MacBee Protocol Documentation

Release 1

Galaxywind

Sep 06, 2017

CONTENTS:

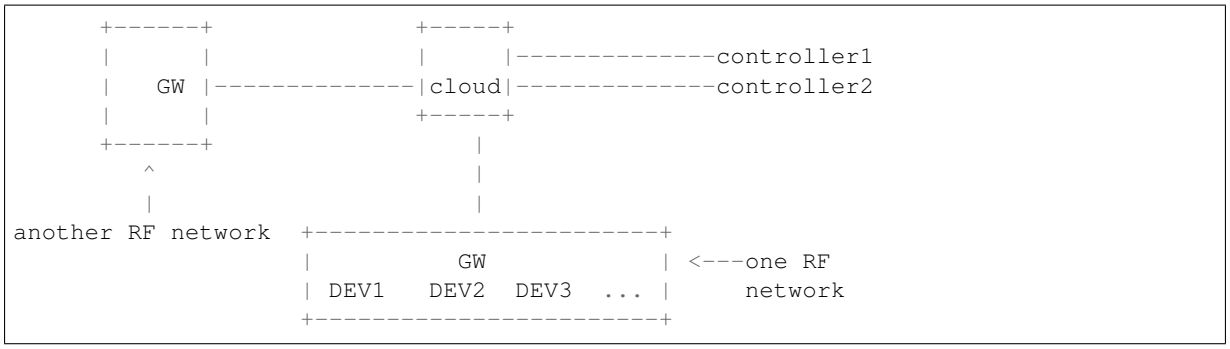
CHAPTER

ONE

MACBEE 协议规范

MacBee 协议是深圳市银河风云网络股份有限公司(以下简称银河风云)牵头制订的低速低功耗传感网络, 可以用于智能建筑, 智能家居, 智能照明等场景. MacBee 协议是传统 IP 网络和专有 RF 网络结合的协议.

1.1 网络拓扑



网关(Gateway)是连接传统 IP 网络和专有 RF 网络的关键部件, 网关连接到云上, 控制端也连接到云上, 在控制端的操作下, 可以完成网络的建立, 管理, 操作, 查询等. 控制端可以是Android或iOS设备上的应用程序, 也可能是PC端应用程序, 或者基于浏览器的网站.

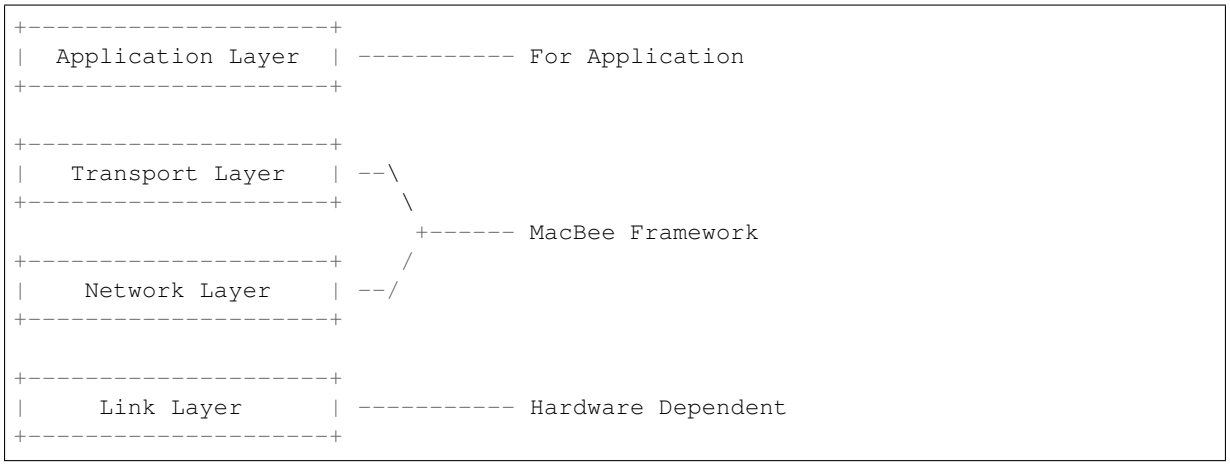
后面分别介绍 RF 网络和 IP 网络.

CHAPTER

TWO

MACBEE RF 层次结构

MacBee RF 网络分层(与 IP 网络类似, 参见 RFC 1122):



一共4层, 从下至上分别为: 链路层, 网络层, 传输层, 应用层. MacBee RF 网络为了移植方便, 不依赖特定链路层, 尽量不对链路层作过多的假定. 仅仅需要链路层(特定芯片)支持多地址单播传输(至少3个地址), 或者混杂模式, 软件作过滤; 连多播, 广播都不用支持, 也不需要芯片特定的功能.

CHAPTER

THREE

MACBEE RF 协议概述

在 MacBee RF 网络中, 有2种角色, 网关(Gateway)和设备(Device). 所有通信只能在网关和设备之间, 设备间不能直接通信. 网络内, 由网关负责协调整个网络的通信. 为了省电, 网关周期广播信标(Beacon); 设备需要侦听信标, 根据信标内容, 可能需要接收信息, 如果没有信息需要接收, 自己也没有信息发送, 那么就尽量休眠. 设备地址1字节(8bits), 一个网络最多240个设备(保留了16个地址).

3.1 时间组织

每个从信标(包含)开始直到下一个信标(不包含)前的时间构成1帧(Frame), 每个帧又划分固定的时隙(Slot), 每个时隙是不可再分的最小传输时间单位. 在当前版本中, 每个帧200ms, 每个时隙5ms, 所以每帧有40个时隙, 为了避免计算误差影响到Beacon, 最后1个时隙不传输数据. 信标编号, 从0开始, 到127(包含)结束; 之后下一个继续从0开始.

```
|       1 Frame        |     Next Frame      |
|00|01|02|03|...|38|39|00|01|02|03|...|38|39|
```

3.2 时隙分配

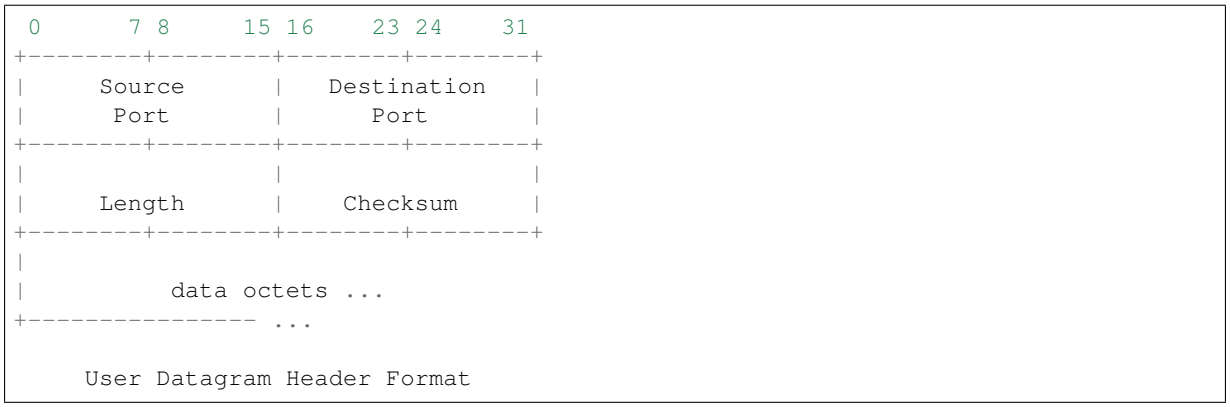
```
|                        1 Frame                        |
|00|01|02|03|...|16|17|18|19|...|32|33|34|35|...|38|39|
|B |R1|R2|R3|...|R |A1|A2|A3|...|A |K |K |S |...|S |N |
```

- 00 信标
- 1-16 设备接收数据(在信标中指定, 依次分配哪些时隙有数据及设备的地址)
- 17-32 设备需要发送应答报文(接收时隙+16).
- 33, 34 用于保活(Keep Alive), 每次会有2个设备报告自己还活着.
- 之后的4个时隙用于设备竞争主动发送给网关的, 网关尽量在下一个帧发送应答; 如果来不及, 必须在下下一个帧发送应答.
- 对于没有使用的接收和应答的时隙, 也可用于竞争主动发送.
- 39 保留, 不使用.

