# LoRa 技术问题解答

**1.什么是 LoRa?**

LoRa 是低功耗广域网通信技术中的一种，是 Semtech 公司采用和推广的一种基于扩频技术的超远距离无线传输技术，是 Semtech 射频部分产生的一种独特的调制格式。

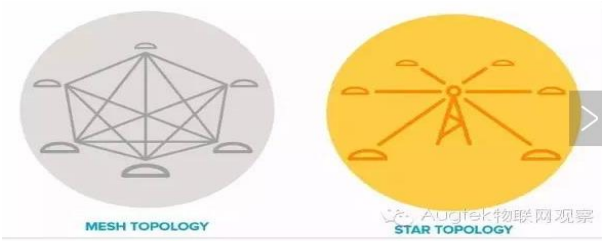
LoRa 射频部分的核心芯片是 SX1276 和 SX1278。这类芯片集成规模小、效率高，为 LoRa 无线模块带来高接收灵敏度。而网关芯片则采用的是集成度更高、信道数更多的 SX1301。用 SX1301 作为核心开发出的 LoRa 网关，可以与许许多多的 LoRa 模块构成多节点的复杂的物联网自组网。

**2. LoRa 是扩频技术吗?**

LoRa 是一种扩频技术，但它不是直接序列扩频。直接序列扩频通过调制载波芯片来传输更多的频谱，从而提高编码增益。而 LoRa 调制与多状态 FSK 调制类似，使用未调制载波来进行线性调频，使能量分散到更广泛的频段。

**3. LoRa 是 Mesh 网络、点对点传输还是星形网络?**

LoRa 调制技术本身是一个物理层（PHY layer）协议，能被用在几乎所有的网络技术中。Mesh 网络虽然扩展了网络覆盖的范围，但是却牺牲了网络容量、同步开销、电池使用寿命。随着 LoRa 技术链路预算和覆盖距离的同时提升，Mesh 网络已不再适合，故采用星形的组网方式来优化网络结构、延长电池寿命、简化安装。**LoRa 网关和模块间以星形网方式组网，而 LoRa 模块间理论上可以以点对点轮询的方式组网，当然点对点轮询效率要远远低于星形网。**




**4. LoRa 技术与 SIGFOX，NWave 的区别在哪里?**

总的来说，LoRa 技术采用的是一种扩频技术；SIGFOX 公司使用窄带 BPSK 调制技术；NWave 公司使用 Weightless 标准，与 SIGFOX 公司使用的技术较为相似。想要了解详细参数，可在菜单干货数据栏查看“**LPWAN 技术比较**”。

目前使用超窄带技术的公司可供选择的收发器芯片较多，而 LoRa 仅能使用 Semtech 提供的芯片

**5. 什么是 LoRa 网关?**

LoRa 网关位处 LoRa 星形网络的核心位置，是终端和服务器（Server）间的信息桥梁，是多信道的收发机。LoRa 网关有时又被称为 LoRa 基站或 LoRa 集中器，虽然定义不同，但其实是同一含义

LoRa 网关使用不同的扩频因子，不同的扩频因子两两正交因而理论上可以在同一信道中对多条不同扩频因子的信号进行解调。网关与网络服务器间通过标准 IP 进行连接，终端通过单跳与一个或多个网关进行通讯，所有的终端通讯都是双向通讯，同时也支持软件远程升级等。

目前来说，定义不同，网关类型也不同。例如，按照应用场景不同可分为为室内型网关和室外型网关；按照通讯方式不同可分为全双工网关和半双工网关；而按照设计标准不同可分为完全符合 LoRaWAN 协议网关和不完全符合 LoRaWAN 协议网关。**AUGTEK 新一代网关为室外型，全双工，并且完全符合 LoRaWAN 协议。完全符合 LoRaWAN 协议的 LoRa 网关及 LoRa 终端能够实现互联互通，这具有很大意义！**

**6. LoRa 网关的容量有多大？单个网关能连多少个终端?**

网关容量是指在一定时间内网关接收数据包数量的能力。**理论上来说，单个 SX1301 芯片拥有 8 个信道，在完全符合 LoRaWAN 协议的情况下最多每天能接收 1500 万个数据包。**如果某应用发包频率为 1 包/小时，单个 SX1301 芯片构成的网关能接入 62500 个终端节点。当然，这只是一个理论值，网关接入终端数量最终还是与网关信道数量、终端发包频率、发包字节数和扩频因子息息相关。

