**3.H.264编码技术**

**3.1.H.264概述**

　　国际上制定视频编解码技术的组织有两个，一个是“国际电联（ITU-T）”，它制定的标准有H.261、H.263、H.263+等，另一个是“国际标准化组织（ISO）”它制定的标准有MPEG-1、MPEG-2、MPEG-4等。而H.264则是由两个组织（ITU-T视频编码专家组（VCEG）和ISO/IEC动态图像专家组（MPEG））联合组建的联合视频组（JVT，Joint Video Team）共同制定的高度压缩数字视频编解码器标准，所以它既是ITU-T的H.264，又是ISO/IEC的MPEG-4高级视频编码（Advanced Video Coding，AVC）的第10 部分。因此，不论是MPEG-4 AVC、MPEG-4 Part 10，还是ISO/IEC 14496-10，都是指H.264。   
　　H264标准各主要部分有Access Unit delimiter（访问单元分割符），SEI（附加增强信息），primary coded picture（基本图像编码），Redundant Coded Picture（冗余图像编码）。还有Instantaneous Decoding Refresh（IDR，即时解码刷新）、Hypothetical Reference Decoder（HRD，假想参考解码）、Hypothetical Stream Scheduler（HSS，假想码流调度器）。   
**优势：**   
　　（1）低码率（Low Bit Rate）：和MPEG2和MPEG4 ASP等压缩技术相比，在同等图像质量下，采用H.264技术压缩后的数据量只有MPEG2的1/8，MPEG4的1/3。   
　　（2）高质量的图像：H.264能提供连续、流畅的高质量图像（DVD质量）。   
　　（3）容错能力强：H.264提供了解决在不稳定网络环境下容易发生的丢包等错误的必要工具。   
　　（4）网络适应性强：H.264提供了网络抽象层（Network Abstraction Layer），使得H.264的文件能容易地在不同网络上传输（例如互联网，CDMA，GPRS，WCDMA，CDMA2000等）。   
　　H.264最大的优势是具有很高的数据压缩比率，在同等图像质量的条件下，H.264的压缩比是MPEG-2的2倍以上，是MPEG-4的1.5～2倍。举个例子，原始文件的大小如果为88GB，采用MPEG-2压缩标准压缩后变成3.5GB，压缩比为25∶1，而采用H.264压缩标准压缩后变为879MB，从88GB到879MB，H.264的压缩比达到惊人的102∶1。低码率（Low Bit Rate）对H.264的高的压缩比起到了重要的作用，和MPEG-2和MPEG-4 ASP等压缩技术相比，H.264压缩技术将大大节省用户的下载时间和数据流量收费。尤其值得一提的是，H.264在具有高压缩比的同时还拥有高质量流畅的图像，正因为如此，经过H.264压缩的视频数据，在网络传输过程中所需要的带宽更少，也更加经济。   
　　H.264／AVC可工作于多种速率，广泛应用于Internet／intranet上的多媒体流服务、视频点播、可视游戏、低码率移动多媒体通信 (视频 手机等)、交互式多媒体应用、实时多媒体监控、数字电视与演播电视和虚拟视频会议等，大有在上述领域一统天下的趋势，有非常广泛的开发和应用前景。

**3.2.H.264编码格式解析**

**3.2.1.名词解释**

　　（1）场和帧：视频的一场或一帧可用来产生一个编码图像。在电视中，为减少大面积闪烁现象，把一帧分成两个隔行的场。   
　　（2）片：每个图象中，若干宏块被排列成片的形式。片分为I片、B片、P片和其他一些片。   
　　I片只包含I宏块，P片可包含P和I宏块，而B片可包含B和I宏块。   
- I宏块利用从当前片中已解码的像素作为参考进行帧内预测。   
- P宏块利用前面已编码图象作为参考图象进行帧内预测。   
- B宏块则利用双向的参考图象（前一帧和后一帧）进行帧内预测。

