# 音视频学习

|  |  |
| --- | --- |
| 日期: | 2021.04.12 |
| By | 叶荣跃 |
| 版本 | V1.0.0 |

抄于：

https://blog.csdn.net/leixiaohua1020/article/details/18893769

# **一．视频播放器原理**

## 视音频技术主要包含

视音频技术主要包含以下几点：封装技术，视频压缩编码技术以及音频压缩编码技术。如果考虑到网络传输的话，还包括流媒体协议技术。

视频播放器的源代码详细解析（Media Player Classic - HC，Mplayer，FFplay，XBMC）可以参考系列文章：

Media Player Classic：Media Player Classic - HC源代码分析 1：整体结构[系列文章]

http://blog.csdn.net/leixiaohua1020/article/details/13280659

Mplayer：MPlayer源代码分析

http://blog.csdn.net/leixiaohua1020/article/details/11885509

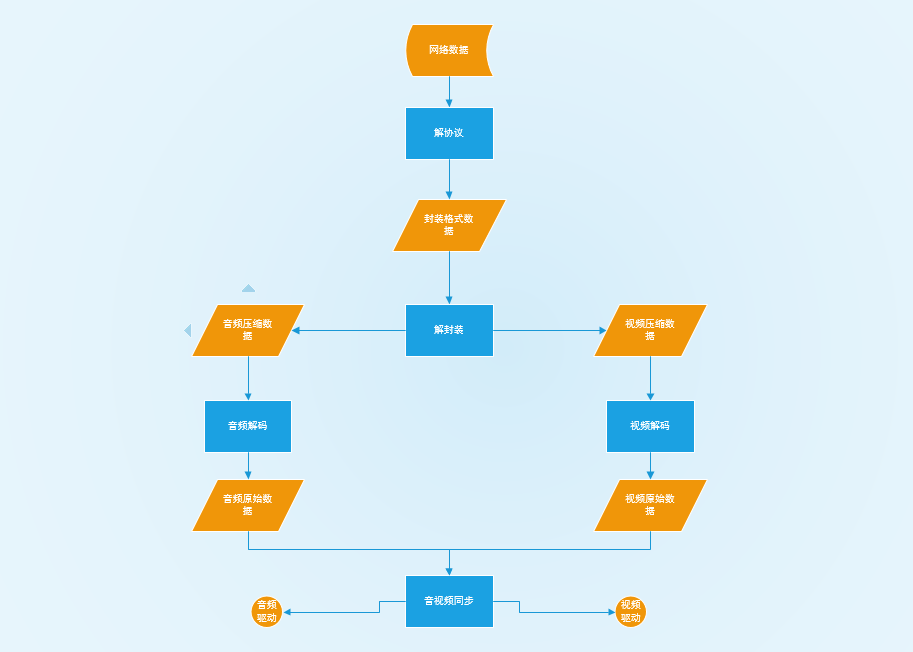
FFplay： FFplay源代码分析：整体流程图

http://blog.csdn.net/leixiaohua1020/article/details/11980843

XBMC： XBMC源代码分析 1：整体结构以及编译方法[系列文章]

<http://blog.csdn.net/leixiaohua1020/article/details/17454977>

视频播放器播放一个互联网上的视频文件，需要经过以下几个步骤：解协议，解封装，解码视音频，视音频同步。如果播放本地文件则不需要解协议，为以下几个步骤：解封装，解码视音频，视音频同步。他们的过程如图所示。



#### 解协议

解协议就是将采用HTTP，RTMP，或是MMS等流媒体协议的数据解析成标准的封装格式数据，这个过程去掉了信令数据，只保留了音视频数据。

#### 解封装

将MP4、MKV、RMVB、TS、FLV、AVI等封装格式数据分离成视频压缩数据和音频压缩数据。FLV格式的数据，经过解封装操作后，输出H.264编码的视频码流和AAC编码的音频码流。

#### 解码

将音视频压缩数据解析成原始数据，视频压缩数据—>YUV420P、RGB数据，音频-->PCM抽样数据

#### 音视频同步

根据解封装的数据信息去同步解码得到的音视频数据，并播放

#### 相关链接

流媒体协议，封装格式，以及视音频编码标准。更详细的比较可以参考：

[视频参数（流媒体系统，封装格式，视频编码，音频编码，播放器）对比](http://blog.csdn.net/leixiaohua1020/article/details/11842919)

<https://www.cnblogs.com/yuweifeng/p/8468744.html>

<https://blog.csdn.net/wudebao5220150/article/details/13016871>

本文中涉及到的协议数据、封装格式数据、视频编码数据、音频编码数据、视频像素数据、音频采样数据的分析可以参考下面系列文章：

[视音频数据处理入门：RGB、YUV像素数据处理](http://blog.csdn.net/leixiaohua1020/article/details/50534150)

[视音频数据处理入门：PCM音频采样数据处理](http://blog.csdn.net/leixiaohua1020/article/details/50534316)

