ICS 35.040
CCS L71

团 体 标 准

T/AI 114—2021

信息技术 高效多媒体编码 第6部分：智能媒体传输

Information technology—Efficient media coding—
Part 6: Smart Media Transport

2021-10-21 发布 2021-10-21 实施

中关村视听产业技术创新联盟 发布

目　次

前　言

《信息技术　高效多媒体编码》包括9个部分：

——第1部分：系统；

——第2部分：视频；

——第3部分：音频；

——第4部分：符合性测试；

——第5部分：参考软件；

——第6部分：智能媒体传输；

——第7部分：图像容器；

——第8部分：互联网图像；

——第9部分：制作域。

其中，第2部分、第4部分已经发布。第1部分主要规定了用于HTTP上的动态自适应流媒体传输的媒体呈现描述格式与片段格式以及内容安全，第2部分主要规定了适应多种比特率、分辨率和质量要求的高效视频压缩方法的解码过程，第3部分主要规定了高质量音频信号的通用音频编码、无损音频编码和三维音频对象编码的表示方法以及通用音频解码、无损音频解码和三维音频对象解码的方法。

本文件为《信息技术 高效多媒体编码》的第6部分。

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由数字音视频编解码技术标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：上海交通大学、三星电子、北京大学、北京工业大学、中兴通讯、上海大学、深圳龙岗智能视听研究院、广东博华超高清创新中心有限公司、中关村视听产业技术创新联盟。

本文件主要起草人：徐异凌、吴越、张行功、牟伦田、黄成、张文军、陈浩、姜志乾、谢绍伟、胡颖、许健伟、常勇健、黄巍、朱文婕、陈杲、刘利、郭宗明、赵海武、赵海英、黄铁军、张伟民、龙仕强、高文。

引　言

本文件的发布机构提请注意，声明符合本文件时，可能涉及6.3、6.4、7.3、7.4、8.1、8.2、8.3、8.4、8.7、8.8、8.9、8.12、8.13、9.3、10.3、10.4、10.5、10.6、11.1、11.2、附录A中如下26项与数字视频编解码技术相关的专利的使用。专利申请号及名称如下：

序号	专利申请号	专利名称
1	201510401550.9	一种多媒体内容分级技术的实现方法
2	201510064427.2	一种在异构网络终端自适应地调整缓存窗口大小的方法
3	201510698388.1	一种异构媒体网络传输下动态提供资源可获取时间的方法
4	201610916013.2	一种基于异构网络的多媒体资源同步推送方法
5	201610107748.0	一种多媒体系统中信息交互机制及网络传输方法
6	201510673115.1	一种基于媒体内容的FEC机制
7	201510673091.X	一种基于媒体内容的自适应FEC机制
8	201610060847.8	多媒体服务中内容组件关系的描述及个性化显示方法
9	201610056411.1	一种基于媒体自身属性以支持空间分块的存储与传输机制
10	201610674633.X	一种面向多媒体内容组件个性化呈现的方法及系统
11	201510955611.6	一种关联多媒体内容个性化呈现信息的描述方法
12	201610915990.0	一种基于广播系统的媒体点播模式控制方法
13	201710027492.7	一种基于广播系统的媒体点播服务控制方法
14	201611141843.9	一种沉浸式媒体中的交互反馈机制
15	201610757375.1	异构网络下基于网络状况的多媒体资源自适应同步方法
16	201610916013.2	一种基于异构网络的多媒体资源同步推送方法
17	201610081443.7	媒体数据传输方法及装置
18	201280029677.7	用于在多媒体系统中发送/接收媒体内容的方法和装置
19	201480042072.0	用于支持下载和流传输的分组传输的方法和设备
20	201380025465.6	用于多媒体传输系统的收发数据的方法和装置
21	201380053506.2	用于在混合网络中传送和接收多媒体数据的装置和方法
22	201280061956.1	用于在广播系统中配置控制消息的装置和方法
23	201280014043.4	用于在广播系统中配置控制消息的装置和方法
24	201380035064.9	提供多媒体内容的方法
25	201380053098.0	用于媒体数据递送控制的方法和装置
26	201480022346.X	用于在多媒体传输系统中发送媒体数据的方法

本文件的发布机构对上述专利的真实性、有效性和范围无任何立场。

上述专利持有人已向本文件的发布机构保证，愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。上述专利持有人的声明已在本文件的发布机构备案，相关信息可以通过以下联系方式获得：

联　系　人：黄铁军（数字音视频编解码技术标准工作组秘书长）

通讯地址：北京大学理科2号楼2641室
邮政编码：100871
电子邮件：tjhuang@pku.edu.cn
电话：+8610-62756172
传真：+8610-62751638
网址：http://www.avs.org.cn

请注意除上述专利外，本文件的某些内容仍可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

信息技术 高效多媒体编码 第6部分：智能媒体传输

1 范围

本文件规定了用于异构包交换网络下多媒体数据传输的智能媒体传输技术，涵盖数据模型、数据传输、信令、媒体呈现、HTTP/2 直播以及自适应前向纠错编码等内容，适用于网络流媒体、网络电视和视频点播等应用。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 18793-2002 信息技术 可扩展置标语言(XML)1.0

GB/T 18030-2005 信息技术 中文编码字符集

ISO/IEC 23009-1 基于 HTTP 的动态自适应流媒体 [Dynamic adaptive streaming over HTTP (DASH)]

ISO/IEC 14496-12 信息技术 音视频对象的编码 第 12 部分：ISO 基本媒体文件格式 [Information technology – Coding of audio-visual objects – Part 12: ISO base media file format (ISO BMFF)]

IEEE 1003.1-2008 信息技术 便携操作系统接口 [Information technology - Portable Operating System Interface (POSIX)]

IETF RFC 1738 统一资源定位符 [Uniform Resource Locators (URL)]