　　片的目的是为了限制误码的扩散和传输，使编码片相互间是独立的。   
　　某片的预测不能以其它片中的宏块为参考图像，这样某一片中的预测误差才不会传播到其它片中去。   
　　（3）宏块：一个编码图像通常划分成若干宏块组成，一个宏块由一个16×16亮度像素和附加的一个8×8 Cb和一个8×8 Cr彩色像素块组成。   
　　（4）数据之间的关系：   
H.264结构中，一个视频图像编码后的数据叫做一帧，一帧由一个片（slice）或多个片组成，一个片由一个或多个宏块（MB）组成，一个宏块由16x16的yuv数据组成。宏块作为H264编码的基本单位。   
　　（5）H264编码过程中的三种不同的数据形式：   
　　　　① SODB数据比特串 —-＞最原始的编码数据，即VCL数据；   
　　　　② RBSP 原始字节序列载荷 —-＞在SODB的后面填加了结尾比特（RBSP trailing bits：一个bit“1”，若干比特“0”），以便字节对齐；   
　　　　③ EBSP 扩展字节序列载荷 —- > 在RBSP基础上填加了仿校验字节（0X03）它的原因是：在NALU加到Annexb上时，需要添加每组NALU之前的开始码StartCodePrefix，如果该NALU对应的slice为一帧的开始则用4位字节表示0x00000001，否则用3位字节表示0x000001（是一帧的一部分）。另外，为了使NALU主体中不包括与开始码相冲突的，在编码时，每遇到两个字节连续为0，就插入一个字节的0x03。解码时将0x03去掉。也称为脱壳操作。

**3.2.2.H.264/AVC结构分析**

　　H.264从层次来看分为两层：视频编码层（VCL，Video Coding Layer）和网络提取层（NAL，Network Abstraction Layer）。VCL输出的是原始数据比特流（SODB，String of data bits），表示H.264的语法元素编码完成后的实际的原始二进制码流。SODB通常不能保证字节对齐，故需要补齐为原始字节序列负荷（RBSP，Raw Byte Sequence Payload）。NAL层实际上就是最终输出的H.264码流，它是由一个个NALU组成的，每个NALU包括一组对应于视频编码数据的NAL头信息和一个原始字节序列负荷（RBSP，Raw Byte Sequence Payload）。以上名词之间的关系如下：

RBSP = SODB + RBSP trailing bits

NALU = NAL header(1 byte) + RBSP

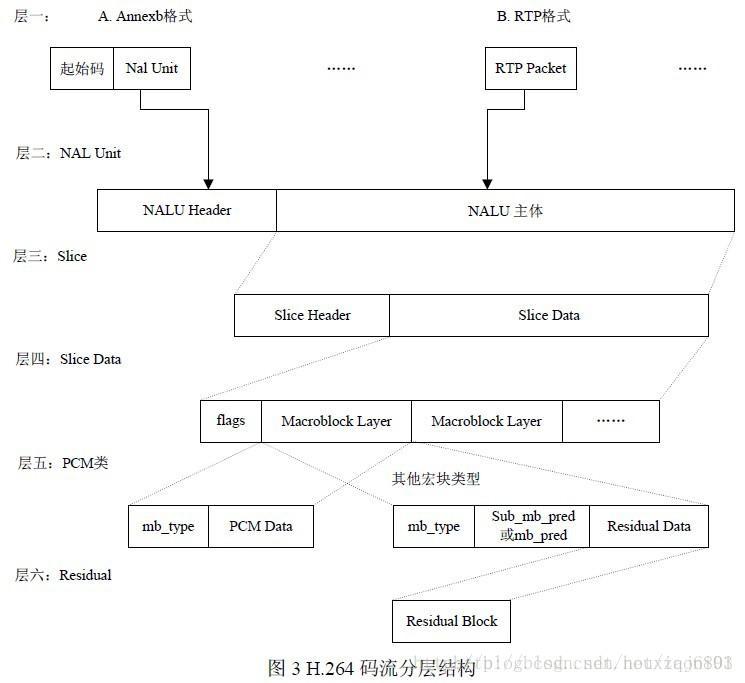
H.264 = Start Code Prefix(3 bytes) + NALU + Start Code Prefix(3 bytes) + NALU +…