[视音频数据处理入门：H.264视频码流解析](http://blog.csdn.net/leixiaohua1020/article/details/50534369)

[视音频数据处理入门：AAC音频码流解析](http://blog.csdn.net/leixiaohua1020/article/details/50535042)

[视音频数据处理入门：FLV封装格式解析](http://blog.csdn.net/leixiaohua1020/article/details/50535082)

[视音频数据处理入门：UDP-RTP协议解析](http://blog.csdn.net/leixiaohua1020/article/details/50535230)

# **二．流媒体协议**

## 流媒体概述

流媒体指的是采用流式传输的方式在Internet播放的媒体格式。流式传输方式是将视频和音频等多媒体文件经过特殊的压缩方式分成一个个压缩包，由服务器向用户计算机连续、实时传送。

[各种流媒体详细介绍](https://blog.csdn.net/xiaomucgwlmx/article/details/102851352)

## RTP：实时传输协议（Real-time Transport Protocol）

RTP是一种基于包的传输协议，用于实时传输数据。用于Internet上针对多媒体数据流的一种传输层协议。在网络上传输数据包延迟是不可避免的，针对此RTP的包头包含了时间戳、丢失保护、载荷标识、源标识以及安全信息。通过这些在应用层实现数据包的丢失恢复、拥塞控制等。运行于UDP上层，利用其复用求和校验功能，采用UDP/IP包封装：

|  |  |  |  |
| --- | --- | --- | --- |
| IP头 | UDP头 | RTP头 | RTP载荷 |

RTP 本身并没有提供按时发送机制或其它服务质量（QoS）保证，它依赖于低层服务去实现这一过程。 RTP 并不保证传送或防止无序传送，也不确定底层网络的可靠性。 RTP 实行有序传送， RTP 中的序列号允许接收方重组发送方的包序列，同时序列号也能用于决定适当的包位置，例如：在视频解码中，就不需要顺序解码。

         RTP 由两个紧密链接部分组成： RTP ― 传送具有实时属性的数据；RTCP 控制协议 ― 监控服务质量并传送正在进行的会话参与者的相关信息。

## RTCP：实时传输控制协议（Real-time Transport Control Protocol）

是实时传输协议（RTP）的一个姐妹协议。RTCP为RTP媒体流提供信道外（out-of-band）控制。RTCP本身并不传输数据，但和RTP一起协作将多媒体数据打包和发送。RTCP定期在流多媒体会话参加者之间传输控制数据。RTCP的主要功能是为RTP所提供的服务质量（Quality of Service）提供反馈。

        RTCP收集相关媒体连接的统计信息，例如：传输字节数，传输分组数，丢失分组数，jitter，单向和双向网络延迟等等。网络应用程序可以利用RTCP所提供的信息试图提高服务质量，比如限制信息流量或改用压缩比较小的编解码器。RTCP本身不提供数据加密或身份认证。SRTCP可以用于此类用途。

## SRTP & SRTCP：安全实时传输协议（Secure Real-time Transport Protocol）

SRTP实际上就是在RTP实时传输协议的基础定义的一个协议，旨在单播或多播的应用程序中的实时数据传输提供数据加密、消息认证、完整性保证和重放保护。最早由David Oran（思科）和Rolf Blom（爱立信）开发，并最早由IETF于2004年3月作为RFC3711发布。

## RTSP：控制声音或影像串流协议（Real Time Streaming Protocol）

 是由Real Networks和Netscape共同提出的。该协议定义了一对多应用程序如何有效地通过IP网络传送多媒体数据。RTSP提供了一个可扩展框架，使实时数据，如音频与视频的受控、点播成为可能。数据源包括现场数据与存储在剪辑中的数据。该协议目的在于控制多个数据发送连接，为选择发送通道，如UDP、多播UDP与TCP提供途径，并为选择基于RTP上发送机制提供方法。

RTSP是一个串流协议，满足多个串流的需求，可降低服务端的网络用量并且支持多方视讯会议。未定传输使用的网络通讯所以服务端可

## RTSP 和RTP的关系

RTP是实时传输协议，不能像http和ftp一样可以下载完整的影响文件，只能按照固定的速率在网络上发送数据，客户端也是按照这个速率观看，不可重复播放。

RTSP与RTP最大的区别：RTSP是一种双向数据传输协议，允许客户端向服务端发送快退、快进、回放灯操作，类似于http协议的网络层协议，可基于RTP来传输数据采用TCP、UDP、UDP组播等通道发送数据。

