< 2018年3月 >

日	一	二	三	四	五	六
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

公告

昵称：bw_0927
园龄：7年6个月
粉丝：61
关注：2
+加关注

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

随笔分类

ATS

随笔档案

2015年10月 (2)
2015年9月 (1)
2015年4月 (1)
2015年3月 (1)
2015年2月 (1)
2011年12月 (1)
2011年7月 (1)
2011年2月 (1)

文章分类

APP(2)
bootstrap(3)
C++11(27)
CDN(11)
Citrix / Virtualization(21)
curl(6)
django(14)
GIT(26)
html/xml/dhtml/CSS(12)
HTML5(4)
HTTP TCP TLS wireshark(18)
java(14)
javascrpit(38)
jquery(8)
kernel(3)
ldap(6)
linux(68)
linux lib TCP/IP(26)

TS流解析【PCR】自己的总结
Posted on 2017-01-18 18:37 bw_0927 阅读(224) 评论(0) 编辑 收藏

http://www.cnblogs.com/ztteng/articles/3166025.html

http://blog.csdn.net/liuhongxiangm/article/details/8981032

http://blog.sina.com.cn/s/blog_6b94d5680101ton7.html

http://blog.csdn.net/jl2011/article/details/47044647

二.TS流包含的内容

一段TS流，必须包含PAT包、PMT包、多个音频包、多个视频包、多个PCR包、以及其他信息包PSI。

解析TS流数据的流程：查找PID为0x0的包，解析PAT，PAT包中的program_map_PID表示PMT的PID；查找PMT，PMT包中的elementary示音视频包的PID，PMT包中的PCR_PID表示PCR的PID，**有的时候PCR的频或者视频的PID相同，说明PCR会融进音视频的包，注意解析，有的时候自己单独的包；CAT、NIT、SDT、EIT的PID分别为: 0x01、0x10、0x12**。

PSI被分为4个表结构，他们应被进一步划分为各个段（SECTION）并插入TS分组中，一些带有预先规定的PID，另一些带有用户可选的PID。

TS包中净荷所承载的信息包括以下3种:

- **1、视频/音频的PES包以及辅助数据**
- **2、描述单路节目复用信息的节目映射表（PMT）**
- **3、描述单路节目复用信息的节目关联表（PAT）**

- （1）系统复用时，对视频和音频的ES流进行打包，形成视频和音频的PES流，**辅助数据不需要打成PES包**.
- （2）视频和音频的PES包以**一帧编码图像为单位**，音频PES包恒定长度，视频PES包长度可变。
- （3）PES包的长度通常都是远大于TS包的长度，一个PES包必须由整数个TS包来传送，没装满的TS包由填充字节填充。
- （4）TS包长度固定，188字节，有效净荷184字节。

一个PMT表包含了与**单路节目**复用有关的节目信息，典型的构成包括1路视频ES流，2-5路音频ES流，1路或多路辅助数据。

- 进行TS流复用时，**各路ES流被分配了唯一的PID,**ES流与被分配的PID值间的关系构成了一张表，称为节目映射表PMT。
- PMT完整描述了**一路**节目由哪些ES流组成，他们的PID分别是什么。
- MPEG-2传送层中，**传送PMT表的码流称为控制码流**，和其他ES流一样，在TS包的净荷中传送，分配唯一的PID.
- PAT包含了与多路节目复用有关的控制信息。
- PAT描述了系统级复用中传送每路节目PMT的码流的PID。
- PAT作为一个独立的码流，装载在TS包的净荷中传送，分配唯一的PID。传送PAT的码流的PID值定义为固定的数值"0"。
- **若复用时(mux时)遇到有不同码流的PID值相同，则必须把它记录下来，记录在PAT和PMT中。**
- 允许单路数字电视节目可由其中某些节目流(例如视频，不同音频，不同字幕)任意组合构成，节目可根据需要ES码流进行增加或删除。
- 允许对多路节目进行灵活复用，若其中某些节目流发生变化，只需要将PAT和PMT做相应修改即可。
- 能够在TS级上提供本地节目插入和条件接收等对广播界非常重要的功能。

