



# 深入理解 NB-IoT

基于 3GPP Release-13 协议

温金辉 著

# 目录

<b>第 1 章</b>	<b>IoT 介绍 .....</b>	<b>1</b>
1.1	蜂窝 IoT 标准演进 .....	1
1.2	NB-IoT 简介 .....	2
1.2.1	NB-IoT 关键技术 .....	3
1.3	NB-IoT 与 LoRa 的比较 .....	5
<b>第 2 章</b>	<b>NB-IoT 部署 .....</b>	<b>9</b>
2.1	部署方式 .....	9
2.2	支持的操作频带 .....	10
2.3	anchor/non-anchor carrier .....	11
2.4	multi-carrier .....	12
2.5	CE level (覆盖增强等级) .....	16
<b>第 3 章</b>	<b>系统架构 .....</b>	<b>18</b>
3.1	CIoT EPS 优化 .....	18
3.2	接入侧协议栈 .....	20
3.2.1	PDCP 层 .....	23
3.2.2	RLC 层 .....	23
3.2.3	MAC 层 .....	24
3.3	空口传输概述 .....	29
3.3.1	下行传输简介 .....	30
3.3.2	上行传输简介 .....	31
<b>第 4 章</b>	<b>下行物理层处理 .....</b>	<b>34</b>
4.1	下行时频资源 .....	34
4.2	窄带参考信号 (NRS) .....	35
4.3	下行物理信道 .....	39
4.3.1	NPDCCH .....	41
4.3.2	NPDSCH .....	54
4.3.3	DL gap .....	62
4.4	下行功率分配 .....	63
<b>第 5 章</b>	<b>上行物理层处理 .....</b>	<b>65</b>
5.1	上行时频资源 .....	65
5.1.1	时频资源 .....	65
5.1.2	RU (Resource Unit) .....	67
5.1.3	single-tone vs multi-tone 和 3.75kHz vs 15kHz .....	68
5.2	窄带解调参考信号 (NDMRS) .....	68

5.3	上行物理信道.....	72
5.3.1	NPUSCH format 1.....	73
5.3.2	NPUSCH format 2.....	81
5.3.3	UL gap.....	81
5.4	上行功率控制.....	82
<b>第 6 章</b>	<b>HARQ.....</b>	<b>84</b>
6.1	下行 HARQ.....	84
6.2	上行 HARQ.....	86
<b>第 7 章</b>	<b>小区接入流程.....</b>	<b>88</b>
7.1	小区搜索过程.....	88
7.2	系统信息.....	91
7.2.1	MIB-NB.....	92
7.2.2	SIB1-NB.....	96
7.2.3	SI 消息.....	99
7.2.4	系统信息有效性和变更通知.....	100
7.3	随机接入过程.....	103
7.3.1	Preamble.....	104
7.3.2	NPRACH.....	105
7.3.3	随机接入过程.....	113
7.3.4	PDCCH order 触发的随机接入过程.....	126
7.4	Paging.....	127
7.4.1	PF/PO.....	128
7.4.2	eDRX 下的 Paging.....	130
<b>第 8 章</b>	<b>连接控制.....</b>	<b>133</b>
8.1	RRC 连接建立.....	134
8.2	RRC 连接释放/挂起.....	135
8.3	RRC 连接恢复.....	136
8.4	RRC 连接拒绝.....	140
8.5	初始安全激活和 RRC 连接重配置.....	141
8.6	RRC 连接重建立.....	142
8.7	Access Barring（接入禁止）.....	143
<b>第 9 章</b>	<b>UE 能力.....</b>	<b>145</b>
<b>第 10 章</b>	<b>UE 节能技术.....</b>	<b>148</b>
10.1	PSM.....	148
10.2	eDRX.....	150
<b>第 11 章</b>	<b>NB-IoT 在 Rel-14 和 5G 的演进.....</b>	<b>152</b>

11.1	Rel-14 对 NB-IoT 的增强 .....	152
11.1.1	定位 .....	152
11.1.2	多播增强 .....	152
11.1.3	Non-anchor 上的 PRB 增强 .....	152
11.1.4	移动性和服务连续性增强 .....	152
11.1.5	新功率类型 .....	152
11.2	5G 对 NB-IoT 的增强 .....	153
<b>第 12 章</b>	<b>MTC 和 eMTC .....</b>	<b>154</b>
12.1	MTC .....	154
12.2	eMTC .....	154
12.3	eMTC 与 NB-IoT 的比较 .....	155

## 第1章 IoT 介绍

IoT (Internet of Things) 是物联网的简称, 简单地说, 就是物物互联的网络。其基础依然是互联网, 但网络连接扩展到任意物体之间。物联网通过使物理上独立的物体实现网络连接功能, 来进行信息交换和通信, 以实现对物体的识别、监控、定位和控制等功能。IoT 可广泛地应用于智能抄表、精准农耕、工业自动化、智能建筑、POS 机、环境监控和远程医疗等场景。

以智能抄表这类业务为例。这些 UE 的位置通常是固定不动的, 因此不存在切换 (handover) 的需求。UE 与基站之间的通信不会很频繁, 而且每次传输的只是很小的数据量, 这些数据传输对时延也不敏感。但与传统的 LTE 设备相比, 单小区需要接入的 UE 数量可能高了几个数量级。如果使用传统的 LTE 技术, 会给现有的网络带来很大的负担。

与此同时, 物联网设备经常安装在没有电源的地方, 可能需要完全依靠电池来供电, 并且换电池的成本可能非常高昂。在某些情况下, 电池的寿命甚至决定了整个设备的寿命。因此, 电池使用寿命的优化对物联网设备来说非常重要。另一方面, 物联网设备的覆盖条件可能很差, 例如位于地下室, 因此需要显著地提高 (室内) 覆盖才能满足物联网的需求。

物联网设备的需求量通常很大, 因此需要将其成本控制在较低的范围。NB-IoT 的目标是将每个模组的价格控制在小于 5 美元的范围。

物联网技术主要包括应用于局域网的物联网技术, 包括 WiFi、Zigbee 或 Bluetooth (蓝牙) 等, 以及应用于广域网的物联网技术, 包括 Sigfox、LoRa 和蜂窝 IoT 等。

物联网的快速发展, 对无线通信技术提出了更高的要求, 主要是为低速率、低功耗、低成本、广覆盖和海量物联网设备连接而设计的 LPWAN (Low Power Wide Area Network) 技术应运而生。LPWAN 主要的技术标准包括非 3GPP 主导的 LoRa 和 SigFox, 以及 3GPP 主导的蜂窝 IoT, 包括 eMTC 和 NB-IoT 等。本书主要关注广域网的蜂窝 IoT 技术 NB-IoT 的实现。在 1.3 节, 我们还将重点比较 LoRa 和 NB-IoT 这两个最具发展前景的低功耗广域网技术。

### 1.1 蜂窝 IoT 标准演进

从 Rel-12 开始, 3GPP 定义了许多新的 LTE 特性来满足蜂窝 IoT 的需求:



图 1-1: 蜂窝 IoT 标准演进

在 **Rel-12** 中:

- **MTC** (Machine Type Communication): 引入了 category 0 的 UE 来降低设备的复杂度;
- **省电模式** (Power Saving Mode, PSM): 当 UE 不需要发送或接收数据时, 使用 PSM 来降低功耗。

在 **Rel-13** 中:

- **eMTC** (enhanced Machine Type Communication): 引入了 category M1 的 UE 来降低设备的复杂度, 并增强覆盖到至少 155.7dB MCL (Maximum Coupling Loss);
- **NB-IoT**: 引入了 category NB1 的 UE 来进一步降低复杂度, 并增强覆盖至 164dB MCL;
- **eDRX** (extended Discontinuous Reception) 优化了终止于终端的应用 (device-terminated application) 的电池使用时间;
- 为 IoT 而进行的**网络架构和协议加强**。

在 **Rel-14** (还未最终定稿) 中, 将对 eMTC 和 NB-IoT 做了进一步增强:

- 为 eMTC 和 NB-IoT 增强**定位** (positioning) 能力;
- 为 eMTC 和 NB-IoT 进行**多播** (multicast) 和**移动性** (mobility) 增强;
- NB-IoT 支持新的**功率类型** (power class), 并对接入和**寻呼**进行增强;
- 为 eMTC 提供**更快的速率**和 **VoLTE** 支持。

注: MCL 的概念可参见 36.824 的 5.1.2 节的介绍。

本书主要针对 Rel-13 版本的 NB-IoT 特性进行介绍。但在本书的最后 (第 12 章), 我们会对 MTC 和 eMTC 进行概要性的描述, 并比较这三种蜂窝 IoT 技术的差异。

## 1.2 NB-IoT 简介

NB-IoT (NarrowBand Internet of Things, 窄带 IoT) 是一种基于蜂窝的窄带物联网技术, 支持低功耗设备在广域网的蜂窝数据连接。它是低功耗广域网 (Low Power Wide Area Network, 简称 LPWA) 技术的一种标

准，并由 3GPP 负责其标准化制定工作。其标准协议核心部分于 2016 年 6 月宣告完成，并写入 3GPP Rel-13 版本中。

NB-IoT 主要应用于低吞吐量、能容忍较大时延且低移动性的场景，如智能抄表、传感器和智能建筑等。

NB-IoT 可直接部署于已有的 GSM 或 LTE 网络中，即可复用现有基站以降低部署成本，实现平滑升级。NB-IoT 是可运营的电信网络，这是 NB-IoT 区别于 LoRa、SigFox 等技术的关键。

但需要说明的是，NB-IoT 是一种与 LTE 不同的 RAT 技术，它并不支持后向兼容。但 NB-IoT 可与 LTE 共存于同一网络中，彼此之间在时频位置上区分开。

NB-IoT 允许 UE 通过 E-UTRA 接入网络服务，但其信道带宽被限制为 180kHz。NB-IoT 的设计目标包括：

- **覆盖增强**：与 GSM 相比，提升 20dB 的覆盖能力；
- **超低功耗**：5Wh 电池可供 UE 使用 10 年（真实的电池寿命取决于应用场景和覆盖需求）；
- **大量的低吞吐量 UE 接入**：单扇区可支持 50000 个 UE 连接；
- **时延不敏感**：上行时延可达到 10 秒；
- **低速率**：支持单用户上下行至少 160 bit/s；
- **低成本**：单模组成本小于 5 美元。

NB-IoT 的主要特征包括：

部署方式（deployment）	带内部署、保护频带部署或独立部署
双工模式（duplex mode）	FDD 半双工 type-B
MCL（覆盖）	164 dB
下行	OFDMA（15kHz 子载波间距），单接收天线，1 或 2 发射天线
上行	SC-FDMA（15kHz 或 3.75kHz 子载波间距），支持 multi-tone/single-tone 传输
带宽（bandwidth）	180 kHz（等同于 LTE 中的 1 个 PRB）
峰值速率（peak data rate）	下行 226.7kbps（见 4.3.2.3 节的介绍）。上行 multi-tone 传输为 250kbps，上行 single-tone 为 20kbps（见 5.3.1.2 节的介绍）
省电技术（power saving）	PSM，eDRX
功率类别（power class）	23 dBm

### 1.2.1 NB-IoT 关键技术

NB-IoT 相比 GSM 提升了 20dB 的增益，其 MCL（Maximum Coupling Loss，最大耦合路损）达到 164dB，以期能覆盖到地下车库、地下室以及地下管道等信号难以到达的地方以及广阔的农村地带。为了支持广/深覆盖，其关键技术点包括：

- **关键技术点 1**：窄带信道提高了功率谱密度，上行功率谱密度增强了 17dB；

- 关键技术点 2：重复传输+编码带来 6~16dB 的增益。NB-IoT 为实现覆盖增强采用了重复传输和低阶调制（只支持 BPSK 和 QPSK）等机制（但这样做会降低吞吐量，需要在二者之间做权衡）；
- 通过引入单音载波（single-tone）NPUSCH 传输和  $\pi/2$  BPSK 调制以保持接近 0dB 的 PAPR（Peak-to-Average Power Ratio），从而减少功率放大器（Power Amplifier，简称 PA）的回退，增强了上行覆盖。

大家可以参考资料《NB-IoT 和 eMTC 覆盖能力浅析》来了解 NB-IoT 的覆盖能力以及达到特定覆盖能力所需的重复次数。

为了降低功耗，NB-IoT 在设计时考虑了多种关键技术：

- 关键技术 1：芯片复杂度降低，工作电流小；
- 关键技术 2：空口信令简化，减小单次数据传输功耗。设备消耗的能量与数据量或速率有关，单位时间内发出的数据包大小决定了功耗的大小。通过减少不必要的信令可以达到省电的目的；
- 关键技术 3：基于覆盖等级的控制和接入，减少单次数据传输时间；
- 关键技术 4：PSM（节能模式），终端功耗仅 15uW；在 PSM 模式下，终端仍旧注册在网，但下行信令不可达，从而使终端更长时间驻留在深睡眠状态以达到省电的目的。在 PSM 状态时，终端不接收寻呼信息；
- 关键技术 5：eDRX（扩展 DRX），减少终端监听网络的频率。eDRX 省电技术进一步延长终端在空闲态和连接态下的睡眠周期，减少接收单元不必要的启动。并且相对于 PSM，大幅度提升了下行数据的可达性；
- 关键技术 6：使用长周期 TAU，降低终端发送位置更新的频率；
- 关键技术 7：移动性管理方面，只支持空闲态下的小区选择和重选，而不支持连接态下的 handover，也不支持与 handover 相关的测量、测量报告、切换流程等，从而减少了相关开销。

低速率、低功耗、低带宽带来的是低成本优势。低成本芯片关键技术主要包括：

- 关键技术 1：180kHz 窄带系统，基带复杂度低，不需要复杂的均衡算法；
- 关键技术 2：小带宽带来的低采样率，缓存 Flash/RAM 要求小，DSP 配置低；
- 关键技术 3：使用单天线、半双工 FDD 传输，RF 成本低；
- 关键技术 4：峰均比低，功放效率高，23dBm 发射功率可支持单片 SoC 内置功放 PA，进一步降低成本；
- 关键技术 5：协议栈简化（500kByte），减少片内 FLASH/RAM。

**大容量连接用户：**NB-IoT 比 2G/3G/4G 有 50~100 倍的上行容量提升，在同一基站的情况下，NB-IoT 可以比现有无线技术提供 50~100 倍的接入数。200kHz 频率下，根据仿真测试数据，单小区可支持 5 万个 NB-IoT 终端接入。其关键技术点包括：

- 关键技术 1：使用窄带技术，上行等效功率（36 信道\*23dBm）提升，大大提升信道容量；
- 关键技术 2：减小空口信令开销，提升频谱效率；
- 关键技术 3：基站优化。包括独立的准入拥塞控制和终端上下文信息存储；
- 关键技术 4：核心网优化。包括终端上下文存储以及下行数据缓存。



NB-IoT 还对 LTE 的功能进行了大幅度的删减。NB-IoT 不支持绝大多数的 LTE-A 特性，如载波聚合（Carrier Aggregation）、双连接（Dual Connectivity）和 D2D（Device to Device）服务等。由于 NB-IoT 不用于时延敏感的数据包传输，因此没有 QoS 的概念。NB-IoT 不支持所有要求保证比特率（GBR）的服务，如实时 IMS 等。

NB-IoT 不支持 VoLTE，因为 VoLTE 对时延要求高，并需要可靠的 QoS 保障。

由于不支持与其它无线技术的交互，因此 NB-IoT 也不支持与其相关的特性。如：不支持 LTE-WLAN 互通，不支持用于设备内共存（In-device coexistence）的干扰避免以及不支持用于监视信道质量的测量等。

除了上面介绍的特性外，NB-IoT 还不支持以下特性：

- 封闭用户组（Closed Subscriber Group, CSG）；
- 中继节点(Relay Node, RN)；
- ACB（Access Class Barring）、EAB（Extended Access Barring）、SSAC（Service Specific Access Control）和 ACDC（Application specific Congestion control for Data Communication）；
- MBMS（Multimedia Broadcast Multicast Service）；
- 自配置和自优化（Self-configuration and self-optimisation）；
- 用于网络性能优化的测量记录和上报（Measurement logging and reporting for network performance optimisation）；
- 公共告警系统（Public warning systems），如：CMAS、ETWS 和 PWS；
- 实时服务（包括紧急呼叫）；
- CS 服务和 CS 回退；
- 网络辅助的干扰消除/抑制（Network-assisted interference cancellation/suppression）。

### 1.3 NB-IoT 与 LoRa 的比较

LoRa（Long Range）是美国的 Semtech 公司于 2013 年 8 月发布的一种基于线性扩频技术的超远距离低功耗无线传输技术。它与 NB-IoT 均采用星型网络拓扑结构，终端均需要通过射频与网关或基站连接，并通过网关或基站来实现大范围的网络信号覆盖。因此，LoRa 和 NB-IoT 天然就存在竞争关系。

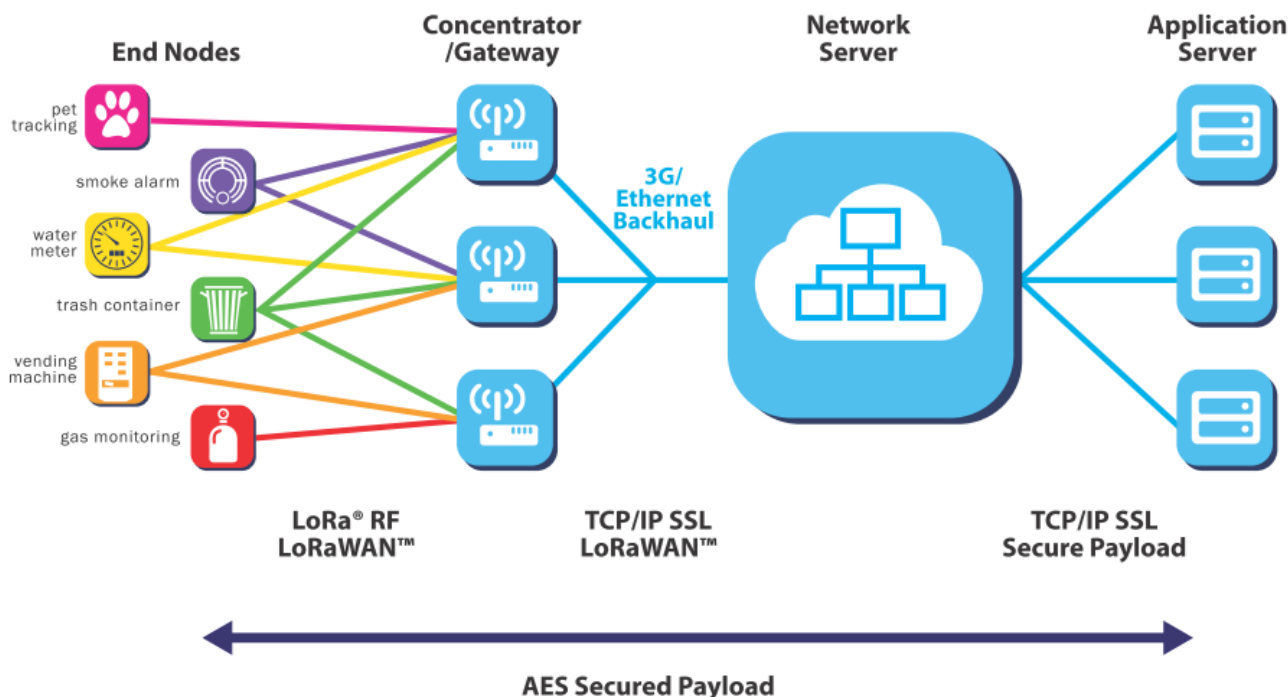


图 1-2: LoRa 网络架构

LoRa 工作在免费的非授权频谱上，任何企业都可以自行组网，因此不依赖于移动运营商的网络覆盖，也无需向运营商支付网络使用费用。而 NB-IoT 工作在授权频谱上，网络运营集中在移动运营商手里，其覆盖也受制于移动运营商的网络覆盖。但受益于无处不在的 LTE 网络部署，理论上说，NB-IoT 的覆盖更广。

LoRa 工作在非授权频谱上，可能遭受使用同一频率的其它无线通信技术的干扰。由于 LoRa 使用线性 Chirp 扩频调制，抗干扰能力强，因此也不必过于担心其遭受的干扰水平。

NB-IoT 工作在授权频谱上，在处理干扰和网络重叠方面特性更好。同时它能够提供更与蜂窝网络类似的 QoS，而 LoRa 无法提供 QoS 保证。

在 NB-IoT 中，终端需要同网络定期同步。终端通过采用 PSM 和 eDRX 等省电技术来延长 IDLE 态下的休眠时间；同时通过减少不必要的信令和不接收（或减少接收）寻呼信息，来降低功耗，延长电池使用时间。LoRa 使用 ALOHA 异步通信协议，终端可根据自身需要进行休眠。相对于 NB-IoT 而言，LoRa 终端的功耗稍占优。但由于 NB-IoT 会与网络定期同步，因此其下行延时更短。

相对 LoRa 来说，NB-IoT 能够提供更高的速率。

基于 LTE 的 NB-IoT 技术，其端到端的安全机制已经被几十亿的 LTE 用户证明是可靠的。而 LoRa 使用应用层的加密技术来保证安全，其可靠性还有待证明。

由于 NB-IoT 的调制机制和协议相对复杂，而且使用的是授权频段，目前的产业规模和成熟度也较 LoRa 晚，所以模组的总体使用成本相对较高。当然，模组成本的决定因素在于出货量。随着全球主流运营商大力部署 NB-IoT 网络，加之电信设备供应商、芯片和模组以及终端设备厂商的共同推动，假以时日，NB-IoT 的模组成本有望达到甚至低于 LoRa 的模组成本。

可以看出，对于具有较频繁的通信和较低的延迟要求或大数据量的应用，NB-IoT 将是最好的选择。然而，对于需要非常长的电池寿命和更优的成本，但不需要频繁通信的应用，LoRa 是更好的选择。参考资料[4]对 NB-IoT 和 LoRa 进行了比较，并对二者的应用部署场景进行了举例。

表 1-1: NB-IoT 与 LoRa 的比较

技术标准	NB-IoT	LoRa
标准化组织	3GPP	LoRa Alliance（非 3GPP 组织）
部署	可重用已有的网络	需要建立一个新的网络
带宽	180kHz	125~500kHz
频谱	使用授权频谱（已授权的 LTE 频带或对 GSM 频带进行重耕）	使用非授权频谱（北美：902 – 928 MHz；欧洲：863 – 870MHz）
速率	~30/60 kbps (DL/UL)	300 bps – 38.4 kbps
覆盖	164dB。由于无处不在的 LTE 网络部署，因此其覆盖更广	声称最大支持 157dB
电池	2AA 电池可用 10 年（真实的电池寿命取决于应用场景和覆盖需求）	1AA 电池可用 5 年；或工业电池可用 10 至 20 年
移动性	有限的移动性（只支持小区重选）	支持移动性和漫游
安全	完全支持已被证明的端到端的安全机制	基于软件的加密
地理定位（Geo-positioning）	Rel-13 不支持；但将在 Rel-14 中加入定位功能	可选
技术演进	有清晰的演进路线，Rel-14 和 5G 将对 NB-IoT 做进一步增强	未来的演进路线还不明确

LoRa 在技术标准成熟度上要早于 NB-IoT，同时由于可以使用非授权频段部署，所以企业可以自建或者会出现基于 LoRa 的“非传统电信运营商”来提供物联网业务。从目前看，因为 LoRa 的开放性，LoRa 在全球的实际部署案例和产业链要好于 NB-IoT。但是随着 NB-IoT 技术标准的确定，特别是各大运营商的使用，NB-IoT 的应用和产业链也会迅速起来。

随着 NB-IoT 标准的确定，基于 NB-IoT 的物联网部署将迎来一个高速增长的时期，与之关联的终端、网络、平台和应用等产业链将进一步成熟壮大。特别是各大电信运营商均将 NB-IoT 标准作为后续物联网部署的主要标准，这将大大推动 NB-IoT 市场的发展。

同时需要看到的是，物联网的每一种应用场景都需要考虑多种因素，如节点成本、网络成本、电池寿命、数据速率（吞吐量）、延迟、移动性、覆盖范围和部署模型等等。不同的应用场景有不同的需求，即使对应同一种应用场景，不同用户也可能有不同的需求，因此没有一种单一技术能够同时解决所有的问题。虽然 LoRa 和 NB-IoT 之间存在一定程度的竞争，但这二者还是具有不同的技术和商业特性，可以服务于不同的市场。从长期来看，二者互相竞争又互为补充。

## 【参考资料】

- [1] 《NB-IoT 解决方案介绍》，华为
- [2] 《NB-IoT 和 eMTC 覆盖能力浅析》，中国移动研究院，李秋香、崔航、徐芙蓉、李新、杨光
- [3] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [4] 《NB-IoT vs LoRa Technology》，LoRa Alliance
- [5] 《A technical overview of LoRa and LoRaWAN》，LoRa Alliance
- [6] <http://www.c114.net/news/131/a993391.html>
- [7] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification

## 第2章 NB-IoT 部署

### 2.1 部署方式

NB-IoT 在频域上占用 180kHz 的有效带宽，对应 LTE 传输中的一个 RB。子载波间距可以为 3.75kHz 或 15kHz。

NB-IoT 支持 3 种部署方式（如图 2-1 所示）：

- 独立部署（standalone operation）：这种部署方式使用独立的频带，主要适用于对 GSM 频段的重耕。GSM 的单载波带宽为 200kHz，可以将 NB-IoT 部署在 GSM 频带上，并在频带的两端各预留 10kHz 作为保护频带；
- 保护频带部署（guardband operation）：利用 LTE 载波的保护频带上的未使用的 RB（Resource Block，资源块）资源来部署 NB-IoT；
- 带内部署（in-band operation）：利用 LTE 载波上的 RB 来部署 NB-IoT。

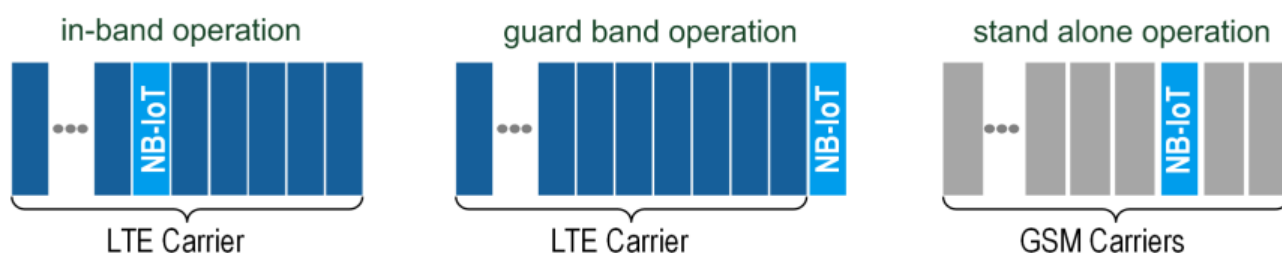


图 2-1：NB-IoT 的 3 种部署方式

三种部署方式的比较见表 2-1。

表 2-1：NB-IoT 的 3 种部署方式的比较

部署方式	比较
独立部署	<ul style="list-style-type: none"><li>• NB-IoT 不占用 LTE 小区的频谱资源；</li><li>• 需要为 NB-IoT 增加额外的频谱资源。</li></ul>
保护频带部署	<ul style="list-style-type: none"><li>• NB-IoT 不占用 LTE 小区的频谱资源；</li><li>• 不需要为 NB-IoT 增加额外的频谱资源。</li></ul>
带内部署	<ul style="list-style-type: none"><li>• 需要占用 LTE 小区的频谱资源；</li><li>• 需要在 NB-IoT 载波和 LTE 容量之间进行权衡；</li><li>• 不需要为 NB-IoT 增加额外的频谱资源。</li></ul>

使用带内部署和保护频带部署时，NB-IoT 可以通过升级的方式共享 LTE 的 RRU 和 BBU 资源。

eNodeB 通过 *MasterInformationBlock-NB* 的 *operationModeInfo-r13* 来指示 NB-IoT 小区当前使用的部署模式。也就是说，UE 需要接收到 MIB-NB 后才能知道小区使用的部署模式。

在使用带内部署时，用于 LTE 系统的小区特定的参考信号（CRS）、可能存在的 CSI-RS 以及每个子帧开始处用于下行 L1/L2 控制信道（PCFICH/PHICH/PDCCH）的控制区域的资源（RE 或 OFDM 符号）需要

预留而不能分配给 NB-IoT 使用。这是因为这些参考信号或控制信道的传输是跨越整个系统带宽的，不能为了部署 NB-IoT 而阻断 LTE 原有的信号传输。而其它部署模式就不存在这个问题。

当前版本（Rel-13）中，NB-IoT 只支持 **FDD** LTE 系统，且只支持**半双工 type-B** 模式。NB-IoT 不支持 TDD LTE 系统。（见 36.101 的 5.5F 节）

NB-IoT UE 使用 FDD 半双工 type-B 模式意味着，上行和下行使用不同的频率，并且 UE 在同一时刻只能接收数据或只能发送数据，而不能同时收发数据。另外，从上行切换到下行，或从下行切换到上行时，中间必须存在至少一个保护子帧（guard subframe），以便 UE 在传输机和接收机之间进行切换。

## 2.2 支持的操作频带

NB-IoT 支持的操作频带（operating band）包括{1, 2, 3, 5, 8, 12, 13, 17, 18, 19, 20, 26, 28, 66 }，该值在 36.101 的 Table 5.5-1 中定义。可以看出，NB-IoT 主要部署在低频的频段上，这是因为低频信号覆盖广，穿透性强，更符合 NB-IoT 的应用场景需求。

**Table 5.5-1 E-UTRA operating bands**

E-UTRA Operating Band	Uplink (UL) operating band BS receive UE transmit		Downlink (DL) operating band BS transmit UE receive		Duplex Mode
	$F_{UL\_low}$	$F_{UL\_high}$	$F_{DL\_low}$	$F_{DL\_high}$	
1	1920 MHz	1980 MHz	2110 MHz	2170 MHz	FDD
2	1850 MHz	1910 MHz	1930 MHz	1990 MHz	FDD
3	1710 MHz	1785 MHz	1805 MHz	1880 MHz	FDD
5	824 MHz	849 MHz	869 MHz	894MHz	FDD
8	880 MHz	915 MHz	925 MHz	960 MHz	FDD
12	699 MHz	716 MHz	729 MHz	746 MHz	FDD
13	777 MHz	787 MHz	746 MHz	756 MHz	FDD
17	704 MHz	716 MHz	734 MHz	746 MHz	FDD
18	815 MHz	830 MHz	860 MHz	875 MHz	FDD
19	830 MHz	845 MHz	875 MHz	890 MHz	FDD
20	832 MHz	862 MHz	791 MHz	821 MHz	FDD
26	814 MHz	849 MHz	859 MHz	894 MHz	FDD
28	703 MHz	748 MHz	758 MHz	803 MHz	FDD
66	1710 MHz	1780 MHz	2110 MHz	2200 MHz	FDD

### 2.3 anchor/non-anchor carrier

在 NB-IoT 中，**锚点载波（anchor carrier）**是用于发送 NPSS / NSSS / NPBCH / SIB-NB 的载波，anchor carrier 被用于小区接入，寻呼消息和随机接入过程只能在 anchor carrier 上进行。**非锚点载波（non-anchor carrier）**是不用于发送 NPSS / NSSS / NPBCH / SIB-NB 的载波，non-anchor carrier 被用于除 NPSS / NSSS / NPBCH / SIB-NB 外的数据传输（non-anchor carrier 也不能用于寻呼消息和随机接入过程中的数据传输。随机接入过程中的数据传输都是在 anchor carrier 上进行的）。

NB-IoT 下行载波的中心频点位于一个 PRB 的第 6 个和第 7 个子载波的中间，且不存在 DC 子载波。

使用带内部署时，分配给 NB-IoT 的资源不是固定的。并且不是 LTE 载波内的所有 RB 都能被用作 anchor carrier，即只有特定的 RB 能被 NPSS / NSSS / NPBCH / SIB-NB 使用，如表 2-2 所示。

表 2-2：带内部署时，LTE 载波内能被用于 NB-IoT anchor carrier 的 PRB 索引集合

LTE 系统带宽	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
可用于 NB-IoT anchor carrier 的 LTE 频域上的 PRB 索引	2, 12	2, 7, 17, 22	4, 9, 14, 19, 30, 35, 40, 45	2, 7, 12, 17, 22, 27, 32, 42, 47, 52, 57, 62, 67, 72	4, 9, 14, 19, 24, 29, 34, 39, 44, 55, 60, 65, 70, 75, 80, 85, 90, 95

与 LTE 相同，用于 NB-IoT 的信道栅格（channel raster）也为 100kHz（见 36.101 的 5.7.2F 节），这意味着用于 NB-IoT 的载波（anchor carrier，不包括 non-anchor carrier）的中心频率必须是 100kHz 的整数倍。也就是说，NB-IoT UE 只需要在位于 100kHz 栅格的载波上进行 NB-IoT 小区搜索。这还意味着，使用带内部署时，anchor carrier 只能位于某些特定的 PRB 上。

接下来，我们会以 10MHz 的 LTE 载波为例，来介绍如何获知该载波内可用于部署 NB-IoT anchor carrier 的 PRB 索引集合，即如何得到表 2-2。

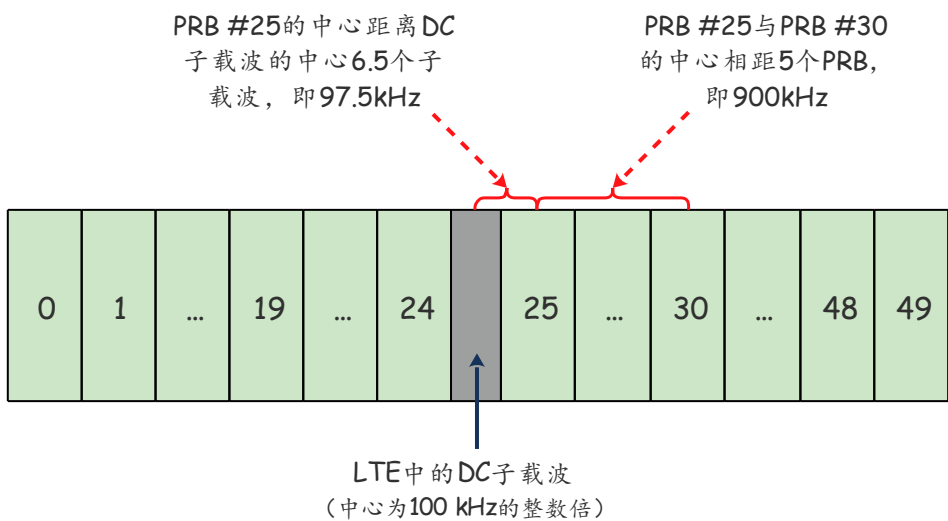


图 2-2：10MHz 的 LTE 小区内，位于 100kHz 栅格内的 PRB

10 MHz 的 LTE 载波，其系统带宽为 50 RB，PRB #24 和 PRB #25 分别位于 DC 子载波的两端。由于 LTE 载波本身的中心频率也是 100kHz 的整数倍，其 DC 子载波必然位于 100kHz 的栅格上。又由于 PRB #25

的中心距离 DC 子载波的中心 6.5 个子载波，即 97.5kHz ( $15\text{kHz}/\text{subcarrier} \times 6.5 \text{ subcarriers}$ )，所以 PRB #25 的中心距离最近的 100kHz 的距离为 2.5kHz。也就是说，PRB #25 位于 100kHz 的栅格上。又因为相邻 PRB 之间的中心相距 180kHz，所以 PRB #30、#35、#40、#45 也同样位于 100kHz 的栅格上，并且其中心距离最近的 100kHz 的距离为 2.5kHz (-2.5kHz)。同理，PRB #24、#19、#14、#9、#4 也同样位于 100kHz 的栅格上，并且其中心距离最近的 100kHz 的距离为 2.5kHz (+2.5kHz)。

与此同时，一个 NB-IoT 的 anchor carrier 不能占用一个 LTE 载波的中心 6 个 PRB，这是因为 LTE 中的 PSS/SSS/PBCH 需要占用中心 6 个 PRB 的大量 RE，使得这 6 个 RB 不能被用于 NB-IoT 传输。也就是说，10 MHz 的 LTE 载波的 PRB #24 和 #25 不能被用于 anchor carrier。

综上，我们可以得出，一个 10MHz 的 LTE 载波上，只有 PRB 索引集合{4, 9, 14, 19, 30, 35, 40, 45}内的 PRB 能被用于 NB-IoT 的 anchor carrier。

同样的，20MHz 的 LTE 载波中，存在一个可用于 NB-IoT anchor carrier 的 PRB 索引集合，该集合中的每个 PRB 的中心距离最近的 100kHz 栅格的距离为 2.5kHz。3MHz、5MHz 和 15MHz 的 LTE 载波中，存在一个可用于 NB-IoT anchor carrier 的 PRB 索引集合，该集合中的每个 PRB 的中心距离最近的 100kHz 栅格的距离为 7.5kHz。

需要说明的是，此限制只针对带内部署的 anchor carrier，对 non-anchor carrier 而言，并不存在此限制。配置的 non-anchor carrier 不需要位于接近 100kHz 的栅格上。

1.4MHz 的 LTE 载波只有 6 个 PRB，所以该载波不能用于带内部署的 NB-IoT。

使用带内部署时，NB-IoT 载波必须与 LTE 载波中的 PRB 对齐，否则会对 LTE 载波上的传输产生不利的影响。anchor carrier 的中心频点与最近的 100kHz 整数倍的频率之间存在偏移，且频偏范围为{+2.5, -2.5, +7.5, -7.5} kHz。

使用保护频带部署时，NB-IoT 载波并不需要与 LTE 载波中的 PRB 对齐。但由于保护频带上的子载波是从 LTE 载波传输带宽的边缘开始向两端划分的，因此其 anchor carrier 的中心频点与最近的 100kHz 整数倍的频率之间同样存在偏移，且频偏范围同样为{+2.5, -2.5, +7.5, -7.5} kHz。

使用独立部署时，NB-IoT 载波的中心频点一定为 100kHz 的整数倍，不存在频率偏移。

UE 进行小区接入时使用的载波即为 anchor carrier。在下一节介绍 multi-carrier 时，我们会介绍如何给 UE 配置 non-anchor carrier。

## 2.4 multi-carrier

通俗点讲，multi-carrier 就是在 anchor carrier 之外，再给连接态的 UE 配置一个 non-anchor carrier 用于数据传输。

使用独立部署、保护频带部署和带内部署的 NB-IoT 均支持 multi-carrier（多载波）配置。multi-carrier 配置是 UE 特定的配置，且只有处于 RRC\_CONNECTED 态的 UE 才会配置 multi-carrier。

对于 multi-carrier 配置而言，存在一个 anchor carrier 用于 UE 的初始同步，同时还可以给 UE 配置一个额外的 non-anchor carrier 用于单播传输，并且配置的 non-anchor carrier 不需要位于接近 100kHz 的栅格上。



UE 需要先向 eNodeB 上报自己是否支持 multi-carrier 的能力（具体见第 9 章的介绍），eNodeB 才能决定是否给 UE 配置 non-anchor carrier。

eNodeB 可以在 RRC 连接建立、重建立、恢复或重配置流程中，给 UE 配置一个 non-anchor carrier。

对于处于 RRC\_CONNECTED 态的 NB-IoT UE 而言，其所有的上下行单播传输可以通过 UE 特定的 *carrierConfigDedicated-r13* 被配置到一个 non-anchor carrier 上。可以看出，针对上行传输和下行传输，可分别配置不同的 non-anchor carrier。也可以只为上行传输配置 non-anchor carrier，或只为下行传输配置 non-anchor carrier。

```

CarrierConfigDedicated-NB-r13 ::= SEQUENCE {
    dl-CarrierConfig-r13      DL-CarrierConfigDedicated-NB-r13,  ----下行non-anchor carrier配置（用于所有的下行单播传输）
    ul-CarrierConfig-r13      UL-CarrierConfigDedicated-NB-r13    ----上行non-anchor carrier配置（用于所有的上行单播传输）
}

DL-CarrierConfigDedicated-NB-r13 ::= SEQUENCE {
    dl-CarrierFreq-r13        CarrierFreq-NB-r13,                ----下行载波频点。该载波不能是LTE中用于传输
    PSS/SSS/PBCH的PRB
    downlinkBitmapNonAnchor-r13 CHOICE { ----non-anchor carrier上用于下行传输的NB-IoT下行子帧配置。关于NB-
    IoT下行子帧的概念，可参见36.213的16.4节，或本书的4.1节
        useNoBitmap-r13      NULL,                                ----指示所有的下行子帧都可用于non-anchor carrier上的下行数据传
        输
        useAnchorBitmap-r13  NULL,                                ----指示non-anchor carrier使用与anchor carrier相同的NB-IoT下行子
        帧配置，即与SIB1-NB中的downlinkBitmap-r13配置相同
        explicitBitmapConfiguration-r13 DL-Bitmap-NB-r13,        ----显式地指定non-anchor carrier特定的NB-IoT下行子帧配
        置，这是一个bitmap
        spare                NULL
    } OPTIONAL, -- Need ON
    dl-GapNonAnchor-r13      CHOICE { ----用于non-anchor carrier的下行传输gap配置，见36.211的10.2.3.4节，或
    本书的4.3.3节
        useNoGap-r13        NULL,                                ----指示不使用下行传输gap
        useAnchorGapConfig-r13 NULL,                                ----指示non-anchor carrier使用与anchor carrier相同的下行传输gap配
        置，即与SIB2-NB中的dl-Gap-r13配置相同
        explicitGapConfiguration-r13 DL-GapConfig-NB-r13,        ----显式地指定non-anchor carrier特定的下行传输gap配置
        spare                NULL
    } OPTIONAL, -- Need ON
    inbandCarrierInfo-r13    SEQUENCE { ----指示使用带内部署的non-anchor carrier的配置
        samePCI-Indicator-r13 CHOICE {
            samePCI-r13      SEQUENCE { ----与LTE载波使用相同的PCI
                indexToMidPRB-r13 INTEGER (-55..54) ----与LTE载波中心相比，偏移的PRB的索引
            },
            differentPCI-r13 SEQUENCE { ----与LTE载波使用不同的PCI
                eutra-NumCRS-Ports-r13 ENUMERATED {same, four} ----LTE载波的CRS天线端口数：该值或者与
                NRS的天线端口数相同，或者为4天线端口
            }
        }
    } OPTIONAL, -- Cond anchor-guardband

```

```

        eutraControlRegionSize-r13          ENUMERATED {n1, n2, n3}      ----指示带内部署模式下，LTE小区中控制区域所占的OFDM符号数。如果MIB-NB中的operationModeInfo被设置为inband-SamePCI或inband-DifferentPCI，则该值需要设置为SIB1-NB中广播的eutraControlRegionSize-r13值

    }
    ...,
    [[ nrs-PowerOffsetNonAnchor-v1330      ENUMERATED {dB-12, dB-10, dB-8, dB-6,
                                                dB-4, dB-2, dB0, dB3} ----non-anchor carrier的NRS EPRE相对于anchor carrier的功率偏移，以dB为单位。见4.4节的介绍

                                                OPTIONAL      -- Need ON

    ]]
}

UL-CarrierConfigDedicated-NB-r13 ::= SEQUENCE {
    ul-CarrierFreq-r13          CarrierFreq-NB-r13          OPTIONAL,      -- Need OP ----上行载波频点（如果存在的话）
    ...
}

```

如果 eNodeB 没有给 UE 配置 non-anchor carrier，则 UE 的所有上下行数据都在 anchor carrier 上传输。

当 UE 的下行配置了 non-anchor carrier 时，处于 RRC\_CONNECTED 态的该 UE 只需要在 non-anchor carrier 上接收下行数据，而不会去 anchor carrier 上接收数据。此时 UE 只需要一个接收机链（receiver chain）。

也许大家会问，如果 UE 只在 non-anchor carrier 上接收下行数据，那如何接收在 anchor carrier 上发送的 NPSS/NSSS/NPBCH/SIB-NB？答案就是，与 LTE 不同，处于 RRC\_CONNECTED 态的 NB-IoT UE 不需要去接收这些信息。由于 NB-IoT 不支持 RRC\_CONNECTED 态下的移动性管理，即不支持 handover，所以处于 RRC\_CONNECTED 态的 UE 不需要去接收 NPSS/NSSS。同时 36.331 的 4.2.1 节明确说明，处于 RRC\_CONNECTED 态的 UE 不需要接收系统信息，因此也就不需要去接收 NPBCH/SIB-NB。

同样的，当 UE 的上行配置了 non-anchor carrier 时，处于 RRC\_CONNECTED 态的该 UE 只会在 non-anchor carrier 上发送上行数据。此时 UE 只需要一个发射机链（transmitter chain）。

当 UE 从 RRC\_CONNECTED 态切换到 RRC\_IDLE 态时，UE 会回到 anchor carrier 上收发数据。

由于在 NB-IoT 中，收发不会同时进行，并且收发都限定在一个频带上，因此即使配置了 multi-carrier，UE 使用一个带宽为 180kHz 的接收机/发射机链（transmitter/receiver chain）就足够了。

从 UE 的角度上看，存在一个 anchor carrier 和至多一个 non-anchor carrier。但从 eNodeB 的角度上看，使用 multi-carrier 允许 eNodeB 根据特定时间内需要服务的 UE 数量，灵活地控制分配给 NB-IoT 的资源量（NB-IoT 的系统容量可以通过增加 NB-IoT 载波的数量进行扩展）。例如，可以给同一小区内所有 UE 配置相同的 anchor carrier，并根据需要，灵活地给不同的 UE 配置不同的 non-anchor carrier。这样，下行的同步信号和系统信息、上行的 NPRACH 资源等只会被限制在一对承载（anchor carrier）上，而其它的承载（non-anchor carrier）可以完全用于数据传输。由于 non-anchor carrier 不需要发送同步信号和系统信

息等，所以可以包含更多的子帧用于数据传输。当然，在不发送公共信息和随机接入相关信息的时候，anchor carrier 也可以用于传输用户数据。

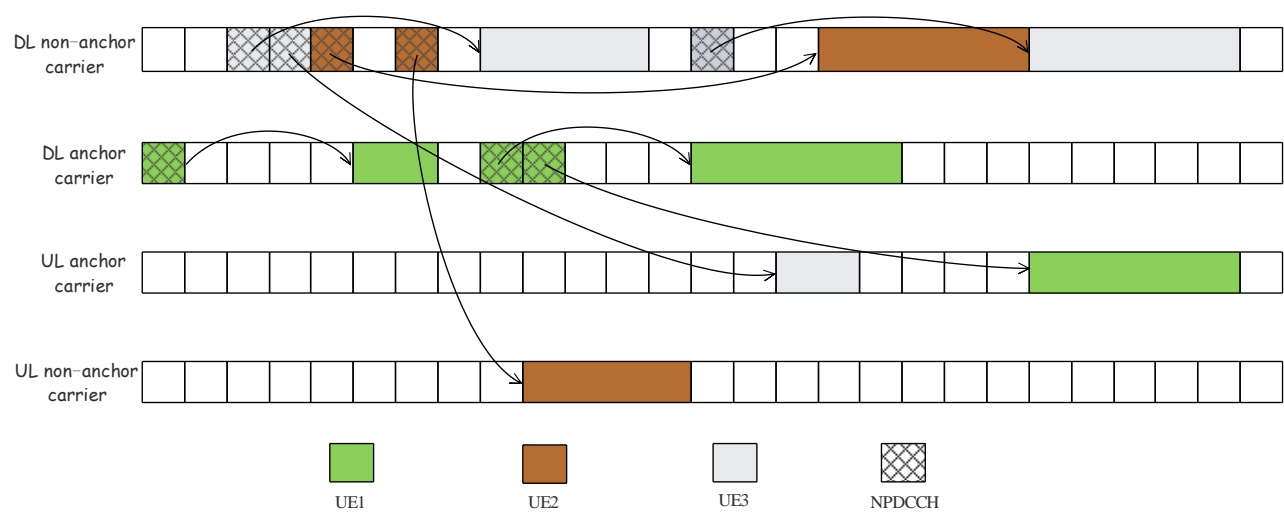


图 2-3: 调度 3UE 的例子，每一个小方格表示一个子帧

图 2-3 是 eNodeB 调度 3 个 UE 的一个例子。UE 1 只配置了 anchor carrier，UE 2 在上下行均配置了一个 non-anchor carrier，UE 3 只在下行配置了一个 non-anchor carrier，而上行使用 anchor carrier。为了简单，这个图没有考虑 NPDCCH 的周期，也没有考虑允许用于 NB-IoT 下行传输的子帧，它只是以原理图的方式来呈现。

NB-IoT 中的 multi-carrier 部署支持 “inband + inband”、“inband + guardband” 和 “guardband + guardband” 的 “anchor carrier + non-anchor carrier” 组合，但要求这些组合中的 2 个载波与同一个 LTE 小区相关联。

NB-IoT 中的 multi-carrier 部署还支持 “standalone + standalone” 的 “anchor carrier + non-anchor carrier” 组合，但要求这 2 个载波的频率跨度不能超过 20MHz 且这 2 个载波是同步的。

NB-IoT 的 multi-carrier 部署不支持 standalone 与 inband 或 guardband 进行组合。

NB-IoT 支持的 anchor carrier 和 non-anchor carrier 组合如表 2-3 所示。

表 2-3: anchor carrier 和 non-anchor carrier 部署组合

	anchor-carrier			
		inband	guardband	standalone
non-anchor carrier	inband	支持	支持	不支持
	guardband	支持	支持	不支持
	standalone	不支持	不支持	支持

使用 multi-carrier 可以带来几点好处：

(1) 节省公共信道的开销：如果只使用 anchor carrier，那么每一个 carrier 都需要发送 NPSS/NSSS/NPBCH/SIB-NB，而这些信道所占的开销是很大的，这就降低了业务信道的容量。使用 multi-carrier 时，不需要在 non-anchor carrier 上发送 NPSS/NSSS/NPBCH/SIB-NB，因此降低了公共信道的开销；

(2) 减少异频小区数：如果只使用 anchor carrier，则可能存在大量的异频小区，这会对空闲态 UE 的移动性管理带来很大的负担。使用 multi-carrier，anchor carrier 的数量大幅减少，从而减少了异频小区数；

(3) 降低初始小区选择的功耗：使用 multi-carrier，anchor carrier 的数量大幅减少，UE 需要测量的小区数也就减少了，从而减少了 UE 初始小区选择的功耗；

(4) 降低 inband 部署模式下对 LTE 系统性能的影响：为了保证小区的覆盖，通常会在发送 NPSS/NSSS/NPBCH 等公共信道时，对相应信道的功率进行提升（power boosting）。在 inband 部署模式下，如果只使用 anchor carrier，就会有更多的 PRB（anchor carrier）进行了功率提升，这会对 LTE 系统的性能产生很大的影响。

简单地说，

NB-IoT UE 在 RRC\_IDLE 态时，只会跟 anchor carrier 打交道。UE 只会在其驻留的 anchor carrier 上接收同步信号、系统信息或 Paging 消息，以及在 anchor carrier 上发起随机接入。

NB-IoT UE 在 RRC\_CONNECTED 态，但 eNodeB 没有给 UE 配置 non-anchor carrier 时，所有的上下行数据都只会 anchor carrier 上传输。

NB-IoT UE 在 RRC\_CONNECTED 态，且 eNodeB 给 UE 配置了 non-anchor carrier 时，UE 只会 non-anchor carrier 上收发单播数据。

## 2.5 CE level（覆盖增强等级）

为了应对不同的无线信道条件，NB-IoT 定义了至多 3 个覆盖增强等级（Coverage Enhancement level，CE 等级）：CE 等级 0 至 CE 等级 2，分别对应可对抗 MCL 为 144dB、154dB 和 164dB 的信号衰落。CE 等

级 0 对应正常覆盖（信道条件最好）；CE 等级 2 对应信道条件最差的情况，并认为该覆盖可能非常差。定义多少个 CE 级别是由网络侧决定的。针对每个 CE 等级，小区会广播一个所接收的参考信号的功率阈值列表。不同 CE 级别主要影响消息重复发送的次数，信道条件差的，就需要多重复发送几次。

需要说明的是，NB-IoT 中各个物理信道的重复传输不是基于 ACK/NACK 的重传。重复是不管接收侧的接收情况如何，对同一数据重复传输几次，这类似于 LTE 中的 TTI bundling，但 NB-IoT 中上下行均需要支持重复传输。如果需要的话，这些传输会作为一个整体进行 ACK/NACK 反馈（如 NPDSCH 和 NPUSCH）。重复类似于把一句话多说几遍，别人听清的概率就提高了。

在下行，独立部署模式可单独设置发射功率。而保护频带部署和带内部署的功率与 LTE 功率有关，即此时下行功率受限于 LTE 载波的功率。因此带内部署或保护频带部署通常需要更多的重复次数才能达到与独立部署相同的覆盖水平。在相同的覆盖水平下，独立部署的下行速率通常优于其它 2 种部署方式。而在上行，3 种部署模式基本没有区别。

参考资料《NB-IoT 和 eMTC 覆盖能力浅析》分析整理了不同条件下各信道为了满足某一覆盖等级要求所需要的重传次数。

关于 CE 等级的配置，UE 如何确定自己所在的 CE 等级，以及 CE 等级对随机接入过程的影响等，将在 7.3 节予以介绍。

## 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 《A Primer on 3GPP Narrowband Internet of Things (NB-IoT) 》，Ericsson, Y.-P.Eric wang, XingQin Lin, Ansuman Adhikary, etc.
- [3] 3GPP TS 36.101 V13.6.1, 2017-01; User Equipment (UE) radio transmission and reception
- [4] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [5] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification

## 第3章 系统架构

NB-IoT 的物理层设计大部分沿用 LTE 技术，如上行采用 SC-FDMA，下行采用 OFDM 等。高层协议设计同样沿用 LTE 协议，但针对非频繁小数据包、低功耗和大规模连接等特性进行了功能增强。与此同时，NB-IoT 支持非 IP 的数据传输，并支持 SMS（可选），协议针对这些业务也做了进一步的增强。

与 LTE 类似，NB-IoT 的控制面与用户面分离，信令走控制面，数据走用户面。但针对低速率业务，NB-IoT 也可以直接走控制面来传输用户数据，不再建立专用的 DRB 承载，从而省去了 NAS 和核心网建立连接的信令流程，缩短唤醒恢复时延。

NB-IoT 的接入网整体结构跟 LTE 相比，并没有变化。eNodeB 使用 S1 接口与 MME 和 S-GW 相连，不同之处在于其携带的是 NB-IoT 消息和数据包。虽然 NB-IoT 中并不存在 handover，eNodeB 之间依然存在一个 X2 接口，以便 UE 进入 IDLE 态后能快速恢复（即跨基站用户上下文恢复，具体见 8.3 节的介绍）。

在本章中，我们会先简单介绍一下核心网针对 NB-IoT 所做的优化以及该优化对接入侧的影响。接下来，我们会简单介绍接入侧协议栈针对 NB-IoT 所做的优化。在最后一节，我们会对 NB-IoT 的空口传输进行宏观上的描述。

### 3.1 CIoT EPS 优化

为了给应用发送数据，EPS（Evolved Packet System）为 CIoT（Cellular Internet of Thing）定义 2 种优化方案：用户面 CIoT EPS 优化和控制面 CIoT EPS 优化。

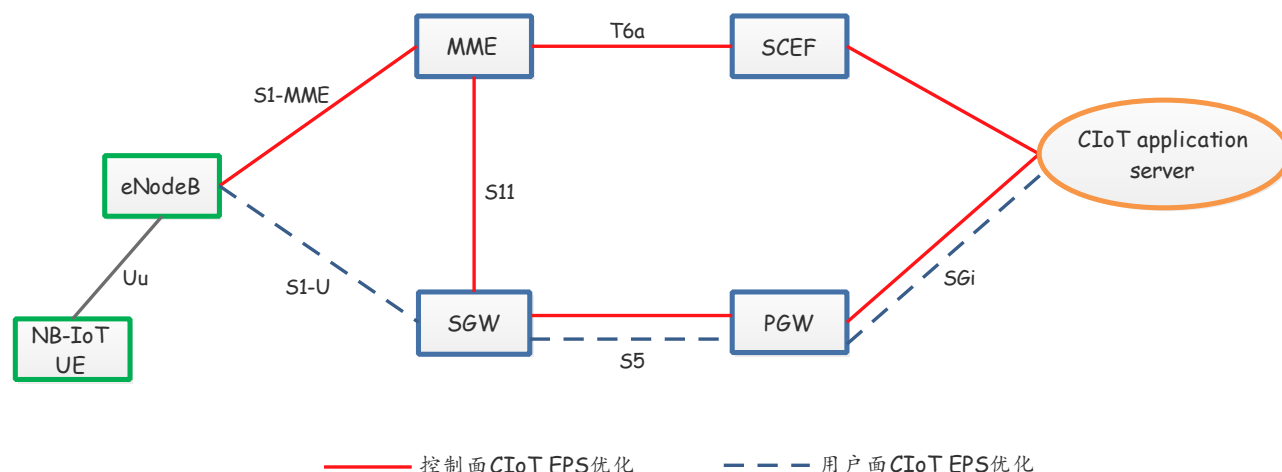


图 3-1：NB-IoT 数据发送和接收的路径。红色实线指示控制面 CIoT EPS 优化的数据通道；蓝色虚线指示用户面 CIoT EPS 优化的数据通道

LTE 中的用户数据均通过用户面进行传输，但在 NB-IoT 中，引入了通过控制面传输小数据的机制。即通过控制面的 NAS（非接入层）消息来携带小数据量业务，而不需要建立用户面承载 DRB，从而减少了用户面承载建立过程中的信令开销。这也是控制面 CIoT EPS 优化的核心所在。

在控制面 CIoT EPS 优化中，上行数据从 eNodeB 发往 MME（而不是直接发往 SGW）。接下来数据有 2 条路径可选：发往 SGW 后再发往 PGW；或发往 SCEF（Service Capability Exposure Function，只能用于

非 IP 数据包的传输)。这些节点最终将数据发送给 CIoT 应用服务器。下行数据在相反方向使用相同的路径来传输。在这种解决方案下,不会为用户数据传输建立专门的数据承载 (DRB),数据包是通过信令承载 (SRB) 来传输的,因此省去了 NAS 层和核心网建立连接的信令流程,缩短了唤醒恢复时延。但这种解决方案只适用于频率低且低速率业务的数据传输。

SCEF 是主要为机器类型数据 (machine-type data) 而设计的一个新节点,它用于在控制面上传输非 IP 数据,并为鉴权等网络服务提供了一个抽象的接口。

对于控制面 CIoT EPS 优化而言,UE 和 eNodeB 之间的数据交换是通过 RRC 信令 (更确切地说,是 RRC 信令上携带的 NAS 消息) 来发送的。在初始接入时,下行用户数据会附带在 *RRCConnectionSetup-NB* 消息中发送给 UE;上行用户数据会附带在 *RRCConnectionSetupComplete-NB* 消息中发送给 eNodeB。如果数据量较大,使用这 2 个 RRC 消息无法全部传输完,则会使用 *DLInformationTransfer-NB* 或 *ULInformationTransfer-NB* 消息继续传输用户数据。

这两类消息中都包含了带有 NAS 消息的 byte 数组,在上面介绍的情况下,该 byte 数组对应 NB-IoT 用户数据。并且对于 eNodeB 来说,该数据是透明的,RRC 层会将其直接转发给上一层 (NAS 层)。

数据到达 MME 后,会由 MME 负责 NAS 数据包与 GTP-U 数据包之间的转换。

NB-IoT 默认支持控制面 CIoT EPS 优化。对于仅使用控制面 CIoT EPS 优化的 NB-IoT UE 而言,

- 会绕过 PDCP 层的处理,即不支持 PDCP 层处理;
- 不支持 AS 安全,即不支持安全激活相关的流程;
- 不支持 RRC 连接重配置和 RRC 连接重建流程;
- 每个 UE 只有一个专用的逻辑信道 DCCH;
- 不支持逻辑信道 DTCH,也不支持 DRB;
- 不支持 RLC 重建;
- 在接入侧 (AS),不会区分不同的数据类型 (如 IP、非 IP 或 SMS 报文)。

对于用户面 CIoT EPS 优化,数据传输的方式与传统的 LTE 用户数据传输方式相同,通过无线承载并经由 SGW 和 PGW 发送给 CIoT 应用服务器,因此它存在一些建立连接的信令开销。这种解决方案适用于序列化的数据传输。该路径同时支持 IP 和非 IP 的数据传输。

对于同时支持控制面 CIoT EPS 优化和用户面 CIoT EPS 优化的 NB-IoT UE 而言,只有在 AS 安全激活之后,才会进行 PDCP 处理。

为了降低 UE 的复杂度,对于支持用户面 CIoT EPS 优化的 NB-IoT UE 而言,默认支持一个 DRB,并支持配置至多 2 个 DRB。

对于用户面 CIoT EPS 优化而言,UE 从 RRC\_IDLE 态到 RRC\_CONNECTED 态的转换过程中,MAC 层不支持 CCCH 与 DTCH 的复用。

在 LTE 中,UE 从连接态到空闲态时,eNodeB 侧和 UE 侧都会释放接入层 (AS) 上下文,并且在有新的数据要发送时,需要重新发起连接建立流程并重新建立上下文。而 NB-IoT 在用户面 CIoT EPS 优化中引

入了 RRC 连接挂起和恢复流程，从而省去了重新建立接入层上下文的过程，并简化了信令流程。在 RRC 连接挂起流程中，UE 从连接态到空闲态时，UE 会保存接入层上下文，eNodeB 会保存接入层上下文和承载相关的 S1AP 信息，MME 会保存承载相关的 S1AP 信息。在 RRC 连接恢复流程中，UE 可以利用之前的挂起流程中保存的上下文信息来重新建立 EPS 承载。

关于如何选择控制面 CIoT EPS 优化还是用户面 CIoT EPS 优化，可参见 23.401 的介绍。在后续章节中，我们还会进一步介绍 CIoT EPS 优化对接入侧的影响。

## 3.2 接入侧协议栈

图 3-2 和图 3-3 分别从下行和上行描述了 NB-IoT 无线侧协议的整体架构，以及无线承载、逻辑信道、传输信道和物理信道之间的对应关系。

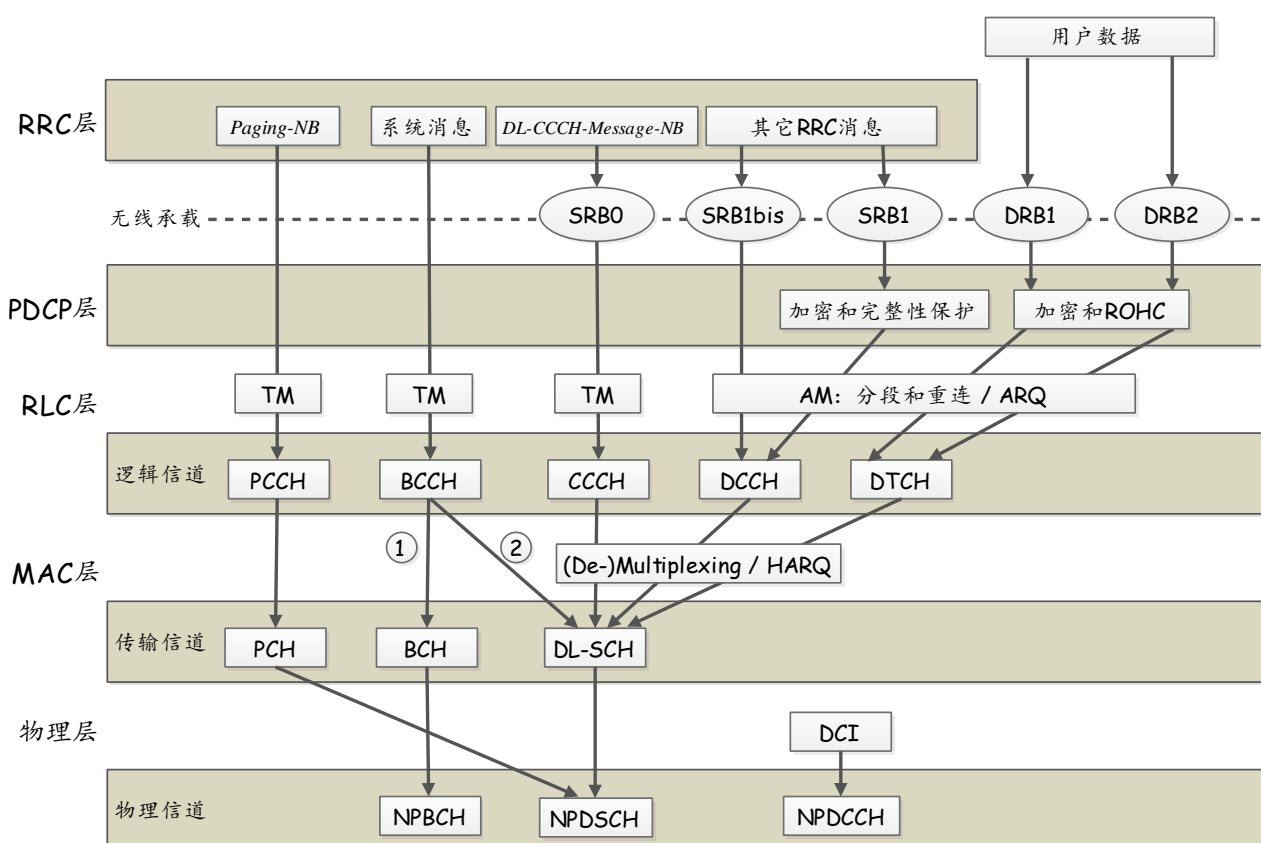


图 3-2: NB-IoT 下行无线协议架构

NB-IoT 相关的 RRC 消息主要在 36.331 的 6.7 节定义，并且在 RRC 消息名称的最后添加了“-NB”后缀，以标识该消息是用于 NB-IoT 的。