IETF RFC 2141 统一资源名称语法 [Uniform Resource Names (URN) Syntax]

IETF RFC 3406 统一资源命名空间定义机制 [Uniform Resource Names (URN) Namespace Definition Mechanisms]

IETF RFC 3986 统一资源标识符：通用语法 [Uniform Resource Identifier (URI): Generic Syntax]

IETF RFC 4122 通用唯一标识符 URN 命名空间 [A Universally Unique Identifier (UUID) URN Namespace]

IETF RFC 6570 统一资源标识符：模板 [Uniform Resource Identifier (URI): Template]

IETF RFC 7231 超文本传送协议：语义和内容 [Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content]

3 术语和定义

下列术语和定义适用于本文件。

3.1

访问单元 access unit

含时间信息的最小媒体数据实体。

3.2

媒体资源 asset

任何与唯一标识符相关的用作构建一个多媒体呈现的多媒体数据实体。

3.3

通用封装单元 common encapsulation unit

与媒体编解码器无关的可独立解码的时序或非时序数据的通用容器。

3.4

媒体内容 media content

媒体数据中包含的具有相同时间基准的音频、视频、字幕等信息。

3.5

非时序数据 non-timed data

没有用于解码和/或呈现媒体内容的内在时间轴的媒体数据。

3.6

数据包 package

使用本文件协议传送的媒体数据逻辑单元。

3.7

包 packet

使用本文件协议传送的媒体数据的格式化单元。

3.8

包流 packet flow

有相同发送和接受实体的数据包序列。

3.9

负载 payload

使用本文件协议或者一个网络应用层传送协议携带数据包或者信令消息的媒体数据的格式化单元。

3.10

接收实体 receiving entity

用于接收和消费媒体数据的实体。

3.11

发送实体 sending entity

将媒体数据发送给一个或多个接收实体的实体。

3.12

智能媒体传输协议　smart media transport protocol

用于在 IP 网络上传输有效载荷的应用层传输协议。

3.13

时序数据　timed data

拥有用于解码和/或呈现媒体内容的内在时间轴的媒体数据。

4　缩略语

下列缩略语适用于本文件。

AL-FEC	应用层前向纠错	（Application Layer Forward Error Correction）
AU	访问单元	（Access Unit）
CEU	通用封装单元	（Common Encapsulation Unit）
CRI	时钟相关信息	（Clock Relation Information）
DASH	基于 HTTP 的动态自适应流媒体	（Dynamic Adaptive Streaming over HTTP）
DCI	设备性能信息	（Device Capability Information）
HRBM	假设接收机缓冲模型	（Hypothetical Receiver Buffer Model）
HTTP	超文本传送协议	（Hypertext Transfer Protocol）
ISO BMFF	ISO 基媒体文件格式	（ISO Base Media File Format）
MIME	多用途互联网邮件扩展类型	（Multipurpose Internet Mail Extensions）
MFU	媒体最小分割单元	（Media Fragment Unit）
MP	媒体呈现	（Media Presentation）
MTU	最大传输单元	（Maximum Transmission Unit）
NTP	网络时间协议	（Network Time Protocol）
PA	包访问	（Package Access）
PID	包标识符	（Packet Identifier）
RAP	随机访问点	（Random Access Point）
RTP	实时传输协议	（Real-time Transport Protocol）
SMT	智能媒体传输	（Smart Media Transport）
SMTP	智能媒体传输协议	（SMT Protocol）
TCP	传输控制协议	（Transmission Control Protocol）
TS	传输流	（Transport Stream）
UDP	用户数据报协议	（User Datagram Protocol）
URI	统一资源标识符	（Uniform Resource Identifier）
URL	统一资源定位器	（Uniform Resource Locator）
URN	统一资源名称	（Uniform Resource Name）
UTC	协调世界时	（Coordinated Universal Time）
UUID	通用唯一标识符	（Universally Unique Identifier）

XML　　可扩展置标语言　　（Extensible Mark-up Language）

5　架构

本条定义了智能媒体传输的架构。

SMT 协议为面向包交换的应用层协议，旨在为异构网络中的媒体数据提供传输服务。异构网络即单向物理网络（如数字广播网络）和双向物理网络（如 IP 网络）组成的混合网络。媒体数据包括时序媒体数据（如视频，音频）、非时序媒体数据（如文本、图片）、以及用户反馈数据（如实时指令）。SMT 协议基于 IP 协议，能动态地自适应于异构网络中的不同网络条件。

SMT 协议从逻辑上可以分为三个逻辑功能区，分别为封装功能区、传送功能区和信令功能区，见图 1。

封装功能区定义内容和服务的逻辑组织结构，实现了数据内容和数据描述的分离以及数据碎片化（详见 6.4）。以此形成了媒体内容的分布式布置，能够依靠数据描述形成的内容关联，完成服务的灵活组织和动态配置。

传送功能区定义异构网络下多媒体内容的传输，包括对多媒体数据包的封装与流式传送(详见 7.4)。传送功能区支持媒体数据的重要性分级、数据存储格式与传输格式的快速转换、数据内容的摘要和检索（如音视频指纹，检索特征向量等），同时加入应用层纠错保护机制，建立用于处理时延和抖动的接收端缓冲模型，以自适应于网络状态的变化。