3.3 报文格式统一说明

所有报文字段都采用大端序, 格式描述类似 RFC.

摘录自 RFC 768(User Datagram Protocol):



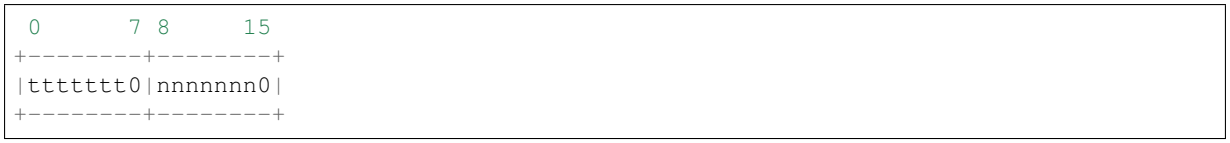
3.4 信标报文格式

信标报文比较特殊, 这里单独介绍. 又因为MacBee 不依赖特定物理层(特定芯片), 所以这里只有 MAC 层之上的报文格式.

3.4.1 头部

短格式:

长格式:



tttttt:

> 时间域(当前beacon发送时间在当前帧的偏移时间量), 对于短格式(1字节), 没有时间域, 等同于时间域为0. 时间单位是16us(也就是时间域*16=此帧开始的微秒数), 网关保证在16us的边界发送信标.

nnnnnnn:

> 编号域, 前面说过, 信标编号, 从0至127(都包含), 周而复始.

为了保证设备尽可能的接收信标, 网关会在时隙0发送3次信标, 第1次在时间0, 所以采用短格式; 之后时间不可能是0, 所以用长格式. 一般来讲, 设备在第一个报文发送时就收到了, 那么在此时隙剩余的时间中, 将不会再接收(节约能耗); 如果第一次没有收到, 继续接收, 直到时隙结束.

3.4.2 数据

在头部之后, 长度不定, 每个字节表示一个设备的地址, 第1个字节代表时隙1有发送给 A 设备的数据(对应的设备需要在此时隙接收, 其他设备知道时隙被使用了, 不得占用此时隙, 一般来讲, 和自己无关, 可以休眠); 第2个字节代表时隙2有发送给 B 设备的数据; 每个帧最多可以下发16个报文, 允许每帧给同一个设备发送多个数据, 必须占用最开始的连续时隙下发数据. 例如只有3个下发数据, 那么只能占用时隙1, 2, 3.

3.5 设备上电加入网络

```
                time ------>
Frame Time |<-0                          200ms->|
Device      -----S:Time?-RRRRRRRRRRR------------
Gateway     SRRRRRRRRRRR--S:Now is X.RRRRRRRRRRR
```

- 设备刚上电(或重启等), 和网关时间不同步, 无法接入网络.
- 随机主动发送对时报文(报文具体格式见后续描述,下同), 在数百 us 内切换到接收状态, 等待数 ms.
- 对于授权设备, 网关会尽快回复当前帧的偏移(offset)时间(特殊处理, 没有按 slot 划分).
- 设备收到网关回复, 计算下一个帧开始的时间(即信标的接收时间); 设备超时没收到回复, 重复随机发送.

CHAPTER

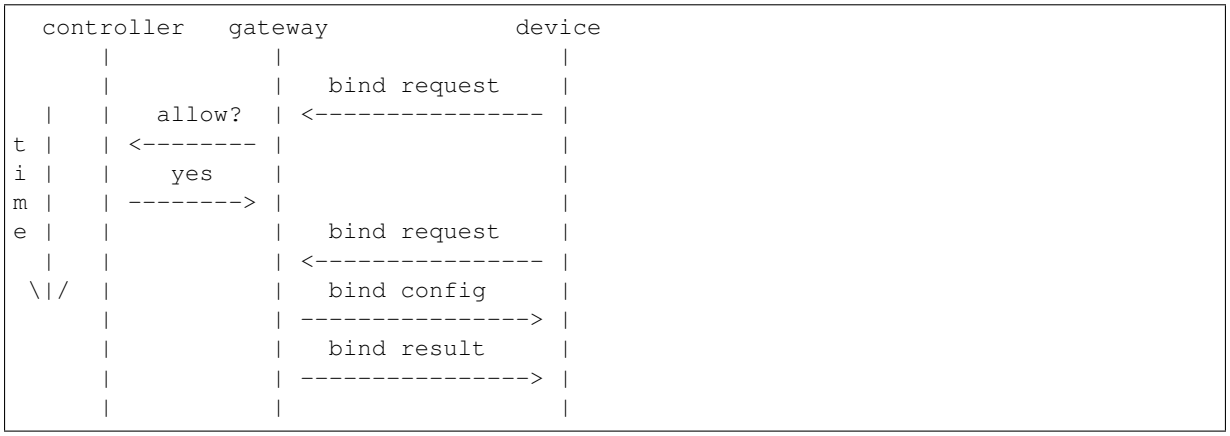
FOUR

MACBEE RF 绑定

绑定是新设备(或恢复出厂设置)需要加入网络的操作. 前2个报文(设备请求及网关的回复)除了绑定外, 还要完成和网关的时间同步, 之后的所有通信都在信标的控制下完成, 参见 设备上电加入网络.

4.1 步骤描述

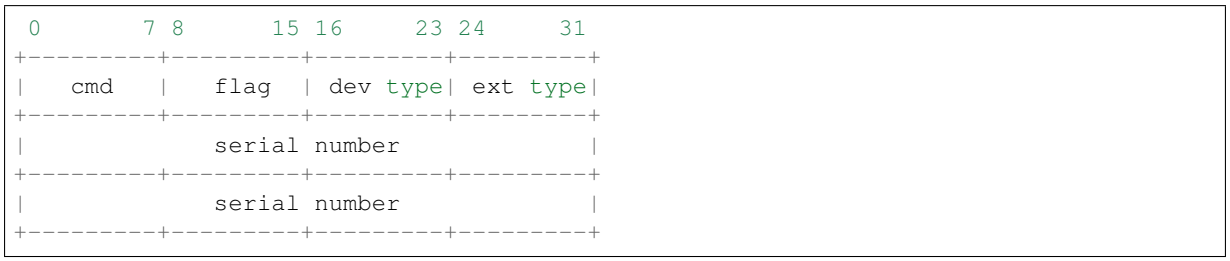
网关和设备交互时序图:



1. 设备发送绑定请求
2. 网关收到了请求, 通知控制端, 由使用者决定是否允许加入网络: 允许, 下次看到请求就会发送配置信息, 并且在下一个帧发送绑定成功, 包含网络配置信息, 设备序列号, 对时信息; 否则不应答.
3. 设备接收一小段时间(3ms-5ms), 期待接收配置信息(并对时), 收到后按照信标指示继续等待接收配置信息; 否则随机迟延一小段时间(平衡功耗与响应速度), 转到1.
4. 设备等待接收绑定成功信息, 收到保存网关发送的配置信息, 完成绑定操作; 如果超时, 转1.