在 36.331 中搜索 *BCCH-BCH-Message-NB*，能够得到映射到逻辑信道 BCCH，并映射到传输信道 BCH 上的 RRC 消息（只有 MIB-NB）。

在 36.331 中搜索 *BCCH-DL-SCH-Message-NB*，能够得到映射到逻辑信道 BCCH，并映射到传输信道 DL-SCH 上的 RRC 消息。



在 36.331 中搜索 *DL-CCCH-Message-NB*，能够得到映射到逻辑信道 CCCH 上的 RRC 消息，这些消息使用无线承载 SRB0 进行传输，并在 RLC 层使用 TM 模式。

在 36.331 中搜索 *DL-DCCH-Message-NB*，能够得到映射到逻辑信道 DCCH 上的 RRC 消息，这些消息使用无线承载 SRB1bis 或 SRB1 进行传输，并在 RLC 层使用 AM 模式。

在 36.331 的 6.7 节搜索 SRB0、SRB1bis 或 SRB1，就可以知道对应的 RRC 消息使用的是哪种 SRB 来进行传输的。有些 RRC 消息既可能使用 SRB1bis，也可能使用 SRB1 进行传输，这就要看发送该消息时，AS 安全是否激活，如果没有激活，就使用 SRB1bis 进行传输，否则使用 SRB1 进行传输。

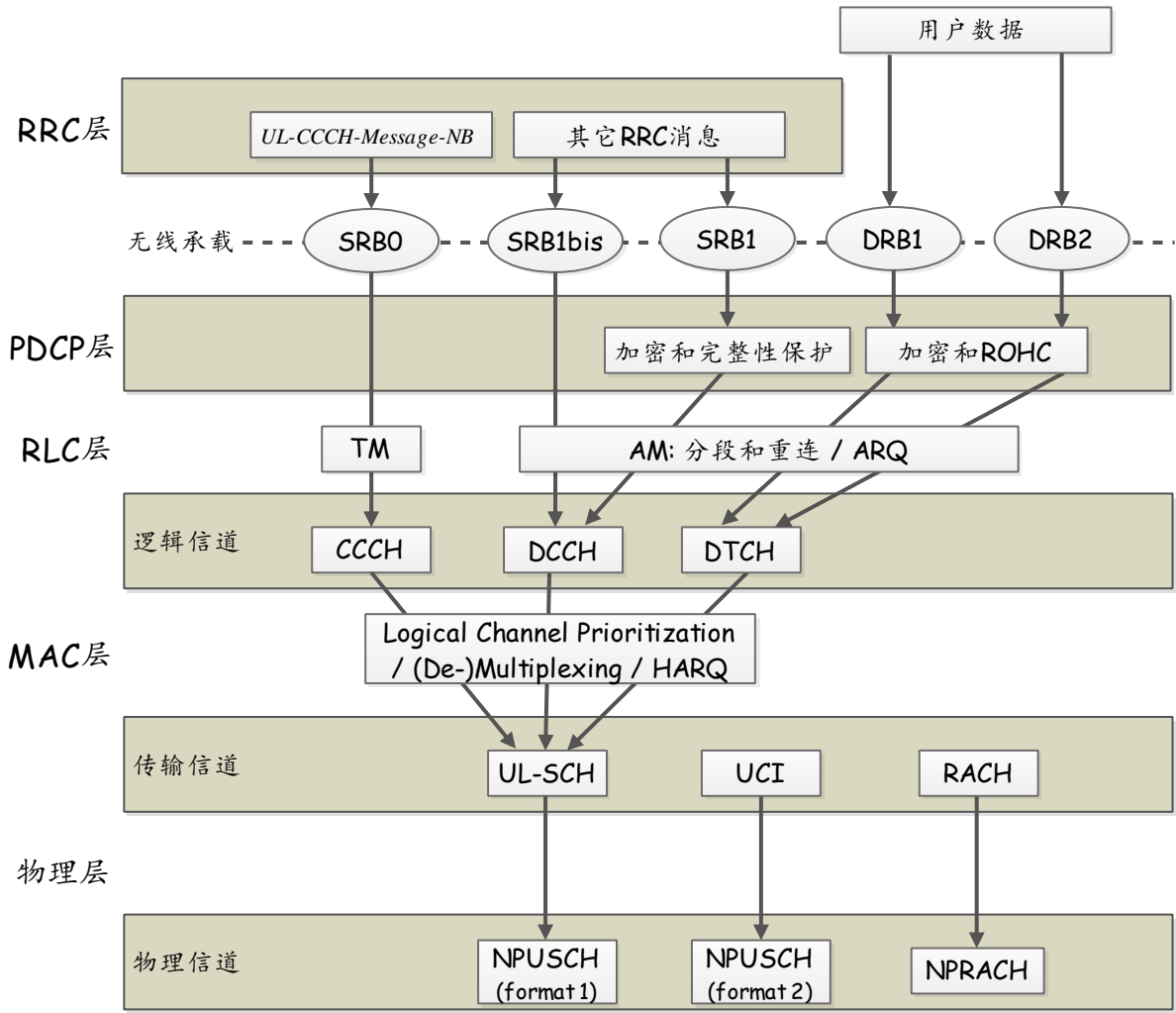


图 3-3: NB-IoT 上行无线协议架构

在 36.331 中搜索 *UL-CCCH-Message-NB*，能够得到映射到逻辑信道 CCCH 上的 RRC 消息，这些消息使用无线承载 SRB0 进行传输，并在 RLC 层使用 TM 模式。

在 36.331 中搜索 *UL-DCCH-Message-NB*，能够得到映射到逻辑信道 DCCH 上的 RRC 消息，这些消息使用无线承载 SRB1bis 或 SRB1 进行传输，并在 RLC 层使用 AM 模式。

**注：**（1）与 LTE 类似，NB-IoT 中的系统消息包括 MIB-NB 和 SIB-NB，二者都映射到逻辑信道 BCCH 上，但 MIB-NB 会在传输信道 BCH 上传输，而 SIB-NB 会在传输信道 DL-SCH 上传输。

(2) 与 LTE 相比，NB-IoT 中不存在 SRB2，但新增了无线承载 SRB1bis。具体可参见第 8 章的介绍。

(3) 图 3-2 和 3-3 中的用户数据映射到 DRB1 和 DRB2 是只针对用户面 CIoT EPS 优化而言的。在 3.1 节已经介绍过，控制面 CIoT EPS 优化中，UE 和 eNodeB 之间的数据交换是通过 RRC 消息（更确切地说，是其携带的 NAS 消息）来发送的，此时无需建立 DRB。

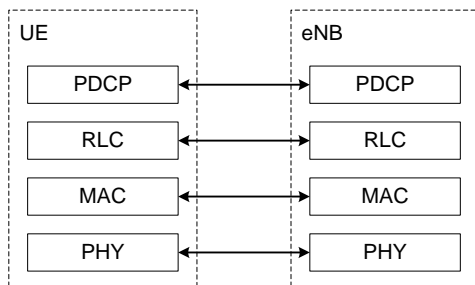
(4) 与 LTE 相比，NB-IoT 的 RLC 层只支持 TM 和 AM 模式，而不支持 UM 模式。

(5) 与 LTE 中的物理信道相对应，NB-IoT 在相应的物理信道/信号的开头加了字母 N，表示 Narrowband 的含义，例如，NB-IoT 中的 NPDCCH 的功能与 LTE 中的 PDCCH 类似，NB-IoT 中的 NPSS/NSSS 的功能与 LTE 中的 PSS/SSS 类似。但与 LTE 相比，NB-IoT 在下行不存在类似于 PCFICH 和 PHICH 的物理信道，在上行不存在类似于 PUCCH 的物理信道。在 Rel-13 中，NB-IoT 不支持多播/广播业务，因此也不存在类似于 LTE 中的逻辑信道 MCCH/MTCH、传输信道 MCH 和物理信道 PMCH 的信道。

关于 NB-IoT 针对 PDCP/RLC/MAC/物理层所做的修改，会在后续章节详细介绍。

NB-IoT 的接入侧协议栈与 LTE 相同，但根据 NB-IoT 的需求进行了精简，或在某些方面进行了加强。从协议栈的角度上看，NB-IoT 可以被认为是一种新的 RAT 技术。

NB-IoT 的**用户面协议栈**（见 36.300 的 Figure 4.3.1-1）与 LTE 相比，没有发生变化。但在使用控制面 CIoT EPS 优化时，数据是通过 NAS 来传输的，此时不使用用户面协议栈。



**Figure 4.3.1-1: User-plane protocol stack**

NB-IoT 的**控制面协议栈**（见 36.300 的 Figure 4.3.2-1）与 LTE 相比，也没有发生变化。但正如前一节的介绍，对于仅支持控制面 CIoT EPS 优化的 NB-IoT UE 而言，是不进行 PDCP 处理的；对于同时支持控制面 CIoT EPS 优化和用户面 CIoT EPS 优化的 NB-IoT UE 而言，只有在 AS 安全激活之后，才会进行 PDCP 处理。

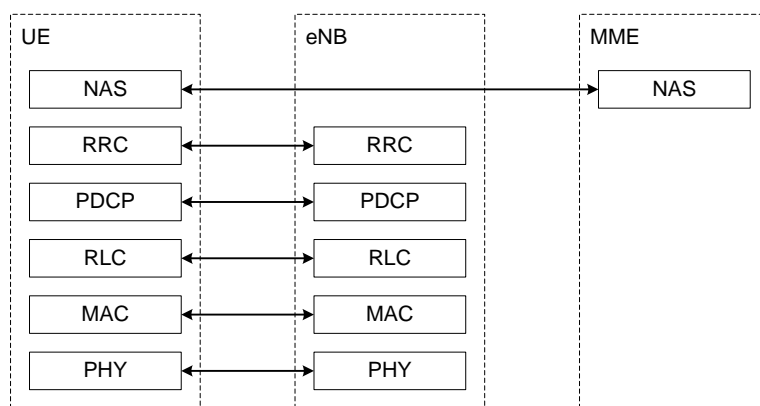


Figure 4.3.2-1: Control-plane protocol stack

### 3.2.1 PDCP 层

NB-IoT 的 PDCP 层处理基于已有的 LTE 流程和协议（见 36.323），但存在几点不同之处。

- 在 NB-IoT 中，只会为 DRB 配置 PDCP 层的参数（通过 *pdcp-Config-r13* 配置），而 SRB（其实只有 SRB1 会存在 PDCP）使用的是默认参数值，且 SRB0 和 SRB1bis 是不存在 PDCP 的。
- 在 LTE 中，PDCP 状态上报（PDCP status report）只用于 handover 的场景。而 NB-IoT 并不支持 handover，因此 NB-IoT UE 不支持 PDCP 状态上报相关的操作。
- 对于 NB-IoT 而言，支持的最大 PDCP SDU 大小和最大 PDCP 控制 PDU 大小均为 1600 字节。
- NB-IoT 只支持 7 比特的 PDCP SN。

### 3.2.2 RLC 层

NB-IoT 的 RLC 层处理同样基于已有的 LTE 流程和协议（见 36.322），但存在几点不同之处。

前面已经介绍过，仅使用控制面 CIoT EPS 优化的 NB-IoT UE 不支持 RLC 重建。

NB-IoT 在 RLC 层只支持 TM 和简化的 AM 模式，而不支持 UM 模式（NB-IoT 不支持 VoIP），且 DRB 固定使用 AM 模式。对于 NB-IoT 的 AM 模式而言，RLC 接收端是按照 *t-Reordering* 和 *t-StatusProhibit* 的值为 0 来进行处理的。*t-Reordering* 的值为 0 意味着一旦发现 RLC PDU 乱序，立即触发与 *t-Reordering* 超时相关的流程。*t-StatusProhibit* 的值为 0 意味着只要 MAC 层指示了传输机会，并需要发送 STATUS PDU 时，无需等待就立即将 STATUS PDU 发往 MAC 层。需要说明的是，这些定时器的值为 0 意味着相关定时器启动后立即超时，而不是意味着该定时器从未启动。

NB-IoT 中还定义了一个 *enableStatusReportSN-Gap-r13* 用于指示 UE 是否发送由于 RLC data PDU 接收失败而产生的状态报告（STATUS report）。如果该值为 TRUE，则 UE 的 AM 实体发现 RLC data PDU 接收失败后，立即发送一个状态报告（因为 *t-Reordering* 和 *t-StatusProhibit* 的值固定为 0）；否则 UE 不上报状态报告。

NB-IoT 也支持 polling 机制，但不支持 *pollPDU* 和 *pollByte* 触发的 polling 操作。即不支持发送端每发送 *pollPDU* 个 AMD PDU，或每发送 *pollByte* 字节数据后，主动要求对端发送一个 STATUS PDU。

NB-IoT 的 AM PDU 默认使用 10 比特的短 SN 字段长度。

### 3.2.3 MAC 层

由于 NB-IoT 只支持传输频率低且无 QoS 保证（不保证 GBR）的低速率业务，因此对 MAC 层的功能做了大量简化。并且为了进一步降低终端功耗，对连接态下的 DRX 也做了优化处理。

#### 3.2.3.1 LCID

NB-IoT 在下行仅支持以下 LCID，见 36.321 的 Table 6.2.1-1：

**Table 6.2.1-1 Values of LCID for DL-SCH**

Index	LCID values
00000	CCCH
00001-01010	Identity of the logical channel
11100	UE Contention Resolution Identity
11101	Timing Advance Command
11110	DRX Command
11111	Padding

NB-IoT 在上行仅支持以下 LCID，见 36.321 的 Table 6.2.1-2：

**Table 6.2.1-2 Values of LCID for UL-SCH**

Index	LCID values
00000	CCCH
00001-01010	Identity of the logical channel
11011	C-RNTI
11101	Short BSR
11111	Padding

NB-IoT 支持的 LCID 集合从侧面反映了 MAC 层支持的功能集合。

#### 3.2.3.2 SR

NB-IoT 中不存在类似 LTE 中的 PUCCH 信道，也不支持类似 LTE 中的 SR 消息。由于使用随机接入过程进行上行调度请求已经能够满足 NB-IoT 系统的需求，因此当 NB-IoT UE 有新的上行数据需要发送且没有被分配上行资源时，UE 会使用随机接入过程来替代 SR，向 eNodeB 请求上行资源。

### 3.2.3.3 BSR 上报

NB-IoT 中所有的逻辑信道都属于同一个 LCG (Logical Channel Group)，其不存在 Long BSR 上报。对于非使用 CCCH 进行 Msg3 传输的场景，BSR 的上报流程与 LTE 中的 BSR 上报流程是相同的。在 NB-IoT 中，一个 Padding BSR 是允许取消那些已经触发的 Regular/Periodic BSR 的。

### 3.2.3.4 上行定时提前

NB-IoT 中的上行定时提前 (Uplink timing advance) 处理与 LTE 是一致的。

### 3.2.3.5 连接态下的 DRX

NB-IoT 下，连接态下的 DRX 的配置参数如下：

```
DRX-Config-NB-r13 ::= CHOICE {
    release                NULL,
    setup                   SEQUENCE {
        onDurationTimer-r13  ENUMERATED {
            pp1, pp2, pp3, pp4, pp8, pp16, pp32, spare}, ----从一个DRX Cycle的起始
            处算起，连续监听的NPDCCH周期数
        drx-InactivityTimer-r13  ENUMERATED {
            pp0, pp1, pp2, pp3, pp4, pp8, pp16, pp32}, ----当UE 成功解码一
            个指示初传的UL或DL用户数据的NPDCCH后，持续处于激活态的连续NPDCCH周期数
        drx-RetransmissionTimer-r13  ENUMERATED {
            pp0, pp1, pp2, pp4, pp6, pp8, pp16, pp24,
            pp33, spare7, spare6, spare5,
            spare4, spare3, spare2, spare1}, ----从UE期待收到下行重传的子帧
            (HARQ RTT之后) 开始，连续监听的NPDCCH周期数
        drx-Cycle-r13          ENUMERATED {
            sf256, sf512, sf1024, sf1536, sf2048, sf3072,
            sf4096, sf4608, sf6144, sf7680, sf8192, sf9216,
            spare4, spare3, spare2, spare1}, ----对应36.321中的longDRX-Cycle，以
            子帧为单位
        drx-StartOffset-r13      INTEGER (0..255), ----对应36.321中的drxStartOffset，以子帧为单位，且步
            进为( drx-cycle / 256 )
        drx-ULRetransmissionTimer-r13  ENUMERATED {
            pp0, pp1, pp2, pp4, pp6, pp8, pp16, pp24,
            pp33, pp40, pp64, pp80, pp96,
            pp112, pp128, pp160, pp320}-----从UE期待收到上行重传的子帧 (UL
            HARQ RTT之后) 开始，连续监听的NPDCCH周期数
    }
}
```

NB-IoT 中的数据传输具有频率低、速率低的特点，UE 在接收或发送完一次数据后，应该尽快进入 DRX 状态以降低功耗。虽然 NB-IoT 重用了 LTE 的 DRX 处理机制，但针对 NB-IoT 数据传输的特点，做了几点优化：

第一点，因为 NB-IoT 中的数据传输多为非频繁发送的数据，因此不再支持短周期，即不支持 *drxShortCycleTimer*。

第二点，长周期的配置参数变为 *drx-Cycle-r13*，其可配的最大值从 Rel-12 中的 2560ms 扩大到 NB-IoT 中的 9216ms。这主要是因为，在 NB-IoT 中，用户数据传输之间的间隔通常较长，扩大 DRX 周期有利于节省 UE 的功耗。这也是连接态下的 eDRX 处理的核心所在。

第三点，为了支持覆盖增强，*onDurationTimer-r13*、*drx-InactivityTimer-r13*、*drx-RetransmissionTimer-r13* 和 *drx-ULRetransmissionTimer-r13* 这 4 个定时器的单位修改为 NPDCCH period，其中 NPDCCH period 是一个长度可动态变化的单位。在 NB-IoT 中，位于不同覆盖等级下的 UE 会配置不同的 NPDCCH 和 NPDSCH/NPUSCH 重复次数，此重复次数是由 eNodeB 动态配置的。与传统的 LTE 相比，NPDCCH 和 NPDSCH/NPUSCH 传输的持续时间随着配置的重复次数而动态变化（不再是 LTE 中固定的 1ms），同时该持续时间也变得更长，尤其是在有大量重复的场景下。因此，这 4 个 DRX 相关的定时器可配置的最大时间变长了，并且这些定时器的单位统一改成了 NPDCCH period（而在 LTE 中，DRX 相关的定时器以子帧为单位）。

第四点，与 eMTC 相同，NB-IoT 在上行也使用异步自适应 HARQ，因此 eMTC 中为上行 HARQ process 定义的 UL HARQ RTT 定时器和 *drx-ULRetransmissionTimer-r13* 同样适用于 NB-IoT，但其取值范围有所扩大。

第五点，如果 NPDCCH 指示了一个上行或下行传输（初传、重传均可），UE 会停止定时器 *drx-InactivityTimer-r13*、*drx-ULRetransmissionTimer-r13* 和 *onDurationTimer-r13*；

第六点，当 HARQ RTT Timer 或 UL HARQ RTT Timer 正在运行时，UE 不会启动 *onDurationTimer-r13*；

第七点，针对 *drx-InactivityTimer* 的启动/重启时机进行了优化。

在 LTE 的现有 DRX 机制中，UE 收到用于调度新传的 PDCCH（DL 和 UL 的均可）后，就启动或重启定时器 *drx-InactivityTimer*，其取值会考虑数据传输的时间和 HARQ 重传的时间。在 LTE 中，这个时间比较容易估计。但在 NB-IoT 中，每次传输（初传或重传）的传输时间与 TBS、覆盖等级以及部署模式相关。与此同时，不同覆盖等级下的 UE，其数据可能需要重复发送很多次以提高重复增益，数据的传输可能需要持续很长时间。还有一点就是，NB-IoT 中的上下行均使用异步自适应 HARQ，因此无法准确地判断 2 次传输之间的时间间隔，从而导致难以准确地配置 *drx-InactivityTimer* 的值。如果 NB-IoT 也使用传统 LTE 的 *drx-InactivityTimer* 机制，该值需要设置得足够长以覆盖一个 MAC PDU 的初传和所有可能的重传，从而导致设置的时间过长，无法达到省电的目的。

在 NB-IoT 中，上下行均只有一个 HARQ process，*drx-InactivityTimer* 可以等到每个 MAC PDU 的初传/重传（针对的是单次传输，而不是所有的初传/重传）结束后再启动/重启。因此在 NB-IoT 中，*drx-InactivityTimer* 在 HARQ RTT timer 或 UL HARQ RTT timer 超时的时候启动/重启。其好处就是更容易准确地配置 *drx-InactivityTimer*。例如，只要能够确定当前数据传输之后不会很快有新的数据传输，就可以将定时器 *drx-InactivityTimer* 配置成一个较小的时间值，这样 UE 就能够快速地进入 DRX 状态。

NPDCCH period（在 36.321 中称为 PDCCH period）的定义见 4.3.1.2 节的介绍。

### 3.2.3.6 数据量和功率余量上报

在 LTE 中，Msg3 中可能不发送 BSR 或 BSR 指示的数据量大小为 0，并且 BSR 只考虑了 RLC 和 PDCP 层中待发送的数据。但在 NB-IoT 中，如果 UE 只使用控制面 CIoT 优化，用户数据会作为 RRC 消息（Msg5）的一部分（NAS 消息）发往网络，因此需要在 Msg3 中发送上行数据量的大小指示，并且该数据量不仅需要包含 RLC/PDCP 层中可用于传输的所有数据，还需要包含 RRC 层中可用于传输的上行数据量，以帮助 eNodeB 正确地进行资源调度。与此同时，由于 NB-IoT 只支持小数据包传输，因此协议对 PHR 上报机制进行了简化，在 Rel-13 中，UE 只会在 Msg3 中上报 Power Headroom，后续不会再上报了。

为此，NB-IoT 引入了一个新的数据量和功率余量上报流程（Data Volume and Power Headroom reporting procedure）。该流程用于向 eNodeB 提供与 MAC 实体相关联的上行 buffer 中的可传输数据量的信息，以及用于向 eNodeB 提供 UE 最大发射功率和用于 UL-SCH 传输的估计发射功率之间的差值信息。该报告使用 DPR MAC CE 进行上报，并与 CCCH SDU 一起在 Msg3 中发送。（见 36.321 的 5.4.5a 节）

**注：**在 NB-IoT 中，使用上行 CCCH 进行传输的 RRC 消息仅包括 *RRCCConnectionRequest-NB*、*RRCCConnectionResumeRequest-NB* 和 *RRCCConnectionReestablishmentRequest-NB*。

DPR MAC CE 仅随着 Msg3 一起传输，并且该 Msg3 必须使用 CCCH 进行传输。也就是说，处于 RRC\_CONNECTED 态的 UE 由于失步而产生的随机接入过程，以及由于有上行数据需要发送而触发的随机接入过程（此时 UE 也处于 RRC\_CONNECTED 态），是不会随着 Msg3 一起发送 DPR MAC CE 的。

Data Volume and Power Headroom Report MAC Control Element，即 DPR MAC CE，由用于 CCCH MAC SDU 的 subheader 来标识，并且该 MAC CE 总是放在 CCCH MAC SDU 之前，因此 DPR MAC CE 并不需要一个额外的 subheader 来指示，也不必为该 MAC CE 定义一个专用的 LCID。（见 36.321 的 6.1.3.10 节）

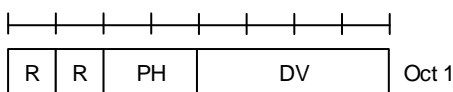


Figure 6.1.3.10-1: Data Volume and Power Headroom Report MAC control element

如 36.321 的 Figure 6.1.3.10-1 所示，DPR MAC CE 包含了 2 部分信息：

- Data Volume (DV) 字段：该字段标识了在一个 TTI 的所有 MAC PDU 已经创建之后，所有逻辑信道上可用的数据量和尚未与逻辑信道相关联的数据量的总和，并以字节数表示。它应包括 RLC/PDCP/RRC 层中可用于传输的所有数据；而各层中可用于传输的数据的定义见各层相关协议的介绍。但在计算上行 buffer 大小时，是不考虑 RLC 和 MAC 报头的大小的。该字段共 4 比特，取值见 36.321 的 Table 6.1.3.10-1。
- Power Headroom (PH) 字段：该字段指示功率余量级别，其长度为 2 比特。上报的 PH 与对应的功率余量级别之间的映射关系见 36.321 的 Table 6.1.3.10-2。

**Table 6.1.3.10-1: Data Volume levels for DV**

Index	Data Volume (DV) value [bytes]	Index	Data Volume (DV) value [bytes]
0	DV = 0	8	67 < DV ≤ 91
1	0 < DV ≤ 10	9	91 < DV ≤ 125
2	10 < DV ≤ 14	10	125 < DV ≤ 171
3	14 < DV ≤ 19	11	171 < DV ≤ 234
4	19 < DV ≤ 26	12	234 < DV ≤ 321
5	26 < DV ≤ 36	13	321 < DV ≤ 768
6	36 < DV ≤ 49	14	768 < DV ≤ 1500
7	49 < DV ≤ 67	15	DV > 1500

**Table 6.1.3.10-2: Power Headroom levels for PH**

PH	Power Headroom Level
0	POWER_HEADROOM_0
1	POWER_HEADROOM_1
2	POWER_HEADROOM_2
3	POWER_HEADROOM_3

如果 UE 在服务小区  $c$  的 NB-IoT 上行 slot  $i$  上发送了 NPUSCH，那么功率余量（Power Header）的计算公式为（见 36.213 的 16.2.1.1.2 节）：

$$PH_c(i) = P_{\text{CMAX},c}(i) - \left\{ P_{\text{O\_NPUSCH},c}(1) + \alpha_c(1) \cdot PL_c \right\} \text{ [dB]}$$

其中  $P_{\text{CMAX},c}(i)$ 、 $P_{\text{O\_NPUSCH},c}(1)$ 、 $\alpha_c(1)$  和  $PL_c$  的定义见 5.4 节的介绍。

功率余量应向下舍入到集合 {PH1, PH2, PH3, PH4} dB 中最接近的值，并发送给 MAC 层。{PH1, PH2, PH3, PH4} 的定义见 36.133 的 9.1.8.4 节的介绍。与 LTE 相比，NB-IoT 只支持 4 个等级的 PH 上报。

可以看出，与同时发送 BSR MAC CE 和 PHR MAC CE 相比，DPR MAC CE 能够节省 Msg3 的开销。

### 3.2.3.7 逻辑信道复用

NB-IoT 主要用于支持时延不敏感、无 GBR 保证（不保证 QoS）且传输频率低的业务，因此与 LTE 相比，NB-IoT 不存在 *prioritisedBitRate*、*bucketSizeDuration* 以及对应的逻辑信道优先级处理流程（不存在令牌桶处理）。也就是说，上行数据会按照逻辑信道的绝对优先级顺序被复用到 MAC PDU 中。只有当所有高优先级逻辑信道的数据都发送完毕且 UL grant 还未耗尽的情况下，低优先级的逻辑信道才能得到



服务。即此时 UE 最大化高优先级的逻辑信道的数据传输。而优先级相同的逻辑信道，则会平等地分享资源。

对于 NB-IoT 中的逻辑信道，其信道优先级 *priority* 应该根据信息重要程度的不同，考虑以下相对优先级顺序（按从高到低的顺序排列）：

- C-RNTI MAC Control Element 或来自 UL-CCCH 的数据；
- DPR MAC Control Element；
- 用于除 “padding BSR” 外的 BSR MAC Control Element；
- 除 UL-CCCH 的数据外，来自任意逻辑信道的数据：不同逻辑信道（SRB1/SRB1bis/DRB）的优先级通过 *priority-r13* 配置，值越大，优先级越低。如果没有为 SRB1/SRB1bis 设置 *priority-r13*，则其优先级使用默认值：SRB1 为 1，SRB1bis 为 3；
- 用于 “padding BSR” 的 MAC Control Element。

### 3.3 空口传输概述

与 LTE 类似，NB-IoT 中的下行（downlink）传输可以简单地理解为由 eNodeB（或者说 NB-IoT 小区）负责发送，UE 负责接收的传输；上行（uplink）传输可以简单地理解为由 UE 负责发送，eNodeB（或者说 NB-IoT 小区）负责接收的传输。

在这一节中，我们将从横向的时间顺序来介绍 NB-IoT 空口上（eNodeB 与 UE 之间）的上下行传输流程。

### 3.3.1 下行传输简介

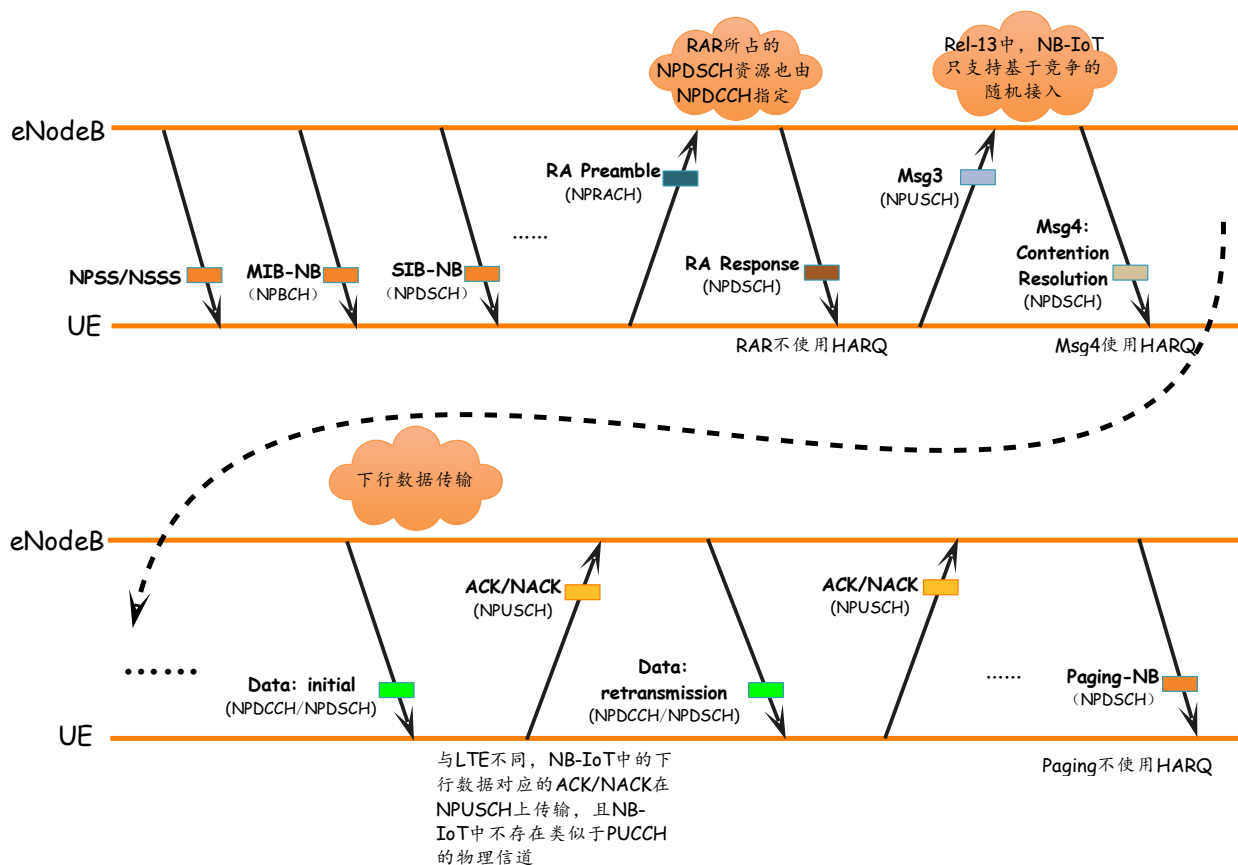


图 3-4: 下行空口流程

我们先来简单介绍一下 NB-IoT 空口传输中与下行相关的流程。

UE 在开机前并不知道小区 (cell) 是否存在，也不知道小区是如何工作的。UE 要与某个小区进行通信，首先要选择一个运营商 (如移动、联通、电信)，即选择 PLMN。选择完 PLMN 后，

1. UE 会进行小区搜索，选择一个它认为最好的小区进行驻留。这是根据 eNodeB (小区) 每隔 10 ms 发送一次的主同步信号 NPSS 和每隔 20 ms 发送一次的辅同步信号 NSSS 来决定的。通过 NPSS/NSSS，UE 能够与小区获得时间和频率上的同步 (但没有获得上行时间同步)，以及得到小区的 PCI 等。(见 7.1 节)
2. UE 确定了要进行通信的小区后，需要获取该小区的系统信息，以便获知如何在该小区上正确地工作。小区会不停地发送与该小区相关的系统信息 (MIB-NB/SIB-NB)。与 LTE 不同的是，只有处于 IDLE 态下的 NB-IoT UE 才会在需要的时候去获取这些系统信息。(见 7.2 节)
3. 获取了小区的系统信息之后，UE 就知道了该如何接入该小区，此时 UE 会发起随机接入过程以便与小区建立连接。但与 LTE 不同的是，在 Rel-13 中，NB-IoT 只支持基于竞争的随机接入过程。(见 7.3 节)
4. UE 与 eNodeB 建立起连接以后，UE 可能需要与 eNodeB 进行数据传输。eNodeB 会通过 NPDSCH 来承载它所发给 UE 的数据，并通过 NPDCCH 告诉 UE 对应的 NPDSCH 在哪些无线资源上传输以及如何传输。而 UE 需要使用 ACK/NACK 来告诉 eNodeB 它是否成功接收到了数据。此时 ACK/NACK 是通过 NPUSCH 来发给 eNodeB 的。如果 UE 没有成功接收到下行数据，eNodeB 需要重传数据。(见第 4、5、6 章等)
5. 与 LTE 不同的是，NB-IoT 在频域上的带宽只有 180kHz (相当于 LTE 中频域上的一个 RB)，可能导致一个子帧无法传输完一个 TB，因此一个 TB 可能会在时域上占用多个子帧。与此同时，为了满

足覆盖的需求，一个 TB 可能在一次传输中重复发送多次。针对一次传输内重复传输多次的同一个 TB，只会对应一个 ACK/NACK 反馈。

6. 类似的，针对一个 TB（可能重复传输多次）的 ACK/NACK 反馈也可能重复发送多次，以满足覆盖的需求。
7. 与 LTE 不同的是，NB-IoT UE 不会通过反馈 CSI 来告知 eNodeB 下行无线信道质量信息，而只会通过随机接入过程中的 NPRACH 资源选择来告诉 eNodeB 自己所处的 CE 等级。（见 7.3 节）
8. 当 UE 与 eNodeB 之间没有数据传输时，UE 并不需要一直保持在连接（RRC\_CONNECTED）状态。UE 可以处于 RRC\_IDLE 态，并每隔一段时间“醒来”一次，去接收 *Paging-NB* 消息，以确定是否有呼叫请求。eNodeB 还可以通过 *Paging-NB* 消息来告诉 UE，系统信息发生了变化。但与 LTE 相比，NB-IoT UE “睡着”的时间可以更长，以降低功耗。（见 7.4 节）

### 3.3.2 上行传输简介

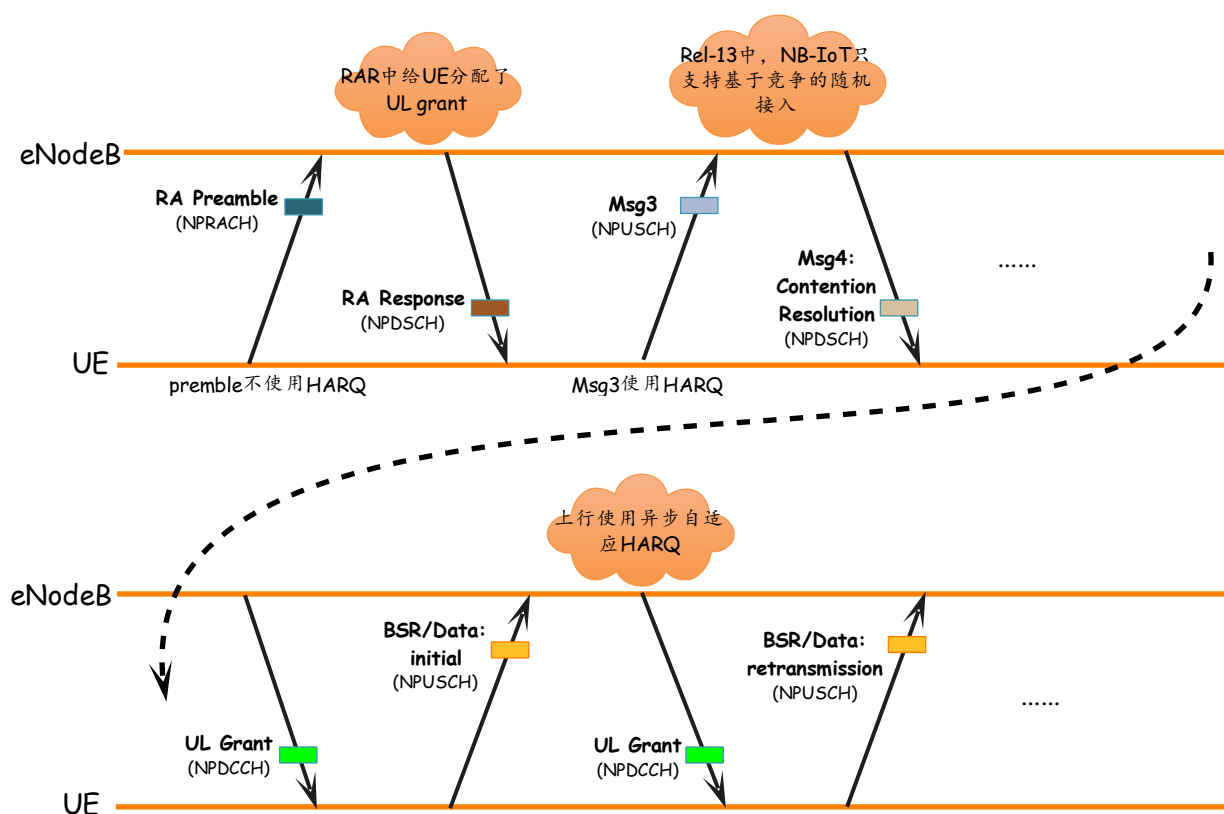


图 3-5: 上行空口流程

接下来，我们将简单介绍 NB-IoT 空口传输中与上行相关的流程。

1. 随机接入过程中，UE 在上行可能需要发送 preamble 和 Msg3 以便与小区建立起连接。UE 会通过专属于某个 CE 等级的 NPRACH 资源上发送 preamble 来告诉小区自己所处的 CE 等级。（见 7.3 节）
2. 对于上行传输，只有当 eNodeB 通过 UL grant (NPDCCH) 给 UE 分配了上行 NPUSCH 资源时，UE 才能够使用对应的资源进行上行传输。但与 LTE 不同的是，当 NB-IoT UE 没有被分配上行 NPUSCH 资源，但又有上行数据要发送时，UE 不会通过发送 SR (Scheduling Request) 来告诉 eNodeB 有数据要发送，而是需要通过发起随机接入过程来向 eNodeB 请求上行资源。

3. UE 需要告诉 eNodeB 自己有多少数据要发送，以便 eNodeB 决定给 UE 分配多少上行资源。UE 需要通过 BSR (Buffer Status Report) 或 DPR MAC CE 来告诉 eNodeB 自己有多少数据需要发送。(见 3.2.3 节)
4. UE 与 eNodeB 建立起连接以后，UE 可能需要与 eNodeB 进行数据传输。UE 会通过 NPUSCH 来承载它所发给 eNodeB 的数据。而 eNodeB 需要使用 ACK/NACK 来告诉 UE 它是否成功接收到了数据。但 NB-IoT 中不存在类似于 PHICH 的物理信道，UE 只能根据后续传输的 DCI format N0 里的 NDI 字段是否翻转来确定是进行重传还是新传。如果 eNodeB 没有成功接收到上行数据，UE 需要重传数据。与 LTE 中上行 HARQ 使用同步自适应/非自适应重传不同，NB-IoT 中上行 HARQ 只使用异步自适应重传。(见 6.2 节)
5. 与下行传输类似，一个 RU (上行传输资源映射的基本单位) 可能无法传输完一个 TB，因此一个 TB 可能会在时域上占用多个 RU。与此同时，为了满足覆盖的需求，一个 TB 可能在一次传输中重复发送多次。
6. 与 LTE 不同，NB-IoT UE 无需反馈 CSI 来告知 eNodeB 其看到的下行无线信道质量信息。这是因为 NB-IoT UE 通常静止不动或以很低的速度移动，其经历的信道变化不会很频繁，因此没有必要上报 CSI。
7. 与 LTE 不同，NB-IoT UE 不会发送 SRS，eNodeB 也无需基于 SRS 进行上行信道估计。

需要说明的是，无论是下行传输还是上行传输，为了满足覆盖的需求，所有的数据传输（包括随机接入过程中的所有传输、MIB-NB 和系统信息、NPDCCH 以及 ACK/NACK 等）都可能需要重复发送多次。但为了描述的方便，重复发送的信息并未在图 3-4 和 3-5 中体现。

### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 《窄带物联网 (NB-IoT) 标准与关键技术》，人民邮电出版社，戴博、袁戈非、余媛芳著
- [3] [《史上最全的 NB-IoT 知识》](#)
- [4] 3GPP TS 23.401 V13.9.0, 2016-12; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access
- [5] 3GPP TS 23.682 V13.8.0, 2016-12; Architecture enhancements to facilitate communications with packet data networks and applications
- [6] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [7] 3GPP TS 36.321 V13.4.0, 2016-12; Medium Access Control (MAC) protocol specification
- [8] 3GPP TS 36.322 V13.2.0, 2016-06; Radio Link Control (RLC) protocol specification
- [9] 3GPP TS 36.323 V13.4.0, 2016-12; Packet Data Convergence Protocol (PDCP) specification

- [10] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification
- [11] “R2-162332, DRX in Connected Mode”, Huawei, Hisilicon, Neul
- [12] “R2-162777, Report of the email discussion ‘MAC layer open issues’”, Ericsson
- [13] “R2-160502, BSR for early transmission of small data”, LG Electronics Inc.
- [14] “R2-163342, Report of 3GPP TSG-RAN WG2 NB-IoT Ad-hoc Meeting#2, Sophia-Antipolis, France, May 3 – 4 2016”

## 第4章 下行物理层处理

为了支持 NB-IoT，下行定义了 3 种物理信道：

- NPBCH (Narrowband Physical Broadcast CHannel)：这部分内容会在 7.2.1 节介绍；
- NPDCCH (Narrowband Physical Downlink Control CHannel)；
- NPDSCH (Narrowband Physical Downlink Shared CHannel)。

为了支持 NB-IoT，下行定义了 2 种物理信号：

- NRS (Narrowband Reference Signal)；
- NPSS (Narrowband Primary Synchronization Signal) 和 NSSS (Narrowband Secondary Synchronization Signal)：这部分内容会在 7.1 节介绍。

### 4.1 下行时频资源

NB-IoT 在下行使用 OFDM，并固定使用 15kHz 的子载波间距 (subcarrier spacing, 即  $\Delta f = 15 \text{ kHz}$ ) 和正常的循环前缀 (normal cyclic prefix)。在频域上，每个 OFDM 符号占据 12 个子载波，即在频域上占据 180kHz 的带宽。也就是说，在频域上，一个 NB-IoT 载波只占用一个 RB (Resource Block)。在时域上，7 个 OFDM 符号组成一个 slot。

下行使用 15kHz 子载波间距使得 NB-IoT 在使用带内部署或保护频带部署时，能更好地与原有 LTE 网络共存。

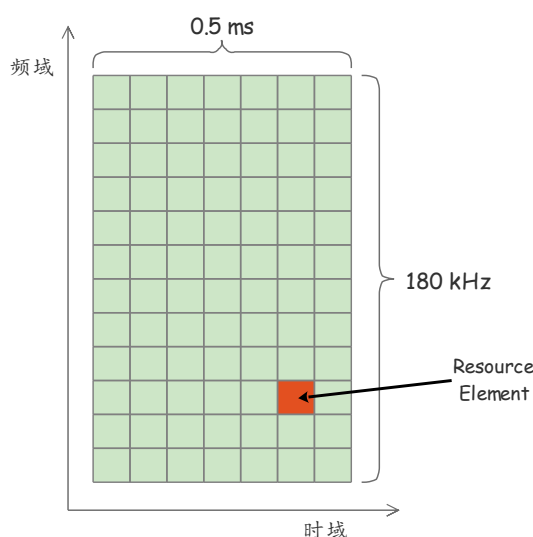


图 4-1: NB-IoT 下行时频资源网格

如图 4-1 所示，NB-IoT 中的下行时频资源网格与 LTE 中使用正常循环前缀且只使用一个 RB 时的时频资源网格是一样的，这对带内部署是很重要的。在 NB-IoT 中，一个 RE (Resource Element) 同样定义为一个 OFDM 符号内的一个子载波，等同于图 4-1 中的一个方格。

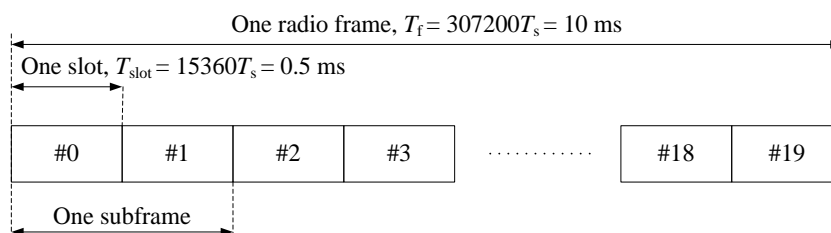


图 4-2: NB-IoT 下行帧结构

NB-IoT 的下行帧结构与 FDD LTE 类似，其每个系统帧长达 10 ms，由 10 个子帧（subframe）组成。每个子帧长达 1 ms，由 2 个连续的 slot 组成。每个 slot 长达 0.5 ms（ $T_{\text{slot}} = 15360 \cdot T_s = 0.5\text{ ms}$ ）。

下行系统帧的编号范围为 0~1023（即 1024 个系统帧为一个系统帧周期）；一个系统帧内的子帧编号范围为 0~9；一个系统帧内的 slot 编号范围为 0~19，即子帧  $i$  包含 slot  $2i$  和  $2i + 1$ 。

NB-IoT 中还定义了超帧（Hyper Frame）的概念，并用于统计系统帧周期的个数。当系统帧号环绕，即系统帧号由 1023 变为 0 时，超帧号会加一。超帧号（Hyper Frame Number）使用 10 比特表示，即一个超帧周期包含了 1024 个系统帧周期，对应差不多 3 小时的时间间隔。

超帧号的低 2 位（LSB）由 MIB-NB 的 *hyperSFN-LSB-r13* 字段配置，超帧号的高 8 位（MSB）由 SIB1-NB 的 *hyperSFN-MSB-r13* 字段配置，二者共同指定了 10 比特的超帧号。

一个 NB-IoT 载波上，并不是所有的下行子帧都能被用于 NPDCCH/NPDSCH 传输。只有满足以下几个条件的子帧才会被 UE 认为是一个 **NB-IoT 下行子帧**：（见 36.213 的 16.4 节）

- 该子帧不包含 NPSS / NSSS / NPBCH / SIB1-NB 传输；
- UE 获取到 SIB1-NB 后，读取 *downlinkBitmap-r13* 得到配置的 NB-IoT 下行子帧。如果 SIB1-NB 中不包含 *downlinkBitmap-r13* 字段，则认为除了用于传输 NPSS/NSSS/NPBCH/SIB1-NB 的子帧外的所有子帧均为 NB-IoT 下行子帧。
- 如果 UE 配置了 non-anchor carrier，则 non-anchor carrier 上的 NB-IoT 下行子帧配置通过 *downlinkBitmapNonAnchor-r13* 指示。（*downlinkBitmapNonAnchor-r13* 字段的解析见 2.4 节的介绍）

在后续章节中，大家一定要注意区分使用的是“下行子帧”，还是“NB-IoT 下行子帧”。

## 4.2 窄带参考信号（NRS）

NRS（Narrowband Reference Signal，窄带参考信号）主要用于下行物理信道的信道估计和下行信道质量测量（测量 RSRP/RSRQ 等）。每个下行 NB-IoT 天线端口对应一个 NRS，下行 NB-IoT 天线端口数为 1 或 2。

NRS 的参考信号序列生成方式与 LTE 中的 CRS 相同（具体可参考《深入理解 LTE-A》的“小区特定的参考信号”一节中的图 3-1），但其中的  $N_{\text{ID}}^{\text{cell}}$  被替换为  $N_{\text{ID}}^{\text{Ncell}}$ 。

图 4-3 从协议角度详细地描述了如何将参考信号序列映射到 RE 上。与 LTE 类似，虽然参考信号序列的长度是以 110 个 RB 来定义的，但 NB-IoT 只选取参考信号序列的中心一段（对应 1 个 RB）作为 NRS 序列来传输。

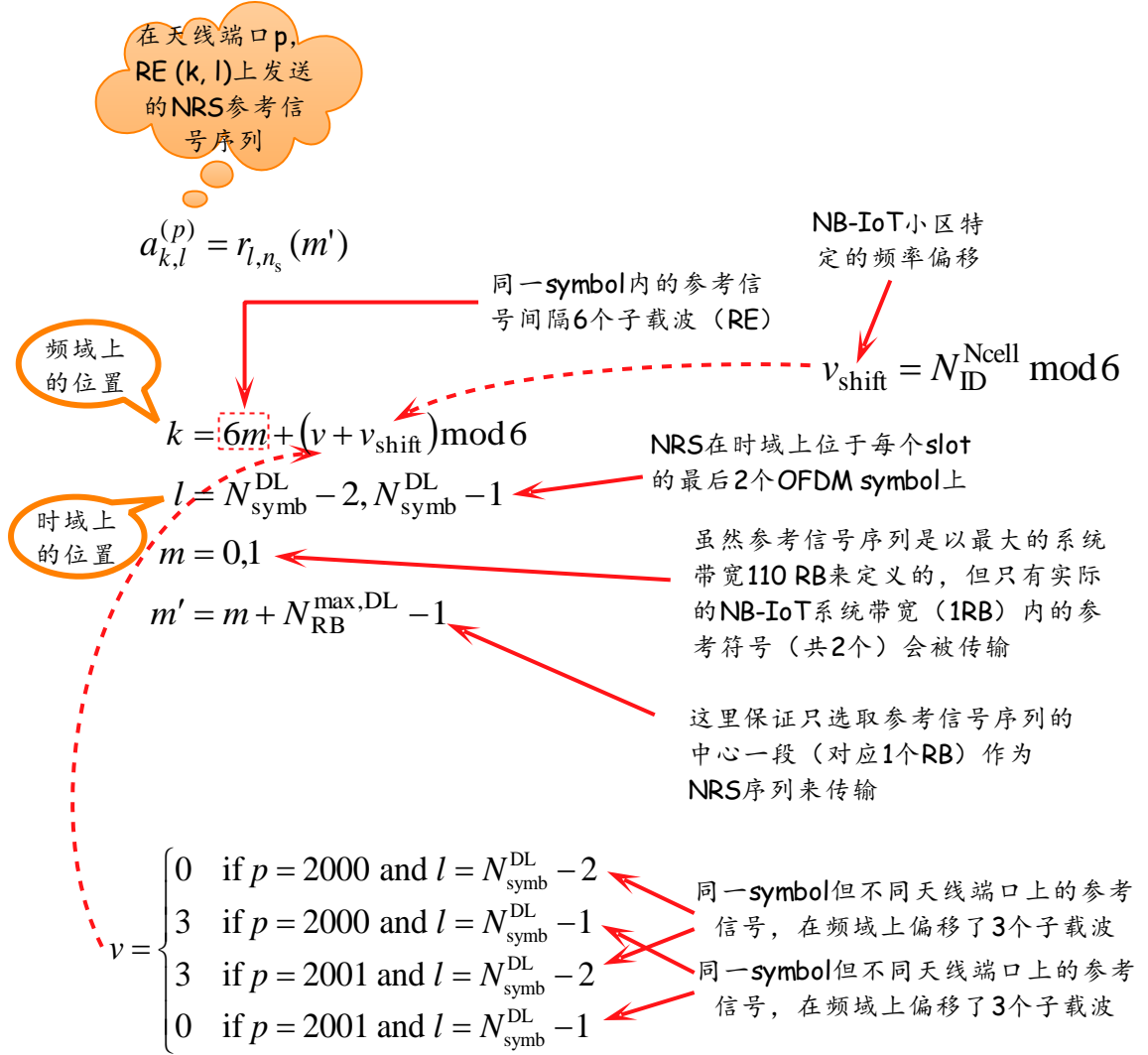


图 4-3: NRS 如何映射到 RE

传输了 NPSS/NSSS 的子帧上是不传输 NRS 的。除了无效的子帧和用于 NPSS/NSSS 的子帧外，下行窄带参考信号在 NB-IoT 天线端口 2000 和 2001 上的每个 slot 的最后 2 个 OFDM 符号上传输。

当 NB-IoT 使用单天线端口传输时，NRS 在时频资源上的映射如图 4-4 所示。



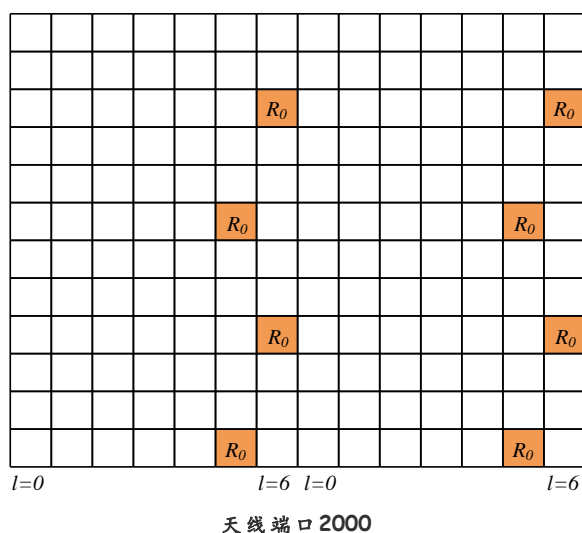


图 4-4: 单天线端口下的 NRS 映射 ( $v_{\text{shift}} = 0$ )

当 NB-IoT 使用 2 天线端口传输时, NRS 在时频资源上的映射如图 4-5 所示。

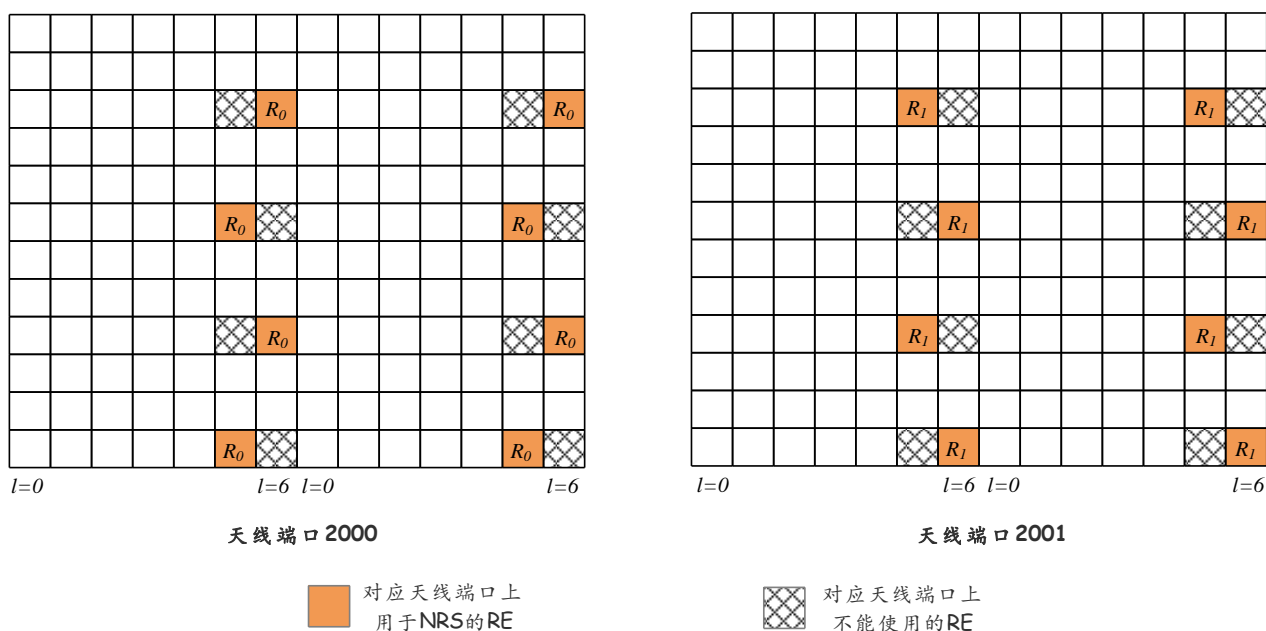


图 4-5: 2 天线端口下的 NRS 映射 ( $v_{\text{shift}} = 0$ )

每个天线端口上, 被 NRS 占据的 RE 不能用于其它传输。与此同时, 如果某个天线端口上的某个 RE 被用于发送 NRS, 则其它天线端口上时频位置相同的 RE 上不能传输任何东西。这是为了避免 NRS 被其它天线端口上的传输所干扰。

与 LTE 中的 CRS 类似, NRS 在 RB 内的起始位置与 NB-IoT 小区特定的频率偏移 (frequency shift) 相关。NB-IoT 也定义了 6 个频率偏移, 该偏移与 NB-IoT 小区的 PCI 相关, 其值为  $v_{\text{shift}} = N_{\text{ID}}^{\text{Ncell}} \bmod 6$ 。

不依赖于子帧上是否存在 NB-IoT 下行数据传输, NRS 总是在存在 MIB-NB (在子帧 0 上传输) 和 SIB1-NB (在子帧 4 上传输) 的子帧、不包含 NSSS 的子帧 9 以及 NB-IoT 下行子帧上传输, 以便 UE 使用

NRS 进行时频同步跟踪或跨子帧/多子帧间的信道估计。其中不依赖于部署模式和 NB-IoT 下行子帧配置，在子帧 0、4 以及不包含 NSSS 的子帧 9 上总是存在 NRS 传输。具体处理如下（见 36.211 的 10.2.6 节）：

1. 在 UE 获取到 NB-IoT 部署模式（部署模式通过 MIB-NB 的 *operationModeInfo-r13* 指示）之前，即接收到 MIB-NB 之前，UE 可以假定小区会在子帧 0、4 以及不包含 NSSS 的子帧 9 上发送 NRS。
2. 当 UE 接收到的 MIB-NB 的 *operationModeInfo-r13* 字段指示部署模式为保护频带部署或独立部署（对应 *operationModeInfo-r13* 的值为 *guardband-r13* 或 *standalone-r13*）时，
  - 在 UE 获取到 SIB1-NB 之前（此时不知道 NB-IoT 下行子帧配置），UE 可以假定小区会在子帧 0、1、3、4 和不包含 NSSS 的子帧 9 上发送 NRS。
  - 在 UE 获取到 SIB1-NB 之后（此时获取到 NB-IoT 下行子帧配置），UE 可以假定小区会在子帧 0、1、3、4、不包含 NSSS 的子帧 9 以及 NB-IoT 下行子帧上发送 NRS。并认为小区不会在其它下行子帧上发送 NRS。
3. 当 UE 接收到的 MIB-NB 的 *operationModeInfo-r13* 字段指示部署模式为带内部署（对应 *operationModeInfo-r13* 的值为 *inband-SamePCI-r13* 或 *inband-DifferentPCI-r13*）时，
  - 在 UE 获取到 SIB1-NB 之前，UE 可以假定小区会在子帧 0、4 和不包含 NSSS 的子帧 9 上发送 NRS。
  - 在 UE 获取到 SIB1-NB 之后，UE 可以假定小区会在子帧 0、4、不包含 NSSS 的子帧 9 以及 NB-IoT 下行子帧上发送 NRS。并认为小区不会在其它下行子帧上发送 NRS。

**注：**UE 只有在接收到 MIB-NB 后才能知道部署模式，且只有在接收到 SIB1-NB（*downlinkBitmap-r13* 字段）后才能知道哪些子帧是 NB-IoT 下行子帧。

带内部署时，UE 会被指示 NB-IoT 小区和 LTE 小区是否使用相同的小区 ID（即 PCI）。如果使用相同的小区 ID，在 NB-IoT 天线端口数与 LTE 小区特定的参考信号天线端口数相同时，UE 也可以使用 LTE 小区特定的参考信号来进行解调和/或测量。但要知道 NB-IoT 载波所占的 PRB 上的 CRS 序列信息，仅知道 PCI 是不够的，还需要知道该 NB-IoT 载波在 LTE 系统带宽内的位置，即相对于 LTE 载波中心的距离以及 NB-IoT 中心频点与最近的 100kHz 整数倍的频率之间的偏移。这是通过 *eutra-CRS-SequenceInfo-r13* 查 36.213 的 Table 16.8-1 得到的。（具体可见 7.2.1.1 节的介绍）

为什么在其它情况下，不能使用 LTE 小区特定的参考信号（CRS）来进行解调和/或测量呢？

首先需要明确一个前提，**NB-IoT UE 在任何时候都不会去读取 LTE 小区的任何信息**。也就是说，NB-IoT UE 不会去读取 LTE 小区的 PSS/SSS/MIB/SIB 以及其它 RRC 消息。否则的话，会极大地增加 NB-IoT 的系统复杂度和开销。NB-IoT UE 只能通过读取 NB-IoT 自身的消息来获取必要的 LTE 小区相关的信息，这些信息通常通过 NB-IoT 的系统消息（MIB-NB/SIB-NB）来告诉 UE。

在确定了这个前提后，我们再来讨论其它情况下，不能使用 CRS 来进行解调和/或测量的原因：

- （1）独立部署和保护频带部署时，NB-IoT 载波上不存在 CRS，也就不存在使用 CRS 来进行解调和/或测量的前提条件。
- （2）在使用带内部署，且 NB-IoT 小区与 LTE 小区使用不同的 PCI 时，NB-IoT UE 很难获取到 CRS 的参考信号序列（序列的生成与小区的 PCI 相关，而只有 NB-IoT 小区和 LTE 小区使用相同的 PCI 时，NB-IoT UE 才能知道 LTE 小区的 PCI），因此不会使用 CRS。

- (3) 在使用带内部署，且 LTE 小区的 CRS 使用 4 天线端口时，由于 NB-IoT 至多支持 2 个天线端口，如果使用 CRS，必然要引入 4 天线端口映射到 2 天线端口的天线虚拟化过程。这会带来额外的系统复杂度，因此在这种情况下，也不会使用 CRS。

### 4.3 下行物理信道

为了支持 NB-IoT，下行定义了 3 种物理信道：

- NPBCH (Narrowband Physical Broadcast CHannel)：用于传输 MIB-NB。该物理信道类似于 LTE 中的 PBCH。NPBCH 会在 7.2.1 节介绍。
- NPDCCH (Narrowband Physical Downlink Control CHannel)：用于发送下行控制信息，主要包括与接收 NPDSCH 相关的调度信息；并用于发送 UL grant，以指示 UE 如何发送 NPUSCH。该物理信道类似于 LTE 中的 PDCCH。
- NPDSCH (Narrowband Physical Downlink Shared CHannel)：用于发送单播数据、paging 消息以及除 MIB-NB 外的系统信息。该物理信道类似于 LTE 中的 PDSCH。

NB-IoT 中的下行传输信道与下行物理信道的映射关系如图 4-6 所示。

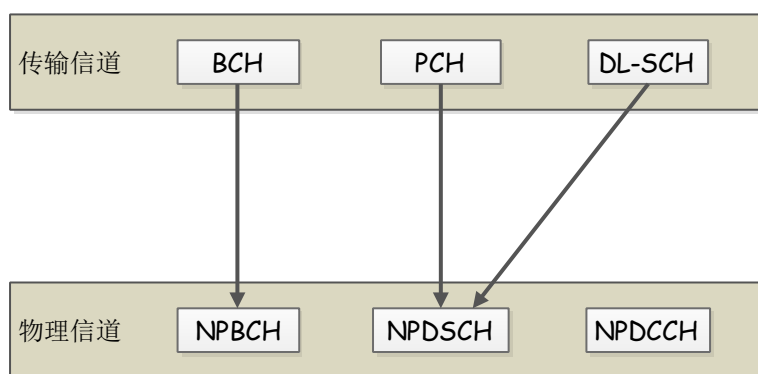


图 4-6: NB-IoT 中，下行传输信道和下行物理信道的映射关系

与 LTE 相比，NB-IoT 在下行取消了类似于 PCFICH 和 PHICH 的物理信道，并且未使用下行控制区域的概念（但在带内部署时，需要考虑 LTE 中的下行控制区域对 NB-IoT 的影响）。NPDCCH 的资源占用方式与 NPDSCH 类似，并且针对 NPUSCH 传输的 ACK/NACK 反馈信息会在 NPDCCH 中指示。又由于 NB-IoT 不支持 MBMS 传输，所以不存在类似于 PMCH 的物理信道。

NB-IoT 中的所有下行物理信道均使用 QPSK 调制。NB-IoT 支持 1 个或 2 个天线端口传输。使用 2 天线端口传输时，只支持传输分集，且只使用 SFBC (Space Frequency Block Coding)。一旦选定，NPBCH / NPDCCH / NPDSCH 将使用相同的传输样式。

