연재

# MPEG-4 다중화 시스템의 이해(하)

**호요성**
광주과학기술원 정보통신공학과 교수

3) BIFS AU(Access Unit)

BIFS 데이터 AU는 BIFS CommandFrame 혹은 BIFS AnimationFrame으로 구성된다. bifs CommandFrame 혹은 AnimationFrame은 주어진 시간에 처리되어야 되는 모든 데이터를 운반하여야 한다. BIFS 비트열에 있는 AU들은 라벨이 붙어서 구분되어야 하고 목적에 맞게 시간정보인 타임스탬프가 붙어야 한다. 이것은 SL 패킷 헤더부분에 있는 관련된 플래그(flag)들과 컴포지션 타임스탬프(composition time stamp, CTS )들을 통하여 수행되어야 한다. 컴포지션 타임은 BIFS AU에 있는 CommandFrame 혹은 AnimationFrame이 유효해지는 시간을 나타낸다. 이것은 BIFS AU로 설명되는 오디오-비쥬얼 객체들에 발생하는 어떠한 변화가 이상적인 컴포지터에서는 이 시간에 정확하게 보이거나 들리도록 되어야 한다는 것을 의미한다. 이때는 BIFS AU에 대한 복호화와 컴포지션 시간은 항상 같은 값을 가져야 한다.

AU는 완전한 화면을 구성하는데 필요한 데이터를 반드시 전송될 필요는 없다. 이 경우 AU는 단지 화면 구성 정보의 현재 상태에 따라 갱신된다. 그렇지만 만약 AU가 주어진 시간에서 완전한 전체화면 구성 데이터를 전송하고 있다면, SL 패킷 헤더에 있는 randomAccessPointFlag는 해당 AU에 대해서 1로 설정되어야 한다. 그렇지 않은 경우는 randomAccessPointFlag는 0값을 가져야 한다.

4) BIFS 비트열을 위한 기준시간(time base)

BIFS 비트열에 해당되는 기준시간은 필요한 부분에서 나타나야 한다. 이것은 이 비트열을 위해서 SL 패킷 헤더에 있는 OCR (object clock reference) 타임스탬프들에 의해 나타내어지거나 기준시간을 가지는 기초비트열에 의해서 나타내어져야 하며, 이 비트열은 OCR 비트열이라 한다. **그림 1**은 이러한 시간적인 관계를 보여주고 있다.

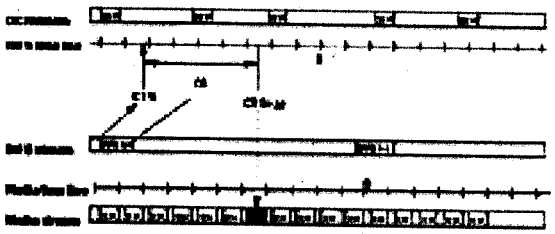

그림 1. 미디어 시작 시간과 CTS

5) BIFS 화면 그래프(scene graph)

개념적으로 BIFS 화면은 3차원 공간상에서 방향성 그래프형태로 분포된 비쥬얼과 오디오 기본개체들(primitives)의 집합으로 표현된다. 그렇지만, BIFS 화면은 이 개념적인 모델의 특정한 경우를 표현하는 여러 개의 하부 계층이 될 수 있다. 특별히, BIFS 화면 설명은 다음과 같은 것으로 구성된 화면을 지원한다.

- 2D만의 기본개체
- 3D만의 기본개체
- 2D와 3D 기본개체의 조합
- 오디오만의 기본개체

2D와 3D 기본개체를 조합한 화면에서 다음과 같은 가능성이 존재한다.

- 깊이를 가진 2D 공간에서 계층화된 완전한 2D와 3D 화면들

- 2D 혹은 3D 기본개체를 위한 텍스춰 맵(texture map)으로서 사용되는 2D와 3D 화면들
- 3D 화면에 있는 지역적인 좌표계 시스템의 지역적인 X-Y 평면에 그려진 2D 화면들

