HEVC 高层语法

与 H.264/AVC 相同，HEVC 分为视频编码层（Video Coding Layer, VCL）和网络提取层（Network Abstraction Layer, NAL）。前者独立于网络，主要包括核心压缩引擎和块、宏块和片的语法句法定义。后者主要是定义数据的封装格式，把 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。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 包以适应不同网络中的视频传输。HEVC 码流在应用过程中与 H.264/AVC 码流的区别就在于 NAL 层。以下将着重介绍 H.264/AVC 与 HEVC 在 NAL 层上的区别，旨在为关注 HEVC 应用并对 HEVC 编码详细技术没有兴趣的开发人员提供帮助。

H.264/AVC 中，视频参数封存于 SPS（Sequence Parameter Set）和 PPS（Picture Parameter Set）两种 NAL 包中。在 HEVC 中，除了继续使用 SPS 和 PPS 封装视频参数等，还新增了一种名为 VPS（Video Parameter Set）的 NAL 包，其中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 的有明显不同，表 1 和表 2 展示了二者 NAL 包头的区别。相比于 H.264，HEVC 中 NAL 头变成两字节长度，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，并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，而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技术的支持，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再是从头到尾的依赖关系，而是可以进行并行地编码和解码。较高复杂度的 CABAC 熵编码方式也能应用于实时编码的场景下，因而传统的 CAVLC 熵编码方式已有被淘汰的趋势。另外，CAVLC 在 HEVC 中不再存在于块一级的编码中，而是仅在头信息编码中使用。由于支持并行的技术细节比较多，而且仅在某些具体的场景才会用到，此处就不再赘述。后续可能会通过其他文章进行详细说明。

表1 H.264/AVC NAL结构

nal_unit(NumBytesInNALunit) {	C	Descriptor
forbidden_zero_bit	All	f(1)
nal_ref_idc	All	u(2)
nal_unit_type	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) {		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
i += 2		
emulation_prevention_three_byte /* equal to 0x03 */	All	f(8)
} else		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
}		
}		

表 2 HEVC NAL 结构

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

nal_unit_header( ) {	
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_reserved_zero_6bits	u(6)
nuh_temporal_id_plus1	u(3)
}	