为了降低 UE 的复杂度，NB-IoT 中的所有下行物理信道使用的信道编码方式均为 TBCC (Tail Biting Convolutional Coding)，且码率固定为 1/3。（见 36.212 的 Table 6.2-1）

对于 NB-IoT 中定义的下行物理信道，在使用带内部署时，其设计原则是尽量避免与 LTE 中定义的下行物理信道/信号相冲突：

- 小区特定的参考信号所占用的 RE 不能用于传输 NB-IoT 的下行物理信道/信号；

- LTE 中的下行 L1/L2 控制信道（PCFICH/PHICH/PDCCH）所占用的控制区域的 OFDM 符号不能用于传输 NB-IoT 的下行物理信道/信号；
- MBSFN 子帧不能用于传输 NB-IoT 的下行物理信道/信号；
- LTE 载波上的中心 6 个 PRB 不能用于传输 NB-IoT 的 NPSS/NSSS/NPBCH。

NB-IoT 在下行物理信道上引入了重复传输机制，通过重复传输的分集增益和合并增益来提高解调门限，更好地增强下行覆盖。

LTE 可以在时域和频域 2 个维度对用户进行调度。但 NB-IoT 在下行仅占用一个 RB，因此只能在时域上采用时分复用方式对用户进行调度。

表 4-1：NB-IoT 下行物理信道/信号与 LTE 中对应的物理信道/信号的比较

NB-IoT	物理信道/信号	说明	与 LTE 的比较
下行	NPSS	<ul style="list-style-type: none"> <li>• NPSS 序列在一个 PRB 内传输，且占用整个 PRB；</li> <li>• 所有 NB-IoT 小区使用相同的 NPSS 序列</li> </ul>	<ul style="list-style-type: none"> <li>• PSS 序列在中心的 6 个 PRB 内传输，且只占用该子帧的一个 OFDM 符号；</li> <li>• LTE 中使用 3 个 PSS 序列，对应 <math>N_{ID}^{(2)}</math></li> </ul>
	NSSS	<ul style="list-style-type: none"> <li>• NSSS 使用 ZC 序列；</li> <li>• NSSS 序列在一个 PRB 内传输，且占用整个 PRB；</li> <li>• 小区 ID（共 504 个值）在 NSSS 中指示</li> </ul>	<ul style="list-style-type: none"> <li>• SSS 使用 M 序列；</li> <li>• SSS 序列在中心的 6 个 PRB 内传输，且只占用该子帧的一个 OFDM 符号；</li> <li>• SSS 携带 168 个值，对应 <math>N_{ID}^{(1)}</math></li> </ul>
	NPBCH	使用 640ms TTI	PBCH 使用 40ms TTI
	NPDCCH	在频域上占用 1 个 PRB，在时域上可能跨多个子帧传输	PDCCH 在频域上跨越多个 PRB，在时域上位于 1 个子帧内
	NPDSCH	<ul style="list-style-type: none"> <li>• 使用 TBCC 编码并只使用一个冗余版本（rv 0）；</li> <li>• 只使用 QPSK 调制；</li> <li>• 最大 TBS 为 680 比特；</li> <li>• 只支持单层传输（单天线端口或传输分集 SFBC）</li> </ul>	<b>PDSCH:</b> <ul style="list-style-type: none"> <li>• 使用 Turbo Code 并使用多个冗余版本；</li> <li>• 可使用 QPSK/16QAM/64QAM 等调制；</li> <li>• 在非空分复用下，最大 TBS 可达到 75376 比特；</li> <li>• 可支持多层传输（单天线端口/传输分集/波束赋形/空分复用）</li> </ul>
注：与 LTE 相比，NB-IoT 在下行不存在类似于 PCFICH、PHICH 和 PMCH 的物理信道。			

在 LTE 中，下行 PDSCH 和对应的 DCI（PDCCH）通常是在同一子帧发送给 UE 的，对 UE 的解码时延要求较高。在 NB-IoT 中，NPDSCH 和对应的 DCI（NPDCCH）是在不同的子帧发送给 UE 的，对解码的时延要求低，降低了 UE 的处理能力要求。

表 4-1 简单地比较了 NB-IoT 下行物理信道/信号与 LTE 中对应的物理信道/信号的差异。

### 4.3.1 NPDCCH

NPDCCH 用于指示 NPDSCH 所占的资源及其重复传输次数。同时，NPDCCH 还用于指示用于 NPUSCH 传输的资源。此外，NPDCCH 还可用于指示用于 Paging 的资源或系统信息变更。与 LTE 类似，NB-IoT 的上下行空口资源都是由 eNodeB 负责调度的。

NPDCCH 可以使用单天线端口或 2 天线端口传输。并且 NPDCCH 和 NPDSCH 在频域上不存在复用，也就是说，NPDCCH 和 NPDSCH 不会在同一子帧上同时发送，二者是通过时分（TDM）的方式进行复用的。这与 LTE 中 PDCCH 与 PDSCH 在同一子帧上发送是不同的。

NPDCCH 支持使用 C-RNTI、TC-RNTI、P-RNTI 和 RA-RNTI 加扰。需要注意的是，与 LTE 不同，NB-IoT 中并不存在使用 SI-RNTI 加扰的 NPDCCH，这是因为 NB-IoT 中并不通过 NPDCCH 来指示如何传输系统信息，具体处理方式可参见 7.2 节的介绍。

#### 4.3.1.1 物理层处理

NPDCCH 用于携带控制信息，一个 NPDCCH 在 1 个或 2 个连续的 NCCE（Narrowband Control Channel Element）上传输。每个子帧上只定义了 2 个 NCCE 资源：NCCE 0 和 NCCE 1。一个 NCCE 由一个子帧上的 6 个连续的子载波组成，其中 NCCE 0 占用子载波 0~5，NCCE 1 占用子载波 6~11。NB-IoT 中不再使用 LTE 中的 REG 的概念。

NPDCCH 支持 2 种格式，见 36.211 的 Table 10.2.5.1-1。一个子帧上只能传输 1 或 2 个 NPDCCH：

NPDCCH format 0 只占用 1 个 NCCE（即聚合等级为 1），因此 1 个子帧上可以传输 2 个 NPDCCH format 0 的 NPDCCH；NPDCCH format 1 占用 2 个 NCCE（即聚合等级为 2），因此 1 个子帧上只能传输 1 个 NPDCCH format 1 的 NPDCCH。注意：这里的 NPDCCH format（指示资源）与 DCI format（指示内容）不是相同的概念。

Table 10.2.5.1-1: Supported NPDCCH formats

NPDCCH format	Number of NCCEs
0	1
1	2

一个 NPDCCH 使用聚合等级 2 时，其占用的 2 个 NCCE 必须位于同一个子帧上，即用于传输 NPDCCH format 1 的 2 个 NCCE 必须位于相同的子帧上。与使用聚合等级 1 相比，使用聚合等级 2 会带来更低的码率和更好的覆盖。而进一步的覆盖增强，可通过重复传输来实现。NPDCCH 的重复次数称为重复等级

$R \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$ 。对于 NPDCCH 而言，每一次重复在时域上只占用一个子帧。

NPDCCH 固定使用 QPSK 调制。

NPDCCH 在映射到 RE 时，

- 用于传输 NPBCH/NPSS/NSSS 的 RE 不能用于 NPDCCH 传输。即 anchor carrier 上发送了 NPBCH/NPSS/NSSS 的子帧不能用于 NPDCCH 传输；
- 用于传输 NRS 的 RE 不能用于 NPDCCH 传输；
- 用于 LTE 的 PBCH/PSS/SSS 或 CRS 的 RE 不能用于 NPDCCH 传输（仅在带内部署时存在）；
- 需要说明的是，在带内部署时，用于 CSI-RS 的 RE 也不会用于 NPDCCH 传输，打孔的操作由 eNodeB 完成，并且 eNodeB 不会通过信令告诉 NB-IoT UE 与 CSI-RS 相关的配置，即 UE 不知道哪些 RE 是因为 CSI-RS 的存在而被打孔的。

同时为了避免与 LTE 中用于下行控制区域的 OFDM 符号相冲突，NPDCCH 在一个子帧的第一个 slot 的起始 OFDM 符号  $l_{\text{NPDCCHStart}}$  通过如下方式确定（见 36.213 的 16.6.1 节）：

- 如果上层配置了 *eutraControlRegionSize-r13*（只有在使用带内部署时才可能存在），则  $l_{\text{NPDCCHStart}}$  由 SIB1-NB 的 *eutraControlRegionSize-r13* 字段（对应 anchor carrier），或 *DL-CarrierConfigDedicated-NB-r13* 中的 *eutraControlRegionSize-r13* 字段（对应 non-anchor carrier）指定。如果 MIB-NB 中的 *operationModeInfo-r13* 的值指示为‘00’或‘01’，即 anchor carrier 使用带内部署，并且也为 non-anchor carrier 设置了 *DL-CarrierConfigDedicated-NB-r13* 中的 *eutraControlRegionSize-r13* 字段，即 non-anchor carrier 也使用带内部署，那么此时 anchor carrier 和 non-anchor carrier 的  $l_{\text{NPDCCHStart}}$  都是由 SIB1-NB 的 *eutraControlRegionSize-r13* 字段指定的。
- 如果使用保护频带部署或独立部署（即 *operationModeInfo-r13* 的值指示为‘10’或‘11’），则控制区域的大小为 0，即  $l_{\text{NPDCCHStart}}=0$ ，从而为 NPDCCH 提供了更多的 RE 资源。

NPDCCH 在映射到 RE 上时，是按先频域后时域的顺序来映射的。

NPDCCH 只能在 NB-IoT 下行子帧上传输。如果一个子帧不是一个 NB-IoT 下行子帧，那么 NPDCCH 会推迟到下一个可用的 NB-IoT 下行子帧上发送。

另外，分配给 DL gap 的子帧也不能用于传输 NPDCCH。当 NPDCCH 的重复传输遇到 DL gap 时，对应的 NPDCCH 重复会在 DL gap 结束后继续传输。具体见 4.3.3 节的介绍。

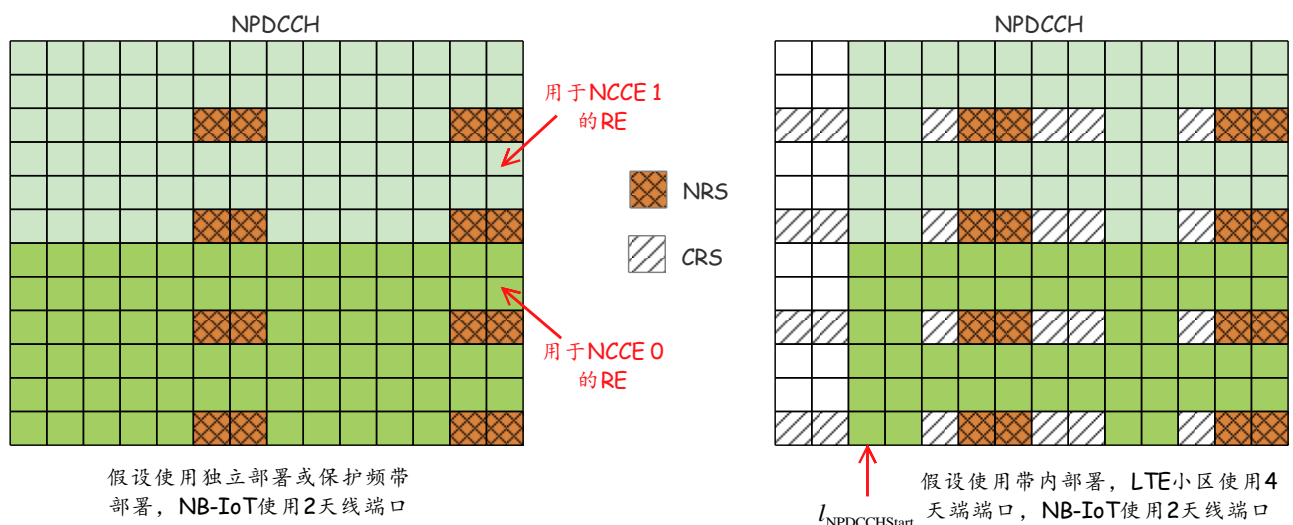


图 4-7：用于 NPDCCH 的资源映射举例

图 4-7 的左图是使用独立部署或保护频带部署，NB-IoT 使用 2 天线端口时，NPDCCH 映射的一个例子。  
图 4-7 的右图是使用带内部署，LTE 小区使用 4 天线端口且控制区域占用 2 个 OFDM symbol，NB-IoT 使用 2 天线端口时，NPDCCH 映射的一个例子。

#### 4.3.1.2 搜索空间

UE 会监听一个 NPDCCH candidate 的集合来接收对应的 DCI，这里的“监听”意味着 UE 会根据需要监听的 DCI 格式去尝试解码该集合内的每一个 NPDCCH，直到成功解码出所需的 DCI 为止。需要监听的 NPDCCH candidate 集合以 NPDCCH 搜索空间的方式定义。

为了使盲检 NPDCCH 的复杂度保持在一个合理的范围，NPDCCH 被组织成下面的搜索空间：

- Type-1 公共搜索空间（CSS for paging）：只用于 Paging；
- Type-2 公共搜索空间（CSS for random access）：只用于随机接入（包括 RAR、Msg3 的重传和 Msg4）；
- UE 特定的搜索空间（USS）。

表 4-2 给出了 NB-IoT 中不同的数据传输与其使用的 RNTI、DCI 格式以及搜索空间的对应关系。

表 4-2：NB-IoT 中数据类型与其使用的 RNTI、DCI 格式以及搜索空间的对应关系

数据类型	使用的 RNTI	DCI 格式	搜索空间
Paging	P-RNTI	DCI format N2 (对应 36.213 的 Table 16.4.1-2)	Type-1 公共搜索空间
RAR (Msg2)	RA-RNTI	DCI format N1 (对应 36.213 的 Table 16.4.1-3)	Type-2 公共搜索空间
Msg3 的重传	TC-RNTI	DCI format N0 (对应 36.213 的 Table 16.5.1-4)	Type-2 公共搜索空间
Msg4 (随机接入过程中)	TC-RNTI 或 C-RNTI	DCI format N1 (对应 36.213 的 Table 16.4.1-5)	Type-2 公共搜索空间
除 Paging 和随机接入过程外的下行数据传输	C-RNTI	DCI format N1 (对应 36.213 的 Table 16.4.1-4)	UE 特定的搜索空间
PDCCH order	C-RNTI	DCI format N1 (对应 36.213 的 Table 16.5.1-3)	UE 特定的搜索空间
除 Msg3 外的上行数据传输	C-RNTI	DCI format N0 (对应 36.213 的 Table 16.5.1-2)	UE 特定的搜索空间
注：（1）NB-IoT 中，系统消息（SIB1-NB 和 SI 消息）的传输是无需 NPDCCH 指示的，因此不存在 SI-RNTI 加扰的 NPDCCH；（2）Msg3 的初传是由 RAR 而不是 DCI 指定的，因此 Msg3 的初传不存在对应的 NPDCCH。			

UE 不会同时监听 UE 特定的搜索空间和 Type-1 公共搜索空间；UE 不会同时监听 UE 特定的搜索空间和 Type-2 公共搜索空间；UE 不会同时监听 Type-1 公共搜索空间和 Type-2 公共搜索空间。



与 LTE 中的搜索空间位于同一个子帧的控制区域内不同, NB-IoT 由于频域上只占用一个 RB, 其 NPDCCH 搜索空间会在时域上跨越多个子帧。

聚合等级  $L' \in \{1, 2\}$  和重复等级  $R \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$  上的一个 NPDCCH 搜索空间  $NS_k^{(L', R)}$  由一个 NPDCCH candidate 的集合定义。从子帧  $k$  开始, 每个 candidate 会在除了用于传输 SI 消息的子帧外的  $R$  个连续的 NB-IoT 下行子帧的集合上重复。NPDCCH 搜索空间可按以下几个步骤来确定:

**步骤一: 确定聚合等级 ( $L'$ )、重复等级 ( $R$ ) 和  $R_{\max}$  的值。**

$R_{\max}$  对应 NPDCCH 的最大重复次数, 即一个搜索空间持续的有效 NB-IoT 下行子帧数, 或者说盲检

NPDCCH 的窗口长度。定义不同的  $R_{\max}$  是为了适应不同的覆盖等级。覆盖等级值越大, 说明信道条件越差, 对应的  $R_{\max}$  就可以配置得大些, 因为给这些用户发的 NPDCCH 需要多重复发送几次。

重复等级  $R$  是 NPDCCH 真实的重复次数, 其值小于等于  $R_{\max}$ 。重复次数  $R$  的增益, 就是个简单的倍数增益。

- 对于 **UE 特定的搜索空间**而言, 聚合等级 ( $L'$ )、重复等级 ( $R$ ) 以及对应的 NPDCCH candidate 在 36.213 的 Table 16.6-1 中定义。其中  $R$  的值由  $R_{\max}$  和对应 DCI (DCI format N0/N1) 的 DCI subframe repetition number 字段共同指定。UE 特定的搜索空间的最大重复次数  $R_{\max}$  由 UE 特定的 *npdcch-NumRepetitions-r13* 指定, 且这是一个 UE 级的配置。
- 对于 **Type-1 公共搜索空间** (用于 Paging) 而言, 聚合等级 ( $L'$ )、重复等级 ( $R$ ) 在 36.213 的 Table 16.6-2 中定义。其中  $R$  的值由  $R_{\max}$  和对应 DCI (DCI format N2) 的 DCI subframe repetition number 字段共同指定。Type-1 公共搜索空间的最大重复次数  $R_{\max}$  由 SIB2-NB 的 *npdcch-NumRepetitionPaging-r13* 指定, 且这是一个小区级的配置。
- 对于 **Type-2 公共搜索空间** (用于随机接入) 而言, 聚合等级 ( $L'$ )、重复等级 ( $R$ ) 以及对应的 NPDCCH candidate 在 36.213 的 Table 16.6-3 中定义。其中  $R$  的值由  $R_{\max}$  和对应 DCI (DCI format N0/N1) 的 DCI subframe repetition number 字段共同指定。Type-2 公共搜索空间的最大重复次数  $R_{\max}$  由 *npdcch-NumRepetitions-RA-r13* 指定, 且这是一个 CE 等级特定的配置, 每个 CE 等级都有其特定的 *npdcch-NumRepetitions-RA-r13* 配置。*npdcch-NumRepetitions-RA-r13* 指定了用于 RAR、Msg3 重传以及 Msg4 的 NPDCCH 公共搜索空间的最大重复次数。



从上面的介绍可知，UE 需要成功盲检并解码 DCI 后，才能知道 NPDCCH 真实的重复次数  $R$ 。也就是说，在盲检时，UE 需要尝试解码每一个可能的重复次数  $R$  对应的 NPDCCH candidate，直到成功解码为止。这个重复次数需要被用于计算后续的 NPDSCH/NPUSCH 传输的 timing。

取决于 eNodeB 的调度，NPDCCH 真正的重复次数  $R$  可能小于最大重复次数  $R_{\max}$ ，此时剩余的子帧可被用于发送另一个 UE 的 NPDCCH。例如，假设  $R_{\max}=4$ ，那么可能某个 UE 的 DCI 就占用了所有子帧，也可能 2 个不同 UE 的 DCI 分别占用 2 个子帧等等。当然，UE 在成功解码对应的 DCI 之前，可能需要监听所有可能的 candidate。

## 步骤二：确定搜索空间。

$R_{\max}$  是搜索空间持续的时域资源长度（可用的 NB-IoT 下行子帧总数），其起始子帧为  $k_0$ 。按照 1/8、1/4、1/2 或 1 的比例来对  $R_{\max}$  进行平均分段，每个分段的大小（包含的连续 NB-IoT 下行子帧数）为  $R$ ，分段数为  $\frac{R_{\max}}{R}$ 。每个分段的索引为  $u$ ， $u$  的取值范围是  $u=0,1,\dots,\frac{R_{\max}}{R}-1$ ，而每个分段在  $R_{\max}$  内的起始索引（相对于  $k_0$  的偏移量）为  $b=u \cdot R$ 。NPDCCH（即一个 candidate）的起始子帧必然满足  $k=k_b$ ，有  $k_b$  与  $k_0$  距离为  $b=u \cdot R$ 。即 NPDCCH 的起始子帧必须为重复次数  $R$  的整数倍。

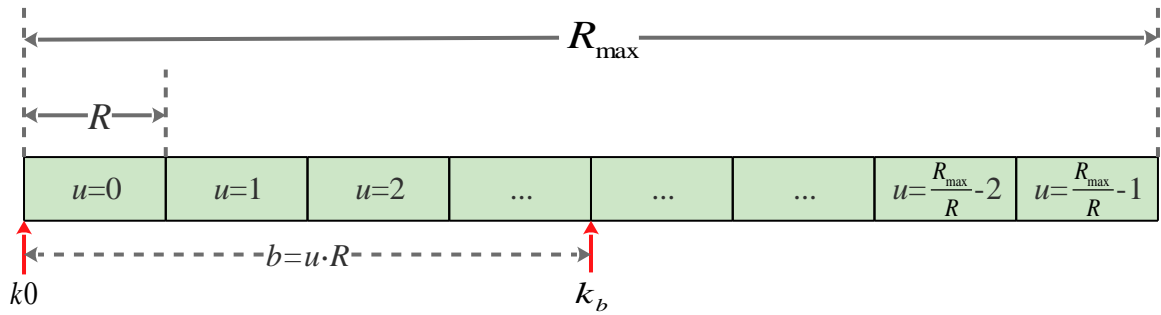


图 4-8: NPDCCH 搜索空间

$k_0$  是满足  $(10n_f + \lfloor n_s/2 \rfloor) \bmod T = \alpha_{\text{offset}} \cdot T$  的子帧，其中  $T = R_{\max} \cdot G$  且  $T \geq 4$ 。

- 对应 UE 特定的搜索空间：G 通过 *npdcch-StartSF-USS-r13* 配置， $\alpha_{\text{offset}}$  通过 *npdcch-Offset-USS-r13* 配置。通过指定搜索空间的周期  $T$  和搜索空间的起始子帧在周期内的偏移  $\alpha_{\text{offset}}$ ，不同 UE 的 NPDCCH 搜索空间之间可以进行时分复用（TDM）。
- 对应 Type-2 公共搜索空间：G 通过 *npdcch-StartSF-CSS-RA-r13* 配置， $\alpha_{\text{offset}}$  通过 *npdcch-Offset-RA-r13* 配置。

- 对应 Type-1 公共搜索空间：  $k = k_0$  并由 PO 子帧的位置决定。如何计算 PO，见 7.4.1 节的介绍。

也就是说，NB-IoT 中的 NPDCCH 搜索空间是周期性地出现的，并非每个子帧上都存在 NPDCCH 搜索空间，并且一个搜索空间在时域上跨越多个子帧。而在 LTE 中，每个子帧上都存在 PDCCH 搜索空间，并且该搜索空间限制在一个子帧内，频域上跨越整个下行系统带宽。

### 步骤三：确定 candidate 集合。

在 NB-IoT 中，NPDCCH candidate 集合是通过  $\{L, R, \#blind\ decode\}$  来定义的，其中  $L$  表示 NPDCCH 的聚合等级， $R$  表示 NPDCCH 的重复次数， $\#blind\ decode$  表示一个子帧上需要的盲检次数。例如  $\{1, 1, 2\}$  表示使用聚合等级 1，重复次数为 1 时，在一个子帧上需要盲检 2 次； $\{2, 4, 1\}$  表示使用聚合等级 2，重复次数为 4 时，在一个子帧上需要盲检 1 次。

与 LTE 中的一个 PDCCH candidate 所占的 CCE 位于同一子帧不同，NB-IoT 中的一个 NPDCCH candidate 可能跨越多个子帧。例如：当重复次数为 4 ( $R=4$ ) 时，用于重复传输 NPDCCH 的 4 个子帧组成了一个 NPDCCH candidate，虽然这 4 个子帧发送的是完全相同的内容，但不同子帧的数据能够进行合并以提供合并增益。

接下来，我们来确定 UE 特定的搜索空间 (USS)、Type-1 公共搜索空间 (CSS for paging) 和 Type-2 公共搜索空间 (CSS for random access) 内的 NPDCCH candidate 集合。

从 36.213 的 Table 16.6-1 可以看出，对于 UE 特定的搜索空间而言，当存在重复传输，即  $R$  的值大于 1 时，仅使用聚合等级 2，即  $L=2$ 。从 36.213 的 Table 16.6-2 和 Table 16.6-3 可以看出，对于 Type-1 和 Type-2 公共搜索空间而言，仅使用聚合等级 2，即  $L=2$ 。

从 36.213 的 Table 16.6-1 和 Table 16.6-3 中可以看出，UE 特定的搜索空间和 Type-2 公共搜索空间至多支持 4 种重复次数 (即  $R$ ) 取值，这里以  $R_1$ 、 $R_2$ 、 $R_3$  和  $R_4$  表示，例如：当  $R_{\max} \geq 8$  时， $R_1 = R_{\max} / 8$ ， $R_2 = R_{\max} / 4$ ， $R_3 = R_{\max} / 2$ ， $R_4 = R_{\max}$ ；当时  $R_{\max} = 4$  时， $R_1 = 1$ ， $R_2 = 2$ ， $R_3 = 4$ 。从 36.213 的 Table 16.6-2 中可以看出，Type-1 公共搜索空间至多支持 8 种重复次数 (即  $R$ ) 取值，这里以  $R_1$ 、 $R_2$ 、 $R_3$ 、 $R_4$ 、 $R_5$ 、 $R_6$ 、 $R_7$  和  $R_8$  表示。

将 36.213 的 Table 16.6-1 和 Table 16.6-3 转换成  $\{L, R, \#blind\ decode\}$  格式后分别如表 4-3 和 4-4 所示。将 36.213 的 Table 16.6-2 转换成  $\{L, R, \#blind\ decode\}$  格式后如表 4-5 所示。

表 4-3: UE 特定搜索空间

搜索空间类型	$R_{\max}$	NPDCCH candidates { $L, R, \#blind\ decode$ }	每一个子帧需要盲检的 NPDCCH candidate 数
UE 特定的搜索空间	1	{1, 1, 2}, {2, 1, 1}	无 NPDCCH 重复。聚合等级为 1 时, 需要盲检 2 个 candidate; 聚合等级为 2 时, 需要盲检 1 个 candidate, 因此 1 个子帧需要盲检 3 个 candidate。
	2	{1, 1, 2}, {2, 1, 1}, {2, 2, 1}	重复次数为 1, 聚合等级为 1 时, 需要盲检 2 个 candidate; 重复次数为 1, 聚合等级为 2 时, 需要盲检 1 个 candidate。重复次数为 2 时, 聚合等级只能为 2, 需要盲检 1 个 candidate。因此一个子帧至多需要盲检 4 个 candidate。
	4	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}	此时聚合等级固定为 2, 针对重复次数 1、2 和 4, 分别需要盲检一次, 因此一个子帧至多需要盲检 3 个 candidate。
	$\geq 8$	{2, $R_{\max}/8, 1$ }, {2, $R_{\max}/4, 1$ }, {2, $R_{\max}/2, 1$ }, {2, $R_{\max}, 1$ }	此时聚合等级固定为 2, 针对重复次数 $R_{\max}/8$ 、 $R_{\max}/4$ 、 $R_{\max}/2$ 和 $R_{\max}$ , 分别需要盲检一次, 因此一个子帧至多需要盲检 4 个 candidate。

表 4-4: Type-2 公共搜索空间 (用于随机接入)

搜索空间类型	$R_{\max}$	NPDCCH candidates { $L, R, \#blind\ decode$ }	每一个子帧需要盲检的 NPDCCH candidate 数
Type-2 公共搜索空间	1	{2, 1, 1}	聚合等级为 2, 无 NPDCCH 重复, 因此 1 个子帧只需要盲检 1 个 candidate。
	2	{2, 1, 1}, {2, 2, 1}	此时聚合等级固定为 2, 针对重复次数 1 和 2, 分别需要盲检一次, 因此一个子帧至多需要盲检 2 个 candidate。
	4	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}	此时聚合等级固定为 2, 针对重复次数 1、2 和 4, 分别需要盲检一次, 因此一个子帧至多需要盲检 3 个 candidate。
	$\geq 8$	{2, $R_{\max}/8, 1$ }, {2, $R_{\max}/4, 1$ }, {2, $R_{\max}/2, 1$ }, {2, $R_{\max}, 1$ }	此时聚合等级固定为 2, 针对重复次数 $R_{\max}/8$ 、 $R_{\max}/4$ 、 $R_{\max}/2$ 和 $R_{\max}$ , 分别需要盲检一次, 因此一个子帧至多需要盲检 4 个 candidate。

表 4-5: Type-1 公共搜索空间（用于 paging）

搜索空间类型	$R_{\max}$	NPDCCH candidates { $L, R, \#blind\ decode$ }
Type-1 公共搜索空间	1	{2, 1, 1}
	2	{2, 1, 1}, {2, 2, 1}
	4	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}
	8	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}, {2, 8, 1}
	16	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}, {2, 8, 1}, {2, 16, 1}
	32	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}, {2, 8, 1}, {2, 16, 1}, {2, 32, 1}
	64	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}, {2, 8, 1}, {2, 16, 1}, {2, 32, 1}, {2, 64, 1}
	128	{2, 1, 1}, {2, 2, 1}, {2, 4, 1}, {2, 8, 1}, {2, 16, 1}, {2, 32, 1}, {2, 64, 1}, {2, 128, 1}
	256	{2, 1, 1}, {2, 4, 1}, {2, 8, 1}, {2, 16, 1}, {2, 32, 1}, {2, 64, 1}, {2, 128, 1}, {2, 256, 1}
	512	{2, 1, 1}, {2, 4, 1}, {2, 16, 1}, {2, 32, 1}, {2, 64, 1}, {2, 128, 1}, {2, 256, 1}, {2, 512, 1}
	1024	{2, 1, 1}, {2, 8, 1}, {2, 32, 1}, {2, 64, 1}, {2, 128, 1}, {2, 256, 1}, {2, 512, 1}, {2, 1024, 1}
	2048	{2, 1, 1}, {2, 8, 1}, {2, 64, 1}, {2, 128, 1}, {2, 256, 1}, {2, 512, 1}, {2, 1024, 1}, {2, 2048, 1}

在由  $R_{\max}$  确定的搜索空间中，UE 特定的搜索空间和 Type-2 公共搜索空间的 NPDCCH candidate 集合如图 4-9 的左图所示。对应 UE 特定的搜索空间和 Type-2 公共搜索空间，UE 需要从搜索空间的起始子帧开始盲检图中的每一个  $R_i$ 。在由  $R_{\max}$  确定的搜索空间中，Type-1 公共搜索空间的 NPDCCH candidate 集合如图 4-9 的右图所示。对应 Type-1 公共搜索空间，UE 仅需要从搜索空间的起始子帧开始，盲检图中的第一个  $R_i$ 。这是因为 Paging 消息对应的搜索空间并不区分 CE 等级，其针对的可能是不同覆盖条件下的 UE，重复次数从小到大跨度较大，如果 UE 需要检测每一个  $R_i$ ，那么功耗会很大。

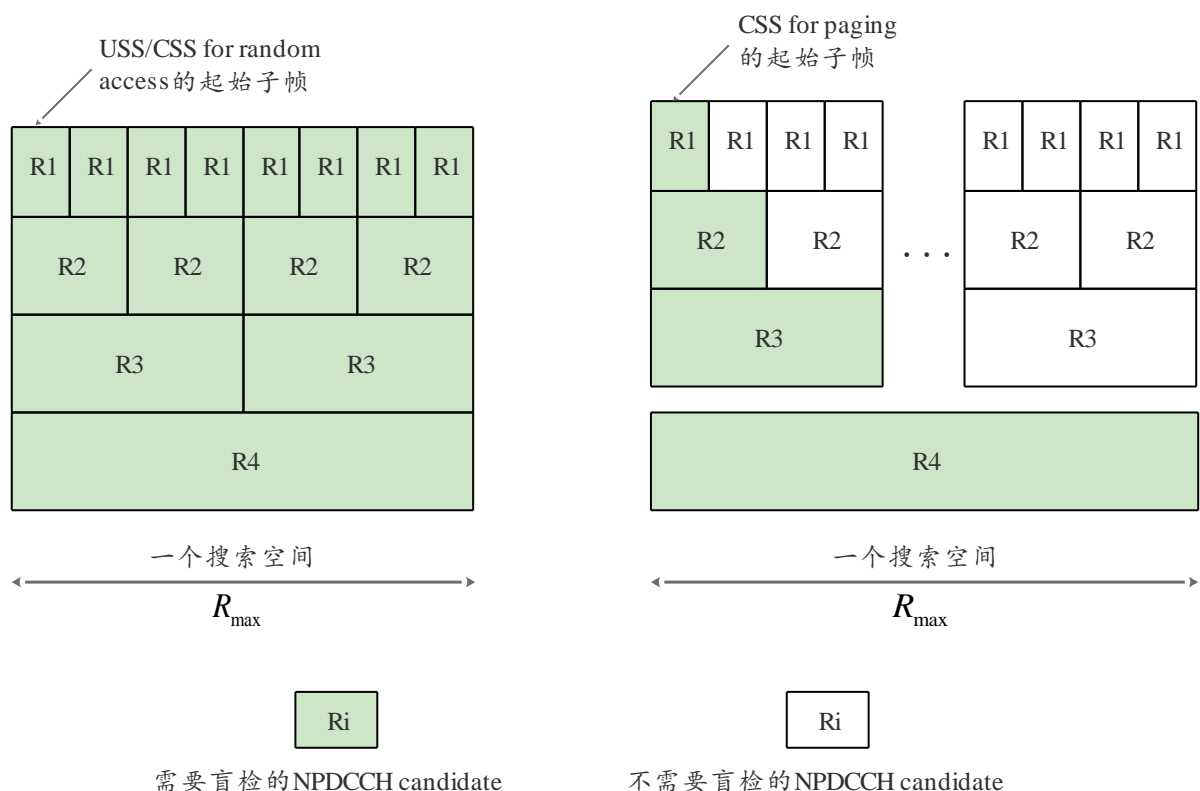


图 4-9: NPDCCH candidate 集合以及需要盲检的 NPDCCH candidate

对于一个重复次数为  $R_i$  的 NPDCCH candidate, UE 需要从搜索空间的开始到结束对每一个“ $R_i$  有效子帧”进行盲解码。

如果上层给 UE 配置了一个下行 non-anchor carrier, 则 UE 会在该 non-anchor carrier 上监听 UE 特定的搜索空间, 且 UE 不会在该载波上去接收 NPSS/NSSS/NPBCH; 否则, UE 会在与 NPSS/NSSS/NPBCH 相同的载波 (anchor carrier) 上去监听 UE 特定的搜索空间。

如果一个 NB-IoT UE 检测到

- DCI Format N0 结束于子帧  $n$ , 且对应的 NPUSCH format 1 传输起始于子帧  $n+k$ , 或
- DCI Format N1 或 N2 结束于子帧  $n$ , 且对应的 NPDSCH 传输起始于子帧  $n+k$ , 或
- DCI Format N1 结束于子帧  $n$ , 且对应的 NPUSCH format 2 传输起始于子帧  $n+k$ , 或
- 用于 PDCCH order 的 DCI Format N1 结束于子帧  $n$ , 且对应的 NPRACH 传输起始于子帧  $n+k$ ,

则 UE 从子帧  $n+1$  开始, 到子帧  $n+k-1$  结束的这段时间内, 不会去监听 NPDCCH。

如果一个 NB-IoT UE 有一个 NPUSCH 传输结束于子帧  $n$ , 则 UE 从子帧  $n+1$  开始, 到子帧  $n+3$  结束的这段时间内, 不会去监听 NPDCCH。

如果一个 NPDCCH 搜索空间的一个 NPDCCH candidate 结束于子帧  $n$ , 并且 UE 配置了监听另一个 NPDCCH 搜索空间的一个 NPDCCH candidate, 该搜索空间的起始子帧  $k_0$ , 则 NB-IoT UE 在子帧  $n+5$  之前不会去监听一个 NPDCCH 搜索空间的 NPDCCH candidate。

在 NPUSCH 上行 gap 期间, NB-IoT UE 不会去监听一个 NPDCCH 搜索空间的 NPDCCH candidates。

在这里，我们需要介绍一下 36.321 和 36.331 中经常使用到的“PDCCH 周期”的概念，NB-IoT 中有许多定时器使用 PDCCH 周期而不是子帧作为计量单位。

PDCCH 周期（PDCCH period, pp）指的是 2 个连续的 PDCCH 机会的起始位置之间的间隔，其取决于当前使用的 PDCCH 搜索空间。其中 PDCCH 机会对应搜索空间的起始位置，并且由前面介绍的子帧  $k_0$  定义。对于 NB-IoT 而言，在 36.321 和 36.331 中使用的 PDCCH period 实际上对应的就是 NPDCCH period。

以 PDCCH 周期为单位配置的定时器，其真正占用的 **PDCCH 子帧** 数的计算方式如下：

- 当 UE 使用 Type-2 公共搜索空间时，占用的 PDCCH 子帧数等于 PDCCH 周期数乘以  $npdcch-NumRepetitions-RA-r13$ ；
- 当 UE 使用 UE 特定的搜索空间时，占用的 PDCCH 子帧数等于 PDCCH 周期数乘以  $npdcch-NumRepetitions-r13$ 。

以 PDCCH 周期为单位配置的定时器，其真正占用的 **子帧** 数是通过将 PDCCH 周期数乘以两个连续的 PDCCH 机会之间的间隔时间来计算的。

#### 4.3.1.3 NPDCCH 盲检

首先，UE 知道自己处于何种状态以及在该状态下期待收到的 DCI 信息、该 DCI 的 CRC 使用哪种 RNTI 加扰以及该 DCI 所在搜索空间（见表 4-2）。例如在 IDLE 态时 UE 期待收到 Paging，就会尝试在 Type-1 公共搜索空间接收 DCI format N2；在随机接入过程中，就会尝试在 Type-2 公共搜索空间接收 DCI format N0/N1；在 RRC\_CONNECTED 态下，就会尝试在 UE 特定的搜索空间接收 DCI format N0/N1。在一个子帧上，UE 只关心一种大小的 DCI（从表 4-6 可以看出，DCI format N0 和 N1 具有相同的比特数），且该 DCI 只在一种搜索空间上传输，因此降低了盲检的复杂度。

其次，UE 知道每种 DCI 格式的大小（如表 4-6），在盲检时，UE 使用对应 DCI 格式的大小来进行 CRC 校验即可。

表 4-6：NB-IoT 中定义的 3 种 DCI 格式

DCI 格式	大小（比特数）	内容
N0	23	UE grant
N1	23	NPDSCH 调度信息； 用于 NPDCCH order 以指示随机接入过程
N2	15	用于 Paging 和直接指示

最后，根据搜索空间类型以及相关的配置，UE 能够计算出对应的搜索空间，因此 UE 知道 DCI 可能分布在哪些子帧、哪些 NCCE 上。对于不同的期望信息，UE 尝试使用相应的 X-RNTI、确定的 DCI format 及其长度、可能的聚合等级和可能的重复次数去与属于自己的搜索空间内的 NCCE 做 CRC 校验。如果 CRC 校验成功，那么 UE 就知道这个信息是自己需要的，也就得到了相应的 DCI，从而进一步解出 DCI 内容。

由于重复次数  $R$  是在成功盲检 NPDCCH 后，读取对应 DCI 中的相应字段才能得到的，所以在成功盲检 NPDCCH 之前，我们还不知道重复次数  $R$  的值。

成功盲检了 NPDCCH，再结合搜索空间的计算方式，UE 就能够知道 NPDCCH 的起始子帧，再加上获取到的重复次数信息，UE 就能够知道 NPDCCH 的结束子帧。NPDSCH/NPUSCH 的起始位置是基于 NPDCCH 的结束子帧来计算的，具体可见后续介绍。

UE 成功接收 NPDCCH 后，通过如下方式来区分不同的 DCI 格式：

- 如果 NPDCCH 的 CRC 使用 P-RNTI 加扰，则收到的是 DCI format N2；
- 如果 NPDCCH 的 CRC 使用 C-RNTI 加扰，则收到的 DCI 的第一个比特（对应 Flag for format N0/format N1 differentiation 字段）指示使用的是 DCI format N0 还是 DCI format N1；
- 如果 NPDCCH 的 CRC 使用 RA-RNTI 加扰，则收到的是 DCI format N1，其内容只包含了用于 RAR 所需的字段。

#### 4.3.1.4 DCI 格式

由于 NB-IoT 只支持单天线端口传输或传输分集，因此其支持的 DCI 格式相对于 LTE 来说，简化了不少：针对上行调度（UL grant），仅支持 DCI format N0 一种格式。针对下行传输，支持 DCI format N1 和 DCI format N2 两种格式。

接下来，我们将介绍 36.212 的 6.4.3 节中定义的用于 NB-IoT 的各种 DCI 格式。并针对各种 DCI 的应用场景予以分别介绍。

DCI format N0: 指示 NPUSCH 的调度信息，即 UL grant。（共 23 比特）		
字段	占用的比特数	作用
Flag for format N0/format N1 differentiation	1 比特	值为 0，则此 DCI 为 DCI format N0；值为 1，则此 DCI 为 DCI format N1
Subcarrier indication	6 比特	对应 $I_{sc}$ ，用于计算 NPUSCH 所占的子载波，具体见 5.3.1.1 节的介绍（或见 36.213 的 16.5.1.1 节）
Resource assignment	3 比特	对应 $I_{RU}$ ，用于计算一个上行 TB 需要占用的 RU 个数，具体见 5.3.1.1 节的介绍（或见 36.213 的 16.5.1.2 节）
Scheduling delay	2 比特	对应 $I_{Delay}$ ，用于计算 $k_0$ ，具体见 5.3.1.1 节的介绍（或见 36.213 的 16.5.1 节）
Modulation and coding scheme	4 比特	对应 $I_{MCS}$ ，用于计算 NPUSCH format 1 使用的调制阶数和 $I_{TBS}$ ，具体见 5.3.1.2 节的介绍（或见 36.213 的 16.5.1.2 节）
Redundancy version	1 比特	对应 $rv_{DCI}$ ，用于计算 NPUSCH format 1 使用的冗余版本，具体见 5.3.1.2 节的介绍（或见 36.213 的 16.5.1.2 节）

Repetition number	3 比特	对应 $I_{\text{Rep}}$ ，用于计算 NPUSCH format 1（或者说上行 TB）的重复次数 $N_{\text{Rep}}$ ，具体见 5.3.1.1 节的介绍（或见 36.213 的 16.5.1.1 节）
New data indicator	1 比特	NDI，用于指示是重传还是初传，并间接地指示针对上行传输的 ACK/NACK 信息，具体见 6.2 节的介绍
DCI subframe repetition number	2 比特	与 $R_{\text{max}}$ 一起指定此 DCI 的重复次数 $R$ ，具体见 4.3.1.2 节的介绍（或见 36.213 的 16.6 节）

DCI format N1: 作为 NPDCCH order 指示随机接入过程（共 23 比特）		
字段	占用的比特数	作用
Flag for format N0/format N1 differentiation	1 比特	值为 0，则此 DCI 为 DCI format N0；值为 1，则此 DCI 为 DCI format N1
NPDCCH order indicator	1 比特	值为 1，则表示此 DCI 用于由 NPDCCH order 触发的随机接入过程，并且其 CRC 使用 C-RNTI 加扰
Starting number of NPRACH repetitions	2 比特	对应 $I_{\text{Rep}}$ ，用于计算 NPRACH 的重复次数，具体见 7.3.3 节的介绍（或见 36.213 的 16.3.2 节）
Subcarrier indication of NPRACH	6 比特	对应 $I_{\text{sc}}$ ，分配给 NPRACH 的子载波 $n_{\text{sc}} = I_{\text{sc}}$ ，具体见 7.3.3 节的介绍（或见 36.213 的 16.3.2 节）
剩余的比特	13 比特	预留，所有比特均设置为 1

DCI format N1: 指示 NPDSCH 的调度信息（共 23 比特）		
字段	占用的比特数	作用
Flag for format N0/format N1 differentiation	1 比特	值为 0，则此 DCI 为 DCI format N0；值为 1，则此 DCI 为 DCI format N1
NPDCCH order indicator	1 比特	值为 0。（值为 1，则表示此 DCI 用于由 NPDCCH order 触发的随机接入过程，并且其 CRC 使用 C-RNTI 加扰）
Scheduling delay	3 比特	对应 $I_{\text{Delay}}$ ，用于计算 $k_0$ ，具体见 4.3.2.2 节的介绍（或见 36.213 的 16.4.1 节）
Resource assignment	3 比特	对应 $I_{\text{SF}}$ ，用于计算一个下行 TB 需要占用的下行子帧数 $N_{\text{SF}}$ 。具体见 4.3.2.2 节的介绍（或见 36.213 的 16.4.1.3 节）



Modulation and coding scheme	4 比特	对应 $I_{\text{MCS}}$ ，并有 $I_{\text{TBS}} = I_{\text{MCS}}$ 。具体见 4.3.2.3 节的介绍（或见 36.213 的 16.4.1.5 节）
Repetition number	4 比特	对应 $I_{\text{Rep}}$ ，用于计算 NPDSCH（或者说下行 TB）的重复次数 $N_{\text{Rep}}$ 。具体见 4.3.2.2 节的介绍（或见 36.213 的 16.4.1.3 节）
New data indicator	1 比特	NDI，用于指示是重传还是初传
HARQ-ACK resource	4 比特	用于指定 $k_0$ 的值以及分配给 ACK/NACK（NPUSCH format 2）的子载波。具体见 6.1 节的介绍（或见 36.213 的 16.4.2 节）
DCI subframe repetition number	2 比特	与 $R_{\text{max}}$ 一起指定此 DCI 的重复次数 $R$ ，具体见 4.3.1.2 节的介绍（或见 36.213 的 16.6 节）
注：如果 DCI format N1 的 CRC 使用 RA-RNTI 加扰，则 New data indicator 和 HARQ-ACK resource 字段是预留的。		

DCI format N2：用于 direct indication（共 15 比特）		
字段	占用的比特数	作用
Flag for paging/direct indication differentiation	1 比特	值为 0，则此 DCI 用于 direct indication；值为 1，则此 DCI 为用于 paging。（这里假设值为 0）
Direct Indication information	8 比特	用于直接指示系统信息是否发生了变化，其每个比特的含义见 36.331 的 6.7.5 节。具体见 7.2.4 节的介绍。
剩余的比特	6 比特	预留

DCI format N2：指示 Paging（共 15 比特）		
字段	占用的比特数	作用
Flag for paging/direct indication differentiation	1 比特	值为 0，则此 DCI 用于 direct indication；值为 1，则此 DCI 为用于 paging。（这里假设值为 1）
Resource assignment	3 比特	对应 $I_{\text{SF}}$ ，用于计算一个下行 TB 需要占用的下行子帧数 $N_{\text{SF}}$ 。具体见 4.3.2.2 节的介绍（或见 36.213 的 16.4.1.3 节）
Modulation and coding scheme	4 比特	对应 $I_{\text{MCS}}$ ，并有 $I_{\text{TBS}} = I_{\text{MCS}}$ 。具体见 4.3.2.3 节的介绍（或见 36.213 的 16.4.1.5 节）

Repetition number	4 比特	对应 $I_{\text{Rep}}$ ，用于计算 NPDSCH（或者说下行 TB）的重复次数 $N_{\text{Rep}}$ 。具体见 4.3.2.2 节的介绍（或见 36.213 的 16.4.1.3 节）
DCI subframe repetition number	3 比特	与 $R_{\text{max}}$ 一起指定此 DCI 的重复次数 $R$ ，具体见 4.3.1.2 节的介绍（或见 36.213 的 16.6 节）

## 4.3.2 NPDSCH

NPDSCH 用于传输 Paging 消息、SIB1-NB、SI 消息、RAR MAC PDU、RRC 消息以及下行用户数据。

NPDSCH 支持使用 C-RNTI、TC-RNTI、P-RNTI、SI-RNTI 和 RA-RNTI 加扰。

### 4.3.2.1 物理层处理

NPDSCH 的物理层处理流程类似于 LTE 中的 PDSCH，但做了一些简化。如图 4-10 所示。

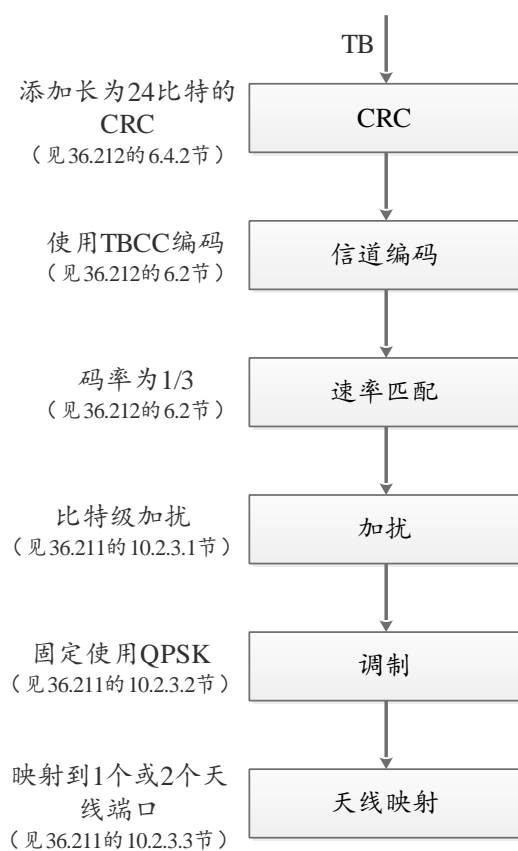


图 4-10: NPDSCH 的物理层处理

物理层在收到来自 MAC 层的 TB 后，会在 TB 的末尾附加上一个 24 比特的 CRC，以便接收端进行错误检测。NB-IoT 中不存在类似于 LTE 中的码块分段，并为每个码块添加一个 CRC 的过程。

NPDSCH 使用的信道编码方式为 TBCC (Tail Biting Convolutional Coding)，且码率固定为 1/3。

NPDSCH 的加扰会以 36.211 的 6.3.1 节介绍的方式进行，并且加扰序列生成器会被初始化为

$$c_{\text{init}} = n_{\text{RNTI}} \cdot 2^{14} + n_f \bmod 2 \cdot 2^{13} + \lfloor n_s / 2 \rfloor \cdot 2^9 + N_{\text{ID}}^{\text{Ncell}}, \text{ 其中 } n_s \text{ 为传输该 codeword 的第一个 slot。对于}$$

NPDSCH 重复和携带 BCCH 的 NPDSCH，每次重复都会使用该表达式重新初始化加扰序列生成器。

NPDSCH 固定使用 QPSK 调制。

NPDSCH 可以使用单天线端口或 2 天线端口传输。NPDSCH 在使用 2 天线端口传输时，只支持传输分集，且只使用 SFBC。由于 NPDSCH 传输不支持波束赋形和空分复用，因此不需要支持 LTE 中定义的各种复杂的传输模式 (TM 模式)。

NPDSCH 在映射到 RE 时，

- 用于传输 NPBCH/NPSS/NSSS 的 RE 不能用于 NPDSCH 传输。即 anchor carrier 上发送了 NPBCH/NPSS/NSSS 的子帧不能用于 NPDSCH 传输。
- 用于传输 NRS 的 RE 不能用于 NPDSCH 传输；
- 用于 LTE 的 CRS 的 RE 不能用于 NPDSCH 传输（仅在带内部署时存在）。

与 NPDCCH 类似，为了避免与 LTE 中用于下行控制区域的 OFDM 符号相冲突，除 SIB1-NB 外的 NPDSCH 传输在一个子帧的第一个 slot 的起始 OFDM 符号  $l_{\text{DataStart}}$  通过如下方式确定（见 36.213 的 16.4.1.4 节）：

- 如果上层配置了 *eutraControlRegionSize-r13*（只有在使用带内部署时才可能存在），则  $l_{\text{DataStart}}$  由 SIB1-NB 的 *eutraControlRegionSize-r13* 字段（对应 anchor carrier），或 *DL-CarrierConfigDedicated-NB-r13* 中的 *eutraControlRegionSize-r13* 字段（对应 non-anchor carrier）指定。如果 MIB-NB 中的 *operationModeInfo-r13* 的值指示为‘00’或‘01’，即 anchor carrier 使用带内部署，并且也为 non-anchor carrier 设置了 *DL-CarrierConfigDedicated-NB-r13* 中的 *eutraControlRegionSize-r13* 字段，即 non-anchor carrier 也使用带内部署，那么此时 anchor carrier 和 non-anchor carrier 的  $l_{\text{DataStart}}$  都是由 SIB1-NB 的 *eutraControlRegionSize-r13* 字段指定的。
- 如果使用保护频带部署或独立部署（即 *operationModeInfo-r13* 的值指示‘10’或‘11’），则控制区域的大小为 0，即  $l_{\text{DataStart}} = 0$ 。从而为 NPDSCH 提供了更多的 RE 资源。

而 SIB1-NB 在一个子帧的第一个 slot 的起始 OFDM 符号  $l_{\text{DataStart}}$  的确定方式见 7.2.2 节的介绍。

NPDSCH 与 NPDCCH 的子帧结构类似，如图 4-11 所示。（用于携带 SIB1-NB 的 NPDSCH 的资源映射见 7.2.2 节的介绍）

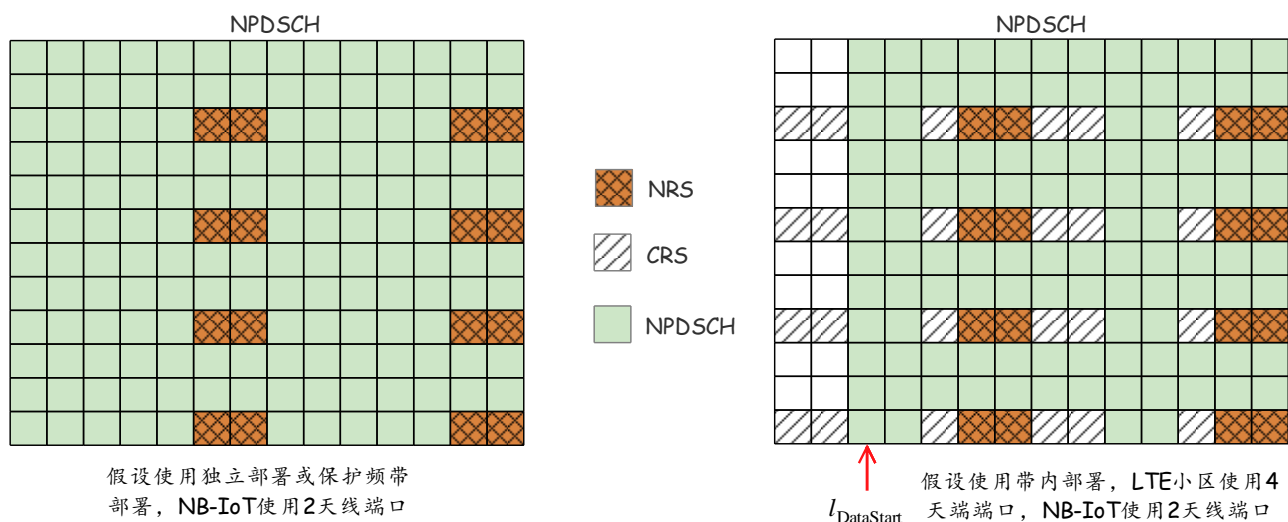


图 4-11：除用于发送 SIB1-NB 外的 NPDSCH 的资源映射举例

图 4-11 的左图是使用独立部署或保护频带部署，NB-IoT 使用 2 天线端口时，NPDSCH 映射的一个例子。图 4-11 的右图是使用带内部署，LTE 小区使用 4 天线端口，NB-IoT 使用 2 天线端口时，NPDSCH 映射的一个例子。

除子帧 4 上用于传输 SIB1-NB 的 NPDSCH 外，NPDSCH 只能在 NB-IoT 下行子帧上传输。如果一个子帧不是一个 NB-IoT 下行子帧，则 NPDSCH 会推迟到下一个可用的 NB-IoT 下行子帧上发送。

#### 4.3.2.2 时频资源分配

在 LTE 中，PDCCH 和对应的 PDSCH 传输（不考虑 SPS 的特殊情况）是在同一个下行子帧上发送的。而在 NB-IoT 中，NPDCCH 和 NPDSCH 并不在同一个下行子帧上发送，而是有一定的延时，这就降低了对硬件处理能力的要求，从而减低了成本。

本节内容介绍的是除系统信息（SIB1-NB 和 SI 消息）外的 NPDSCH 传输的时频资源分配。而系统信息的时频资源分配见 7.2.2 节和 7.2.3 节的介绍。

首先，我们来介绍 NPDSCH 在时域上的 timing 关系和资源分配。

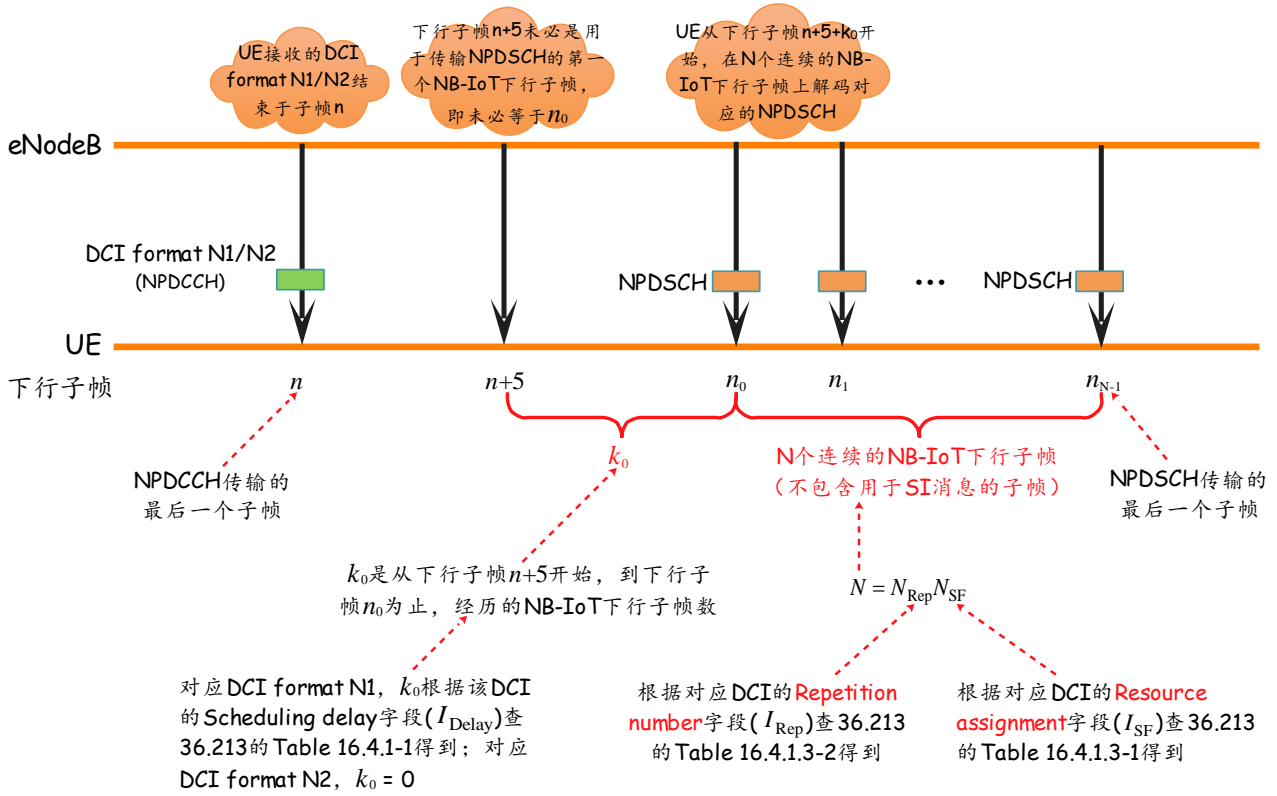


图 4-12: 除 SIB1-NB 外的 NPDSCH 传输在时域上的资源分配

假设 UE 接收的 DCI format N1, N2 结束于子帧  $n$ ，那么 UE 会从下行子帧  $n+5$  开始，根据 DCI 中携带的信息，在  $N$  个连续的 NB-IoT 下行子帧  $n_i$  (其中  $i=0, 1, \dots, N-1$ ) 上解码对应的 NPDSCH 传输。其中，

- 子帧  $n$  是 NPDCCH 传输的最后一个子帧，并由 NPDCCH 传输的起始子帧和对应 DCI 里的“DCI subframe repetition number”字段共同决定。具体的计算方式见 4.3.1 节的介绍。
- 子帧  $n_i$  ( $i=0, 1, \dots, N-1$ ) 是  $N$  个连续的 NB-IoT 下行子帧，不包含用于 SI 消息的子帧在内且有  $n_0 < n_1 < \dots, n_{N-1}$ 。
- $N = N_{\text{Rep}} N_{\text{SF}}$ 。一个 TB 映射到  $N_{\text{SF}}$  个子帧，同一 TB 会重复传输  $N_{\text{Rep}}$  次，因此一次 NPDSCH 传输

(对应同一个 TB) 需要占用  $N = N_{\text{Rep}} N_{\text{SF}}$  个 NB-IoT 下行子帧。 $N_{\text{SF}}$  通过对应 DCI 的 Resource

assignment 字段 ( $I_{\text{SF}}$ ) 查 36.213 的 Table 16.4.1.3-1 得到。 $N_{\text{Rep}}$  通过对应 DCI 的 Repetition number 字段

( $I_{\text{Rep}}$ ) 查 36.213 的 Table 16.4.1.3-2 得到。

**Table 16.4.1.3-1: Number of subframes ( $N_{\text{SF}}$ ) for NPDSCH.**

$I_{\text{SF}}$	$N_{\text{SF}}$
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

**Table 16.4.1.3-2: Number of repetitions ( $N_{\text{Rep}}$ ) for NPDSCH.**

$I_{\text{Rep}}$	$N_{\text{Rep}}$
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	192
9	256
10	384
11	512
12	768
13	1024
14	1536
15	2048

- $k_0$  是从下行子帧  $n+5$  开始，到下行子帧  $n_0$  为止，间隔的 NB-IoT 下行子帧数。对应 DCI format N1,  $k_0$  由对应 DCI 的 “Scheduling delay” 字段 ( $I_{\text{Delay}}$ ) 查 36.213 的 Table 16.4.1-1 得到，其中对应 DCI format N1 的  $R_{\text{max}}$  的计算见 4.3.1.2 节的介绍。对应 DCI format N2,  $k_0 = 0$ 。

**Table 16.4.1-1:  $k_0$  for DCI format N1.**

$I_{\text{Delay}}$	$k_0$	
	$R_{\text{max}} < 128$	$R_{\text{max}} \geq 128$
0	0	0
1	4	16
2	8	32
3	12	64
4	16	128
5	32	256
6	64	512
7	128	1024

从前面的介绍可以看出，NPDSCH 传输的起始时间与其相关的 DCI format N1/N2 的结束时间之间的间隔会大于等于 4ms。同时，针对同一 UE，NPUSCH 传输结束后紧接着的 3 个下行子帧，UE 是不会去接收下行传输的。

一个下行传输块（TB）在时域上可能会在一个或多个 NB-IoT 下行子帧上被调度。也就是说，一个单一的 TB 可能需要使用多个子帧来传输，这也是  $N_{\text{SF}}$  的物理意义，即一个 TB 被调度在  $N_{\text{SF}}$  个子帧上传输。这与 LTE 中，一个 TB 通常只在一个子帧上发送是不同的。

**接下来，我们来确定 NPDSCH 在频域上占用的资源。**

对于 UE 而言，如果一个下行子帧分配给 NPDSCH 传输，则该 NPDSCH 传输会占用该子帧的所有频域资源（即所有的 12 个子载波），也即 NPDSCH 传输不支持频分复用。因此针对一个 UE，只需要确定该 UE 的 NPDSCH 传输使用的是 anchor carrier，还是 non-anchor carrier，就能够知道 NPDSCH 在频域上的位置。具体可见 2.4 节关于“multi-carrier”的介绍。

#### 4.3.2.3 调制阶数、冗余版本和 TBS

**确定调制阶数：**所有的下行传输固定使用 QPSK 调制，即调制阶数  $Q_m = 2$ 。

**确定冗余版本：**NB-IoT 在下行只支持一种冗余版本，即 RV 0。因此也可以认为 NPDSCH 传输是不支持冗余版本的。

**除 SIB1-NB 外的 NPDSCH 传输的 TBS 计算方式如下：**

首先，读取 DCI 中的 4 比特 “Modulation and coding scheme” 字段得到  $I_{\text{MCS}}$ ，并设置  $I_{\text{TBS}} = I_{\text{MCS}}$ ；然后，读取 DCI 中的 3 比特 “Resource assignment” 字段得到  $I_{\text{SF}}$ ；最后根据二元组  $(I_{\text{TBS}}, I_{\text{SF}})$  查 36.213 的 Table 16.4.1.5.1-1 得到 TBS。如果使用带内部署（即 *operationModeInfo-r13* 的值指示‘00’或‘01’），有  $0 \leq I_{\text{TBS}} \leq 10$ 。

可以看出，下行支持的最大 TBS 为 680 比特。

**Table 16.4.1.5.1-1: Transport block size (TBS) table.**

$I_{\text{TBS}}$	$I_{\text{SF}}$							
	0	1	2	3	4	5	6	7
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	
6	88	176	256	392	504	600		
7	104	224	328	472	584	680		
8	120	256	392	536	680			
9	136	296	456	616				
10	144	328	504	680				
11	176	376	584					
12	208	440	680					