linux programing(76)
linux shell(29)
linux程序分析(35)
linux网络编程(29)
makefile/gcc(14)
MMORPG(3)
nginx(18)
perl学习笔记(3)
pjsip/opensips/kamailio(2)
python(79)
sip概念(6)
stock(3)
UML(2)
VB/ActiveX(10)
VI(22)
VMware(1)
web / websocket(47)
windows 网络编程整理_转载(5)
windows编程(10)
windows常见问题(5)
ZeroMQ(4)
安全，SSL(41)
分布式(17)
服务器开发与架构(75)
概念(5)
健身(1)
流媒体(58)
路由器(8)
牛掰(5)
设计模式(38)
数据结构(28)
数据库(29)
网络(42)
硬件(3)
语言常识(17)
云技术、虚拟化(21)
中医(7)

猪八戒

最新评论

1. Re:高性能服务器架构思路
序列化 主要是为了双方都认识这个数据，是 统一语言。
--岁月漫步

2. Re:with as 异常处理
楼主，我在你的博客中看到引用了我的文章，我想说一下，原来那个博客被墙了，我就启用了，现在那篇文章在CSDN上。
--黑翼天使23

3. Re:App后台开发运维和架构实践： 15.app后端怎么设计用户登录方案
短信登录的成本还是太高了。。。
--CEMaster

4. Re:rtsp rtmp http 直播 点播
国内做的真正好用的，可以看看这个项目
--曾经在奋斗

5. Re:js 事件捕获 事件冒泡
好文章，简明易懂!
--biyesheng

阅读排行榜

1. 服务器开发知识要点(661)
2. c中printf必须在所有的变量申明之后才能用? (652)
3. 找到关注点(221)
4. Who is using Asio?(216)
5. 牛人博客(214)

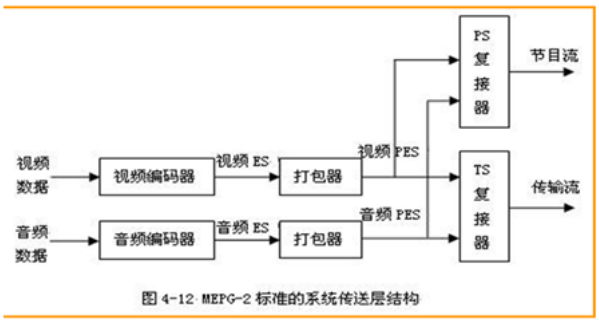
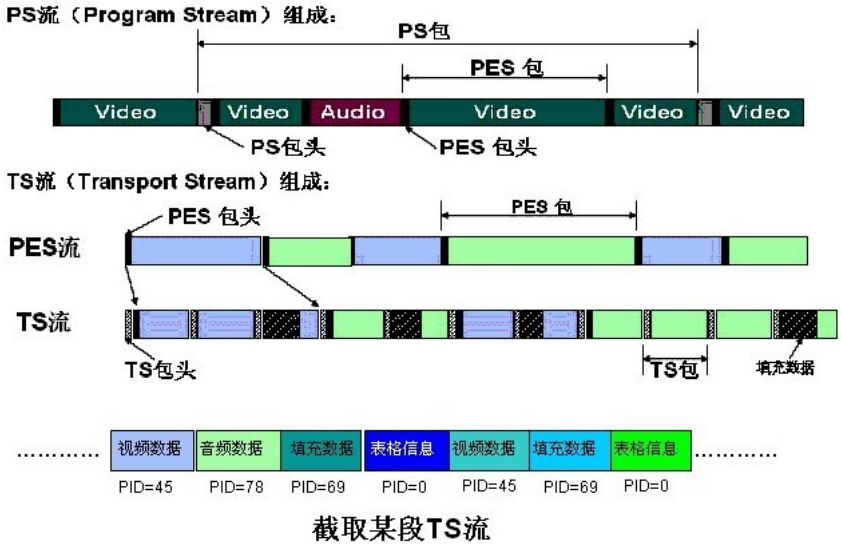


