

## HEVC 高层语法

对于只对 HEVC 的应用感兴趣，而对于 HEVC 编码不感兴趣的读者，只看这一节应该就足够了。这一节讲的是 HEVC 与 H.264/AVC 码流在使用时的区别。与 H.264/AVC 相同，HEVC 也分为视频编码层（Video Coding Layer, VCL）和网络虚拟层（Network Abstraction Layer, NAL）。原始视频经过 VCL 层，被编码成视频数据，然后经过 NAL 层，封装成一个个 NAL 包。这一节主要讲 H.264/AVC 与 HEVC 在 NAL 层上的区别。

H.264/AVC 的视频参数存在于 SPS（Sequence Parameter Set）和 PPS（Picture Parameter Set）两种 NAL 包中。在 HEVC 中，新增了一种 VPS（Video Parameter Set）包，包含视频的一些全局信息，如 Profile、Level 等。

HEVC 的 NAL 包结构与 H.264/AVC 的不同，表 1-1 和表 1-2 显示了 NAL 包头的区别。可以看到，HEVC 中 NAL 头变为了两字节长度，同时加入了该 NAL 所在的时间层的 id，去掉了表示是否被参考的 `nal_ref_idc`，将是否被参考的信息定义在了 `nal_unit_type` 中，即只有某些类型的 NAL 包被参考。

由于近年来并行技术的兴起，同时语法元素解析在硬件实现中通常是瓶颈，HEVC 在高层语法中加入了大量对并行的支持。包括传统的 Slice，和新的 Tiles、WPP（Wavefront Parallel Processing）和 Dependent Slices 技术。这些技术使得一帧之内的语法元素不是从头到尾的依赖关系，可以并行地编码和解码。这些技术使得较高复杂度的 CABAC 熵编码方式也能应用在实时编码的场景下。传统的 CAVLC 熵编码方式因此有被淘汰的趋势。CAVLC 在 HEVC 里不再存在于块一级的编码中，仅在头信息编码中使用。这些支持并行的技术细节比较多，仅仅某些具体的场景才会用到，这里都不进行具体介绍了。以后有机会笔者会通过另外的文章进行详细说明。

nal_unit( NumBytesInNALunit ) {	<b>C</b>	<b>Descriptor</b>
<b>forbidden_zero_bit</b>	All	f(1)
<b>nal_ref_idc</b>	All	u(2)
<b>nal_unit_type</b>	All	u(5)
NumBytesInRBSP = 0		
nalUnitHeaderBytes = 1		
if( nal_unit_type == 14    nal_unit_type == 20 ) {		
nal_unit_header_svc_extension( ) /* specified in Annex G */		
nalUnitHeaderBytes += 3		
}		
for( i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++ ) {		
if( i + 2 < NumBytesInNALunit && next_bits( 24 ) == 0x000003 ) {		
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	All	b(8)
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	All	b(8)
i += 2		
<b>emulation_prevention_three_byte</b> /* equal to 0x03 */	All	f(8)
} else		
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	All	b(8)
}		
}		

表 1-1 H.264 NAL 结构

nal_unit( NumBytesInNALunit ) {	<b>Descriptor</b>
nal_unit_header( )	
NumBytesInRBSP = 0	
for( i = 2; i < NumBytesInNALunit; i++ ) {	
if( i + 2 < NumBytesInNALunit && next_bits( 24 ) == 0x000003 ) {	
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	b(8)
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	b(8)
i += 2	
<b>emulation_prevention_three_byte</b> /* equal to 0x03 */	f(8)
} else	
<b>rbsp_byte</b> [ NumBytesInRBSP++ ]	b(8)
}	
}	

<code>nal_unit_header() {</code>	
<code>    <b>forbidden_zero_bit</b></code>	<code>f(1)</code>
<code>    <b>nal_unit_type</b></code>	<code>u(6)</code>
<code>    <b>nuh_reserved_zero_6bits</b></code>	<code>u(6)</code>
<code>    <b>nuh_temporal_id_plus1</b></code>	<code>u(3)</code>
<code>}</code>	

表 1-2 HEVC NAL 结构