**7. LoRa 网关接入的节点数目取决于哪些因素?**

LoRa 网关接入的节点数取决于 LoRa 网关所能提供的**信道资源**以及单个 LoRa 终端占用的信道资源。LoRa 网关如果采用 Semtech 标准参考设计，网关采用 SX1301 芯片，那么信道数是固定的 **8 个上行信道 1 个下行信道**。物理信道数确定了， LoRa 网关所能提供的信道资源也就确定了。**（网关设计不同，信道数不同，AUGTEK 网关能实现 8 个上行，4 个下行。）**单个 LoRa 终端占用的信道资源与终端**占用信道**的时间一致，也就与终端的**发包频率**、**发包字节数**以及 LoRa 终端的**扩频因子**息息相关。当 LoRa 终端的发包频率和发包字节数上升，该终端占据信道收发的时间就会增加，就占用了更多的信道资源。而当 LoRa 终端采用更大的扩频因子时，信号可以传的更远，但是代价是传递单位字节的信息会花费更多的时间。

**8. 什么是 LoRa 终端或节点?**

LoRa 终端是 LoRa 网络的组成部分，一般由 LoRa 模块和传感器等器件组成。LoRa 终端可使用电池供电，能够远程定位。每一个符合 LoRaWAN 协议的终端都能与符合 LoRaWAN 的网关直接通讯，从而实现互联互通。

**9. 采用 LoRa 技术，我可以使用哪些 ISM 频段？**

按理论来说，你可以使用 150 MHz 到 1 GHz 频段中的任何频率。但是 Semtech 的 LoRa 芯片并不是所有的 sub-GHz 的频段都可以使用，在常用频段（如 433MHz，470MHz~510MHz，780MHz 以及欧美常用的 868MHz 和 915MHz 都属于常用频段）以外的一些频率并不能很好的支持。AUGTEK 主导 LoRaWAN 470~510MHz 协议标准的制定，目前在中国提供 **470-510MHz** 频段网关

**10. LoRa 网关使用免费频段，会不会容易受到频率干扰？**

抗干扰能力取决于 LoRa 技术本身的特性和网关的设计。LoRa 技术本身拥有超高的**接收灵敏度（RSSI）**和**超强信噪比（SNR）**。以 AUGTEK 的 LoRa 网关与 LoRa 模块为例，其接收灵敏度达到惊人的-142dBm，而超强的信噪比可以让 AUGTEK 网关和终端工作在噪声门限以下 20dB。此外，AUGTEK 网关使用**跳频技术**，通过伪随机码序列进行频移键控，使载波频率不断跳变而扩展频谱，防止定频干扰。

**11. LoRa 的数据传输速率是多少？** LoRaWAN 协议定义了一系列的数据传输速率，不同的芯片可供选择的速率范围不同，例如 SX1272 支持 0.3-38.4kbps，SX1276 支持 0.018-38.4kbps 的速率范围。**目前 AUGTEK 能实现 0.3-37.5kbps 的传输速率。**

**12. 使用 LoRa 设备发送或接收的数据长度有限制吗？**有限制，理论来说 SX127x 系列芯片有 256 Bytes 的 FIFO，发射或接收 256 Bytes 都行得通。但是，并不是在任何传输速率下 LoRa 模块的负载长度都能为 256 Bytes。在传输速率较低的情况下，一次传输 256 Bytes 需要花费的时间极长（可能需要花费几秒甚至更长），这不利于抗干扰和交互，因此在技术处理上一般**建议用户将一条长数据分割成数条小数据来进行传输。**

**13. 什么是速率自适应（ADR）？**

速率自适应（Adaptive Date Rate, ADR）是调整数据传输速率来保证可靠数据传输、优化网络性能、扩充网络容量的一种技术。当节点靠近网关时，数据传输速率可以更快、发射功率也更低。而在链路预算边缘处的节点，其数据传输速率更慢，发射功率更高。ADR 方法能适应不同的网络构造，支持不同的路径损耗，可以最大化终端的电池使用寿命和整体的网络容量，LoRa 网络能够从整体上管理每个终端的数据传输速率和扩频因子。

**14. 就 LoRa 设备而言，其天线所能实现的发射功率是多少？**

从芯片管脚输出的功率为+ 20 dbm，经过天线匹配/滤波损失一定功率，最终能输出的功率为+ 19 dbm + （-） 0.5 db。不同地区对最大输出功率有不同的规定，LoRaWAN 协议定义了不同地区在最大化链路预算的情况下的不同输出功率。

## LORA 参数说明：

1.扩大带宽，减少干扰

当扩频因子为 1 时，数据 1 就用 1 表示，扩频因子为 4 时，可能用 1011 表示 1，这样传输的时候可以降低误码率即信噪比，但是却减少了实际可以传输的实际数据，扩频因子越大，可以传输的数据数率越小

2.扩频因子另一个用途正交码 QVSF,通过正交可变扩频因子可以获得正交的扩频码，扩频因子为 4 有四个正交的扩频码，可以让同时传输的无线信号互不干扰,，可以同时传输 4 个人的信息

