浙江大学
Zhejiang University

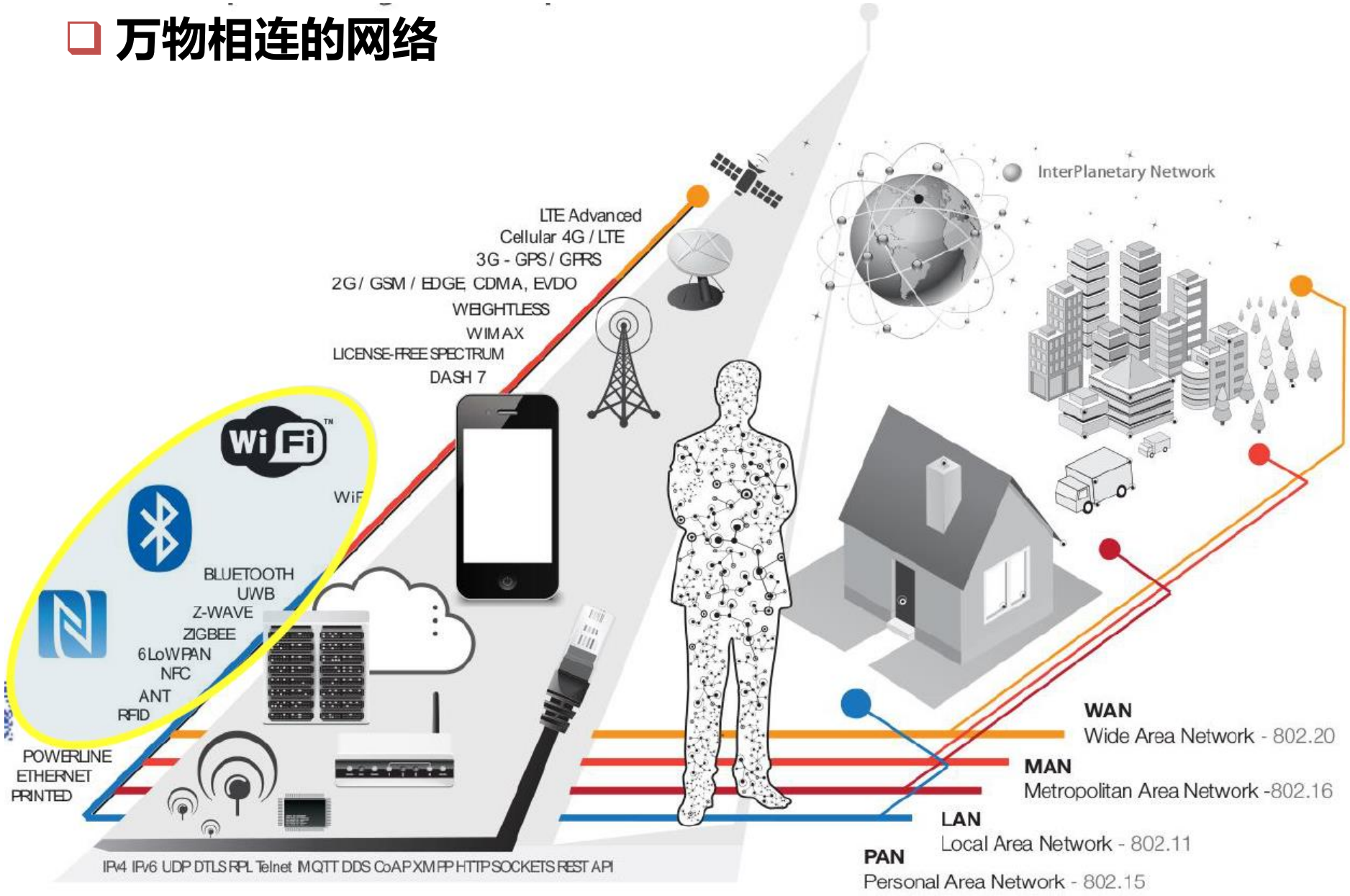
第五章：低功耗短距离通信协议

目录

- 概述
- 关键特性
- 基于IEEE802.15.4的传感网协议架构
- Bluetooth
- 研究进展

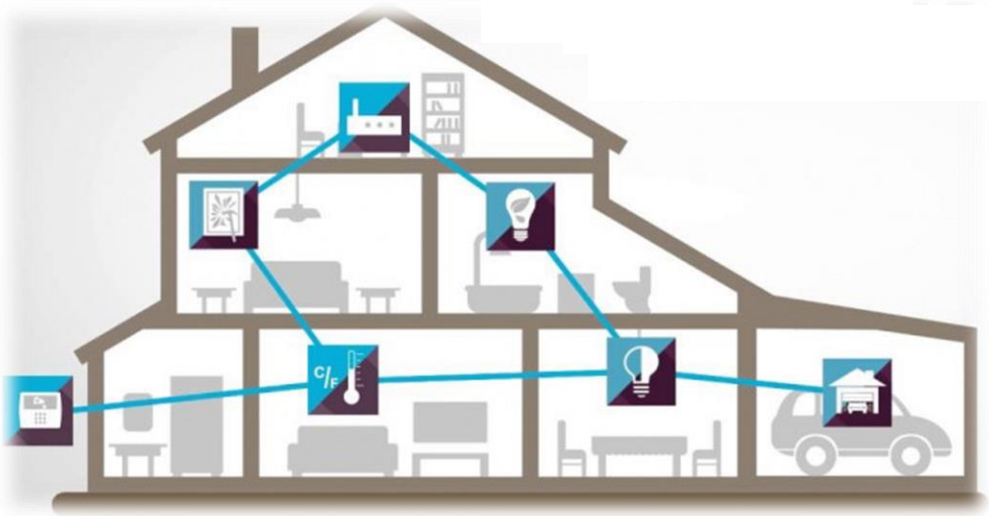
低功耗短距离通信需求

- 万物相连的网络



低功耗短距离通信需求

- 在物联网中，**万物互联**是关键
  - 例如，智慧家居



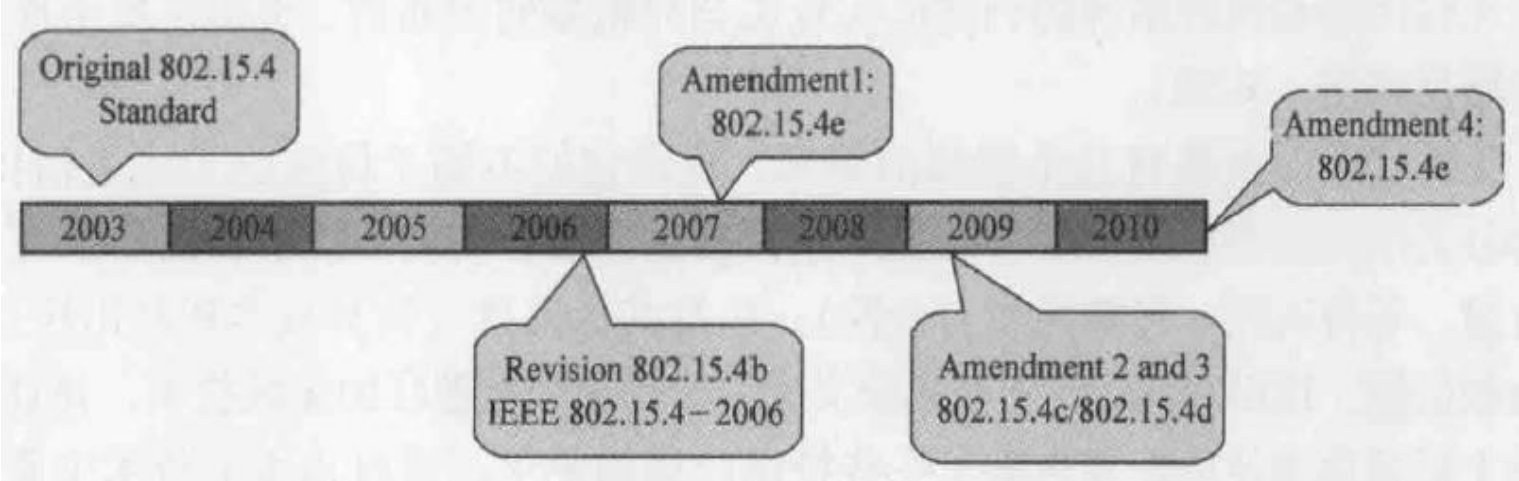
- 物联网通信的特点
  - 无线
  - 低速
  - 低功耗

低功耗短距离通信需求

- 低功耗 → 无线短距离通信技术
  - 发射功率与传输距离的关系：$P \propto d^n$
- 无线低功耗短距离通信技术
  - IEEE 802.15.4
  - ZigBee
  - 传感网协议
  - BLE
  - WiFi Halow/802.11ah

IEEE 802.15.4技术

- 历史发展
  - 2003年10月, 由IEEE发布，覆盖低速无线个域网的物理层和 MAC 层
  - 2001年3月, IEEE 802.15.4c修订版，增加专门用来解决我国新开放的881~787MHz无线传感网络频段的物理层（也称为780MHz中国频段）
  - 2010年，IEEE 802.15.4e, 定义了MAC层时分多址（TDMA）和信道跳频的支持，更好的支持了工业市场需求



ZigBee

- 历史发展
  - “紫蜂”，2004年诞生，是基于IEEE 802. 15. 4标准的低功耗短距离无线网络协议
  - 2007年，推出ZigBee PRO
  - 2011年，推出ZigBee RF4CE, 具备更强的灵活性和远程控制能力
- ZigBee 与 IEEE 802.15.4
  - 其协议栈的物理层和MAC层由15.4定义，而上层的网络层和应用层由ZigBee联盟定义
- ZigBee技术具有低功耗、低速率、低成本、组网灵活等优点

传感网协议

- 传感网是由大量低功耗节点组成的无线自组织多跳网络，实时感知、处理和传输网络覆盖区域内监测对象的信息
- 传感网协议是传感网应用的基础，自2002年以来大量研究围绕低功耗与组网技术展开
  - 低功耗MAC的研究
    - 2002年,S-MAC；2004年, LPL-MAC, Wise-MAC; 2008年, RI-MAC; 2010年A-MAC
  - 组网
    - 2009年, CTP协议支持多对一的路由模式
    - 2010年, IETF ROLL工作组提出RPL支持多对一, 一对多，一对一的路由模式
- 早期的传感网技术
  - CC1000, 38.4kbps, FSK调制
- 近期的传感网协议是基于802.15.4的协议架构
- 802.15.4具有高度开放性，促进了大量研究工作的产生

BLE

- 低功耗短距离协议IEEE 802.15.1是基于Bluetooth v1.1在2002和2005年被提出来的。
- 此后，协议IEEE 802.15.1便没有再演进，而是Bluetooth继续演进到了Bluetooth v4.0。此版本为Bluetooth的低功耗版本，即BLE（Bluetooth Low Energy）。
- 在2016年6月16日伦敦举行的媒体活动中，蓝牙 SIG正式公布了蓝牙5标准。
- 随着功能的逐步增加，在2017年7月18日，蓝牙 SIG 正式发布了蓝牙多跳的功能定义和多跳模型定义，并增加到了蓝牙5.0中，即Bluetooth v5.0。在家居物联网中，蓝牙多跳标准的确定使得多设备和多设备之间的通信成为可能。

其它低功耗协议

- RFID
  - 射频识别，无线电的信号是通过调成无线电频率的电磁场，把数据从附着在物品上的标签上传送出去，以自动辨识与追踪该物品
- NFC
  - 近场通信技术，2004年由飞利浦半导体、诺基亚和索尼共同研制开发，是一种短距高频的无线电技术，在13.56 MHz频率运行于20厘米距离内，由RFID演变而来
- WiFi HaLow
  - 即 IEEE 802.11ah，2017年由WiFI联盟发表，目的是在1km的距离下保持不低于100kb/s的数据传输率以及低功耗

关键特性

- ①功耗
- ②通信距离
- ③频带
- ④数据率
- ⑤安全性

通信功耗

❑ 不同协议的最大传输功耗

协议	最大传输功率
802.15.4	100mW
BLE	10mW
802.11b	1W

TIPS:
D(dBm)与P(mW)换算
D= 10lgP

❑ 无线收发器状态与能耗

- 符合15.4标准的CC2420 radio工作电压为 2.1~3.6V，通常为 3.0V
- 对于两节750mAh的电池来说, 以最大输出功率不停发数据最多工作86h

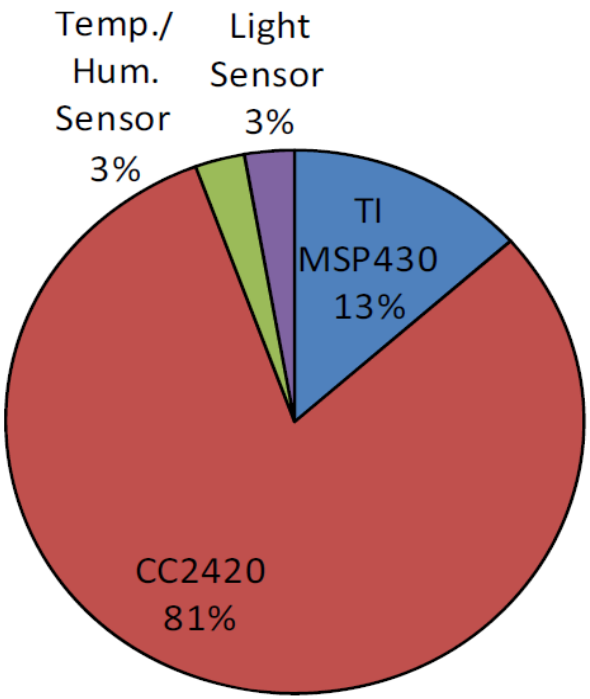
CC2420	电流能耗
TX	8.5~17.4mA
RX	18.8mA
CCA	0.426 mA

可调节
能耗级别

PA_LEVEL	电流能耗(mA)	输出功率(dBm)
31	17.4	0
27	16.5	-1
23	15.2	-3
11	13.1	-5
15	12.5	-7
11	11.2	-10
7	1.1	-15
3	8.5	-25

通信功耗

- 嵌入式设备的功耗很大部分来源于通信功耗
  - 例如，对于Telosb节点，cc2420能耗占其总能耗的81%
- 通信<->计算功耗的关系
  - TI MSP430 : 250uA/MIPS
  - 1ms TX 相当于69.6 MIPS
- 降低无线收发器占空比
  - 无线收发器占空比=活跃时间/总运行时间
  - 示例：设备活跃时的能耗为10mW, 休眠时的能耗为10μW,若占空比为0.1%, 则平均功耗为19.99μW。



通信距离

- 不同技术的通信距离
- 通信距离的计算模型
  - 常用的路径损耗模型

协议	最大通信距离
802.15.4	~100m
BLE	~100m
802.11b	~140m

$$PL(d) = PL_0 + 10n\log\left(\frac{d}{d_0}\right) + X_\sigma$$

- PL_0: 传输d_0距离的路径损耗，通常=1m; n: 衰减指数; X_σ:高斯随机误差

  - 最远通信距离：使得接收信号强度不低于接收器灵敏度的距离，可以解以下公式求解

$$RSens = P_{tx} + P_{ta} + P_{ra} - PL(d)$$

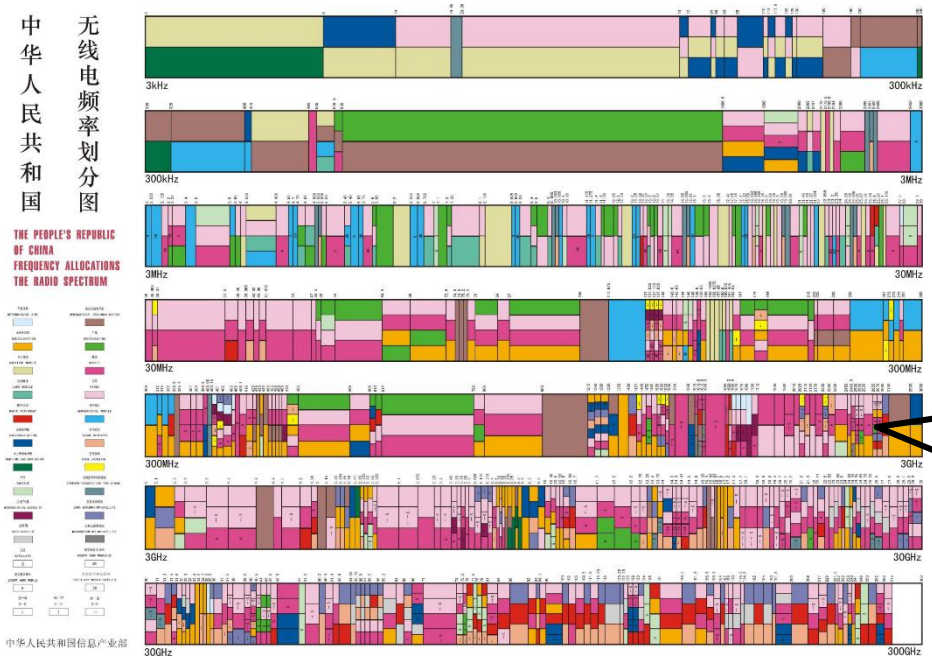
- P_{tx}:发射功率，P_{ta}，P_{ra}：发射，接收天线增益

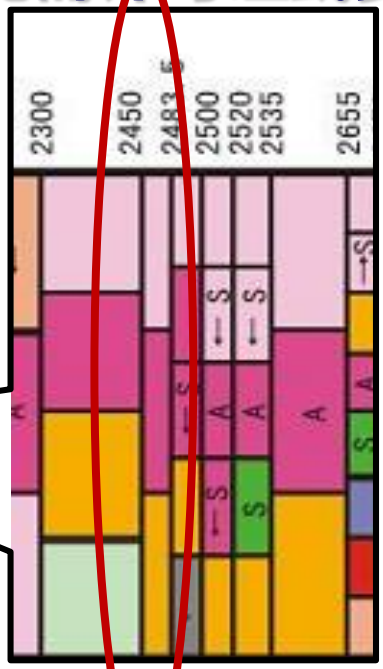
  - 示例
    - 不考虑天线增益与随机误差，假设发射功率为0dBm，接收灵敏度-85 dBm, PL_0为28.5dBm, 衰减指数3，那么最远通信距离为76.4m

- 多跳网络--更大的通信距离

频带和数据率

- 授权频段与ISM band
  - 授权频段
    - 使用授权频段需要许可证, 例如AM,FM广播频段
  - ISM频段(Industrial Scientific Medical Band),主要开放给工业，科学和医学机构使用, 无需许可证或费用





2.45GHz ISM Band:
2.4–2.4835 GHz

- 常见的ISM band
  - 包括2.45GHz , 5.8GHz, 115MHz 等

频带和数据率

- 不同协议频带与数据率

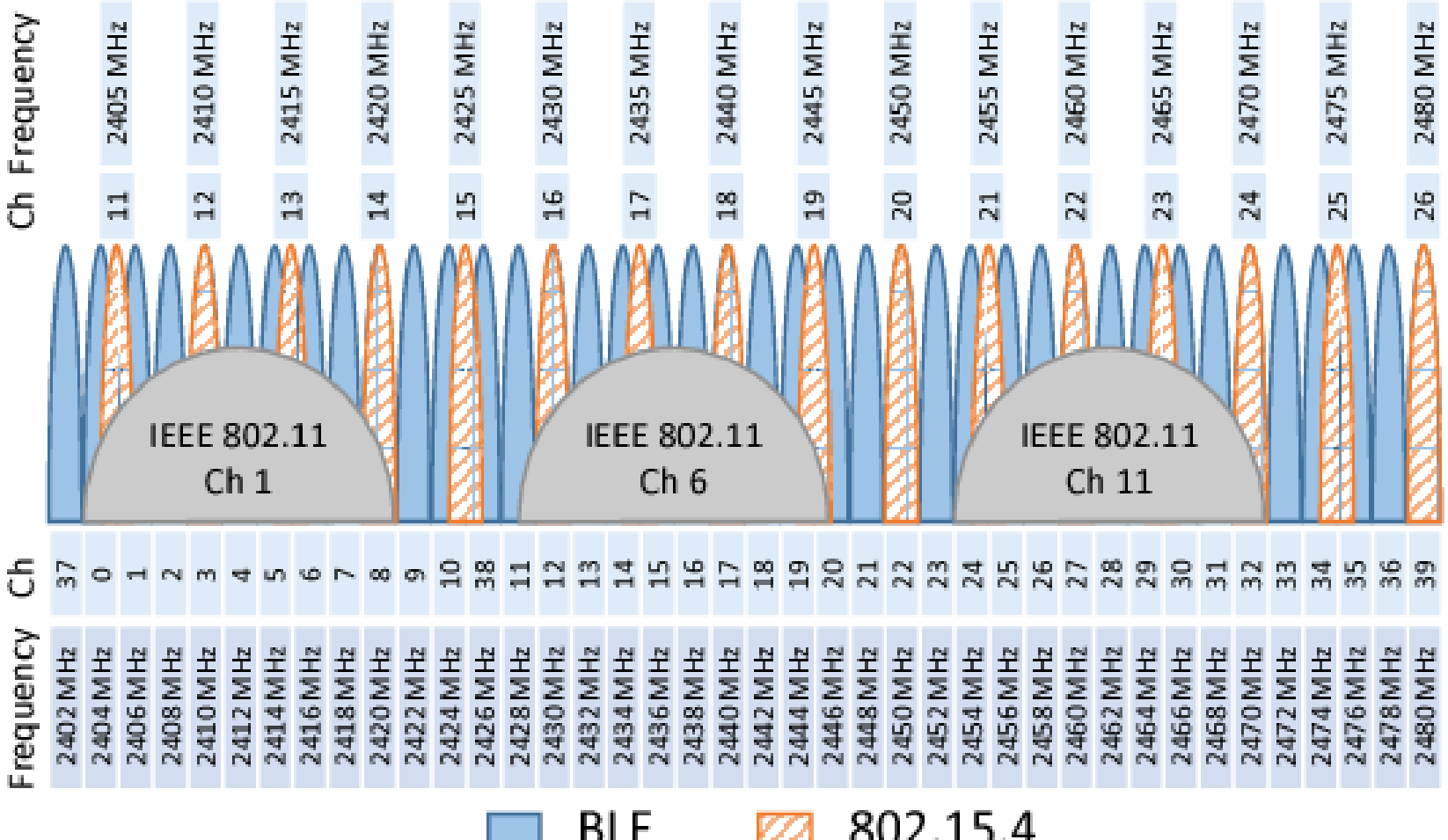
协议	中心频率(Hz)		频宽(Hz)	最高数据率
802.15.4	全球通用	2.45G	2M	250kbps
	北美	115M		40kbps
	欧洲	868M		20kbps
BLE	2.4G		2M	2Mbps
802.11b	2.45G		22M	11Mbps

频带和数据率

- 实际数据率与最大数据率
  - 最大数据率
    - 802.15.4 物理层最大数据率是250kbps
    - BLE 最大数据率为2Mbps
  - 实际中通常不能最高数据率
    - 802.15.4 数据包发送前会主动退避一段时间($initial_{backoff} \in [0, CW_{init})$，通常$CW_{init} \approx 10ms$，实际数据率一般为最大数据率的1/3
    - 在BLE 中，master在每个连接事件间隔会分配时间片扫描邻居的BLE 节点，其数据率与连接事件间隔长短以及扫描时间有关

2.4GHz频段信道分配

- 802.15.4：16个信道，2MHz
- BLE： 40个2MHz信道
- 802.11b：3个信道



安全性

- 芯片加密
  - CC2420支持AES-128硬件加密

- 协议提供的安全服务
  - IEEE 802.15.4标准在MAC层向上提供帧安全服务
    - 数据保密性
      - 使用AES-128对称加密算法
    - 数据完整性
      - 在消息中附加消息完整性代码(MIC)来校验
  - 蓝牙
    - 跳频
    - 应用层加密

非技术特征

- 模组价格
  - 2018年, 德州仪器 Texas Instruments (TI)官网价格

TI模组	核心产品	价格（美元）
BLE	CC2642R	2.70
	CC2640R2F	1.80
	CC1352R	4.58
802.15.4	CC1352R	4.58
	CC2652R	3.05
	CC2420	4.35

- 常用设备支持
  - BLE: 手机，共享单车
  - ZigBee: 小米智能家居，亚马逊Alexa
- 上游芯片厂商
  - ZigBee:TI, 美国飞思卡尔, 美国Ember, 美国Atmel等
  - BLE: TI, 英国CSR, 美国博通, 挪威Nordic, 上海锐迪科等
- 标准化
  - 标准化组织：ZigBee 联盟，蓝牙技术联盟(Bluetooth SIG)
- 协议的开放性
  - 15.4 比BLE更开放

典型的低功耗短距离协议

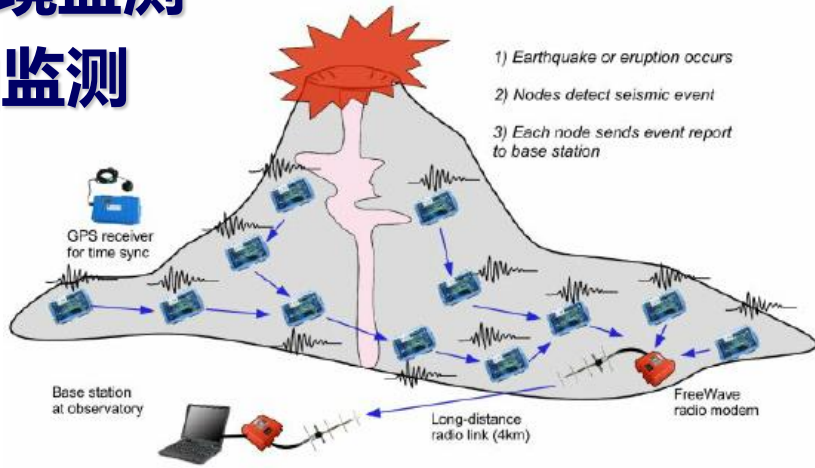
协议	802.15.4/ZigBee	BLE	802.11b
通信距离	~100m	~100m	~100m
功耗	~10mW	~10mW	~1W
频带	868/115MHz, 2.4GHz	2.4~2.5GHz	2.4~2.5GHz
数据率	250kbps	2Mbps	11Mbps
数据包长度	127 B	265 B	2304B

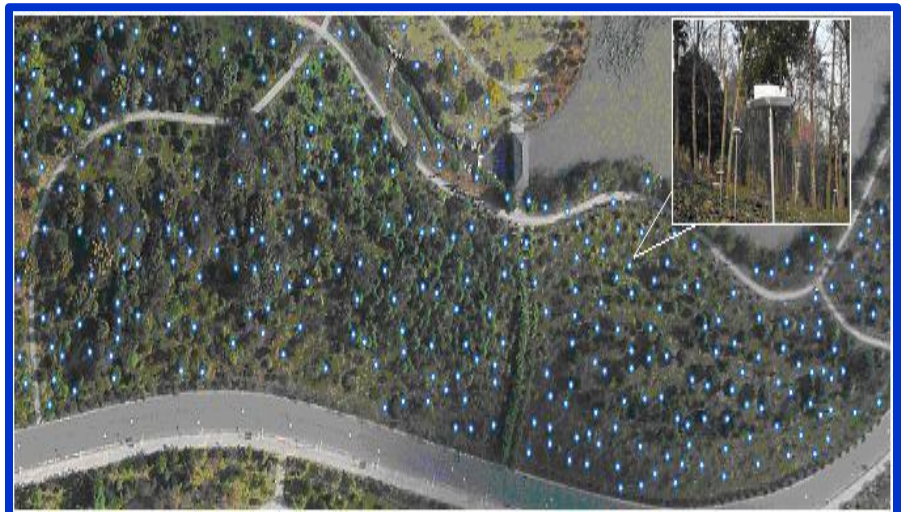
目录

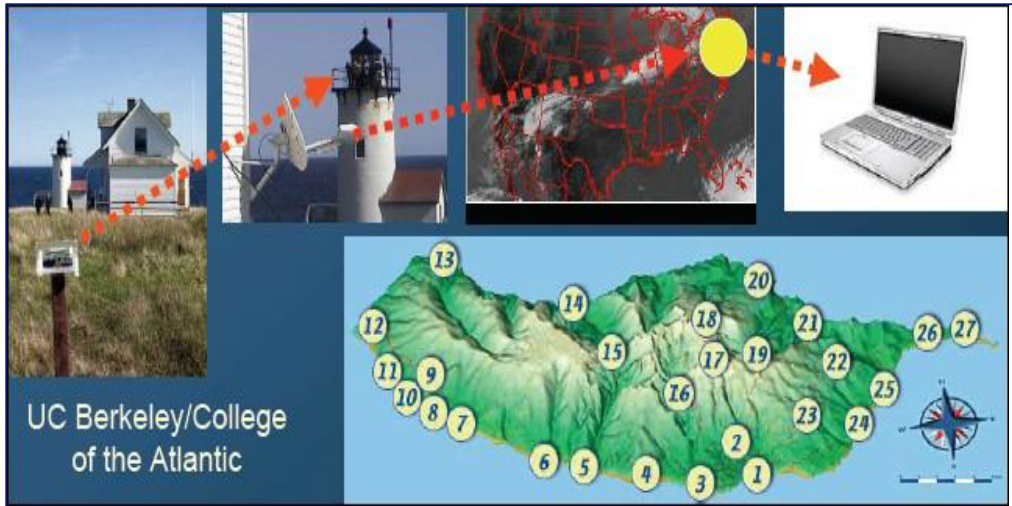
- 概述
- 基于IEEE802.15.4的传感网协议架构
  - IEEE 802.15.4技术物理层
  - 低功耗MAC
  - 链路质量估计
  - 网络层协议
  - 应用案例
- Bluetooth
- 研究进展

传感网协议的使用场景

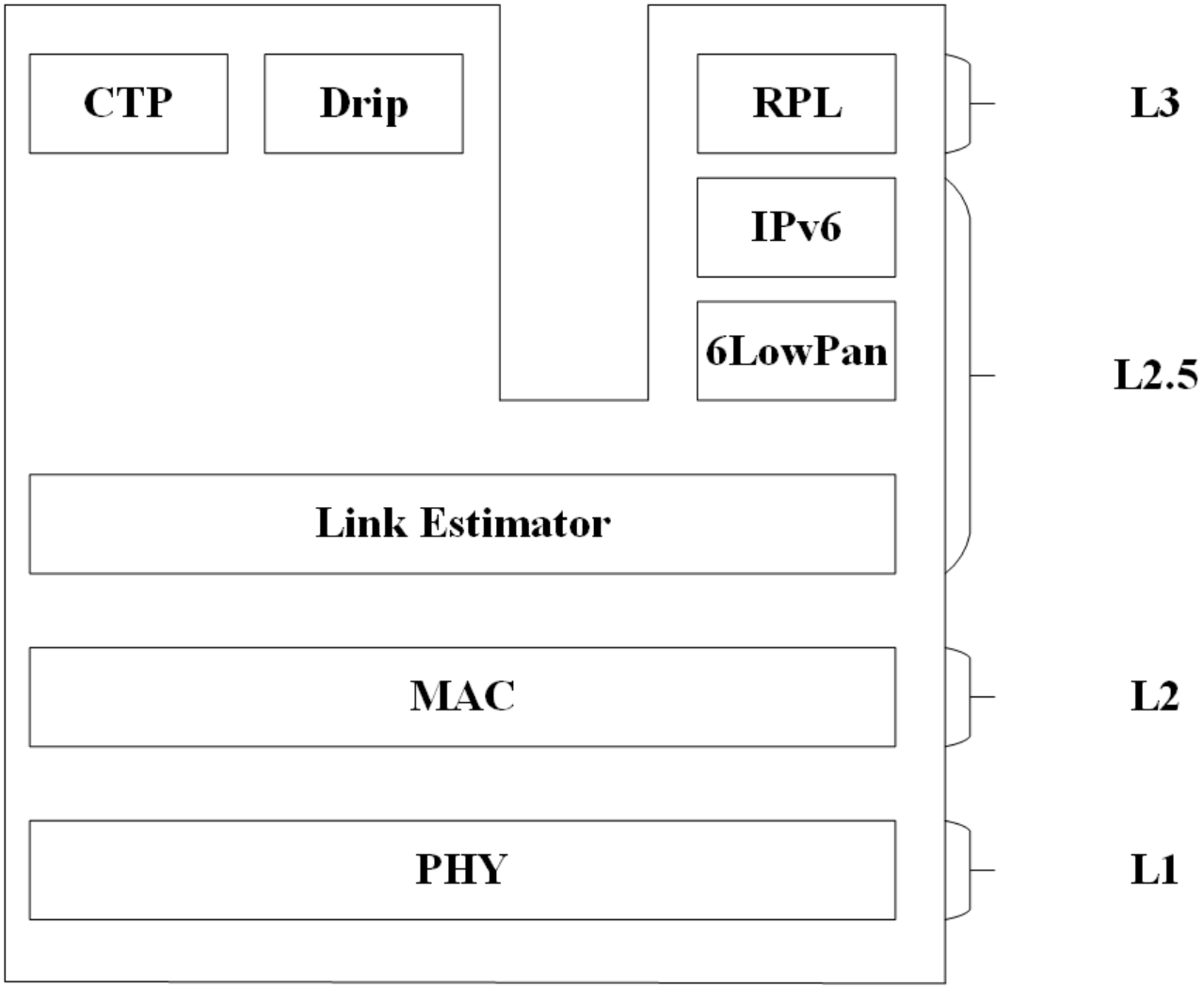
- 传感网协议的典型使用场景: 自组织无线多跳网络
- 传感网应用场景
  - Great Duck Island，生态环境监测
  - Volcano monitoring，火山监测
  - 绿野千传，森林火灾监测