* 1
* 2
* 3

所以H.264码流的结构如下：

这里写图片描述

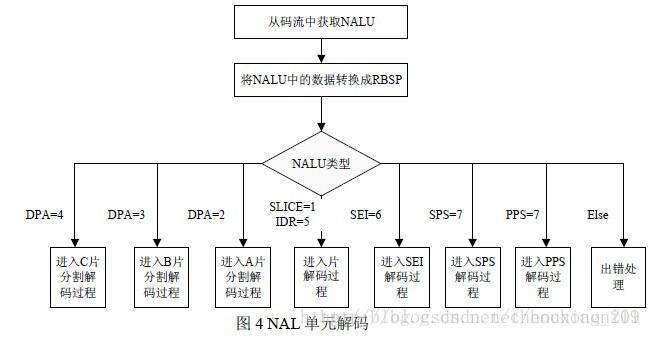
H.264码流层次结构为：



　　编码器将每个NALU各自独立、完整地放入一个分组，因为分组都有头部，解码器可以方便地检测出NALU的分界，并依次取出NALU进行解码。每个NALU之间由起始码（Start Code Prefix）分隔，起始码分为两种：0x000001（3 bytes）or 0x00000001（4 bytes），解码器检测每个起始码，作为一个NALU的起始标识，当检测到下一个起始码时，当前NALU结束。如果NALU 对应的Slice 为一帧的开始，则用4 字节表示，即0x00000001；否则用3 字节表示，0x000001。NALU针对起始码设计了防止冲突机制，如果编码器检测到NALU数据存在0x000000，0x000001，0x000002或0x000003时，编码器会在最后一个字节前插入一个新的字节0x03，如下：   
0x00 00 00 -> 0x00 00 03 00   
0x00 00 01 -> 0x00 00 03 01   
0x00 00 02 -> 0x00 00 03 02   
0x00 00 03 -> 0x00 00 03 03   
　　解码器检测到0x000003时，把03抛弃，恢复原始数据（脱壳操作）。解码器在解码时，首先逐个字节读取NALU的数据，统计NALU的长度，然后再开始解码。

**3.2.3.H.264/AVC解码**

　　一个NALU就是编码后的一帧数据。每个NAL单元是一个一定语法元素的可变长字节字符串，包括一个字节的头信息（用来表示数据类型），以及若干整数字节的负荷数据。一个NAL单元可以携带一个编码片、A/B/C型数据分割或一个序列或图像参数集。NALU头用来标识后面的RBSP是什么类型的数据，它是否会被其他帧参考以及网络传输是否有错误。   
NAL的解码单元的流程如下：



**（1）NAL header是一个字节（length = 1byte）：**   
　　　　　　　　　　　　　　　forbidden\_bit(1bit) + nal\_reference\_bit(2bit) + nal\_unit\_type(5bit)   
　　① forbidden\_bit：禁止位，初始为0，当网络发现NAL单元有比特错误时可设置该比特为1，以便接收方纠错或丢掉该单元。   
　　② nal\_reference\_bit：nal重要性指示，标志该NAL单元的重要性，取值范围为0~3，值越大，越重要，解码器在解码处理不过来的时候，可以丢掉重要性为0的NALU。H.264规定，如果当前NALU是序列参数集，或是图像参数等，该值必须大于0。比如nal\_unit\_type等于5时，nal\_reference\_bit大于0；nal\_unit\_type等于6,9,10,11或12时，nal\_reference\_bit等于0。