**그림 2**는 전형적인 BIFS 화면 구조를 보여주고 있다.

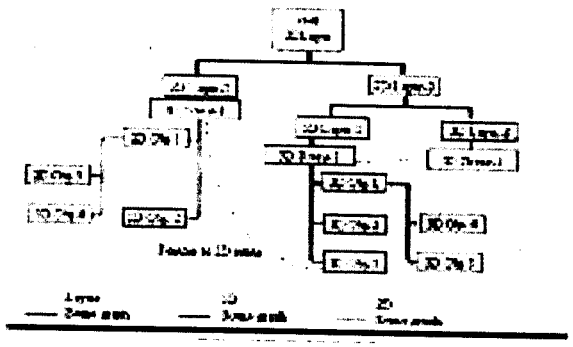

그림 2. 화면 그래프의 예제

## 2. MPEG-4 시스템의 비트열 구조

### 가. 기본 동작

MPEG-4 시스템 표준은 **그림 3**과 같은 동작을 수행한다. 화면구성 기술(scene description, SD)과 기초비트열은 MPEG-4 시스템 표준의 구조를 형성하는 기본 블록이다. 기초비트열은 SD뿐만 아니라 오디오 혹은 비쥬얼 객체에 대한 정보를 운반한다. 객체 기술자(object descriptor, OD)는 기초비트열과 SD간의 연결관계를 제공한다. SD는 오디오-비쥬얼 객체들의 공간적인 관계와 시간적인 관계를 정의한다. OD는 화면에 따라 시간적으로 변화하는 데이터를 제공하는 기초비트열 자원을 규정한다.

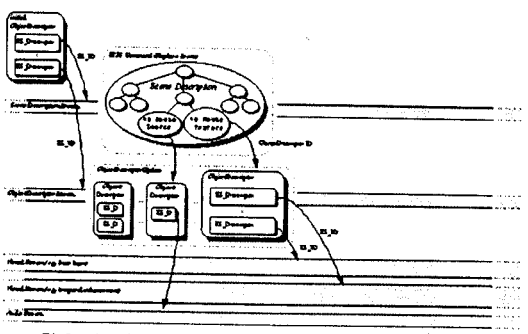

그림 3. 기초비트열과 여러 가지 기술어들간의 상호연관 및 동작

해당 기술어(descriptor)들을 유일하게 식별하기 위하여 tag가 사용된다. 각 기술어들에 대해 할당된 tag값 중 중요한 몇 가지는 **표 1**과 같다.

표 1. 기술어(Descriptor)들에 할당된 tag값

| Tag값 | Tag 이름 |
|---|---|
| 0x00 | 사용금지됨 |
| 0x01 | ObjectDescrTag |
| 0x02 | InitialObjectDescrTag |
| 0x03 | ES_DescrTag |
| 0x04 | DecoderConfigDescrTag |
| 0x05 | DecSpecificInfoTag |
| 0x06 | SLConfigDescrTag |

### 나. OD (Object Descriptor)

1) 개요

OD는 개별적인 기초비트열들과 그것들의 성질을 설명하기 위하여 그들 항목에 대한 다양한 추가적인 기술어(descriptor)들을 포함한다. **그림 4**는 OD의 전체 구조이고 여기서는 OD의 구문구조(synta)와 의미(semantics)를 정의한다.

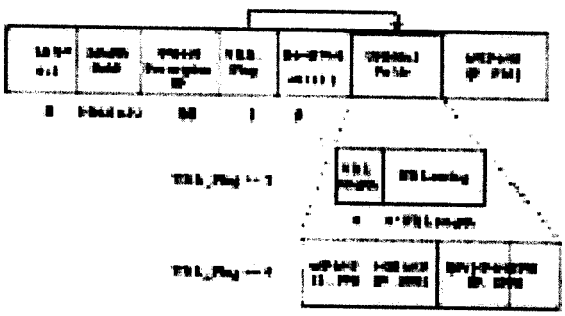

그림 4. OD의 구조

2) 의미

OD는 세 개의 다른 부분으로 이루어져 있다. 첫번째 부분은 objectDescriptorID와 같이 OD를 유일하게 식별하기 위한 라벨이다. SD에 있는 노드들은 이 값을 이용하여 해당되는 OD를 참조한다. 선택적인 URLstring은 실제 객체가 존재하는 원격 위치를 나타낸다. 두번째 부분은 ES_Descriptor들의 배열로 구성되어 있다.

이것은 각각의 기초비트열의 특성을 기술한다. 세번째 부분은 미래의 확장이나 역방향 호환성(backward compatible)을 제공하기 위한 선택적인 기술어들이다.