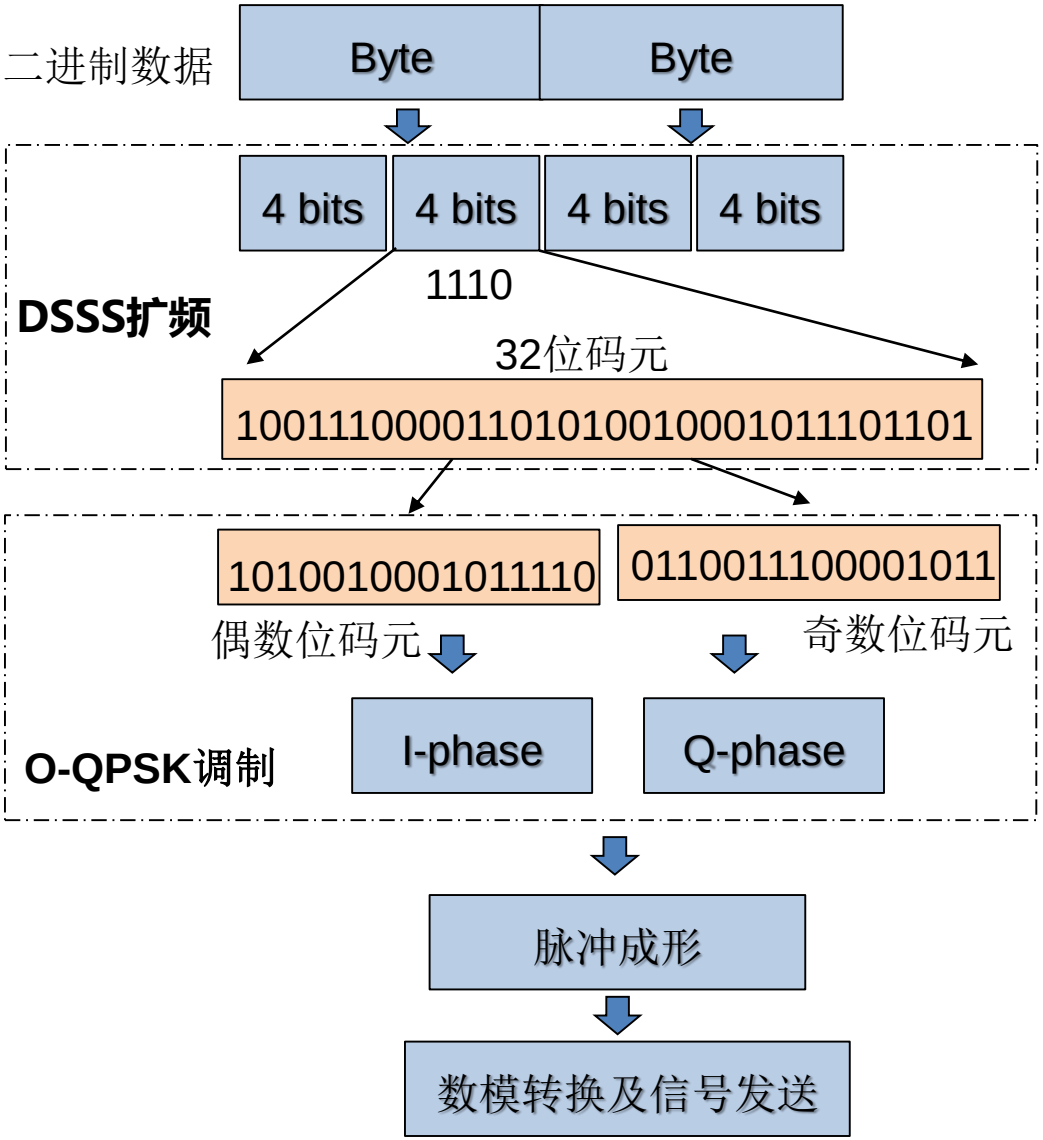


传感网协议架构



IEEE 802.15.4技术物理层

- 物理层功能：从比特流到RF信号
- 工作流程
  - DSSS扩频
  - O-QPSK调制
  - 脉冲成形
  - 数模转换及信号发送



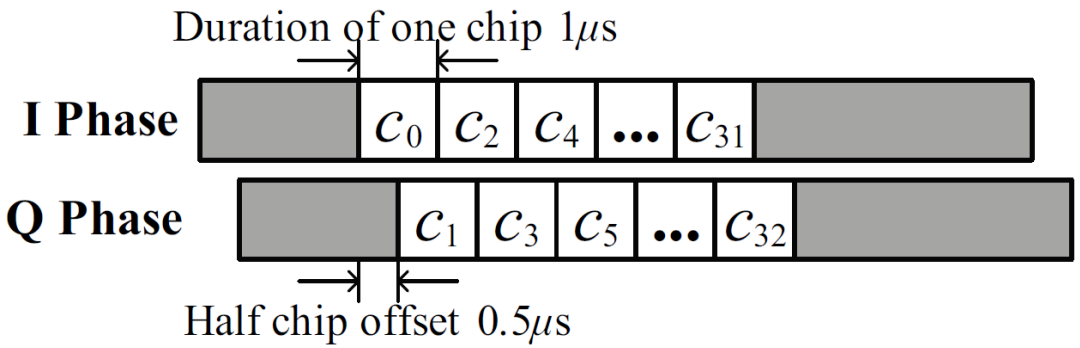
IEEE 802.15.4技术物理层

- DSSS 扩频
  - 每4bit数据转化为32位码元(chips)
  - Chip rate：2M chip/s， data rate: 250kb/s

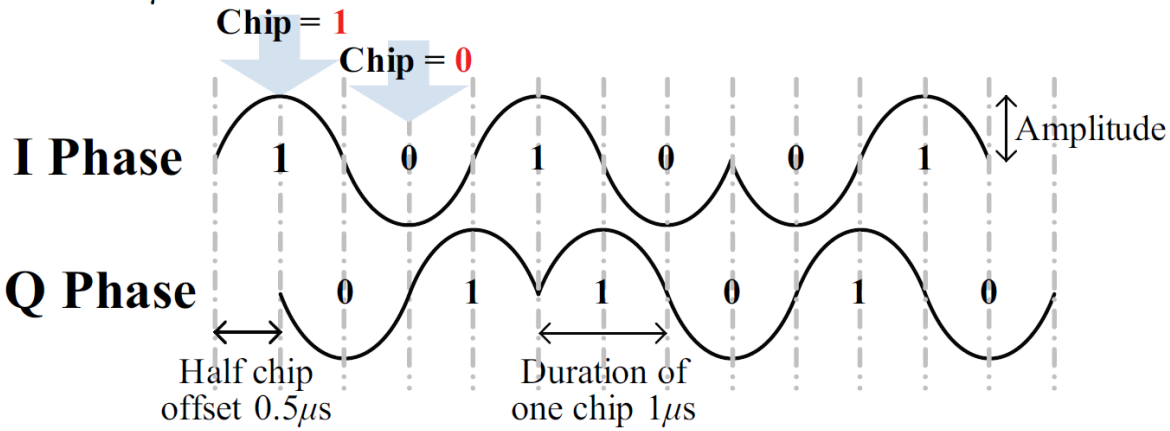
数据符号（二进制）（$b_3\ b_2\ b_1\ b_0$）	码元值（$c_0\ c_1 \cdots c_{30}\ c_{31}$）
0000	11011001110000110101001000101110
0001	11101101100111000011010100100010
0010	00101110110110011100001101010010
0011	00100010111011011001110000110101
0100	01010010001011101101100111000011
0101	00110101001000101110110110011100
0110	11000011010100100010111011011001
0111	10011100001101010010001011101101
1000	10001100100101100000011101111011
1001	10111000110010010110000001110111
1010	01111011100011001001011000000111
1011	01110111101110001100100101100000
1100	00000111011110111000110010010110
1101	01100000011101111011100011001001
1110	10010110000001110111101110001100
1111	11001001011000000111011110111000

IEEE 802.15.4技术物理层

- O-QPSK调制
  - 将32位码元序列调制到信号的I和Q相位上
    - 每个相位的chip rate = C/2 = 1M chip/s
    - 每个码元的发送时间：1s/(1Mchip/s)= 1us
  - O表示offset，表示相位偏移，offset=0.5 us

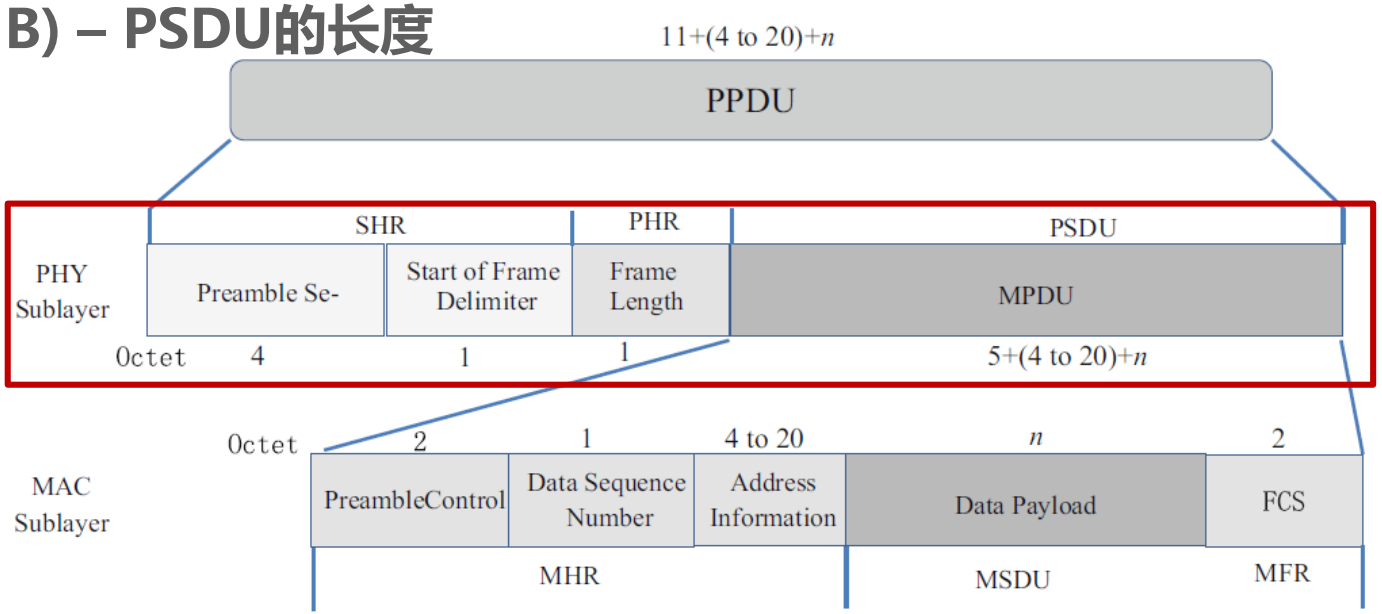


- 脉冲成形
  - 上半正弦表示1
  - 下半正弦表示0



IEEE 802.15.4技术物理层

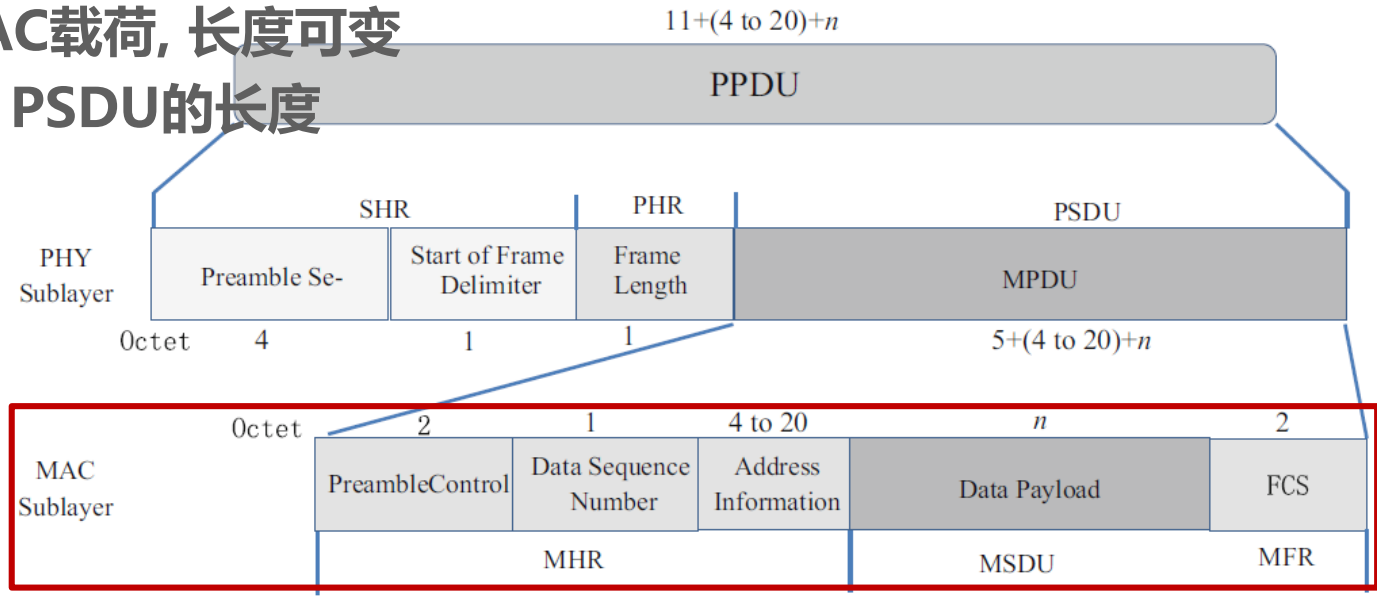
- 帧结构
  - 物理层数据单元(PPDU)=物理层头部+MAC层数据单元(MPDU)
  - 物理层头部
    - SHR (4B) – 用于实现码元和位同步
    - SFD (1B) –帧定界符，固定的特殊值，用于标志SHR的结束和实际数据报文的开始
    - PSDU(0~127B): 实际数据报文，从MAC层传过来的数据单元
    - PHR (1B) – PSDU的长度



IEEE 802.15.4技术物理层

帧结构

- 物理层数据单元(PPDU)=物理层头部+MAC层数据单元(MPDU)
- MAC层数据单元(MPDU)
  - MHR: MAC头部
    - Preamble Control (2B) – 帧控制字段
    - Data Seq. No. (1B)-序列号字段
    - Address Information (4~20B) –地址信息字段
  - MSDU: MAC载荷, 长度可变
  - MFR(2B) – PSDU的长度



目录

- 概述
- 基于IEEE802.15.4的传感网协议架构
  - IEEE 802.15.4技术物理层
  - 低功耗MAC
  - 链路质量估计
  - 网络层协议
  - 传感网应用案例
- Bluetooth
- 研究进展

CSMA MAC

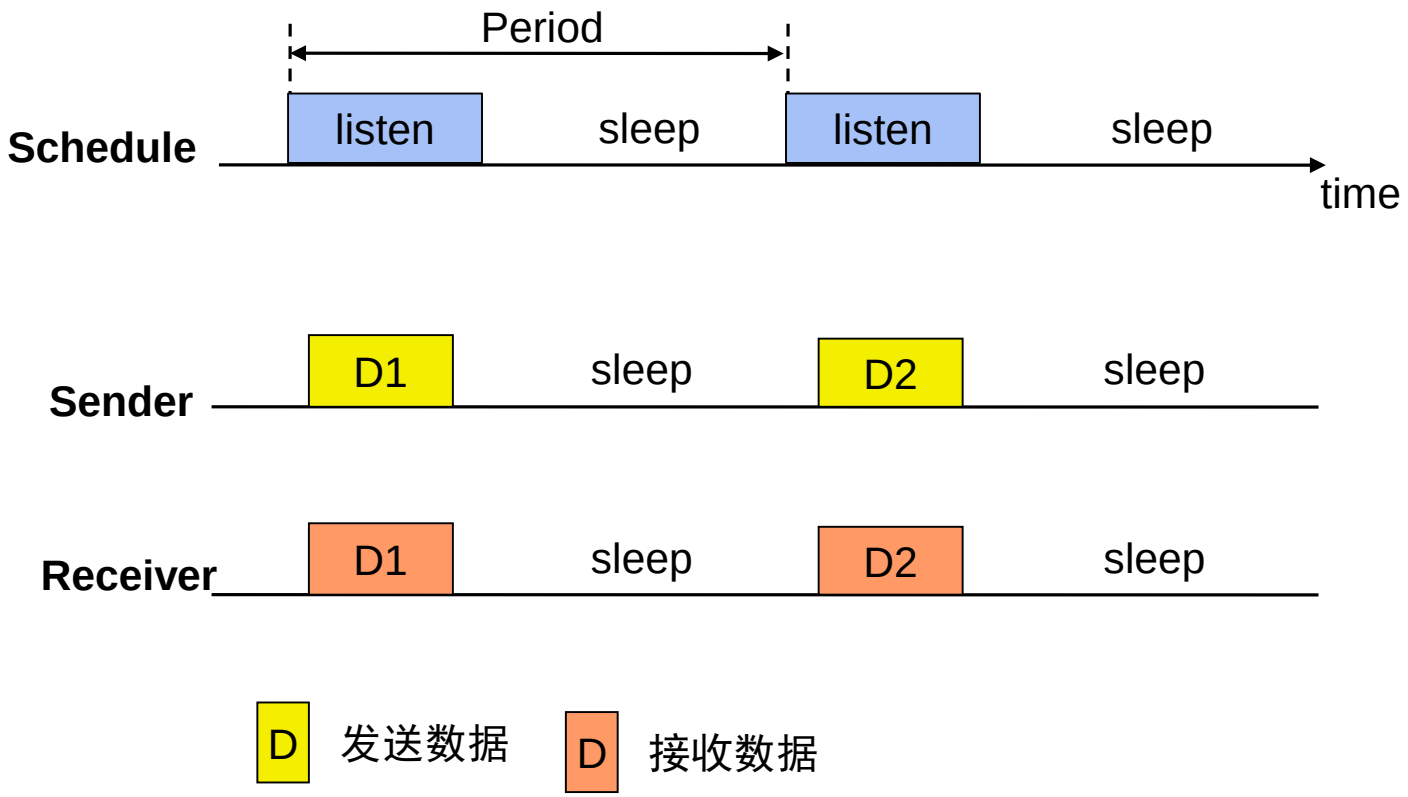
- Carrier Sense Multiple Access
  - CS (Carrier Sense) 载波侦听
    - 在发数据前先侦听，如果信道空闲，则立刻发送数据；如果有数据在传输，则退避一段时间
  - 问题： 接收者始终需要在线
- 链路层ACK
  - Hardware ACK （收到包，硬件自动固定112us延迟后发送ACK)
    - 有点：很快，硬件自动回复ACK
    - 缺点：但是容易出错，很可能无线射频芯片收到的包并没有传给上层软件
  - Software ACK
    - 比较慢，但是更可靠

低功耗MAC

- 低功耗MAC
  - 基于同步机制的MAC
  - 基于异步机制的MAC
    - 发送者发起
    - 接收者发起

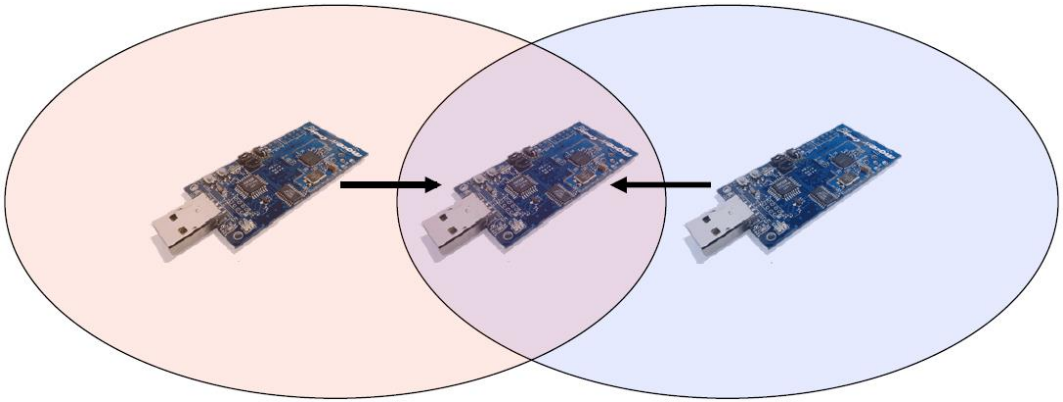
低功耗MAC：S-MAC

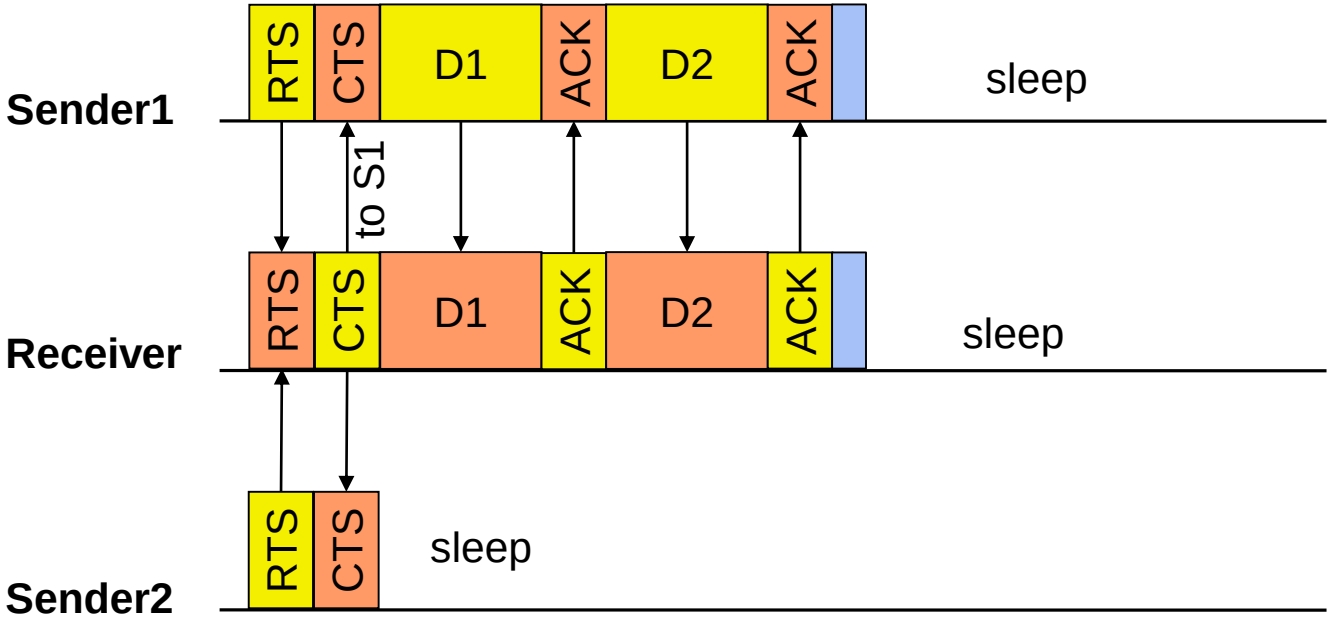
- 基本模式：节点与邻居节点同步苏醒和睡眠时间，并在苏醒时间内传输数据



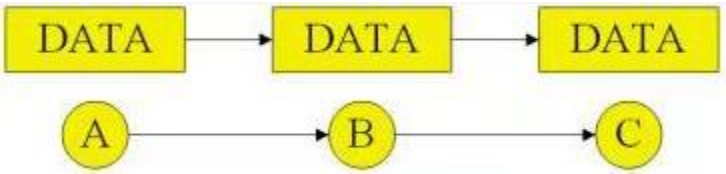
低功耗MAC：S-MAC

- RTS-CTS机制
  - 隐藏/暴露终端问题
  - 减少空闲侦听

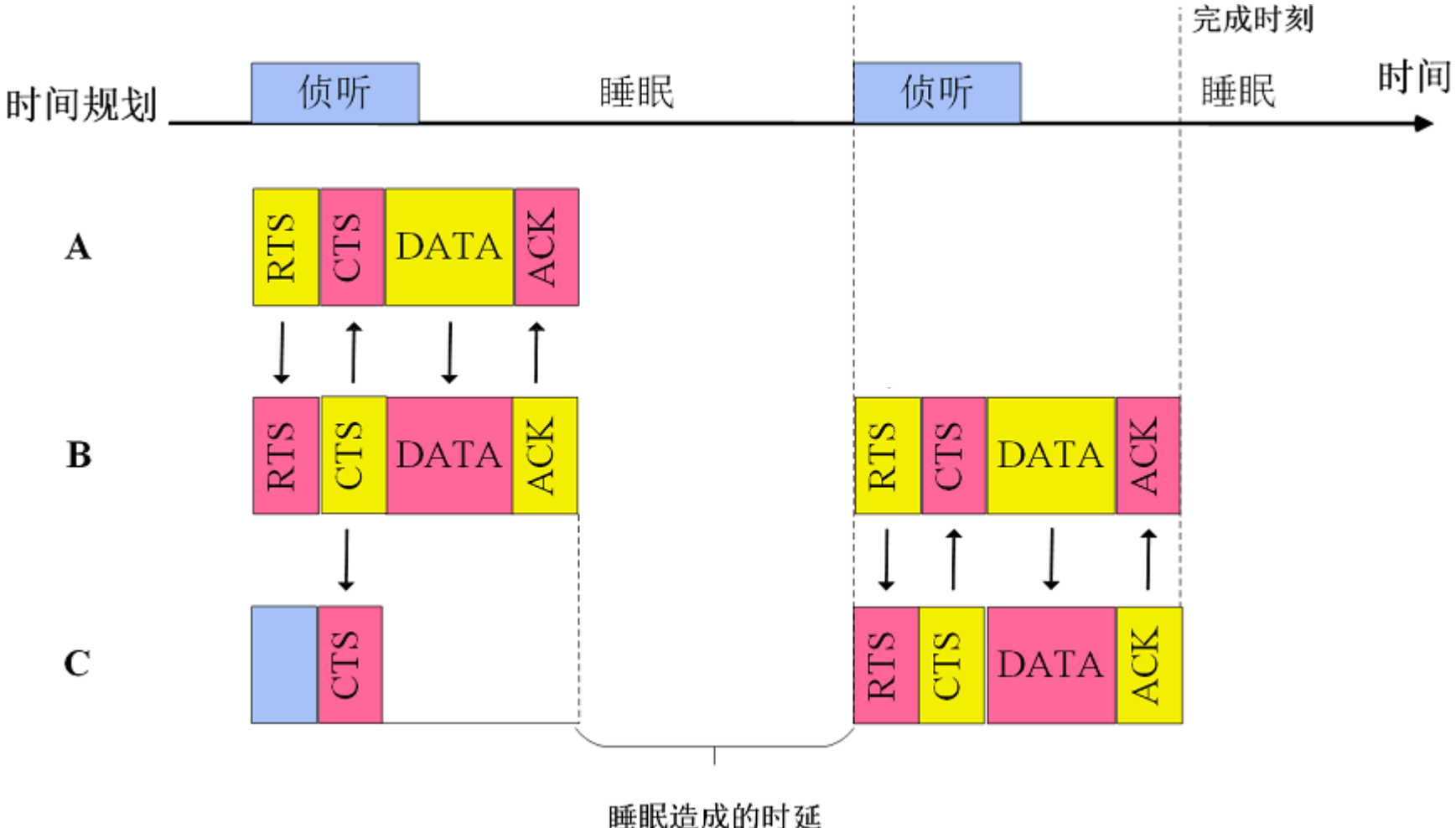




低功耗MAC：S-MAC

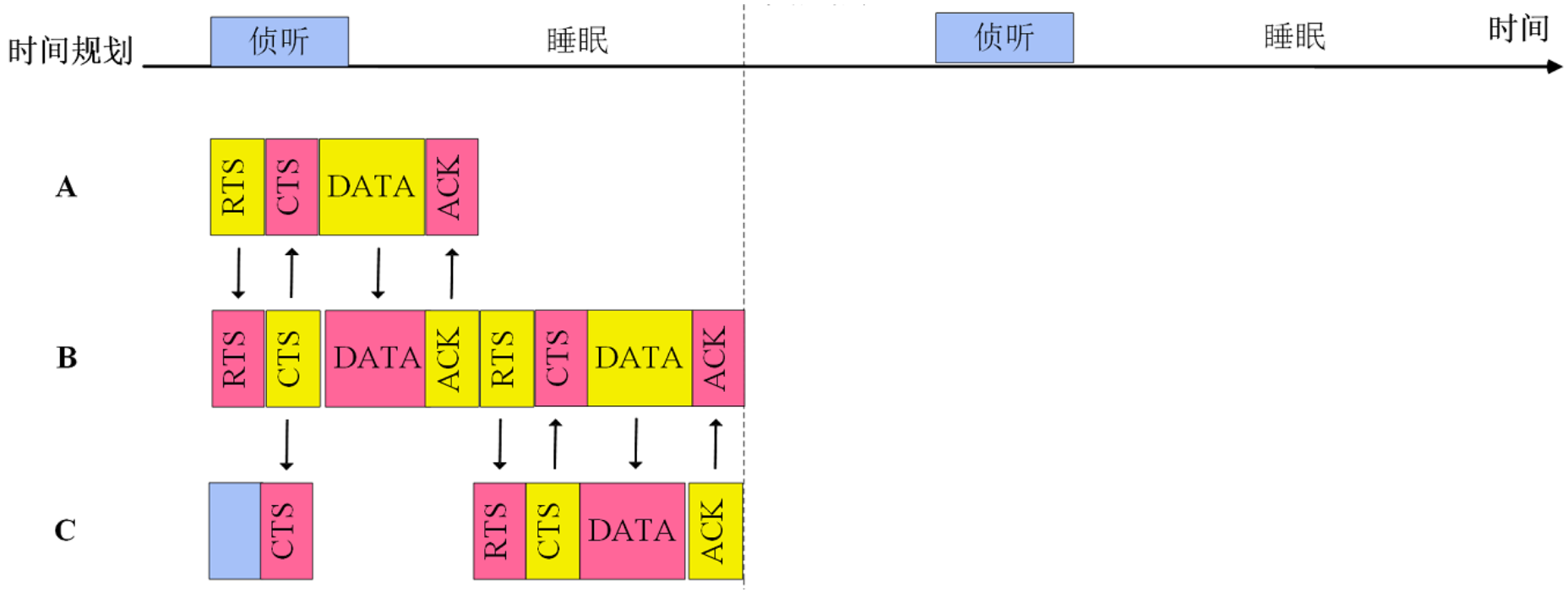


- 多跳传输延迟问题
  - 多跳传输中睡眠造成的延时会逐跳累加的问题
  - 示例



低功耗MAC：S-MAC

- 自适应侦听机制
  - 邻居C依据CTS包知道B传输结束的时间，并在C传输结束时短暂醒来侦听
  - 该例中，C正好是B的下一跳，减少了延迟

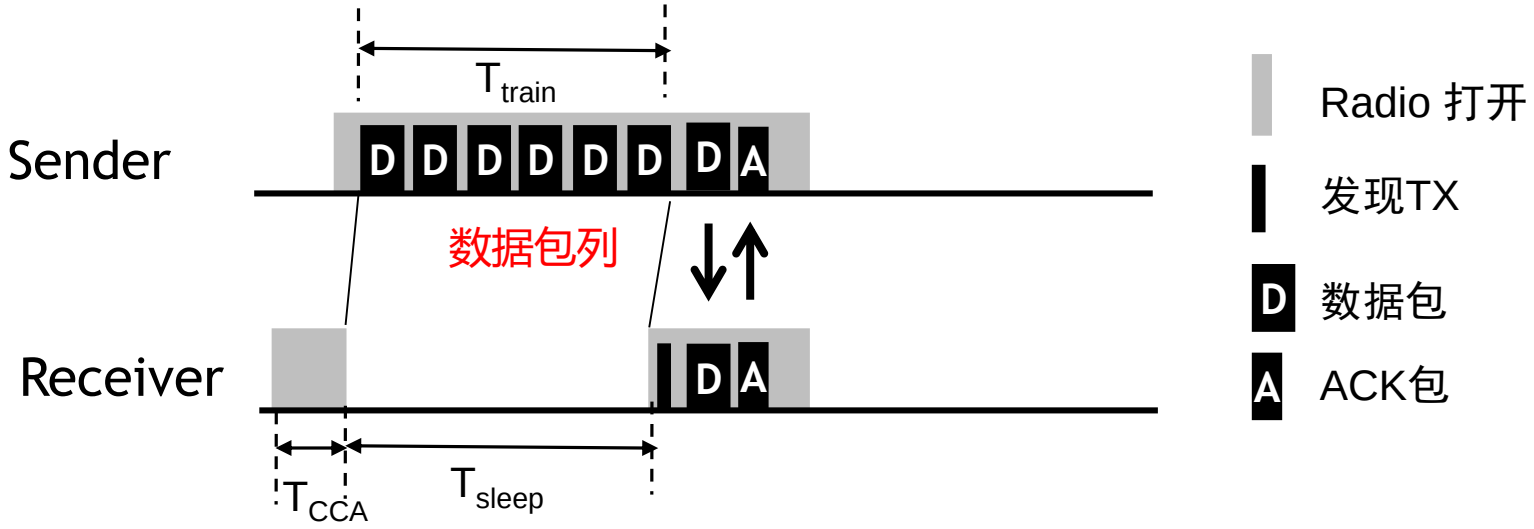


发送方发起网络：S-MAC

- S-MAC 回顾
  - 比传统802.11中的CSMA/CA MAC更节能
  - 空闲侦听依然开销很大
    - 所有人一开始都必须醒着，直到某个人开始传输数据
  - 网络空闲时有同步的开销
  - 发送数据时有RTS/CTS开销

发送方发起网络：LPL MAC

- LPL (Low Power Listening) MAC
  - 发送方发送足够长的探测包(preamble)列来唤醒接收者，LPL MAC直接使用数据包作为探测包
  - 接收方周期性做CCA，一旦发现能量大于阈值(例如，-77 dBm），就进入接收状态接收数据
    - CCA: Clear Channel Assessment, 用于评估信道上是否有数据传输