## RTSP和HTTP的比较

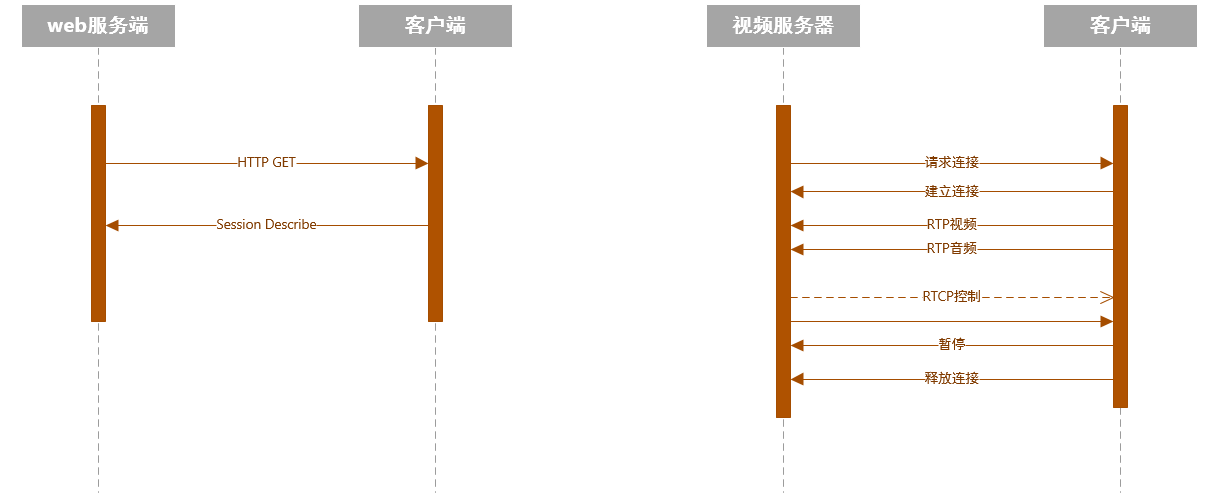
相同点：提供的服务相同

不同点：

1、RTSP使用音视频流形式，HTTP使用文本和图片形式

2、RTSP服务器需要维持会话状态，将请求和流关联起来

3、HTTP不是对称的（客户端发出请求，服务器响应)，但在RTSP协议中客户端和服务器都可以发出请求



## **SDP：**会话描述协议（SDP:Session Description Protocol）

## **RTMP/RTMPS：**RTMP实时消息传送协议(Real Time Messaging Protocol)

## MMS： 微软媒体服务器协议(Microsoft Media Server Protocol)

## **HLS：**HTTP的流媒体传输协议（ Live Streaming）

## http-flv、rtmp和hls直播的优缺点:



参考资料

[从零开始写一个RTSP服务器（一）RTSP协议讲解](https://blog.csdn.net/weixin_42462202/article/details/98986535?utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-13.control&dist_request_id=&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromMachineLearnPai2%7Edefault-13.control)

## 结构组成

ZigBee协议栈建立在IEEE 802．15 4的PHY层和MAC子层规范之上。它实现了网络层(networklayer，NWK)和应用层(applicationlayer，APL)。在应用层内提供了应用支持子层(application support sub—layer，APS)和ZigBee设备对象(ZigBee Device Object，ZDO)。应用框架中则加入了用户自定义的应用对象

## 层的概念

ZigBee 的体系结构由称为层的各模块组成。每一层为其上层提供特定的服务：即由数据服务实体提供数据传输服务；管理实体提供所有的其他管理服务。每个服务实体通过相应的服务接入点(SAP)为其上层提供一个接口，每个服务接入点通过服务原语来完成所对应的功能。 ZigBee 协议的体系结构如图2-1所示

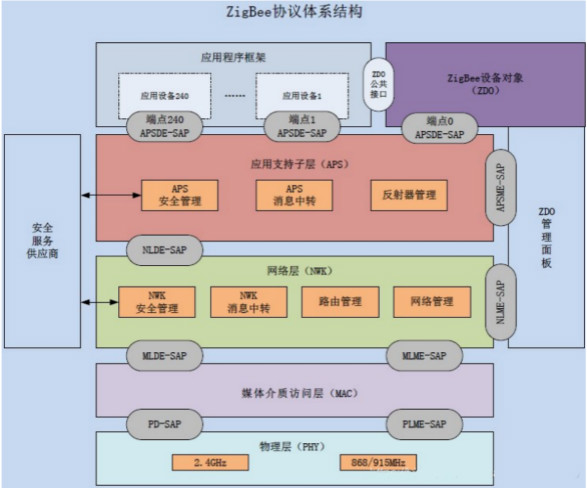


图 2-1

### 物理层(PHY)

物理层定义了物理无线信道和 MAC 子层之间的接口，提供物理层数据服务和  
物理层管理服务。

* **物理层内容**1)ZigBee 的激活；  
  2)当前信道的能量检测；  
  3)接收链路服务质量信息；  
  4)ZigBee 信道接入方式；

5)信道频率选择；

6)数据传输和接收。

### 介质接入控制子层（MAC）

MAC层负责处理所有的物理无线信道访问，并产生网络信号、同步信号；支持PAN连接和分离，提供两个对等MAC实体之间可靠的链路。

* **MAC层功能**

1）网络协调器产生信标；  
2）与信标同步；  
3）支持 PAN(个域网)链路的建立和断开；  
4）为设备的安全性提供支持；  
5）信道接入方式采用免冲突载波检测多址接入(CSMA-CA)机制；  
6）处理和维护保护时隙(GTS)机制；  
7）在两个对等的 MAC 实体之间提供一个可靠的通信链路。