NB-IoT 的下行峰值速率是在使用最大 TBS 为 680 比特（此时一个 TB 需要使用 3 个子帧（3ms）来传输。一个 RB 共  $14 * 12 = 168$  个 RE，并且下行使用 QPSK 调制，一个 RB 在不考虑参考信号的情况下，最多能传输  $168 * 2 = 336$  比特的信息，因此 TBS 为 680 比特时，至少需要 3 个子帧来传输该 TB），并且重复次数为 1（最好的信道条件，不需要重复传输）时的速率，也即 L1 的下行峰值速率为 226.7 kbps。当将 DCI 占用的子帧、HARQ timing 关系等因素考虑在内后，实际得到的下行峰值速率必然小于 226.7 kbps。

**携带 SIB1-NB 的 NPDSCH 传输的 TBS 计算方式如下：**

首先，将  $I_{\text{TBS}}$  设置为 MIB-NB 中的 *schedulingInfoSIB1-r13* 的值。然后，通过  $I_{\text{TBS}}$  查 36.213 的 Table 16.4.1.5.2-1 得到 SIB1-NB 的 TBS。可以看出 SIB1-NB 的 TBS 只有 4 种可能取值{208, 328, 440, 680}。



**Table 16.4.1.5.2-1: Transport block size (TBS) table for NPDSCH carrying *SystemInformationBlockType1-NB***

$I_{\text{TBS}}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TBS	208	208	208	328	328	328	440	440	440	680	680	680	Reserved			

#### 4.3.2.4 物理层映射

NPDSCH 在映射到 RE 上时，是按先频域后时域的顺序来映射的，并通过使用子帧交织的方式来优化 UE 的接收。（见 36.211 的 10.2.3.4 节）

对于**不携带 BCCH 的 NPDSCH 传输**而言，调制符号映射完一个子帧后，相同的内容会在接下来的  $\min(M_{\text{rep}}^{\text{NPDSCH}}, 4) - 1$  个子帧里重复发送（也就是说，重复发送  $\min(M_{\text{rep}}^{\text{NPDSCH}}, 4)$  次）。然后，剩余的调制符号会继续映射到接下来的一个子帧里，并且相同的内容会在接下来的  $\min(M_{\text{rep}}^{\text{NPDSCH}}, 4) - 1$  个子帧里重复发送，依次类推。当所有的调制符号都映射完后，会重复之前的过程，直到  $M_{\text{rep}}^{\text{NPDSCH}} N_{\text{SF}}$  个子帧都发送完为止。（注：36.211 的 10.2.3.4 节中的  $M_{\text{rep}}^{\text{NPDSCH}}$  即为前面介绍的  $N_{\text{Rep}}$ ）

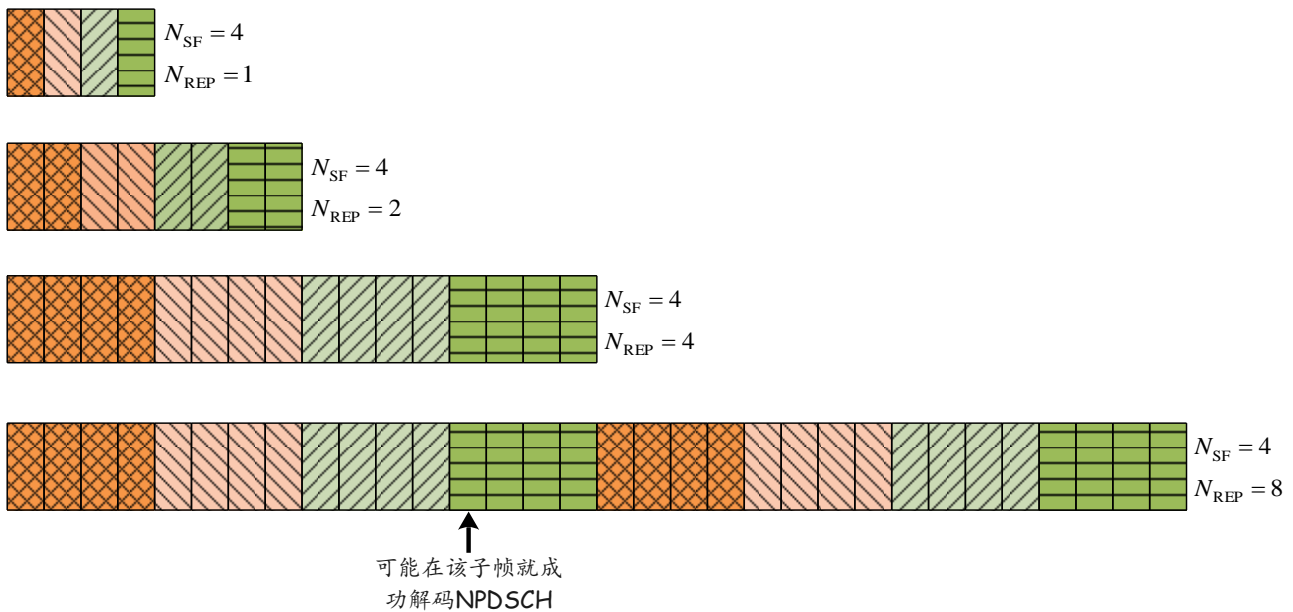


图 4-13: 不携带 BCCH 的 NPDSCH 在时域上的映射方式举例

图 4-13 是不携带 BCCH 的 NPDSCH 在时域上的映射方式的一个例子。假设一个 TB 需要映射到 4 个 NB-IoT 下行子帧上传输，即  $N_{\text{SF}}=4$ 。如果只重复 1 次（即  $N_{\text{Rep}}=1$ ），则该 TB 按顺序将调制后的符号映射到 4 个子帧上。如果重复 2 次（即  $N_{\text{Rep}}=2$ ），则第一个子帧映射完成后，会将该子帧重复发送一次（第二个子帧），然后再将剩余的符号映射到下一个（第三个）子帧，再重复发送一次（第四个子帧），依次类推。

如果重复 8 次（即  $N_{\text{Rep}}=8$ ），则第一个子帧映射完成后，相同的内容会在接下来的

$\min(M_{\text{rep}}^{\text{NPDSCH}}, 4)-1=3$  个子帧里重复发送，然后再将剩余的符号映射到下一个（第 5 个）子帧，再重复发送  $\min(M_{\text{rep}}^{\text{NPDSCH}}, 4)-1=3$  次，依次类推。从第 17 个子帧开始，又会重复前面的过程，直到  $M_{\text{rep}}^{\text{NPDSCH}} N_{\text{SF}}=32$  个子帧都发送完为止。

对携带 BCCH（SIB1-NB 或 SI 消息）的 NPDSCH 传输而言，是将一个 TB 映射到  $N_{\text{SF}}$  个子帧后，再重复前面的流程直到  $M_{\text{rep}}^{\text{NPDSCH}} N_{\text{SF}}$  个子帧都发送完为止。

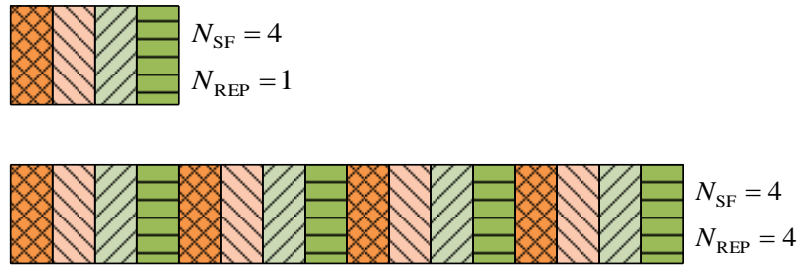


图 4-14：携带 BCCH 的 NPDSCH 在时域上的映射方式举例

图 4-14 是携带 BCCH 的 NPDSCH 在时域上的映射方式的一个例子。假设一个 TB 需要映射到 4 个 NB-IoT 下行子帧上传输，即  $N_{\text{SF}}=4$ 。如果只重复 1 次（即  $N_{\text{Rep}}=1$ ），则该 TB 按顺序将调制后的符号映射到 4 个子帧上。如果重复 4 次（即  $N_{\text{Rep}}=4$ ），则该 TB 按顺序将调制后的符号映射到 4 个子帧后，会重复之前的过程，直到  $M_{\text{rep}}^{\text{NPDSCH}} N_{\text{SF}}=16$  个子帧都发送完为止。

### 4.3.3 DL gap

如果一个 UE 的 NPDCCH/NPDSCH 传输需要大量的重复，那么该 UE 就会长时间霸占着时域资源，从而导致其它 UE 长时间得不到调度（包括上行资源调度和下行资源调度），并可能因此带来过大的延时。

为了提供足够的调度灵活性，以及避免那些需要大量重复/子帧的 NPDCCH/NPDSCH 传输的 UE 阻塞其它 UE 的 NPDCCH/NPDSCH 传输，引入了 DL gap 的概念。如果一个 UE 配置了 DL gap，那么该 UE 在 DL gap 期间将停止 NPDCCH/NPDSCH 传输，而其它没有配置相同 DL gap 的 UE 就可以在此 DL gap 期间进行 NPDCCH/NPDSCH 传输。当 DL gap 结束后，配置了该 DL gap 的 UE 可以继续之前没有完成的 NPDCCH/NPDSCH 传输。

也就是说，通过配置 DL gap，使得需要大量重复的 NPDCCH/NPDSCH 传输在时域资源上不连续映射（在中间插入了 DL gap），以便其它 UE 能够在 DL gap 期间发送其 NPDCCH/NPDSCH。

例如，位于极端覆盖下的 UE，其下行数据需要大量的重复传输才能保证其可靠性（从 4.3.2.2 节可以看出，一个 TB 至多占用 10 个子帧，并至多重复 2048 次，导致 TB 的一次传输至多需要占用约 20 秒的时间），通过在 NPDCCH 和 NPDSCH 的重复期间插入 DL gap，可以避免其它 UE 过大的延时。在 DL gap 期间，除了那些配置有该特定 DL gap 的 UE 之外的其它 UE 可以接收它们的 NPDCCH / NPDSCH。

对一个特定的 UE 而言，那些分配给 DL gap 的子帧是一个无效的下行子帧。当 NPDCCH/NPDSCH 的重复与 DL gap 重叠时，这些重复会推迟到下一个有效的下行子帧再发送。但对于携带 BCCH 的 NPDSCH 传输而言，其传输过程是不受 DL gap 的影响的。

eNodeB 可以通过 SIB2-NB 的 *dl-Gap-r13* 字段为 anchor carrier 配置一个 DL gap。或通过 UE 特定的 *dl-GapNonAnchor-r13* 字段为 non-anchor carrier 配置一个 DL gap。DL gap 的配置参数如下：

```
DL-GapConfig-NB-r13 ::= SEQUENCE {
    dl-GapThreshold-r13      ENUMERATED {n32, n64, n128, n256},      ----  $N_{\text{gap,threshold}}$ 
    dl-GapPeriodicity-r13    ENUMERATED {sf64, sf128, sf256, sf512}, ---- DL gap 的周期  $N_{\text{gap,period}}$ 
    dl-GapDurationCoeff-r13  ENUMERATED {oneEighth, oneFourth, threeEighth, oneHalf} ----  $N_{\text{gap,coeff}}$ 
}
```

如果  $R_{\max} < N_{\text{gap,threshold}}$ ，则在 NPDCCH/NPDSCH 传输过程中不存在 DL gap，其中  $N_{\text{gap,threshold}}$  通过 *dl-*

*GapThreshold-r13* 配置。DL gap 的起始系统帧和子帧满足条件  $(10n_f + \lfloor n_s/2 \rfloor) \bmod N_{\text{gap,period}} = 0$ ，其中 DL

gap 的周期  $N_{\text{gap,period}}$  通过 *dl-GapPeriodicity-r13* 配置。DL gap 的持续时间（以子帧数为单位）为

$N_{\text{gap,duration}} = N_{\text{gap,coeff}} N_{\text{gap,period}}$ ，其中  $N_{\text{gap,coeff}}$  通过 *dl-GapDurationCoeff-r13* 配置。（见 36.211 的 10.2.3.4 节）

## 4.4 下行功率分配

在下行，eNodeB 需要确定每个 RE 的下行发射功率，即确定 EPRE（Energy Per Resource Element）。

下行发射功率是参考于 NRS 的发射功率。eNodeB 会在广播消息中向 UE 指示 anchor carrier 的 NRS 发射功率值（EPRE），并通过一个 UE 特定的 RRC 消息来指示 non-anchor carrier 的 NRS EPRE 相对于 anchor carrier 的偏移，以便 UE 估计路径损耗。具体处理如下：

NRS 的 EPRE 值为  $\text{nrs-Power-r13} + \text{nrs-PowerOffsetNonAnchor-v1330}$ 。其中 *nrs-Power-r13* 是小区级的配置，通过 SIB2-NB 发送给 UE，该值指示了 anchor carrier 的 NRS EPRE；*nrs-PowerOffsetNonAnchor-v1330* 是 UE 级的配置，用于指示 non-anchor carrier 的 NRS EPRE 相对于 anchor carrier 的 NRS EPRE 的功率偏移（以 dB 为单位）。如果上层未提供 *nrs-PowerOffsetNonAnchor-v1330*，则其值为 0。可以看出，anchor carrier 和 non-anchor carrier 的 NRS EPRE 可以配置成不同的值。

NRS 发射功率定义为在 NB-IoT 系统带宽内携带 NRS 的所有 RE 的功率贡献（以[W]）的线性平均。除非重新配置一个不同的 NRS 功率值，否则 NRS 的 EPRE 在所有子帧的整个下行 NB-IoT 系统带宽上都保持在一个恒定的值。

其它下行物理信道的发射功率（EPRE）通过其与 NRS EPRE 的比值来定义，其值与使用的传输样式相关。如果使用单天线端口传输，则 NPDCCH/NPDSCH/NPBCH 的 EPRE（不包含那些 0 EPRE 的 RE）与 NRS EPRE 的比值为 0 dB，即与 NRS 的发射功率相同；如果使用 2 天线端口传输，则 NPDCCH/NPDSCH/NPBCH 的 EPRE（不包含那些 0 EPRE 的 RE）与 NRS EPRE 的比值为-3 dB。

如果使用带内部署且 NB-IoT 小区与 LTE 小区使用相同的 PCI（即 *operationModeInfo-r13* 的值对应为‘00’），那么 eNodeB 可以通过 SIB1-NB 的 *nrs-CRS-PowerOffset-r13* 来给 UE 设置 NRS 与 LTE 的 CRS 之间的功率偏移，也即 NRS EPRE 与 CRS EPRE 的比值（如果不提供该字段，则 NRS EPRE 与 CRS EPRE 的比值为 0dB）。其目的是使得在这种场景下，UE 也可以使用 CRS 来进行信道估计。当前协议定义可设置 NRS 比 CRS 最大高 9dB，而实际 power boosting 的大小需根据设备的发射能力而定。

关于下行功率分配，具体可参见 36.213 的 16.2.2 节的介绍。

#### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] <http://www.sharetechnote.com/> 关于 IoT 的介绍
- [3] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [4] 3GPP TS 36.211 V13.4.0, 2016-12; Physical channels and modulation (Release 13)
- [5] 3GPP TS 36.212 V13.4.0, 2016-12; Multiplexing and channel coding (Release 13)
- [6] 3GPP TS 36.213 V13.4.0, 2016-12; Physical layer procedures(Release 13)
- [7] 3GPP TS 36.321 V13.4.0, 2016-12; Medium Access Control (MAC) protocol specification
- [8] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification
- [9] “R1-161933, NB-PDCCH design”, Qualcomm Incorporated

## 第5章 上行物理层处理

为了支持 NB-IoT，上行定义了 2 种物理信道：

- NPUSCH (Narrowband Physical Uplink Shared CHannel) ；
- NPRACH (Narrowband Physical Random Access CHannel) ：这部分内容会在 7.3 节介绍。

为了支持 NB-IoT，上行定义了 1 种物理信号：

- NDMRS (Narrowband DeModulation Reference Signal) 。

NB-IoT 在上行只使用单天线端口 ( $p=0$ ) 传输。

### 5.1 上行时频资源

#### 5.1.1 时频资源

NB-IoT 在上行使用 SC-FDMA，并支持 multi-tone 和 single-tone 传输：

- **multi-tone** 传输使用 15kHz 子载波间距（即  $\Delta f = 15\text{ kHz}$ ），slot 长为 0.5ms。
- **single-tone** 传输支持 15kHz（即  $\Delta f = 15\text{ kHz}$ ）和 3.75kHz（即  $\Delta f = 3.75\text{ kHz}$ ）这 2 种子载波间距。其中 15kHz 子载波等同于 LTE 中的 15kHz 子载波，slot 长为 0.5ms，并与 LTE 的上行提供一致的性能。3.75kHz 子载波的符号长度是 15kHz 子载波的符号长度的 4 倍，其一个 slot 长为 2ms，此时一个系统帧（10ms）包含 5 个 slot。

single-tone 对应频域上只使用一个子载波来传输，multi-tone 对应频域上使用多个子载波来传输。在第 9 章关于“UE 能力”的介绍中，会阐述 NB-IoT UE 如何告诉 eNodeB 其是否具备支持 multi-tone 传输的能力。

对于 3.75kHz 子载波间距而言，一个 SC-FDMA 符号的长度为  $8448 \cdot T_s$ ，其中 CP 长度为  $256 \cdot T_s$ ，可用的符号时间为  $8192 \cdot T_s$ 。并且一个 slot 的最后会预留  $2304 \cdot T_s$  作为保护间隔。slot 之间预留的保护间隔是为了最小化 NB-IoT 符号与 LTE 中的 SRS 之间的冲突。（见 36.211 的 10.1.5 节）

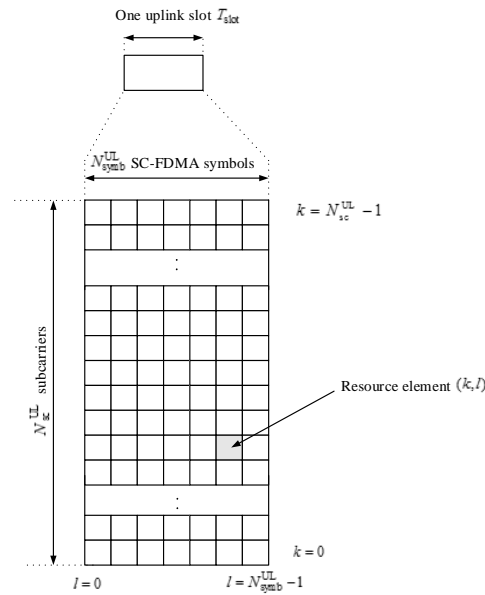
3.75kHz 子载波间距的 slot 边界与 FDD 中的子帧边界是对齐的。上行无论使用哪种子载波间距，一个 slot 都只包含 7 个 SC-FDMA 符号。

表 5-1: NB-IoT 的上行时频资源

x-tone	子载波间距	时域上每个 slot 的长度 ( $T_{\text{slot}}$ )	频域上的子载波数 ( $N_{\text{sc}}^{\text{UL}}$ )	一个 slot 内包含的符号数 ( $N_{\text{sym}}^{\text{UL}}$ )
single-tone	3.75kHz	2ms ( $61440 \cdot T_s$ )	48	7
	15kHz	0.5ms ( $15360 \cdot T_s$ )	12	
multi-tone	15kHz	0.5ms ( $15360 \cdot T_s$ )	12	

与 LTE 不同，NB-IoT 在上行并未定义 RB（Resource Block）的概念。在频域上，上行使用的子载波间距为 15kHz 时，频域上包含 12 个连续的子载波；上行使用的子载波间距为 3.75kHz 时，频域上包含 48 个连续的子载波。

一个 slot 上传输的上行物理信道或信号可以描述成一个或多个频域上包含  $N_{sc}^{UL}$  个子载波，时域上包含  $N_{symb}^{UL}$  个 SC-FDMA 符号的时频资源网格（resource grid）。36.211 的 Figure 10.1.2.1-1 对应上行的时频资源网格。对于 15kHz 子载波间距而言，一个系统帧内的 slot 编号  $n_s \in \{0,1,...19\}$ ；对于 3.75kHz 子载波间距而言，一个系统帧内的 slot 编号  $n_s \in \{0,1,...4\}$ 。



**Figure 10.1.2.1-1: Uplink resource grid for NB-IoT**

时频资源网格上的每个元素称为一个 RE（Resource Element），并且通过  $(k,l)$  唯一指定的，其中  $k = 0, ..., N_{sc}^{UL} - 1$ （对应频域上的每个子载波的索引）且  $l = 0, ..., N_{symb}^{UL} - 1$ （对应 slot 上的每个 SC-FDMA 符号的索引）。索引为  $(k,l)$  的 RE 对应的值使用  $a_{k,l}$  表示，一个 slot 内没有用于传输物理信道或信号的 RE 上携带的  $a_{k,l}$  会被设置为 0。

RE 是 NB-IoT 中的最小物理资源单位。一个 RE 可存放一个调制符号（modulation symbol），该调制符号可使用 BPSK（对应一个 RE 存放 1 比特数据）或 QPSK（对应一个 RE 存放 2 比特数据）调制。

### 5.1.2 RU (Resource Unit)

上行调度以及 HARQ 信息发送的基本单位是 RU (Resource Unit)，RU 用于描述如何将 NPUSCH 映射到 RE 上。一个 RU 在时域上由  $N_{\text{symb}}^{\text{UL}} N_{\text{slots}}^{\text{UL}}$  个连续的 SC-FDMA 符号组成，在频域上由  $N_{\text{sc}}^{\text{RU}}$  个连续的子载波组成。其中  $N_{\text{sc}}^{\text{RU}}$  和  $N_{\text{symb}}^{\text{UL}}$  的定义如 36.211 的 Table 10.1.2.3-1 所示。

**Table 10.1.2.3-1: Supported combinations of  $N_{\text{sc}}^{\text{RU}}$ ,  $N_{\text{slots}}^{\text{UL}}$ , and  $N_{\text{symb}}^{\text{UL}}$ .**

NPUSCH format	$\Delta f$	$N_{\text{sc}}^{\text{RU}}$	$N_{\text{slots}}^{\text{UL}}$	$N_{\text{symb}}^{\text{UL}}$
1	3.75 kHz	1	16	7
	15 kHz	1	16	
		3	8	
		6	4	
		12	2	
2	3.75 kHz	1	4	7
	15 kHz	1	4	

当  $\Delta f = 3.75 \text{ kHz}$  时，一个 slot 长为 2ms；当  $\Delta f = 15 \text{ kHz}$  时，一个 slot 长为 0.5ms。结合 36.211 的 Table 10.1.2.3-1，就能够知道一个 RU 在时域上的长度 ( $T_{\text{slot}} \cdot N_{\text{slots}}^{\text{UL}}$ )：

- 对于携带 UL-SCH 传输（对应 NPUSCH format 1）的 single-tone NPUSCH 而言，如果使用一个单一的 3.75kHz 子载波，则其 RU 在时域上的跨度为 32ms；如果使用一个单一的 15kHz 子载波，则其 RU 在时域上的跨度为 8ms。
- 对于携带 UL-SCH 传输（对应 NPUSCH format 1）的 multi-tone NPUSCH 而言，使用 3 个子载波时，其 RU 在时域上的跨度为 4ms；使用 6 个子载波时，其 RU 在时域上的跨度为 2ms；使用 12 个子载波时，其 RU 在时域上的跨度为 1ms。
- 对于携带 ACK/NACK（对应 NPUSCH format 2）的 NPUSCH 而言，如果使用一个单一的 3.75kHz 子载波，则其 RU 在时域上的跨度为 8ms，如果使用一个单一的 15kHz 子载波，则其 RU 在时域上的跨度为 2ms。

表 5-2：一个上行 RU 占用的时频资源

x-tone	RU 携带的内容	子载波间距 (子载波数)	RU 在时域上的跨度 (一个 RU 持续的时间 以及占用的 slot 数)	RU 在频域上的跨度 (一个 RU 占用的子载 波数)
single-tone	UL-SCH (NPUSCH format 1)	3.75kHz	32ms (16 slots)	1
		15kHz	8ms (16 slots)	1
multi-tone	UL-SCH (NPUSCH format 1)	15kHz (3 个子载波)	4ms (8 slots)	3
		15kHz (6 个子载波)	2ms (4 slots)	6



		15kHz (12 个子载波)	1ms (2 slots)	12
single-tone	ACK/NACK (NPUSCH format 2)	3.75kHz	8ms (4 slots)	1
		15kHz	2ms (4 slots)	1

可以看出，只使用 single-tone 的 NPUSCH format 2 传输，针对不同的子载波间距，一个 RU 内的有效 RE 数是一样的。使用 single-tone 的 NPUSCH format 1 传输，针对不同的子载波间距，一个 RU 内的有效 RE 数是一样的。使用 multi-tone 的 NPUSCH format 1 传输，针对不同的频域子载波数，一个 RU 内的有效 RE 数是一样的。并且 RU 在时域上的长度均为 2 的整数幂，这样做是为了在调度不同子载波数的 RU 时，降低资源的碎片率。

### 5.1.3 single-tone vs multi-tone 和 3.75kHz vs 15kHz

single-tone 对应频域上只使用一个子载波，multi-tone 对应频域上使用多个子载波。

single-tone 的峰均比小，终端功耗小，能够提供更好的覆盖、系统容量和终端功耗；multi-tone 则支持更大的峰值速率。

对于覆盖增强场景，3.75kHz 子载波间距比 15kHz 子载波间距提供了更大的上行系统容量。但是在带内部署场景下，15kHz 子载波间距比 3.75kHz 子载波间距有更好的 LTE 兼容性。

使用 3.75kHz 子载波间距时，相当于把频域上的 180kHz 切成 48 块（48 个子载波），相比于 15kHz 子载波间距的 12 块，自然提供了更大的容量来传输更多 UE 的上行传输（NPUSCH 和 NPRACH）。

single-tone 中灵活地使用 3.75kHz 和 15kHz 是为了适应不同的场景：

- 3.75kHz：由于使用更长的 CP 长度，其定时要求更宽裕，并提供了更好的覆盖；为小区提供了更大的 NPRACH 和 NPUSCH 容量；功耗更低。
- 15kHz：更好地向前兼容 LTE；更好地与原有 LTE 网络共存。

传统的 LTE UE 仅支持 12 子载波 multi-tone 传输，为 NB-IoT UE 引入 3 子载波或 6 子载波 multi-tone 传输的原因在于覆盖的限制会导致 NB-IoT UE 不能从更高的带宽分配（12 子载波）中获益。

从市场角度上看，single-tone 传输的功耗低、成本低，使其能够更好地与非蜂窝设备竞争，并加快上市时间；包括蜂窝行业和低成本芯片供应商都对构建 single-tone 芯片感兴趣。multi-tone 的功耗和成本比 single-tone 高，但能够提供更高的数据速率，它可以用于取代大多数原有的 CDMA / GPRS 物联网应用。

## 5.2 窄带解调参考信号（NDMRS）

NDMRS（Narrowband DeModulation Reference Signal，窄带解调参考信号）主要用于上行物理信道的信道估计，以便正确地解码 NPUSCH。上行定义的窄带解调参考信号与数据复用在一起传输的，因此只有包含了数据传输的 RU 上才会发送 NDMRS。NB-IoT 在上行不支持多天线传输，所有的上行传输都使用单天线端口。



当子载波个数为 1 时，NDMRS 的参考信号序列的生成方式如图 5-1 所示。当子载波个数大于 1 时，NDMRS 的参考信号序列的生成方式如图 5-2 所示。

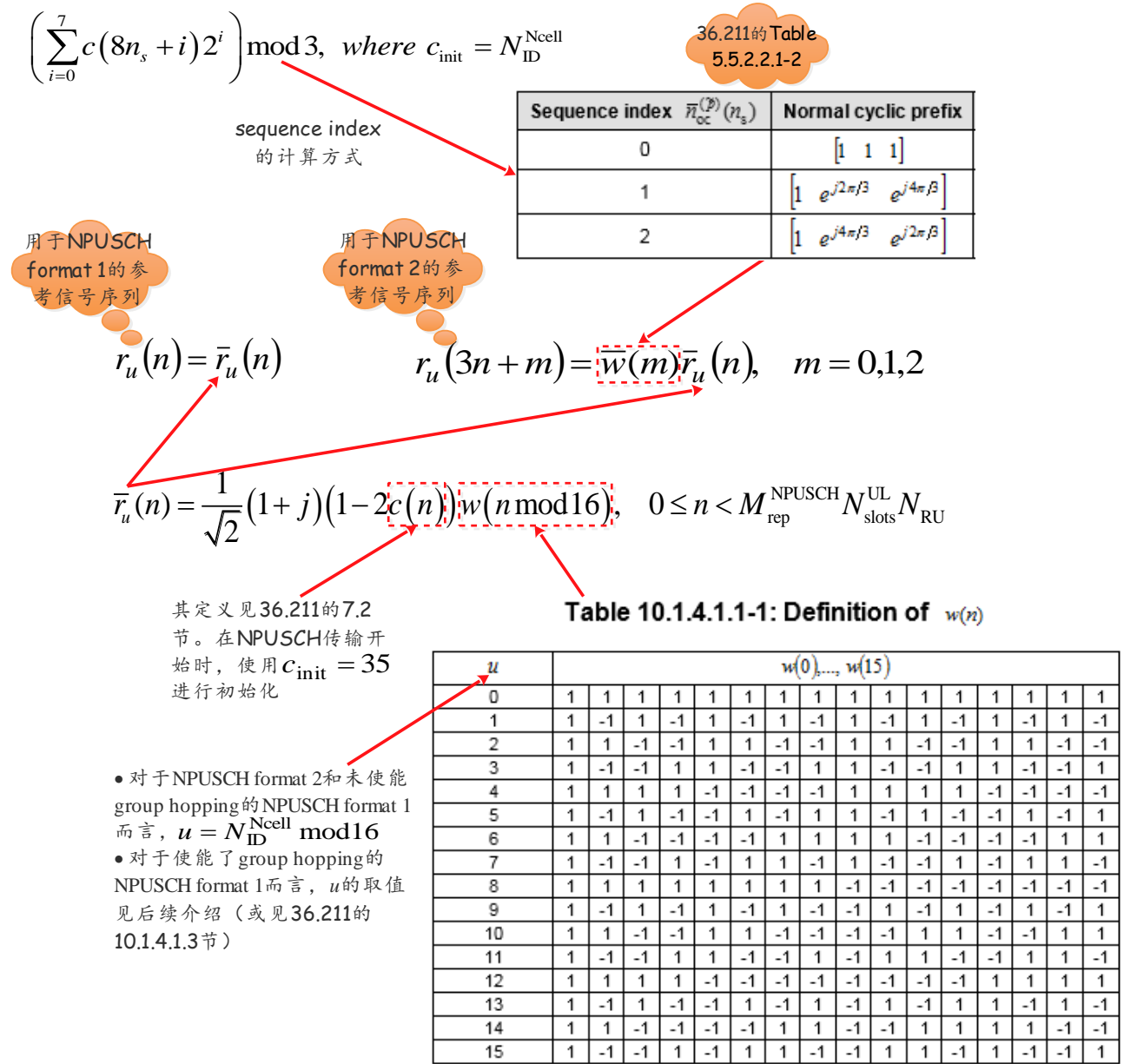


图 5-1:  $N_{\text{sc}}^{\text{RU}} = 1$  时的 NDMRS 参考信号序列生成方式

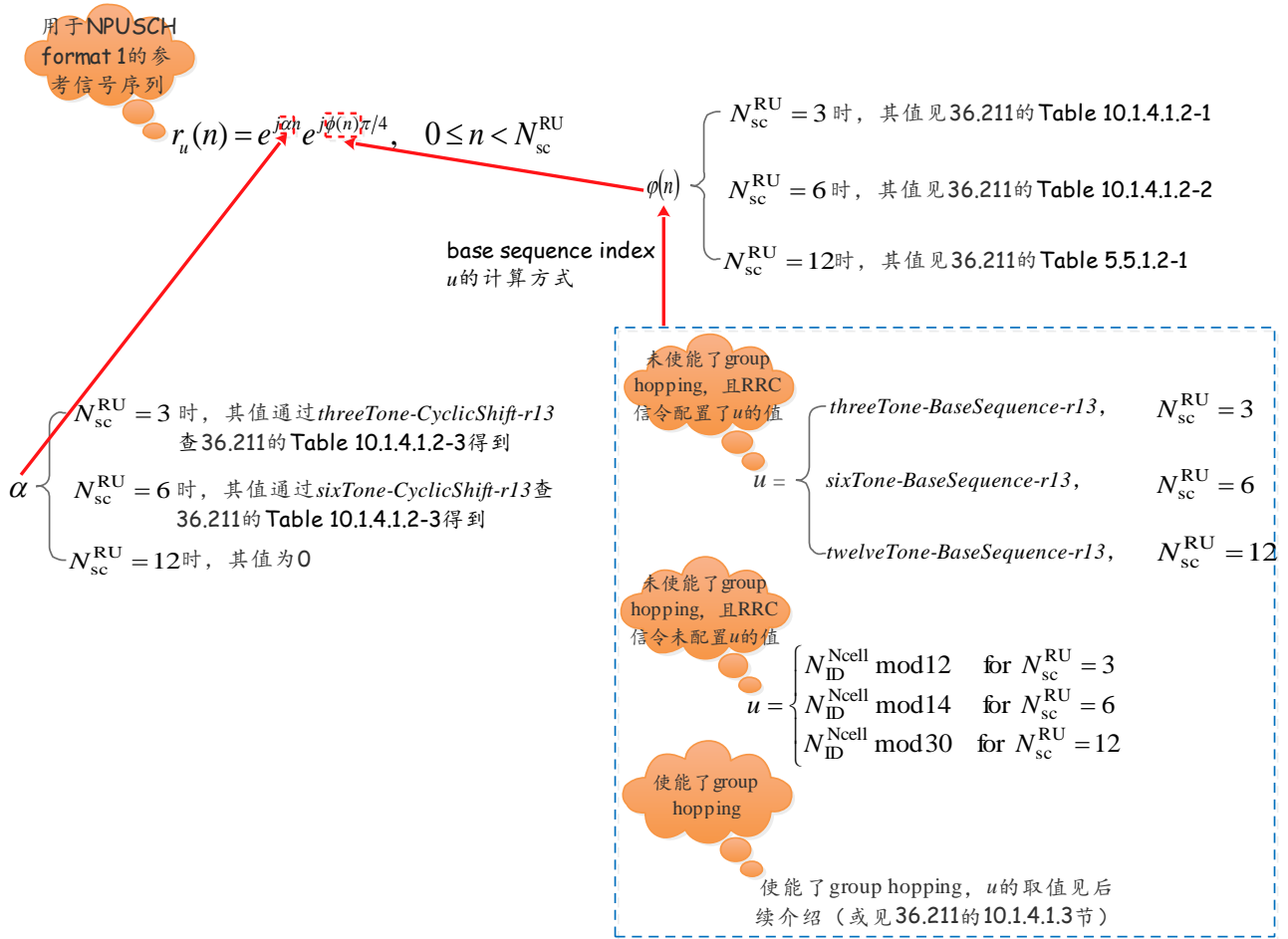


图 5-2:  $N_{sc}^{RU} > 1$  时的 NDMRS 参考信号序列生成方式

NDMRS 符号与相关的数据使用相同的调制。对于 NPUSCH format 2 而言, NDMRS 符号使用与 LTE 中的 PUCCH format 1/1a/1b 相同的正交序列。

不同的 RU 大小 (频域上的子载波数不同) 拥有不同的基序列数, 对应关系见 36.211 的 Table 10.1.4.1.3-1。

与 LTE 类似, 为了随机化小区间 (inter-cell) 的参考信号干扰, NB-IoT 为 NPUSCH format 1 引入了 group hopping (组跳) 的概念。同一小区内, 不同的 slot 可以使用相同的 group, 也可以使用不同的 group。如果不同的 slot 使用不同的 group (即使用不同的  $u$  值), 就形成组跳。而不同 NB-IoT 小区的  $u$  的取值还与该小区的 PCI ( $N_{ID}^{Ncell}$ ) 相关, 进一步地随机化了小区间的干扰。

NB-IoT 中的 NDMRS 是否使能组跳可以通过小区特定的配置参数 *groupHoppingEnabled-r13* 来指定。与此同时, 还可以使用一个 UE 特定的配置参数 *groupHoppingDisabled-r13* 来配置是否去使能某个特定 UE 的组跳, 此时不需要关心是否在小区级的配置中使能了组跳。即对于某个 UE 来说, 该 UE 级的配置会覆盖小区级的配置。但基于竞争的随机接入过程中的 Msg3 的初传和重传, 是不支持 UE 级的组跳去使能的。

如果未使能组跳， $u$  的取值是一个固定的值，见图 5-1 和图 5-2。如果使能了组跳， $u$  的计算方式见图 5-3。

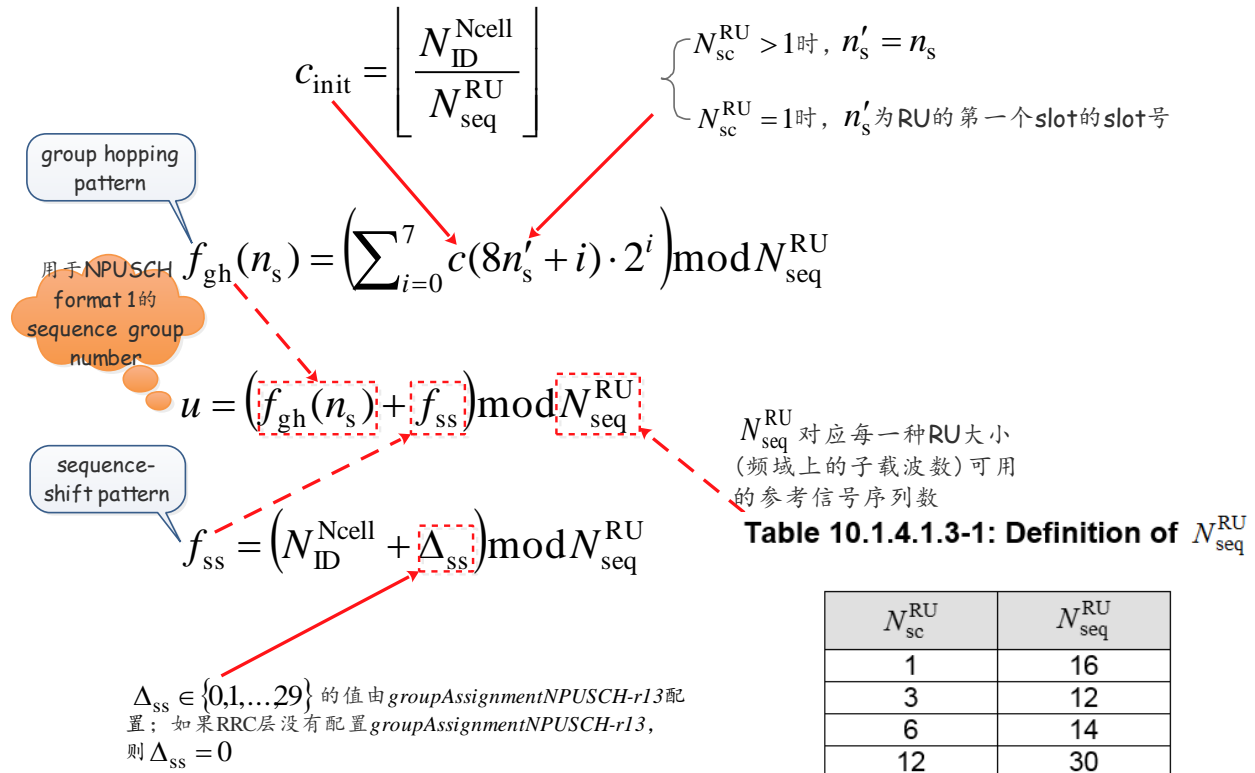


图 5-3: slot  $n_s$  上的 group number  $u$  的计算方式

取决于 NPUSCH 格式和使用的子载波间距，NDMRS 在每个 slot 的 1 个或 3 个 SC-FDMA 符号上传输。

- 对于 NPUSCH format 1 而言，如果使用子载波间距 3.75kHz，则 NDMRS 在每个 slot 的第 5 个 SC-FDMA 符号 ( $l=4$ ) 上传输；如果使用子载波间距 15kHz，则 NDMRS 在每个 slot 的第 4 个 SC-FDMA 符号 ( $l=3$ ) 上传输。如图 5-4 所示。

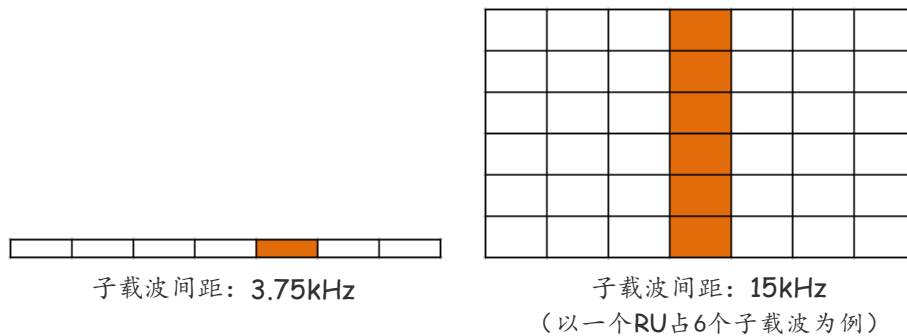


图 5-4: NPUSCH format 1 的 NDMRS 所占的 RE

- 对于 NPUSCH format 2 而言，如果使用子载波间距 3.75kHz，则 NDMRS 在每个 slot 的前 3 个 SC-FDMA 符号 ( $l=0, 1, 2$ ) 上传输；如果使用子载波间距 15kHz，则 NDMRS 在每个 slot 的中间 3 个 SC-FDMA 符号 ( $l=2, 3, 4$ ) 上传输。如图 5-5 所示。

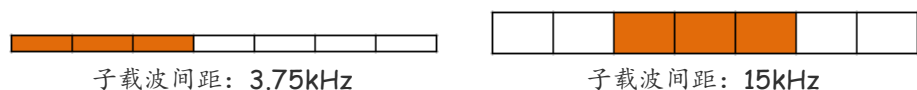


图 5-5: NPUSCH format 2 的 NDMRS 所占的 RE

### 5.3 上行物理信道

为了支持 NB-IoT，上行定义了 2 种物理信道：

- NPUSCH (Narrowband Physical Uplink Shared CHannel)：用于传输上行单播数据 (UL-SCH) 以及上行控制信息。
- NPRACH (Narrowband Physical Random Access CHannel)：用于传输随机接入过程中的 preamble。NPRACH 将在 7.3.2 节介绍。

NPUSCH 支持 2 种格式：

- NPUSCH format 1：用于传输 UL-SCH 数据，其 TBS 不会超过 1000 比特；
- NPUSCH format 2：用于传输上行控制信息 (UCI)。在 Rel-13 中，上行控制信息只有针对下行 NPDSCH 传输的确认信息，即 ACK/NACK。

NB-IoT 中，上行传输信道与上行物理信道的映射关系如图 5-6 所示。

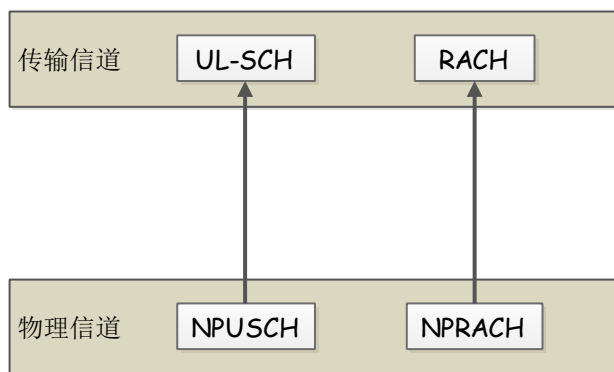


图 5-6: NB-IoT 中，上行传输信道和上行物理信道的映射关系

与 LTE 相比，NB-IoT 在上行取消了类似于 PUCCH 的物理信道。NB-IoT UE 在上行不支持 SR 和 CSI 上报，其上行 UCI 信息只包括针对下行 NPDSCH 传输的 ACK/NACK 信息，并且该 ACK/NACK 信息在 NPUSCH format 2 上反馈。

可以看出，除了 RACH 传输，包括上行控制信息 (UCI) 在内的所有上行数据都是通过 NPUSCH 来传输的。并且 UL-SCH 数据和上行控制信息使用不同的 NPUSCH 格式。

为了更好地支持上行覆盖增强，NB-IoT 系统在上行物理信道上也引入了重复传输机制。

表 5-3 简单地比较了 NB-IoT 上行物理信道与 LTE 中对应的物理信道的差异。

表 5-3: NB-IoT 上行物理信道与 LTE 中对应的物理信道的比较

NB-IoT	物理信道/信号	说明	与 LTE 的比较
上行	NPARCH	<ul style="list-style-type: none"> <li>基于单子载波跳频且使用 3.75kHz 子载波间距的新 preamble 结构</li> </ul>	PRACH 在频域上占 6 个 PRB, 使用 multi-tone 传输且使用 1.25kHz 子载波间距
	NPUSCH format 1	<ul style="list-style-type: none"> <li>分配给 UE 的带宽可以小于一个 PRB;</li> <li>支持 2 种子载波间距: 3.75kHz 和 15kHz;</li> <li>single-tone 传输使用 <math>\pi/2</math> BPSK 或 <math>\pi/4</math> QPSK 调制; multi-tone 传输使用 QPSK 调制;</li> <li>最大 TBS 为 1000 比特;</li> <li>只支持单层传输 (单天线端口)</li> </ul>	<b>PUSCH:</b> <ul style="list-style-type: none"> <li>分配给 UE 的最小带宽为一个 PRB;</li> <li>只支持 15kHz 子载波间距;</li> <li>使用标准的 QPSK 调制或更高阶的 16QAM/64QAM 调制;</li> <li>在非空分复用下, 最大 TBS 可达到 75376 比特;</li> <li>可支持多层传输 (支持多天线传输)</li> </ul>
	NPUSCH format 2	<ul style="list-style-type: none"> <li>使用一种新的编码: 重复码;</li> <li>只使用 single-tone 传输</li> </ul>	不存在与 NPUSCH format 2 对应的物理信道
注: 与 LTE 相比, NB-IoT 在上行不存在类似于 PUCCH 的物理信道。			

### 5.3.1 NPUSCH format 1

UL-SCH 传输 (即 NPUSCH format 1, 这里不考虑 RAR 的情况) 对应的 UL grant 在 DCI format N0 中指示。DCI format N0 中包含了指示 NPUSCH 传输的起始时间、重复次数、传输一个 TB 所需的 RU 数以及频域上使用的子载波数信息等。MCS 信息也在 DCI format N0 中发送, 该 MCS 提供了单子载波 RU 使用的调制方式, 并与 RU 数一起, 确定一个 TB 的 TBS。

NPUSCH format 1 使用的信道编码为 Turbo coding, 且码率为 1/3。 (见 36.212 的 Table 6.2-1)

如果 NPUSCH format 1 使用 single-tone 传输, 则一个 RU 由频域上的一个子载波和时域上的 16 个 slot 组成; 如果 NPUSCH format 1 使用 multi-tone 传输, 则一个 RU 在频域上可能包含 3、6 和 12 个子载波, 如表 5-4 所示。

NPUSCH format 1 使用的调制方式取决于所选的 RU (见 36.211 的 10.1.3.2 节):

- 如果一个 RU 占用一个子载波, 则可以使用  $\pi/2$  BPSK 或  $\pi/4$  QPSK 调制来最小化 PAPR (Peak-to-Average Power Ratio);
- 如果一个 RU 占用多于一个子载波, 则固定使用 QPSK 调制。

表 5-4: NPUSCH format 1 占用的一个 RU 资源

NPUSCH format	携带的内容	x-tone	子载波间距 (子载波数)	一个 RU 持续的时间 (占用的 slot 数 $N_{\text{slots}}^{\text{UL}}$ )	一个 RU 占用的 子载波数 $N_{\text{sc}}^{\text{RU}}$
1	UL-SCH	single-tone	3.75kHz	32ms (16 slots)	1
			15kHz	8ms (16 slots)	1
		multi-tone	15kHz (3 个子载波)	4ms (8 slots)	3
			15kHz (6 个子载波)	2ms (4 slots)	6
			15kHz (12 个子载波)	1ms (2 slots)	12

### 5.3.1.1 时频资源分配

首先，我们来介绍 NPUSCH format 1 在时域上的 timing 关系和资源分配。

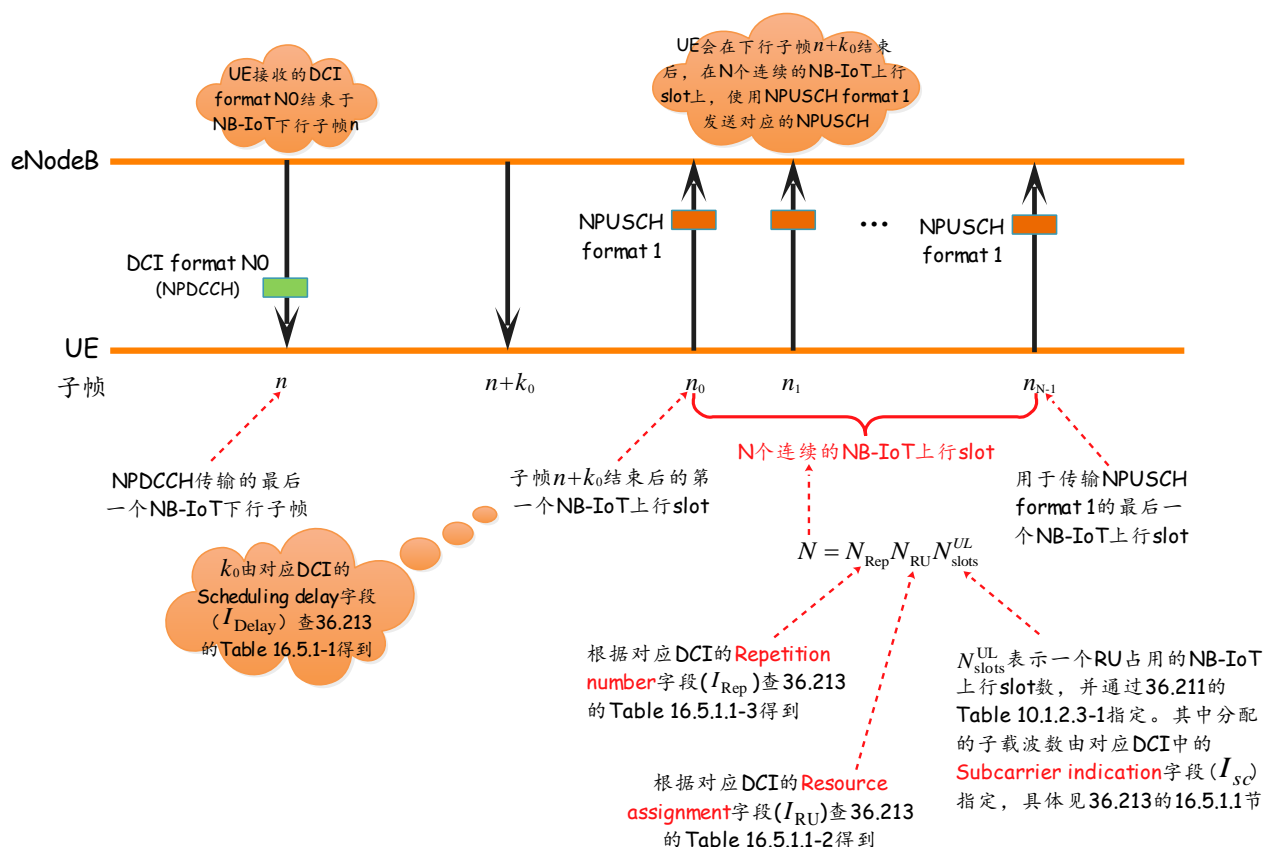


图 5-7: NPUSCH format 1 传输在时域上的资源分配

如果 UE 接收到的 DCI format N0 结束于 NB-IoT 下行子帧  $n$ ，那么 UE 会在下行子帧  $n+k_0$  结束之后，根据 DCI 中携带的信息，在  $N$  个连续的 NB-IoT 上行 Slot  $n_i$ （其中  $i=0, 1, \dots, N-1$ ）上，使用 NPUSCH format 1 发送对应的 NPUSCH。（见 36.213 的 16.5.1）

- 子帧  $n$  是 NPDCCH (DCI format N0) 传输的最后一个子帧, 并由 NPDCCH 传输的起始子帧和对应 DCI 里的 “DCI subframe repetition number” 字段共同决定。具体的计算方式见 4.3.1 节的介绍。
- $N = N_{\text{Rep}} N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$ 。其中  $N_{\text{Rep}}$  表示 NPUSCH (或者说 TB) 的重复次数, 并根据对应 DCI 的 Repetition number 字段 ( $I_{\text{Rep}}$ ) 查 36.213 的 Table 16.5.1.1-3 得到。  $N_{\text{RU}}$  表示一个 TB 需要占用的 RU 个数, 并根据对应 DCI 的 Resource assignment 字段 ( $I_{\text{RU}}$ ) 查 36.213 的 Table 16.5.1.1-2 得到。  $N_{\text{slots}}^{\text{UL}}$  表示一个 RU 占用的 NB-IoT 上行 slot 数, 并通过 36.211 的 Table 10.1.2.3-1 指定。其中分配的子载波数由对应 DCI 中的 Subcarrier indication 字段 ( $I_{\text{sc}}$ ) 指定, 具体见接下来的介绍, 或见 36.213 的 16.5.1.1 节。
- $n_0$  是子帧  $n+k_0$  结束之后的第一个 NB-IoT 上行 slot;
- $k_0$  由对应 DCI 的 Scheduling delay 字段 ( $I_{\text{Delay}}$ ) 查 36.213 的 Table 16.5.1-1 得到。

**Table 16.5.1-1:  $k_0$  for DCI format N0.**

$I_{\text{Delay}}$	$k_0$
0	8
1	16
2	32
3	64

从前面的介绍可以看出, NPUSCH format 1 传输的起始时间与其相关的 DCI format N0 的结束时间之间的间隔大于等于 8ms。

一个 UL-SCH 传输块 (TB) 在时域上可以在一个或多于一个 RU 上被调度。也就是说, 一个单一的 TB 可能需要使用多个 RU 来传输, 这也是  $N_{\text{RU}}$  的物理意义, 即一个 TB 被调度在  $N_{\text{RU}}$  个 RU 上传输。这与 LTE 中, 一个 TB 通常只在一个子帧上发送是不同的。

**接下来, 我们来确定 NPUSCH format 1 在频域上占用的子载波数以及占用了哪些子载波。**

第一步, 我们要确定 NPUSCH 传输使用的子载波间距  $\Delta f$ 。  $\Delta f$  是由 Narrowband Random Access Response Grant 中的 Uplink subcarrier spacing 字段指定的。(见 36.213 的 16.3.3 节)

第二步, 读取对应 DCI 的 subcarrier indication 字段, 得到  $I_{\text{sc}}$ 。

如果 NPUSCH format 1 的子载波间距  $\Delta f = 3.75 \text{ kHz}$ , 则此时频域上使用 1 个子载波, 且分配给该 NPUSCH 传输的子载波索引  $n_{\text{sc}} = I_{\text{sc}}$ 。其中  $I_{\text{sc}}$  由对应 DCI 的 Subcarrier indication 字段指定。  $\Delta f = 3.75 \text{ kHz}$  时, 一个 RU 在频域上占 48 个子载波, 因此  $I_{\text{sc}} = 48, 49, \dots, 63$  是预留的。

如果 NPUSCH format 1 的子载波间距  $\Delta f = 15 \text{ kHz}$ ，则根据对应 DCI 的 subcarrier indication 字段 ( $I_{sc}$ ) 查 36.213 的 Table 16.5.1.1-1 得到分配给该 NPUSCH 传输的连续子载波集合  $n_{sc}$ ，该集合中元素的个数就是使用的子载波数  $N_{sc}^{RU}$ 。

**Table 16.5.1.1-1: Allocated subcarriers for NPUSCH with  $\Delta f = 15 \text{ kHz}$ .**

Subcarrier indication field ( $I_{sc}$ )	Set of Allocated subcarriers ( $n_{sc}$ )
0 – 11	$I_{sc}$
12-15	$3(I_{sc} - 12) + \{0,1,2\}$
16-17	$6(I_{sc} - 16) + \{0,1,2,3,4,5\}$
18	$\{0,1,2,3,4,5,6,7,8,9,10,11\}$
19-63	Reserved

举个例子，假设对应 DCI 的 Subcarrier indication 字段  $I_{sc}=10$ ，查 36.213 的 Table 16.5.1.1-1 可知，此时分配给 NPUSCH 传输的子载波个数为 1，且该子载波的索引为 10；假设对应 DCI 的 Subcarrier indication 字段  $I_{sc}=13$ ，查 36.213 的 Table 16.5.1.1-1 可知，此时分配给 NPUSCH 传输的子载波个数为 3，且子载波的索引集合  $n_{sc}$  为  $\{3,4,5\}$ ；假设对应 DCI 的 Subcarrier indication 字段  $I_{sc}=18$ ，查 36.213 的 Table 16.5.1.1-1 可知，此时分配给 NPUSCH 传输的子载波个数为 12，且子载波的索引集合  $n_{sc}$  为  $\{0,1,2,3,4,5,6,7,8,9,10,11\}$ 。

至此，UE 就知道在哪些时频资源上去接收 NPUSCH format 1 了。

### 5.3.1.2 调制阶数、冗余版本和 TBS

为了确定 NPUSCH format 1 使用的调制阶数、冗余版本和 TBS，UE 首先要获取下面的信息：（见 36.213 的 16.5.1.2 节）

- 读取对应 DCI 中的“modulation and coding scheme”字段，得到  $I_{MCS}$ ；
- 读取对应 DCI 中的“redundancy version”字段，得到  $rv_{DCI}$ ；
- 读取对应 DCI 中的“resource assignment”字段，得到  $I_{RU}$ ；
- 计算分配的子载波个数  $N_{sc}^{RU}$ 、RU 的个数  $N_{RU}$  以及重复次数  $N_{Rep}$ ：计算方式见上一节的介绍。

**确定调制阶数：**如果一个 RU 在频域上占用多于一个子载波，即  $N_{sc}^{RU} > 1$ ，则固定使用 QPSK 调制，即调制阶数  $Q_m = 2$ ；如果一个 RU 在频域上占用一个子载波，即  $N_{sc}^{RU} = 1$ ，则 UE 会使用  $I_{MCS}$  查 36.213 的 Table 16.5.1.2-1 得到使用的调制阶数。



**Table 16.5.1.2-1: Modulation and TBS index table for NPUSCH with  $N_{\text{sc}}^{\text{RU}} = 1$ .**

MCS Index $I_{\text{MCS}}$	Modulation Order $Q_m$	TBS Index $I_{\text{TBS}}$
0	1	0
1	1	2
2	2	1
3	2	3
4	2	4
5	2	5
6	2	6
7	2	7
8	2	8
9	2	9
10	2	10

**确定冗余版本 RV:** NPUSCH 在  $N$  ( $N = N_{\text{Rep}} N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$ ) 个连续的 NB-IoT 上行 slot 传输, 其中一个上行

TB 需要占用  $N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$  个上行 slot。这里把  $N_{\text{Rep}}$  个  $N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$  分成  $\frac{N_{\text{Rep}}}{L}$  份, 每份的索引为

$j = 0, 1, \dots, \frac{N_{\text{Rep}}}{L} - 1$ , 且每份包含  $B = L N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$  个连续的 NB-IoT 上行 slot (如果  $N_{\text{sc}}^{\text{RU}} = 1$ , 则  $L = 1$ ; 否

则  $L = \min(4, \lceil N_{\text{Rep}} / 2 \rceil)$ )。第  $j$  份里包含的  $B$  个连续的 NB-IoT 上行 slot 的索引为  $n_i$ , 且有

$i = jB + b$ ,  $b = 0, 1, \dots, B - 1$ 。第  $j$  份使用的冗余版本为  $rv_{\text{idx}}(j) = 2 \cdot \text{mod}(rv_{\text{DCI}} + j, 2)$ 。可以看出, NPUSCH

format 1 传输的冗余版本  $rv$  只能为 0 或 2。关于 NPUSCH 如何映射到 RU 以及使用的 RV 的举例, 见 5.3.1.3 节的介绍。

**确定 TBS:** 如果  $N_{\text{sc}}^{\text{RU}} = 1$ , UE 会使用  $I_{\text{MCS}}$  查 36.213 的 Table 16.5.1.2.1-1 得到对应 NPUSCH 传输使用的

$I_{\text{TBS}}$  (此时  $I_{\text{TBS}}$  的取值范围为 0~10); 否则  $I_{\text{TBS}} = I_{\text{MCS}}$  (此时  $I_{\text{TBS}}$  的取值范围为 0~12)。然后, UE 会使

用  $(I_{\text{TBS}}, I_{\text{RU}})$  查 36.213 的 Table 16.5.1.2-2 得到对应 NPUSCH 传输的 TBS。

可以看出, 上行支持的最大 TBS 为 1000 比特。

**Table 16.5.1.2-2: Transport block size (TBS) table for NPUSCH.**

$I_{\text{TBS}}$	$I_{\text{RU}}$							
	0	1	2	3	4	5	6	7
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	872
6	88	176	256	392	504	600	808	1000
7	104	224	328	472	584	712	1000	
8	120	256	392	536	680	808		
9	136	296	456	616	776	936		
10	144	328	504	680	872	1000		
11	176	376	584	776	1000			
12	208	440	680	1000				

使用 multi-tone 传输时，NB-IoT 的上行峰值速率是使用最大 TBS 为 1000 比特（此时一个 TB 至少需要占用 4ms 来传输），并且重复次数为 1（最好的信道条件，不需要重复传输）时计算出的速率（L1 的峰值速率）： $1000(\text{bits}) \times 1000(\text{ms/second}) / 4(\text{ms}) = 250 \text{ kbps}$ 。这里假设频域上的所有 12 个子载波均用于传输该 TB。由于 1ms 内有 2 个 SC-FDMA 符号用于传输 NDMRS，因此 1ms 内只能使用 12 个 SC-FDMA 符号，即一个 RB 共有  $12 \times 12 = 144$  个 RE 可用于传输 NPUSCH format 1。在上行使用最高 QPSK 调制的情况下，1ms 至多能传输  $144 \times 2 = 288$  比特的信息。因此 TBS 为 1000 比特时，至少需要占用 4ms 来传输该 TB。

使用 single-tone 传输时，NB-IoT 的上行峰值速率是使用最大 TBS 为 1000 比特（ $I_{\text{TBS}}=10$ ），子载波间距为 15kHz，一个 TB 占用 6 个 RU 来传输，一个 RU 占用 8ms（16 slots），并且重复次数为 1（最好的信道条件，不需要重复传输）时计算出的速率（L1 的峰值速率）： $1000(\text{bits}) \times 1000(\text{ms/second}) / (6 \times 8) (\text{ms}) \approx 20 \text{ kbps}$ 。使用 single-tone 传输，频域上只有 1 个子载波可用于传输该 TB。使用 15kHz 子载波间距，一个 RU 占用 8ms（16 slots），每个 slot 上有 1 个 SC-FDMA 符号用于传输 NDMRS，即一个 RU 共有  $6 \times 16 = 96$  个 RE 可用于传输 NPUSCH format 1。在上行使用最高 QPSK 调制的情况下，1 个 RU 至多能传输  $96 \times 2 = 192$  比特的信息。因此 TBS 为 1000 比特时，至少需要占用 6 个 RU（48ms）来传输该 TB。

注意：这里并未将 DCI 占用的子帧、HARQ timing 关系等因素考虑在内，如果考虑这些因素，实际得到的上行峰值速率必然小于上面给出的理论值。

UE 收到的 NPDCCH 中携带的 NDI，以及通过上面介绍得到的 RV 和 TBS，会被发送给 MAC 层。

### 5.3.1.3 物理层映射

NPUSCH 在映射到 RE 上时，是按先频域后时域的顺序来映射的。调制后的符号映射到  $N_{\text{slots}}$  个 slot 后，会将这  $N_{\text{slots}}$  个 slot 继续重复  $M_{\text{identical}}^{\text{NPUSCH}} - 1$  次。然后继续映射接下来的  $N_{\text{slots}}$  个 slot，再继续重复  $M_{\text{identical}}^{\text{NPUSCH}} - 1$  次，直到  $M_{\text{rep}}^{\text{NPUSCH}} N_{\text{RU}} N_{\text{slots}}^{\text{UL}}$  个 slot 都映射完毕为止（见 36.211 的 10.1.3.6 节）。其中

$$M_{\text{identical}}^{\text{NPUSCH}} = \begin{cases} \min\left(\left\lceil M_{\text{rep}}^{\text{NPUSCH}} / 2 \right\rceil, 4\right) & N_{\text{sc}}^{\text{RU}} > 1 \\ 1 & N_{\text{sc}}^{\text{RU}} = 1 \end{cases}$$

$$N_{\text{slots}} = \begin{cases} 1 & \Delta f = 3.75 \text{ kHz} \\ 2 & \Delta f = 15 \text{ kHz} \end{cases}$$