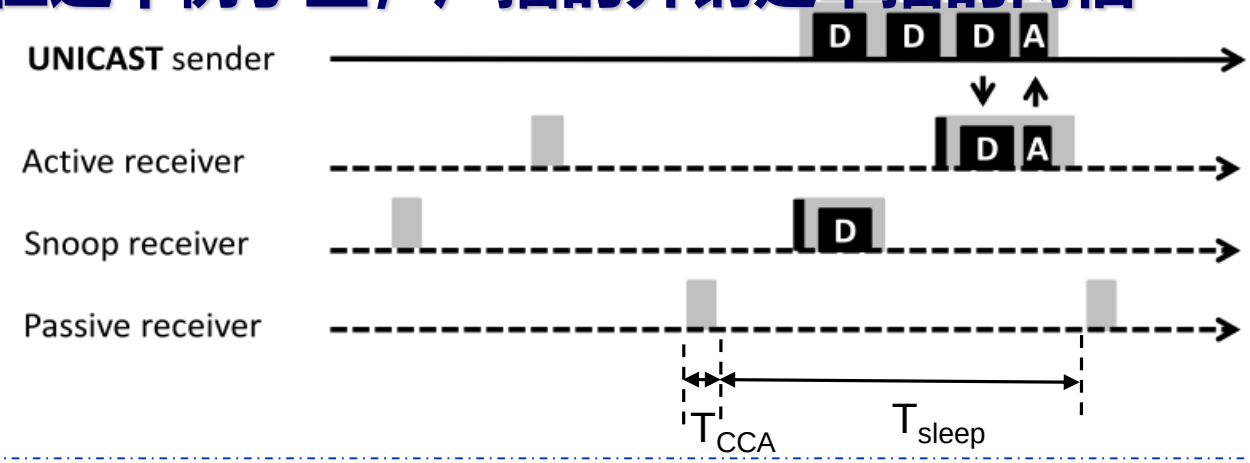
发送方发起网络：LPL MAC

- CS和CCA对比
- 相似点：检测能量是否高于阈值，探测周围是否有人发包
- 不同点：结果不同
  - CS发现则退避
  - CCA发现则进入唤醒
- 阈值设置 (默认-77dBm)
  - 偏高
    - CS可能无法感知潜在的传输，引起冲突
    - CCA可能无法检测潜在的传输，错过接收
  - 偏小
    - CS： 环境噪音也可能引起不必要的退避
    - CCA: 环境噪音也可能唤醒节点

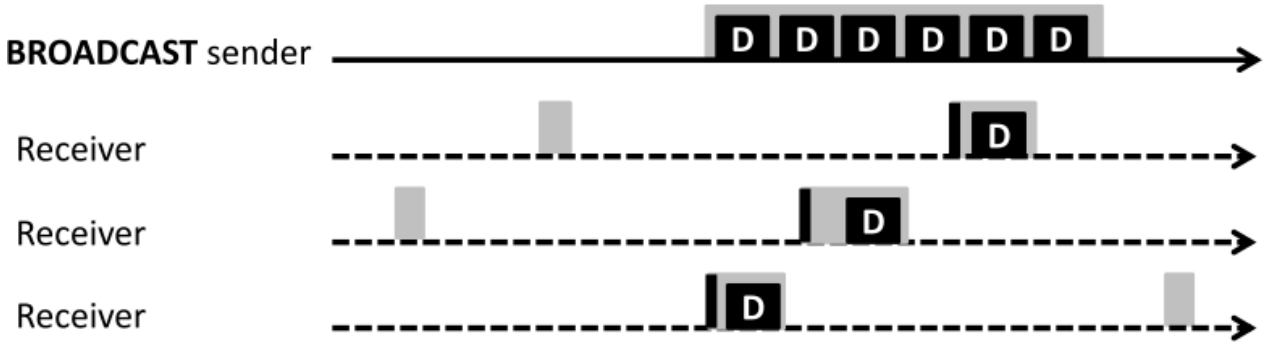
发送方发起网络：LPL MAC

- LPL 单播vs.广播
  - 单播(UNICAST)：ACK包可以缩短数据包列
  - 广播(BROADCAST)：发送的数据包列持续时间必须大于等于接收者的睡眠时间（$T_{train} >= T_{sleep}$）
  - 在这个例子里，广播的开销是单播的两倍

单播



广播

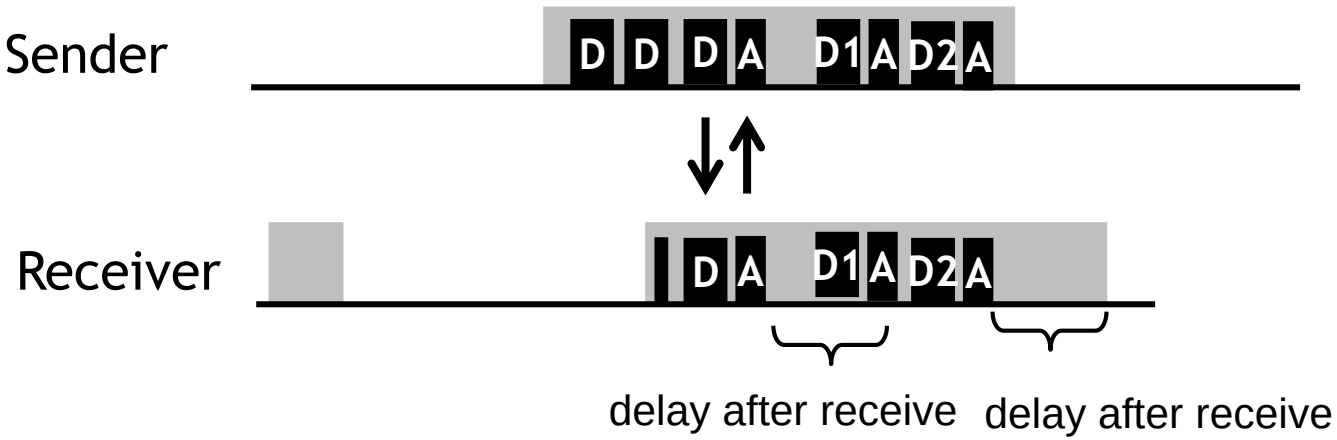


思考题

- 已知节点在CCA时间中采样400次，平均值超过-77dBm才醒来。假设，CCA的长度 11ms,探测包长度是2ms，接收的信号强度-56dbm，环境噪声-100dbm。那么，为了使接收方正确收到包，探测包间隔最大为多少?

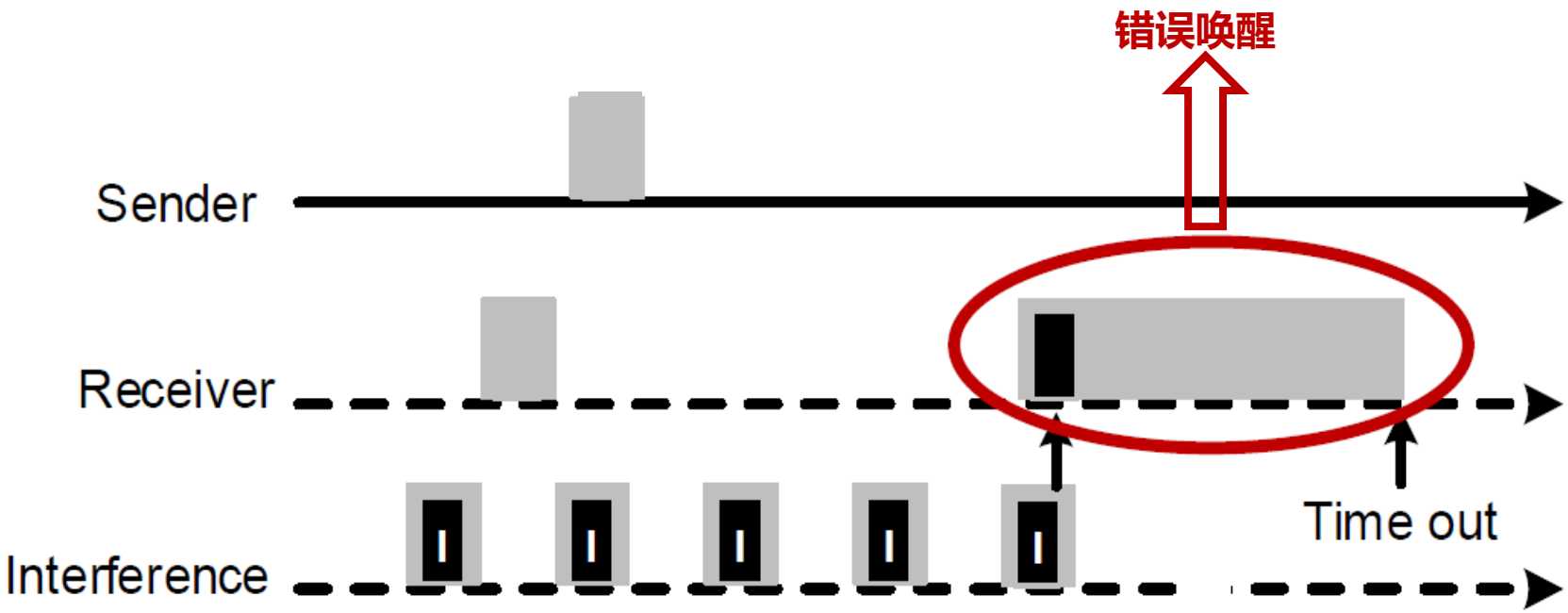
发送方发起网络：LPL MAC

- 自适应机制
  - 接收方回复完ACK之后持续监听一段时间(delay after receive)来接收发送者接下来可能会连续发送的数据包



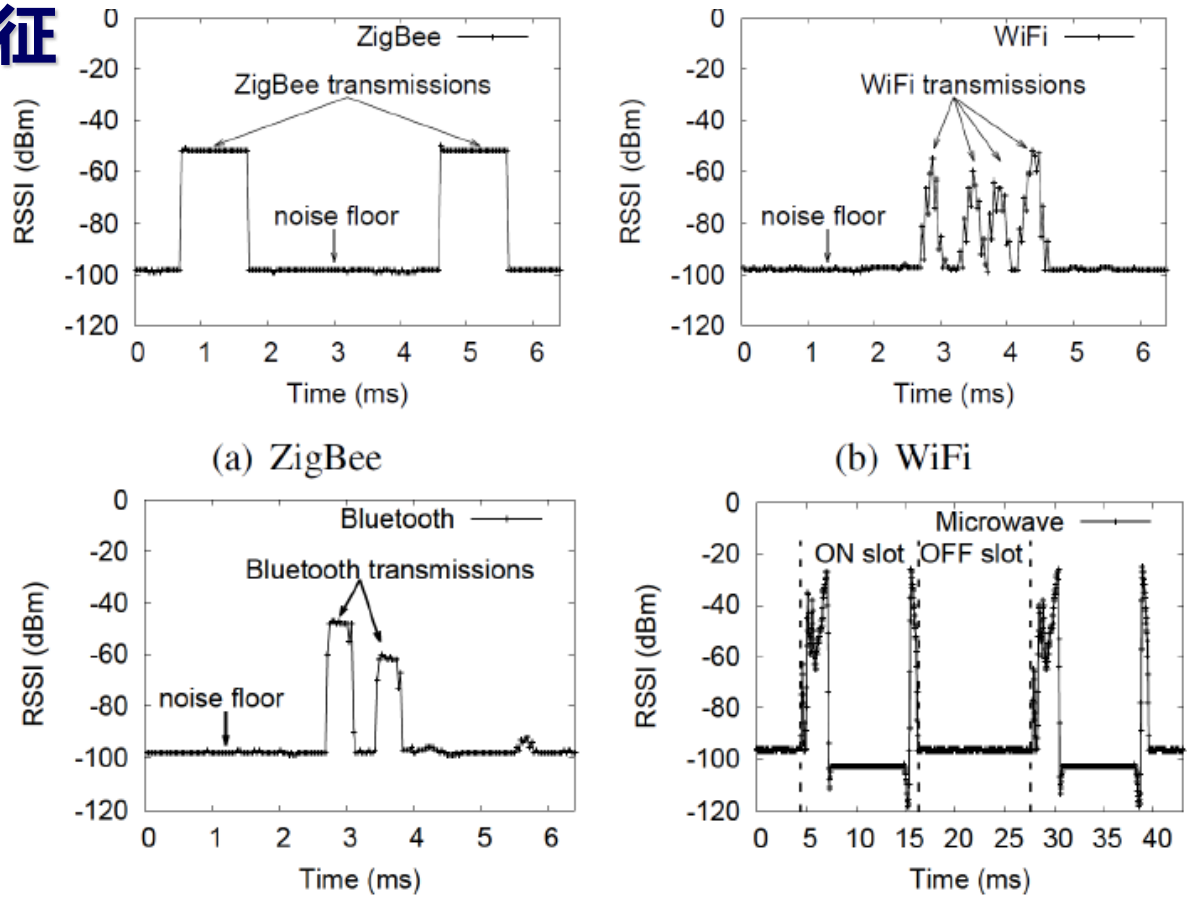
发送方发起网络：LPL MAC

- 错误唤醒问题
  - 由于WiFi，蓝牙等与802.15.4公用2.4G频段，节点可能会被信道上的WiFi信号错误唤醒
  - 持续空听(delay after receive)将引起不必要的开销



发送方发起网络：LPL MAC

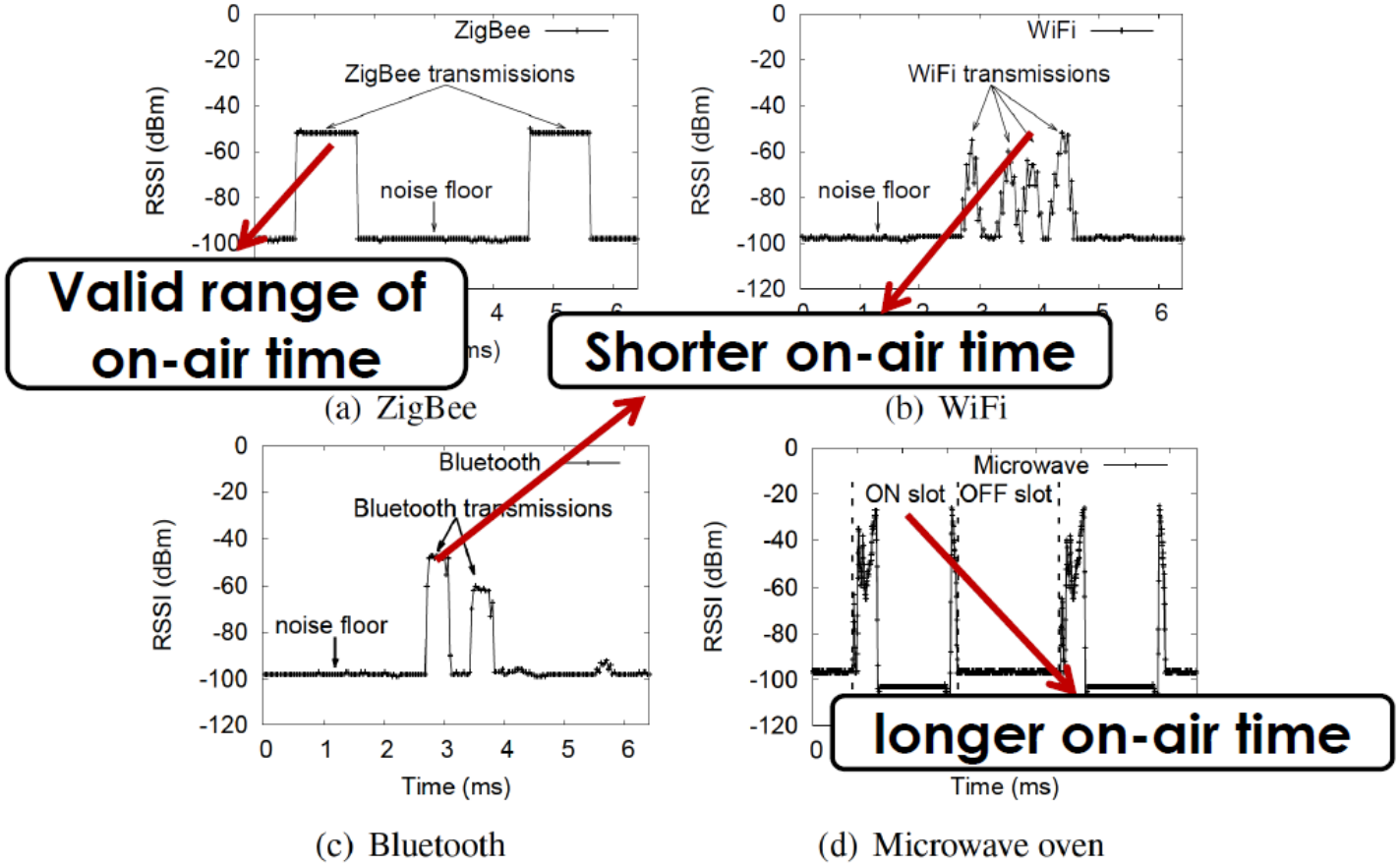
- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征
    - 广播时间
    - 数据包间隔
    - 峰均比
    - RSSI是否可能小于噪声强度



(a) ZigBee (b) WiFi (c) Bluetooth (d) Microwave oven

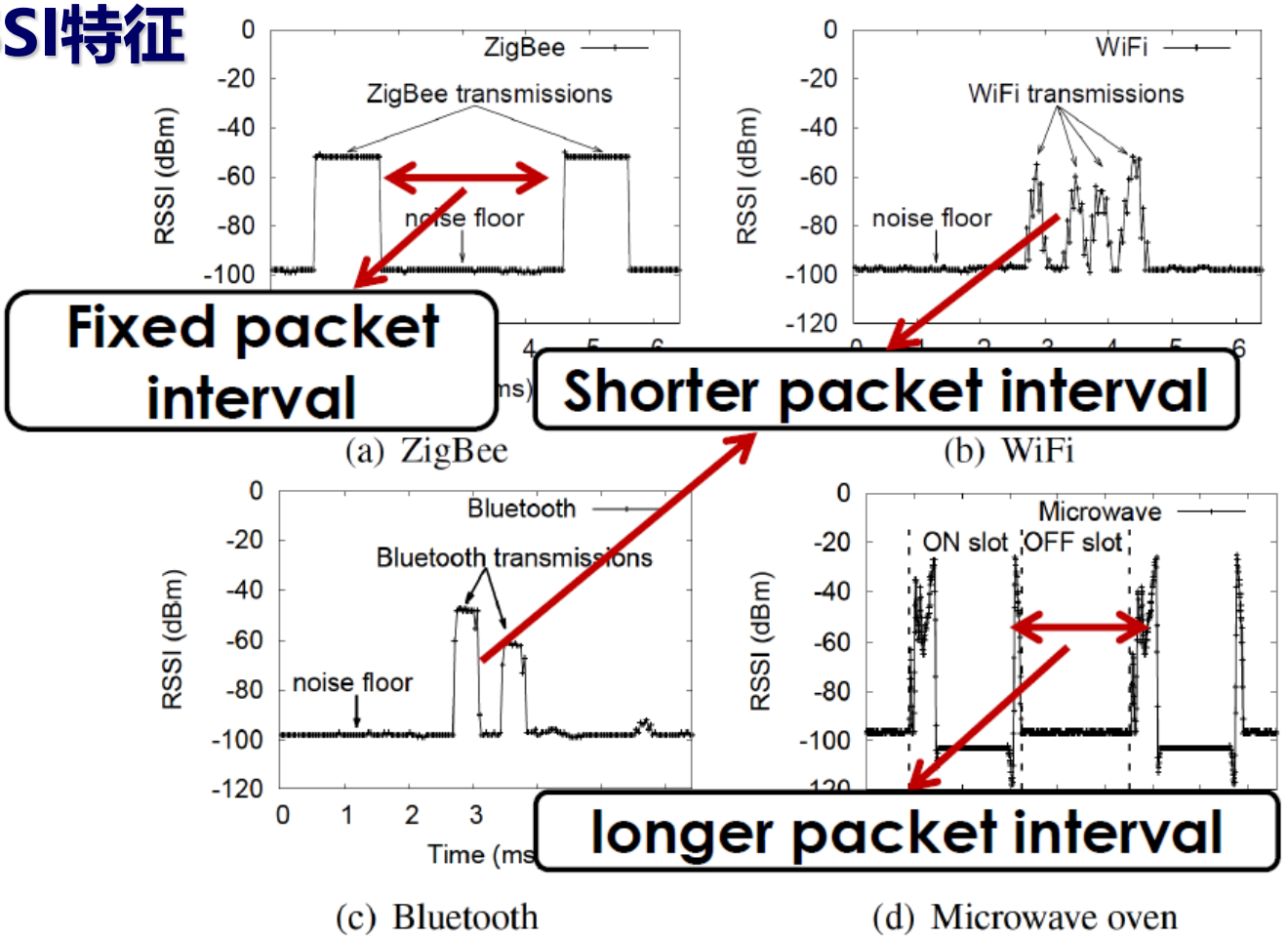
发送方发起网络：LPL MAC

- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征
    - 广播时间



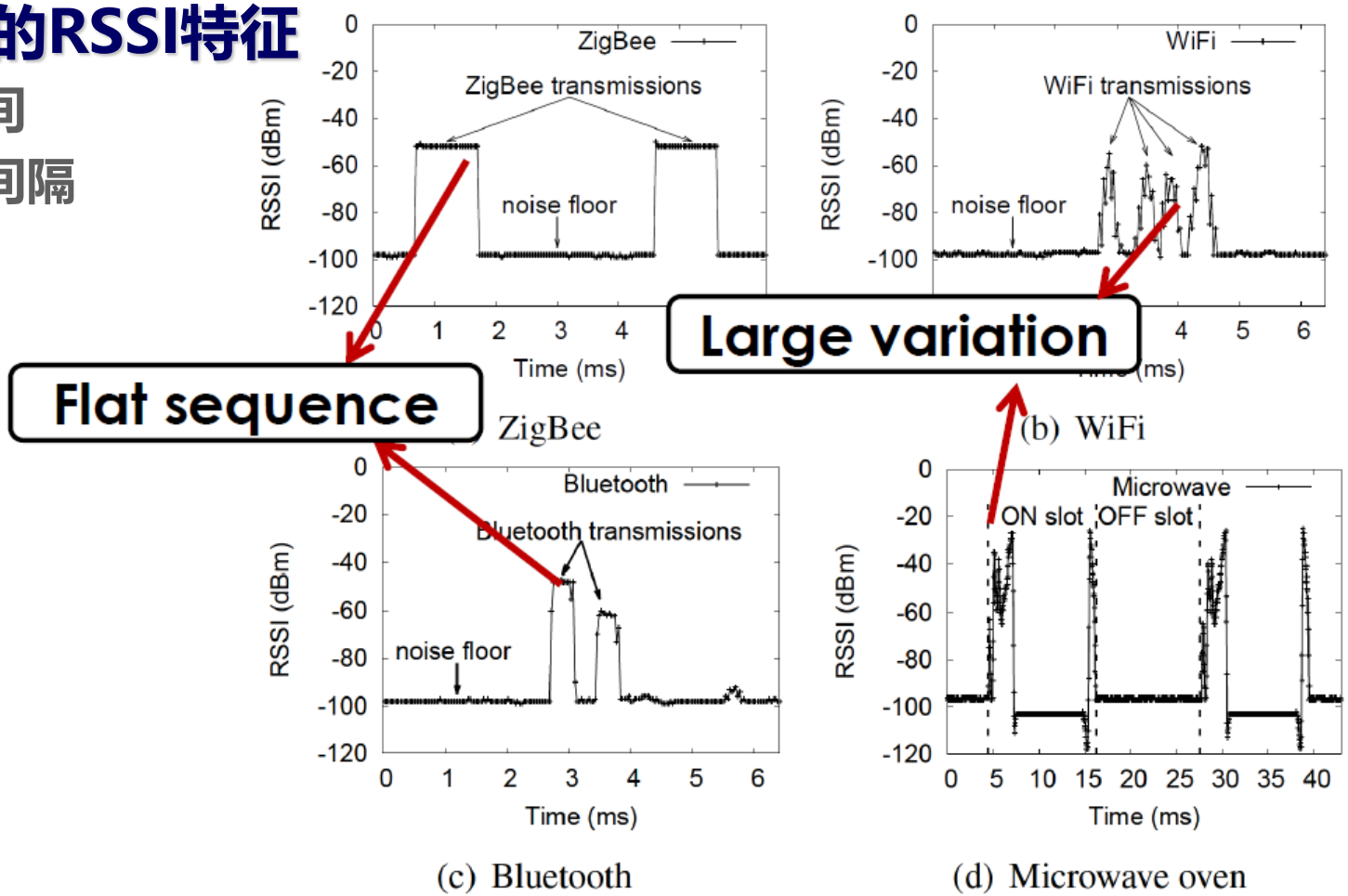
发送方发起网络：LPL MAC

- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征
    - 广播时间
    - 数据包间隔



发送方发起网络：LPL MAC

- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征
    - 广播时间
    - 数据包间隔
    - 峰均比

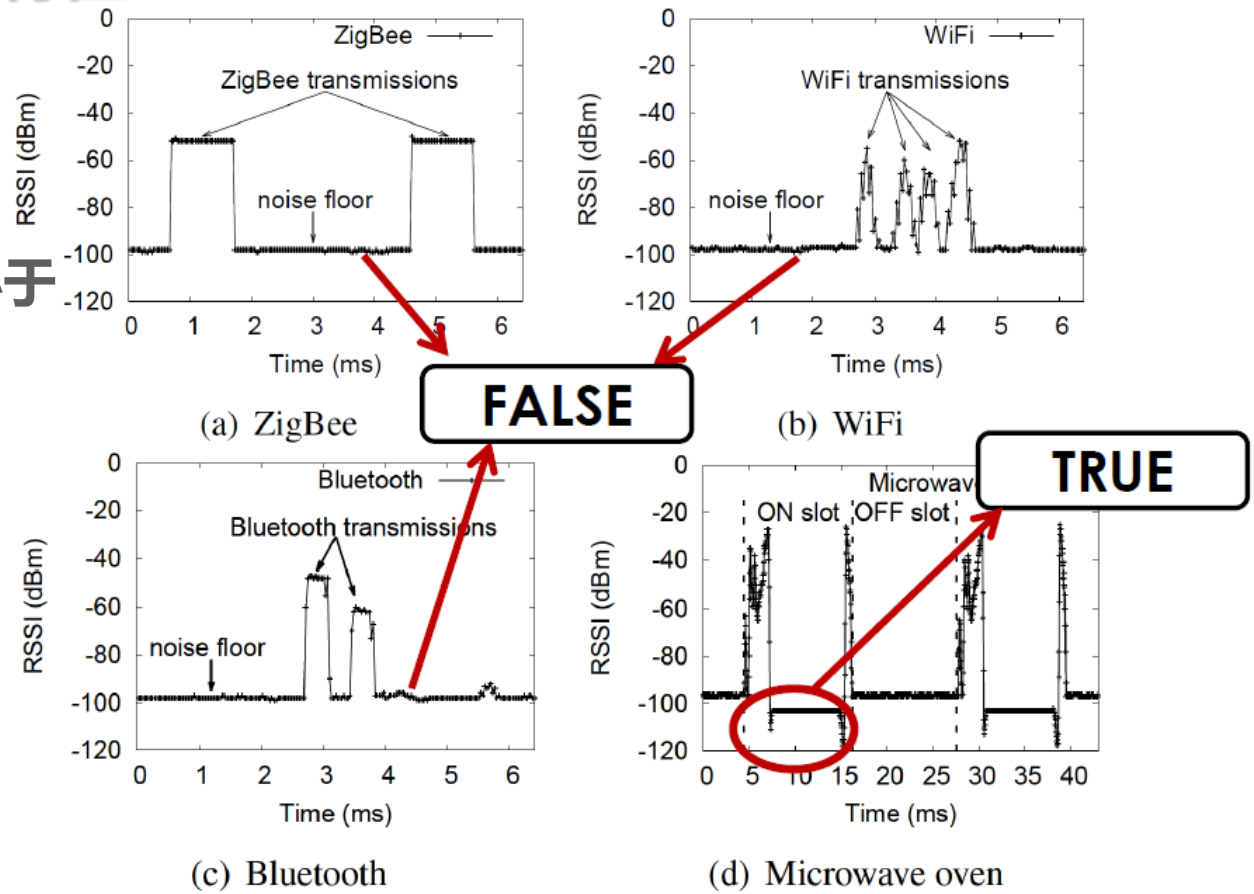


ZigBee (b) WiFi

(c) Bluetooth (d) Microwave oven

发送方发起网络：LPL MAC

- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征
    - 广播时间
    - 数据包间隔
    - 峰均比
    - RSSI是否可能小于噪声强度



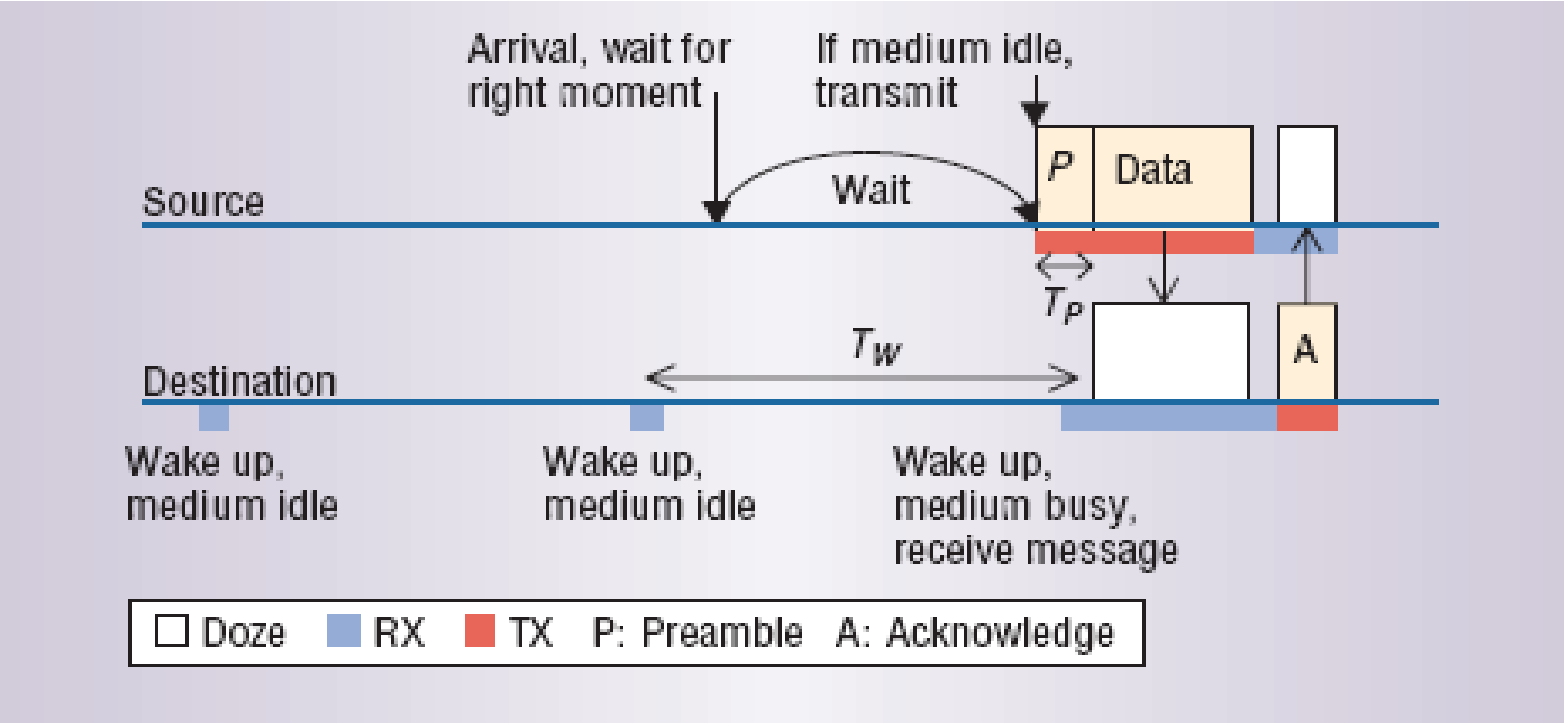
发送方发起网络：LPL MAC

- 错误唤醒问题可行解决方法
  - 通过RSSI特征来区分ZigBee信号与其它微波信号，在CCA时判断是否为ZigBee信号，是则醒过来收包，否则直接睡眠
  - 可以利用的RSSI特征

协议\RSSI特征	广播时间	数据包间隔	峰均比	RSSI是否会小于噪声强度?
ZigBee	[576, 4256]μs	2.8ms or 192μs	≤ 1.3	FALSE
WiFi	[194, 542]μs	≥ 28μs	≥ 1.9	FALSE
Bluetooth	366μs	NA	≤ 1.3	FALSE
Microwave	10ms	10ms	≥ 2.9	TRUE