### 网络层（NWK）

ZigBee协议栈的核心部分在网络层。网络层主要实现节点加入或离开网络、接收或抛弃其他节点、路由查找及传送数据等功能。

* **网络层功能：**

1)网络发现；  
2)网络形成；  
3)允许设备连接；  
4)路由器初始化；  
5)设备同网络连接；  
6)直接将设备同网络连接；  
7)断开网络连接；  
8)重新复位设备；  
9)接收机同步；  
10)信息库维护。

### 应用层（APL）

ZigBee应用层框架包括应用支持层(APS)、 ZigBee设备对象(ZDO)和制造商所定义的应用对象。ZigBee应用层除了提供一些必要函数以及为网络层提供合适的服务接口外，一个重要的功能是应用者可在这层定义自己的应用对象。

* **应用支持层的功能**

维持绑定表、在绑定的设备之间传送消息。

* **ZigBee设备对象的功能**

定义设备在网络中的角色(如ZigBee协调器和终端设备)，发起和响应绑定请求，在网络设备之间建立安全机制。ZigBee设备对象还负责发现网络中的设备，并且决定向他们提供何种应用服务。

### 应用程序框架（AF）

运行在ZigBee协议栈上的应用程序实际上就是厂商自定义的应用对象，并且遵循规范（profile）运行在端点1~ 240上。在ZigBee应用中，提供2种标准服务类型：键值对（KVP）  
或报文（MSG）。

* **设备对象（ZDO）**ZigBee设备对象(ZDO)的功能包括负责定义网络中设备的角色，如：协调器或者终端设备。还包括对绑定请求的初始化或者响应，在网络设备之间建立安全联系等。实现这些功能，ZDO使用APS层的APSDE-SAP和网络层的NLME-SAP。 ZDO是特殊的应用对象，它在端点(entire)0上实现。远程设备通过ZDO请求描述符信息，接收到这些请求时，ZDO会调用配臵对象获取相应描述符值。

# **三．ZigBee设备类型和网络节点类型**

## 设备类型

**ZigBee网络支持两种功能类型的网络节点**：全功能器件(FullFunctionDevice,FFD)和精简功能器件(ReduceFunctionDevice，RFD)。全功能器件拥有完整的协议功能，在网络中可以作为协调器(Coordinator)，路由器(Router)和普通节点(Device)；而精简功能器件旨在实现最简单的协议功能而设计，只能作为普通节点存在于网络中。全功能器件可以与精简功能器件或其他的全功能器件通信，而精简功能器件只能与全功能器件通信，精简功能器件之间不能直接通信。

## ZigBee网络节点类型

在ZigBee网络中存在三种节点类型：Coordinator(协调器)，Router(路由器)和End-Device(终端设备)。 ZigBee网络由一个Coordinator以及多个Router和多个End\_Device组成。

* **Coordinator(协调器)**

协调器负责启动整个网络。它也是网络的第一个设备。协调器选择一个信道和一个网络ID(也称之为PAN ID，即Personal Area Network ID)，随后启动整个网络。协调器也可以用来协助建立网络中安全层和应用层的绑定(bindings)。

注意，协调器的角色主要涉及网络的启动和配置。一旦这些都完成后，协调器的工作就像一个路由器(或者消失go away)。由于ZigBee网络本身的分布特性，因此接下来整个网络的操作就不在依赖协调器是否存在。

* **Router(路由器)**

路由器的功能主要是：允许其他设备加入网络，多跳路由和协助它自己的由电池供电的终端设备的通讯。通常，路由器希望是一直处于活动状态，因此它必须使用主电源供电。但是当使用树状网络拓扑结构时，允许路由间隔一定的周期操作一次，这样就可以使用电池给其供电。

* **End-Device(终端设备)**

终端设备没有特定的维持网络结构的责任，它可以睡眠或者唤醒，因此它可以是一个电池供电设备。通常，终端设备对存储空间(特别是RAM的需要)比较小。

# **四．ZigBee协议栈规范（Stack Profile）**

协议栈规范由ZigBee联盟定义指定。在同一个网络中的设备必须符合同一个协议栈规范（同一个网络中所有设备的协议栈规范必须一致）。  
ZigBee联盟为ZigBee协议栈2007定义了2个规范：ZigBee和ZigBee PRO。所有的设备只要遵循该规范，即使在不同厂商买的不同设备同样可以形成网络。  
 如果应用开发者改变了规范，设备只有在自己的产品中使用，不能与其他产品通信。称之为“特定网络”规范。协议栈规范的 ID 号可以通过查询设备发送的 beacon 帧获得。在设备加入网络之前，首先需要确认协议栈规范的 ID。