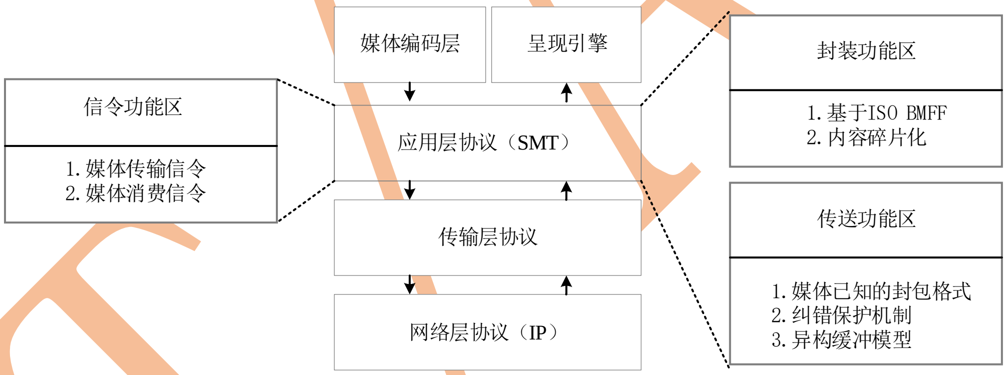


图1　SMT 架构

信令功能区定义用于控制媒体内容消费和传送两方面的信息格式（详见8.2），能根据需求定制信令消息以实现动态配置服务、多种QoS（Quality of Service）类型、网络时钟的同步以及多屏设备之间的交互等，同时能够保证客户端实时反馈和超低延时控制。

6　数据模型

6.1　概述

本条介绍了 SMT 协议数据的逻辑结构。SMT 协议提供了媒体数据的流式传输和存储式传输。在传输过程中，SMT 协议用信令消息保护数据模型。为满足 SMT 内容灵活组织的技术需求，SMT 服务中内容的关联关系由描述文件指定，服务的改变不需要对数据流层级进行更改，只需要更新描述文件。同时为适配不同的网络环境，需要有单独的媒体传输控制信息。在异构网络中，传输控制信息能够根据网

络环境的变化动态更新。分离信令消息与媒体数据，保证了内容自组织的动态灵活性，提供了更好的跨层优化。

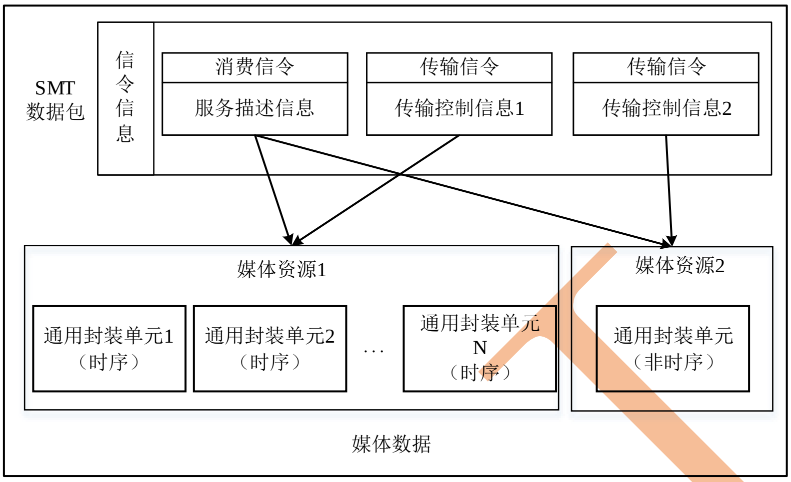


图2　数据模型

6.2　数据包

数据模型见图 2。SMT 数据包是一个逻辑实体，可以看作一种服务，它主要由信令描述文件（详见第 8 章）和媒体资源组成。信令文件包括前向信令和反馈信令，可以分为两种类型：一种是消费信令，一种是传输信令。消费信令主要包含该服务的描述信息，如媒体资源的构成、存放位置、类型、呈现策略等；传输信令主要包含传输过程中的控制信息，如 QoS 参数、缓冲区设置信息等。媒体资源也可分为两种类型：一种是时序的媒体，如视音频内容；另一种是非时序的媒体，如文本、图片信息。为支持异构网络条件下媒体资源的有效传输，以及传输过程中内容动态的配置，设计 SMT 媒体资源的通用封装单元，该单元应具有碎片化、自包含性、统一性等特点，从而支持内容动态组织和传输动态适配的需求。

6.3　媒体资源

一个媒体资源指的是建立多媒体呈现所用到的任意多媒体数据，是封装了编码媒体数据且具有相同媒体资源标识符的内容碎片的逻辑集合，其数据结构见图 3。内容碎片命名为通用封装单元（CEU），其包含的编码媒体数据可以是时序的，也可以是非时序的。时序数据是指有内在时间轴的编码媒体数据，要求数据单元在指定时间同步解码并呈现。相对的，非时序数据指的是在解码并呈现媒体内容时，没有内在时间轴的数据类型。也就是说，非时序数据中每个数据单元的解码及呈现未必要和该数据中的其他数据单元相互依赖。

同一媒体资源中，包含时序数据的 CEU 之间在呈现时间上不能有任何重叠。

一个独立媒体资源的媒体数据类型可以是音频、视频或者网页等。Edit_list 定义描述多媒体文件与不同版本的关联内容的映射关系，并标注出该 Edit_list 包含的媒体数据单元的标识号。属于同一媒体资源的不同 Edit_list 层级可以包含完全不同的媒体数据单元，也可含有相同的媒体数据单元。Edit_list 需要信令的支持，相关信令见 MP 表的 MUR_descriptor。