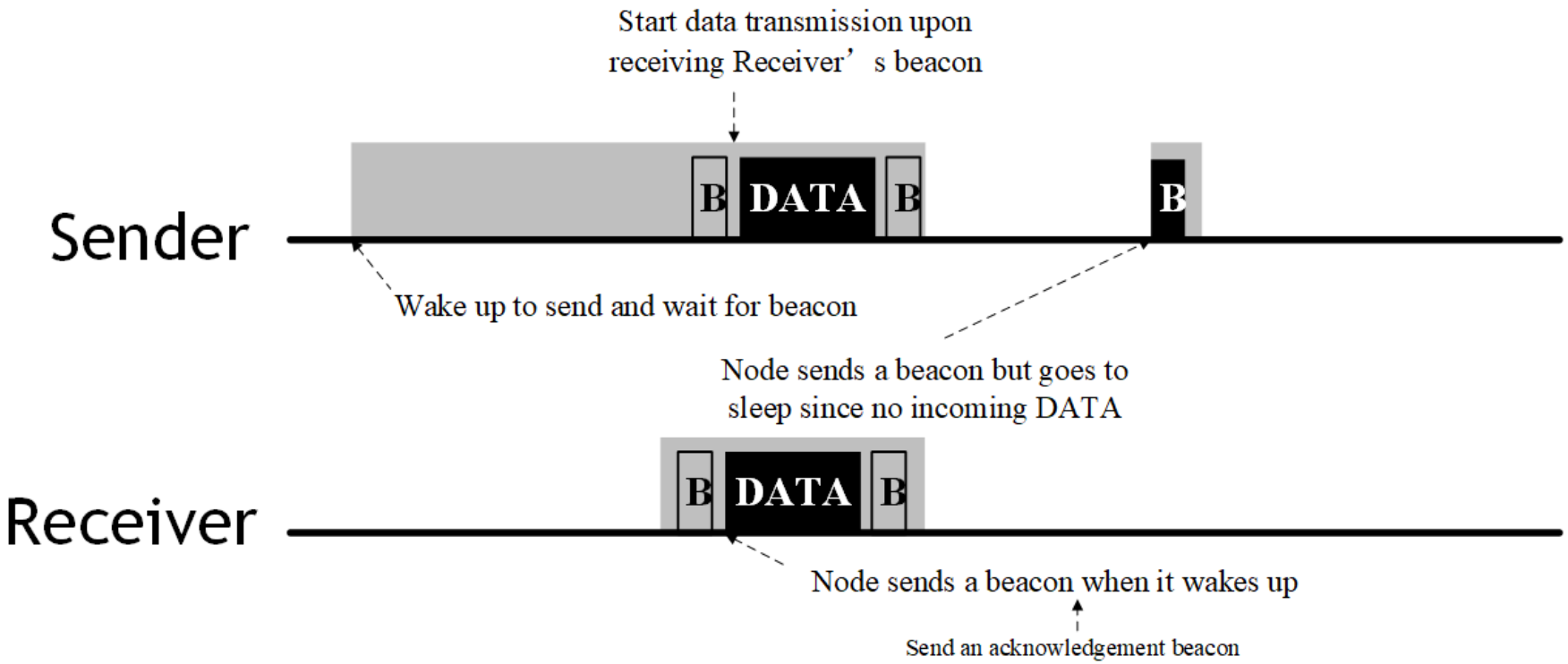
发送方发起网络： WiseMAC

- 进一步减小发送方发送探测包的开销
  - WiseMAC中，节点学习其邻居节点苏醒模式，并在其快苏醒时发送探测包



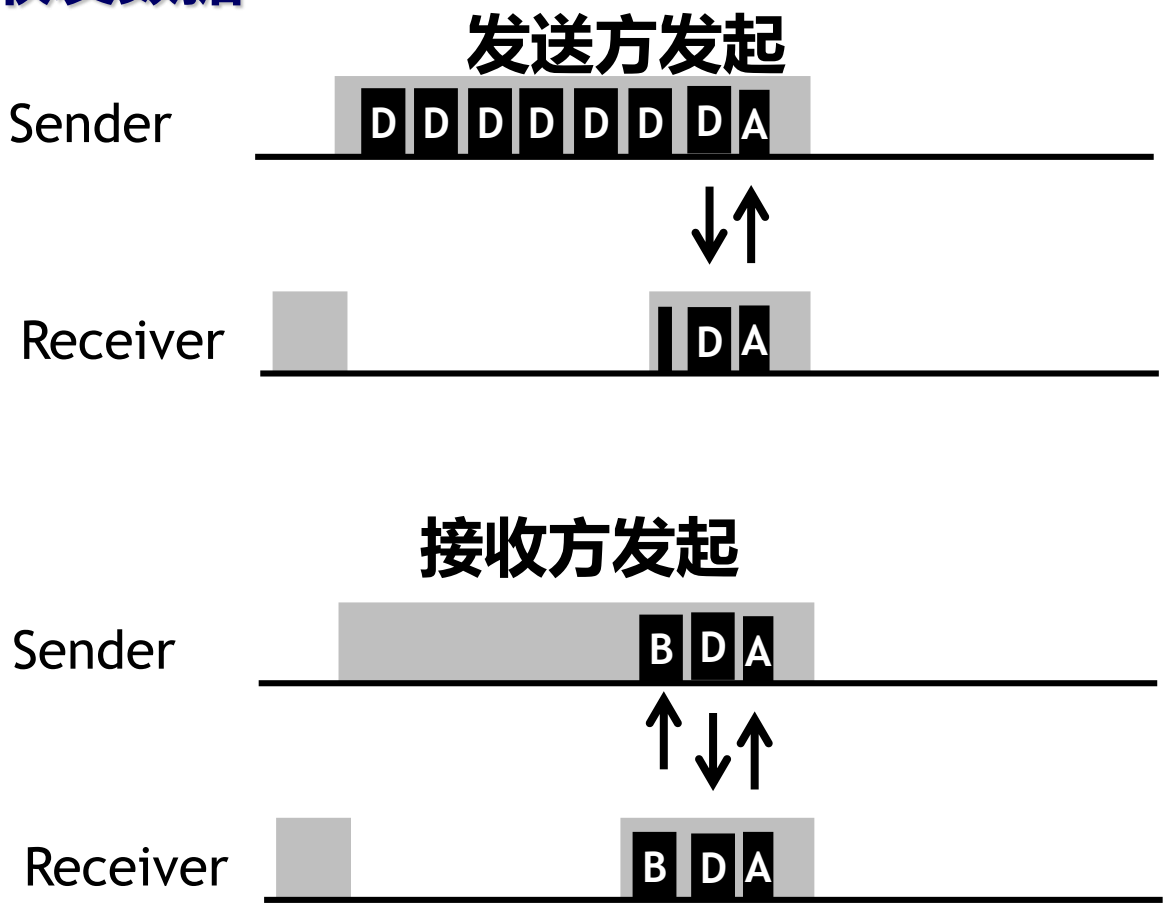
接收方发起网络：RI-MAC

- RI-MAC 工作机制
  - 发送者如果有数据需要发送，一直处于侦听状态，直到收到接收者的探测包才可以发送数据
  - 接收者周期性醒来，并发送探测包



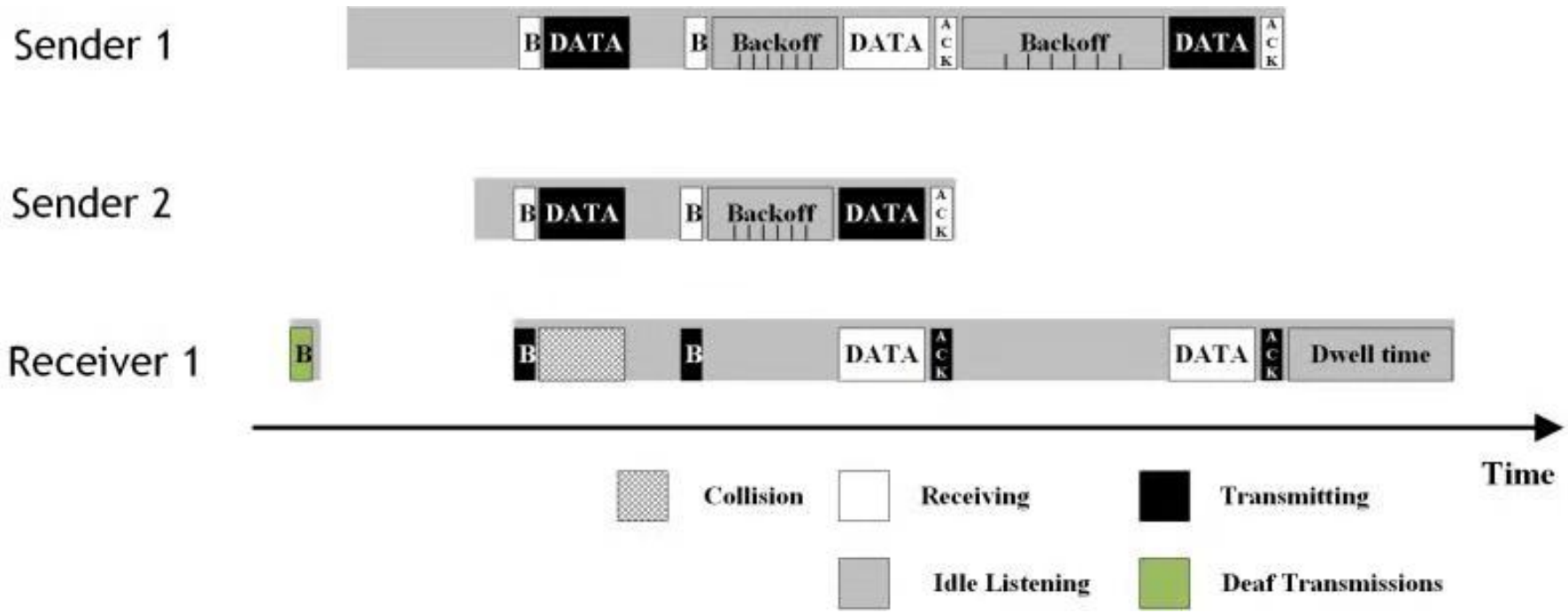
接收方发起网络：RI-MAC

- 接收方发起 vs. 发送方发起
  - 发送方发起的MAC机制要持续占用信道，导致其他邻居无法收发数据



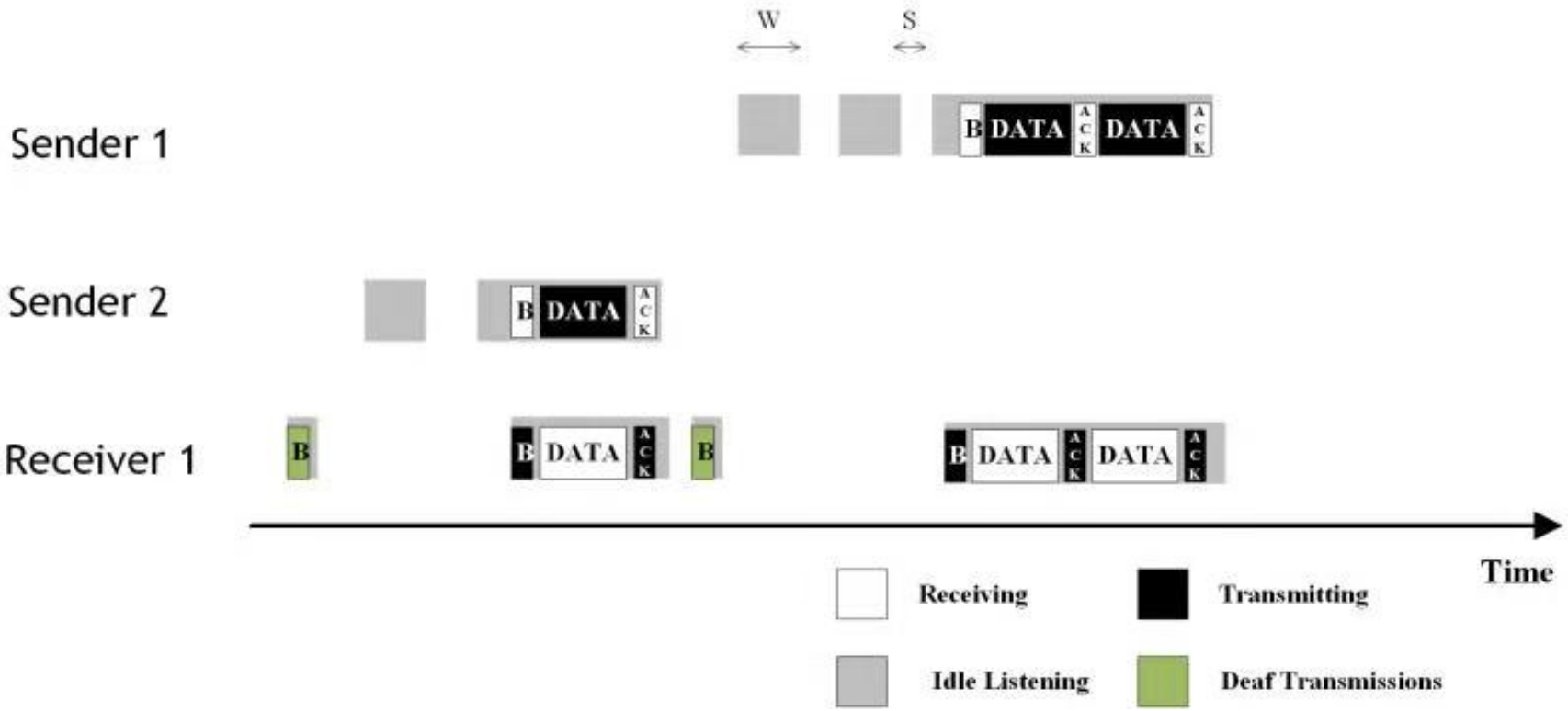
接收方发起网络：RI-MAC

- 数据冲突问题:
  - 冲突避免
  - 二进制指数退避



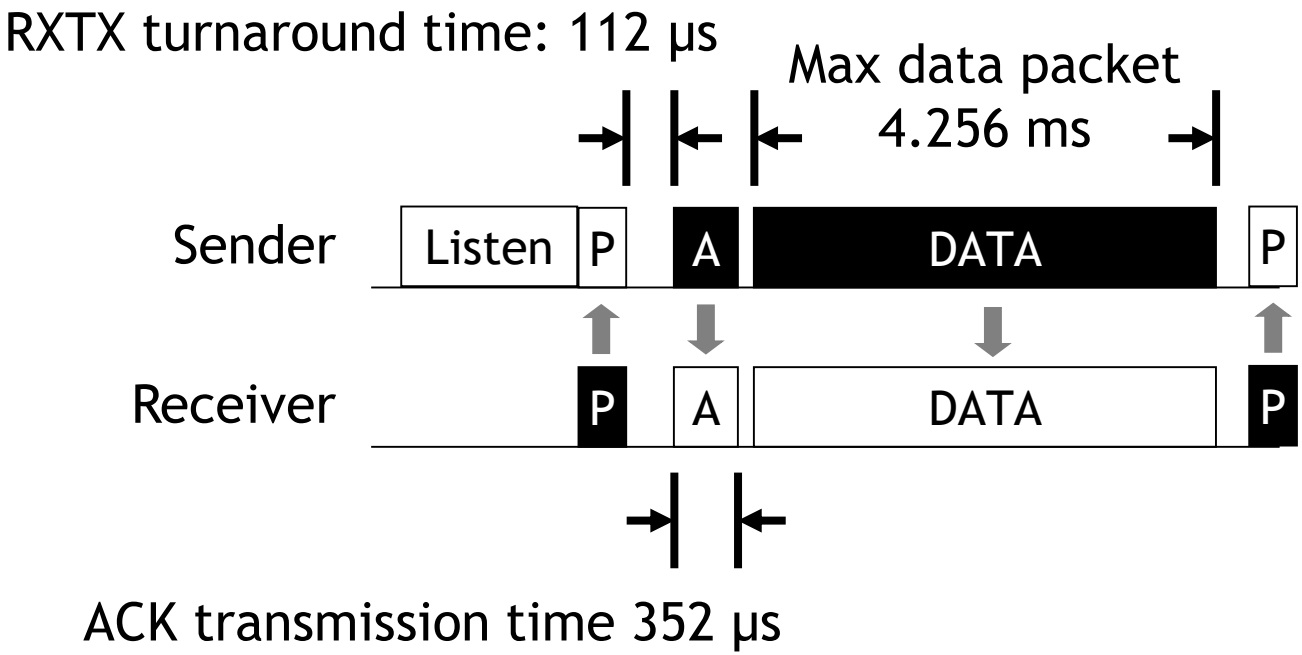
接收方发起网络：EE-RI-MAC

- 进一步减少RI-MAC发送者的能耗
  - 受到X-MAC的启发
- 在发送者空闲侦听的过程中，时醒时睡，而不是持续侦听



接收方发起网络：A-MAC

- 进一步减少接收者的能耗
- RI-MAC问题：Receiver发送probe后需要等待一段时间试图去接收可能收到的数据
- 解决方法：A-MAC可以减少等待时间，通过backcast机制测试网络中是否有潜在的sender



目录

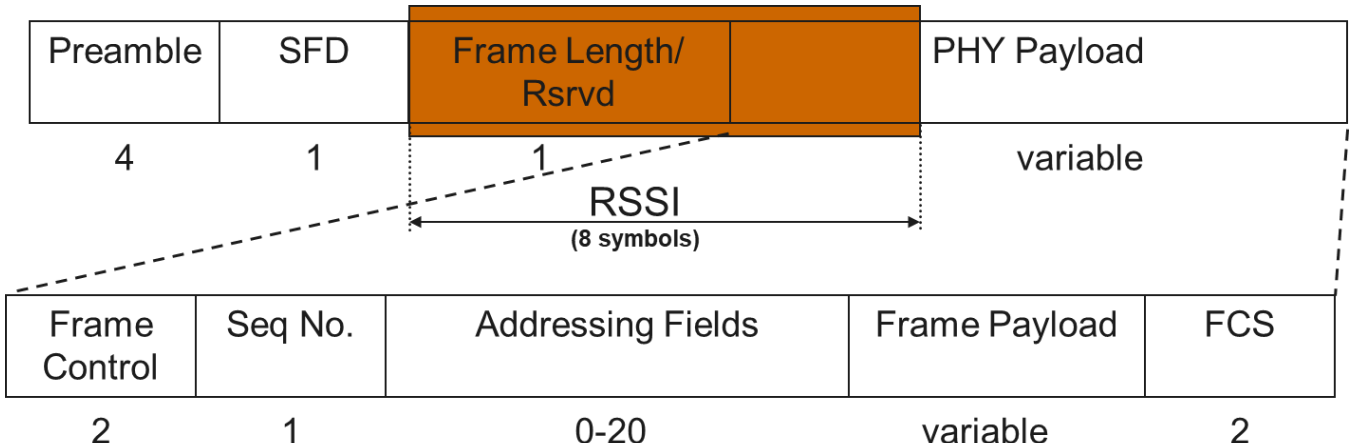
- 概述
- 基于IEEE802.15.4的传感网协议架构
  - IEEE 802.15.4技术物理层
  - 低功耗MAC
  - 链路质量估计
  - 网络层协议
  - 传感网应用案例
- Bluetooth
- 研究进展

链路质量估计: 重要性与目标

- 链路质量估计是路由协议的基础
  - 一个好的路由协议会选择链路质量好的链路进行通信
- 链路质量的评估指标
  - RSSI/SNR
  - LQI
  - PRR
  - BER
  - Symbol Error Rate

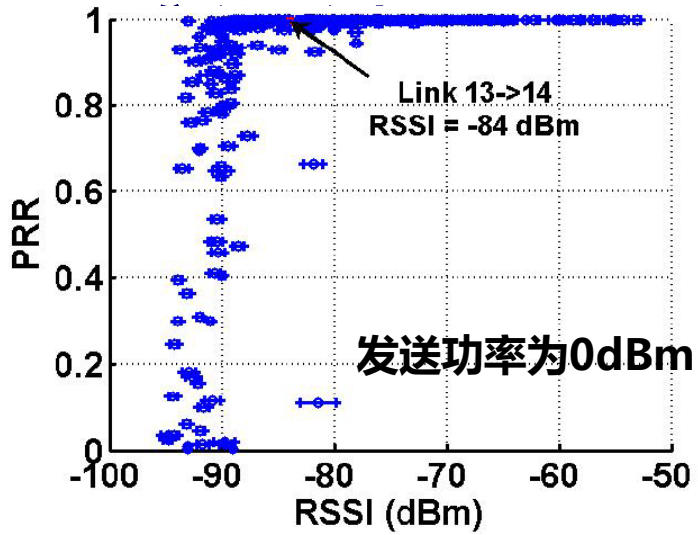
链路质量估计: RSSI

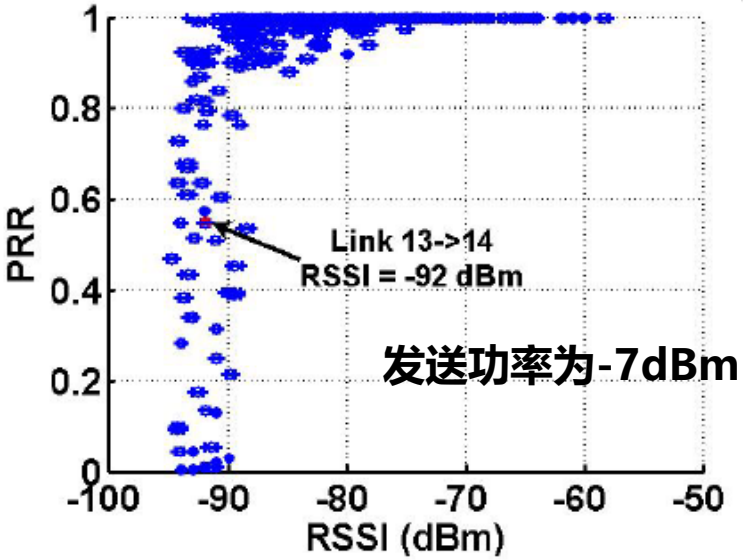
- 接收信号强度(Radio Signal Strength Indicator, RSSI)
  - 是接收信号总能量的度量
- RSSI 获得方法
  - 计算方法:
    - 8个symbol(128us)周期的信号强度平均值
    - CC2420记录在RSSI VAL寄存器中
    - CC2420的RSSI计算公式P = RSSI VAL + RSSI OFFSET
      - RSSI_OFFSET≈-45dBm
  - Per byte: 读寄存器
  - Per packet：根据SFD后8个symbol计算
  - 经验值
    - -95dbm ~-55dBm



链路质量估计: RSSI

- RSSI 与PRR之间的关系
  - 当RSSI>-87dBm时, PRR>85%, 链路质量好
  - 当RSSI<-92dBm时, PRR<50%, 链路质量差
  - 当RSSI低于-87dBm时，进入灰色区域，PRR非常不稳定

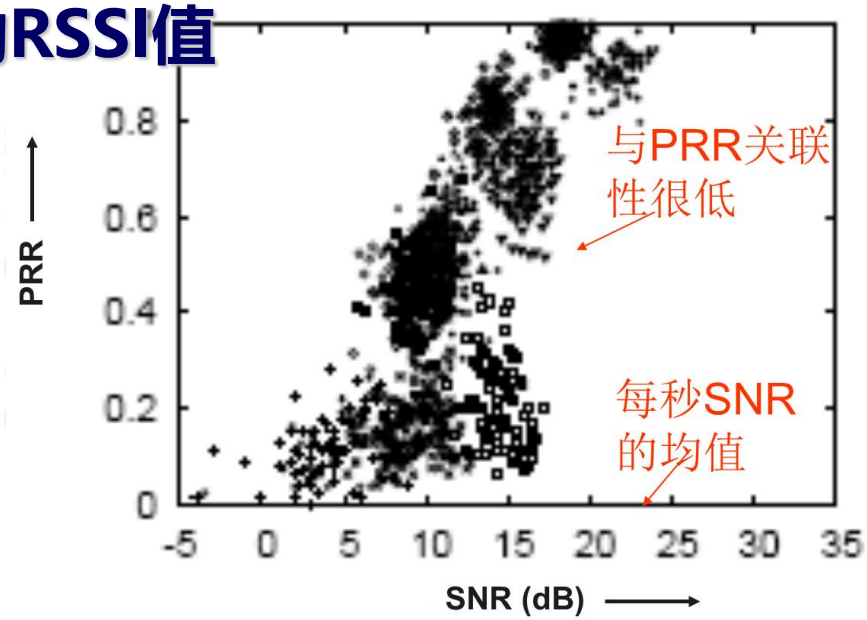




- RSSI问题
  - 灰色区域测不准
  - 和接收器灵敏度有关

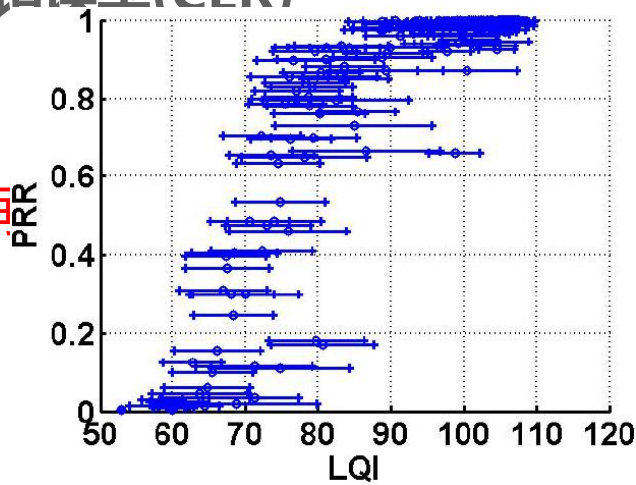
链路质量估计: SNR

- SNR (Signal to Noise Ratio)
  - 信号功率与噪声功率的比率, 单位dB
  - SNR=10log(S/N), S为信号功率，N为噪声功率
- 802.15.4 硬件不支持直接读取SNR的接口
- 获取方法(以cc2420为例)
  - 噪声功率(Noise Floor): RSSI寄存器在不收包时的读数
    - 通常为-98dBm 左右
  - 信号功率：收到的数据包的RSSI值
  - 例如，RSSI为-75dBm时，SNR为23dB
- SNR与PRR的关系
- SNR经验值: 3~43dB

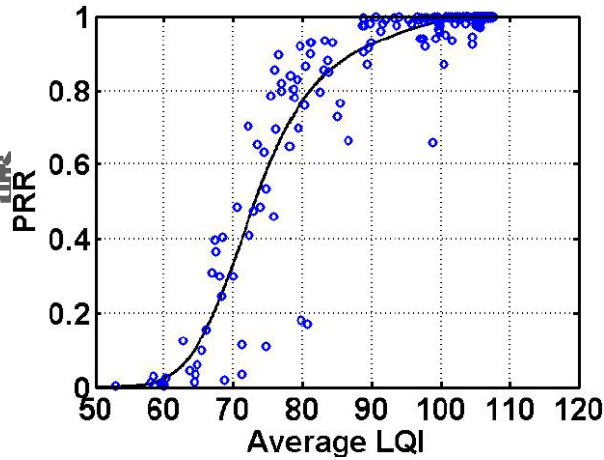


链路质量估计: LQI

- LQI (Link Quality Indicator)
  - 计算方法
    - 帧定界符（SFD）之后8个symbol的码元错误率(CER)
    - LQI=(CORR-a)*b
      - CORR为数据包SFD后8个symbol的平均correlation value，在0~255之间
  - LQI经验值：50(最差)~110(最好)
  - 在CTP中，认为LQI高于105的链路为高质量链路



- LQI与PRR之间的关系
  - 单个LQI值与PRR关联性低(上图)
    - 例如LQI 85 对应的PRR从10%到100%不等
  - 平均LQI与PRR关联性高(下图)

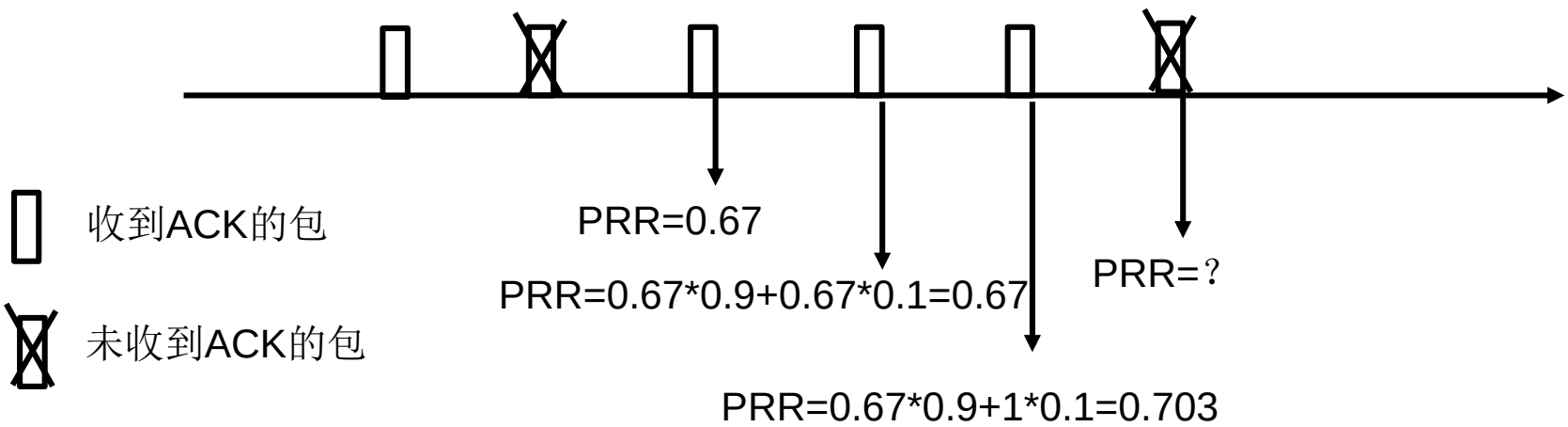


链路质量估计: PRR

- PRR (Packet Reception Ratio)
  - 数据包成功接收率 （不考虑重传）
  - 衡量链路质量的最终指标
- PRR估计方法：一段时间内，成功接收的包数与发送的总数的比率
  - 通过Beacon包
    - 缺陷：包长不同，预测不准
  - 通过数据包
    - 通过receiver feedback
    - 通过ack计数

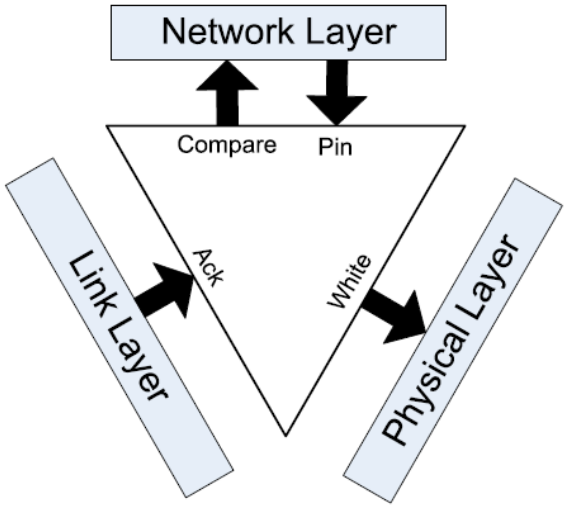
链路质量估计: PRR

- 通过ack计数的PRR估计方法
  - 固定时间窗口中, 统计收到的ACK数据包个数与发送的数据包总数的比值
  - 指数加权移动平均(Eexponentially weighted moving average EWMA)
    - PRR = PRR*α+ (1-α)PRR_{new}
    - 可调节参数：滑动窗口W，与权重α
      - 示例中，W=3,α=0.9



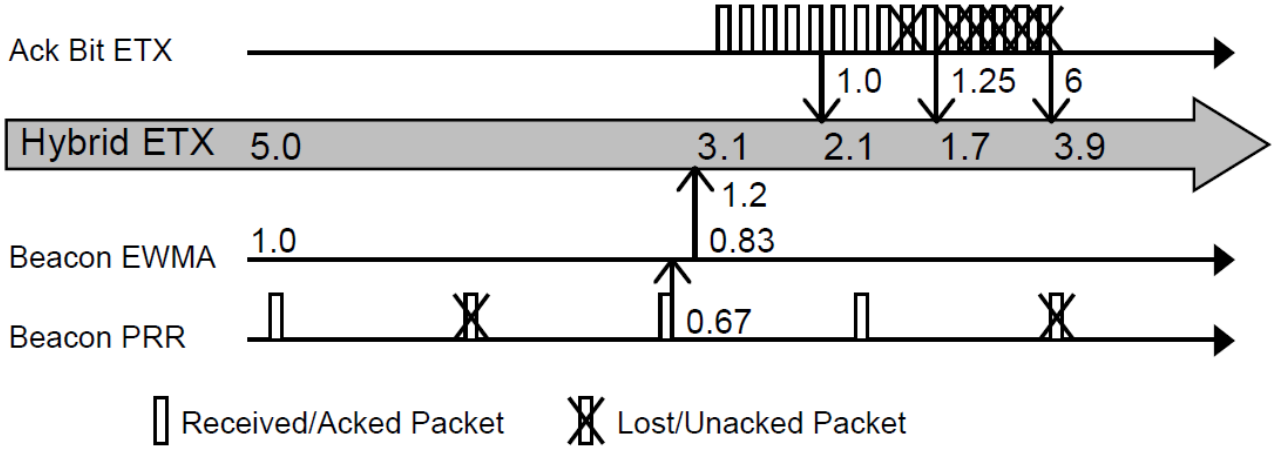
链路质量估计: 基于跨层特征的估计

- 4-bit 链路质量估计的信息来源
  - **物理层(1 bit)：数据包解码质量情况**
    - White bit: 设置为1表示收到的数据包中每个符号误码率很低
    - Bit = 0, if LQI < 105
  - **链路层(1 bit)：数据包ACK情况**
    - Ack bit: 设置为1表示收到L2层的ack，上一个数据包成功发送，为0则表示发出去的包可能丢了
  - **网络层(2 bits)：标注相对重要的链路**
    - Pin bit：当网络层在一个路由表的某链路上设置pin位时，链路估计器不能从表中删除该链路，直到该位清零为止
    - Compare bit：指示数据包发过来的链路是否优于现在路由表中一个或多个链路



链路质量估计: 基于跨层特征的估计

- 4-bit 链路质量估计
  - Compare bit & White bit & Pin bit: 用于补充现有的路由表更新策略
    - 当收到数据包时, 如果white bit置位, 则LE向查询查询compare bit 是否置位，如果置位，LE 在路由表中没有标志pin 的条目上随机选择一条更新
  - ACK bit: 用于提升ETX估计的准确度
    - 主要思想：结合了beacon包与数据包的ack情况来估计链路ETX
    - 分别计算beacon的EWMA(2,0.5) ETX，以及数据包的EWMA(5,0.5) ETX, 再将二者的平均值作为ETX估计值



目录

- 概述
- 基于IEEE802.15.4的传感网协议架构
  - IEEE 802.15.4技术物理层
  - 低功耗MAC
  - 链路质量估计
  - 网络层协议
  - 应用案例
- Bluetooth
- 研究进展

网络层协议: CTP

- CTP
  - 多对一的路由模式
  - 基于距离向量路由协议, 选择距离最短的路径
  - 建立在CSMA MAC或LPL MAC，及链路质量估计器上
- 路由指标：ETX(expected number of transmissions)
  - 链路ETX：在一条链路上成功发送一个数据包所需的传输次数
    - $ETX_f = 1 / (d_f X d_r) \approx 1/d_f$
    - d_f: 数据包成功到达接收者的概率， d_r : 接收者回复的ACK包成功到达发送者的概率
  - 路径ETX：在一条路径上成功发送一个数据包所需的传输次数
    - 证明：路径ETX=链路ETX之和
  - 思考题： 对于两跳链路，前一跳ETX 1, 后一跳ETX 100，与前一跳ETX 100，后一跳ETX1,收包率是否一样?

