

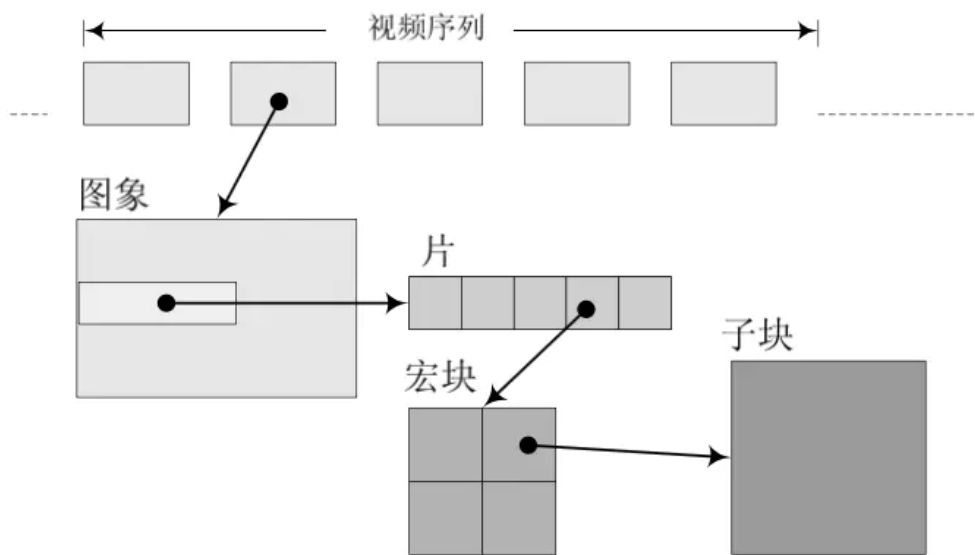
# H.264(一)NALU解析

noobyard.com/article/p-xcchbqon-rz.html

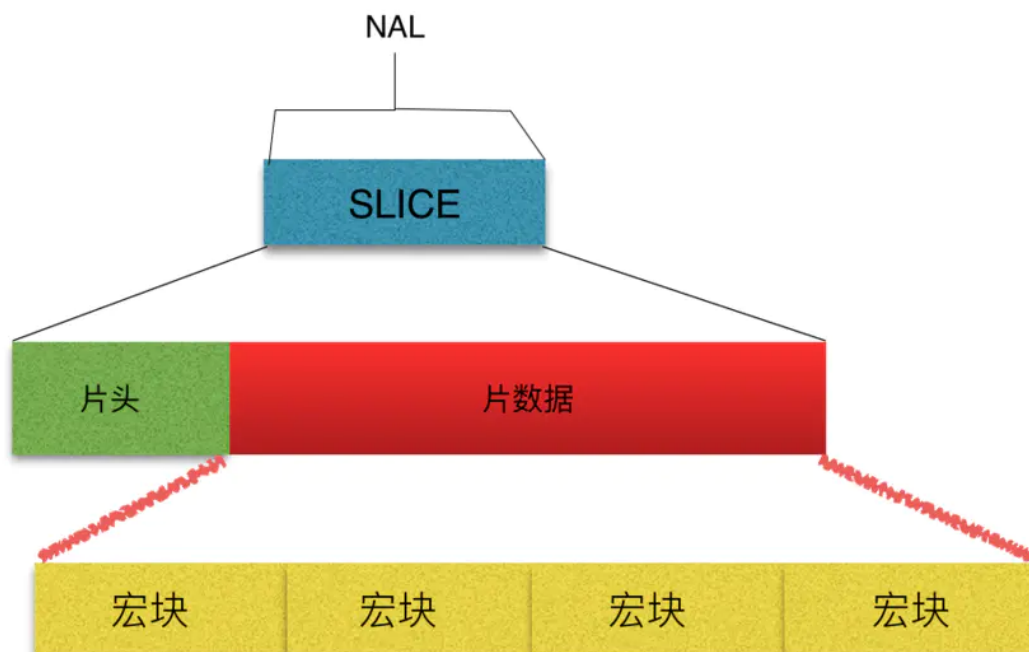
时间 2021-01-07

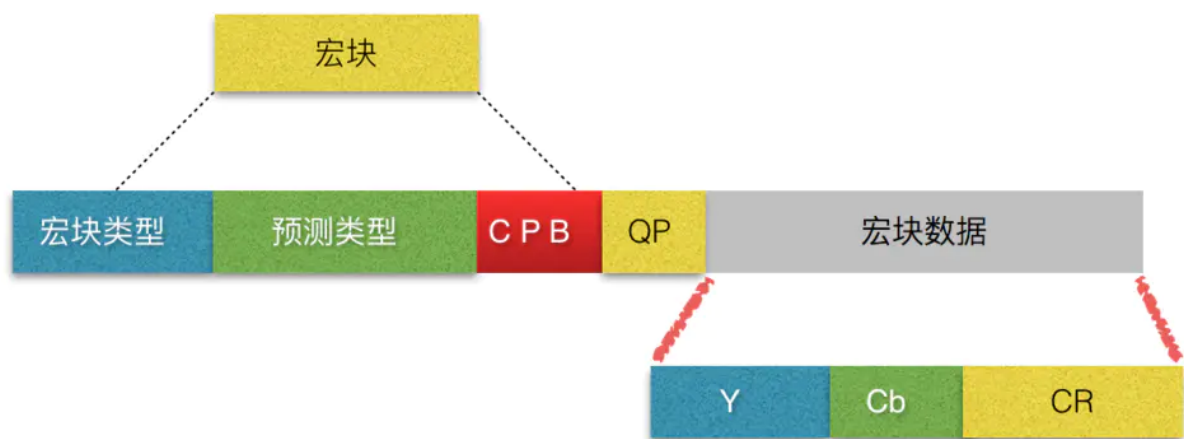
原文 [https://blog.csdn.net/qg\\_40732350/article/details/89339242](https://blog.csdn.net/qg_40732350/article/details/89339242)