**objectDescriptorId** - 이 구문요소는 적용 범위 내에서 유일하게 ObjectDescriptor를 식별하는데 사용된다. 0값은 사용금지되어 있고 1023은 예약되어 있다.
**URL_Flag** - URLstring의 존재를 지시하는 플래그이다.
**URLlength** - 바로 뒤따라오는 URLstring의 바이트 길이이다.
**URLstring []** - UTF-8 코드로 부호화된 URL 문자열로서 다른 OD를 지시할 수 있다. 이 OD의 컨텐트는 이 URL을 통한 접근에 의해서만 전송 개체에 전달되어야 한다. 새로운 OD는 URLstring을 운반하는 OD의 objectDescriptorID에 의해서 참조되어야 한다. 허용가능한 URL은 규정된 전송 계층뿐만 아니라 프로화일(profile)과 레벨(level)에 의해서도 제약받을 수 있다.
**esDescr []** - ES_Descriptor들의 배열이다. 이 배열은 1개 이상 30개까지의 요소를 가져야 한다.

### 다. IOD(InitialObjectDescriptor)

1) 개요

InitialObjectDescriptor는 ObjectDescriptor의 변형된 형태로서 프로화일과 레벨 정보를 나타내는 것을 허용한다. 이것은 MPEG-4 표준 컨텐트의 초기 설정값을 위하여 사용되어야 한다. **그림 5**는 IOD의 구조를 보여주고 있다.

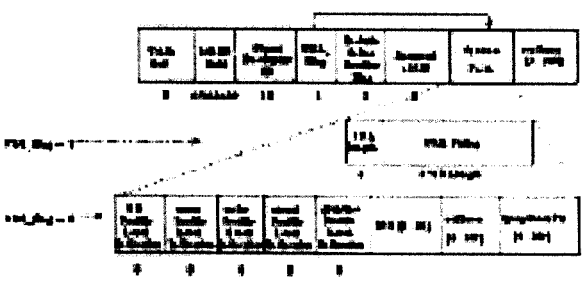

그림 5. IOD의 구조

2) 의미

**ODProfileLevelIndication** - 이 IOD와 연관된 컨텐트의 처리 시에 요구되는 OD 프로화일과 레벨을 나타낸다.
**sceneProfileLevelIndication** - 이 IOD와 연관된 컨텐트의 처리 시에 요구되는 scene graph 프로화일과 레벨을 나타낸다.
**audioProfileLevelIndication** - 이 IOD와 연관된 컨텐트의 처리 시에 요구되는 audio 프로화일과 레벨을 나타낸다.
**visualProfileLevelIndication** - 이 IOD와 연관된 컨텐트의 처리 시에 요구되는 visual 프로화일과 레벨을 나타낸다.
**graphicsProfileLevelIndication** - 이 IOD와 연관된 컨텐트의 처리 시에 요구되는 graphics 프로화일과 레벨을 나타낸다.

### 라. ES_Descriptor(ESD)

1) 개요

ESD는 특정한 기초비트열과 연관된 정보를 운반하며, 세 개의 다른 부분으로 이루어져 있다. 첫번째 부분은 ES_ID와 같이 ESD를 유일하게 식별하기 위한 라벨이다. 부모 OD와의 의존성을 설명하는 메커니즘이다. 선택적인 URLstring은 실제 객체가 존재하는 원격 위치를 나타낸다. 두번째 부분은 기초비트열들이 요구하는 특성을 기술하는 해당 항목 기술어들로 구성되어 있다. 세번째 부분은 미래의 확장이나 역방향 호환성(backward compatible)을 제공하기 위한 선택적인 기술어들이다. **그림 6**은 ESD의 구문구조이다.

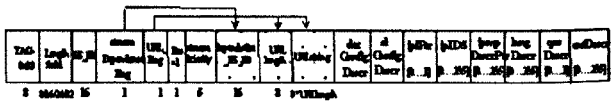

그림 6. ESD의 구조

2) 의미

**ES_ID** - 이 구문요소는 적용 범위 내에서 유일하게

ES_Descriptor를 식별하는데 사용된다. 0값은 사용 금지되어 있고 0xFFFF값은 예약되어 있다.

**streamDependenceFlag** - 만약 이 값이 1로 설정되어 있으면 dependsOn_ES_ID가 뒤따라옴을 나타낸다.

**streamPriority** - 이 기초비트열의 우선순위를 측정하기 위한 상대적인 값이다. 높은 streamPriority를 가지는 기초비트열이 낮은 streamPriority를 가지는 기초비트열보다 더 중요하다. streamPriority의 절대적인 값은 표준상에 정의되어 있지 않다.