“特定网络” 规范 ID 号为 0；ZigBee 协议栈规范的 ID 号为 1；ZigBee PRO 协议栈规范的ID 号为 2。协议栈规范的 ID（STACK\_PROFILE\_ID）在nwk\_globals.h 中定义：

#define NETWORK\_SPECIFIC 0  
#define HOME\_CONTROLS 1  
#define ZIGBEEPRO\_PROFILE 2  
#define GENERIC\_STAR 3  
#define GENERIC\_TREE 4  
#if defined ( ZIGBEEPRO )  
#define STACK\_PROFILE\_ID ZIGBEEPRO\_PROFILE  
#else  
#define STACK\_PROFILE\_ID HOME\_CONTROLS  
#endif

# **五．ZigBee拓扑结构**

 zigbee的拓扑结构可以典型的分为三种类型，星形(Star), 树形(Tree Topology) 还有网状型(Mesh Topology).

## 星形(Star)

星状网络由一个PAN 协调器和多个终端设备组成，只存在PAN 协调器与终端的通讯，终端设备间的通讯都需通过PAN 协调器的转发。该拓扑结构的最大特点就是任意两个节点的通信都需要依赖协调器的辅助转发才能完成通信，即便是两个节点十分靠近。如图 5-1

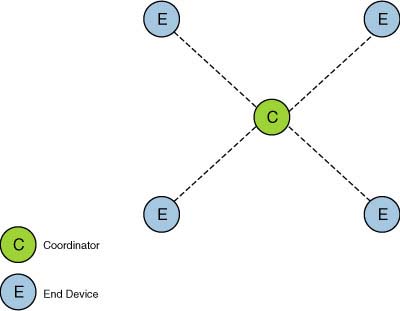


图 5-1

## 树形(Tree Topology)

树状网络由一个协调器和一个或多个星状结构连接而成，设备除了能与自己的父节点或子节点进行点对点直接通讯外，其他只能通过树状路由完成消息传输。如图 5-2

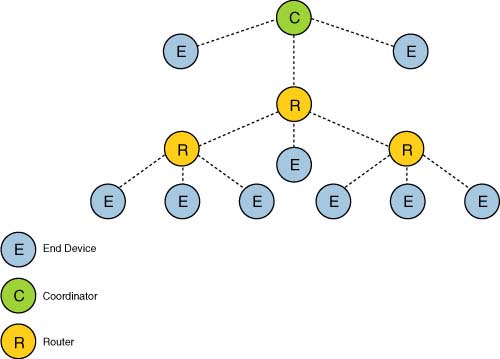


图 5-2

## 网状型(Mesh Topology)

网状网络是树状网络基础上实现的，与树状网络不同的是，它允许网络中所有具有路由功能的节点直接互连，由路由器中的路由表实现消息的网状路由。即Mesh结构的最大特点在于，路由器之间是可以相互通信的，而不需要通过她们的父节点才可以通信。如图 5-3。该拓扑的优点是减少了消息延时，增强了可靠性，缺点是需要更多的存储空间开销。

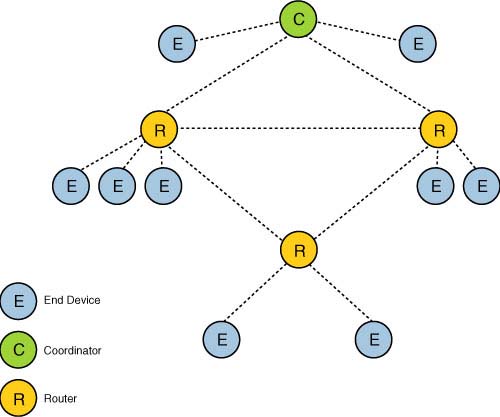


图 5-3

# **六．ZigBee网络通讯方式**

Zigbee网络中进行数据通信有三种主要的类型：单播、组播、广播。如果与TCP/IP对比，ZigBee通讯当然也需要地址，所以传递数据是目的，那么其过程就是寻址。所以先了解一下ZigBee地址的概念。

## 地址

* 地址定义

ZigBee设备有两种类型的地址。一种是64位IEEE地址，即MAC地址，另一种是16位网络地址。

* 网络地址分配

使用分布式寻址方案来分配网络地址，保证在整个网络中所有分配的地址是唯一的。保证一个特定的数据包能够发给它指定的设备，而不出现混乱。寻址算法本身的分布特性保证设备只能与他的父辈设备通讯来接受一个网络地址。不需要整个网络范围内通讯的地址分配，这有助于网络的可测量性

## 寻址（**Addressing**）

### AF\_DataRequest函数

为了向一个在ZigBee网络中的设备发送数据，应用程序通常使用AF\_DataRequest()函数。  
数据包将要发送给一个afAddrType\_t(在ZComDef.h中定义)类型的目标设备。