图 4-12 MEPG-2 标准的系统传送层结构



PES包格式：

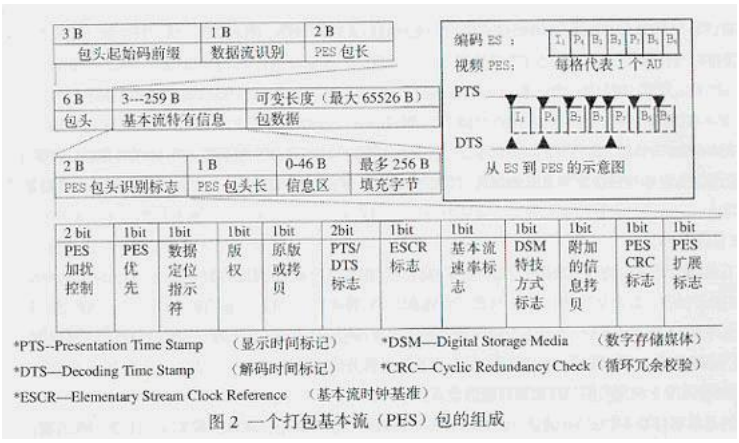


图 2 一个打包基本流（PES）包的组成

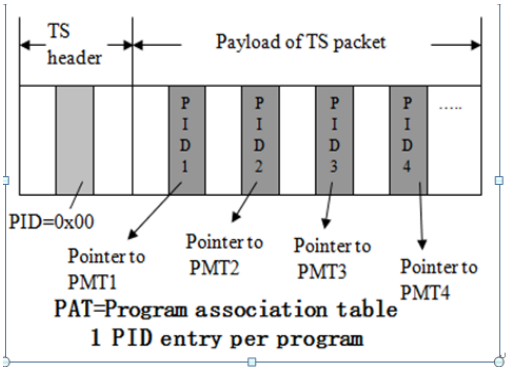
PES再打包成TS流或PS流，往往一个PES会分存到多个ts包中

每个ES都由若干个**存取单元（AU）**组成，每个视频AU或音频AU都是由头部和编码数据两部分组成，1个AU相当于编码的1幅视频图像或1个音频帧，实际上是编码数据流的显示单元，即相当于解码的1幅视频图像或1个音频帧的取样。

简而言之：**一个AU对应一个帧，一个ES由多个帧组成**

- **PAT**

- **每个TS流一个，每隔0.5秒重复。**
- **描述TS流中有多少个节目。**
- **包含该表的TS包的PID为0，便于识别。**
- **PAT的payload中传送特殊PID的列表，每个PID对应一个节目。**
- **这些PID是描述每个独立节目详细信息的指针。**
- **PID指向PMT表。**



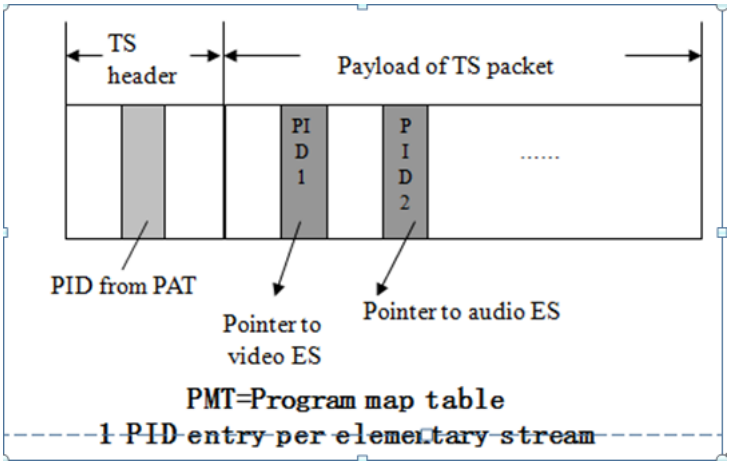
- **PMT**

– **对应TS包有特殊的PID和特殊的payload。**

– **PMT的PID由PAT传送。**

– **例如要接收节目3时，先从PAT的payload中的所有PID列表中选出节目3的PID为1FF3hex，然后查找包头中PID=1FF3hex的TS包，就是**