4.2 报文格式

4.2.1 绑定请求



cmd=2

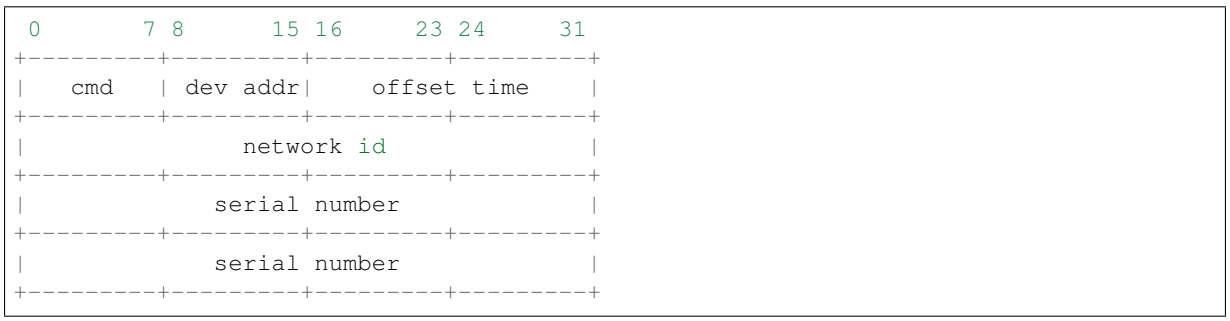
flag(按比特划分的某些特性, 可以组合):

- bit 1 设备升级时为1, 否则为0(在升级部分详细描述)
- 其他未使用比特保留, 必须是0

dev type, ext type: 设备的类型和子类型. 由 MacBee 联盟负责统一分配, 请咨询相关机构.

serial number: 设备的序列号(出厂时, 每个设备都会分配一个唯一的序列号).

4.2.2 绑定配置



cmd=3

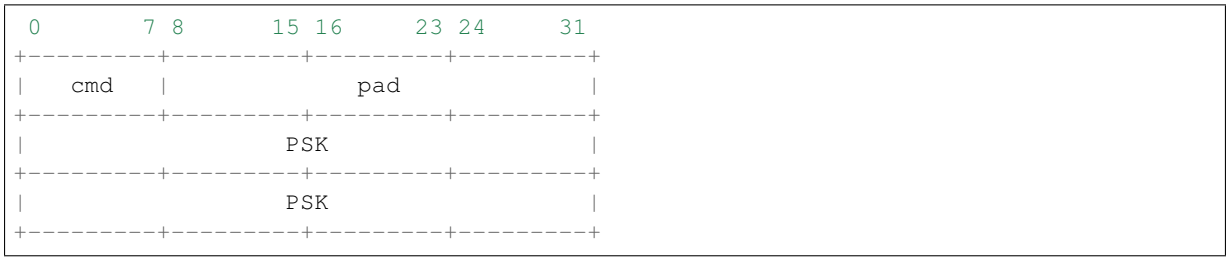
dev addr: 网关分配给设备的地址.

offset time: 报文开始发送时, 相对当前帧开始的偏移时间(以16us为单位, 即*16得到偏移的微秒).

network id: 网关的网络编号, 类似 WiFi 的 ESSID, 用来唯一区分网络.

serial number: 设备的序列号, 用来确认是网关是发送给自己的报文.

4.2.3 绑定成功



cmd=4

pad: 保留, 必须是0.

PSK: 预共享密钥, 用于后续认证及加密.

CHAPTER

FIVE

MACBEE RF 认证

认证是已经绑定的设备完成鉴权的操作. 前2个报文(设备请求及网关的回复)除了认证外, 还要完成和网关的时间同步, 之后的所有通信都在信标的控制下完成, 参见 设备上电加入网络. 这里只有网关对设备的认证, 设备对网关没有认证.

5.1 标准认证步骤描述

网关和设备交互时序图:

1. 设备发送认证请求, 包含序列号, 随机数1.
2. 网关收到报文, 就发送认证提问作为应答, 报文包含这次认证分配的地址, 设备序列号, 随机数2, 对时信息. 使用了保留的特殊地址表达网关没有绑定该设备.
3. 设备接收一小段时间(3ms-5ms), 期待接收认证提问(并对时).
4. 设备如果收到正确的认证提问, 且地址在[0,240), 注意左边包含, 右边不包含, 设置计数器10次, 转到5; 如果地址是网关未绑定, 就休眠直到下次上电或用户发出恢复出厂设置的指令; 如果报文错误或超时未收到, 随机迟延一小段时间(平衡功耗与响应速度), 转到1.
5. 设备在信标控制下, 随机选择未分配的时隙, 发送认证答案, 包含PSK及2个随机数的单向哈希, 设备的版本信息等.
6. 网关收到认证答案, 验证单向哈希是否正确, 无论是否正确, 都发送认证结果报文, 包含是否通过认证, 当前的 unix 时间戳(UTC 1970年1月1日0时0分以来的秒数)等信息.
7. 设备期待接收认证结果, 如果超过1-2秒(随机的)没有收到, 计数器-1, 不是0, 转到5; 计数器是0, 转到1. 收到认证结果, 转到8.
8. 收到认证结果, 检查是否认证通过, 如果通过认证, 认证结束, 进入正常通信; 否则, 转到1.

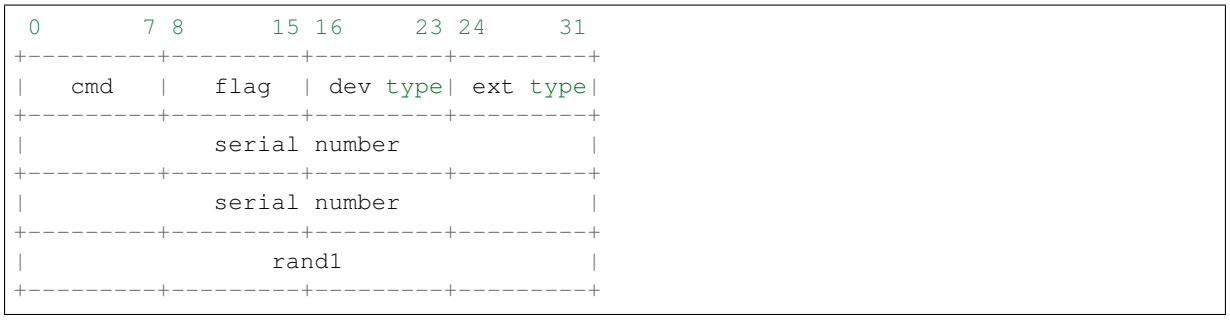
5.2 快速认证

对于已经认证通过的设备, 如果由于某些原因和网关时间不同步了, 可以发送快速认证报文, 设备自己的状态和网关认为该设备的状态都不会变化, 并且可以快速和网关取得时间同步.