## 视频序列



## 宏块结构



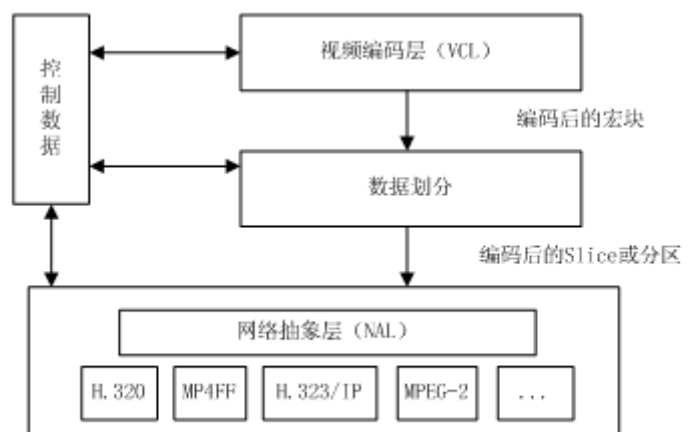


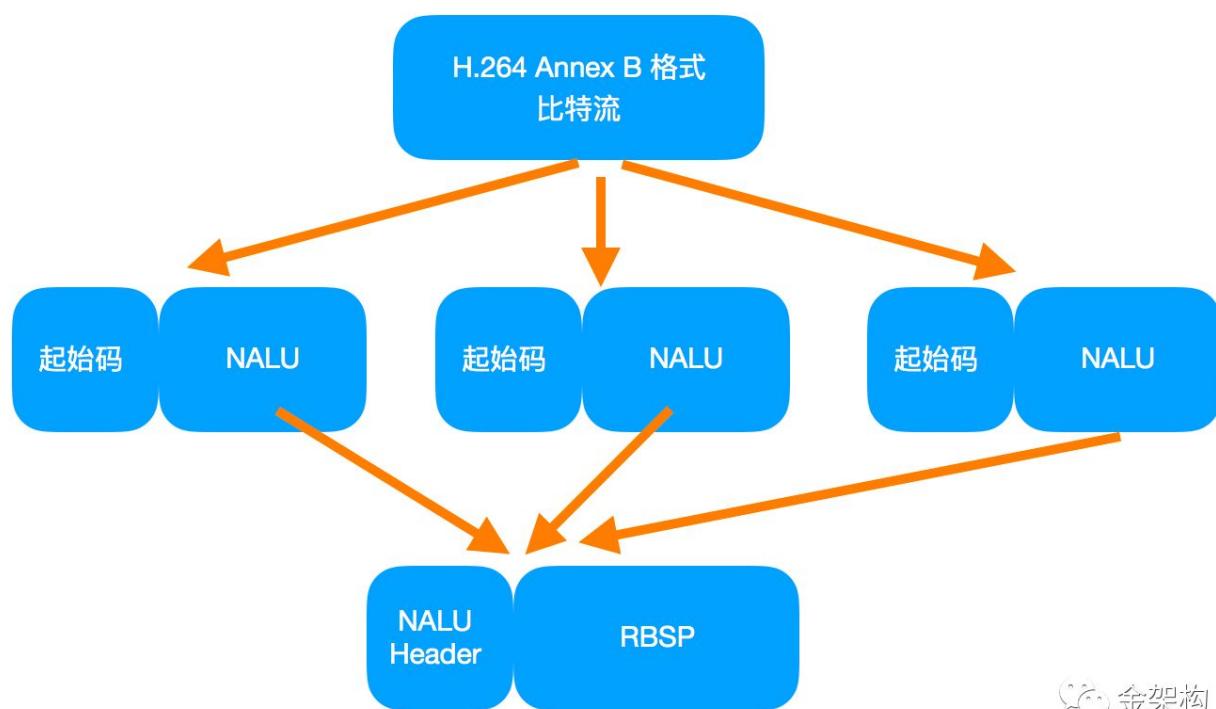
## NALU分层

H264的主要目标是为了有高的视频压缩比和良好的网络亲和性，为了达成这两个目标，H264的解决方案是将系统框架分为两个层面，

**VCL(视频编码层)和 NAL(网络提取层)。**

- VCL：包括核心压缩引擎和块，宏块和片的语法级别定义，设计目标是尽可能地独立于网络进行高效的编码。
- NAL：负责将VCL产生的比特字符串**适配到各种各样的网络**和多元环境中，覆盖了所有片级以上的语法级别。