二，传输速率和距离

无线传输距离由接收机灵敏度和发射功率共同决定，两种之间的差值称为链路运算。输出功率受限于标准规范，只有通过提高灵敏度提高传输距离；而灵敏度受数据速率非常重要的影响。对所有调制方式来说，越低的速率，接收机的

带宽越窄，接收灵敏度越高。在现今高性价比无线收发机中应用最广泛的调制方式是 FSK 或者 GFSK。要进一步减小 FSK 系统的接收机带宽，唯一可行的办法就是提高参考晶体的精确度

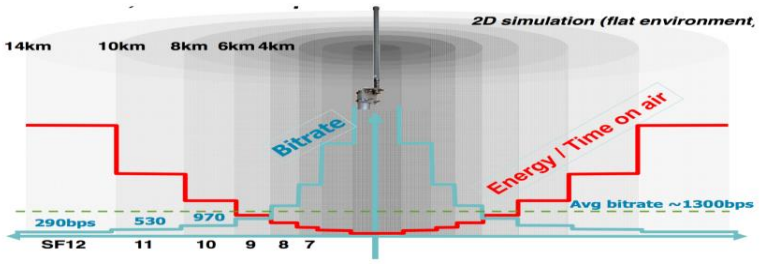
三：编码率

编码率越小，空中传输时间越长

编码率 4/5 4/6 4/7 4/8（最小为 4/8）

以下示意图形象地说明了 LORA 各参数指标的特点，关联的参数包括：扩频因子、数据率、传输距离、功耗等：

**扩频因子越大，数据率就越小，但传输的距离就越远，数据传输耗时越长**



# LoRaWAN 规范：

# 6 终端激活（End-Device Activation）

所有终端设备在正式加入 LoRaWAN 网络之前必须先进行初始化并激活。有两种激活方式：

无线激活（Over-The-Air Activation (OTAA)），设备部署和重置时使用;

手动激活（Activation By Personalization (ABP)），此时初始化和激活一步完成。

## 6.1 激活成功后存储在终端设备的数据

以下信息在激活成功后回存储在终端设备：设备地址（DevAddr）、应用 ID（AppEUI）、网络会话密钥（NwkSKey）和应用会话密钥（AppKey）。

- 6.1.1 终端设备地址（DevAddr）

  DevAddr 是终端在当前网络中的识别码，大小 32bits。结构如下：

| **Bit** | **[31..25]** | **[24..0]** |
| --- | --- | --- |
| DevAddr bits | NwkID | NwkAddr |

  最高 7 位是网络 ID（**NwkID**），用以区分有地域重叠的不同网络运营商和弥补有路由问题的网络。接下来的 25bits，终端设备网络地址（**NwkAddr**），该地址可以有由网络管理员分配。

- 6.1.2 应用唯一识别 ID（AppEUI）

  **AppEUI** 是 IEEE EUI64 的全球唯一应用 ID，用以识别终端设备的应用服务提供商（等等）。**AppEUI** 在进行激活操作之前就存储在终端设备中了。就是说 **AppEUI** 是出厂时烧录进去的。

- 6.1.3 网络会话密钥（NwkSKey）

  **NwkSKey** 是分配给终端设备的网络会话密钥。**网络服务器**和**设备用**它来计算和校验所有消息的 MIC（消息一致码），来保证收发的数据一致。**也可以用来对 MAC 负载（MAC 命令放在 Payload 里面）的消息进行加/解密**。

- 6.1.4 应用会话密钥（**AppSKey**）

  **AppSKey** 是分配给终端设备的应用会话密钥。网络服务器和设备用来对 应用指定的 Payload 字段进行加解密。也可以用来计算和校验应用层 MIC（可能存放在应用指定 消息的 Payload 中）。

## 6.2 无线激活（Over-the-Air Activation）

终端设备在与网络服务器交流（数据交换）之前，必须先通过加入过程加入网络服务器。每次终端设备会话的上下文丢失（与服务器通信断开）后都要重新加入。加入服务器之前，要使用以下信息初始化终端设备：全局唯一设备 ID（**DevEUI**）、应用 ID（**AppEUI**）、AES-128 密钥（AppKey）。 **AppEUI** 在上面 *6.1.2* 中有介绍。

** 注意：**无线激活时，网络密钥初不会向初始化那样写死到终端，而是在终端加入网络时由网络层衍生并分发，该密钥用来对传输数据进行加密和校验。这样，终端设备能很方便的在不同的网络服务器和应用提供商之间切换。使用**网络会话密钥和应用会话密钥***可以避免应用数据被网络供应商（网络服务器拥有者）解析或篡改，从而接入大量的网络服务器。*

- 6.2.1 终端设备 ID（DevEUI）

  DevEUI 是全球终端 ID，符合 IEEE EUI64，用来唯一辨识终端设备。

- 6.2.2 应用密钥（AppKey）

  **AppKey** 是 AES-128 的应用密钥，由应用拥有者通过 应用指定的 根密钥衍生并分配给终端设备，根密钥只有应用供应商知晓和掌握。终端设备通过无线激活入网时，通过 AppKey 衍生会话密钥 **NwkSKey** 和 **AppSKey**，并分发相应的终端设备，用来加密和校验网络通讯和应用数据。

- 6.2.3 入网流程

  从终端的角度看，和服务器交互的入网流程包含两个 MAC 消息：join request 和 join accept.