1. 设备设置一个计数器(没有强制要求, 可以取3-100).
2. 设备发送认证请求(注意后面的报文描述, 包含特殊信息, 让网关知道是快速认证), 包含序列号, 随机数1(快速认证不使用, 可以填充任意数值).
3. 网关收到报文, 就发送认证提问作为应答, 报文包含这次认证分配的地址, 设备序列号, 随机数2(快速认证不使用), 对时信息. 使用了保留的特殊地址表达网关没有绑定该设备, 再用了一个保留地址表达快速认证通过.
4. 设备接收一小段时间(3ms-5ms), 期待接收认证提问(并对时).
5. 设备如果收到正确的认证提问, 且地址是快速认证通过, 完成对时并结束快速认证; 如果地址是网关未绑定, 就休眠直到下次上电或用户发出恢复出厂设置的指令; 如果报文错误或超时未收到, 随机迟延一小段时间(可以比标准认证更短的时间, 建议不超过1秒), 计数器-1, 如果不是0, 转到2; 如果计数器是0, 结束快速认证, 转到标准认证流程.

5.3 报文格式

5.3.1 认证请求



cmd=5

flag(按比特划分的某些特性, 可以组合):

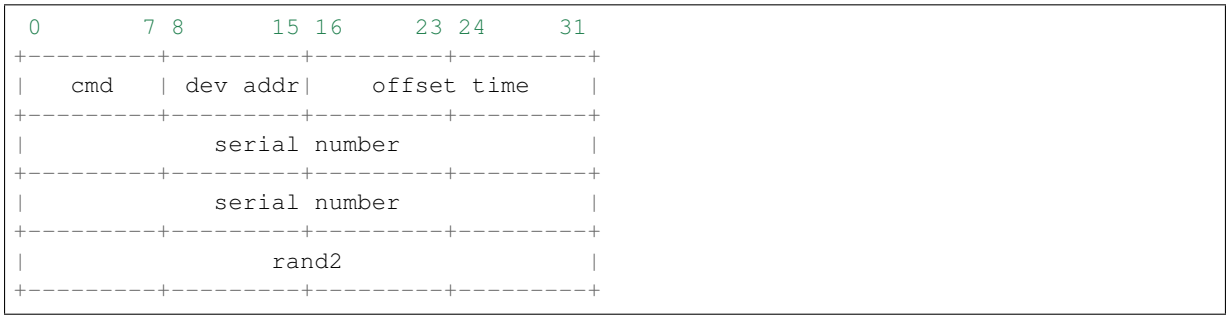
- bit 2 快速认证为1, 正常认证为0
- 其他未使用比特保留, 必须是0

dev type, ext type: 设备的类型和子类型. 由 MacBee 联盟负责统一分配, 请咨询相关机构.

serial number: 设备的序列号(出厂时, 每个设备都会分配一个唯一的序列号).

rand1: 设备产生的随机数.

5.3.2 认证提问



cmd=6

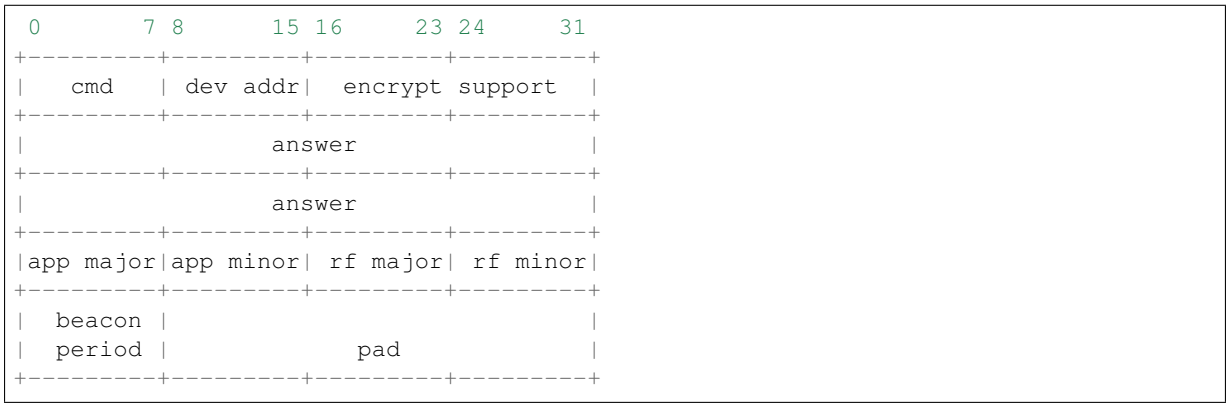
dev addr: 网关分配给设备的地址. 0xFF 表示快速认证通过, 0xFD 表示网关没有绑定该设备(或已经解除绑定).

offset time: 报文开始发送时, 相对当前帧开始的偏移时间(以16us为单位, 即*16得到偏移的微秒).

serial number: 设备的序列号(出厂时, 每个设备都会分配一个唯一的序列号).

rand2: 网关产生的随机数.

5.3.3 认证答案



cmd=7

dev addr: 网关分配给设备的地址.

encrypt support: 设备选择支持的加密算法, 16bits 代表16种算法, 0表示不加密.

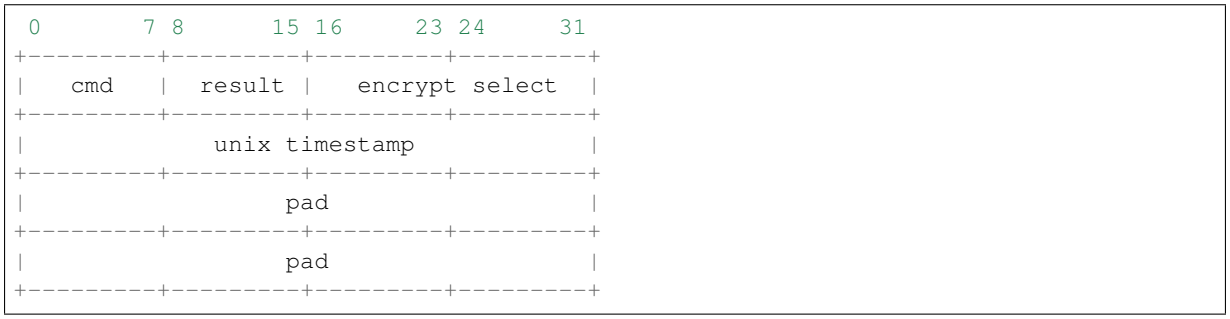
answer: 设备根据 PSK, rand1, rand2 计算单向哈希.

app major 和 app minor: 应用程序的主次版本号.

rf major 和 rf minor: MacBee RF 的主次版本号.

beacon period: 设备的周期选择, 只在信标编号是 beacon period 整倍数的帧才会接收, 进一步节省能耗.

5.3.4 认证结果



cmd=8

result: 0 认证答案错误, 1 认证答案正确, 通过认证.

encrypt select: 网关在设备支持的算法中选择1种用于本次会话. 在下次标准认证前, 一直参用此加密方式.

unix timestamp: 当前的 unix 时间戳(UTC 1970年1月1日0时0分以来的秒数).

CHAPTER

SIX

MACBEE RF 设备周期选择

前面说的都是 MacBee 标准周期设备, 即会在每帧开头接收信标; 这对某些设备而言, 功耗还是比较高. 我们允许设备选择只接收部分信标, 进一步降低功耗. 在设备认证成功之前, 都使用标准周期; 认证成功后, 使用 认证答案 定义的信标周期(beacon period).