金架构

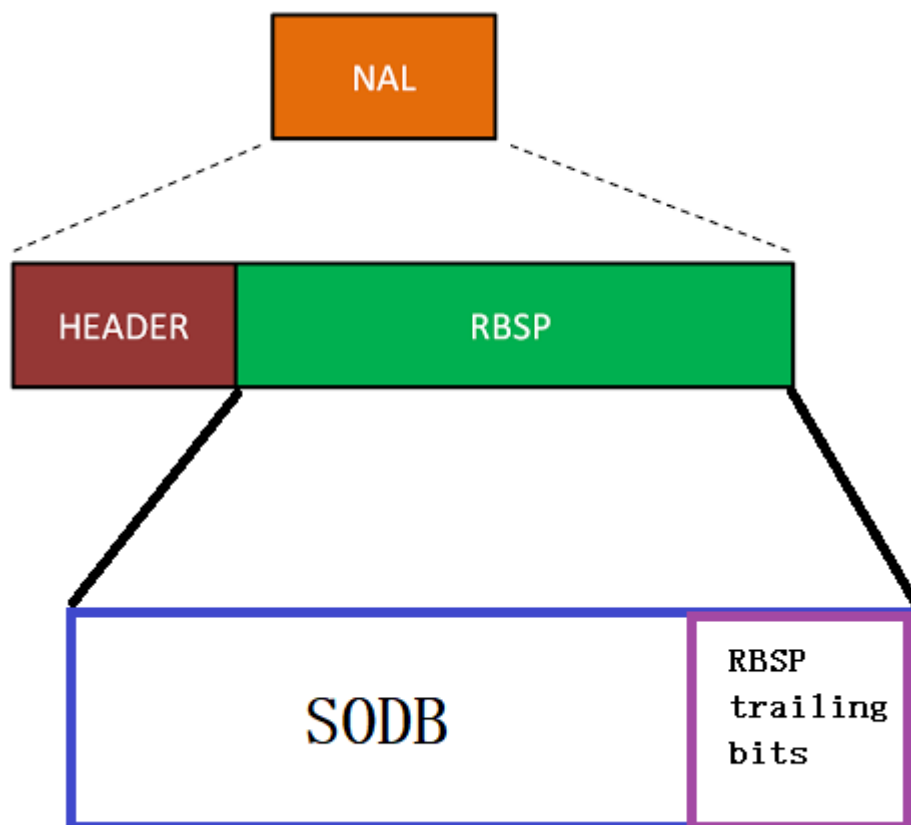
NALU：（Network Abstract Layer Unit）网络抽象层单元。

RBSP：（Raw Byte Sequence Payload）原始字节序列载荷。

SODB：String Of Data Bits （原始数据比特流, 长度不一定是8的倍数，故需要补齐，是由VCL产生）。

SODB是以值为1的一个比特结束，如果没有字节对齐，就用0补齐，所以从后往前第一个值为1的位置就为，SODB的最后一个字节。

逻辑关系：RBSP trailing bits 是拖尾字节，用于字节对齐。



其实严格来说，这个等式是不成立的，因为RBSP并不等于NALU刨去NALU Header。严格来说，NALU的组成部分应为：

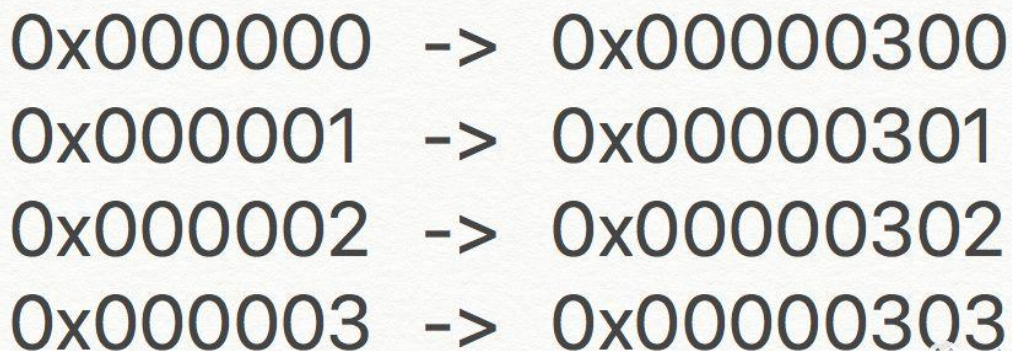
$$\text{NALU} = \text{NALU Header} + \text{EBSP}$$