**dependsOn_ES_ID** - 이 기초비트열에 의존하는 또 다른 기초비트열의 ES_ID이다. dependsOn_ES_ID를 가지는 비트열은 현재의 ES_Descriptor와 같은 ObjectDescriptor안에 존재해야 한다.

**decConfigDescr** - 뒤의 DecoderConfigDescriptor에 규정되어 있다.

**slConfigDescr** - 뒤의 SLConfigDescriptor에 규정되어 있다.

### 마. DecoderConfigDescriptor

1) 개요

decoderConfigDescriptor는 복호화기 타입과 해당 기초비트열이 요구하는 복호화기 자원에 관한 정보를 제공한다. 이것은 수신단 측에서 해당 기초비트열을 복호화할 수 있는지의 여부를 결정하기 위하여 필요하다. **그림 7**은 decoderConfigDescriptor의 구조이다.

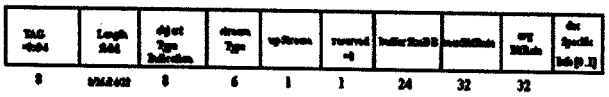


그림 7. decoderConfigDescriptor의 구조

2) 의미

**objectTypeIndication** - 이 기초비트열에 대한 복호화기에 의해 지원되는 객체(object) 혹은 SD 타입을 지시한다. 오디오 비트열과 비쥬얼비트열 이외의 다른 비트열의 streamType값은 0xFF이어야 하고 object type이 규정되지 않았음을 나타낸다.

표 2. streamType 값

| streamType | 비트열 타입 설명 |
|---|---|
| 0x00 | 사용금지 |
| 0x01 | Object Descriptor 비트열 |
| 0x02 | Clock Reference 비트열 |
| 0x03 | Scene Description 비트열 |
| 0x04 | Visual 비트열 |
| 0x05 | Audio 비트열 |
| 0x06 | MPEG-7 비트열 |
| 0x07 | IPMP 비트열 |
| 0x08 | Object Content Information(OCI) 비트열 |
| 0x09-0x1F | ISO에서 사용을 위해 예약됨 |
| 0x20-0x3F | 개인 사용자용 |

**streamType** - **표 2**와 같이 이 기초비트열의 타입을 나타낸다.

**upStream** - 이 비트열이 상향비트열 정보임을 나타낸다.

**bufferSizeDB** - 이 기초비트열을 위한 복호화 버퍼의 크기를 바이트 단위로 나타낸다.

**maxBitrate** - 1초의 간격동안 이 기초비트열의 최대 비트율을 초당 비트수로 나타낸다.

**avgBitrate** - 이 기초비트열의 평균 비트율을 초당 비트수로 나타낸다. 가변 비트율을 가진 비트열들에 대해서 이 값은 0으로 설정되어야 한다.

### 바. FlexMux 패킷과 SL 패킷

1) FlexMuxPacket의 구조

FlexMux도구는 여러 가지 다양한 비트율로 변화하는 SL-패킷화된 비트열의 인터리빙(interleaving)을 가진 유연한 다중화기이다. FlexMux의 기본 데이터는 FlexMux패킷이며 가변 길이를 가지고 있다. 하나 이상의 SL 패킷들은 하나의 FlexMux패킷으로 묶여진다. FlexMux도구는 FlexMuxChannel 번호로서 서로 다른 기초비트열들로부터 이루어지는 SL 패킷들의 식별방법을 제공한다. 각 SL-패킷화된 비트열은 하나의 FlexMuxChannel에 해당된다. 그러므로 서로 다른 SL-패킷화된 비트열들로부터의 데이터를 가진 FlexMux패

킷들은 임의로 인터리빙될 수 있다. 하나의 비트열로 인터리빙된 FlexMux 패킷들의 시퀀스는 FlexMux 비트열이라고 불린다.

가) 단순모드 (Simple Mode)

단순 모드에서 하나의 SL 패킷은 하나의 FlexMux 패킷과 FlexMux Channel 번호와 같은 인덱스(index)로 구별되도록 묶여진다. **그림 8**은 이 모드에서 FlexMux 패킷의 구조를 보여주고 있다. 이 모드는 수신단 측에서 별다른 상태의 설정이나 관리를 필요로 하지 않는다.