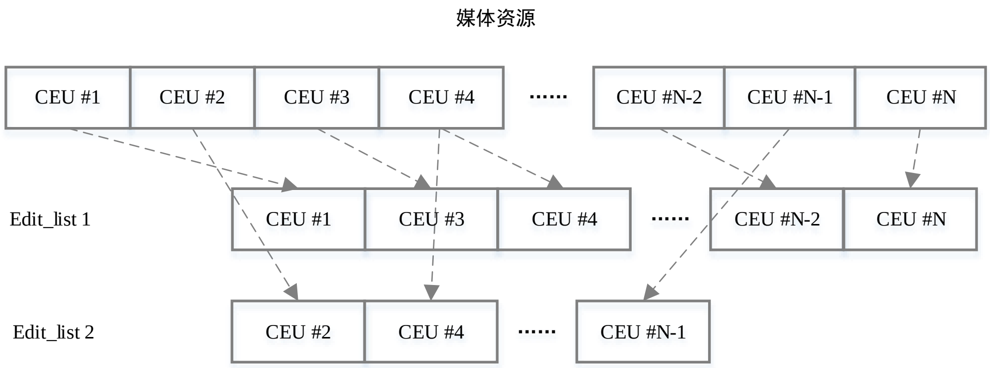


图3　媒体资源的数据构成

6.4　通用封装单元

6.4.1　概述

通用封装单元是一个根据 6.4.2 规则产生的符合 ISO BMFF 的文件。Asset 标识、CEU 序列号以及相关信息由‘cceu’ 盒子提供，以明确地标识出封装进 CEU 文件中的媒体数据。‘moov’盒子包含所有编码器配置信息，以解码和呈现媒体数据。

时序媒体数据作为 ISO BMFF 的轨道存储，CEU 中允许单媒体轨道。非时序媒体作为 ISO BMFF 的元数据的部分存储。图 4 描绘了两个 SMT 封装的例子，一个是时序媒体，另一个是非时序媒体。对于封包化的 CEU 传送，SMT 提示轨道提供将封装的 CEU 转化成 SMTP 负载和 SMTP 包的信息。

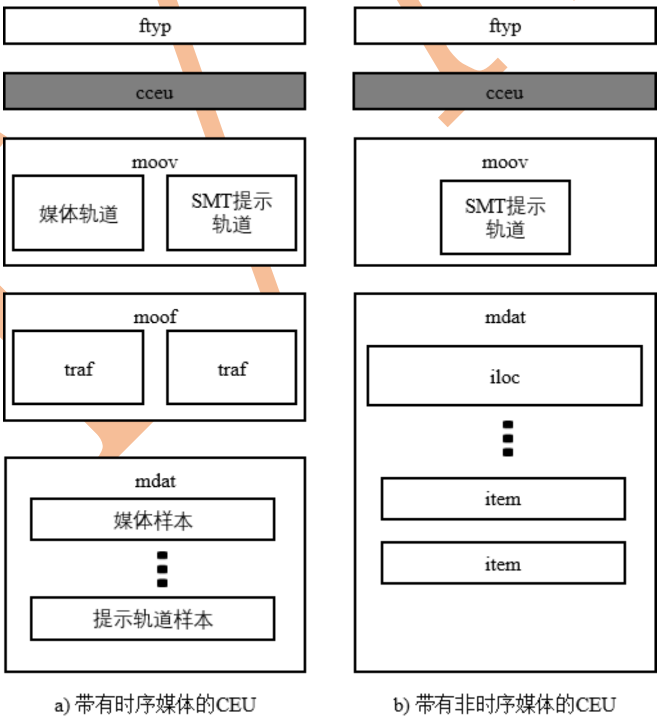


图4　CEU 封装结构

6.4.2　CEU 标签定义

本条定义的标签‘ceuf’(CEU file)确定了遵从 CEU 封装规则的文件。‘ceuf’标签需要‘isom’标签的支持。对其他如‘dash’标签（参见 ISO/IEC 23009-1）的支持也可以另外说明。

一个 CEU 文件由一组使 CEU 自包含的元数据盒子组成。一个 CEU 文件应包含一个‘ftyp’，‘cceu’，‘moov’盒子，一个可选‘sidx’盒子，以上所有都是 CEU 的元数据一部分。其他盒子也被允许，但如果分析程序不识别它们则会被忽略掉。

‘moov’盒子应最多包含一条媒体轨道，且可以包含用于标识传输格式中媒体最小分割单元的 SMT 提示轨道。‘moov’盒子中的轨道应不包含媒体帧以保证开销较小（就是说‘stts’，‘stsc’和‘stco’盒子中的 entry_count 应被设成 0）。存储时序媒体数据 CEU 的文件中，‘moov’盒子应包含‘mvex’盒子，以指示使用了 moof 盒子结构。‘mvex’盒子也设定了以后 moof 盒子的轨道和样本的默认值。

此外，一个‘cceu’盒子应出现于文件级别，且应用如下规则，包含如何决定盒子们的顺序。

a) 如果出现，‘cceu’盒子应紧跟‘ftyp’盒子放置其后；
b) 对于时序媒体数据，零个或更多‘sidx’盒子可以在文件中出现。如果出现，它们应索引构成当前 CEU 的 moof 盒子。

除了盒子顺序，如下限制也应用于‘ceuf’标签。

a) 文件中独立媒体轨道的最大数量为 1（如，空的‘tref’盒子）。并且，非空‘tref’盒子的轨道（如提示轨道）也可用；
b) 对时序媒体数据，文件应至少包含一个 moof 盒子；
c) 对非时序媒体数据，‘meta’盒子应出现于文件级别且应包含 CEU 的非时序媒体元；
d) 如果出现，编辑列表盒子（‘elst’）应仅提供初始偏移；
e) 样本数据序列应以解码顺序放置于‘mdat’盒子，并且两者间无任何其他数据；
f) 任何样本辅助数据，如‘saio’和‘saiz’描述，应放置在‘mdat’盒子开始，所有样本数据之前；
g) 任何提示数据应放置于‘mdat’样本数据之后（或者样本数据之后的另一个 mdat），以使传输前后不改变样本偏移。

‘tfdt’盒子应在每个 moof 盒子的‘traf’盒子内部，以提供该片段按照解码顺序第一个样本的的解码时间。

如果任何‘elst’盒子可用，则它指示的偏移应适用于该 CEU 中依照呈现顺序的第一个样本的合成时间，和任何呈现信息提供的呈现时间。

时序媒体数据作为 ISO BMFF 的一条轨道存储，被‘moov’和‘moof’盒子索引，支持反相兼容。SMT 提示轨道指导 SMT 发送实体将封装的 CEU 转化成封包化的媒体流以采用诸如 SMT 的传输协议来传送。