其中的EBSP为扩展字节序列载荷（Encapsulated Byte Sequence Payload），而RBSP为原始字节序列载荷（Raw Byte Sequence Payload）。那为什么我们上面，没有使用2式而使用了1式呢？那是因为，在h264的文档中，并没有EBSP这一名词出现，但是在h264的官方参考软件JM里，却使用了EBSP。

**EBSP相较于RBSP，多了防止竞争的一个字节：0x03。**

我们知道，NALU的起始码为0x000001或0x00000001，同时H264规定，当检测到0x000000时，也可以表示当前NALU的结束。那这样就会产生一个问题，就是如果在NALU的内部，出现了0x000001或0x000000时该怎么办？

所以H264就提出了“防止竞争”这样一种机制，当编码器编码完一个NAL时，应该检测NALU内部，是否出现如下左侧的四个序列。当检测到它们存在时，编码器就在最后一个字节前，插入一个新的字节：0x03。



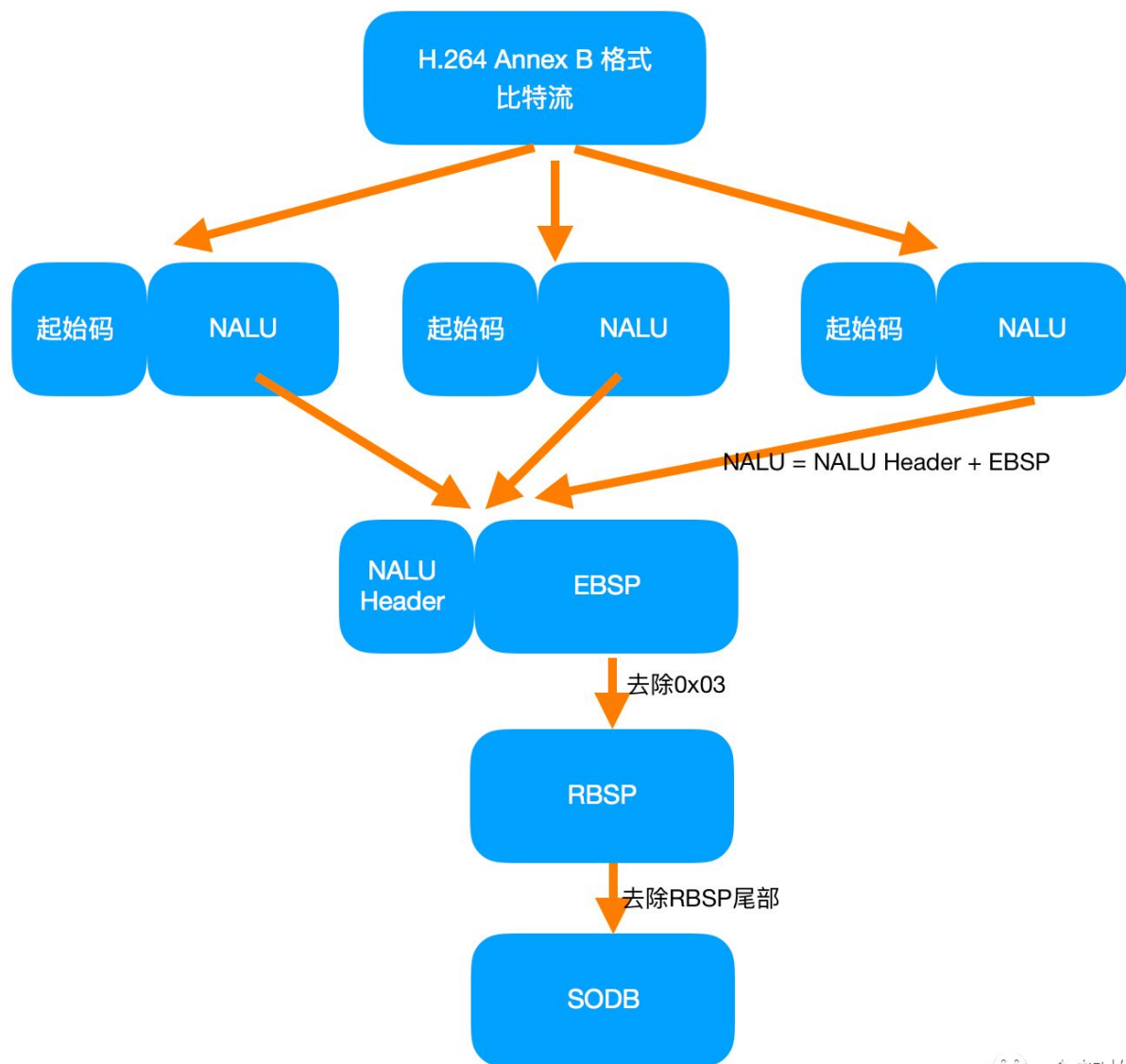
0x000000	->	0x000000300
0x000001	->	0x000000301
0x000002	->	0x000000302
0x000003	->	0x000000303