不同类型的NALU的重要性指示如下表所示。

| Nal\_unit\_type | NAL类型 | Nal\_reference\_bit |
| --- | --- | --- |
| 0 | 未使用 | 0 |
| 1 | 非IDR的片 | 此片属于参考帧，则不等于0，不属于参考帧，则等于0 |
| 2 | 片数据A分区 | 同上 |
| 3 | 片数据B分区 | 同上 |
| 4 | 片数据C分区 | 同上 |
| 5 | IDR图像的片 | 5 |
| 6 | 补充增强信息单元（SEI） | 0 |
| 7 | 序列参数集 | 非0 |
| 8 | 图像参数集 | 非0 |
| 9 | 分界符 | 0 |
| 10 | 序列结束 | 0 |
| 11 | 码流结束 | 0 |
| 12 | 填充 | 0 |
| 13…23 | 保留 | 0 |
| 24…31 | 不保留 | 0 |

　　所谓参考帧，就是在其他帧解码时需要参照的帧。比如一个I帧可能被一个或多个B帧参考，一个B帧可能被某个P帧参考。   
　　从这个表我们也可以看出来，DIR的I帧是非常重要的，它一丢，那么这个序列的所有帧都没办法解码了；序列参数集和图像参数集也很重要，没有序列参数集，这个序列的帧就没法解；没有图像参数集，那用到这个图像参数集的帧都没法解。   
　　③ nal\_unit\_type：NALU类型取值如下表所示：

| Nal\_unit\_type | NAL类型 | C |
| --- | --- | --- |
| 0 | 未使用 |  |
| 1 | 非IDR图像中不采用数据划分的片段 | 2,3,4 |
| 2 | 非IDR图像中A类数据划分片段 | 2 |
| 3 | 非IDR图像中B类数据划分片段 | 3 |
| 4 | 非IDR图像中C类数据划分片段 | 4 |
| 5 | IDR图像的片 | 2,3 |
| 6 | 补充增强信息单元（SEI） | 5 |
| 7 | 序列参数集 | 0 |
| 8 | 图像参数集 | 1 |
| 9 | 分界符 | 6 |
| 10 | 序列结束 | 7 |
| 11 | 码流结束 | 8 |
| 12 | 填充 | 9 |
| 13…23 | 保留 |  |
| 24…31 | 不保留（RTP打包时会用到） |  |

RTP打包时的扩展类型

|  |  |  |
| --- | --- | --- |
| 24 | STAP-A | Single-time aggregation packet |
| 25 | STAP-B | Single-time aggregation packet |
| 26 | MTAP16 | Multi-time aggregation packet |
| 27 | MTAP24 | Multi-time aggregation packet |
| 28 | FU-A | Fragmentation unit |
| 29 | FU-B | Fragmentation unit |
| 30-31 | undefined |  |