非时序媒体数据作为元数据项目存储在‘meta’盒子里面。‘meta’盒子应出现在文件层级。每个非时序媒体数据文件应作为单独项目分别存储在‘meta’盒子里。非时序媒体的进入点应被标记为‘meta’盒子的主要项目。（参见 14496-12）

6.4.3 CEU 盒子

6.4.3.1 概述

- 盒子类型：‘cceu’
- 容器：文件
- 强制性：是
- 数量：一或多个

通用封装单元(‘cceu’)盒子包含了当前 CEU 所属 Asset 的 Asset 标识和当前 CEU 的其他属性信息。Asset 标识可全局性无歧义地标识 Asset。CEU 信息包含该 CEU 在 Asset 中的序号以及相关属性信息。

6.4.3.2 语法

```
aligned(8) class CEUBox
extends FullBox('cceu', version, 0){
    unsigned int(1) is_complete;
    unsigned int(7) reserved;
    unsigned int(32) ceu_sequence_number;
    AssetIdentifierBox();
}

aligned(8) class AssetIdentifierBox {
    unsigned int(32) asset_id_scheme;
    unsigned int(32) asset_id_length;
    unsigned int(8) asset_id_value[asset_id_length];
}
```

6.4.3.3 语义

is_complete：指示该 CEU 是否包含了 MFU 结构中描述的所有 MFU。

ceu_sequence_number：当前 CEU 的序列号。媒体资源（Asset）中的第一个 CEU 序列号为 0，之后的 CEU 序列号依次递增。此序列号在一个媒体资源中是唯一的。

asset_id_scheme：区分用来表示Asset ID的策略，决定了asset_id_value的类型。有效的策略见表1。

表1　asset_id_scheme 取值列表

取值	描述
"UUID"	UUID (通用唯一标识符)
"URI"	URI (统一资源标识符)

asset_id_length：asset_id_value 的长度。

asset_id_value：即媒体资源的标识符。其取值格式依赖于 asset_id_scheme 的类型。

6.4.4 SMT 提示轨道

6.4.4.1 概述

出于传送的目的，SMT 提示轨道为 SMT 发送实体提供将 CEU 分解(或分割)为传输格式中媒体最小分割单元的信息。该最小分割单元是媒体已知的，且被用来搭建 SMTP 的负载，即 CEU 中的媒体数据在传送时被 SMT 发送实体提取进 SMTP 负载。

SMT 提示轨道也提供了从 SMTP 负载中提取和重建 CEU 的信息。SMTP 负载可以包含 CEU 元数据、分片元数据、或是一个或多个传输格式中的最小分割单元。CEU 元数据可以包含'ftyp', 'sidx', 'cceu', 和'moov'盒子。

6.4.4.2 提示轨道样本属性

6.4.4.2.1 语法

```
aligned(8) class SMTHintSampleEntry() extends SampleEntry('smth') {
```

```
        unsigned int(16) hinttrackversion = 1;
        unsigned int(16) highestcompatibleversion = 1;
        unsigned int(16) packet_id;
        unsigned int(1) is_fragment;
        unsigned int(1) is_timed;
        unsigned int(6) reserved;
    }
```

6.4.4.2.2　语义

packet_id：指示该提示轨道应用于哪一个 Asset 的唯一标识符。

is_fragment：指示 CEU 是否切分为 MFU 的标识。若该标识位置‘0’，该提示轨道应用于完整的 CEU，即每个轨道片段（track fragment）对应于一个媒体片段；否则，每个提示样本应用于单个 MFU。

is_timed：指示该轨道提示的媒体数据是否是时序的。

6.4.4.3　提示轨道样本格式

6.4.4.3.1　定义

每个媒体样本被划分为一个或多个 MFU。每个 SMT 提示轨道样本对应一个或多个 MFU。

6.4.4.3.2　概述

```
aligned(8) class SMTHSample {
    unsigned int(32)     sequence_number;
    if (is_timed) {
        signed int(8)        trackrefindex;
        unsigned int(32)     movie_fragment_sequence_number;
        unsigned int(32)     samplenumber;
        unsigned int(8)      priority;
        unsigned int(8)      dependency_counter;
        unsigned int(32)     offset;
        unsigned int(32)     length;
    }
    else {
        unsigned int(16)     item_ID;
    }
}
```

6.4.4.3.3　语义

sequence_number：指示该 MFU 在 CEU 中序列号的整数值。CEU 中序列号的断续是允许的，用于指示特定 MFUs（其序列号在序列中缺失）在 CEU 组包后未被处理。由下层网络实体完成传送和缓冲是处理 MFU 的一个例子。

trackrefindex：所描述的媒体对应的 trackID。

movie_fragment_sequence_numbe：该 MFU 中媒体数据所属媒体片段的序列号。

samplenumber：该 MFU 提取自媒体样本的编号。样本编号 n 指向当前媒体片段累计媒体样本中的第 n 个。媒体样本中第一个样本的样本编号置为‘1’。

priority：指示 CEU 中该 MFU 相对于其它 MFU 的优先级。

dependency_counter：指示依赖于该 MFU 解码的 MFU 的个数。该字段的值等于按 sequence_number 排列的可能无法正确解码的后续 MFU 的个数。例如，该字段的值为 n，则若无此 MFU 后续有 n 个 MFU 可能无法正确解码。

offset：该 MFU 中媒体数据偏移。偏移基准为包含‘mdat’盒的开始。MFU 应放置在偏移指示的位置。

length：该 MFU 中数据的长度，以字节为单位。

item_ID：对非时序媒体数据，这是该 MFU 包含该项目的标识。

7 数据传输

7.1 概述

本条定义SMT负载格式，规定数据包在应用层的封包传输协议。

SMT负载格式被定义为对包内容封包的通用负载格式。SMT负载格式对用于编码媒体数据的特定编解码器不可知，因此任何封装成CEU的媒体数据可被封包为SMTP负载，其支持媒体数据流传输的应用层传输协议。SMT负载可用于RTP协议、SMT协议以及其它传输协议。SMT负载亦可用于封包信令消息。