– **PMT包含该节目中所有ES流（视频、音频或数据）的PID。**



一个节目可能有多个视频和音频流，解码器必须选择2个PID，一个视频流的PID（100hex），一个音频流的PID（200hex）。

此后解码器只收集这些TS包，**解复用，重新组成PES包，这些PES包再送到视频或音频解码器。**

传输过程中TS流的结构也可能发生改变。解码端机顶盒，如DVB-S，必须连续检测TS流瞬时结构，读出PAT和PMT，做自适应调整。

- PAT和PMT读出以后，用户确定出一个节目的两个PID：

  待解码视频信号的PID（如100hex）

  待解码音频信号的PID（如200hex）

解码器只处理这两个PID的TS包：

解复用过程中，PID为100hex的所有TS包集合成视频PES包，送到视频解码器。

同样，PID为200hex的所有TS包重新集合成音频PES包，送到音频解码器。

如果ES流没有加扰，这时可以直接解码。

对付费电视或许可证和地域限制等情况，ES流利用电子码进行传输保护。

– ES流利用各种方法进行混扰，接收端必须配有附加硬件并授权。

– 附加硬件必须有TS流中合适的解扰和授权数据。

– 因此TS流中传送一个特殊的表CAT(conditionalaccess table)

- **CAT提供了TS流其他数据包的PID，该数据包传送了解扰所需信息：**

– ECM(entitlement control message)　用于传送加扰码

– EMM(entitlement management message)　用于用户管理

- 只有ES流本身可以加扰，TS包头、表格和**adaptation field**不能加扰。
- 解扰本身在MPEG解码器以外的附加硬件设备进行，附加硬件与解扰方法相关，可以做成智能板卡通过CI(common interface)插入机顶盒。
- 在MPEG解码器做进一步处理之前，TS流在该硬件设备中循环。
- ECM和EMM的信息，以及用户的个人码可以将码流解扰。

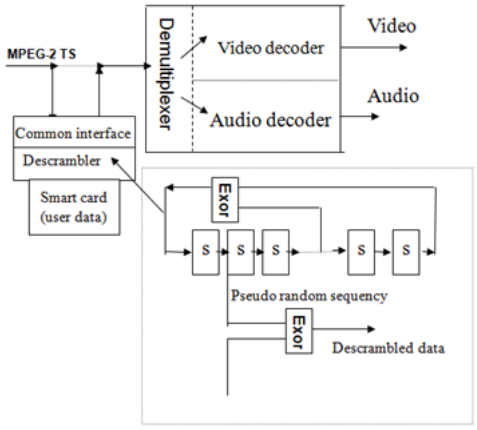


图3. 16 解码器的解扰

- 亮度信号采样频率13.5MHz，色度信号6.75MHz。27MHz是采样频率的倍数，作为发送端MPEG编码器所有处理过程的参考或基本频率。
- 编码器中27MHz振荡器作为系统时钟(STC)的输入。
- STC是42bit计数器，由27MHz时钟计数，溢出后重新从0开始。
- 接收端也必须提供STC，其27MHz振荡器和42bit计数器必须与编码器STC完全同步。