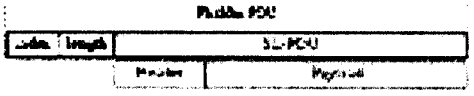

그림 8. SL 패킷 정보를 운반하는 단순모드 FlexMux 패킷구조

나) 먹스코드 모드 (MuxCode mode)

먹스코드 모드에서는 하나 이상의 SL 패킷들은 **그림 9**와 같은 하나의 FlexMux 패킷으로 묶여진다. 이 모드는 수신단 측에서 상태의 설정과 관리를 필요로 한다. 설정값은 어떻게 FlexMux 패킷들이 다중의 SL 패킷들 간에서 공유되는지를 설명한다. 이 모드에서 index 값은 서로 다른 FlexMux Channel들에서 FlexMux 패킷 페이로드의 배치를 정의하는 재참조 설정 정보로서 사용된다.

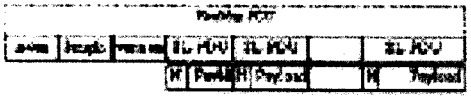

그림 9. SL 패킷 정보를 운반하는 먹스코드모드 FlexMux 패킷구조

간략하게 전체 과정을 통틀어 설명하자면, 시스템 부분은 DMIF로부터 FlexMux Channel에 해당하는 FlexMux 패킷을 입력 받은 후, 그것을 구성하고 있는 SL 패킷들을 패킷 헤더와 패킷 페이로드로 분리하고, 패킷 헤더에 있는 타이밍과 패킷화 정보를 이용하여 기초비트열들을 분리해낸다. 분리된 기초비트열은 해당되는 복호화기에 입력되어 복호화가 이루어진다.

2) SL 패킷 계층

싱크 계층(sync layer)에서 ES는 패킷들의 시퀀스로 매핑된다. 패킷화 정보는 ES와 싱크 계층을 생성하는 개체간에서 교환된다. 이 관계는 두 계층간의 개념적인 인터페이스를 잘 설명하는 ESI(elementary stream interface)로 불린다. 이 관계는 **그림 10**과 같이 간략한 그림으로 나타낼 수 있다.

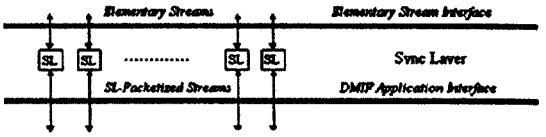


그림 10. 전체 개념도 중 SL 패킷 역다중화 부분

싱크 계층(Sync Layer, SL)은 AU 단위로 ES를 패킷화하기 위한 구문법을 규정한다. 그러한 패킷은 SL 패킷이라 불린다. 하나의 ES로부터의 결과인 SL 패킷들의 시퀀스는 SL 패킷화된 비트열(SL-packetized stream, SPS )이라고 불린다. AU는 동기화(synchronisation)를 위한 기본 단위로 사용된다. SL 패킷의 구조는 **그림 11**과 같이 SL 패킷 헤더와 SL 패킷 페이로드로 구성되어 있다. SL 패킷 헤더는 타임스탬프(time stamp)와 관련 정보를 운반한다. SL 패킷 페이로드(payload)는 실제 데이터 페이로드 부분을 운반한다.

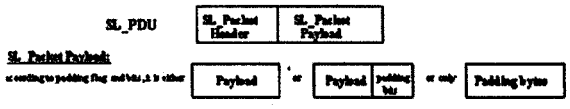


그림 11. SL 패킷의 구조

### 사. SLConfigDescriptor

1) 개요

SL 패킷 헤더는 각 개별적인 기초비트열의 요구사항에 따라서 개별적으로 설정된다. 파라메터들은 타임스탬프들과 클록 참조값의 존재유무, 해상도, 시간의 정확도에 관한 사항을 포함한다. 예를 들면, 유연성(flexibility)은 저속 비트율 기초비트열을 발생하도록 매우 짧은 오버헤드를 가진 SL 패킷 헤더를 만든다. 각 기초비트열에 대하여 설정값은 OD 내에 있는 ES_Descriptor의 부분인 SLConfigDescriptor에서 운반된다. SL 패킷 헤더에서 설정가능한 파라메터들은 두 그룹으로 나누어진다. 하나는 OCR, sequenceNumber 등과 같이 각 SL 패킷에 대해 적용되는 것이고, 다른 하나는 타임스탬프(DTS, CTS), accessUnitLength, instantBitrate,

degradationPriority등과 같은 AU(access unit)와 직접 연관된 것들이다. SLConfigDescriptor의 구조는 **그림 12**와 같다.