NALU的顺序要求：   
　　H.264/AVC标准对送到解码器的NAL单元顺序是有严格要求的，如果NAL单元的顺序是混乱的，必须将其重新依照规范组织后送入解码器，否则解码器不能够正确解码。   
　　1)序列参数集NAL单元   
　　必须在传送所有以此参数集为参考的其他NAL单元之前传送，不过允许这些NAL单元中间出现重复的序列参数集NAL单元。   
所谓重复的详细解释为：序列参数集NAL单元都有其专门的标识，如果两个序列参数集NAL单元的标识相同，就可以认为后一个只不过是前一个的拷贝，而非新的序列参数集。   
　　2)图像参数集NAL单元   
　　必须在所有以此参数集为参考的其他NAL单元之前传送，不过允许这些NAL单元中间出现重复的图像参数集NAL单元，这一点与上述的序列参数集NAL单元是相同的。   
　　3)不同基本编码图像中的片段（slice）单元和数据划分片段（data partition）单元在顺序上不可以相互交叉，即不允许属于某一基本编码图像的一系列片段（slice）单元和数据划分片段（data partition）单元中忽然出现另一个基本编码图像的片段（slice）单元片段和数据划分片段（data partition）单元。   
　　4)参考图像的影响：如果一幅图像以另一幅图像为参考，则属于前者的所有片段（slice）单元和数据划分片段（data partition）单元必须在属于后者的片段和数据划分片段之后，无论是基本编码图像还是冗余编码图像都必须遵守这个规则。   
　　5)基本编码图像的所有片段（slice）单元和数据划分片段（data partition）单元必须在属于相应冗余编码图像的片段（slice）单元和数据划分片段（data partition）单元之前。   
　　6)如果数据流中出现了连续的无参考基本编码图像，则图像序号小的在前面。   
　　7)如果arbitrary\_slice\_order\_allowed\_flag置为1，一个基本编码图像中的片段（slice）单元和数据划分片段（data partition）单元的顺序是任意的，如果arbitrary\_slice\_order\_allowed\_flag置为零，则要按照片段中第一个宏块的位置来确定片段的顺序，若使用数据划分，则A类数据划分片段在B类数据划分片段之前，B类数据划分片段在C类数据划分片段之前，而且对应不同片段的数据划分片段不能相互交叉，也不能与没有数据划分的片段相互交叉。   
　　8)如果存在SEI（补充增强信息）单元的话，它必须在它所对应的基本编码图像的片段（slice）单元和数据划分片段（data partition）单元之前，并同时必须紧接在上一个基本编码图像的所有片段（slice）单元和数据划分片段（data partition）单元后边。假如SEI属于多个基本编码图像，其顺序仅以第一个基本编码图像为参照。   
　　9)如果存在图像分割符的话，它必须在所有SEI 单元、基本编码图像的所有片段slice）单元和数据划分片段（data partition）单元之前，并且紧接着上一个基本编码图像那些NAL单元。   
　　10)如果存在序列结束符，且序列结束符后还有图像，则该图像必须是IDR（即时解码器刷新）图像。序列结束符的位置应当在属于这个IDR图像的分割符、SEI 单元等数据之前，且紧接着前面那些图像的NAL单元。如果序列结束符后没有图像了，那么它的就在比特流中所有图像数据之后。   
　　11)流结束符在比特流中的最后。   
**（2）RBSP**   
RBSP数据是下表中的一种：

| RBSP类型 | 缩写 | 描述 |
| --- | --- | --- |
| 参数集 | PS | 序列的全局信息，如图像尺寸、视频格式等 |
| 增强信息 | SEI | 视频序列解码的增强信息 |
| 图像界定符 | PD | 视频图像的边界 |
| 编码片 | SLICE | 编码片的头信息和数据 |
| 数据分割 |  | DP片层的数据，用语错误恢复解码 |
| 序列结束符 |  | 表明一个序列的结束，下一个图像为IDR图像 |
| 流结束符 |  | 表明该流中已没有图像 |
| 填充数据 |  | 亚元数据，用于填充字节 |