SMT协议定义应用层传输协议，支持基于数据包异构网络（包括IP网络环境）中数据包流传输。SMT协议提供包传输的必要特性，例如使不同媒体资源通过一个SMT包传输的协议级复用，以及独立于呈现时间的传输时间模型以适应大范围网络抖动。

7.2 传输模型

SMT传输模型能够解决多方面问题，如不同网络通道QoS参数各异情况下如何保证数据的可靠传输，异构网络条件下如何设计终端缓存模型使多源媒体数据同步呈现，以及当网络条件糟糕时提供基于数据重要等级的保护机制差异化等。

为支持SMT传输需求，提出图5的传输模型。SMT信令文件和传输包均根据媒体服务逻辑包进行组织，媒体组织和传输都与媒体服务相关，有利于根据不同服务类型差异化实现媒体内容组织和传输的自适应性。

SMT逻辑包可以序列化为SMT文件，支持媒体文件式的存储、传输和下载；也可以打包为SMT传输包，以支持媒体的流化传输。由于文件格式和传输包格式内容上的高相关性，SMT支持两者的便捷转换，便于中继转发，同时传输模型也可以高效响应服务内容的动态配置。传输过程中服务描述和传输控制等信令文件可以采用与媒体内容不同的传输模式，既能满足带外传输需求，又能对信令文件加以更高等级的保护措施。

媒体通用封装单元的碎片化、自包含性也决定了SMT传输模型能够应对传输过程中的包错误问题，实现弹性容错和错误隐藏，支持接收端快速恢复错误。每个通用封装单元及其切片单元都被唯一标识，头部信息与信令消息相结合，指定数据范围、优先级、QoS要求、保护方式、呈现时间等不同层级的参数设置，对于媒体内容的分级、异构、多通道、可靠性传输有重要参考意义。

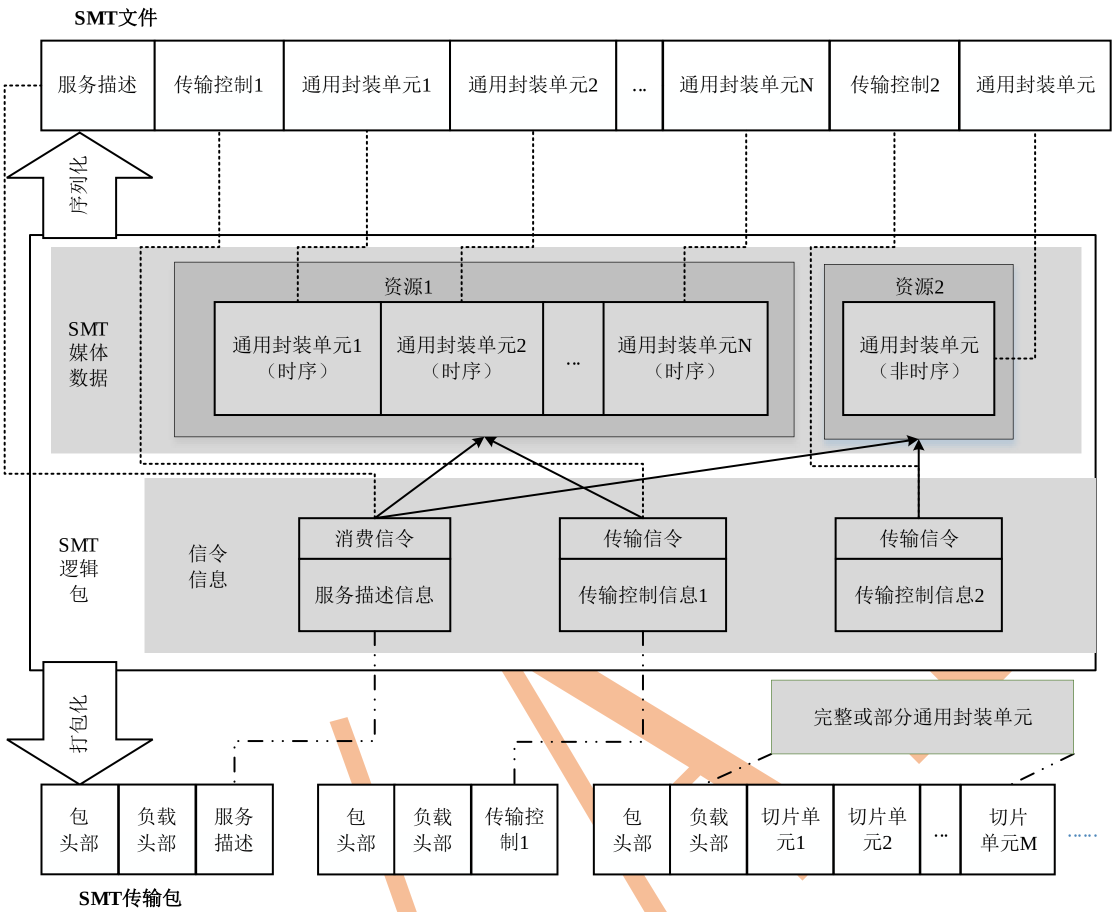


图5　传输模型

7.3　传输协议

7.3.1　基于封装结构的最小分割单元

MFU与CEU的结构关系见图6和图7。包含时序媒体的一个媒体最小分割单元是媒体样本或子样本，而由一个‘moof’盒子和一个‘mdat’盒子构成的每个媒体片段（movie fragment）可以包含一个或多个媒体样本（media sample）。包含非时序媒体的一个MFU是一个项（item）。