具体规则如下:

- 设备在所有编号整除自己周期的帧(称为对齐), 都必须接收信标(所有设备都要接收帧0的信标).
- 任何时候, 设备接收信标失败或者当前帧有自己的数据或当前帧发送了数据, 下一帧必须接收信标; 否则在下一个对齐的周期的帧接收信标.
- 网关发送后没有收到应答, 持续在下一帧发送(目前发5次).
- 通过前2条, 可以爆发传输. 在网关主动下发时, 开始需要对齐设备周期, 之后爆发传输. 设备主动上报, 直接爆发传输. 同时, 最大程度简化了两者间的周期变化机制(没有同步报文).

CHAPTER

SEVEN

MACBEE RF 命令与报告

RF 设备与网关直接通信, 网关主动查询设备的信息使用命令报文(例如升级, 主动保活), 设备主动向网关报告信息使用报告报文(例如取得 UTC 时间).

7.1 报文格式

7.1.1 命令请求

头部:

```
 0        7 8       15 16      23 24      31
+---------+---------+---------+---------+
|   cmd   |   seq   | sub cmd |   pad   |
+---------+---------+---------+---------+
```

cmd=14

seq: 传输序列号, 过滤重传.

sub cmd: 进一步细分命令类型, 例如 主动保活请求 是4.

根据 sub cmd, 头部之后可能会有数据. 由链路层保证单播传输, 所以这里没有设备地址.

7.1.2 命令应答

头部:

```
 0        7 8       15 16      23 24      31
+---------+---------+---------+---------+
|   cmd   |   seq   |   addr  | sub cmd |
+---------+---------+---------+---------+
```

cmd=15

seq: 与请求保持一致.

addr: 设备地址.

sub cmd: 与请求保持一致.

CHAPTER

EIGHT

MACBEE RF 保活

RF 设备在多数时候, 不会发送报文给网关, 只是有选择的接收信标报文, 那么网关是不知道设备是否在线的. 所以, 设计了保活的机制.

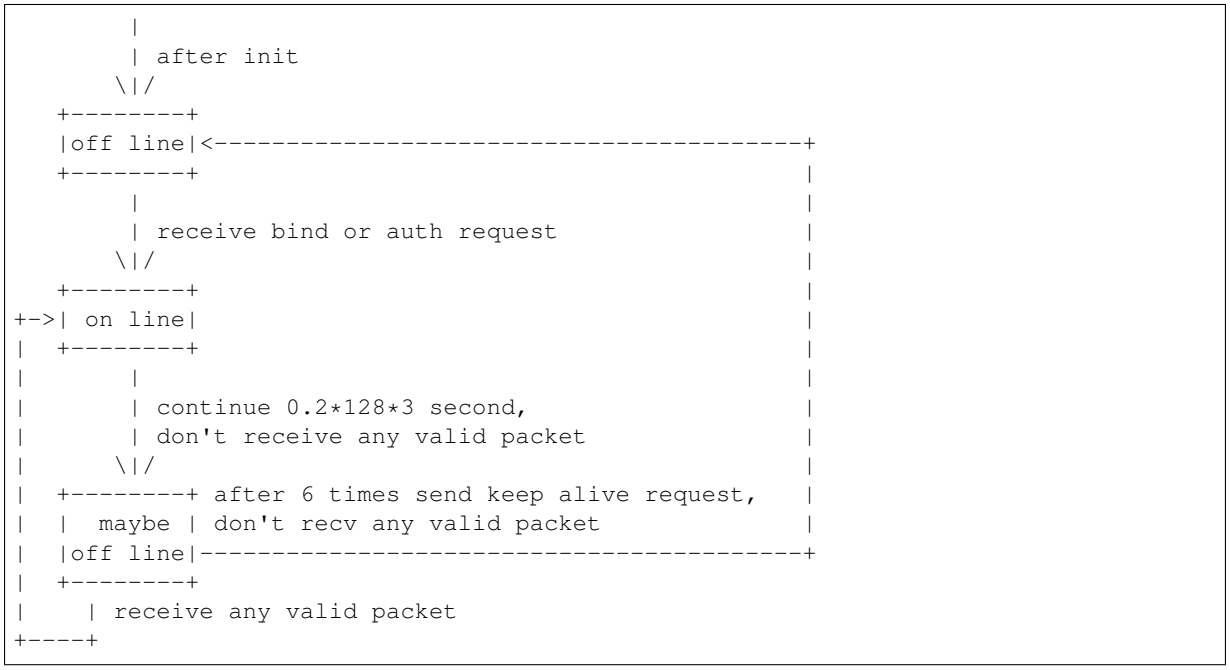
8.1 设备侧流程

在 信标报文格式, 我们描述了信标报文包含一个 [0, 128) 的编号(左闭右开), 并且在 时隙分配, 我们知道了有2个保留给保活的时隙. 信标编号和自己地址相同的设备使用第1个时隙发送, 信标编号+128和自己地址相同的设备使用第2个时隙发送. 这个称为被动保活. 如果设备由于某些意外情况, 错过了标准发送时隙, 应该尽快在竞争发送时隙发送理想保活报文. 网关也可能会要求设备应答还存活的报文, 设备收到后应该在应答时隙发送自己还存活的报文. 这个称为主动保活.

8.2 网关侧流程

一轮保活时间(beacon 编号从0开始, 再次回到0所经过的时间), 为25.6秒. 网关如果连续3轮时间没有收到设备发送的任何有效报文(可解析, 符合规范), 设置设备状态为可能离线(和在线等同, 除了会发送主动保活请求). 针对每个设备, 在可能离线状态, 每间隔至少32个帧周期(同时还要符合设备接收周期), 发送主动保活请求(仍然要符合设备的周期), 发送6次, 之后再等待1秒. 如果到此时, 网关没有收到设备的任何有效报文, 认为设备超时, 状态设置为离线.

设备保活状态变迁:



8.3 报文格式

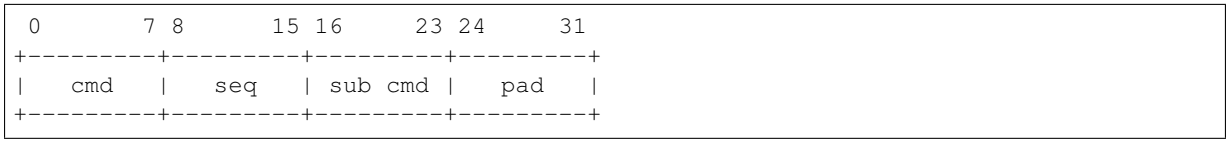
8.3.1 被动保活报告

cmd=9

addr: 设备的地址

网关不会应答该报文.