可以看出，当使用 3.75kHz 子载波间距时，重复是基于一个 slot 的；当使用 15kHz 子载波间距时，重复是基于一个子帧（2 个 slot）的。

注：36.211 中的  $M_{\text{rep}}^{\text{NPUSCH}}$  等同于 36.213 中的  $N_{\text{Rep}}$ 。

接下来，我们会举几个例子来介绍 NPUSCH 在时域上的映射，以及使用的冗余版本 RV。

一个使用 multi-tone 的 TB 映射  
到 2 个 RU 上，每个 RU 占 2 个 slot

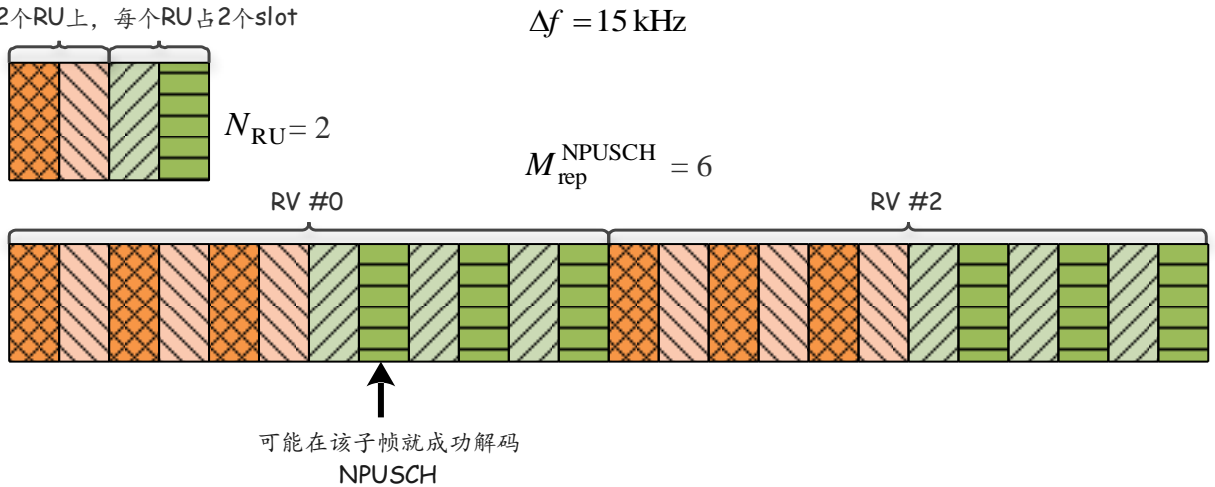


图 5-8: multi-tone 下的 NPUSCH 资源映射和使用的 RV 举例

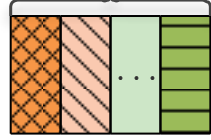
图 5-8 假设一个使用 multi-tone 的 TB 映射到 2 个 RU 上，每个 RU 占 2 个上行 slot，即一个 TB 要映射到 4 个上行 slot 上，并有  $\Delta f = 15 \text{ kHz}$ 、 $N_{\text{RU}} = 2$  且  $N_{\text{sc}}^{\text{RU}} > 1$ 。同时，重复次数  $M_{\text{rep}}^{\text{NPUSCH}} = 6$ 。根据前面的计算公式，我们可以得出  $M_{\text{identical}}^{\text{NPUSCH}} = 3$ ， $N_{\text{slots}} = 2$ 。即调制后的符号映射到 2 个 slot 后，会将这 2 个 slot 再重复 2 次（共 3 次）。然后继续映射后 2 个 slot 后，再重复 2 次。由于总的重复次数是 6 次，所以之前的过程需要再重复一次。

结合 5.3.1.2 节关于 RV 计算的介绍，这里  $N_{\text{Rep}}=6$ ， $N_{\text{sc}}^{\text{RU}} > 1$  有  $L = \min\left(4, \lceil N_{\text{Rep}} / 2 \rceil\right) = 3$ ，整个时域上的

资源会被分成  $\frac{N_{\text{Rep}}}{L} = 2$  份，这里假设  $rv_{\text{DCI}} = 0$ ，则有第一份 ( $j = 0$ ) 使用的 RV 等于

$rv_{\text{idx}}(0) = 2 \cdot \text{mod}(rv_{\text{DCI}} + 0, 2) = 0$ ，第二份 ( $j = 1$ ) 使用的 RV 等于  $rv_{\text{idx}}(1) = 2 \cdot \text{mod}(rv_{\text{DCI}} + 1, 2) = 2$ 。

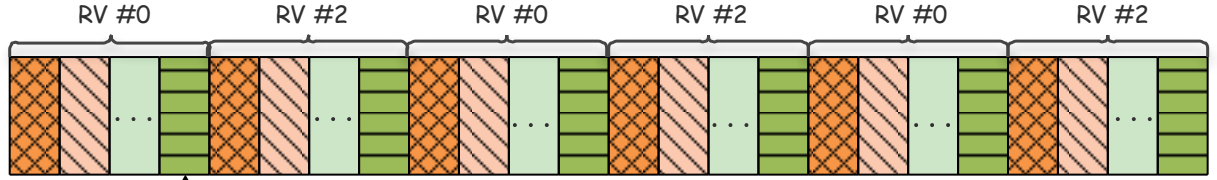
一个使用 single-tone 的 TB 映射  
到 1 个 RU 上，每个 RU 占 16 个 slot



$N_{\text{RU}} = 1$

$\Delta f = 3.75 \text{ kHz}$

$M_{\text{rep}}^{\text{NPUSCH}} = 6$



可能在该子帧就成功解码  
NPUSCH

图 5-9: single-tone 下的 NPUSCH 资源映射和使用的 RV 举例

图 5-9 假设一个使用 single-tone 的 TB 映射到 1 个 RU 上，该 RU 占 16 个 slot，即一个 TB 要映射到 16 个 slot 上，并有  $\Delta f = 3.75 \text{ kHz}$ 、 $N_{\text{RU}} = 1$  且  $N_{\text{sc}}^{\text{RU}} = 1$ 。同时，重复次数  $M_{\text{rep}}^{\text{NPUSCH}} = 6$ 。根据前面的计算公式，我们可以得出  $M_{\text{identical}}^{\text{NPUSCH}} = 1$ ， $N_{\text{slots}} = 1$ 。即调制后的符号映射到 1 个 slot 后，接着映射下一个 slot，直到该 RU 的所有调制符号都映射完。由于总的重复次数是 6 次，所以之前的过程需要再重复 5 次。

可以看出，使用 multi-tone 传输时，NPUSCH 传输采用的是“子帧级（2 个 slot）重复 + TB 级重复”的方式来实现重复传输的；使用 single-tone 传输时，NPUSCH 传输采用的是“TB 级（RU 级）重复”的方式来实现重复传输的。

结合 5.3.1.2 节关于 RV 计算的介绍，这里  $N_{\text{Rep}}=6$ ， $N_{\text{sc}}^{\text{RU}} = 1$ ，有  $L=1$ ，整个时域上的资源会被分成  $\frac{N_{\text{Rep}}}{L}$

=6 份，这里假设  $rv_{\text{DCI}} = 0$ ，则有第 1、3、5 份 ( $j = 0, 2, 4$ ) 使用的 RV 等于

$rv_{\text{idx}}(j) = 2 \cdot \text{mod}(rv_{\text{DCI}} + j, 2) = 0$ ，第 2、4、6 份 ( $j = 1, 3, 5$ ) 使用的 RV 等于

$rv_{\text{idx}}(j) = 2 \cdot \text{mod}(rv_{\text{DCI}} + j, 2) = 2$ 。

当 NPUSCH 传输与配置的 NPRACH 资源重叠时，重叠部分的 NPUSCH 会推迟到配置的 NPRACH 资源之后再传输。

在 NPUSCH 连续传输了 256ms 之后，会插入一个 40ms 的 UL gap，之后再继续 NPUSCH 传输，具体原因见 5.3.3 节的介绍。由 NPRACH 导致的延迟部分与一个 UL gap 相冲突时，该延迟部分也统计为 UL gap 的一部分。

当 UE 特定的配置 *npusch-AllSymbols-r13* 设置为 false 时，说明 NPUSCH 与 LTE 中的 SRS 传输在某些 SC-FDMA 符号上有重叠，此时在映射 NPUSCH 到 RE 上时，与 SRS 重叠的 SC-FDMA 符号上的 RE 会被统计，但不会用于 NPUSCH 传输。当 *npusch-AllSymbols-r13* 设置为 true 时，所有的符号都能被用于 NPUSCH 传输。

### 5.3.2 NPUSCH format 2

NPUSCH format 2 用于传输上行控制信息（UCI）。在 Rel-13 中，上行控制信息只有针对下行 NPDSCH 传输的确认信息，即 ACK/NACK。并且该 ACK/NACK 只支持 single-tone 传输。

对于 NPUSCH format 2 而言，一个 RU 总是由频域上的一个子载波和时域上的 4 个 slot 组成。因此，对应 3.75kHz 子载波间距，一个 RU 具有 8ms 的持续时间；对应 15kHz 子载波间距，一个 RU 具有 2ms 的持续时间。

表 5-5: NPUSCH format 2 占用的一个 RU 资源

NPUSCH format	携带的内容	x-tone	子载波间距	一个 RU 持续的时间 (占用的 slot 数 $N_{\text{slots}}^{\text{UL}}$ )	一个 RU 占用的子载波数 $N_{\text{sc}}^{\text{RU}}$
2	ACK/NACK	single-tone	3.75kHz	8ms (4 slots)	1
			15kHz	2ms (4 slots)	1

NPUSCH format 2 固定使用  $\pi/2$ -BPSK 调制。

NPUSCH format 2 使用的信道编码为 Block code（重复码），且码率固定为 1/16。其传输的数据是全‘0’（对应 NACK）或全‘1’（对应 ACK），以降低复杂度。（见 36.212 的 Table 6.2-2）

对于 NPUSCH format 2 而言，其传输在时域上的起始位置、重复次数以及所占的子载波，会在 6.1 节介绍下行 HARQ 时予以介绍。

### 5.3.3 UL gap

为了更好地支持上行覆盖增强，NB-IoT 系统在上行物理信道上也引入了重复传输的机制。通常位于极限覆盖下的 UE，其 NPUSCH 传输可能持续很长时间（至多 2.048s），主要原因在于：（1）大量的重复；（2）在极限覆盖下，通常会使用 3.75kHz 的 single-tone 传输，即只使用一个子载波，从而导致每个速率匹配后的码块（code block）在时域上要跨越较长的时间，简单地说，就是一次重复本身需要较长的时间来传输。

由于 NB-IoT UE 的低成本需求，配备了较低成本晶振的 NB-IoT UE 在连续长时间的上行传输后，UE 功率放大器的热耗散会导致发射机温度升高，进而导致晶振频率偏移。与此同时，使用半双工 FDD（HD-FDD）的 NB-IoT UE 在发送上行数据时，是不能同步接收下行信号的，因此无法进行时频偏补偿。连续长时间的上行传输所导致的频率偏移会严重影响到 UE 的上行传输性能，降低数据传输效率。为了纠正这种频率漂移，NB-IoT 中引入了 UL gap 的概念，即让需要长时间连续上行传输的 UE 在 UL gap 期间暂时中断上行传输，并在 UL gap 期间切换到下行链路，利用 NPSS/NSSS/NRS 信号进行同步跟踪以及时频偏补偿，经过一定时间的补偿后，UE 再切换回上行继续传输。

更确切地说，在上行（包括 NPUSCH 和 NPRACH 传输）连续传输了 256ms（对于 NPRACH，是连续传输  $4 \cdot 64(T_{CP} + T_{SEQ})$  个时间单元）之后，会加入一个 40ms 的 UL gap，在 UL gap 期间，UE 会切换到下行并进行时频同步。UL gap 之后，UE 可以切换回上行继续之前的上行传输。

## 5.4 上行功率控制

上行功率控制用于控制不同上行物理信道的发射功率。本节主要介绍 NPUSCH 的上行功率控制，NPRACH 的上行功率控制将在 7.3.3 节介绍。

在服务小区  $c$  的 NB-IoT 上行 slot  $i$  上发送的 NPUSCH 的发射功率  $P_{NPUSCH,c}(i)$  通过如下方式确定：

- 如果分配的 NPUSCH RU 的重复次数大于 2（说明 UE 处于深度覆盖），则  $P_{NPUSCH,c}(i) = P_{CMAX,c}(i)$  [dBm]；

- 如果分配的 NPUSCH RU 的重复次数小于等于 2，则

$$P_{NPUSCH,c}(i) = \min \left\{ P_{CMAX,c}(i), 10 \log_{10}(M_{NPUSCH,c}(i)) + P_{O\_NPUSCH,c}(j) + \alpha_c(j) \cdot PL_c \right\} \text{ [dBm]}.$$

其中

- $P_{CMAX,c}(i)$  是在 36.101 的 6.2.5F 节中定义的服务小区  $c$  的 NB-IoT 上行 slot  $i$  上配置的 UE 最大发射功率。
- $M_{NPUSCH,c}(i)$  的值与所选 RU 的带宽以及子载波间距相关：如果使用 3.75kHz 子载波间距，则  $M_{NPUSCH,c}(i)$  等于 1/4；如果使用 15kHz 子载波间距，则  $M_{NPUSCH,c}(i)$  的取值范围为 {1, 3, 6, 12}。
- $P_{O\_NPUSCH,c}(j)$  是一个由  $P_{O\_NOMINAL\_NPUSCH,c}(j)$ （单位为 dBm）和  $P_{O\_UE\_NPUSCH,c}(j)$ （单位为 dB）之和组成的一个参数。 $j=1$  对应于动态调度的 UL grant 调度的 NPUSCH 传输/重传； $j=2$  对应于 RAR 的 UL grant 调度的 NPUSCH 传输/重传（即 Msg3）。

- 对应  $j=1$ ,  $P_{O\_NOMINAL\_NPUSCH,c}(j)$  通过 SIB2-NB 的  $p0-NominalNPUSCH-r13$  配置,  
 $P_{O\_UE\_NPUSCH,c}(j)$  通过 UE 特定的  $p0-UE-NPUSCH-r13$  配置;
- 对应  $j=2$ ,  $P_{O\_UE\_NPUSCH,c}(2)=0$  且  $P_{O\_NOMINAL\_NPUSCH,c}(2)=P_{O\_PRE}+\Delta_{PREAMBLE\_Msg3}$ 。其中  
 $P_{O\_PRE}$  通过  $preambleInitialReceivedTargetPower$  配置;  $\Delta_{PREAMBLE\_Msg3}$  通过 SIB2-NB 的  
 $deltaPreambleMsg3-r13$  配置。
- 对于  $j=1$  而言, NPUSCH format 2 的  $\alpha_c(j)=1$ , NPUSCH format 1 的  $\alpha_c(j)$  值通过 SIB2-NB 的  $alpha-r13$  配置; 对于  $j=2$  而言,  $\alpha_c(j)=1$ 。  $\alpha_c(j)$  对应路径损耗的补偿因子: 1 对应完全路损补偿, 0 对应不补偿路损, 大于 0 小于 1 的值对应部分路损补偿。
- $PL_c$  是 UE 针对服务小区  $c$  计算的下行路径损耗估计值, 以 dB 为单位。并且  $PL_c = nrs-Power-r13 + nrs-PowerOffsetNonAnchor-r13 - \text{高层过滤后的 NRSRP}$ , 其中  $nrs-Power-r13$  通过 SIB2-NB 配置。  
 $nrs-PowerOffsetNonAnchor-r13$  是 UE 级的配置, 对应 non-anchor carrier 的 NRS EPRE 相对于 anchor carrier 的功率偏移, 并以 dB 为单位。NRSRP 的定义见 36.214, 高层过滤配置的定义见 36.331。

关于上行功率控制, 可参见 36.213 的 16.2.1 节的介绍。

#### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》, ROHDE & SCHWARZ
- [2] <http://www.sharetechnote.com/> 关于 IoT 的介绍
- [3] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [4] 3GPP TS 36.211 V13.4.0, 2016-12; Physical channels and modulation (Release 13)
- [5] 3GPP TS 36.212 V13.4.0, 2016-12; Multiplexing and channel coding (Release 13)
- [6] 3GPP TS 36.213 V13.4.0, 2016-12; Physical layer procedures(Release 13)
- [7] 3GPP TS 36.321 V13.4.0, 2016-12; Medium Access Control (MAC) protocol specification
- [8] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification
- [9] “R4-160884, Considerations on Frequency error in NB-IoT”, Sony
- [10] “R1-161904, On the need for UL transmission gaps for HD-FDD UEs in extreme coverage”, Intel Corporation

## 第6章 HARQ

在 NB-IoT 中，无论是上行 HARQ 还是下行 HARQ，均只使用一个 HARQ process，从而降低了对 UE 并行处理能力和缓存的要求。并且上下行均使用自适应（adaptive）、异步（asynchronous）HARQ。这意味着重传与前一次传输（包括新传和前一次重传）之间没有固定的 timing 关系，这使得调度可以更为灵活。

在 LTE 中，某次传输（包括新传和重传，PDSCH 和 PUSCH）与其对应的 ACK/NACK 之间有着固定的 timing 关系。但在 NB-IoT 中，某次传输（包括新传和重传，NPDSCH 和 NPUSCH）与其对应的 ACK/NACK 之间的 timing 关系并不固定，而是由该次传输对应的 DCI 指定的。具体处理见接下来的介绍。

### 6.1 下行 HARQ

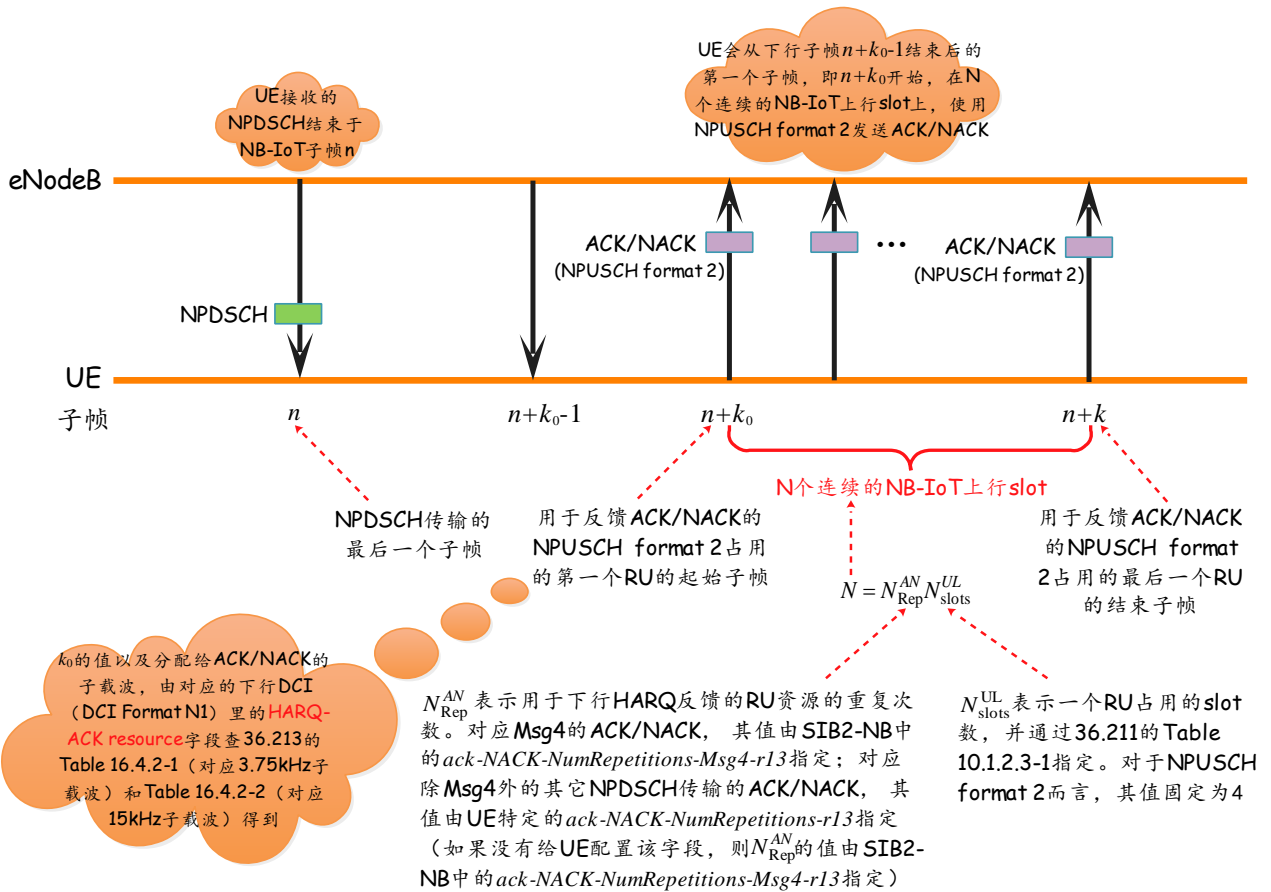


图 6-1: NB-IoT 下行 HARQ 处理流程

如果 UE 接收的 NPDSCH 传输结束于 NB-IoT 下行子帧  $n$ ，则 UE 会在下行子帧  $n+k_0-1$  结束后，在  $N$  个连续的 NB-IoT 上行 slot 上使用 NPUSCH format 2 来发送对应的 ACK/NACK（见 36.213 的 16.4.2 节）。其中，



- $N = N_{\text{Rep}}^{\text{AN}} N_{\text{slots}}^{\text{UL}}$ 。  $N_{\text{Rep}}^{\text{AN}}$  表示用于携带下行 HARQ 反馈的 RU 资源的重复次数（也即 ACK/NACK 的重复次数）。对应 Msg4 的 ACK/NACK，  $N_{\text{Rep}}^{\text{AN}}$  的值由 SIB2-NB 中的 *ack-NACK-NumRepetitions-Msg4-r13* 指定；对应除 Msg4 外的其它 NPDSCH 传输的 ACK/NACK，  $N_{\text{Rep}}^{\text{AN}}$  的值由 UE 特定的 *ack-NACK-NumRepetitions-r13* 指定（如果没有给 UE 配置该字段，则  $N_{\text{Rep}}^{\text{AN}}$  的值由 SIB2-NB 中的 *ack-NACK-NumRepetitions-Msg4-r13* 指定）。  $N_{\text{slots}}^{\text{UL}}$  表示一个 RU 占用的 slot 数，并通过 36.211 的 Table 10.1.2.3-1 指定（由于 ACK/NACK 在 NPUSCH format 2 上传输，查该表可知，  $N_{\text{slots}}^{\text{UL}}$  固定等于 4）。注：NPUSCH format 2 的一次重复（或者说一次传输）只占用一个 RU。
- $k_0$  的值以及分配给 ACK/NACK（NPUSCH format 2）的子载波，由对应的下行 DCI（DCI Format N1）里的 HARQ-ACK resource 字段查 36.213 的 Table 16.4.2-1 和 Table 16.4.2-2 得到。也就是说，用于指示 NPDSCH 传输的 DCI format N1，同时指定了对该 NPDSCH 传输进行 ACK/NACK 反馈的 NPUSCH format 2 的资源。

**Table 16.4.2-1: ACK/NACK subcarrier and  $k_0$  for NPUSCH with subcarrier spacing  $\Delta f = 3.75$  kHz.**

ACK/NACK resource field	ACK/NACK subcarrier	$k_0$
0	38	13
1	39	13
2	40	13
3	41	13
4	42	13
5	43	13
6	44	13
7	45	13
8	38	21
9	39	21
10	40	21
11	41	21
12	42	21
13	43	21
14	44	21
15	45	21

**Table 16.4.2-2: ACK/NACK subcarrier and  $k_0$  for NPUSCH with subcarrier spacing  $\Delta f = 15 \text{ kHz}$ .**

ACK/NACK resource field	ACK/NACK subcarrier	$k_0$
0	0	13
1	1	13
2	2	13
3	3	13
4	0	15
5	1	15
6	2	15
7	3	15
8	0	17
9	1	17
10	2	17
11	3	17
12	0	18
13	1	18
14	2	18
15	3	18

从前面的介绍可以看出，NPUSCH format 2（ACK/NACK）传输的起始时间与其相关的 NPDSCH 传输的结束时间之间的间隔大于等于 12ms。

## 6.2 上行 HARQ

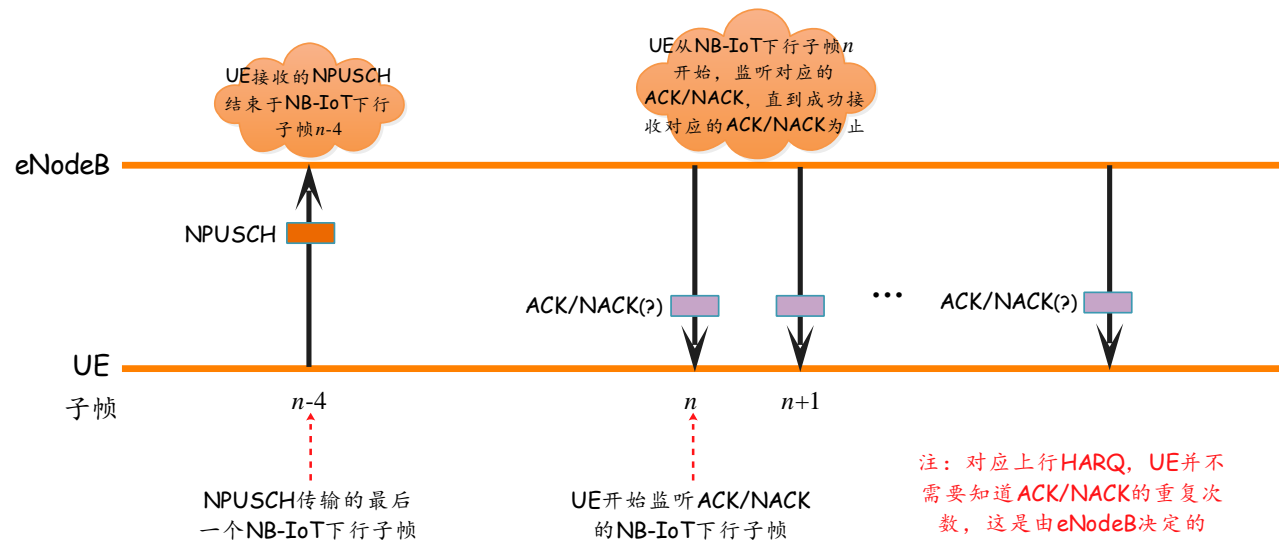


图 6-2: NB-IoT 上行 HARQ 处理流程

如果一个 NPUSCH 传输结束于 NB-IoT 下行子帧  $n-4$ ，则 UE 会从 NB-IoT 下行子帧  $n$  开始，去尝试接收对应的 ACK/NACK（监听 NPDCCH）。（见 36.213 的 16.5.2 节）

也就是说，针对 NPUSCH 传输的 ACK/NACK 的起始时间与其相关的 NPUSCH 传输的结束时间之间的间隔大于等于 3ms。

对应上行 HARQ，并不存在一个类似于 PHICH 的下行物理信道用来发送 ACK/NACK，更确切地说，eNodeB 并不会发送实际的 ACK/NACK 信息。UE 只能根据 DCI format N0 里的 NDI 字段是否翻转来决定是否要释放上行 HARQ buffer 里的内容，并确定是进行重传还是新传。也就是说，发完上行数据之后 UE 不能释放 HARQ buffer，只有到下一次 DCI format N0 到来之后，UE 才能确定是否清空 HARQ buffer 里的内容。

由于 NB-IoT 在上行使用异步自适应 HARQ，因此不需要配置上行的最大传输次数，即不需要配置类似于 LTE 中的  $maxHARQ-Tx$  和  $maxHARQ-Msg3Tx$ 。

#### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 3GPP TS 36.213 V13.4.0, 2016-12; Physical layer procedures(Relase 13)
- [3] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification

# 第7章 小区接入流程

当一个 UE 需要接入一个 NB-IoT 小区时，它遵循的原则与 LTE 是一样的：UE 需要先在合适的频率上搜索小区（小区搜索过程），然后读取相关的系统信息，再通过随机接入过程建立起与小区之间的 RRC 连接。如果 UE 还未向核心网注册，那么 UE 还会通过 NAS 层信令向核心网发起注册流程。在 UE 回到 RRC\_IDLE 态后，它需要重新发起随机接入过程才能向网络发送数据，或等待被寻呼。

## 7.1 小区搜索过程

为了获取下行的时间和频率的同步以及确定 NB-IoT 小区的 PCI（又称为 NCellID， $N_{ID}^{Ncell}$ ），NB-IoT 中定义了 2 种同步信号：NPSS（Narrowband Primary Synchronization Signal）和 NSSS（Narrowband Secondary Synchronization Signal）。NPSS 和 NSSS 被 NB-IoT UE 用于小区搜索，其作用类似于 LTE 中的 PSS 和 SSS。

NPSS 在每个系统帧的子帧 5 上传输，并在频域上占据 11 个子载波（ $k=0,1,...,N_{sc}^{RB}-2$ ，即 NPSS 并不占用第 12 个子载波），时域上占据最后 11 个 OFDM 符号（ $l=3,4,...,2N_{symb}^{DL}-1$ ）。（见 36.211 的 10.2.7.1.2 节）

NSSS 在每个满足  $n_f \bmod 2=0$  的系统帧（即在偶数系统帧）的子帧 9 上传输，并在频域上占据 12 个子载波，时域上占据最后 11 个 OFDM 符号（ $l=3,4,...,2N_{symb}^{DL}-1$ ）。（见 36.211 的 10.2.7.2.2 节）

NPSS/SSS 的时频资源位置如图 7-1 所示。

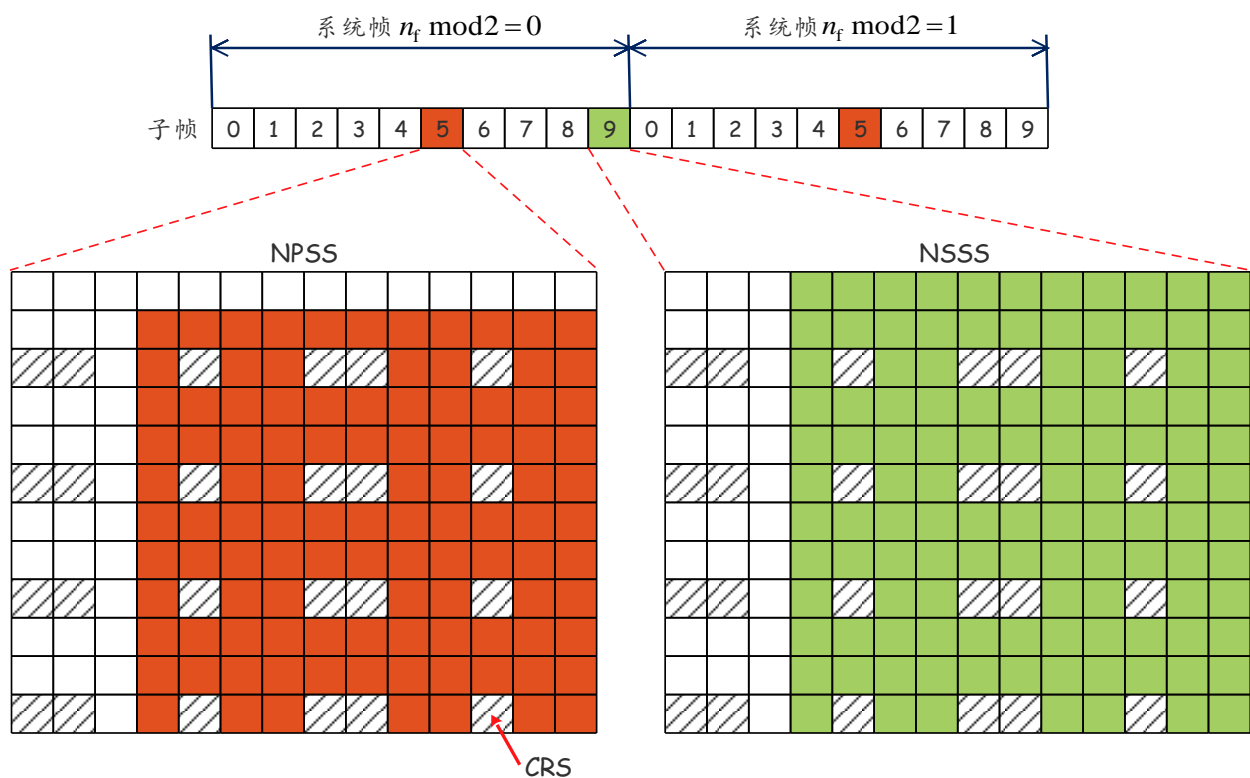


图 7-1: NPSS/NSSS 的时频资源位置

一个子帧的前 3 个 OFDM 符号 ( $l=0,1,2$ ) 因为可能被用于 LTE 的下行控制区域 (携带 LTE 中的 PCFICH/PHICH/PDCCH) 而不能被 NPSS 和 NSSS 使用。虽然只有带内部署时会存在 LTE 的下行控制区域, 但 NB-IoT UE 在进行小区搜索时, 还不知道小区会使用哪种部署方式, 并且 UE 要收到 SIB1-NB 后, 才能知道 LTE 的下行控制区域占用的确切 OFDM 符号数。因此对于 NPSS/NSSS 而言, 无论小区使用哪种部署方式, 前 3 个 OFDM 符号都是预留而不能用于 NPSS/NSSS 传输的。

LTE 中的小区特定的参考信号 (CRS) 所占的 RE 不能用于传输 NPSS/NSSS。虽然只有带内部署时会存在 CRS, 但 NB-IoT UE 在进行小区搜索时, 还不知道小区会使用哪种部署方式, 也不知道 CRS 使用的天线端口数以及 CRS 在频域上的偏移。但这些信息 eNodeB 是知道的, eNodeB 在映射 NPSS/NSSS 时, 会对 CRS 占用的 RE 进行“打孔”, 即这些 RE 不传输任何东西 (见 36.211 的 10.2.7.1.2 节和 10.2.7.2.2 节)。虽然 UE 并不知道哪些 RE 被“打孔”了, 但由于被“打孔”RE 的比例相对较低, NPSS 和 NSSS 依然能被正确地接收。注意: 这里并不一定是按 CRS 使用 4 天线端口来打孔的, 这是因为此时 UE 无法获知 LTE 小区的 PCI, 也就不知道 CRS 在频域上偏移, UE 无法知道 CRS 占用了哪些 RE, 所以 UE 按 4 天线端口来假设对 CRS 所占的 RE 进行打孔也没有意义。

下行的窄带参考信号 (NRS) 不会在包含 NPSS 或 NSSS 的子帧上传输, 因此 NPSS/NSSS 在映射到 RE 时, 不需要考虑 NRS 的影响。

NPSS/NSSS 未必与其它任何下行参考信号在相同的天线端口上传输。在某个子帧上发送的 NPSS/NSSS 可能与其它子帧上发送的 NPSS/NSSS 使用不同的天线端口。

从 36.211 的 10.2.7.1.1 节可以看出, NPSS 在频域上使用一个长为 11 的 ZC 序列, 并在时域上乘以一个长为 11 的  $S(l)$  (见 36.213 的 Table 10.2.7.1.1-1。对应时域上一个子帧的最后 11 个 OFDM 符号), 如图 7-2 所示。可以看出, 任一 RB 上传输的 NPSS 序列是固定不变的, 且 NPSS 本身不携带与小区相关的任何信息 (这点与 LTE 不同, LTE 中的 PSS 携带了  $N_{ID}^2$ )。由于 NPSS 固定在子帧 5 上发送, 因此**当 UE 接收到 NPSS, UE 就确定了 anchor carrier 以及子帧 5 的位置, 也就确定子帧 0 的位置, 即确定了帧边界。**

$$d_l(n) = S(l) \cdot e^{-j \frac{\pi \min(n+1)}{11}}, \quad n = 0, 1, \dots, 10$$

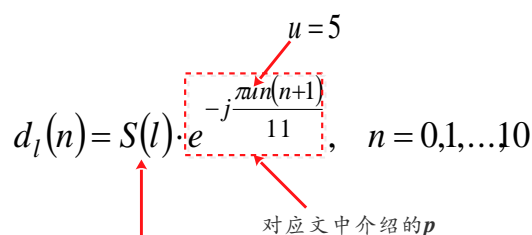


Table 10.2.7.1.1-1: Definition of  $S(l)$ .

Cyclic prefix length	$S(3), \dots, S(13)$										
Normal	1	1	1	1	-1	-1	1	1	1	-1	1

图 7-2: NPSS 序列的生成方式

从 UE 的角度上看, NPSS 检测是其计算量最大的操作之一。从 NPSS 序列的定义可以看出, 一个子帧上用于 NPSS 的每个 OFDM 符号, 传输的数据是  $p$  或  $-p$ , 其中  $p$  为基于一个 root index 为 5 且长为 11 的 ZC

序列生成的基序列，每个序列会映射到对应 PRB 的最低 11 个子载波上。这种设计有效地降低了 NPSS 检测的复杂性。

综上所述，通过 NPSS，UE 可以得到如下信息：

- 10 ms timing，即子帧 0 所在的位置；
- 频率同步，确定了 anchor carrier 的位置。

从 36.211 的 10.2.7.2.1 节可以看出，NSSS 序列长为 132（12 个子载波×11 个 OFDM 符号），并基于一个频域上长为 131 的 ZC 序列生成，如图 7-3 所示。NSSS 序列的生成与 NCellID ( $N_{ID}^{Ncell}$ ) 相关，UE 通过成功检测到的 NSSS，能够得到小区的 NCellID。与 LTE 类似，NB-IoT 也定义了 504 个不同 PCI 值，并称为 NCellID (Narrowband physical cell ID)。但 NB-IoT 中的 NCellID 值只在 NSSS 上携带，UE 通过接收 NSSS 来获取小区的 NCellID。

见 36.211 的 Table 10.2.7.2.1-1

$$d(n) = b_q(m) e^{-j2\pi\theta_f n} e^{-j\frac{\pi u n'(n'+1)}{131}}$$

$n = 0, 1, \dots, 131$   
 $n' = n \bmod 131$   
 $m = n \bmod 128$   
 $u = N_{ID}^{Ncell} \bmod 126 + 3$   
 $q = \left\lfloor \frac{N_{ID}^{Ncell}}{126} \right\rfloor$

循环移位 cyclic shift

$$\theta_f = \frac{33}{132} (n_f / 2) \bmod 4$$

结合 NSSS 只会在偶数系统帧上发送，可以推导出 80ms 时间块内的 timing，这可为后续 MIB-NB 盲检服务

NCellID

图 7-3: NSSS 序列的生成方式

通过图 7-3 我们还可以看出，NSSS 序列使用的循环移位  $\theta_f$  与系统帧号  $n_f$  相关，即有

$\theta_f = \frac{33}{132} (n_f / 2) \bmod 4$ 。又由于 NSSS 只在偶数系统帧上发送，基于  $\theta_f$  的计算公式，我们就能够得知一个 80ms 时间块内的 timing，即 SFN mod 8 等于 0、2、4 或 6 的系统帧位置，这是为减轻后续的盲检 MIB-NB 的负担而设计的。

综上所述，通过 NSSS，UE 可以得到如下信息：

- 小区的窄带 PCI，即  $N_{ID}^{Ncell}$ ；
- 80ms 时间块内的 timing。即 SFN mod 8 等于 0、2、4 或 6 的系统帧位置。

NB-IoT UE 在小区搜索过程之后，已经获得了继续解码 NPBCH 所需的所有信息。接下来，UE 会去接收 MIB-NB 以获取最重要的系统信息。

## 7.2 系统信息

与 LTE 类似，NB-IoT 也使用系统信息来广播小区级的信息。为了减少资源的占用和降低功耗，NB-IoT 中的系统信息更小，发送的频率也更低。

NB-IoT 中定义了多种系统信息，如表 7-1 所示。

表 7-1：NB-IoT 系统信息类型

类型	内容	周期（ms）	传输信道
MIB-NB	包含一些数量有限但最重要也最频繁发送的参数，UE 必须使用这些参数来获取其它的系统信息。见 36.331 的 IE: <i>MasterInformationBlock-NB</i>	640	BCH
SIB1-NB	包含小区接入和小区选择相关的参数，以及其它 SIB 的调度信息。见 36.331 的 IE: <i>SystemInformationBlockType1-NB</i>	2560	DL-SCH
SIB2-NB	包含了公共的无线资源配置信息，对所有 UE 通用	由SIB1-NB配置，见36.331的 IE: <i>SystemInformationBlockType1-NB</i> 的字段: <i>SchedulingInfoList-r13</i>	
SIB3-NB	包含了 intra-frequency 和 inter-frequency 小区重选相关的参数		
SIB4-NB	包含了用于 intra-frequency 小区重选的邻居小区的相关信息		
SIB5-NB	包含了用于 inter-frequency 小区重选的邻居小区的相关信息		
SIB14-NB	包含了 Access Barring 参数		
SIB16-NB	包含了与 GPS 时间和 UTC 时间相关的信息		

为了与 LTE 区分开，NB-IoT 中的 MIB/SIB 信息以“-NB”后缀结尾。对于 UE 而言，MIB-NB、SIB1-NB 以及 SIB2-NB 至 SIB5-NB 是必须的。而其它 SIB 只有在部署了相关功能时才需要。例如，如果在 MIB-NB 中指示了使能接入禁止（access barring）功能，则 UE 必须获取一个有效的 SIB14-NB。

LTE 不支持公共告警系统，如 CMAS、ETWS 和 PWS，因此不存在相关的 SIB。

与 LTE 不同，NB-IoT UE 只有在 RRC\_IDLE 态时才会去获取系统信息或检测系统信息变更。UE 在 RRC\_CONNECTED 态时是不会去读取系统信息的。如果系统信息发生了变化，小区会通过 *Paging-NB* 或 DCI format N2 中的 Direct Indication information 字段直接指示系统信息发生了变化。eNodeB 还可以将 UE 释放到 RRC\_IDLE 态，以便其获取变更后的系统信息。

## 7.2.1 MIB-NB

MIB-NB (*MasterInformationBlock-NB*) 在 NPBCH 上传输。UE 通过检测 NPBCH，能得到以下信息：

1. 通过接收到的 MIB-NB 可以知道小区的部署方式、SFN 的高 4 位、超帧（H-SFN）的低 2 位以及 SIB1-NB 的调度信息和大小等。另外，SFN 的低 6 位可以隐式地通过盲检 NPBCH 得到；
2. NRS 使用的天线端口数：1 或 2；
3. 如果使用的是带内部署，还可知道 LTE 中的 CRS 使用的天线端口数及其在频域上的偏移。

MIB-NB 包含了 34 比特的信息，如下所示。其中 11 比特是预留给后续扩展用的。

```
MasterInformationBlock-NB ::= SEQUENCE {
    systemFrameNumber-MSB-r13      BIT STRING (SIZE (4)),    ----SFN的高4位（4比特）。SFN的低6位通过NPBCH盲检得到
    hyperSFN-LSB-r13               BIT STRING (SIZE (2)),    ----超帧（H-SFN）的低2位（2比特）。H-SFN的高8位在SIB1-NB
中指示
    schedulingInfoSIB1-r13          INTEGER (0..15),          ----指定SIB1-NB的调度信息和大小（TBS），见7.2.2节的介绍（4
比特）
    systemInfoValueTag-r13          INTEGER (0..31),          ----用于指示除MIB-NB/SIB14-NB/SIB16-NB外的系统信息变更（5
比特），见7.2.4节的介绍
    ab-Enabled-r13                  BOOLEAN,                  ----指示是否使用access barring（1比特）。如果使能，UE会
在RRC连接建立或恢复之前，接收SIB14-NB并进行access barring检查。具体见8.7节的介绍
    operationModeInfo-r13           CHOICE {                  ----指示NB-IoT的部署模式（7比特）
        inband-SamePCI-r13          Inband-SamePCI-NB-r13,    ----使用带内部署，且NB-IoT小区与LTE小区使用相同的
PCI，NRS和CRS的天线端口数也相同
        inband-DifferentPCI-r13     Inband-DifferentPCI-NB-r13,----使用带内部署，且NB-IoT小区与LTE小区使用不同的PCI
        guardband-r13              Guardband-NB-r13,          ----使用保护频带部署
        standalone-r13             Standalone-NB-r13          ----使用独立部署
    },
    spare                            BIT STRING (SIZE (11))    ----预留给后续扩展使用（11比特）
}
```

```
ChannelRasterOffset-NB-r13 ::= ENUMERATED {khz-7dot5,kHz-2dot5,kHz2dot5,kHz7dot5}
```

```
Guardband-NB-r13 ::= SEQUENCE {
    rasterOffset-r13                ChannelRasterOffset-NB-r13, ----NB-IoT小区中心频点相对于最近的100kHz频点的偏移。取值范
围为{-7.5, -2.5, 2.5, 7.5}，并以kHz为单位
    spare                            BIT STRING (SIZE (3))
}
```

```
Inband-SamePCI-NB-r13 ::= SEQUENCE {
    eutra-CRS-SequenceInfo-r13      INTEGER (0..31)          ----见7.2.1.1节的介绍
}
```

```
Inband-DifferentPCI-NB-r13 ::= SEQUENCE {
    eutra-NumCRS-Ports-r13          ENUMERATED {same, four}, ----指示LTE载波中使用的CRS天线端口数。该值或等于NRS
的天线端口数，或等于4。
}
```



```

    rasterOffset-r13          ChannelRasterOffset-NB-r13,    ----NB-IoT小区中心频点相对于最近的100kHz频点的偏移。取值范
围为{-7.5, -2.5, 2.5, 7.5}, 并以kHz为单位
    spare                      BIT STRING (SIZE (2))
}

Standalone-NB-r13 ::=      SEQUENCE {
    spare                      BIT STRING (SIZE (5))
}

```

MIB-NB 使用 640ms 的固定调度周期，同一 MIB-NB 会在 640 ms 内的每一个系统帧上重复发送多次。MIB-NB 的第一次传输在满足  $n_f \bmod 64 = 0$  的系统帧的子帧 0 上发送，并在接下来的 640ms 内的每个系统帧的子帧 0 上重复。

物理层收到一个 MIB-NB 后，会先添加一个 16 比特的 CRC，然后使用 TBCC 进行信道编码，并经过速率匹配后，输出 1600 比特的数据。这 1600 比特会被分成 8 个等长的可自解码的子块（sub-block），每个子块重复发送 8 次，即每个子块会在 8 个连续的系统帧的子帧 0 上重复发送。这样设计的目的是为了处于深度覆盖的 UE 能够成功接收 MIB-NB。而使用子帧 0 来传输 NPBCH 的目的是为了避免在带内部署时，与 LTE 中可能存在的 MBSFN 传输相冲突。

NPBCH 固定使用 QPSK 调制。

NRS 和 LTE CRS 所占的 RE 不能用于 NPBCH 传输，并且 NPBCH 映射到 RE 时，总是假定 NRS 使用 2 天线端口（天线端口 2000 和 2001），CRS 使用 4 天线端口（天线端口 0~3）来剔除那些不能用于 NPBCH 的 RE。这样做的原因是 UE 只有读取 MIB-NB 后才能得到真实的天线端口信息，在这之前，只能按照最大天线端口数来剔除那些不能使用的 RE。另外，UE 在接收 NPBCH 时，是使用  $N_{ID}^{Ncell}$  替换  $N_{ID}^{cell}$  来计算 CRS 的频率偏移的。也就是说，UE 在小区搜索过程之后，就知道了 CRS 在频域上的偏移。

与 NPSS/NSSS 类似，一个子帧的前 3 个 OFDM 符号可能被 LTE 的下行控制区域使用，所以不能用于 NPBCH 传输。

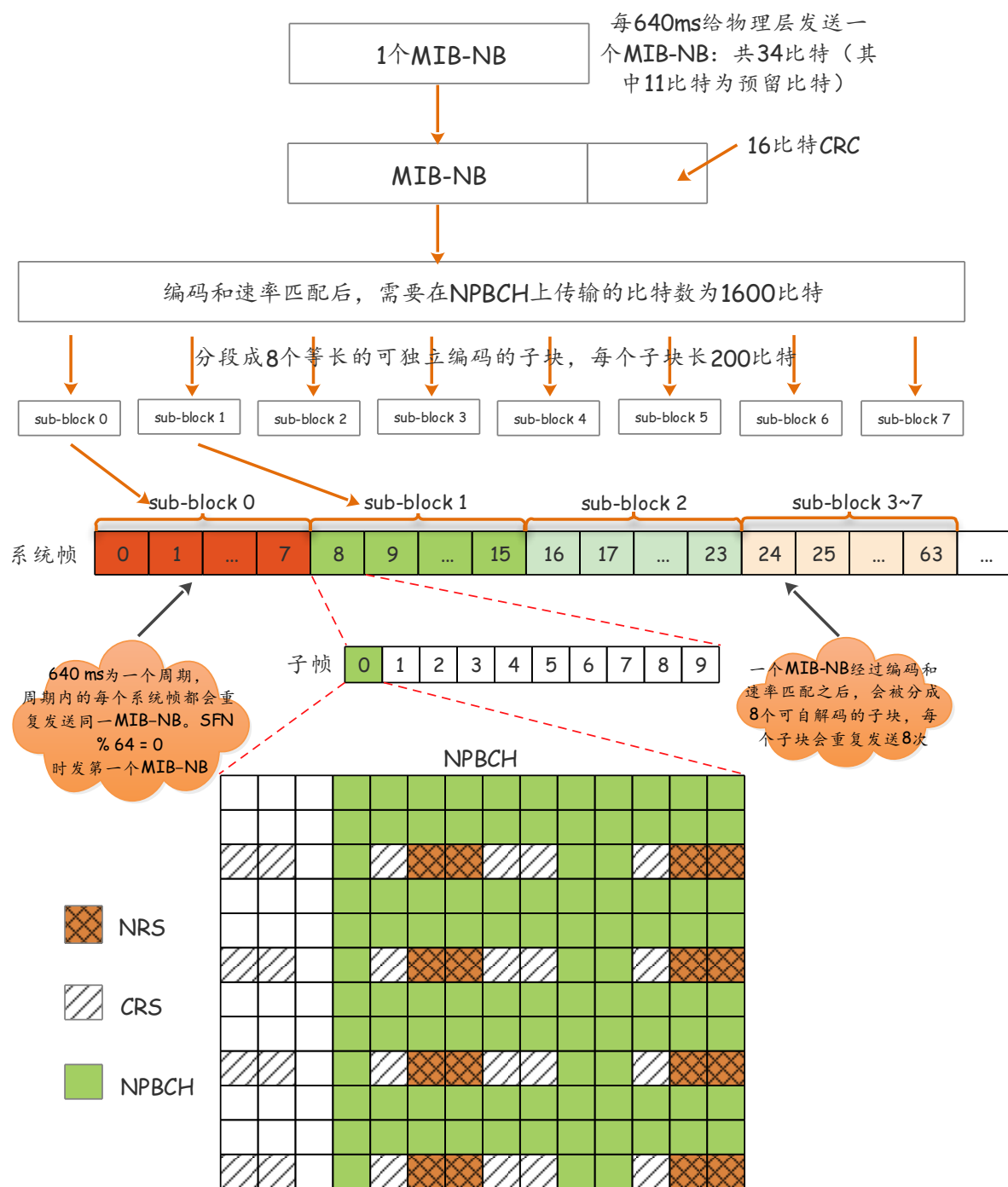


图 7-4: NPBCH 的物理层处理和资源映射

NB-IoT 中的下行物理信道 (NPDCCH/NPDSCH) 在映射到 RE 时, 需要剔除 NRS 占用的那些 RE。在带内部署时, 还要剔除 LTE 的 CRS 占用的那些 RE。NRS 和 CRS 占用的 RE 与其使用的天线端口数相关, 而 NRS 和 CRS 占用的 RE 在频域上的位置还与其频率偏移相关, 因此我们首先要确定 NRS 和 CRS 占用的天线端口数以及在频域上的频率偏移。

NPBCH 使用的 CRC mask, 跟 LTE 中的 1 或 2 天线端口 PBCH 使用的 CRC mask 相同, 如 36.212 的 Table 5.3.1.1-1 所示。因此通过使用不同的 CRC mask 去盲检 NPBCH, 就知道了 NB-IoT 中使用的 NRS 天线端口数。

**Table 5.3.1.1-1: CRC mask for PBCH.**

Number of transmit antenna ports at eNodeB	PBCH CRC mask $\langle x_{ant,0}, x_{ant,1}, \dots, x_{ant,15} \rangle$
1	$\langle 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$
2	$\langle 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 \rangle$

成功盲检了 NPBCH 后，UE 就获取到了 MIB-NB，并知道了 NB-IoT 的部署方式，CRS 占用的天线端口数也就确定了：

- 如果使用独立部署或保护频带部署，那么 NB-IoT 下行传输占用的资源上是不存在 CRS 的，此时不需要计算 CRS 占用的天线端口数。
- 如果使用带内部署，并且 NB-IoT 小区与 LTE 小区使用相同的 PCI（即  $N_{ID}^{Ncell}$  等于  $N_{ID}^{cell}$ ），那么 NRS 和 CRS 的天线端口数也相同，此时 CRS 的天线端口数等于前面得到的 NRS 的天线端口数。
- 如果使用带内部署，并且 NB-IoT 小区与 LTE 小区使用不同的 PCI（即  $N_{ID}^{Ncell}$  不等于  $N_{ID}^{cell}$ ），那么 CRS 的天线端口数会在 MIB-NB 的 *eutra-NumCRS-Ports-r13* 中指定。

虽然带内部署时， $N_{CellID}$  与 LTE 的 PCI 可能不同，但在选择  $N_{CellID}$  时，会限制其范围以便它与 PCI 指向相同的频率偏移，这样 UE 也就能知道 CRS 在频域上的偏移，即使用  $N_{ID}^{Ncell}$  替换  $N_{ID}^{cell}$  来计算小区特定的参考信号（CRS）的频率偏移，这样也知道了 CRS 所占的 RE 在频域上的位置。

80ms 的边界在检测 NSSS 中已经得到，接下来 UE 可以通过使用 8 个可能的子块位置中的每一个去尝试解码 NPBCH，如果解码成功，也就知道了小区是在 640ms 内的第几个 80ms 时间块上发送 MIB-NB，也就知道了 SFN 的低 6 位。

#### 7.2.1.1 UE 获取 CRS 的参考信号序列及其栅格偏移的流程

在 4.4 节介绍“下行功率分配”时，我们提到过，使用带内部署且 NB-IoT 小区与 LTE 小区使用相同的 PCI 时，UE 也可以使用 CRS 来进行信道估计。NB-IoT UE 想要使用 CRS 来进行信道估计，就需要知道 NB-IoT 载波上的 CRS 使用的参考信号序列及其在频域上的确切位置。

在《深入理解 LTE-A》一书中提到过，不管 LTE 小区的系统带宽是多少，CRS 的参考信号序列都是以频域上的最大可能的系统带宽 110 RB 来定义的。在映射到 RE 时，只有实际的系统带宽内的参考符号会被传输，并且不管小区实际系统带宽是多少，频带中心部分的参考符号是相同的。也就是说，只要我们知道 anchor carrier 所在的 RB 偏离 LTE 小区中心多少个 PRB，就能知道该 PRB 上所传输的 CRS 参考信号序列。

另外我们在 2.3 节还介绍过，当 LTE 小区使用 10MHz 或 20MHz 带宽（此时  $N_{RB}^{DL}$  为偶数）时，anchor carrier 的中心距离最近的 100kHz 栅格的距离可能为 +2.5kHz/-2.5kHz。当 LTE 小区使用 3MHz、5MHz 或

15MHz 带宽（此时  $N_{RB}^{DL}$  为奇数）时，anchor carrier 的中心距离最近的 100kHz 栅格的距离可能为+7.5kHz/-7.5kHz。也就是说，在（通过小区搜索过程）确定了 NB-IoT 小区的中心频点后，我们还需要知道 NB-IoT 小区相对于 LTE 信道栅格的偏移，以便进行频偏补偿。

使用带内部署且 NB-IoT 小区与 LTE 小区使用相同的 PCI 时，上述信息是通过 MIB-NB 的 *eutra-CRS-SequenceInfo-r13* 字段通知 UE 的，并通过查 36.213 的 Table 16.8-1 得到所需的信息。其中

$$n'_{PRB} = n_{PRB} - \lfloor N_{RB}^{DL} / 2 \rfloor。$$

**Table 16.8-1: Definition of *eutra-CRS-SequenceInfo***

<i>eutra-CRS-SequenceInfo</i>	E-UTRA PRB index $n'_{PRB}$ for odd number of $N_{RB}^{DL}$	Raster offset	<i>eutra-CRS-SequenceInfo</i>	E-UTRA PRB index $n'_{PRB}$ for even number of $N_{RB}^{DL}$	Raster offset
0	-35	-7.5 kHz	14	-46	+2.5 kHz
1	-30		15	-41	
2	-25		16	-36	
3	-20		17	-31	
4	-15		18	-26	
5	-10		19	-21	
6	-5		20	-16	
7	5	+7.5 kHz	21	-11	
8	10		22	-6	
9	15		23	5	
10	20		24	10	-2.5 kHz
11	25		25	15	
12	30		26	20	
13	35		27	25	
			28	30	
			29	35	
			30	40	
			31	45	

### 7.2.2 SIB1-NB

SIB1-NB 包含了小区接入和小区选择相关的参数，超帧（H-SFN）的高 8 位以及其它 SIB-NB 的调度信息和 TBS 等。

SIB1-NB 在 NPDSCH 上传输，并使用 SI-RNTI 进行加扰，其调度周期固定为 2560ms。一个 SIB1-NB 可重复发送多次，携带 SIB1-NB 的 NPDSCH 的重复次数（4、8 或 16）通过 MIB-NB 的 *schedulingInfoSIB1-r13* 字段查 36.213 的 Table 16.4.1.3-3 得到。此重复次数是在 2560ms 周期内的重复次数，并在该周期内等间隔地重复。每次重复会在 16 个连续帧内的每隔一帧（共 8 个无线帧）的子帧 4 上传输。

**Table 16.4.1.3-3: Number of repetitions for NPDSCH carrying *SystemInformationBlockType1-NB*.**

Value of <i>schedulingInfoSIB1</i>	Number of NPDSCH repetitions
0	4
1	8
2	16
3	4
4	8
5	16
6	4
7	8
8	16
9	4
10	8
11	16
12-15	Reserved

携带 SIB1-NB 的 NPDSCH 的第一次传输的起始系统帧由重复次数和  $N_{ID}^{Ncell}$  确定，并通过 36.213 的 Table 16.4.1.3-4 指定。相邻小区可以通过配置不同的  $N_{ID}^{Ncell}$  来使得不同小区的 SIB1-NB 的起始无线帧不同，从而保证不同小区的 SIB1-NB 在时域上错开，避免小区间的 SIB1-NB 传输干扰。

**Table 16.4.1.3-4: Starting radio frame for the first transmission of the NPDSCH carrying *SystemInformationBlockType1-NB*.**

Number of NPDSCH repetitions	$N_{ID}^{Ncell}$	Starting radio frame number for NB-SIB1 repetitions ( $n_f \bmod 256$ )
4	$N_{ID}^{Ncell} \bmod 4 = 0$	0
	$N_{ID}^{Ncell} \bmod 4 = 1$	16
	$N_{ID}^{Ncell} \bmod 4 = 2$	32
	$N_{ID}^{Ncell} \bmod 4 = 3$	48
8	$N_{ID}^{Ncell} \bmod 2 = 0$	0
	$N_{ID}^{Ncell} \bmod 2 = 1$	16
16	$N_{ID}^{Ncell} \bmod 2 = 0$	0
	$N_{ID}^{Ncell} \bmod 2 = 1$	1

SIB1-NB 在一个子帧的第一个 slot 的起始 OFDM 符号  $l_{\text{DataStart}}$  通过如下方式确定：如果使用带内部署（即 *operationModeInfo-r13* 的值指示‘00’或‘01’），则  $l_{\text{DataStart}} = 3$ ；否则  $l_{\text{DataStart}} = 0$ 。（见 36.213 的 16.4.1.4 节）。

在带内部署时，一个子帧要固定预留前 3 个 OFDM 符号的原因在于 UE 只有解码 SIB1-NB 后，才能知道一个子帧内有多少个 OFDM 符号要预留给 LTE 中的下行控制区域使用。因此在解码 SIB1-NB 本身时，只能按照最大 3 个 OFDM 符号来预留给 LTE 中的下行控制区域使用。注：LTE 中的下行控制区域所占的 OFDM 符号数是通过 SIB1-NB 的 *eutraControlRegionSize-r13* 字段设置的（针对 anchor carrier）。

图 7-5 是使用带内部署时，SIB1-NB 映射到时频资源上的一个例子。

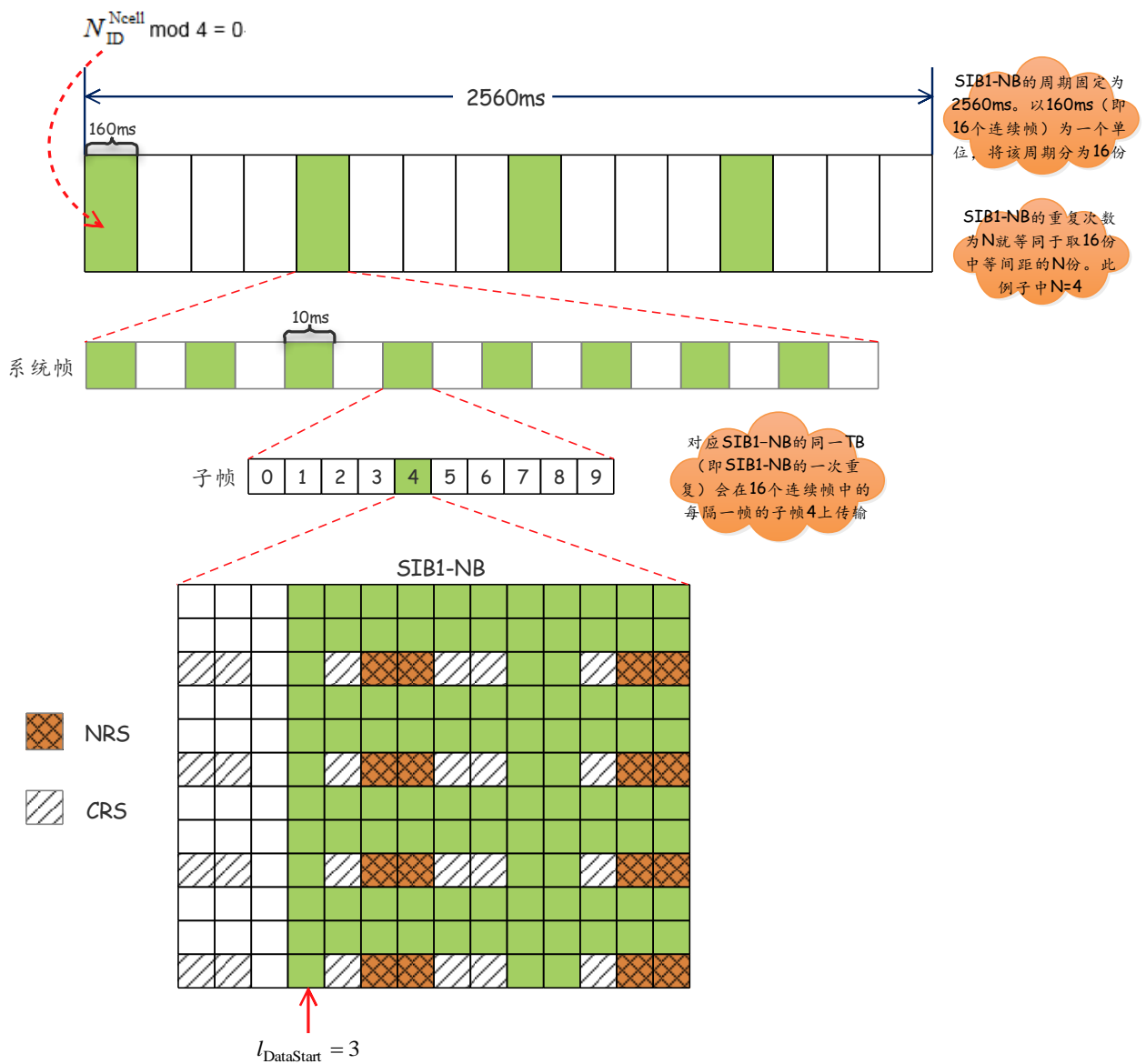


图 7-5: SIB1-NB 的资源映射（带内部署）

SIB1-NB 的 TBS 的计算方式见 4.3.2.3 节的介绍。

由于 SIB1-NB 相关的资源配置参数都是固定的：时域上的位置、重复次数、TBS 由 MIB-NB 和小区 PCI 指示，频域上位于 anchor carrier 上，并固定使用 QPSK 调制，因此并不需要任何控制信道（NPDCCH）相关的信息来指示如何传输 SIB1-NB。即 UE 接收 SIB1-NB 时，并不需要去检测对应的 NPDCCH。这与 LTE 中 SIB1 的频域位置和传输格式由 SI-RNTI 加扰的 PDCCH 指示是不同的。

### 7.2.3 SI 消息

除了 SIB1-NB 外的其它 SIB-NB 信息的调度方式类似于 LTE：一个 SI 消息包含一个或多个除 SIB1-NB 外的拥有相同调度需求的 SIB-NB（这些 SIB-NB 有相同的传输周期），并在独立的 SI 窗口内发送。不同 SI 消息的 SI 窗口互不重叠。SI 窗口的长度在 SIB1-NB 中指示，并且所有 SI 消息的 SI 窗口的长度是相同的。一个 SIB-NB 或 SI 消息的最大长度为 680 比特。SI 消息固定使用 QPSK 调制。