网络层协议: CTP

- CTP基本工作流程
  - S 广播beacon, A B更新路由表

A

下一跳	开销
S	1

E

下一跳	开销

B

下一跳	开销
S	3

F

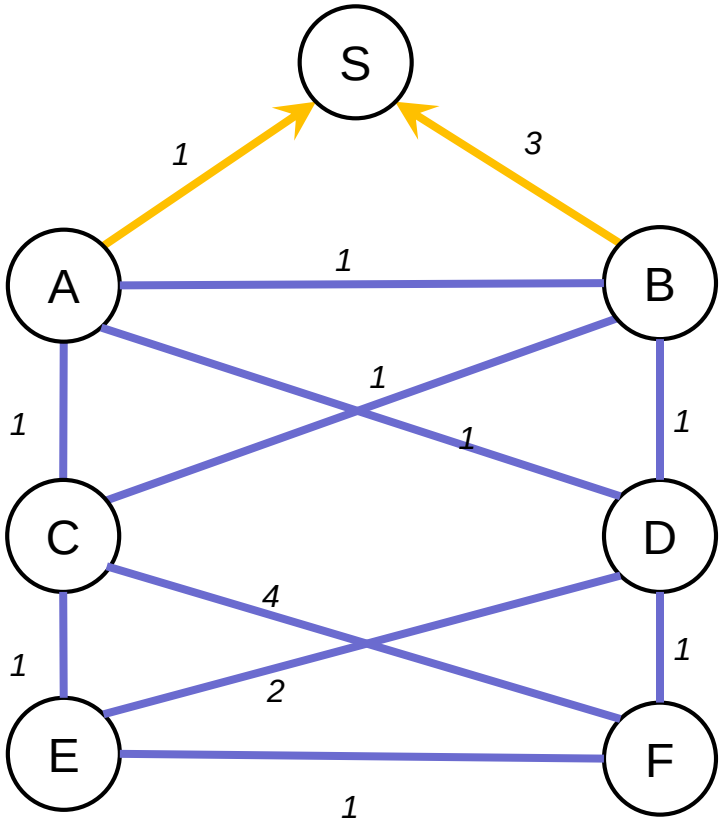
下一跳	开销

C

下一跳	开销

D

下一跳	开销



网络层协议: CTP

- CTP基本工作流程
  - A 广播beacon, B 发现走A路径更优，更新父节点为A；其余邻居节点C和D分别更新路由

A

下一跳	开销
S	1

E

下一跳	开销

B

下一跳	开销
A	2
S	3

F

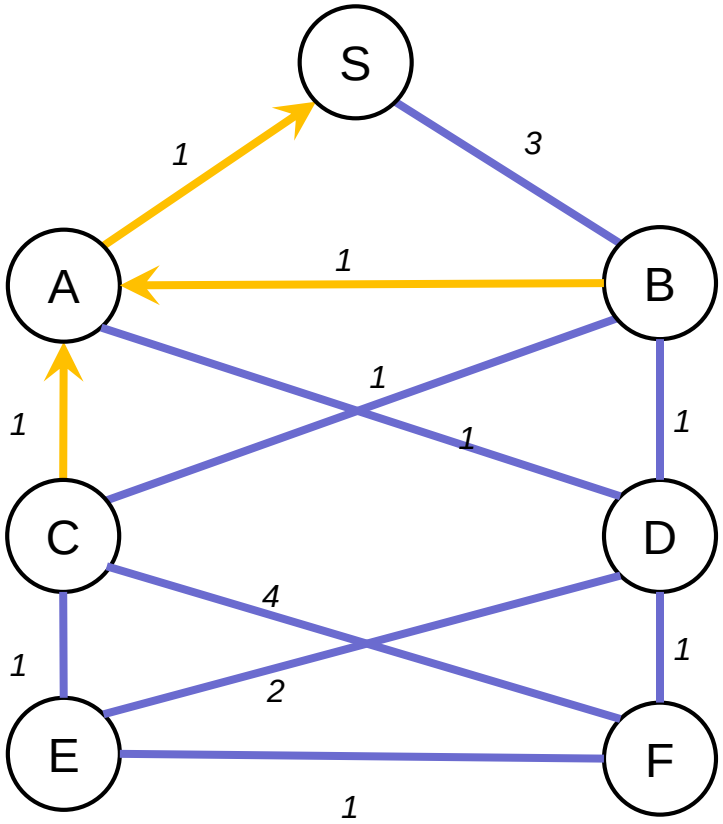
下一跳	开销

C

下一跳	开销
A	2

D

下一跳	开销
A	2



网络层协议: CTP

- CTP基本工作流程
  - 重复此过程

A

下一跳	开销
S	1

B

下一跳	开销
A	2
S	3

C

下一跳	开销
A	2
B	3

D

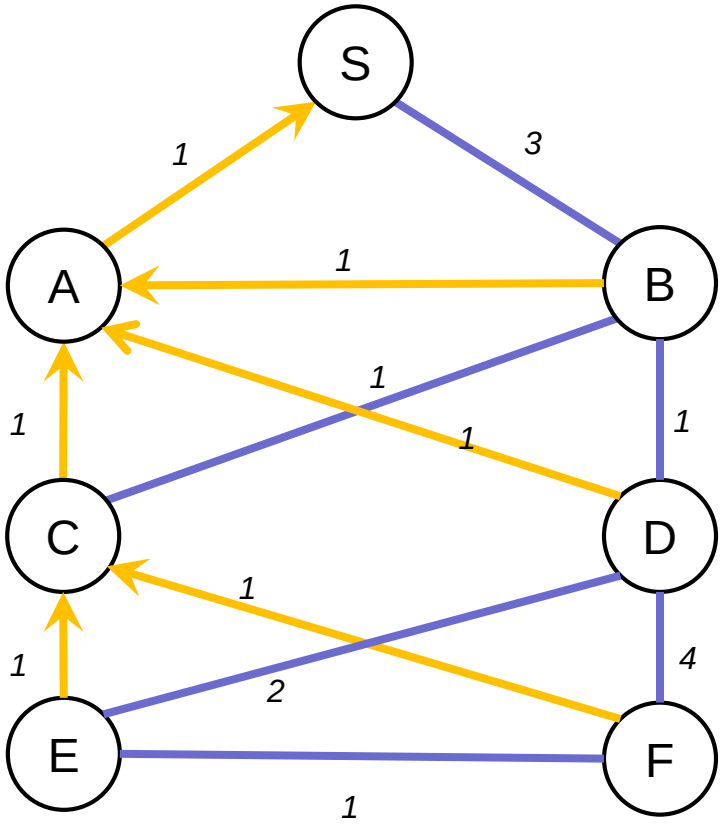
下一跳	开销
A	2
B	3

E

下一跳	开销
C	3
D	4

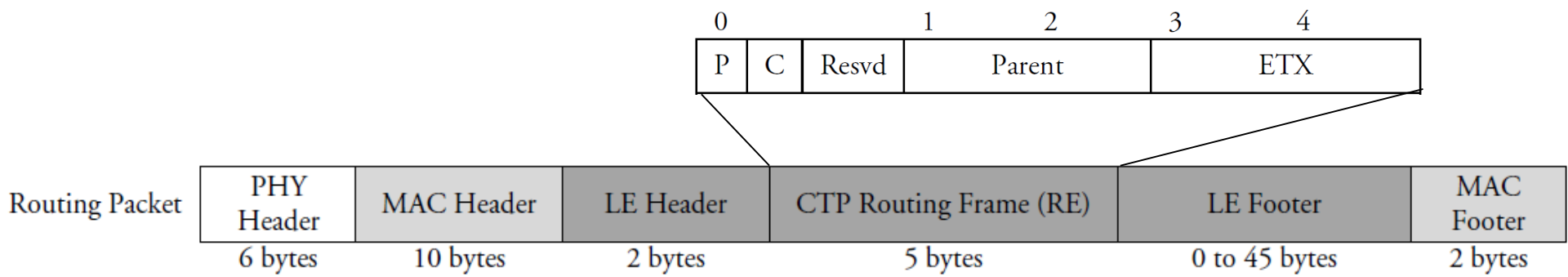
F

下一跳	开销
C	3
E	4
D	6



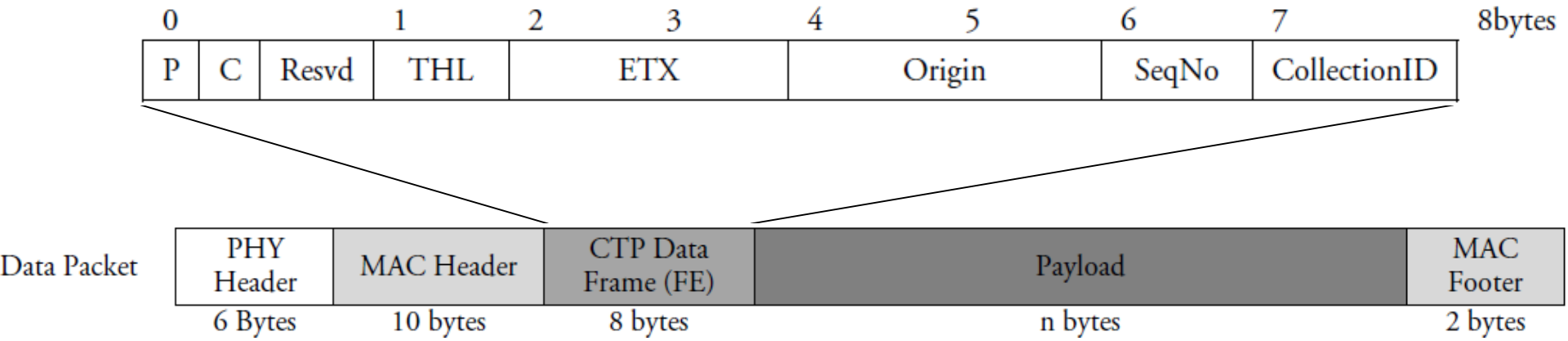
网络层协议: CTP

- CTP 路由帧(beacon)格式
  - P: 路由信息请求标志位。置位, 则收到该包的节点将发送路由帧更新路由信息
  - C: 拥塞标志位。置位，表示上一个CTP包丢失，发生拥塞
  - Parent: beacon发送节点的parent
  - ETX: 发送节点的path ETX



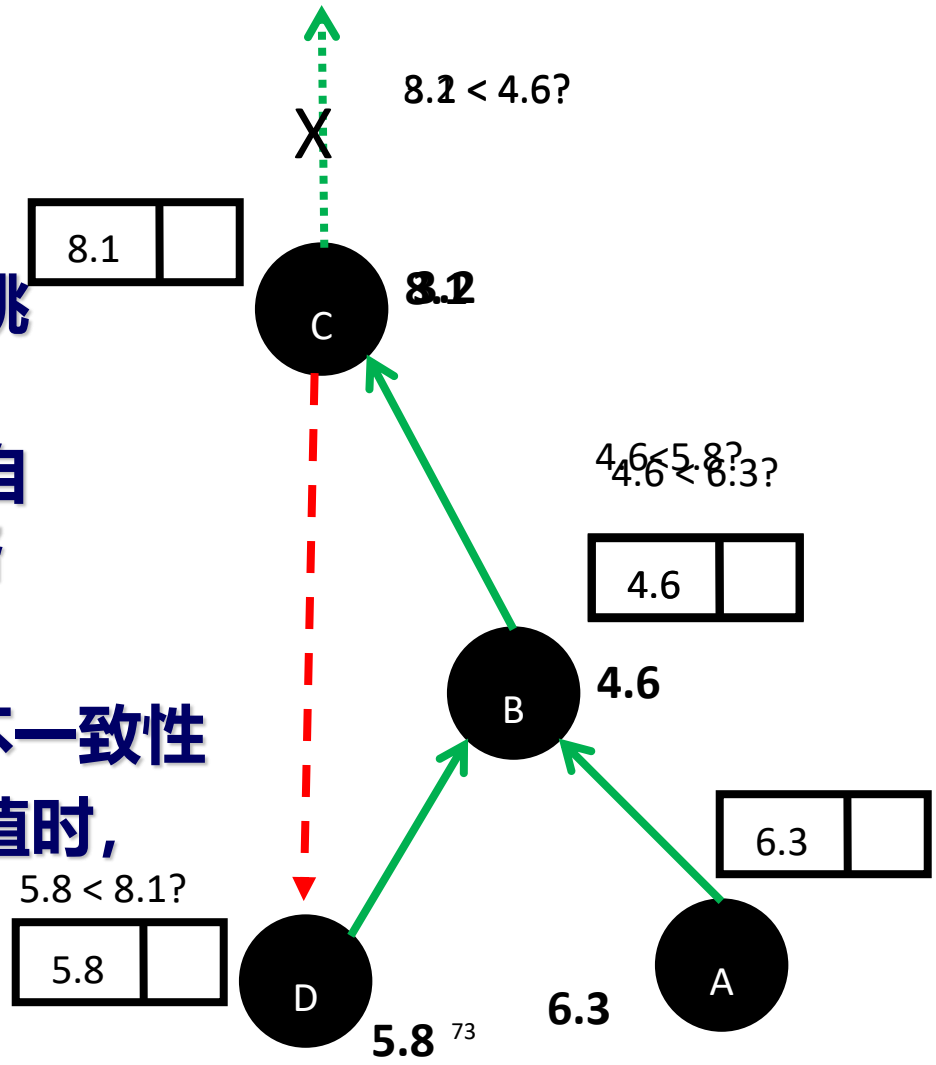
网络层协议: CTP

- CTP数据帧格式
  - P: 路由信息请求标志位
  - C：拥塞标志位
  - THL: Time Has Lived, 数据包生成时为0, 每经过一跳增加1
  - ETX: 发送节点的path ETX
  - Origin：源节点ID
  - SeqNo: 序列号



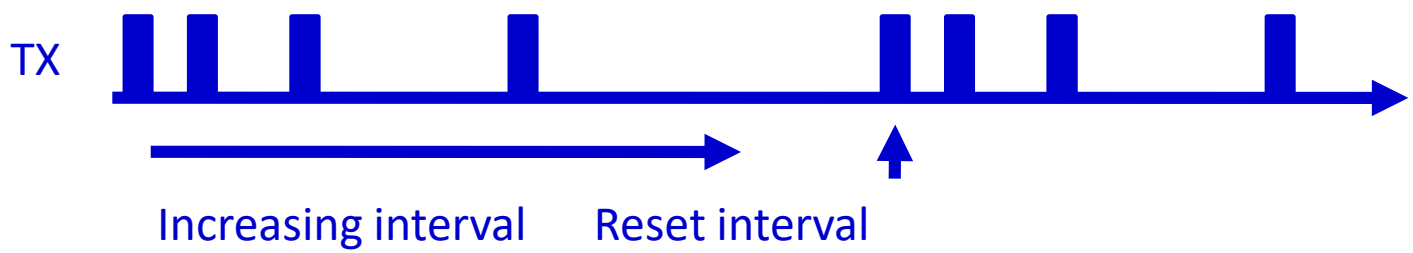
网络层协议: CTP

- 路由回路问题
  - 发生原因
    - 路由信息陈旧, 没有及时更新
- 如何发现
  - 数据包中的节点ETX应该逐跳减小
  - 当节点收到的数据包ETX比自己的ETX低时, 说明发生回路
- 两种解决机制
  - 通过广播beacon包来解决不一致性
  - 当数据包ETX高于设定的阈值时, 不再发送该数据包



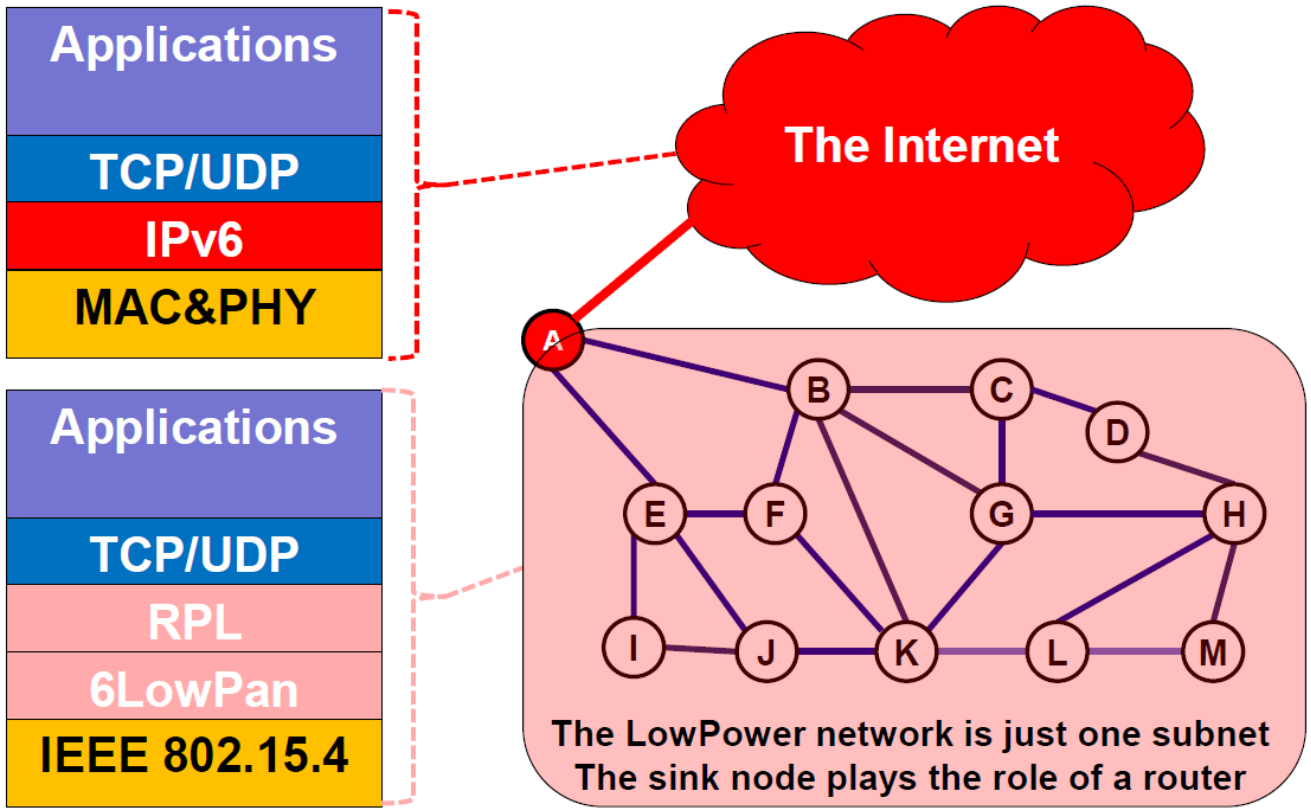
网络层协议: CTP

- Beacon频率问题
  - Beacon用于更新ETX及路由信息, 频率过高, 路由信息准确度高, 但开销大; 频率过低, 则无法及时更新路由
- 解决方法
  - 自适应的调节机制(Adaptive beaconing), beacon间隔从小到大增加
  - 当检测到网络不一致性时，将beacon间隔设成最小
    - E.g. ,发生路由回路时



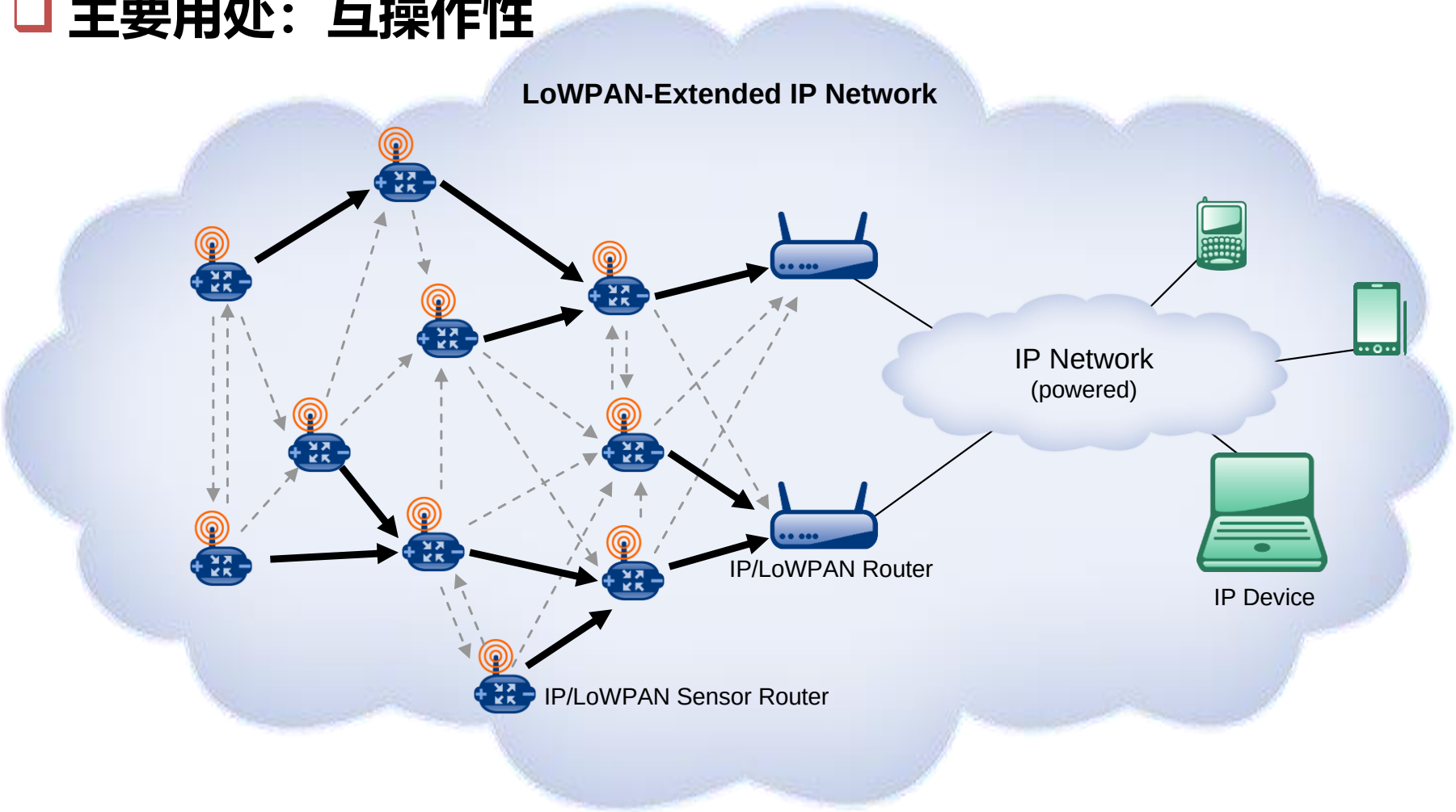
网络层协议: RPL

- 低功耗有损网络中的IPv6路由协议 (IPv6 Routing Protocol for Low Power and Lossy Networks)
  - IPv6与IEEE 802.15.4的结合: 6LoWPAN
  - 基于6LoWPAN的路由策略：RPL



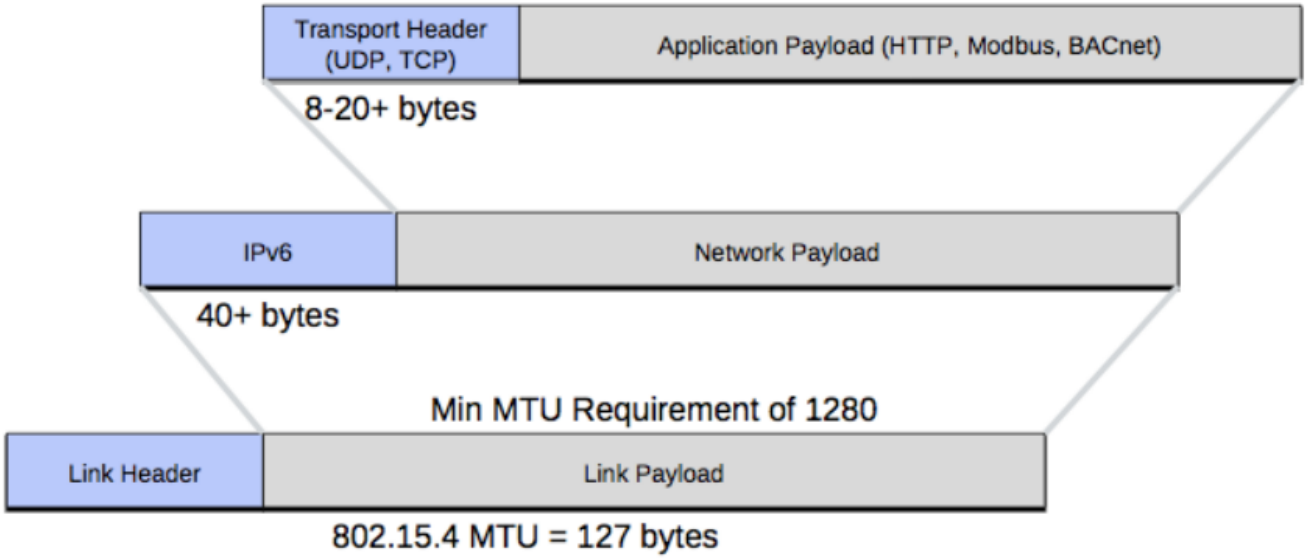
6LoWPAN简介

- 专为低功耗易失网络（Low power and Lossy Networks, LNN）而设计的IPv6解决方案
- 主要用处：互操作性



6LoWPAN：适配层功能

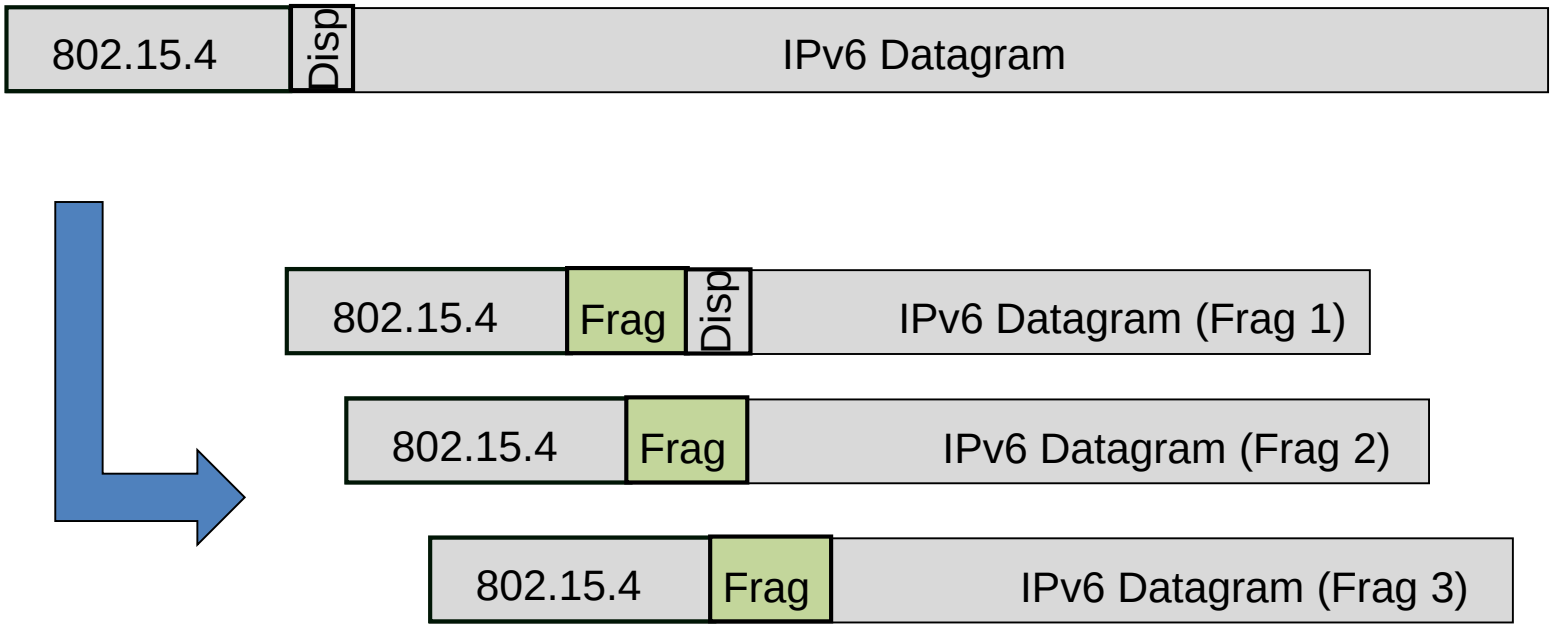
- 目的：将一个大体积的IPv6数据报文映射到多个15.4报文里



- 两种主要技术
  - **分片与重组:** IPv6最小MTU(1280B)>>802.15.4 MTU(127B)
  - **头部压缩:** 48 B UDP IPv6 报头，即使15.4的有效载荷达到最大MTU， 占用超过40%的802.15.4有效载荷

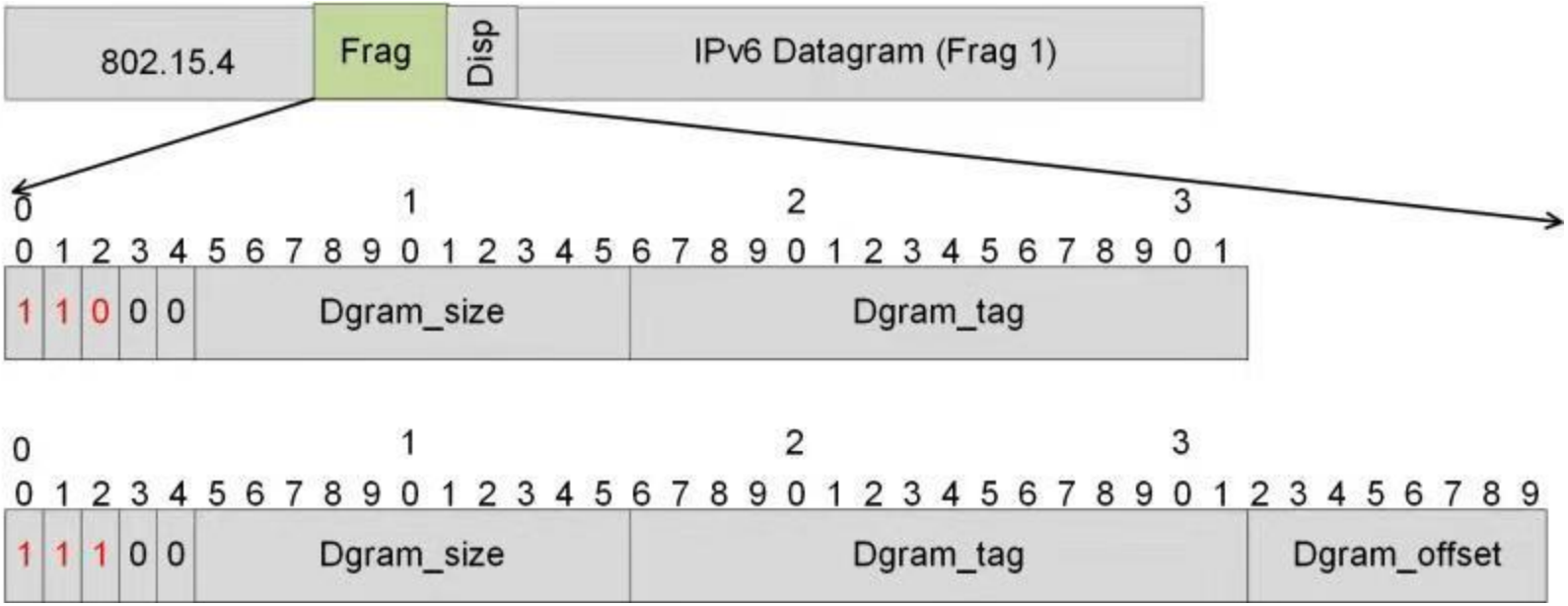
6LoWPAN：分段与重组

- 802.15.4 数据包最大127 B
- IPv6 需要链路层支持最小1280B的数据包 [RFC 2460]
  - 6LoWPAN必须支持分段与重组



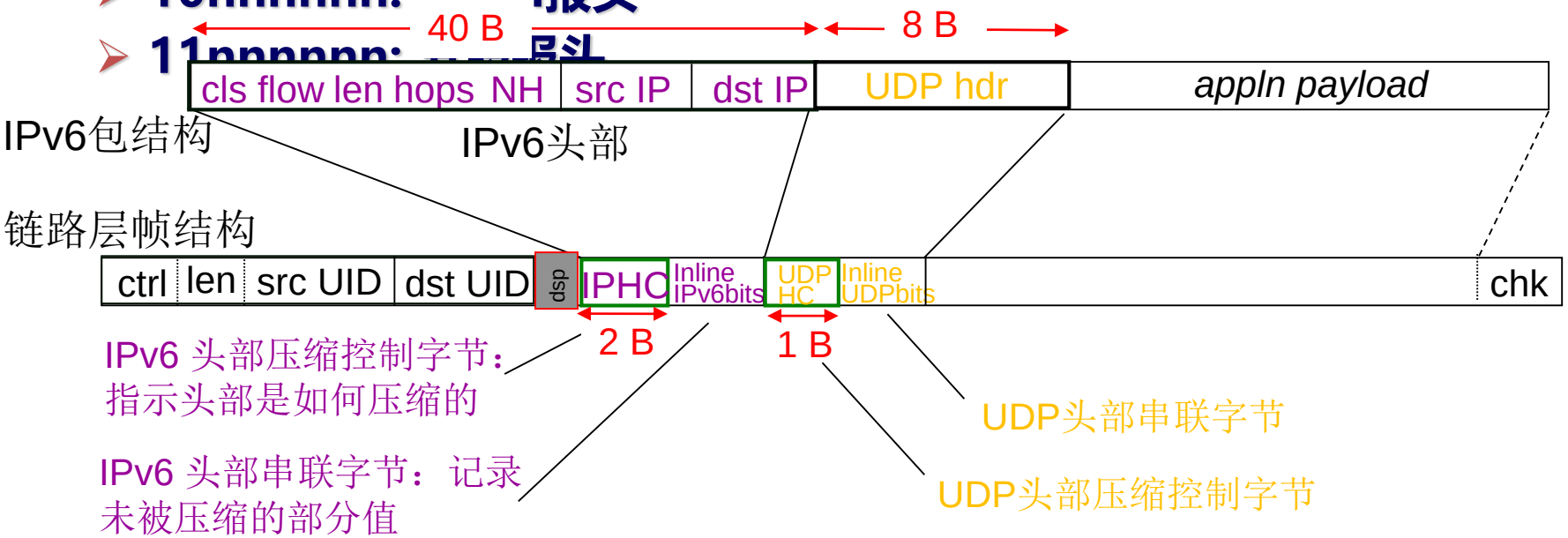
6LoWPAN：分段与重组

- Size: 每一个分段的大小(bytes)，所有分段中都包含
- Tag: 属于同一个IPv6报文的，Tag相同
- Offset: 表示在原来IPv6报文中的位置，用于重组
  - 在第一个分段中省略，其它分段需要包含该字段