金架构

这样一来，当我们拿到EBSP时，就需要检测EBSP内是否有序列：0x000003，如果有，则去掉其中的0x03。这样一来，我们就能得到原始字节序列载荷：RBSP。

**总结：H264的码流结构如下：**

---

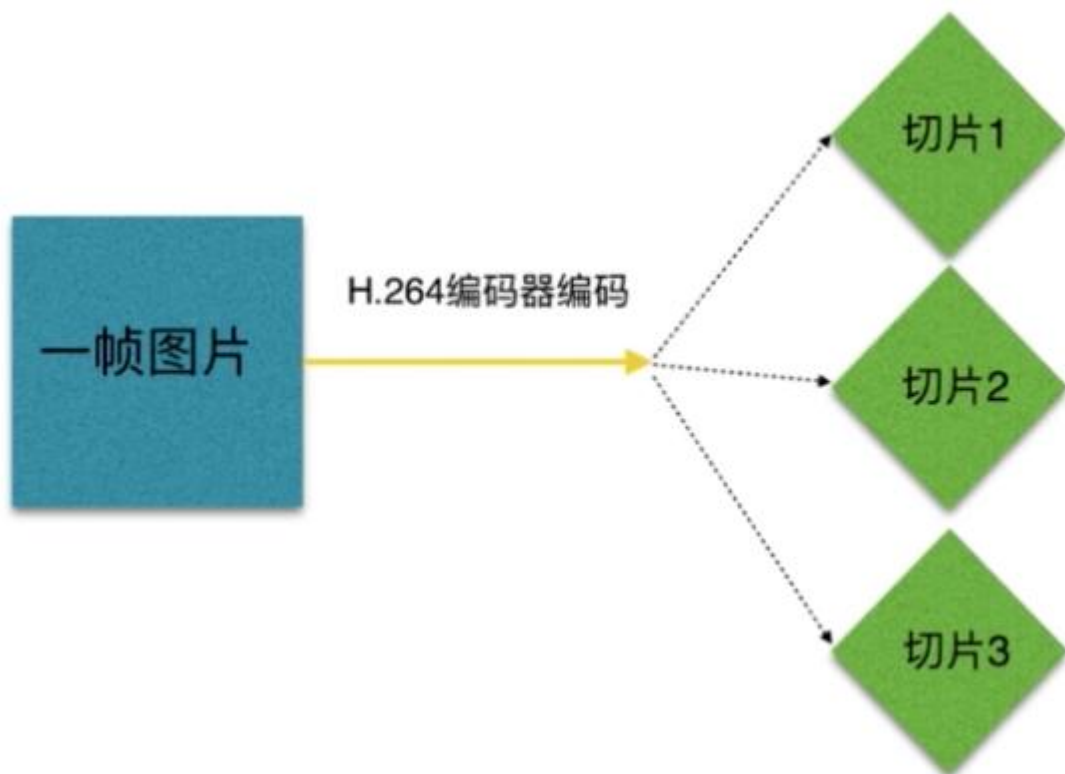
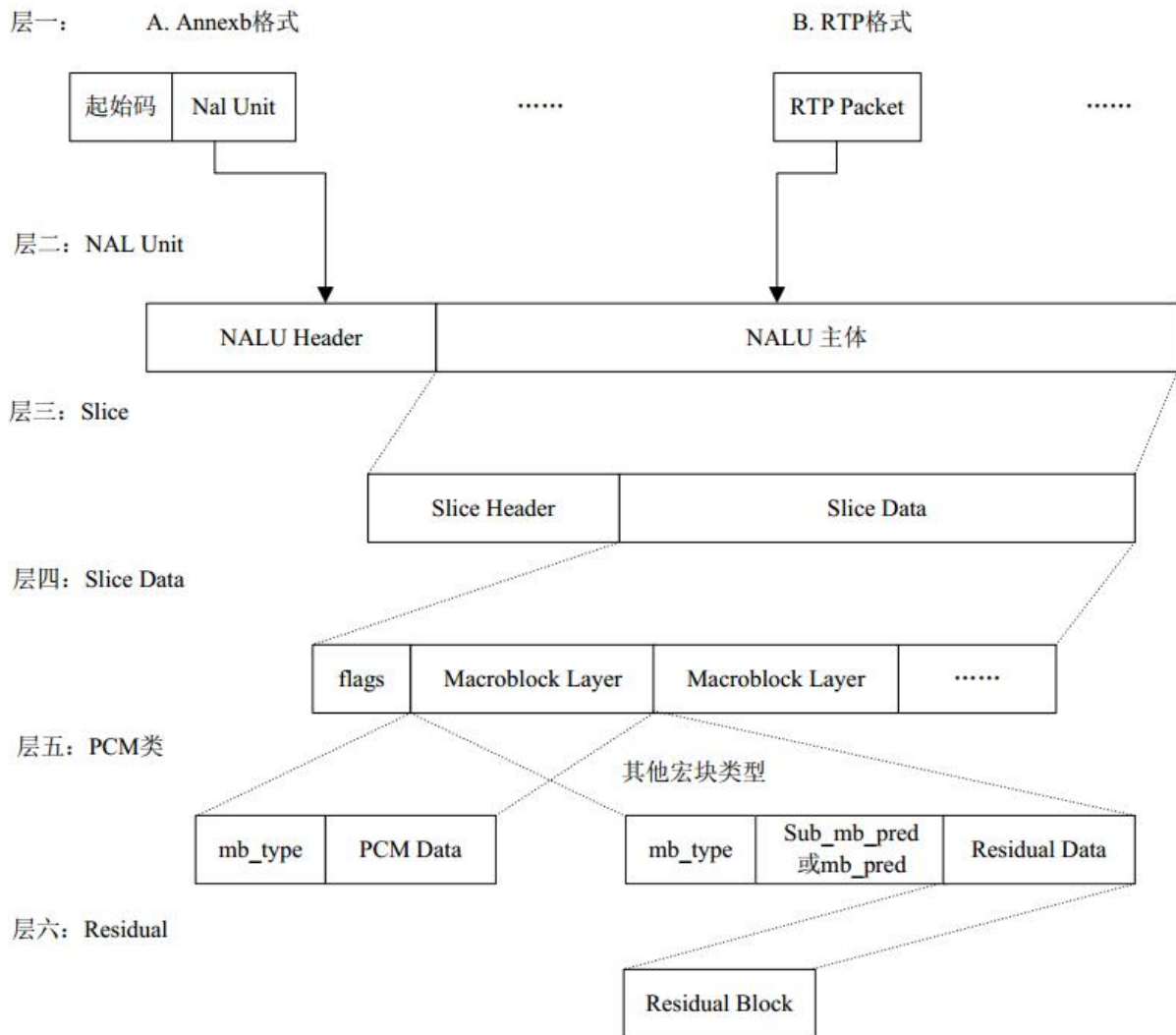