在一个 SI 窗口内，SI 消息根据他们的传输块大小（TB Size）在 2 个或 8 个连续的有效下行子帧上发送，在一个 SI 窗口内，对应的 SI 消息可以重复传输多次。SI 消息的所有调度信息在 SIB1-NB 中指示，因此不需要再去接收对应的 NPDCCH。

每个 SI 消息只在一个 SI 窗口（SI-window）内传输：（1）一个 SI 消息跟一个 SI 窗口相关联，该 SI 窗口内只能发这个 SI 消息且可以重复发送多次，但不能发送其它 SI 消息；（2）SI 窗口之间是紧挨着的（如果相邻的话），既不重叠，也不会有空隙；（3）所有 SI 消息的 SI 窗口长度都相同；（4）不同 SI 消息的周期是相互独立的。

SI 消息的调度信息以及 TBS 等信息在 SIB1-NB 的 *schedulingInfoList-r13* 字段指定（而在 LTE 中，SI 消息的调度信息和传输格式等信息是在 SI-RNTI 加扰的 PDCCH 中指定的），因此与 SIB1-NB 类似，SI 消息无需 SI-RNTI 加扰的 NPDCCH 来指示如何传输。SIB1-NB 中与 SI 消息调度相关的字段如下：

```
SystemInformationBlockType1-NB ::= SEQUENCE {
    ...
    downlinkBitmap-r13                DL-Bitmap-NB-r13                OPTIONAL, -- Need OP,      ----用于下行
    schedulingInfoList-r13             SchedulingInfoList-NB-r13, ----SI消息调度信息列表
    si-WindowLength-r13                ENUMERATED {ms160, ms320, ms480, ms640,
                                                ms960, ms1280, ms1600, spare1}, ----SI窗口的大小
    si-RadioFrameOffset-r13            INTEGER (1..15)                OPTIONAL, -- Need OP      ----用于计算SI窗口在周期内的
    ...                                偏移，以系统帧为单位
}

SchedulingInfoList-NB-r13 ::= SEQUENCE (SIZE (1..maxSI-Message-NB-r13)) OF SchedulingInfo-NB-r13

SchedulingInfo-NB-r13 ::= SEQUENCE { ----每个SI消息的调度信息
    si-Periodicity-r13                ENUMERATED {rf64, rf128, rf256, rf512,
                                                rf1024, rf2048, rf4096, spare}, ----SI消息的周期，以系统帧为单位
    si-RepetitionPattern-r13          ENUMERATED {every2ndRF, every4thRF,
                                                every8thRF, every16thRF}, ----SI消息的重复样式
}
```

sib-MappingInfo-r13	SIB-MappingInfo-NB-r13,	----该SI消息里包含的SIB-NB列表
si-TB-r13	ENUMERATED {b56, b120, b208, b256, b328, b440, b552, b680}	----指定SI消息的TBS和用于SI的连续NB-IoT下行子帧数
}		
SIB-MappingInfo-NB-r13 ::=	SEQUENCE (SIZE (0..maxSIB-1)) OF SIB-Type-NB-r13	
SIB-Type-NB-r13 ::=	ENUMERATED { sibType3-NB-r13, sibType4-NB-r13, sibType5-NB-r13, sibType14-NB-r13, sibType16-NB-r13, spare3, spare2, spare1}	
		----SIB-NB类型。没有SIB2-NB，因为它总是放在第一个SI消息中

SIB1-NB的 *schedulingInfoList-r13* 指定了 SI 消息的列表，每个 SI 消息在该列表中的顺序以  $n$  表示（从 1 开始计数）。假如 *schedulingInfoList-r13* 中指定了 3 个 SI 消息，则会有 3 个 SI 窗口用于发送这 3 个 SI 消息，而  $n$  表明了 SI 消息在第几个 SI 窗口。

此时每个 SI 消息有一个  $x = (n - 1) * w$ ，其中  $w$  为 SI 窗口的长度，并通过 *si-WindowLength-r13* 设置。可以看出， $x$  是以 ms 为单位的。

SI 窗口起始于满足  $(H-SFN * 1024 + SFN) \bmod T = \text{FLOOR}(x/10) + \text{Offset}$  的系统帧的子帧 0。其中  $T$  为对应 SI 消息的周期，由 *si-Periodicity-r13* 指定（以系统帧为单位，即以 10ms 为单位）， $\text{FLOOR}(x/10) + \text{Offset}$  用于确定 SI 窗口在周期内的起始位置，其中 *Offset* 由 *si-RadioFrameOffset-r13* 指定。

UE 会从 SI 窗口的开始，去接收和累积在 DL-SCH 上传输的 SI 消息，并持续到 SI 窗口的结束。在每个 SI 窗口内，UE 会从 *si-RepetitionPattern-r13* 指示的系统帧，*downlinkBitmap-r13* 指示的子帧开始，直到成功地解码累积的 SI 消息传输为止。即 SI 消息会在重复模式定义的系统帧内的第一个有效下行子帧开始，占用连续的有效下行子帧，直到一次完整的重复发送完毕为止。用于传输 SI 消息的子帧是不包含那些用于传输 NPSS/NSSS/MIB-NB/SIB1-NB 的子帧的。

*si-TB-r13* 指定了 SI 消息的 TBS 以及用于传输 SI 消息的连续 NB-IoT 下行子帧数。如果 TBS 为 56 或 120 比特，则一个 TB 需要使用 2 个下行子帧传输；而对应其它的 TBS，一个 TB 需要使用 8 个下行子帧传输。

如果 *si-RepetitionPattern-r13* 指示的系统帧内没有足够的子帧用来传输一个 SI 消息，那么 UE 会在紧接着 *si-RepetitionPattern-r13* 指示的系统帧之后的系统帧内继续接收 SI 消息。

协议中并不要求 UE 并行地累积多个 SI 消息，但取决于覆盖条件，可能要求 UE 累积多个 SI 窗口上传输的同一个 SI 消息。如果在 SI 窗口结束时无法从累积的 SI 消息传输中解码出 SI 消息，则 UE 会在相关 SI 消息的下一个 SI 窗口内继续在 DL-SCH 上进行接收和累积 SI 消息传输。

#### 7.2.4 系统信息有效性和变更通知

与 LTE 类似，NB-IoT 中的系统信息（除 SIB14-NB 和 SIB16-NB 外）的变更只发生在某些特定的系统帧，这里也用到了变更周期（modification period）的概念。系统信息可能会在一个变更周期内传输多次，但在同一变更周期内，系统信息的内容不会发生变化。



当小区修改了某些系统信息时，它会先在一个变更周期内通知 UE 系统信息将发生变化（但并不发送更新后的内容），然后在紧接着的下一个变更周期，小区才会发送更新后的系统信息。如 36.331 的 Figure 5.2.1.3-1 所示（不同的颜色表示不同的系统信息）。

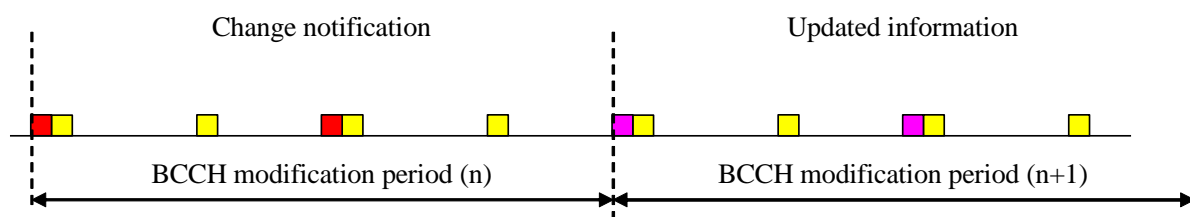


Figure 5.2.1.3-1: Change of system Information

对于 NB-IoT 而言，变更周期的起始系统帧（也即变更周期的边界）必须满足  $(H-SFN * 1024 + SFN) \bmod m = 0$ ，其中  $m$  是组成一个变更周期的系统帧数。一个变更周期包含  $m = \text{modificationPeriodCoeff} \cdot r13 * \text{defaultPagingCycle} \cdot r13$  个系统帧。这是通过 SIB2-NB 来设置的。

如果 RRC\_IDLE 态的 UE 配置的 DRX 周期长于系统信息变更周期，那么为了使能该 UE 的系统信息变更通知，定义了一个 eDRX 获取周期（eDRX acquisition period）。对于 NB-IoT 而言，eDRX 获取周期的边界是满足  $H-SFN \bmod 1024 = 0$  的超帧。

与 LTE 不同的是，在 NB-IoT 中，处于 RRC\_IDLE 态的 UE 配置的 eDRX 周期（见 7.4.2 节）可能大于变更周期，因此需要分别对 eDRX 周期大于变更周期和 eDRX 周期小于等于变更周期这两种情况进行处理。

UE 收到了一个系统信息变更通知后，如果 UE 没有配置一个长于系统信息变更周期的 DRX 周期，则 UE 会从下一个变更周期的开始处去接收新的系统信息。这与传统的 LTE 处理方式是一样的。

UE 收到了一个应用于 eDRX 的系统信息变更通知后，如果 RRC\_IDLE 态的 UE 配置了一个长于系统信息变更周期的 DRX 周期，则 UE 会从下一个 eDRX 获取周期的开始处去接收新的系统信息。

UE 在收到新的系统信息之前，会继续使用旧的系统信息。如果一个重要的参数发生改变，可能会对通信造成严重影响，但由于任何服务的中断都很短且不会频繁发生，所以认为结果是可以接受的。

对于 NB-IoT 而言，SIB1-NB 的可能变更的边界是满足  $(H-SFN * 1024 + SFN) \bmod 4096 = 0$  的系统帧。

小区有 2 种方式通知 UE 系统信息发生了变化：

- 使用 *Paging-NB* 消息或直接指示消息（Direct Indication information）指示 SI 消息是否发生了变化；
- MIB-NB 中包含了一个 *systemInfoValueTag-r13* 字段，每当 SI 消息发生变化时，该字段的值会加 1。与此同时，SIB1-NB 中还包含了 *systemInfoValueTagSI-r13* 字段，用于指示哪个 SI 消息发生了变化。

注：将 *systemInfoValueTag-r13* 放在 MIB-NB 中有助于 UE 快速检测到系统信息的变更。

小区可以使用 *Paging-NB* 消息来通知处于 RRC\_IDLE 态和 RRC\_CONNECTED 态的 UE 系统信息发生了变化。如果 UE “处于 RRC\_CONNECTED 态，或 UE 处于 RRC\_IDLE 态但没有配置一个长于系统信息变

更周期的 DRX 周期”，并且接收到的 *Paging-NB* 消息里包含了 *systemInfoModification-r13* 字段，则 UE 知道系统信息将在下一个变更周期边界处发生改变。如果 UE 处于 RRC\_IDLE 态并且配置了一个长于系统信息变更周期的 DRX 周期，并且在一个 eDRX 获取周期内收到了至少一个包含 *systemInfoModification-eDRX-r13* 的 *Paging-NB* 消息，则 UE 会在下一个 eDRX 获取周期边界处去接收更新后的系统信息。UE 只是被通知系统信息发生了变化，但并不知道哪些系统信息发生了变化，除非收到了 SIB1-NB 中的 *systemInfoValueTagSI-r13*。

直接指示信息（Direct Indication information）在使用 P-RNTI 加扰的 NPDCCH（DCI format N2）上传输，但该 NPDCCH 只用于指示系统信息发生了变化，此时不会再发送关联的 *Paging-NB* 消息。对应的 DCI format N2 的定义见 4.3.1.4 节的介绍。8 比特的 Direct Indication information 字段的解析见 36.331 的 Table 6.7.5-1。如果某一比特设置为 1，则 UE 按照 *Paging-NB* 的对应字段被设置来进行处理。

**Table 6.7.5-1: Direct Indication information**

Bit	Field in <i>Direct Indication information</i>
1	<i>systemInfoModification</i>
2	<i>systemInfoModification-eDRX</i>
3, 4, 5, 6, 7, 8	Not used, and shall be ignored by UE if received

直接指示信息可以看做是 *Paging-NB* 消息的简化版本。其好处在于，在 *Paging-NB* 消息仅用于指示系统信息发生了变化时，只需发送一个 NPDCCH（Direct Indication information）而不需要再占用 NPDSCH 的资源（因为不发送 *Paging-NB*）。在 *Paging-NB* 消息可能需要大量重复的情况下，使用 Direct Indication information 能有效地降低 UE 读取 *Paging-NB* 消息的开销。

NB-IoT UE 还可以基于 MIB-NB 中的 *systemInfoValueTag-r13* 字段来判断 SI 消息是否发生了变化。UE 可能在从覆盖外返回到覆盖内，或从较长 DRX 周期醒来时，使用该字段来判断保存的系统信息是否依然有效。在这些情况下，UE 会通过检查 *systemInfoValueTag-r13* 来判断系统信息是否发生变化。如果存在 SIB 更改，eNodeB 则会修改 *systemInfoValueTag-r13* 的值。另外，NB-IoT 中的系统消息的有效期为 24 个小时，即 NB-IoT UE 会认为从确认接收到的系统消息有效的那一刻算起的 24 个小时之后，所保存的系统信息是无效的。如果 UE 在覆盖外超过了 24 小时，那么无论如何，UE 都必须重新读取系统信息。如果处于 RRC\_CONNECTED 态的 NB-IoT UE 认为保存的系统信息失效了，UE 会继续使用保存的系统信息。

MIB-NB 中的 *systemInfoValueTag-r13* 只用于指示系统信息是否发生了变化，而不指示哪个系统信息发生了变化。对于 NB-IoT UE 而言，还可以进一步地通过 SIB1-NB 的 *systemInfoValueTagSI-r13* 字段来判断某个特定的 SI 消息是否发生的变化。如果 MIB-NB 中的 *systemInfoValueTag-r13* 字段的值与 UE 保存的值不同，并且 SIB1-NB 中用于某个特定 SI 消息的 *systemInfoValueTagSI-r13* 字段的值与保存的值不同，则 UE 会认为这个特定的 SI 消息是无效的，此时 UE 无需去重新接收那些没有发生变化的 SI 消息，以达到省电目的。如果只有 MIB-NB 中的 *systemInfoValueTag-r13* 字段的值与保存的值不同，则 NB-IoT UE 会认为除 SIB14-NB 外的保存的系统信息都是无效的。

SIB14-NB 中携带的 access barring 参数，在任意时间点都可能发生变化。SIB16-NB 会规律地改变它的内容。对于这 2 类 SIB-NB 的变更，不需要通过 MIB-NB 的 *systemInfoValueTag-r13* 或 *Paging-NB* 消息来指示系统信息发生了变化。同时，SIB1-NB 的 *hyperSFN-MSB* 字段发生变化也不需要发送变更通知。

对于处于 RRC\_CONNECTED 态的 NB-IoT UE 而言，并不要求去接收系统信息，因此必要的系统信息发生改变时，E-UTRAN 可以发起 RRC 连接释放流程，以便 UE 进入 RRC\_IDLE 态去接收更新后的系统信息。

如果 UE 没有配置一个长于系统信息变更周期的 DRX 周期，则 UE 会通过变更周期边界之后校验 MIB-NB 的 *systemInfoValueTag-r13* 字段；或在每个变更周期，如果 UE 在变更周期内没有收到 *Paging-NB*，则会尝试寻找 *systemInfoModification-13* 至少 *modificationPeriodCoeff* 次。如果 UE 在一个变更周期内没有收到 *Paging-NB* 消息，则 UE 认为在接下来的一个变更周期范围不会发生系统信息的改变。如果处于 RRC\_CONNECTED 态的 UE 在一个变更周期内收到了一个 *Paging-NB* 消息，则 UE 会通过是否存在 *systemInfoModification-13* 来推断接下来的一个变更周期内，系统信息是否会发生变化。

当处于 RRC\_IDLE 态的 UE 配置了一个长于系统信息变更周期的 DRX 周期，并且自从 UE 最后验证所保存的系统信息的有效性起已经过去了至少一个变更周期边界时，UE 在建立或恢复 RRC 连接之前，会通过检查 MIB-NB 的 *systemInfoValueTag-r13* 来校验保存的系统信息是否依然有效。

基于前面的介绍，在本节的最后，我们对 NB-IoT 和 LTE 的系统信息相关的主要差异进行了总结。如表 7-2 所示。

表 7-2: NB-IoT 和 LTE 的系统信息相关的主要差异

NB-IoT		LTE	
<b>MIB-NB</b>	<ul style="list-style-type: none"> <li>周期固定为 640ms</li> </ul>	<b>MIB</b>	<ul style="list-style-type: none"> <li>周期固定为 40ms</li> </ul>
<b>SIB1-NB</b>	<ul style="list-style-type: none"> <li>周期固定为 2560ms;</li> <li>获取 SIB1-NB 所需的所有调度信息由 MIB-NB 提供，无需 DCI 指示</li> </ul>	<b>SIB1</b>	<ul style="list-style-type: none"> <li>周期固定为 80ms;</li> <li>SIB1 的时域位置固定，频域位置以及其它调度信息（MCS/TBS 等）由 SI-RNTI 加扰的 DCI 来指示</li> </ul>
<b>SIB-NB (除 SIB1-NB 外)</b>	<ul style="list-style-type: none"> <li>获取 SIB-NB（SI 消息）所需的所有调度信息由 SIB1-NB 提供，无需 DCI 指示</li> </ul>	<b>SIB（除 SIB1 外）</b>	<ul style="list-style-type: none"> <li>时域位置的信息（周期、SI 窗口）由 SIB1 提供，频域位置以及其它调度信息（MCS/TBS 等）由 SI-RNTI 加扰的 DCI 来指示</li> </ul>
<b>其它</b>	<ol style="list-style-type: none"> <li>NB-IoT 中，BCCH 和其它的逻辑信道不能在同一个子帧上发送（针对同一载波）。但在 LTE 中可以；</li> <li>NB-IoT 中，RRC_CONNECTED 态的 UE 并不要求会去监听系统信息是否发生了变化，E-UTRAN 需要通过触发 RRC 连接释放流程使 UE 进入 RRC_IDLE 态，以便其检测并接收更新后的系统信息。但在 LTE 中，无论 UE 是处于 RRC_IDLE 态，还是处于 RRC_CONNECTED 态，都可以检测并接收更新后的系统信息。</li> <li>NB-IoT 中系统信息的有效时间为 24 小时；而 LTE 中为 3 小时。</li> </ol>		

### 7.3 随机接入过程

在 NB-IoT 中，随机接入过程的主要作用与 LTE 类似：（1）获取上行同步；（2）与网络建立起一个无线连接，并为 UE 分配一个唯一的标识 C-RNTI；（3）发送上行调度请求。

NB-IoT 中，随机接入过程通常由以下 5 类事件之一触发：

1. 初始接入时建立无线连接：UE 会从 RRC\_IDLE 态到 RRC\_CONNECTED 态；
2. RRC 连接重建立流程（RRC Connection Re-establishment procedure）：以便 UE 在无线链路失败（Radio Link Failure）后重建无线连接。对于只支持控制面 CIoT EPS 优化的 UE 而言，不支持 RRC 连接重建立流程，因此也不支持此类事件触发的随机接入过程；
3. RRC\_CONNECTED 态下，下行数据到达，但 UE 的上行处于“不同步”状态，此时 eNodeB 会通过发送一个 PDCCH order 来触发 UE 的随机接入过程；
4. RRC\_CONNECTED 态下，上行数据到达，但 UE 的上行处于“不同步”状态；
5. RRC\_CONNECTED 态下，为了定位 UE，需要 timing advance。（注：在 Rel-13 中，NB-IoT 不支持定位功能。该事件触发的随机接入过程会在 Rel-14 中定义）

NB-IoT 不支持切换（handover），因此不支持切换触发的随机接入过程。

在 3.2.3.2 节介绍过，NB-IoT 中不存在类似 LTE 的 PUCCH 信道，也不支持类似 LTE 的 SR 消息。当 NB-IoT UE 有新的上行数据需要发送且没有被分配上行资源时，UE 会使用随机接入过程来替代 SR 的作用，向 eNodeB 请求上行资源。

在 Rel-13 中，NB-IoT 只要求支持基于竞争的随机接入。虽然在 7.3.2.1 节中会介绍用于基于非竞争的随机接入的 NPRACH 资源，但这部分资源只是为了满足未来可能存在的扩展而定义的，并不要求一定要使用这部分资源。在本章中，如无特殊说明，我们将按 NB-IoT 只支持基于竞争的随机接入过程来介绍相关流程。

### 7.3.1 Preamble

随机接入过程的步骤一就是 UE 发送 random access preamble。Preamble 的主要作用是告诉 eNodeB 有一个随机接入请求，并使得 eNodeB 能估计其与 UE 之间的传输时延，以便 eNodeB 校准 uplink timing 并将校准信息通过 timing advance command 告知 UE。在 NB-IoT 中，UE 通过发送 preamble 还可以告诉 eNodeB 自己所处的 CE 等级及其 Msg3 是否支持 multi-tone 传输。

preamble 序列基于与子载波位置相关的 Zadoff-Chu 序列生成，载波频率上的调制和变频的方式与 LTE 类似，具体见 36.211 的 10.1.6.2 节的介绍。

NB-IoT 中的 preamble 基于单子载波跳频符号组（single-subcarrier frequency-hopping symbol groups）。每个符号组（symbol group）由一个长为  $T_{CP}$  的循环前缀和 5 个长度相等的符号组成，这 5 个符号的总长度为  $T_{SEQ}$ ，且这 5 个符号上发送的信号是完全相同的。如图 7-6 所示。

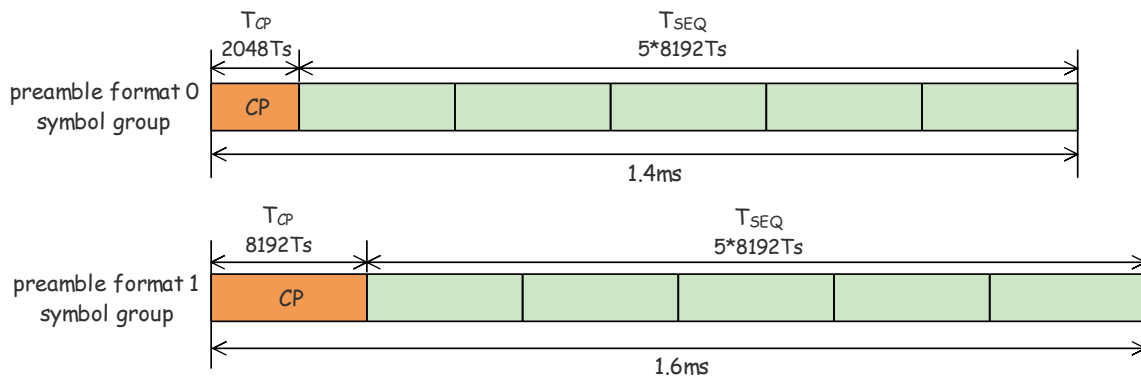


图 7-6: Random Access symbol group

NB-IoT 中定义了 2 种 preamble 格式: format 0 和 format 1, 它们的不同之处在于 CP 的长度, 如 36.211 的 Table 10.1.6.1-1 所示。具体使用哪种格式是通过 *nprach-CP-Length-r13* 配置的。不同的 preamble 格式分别对应不同的最大小区覆盖半径, preamble format 0 的最大覆盖半径约为 10km, preamble format 1 的最大覆盖半径大约为 40km。

Table 10.6.1.1-1: Random access preamble parameters

Preamble format	$T_{CP}$	$T_{SEQ}$
0 (1.4ms)	$2048T_s$ (0.0667ms)	$5 \cdot 8192T_s$ (1.3333ms)
1 (1.6ms)	$8192T_s$ (0.2667ms)	$5 \cdot 8192T_s$ (1.3333ms)

一个 preamble 由没有间隙的 4 个符号组 (symbol group) 组成, 并会重复传输  $N_{rep}^{NPRACH}$  次。如图 7-7 所示。

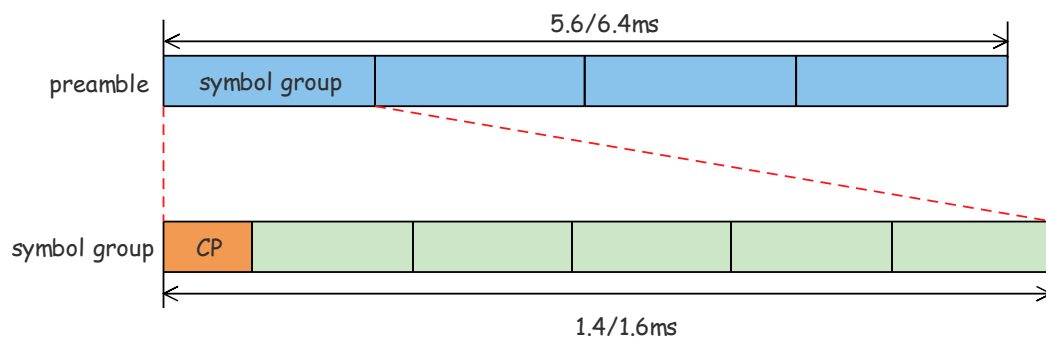


图 7-7: preamble 的定义

## 7.3.2 NPRACH

### 7.3.2.1 NPRACH 资源

NPRACH 用于传输 preamble。如果 MAC 层触发了 preamble 的发送, 则该 preamble 只能在特定的时频资源上发送。首先我们来介绍小区级的 NPRACH 资源。

小区级的 NPRACH 配置会通过 SIB2-NB 广播给所有的 UE。对于 NB-IoT 而言，一个小区支持的 NPRACH 资源集合通过 *nprach-ParametersList-r13* 配置。*nprach-ParametersList-r13* 列表中的元素个数就等于 NPRACH 资源的个数。一个 NPRACH 资源对应一个 CE 等级。CE 等级的个数等于 *rsrp-ThresholdsPrachInfoList-r13* 中定义的 RSRP 阈值数加一。每个 CE 等级在 *nprach-ParametersList-r13* 中有一个对应的 NPRACH 资源。CE 等级的编号从 0 开始，NPRACH 资源是根据 *numRepetitionsPerPreambleAttempt-r13* 递增的顺序与 CE 等级映射的。这是因为 CE 等级的编号越小，说明其覆盖范围内的信道条件越好，此时所需的 preamble 重复次数也就越少。

注意：NB-IoT 中的一个 NPRACH 资源实际上是包含多个子载波的集合。UE 在发起随机接入时，只会选择该集合中的其中一个子载波来发送 preamble。

每个 NPRACH 资源是通过对应的 *NPRACH-Parameters-NB-r13* 来配置的。

NPRACH-Parameters-NB-r13::=		SEQUENCE {
nprach-Periodicity-r13	ENUMERATED {ms40, ms80, ms160, ms240, ms320, ms640, ms1280, ms2560},	----NPRACH资源的周期
$N_{\text{period}}^{\text{NPRACH}}$		
nprach-StartTime-r13	ENUMERATED {ms8, ms16, ms32, ms64, ms128, ms256, ms512, ms1024},	----NPRACH资源在周期内的起始时间
$N_{\text{start}}^{\text{NPRACH}}$		
nprach-SubcarrierOffset-r13	ENUMERATED {n0, n12, n24, n36, n48, n60, n72, n84, n96, n108, n120, n132, n144, n156, n168, n180, n192, n204, n216, n228, n240, n252, n264, n276, n288, n300, n312, n324, n336, n348, n360, n372, n384, n396, n408, n420, n432, n444, n456, n468, n480, n492, n504, n516, n528, n540, n552, n564, n576, n588, n600, n612, n624, n636, n648, n660, n672, n684, n696, n708, n720, n732, n744, n756, n768, n780, n792, n804, n816, n828, n840, n852, n864, n876, n888, n900, n912, n924, n936, n948, n960, n972, n984, n996, spare1},	----分配给NPRACH资源的第一个子载波的频域位置
$N_{\text{scoffset}}^{\text{NPRACH}}$		
nprach-NumSubcarriers-r13	ENUMERATED {n12, n24, n36, n48},	----分配给NPRACH资源的子载波数
$N_{\text{sc}}^{\text{NPRACH}}$		
nprach-SubcarrierMSG3-RangeStart-r13	ENUMERATED {zero, oneThird, twoThird, one},	---- $N_{\text{MSG3}}^{\text{NPRACH}}$ ，用于计算NPRACH子载波范围内的哪些子载波索引是用于指示UE支持multi-tone Msg3传输的
maxNumPreambleAttemptCE-r13	ENUMERATED {n3, n4, n5, n6, n7, n8, n10, spare1},	----每个NPRACH资源（CE等级）对应的preamble最大传输尝试次数
numRepetitionsPerPreambleAttempt-r13	ENUMERATED {n1, n2, n4, n8, n16, n32, n64, n128},	----每个NPRACH资源（CE等级）对应的每次preamble传输尝试的重复次数
$N_{\text{rep}}^{\text{NPRACH}}$		
npdcch-NumRepetitions-RA-r13	ENUMERATED {r1, r2, r4, r8, r16, r32, r64, r128, r256, r512, r1024, r2048, spare4, spare3, spare2, spare1},	----指定了用于RAR、Msg3重传以及Msg4的NPDCCH公共搜索空间的最大重复次数。具体见4.3.1.2节的介绍
npdcch-StartSF-CSS-RA-r13	ENUMERATED {v1dot5, v2, v4, v8, v16, v32, v48, v64},	----对应Type-2公共搜索空间的G值。具体见4.3.1.2节的介绍
npdcch-Offset-RA-r13	ENUMERATED {zero, oneEighth, oneFourth, threeEighth},	----对应Type-2公共搜索空间的 $\alpha_{\text{offset}}$ 值。具体见4.3.1.2节的介绍

```

}

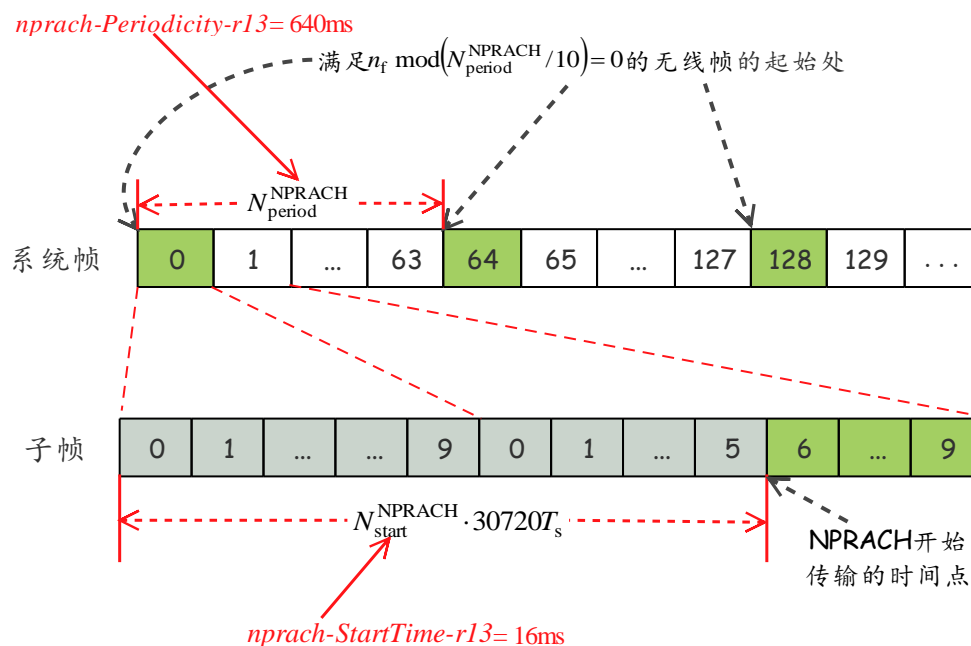
NPRACH-Parameters-NB-v1330 ::= SEQUENCE {
    nprach-NumCBRA-StartSubcarriers-r13 ENUMERATED {n8, n10, n11, n12, n20, n22, n23, n24,
                                                    n32, n34, n35, n36, n40, n44, n46, n48} ----分配给基于竞争的随
机接入的子载波个数  $N_{sc\_cont}^{NPRACH}$ 
}

```

一个 NPRACH 资源在时域上的配置包括：

- NPRACH 资源的周期  $N_{period}^{NPRACH}$ ：通过 *nprach-Periodicity-r13* 配置；
- NPRACH 资源在周期内的起始时间  $N_{start}^{NPRACH}$ ：通过 *nprach-StartTime-r13* 配置；
- 每次接入尝试的 NPRACH 重复次数  $N_{rep}^{NPRACH}$ ：通过 *numRepetitionsPerPreambleAttempt-r13* 配置；

一个 NPRACH 资源只能在满足  $n_f \bmod (N_{period}^{NPRACH}/10) = 0$  的系统帧开始处算起， $N_{start}^{NPRACH} \cdot 30720T_s$  个时间单元（即 *nprach-StartTime-r13* ms）之后开始传输。为 NPRACH 添加周期配置的目的是为了让相邻小区之间的 NPRACH 资源能够时分复用（TDM），以避免小区间的干扰（inter-cell interference）。举个例子，相邻的小区可以使用相同的 NPRACH 周期  $N_{period}^{NPRACH}$ ，但不同的起始时间  $N_{start}^{NPRACH}$ ，从而保证彼此间的 NPRACH 资源在时域上复用。图 7-8 是 NPRACH 资源在时域上的起始位置的一个例子，并假设  $N_{period}^{NPRACH} = 640\text{ms}$ ， $N_{start}^{NPRACH} = 16\text{ms}$ 。



在传输了  $4 \cdot 64(T_{CP} + T_{SEQ})$  个时间单元后，会插入一个长为  $40 \cdot 30720T_s$  时间单元的 UL gap，这么做的目的可见 5.3.3 节的介绍。（注： $30720 \cdot T_s$  正好对应 1ms 的长度）



- $N_{MSG3}^{NPRACH}$ ：通过 *nprach-SubcarrierMSG3-RangeStart-r13* 配置，并用于计算 NPRACH 子载波范围内的哪些子载波索引是用于指示 UE 支持 multi-tone Msg3 传输的。

一个 NPRACH 资源在频域上的划分见图 7-9 所示。

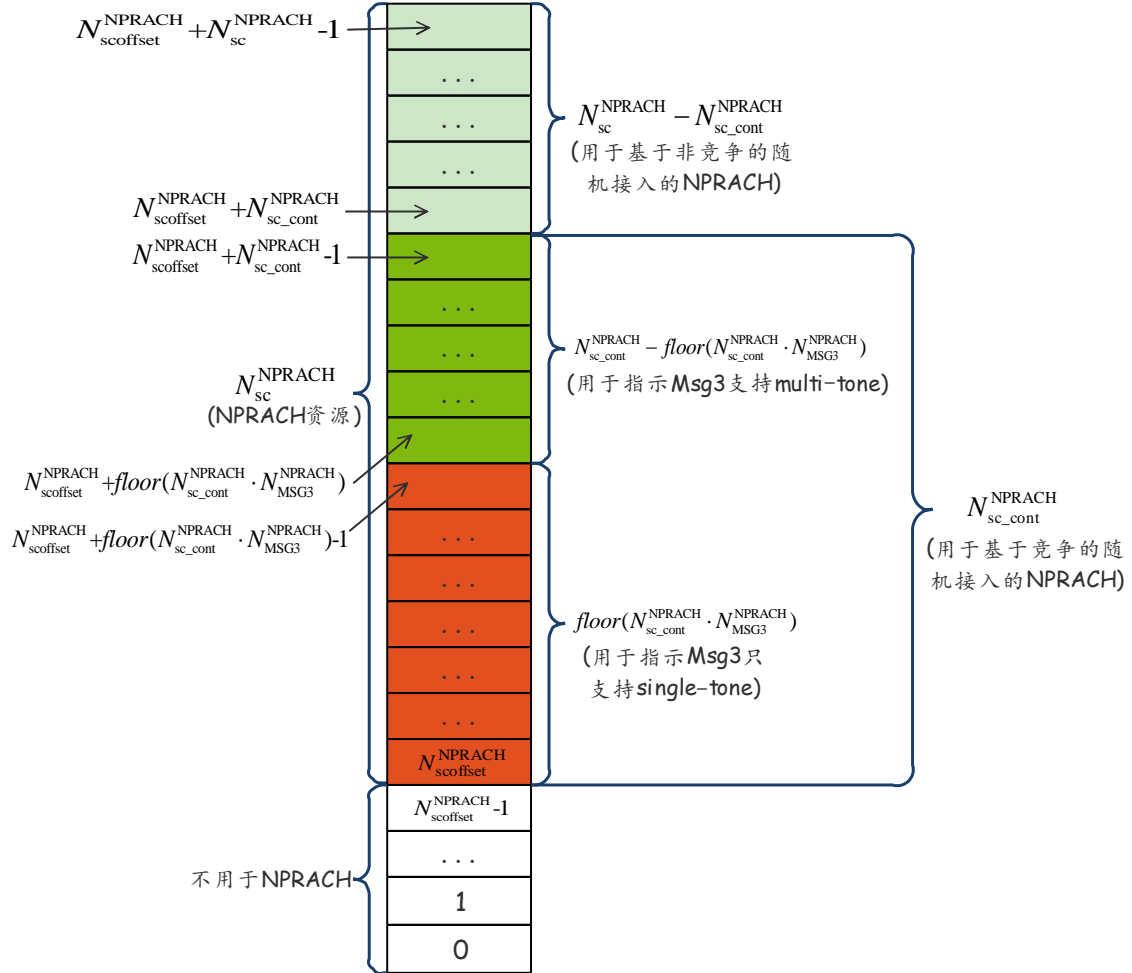


图 7-9: 一个 NPRACH 资源在频域上的划分

在频域上，NPRACH 使用 3.75kHz 的子载波间距，对应在频域上存在 48 个子载波（对应  $N_{sc}^{UL}=48$ ）。一个 NPRACH 资源在频域上可占据连续的 12/24/36/48 个子载波（对应  $N_{sc}^{NPRACH}$ ），并通过 *nprach-NumSubcarriers-r13* 配置。该 NPRACH 资源在频域上的起始子载波（对应  $N_{sc\_offset}^{NPRACH}$ ）通过 *nprach-SubcarrierOffset-R13* 配置，其取值范围为 {0, 12, 24, 36, 2, 18, 34}。基于一个 NPRACH 资源在频域上可占用的子载波数，大家很容易理解分配给 NPRACH 的起始子载波的频域位置取值为 {0, 12, 24, 36}。而其另一个取值集合 {2, 18, 34} 是为了在 NPRACH 和 NPUSCH 相邻时，保证 NPRACH 和 NPUSCH 之间保留 7.5kHz 的保护频带。

如果  $N_{\text{scoffset}}^{\text{NPRACH}} + N_{\text{sc}}^{\text{NPRACH}} > N_{\text{sc}}^{\text{UL}}$ ，则该 NPRACH 配置是无效的。也就是说，分配给 NPRACH 资源的起始子载波的位置加上分配给该 NPRACH 资源的子载波数，不能超过上行的子载波数，即不能超过频域资源的上边界。

虽然在 Rel-13 中，并不要求 NB-IoT 支持基于非竞争的随机接入。但为了后续版本可能存在的扩展需求，36.331 在 v13.3.0 版本中引入了 IE: *nprach-NumCBRA-StartSubcarriers-r13*（对应  $N_{\text{sc\_cont}}^{\text{NPRACH}}$ ），以将前面介绍的 NPRACH 资源进一步地划分为 2 个集合：其中一个集合中的子载波用于基于竞争的随机接入，另一个集合中的子载波用于基于非竞争的随机接入。 $N_{\text{sc\_cont}}^{\text{NPRACH}}$  指定了分配给基于竞争的随机接入的子载波数，对应用于竞争的随机接入的子载波索引的取值范围为： $N_{\text{scoffset}}^{\text{NPRACH}} + [0, N_{\text{sc\_cont}}^{\text{NPRACH}} - 1]$ 。隐式地，用于非竞争的随机接入的子载波索引的取值范围为： $N_{\text{scoffset}}^{\text{NPRACH}} + [N_{\text{sc\_cont}}^{\text{NPRACH}}, N_{\text{sc}}^{\text{NPRACH}} - 1]$ 。如果将  $N_{\text{sc\_cont}}^{\text{NPRACH}}$  的值设置成等于  $N_{\text{sc}}^{\text{NPRACH}}$  的值，则对应 NPRACH 资源内的所有子载波都用于基于竞争的随机接入。

在早期的 NB-IoT 现场试验和部署中，某些 UE 可能不支持 multi-tone 传输。为了更好地调度上行传输，eNodeB 需要知道 UE 是否具有支持 multi-tone 传输的能力。因此，UE 会在随机接入的步骤一中（发送 preamble 时）指示其是否支持 Msg3 的 multi-tone 传输，以便 eNodeB 确定在 RAR 中如何调度用于 Msg3 的上行资源。为此，eNodeB 可以将频域上的 NPRACH 子载波（更确切地说，是用于基于竞争的随机接入的子载波集合）划分为两个非重叠的集合：一个指示 UE 不支持 multi-tone Msg3 传输，另一个指示 UE 支持 multi-tone Msg3 传输。UE 通过选择两个集合中的其中一个来发送 preamble，以向 eNodeB 指示其是否支持 multi-tone Msg3 传输。

一个 NPRACH 资源内用于基于竞争的随机接入的子载波集合可通过 *nprach-SubcarrierMSG3-RangeStart-r13*（对应  $N_{\text{MSG3}}^{\text{NPRACH}}$ ）进一步被分成 1 或 2 组，以指示 Msg3 是否支持 multi-tone 传输。确切地说，分配给

基于竞争的随机接入的 NPRACH 起始子载波可被分成 2 个子载波集合： $\{0, 1, \dots, N_{\text{sc\_cont}}^{\text{NPRACH}} N_{\text{MSG3}}^{\text{NPRACH}} - 1\}$  和

$\{N_{\text{sc\_cont}}^{\text{NPRACH}} N_{\text{MSG3}}^{\text{NPRACH}}, \dots, N_{\text{sc\_cont}}^{\text{NPRACH}} - 1\}$ 。如果 UE 选择前一个子载波集合中的 NPRACH 来发送 preamble，

则说明该 UE 的 Msg3 只能使用 single-tone 传输；如果 UE 选择后一个子载波集合中的 NPRACH 来发送 preamble，则说明该 UE 的 Msg3 支持 multi-tone 传输。eNodeB 会根据接收到的 preamble 属于哪一个集合，来决定是基于 single-tone 还是 multi-tone 来给 UE 分配 Msg3 资源。

注意：这里介绍的是传输 Msg3 时是否支持 multi-tone 传输。这与 Msg3 消息本身携带的指示 UE 是否支持 multi-tone 传输的能力是不同的。

如果 *nprach-SubcarrierMSG3-RangeStart-r13* 配置为 0，则对应的 NPRACH 资源均用于指示 UE 支持 Msg3 的 multi-tone 传输；如果 *nprach-SubcarrierMSG3-RangeStart-r13* 配置为 1，则对应的 NPRACH 资源均用于指示 UE 只支持 Msg3 的 single-tone 传输。

如果  $nprach\text{-}SubcarrierMSG3\text{-}RangeStart\text{-}r13$  的值等于 {oneThird} 或 {twoThird}，则 2 部分的起始子载波索引分别为：

- 用于 single-tone Msg3 NPRACH 的部分：
$$N_{scoffset}^{NPRACH} + \left[ 0, \text{floor}(N_{sc\_cont}^{NPRACH} \cdot N_{MSG3}^{NPRACH}) - 1 \right]$$
- 用于 multi-tone Msg3 NPRACH 的部分：
$$N_{scoffset}^{NPRACH} + \left[ \text{floor}(N_{sc\_cont}^{NPRACH} \cdot N_{MSG3}^{NPRACH}), N_{sc\_cont}^{NPRACH} - 1 \right]$$

至少有一个具有除 {32,64,128} 之外的 NPRACH 重复次数的 NPRACH 资源，其  $nprach\text{-}SubcarrierMSG3\text{-}RangeStart\text{-}r13$  的值不为 0，即一个小区必须预留部分资源用于指示 single-tone Msg3 传输。

multi-tone 传输主要用于正常覆盖范围内的 UE，而正常覆盖范围内的 UE 在传输数据时，重复次数不会太多。因此 Msg3 使用 multi-tone 传输时，说明 UE 所属 CE 等级的信道条件较好，而 CE 等级的信道条件较好时，所需的 NPRACH 重复次数也不需要太多，而协议中将其限制在范围 {1, 2, 4, 8, 16} 内。这也是协议 (36.331) 中规定，当某个 CE 等级对应的 NPRACH 重复次数为 {32, 64, 128} 时，该 CE 等级上的 Msg3 不支持 multi-tone 传输的原因。也就是说，如果某个 CE 等级对应的 NPRACH 资源里的  $N_{rep}^{NPRACH}$  值配置成 {32, 64, 128} 中的一个，且 UE 使用该 NPRACH 资源发送 preamble，那么 eNodeB 在调度 Msg3 时，只会使用 single-tone 传输。

### 7.3.2.2 NPRACH 上的 preamble 传输

接下来我们将介绍 UE 如何在 NPRACH 上发送 preamble。

Preamble 在使用 NPRACH 进行传输时，只能使用 single-tone 传输，即在频域上只占用一个子载波。并且该传输支持跳频。使用 single-tone 而不是 multi-tone 来传输 preamble 能提供更高的功率谱密度，并提高 NPRACH 的覆盖范围。

NPRACH 跳频的粒度为一个符号组，即组成一个 preamble 的 4 个符号组在不同的子载波上传输，并且跳频会被限制在一个连续的 12 ( $N_{sc}^{RA} = 12$ ) 个子载波的集合内。第  $i$  个符号组的频域位置  $n_{sc}^{RA}(i)$  的计算公式如图 7-10 所示（跳频的方式也体现在该公式中）。

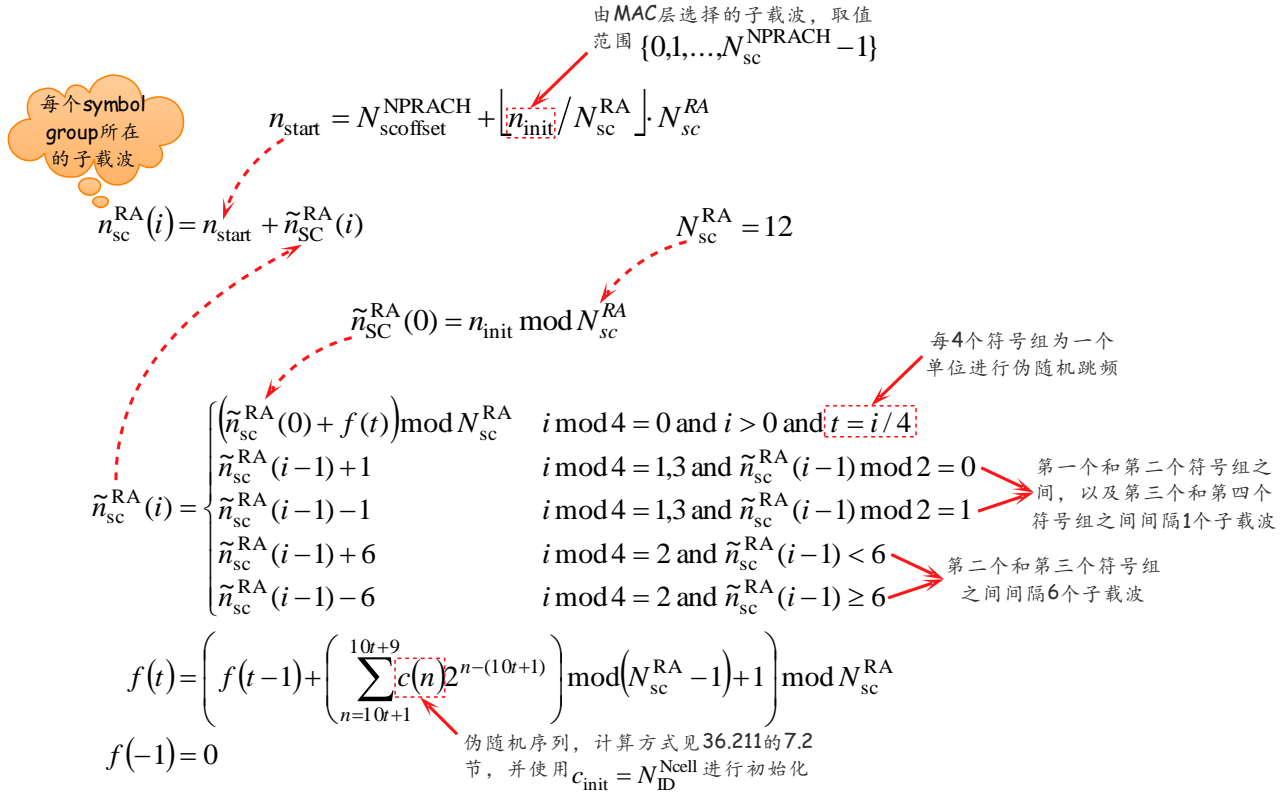


图 7-10: NPRACH 的第  $i$  个符号组的频域位置

由于跳频被限制在一个连续的 12 个子载波的集合内，因此有 12 种可能性。如果 eNodeB 没有通过 PDCCH order 指示用于第一个 preamble 符号组的子载波，则 UE 会自己选择一个子载波，接下来的 3 个符号组使用的子载波的位置取决于第一个符号组的子载波位置。对于下一次重复的第一个符号组的子载波选择，会使用  $N_{ID}^{Ncell}$  和重复的编号（即  $t$ ）用作输入来进行伪随机跳频，且该重复接下来的符号组的子载波选择再次取决于该重复的第一个符号组的结果。

这种跳频算法设计使得只要给 UE 分配不同的起始子载波，就会带来不重叠的跳频样式。因此，分配给 NPRACH 资源的子载波数，就等于可用的 preamble 数。每个子载波位置就对应一个 preamble，因此在 NB-IoT 中，不存在类似于 LTE 中的 preamble index 概念。

图 7-11 是一次 preamble 尝试重复 4 次的跳频举例。这里，每一个蓝色的矩形表示一个 preamble 符号组，因此一个 preamble 重复包含了 4 个矩形。

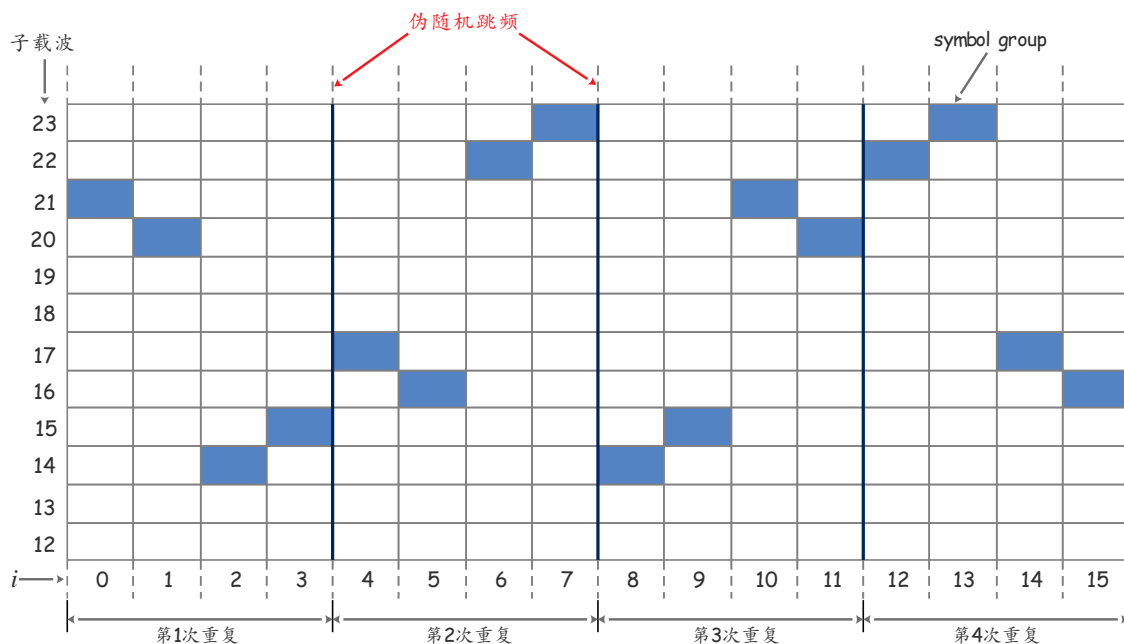


图 7-11: preamble 在频域上的传输举例（重复 4 次）

NPRACH 的设计使其有非常低的 PAPR (Peak-to-Average Power Ratio)，并因此大幅度地降低了对功率放大器补偿的需求。与 GSM/GPRS 相比，NPRACH 提升了超过 20dB 的随机接入覆盖。

### 7.3.3 随机接入过程

无论 UE 是否配置了 non-anchor carrier，整个随机接入过程（包括 Msg3 和 Msg4）都是在 anchor carrier 上进行的。NB-IoT 中的随机接入过程如图 7-12 所示。

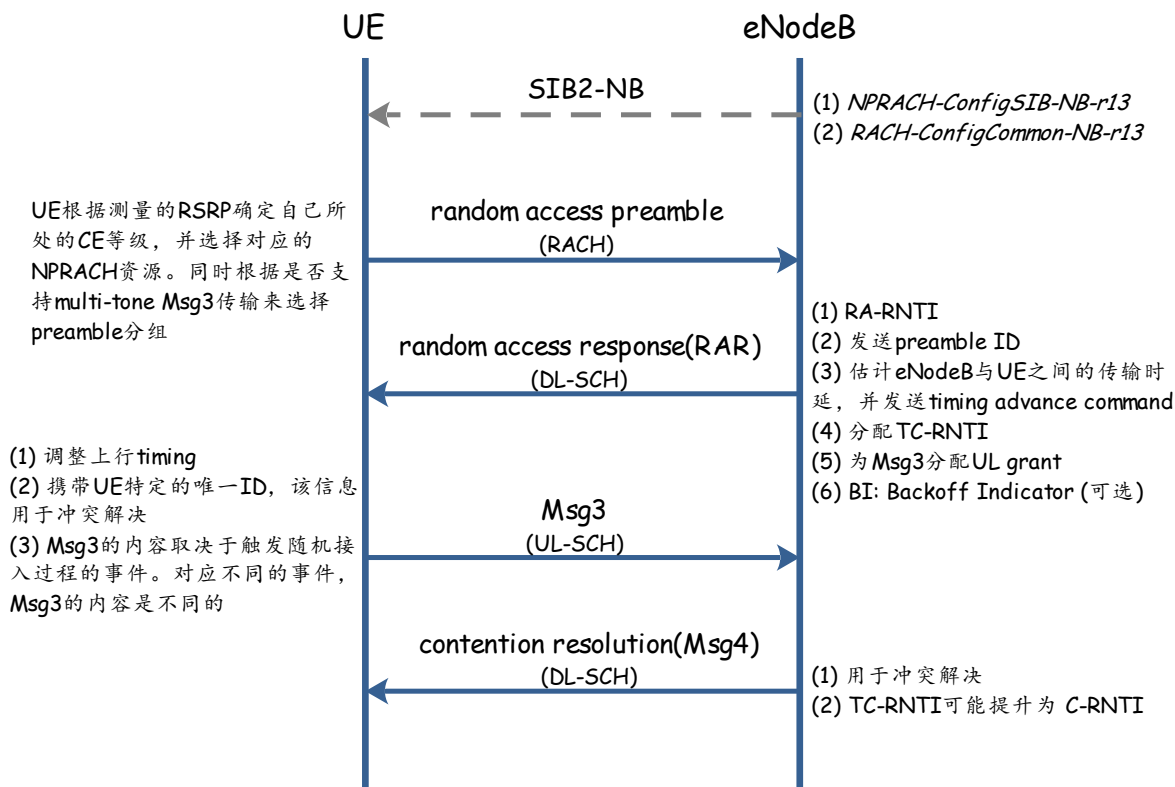


图 7-12: 基于竞争的随机接入

在 Rel-13 中，NB-IoT 只要求使用基于竞争的随机接入。并且与 LTE 一样，该过程包含 4 个步骤：

**步骤一：** UE 发送随机接入 preamble；

**步骤二：** eNodeB 发送一个 RAR。该 RAR 中包含了 timing advance command 和用于步骤三的上行资源调度信息；

**步骤三：** UE 使用步骤二指示的上行资源来将其唯一的 ID 信息发给 eNodeB；

**步骤四：** eNodeB 发送一个冲突解决消息，来解决由于多个 UE 发送相同的 preamble 而导致的冲突。

在发起随机接入之前，UE 必须接收 SIB2-NB 以获取随机接入相关的配置信息。

### 7.3.3.1 步骤一：UE 发送 preamble

小区会定义好 CE 等级个数以及每个 CE 等级所属的 RSRP 阈值范围，同时定义好每个 CE 等级对应的 NPRACH 资源，并通过 SIB2-NB 告诉 UE。UE 通过测量下行链路信号的接收功率得到 RSRP 值，并以此确定它所在的 CE 等级，同时 UE 在此 CE 等级对应的 NPRACH 资源上发送 preamble，也即通过所选的 NPRACH 资源来告诉小区该 UE 本身所在的 CE 等级。与此同时，一个 CE 等级的 NPRACH 资源可能分成 2 组，UE 通过选择其中一组里的来指示自己的 Msg3 是否支持 multi-tone 传输。小区接收到 preamble 后，就知道了 UE 所在的 CE 等级以及 UE 的 Msg3 是否支持 multi-tone 传输，并基于这些信息来调度该 UE（确定重复次数、以及 Msg3 使用的子载波数等）。UE 发送 preamble 的整体步骤如图 7-13 所示：

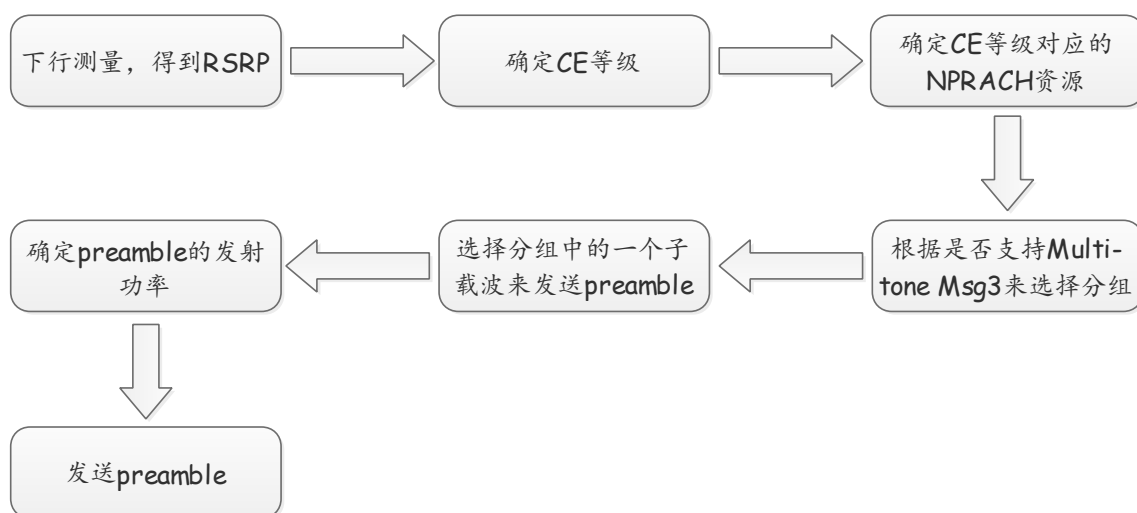


图 7-13: UE 发送 preamble 的处理步骤

#### 第一步：确定 UE 所属的 CE 等级

eNodeB 通过 SIB2-NB 的 *radioResourceConfigCommon-r13* -> *nprach-Config-r13* -> *rsrp-ThresholdsPrachInfoList-r13* 来广播小区支持的 CE 等级个数以及每个 CE 等级的阈值。

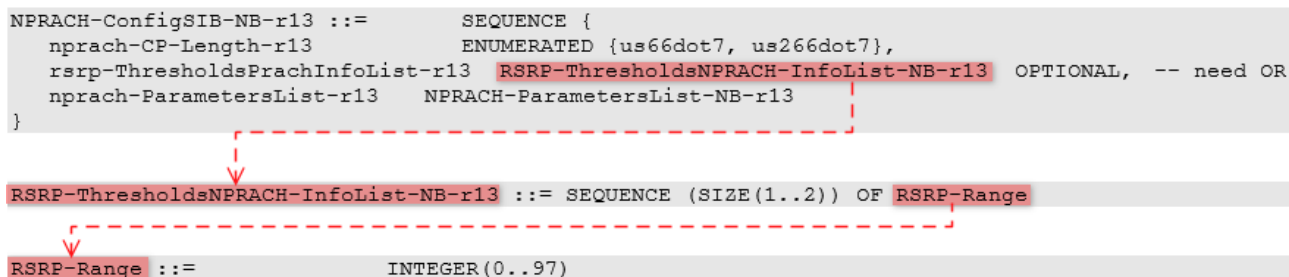


图7-14: CE等级个数以及每个CE等级的阈值

CE 等级的个数等于 *rsrp-ThresholdsPrachInfoList-r13* 中定义的 RSRP 阈值个数加一:

- (1) 如果不存在 *rsrp-ThresholdsPrachInfoList-r13*, 则小区只定义了 1 个 CE 等级, 此时无需定义相关阈值。
- (2) 如果 *rsrp-ThresholdsPrachInfoList-r13* 列表包含的元素个数为 1, 即定义了一个 RSRP 阈值, 则小区只定义了 2 个 CE 等级: RSRP 小于该阈值对应 CE 等级 1, RSRP 大于该阈值对应 CE 等级 0;
- (3) 如果 *rsrp-ThresholdsPrachInfoList-r13* 列表包含的元素个数为 2, 即从大到小顺序定义了 2 个 RSRP 阈值 T1 和 T2, 则小区定义了 3 个 CE 等级: RSRP 小于阈值 T2 对应 CE 等级 2, RSRP 大于等于 T2 但小于 T1 对应 CE 等级 1, RSRP 大于等于 T1 对应 CE 等级 0。

如果随机接入过程不是由 PDCCH order 触发, 则 UE 会通过测量下行链路信号的接收功率得到 RSRP 值, 并与上面给出的阈值比较, 从而确定自己所在的 CE 等级。

如果随机接入过程是由 PDCCH order 触发, 则 PDCCH order 中会携带 NPRACH 的重复次数。与该重复次数相等的 *numRepetitionsPerPreambleAttempt-r13* 对应的 NPRACH 资源对应的 CE 等级, 就是 UE 所处的 CE 等级, 且该 NPRACH 资源即为发送 preamble 时使用的资源。也就是说, 此时并不基于测量的 RSRP 来确定 UE 所在的 CE 等级。

## 第二步: 确定所属 CE 等级对应的 NPRACH 资源

每个 CE 等级有一个对应的 NPRACH 资源。确定了 CE 等级后, UE 也就确定了应该使用哪个 NPRACH 资源来发送 preamble。具体可见 7.3.2.1 节的介绍。

## 第三步: UE 根据 Msg3 是否支持 multi-tone 传输来选择子载波集合

接下来, UE 根据自己是否支持 Msg3 的 multi-tone 传输, 以及第二步得到的 NPRACH 资源中是否包含了用于指示 multi-tone Msg3 传输的子载波集合, 来确定是选择 single-tone Msg3 传输对应的子载波集合, 还是选择 multi-tone Msg3 传输对应的子载波集合来发送 preamble, 即确定 NPRACH 资源的分组。该分组中的每一个子载波就对应一个 preamble。

当然，如果对应的 NPRACH 资源中不存在对应 multi-tone Msg3 传输的分组，则只会选择 single-tone Msg3 分组中的子载波。

至此，UE 已经选定了 NPRACH 资源以及该资源内的一个分组，但 UE 发送的 preamble 只占用一个子载波，因此 UE 还需要从该分组中选择一个子载波来发送 preamble。

#### 第四步：选择一个子载波发送 preamble

前面已经介绍过，一个子载波对应一个 preamble，因此选择 preamble 的过程就是选择子载波的过程。这里分为 2 种情况：（1）PDCCH order 触发的随机接入过程，此时用于发送 preamble 的子载波可能由 eNodeB 指定；（2）UE 的 MAC 层自己选择一个子载波来发送 preamble。

对于 **PDCCH order 触发的随机接入过程**，eNodeB 会通过 PDCCH order 的 Subcarrier indication of NPRACH 字段显式地给 UE 设置用于发送 preamble 的子载波索引值  $I_{sc}$ ，且该值在 36.321 中被称为 *ra-PreambleIndex*。如果 *ra-PreambleIndex* 的值不等于 000000，则 preamble（即子载波的位置）会被设置为

$N_{scoffset}^{NPRACH} + I_{sc} \% N_{sc}^{NPRACH}$ ，其中  $N_{scoffset}^{NPRACH}$  和  $N_{sc}^{NPRACH}$  的定义见 7.3.2 节的介绍；如果 *ra-PreambleIndex* 的值等于 000000，则 UE 会根据第一步和第二步中确定的 NPRACH 资源以及该 UE 是否支持 multi-tone Msg3 传输来选择子载波集合（分组），并从该集合中随机选择一个子载波来发送 preamble。当然，如果该 NPRACH 资源中不存在对应 multi-tone Msg3 的分组，则只会选择 single-tone Msg3 分组中的子载波。

需要说明的是，虽然 PDCCH order 明确地指定了 preamble 传输使用的 NPRACH 资源和子载波，但是依然进行的是基于竞争的随机接入过程。（关于 PDCCH order，见 7.3.4 节的介绍）

对于 **UE 自己选择 preamble** 的情况，当 Msg3 还未发送时，UE 会根据前面的介绍，确定所选的 CE 等级、所选的 NPRACH 资源以及根据是否支持 multi-tone Msg3 传输来选择子载波集合（分组）；当 Msg3 正在被重传时，UE 会选择与 Msg3 初传相对应的 preamble 传输相同的子载波集合（分组）。接下来，UE 会从所选的子载波集合中随机选择一个子载波来发送 preamble。