MPEG码流中需传送参考信息——**PCR(program clock reference)，即在固定时刻将最新的STC计数器值复制到TS流中**

- **码流中传送的PCR值必须足够多，有最大间隔的限制；而且要相对准确，没有抖动。**

MPEG标准规定：

- **每个节目PCR的最大间隔为40ms。**
- **PCR的抖动小于±500ns。**

- PCR如果出错：
- 本来应该显示彩色图像，却显示出黑白图像。
- TS流重复用时会出现抖动，因为TS包顺序改变，但其中PCR信息却没变。经常会有最大±30μs的PCR抖动，该问题许多机顶盒可以解决。
- **PCR信息在相应节目TS包的adaptation field中传送**，而TS包类型的准确信息可以从PMT中获得。
- 节目时钟同步以后，视音频编码就可以锁定系统时钟进行了。
- 欧洲DVB项目组和美国ATSC项目组都定义了数字视音频节目传输的附加信息，以便简化机顶盒操作，使其更加人性化：
- 在TS流中传送节目名称来分辨不同节目；
- MPEG-2为扩展留有空间，在PSI、PMT和CAT之外，**TS流中还可以有private tables，定义了用户表的结构以及如何将用户表插入到TS流中。**

TS包头定义：

```
typedef struct TS_packet_header
{
    unsigned sync_byte                  : 8; //同步字节, 固定为0x47,表示后面的是一个TS分组
    unsigned transport_error_indicator      : 1; //传输误码指示符
    unsigned payload_unit_start_indicator   : 1; //有效荷载单元起始指示符

    unsigned transport_priority         : 1; //传输优先, 1表示高优先级,传输机制可能用到，解码用不着
    unsigned PID                        : 13; //PID
    unsigned transport_scrambling_control   : 2; //传输加扰控制
    unsigned adaption_field_control         : 2; //自适应控制 01仅含有效负载，10仅含调整字段，11含有调整字段和有效负载。为00解码器
    unsigned continuity_counter             : 4; //连续计数器 一个4bit的计数器，范围0-15
} TS_packet_header;   //总共32位，4个字节
```

TS包， 很多地方packet译为分组，即传输分组，XX分组

- **sync_byte**

是包中的第一个字节，TS包以固定的8bit的同步字节开始，所有的TS传送包，同步字都是唯一的OX47，用于建立发送端和接收端包的同步。

MPEG-2解码器接收到MPEG-2 TS流时，首先检测包结构，在TS流中查找同步字节：

总是OX47，总位于TS包开始位置，固定间隔为188字节。同时满足这两个条件，可以确定同步。

如果出现一个字节为47hex(OX47)，解码器将检测这个字节前后n倍188字节的位置是否也是同步字节。

如果是，则当前字节为同步字节；否则，当前字节只是码流中偶尔出现的47hex，不是同步字节。

接收端收到5个TS包之后开始同步。丢包3个之后解码器即失步，需重新同步。

- **transport_error_indicator**

用于从解码器向分接器指示传输误码。若这个比特被设置，表示此TS包中所携带的净荷信息有错误，无法使用。

传输错误标志位，一般传输错误的话就不会处理这个包了

- **payload_unit_start_indicator**

有效负载的开始标志
标志PES包头以及包含节目特定信息的表（PMT，PAT）的头是否出现在该包中，在失步后的重新同步中起着重要的作用
一个PES包会被封装在很多小的TS包中，该标识指示是否是一个PES包的第一个TS包，用于重新同步

- **PID**

PID是识别TS包的重要参数，用来识别TS包所承载的数据类型。在TS码流生成时，每一类业务（视频，音频，数据）的基本码流均被赋予一个不同的借助于PID判断某一个TS包属于哪一类业务的基本码流。

//0X00 -> PAT，0x01->CAT, PMT与NIT的TS包的PID在PAT中指定

//0X1FFF->空分组、空包， 0X0002-0X000F也被保留

//ISO/IEC13818-1中定义，通过传输流传送PSI表时，PSI应被划分为一个或多个段（SECTION）后，将SECTION映送，ISO/IEC13818-1 中定义了这种传输段（SECTION）的语法结构，通过这种结构，将PSI的数据填充到传输流中进行传送. 为什么要把PSI来传输了，一次传输不就行了？ 因为每个TS包的数据负载能力是有限的，即每个TS包的长度是有限的，所以当有些PSI表很长很大时，就需要将表拆SECTION语法数据段，再把这钟结构的SECTION填充到TS包中进行传输