金架构

---

## NALU分层结构

---



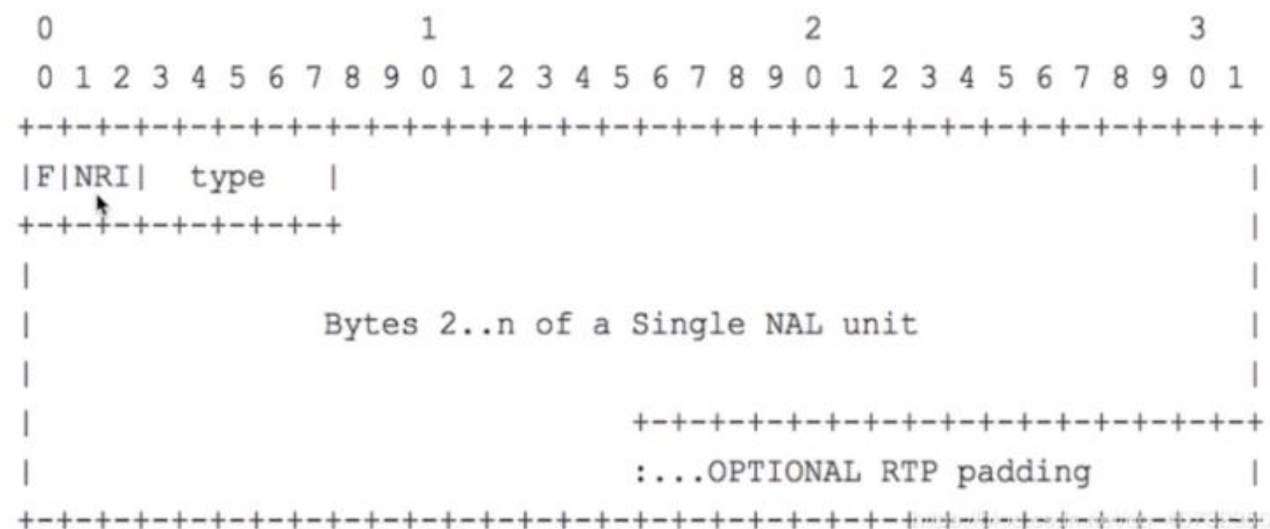
## RTP包的NALU类型介绍

单一类型：一个RTP包只包含一个NALU

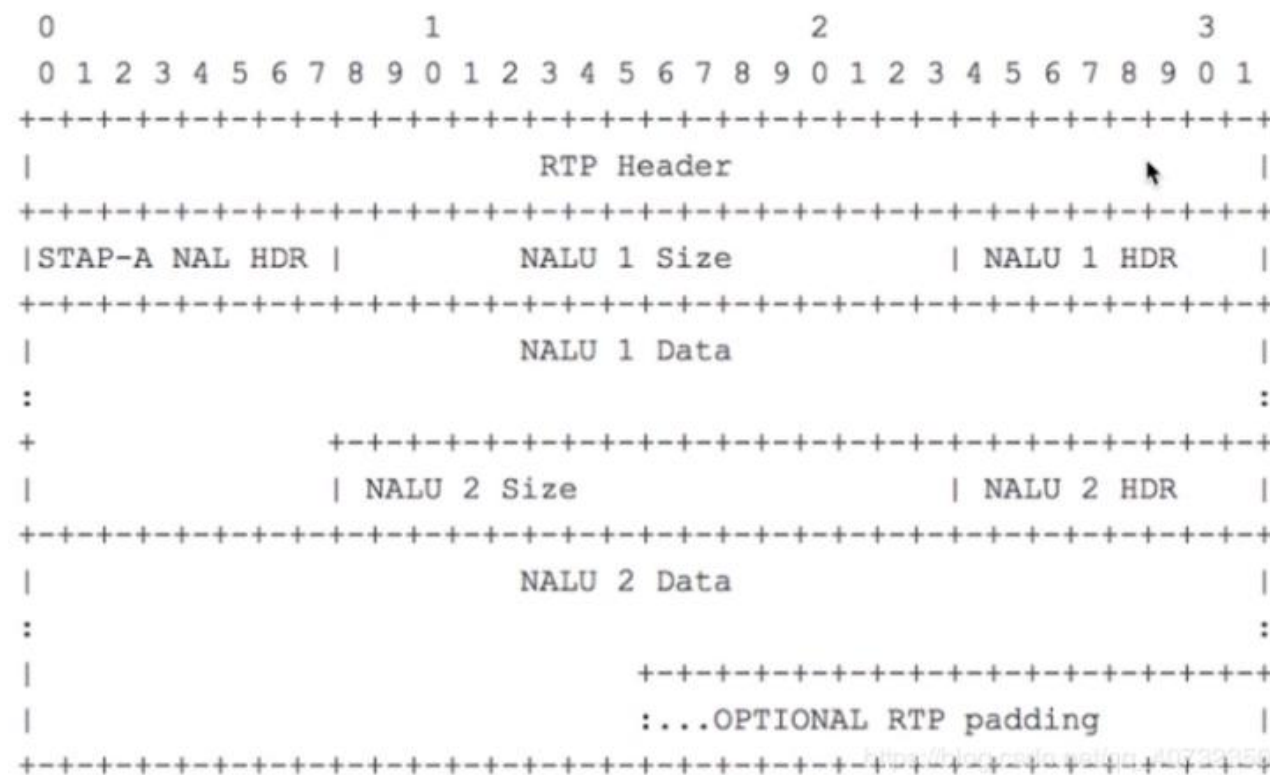
组合类型：一个RTP包包含多个NALU，类型是24 — 27

分片类型：一个NALU单元分成多个RTP包，类型是28和29

