

Syntax

Break-out group report



GE imagination at work

Syntax changes

> Purpose:

- “Optimize” the bytes in the NAL header
- Allow NAL 20/21 payload compatible with AVC

> Changes NAL Header

- Using 3 bytes
- Moved the fragment management to the NAL header
- Moved a “layer base” flag to indicate AVC syntax compliant payload
- Moved the use_base_rep_flag to the NAL header
- Moved reserved_zero_byte to the beginning



GE imagination at work

Problems solved

- > Can have SVC NALU payload with pure AVC syntax
- > Can identify FGS slices
 - fragmented_flag=1
 - If there is no fragment then **fragment_order=0** and **last_fragment_flag=1**
- > All the information for extraction is in the header
- > Mostly a modification of the position of the syntax elements
- > We have two more bits left ☺



GE imagination at work

New syntax

nal_unit_header_svc_extension() {	C	Descriptor
reserved_zero_bit	All	u(1)
priority_id	All	u(6)
discardable_flag	All	u(1)
temporal_level	All	u(3)
dependency_id	All	u(3)
quality_level	All	u(2)
layer_base_flag	All	u(1)
use_base_prediction_flag	All	u(1)
fragmented_flag	All	u(1)
last_fragment_flag	All	u(1)
fragment_order	All	u(2)
reserved_zero_two_bits	All	u(2)
nalUnitHeaderBytes += 3		
}		



GE imagination at work

Julien's question

- > Why do we need NALU = 21 in SVC ?
 - IDR are identified by the presence of NALU=5 in the AU
 - NALU = 21 just indicates EIDR
 - This could be indicated as a flag
- > Guess what... 😊
 - we have a flag free in the new NAL header



GE imagination at work

Idea

nal_unit_header_svc_extension() {	C	Descriptor
reserved_zero_bit	All	u(1)
priority_id	All	u(6)
discardable_flag	All	u(1)
temporal_level	All	u(3)
dependency_id	All	u(3)
quality_level	All	u(2)
layer_base_flag	All	u(1)
use_base_prediction_flag	All	u(1)
fragmented_flag	All	u(1)
last_fragment_flag	All	u(1)
fragment_order	All	u(2)
eidr_flag	All	u(1)
reserved_zero_two_bits	All	u(1)
nalUnitHeaderBytes += 3		
}		



GE imagination at work



GE imagination at work

Alternative approach

> Purpose:

- “Optimize” the bytes in the NAL header
- Allow NAL 20/21 payload compatible with AVC

> Changes NAL Header

- Move quality_level to the slice header
- Add a “layer base” flag to indicate AVC syntax compliant payload
- Add the use_base_rep_flag
- Moved reserved_zero_byte to the beginning

> Changes in the slice header



GE imagination at work

Propose modifications

<code>nal_unit_header_svc_extension() {</code>	C	Descriptor
<code>reserved_zero_bit</code>	All	u(1)
<code>priority_id</code>	All	u(6)
<code>discardable_flag</code>	All	u(1)
<code>temporal_level</code>	All	u(3)
<code>dependency_id</code>	All	u(3)
<code>layer_base_flag</code>	All	u(1)
<code>use_base_prediction_flag</code>	All	u(1)
<code>nalUnitHeaderBytes += 2</code>		
<code>}</code>		



GE imagination at work

Advantages

When **layer_base_flag=1**

- > **base_id = -1** and **quality_level=0**
- > Syntax of the NAL payload is AVC syntax
- > Notes:
 - To see if you are in PR, you need to get the slice_type
 - You can have quality_level==0 even if avc_syntax_flag==0
 - All NAL of a layer base of a scalable layer (of an AU) HAVE the same priority_id. (consequence of the constraint on PI)
 - store_base_rep_flag is sent in the PR slices



GE imagination at work