【每一个段的长度不一，一个段的开始由TS包的有效负载中的payload_unit_start_indicator来标识】

Table 1: PID allocation for SI

Table	PID value
PAT	0x0000
CAT	0x0001
TSDT	0x0002
reserved	0x0003 to 0x000F
NIT, ST	0x0010
SDT, BAT, ST	0x0011
EIT, ST CIT (TS 102 323 [13])	0x0012
RST, ST	0x0013
TDT, TOT, ST	0x0014
network synchronization	0x0015
RNT (TS 102 323 [13])	0x0016
reserved for future use	0x0017 to 0x001B
inband signalling	0x001C
measurement	0x001D
DIT	0x001E
SIT	0x001F

```
if (adaption_field_control == '10' || adaption_field_control == '11')
{
    adaption_fields() //调整字段的处理
}
if (adaption_field_control == '01' || adaption_field_control == '11')
{
    for(i = 0; i < N ; i++) //N值 = 184 - 调整字段的字节数
    {

    }
}
```

- **transport_scrambling_control**

传送信息通过加入扰码来加密，各个基本码流可以独立进行加扰。加扰控制字段说明TS包中的净荷数据是否加扰。如果加扰，标志出解扰的密匙。

- **adaption_field_control**

```
调整字段控制
    0x0: // reserved for future use by ISO/IEC
    0x1: // 无调整字段，仅含有效负载
    0x2: // 仅含调整字段，无有效负载
    0x3: // 调整字段后含有效负载
```

适配域是一个可变长度的域，它在TS包中是否存在，由适配域控制标识决定。

- **continuity_counter**

用于对传输误码进行检测。在发送端对所有的包都做0-15的循环计数，在接收终端，如发现循环计数器的值有中断，表明数据在传输中有丢失。

==========

TS包头解析：

```
oid adjust_TS_packet_header(TS_packet_header* pheader)
{
    unsigned char buf[4];
    memcpy(buf, pheader, 4);
    pheader->transport_error_indicator      = buf[1] >> 7;
    pheader->payload_unit_start_indicator   = buf[1] >> 6 & 0x01;
    pheader->transport_priority             = buf[1] >> 5 & 0x01;
    pheader->PID                        = (buf[1] & 0x1F) << 8 | buf[2];
    pheader->transport_scrambling_control   = buf[3] >> 6;
    pheader->adaption_field_control         = buf[3] >> 4 & 0x03;
```

```
    pheader->continuity_counter             = buf[3] & 0x03;
}
```

例如要提取结构体中的adaption_field_control，因为它是第26~27位，也就是第3（基于0）字节的第2~3位（共两位），所以要取第3字节，然后右移（去掉多余的2位抛弃），再“与”3（屏蔽左边多余的位）。

==========**PCR**==========

http://dekst.awardspace.com/project/download/PCR.pdf

时间上的同步包括音频和视频上的同步，然而 在传输流（Transport Stream，TS）的传输过程中， 由于网络延迟、复接，以及在适配器（网卡）中进行 兆帧初始化包（ Mega-frame Initialization Packet，MIP）等种种因素，使得附 带音频和视频时间信息的两个相邻 PCR 的相对位 置发生了改变，解出的图像 出现马赛克和唇音不同步等现象，即 PCR 发生了 抖动，因此，在单频网适配器中，需要对接收到 的 PCR 信息进行校正。