每个MFU由一个头文件和相关联的媒体数据组成。MFU头应是MFU提示样本的一个拷贝，媒体数据应是该MFU提示样本索引的媒体数据的拷贝。MFU的提取与重建由SMT提示轨道负责，见6.4.4。

MFU在SMTP包内的SMTP负载中传送。

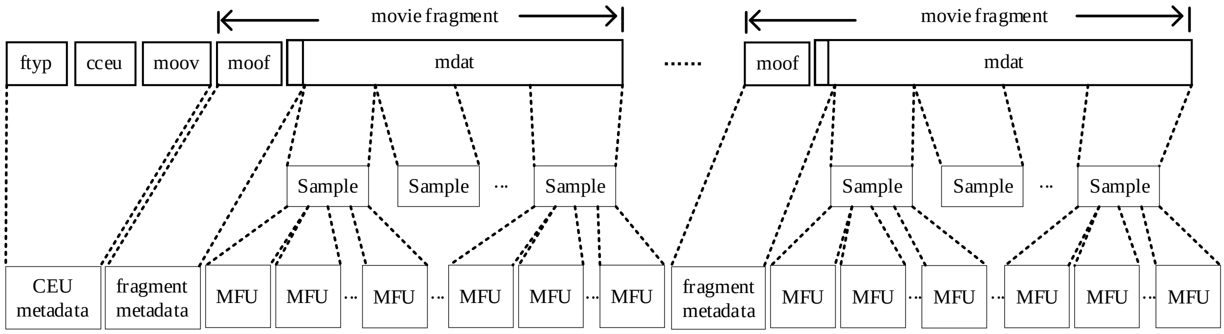


图6　时序媒体中 CEU 与 MFU 关系

图7　非时序媒体中 CEU 与 MFU 关系

7.3.2　SMTP 包

7.3.2.1　概述

SMT 协议为高效、可靠包传输的应用层协议，适用于时序与非时序媒体数据的传输。该协议支持若干增强特性，例如媒体多路复用与网络抖动计算。设计这些特性是为了更高效地传送不同编码类型的媒体数据内容。SMT 协议可运行于现行的网络协议之上，如 UDP 和 IP。

SMT 协议支持不同媒体数据的复用，例如，来自不同媒体资源的多个 CEU 复用到一个 SMTP 流上。在不引入长延迟和大缓存的情况下，SMT 协议按照接收实体媒体数据的消费顺序传送多种类型的数据以帮助不同类型媒体数据之间同步。SMT 协议也支持在单一数据流内，媒体数据和信令消息的复用。一个 SMTP 包内仅能包含一个 SMTP 负载。SMTP 包格式不支持多 SMTP 负载的聚合和单 SMTP 负载的切分。

SMT协议也提供了计算和消除底层传输网络所引入抖动的方法，以实现数据流的恒定延迟。通过使用包头的时间戳字段，可以在不获取额外信息的情况下精确地计算抖动。

7.3.2.2　SMTP 包结构

图 8 描述了 V=0 时 SMTP 包的结构。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=0 | C | FEC | r | X | R | RES |    type     |           packet_id            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     packet_sequence_number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          packet_counter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         header_extension       ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         payload data        ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      source_FEC_payload_ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图8　SMTP 包结构（V=0）

图 9 描述了 V=1 时 SMTP 包的结构，该版本定义了简化的 SMTP 数据包结构，以实现灵活多样的信息交互。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| V=1 | C | FEC | T | X | R | P | F |    type     |           packet_id            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            timestamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     packet_sequence_number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          packet_counter                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         header_extension       ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         payload data        ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      source_FEC_payload_ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图9　SMTP 包结构（V=1）

7.3.2.3　语义

version（V：2 bits）：标识 SMT 协议版本号。

packet_counter_flag（C：1 bit）：该字段置‘1’则使用 packet_counter 字段。

FEC_type（FEC：2 bits）：标识用于纠错保护 SMTP 包的 FEC 方案。该字段有效值见表 2。

表2　FEC_type

值	描述
0	无 AL-FEC 保护的 SMTP 包
1	有 AL-FEC 保护（FEC 数据包）的 SMTP 包
2	带修复符号的 SMTP 包（FEC 修复包）
3	保留以后使用

reserved（v=0，r：1 bit）：保留字段。

timestamp_flag（v=1，T：1 bit）：该字段置‘1’则使用 timestamp 字段。

extension_flag（X：1 bit）：该字段置‘1’则使用 header_extension 字段。

RAP_flag（R：1 bit）：该字段置‘1’时，表示该 SMTP 包负载包含一个该数据类型数据流的随机接入点（RAP）。此标志位的准确含义由数据类型本身定义。当数据单元类型为 CEU 元数据或媒体片段元数据时，RAP_flag 字段需置‘1’。当数据单元类型为时序 CEU 中包含同步信息的 MFU，或非时序 CEU 中包含同步信息的主要项时，RAP_flag 字段需置‘1’。

reserved（v=0，RES:2 bit）：保留字段。

packet_id_flag（v=1，P:1 bit）：该字段置‘1’则使用 packet_id 字段。

fragmentation_flag（v=1，F:1 bit）：该字段置‘1’则使用 packet_sequence_number 字段。

type（6bits）：标识负载数据类型。负载类型值见表 3。

表3　数据类型与数据单元定义

值	数据类型	数据单元定义
0x00	CEU	媒体感知的 CEU 片段
0x01	信令消息	一条或多条信令消息或信令消息的片段
0x02~0x03	为其它数据类型保留	
0x04 ~ 0x3F	私有用途保留	

packet_id（16bits）：该字段为整数值，用于区分不同媒体资源。该字段数值源自该包所属媒体资源的 asset_id 字段。packet_id 与 asset_id 间的映射由信令消息中的 MP 表标识。信令消息与 FEC 修复流分配不同的值。整个传输会话周期同一 SMT 发送实体所有 SMT 流的 packet_id 值是唯一的。对 AL-FEC，packet_id 与 FEC 修复流间的映射由 AL-FEC 信息提供。