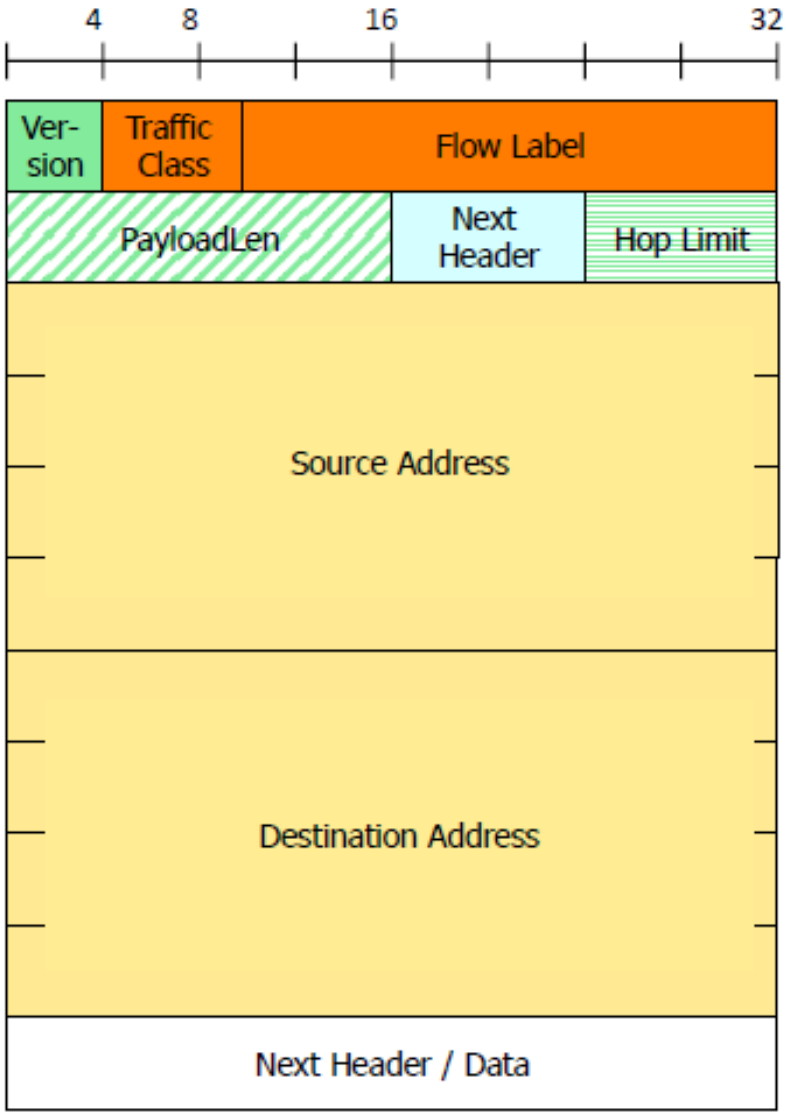
6LoWPAN: 帧结构

- IPv6数据包经压缩嵌入到链路层帧结构中
  - IPHC：2B的控制字节，用于说明IPv6头部如何压缩
  - UDPHC: 1B的控制字节，用于说明UDP头部如何压缩
- 在嵌入之前，6LoWPAN在IPv6报头前扩展一个字节(dispatch)，说明数据报类型
  - 00nnnnnn: 不是LoPAN帧
  - 01nnnnnn: 普通IPv6报头
  - 10nnnnnn: mesh报头
  - 11nnnnnn: [illegible]报头



6LoWPAN：头部压缩的基本原则

- IPv6头部(40B)压缩
  - IPv6地址压缩
    - 本地链路地址
      - 16B -> 3bits~19bits
    - 全局地址
      - 16B->7bits~ 23bits
  - IPv6头部其他字节的压缩
    - *Version: 4bits->0*
    - *Taffic class+Flow label:* 28bits -> 2bit
    - *PayloadLen:* 16bits->0
    - Hop Limit: 8bits -> 2bits
- UDP头部(8B)压缩
  - 端口号压缩
    - 2B->4bits



6LoWPAN：IPv6地址生成

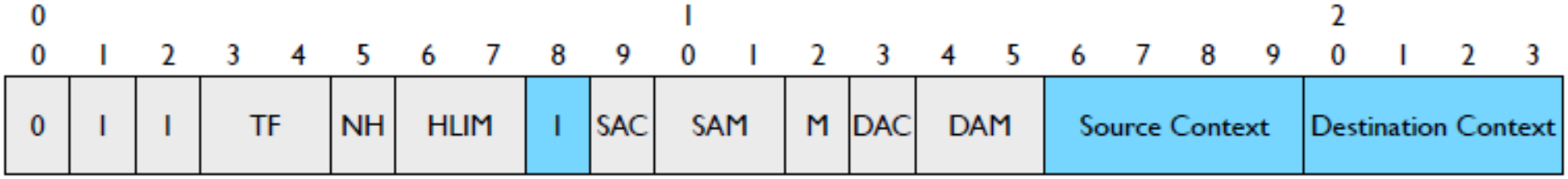
- 128-bit IPv6 地址
  - 64-bit 前缀+64-bit Interface ID （IID）
  - 主要地址类型
    - 本地链路单播地址 (FE80:: 开头)
    - 全球单播地址
    - 多播地址(FF:: 开头)
- 6LowPAN在链路层支持两种地址格式
  - 64位的IEEE EUI-64地址: 15.4全球唯一地址，由设备生产商分配
    - 24-bit OUI 组织唯一标识符 （IEEE分配）+ 40-bit 扩展标识符 （生产商分配）
  - 16位动态分配的短地址

6LoWPAN：IPv6地址生成

- 从链路层地址到IPv6地址
  - 在链路层地址前加上前缀构成
    - EUI-64 地址直接加上前缀
    - 16位短地址首先加上48位伪地址: 16位PAN ID+32位指定值，再加前缀
  - 前缀
    - 本地链路单播地址, 直接加上FE80::/64
    - 全局单播地址，由路由器分配(最多支持16种前缀)

6LoWPAN：IPv6地址压缩

- 6LoWPAN IPv6 地址特点
  - 前缀
    - 节点地址通常有相同的前缀
    - 路由器为前缀建立状态(context)，支持最多16个前缀, 编号0~15
  - IID：通常由MAC地址自动配置而成，可以从MAC地址推导出
- IPv6前缀的压缩
  - 压缩方式指示位：CID, SAC/DAC
  - 无状态模式(Stateless, SAC/DAC=0)
    - 前缀是本地链路前缀(FE80::/16), 无需额外标记
  - 基于上下文的模式(Context-based, SAC/DAC=1)
    - 前缀从事先存储的16个前缀中选择
    - CID=0, 使用编号为0的前缀，即ContextID=0
    - CID=1, 额外存储1byte信息, 源地址和目的地址各用4-bit用于记录其ContextID



6LoWPAN：IPv6 头部其它字节压缩

2B 控制字节

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
0	1	1	TF	NH	HLIM	CID	SAC	SAM		M	DAC	DAM	In-line IPv6 Header Bits		

- TF （Traffic Class &Flow Label)
  - 流量类型, 在IPv6中用于QoE控制, 在传感网中基本总是0
  - 0: 不压缩,在Inline字节中携带 (ECN+DSCP+Flow)
  - 1: ECN+Flow； 2: ECN+DSCP
  - 3: 全0，完全省略(最完美)

TF = 0

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
IPHC	ECN	DSCP								rsv				Flow Label																	

TF = 1

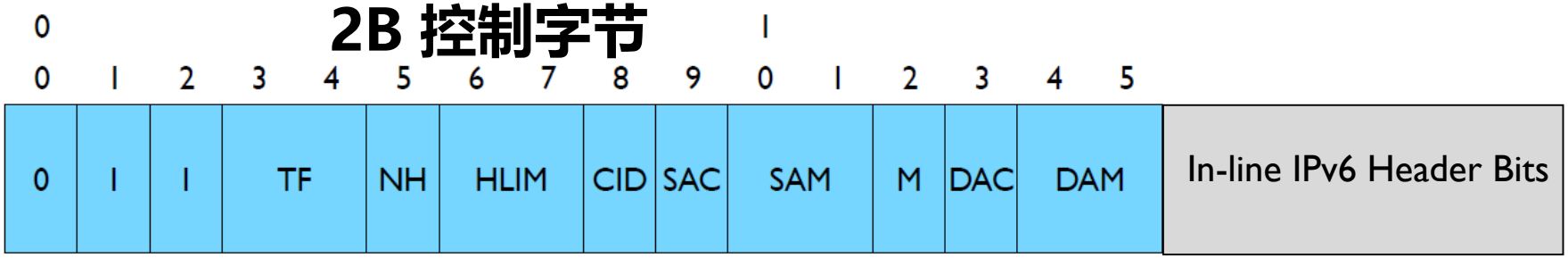
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
IPHC	ECN	rsv	Flow Label																				

TF = 2

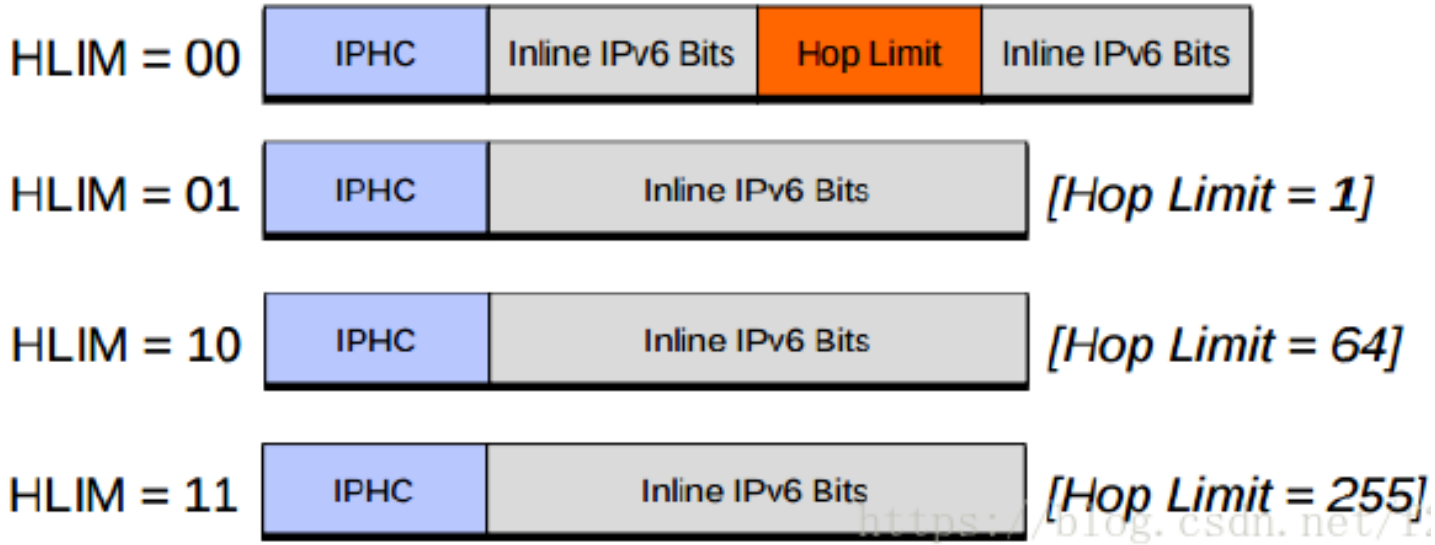
0	1	2	3	4	5	6	7
IPHC	ECN	DSCP					

TF = 3 IPHC

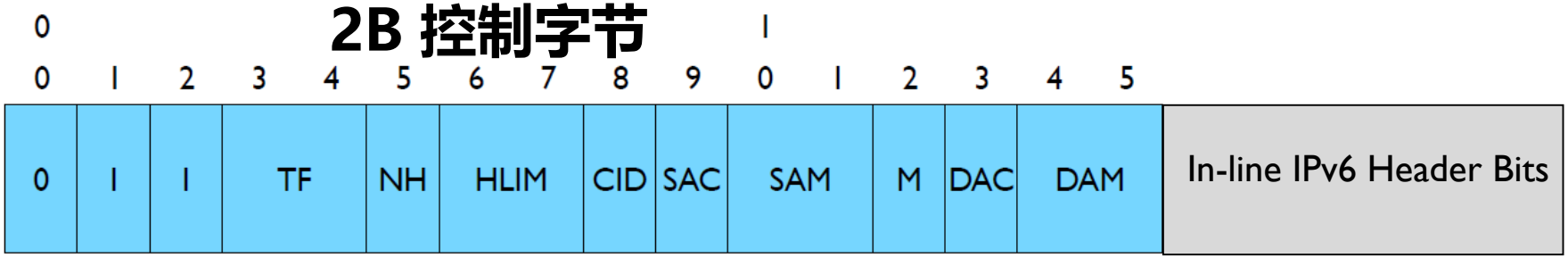
6LoWPAN：IPv6 头部其它字节压缩



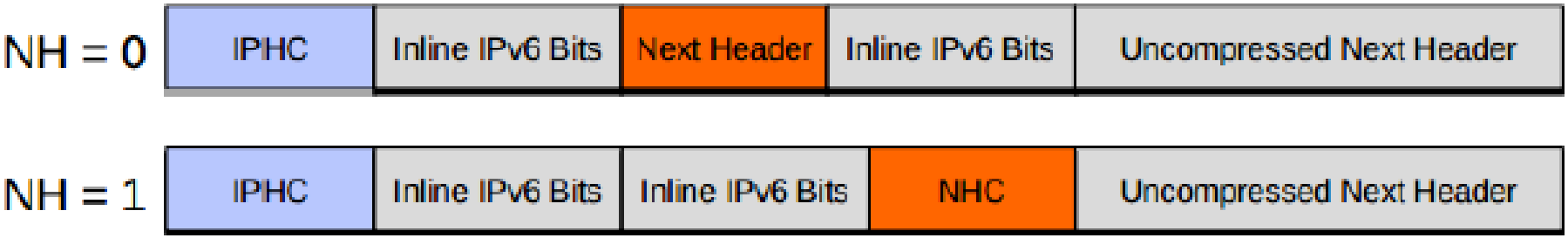
- HLIM (Hop Limit=Inline, 1, 64, 255)
  - 最大跳数
  - 0: 不压缩, Inline; 1:1; 2: 64; 3: 255



6LoWPAN：IPv6 头部其它字节压缩



- NH（Next Header Compression）
  - 表示下一个头部有没有压缩,例如UDP header
  - 0：不压缩, Inline；1：压缩



6LoWPAN：UDP头部压缩

- UDP头部 （8 bytes）

	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
0	Source Port	Destination Port
4	Length	Checksum

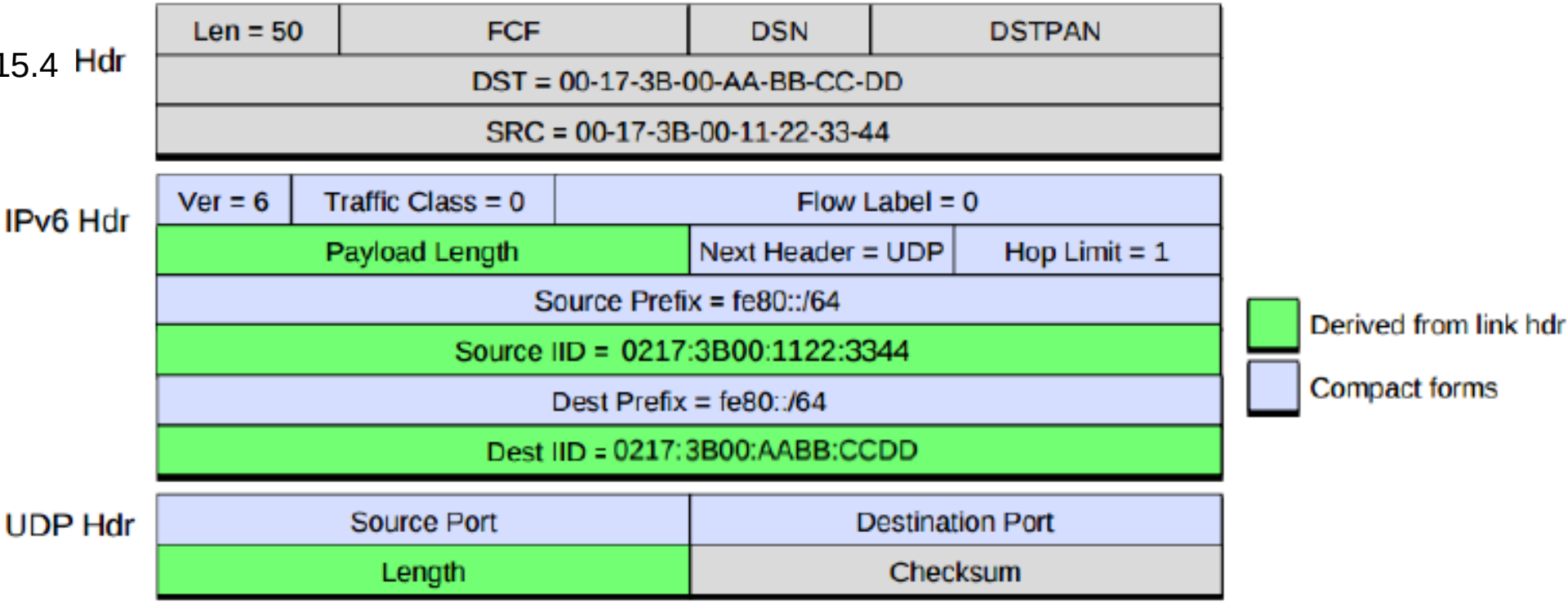
- 找到头部字段的共同值并压缩
  - 规定端口值(Port)范围：61616~61632 (16个值, 4-bit)
  - 长度(Length)可以从IPv6长度中推出
  - 如果在其它的协议(e.g., IPSec)已有checksum，则可省略，如果没有，则需要保留
- UDP 头部压缩控制字节

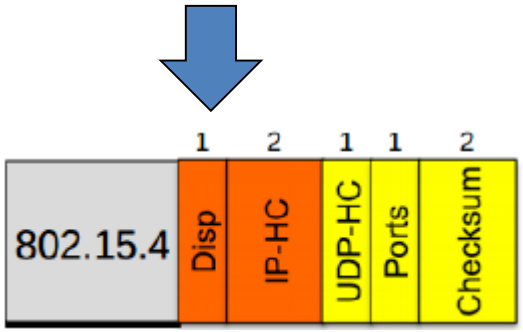
0	1	2	3	4	5	6	7
1	1	1	1	0	C	P	

  - C(CheckSum): 0:Inline, 1:省略
  - P(Port): 00:Inline, 01:省略目的端口前8位，10: 省略源端口前8位，11: 省略源端口和目的端口前12位

6LoWPAN：头部压缩示例

❑ 本地链路(Link-local)单播报头压缩示例

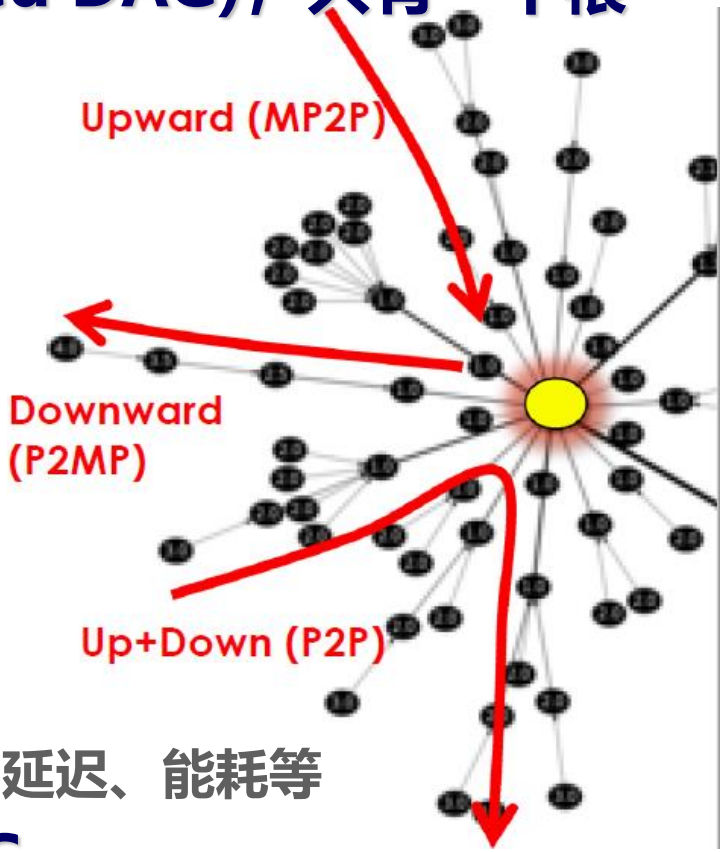




48-byte UDP/IPv6 Hdr➔ 7 bytes

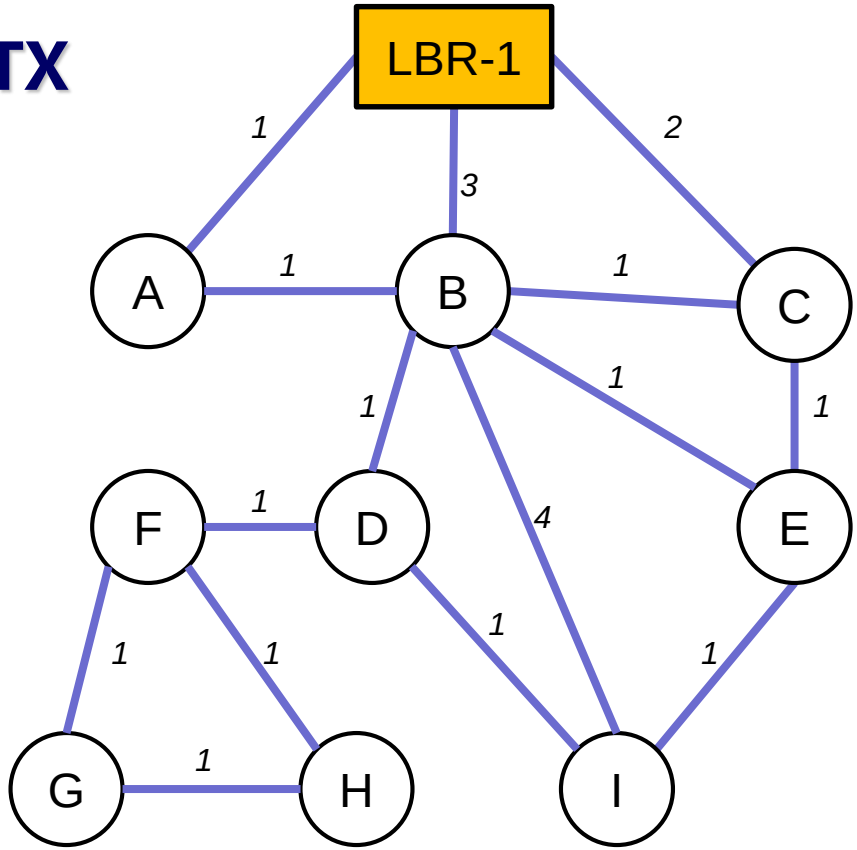
网络层协议: RPL

- RPL(Ripple) , Routing Protocol for Low Power and Lossy Networks
- 组网模式:
  - DODAG(Destination-Oriented DAG)，只有一个根节点的有向无环图
- 通信模式： MP2P, P2MP,P2P
  - MP2P: 数据收集
  - P2MP: 数据分发
  - P2P: 点对点, 由前两种模式结合实现
- 组网过程
  - 基于距离向量路由建立DAG
    - 距离由目标函数而定, 可能为ETX、延迟、能耗等
  - 节点交换DIS DIO 包来加入DAG



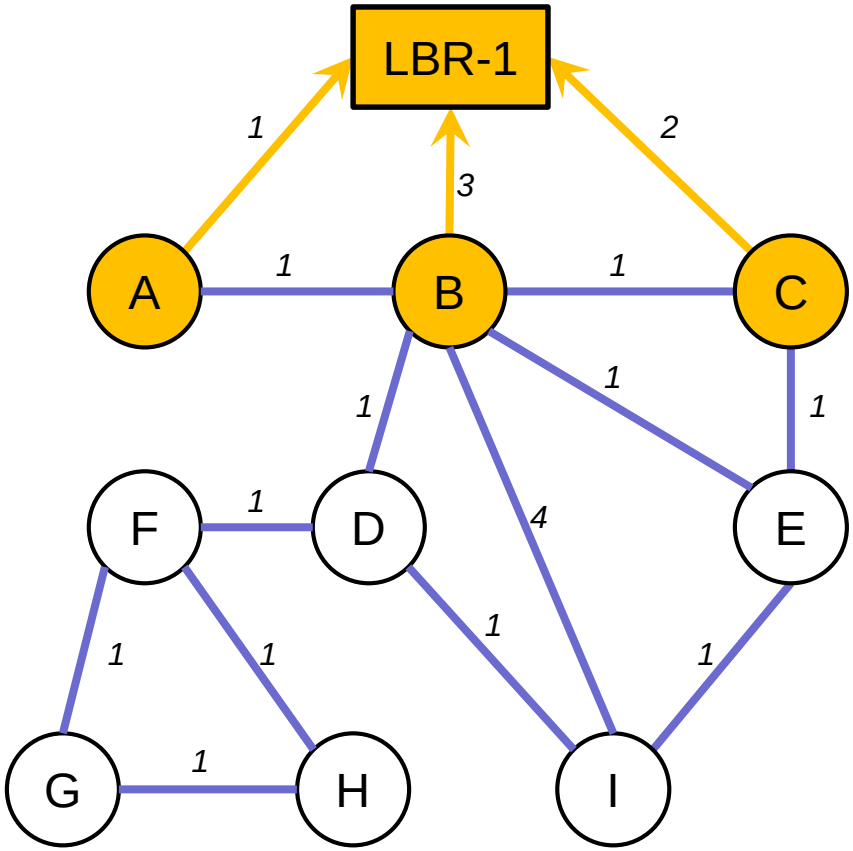
网络层协议: RPL

- DAG 建立
  - 距离：ETX, 标在每条边上
  - 目标：最小化节点到根的ETX
- 建立过程
  - LBR-1广播DIO信息



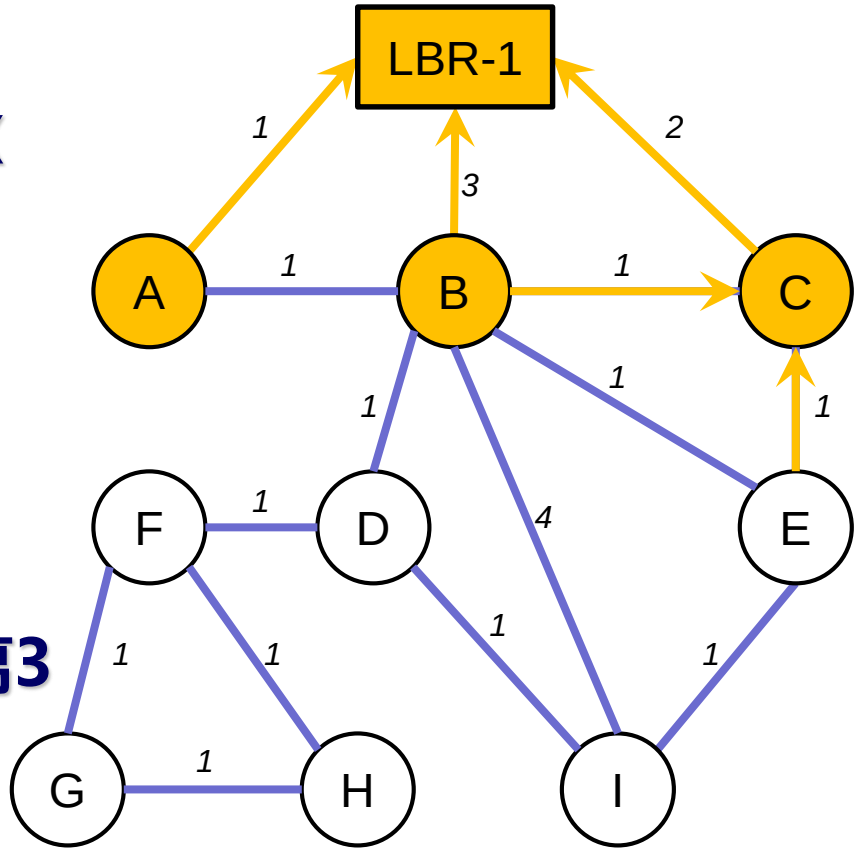
网络层协议: RPL

- DAG 建立
  - 距离：ETX, 标在每条边上
  - 目标：最小化节点到根的ETX
- 建立过程
  - LBR-1广播DIO信息
  - ABC收到，设置LBR-1为父节点，加入DAG



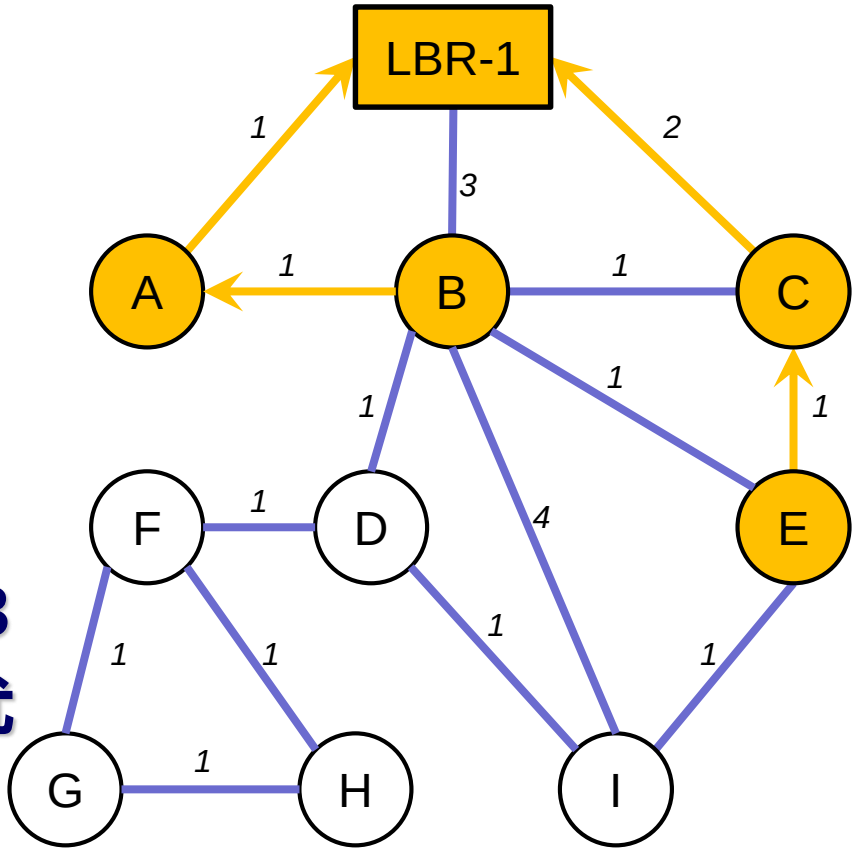
网络层协议: RPL

- DAG 建立
  - 距离：ETX, 标在每条边上
  - 目标：最小化节点到根的ETX
- 建立过程
  - LBR-1广播DIO信息
  - ABC收到，设置LBR-1为父节点，加入DAG
  - C广播DIO信息，B添加C为可选父节点; E加入DAG, 距离3