编码器中使用一个 27MHz 的系统时钟来产 生一系列时间标签，包括指示音频、视频正确显 示（PTS）和解码的时间(DTS)，以及采样过程中系统时
在 TS 流中描述该系统时钟瞬时值的时 间标签称为节目参考时钟标签（PCR），是编码器 27MHz 系统时钟的 42 比特采样值。
解码器中也 有一个 27MHz 的系统时钟，它根据打包的基本码 流 (Packetized Elementary Stream, PES)中的显 示时间标签 (Presentation Time
解码时间标签 (Decoding Time Stamp, DTS)字段 所指示的时间进行解码和显示。
如果前端编码器 的时钟与后端解码器中时钟“绝对”同步，那么 TS 传输流中的 PCR 就没有任何意义了。
但是如 果“绝对”变为“相对”以后，它们之间的“微 小”误差经过长时间的累积（1~2 小时足以），机 顶盒中解码器就会因为自己的时钟“快”了而造 成（即停帧），或是因为时钟“慢” 了而造成 buffer 中数据溢出（即丢帧）。
所以**解码 器就需要用 TS 流中的 PCR 字段来不断修正自己 与编码器时钟之间的“微小”误差。**
同时 **TS 流经 过适配器时，在原有的 TS 流中插入了用于调整 速率的空包和控制TS流传输的MIP包，因此PCR 必须进行非均匀延迟修正。**

- **PCR的查找和提取**

在 TS 流的传输过程中，并不是每一个 TS 包 （188 个字节）都带有 PCR 信息，这样，在处理 流入单频网适配器中的 TS 流的 PCR 信息时，就 需 息，将其提取出来，然后再 对 PCR 信息进行修正。

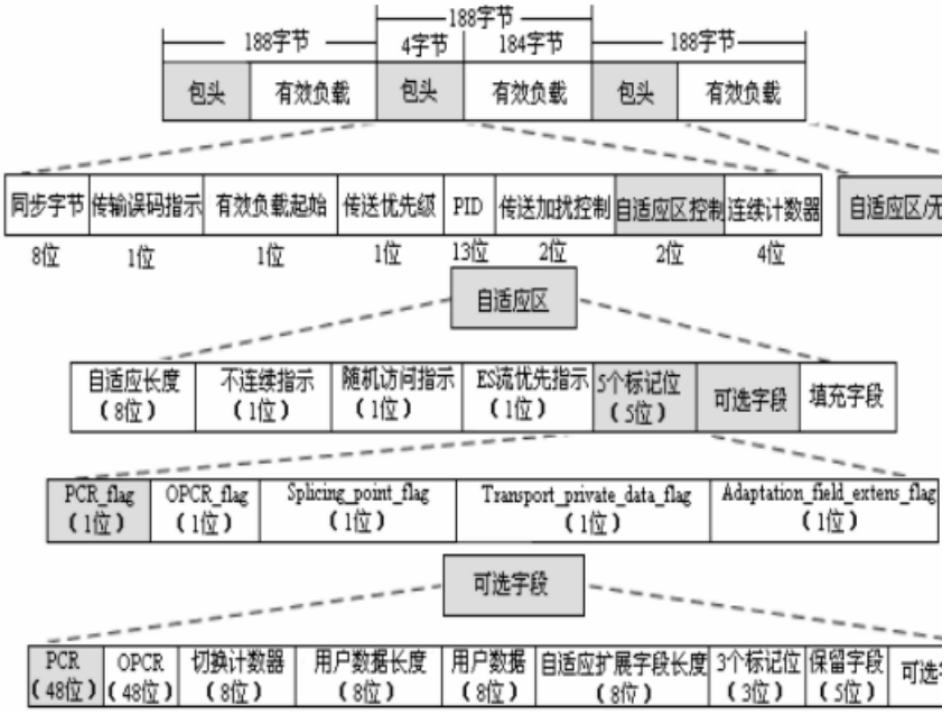


图 1　TS 流的帧结构框

如上图所示，PCR是存在自适应区的。有没有自适应区由TS包头中的adaption_field_control字段控制。

如果有自适应区，那么其内容存放在TS流中数据域中，在有效负载(音视频数据)之前。

图 1 给出了 TS 流的帧结构，由此可以看出， 要查找 PCR，首先应检查包头中自适应控制位**adaption_field_control**的 值，当其为“10”或“11”时 后紧跟 着自适应区，当其为“00”或“01”时，就不存在自适 应区，那么，连续计数器后紧跟着的就是有效负 载。