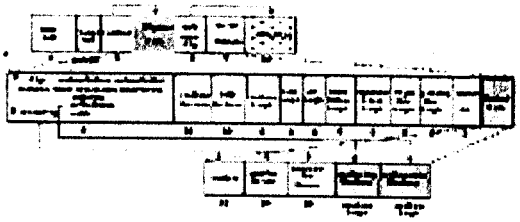

그림 12. SLConfigDescriptor의 구조

2) 의미

**predefined** - predefined 파라메터 집합으로부터 설정된 기본 설정값이다.

**useAccessUnitStartFlag** - 이 기초비트열의 각 SL 패킷 헤더에 accessUnitStartFlag가 존재하는지를 나타낸다.

**useAccessUnitEndFlag** - 이 기초비트열의 각 SL 패킷 헤더에 accessUnitEndFlag가 존재하는지를 나타낸다.

만약 useAccessUnitStartFlag나 useAccessUnitEndFlag가 존재하지 않으면 각 SL 패킷은 완전한 AU로 구성되어 있음을 내포한다.

**useRandomAccessPointFlag** - 이 기초비트열의 각 SL 패킷 헤더에 RandomAccessPointFlag가 존재하는지를 나타낸다.

**hasRandomAccessUnitsOnlyFlag** - 임의 접근 위치(random access point)에 해당되는 각 SL 패킷을 나타낸다. 이 경우 randomAccessPointFlag는 사용되지 않아도 된다.

**usePaddingFlag** - 이 기초비트열의 각 SL 패킷 헤더에 paddingFlag가 존재하는지를 나타낸다.

**useTimeStampsFlag** - 이 기초비트열의 각 SL 패킷 헤더에 동기일치를 위해 사용되는 타임스탬프가 존재하는지를 나타낸다.

타임스탬프들은 SL 패킷 헤더에서 운반되고, 이 SL 패킷 헤더 부분에서 운반되는 다른 파라메터인 accessUnitRate, compositionUnitRate, startDecodingTimeStamp, startCompositionTimeStamp도 동기일치를 위해 사용된다.

**useIdleFlag** - idleFlag가 이 기초비트열에서 사용되는지를 나타낸다.

**durationFlag** - 이 기초비트열에 대한 AU와 CU가 상수 갯수만큼 지속됨을 나타낸다.

**yimeStampResolution** - 초당 클록수로 나타낸 타임스탬프의 해상도를 나타낸다.

**OCRResolution** - 초당 사이클 수로 나타낸 OTB의 해상도를 나타낸다.

**timeStampLength** - SL 패킷 헤더에 있는 타임스탬프의 길이를 나타낸다. 이 값은 0에서 64사이의 값을 가져야 한다.

**OCRlength** - SL 패킷 헤더에 있는 objectClockReference의 길이를 나타낸다. 길이 0은 이 기초비트열에 objectClockReferences가 존재하지 않음을 나타낸다. 만약 OCRstreamFlag가 설정되어 있으면 OCRLength는 0이 되어야 하고, 그렇지 않으면 OCRlength는 0에서 64사이의 값이어야 한다.

**AU_Length** - 이 기초비트열에 대한 SL 패킷 헤더에 있는 accessUnitLength 필드들의 길이이다. AU_Length는 0에서 32사이의 값을 취해야 한다.

**instantBitrateLength** - 이 기초비트열에 대한 SL 패킷 헤더에 있는 instantBitrate 필드의 길이이다.

**degradationPriorityLength** - 이 기초비트열에 대한 SL 패킷 헤더에 있는 degradationPriority 필드의 길이이다.

**AU_seqNumLength** - 이 기초비트열에 대한 SL 패킷 헤더에 있는 AU_sequenceNumber 필드의 길이이다.

**packetSeqNumLength** - 이 기초비트열에 대한 SL 패킷 헤더에 있는 packetSequenceNumber 필드의 길이 이다.

**timeScale** - AU와 CU의 지속 시간을 나타내는데 사용된다. 1초는 timeScale에 있는 값으로 균일하게 나누어진다.

**accessUnitDuration** - AU의 지속시간은 accessUnitDuration*1/timeScale 초이다.

**compositionUnitDuration** - CU의 지속시간은 compositionUnitDuration*1/timeScale 초 이다.

**startDecodingTimeStamp** - 이 기초비트열의 복호화되어야 하는 첫번째 AU에 대한 시간을 운반한다. timeStampResolution 값에 의해서 해상도가 운반된다.
**startCompositionTimeStamp** - 이 기초비트열의 복호화된 첫번째 AU와 관련된 CU에 대한 시간을 운반한다. timeStampResolution 값에 의해서 해상도 값이 운반된다.
**OCRstreamFlag** - OCR_ES_ID 필드가 뒤따라옴을 나타낸다.
**OCR_ES_Id** - 이 기초비트열의 기준 시간(time base)을 유도하는데 사용되는 기초비트열의 ES_Id를 나타낸다.