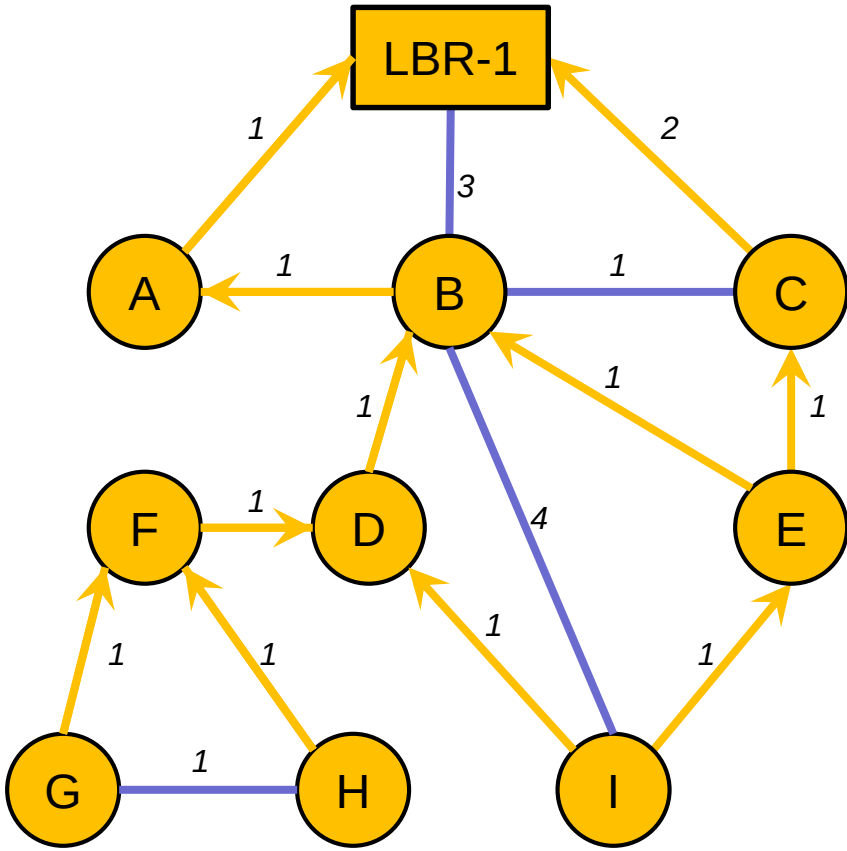
网络层协议: RPL

- DAG 建立
  - 距离：ETX, 标在每条边上
  - 目标：最小化节点到根的ETX
- 建立过程
  - LBR-1广播DIO信息
  - ABC收到，设置LBR-1为父节点，加入DAG
  - C广播DIO信息，B添加C为可选父节点; E加入DAG, 距离3
  - A广播DIO信息, B发现A是更优的父节点, 因而改路由到A



网络层协议: RPL

- DAG 建立
  - 距离：ETX, 标在每条边上
  - 目标：最小化节点到根的ETX
- 建立过程
  - LBR-1广播DIO信息
  - ABC收到，设置LBR-1为父节点，加入DAG
  - C广播DIO信息，B添加C为可选父节点; E加入DAG, 距离3
  - A广播DIO信息, B发现A是更优的父节点, 因而改路由到A
  - 继续该过程，直到所有节点都加入DAG



目录

- 概述
- 基于IEEE802.15.4的传感网协议架构
  - IEEE 802.15.4技术物理层
  - 低功耗MAC
  - 链路质量估计
  - 网络层协议
  - 应用案例
- Bluetooth
- 研究进展

编程实例-基于RPL的数据收集

- 在本例中，我们基于低功耗和有损网络（LLNs）路由协议（RPL）的数据收集。
- 我们将具体介绍如何编写传输层的代码（即UDP）完成数据的接收，网络层基于RPL实现，来自网络中的节点将基于RPL协议通过单跳或者多跳的数据传输到达汇聚节点。汇聚节点即为Server。

网络层名称	使用的协议名称
传输层	UDP
网络层	IPv6/RPL
访问控制层	ContikiMAC
物理层	IEEE 802.15.4

编程实例-基于RPL的数据收集

- **1. 首先编写数据节点端的代码。下面是建立UDP连接的主要代码。在udp-client.c中的PROCESS_THREAD(udp_client_process, ev, data)。**

```
1.   /* new connection with remote host */
2.    client_conn = udp_new(NULL,
     UIP_HTONS(UDP_SERVER_PORT), NULL);
3.    udp_bind(client_conn,
     UIP_HTONS(UDP_CLIENT_PORT));

4.    PRINTF("Created a connection with the server ");
5.    PRINT6ADDR(&client_conn->ripaddr);
```

编程实例-基于RPL的数据收集

- **2.接着是编写UDP-client发送数据的代码。在udp-client.c中。**

```
static void send_packet(void *ptr)
{
  char buf[MAX_PAYLOAD_LEN];
  seq_id++;
  PRINTF("DATA send to %d 'Hello %d'\n",
         server_ipaddr.u8[sizeof(server_ipaddr.u8) - 1], seq_id);
  sprintf(buf, "Hello %d from the client", seq_id);
  uip_udp_packet_sendto(client_conn, buf, strlen(buf),
                        &server_ipaddr,
UIP_HTONS(UDP_SERVER_PORT));
}
```

编程实例-基于RPL的数据收集

- 3.编写定时器周期性发送数据。在udp-client.c中的PROCESS_THREAD(udp_client_process, ev, data) 。

```
etimer_set(&periodic, SEND_INTERVAL);
while(1) {
    PROCESS_YIELD();
    if(etimer_expired(&periodic)) {
        etimer_reset(&periodic);
        ctimer_set(&backoff_timer, SEND_TIME, send_packet, NULL);}
}
```

编程实例-基于RPL的数据收集

- 4.服务端代码。初始化RPL的DAG网络。在udp-server.c中的PROCESS_THREAD(udp_server_process, ev, data) 。

```
uip_ds6_addr_add(&ipaddr, 0, ADDR_MANUAL);
root_if = uip_ds6_addr_lookup(&ipaddr);
if(root_if != NULL) {
    rpl_dag_t *dag;
    dag = rpl_set_root(RPL_DEFAULT_INSTANCE,(uip_ip6addr_t
*)&ipaddr);
    uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
    rpl_set_prefix(dag, &ipaddr, 64);
    PRINTF("created a new RPL dag\n");
} else {
    PRINTF("failed to create a new RPL DAG\n");
}
```

编程实例-基于RPL的数据收集

- 5.服务端代码。建立UDP连接。在udp-server.c中的PROCESS_THREAD(udp_server_process, ev, data)。

```
1. server_conn = udp_new(NULL, UIP_HTONS(UDP_CLIENT_PORT), NULL);
2. udp_bind(server_conn, UIP_HTONS(UDP_SERVER_PORT));
3. PRINTF("Created a server connection with remote address ");
4. PRINT6ADDR(&server_conn->ripaddr);
```

编程实例-基于RPL的数据收集

- 6.服务端代码。接收数据并且打印。在udp-server.c中

```
static void tcpip_handler(void)
{
  PRINTF("DATA recv '%s' from ", appdata);
  PRINTF("%d",
  UIP_IP_BUF->srcipaddr.u8[sizeof(UIP_IP_BUF->srcipaddr.u8) - 1]);
  PRINTF("\n");
}
```

编程实例-基于RPL的数据收集

❑ 6.编写Makefile和project_conf.h

```
all: udp-client udp-server
CONTIKI=../../..

CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\"

ifdef WITH_COMPOWER
APPS+=powertrace
CFLAGS+= -DCONTIKIMAC_CONF_COMPOWER=1 -
DWITH_COMPOWER=1 -DQUEUEBUF_CONF_NUM=4
endif

ifdef SERVER_REPLY
CFLAGS+=-DSERVER_REPLY=$(SERVER_REPLY)
endif
ifdef PERIOD
CFLAGS+=-DPERIOD=$(PERIOD)
endif

ifeq ($(MAKE_WITH_NON_STORING),1)
CFLAGS += -DWITH_NON_STORING=1
endif

CONTIKI_WITH_IPV6 = 1
include $(CONTIKI)/Makefile.include
```

```
#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_

/* configure number of neighbors and routes */
#define NBR_TABLE_CONF_MAX_NEIGHBORS     10
#define UIP_CONF_MAX_ROUTES   30
#else
/* configure number of neighbors and routes */
#define NBR_TABLE_CONF_MAX_NEIGHBORS     10
#define UIP_CONF_MAX_ROUTES   10
#endif /* TEST_MORE_ROUTES */
#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC     nullrdc_driver
#undef NULLRDC_CONF_802154_AUTOACK
#define NULLRDC_CONF_802154_AUTOACK       1
/* Define as minutes */
#define RPL_CONF_DEFAULT_LIFETIME_UNIT   60

/* 10 minutes lifetime of routes */
#define RPL_CONF_DEFAULT_LIFETIME        10

#define
RPL_CONF_DEFAULT_ROUTE_INFINITE_LIFETIME 1

#endif
```

编程实例-基于RPL的数据收集

- 7. 执行下面的命令查看节点id。

```
$ motelist
Reference  Device          Description
---------- ---------------- ----------------------------------------------
XBTOM4UK   /dev/ttyUSB0     XBOW Crossbow Telos Rev.B
XBTOM8JM   /dev/ttyUSB1     XBOW Crossbow Telos Rev.B
```

- 8.给server和client分别烧写的时候，在MakeFile添加节点MACID，会自动生成对应的ipv6地址，

```
#server ID
CFLAGS += -DMACID = 0
#client ID
CFLAGS += -DMACID = 1
```

- 9.执行下面的命令分别给client和server烧写程序

```
$ make TARGET=sky MOTES=/dev/ttyUSB0 udp-client.upload
$  make TARGET=sky MOTES=/dev/ttyUSB1 udp-server.upload
```

目录

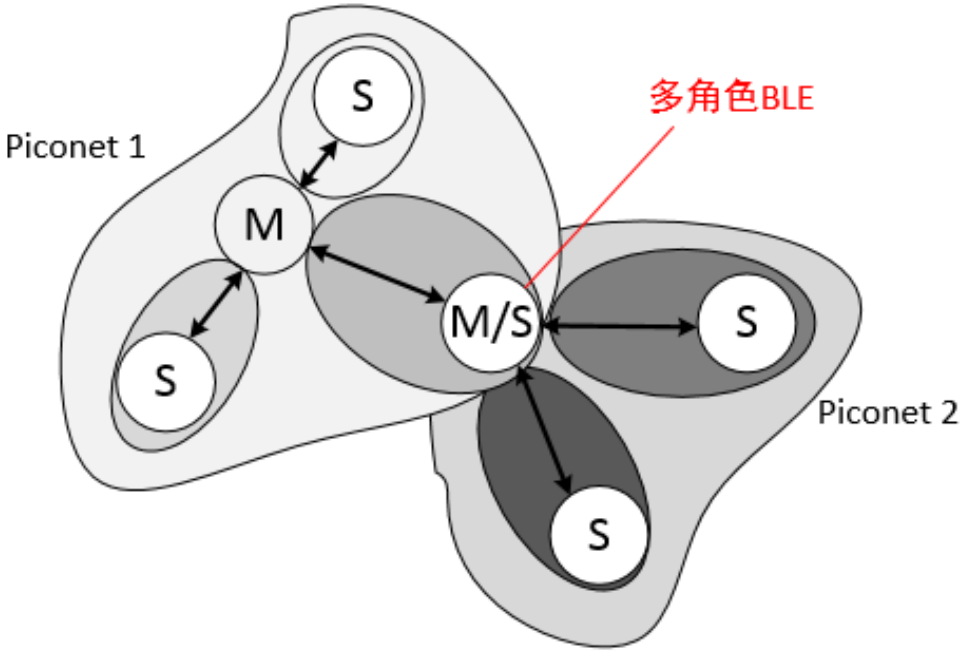
- 概述
- 基于IEEE802.15.4的传感网协议架构
- Bluetooth
  - 概述
  - 物理层协议
  - 链路层协议
  - L2CAP协议
  - ATT属性检索协议和GATT通用属性框架
  - SMP安全管理协议
  - BLE mesh
  - 基于BLE的广告推送iBeacon应用
- 研究进展

Bluetooth

- 概述
- 物理层协议
- 链路层协议
- L2CAP协议
- ATT属性检索协议和GATT通用属性框架
- SMP安全管理协议
- BLE mesh
- 基于BLE的广告推送iBeacon应用

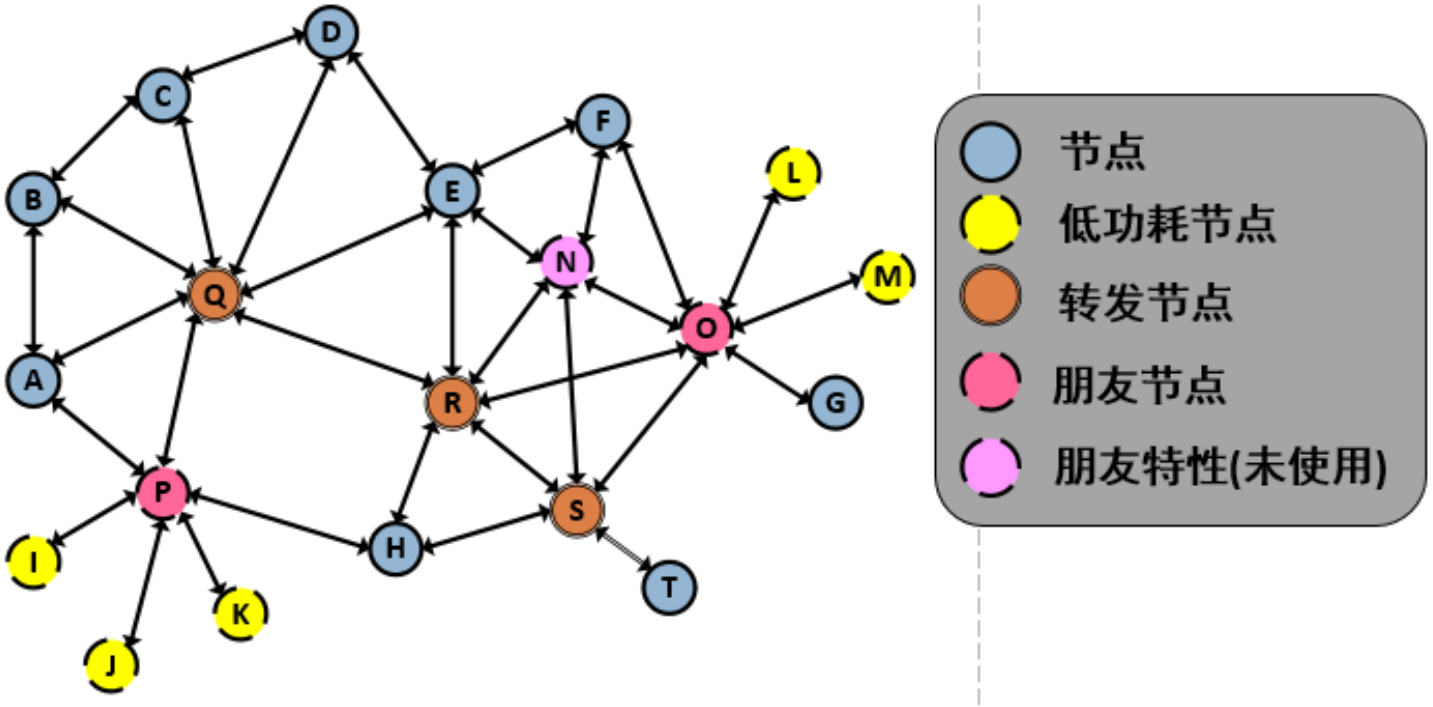
场景-BLE的星型组网（单跳）

- 一个星型网络（piconet）
  - 一个master，多个slave（传统Bluetooth限制为7个,BLE没有限制）
- BLE设备多重角色
  - 可以是一个piconet的slave，
  - 又是另一个piconet的master。但一个时间点只能是一个角色
- BLE组网：一个piconet每个slave使用不同的信道（不同的跳频序列）
  - 下图中，相同颜色为同一个信道



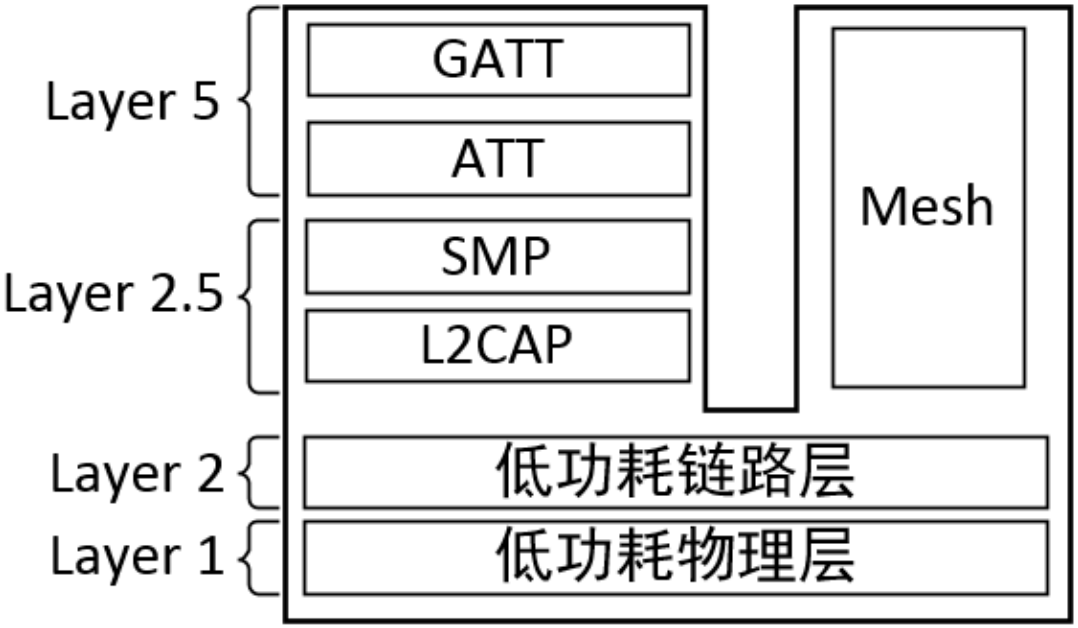
场景-BLE的mesh组网（多跳）

- 组网：网络中节点不同的角色（五种节点）
  - 节点: 网络边缘的节点，没有relay功能。
  - 转发节点: 具有转发功能，收到数据立即转发
  - 朋友节点: 特殊的转发节点，还可以暂存数据
  - 低功耗节点：可从朋友节点查询数据，无需一直监听数据包。
  - 朋友特性（未使用）: 可以作为朋友节点，但未启用对应的功能



BLE协议栈

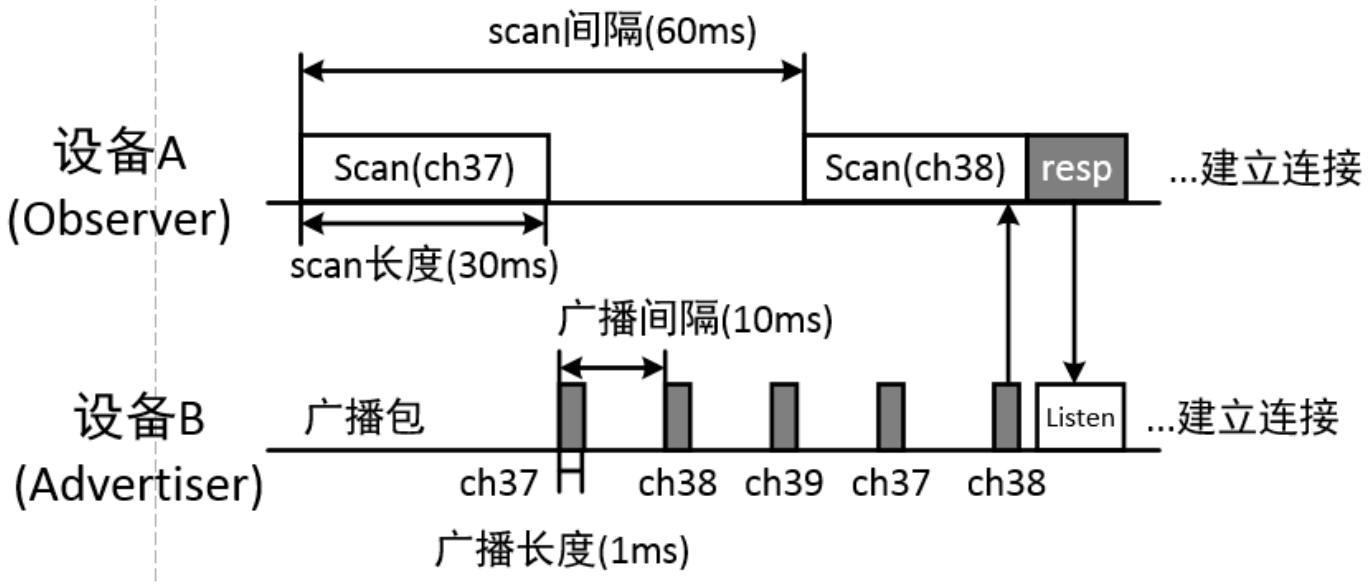
- 低功耗物理层
  - 40个跳频信道：37个数据传输信道，3个广播信道
- 低功耗链路层
  - 周期性醒睡
- L2CAP
  - 数据包融合/分割
  - 控制数据包重传
- GATT，ATT
  - 服务交换的数据结构
- SMP
  - 密钥生成与分发



蓝牙低功耗协议栈

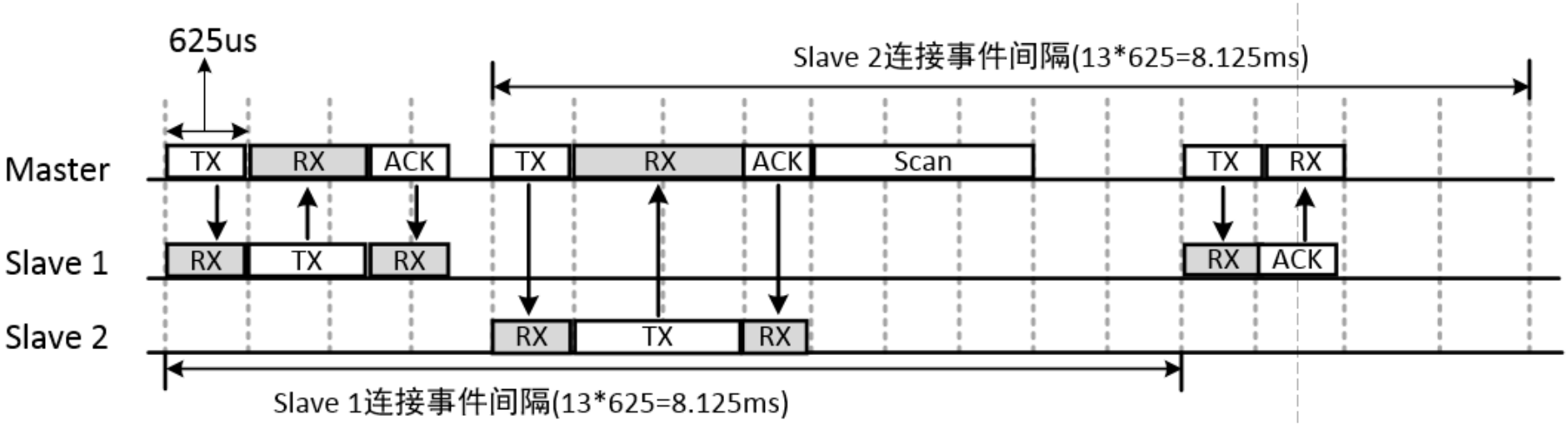
物理层协议

- 信道分配
  - 从2.4GHz到2.480GHz，
  - 每个信道2MHz，总共40个信道
  - 37,38,39为广播信道，其他为数据传输信道
- 邻居发现
  - 扫描长度：3倍以上的广播间隔，37,38,39轮流扫描
  - 广播间隔：10ms以上，37,38,39轮流广播



物理层协议

- 数据传输信道-基础跳频
  - 下一个跳频信道：$F_{n+1} = gMap[(F_n+hop)\%37]$
  - gMap:可用信道数组，如[1,13,24,27]
  - hop: 跳频增量，为[5,16]之间的随机数，
- 数据传输信道-自适应跳频
  - 根据链路质量更新gMap
- Master维护多个Slave连接
  - 每个slave有单独的连接事件间隔
  - Master和slave跳频以625us的slot为单位同步
  - 对于master,一个连接事件间隔内会处理完所有的slave通信，并且执行扫描
  - 对于slave，piconet中slave越多，连接事件间隔越长

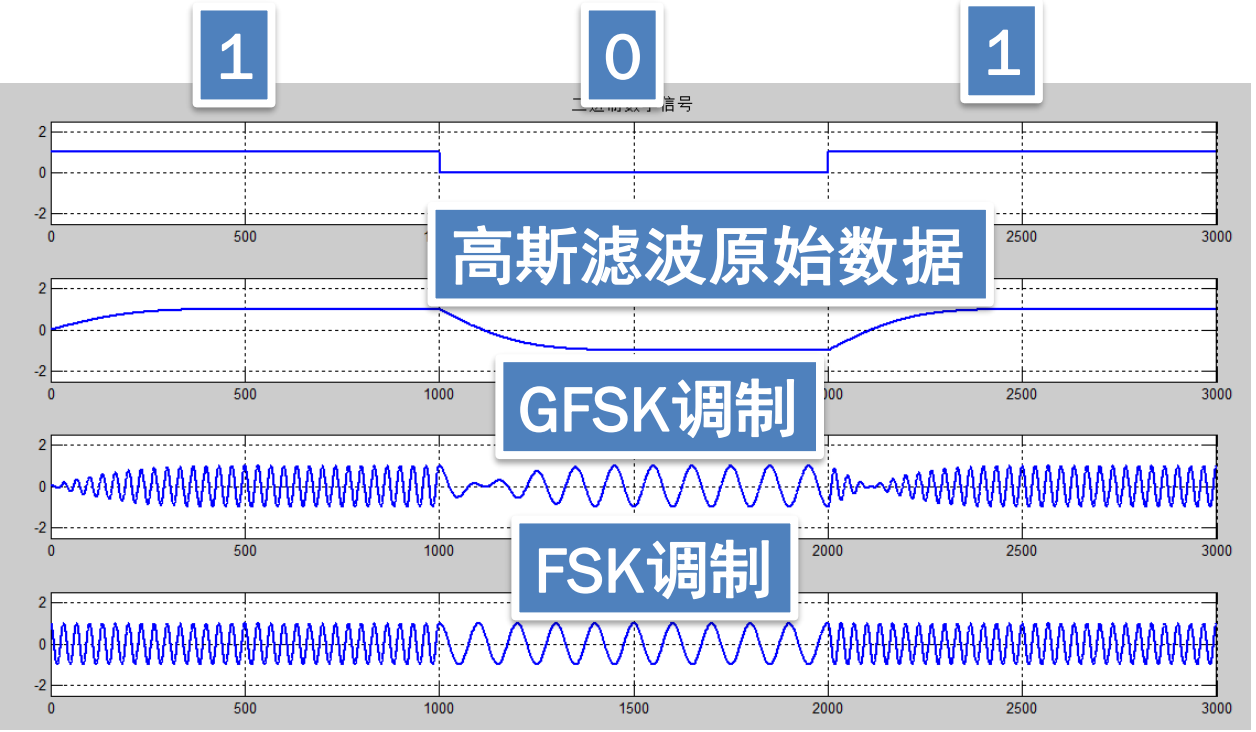


物理层协议

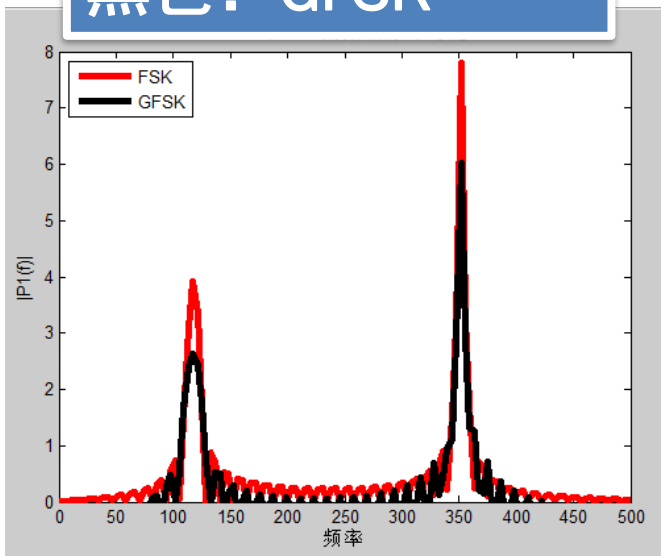
❑ BLE的调制方式: GFSK：高斯频移键控

bit数据 → 高斯滤波 → FSK → I/Q调制

- 给定传输数据101，FSK和GFSK调制波形图如下所示
- 降低不同频率调制之间的干扰。



频谱图对比
红色：FSK
黑色：GFSK



物理层协议

- 物理帧结构。
  - preamble: 0,1交替， 首位bit和Access Address的首位bit相同
    - *10101010 ，8位(1Mbps模式)*
  - Access Address (AA):
    - *广播包：固定的值0x8E89BED6*
    - *数据包：在建立连接过程中确定，每一对链路唯一的AA*
  - PDU
    - *广播包：最多37字节，最少6字节*
    - *数据包：最多257字节，最少2字节*

Preamble (1字节)	Access Address (4字节)	PDU (2 to 251 字节)	CRC (3字节)

链路层协议

- **广播包帧结构**
  - **Header**
    - *PDU type*
    - *TxAdd: 指示发送方的地址类型*
      - 0->使用厂商分配的静态地址,1->自己生成的随机地址
    - *RxAdd:指示接收方的地址类型*
      - 0->使用厂商分配的静态地址,1->自己生成的随机地址
    - *RFU*
      - 未来使用的
  - **PDU type（7种）**
    - *ADV_IND (可连接) <-> CONNECT_REQ (连接请求)*
    - *ADV_DIRECT_IND (定向可连接), <->CONNECT_REQ (连接请求)*
    - *ADV_SCAN_IND (可扫描) <- > SCAN_REQ (扫描请求), <->SCAN_RSP (扫描反馈)*
    - *ADV_NONCONN_IND (不可连接) –x*
- **广播包间隔**
  - **Low duty cycle: 3.75 ~ 10 ms**
  - **High duty cycle: 0 ~ 3.75 ms**

广播信道数据包

Header (2字节)	Payload (0-37 字节)

Header

PDU type (4 bits)	RFU (2 bits)	TxAdd (1 bit)	RxAdd (1 bit)	Length (6 bits)	RFU (2 bits)

链路层协议

- **连接请求广播包结构(CONNECT_REQ)**
  - **initA: 发起者的地址（6字节）**
  - **AdvA: 接收方的地址（6字节）**
  - **LLData**
    - *AA: 建立连接后确定链路的唯一标记(广播包是固定的)*
    - *WinSize,Winoffset: BLE处理多个连接的缓冲时间*
    - *Interval: 连接事件的间隔*
    - *Latency: slave隔多少个连接事件再反馈master*
    - *Timeout: 等待反馈的延迟*
    - *ChM: 信道可用情况标记(40 bits, 每个bit代表一个信道)，与gMap数组对应*
    - *Hop:随机跳频的增量因子。*

连接请求数据包 (payload)

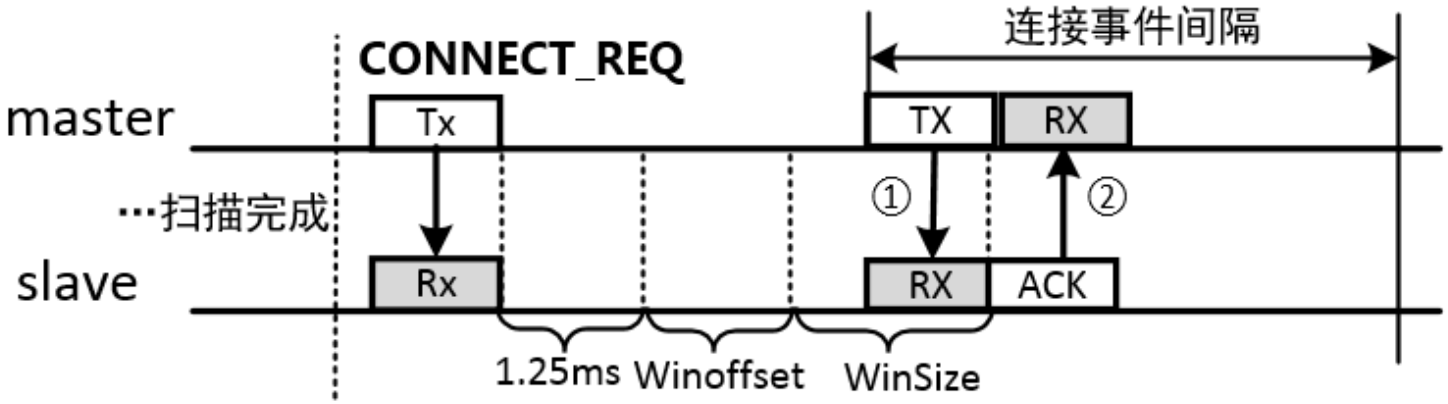
InitA (6 字节)	AdvA (6 字节)	LLData (22 字节)

LLData

AA (4 字节)	CRCinit (3 字节)	Winsize (2 字节)	WinOffset (2 字节)	Interval (2 字节)	Timeout (2 字节)	ChM (5 字节)	Hop (5 字节)