当判定 TS 包中存在自适应区后，就查询自适应区中的 PCR 标志位是否为 1，如果是 1，则该 TS 包中有 PCR 信息，且 PCR 信息就存放在可选 字段

4. PCR 改进校正算法

对 PCR 字段进行校正的基本思路是在原始的 PCR 上加上一个校正值。一般校正值的计算公式 如下：

ΔPCR = delact − delconst　(4)

其中，delact是某 TS 流的 PCR 从进入本地系统到 离开系统所经历的实际延迟，delconst 是节目的所 有 PCR 使用的一个常数。

假定 PCR 进入适配器时，本地有一个以 27MHZ 时钟计数的计数器，此时该计数器的值记 为 PCRin；当 PCR 离开适配器时，计数器的值记 为 PCR

那么： ΔPCR = PCRout − PCRin （5）

$$PCR_{new} = PCR + \Delta PCR$$
$$= PCR + PCR_{out} - PCR_{in} \qquad (6)$$
$$= (PCR - PCR_{in}) + PCR_{out}$$

对 PCR 先做减法后做加法，中间的差值实质 上就是系统引入非恒定延时所需要校正补偿的 值，这样就间接地实现了 PCR 校正。

按照上述原理，传统的 PCR 间接校正算法就 是在 TS 流进入单频网适配器时，经过 PCR 包查 找和 PCR 域定位，将 PCR 域中的值提取出来和 本地相减，差值存入 FIFO 当中。

TS 流输出时，又一次对 PCR 包进行查找 和域定位，然后从 FIFO 中取出数据，和本地计数 器的当前值相加，将结果按照 TS 包的格式转换为6 个字流相应的 PCR 域里，从 而完成对 PCR 的校正。

这种设计方式和 PCR 直接校正算法（用 27MHZ 的计数器，直接记录 PCR 信息从流入适 配器到流出适配器的时间，然后将这段时间直接 在输出时加上）相比具有占用逻 辑资源少，不用记录与当前 PCR 相匹配的计数器 号，及时序控制相对简单等优点。

但是，由于本 文设计的是单频网适配器中的 PCR 校正，在适配 器中，对TS流进行去空包处理时用到了一个FIFO 作为缓冲器，若用上诉 PCR 间接算将用到一个 FIFO，两个内部 FIFO 的使用将大 大提高对 FPGA 逻辑资源的占用。

另外，在 TS 输入和输出适配器时，都要进行 PCR 的查找和定 位，两个过程完全相同，这也同样增加了 FPGA 的逻辑资源占用。

为了减少 FPGA 的逻辑资源占用率，优化设 计，我们在对 TS 流进行第一次 PCR 查找和定位 以后，就在原有的每个字节前均加上一个标志位。 当定将 PCR 域的每个字节前 的标志位置 1；其他情况下，每个字节前的标志位 均为 0。这样，在输出 TS 流进行第二次查找和定 位 PCR 时，只需看字节是为 0， 只对标志位为 1 的字节进行处理即可。在第一次 查找定位 PCR 信息后，将 PCR 值提取出来与本 地计数器做了减法，差值不再放入 FIFO 流中；输出时再从 TS 流中取出这个 差值与本地计数器做加法，将结果再插回 TS 流， 完成 PCR 校正的间接算法。通过 TS 流本身来 传递减法差值个 FIFO 所要 占用的逻辑资源，又避免了在减法差值存储过程 中对 FIFO 的读写控制，彻底杜绝了数据竞争冒 险的潜在危险，同时还简化了程序。

分类: 流媒体

bw_0927
关注 - 2
粉丝 - 61
+加关注

0

« 上一篇：第四章：深入学习SI
» 下一篇：振荡器 晶振

刷新评论

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。