### 아. SL Packet Header

1) 구조

SL 패킷 헤더는 SLConfigDescriptor에 부호화된 값에 대응되는 비트 길이와 플래그에 따라서 타이밍과 동기 일치, 임의 접근 등의 처리를 위한 정보들이 부호화되며 **그림 13**과 같은 구조를 가지고 있다.

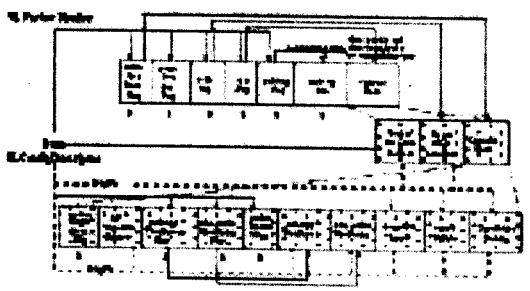

그림 13. SL 패킷 헤더의 구조

2) 의미

**accessUnitStartFlag** - 이 필드가 1로 설정되어 있으면 이 SL 패킷에서 AU가 시작됨을 나타낸다. 만약 이 구문 요소가 SL 패킷 헤더에서 빠져있으면 다음과 같은 규칙에 따라 이전 SL 패킷의 값이 기본 설정값으로 된다.

accessUnitStartFlag=(이전 SL 패킷의 accessUnitEndFlag==1) ? 1: 0

**accessUnitEndFlag** - 이 필드가 1로 설정되어 있으면 이 SL 패킷에서 AU가 끝남을 나타낸다. 만약 이 구문 요소가 SL 패킷 헤더에서 빠져있으면 다음과 같은 규칙에 따라 이후 SL 패킷의 값이 기본 설정값으로 된다.

accessUnitEndFlag=(이후 SL 패킷의 accessUnitStartFlag==1) ? 1: 0

만약 accessUnitStartFlag 혹은 accessUnitEndFlag가 SL 패킷 헤더에 설정되어 있지 않으면, 이것은 각 SL 패킷이 단일의 AU에 해당됨을 내포한다. 그리고 accessUnitStartFlag와 accessUnitEndFlag가 모두 1 의 값을 가진다.

**OCRflag** - 1로 설정되었을 때 objectClockReference 항목이 뒤따라옴을 나타낸다. 기본 설정값은 0이다.
**idleFlag** - 이 기초비트열이 결정되지 않은 시간동안 데이터를 제공하지 않는 휴지(idle)상태임을 나타낸다.
**paddingFlag** - 이 SL 패킷에 padding의 존재여부를 나타낸다. paddingFlag에 대한 기본 설정값은 0이다.
**packetSequenceNumber** - 만약 존재한다면, 이 값은 각 SL 패킷에 대해서 모듈로 계수기로 연속적으로 증가해야 한다. 복호화기에서 이 값의 불연속은 하나이상의 SL 패킷을 잃어버렸음을 의미한다.
**objectClockReference** - Object Clock Reference 타임스탬프를 포함한다. OTB(Object Time Base) 시간 t는 다음 식에 따라 OCR 타임스탬프로부터 재구성된다.

$$t = (objectClockReference / SL.OCRResolution) + k*(2^{SL.OCRLength} / SL.OCRResolution)$$

여기서 k는 objectClockReference 계수기가 몇 번 돌았는지의 수를 나타낸다.

objectClockReference는 OCRflag가 설정되었을 때만 존재한다.

**randomAccessPointFlag** - 1로 설정되었을 때, 이 기초비트열의 컨텐트에 대한 임의 접근이 여기에서 가능함을 나타낸다. randomAccessPointFlag는 accessUnitStartFlag가 설정되었을 때만 존재한다.
**AU_sequenceNumber** - 만약 존재한다면, 각 AU에 대해서 모듈로 계수기로 연속적으로 증가해야 한다. 복호화기에서 이 값의 불연속은 하나이상의 AU를 잃어버렸음을 의미한다.
**decodingTimeStampFlag** - 이 패킷에 decodingTimeStamp가 존재함을 나타낸다.