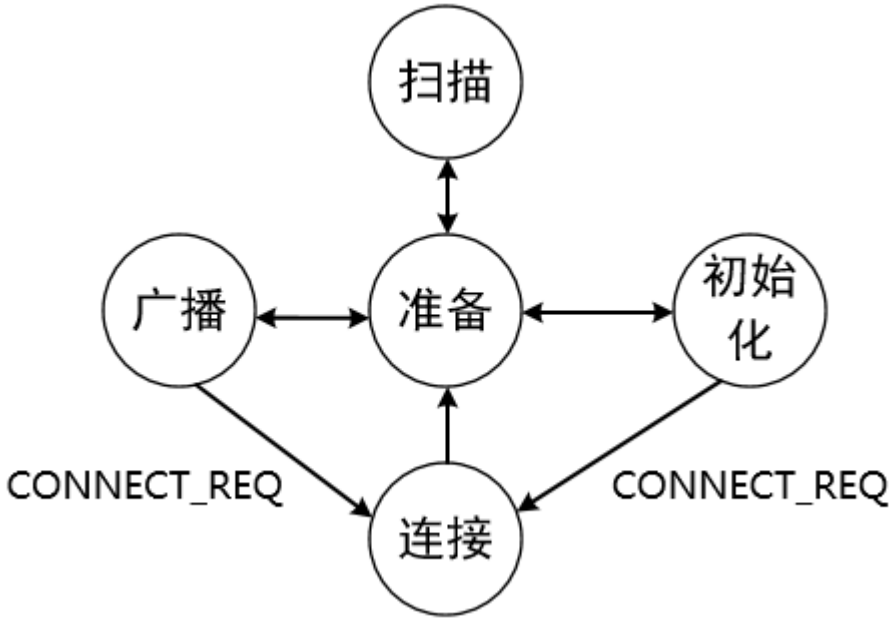
链路层协议

- 连接建立过程
  - Master发起连接请求
  - 等待1.25ms+winoffset时间间隔（BLE处理多个连接的schedule时间）
  - Master发送数据包，slave成功反馈，则连接成功，且连接事件间隔确认
  - 连接失败
    - *①slave没有在winsize时间内收到master数据包，失败*
    - *②master没有在timeout时间内收到salve反馈，失败*



链路层协议

- **BLE链路层状态机**
  - 扫描状态:，对广播信道37,38，39轮流扫描
  - 准备状态：可以向其他状态切换的中间状态
  - 广播状态：纯广播或者可以被连接
  - 初始化状态：监听指定的device信道，等待发送连接反馈
  - 连接状态：从广播状态或者初始化状态通过发送连接广播包CONNECT_REQ切换。

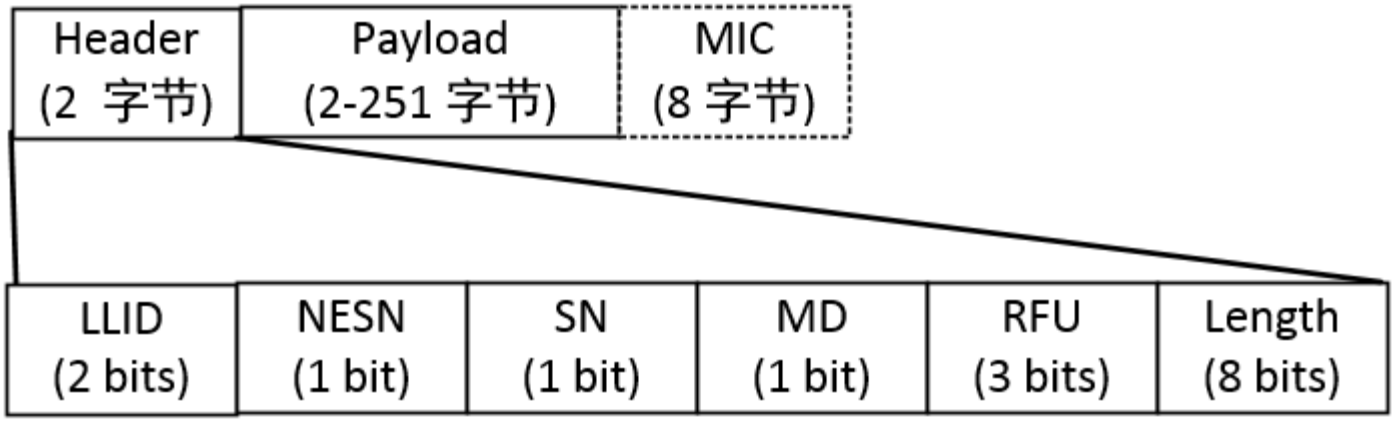


链路层协议

❑ 数据包帧结构

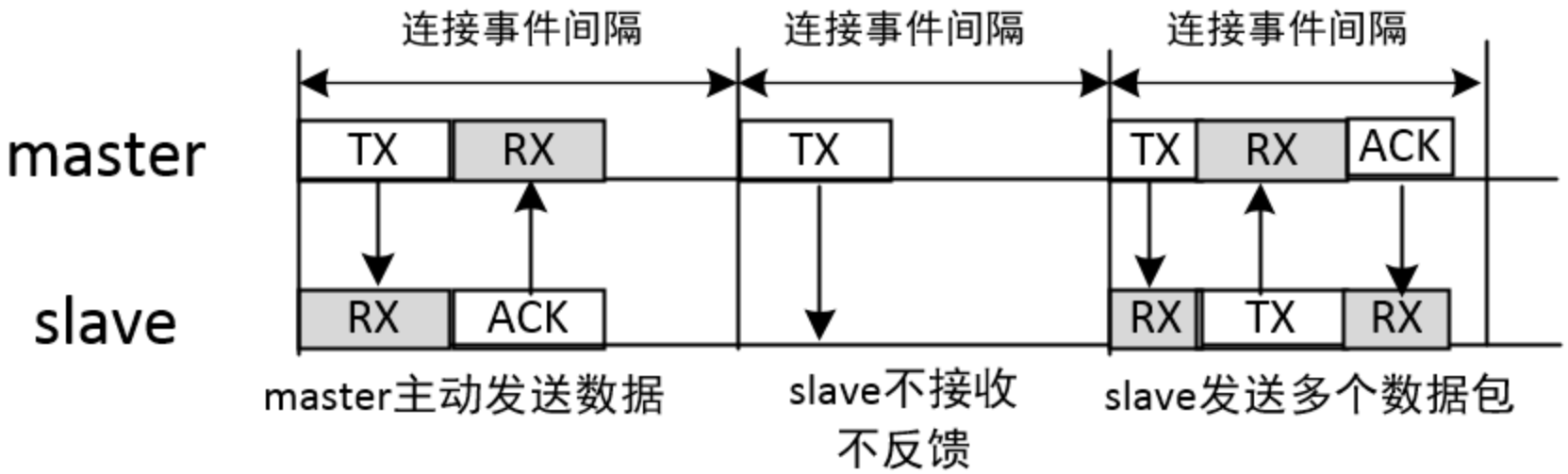
- **Header**
  - *LLID: 表示数据包或者控制包*
  - *NESN： 下一个期望的包序号（停等协议,序号为0/1）*
  - *SN：包序号（停等协议,序号为0/1）*
  - *MD： more data,是否还有继续的数据包*
- **MIC：完整性校验**

数据包格式



链路层协议

- 数据包传输
  - 7.5ms 到 4s之间, 1.25ms的倍数
  - Master发起每次通信事件，slave反馈
    - *即使没有实际数据发送，也会周期性发送/反馈空数据包*
    - *Slave可以设定**latency**，表示经过多少连接事件再反馈（节约能量）*
    - *Slave在一个连接事件间隔中，可以连续发送多个数据包*
  - 通信完成，立刻进入休眠



L2CAP

❑ 数据包的分割和重组

L2CAP数据帧

Length (2 字节)	Channel ID (2 字节)	L2CAP PDU length (2 字节)	Payload (2-251 字节)

➢ L2CAP帧格式

- *Length: 数据帧长度*
- *Channel ID: 用于唯一表示当前通信链路的标识*
- *L2CAP SDU length: 传输数据帧总长度（子数据帧重组后的长度）。每个子数据帧最大251字节。*

➢ 示例：

- *发送90字节数据，CID为0x4b，每个子数据帧长度最大为44字节（包括包头）。（master, slave协商达成）*
- *注：子数据帧没有偏移编号，因为BLE使用停等协议。*

	长度(字节)	Channel ID	数据帧总长度	数据
（1）	40	0x004b	90	Data[0-37]
（2）	40	0x004b	Data[38-77]	
（3）	12	0x004b	Data[77-89]	

L2CAP

- **差错控制和重传**
  - **L2CAP层的差错控制主要通过重传来完成**
  - **信息帧：用来发送数据。control域主要定义了两种重传模式**
    - *ACK超时，sender重传*
    - *sender等待receiver通知再重传*

L2CAP信息帧

Length (2 字节)	Channel ID (2 字节)	Control (2 或4 字节)	Payload (2-251 字节)

  - **监督帧：反馈ACK/NACK信息. Control域主要定义了反馈的类型**
    - *成功接收数据包*
    - *没有成功接收*
    - *拒绝接收数据包*
    - *拒绝接收部分数据包*

L2CAP监督帧

Length (2 字节)	Channel ID (2 字节)	Control (2 或4 字节)	FCS (0 或2 字节)

ATT属性检索协议和GATT通用属性框架

- **Profile:定义了一系列slave提供的服务的组合。**
  - 服务：提供一系列具体的功能集合。
  - 特征：即为功能。
- **ATT(Attribute Protocol): 提供对服务的检索功能**
- **GATT(Generic Attribute Profile)：定义了服务存储的数据结构**
  - *Handler: 句柄，区分不同的服务或特征.*
  - *UUID: 服务/特征的标志，可用系统默认，也可自定义*
  - *Value:对应的值*
  - *Permission: 可读，可写 等*

Handler	UUID	Value	Permission	Profile
0x0100	0x2800	UUID=0x1800	Read	Generic Access [service]
0x0101	0x1800	UUID=0x2a00, handler=0x0102	Read,write	服务 [characteristic]
0x0102	0x2a00	BLE device	Read,write	特征
0x0111	0x2a01	0x0100	Read	[characteristic] 手机
0x0200	0x2800	UUID=0x1809	Read	温度计 [service]

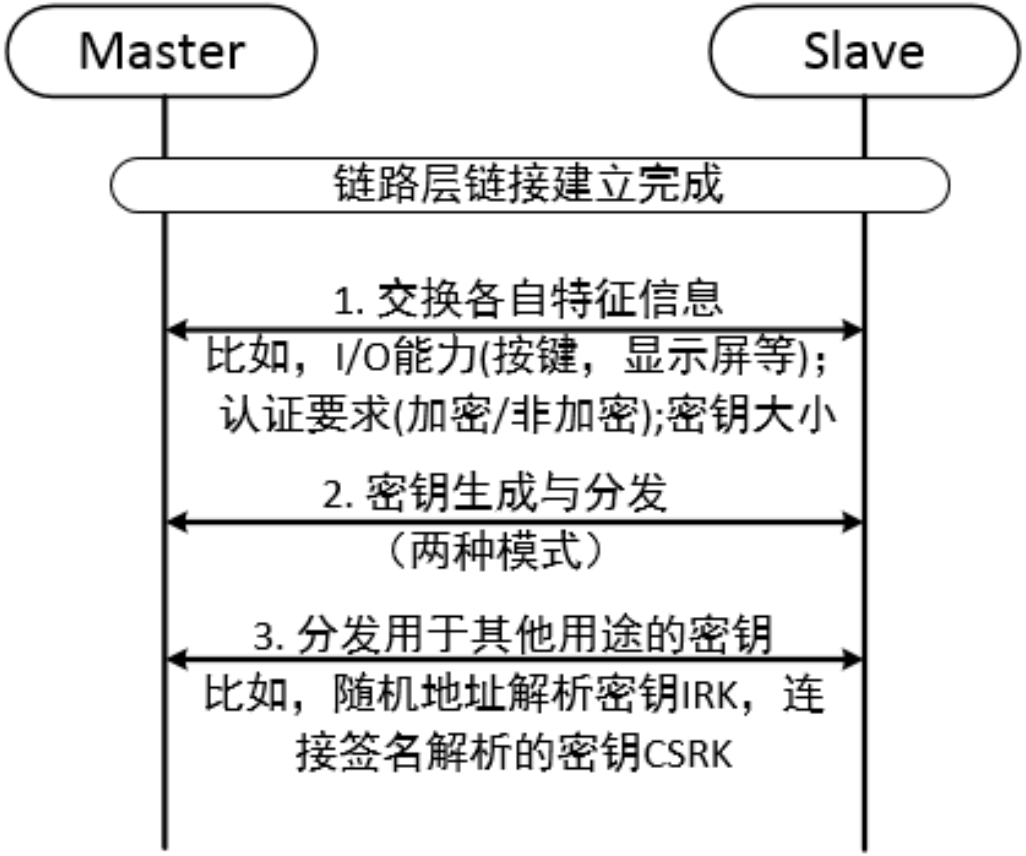
ATT属性检索协议和GATT通用属性框架

- 示例：一个提供了设备名称服务和温度查询服务的手机
  - 每个服务以UUID=0x2800表示开始和结束，value即为其包含的具体服务
  - 服务中的value包含提供的特征UUID和handler.(比如设备名称BLE device, 外观编号0x0100表示手机)

Handler	UUID	Value	Permission	备注
0x0100	0x2800	UUID=0x1800	Read	Generic Access [service]
0x0101	0x1800	UUID=0x2a00, handler=0x0102	Read,write	设备名[characteristic]
0x0102	0x2a00	BLE device	Read,write	
0x0111	0x2a01	0x0100	Read	外观[characteristic] 手机
0x0200	0x2800	UUID=0x1809	Read	温度计 [service]
0x0201	0x1809	UUID=0x2a1c, Handler=0x0202	Read	
0x0202	0x2a1c	20	Read	读取温度[characteristic]

SMP安全管理协议

- 建立链路层连接后，开始密钥生成与分发
- 三步：
  - 其中，第2步最为关键，涉及后续通信加密的安全性。



SMP安全管理协议

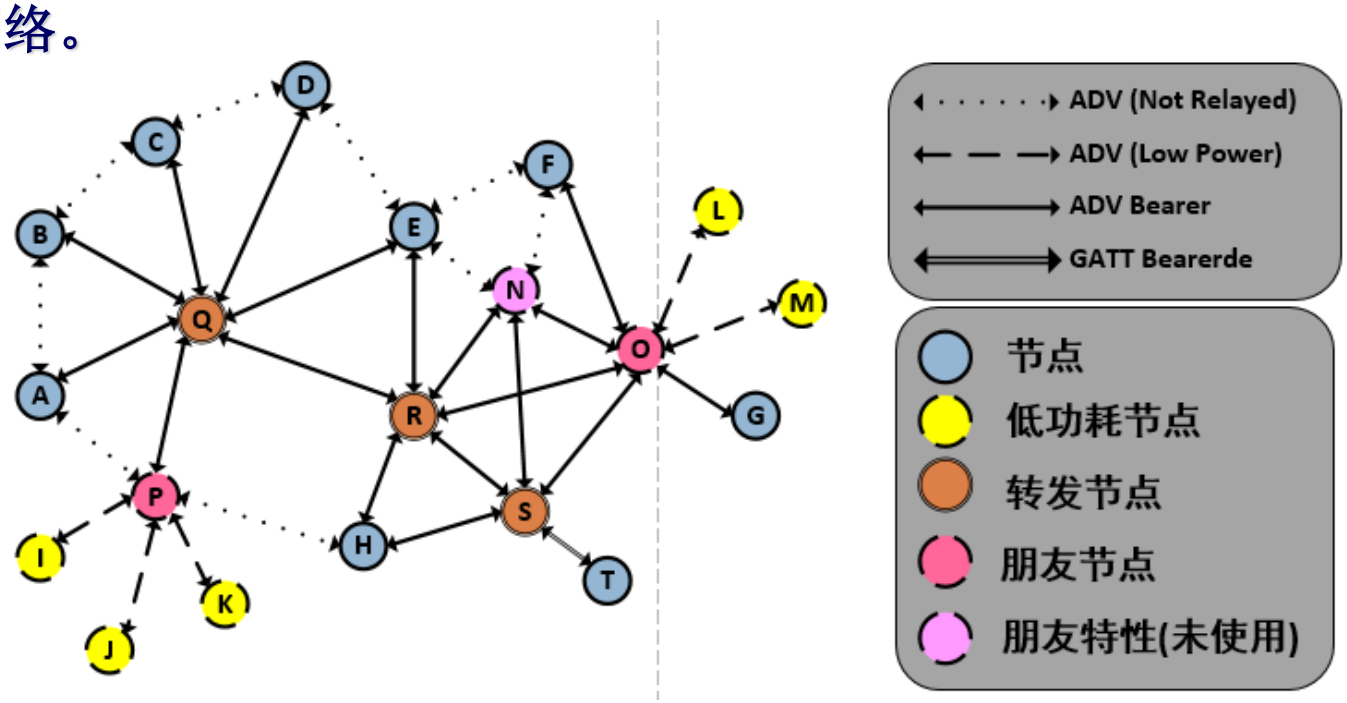
- 关键步骤2，密钥生成与分发
- 第一种机制：LE legacy pairing
  - 密钥生成方式
    - *master,slave 各自生成128位的STK，用于后续通信的对称加密*
    - *STK = s1(TK, Srand, Mrand)*
    - *s1()密钥生成函数*
    - *Mrand/Srand: Master/Slave生成的随机数*
      - 非加密通信下交换（*可能被监听*）
    - *TK: 128位的协商密钥（关键元素）*
  - TK密钥协商方式
    - *Just works (TK=0)*
    - *Passkey Entry*
      - 其中一个设备产生6位数字，用户口头告知另外一个用户，TK通过非无线信号方式传输，提供安全性
    - *OOB (Out of Band)*
      - NFC传输TK，假设NFC安全
  - TK密钥通过选择上述三种方式之一，得到了交换

SMP安全管理协议

- 关键步骤2，密钥生成与分发
- 第二种机制：LE Secure Connections
  - 非对称加密，Master和Slave交换公钥PKm,PKs。各自保存私钥SKm, SKs.
  - 密钥生成方式
    - *生成128位的LTK，用于后续通信的对称加密*
    - *x || LTK = f5(DHKey, Mr, Sr, Maddr, Saddr)*
    - *DHKey: Diffie-Hellman加密算法，MasterSlave各自得到*
      - DHKey=p256(PKm, SKs)
      - DHKey=p256(PKs, SKm)
      - 能够通过加密计算的方式共享key
    - *Mr/Sr: Master/Slave生成的随机数*
    - *Maddr/Saddr: Master/Slave的地址*
  - DHKey通过加密算法的方式得到了交换和保密

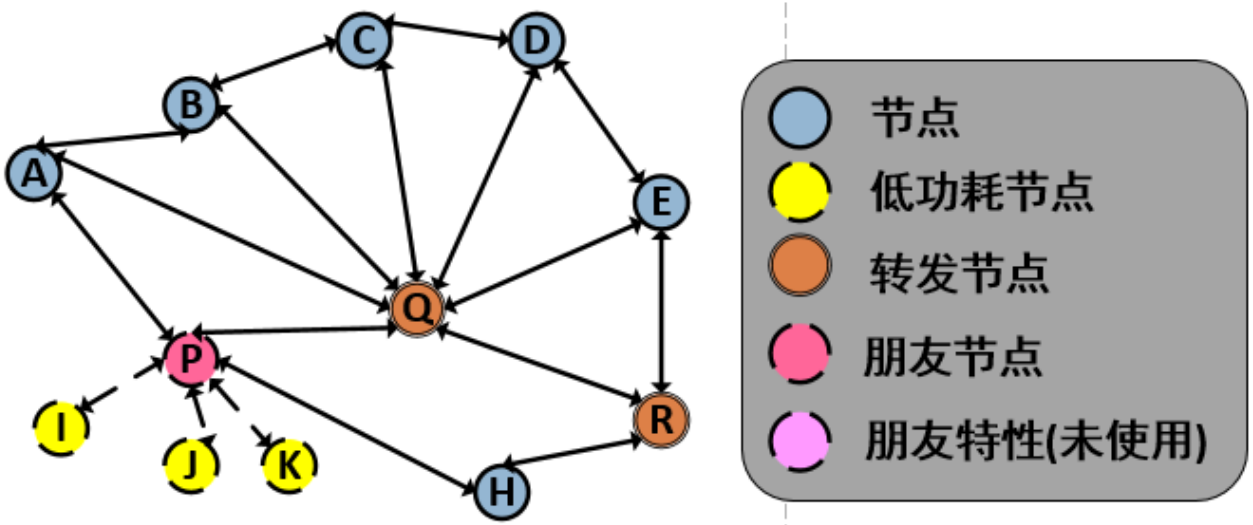
BLE Mesh

- **组网：四种链路类型**
  - (1) ADV(Not Relayed)：非转发节点的广播包收发
  - (2) ADV(Low power)：用于低功耗节点与朋友节点之间收发数据包，
  - (3) ADV Bearer：两个节点之间可以基于mesh协议收发广播消息，并且可以作为中继转发；
  - (4) GATT Bearer：用于没有mesh能力的传统节点也能参与MESH网络。



BLE Mesh

- 场景：节点J需要发送数据包给节点E，TTL值设定为3.
  - (1) 节点J发送数据包给节点P；
  - (2) relay节点P收到收据包。节点P将TTL修改为2，广播该数据包。
  - (3) 此时，节点A、Q以及H都会收到该数据包；
  - (4) 由于节点A以及节点H不是relay节点，将不会转发该数据包。
  - (5) relay节点Q收到数据包后，将TTL修改为1，广播该数据包。
  - (6) 此时，节点A、B、C、D、E以及R都会收到该数据包；
  - (7) 由于节点A、B、C、D不是relay节点，将不会转发该数据包。节点E收到数据包之后，判断DST地址是自己，收到数据包。节点R收到数据包时，虽然它是relay节点，但是由于TTL为1，节点R会丢弃该数据包。



BLE Mesh

- BLE的官方mesh仍然在改进中
- 最新的停留在2017年7月13日，第一个版本
  - https://www.bluetooth.com/specifications/mesh-specifications

Adopted Mesh Networking Specifications

Specification	Date Adopted	Status
Mesh Profile Specification 1.0	13 July 2017	Active
Mesh Model Specification 1.0	13 July 2017	Active
Mesh Device Properties 1.0	13 July 2017	Active

基于BLE的广告推送iBeacon应用

- 案例：利用BLE推送广告。
- 方案：BLE可以发送非连接的广播包，而广播包的数据对于开启了蓝牙的设备都能够监听到，从而达到广播推送。
  - 通过在广播包中加入公司特定的编号和数据，达到推送效果
  - 广播包payload格式
    - *Length: 数据长度(包括data type)，3字节~31字节*
    - *Data type: 数据类型, 比如，Flags(0x01), 设备名称(0x08)，发送能量(0x0A)，厂商特定ID(0xff).*
    - *Data: 实际的数据，长度为length-1.*

Length (1 字节)	Data type (1 字节)	Data (Length-1 字节)	…	Length (1 字节)	Data type (1 字节)	Data (Length-1 字节)

基于BLE的广告推送iBeacon应用

❑ 广播包的数据内容设定

```
static beaconAdvData_t beaconAdv =
{
  0x02,  // 长度2 字节
  0x01, // flag 的类型，表明广播包类型
  0x06, // 非可连接，可发现的广播包类型标志

  0x0B   // 长度 12字节
  0x08,   // 设备名称的数据类型
  “BLE devices”,  // 设备名称

  0x03,   //长度3 字节
   0xFF    //厂商ID
  0x4C,
  0x00,  // 苹果公司的ID，  0x004c

  0x03,  // 长度3字节
  0x0A,  // 发送能量的数据类型
  0x00,
  0x00   //测量的能量级别，可以在后面的操作中设置
};
```

基于BLE的广告推送iBeacon应用

❑ 广播事件间隔设定，100ms广播一次

```
// 广播间隔(625us的倍数, 因此160*625=100ms)
#define DEFAULT_ADVERTISING_INTERVAL        160
```

❑ BLE事件处理函数

```
static void SimpleBeacon_processStateChangeEvt(gaprole_States_t newState)
{
  switch ( newState )
  {
    case GAPROLE_STARTED:
      SimpleBeacon_startRegularAdv();   break; //开始广播
    case GAPROLE_ADVERTISING_NONCONN:
      SimpleBeacon_freeAttRsp(bleNotConnected);   break; //清空GATT反馈消息
    case GAPROLE_WAITING_AFTER_TIMEOUT:
     SimpleBeacon_freeAttRsp(bleNotConnected);
     SimpleBeacon_startRegularAdv();   break; //连接事件间隔之后，继续广播
   }
}
```

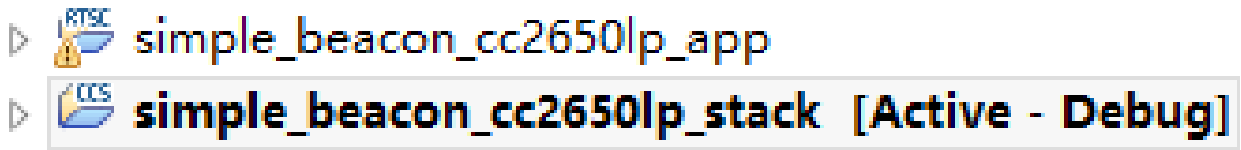
基于BLE的广告推送iBeacon应用

❑ BLE广播数据包发送函数：

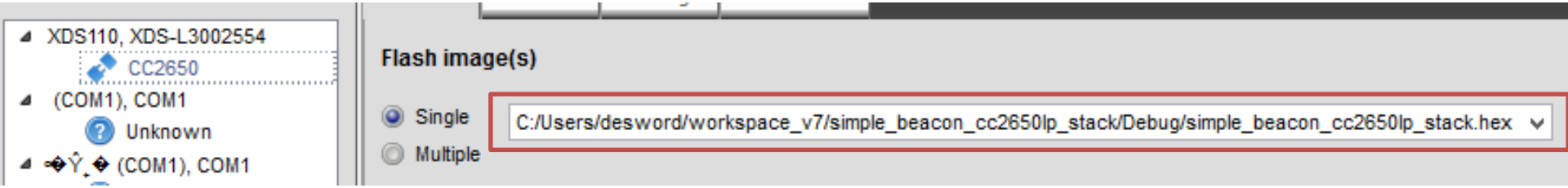
```
static void SimpleBeacon_startRegularAdv(void)
{
  uint8 advertEnabled = FALSE;
  uint8 advType = GAP_ADTYPE_ADV_NONCONN_IND; //设置不可连接的广播包类型

  advertEnabled = TRUE;

  GAPRole_SetParameter(GAPROLE_ADV_EVENT_TYPE,
              sizeof(uint8_t), &advType);                // 设置广播包类型

  GAPRole_SetParameter(GAPROLE_ADVERT_DATA,
                      beacon_PACKET_SIZE,
                      &beaconAdv); // 设置广播包的内容

  GAPRole_SetParameter(GAPROLE_ADV_NONCONN_ENABLED, sizeof(uint8_t),
                  &advertEnabled);                       // 设置启用广播
}
```

基于BLE的广告推送iBeacon应用

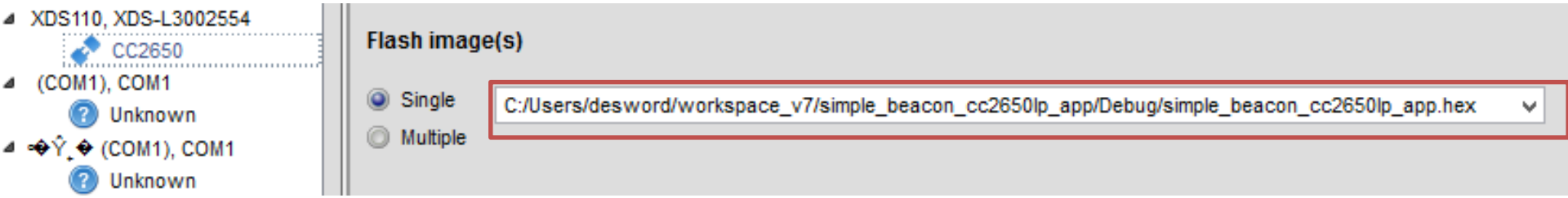
- 编译BLE协议栈和BLE应用



- 连接CC2650 sensortag到主机
- 使用smartRF flash 先上传BLE协议栈:xx_stack.hex

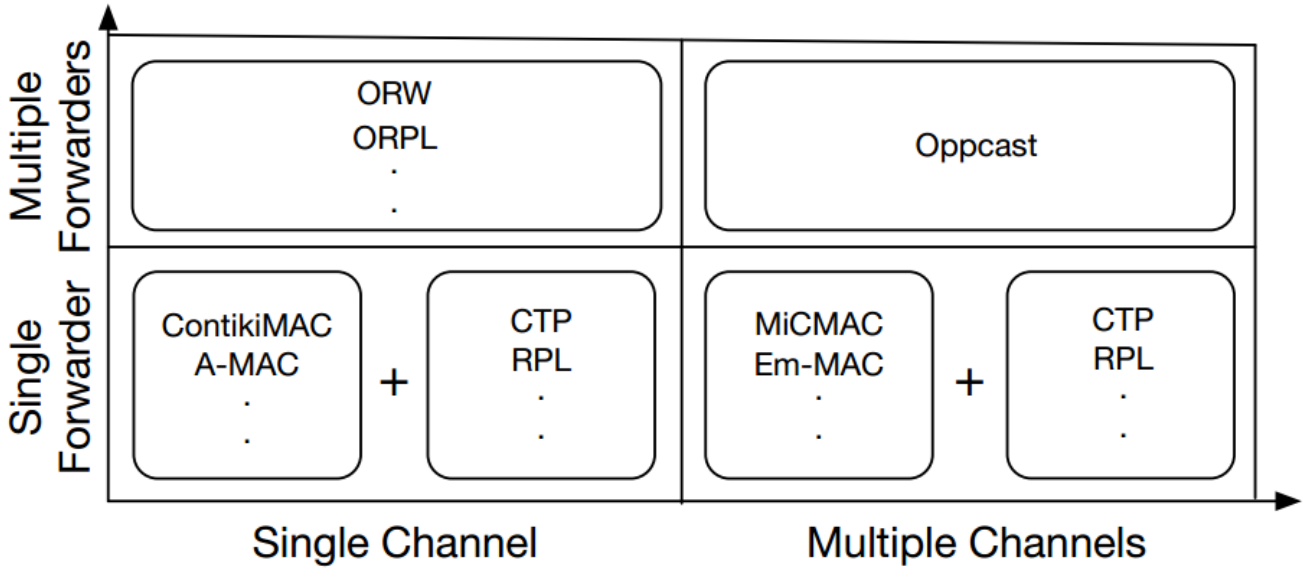


- 再上传BLE 应用:xx_app.hex



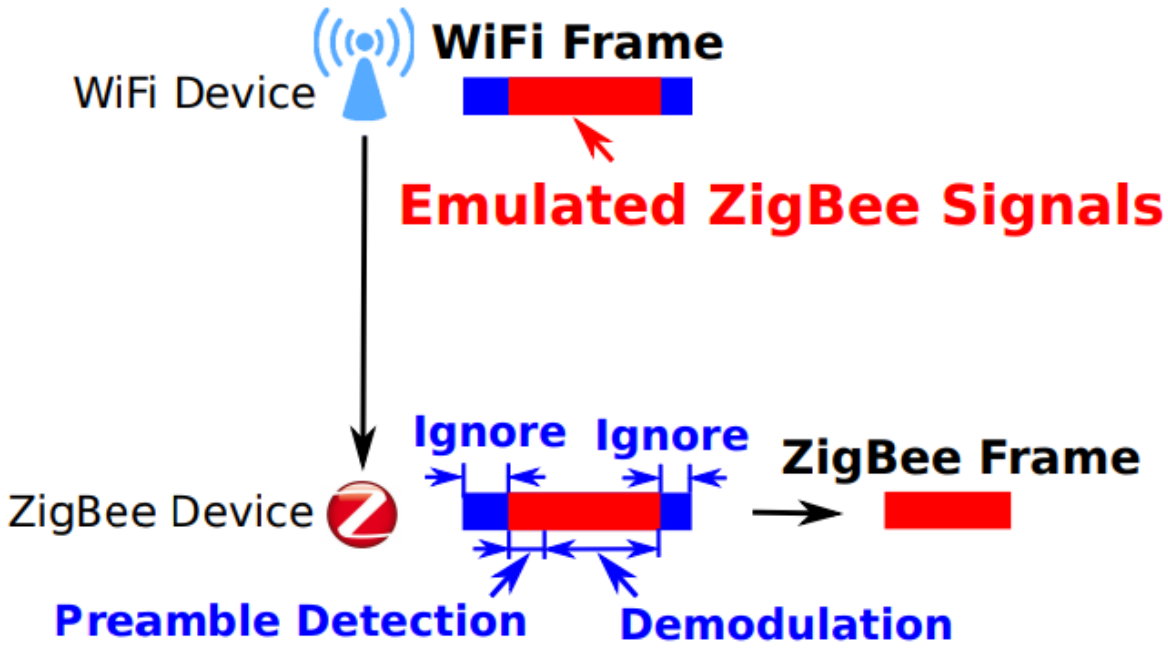
研究进展

- [Ipsn16] Oppcast: Exploiting Spatial and Channel Diversity for Robust Data Collection in Urban Environments
  - 提供了一种能够在干扰源动态变化的情况下，实现稳定的数据收集的路由协议。
  - Oppcast背后的关键理念是它同时利用了空间和信道多样性。
  - 通过结合两种机制：机会路由和多信道跳频的使用，Oppcast能够显著地增加可以用来稳定传输数据的链路，实现可靠的网络数据收集。



研究进展

- [mobicom17] WEBee: Physical-Layer Cross-Technology Communication via Emulation
  - 实现了WiFi到ZigBee的跨技术通信，能够达到以126Kbps的速率实现99%以上收包率的传输
  - 其背后原理是WiFi的物理层和ZigBee的物理层能够相互转换，通过在WiFi端选择特定的发送的数据，就能够在ZigBee端成功解码出跨技术通信的数据。



浙江大学
Zhejiang University

谢谢！

Email: dongw@zju.edu.cn