**afStatus\_t** AF\_DataRequest**(**afAddrType\_t **\***dstAddr**, //目的地址指针**

endPointDesc\_t **\***srcEP**, //发送节点的端点描述符指针**uint16cID**,    //ClusID 簇ID号**

uint16len**,   //发送数据的长度**

uint8 **\***buf**,   //指向存放发送数据的缓冲区指针**

uint8 **\***transID**,**

uint8options**, //发送选项**

uint8radius**//最大传输半径（发送的跳数）**

**)**

**参数1：**afAddrType\_t \*dstAddr 该参数包含了目的节点的网络地址、端点号及数据传送的模式，如单播、广播或多播等。

     afAddrType\_t 是个结构体如下：

    typedef struct  
   {  
      union  
     {  
       uint16 shortAddr;      //用于标识该节点网络地址的变量  
     } addr;  
    afAddrMode\_t addrMode;   //用于指定数据传送模式，单播、多播还是广播  
    byte endPoint;           //端点号  
   } afAddrType\_t;  **// 其定义在AF.h中**

  在Zigbee 中，数据包可以单点传送(unicast)，多点传送(multicast)或者广播传送，所以必须有地址模式参数。一个单点传送数据包只发送给一个设备，多点传送数据包则要传送给一组设备，而广播数据包则要发送给整个网络的所有节点。因此上述结构体中的afAddrMode\_t addrMode 就是用于指定数据传送模式，是个枚举类型，可以设置为以下几个值，

typedef enum  
{  
  afAddrNotPresent = AddrNotPresent,   //表示通过绑定关系指定目的地址  
  afAddr16Bit = Addr16Bit,   //单播发送  
  afAddrGroup = AddrGroup,   //组播  
  afAddrBroadcast = AddrBroadcast //广播  
} afAddrMode\_t;

enum  
{  
  AddrNotPresent = 0,  
  AddrGroup = 1,  
  Addr16Bit = 2,  
  Addr64Bit = 3,  
  AddrBroadcast = 15  
};

### 单播

注意：其实单播有两种方式一种是绑定传输my\_DstAddr.addrMode=(afAddrMode\_t) AddrNotPresent，一种是直接指定目标地址的单播传输，比如协调器就是0x0000。

* **单播绑定传输**

my\_DstAddr.addrMode=(afAddrMode\_t)Addr16Bit;   //单播发送  
   my\_DstAddr.endPoint=GENERICAPP\_ENDPOINT;       //目的端口号  
   my\_DstAddr.addr.shortAddr=0;   //按照绑定的方式进行单播，不需要指定目标地址，需要先将两个设备 绑定，将两个设备绑定后即可通信

* **直接指定目标地址的单播传输：**

是标准寻址模式，它将数据包发送给一个已经知道网络地址的网络设备，将afAddrMode 设置为Addr16Bit 并且在数据包中携带目标设备地址。  
  my\_DstAddr.addrMode=(afAddrMode\_t)Addr16Bit;   //单播发送  
  my\_DstAddr.endPoint=GENERICAPP\_ENDPOINT;       //目的端口号  
  my\_DstAddr.addr.shortAddr=0x0000;              //目标设备网络地址

### 广播

当应用程序需要将数据包发送给网络的每一个设备时，使用这种模式。地址模式设置为  
AddrBroadcast。目标地址 my\_DstAddr.addr.shortAddr可以根据需求设置为下面广播地址的一种：

NWK\_BROADCAST\_SHORTADDR\_DEVALL(0xFFFF)——数据包将被传送到网络上的所有设备，包括睡眠中的设备。对于睡眠中的设备，数据包将被保留在其父亲节点直到查询到它，或者消息超时(NWK\_INDIRECT\_MSG\_TIMEOUT 在f8wConifg.cfg 中)。

 NWK\_BROADCAST\_SHORTADDR\_DEVRXON(0xFFFD)——数据包将被传送到网络上的所有在空闲时打开接收的设备(RXONWHENIDLE)，也就是说，除了睡眠中的所有设备。

  NWK\_BROADCAST\_SHORTADDR\_DEVZCZR(0xFFFC)——数据包发送给所有的路由器，包括协调器。

   my\_DstAddr.addrMode=(afAddrMode\_t)AddrBroadcast;//广播发送  
  my\_DstAddr.endPoint=GENERICAPP\_ENDPOINT;       //目的端口号  
  my\_DstAddr.addr.shortAddr=0xFFFF;              //协调器网络地址

### 组播

当应用程序需要将数据包发送给网络上的一组设备时，使用该模式。地址模式设置为  
afAddrGroup 并且addr.shortAddr 设置为组ID。使用组播的方式需要加入特定的组。