UE 会根据物理层的 timing 要求以及对应 CE 级别 NPRACH 资源所占的子帧，来确定下一个可用的 NPRACH 子帧。

#### 第五步：确定 preamble 的发射功率

对于 CE 等级 0 而言，preamble 的发射功率  $P_{NPRACH}$  会设置为  $P_{NPRACH} = \min\{P_{CMAX,c}(i), \text{PREAMBLE\_RECEIVED\_TARGET\_POWER} + PL_c\}$  [dBm]；对于 CE 等级 1 或 2 而言，preamble 的发射功率  $P_{NPRACH}$  会设置为  $P_{CMAX,c}(i)$ 。其中  $P_{CMAX,c}(i)$  是在 36.101 的 6.2.5F 节中定义的服务小区  $c$  的 NB-IoT



上行子帧  $i$  上配置的 UE 最大发射功率。 $PL_c$  是 UE 针对服务小区  $c$  计算的下行路径损耗估计值，以 dB 为单位。（见 36.213 的 16.3.1 节）

对于 CE 等级 0 而言，PREAMBLE\_RECEIVED\_TARGET\_POWER（preamble 的目标接收功率）设置为  $preambleInitialReceivedTargetPower + DELTA\_PREAMBLE + (PREAMBLE\_TRANSMISSION\_COUNTER - 1) * powerRampingStep - 10 * \log_{10}(numRepetitionsPerPreambleAttempt-r13)$ ；而对于其它 CE 等级而言，PREAMBLE\_RECEIVED\_TARGET\_POWER 设置为 UE 的最大输出功率。（见 36.213 的 5.1.3 节）

$preambleInitialReceivedTargetPower$  为 preamble 的初始目标接收功率。对于 NB-IoT 而言，存在 2 种不同的 preamble 格式，二者的 CP 长度不同，但序列长度是相同的，不需要额外的功率差异，因此 DELTA\_PREAMBLE 的值固定为 0。在发送 preamble 时，NB-IoT 沿用了 LTE 中的 power-ramping 机制，即当一次随机接入尝试失败后，UE 会在上次发射功率的基础上，提升功率  $powerRampingStep$  来发送下次 preamble，以提高 preamble 发射成功的概率。这里的 PREAMBLE\_TRANSMISSION\_COUNTER 为随机接入尝试的次数， $powerRampingStep$  为功率提升的步长。NPRACH 的重复次数越多，对 SNR 的要求就越低，相应地，preamble 的目标接收功率可以降低，这也是公式中加入  $-10 * \log_{10}(numRepetitionsPerPreambleAttempt-r13)$  来调整 NPRACH 重复次数带来的影响的原因。

## 第六步：发送 preamble

最后，UE 的 MAC 层会通知物理层关于 preamble 的重复次数（ $numRepetitionsPerPreambleAttempt-r13$ ）、对应的 RA-RNTI、NPRACH 资源、子载波索引以及 PREAMBLE\_RECEIVED\_TARGET\_POWER 等信息，以便物理层基于这些信息来发送 preamble。

每个 CE 等级的最大 preamble 传输尝试次数通过  $maxNumPreambleAttemptCE-r13$  设置，而每次尝试中，preamble 传输的重复次数通过  $numRepetitionsPerPreambleAttempt-r13$  设置。也就是说，重复多次的 preamble 传输被认为只是一次 preamble 传输尝试。

RA-RNTI 的计算见下一节的介绍。

### 7.3.3.2 步骤二：eNodeB 发送 Random Access Response

UE 发送了 preamble 之后，将在 RAR 时间窗（RA Response window）内监听 NPDCCH，以接收对应 RA-RNTI 的 RAR。如果在此 RAR 时间窗内没有接收到 eNodeB 回复的 RAR MAC PDU，或 UE 接收到的 RAR MAC PDU 中没有包含了对所发送的 preamble 的响应，则认为此次随机接入过程失败。

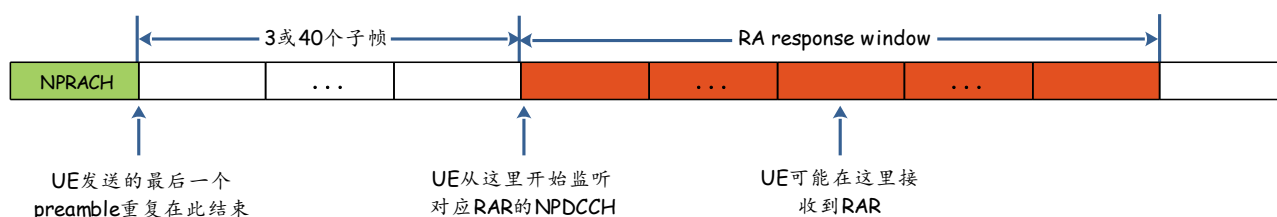


图 7-15: RA Response window

对于 NB-IoT UE 而言，对应所选的 CE 等级，在 NPRACH 重复次数大于或等于 64 的情况下，RAR 时间窗从“最后一个 preamble 重复的结束 + 41 个子帧”（原因见 5.3.3 节的介绍）处开始，并持续 *ra-ResponseWindowSize* 窗长；在 NPRACH 重复次数小于 64 的情况下，RAR 时间窗从“最后一个 preamble 重复的结束 + 4 个子帧”处开始，并持续 *ra-ResponseWindowSize* 窗长。UE 会在该窗口内去尝试接收使用对应 RA-RNTI 加扰的 RAR MAC PDU。

在 NB-IoT 中，*ra-ResponseWindowSize* 是 CE 等级相关的配置，不同的 CE 等级可能配置不同的值。对于 NB-IoT 而言，*ra-ResponseWindowSize* 的单位是 PDCCH 周期（见 4.3.1.2 节中关于 PDCCH period 的介绍），而不是子帧。其配置可以从 2 个 PDCCH 周期到 10 个 PDCCH 周期，并有  $ra-ResponseWindowSize = \text{Min}(ra-ResponseWindowSize-r13 \text{ 对应的 PDCCH 周期数} \times \text{PDCCH period 持续的时间}, 10.24s)$ 。

对应 NB-IoT UE，与某个 preamble 传输所在的 NPRACH 相关联的 RA-RNTI 通过如下方式计算：

$$RA-RNTI=1+\text{floor}(SFN\_id/4)$$

其中，*SFN\_id* 是该特定 NPRACH 资源所占的第一个系统帧的索引。也就是说，只要不同 UE 使用的 NPRACH 的第一个系统帧的索引相同，则这些 UE 的 RAR 就可以复用到同一个 RAR MAC PDU 中，并使用相同的 RA-RNTI 加扰。

某个 UE 发送的 preamble 时域位置是确定的，eNodeB 在成功解码 preamble 后，也就获得了该 preamble 的时域位置信息，进而知道了 RAR 中需要使用的 RA-RNTI。

当 UE 成功地接收到一个 RAR（使用前面介绍的 RA-RNTI 来解码），且该 RAR 中的 preamble（确切地说，是其对应的 subheader 中的 RAPID 字段）与 UE 发送的 preamble（确切地说，是 preamble 所占用的子载波索引）相同时，则认为成功接收了 RAR，此时 UE 就可以停止监听 RAR 了。

如果 UE 在 RAR 时间窗内没有接收到 RAR，或接收到的 RAR 中没有有一个 preamble 与自己发送的 preamble 相符合，则认为此次 RAR 接收失败。

针对 RAR 接收失败时的处理，NB-IoT 定义了几个相关的 RRC 层配置参数。

*maxNumPreambleAttemptCE-r13* 是对应一个 CE 等级内的 preamble 传输的最大尝试次数。如果 UE 使用同一个 CE 等级内的 NPRACH 资源来传输 preamble 的尝试次数达到了该 CE 等级定义的 preamble 传输的最大尝试次数时，它会尝试使用下一个 CE 等级的 NPRACH 资源来发送 preamble；如果不存在下一 CE 等级，则 UE 会留在当前的 CE 等级上，并继续使用当前 CE 等级的 NPRACH 资源来发送 preamble。接下来，UE 会根据步骤一的介绍，基于选择的 CE 等级、NPRACH 资源和是否支持 multi-tone Msg3 传输等，来选择一个子载波发送 preamble。

*preambleTransMax-CE-r13* 是 preamble 传输的最大尝试次数，如果在达到 preamble 最大传输尝试次数时，随机接入没有成功，则 MAC 层会给上层发送一个随机接入问题指示，并认为随机接入过程不成功地结束了。也就是说，在达到 preamble 最大传输尝试次数之前，即使 UE 在一个 CE 等级内的 preamble 尝试次数达到了 *maxNumPreambleAttemptCE-r13*，UE 依然会继续发送 preamble。

这里要注意区分“一次 preamble 传输尝试的重复次数（*numRepetitionsPerPreambleAttempt-r13*）”、“一个 CE 等级内的 preamble 传输的最大尝试次数（*maxNumPreambleAttemptCE-r13*）”和“preamble 传输的最大尝试次数（*preambleTransMax-CE-r13*）”的区别。

如果 UE 接收到 RAR MAC PDU，但其中没有一个 RAR 包含了对所发送的 preamble 的响应（即 UE 接收 RAR 失败），且该 MAC PDU 结束于子帧  $n$ ，则 UE 应该准备好在不迟于“子帧  $n$  的结束之后 12 ms 开始的 NB-IoT UL slot”上重新发送 preamble。（见 36.213 的 16.3.2 节）

如果 UE 在子帧  $n$  没有接收到一个 RAR MAC PDU，其中子帧  $n$  为 RAR 时间窗的最后一个子帧，则 UE 应该准备好在不迟于“子帧  $n$  的结束之后 12 ms 开始的 NB-IoT UL slot”上重新发送 preamble。（见 36.213 的 16.3.2 节）

### RAR MAC PDU

接下来，我会从 Random Access Response 的 MAC PDU 构成的角度来介绍 RAR 携带的信息。

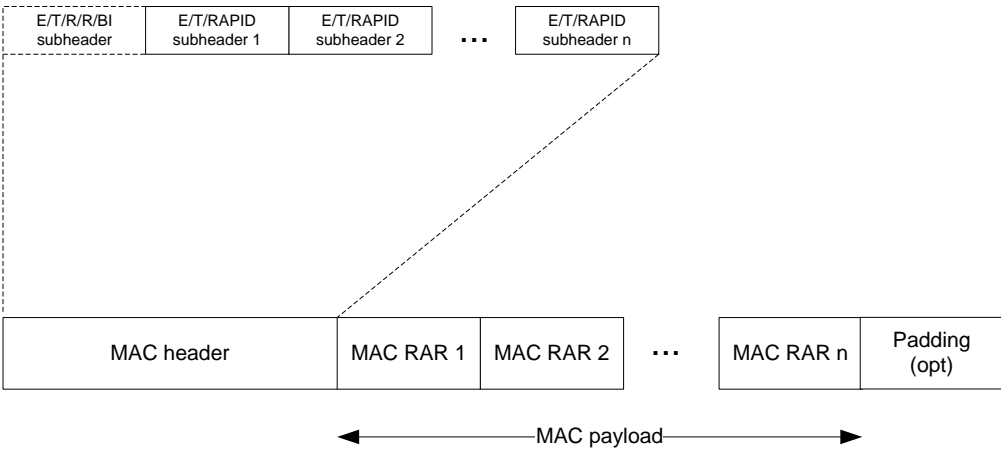


图 7-16：由 MAC header 和 MAC RARs 组成的 RAR MAC PDU

从图 7-16 可以看出，RAR MAC PDU 由 1 个 MAC 头（MAC header）+ 0 个或多个 MAC RAR（MAC Random Access Response，简称为 RAR）+ 可能存在的 padding 组成。

注意 RAR MAC PDU 与 RAR 的区别：1 个 RAR MAC PDU 包含 1 个或多个 RAR。

从 RAR MAC PDU 的结构可以看出，如果 eNodeB 在同一 NPRACH 资源（SFN\_id 相同）上检测到来自多个 UE 的随机接入请求，则使用一个 RAR MAC PDU 就可以对这些接入请求进行响应，每个随机接入请求（对应一个 preamble 子载波索引）的响应对应一个 RAR。

如果多个 UE 在同一 NPRACH 资源（SFN\_id 相同，使用同一 RA-RNTI）发送 preamble，则对应的 RAR 复用在同一 RAR MAC PDU 中。

RAR MAC PDU 在 DL-SCH 上传输，并用以 RA-RNTI 加扰的 NPDCCH 来指示。前面已经介绍过，使用相同 NPRACH 资源发送 preamble（preamble 使用的子载波索引不一定需要相同）的所有 UE 都监听相同的 RA-RNTI 加扰的 NPDCCH，并接收相同的 RAR MAC PDU，但不同 preamble 对应不同的 RAR。

由于 RAR MAC PDU 只能使用一个 RA-RNTI 加扰，这也意味着使用不同 NPRACH 资源（SFN\_id 不同）发送的 preamble 对应的 RAR 不能复用到同一个 RAR MAC PDU 中。

### MAC header

MAC header 由一个或多个 MAC subheader 组成。除了 Backoff Indicator subheader 外，每个 subheader 对应一个 RAR。如果包含 Backoff Indicator subheader（见 36.213 的 Figure 6.1.5-2），则该 subheader 只出现一次，且位于 MAC header 的第一个 subheader 处。

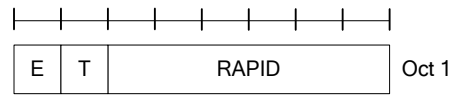


Figure 6.1.5-1: E/T/RAPID MAC subheader

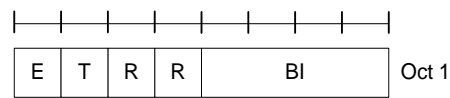


Figure 6.1.5-2: E/T/R/R/BI MAC subheader

如果 UE 收到了一个 Backoff Indicator subheader，则会保存一个 backoff 值，该值等于该 subheader 中的 BI 值；否则 UE 会将 backoff 值设为 0。

BI（Backoff Indicator）指定了 UE 重发 preamble 前需要等待的时间范围（取值范围见 36.321 的 Table 7.2-2）。如果 UE 认为 RAR 接收失败，那么 UE 会等待一段时间后，再发起随机接入。等待的时间为在 0 至 BI 指定的等待时间区间内选取一个随机值。（注：如果在步骤四中，冲突解决失败，也会有这样的回避机制）

BI 的取值从侧面反映了小区的负载情况，如果同时接入的 UE 多，则该值可以设置得大些；如果同时接入的 UE 少，该值就可以设置得小些。与 LTE 中的 BI 取值范围相比，NB-IoT 中的 BI 取值可以更大。

RAPID 为 Random Access Preamble IDentifier 的简称，为 eNodeB 在检测 preamble 时得到的子载波索引。如果 UE 发现该值与自己发送 preamble 时使用的子载波索引相同，则认为成功接收到对应的 RAR。在 NB-IoT 中，RAPID 对应的是发送 preamble 时使用的子载波索引，这与 LTE 中对应 preamble index 是不同的。

### MAC RAR

NB-IoT 中，RAR 的结构如 36.321 的 Figure 6.1.5-3b 所示。

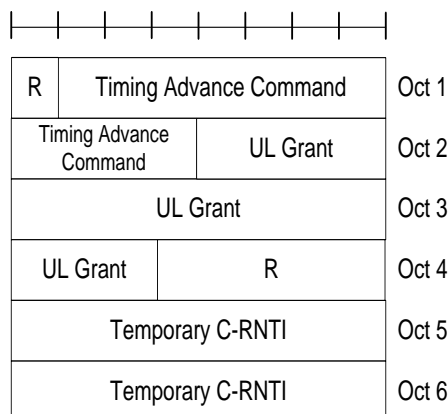


Figure 6.1.5-3b: MAC RAR for NB-IoT UEs

11 比特的 Timing advance command 用于指定 UE 上行同步所需要的时间调整量。

16 比特的 TC-RNTI 用于 UE 和 eNodeB 的后续传输。冲突解决后，该值可能变成 C-RNIT。

RAR 中包含的 15 比特的 UL Grant 信息，用于指示 Msg3 初传的调度信息。这 15 比特从高到底依次解析为：

RAR 中包含的 15 比特 UL Grant 信息（见 36.213 的 16.3.3 节）		
字段	占用的比特数	说明
Uplink subcarrier spacing $\Delta f$	1 比特	指示上行子载波间距。值为 0 表示使用 3.75kHz 子载波间距；值为 1 表示使用 15kHz 子载波间距
Subcarrier indication field $I_{sc}$	6 比特	指示分配给 Msg3 的子载波（见 5.3.1.1 节的介绍）
Scheduling delay field ( $I_{Delay}$ )	2 比特	用于计算 Msg3 的时域上的起始位置，其计算方式见 5.3.1.1 节的介绍。并有 $I_{Delay} = 0$ 时， $k_0 = 12$ ；而 $I_{Delay}$ 的其它取值与 $k_0$ 的对应关系见 36.213 的 Table 16.5.1-1。但这里的 NB-IoT 下行子帧 $n$ 是用于发送此 RAR 的 NPDSCH 的最后一个子帧
Msg3 repetition number $N_{Rep}$	3 比特	Msg3 的重复次数 $N_{Rep}$
MCS index	3 比特	根据 36.213 的 Table 16.3.3-1 来指示 TBS、调制阶数以及用于 Msg3 的 RU 数

Table 16.3.3-1: MCS index for Msg3 NPUSCH

MCS Index $I_{\text{MCS}}$	Modulation $\Delta f = 3.75 \text{ kHz}$ or $\Delta f = 15 \text{ kHz}$ and $I_{\text{sc}} = 0, 1, \dots, 11$	Modulation $\Delta f = 15 \text{ kHz}$ and $I_{\text{sc}} > 11$	Number of RUs $N_{\text{RU}}$	TBS
'000'	pi/2 BPSK	QPSK	4	88 bits
'001'	pi/4 QPSK	QPSK	3	88 bits
'010'	pi/4 QPSK	QPSK	1	88 bits
'011'	reserved	reserved	reserved	reserved
'100'	reserved	reserved	reserved	reserved
'101'	reserved	reserved	reserved	reserved
'110'	reserved	reserved	reserved	reserved
'111'	reserved	reserved	reserved	reserved

RAR 指示的 Msg3 传输一定是初传，而 Msg3 的初传的冗余版本为 0，因此冗余版本信息不需要在 RAR 中指示。

对于 NB-IoT 而言，当有上行数据传输时，例如需要解决冲突，eNodeB 在 RAR 中分配的 UL grant 不能小于 88 比特（见 36.321 的 5.1.4 节。另外从 36.213 的 Table 16.3.3-1 也可以看出，至少在 Rel-13 中，RAR 中分配的 UL grant 的大小只有 88 比特一种取值）。

由于 UE 随机选择一个 preamble 用于随机接入，可能导致位于同一 CE 等级内的多个 UE 同时选择同一 NPRACH 资源里的同一个子载波来发送 preamble，从而导致冲突的出现（使用相同的 RA-RNTI 和子载波索引，因此不能确定 RAR 是对哪个 UE 的响应），这时需要一个冲突解决机制来解决这个问题。冲突的存在也是 RAR 不使用 HARQ 的原因之一。另外需要说明的是，在 Rel-13 中，虽然 PDCCH order 里可能给 UE 指定使用的 NPRACH 资源和子载波，但这些资源并不是独占的资源，其它 UE 在随机选择时，也有可能选到这些资源来发送 preamble。（这里我们认为 Rel-13 中不会使用基于非竞争的那部分资源）

### 7.3.3.3 步骤三：UE 发送 Msg3

之所以将随机接入过程中的第 3 条消息称为 Msg3 而不是某一条具体消息的原因在于，根据 UE 状态的不同和应用场景的不同，这条消息也可能不同，因此统称为 Msg3，即第 3 条消息。

与随机接入的触发事件对应起来，Msg3 携带的信息如下：

1. 如果是初次接入（initial access），Msg3 为在 CCCH 上传输的 RRC 连接请求 *RRCCConnectionRequest-NB*，且至少需要携带 NAS UE 标志信息，但并不会携带 NAS 消息。（此时 UE 的唯一标志为 S-TMSI 或一个随机数，并通过 *ue-Identity-r13* 字段指示）
2. 如果是 RRC 连接重建（RRC Connection Re-establishment），Msg3 为 CCCH 上传输的 RRC 连接重建请求 *RRCCConnectionReestablishmentRequest-NB*，且不携带任何 NAS 消息。（此时 UE 的唯一标志为 *ue-Identity-r13*，其中包含了 UE 之前使用的 C-RNTI 以及 NB-IoT 小区的 PCI 等信息）
3. 如果是 RRC 连接恢复（RRC Connection Resume），Msg3 为在 CCCH 上传输的 RRC 连接恢复请求 *RRCCConnectionResumeRequest-NB*，且至少需要携带 Resume ID 信息，以用于恢复 RRC 连接。（此时 UE 的唯一标志为 Resume ID，并通过 *resumeID-r13* 字段指示）

4. 对于其它触发事件，则至少需要携带 C-RNTI。（此时 UE 的唯一标志为 C-RNTI）

Msg3 在 UL-SCH 上传输，使用 HARQ，且 RAR 中带的 UL grant 指定的用于 Msg3 的 TB 大小至少为 88 比特。这里，我们整理了这 88 比特包含了哪些信息（参考 R2-163232 的说明）。

初始接入过程中，Msg3 中包含的信息：

元素	比特数
CCCH 对应的 MAC subheader	8
DPR MAC CE	8
<i>RRCCoalitionRequest-NB-r13</i>	72。其分布如下： <ul style="list-style-type: none"> <li>• <i>ue-Identity-r13</i>: 41 比特；</li> <li>• <i>establishmentCause-r13</i>: 3 比特；</li> <li>• <i>multiToneSupport-r13</i>: 1 比特；</li> <li>• <i>multiCarrierSupport-r13</i>: 1 比特；</li> <li>• <i>spare</i>: 22 比特；</li> <li>• RRC 层的开销: 4 比特。</li> </ul>

RRC 连接恢复过程中，Msg3 中包含的信息：

元素	比特数
CCCH 对应的 MAC subheader	8
DPR MAC CE	8
<i>RRCCoalitionResumeRequest-NB</i>	72。其分布如下： <ul style="list-style-type: none"> <li>• <i>resumeID-r13</i>: 40 比特；</li> <li>• <i>shortResumeMAC-I-r13</i>: 16 比特；</li> <li>• <i>resumeCause-r13</i> : 3 比特；</li> <li>• <i>spare</i>: 9 比特；</li> <li>• RRC 层的开销: 4 比特。</li> </ul>

RRC 连接重建过程中，Msg3 中包含的信息：

元素	比特数
CCCH 对应的 MAC subheader	8
DPR MAC CE	8
<i>RRCCoalitionReestablishmentRequest-NB</i>	72。其分布如下： <ul style="list-style-type: none"> <li>• <i>ue-Identity-r13</i>: 41 比特； <ul style="list-style-type: none"> <li>- <i>c-RNTI</i>: 16 比特；</li> <li>- <i>physCellId</i>: 9 比特；</li> <li>- <i>shortMAC-I</i>: 16 比特。</li> </ul> </li> <li>• <i>reestablishmentCause-r13</i>: 2 比特；</li> <li>• <i>spare</i>: 25 比特；</li> <li>• RRC 层的开销: 4 比特。</li> </ul>

对于其它事件触发的随机接入过程，UE 原本处于 RRC\_CONNECTED 态，Msg3 中至少需要携带 C-RNTI MAC CE。

如果 UE 检测到使用相关 RA-RNTI 加扰的 NPDCCH，且对应的 RAR MAC PDU 中包含了对所发送的 preamble 的响应（即 UE 成功地接收了 RAR），且该 MAC PDU 结束于子帧  $n$ ，则 UE 应当在子帧  $n+k_0$  结束后的第一个 NB-IoT 上行 slot 发送 Msg3。其中  $k_0$  由对应 RAR 中的 UL grant 的 Scheduling delay field ( $I_{\text{Delay}}$ ) 字段决定。（具体见 7.3.3.2 节，或 36.213 的 16.3.2 节和 16.3.3 节的介绍）

Msg3 中需要包含一个重要信息：每个 UE 唯一的标志。该标志将用于步骤四的冲突解决。

对于处于 RRC\_CONNECTED 态的 UE 来说，其唯一标志是 C-RNTI。

对于非 RRC\_CONNECTED 态的 UE 来说，将使用一个来自核心网或 eNodeB 的唯一的 UE 标志（S-TMSI、一个随机数或 Resume ID 等）作为其标志。

当 UE 处于 RRC\_CONNECTED 态但上行不同步时，UE 有自己的 C-RNTI，在随机接入过程的 Msg3（此时 Msg3 不使用 CCCH 来传输）中，UE 会通过 C-RNTI MAC Control Element 将自己的 C-RNTI 告诉 eNodeB，eNodeB 在步骤四中会使用这个 C-RNTI 来解决冲突。

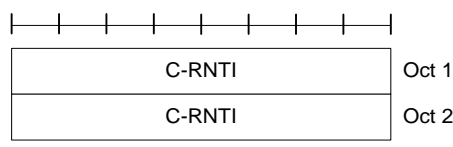


Figure 6.1.3.2-1: C-RNTI MAC control element

当 UE 在随机接入过程中使用上行 CCCH 来发送 Msg3 消息时，UE 还没有 C-RNTI，此时 UE 会使用来自核心网（或 eNodeB）的 UE 标志。在步骤四中，eNodeB 会通过发送 UE Contention Resolution Identity MAC Control Element（携带了这个 UE 标志）来解决冲突。

上行传输通常使用 UE 特定的信息，如 C-RNTI，对 UL-SCH 的数据进行加扰。但此时冲突还未解决，加扰不能基于 C-RNTI，而只能使用 TC-RNTI。也就是说，Msg3 只会使用 TC-RNTI 进行加扰。

#### 7.3.3.4 步骤四：eNodeB 发送 contention resolution

在步骤三中已经介绍过，UE 会在 Msg3 中携带自己唯一的标志：C-RNTI 或来自核心网（或 eNodeB）的 UE 标志。eNodeB 在冲突解决机制中，会在 Msg4（我们把步骤四的消息称为 Msg4）中携带该唯一的标志以指定胜出的 UE。而其它没有在冲突解决中胜出的 UE 将重新发起随机接入。

eNodeB 会通过使用 C-RNTI 加扰的 NPDCCH，或 DL-SCH 上传输的 UE Contention Resolution Identity MAC Control Element 来指明哪个 UE 在冲突解决中胜出。

UE 发送了 Msg3 后，会启动一个 *mac-ContentionResolutionTimer-r13*（对于 NB-IoT 而言，每一个 CE 等级可能有其独自の *mac-ContentionResolutionTimer-r13* 配置），并在 Msg3（可能重复多次，称为一个



bundle) HARQ 重传结束时，重启该定时器。在该 timer 超时或停止之前，UE 会一直监听对应的 NPDCCH 以便接收 Msg4。

与 *ra-ResponseWindowSize-r13* 类似，在 NB-IoT 中，*mac-ContentionResolutionTimer-r13* 的单位也是 PDCCH 周期，而不是子帧。UE 是基于所选的 CE 等级，来确定所选的 *ra-ResponseWindowSize-r13* 和 *mac-ContentionResolutionTimer-r13* 的。

如果 UE 监听到了 NPDCCH，且 UE 在发送 Msg3 时携带了 C-RNTI MAC Control Element，则在以下 2 种情况下，UE 认为冲突解决成功（即该 UE 成功接入，此时 UE 会停止 *mac-ContentionResolutionTimer-r13*，并丢弃 TC-RNTI，并且在 NB-IoT UE 配置了 non-anchor carrier 的情况下，NPDCCH 指示的上行或下行资源是针对于 non-anchor 上的资源的。**注意：**这 2 种情况下 TC-RNTI 不会提升为 C-RNTI）：

- 1. 随机接入过程由 MAC 子层或 RRC 子层触发，且 UE 在 Msg3 中接收到的 NPDCCH 由 Msg3 携带的 C-RNTI 加扰，且给新传的数据分配了上行资源；
- 2. 随机接入过程由 PDCCH order 触发，且 UE 在 Msg4 中接收到的 NPDCCH 由 Msg3 携带的 C-RNTI 加扰。

如果 Msg3 在 CCCH 上发送，且 UE 接收到的对应 Msg4 的 NPDCCH 由 RAR 中指定的 TC-RNTI 加扰，则当成功解码出的 Msg4 对应的 MAC PDU 中包含的 UE Contention Resolution Identity MAC Control Element 与 Msg3 发送的 CCCH SDU 的前 48 比特相匹配时，UE 会认为随机接入成功并将自己的 TC-RNTI 设置成 C-RNTI；如果不匹配，则认为冲突解决失败，并丢弃 TC-RNTI。（只要 UE 成功接收到 TC-RNTI 加扰的 NPDCCH，并成功解码 Msg4，UE 就停止 *mac-ContentionResolutionTimer-r13*，但不需要等到冲突解决成功才停止该定时器。**注意：**这种情况下 TC-RNTI 会提升为 C-RNTI）

UE Contention Resolution Identity MAC Control Element 长度固定为 48 比特，并由 LCID 为“11100”的 MAC subheader 指示。该 48 比特的值包含了 Msg3 的上行 CCCH SDU 中的前 48 个比特。

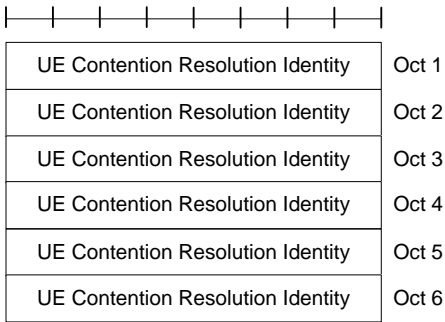


Figure 6.1.3.4-1: UE Contention Resolution Identity MAC control element

如果 *mac-ContentionResolutionTimer-r13* 超时，UE 会丢弃 TC-RNTI 并认为冲突解决失败。

如果冲突解决失败，UE 需要

- 1. 清空 Msg3 对应的 HARQ buffer；
- 2. 将 PREAMBLE\_TRANSMISSION\_COUNTER 加 1，如果此时 PREAMBLE\_TRANSMISSION\_COUNTER = *preambleTransMax-CE-r13* + 1，则通知上层随机接入出错（Random Access problem indication），对于 NB-IoT 而言，随机接入过程就不成功地结束了（这与 LTE 是不同的，LTE 中的

preamble 达到最大传输次数却还未成功时，MAC 层并不会停止发送 preamble。而在 NB-IoT 中，preamble 达到最大传输次数却还未成功时，就停止发送 preamble 了）；

3. 在 0~BI 值之间随机选择一个 backoff time，UE 延迟 backoff time 后，再次发起随机接入。

如果 UE 接入成功，UE 会

1. 如果收到显式指示的 *ra-PreambleIndex*，则丢弃；
2. 清空 Msg3 对应的 HARQ buffer。

对于 Msg4 而言，也使用 HARQ，但不需要与 Msg3 同步。从前面的介绍可以看出，对于初始接入、RRC 连接恢复和无线链路失效而言，Msg4（包括对应的 NPDCCH）使用 TC-RNTI 加扰，且使用 RLC-TM 模式；而对于处于 RRC\_CONNECTED 态的 UE 而言，Msg4（包括对应的 NPDCCH）使用 C-RNTI 加扰。

简单地说：

1. 如果 UE 原本就处于 RRC\_CONNECTED 态，则该 UE 在小区内有唯一的标志 C-RNTI。步骤三中，Msg3 会通过 C-RNTI MAC Control Element 把这个 C-RNTI 带给 eNodeB；步骤四中，如果此 UE 在冲突解决中胜出，eNodeB 就使用这个 C-RNTI 对 NPDCCH 进行加扰。UE 收到以此 C-RNTI 加扰的 NPDCCH，就知道自己接入成功了。
2. 如果 UE 原本不处于 RRC\_CONNECTED 态，则该 UE 在小区内不存在 C-RNTI，其唯一标志来自核心网或 eNodeB。步骤三中，Msg3 会将该唯一标志带给 eNodeB；步骤四中，如果此 UE 在冲突解决中胜出，eNodeB 会通过 UE Contention Resolution Identity MAC Control Element 将步骤三中的 Msg3 中的 CCCH SDU 的前 48 比特发回给 UE，UE 比较 Msg3 和 Msg4，发现二者匹配，就知道自己接入成功了。

### 7.3.4 PDCCH order 触发的随机接入过程

如果随机接入过程是由一个 PDCCH order 触发的，且该 PDCCH order 结束于子帧  $n$ ，则 UE 会在  $n+k_2$ （ $k_2 \geq 8$ ）结束后的第一个有可用 NPRACH 资源的子帧上开始发送 preamble。（见 36.213 的 16.3.2 节）

PDCCH order 通过 DCI format N1（见 4.3.1.4 节）发送，并包含了如下信息：

- 分配给 NPRACH 的子载波  $n_{sc} = I_{sc}$ ，其中  $I_{sc}$  由对应 DCI 的 **Subcarrier indication of NPRACH** 字段指定，且  $I_{sc} = 48, 49, \dots, 63$  是预留而不使用的。该字段指定了使用的 preamble 以及该 preamble 所在的子载波（其实二者对应同一个东西）。
- NPRACH 的重复次数（ $N_{Rep}$ ）根据对应 DCI 的 **Starting number of NPRACH repetitions**（ $I_{Rep}$ ）字段查 36.213 的 Table 16.3.2-1 得到，其中  $R_1 < R_2 < R_3$ 。 $R_1$  对应拥有最小 *numRepetitionsPerPreambleAttempt-r13* 的 NPRACH 资源配置，且  $R_1$  的值等于该 *numRepetitionsPerPreambleAttempt-r13*；同样的， $R_2$ （如果存在的话）和  $R_3$ （如果存在的话）的值等于对应 NPRACH 资源配置中的 *numRepetitionsPerPreambleAttempt-r13* 值。

**Table 16.3.2-1: Number of repetitions ( $N_{\text{Rep}}$ ) for NPRACH following a “PDCCH order”**

$I_{\text{Rep}}$	$N_{\text{Rep}}$
0	$R_1$
1	$R_2$
2	$R_3$
3	Reserved

## 7.4 Paging

在 NB-IoT 中，Paging 消息的作用包括：（1）向处于 RRC\_IDLE 态的 UE 发送呼叫请求；（2）通知处于 RRC\_IDLE 和 RRC\_CONNECTED 态的 UE，系统信息发生了变化。

Paging 作为系统信息变更指示的作用见 7.2.4 节关于“系统信息有效性和变更通知”的介绍，本节主要关注其呼叫请求的功能以及 UE 如何接收 Paging 消息。

与 LTE 类似，传输信道 PCH（Paging CHannel）用于传输来自逻辑信道 PCCH 的 *Paging-NB* 信息，并直接映射到物理信道 NPDSCH 上。*Paging-NB* 信息可能包含被寻呼的 UE 列表（通过 *pagingRecordList-r13* 指示），以及指示系统信息是否发生了变化的信息（通过 *systemInfoModification-r13* 或 *systemInfoModification-eDRX-r13* 指示）。

```

Paging-NB ::= SEQUENCE {
    pagingRecordList-r13      PagingRecordList-NB-r13      OPTIONAL,    -- Need ON----指示被寻呼的UE列表
    systemInfoModification-r13  ENUMERATED {true}          OPTIONAL,    -- Need ON----指示除SIB14-NB和
SIB16-NB外的BCCH变更。此字段用于UE使用的eDRX cycle不长于BCCH变更周期的场景
    systemInfoModification-eDRX-r13  ENUMERATED {true}      OPTIONAL,    -- Need ON----指示除SIB14-NB和
SIB16-NB外的BCCH变更。此字段用于UE使用的eDRX cycle长于BCCH变更周期的场景
    nonCriticalExtension       SEQUENCE {}                  OPTIONAL
}

PagingRecordList-NB-r13 ::= SEQUENCE (SIZE (1..maxPageRec)) OF PagingRecord-NB-r13

PagingRecord-NB-r13 ::= SEQUENCE {
    ue-Identity-r13           PagingUE-Identity,    ----指示被寻呼的UE的NAS ID
    ...
}

```

只有处于 RRC\_IDLE 态的 UE，才会读取 *pagingRecordList-r13* 中的信息。UE 接收到 Paging 消息后，如果发现它的 ID 在列表中指示，就知道自己被呼叫了。并且它会把自己被呼叫的信息告诉上层，然后上层会触发 UE 建立一条 RRC 连接。

Paging 支持非连续接收（Discontinuous Reception, DRX），使得处于 RRC\_IDLE 态的 UE 只在预先定义好的时间段内“醒来”以接收 Paging 消息，而在其它时间可以保持“休眠”状态，这样就能够降低功耗，提升 UE 的电池使用时间。

7.4.1 PF/PO

为了接收来自 eNodeB 的 Paging 消息，处于 RRC\_IDLE 态的 UE 会监听使用 P-RNTI（值为 0xFFFFE）加扰的 NPDCCH。处于 RRC\_IDLE 态的 UE 只会监听与 Paging 相关的某些子帧，即在寻呼帧（Paging Frame，PF）内的寻呼机会（Paging Occasion，PO）上去尝试接收 Paging 消息。

对于使用 P-RNTI 加扰的 NPDCCH，PO 指示的是 NPDCCH 重复（而不是 NPDSCH）的起始子帧，如果 PO 指示的子帧不是一个有效的 NB-IoT 下行子帧，则 PO 之后的第一个有效的 NB-IoT 下行子帧才是 NPDCCH 重复的起始子帧。

PF 是一个无线帧，该帧可能包含一个或多个 PO。使用 DRX 时，UE 在每个 DRX cycle 只需要监听一个 PO。也就是说，对应每个 UE，在每个 Paging 周期内只有 1 个子帧可用于发送 Paging。这里的 DRX cycle 与 Paging 周期是同一概念。

PF 是满足如下公式的系统帧：

SFN mod T= (T div N)\*(UE\_ID mod N)

使用索引 i\_s 查 36.304 的 7.2 节定义的下表，可得到 PO。其中 i\_s 通过下面的公式得到：

i\_s = floor(UE\_ID/N) mod Ns

FDD:

Ns	PO when i_s=0	PO when i_s=1	PO when i_s=2	PO when i_s=3
1	9	N/A	N/A	N/A
2	4	9	N/A	N/A
4	0	4	5	9

Paging 相关的配置信息在 SIB2-NB 的 pcch-Config-r13 中指定，其结构 PCCH-Config-NB-r13 如下：

```
PCCH-Config-NB-r13 ::= SEQUENCE {
    defaultPagingCycle-r13    ENUMERATED {rf128, rf256, rf512, rf1024}, ----小区特定的默认Paging周期，对应36.304
    nB-r13                    ENUMERATED {
                                fourT, twoT, oneT, halfT, quarterT, one8thT,
                                one16thT, one32ndT, one64thT,
                                one128thT, one256thT, one512thT, one1024thT,
                                spare3, spare2, spare1 }, ----nB表示在每个DRX cycle内包含了多少个PO。
    npdcch-NumRepetitionPaging-r13    ENUMERATED {
                                r1, r2, r4, r8, r16, r32, r64, r128,
                                r256, r512, r1024, r2048,
                                spare4, spare3, spare2, spare1 } ----用于Paging的NPDCCH公共搜索空
}
间（Type-1公共搜索空间）的最大重复次数
```

而基于上面的配置，PF/PO 计算相关参数说明如表 7-3 所示：

表 7-3：PF/PO 公式的相关参数说明

参数	说明
$T$ （以系统帧为单位）	UE 使用的 DRX cycle。NB-IoT 不支持配置 UE 特定的 DRX 值，因此 $T$ 等于 SIB2-NB 中的 <i>defaultPagingCycle-r13</i>
$nB$	取值范围：{4T, 2T, T, T/2, T/4, T/8, T/16, T/32, T/64, T/128, T/256, T/512, T/1024}。实际上 $nB$ 表示在每个 DRX cycle 内包含了多少个 PO。 (通过 SIB2-NB 的 IE: <i>PCCH-Config-NB-r13</i> -> <i>nB-r13</i> 配置)
$N$	$N = \min(T, nB)$ 。实际上 $N$ 表示在每个 DRX cycle 内包含了多少个 PF
$N_s$	$N_s = \max(1, nB/T)$ 。实际上 $N_s$ 表示在每个 PF 内包含了多少个 PO
UE_ID	$UE\_ID = IMSI \bmod 4096$ （针对 NB-IoT 而言）

IMSI 是一连串的 10 进制数字 (0..9)，IMSI 在公式中被解释成 10 进制整数，其中第 1 位为最高位，依次类推。例如：IMSI = 12 (digit 1 = 1, digit 2 = 2)，计算时该 IMSI 被解释成 10 进制数“12”，而不是“ $1 \times 16 + 2 = 18$ ”。

如果 UE 没有 IMSI，例如 UE 因为没有 USIM 而只能进行紧急呼叫，则该 UE 的 UE\_ID = 0。

可以看出，PF 和 PO 的位置取决于 IMSI，不同的 UE 拥有不同的 IMSI，进而拥有不同的 PO，这些 PO 在时间上是均匀分布的。对于 UE 来说，监听一个 DRX cycle 内的一个 PO 就足够了，如果存在多个 PO，Paging 消息会在每个 PO 上重复发送。

接下来，我们通过 1 个例子来说明如何计算 NB-IoT UE 的 PF 和 PO。假设某个 UE 的配置如表 7-4 所示：

表 7-4：某个 UE 的 Paging 配置举例

NB-IoT UE 的配置	
IMSI	404685505598162
$T$	32
$nB$	$nB = T/2 = 16$
计算 PF 和 PO	
UE_ID	$UE\_ID = IMSI \bmod 4096 = 722$
$N$	$N = \min(T, nB) = 16$
$N_s$	$N_s = \max(1, nB/T) = 1$
$i_s$	$i_s = \text{floor}(UE\_ID/N) \bmod N_s = 0$
PF	满足 $SFN \bmod T = (T \text{ div } N) * (UE\_ID \bmod N)$ 。计算出 PF 需满足： $SFN \bmod 32 = 4$
PO	通过 $N_s$ 和 $i_s$ 查表可知，子帧号为 9

从计算结果可以看出：

该 UE 需要每 320 ms ( $T = 32 \times 10\text{ms}$ ) 醒来一次，并尝试接收 Paging 消息（即该 UE 的 DRX cycle 为 32 个系统帧）。

在该 UE 的 DRX cycle 内，每 2 个系统帧 ( $N = 16$ ) 中会有一个 PF 可用于 Paging。该 UE 对应的 DRX cycle 内有 16 个 PO ( $nB = 16$ )。不同的 UE 基于其不同的 UE\_ID，可以在这 16 个 PO 中选择 1 个来接收 Paging。

每个 PF 内只有 1 个 PO ( $N_s = 1$ )。根据预先定义好的样式，对于  $N_s = 1$ ， $i_s = 0$ ，PF 内的子帧 9 用于 PO。

对应表 7-4 中的 UE 配置，其 PF、PO 如图 7-17 所示：

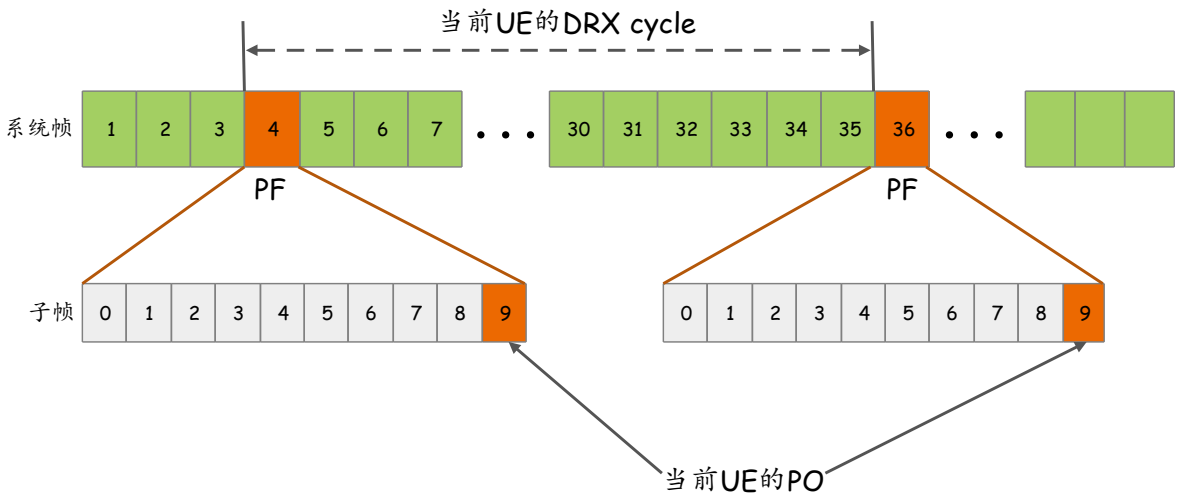


图 7-17: IMSI = 404685505598162、 $T = 32$ 、 $nB = 16$  的 NB-IoT UE 的 PF/PO

基于前面的介绍，UE 就知道了在时域上，Paging 相关的 NPDCCH 所在的起始子帧（即 PF/PO）。接下来，UE 就会在对应的搜索空间（见 4.3.1.2 节）内去盲检使用 P-RNTI 加扰的 NPDCCH。盲检成功并解码对应的 DCI format N1（见 4.3.1.4 节）后，UE 就知道了用于传输 Paging 的 NPDSCH 如何传输以及对应的 timing 关系（见 4.3.2 节），从而接收到 Paging 消息。

### 7.4.2 eDRX 下的 Paging

与 DRX 相比，eDRX（extended DRX）特性允许 IoT 设备在更长的时间内处于休眠状态，从而达到更为省电的目的（关于 eDRX，详见 10.2 节的介绍）。与 eMTC 类似，eDRX 同样适用于 NB-IoT，并引入了超帧（H-SFN）的概念。如果支持 eDRX，则 UE 不监听寻呼消息的时间间隔可以大大延长，最大可以间隔差不多 3 小时。同时为了提升 UE 在 eDRX 周期内的寻呼接收成功率，引入了寻呼传输窗（PTW）的概念，以允许网络在一个 PTW 内多次寻呼 UE。UE 只会在其对应的 PTW 内的 PF/PO 上去监听寻呼消息。

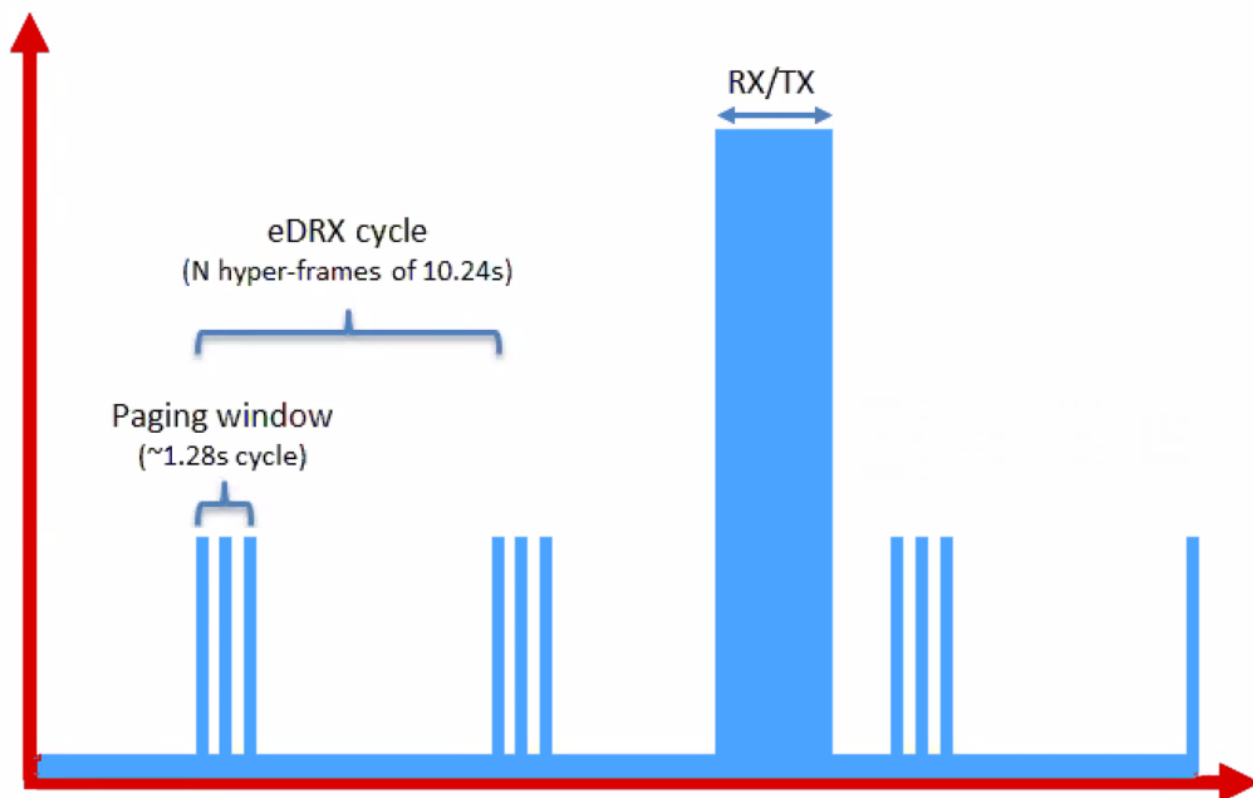


图 7-18: eDRX 下的 Paging

Paging 的 eDRX 处理是 NAS 层的功能，eNodeB 不会通过 RRC 消息给 UE 下发 eDRX 相关的配置。eNodeB 和 UE 均需要知道该 UE 的 eDRX 配置（针对同一 UE，二者得到的应该是相同的配置信息），才能在正确的时频资源上去收发空口上的 Paging 消息。

由 MME 发往 eNodeB 的 Paging 消息中，可能携带一个 IE: *NB-IoT Paging eDRX Information*。该 IE 包含了用于 NB-IoT Paging eDRX 的参数信息：*NB-IoT Paging eDRX Cycle*，对应 36.304 中的 eDRX cycle  $T_{eDRX}$ （以超帧为单位），和 *NB-IoT Paging Time Window*，对应 36.304 中的 PTW 长度  $L$ （以 2.56s 为单位）。eNodeB 收到来自 MME 的 Paging 消息后，会根据 36.304 的介绍来确定 UE 的 PO。至此，eNodeB 就知道了 UE 的 eDRX 配置了。（见 36.413 的 8.5 节和 9.2.1.115 节）

在 Attach 或 TAU 流程中，UE 可以通过在 *ATTACH REQUEST* 或 *TRACKING AREA UPDATE REQUEST* 消息中携带 IE: *extended DRX parameters* 来请求网络给它配置一个 eDRX。当网络接受 Attach 或 TAU 流程时，网络可以通过在 *ATTACH ACCEPT* 或 *TRACKING AREA UPDATE ACCEPT* 消息中携带 IE: *Extended DRX parameters* 来接受 UE 使用 eDRX 的请求，并将 eDRX 的参数配置（ $T_{eDRX}$  和 PTW 长度  $L$ ）下发给 UE。如果 UE 希望一直使用 eDRX，则它需要在每次 Attach 或 TAU 流程中，都携带 IE: *Extended DRX parameters*。至此，UE 就知道了自己的 eDRX 配置了。（见 24.301，其中 *Extended DRX parameters* 的定义见 24.008 的 10.5.5.32 节）

前面介绍的是针对同一 UE，eNodeB 和 UE 分别如何获取该 UE 的 eDRX 配置： $T_{eDRX}$  和 PTW 长度  $L$ 。接下来，我们来看看基于这个配置，UE 如何确定何时去监听 PO（或者说，eNodeB 何时在对应的 PO 发送 Paging）。

如果 UE 配置的  $T_{eDRX}$  为 512 个系统帧，它会按照参数  $T=512$  来监听 PO（对于 NB-IoT 而言， $T_{eDRX}$  最小为 2 个超帧，即 2 个 H-SFN，因此不用考虑这种情况）；否则，UE 会在上层配置给 UE 的 PTW 期间，按照 7.4.1 节的介绍来监听 PO（也就是说，在 PTW 内，PF/PO 的计算方式与非 eDRX 情况下的 PF/PO 计算方式相同），直到 UE 在 PTW 期间接收到一个包含 UE 的 NAS ID 的 Paging 消息，或 PTW 结束为止。

PTW 是 UE 特定的，并由 Paging Hyperframe (PH)、PH 内的起始位置 (PTW\_start) 和结束位置 (PTW\_end) 共同决定。PH、PTW\_start 和 PTW\_end 通过下面的公式得到：

PH 是满足  $H-SFN \bmod T_{eDRX,H} = (UE\_ID\_H \bmod T_{eDRX,H})$  的 H-SFN（超帧）。其中

- UE\_ID\_H: 对应使用 P-RNTI 加扰的 NPDCCH，其值为 Hashed ID 的最高 12 个比特。Hashed\_ID 是 S-TMSI 的 b31, b30..., b0 的循环冗余校验 (CRC) 值，其计算方式见 36.304 的 7.3 节的介绍。
- $T_{eDRX,H}$ : UE 的 eDRX cycle，以超帧 H-SFN 为单位。其配置方式见之前关于  $T_{eDRX}$  的介绍。（对于 NB-IoT 而言， $T_{eDRX,H}=2, \dots, 1024$  Hyper-frames，对应 eDRX 下的 DRX cycle 最大约为 3 小时。而从前一节的介绍可以看出，NB-IoT 在非 eDRX 下的 DRX cycle 最大为 10.24s）

PTW\_start 表示 PTW 内的 PH 上的第一个系统帧，其 SFN 值满足： $SFN = 256 * i_{eDRX}$ 。其中  $i_{eDRX} = \text{floor}(UE\_ID\_H / T_{eDRX,H}) \bmod 4$ 。

PTW\_end 表示 PTW 的最后一个系统帧，其 SFN 值满足： $SFN = (PTW\_start + L * 100 - 1) \bmod 1024$ 。其中 L=上层配置的 PTW 长度（以秒为单位，最大为  $16 \times 2.56s = 40.96s$ ）。

## 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 《A Primer on 3GPP Narrowband Internet of Things (NB-IoT)》，Ericsson, Y.-P. Eric wang, XingQin Lin, Ansuman Adhikary, etc.
- [3] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [4] 3GPP TS 36.211 V13.4.0, 2016-12; Physical channels and modulation (Release 13)
- [5] 3GPP TS 36.212 V13.4.0, 2016-12; Multiplexing and channel coding (Release 13)
- [6] 3GPP TS 36.213 V13.4.0, 2016-12; Physical layer procedures (Release 13)
- [7] 3GPP TS 36.321 V13.4.0, 2016-12; Medium Access Control (MAC) protocol specification
- [8] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification
- [9] 3GPP TS 36.304 V13.4.0, 2016-12; User Equipment (UE) procedures in idle mode (Release 13)



## 第8章 连接控制

由于 NB-IoT 不支持 handover 到其它 RAT（如 GSM/UTRA/LTE）上，UE 在 RRC 层只支持 2 种状态：RRC\_IDLE 态和 RRC\_CONNECTED 态，其状态机很简单：

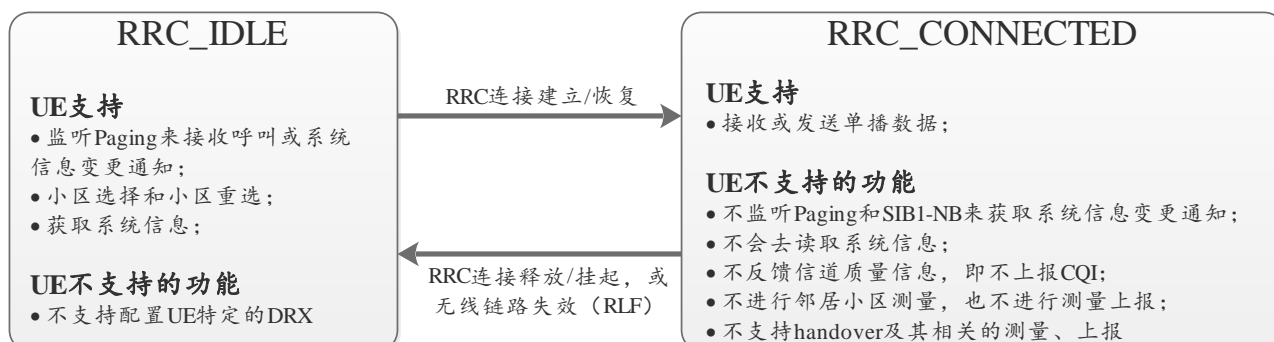


图 8-1: UE 的 RRC 状态以及它们的切换

与 LTE 类似，NB-IoT 中，SRB0 用于传输使用逻辑信道 CCCH 的 RRC 消息；SRB1 用于传输使用逻辑信道 DCCH 的 RRC 消息和 NAS 消息。NB-IoT 中并不存在 SRB2，但定义了新的无线承载 SRB1bis。SRB1bis 在 RRC 连接建立流程中，隐式地与 SRB1 一起建立并使用相同的配置。但 SRB1bis 的逻辑信道 ID 为 3，且 SRB1bis 不存在 PDCP。在安全激活之前，SRB1bis 会取代 SRB1 的角色用于传输 RRC 消息。但在安全激活之后，SRB1bis 就不再使用了，此时 SRB1bis 会自动变为 SRB1，并用于后续的 RRC 消息传输。这隐式地说明了对控制面 CIoT EPS 优化而言，只会使用 SRB1bis 而不会使用 SRB1，这是因为控制面 CIoT EPS 优化是不会进行安全激活的。

在 Rel-13 中，NB-IoT 只支持表 8-1 中列出的连接控制流程。（见 36.331 的 Table 5.3.1.4-1）

表 8-1: NB-IoT UE 支持的连接控制流程

流程	36.331 中的相关章节
Paging	5.3.2
RRC connection establishment	5.3.3
RRC connection resume（只支持控制面 CIoT EPS 优化的 UE 不支持此流程）	
Initial security activation（只支持控制面 CIoT EPS 优化的 UE 不支持此流程）	5.3.4
RRC connection reconfiguration（只支持控制面 CIoT EPS 优化的 UE 不支持此流程）	5.3.5
RRC connection re-establishment（只支持控制面 CIoT EPS 优化的 UE 不支持此流程）	5.3.7
RRC connection release	5.3.8
RRC connection release requested by upper layers	5.3.9
Radio resource configuration	5.3.10
Radio link failure related actions	5.3.11

## 8.1 RRC 连接建立

NB-IoT 中的 RRC 连接建立（RRC Connection Establishment）流程与 LTE 类似，但携带的内容有所不同：

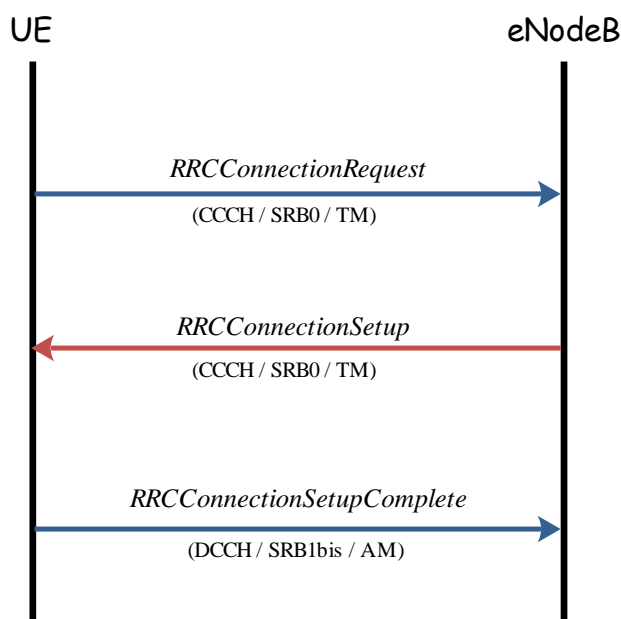


图 8-2：RRC 连接建立：成功

RRC 连接建立流程用于建立一条 RRC 连接，其中包含了 SRB1（针对 NB-IoT，还包括 SRB1bis）的建立，同时，该流程还被 UE 用于发送初始的 NAS 专用信息/消息。

UE 通过在上行逻辑信道 UL-CCCH 上（SRB0）发送 *RRCConnectionRequest-NB* 来接入网络，并通过其中的 *establishmentCause-r13* 字段来告诉 eNodeB 其接入的原因：终端发起的信令、终端发起的数据、终止于终端的接入、异常上报和延迟容忍业务。此外，在 *RRCConnectionRequest-NB* 中还会携带 UE 的唯一标识（来自核心网的 S-TMSI，如果不存在 S-TMSI，则 UE 会生成一个随机数），以及指示 UE 是否具备支持 multi-tone 和 multi-carrier 传输的能力，具体见第 9 章关于“UE 能力”的介绍。

作为响应，eNodeB 会在下行逻辑信道 DL-CCCH 上（SRB0）发送 *RRCConnectionSetup-NB*，并在该消息中提供 SRB1（SRB1bis）、DRB（至多 2 个）的配置信息以及其它 UE 特定的配置（如 MAC/PHY 层配置）信息等。UE 收到该消息后，会同时建立 SRB1 和 SRB1bis。

最后，UE 会在上行逻辑信道 UL-DCCH 上（SRB1bis。此时 AS 安全还未激活，所以使用的是 SRB1bis）发送 *RRCConnectionSetupComplete-NB*，并在该消息中将其选择的 PLMN 和 MME 等信息发给 eNodeB，同时，该消息还可以携带 NAS 消息。对于支持控制面 CIoT 优化的 UE 而言，可在该 NAS 消息中携带用户数据。eNodeB 收到此消息后，会将其中的 NAS 消息转发给 MME。至此，RRC 连接建立流程就完成了。此外，UE 还会通过 *RRCConnectionSetupComplete-NB* 消息中的 *up-CIoT-EPS-Optimisation-r13* 字段来

告诉网络它是否支持用户面 CIoT EPS 优化（NB-IoT UE 默认支持控制面 CIoT EPS 优化，因此不需要告诉网络其是否支持控制面 CIoT EPS 优化）。

如果使用的是用户面 CIoT EPS 优化，在建立了 RRC 连接后，会以与 LTE 相同的方式进行安全激活和 RRC 连接重配置流程，协议同时还定义了 RRC 连接重建立流程。但对控制面 CIoT EPS 优化而言，这些流程都是不存在的。

## 8.2 RRC 连接释放/挂起

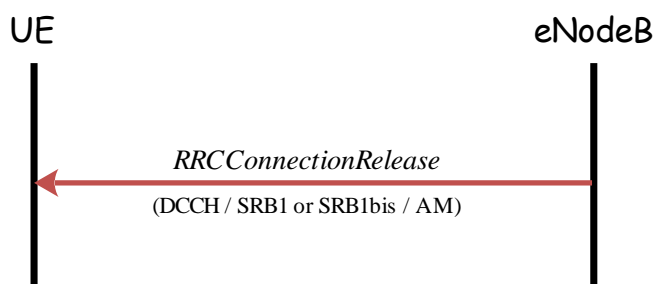


图 8-3: RRC 连接释放

RRC 连接释放流程是由 eNodeB 触发的。eNodeB 可以选择释放已建立的 RRC 连接、无线承载和无线资源；也可以选择挂起 RRC 连接，包括挂起已建立的无线承载。

对于用户面 CIoT EPS 优化而言，eNodeB 可以通过将下行逻辑信道 DL-DCCH 上（SRB1 或 SRB1bis）传输的 *RRCConnectionRelease-NB* 消息中的 *releaseCause-r13* 字段设置为 *rrc-Suspend* 来指示挂起 RRC 连接。在这种情况下，UE 会保存 AS 上下文以及来自 *RRCConnectionRelease-NB* 消息的 *resumeIdentity-r13*，并挂起所有的 SRB 和 DRB。如果以后需要，UE 会使用 8.3 节介绍的 RRC 连接恢复流程来恢复 RRC 连接。

如果是释放而不是挂起 RRC 连接，则 AS 上下文会被删除，且在需要时，UE 可以使用 8.1 节介绍的 RRC 连接建立流程来重新建立一条新的 RRC 连接。

在 RRC 连接释放/挂起后，UE 会进入 RRC\_IDLE 态。

### 8.3 RRC 连接恢复

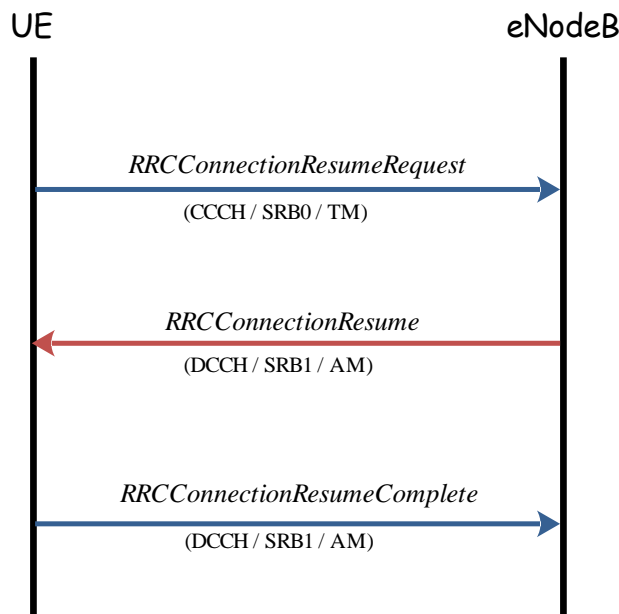


图 8-4: RRC 连接恢复: 成功

8.2 节介绍过，当 eNodeB 发起 RRC 连接释放流程时，它可能会要求 UE 挂起而不是释放 RRC 连接。此时 UE 会保存 AS 上下文以及来自 eNodeB 的 *resumeIdentity-r13*，挂起所有的 SRB 和 DRB，并切换到 RRC\_IDLE 态。当然，eNodeB 侧也会继续保留相关 UE 的 AS 上下文等信息。