8.3.2 主动保活请求

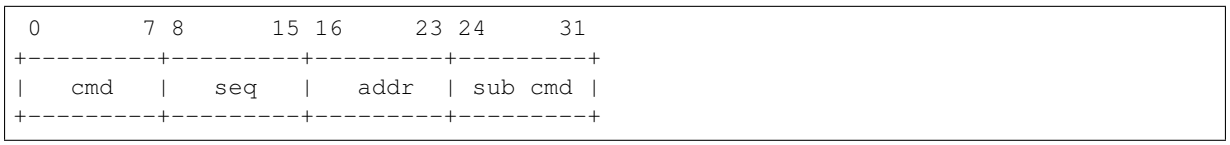


cmd=14

seq: 传输序列号, 过滤重传.

sub cmd=4

8.3.3 主动保活应答



cmd=15

seq: 与请求保持一致.

addr: 设备地址.

sub cmd=4

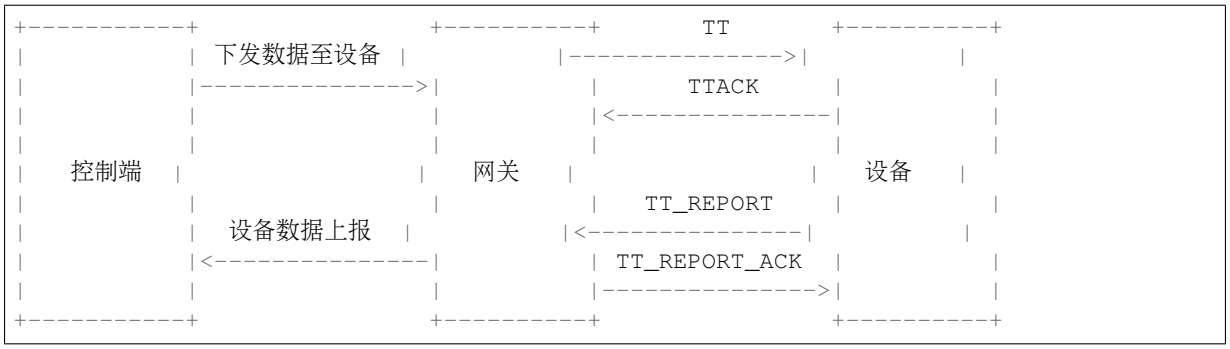
CHAPTER

NINE

MACBEE 透传

透传即控制端、网关、设备建立连接后，控制端与设备通过网关直接进行数据透明传输。

9.1 透传示意图

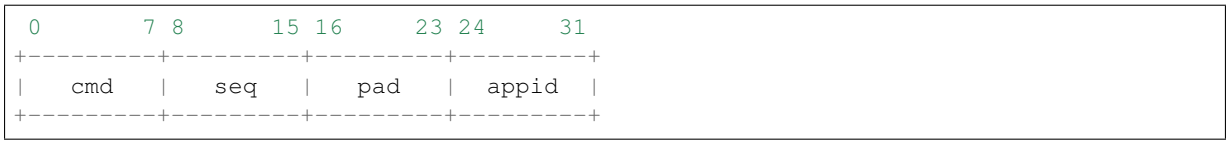


Note:

1. 下行数据透传：网关通过TT命令向设备发送透传数据，设备收到后回复TT_ACK进行确认，未收到确认报文，网关需进行重传。
2. 上行数据透传：设备通过TT_REPORT命令向网关透传数据，网关收到后回复TT_REPORT_ACK进行确认，未收到确认报文，设备需进行重传。

9.2 报文格式

9.2.1 TT



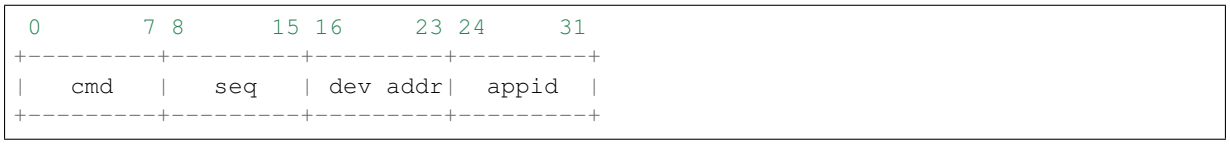
cmd=10

seq:传输序列号，抗重传。

pad:保留。

appid:控制端标识。网关通过该标识区分控制端。

9.2.2 TTACK



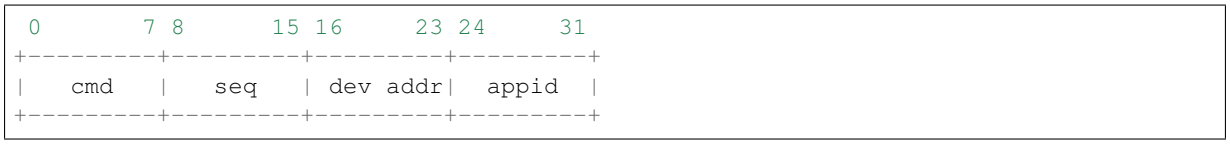
cmd=11

seq:传输序列号，抗重传。

dev addr:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

9.2.3 TT_REPORT



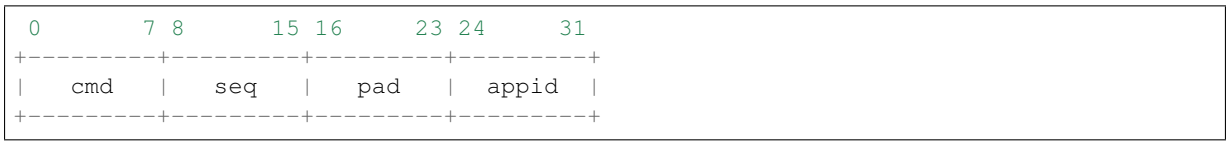
cmd=12

seq:传输序列号，抗重传。

dev addr:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

9.2.4 TT_REPORT_ACK



cmd=13

seq:传输序列号，抗重传。

pad:保留。

appid:控制端标识。网关通过该标识区分控制端。

CHAPTER

TEN

MACBEE CACHE 机制

设备将一些关键数据缓存在网关，便于控制端快速获取，及时同步。网关支持每个设备最大28字节的缓存数据。

10.1 交互流程

1.网关无设备的cache数据，通过TT_CACHE命令向设备查询，设备收到后通过TT_CACHE_ACK回复cache数据，未收到回复，网关进行重传。

```
+--------+  TT_CACHE     +-------+
|        |-------------->|       |
| 网关   | TT_CACHE_ACK | 设备 |
|        |<--------------|       |
+--------+               +-------+
```

2.设备的cache数据有变化，通过TT_REPORT_CACHE命令上报至网关，网关收到后回复TT_REPORT_CACHE_ACK进行确认，未收到回复，设备进行重传。

```
+--------+  TT_REPORT_CACHE     +-------+
|        |--------------------->|       |
| 网关   | TT_REPORT_CACHE_ACK | 设备 |
|        |<---------------------|       |
+--------+                      +-------+
```

10.2 报文格式

10.2.1 TT_CACHE

```
 0       7 8      15 16     23 24     31
+---------+---------+---------+---------+
|   cmd   |   seq   |   pad   |  appid  |
+---------+---------+---------+---------+
```