* **1.首先声明一个组对象aps\_Group\_t SampleApp\_Group;**
* **2.对SampleApp\_Group赋值：**
* **3.在本任务里将端点加入到组中：**
* **4.设定通信的目标地址及模式：**
* **5.若要把一个设备加入到组中的端点从组中移除，调用aps\_RemoveGroup：**

**注意**：组可以用来关联间接寻址。再绑定表中找到的目标地址可能是是单点传送或者是一个组地址。另外，广播发送可以看做是一个组寻址的特例。

关于Zigbee通讯更多了解

（<https://blog.csdn.net/xiao11xiang123/article/details/26497453>）

（<https://blog.csdn.net/aaa1163548340/article/details/87871218>）（<https://blog.csdn.net/zgrjkflmkyc/article/details/8892296>）

### 重要设备地址(Important Device Addresses)

应用程序可能需要知道它的设备地址和父亲地址。

* 使用下面的函数获取设备地址(在ZStackAPI中定义)：  
  NLME\_GetShortAddr()——返回本设备的16位网络地址  
  NLME\_GetExtAddr()—— 返回本设备的64位扩展地址
* 使用下面的函数获取该设备的父亲设备的地址：  
  NLME\_GetCoordShortAddr()——返回本设备的父亲设备的16位网络地址  
  NLME\_GetCoordExtAddr()—— 返回本设备的父亲设备的64位扩展地址

# **七．ZigBee-Z-Stack协议栈（入网组网的理解）**

## ZigBee组网过程

### 建立信道

通过底层能量扫描，选择一个合适的信道建立

### 广播信标(Beacon)

协调器或者路由器不断地广播信标

### 终端设备请求入网

终端设备（End\_Device）或路由器发现信标并且请求入网

### 协调器或者路由器分配短地址

协调器会为入网的终端设备或者路由设备分配一个短地址

### 5. 入网成功

# **八．ZigBee术语**

以上内容已经有涉及一些术语概念，本章重新来总结一下一些术语

### 1．Attribute（属性）

属性Attribute是一个反映物理数量或状态的数据值，比如开关值(On/Off)、灯的状态值(On/Off)、温度值、百分比等等

### 2．Cluster（集群，簇）

群集Cluster是包含一个或多个属性(attribute)的群集。简单的说，群集就是属性的集合。每个群集都被分配一个唯一群集ID且每个群集最多有65536个属性。比如一个群集包含了不同情况下的开关、不同情况下的灯、不同情况下的温度值、不同情况下的百分比等等

### 3．Device Description（设备描述）

设备描述Device Description是指一个大型目标应用的一部分，包括一个或多个群集。描述符有：节点描述符、电源描述符、简单描述符、端点描述符。

端点描述符： typedef struct  
{  
  byte endPoint;  
  byte \*task\_id; // Pointer to location of the  
  Application task ID.  
  SimpleDescriptionFormat\_t

  \*simpleDesc;  
  afNetworkLatencyReq\_t latencyReq;  
} endPointDesc\_t;

简单描述符：  
typedef struct  
{  
  byte EndPoint;  
  uint16 AppProfId;  
  uint16 AppDeviceId;

  byte AppDevVer:4;  
  byte Reserved:4; // AF\_V1\_SUPPORT uses for AppFlags:4.  
  byte AppNumInClusters;  
  cId\_t \*pAppInClusterList;  
  byte AppNumOutClusters;  
  cId\_t \*pAppOutClust  
} SimpleDescriptionFormat\_t;

### 4．Endpoint（端点）

端点EndPoint是协议栈应用层的入口，即入口地址，也可以理解应用对象(Application Object)存在的地方，它是为实现一个设备描述而定义的一组群集。每个Zigbee设备可以最多支持240这样的端点，端口0用于整个Zigbee设备的配置和管理，应用程序可以通过端点0与Zigbee堆栈的其他层通信，从而实现对这些层的初始化和配置。附属在端点0的对象被称为Zigbee设备对象(ZDO)。端点255用于向所有端点的广播，端点241~254是保留端点。

### 5．Coordinator、Router、EndDevice 第三章

### 6． 通信（单播、广播） 第六章

### 7．ZigBee Home Automation profile 0x0104

**profile**：可以理解成为了实现/规范化某个功能或应用而定义的一套规定。每个应用对应一个profile ID。分为公共profile和私有profile。如果对Bluetooth有一定的基础，可以对比Bluetooth 的profile进行理解，概念基本上互通。

**cluster**：一个profile包含多个cluster。可以这样理解，一个具体的profile，包含了很多特性对象及细节上的小规范，这个规范即称为cluster。每个cluster有对应的cluster ID。每个cluster ID都是唯一的。

              不同的cluster有不同的命令（command），除过命令，cluster还有对应的属性（attribute）。命令和属性都有对应的ID，每个cluster下，命令和属性的ID必须是唯一的。