当 UE 需要恢复 RRC 连接时，它会在上行逻辑信道 UL-CCCH 上（SRB0）发送 *RRCConnectionResumeRequest-NB*，并基于之前保存的 *resumeIdentity-r13* 来设置该消息中的 *resumeID-r13* 字段，以请求 eNodeB 恢复相应的 RRC 连接。如果 UE 在下行逻辑信道 DL-DCCH 上（SRB1）接收到响应的 *RRCConnectionResume-NB* 消息，则它会基于保存的 AS 上下文来恢复 RRC 连接，包括恢复无线承载（SRB/DRB），并使用更新后的 key 来激活安全等。当然，部分 AS 上下文是可以被修改的。

最后，UE 会在上行逻辑信道 UL-DCCH 上（SRB1）发送 *RRCConnectionResumeComplete-NB* 消息，将其选择的 PLMN 等信息发给 eNodeB，同时，该消息还可以携带 NAS 消息。至此，RRC 连接恢复流程就完成了。

RRC 连接恢复后，安全继续。RRC 连接恢复流程中不支持密钥更新。在 RRC 连接重建流程和 RRC 连接恢复流程中，短 MAC-I 继续被重用为认证令牌。eNodeB 还会在 *RRCConnectionResume-NB* 消息中提供 Next Hop Chaining Count (NCC)，并且 UE 会重置 COUNT。

RRC 连接恢复流程只适用于配置了用户面 CIoT EPS 优化的 UE，且该 UE 需要配置至少一个 DRB。该流程在发送非频繁的小数据包时，能够减少大量的信令开销（此时绝大部分配置可以通过保存的 AS 上下文来恢复，而不需要在信令中发送）。

当收到 *RRCConnectionResumeRequest-NB* 后，eNodeB 会决定是接受该请求（见图 8-4）还是回退到传统的 RRC 连接建立流程（即不接受该请求，见图 8-5）。后一种情况等同于重新建立一条 RRC 连接，且 UE 会释放之前存储的 AS 上下文，不再使用该上下文来恢复 RRC 连接。

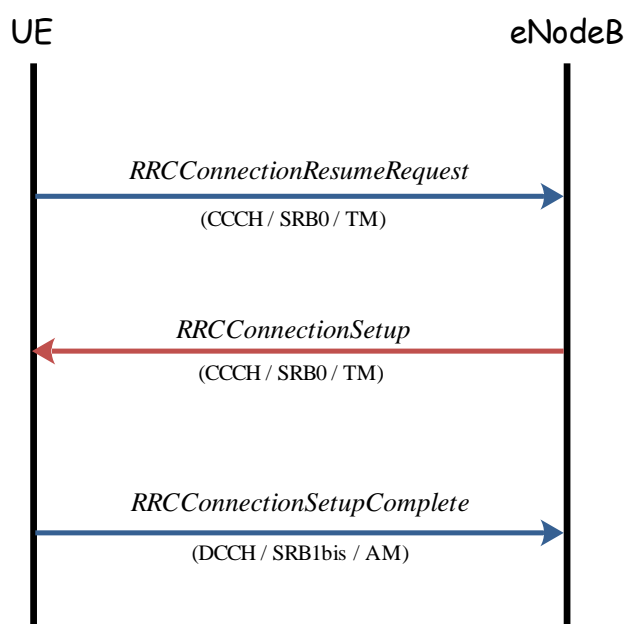


图 8-5: RRC 连接恢复：回退到 RRC 连接建立流程

前面我们已经针对接入侧的 RRC 连接挂起/恢复流程做了深入的分析。但实际上，RRC 连接的挂起/恢复不仅涉及接入侧的处理，eNodeB 还需要与核心网有一定的交互。并且存在跨 eNodeB 恢复 RRC 连接的场景。（见 36.300 的 7.3a.3 节）

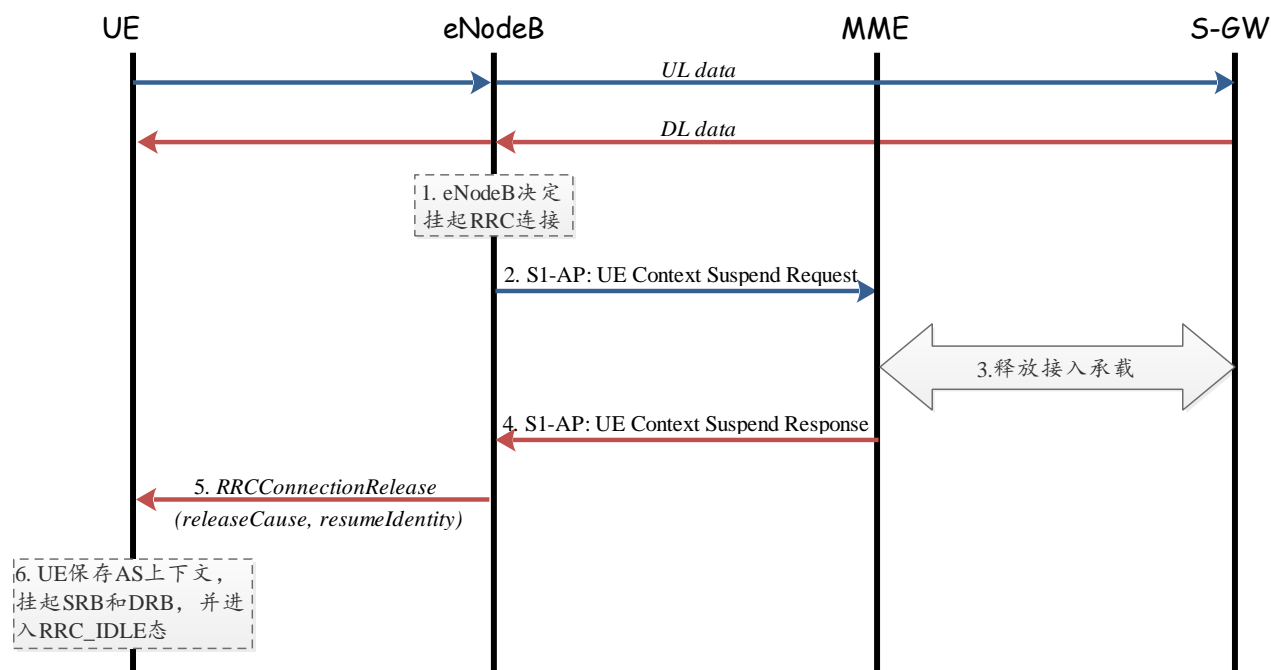


图 8-6: RRC 连接挂起流程

RRC 连接挂起的整体流程如图 8-6 所示：

1. 由于某些原因，如 UE inactivity timer（DRX 相关的定时器）超时等，eNodeB 决定挂起 RRC 连接。
2. eNodeB 通过发送 S1-AP UE Context Suspend Request 来通知 MME，RRC 连接正在被挂起。

3. MME 请求 S-GW 释放对应 UE 的所有 S1-U 承载。
4. MME 通过发送 S1-AP UE Context Suspend Response 来响应 eNodeB 的 UE 上下文挂起请求。
5. eNodeB 通过发送 *RRCCConnectionRelease-NB* 消息，并将其 *releaseCause-r13* 字段设置为 *rrc-Suspend* 来通知 UE 挂起 RRC 连接。该消息中还包含了会被 UE 保存的 Resume ID。
6. UE 保存 AS 上下文，挂起所有的 SRB 和 DRB，并进入 RRC\_IDLE 态。

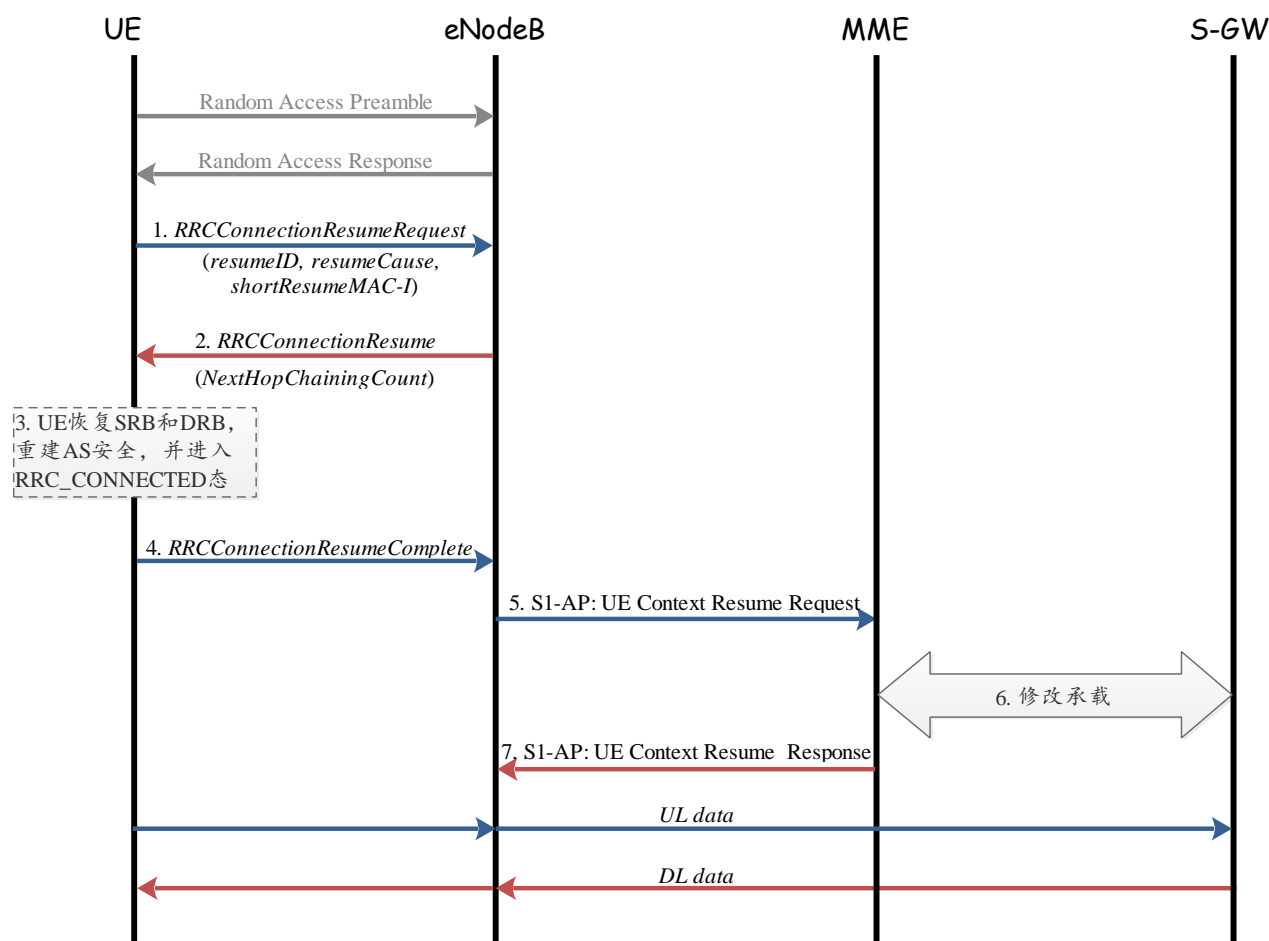


图 8-7: RRC 连接恢复流程

RRC 连接恢复的整体流程如图 8-7 所示:

1. 在某些时间点（如：当 UE 被寻呼、或上行有新的数据要发送时），UE 会通过向 eNodeB 发送 *RRCCConnectionResumeRequest-NB* 消息来恢复 RRC 连接。该消息中会包含 Resume ID、建立原因以及认证令牌。认证令牌会以与 RRC 连接重建过程中使用短 MAC-I 相同的方式来计算，以允许 eNodeB 验证 UE 的身份。
2. 如果 eNodeB 中存在对应的 Resume ID，并且认证令牌被成功验证是有效的，则 eNodeB 会通过向 UE 发送 *RRCCConnectionResume-NB* 消息来响应 RRC 连接恢复请求。在该消息中会包含重新建立 AS 安全所需的 Next Hop Chaining Count (NCC) 值。
3. UE 恢复所有的 SRB 和 DRB，重建 AS 安全，并进入 RRC\_CONNECTED 态。
4. UE 向 eNodeB 回应 *RRCCConnectionResumeComplete-NB* 消息来确认 RRC 连接已经成功恢复。
5. eNodeB 发起 S1-AP UE Context Resume Request 来通知 MME，UE 状态发生的变化。
6. MME 请求 S-GW 为 UE 激活 S1-U 承载。
7. MME 通过发送 S1-AP UE Context Resume Response 来响应 eNodeB 的 UE 上下文恢复请求。

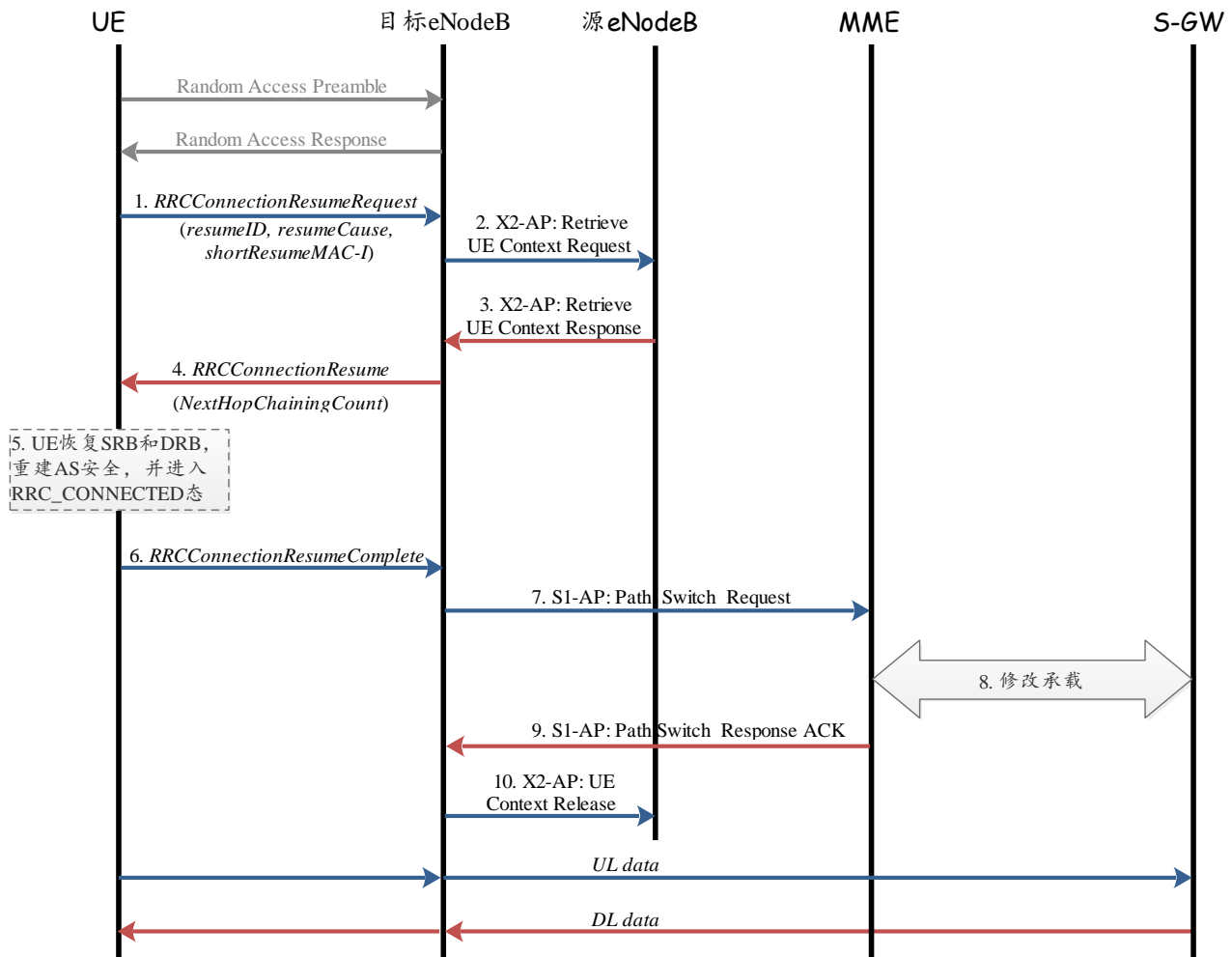


图 8-8: 在不同的 eNodeB 上进行 RRC 连接恢复流程

RRC 连接可以在与挂起该连接的 eNodeB（旧 eNodeB）不同的 eNodeB（新 eNodeB）上恢复。跨 eNodeB 间的 RRC 连接恢复需要通过上下文获取过程来处理，其中新的 eNodeB 会通过 X2 接口从旧的 eNodeB 上获取 UE 的上下文。新的 eNodeB 会将从 UE 得到 Resume ID 提供给旧的 eNodeB 以便其识别 UE 的上下文。

跨 eNodeB 的 RRC 连接恢复的整体流程如图 8-8 所示：

1. 在某些时间点（如当 UE 被寻呼、或上行有新的数据要发送时），UE 会通过向 eNodeB 发送 *RRCConnectionResumeRequest-NB* 消息来恢复 RRC 连接。该消息中会包含 Resume ID、建立的原因以及认证令牌。认证令牌会以与 RRC 连接重建过程中使用短 MAC-I 相同的方式来计算，以允许 eNodeB 验证 UE 的身份。
2. 新的 eNodeB 使用 Resume ID 来定位旧的 eNodeB，并通过 X2-AP Retrieve UE Context 流程来获取 UE 的上下文。
3. 旧的 eNodeB 给新的 eNodeB 响应与 Resume ID 相关联的 UE 上下文。
4. 如果 eNodeB 中存在对应的 Resume ID，并且认证令牌成功验证是有效的，则 eNodeB 会通过向 UE 发送 *RRCConnectionResume-NB* 消息来响应 RRC 连接恢复请求。在该消息中会包含为重新建立 AS 安全所需的 Next Hop Chaining Count (NCC) 值。
5. UE 恢复所有的 SRB 和 DRB，重建 AS 安全，并进入 RRC\_CONNECTED 态。

6. UE 向 eNodeB 回应 *RRCConnectionResumeComplete-NB* 消息来确认 RRC 连接已经成功恢复。
7. 新的 eNodeB 发起 S1-AP Path Switch 流程来建立到服务 MME 的 S1 UE 相关联的信令连接，并请求 MME 恢复 UE 上下文。
8. MME 请求 S-GW 激活 UE 的 S1-U 承载，并更新下行路径。
9. MME 通过发送 S1-AP Path Switch Response ACK 来响应步骤 7 中的 eNodeB 的 Path Switch 请求。
10. 在 S1-AP Path Switch 流程之后，新的 eNodeB 会通过 X2-AP UE Context Release 流程来触发旧的 eNodeB 去释放其保存的 UE 上下文。

其实跨 eNodeB 的 RRC 连接恢复流程增加的一些信令流程很好理解：UE 的挂起流程是在旧的 eNodeB 上进行的，在新的 eNodeB 上不存在该 UE 的上下文。UE 在新的 eNodeB 上发起 RRC 连接恢复流程时，新的 eNodeB 必然要去旧的 eNodeB 上获取该 UE 的上下文，因此需要 X2-AP: Retrieve UE Context 流程。UE 切换到新的 eNodeB，eNodeB 与核心网之间路径必然发生变化，因此需要 S1-AP: Path Switch 流程。UE 成功切换到新的 eNodeB 后，旧的 eNodeB 上就没必要保存 UE 的上下文了，因此需要 X2-AP UE Context Release 流程来触发旧的 eNodeB 去释放其保存的 UE 上下文。

## 8.4 RRC 连接拒绝

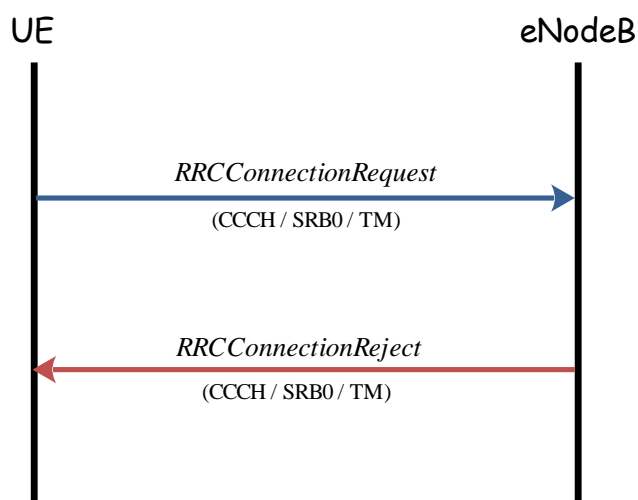


图 8-9: RRC 连接建立：网络拒绝



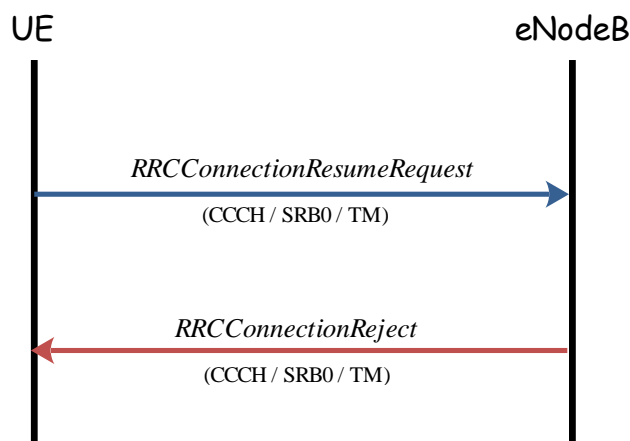


图 8-10: RRC 连接恢复：网络拒绝或失败

由于某些原因，如不再有空闲的资源等，RRC 连接建立请求或 RRC 连接恢复请求会被 eNodeB 拒绝，此时 eNodeB 会在下行逻辑信道 DL-CCCH 上（SRB0）回复一个 *RRCConnectionReject-NB* 消息。接下来，UE 必须等待 *RRCConnectionReject-NB* 消息中指定的时间（*extendedWaitTime-r13*）后才能再次发起连接请求。通过这种方式，eNodeB 能够避免过多 UE 同时进行网络连接等原因而带来的过度拥塞。如果拒绝的是一个 RRC 连接恢复请求，则 eNodeB 会指示 UE 是释放当前的 AS 上下文和 *resumeIdentity-r13*（当不存在 *rrc-SuspendIndication-r13* 字段时），还是继续保存该上下文以便用于后续的恢复请求（当存在 *rrc-SuspendIndication-r13* 字段时）。

## 8.5 初始安全激活和 RRC 连接重配置

初始安全激活流程和 RRC 连接重配置流程只适用于配置了用户面 CIoT EPS 优化的 UE。

在 RRC 连接成功建立之后，或者说，SRB1 和 SRB1bis 成功建立之后，eNodeB 会发起初始安全激活流程，以建立 AS 安全。初始安全激活流程如下：

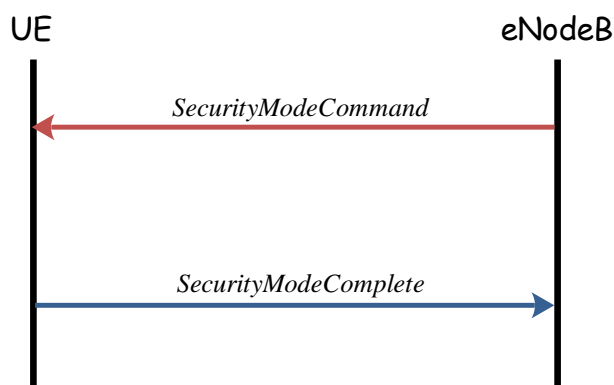


图 8-11: 初始 AS 安全激活流程

在 *SecurityModeCommand* 消息中，eNodeB 会向 UE 提供应用于 SRB1 和 DRB 的加密算法，以及用于 SRB1 的完整性保护算法。安全激活之后，SRB1bis 将自动变为 SRB1，且 SRB1 会用于后续的 RRC 消息传输。

安全激活之后，eNodeB 会通过 RRC 连接重配置流程来为 UE 建立 DRB 并更新各层的配置参数。

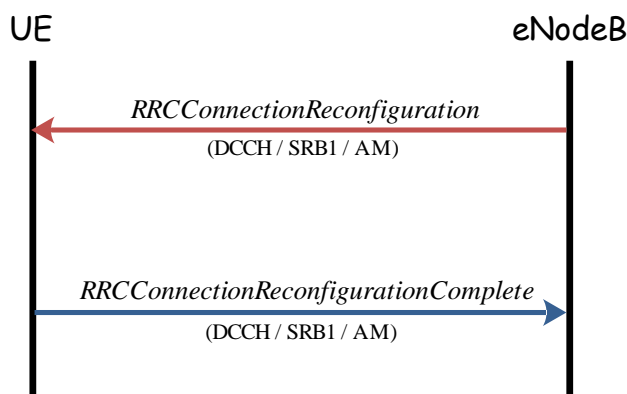


图 8-12: RRC 连接重配置流程（成功）

在 *RRCConnectionReconfiguration-NB* 中，eNodeB 会给 UE 提供与无线承载相关的配置参数，包括 RLC 和逻辑信道的配置。由于 SRB 只使用默认的 PDCP 配置，因此 eNodeB 只会为 DRB 配置 PDCP。与此同时，eNodeB 还会给 UE 配置 MAC 层和物理层的相关参数。

对于 RRC 连接恢复流程而言，*RRCConnectionResume-NB* 消息在 SRB1 上传输且已进行了完整性保护，并且该消息可以携带对 DRB 和各层参数进行重配的字段，因此对 NB-IoT 而言，RRC 连接恢复流程之后的 RRC 连接重配置流程是可选的。这样做的目的主要是为了减少 RRC 连接恢复过程中的信令开销，以降低终端的功耗。

如果 UE 无法执行 *RRCConnectionReconfiguration-NB* 消息中的部分或全部配置，如某些参数（IE）的内容出错或配置了 UE 不支持的功能等，则 UE 会继续使用接收到这条 RRC 连接重配置消息之前的配置。并且如果此时 UE 的安全还未激活，则 UE 会离开 RRC\_CONNECTED 态；否则 UE 会发起 RRC 连接重建流程。

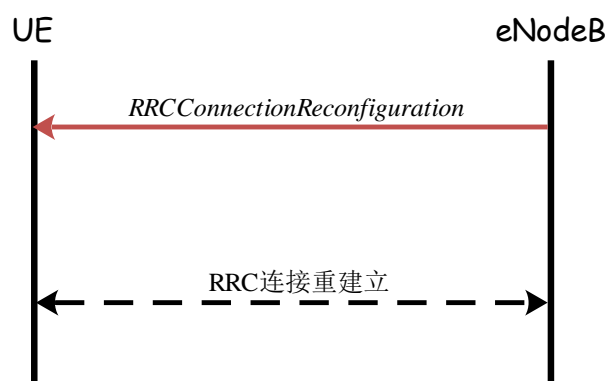


图 8-13: RRC 连接重配置流程（失败）

RRC 连接重配置流程中，要么执行所有的配置要求，要么所有的配置要求都不执行，不存在部分执行的情况。

## 8.6 RRC 连接重建

只有当 AS 安全已经激活后，UE 才可能发起 RRC 连接重建流程。也就是说，RRC 连接重建流程只适用于配置了用户面 CIoT EPS 优化的 UE。

在 NB-IoT 中，当发生以下几类事件之一时，UE 会触发 RRC 连接重建立流程：

1. 检测到无线链路失效（RLF）；
2. PDCP 层的完整性校验失败；
3. RRC 连接重配置失败。

由于 NB-IoT 不支持切换，因此 NB-IoT 不支持切换失败所触发的 RRC 连接重建立流程。

NB-IoT 中的 RRC 连接重建立流程与 LTE 类似，成功的流程如图 8-14 所示，失败的流程如图 8-15 所示。

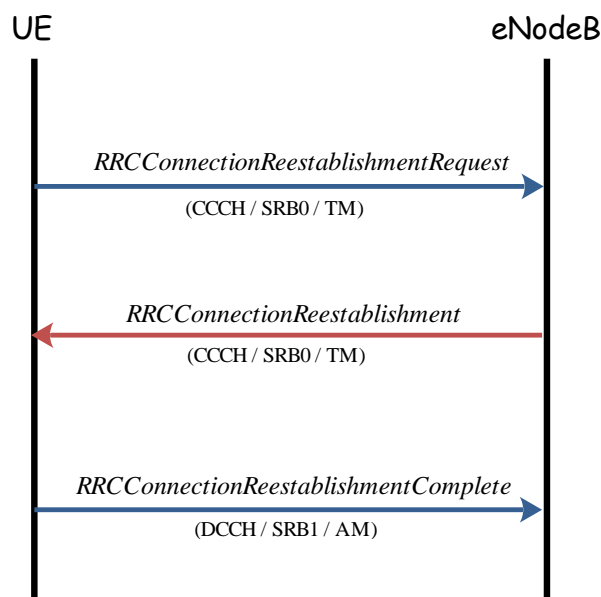


图 8-14: RRC 连接重建立流程（成功）

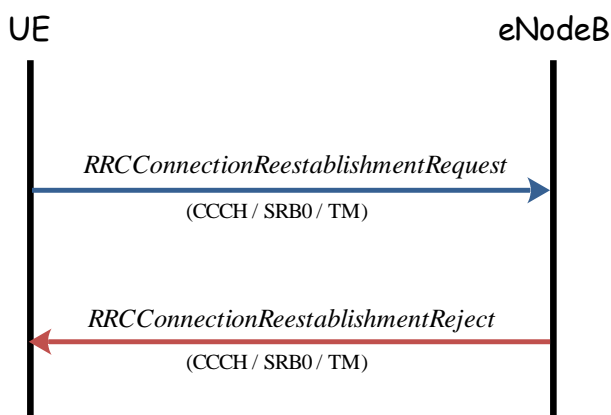


图 8-15: RRC 连接重建立流程（失败）

## 8.7 Access Barring（接入禁止）

Access Barring 参数在 SIB14-NB 上传输，并可以在任意时刻发生变化。Access Barring 参数的变更不会影响 MIB-NB 的 *systemInfoValueTag-r13* 字段，或 SIB1-NB 的 *systemInfoValueTagSI-r13* 字段。

NB-IoT UE 通过校验 MIB-NB 的 *ab-Enabled-r13* 字段来确定小区是否是能了 access barring。当 eNodeB 在 MIB-NB 中指示进行接入禁止检查（此时 *ab-Enabled-r13* 设置为 TRUE），并广播 SIB14-NB 时，UE 会在除（由网络发起的）终端终止的呼叫之外的原因引起的 RRC 连接建立/恢复流程之前，进行接入禁止检查。

商业上可用的 UE 具有接入类别（Access Class，简称 AC）0~9。在 SIB14-NB 中，存在关联的 bitmap（对应字段 *ab-BarringBitmap-r13*，共 10 比特，从左到右分别对应 AC 0~9），其每个比特对应一个接入类别。如果与接入类别相关联的比特被置位（设置为 1），则具有该接入类别的 UE 禁止访问该小区。被禁止接入的 UE 会等待 SIB14-NB 的更新以再次检查禁止状态。需要注意的是，取决于 SIB14-NB 的设置，可能会跳过某些特殊数据的访问限制检查。

#### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 3GPP TS 36.300 V13.7.0, 2016-12; Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2
- [3] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification

## 第9章 UE 能力

本章主要关注 UE 的接入侧能力及其上报过程。在 NB-IoT 中，UE 可能会在 3 个地方指示自己的能力。

一个 NPRACH 资源在频域上占用的子载波可通过 *nprach-SubcarrierMSG3-RangeStart-r13* 被分成 1 或 2 组，UE 通过选择其中一组来发送 preamble 以指示 Msg3 是否支持 multi-tone 传输。

在建立 RRC 连接时，NB-IoT UE 会在 *RRCConnectionRequest-NB* 消息中发送其是否支持 multi-carrier（通过 *multiCarrierSupport-r13* 指定）以及是否支持 NPUSCH 上的 multi-tone 传输（通过 *multiToneSupport-r13* 指定）的能力指示。其目的是为了让 eNodeB 在接下来的上行调度中，能够基于 UE 的这些能力给 UE 分配合适的上行资源。

这 2 种在随机接入过程的 Msg1 或 Msg3 中指示的 UE 能力内容非常有限。更多的 UE 能力信息是在 UE 能力查询流程中获得的。

当 UE 连接到网络时，eNodeB 既不知道 UE 建立在哪个 Rel 版本上，也不知道其支持哪些可选的特性。为了获取这些信息，定义了 UE 能力查询过程，如图 9-1 所示：

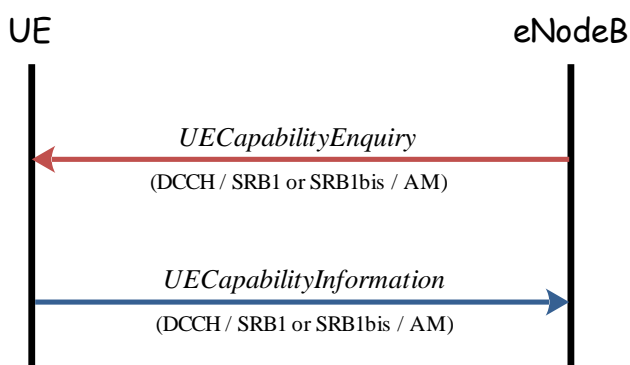


图 9-1: UE 能力查询流程

UE 能力查询总是由 eNodeB 发起，因为 UE 不知道 eNodeB 是否已经从核心网或从先前的会话中获取到了该信息。

NB-IoT UE 的能力信息包括 UE 所支持的 Rel 版本，UE 类别（UE category），支持的频带列表和建立多个承载（多个 DRB）的能力。此外，UE 可以指示其是否支持 multi-carrier，以及上行的 multi-tone 传输。其还可以包含 RoHC 上下文会话的最大数量和支持的 profile 等信息。

NB-IoT UE 的类别（category）为 Category NB1（Cat-NB1），并通过 *ue-Category-NB* 指定，其上下行物理层参数见 36.306 的 Table 4.1C-1 和 Tables 4.1C-2。主要参数包括下行支持的最大 TBS 为 680 比特，上行支持的最大 TBS 为 1000 比特，且上下行均不支持空分复用。NB-IoT UE 的层 2 缓存大小为 4000 字节（见 36.306 的 Table 4.1C-3）。同时 NB-IoT UE 只支持 FDD 半双工 type-B 模式（见 36.306 的 Table 4.1C-5）。

NB-IoT UE 的接入侧能力主要通过 *UE-Capability-NB-r13* 定义。该消息比 LTE 中对应的消息要小，这是因为有很多 LTE 特性是 NB-IoT 所不支持的，如载波聚合等。

UE-Capability-NB-r13 ::=	SEQUENCE {		
accessStratumRelease-r13	AccessStratumRelease-NB-r13,	----	Rel-13中，该值固定为rel13
ue-Category-NB-r13	ENUMERATED {nb1}	OPTIONAL,	---- UE category，Rel-13中总是包含此字段
multipleDRB-r13	ENUMERATED {supported}	OPTIONAL,	---- UE是否支持多个DRB。只有UE支持用户面CIoT EPS优化和ue-Category-NB时，才会应用该字段。在Rel-13中，如果UE支持多个DRB，那么UE只支持至多2个DRB同时传输
pdcp-Parameters-r13	PDCP-Parameters-NB-r13	OPTIONAL,	---- PDCP层参数
phyLayerParameters-r13	PhyLayerParameters-NB-r13,		---- 物理层参数
rf-Parameters-r13	RF-Parameters-NB-r13,		---- 射频参数
nonCriticalExtension	SEQUENCE { }	OPTIONAL	
}			
AccessStratumRelease-NB-r13 ::=	ENUMERATED {rel13, spare7, spare6, spare5, spare4, spare3, spare2, spare1, ...}		
PDCP-Parameters-NB-r13 ::=	SEQUENCE {		
supportedROHC-Profiles-r13	SEQUENCE {	----	指示UE支持哪种ROHC profile，只有UE支持用户面CIoT EPS优化和ue-Category-NB时，才会应用该字段
profile0x0002	BOOLEAN,		
profile0x0003	BOOLEAN,		
profile0x0004	BOOLEAN,		
profile0x0006	BOOLEAN,		
profile0x0102	BOOLEAN,		
profile0x0103	BOOLEAN,		
profile0x0104	BOOLEAN		
},			
maxNumberROHC-ContextSessions-r13	ENUMERATED {cs2, cs4, cs8, cs12}	DEFAULT cs2,	----指示RoHC上下文会话的最大数量
...			
}			
PhyLayerParameters-NB-r13 ::=	SEQUENCE {		
multiTone-r13	ENUMERATED {supported}	OPTIONAL,	----UE是否支持NPUSCH的multi-tone传输
multiCarrier-r13	ENUMERATED {supported}	OPTIONAL	----UE是否支持多载波部署
}			
RF-Parameters-NB-r13 ::=	SEQUENCE {		
supportedBandList-r13	SupportedBandList-NB-r13,		----UE支持的NB-IoT频带列表，取值范围见2.2节的介绍
multiNS-Pmax-r13	ENUMERATED {supported}	OPTIONAL	----UE是否支持NB-IoT小区广播NS-PmaxList的机制
}			
SupportedBandList-NB-r13 ::=	SEQUENCE (SIZE (1..maxBands)) OF SupportedBand-NB-r13		
SupportedBand-NB-r13 ::=	SEQUENCE {		

band-r13	FreqBandIndicator-NB-r13,	----	UE支持的NB-IoT频带
powerClassNB-20dBm-r13	ENUMERATED {supported}	OPTIONAL	----对应此频带，UE是否支持功率类别20dBm，见36.101。如果不包含此字段，则UE在此频带上支持功率类别为23dBm

}

#### 【参考资料】

- [1] 《Narrowband Internet of Things Whitepaper》，ROHDE & SCHWARZ
- [2] 3GPP TS 36.101 V13.6.1, 2017-01; User Equipment (UE) radio transmission and reception
- [3] 3GPP TS 36.331 V13.4.0, 2016-12; Radio Resource Control (RRC) Protocol specification

## 第10章 UE 节能技术

### 10.1 PSM

PSM (Power Saving Mode, 即省电模式) 是为了满足 IoT 和 M2M 的省电要求而在 Rel-12 中引入的, 其目标是使 UE 的功耗与其关机时几乎一致, 并低于空闲态的功耗。

PSM 是一种新的低功耗模式, 它允许 UE 不去监听周期性的寻呼 (Paging) 消息, 从而使得 UE 可以进入长时间的深度睡眠状态。当 PSM 激活时, UE 变得不可达, 即网络无法联系上 UE。因此, PSM 最好是由 UE 发起 (或者说终端发起) 或被调度的应用使用, 这些应用均由 UE 来发起与网络的通信。如果没有终端终止 (由网络侧发起) 的数据, IoT 设备可以保持在 PSM 状态长达 10 天左右, 其上限由跟踪区域更新 (TAU) 定时器的最大值确定。

PSM 旨在用于非频繁发生的终端发起的业务, 或终端终止但可以接受较大延迟的业务。PSM 可应用于包括智能电表、环境监控、传感器和定期将数据推送到网络的任何物联网设备中。PSM 适用于 Cat-0, Cat-M1 和 Cat-NB1 的 UE。

PSM 的原理其实很简单: UE 处于 PSM 态, 即休眠态的时候, 是不能接收和发送数据的, 网络侧也无法叫醒它 (这与 LTE 不同, LTE 中处于 IDLE 态的 UE, 网络侧可以通过 Paging 来唤醒它)。所以, 只有等 UE 自己主动醒来的时候 (如周期性地醒来, 或 UE 有上行数据要发送), 网络侧才能发送下行数据。

PSM 模式类似于关机, 但 UE 依然注册在网络中, 并保持了其非接入层 (NAS) 的状态, 因此 UE 在退出 PSM 状态时, 无需花费额外的时间来重新附着或重新建立 PDN 连接, 从而实现了更有效的 PSM 模式进入/退出机制。

UE 处于 PSM 状态时, 其接入层 (AS) 会被关闭, 这意味着基带和 RF 单元可以停止供电, 并且 UE 不必监听寻呼消息或执行任何无线资源管理 (RRM) 相关的测量。

PSM 是 NAS 层的功能, 并由 UE 发起的。当 UE 希望使用 PSM 时, 它会在附着 (Attach) 或 TAU (Tracking Area Updating) 流程中通过 *ATTACH REQUEST* 或 *TRACKING AREA UPDATE REQUEST* 消息向网络请求一个激活时间值 (Active Time value, 对应 IE: T3324 value) 来请求使用 PSM。如果网络支持 PSM 并接受 UE 使用 PSM, 则网络会通过给 *ATTACH ACCEPT* 或 *TRACKING AREA UPDATE ACCEPT* 消息给 UE 分配一个激活时间值 (对应 IE: T3324 value) 来确认接受 UE 使用 PSM。仅在网络提供的 T3324 值不等于 “deactivated” 时, UE 才会使用 PSM。



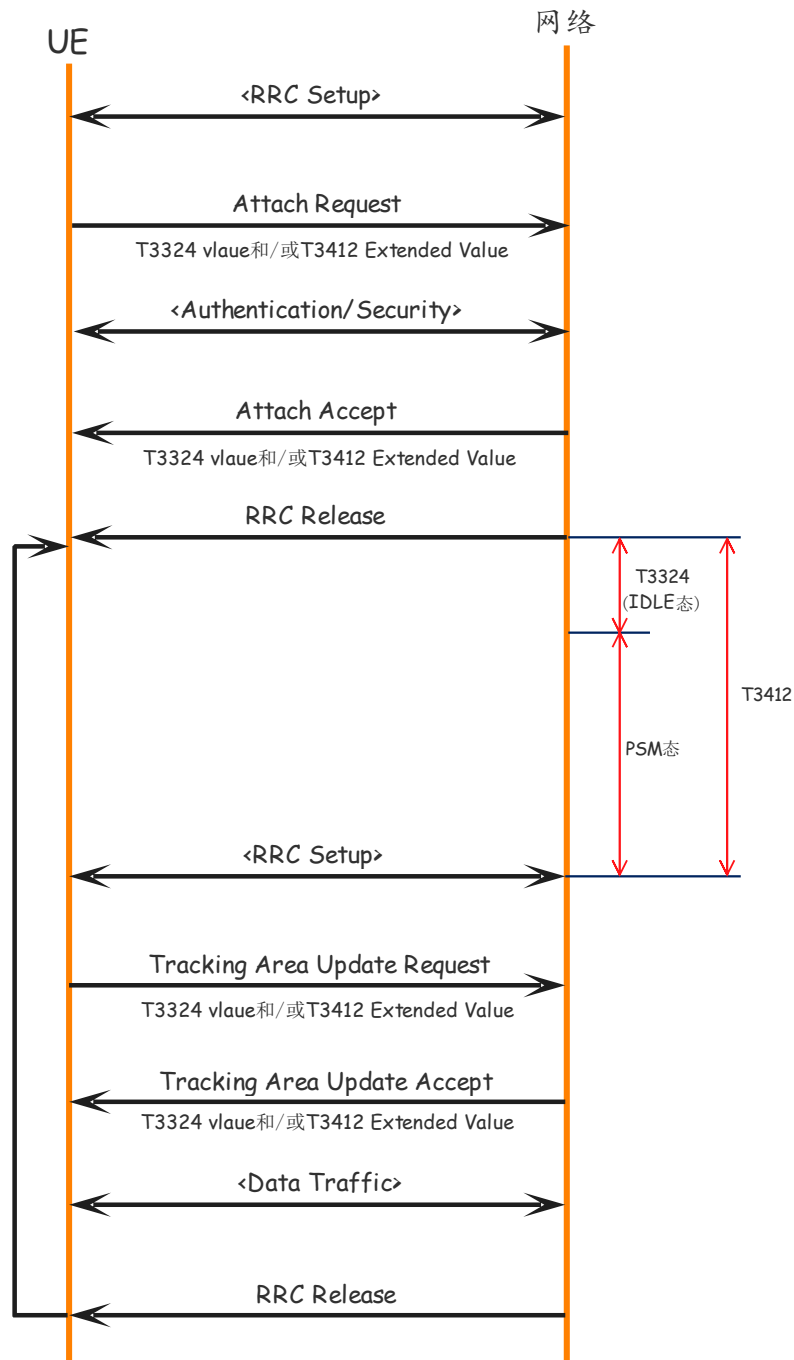


图 10-1: PSM 配置流程

定时器 T3324 指示了 UE 在进入 PSM 状态之前，在多长的时间内是下行可达的（此时 UE 处于 IDLE 态，并且会监听寻呼消息），如果该定时器超时，UE 将关闭 AS 层并激活 PSM（进入休眠态）。

定时器 T3412 是周期性的 TAU 定时器，它与 T3324 一起指示了 UE 在多长的时间内处于 PSM 状态，此时下行是不可达的。如果 UE 想改变周期性的 TAU 定时器值，UE 会在 TAU 过程中请求它希望的值。

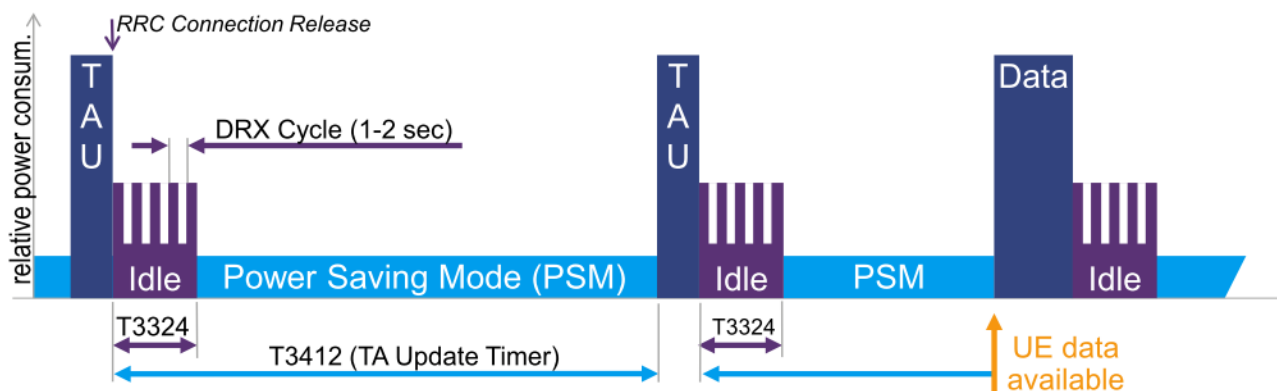


图 10-2: PSM

从图 10-2 可以看出，在周期性 TAU 完成、或数据传输结束后，UE 会先进入空闲态，使能非连续接收（DRX），并监听寻呼消息（此时终端终止业务可达）。当定时器 T3324 超时后，UE 才会进入 PSM 态。UE 会一直处于 PSM 态中，直到终端发起的事件（例如：周期性 TAU、UE 需要发送上行数据或去附着）要求 UE 向网络发起相关流程，UE 才会退出 PSM 态。

使用 PSM 的 UE 在其处于连接态期间以及在连接态之后的活动时间期间（定时器 T3324 运行期间的激活时段）可用于终止于终端的服务，而连接态是由在周期性 TAU 过程之后发生的由终端发起的诸如数据传输或信令传输触发的。对于终端终止（网络侧发起）的业务，其数据不能立即到达处于 PSM 状态的 UE，因此支持终端终止业务（由网络侧发起）并保证其时延对 PSM 来说是一个很大的挑战。随着周期性的 TAU 定时器超时或 UE 主动发起上行传输，设备才能变得网络可达，这可能导致终端终止业务显著的延时。虽然周期性的 TAU 可以配置为更频繁地发生以满足 UE 的延迟要求，但这样的配置会导致不必要的周期性 TAU 过程带来的附加信令开销，并增加设备功耗。为了解决 PSM 的这个缺点，在 Rel-13 中引入了扩展非连续接收（eDRX），这将在下一节介绍。

PSM 的详细描述可参见 24.301 的 5.3.11 节、23.401 的 4.3.22 节和 23.682 的 4.5.4 节的介绍。

## 10.2 eDRX

虽然 Rel-12 中引入的省电模式（PSM）可以有效地降低终端发起或被调度应用的功耗，但是其对于终端终止应用的支持是一个很大的挑战，并可能带来不必要的信令和功耗开销。为了解决这个问题，Rel-13 引入了扩展的非连续接收（eDRX, extended Discontinuous Reception）来增强连接态的非连续接收（Connected mode DRX, C-DRX）和空闲态的非连续接收（Idle mode DRX, I-DRX），以进一步增强对终端终止应用的支持，例如，设备可以被网络寻呼。eDRX 可用于 Cat-M1 和 Cat-NB1 的 UE，除了能降低实现的复杂度外，还能够节省额外的功耗。

空闲态下的 DRX 周期即为 Paging 周期。传统的 LTE 空闲态 DRX 周期被限制为至多 2.56 秒

（defaultPagingCycle 的最大取值为 2560ms）。eDRX 通过将空闲态下监听 Paging 消息的时间间隔（即 Paging 周期）和 TAU 的间隔延长到 Cat-M1 的最大 430.39 分钟和 Cat-NB1 的最大约 3 小时，来优化电池的使用时间。空闲态下的 eDRX 处理见 7.4.2 节的介绍。

传统的 LTE 连接态 DRX 周期被限制为至多 2.56 秒（指示的最大 DRX 周期取值为 2560ms）。eDRX 通过将连接态下监听控制信道（NPDCCH）/数据接收的最大时间间隔延长到 10.24 秒（NB-IoT 下为 9.216 秒），来优化电池的使用时间。连接态下的 eDRX 处理见 3.2.3.5 节的介绍。

eDRX 要求网络和 UE 同步休眠时段，从而使得 UE 不必频繁地检查网络消息。eDRX 周期的选择取决于 UE 所需的时延，并可以通过配置更长的 eDRX 周期来实现更低的功耗。除了降低功耗之外，与传统的 DRX 和/或 PSM 相比，eDRX 还可以减少信令负载（eDRX 无需额外的信令即可快速进入连接态）。

eDRX 的基本概念听起来很简单，仅仅是通过扩大休眠的持续时间来降低功耗。但实现起来却不能简单地给 Paging Cycle 和连接态下的 DRX Cycle 配置更大的值。如果希望 DRX 正确地运作，需要 UE 侧和网络侧之间精确的时间同步。传统 LTE 的 SFN 最大只能表示 1024 个系统帧，即最大只能表示 10240ms（10.24 秒），不足以表示空闲态 eDRX 下最大约 3 小时的休眠时间，为此，引入了超帧（Hyter Frame）的概念。

从本质上讲，PSM 和 eDRX 都是通过提高“睡眠”时间的占比来达到省电的效果，但这又牺牲了数据的实时性。与 PSM 相比，eDRX 的实时性更好，但省电效果差些。二者可满足不同的场景需求，如 PSM 更适用于智能抄表业务，而 eDRX 更适合宠物跟踪类业务等。

#### 【参考资料】

- [1] [http://www.sharetechnote.com/html/Handbook\\_LTE\\_PSM.html](http://www.sharetechnote.com/html/Handbook_LTE_PSM.html)
- [2] [http://www.sharetechnote.com/html/Handbook\\_LTE\\_eDRX.html](http://www.sharetechnote.com/html/Handbook_LTE_eDRX.html)
- [3] <http://www.eleven-x.com/2015/04/29/lte-cat-0s-power-saving-mode-what-it-could-mean-for-cellular-iot/>
- [4] <http://www.ednchina.com/news/article/201612280003>
- [5] [http://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-1/MME/21-1-MME-Admin/b\\_21\\_1\\_MME\\_Admin\\_chapter\\_010000.html](http://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-1/MME/21-1-MME-Admin/b_21_1_MME_Admin_chapter_010000.html)
- [6] 《Emerging communication technologies enabling the Internet of Things (IoT) 》， AK EMARIEVBE, ROHDE&SCHWARZ
- [7] 《LTE and 5G Technologies Enabling the Internet of Things》， 5G Americas
- [8] 3GPP TS 24.301 V13.8.0, 2016-12; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3 (Release 13)
- [9] 3GPP TS 23.401 V13.9.0, 2016-12; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access (Release 13)
- [10] 3GPP TS 23.682 V13.8.0, 2016-12; Architecture enhancements to facilitate communications with packet data networks and applications (Release 13)

## 第11章 NB-IoT 在 Rel-14 和 5G 的演进

NB-IoT 在 2016 年 6 月已经写入了 3GPP Rel-13 标准，它满足了改善室内覆盖和支持极大数量的低吞吐量设备接入的需求。但 IoT 市场的应用场景众多，Rel-13 中定义的 NB-IoT 无法满足某些特定的场景需求，如支持定位、非实时语音等，同时客户可能希望进一步地降低 IoT 设备的成本，如降低发射功率等。基于市场期望的推动，在 Rel-14 中，NB-IoT 将被进一步增强，以支持更多的功能，如：定位和多播等。

### 11.1 Rel-14 对 NB-IoT 的增强

在接入侧，Rel-14 将在以下几个方面对 NB-IoT 进行增强。

#### 11.1.1 定位

NB-IoT 中的定位功能可应用于资产跟踪和宠物定位等场景。目前主要考虑几种定位方案：ECID

（Enhanced Cell ID）、上行到达时间差（Uplink Time Difference of Arrival, UTDOA）和观察到的到达时间差（Observed Time Difference Of Arrival, OTDOA）。在研究中，不仅要考虑定位的精确性，还要考虑对网络/UE 的复杂性和功耗的影响，以便选择合适的方案。

基于 ECID 的定位是 Rel-9 中引入的定位技术。在这种技术中，UE 会向网络上报服务小区 ID、测量的参考信号接收功率/参考信号接收质量（RSRP / RSRQ）以及测量的接收和发送之间的时差等信息，再加上 eNodeB 侧的一些信息（如 eNodeB 的地理位置信息等），网络能够对 UE 进行定位。这种方案实现简单但定位精度不高。

在 OTDOA 中，UE 会测量来自多个基站的下行信号以实现 UE 的定位。而在 UTDOA 中，需要在多个基站测量 UE 的上行信号以定位 UE 的位置。

#### 11.1.2 多播增强

Rel-14 中，NB-IoT 考虑新增对 SC-PTM（Single Cell Point-to-Multipoint，单小区点对多点）的支持，以支持下行多播传输（如支持固件或软件更新，组消息传递等）。针对 NB-IoT 的窄带特性，协议还将考虑对 NPDCCH 和覆盖进行增强以便更好地支持多播传输。

#### 11.1.3 Non-anchor 上的 PRB 增强

该功能增强主要是为了支持在 non-anchor PRB 上传输 NPRACH 和寻呼消息，以增加 NB-IoT 的随机接入和寻呼容量。

#### 11.1.4 移动性和服务连续性增强

在 Rel-13 中，NB-IoT 不支持连接态下的移动性管理。而在 Rel-14 中，将考虑增强连接态下的移动性，以提高服务的连续性。同时在不增加 UE 功耗的前提下，避免控制面（CP）和用户面（UP）二者的非接入层（NAS）恢复。

#### 11.1.5 新功率类型

为了支持适合于形状受限的小型电池（如纽扣电池）的具有较低最大发射功率以及较小 MCL 的设备（如可穿戴设备），Rel-14 将对引入新的 UE 功率类别进行评估，并在必要时指定一个具有 14dBm 级别的新的 UE 功率类别。如果需要，还将引入必要的信令以支持该功率类别。

除了上面的增强外，Rel-14 中还将支持更大的 TBS（下行最大 TBS 为 1352 比特，上行最大 TBS 为 1800 比特），并且在上下行分别支持最多 2 个 HARQ process。这将使得 Rel-14 中的 NB-IoT UE 能够支持更高的速率，并减少延迟和功耗。

## 11.2 5G 对 NB-IoT 的增强

mMTC（massive Machine Type Communications）是 5G 的 3 大主要应用场景之一（另外 2 大应用场景为 eMBB 和 URLLC）。这意味着从 5G 标准制定的开始，就将 IoT 考虑在内了。对于 5G 下的 IoT，将在电池寿命（考虑支持至多 15 年的使用寿命）、连接密度（城市环境下，1 000 000 设备/km<sup>2</sup>）等方面进行加强。

## 第12章 MTC 和 eMTC

### 12.1 MTC

在 Rel-12 中，为了降低终端功耗和复杂度，引入了类型为 Cat-0 (Category 0) 的 UE，以支持 MTC (Machine-Type Communications, 机器类型通信) 数据传输。Cat-0 UE 支持：(1) 降低的数据速率，单播的最大 TBS (传输块大小) 为 1000 比特，广播 (Paging/系统消息/RAR MAC PDU) 的最大 TBS 为 2216 比特；(2) 支持新的拥有宽裕切换时间的 FDD 半双工类型；(3) 降低了数据速率能力的单接收天线；(4) 可选的 MBMS 支持：PMCH 的最大 TBS 为 4584 比特；(5) 最大速率为 1 Mbps。

MTC 降低了基带中数据信道的下行信道带宽 (尽管仅通过隐式方式限制)，而下行控制信道 (PCFICH/PHICH/PDCCH) 仍然允许使用整个 LTE 载波带宽。上行信道带宽和上下行 RF 的带宽保持与传统的 LTE UE 相同。对于广播流量，MTC 没有对 UE 的资源分配大小做明显的限制；但对于单播流量，限制了最大 TBS (上下行均为 1000 比特)；

- 对于 PDSCH 上的单播传输，Category 0 UE 支持的最大 TBS 为 1000 比特；
- 使用 SI-RNTI 加扰的系统信息，使用 P-RNTI 加扰的 Paging 以及使用 RA-RNTI 加扰的 RAR MAC PDU，支持的最大 TBS 为 2216 比特；
- 如果一个 Category 0 UE 被调度，且下行传输超过了上述限制，则一个 TTI 内的这些下行传输之间的优先级取决于 UE 的实现；
- 用于单播的 soft buffer 比特数为 25334；
- 低复杂度的 MTC UE 可选择支持 eMBMS。这种情况下，它必须支持接收 TBS 最大为 4584 比特的 MBMS 传输。
- MTC UE 在上下行均只支持单层传输，不支持空分复用。

Category 0 走出了降低 IoT 设备的复杂度和成本的一步。Rel-13 定义了新的 UE 类别 (Category M1 和 Category NB1)，进一步降低了 IoT 设备的复杂度和成本，并提高了覆盖和电池使用时间。因此，许多运营商商会选择直接部署 Rel-13 中的 eMTC 和/或 NB-IoT，并跳过 Category 0 的部署。

### 12.2 eMTC

3GPP 在 Rel-13 中引入了新的 UE 类型 Cat-M1 以支持 eMTC (enhanced Machine-Type Communications, 增强型机器类型通信) 传输。eMTC 是针对 Rel-12 中定义的 MTC 的直接扩展，主要目标是进一步降低设备复杂度和成本，扩大覆盖范围和延长电池寿命。eMTC 的标准化工作已于 2016 年第一季度完成。

虽然 Cat-0 UE 降低了上下行峰值速率需求，但依然需要支持所有指定的 LTE 系统带宽 (从 1.4MHz 至 20MHz)，并使用 6 至 100 个资源块 (RB)，且控制信道需要跨越整个带宽。而 eMTC 将带宽限制在 1.08MHz (6 个 RB)，因此 Cat-M1 UE 的基带和 RF 只需支持 1.08MHz 的系统带宽，从而降低了设备的复杂度和成本。

Cat-M1 UE 在上下行的吞吐量最高可达到 1Mbps。用于公共控制消息（系统消息/Paging/RAR）的最大传输块大小（TBS）从 Cat-0 的 2216 比特减少到 1000 比特（即与单播数据相同）。峰值速率的降低能够节省 UE 的处理时间和内存消耗。同时通过减少支持的下行传输模式（TM 模式）的数量以及放宽对无线链路质量测量和报告的要求，来进一步降低 UE 的复杂度。

Cat-0 UE 需要支持 23 dBm（200mW）的上行最大发射功率，但 Cat-M1 UE 可选择支持 23 dBm 或 20 dBm 的功率等级。20 dBm 的最大发射功率允许 UE 集成功率放大器（PA），而不是使用专用的 PA，这将进一步降低设备成本。通过 eMTC 引入的覆盖增强技术可以弥补较低发射功率所导致的上行覆盖范围降低。

eMTC 可以部署在常规的 LTE 载波内，并与其它 LTE 服务共存。由于 Cat-M1 只支持 1.08MHz 的带宽，因此引入了一个新的下行控制信道 MPDCCH（类似于 EPDCCH）来代替传统的控制信道

（PCFICH/PHICH/PDCCH 已不适用于 eMTC）。Cat-M1 设备利用了原有的 LTE 载波上位于中心 6 个 PRB 的 LTE 同步信号（PSS/SSS）和 PBCH，并引入了新的系统信息（SIB1-BR）。

eMTC 可在 LTE 载波内的任意区域为窄带 PDSCH 和用于数据调度的 MPDCCH 配置多个窄带区域，每个窄带区域包含 6 个 PRB。通过在时域和频域上复用 IoT 和非 IoT 数据，LTE 网络可以为 eMTC 动态地扩展资源分配，并提供灵活的容量来满足 IoT 需求。Cat-M1 设备允许跳频以实现在整个 LTE 载波上的频率分集。RACH 可在由 SIB1-BR 指定的窄带区域上传输。

为了将覆盖范围扩展到部署在室内或较远位置的 IoT 设备，eMTC 增加了 15dB 的链路预算，并将 MCL 从 LTE 的 140.7dB 提高到 155.7dB 或更高。用于 eMTC 的所有信道和信令的设计均满足 MCL 的目标。

基于覆盖范围与传输效率和延迟之间的折衷，增强覆盖的关键技术包括 TTI bundling 和重复传输。在早期版本中，TTI bundling 仅用于 LTE 上行传输，但在 eMTC 中，TTI bundling 的应用进一步扩展。TTI bundling 被应用于 eMTC 的 PRACH 和下行信道/信令中，如 PDSCH/MPDCCH/PBCH/MTC\_SIB。对于上下行数据业务信道，通过持久性的分配来确定 bundling 大小，这些分配是在连接建立期间确定的，并且可以通过事件驱动的反馈进行更新。同时引入了一种异步 HARQ 的 timing 关系，用于上下行的绑定。

eMTC 可延长电池寿命，目标是通过 5 瓦特电池为某些 IoT 流量模式和覆盖需求提供近 10 年的运行。Cat-M1 终端的功率效率提高通过基带和 RF 两者的较窄带宽操作实现。减少处理要求，例如较低的数据速率和功率效率的信道反馈也降低功耗。Rel-13 中引入的 eDRX 功能，进一步提高了电池寿命。

总而言之，Cat-M1（eMTC）使用仅 1.08MHz 的带宽可提供高达 1Mbps 的数据速率。eMTC 支持全双工 FDD，半双工 FDD 和 TDD 模式，可以部署在任何 LTE 频谱内。Cat-M1 还可以支持语音（VoLTE）和完全或有限的移动性，并且能够与常规 LTE 流量（Cat-0 及以上）完全共存。

## 12.3 eMTC 与 NB-IoT 的比较

eMTC 与 NB-IoT 的比较如表 12-1 所示。

表 12-1: eMTC 和 NB-IoT 的比较

技术标准	eMTC (Cat-M1)	NB-IoT (Cat-NB1)
部署方式 (deployment)	LTE 带内部署	带内部署、 <b>保护频带部署</b> 和 <b>独立部署</b>
下行	OFDMA (15kHz), Turbo code, 16QAM, 1RX	OFDMA (15kHz), TBCC, QPSK, 1RX
上行	SC-FDMA (15kHz) , Turbo code, 16QAM	SC-FDMA (15/3.75kHz) Turbo code, QPSK
带宽	1.08MHz	180kHz
UE 接收带宽	1.4MHz	200kHz
双工模式 (duplex mode)	半双工 FDD(type-B)、全双工 FDD 和 TDD	FDD 半双工 type-B
峰值速率	UL: 1Mbps DL: 1Mbps	UL: 250kbps DL: 226.7kbps
覆盖 (MCL)	155.7dB	164dB
功率类别	23dBm, 20dBm	23dBm, 其它类别待定
移动性	既支持空闲态下的小区重选, 也支持 连接态下的小区切换	支持空闲态下的小区重选, 但不支持连接态 下的小区切换
省电	PSM, eDRX	PSM, eDRX
地理定位 (Geo- positioning)	支持定位功能	Rel-13 不支持; 但将在 Rel-14 中加入定位 功能
语音	将在 Rel-14 中支持 VoLTE	不支持
连接数	协议未做专门优化, 其 UE 连接数小 于 NB-IoT	50000 UE/小区

NB-IoT 在覆盖、功耗、成本和 UE 连接数等方面占优, 但其速率较低且无法满足移动性要求, 并且不支持语音业务, 因此其比较适合于低速率且位置固定或移动很慢的 LPWA 应用, 如智能抄表等。

eMTC 在峰值速率、移动性以及语音业务方面要强于 NB-IoT, 但其覆盖、成本和 UE 连接数等方面要弱于 NB-IoT。因此 eMTC 适用于中等速率, 且对移动性或语音有较高需求的物联网应用, 如智能手表等。

运营商可根据具体的需求来选择部署对应的物联网技术。相对而言, eMTC 的 ARPU 值更高。

#### 【参考资料】

- [1] 《LTE and 5G Technologies Enabling the Internet of Things》, 5G Americas