### 单一NALU的RTP包

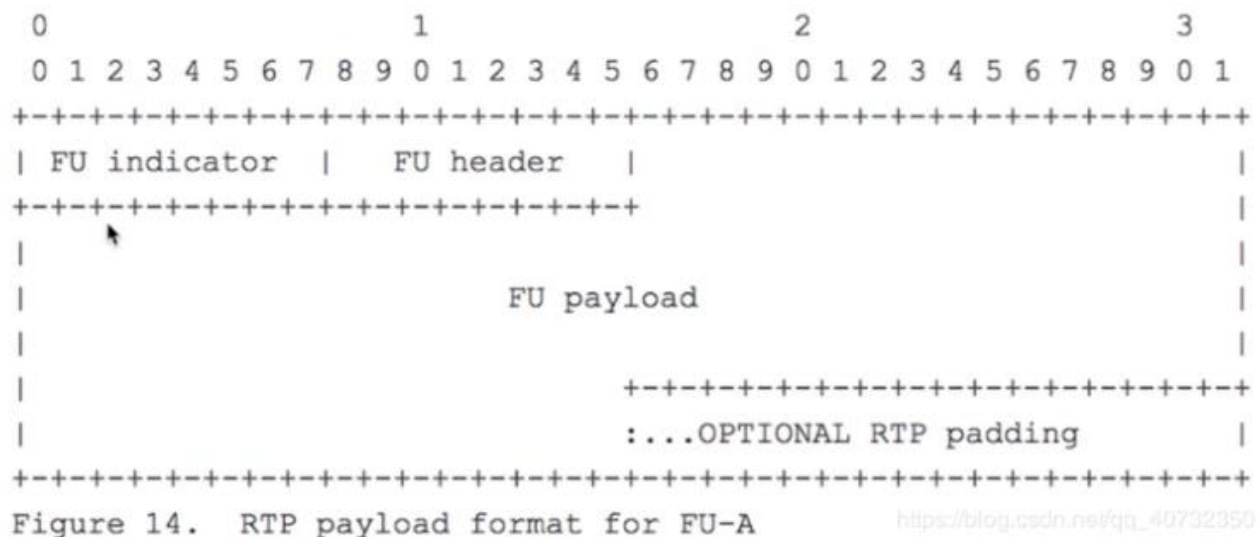


### 组合NALU的RTP包



### 分片NALU的RTP包





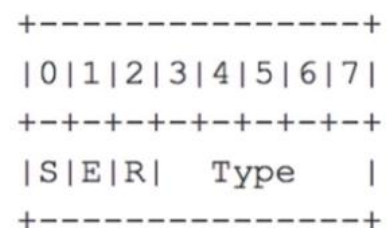
## FU Header

◆ **S** start bit , 用于指明分片的开始

◆ **E** end bit , 用于指明分片的结束

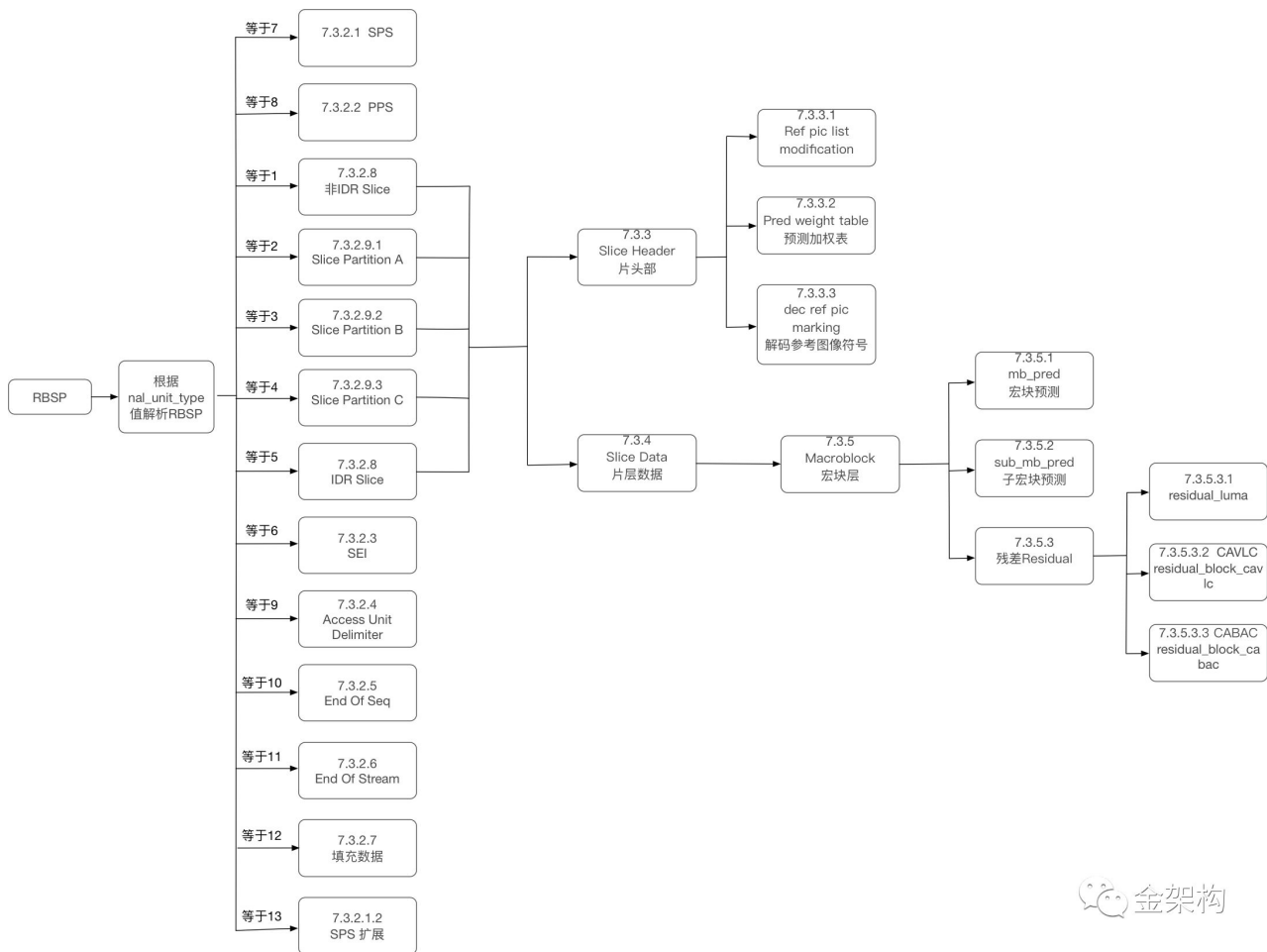
◆ **R** 未使用 , 设置为 0

◆ **Type** 指明分片NAL类型



### H264句法元素解析流程

而当我们拿到RBSP或SODB之后，就可以对照各类型的NALU，去解析它们的语法元素，进而再根据语法元素，重建图像。其中解析语法元素的框图如下：



由图可见，解析NALU的各个句法元素并不难，只要根据h264文档对应章节的句法，并配合相应的编解码算法解析即可。而相应的编解码算法如指数哥伦布编码、CAVLC、CABAC、算术编码，我们会一步步涉猎。