              而不同的cluster下，命令和属性ID可以重复。不同的profile下，cluster ID也可以重复。

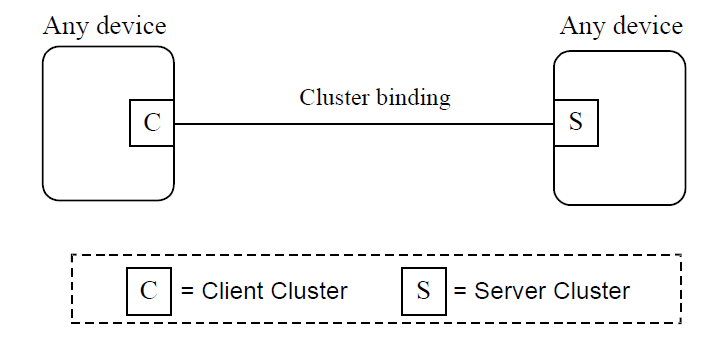
              例如：一个智能家居profile下的一个调光器，就是可以是一个cluster。而操作这个调光器，就需要很多命令，比如开、关、调大、调小等，这些都是command。

              而一个调光器，都有一些当前的状态和参数，比如当前是开状态，从开到关需要多长时间等，这些统一起来为属性。

**ZHA**：ZigBee Home Automation profile。 智能家居自动化profile，即规定了智能家居要做什么，其中的设备要遵守的规范协议，有了规范的标准化协议，各个厂家的智能设备才能协同工作，不同厂家的产品才能兼容使用。

### 8．ZCL: ZigBee Cluster Library

可以理解为Zigbee在开发一些特性功能的cluster时所用的一些库。开发者在开发应用profile时需要用到对应ZCL API，去操作或控制对应的cluster。



### 9．Channel

Zigbee在3个频段定义了27个物理信道：868MHz频段中定义了1个**20Kb/s**信道，915MHz频段中定义了10个**40Kb/s**信道，信道间隔为2MHz，2.4GHz频段上定义了16个**250Kb/s**信道，信道间隔为5MHz.

Z-stack中可以在f8wConfig.cfg里设置信道。当建网过程开始后，网络层将请求MAC层对规定的信道或由物理层默认的有效信道进行能量检测扫描，以检测可能的干扰。网络层管理实体对能量扫描的结果以递增的方式排序，丢弃那些能量值超出可允许能量水平的信道，然后再由网络层管理实体执行一次主动扫描，结合检查PAN描述符，对剩下的信道选择一个合适的建立网络。

**-DDEFAULT\_CHANLIST=0x00000800 // 11 - 0x0B    默认使用的是编号为11的信道**

若要在应用中查看信道，可以这样获得，\_NIB.nwkLogicalChannel，读取这个就OK了。

### 10．PanId

在确定信道以后，下一步将是确定PANID，如果ZDAPP\_CONFIG\_PAN\_ID被定义为0xFFFF，那么协调器将根据自身的IEEE地址建立一个随机的PANID（0～0x3FFF），如ZDAPP\_CONFIG\_PAN\_ID没有被定义为0xFFFF，那么网络的PANID将由ZDAPP\_CONFIG\_PAN\_ID确定。

“如果ZDAPP\_CONFIG\_PAN\_ID被定义为0xFFFF，那么协调器将根据自身的IEEE地址建立一个随机的PANID（0～0x3FFF）”这句话怎么理解呢，我经过试验发现，这个随机的PANID并非完全随机，它有规律，与IEEE地址有一定的关系：要么就是IEEE地址的低16位，要么就是一个与IEEE地址低16位非常相似的值。如IEEE地址为0x8877665544332211，PANID很有可能就是2211，或相似的值；IEEE地址为0x8877665544337777，PANID很有可能就是3777，或其它相似的值；

-DZDAPP\_CONFIG\_PAN\_ID=0xFFFF

若要在应用中查看PANID可以这样获得，\_NIB.nwkPanId，读取这个就OK了。

# **九．谁能称霸物联网时代**

根据IDC市场研究机构报告显示，到2020年全球将有260亿件设备接入物联网，全球物联网产业规模将达到1．7万亿美元。

物联网悄然拉开了“万物互联”的帷幕。在“万物互联”时代，首先要解决“物”与“网”的连接问题，物联网生态系统的核心组成部分是连接和通信，使得远程和分散物体（设备，传感器节点，网关单元，云服务器）之间的连接和通信成为可能，所以“连接技术”决定了物联网发展的走向。

物联网无线接入技术种类众多，包括Zigbee、WiFi、蓝牙等短距离通信技术和LoRa、SigFox、eMTC、NB－IoT等长距无线通信技术。其中，受业界青睐的低功耗广域技术LPWA既包括广域非授权频谱技术LoRa和SigFox，也包括授权频谱的eMTC和NB－IoT等。

## 远距离无线通信

这里罗列了两种远距离通信技术NB-IOT和LoRa与ZigBee的比较



**注意：**更多请看<https://blog.csdn.net/ymj321/article/details/103022353>

## 短距离无线通信

几种短距离通信技术与ZigBee的比较