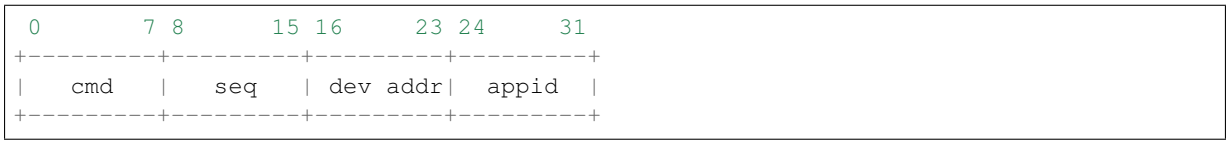
cmd=25

seq:传输序列号，抗重传。

pad:保留。

appid:控制端标识。网关通过该标识区分控制端。

10.2.2 TT_CACHE_ACK



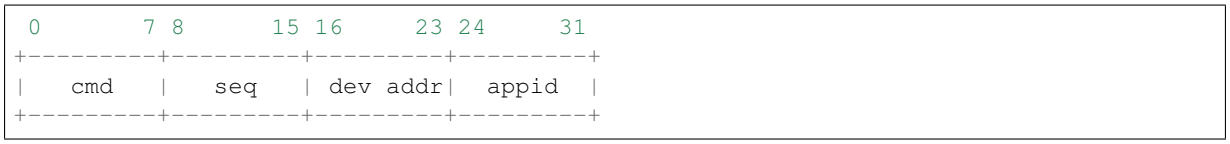
cmd=26

seq:传输序列号，抗重传。

dev addr:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

10.2.3 TT_REPORT_CACHE



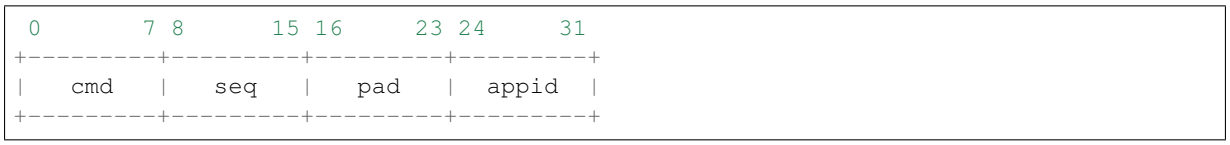
cmd=27

seq:传输序列号，抗重传。

dev addr:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

10.2.4 TT_REPORT_CACHE_ACK



cmd=28

seq:传输序列号，抗重传。

pad:保留。

appid:控制端标识。网关通过该标识区分控制端。

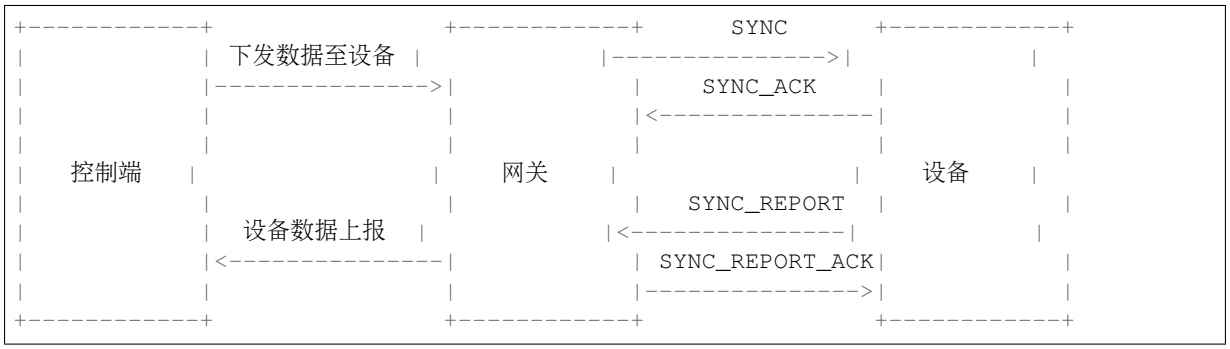
CHAPTER

ELEVEN

MACBEE 数据同步

无线传输低速率和不可靠的特点，使得报文在控制端，MacBee网关和MacBee设备整条传输路径中，容易在MacBee网关堆积或丢失。 而实际应用中，一种控制命令下发或一种状态的获取，不需要每个报文都传输成功，只需要最后的控制命令或状态发送成功，不是最终状态的报文都不必发送成功，以下称这种通信方式为“数据同步”。 数据同步的以一个MTU能容纳的数据为基本单位。MacBeeMTU是32字节，数据同步报文头部6字节，剩下26字节用作数据传输。数据同步报文比透传报文头部多2字节，用以传输数据同步需要的额外信息。 在数据同步报文头部有一个idx字段，同一idx代表同一个数据同步单位，称为“块”。用idx来区分不同的块，即对同一块报文传输只需保证最后一次可靠传输。 数据同步报文头部另外有一个字段digest，代表块的摘要，就是设备中块的变化次数。

11.1 数据同步示意图

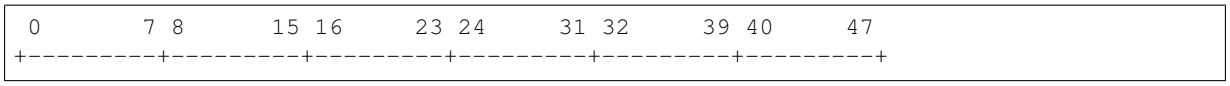


Note:

1. 下行数据同步：下行同步又分为GET和SET，而GET和SET又细分为网关缓存方式和非网关缓存方式。网关缓存方式是设备在回SET或GET ACK的时候，使用ACK_CACHE命令，网关会将收到的数据缓存起来，可以用来直接回控制端的数据请求。重传可由控制端做，对每一块都保证收到最后一个报文的ACK。
2. 上行数据同步：上行同步分网关缓存和非网关缓存方式。网关缓存方式是网关会将收到的数据缓存起来，可以用来直接回控制端的数据请求。设备端控制重传，对每一块保证收到最后一个报文的ACK。

11.2 报文格式

11.2.1 GET



```
|   cmd    |  subidx | pad      | pad      |  idx     |    pad   |
+---------+---------+---------+---------+---------+---------+
```

cmd=32

subidx:由于idx最多256个，在某些应用中不够，尤其是GET，所以GET在每个idx的基础上，可以扩展256个，总量达到256*256的GET块。

pad:保留。

idx:块号。

11.2.2 GET_ACK/GET_ACK_CACHE

```
 0       7 8      15 16     23 24     31 32     39 40     47
+---------+---------+---------+---------+---------+---------+
|   cmd    |  subidx |client_id|  app_id |  idx     | digest   |
+---------+---------+---------+---------+---------+---------+
```

cmd=34/35

subidx:参考GET。

client_id:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

idx:块号。

digest:块变化次数。

11.2.3 SET

```
 0       7 8      15 16     23 24     31 32     39 40     47
+---------+---------+---------+---------+---------+---------+
|   cmd    |  seq     | mask_hi |  app_id |  idx     | mask_low|
+---------+---------+---------+---------+---------+---------+
```

cmd=36

seq:序列号，发送者以此来确定最后发送的报文收到ACK没有。

mask_hi:mask一共16位，前13位对应数据26字节是否有效，1位代表2字节是否有效。

appid:控制端标识。网关通过该标识区分控制端。

idx:块号。

mask_low:参考mask_hi。

11.2.4 SET_ACK/SET_ACK_CACHE

```
 0       7 8      15 16     23 24     31 32     39 40     47
+---------+---------+---------+---------+---------+---------+
|   cmd    | seq      |client_id|  app_id |  idx     | digest   |
+---------+---------+---------+---------+---------+---------+
```

cmd=37/38

seq:回SET一样的序列号。

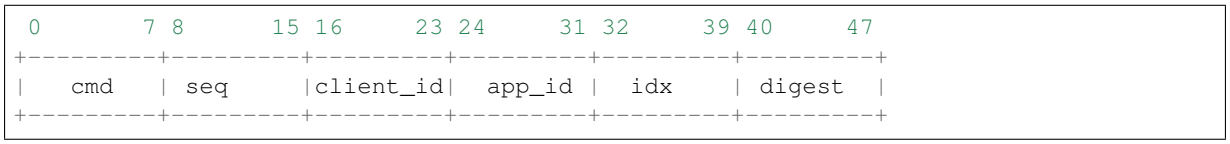
client_id:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