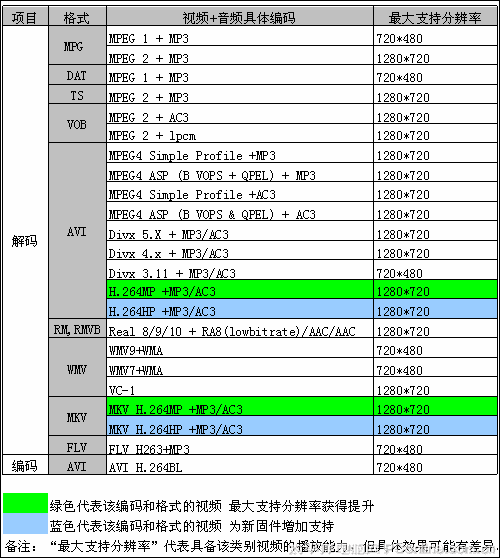
　　从前面的分析我们知道，VCL层出来的是编码完的视频帧数据，这些帧可能是I、B、P帧，而且这些帧可能属于不同的序列，再者同一个序列还有相对应的一套序列参数集和图片参数集等等，所以要完成视频的解码，不仅需要传输VCL层编码出来的视频帧数据，还需要传输序列参数集、图像参数集等数据。   
　　参数集：包括序列参数集 SPS和图像参数集 PPS。SPS包含的是针对一连续编码视频序列的参数，如标识符 seq\_parameter\_set\_id、帧数及 POC 的约束、参考帧数目、解码图像尺寸和帧场编码模式选择标识等等。PPS对应的是一个序列中某一幅图像或者某几幅图像，其参数如标识符 pic\_parameter\_set\_id、可选的 seq\_parameter\_set\_id、熵编码模式选择标识、片组数目、初始量化参数和去方块滤波系数调整标识等等。   
　　数据分割：组成片的编码数据存放在3个独立的DP（数据分割，A、B、C）中，各自包含一个编码片的子集。分割Ａ包含片头和片中每个宏块头数据。分割Ｂ包含帧内和 SI 片宏块的编码残差数据。分割 C包含帧间宏块的编码残差数据。每个分割可放在独立的 NAL 单元并独立传输。

**3.2.4.H.264封装模式**

　　H.264有两种封装模式：   
　　（1）annexb模式：传统模式，有start code， SPS和PPS是在ES中；   
　　（2）mp4模式：没有start code，SPS和PPS是封装在container中，每一个frame前面是这个frame的长度；SPS的头部是0x67，PPS的头部是0x68，要保持对数据的敏感性。

从我们在接触网络上面看电影以来，就会遇到从原先我们熟知的RMVB格式的视频到后来接触的MP4、3GP等格式的视频就会感到为什么同样是AVI的视频，我的MP4却不支持，而别人的就可以呢？这里就涉及到了视频编码与封装格式的问题。本文主要是针对视频编码进行讨论，以后会在下次为大家讲解一下关于封装格式的问题。

所谓**视频编码方式**就是指通过特定的压缩技术，将某个视频格式的文件转换成另 一种视频格式文件的方式。目前视频流传输中最为重要的编解码标准有国际电联的H.264，运动静止图像专家组的M-JPEG和国际标准化组织运动图像专家 组的MPEG系列标准，此外在互联网上被广泛应用的还有Real-Networks的RealVideo、微软公司的WMV以及Apple公司的 QuickTime等，到目前google力推的WebM格式都收到了我们的关注。



**视频国际标准化相关组织的的ISO和ITU-T**

格式的统一肯定会极大地提高人们的生活的便利以及数据的传播，为什么还会有如此繁多的视频编码的方式，难道就没有专门机构或者组织来管理一下吗？带着这些疑问我们认识一下底下的两个机构。

**■ ITU-T**

ITU-T的中文名称是国际电信联盟远程通信标准化组织(ITU-T for ITU Telecommunication Standardization Sector), 它是国际电信联盟管理下的专门制定远程通信相关国际标准的组织。由ITU-T指定的国际标准通常被称为建议(Recommendations)。由于 ITU-T是ITU的一部分，而ITU是联合国下属的组织，所以由该组织提出的国际标准比起其它的组织提出的类似的技术规范更正式一些。

它制定的标准有H.261、H.263、H.263+等，目前流行最广的，影响也是最大的H.264也有他的一份功劳。底下附上

H - 视频音频以及多媒体系统复合方法

H.223 低码率多媒体通信复合协议

H.225.0 也被称为实时传输协议

H.261 视频压缩标准, 约1991年

H.262 视频压缩标准(和MPEG-2第二部分内容相同), 约1994年

H.263 视频压缩标准, 约1995年

H.263v2 (也就是 H.263+) 视频压缩标准, 约1998年

H.264 视频压缩标准(和MPEG-4第十部分内容相同), 约2003年

H.323 基于包传输的多媒体通信系统