**compositionTimeStampFlag** - 이 패킷에 compositionTimeStamp가 존재함을 나타낸다.
**accessUnitLengthFlag** - 이 패킷에 존재하는 이 AU의 길이를 나타낸다.
**instantBitrateFlag** - 이 패킷에 instantBitrate가 존재함을 나타낸다.
**decodingTimeStamp** - 연관된 SLConfigDescriptor에 설정된 decoding 타임스탬프 값이다. 이 AU의 decoding time $t_d$는 다음 식에 따라 이 타임스탬프로부터 재구성된다.

$t_d$ = (decodingTimeStamp/SL.timeStampResolution + k*$2^{SL.timeStampLength}$/SL.timeStampResolution

여기서 k는 decodingTimeStamp 계수기가 몇 번 돌았는지의 수를 나타낸다.

**compositionTimeStamp**-연관된 SLConfigDescriptor에 설정된 composition 타임스탬프 값이다. 이 AU의 재구성 결과인 첫번째 CU의 composition time tc는 다음 식에 따라 이 타임스탬프로부터 재구성된다.

$t_c$ = (compositionTimeStamp/SL.timeStampResolution + k*$2^{SL.timeStampLength}$/SL.timeStampResolution

여기서 k는 compositionTimeStamp 계수기가 몇 번 돌았는지의 수를 나타낸다.
**accessUnitLength** - AU의 길이를 바이트 단위로 나타낸 것이다.
**instantBitrate**-이 기초비트열에서 다음 instantBitrate필드가 나올 때까지의 순시 비트율이다.
**degradationPriority** - 이 AU 페이로드의 중요도를 나타낸다. streamPriority는 기초비트열의 기본 중요도를 정의하고, degradationPriority는 기본 중요도(priority)와 이 AU와의 상대적인 감소값을 정의한다. 이 AU에 대한 priority는 다음 식과 같이 주어진다.

accessUnitPriority = streamPriority - degradationPriority

## 3. 결론

본 기고서에서는 MPEG-4 시스템의 전반적인 내용을 기술하였다. MPEG-4 시스템 버전 1은 표준화가 완료되었고 현재는 확장 버전의 표준화가 진행 중이다. MPEG-4 시스템은 세부항목 중에서 구현 시에 조건이 필요 없는 터미널의 작동에 대한 것만을 명시하므로 구현자에게 실제로 여러 가지 방법으로 MPEG-4 터미널과 복호화기를 구현하는 가능성을 제공하고 있다. 또한 양방향 수신이 불가능한 TV 수신기에서부터 완전한 양방향 통신이 가능한 컴퓨터까지를 목적으로 한 것이다. 또한 일부 장치는 MPEG-4 비트열을 등시 네트워크(Isochronous network)에서 수신하고, 다른 장치는 MPEG-4 비트열을 인터넷과 같은 비등시성 수단에서 수신할 것이다. 즉, 이전의 다른 어떤 표준보다도 많은 다양성 및 유연성을 제공한다.

MPEG-4 시스템 표준은 비쥬얼 표준, 오디오 표준, SNHC 표준, DMIF 표준을 총괄한다고 볼 수 있으며 개별 부분의 작업이 모두 완료된 후에야 비로소 가치를 가지는 부분이다. 현재 MPEG-4 표준의 모든 부분들은 1차 버전이 완성되었으므로 머지 않아 이 방식을 적용한 여러 가지 제품군이 경쟁적으로 개발될 것이고 이에 대한 관심이 증가될 것이다. MPEG-4 시스템은 자연영상 부호화뿐만 아니라 가상 공간상에서 2차원 및 3차원 인공영상 부호화, 여러 언어 문자 부호화, 그래픽스, 그리고 이들을 결합한 여러 가지 다양한 화면 구성이 가능하기 때문에 미래의 멀티미디어 통신 서비스는 우리의 상상을 초월한 상품도 창출될 것이며 세계적으로 널리 퍼져있는 인터넷망을 통해서 동시 다발적으로 전파될 것이다.

### 참 고 문 헌

[1]I SO/IEC 14496-1:1999, Information Technology Coding of audio-visual objects - Part 1: Systems.
[2] ISO/IEC 14496-2:1999, Information Technology Coding of audio-visual objects - Part 2: Visual.
[3] ISO/IEC 14496-3:1999, Information Technology Coding of audio-visual objects - Part 3: Audio.
[4] ISO/IEC 14496-6:1999, Information Technology Coding of audio-visual objects - Part 6: Delivery Multimedia Integration Framewo-rk. (DMIF).