idx:块号。

digest:块变化次数。

11.2.5 REPORT/REPORT_CACHE



cmd=39/40

seq:设备产生的序列号。

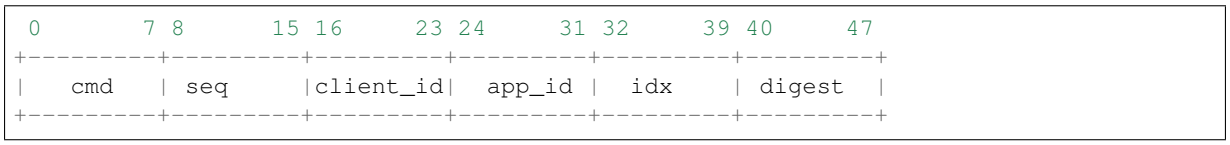
client_id:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

idx:块号。

digest:块变化次数。

11.2.6 REPORT_ACK



cmd=41

seq:网关回复和设备发送的序列号相同。

client_id:网关分配给设备的地址。

appid:控制端标识。网关通过该标识区分控制端。

idx:块号。

digest:块变化次数。

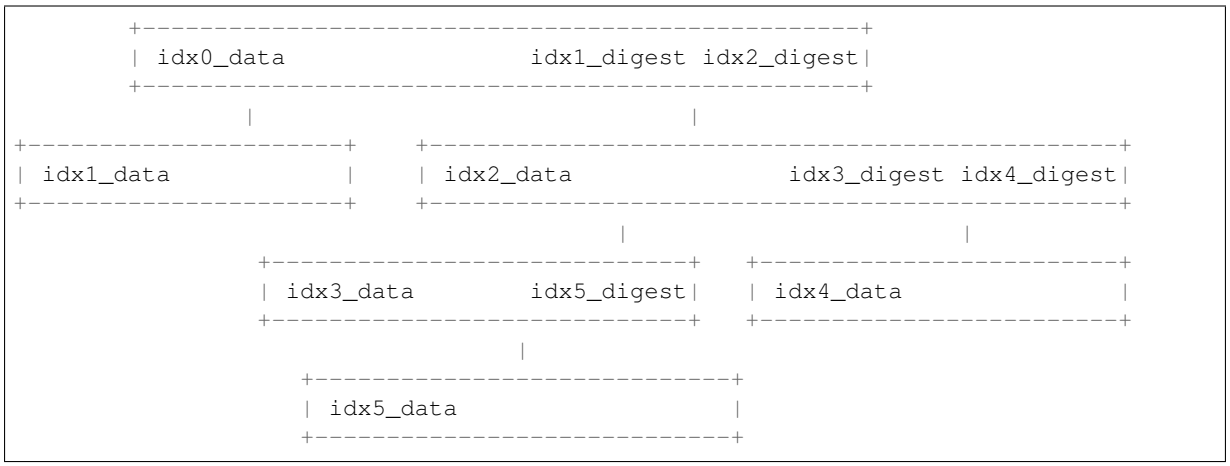
11.3 同一个数据块多个操作同时发生

GET只要有该块的数据任何数据收到，就认为完成，不论是GET ACK，SET ACK，还是REPORT。SET一定要收到自己发送的最后一个seq的SET_ACK才认为完成。 REPORT不要求发送成功，REPORT_CACHE要求收到和最后的报文头部一样的ACK才认为完成。 在控制端，一个块同时有GET，SET，就发送SET，SET ACK收到后，SET和GET都完成了。 在设备端，一个块同时存在REPORT，SET ACK和GET_ACK,优先级REPORT>SET ACK，GET ACK在发送完SET ACK后，不用发。

11.4 树形结构数据同步

树形结构数据同步建立在基本块结构同步基础之上，主要解决设备可靠上传报文的问题。 设备的数据变化了，由设备主动可靠推送给所有控制端，代价太大了，网关，设备都可能成为瓶颈。 设计思想是只让第0块可靠地同步到每个控制端，第0块为树根，数据区包含第二级子节点块的摘要，第二级子节点又包

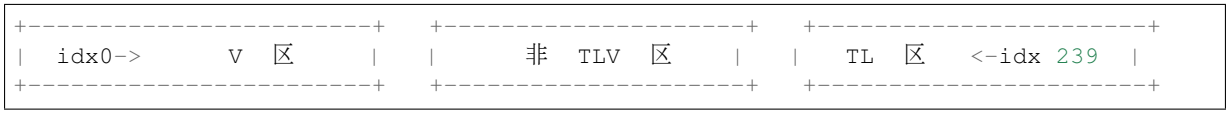
含第三级子节点的摘要。 当控制端收到第0块，就能至上而下地根据摘要判断每一块是不是最新，如果不是最新，就重新取这一块。



同步数据块的数据区是26字节，子节点的摘要放在块的最后，即右对齐。 若一个块有几个子节点，那么能容纳的同步数据就少几个字节。比如上图中，idx0有两个子节点，那么必定是数据区第25个字节为idx较小的子节点摘要，第26个字节为idx较大的子节点摘要。 数据区可以不填满，遵循左对齐原则，数据区到摘要之间的区域不使用。 树排使用紧凑排列的块，比如总共有10块数据同步，那么一定是idx0~idx9，且idx0为树根。 树排好后，idx的分配，按照层数越大，idx越大的原则。同一层中，idx按照右比左大的原则。 树的描述：可以用一组数据来描述树的组成，规则为：一个uint8数组，数组下表和idx号是相同的，数组内容是该数组单元对应的idx所含子节点数量，如果从某个idx开始，子节点数量全为0，可以不写。 上图中树的数组描述为：202100，或省略后面的全0为2021。 定义在以idx0为根的树中的块，由框架根据摘要提供自动同步机制，使用尽量少的通信量。未定义在以idx0为根的树中的块，由应用程序编写者决定何时触发同步，即触发一次，同步一次，同步仍是可靠的。

11.5 TLV同步

树形块的同步框架实现了块的可靠传输机制，从应用程序编写人员看到的是从idx0到idx255的线性块。在设备端，修改某个块会同步到所有控制端。一个控制端修改某个块会同步到设备端，并同步到其他控制端。 这样就可以用块来映射设备的数据结构，而控制端编程人员可以像操作本地内存一样去操作它，并得到变化通知。 但程序出新版本的时候，数据结构不能随便修改，否则控制端就和设备端不兼容了。为了解决这个问题，可以将块映射为TLV缓冲区。 但TLV又会带来同步数据增大，进而增加了MacBee通信量，每次修改TLV的时候，实际上只修改V，但都要把TL也传输一次，浪费传输带宽。所以将TL和V分开，V变化了就要传，但TL只传输一次。 把所有V紧凑排列单独放在一些块，TL也紧凑排列放在另外一些块，那么TL所在块只要设备端软件不变，就永远不会变。 从idx0向上生长，划为V区，从idx239向下生长，划为TL区，中间为非TLV区，应用程序自己决定使用方式。



一般来说，V区放在以idx0为树的根中，由框架自动同步，TL区也由框架获取，非TLV区由应用程序编写者触发同步。 这样，展现在应用程序编写者面前的，就是一组TLV，对其操作能反映到对方，在新版本中，可以增加或删除TLV，很容易实现向前向后兼容。 TL区除了放TL信息外，还存放树的描述，控制端收到描述后，可以还原树的结构来构造树。所以在新版本中，可以修改树的结构而不影响原有控制端。 不同设备的TL区都不一样，同一设备不同版本TL区也不一样，但只要同一设备，同一版本，TL区是一样的，可以根据这个特性，减少TL区的次数，所以TL区也称为“模板”。 控制端根据设备认证时的版本号，查询库中是否有模板，如果没有，才从设备获取。

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search