- 6.2.4 入网请求（Join-request message）

  入网流程由终端发起，终端入网时发送入网请求，消息格式（MACPayoad）如下：

| 字节数 | 8 | 8 | 2 |
| --- | --- | --- | --- |
| Join Request | AppEUI | DevEUI | DevNonce |

  入网请求消息包含：**AppEUI**、**DevEUI** 和终端设备产生的 2 字节的**随机数（DevNonce）**

  **DevNonce** 是个随机值，终端设备最近使用的一些(数量自定义)**DevNonce** 会保存在网络服务器（NS）。如果终端发送的入网请求中的 **DevNonce** 在 NS 中可以查到，该请求就会被忽略。

  注意：该机制的目的是防止重放攻击(replay attacks)，避免其它人通过发送之前的入网请求来断开终端设备和网络的连接。

  入网请求的消息一致性校验码（MIC）（见第 4 章 MAC 消息部分）通过下述**算法**计算：

```
cmac = aes128_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)

MIC = cmac[0..3]
```

  入网请求以明文发送。

- 6.2.5 接受入网消息（Join-accept）

  服务器同意终端入网后网络服务器（NS）会回复 **接受入网** 消息。**接受入网**使用 `JOIN_ACCEPT_DELAY1` 或 `JOIN_ACCEPT_DELAY2`（而不是 RECEIVE_DELAY1 和 RECEIVE_DELAY2），和普通消息一样发送。这两种接收窗口使用的信道频率和数据率与 RX1 和 RX2 的接收窗口（见章节 “物理层” 之” 接收窗口” ）相同。

入网请求被拒绝则服务器不发送任何数据。

**接受入网**消息包含以下字段：应用层随机数（AppNonce），3 字节；网络 ID（NetID）；终端地址（DevAddr）；介于 TX 和 RX（RxDelay） 之间的延迟；信道频率的一系列配置（CFList）。CFList 相关内容见第 7 章。

| 大小（字节） | 3 | 3 | 4 | 1 | 1 | 变长(16) |
|---|---|---|---|---|---|---|
| 接受入网 | AppNonce | NetID | DevAddr | DLSeting | RxDelay | CFList |

**AppNonce** 是由网络服务器产生的一个随机数或唯一 ID，终端设备用它来衍生两个会话密钥：**NwkSKey** 和 **AppSKey**。衍生算法如下：

```
NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce |NetID | DevNonce | pad16)

AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)
```

接受入网的 MIC 计算方式如下：

```
cmac = aes128_cmac(AppKey,MHDR | AppNonce | NetID | DevAddr | RFU | RxDelay | CFList)

MIC = cmac[0..3]
```

消息本身采用 AppKey 加密，加密方式如下：

```
aes128_decrypt(AppKey, AppNonce | NetID | DevAddr | RFU | RxDelay | CFList | MIC)
```

注意：网络服务器加密接收入网消息使用的是 AEC ECB 模式的**解密**算法。这样终端就不必再实现 AES 解密算法，只用 AES 加密即可。

**PS ：之前关于 NetID 的翻译有误，特此更正。**

**NetID** 包含：NetID 的 7 个最低有效位称作 NwkID，即 DevAddr（终端短址）的 7 个最高有效位。区域相邻或重叠的网络的 NwkID 不能相同。余下的 17 个最高有效位由网络运营商自由分配。

DLsetting 字段含有下行配置：

| 位（Bits） | 7 | 6:4 | 3:0 |
| --- | --- | --- | --- |
| DLsettings | RFU | RX1DRoffset | RX2 Data rate |

**RX1DRoffset** 设置上下行数据速率之间的偏移（偏差），和终端设备交互的首个接收时隙（RX1）。*（感冒好几天了，头僧，不知道该怎么翻译）*offset 默认为 0。下行数据速率不能比上行的大。考虑到一些地区基站的最大功率密度限制，offset 用来平衡上行和下行的无线链路余量。

上行和下行链路数据速率之间的确切关系见章节 “物理层（Physical Layer）”

延时 **RxDelay** 规则和 **TXTimingSetupReq** 命令中的 **Delay** 字段相同。

- 6.3 手动激活

手动激活需要生产商提供的工具和配套网关等