timestamp（32bits）：基于 UTC 时间标识 SMTP 包传输瞬时时间。此格式定义于 IETF RFC5905 条款 6，NTP 第四版的‘短格式’。该时间戳表示 SMTP 包第一个字节的发送时间。SMT 发送实体要求能够提供与 UTC 同步的准确时间信息。

packet_sequence_number（32bits）：该字段为整数值，用于区分包含相同 packet_id 的不同包。该字段初始为任意值，每收到一个 SMTP 包该字段的值加 1，超出最大值后返回 0。

packet_counter（32bits）：该字段为整数值，用于计数 SMTP 包。每发送一个 SMTP 包该字段的值加 1，不考虑 packet_id 值。该字段初始为任意值，超出最大值后返回 0。

header_extension：包头扩展机制允许对负载格式进行适当的扩展，这使得要求负载格式头携带额外信息的应用和媒体类型成为可能。包头扩展机制被设计成可在不影响 SMTP 负载正确处理的情况下丢弃。包头扩展格式见图 10。该协议不定义任何具体的包头扩展。

Source_FEC_payload_ID（32bits）：仅当‘FEC_type’字段置‘1’时使用该字段。‘FEC_type’字段置‘1’时 SMTP 包使用 AL-FEC 保护，使用 AL-FEC 保护后该字段应增加至 SMTP。

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              type               |            length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           header__extension__value      ......
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图10　头部扩展结构

type（16bits）：后续头部扩展的唯一标识。

length（16 bits）：表示 header_extension_value 字段长度（字节为单位）。

header_extension_value ：提供扩展信息。该协议不定义该字段格式。

7.3.3　SMTP 会话描述信息

SMTP 会话描述信息可通过不同方式传输至接收实体，以适应不同的配置环境。在加入 SMTP 会话前接收实体需要获知以下信息。

a) 目的地信息，在 IP 网络环境中，目的地信息为 IP 地址与端口号；
b) 指示该会话为 SMTP 会话；
c) 指示该会话为 SMTP 会话；
d) SMTP 会话的开始与结束时间。

7.4　传输负载结构

7.4.1　概述

SMTP包负载是一种通用负载，使用SMT协议封包并携带SMT媒体数据。SMTP包负载可以是一个或多个完整CEU或CEU片段，或者信令消息等。每一种负载类型都有着独立的传输数据单元以及针对该类型负载的负载头。例如，SMTP负载携带CEU片段时CEU片段被视为一个数据单元。SMT协议能够整合多个同类型数据单元到一个SMTP负载中，也能够将数据单元分割至多个SMTP包中。

7.4.2　CEU 模式

7.4.2.1　概述

通过SMT协议传输CEU要求在SMT发送与接收实体分别配置封包与解包程序。封包程序将CEU封包成一组被SMTP包携带的SMTP负载。SMTP负载格式支持SMTP负载分段传输，以使大容量负载可以传输。SMTP负载格式也支持将多个SMTP负载的数据单元整合到一个SMTP负载中，以便于小容量数据单元聚合传输。接收实体解包以恢复原始CEU数据。

7.4.2.2　CEU 模式下 SMTP 负载头

CEU 模式下 SMTP 负载头部结构见图 11。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           length              |  FT   |T| f_i |A| frag__counter |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     CEU_sequence_number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         DU_length             |         DU_Header    ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         DU_ payload      ……
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图11　CEU 模式下 SMTP 负载头部结构

对携带 MFU 的负载，数据单元头部由字段“T”指示其为时序媒体或非时序媒体。时序媒体数据单元头部结构见图 12，非时序媒体数据单元头部结构见图 13。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                movie_fragment_sequence_number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        sample_number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           offset                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    priority   |  dep_counter  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图12　时序媒体数据单元头部

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           item_ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图13　非时序媒体数据单元头部

7.4.2.3　语义

length（16bits）：除此字段外负载长度（单位为字节）。

CEU Fragment Type（FT：4bits）：该字段指示片段类型，有效值见表4。

表4　数据类型及数据单元定义

FT	描述	内容
0	CEU metadata	包含 ftyp，mceu，moov 和其它出现在这之间的元数据盒子。
1	Movie fragment metadata	包含 moof 和除去所有媒体数据的 mdat 盒子。
2	MFU	包含时序媒体数据的样本或子样本，或非时序媒体数据的一个 item。
3～15	私有用途保留	保留。

Timed_Flag（T：1bit）：指示数据单元是时序（该字段置‘1’）媒体或非时序（该字段置‘0’）媒体。

Fragmentation_Indicator（f_i：2bits）：指示负载中数据单元的分片信息。有效值见表5。当此字段置‘00’时有可能设置aggregation_flag字段。

表5　数据单元指示值

值	描述
00	负载包含一或多个完整数据单元。
01	负载包含数据单元的第一个片段。
10	负载包含数据单元的中间片段。
11	负载包含数据单元的最后一个片段。

aggregation_flag（A：1bit）：该字段置‘1’表示负载中包含2个或以上数据单元。

fragment_counter（frag_count：8bits）：该字段指示此SMTP负载后包含同一数据单元片段的SMTP负载个数。当aggregation_flag字段置‘1’时该字段置‘0’。

CEU_sequnece_number（32bits）：CEU片段所属CEU的序号。

DU_length（16bits）：指示该字段后续数据长度。当aggregation_flag置‘0’时，该字段不呈现。当aggregation_flag置‘1’时，该字段呈现次数与整合进负载中的数据单元个数相同，并出现在每个数据单元前面。

DU_header：数据单元头部，仅当FT为2，也即MFU有效时存在，且时序媒体与非时序媒体对应的语义也不同。

DU_payload：数据单元负载。

moive_fragment_sequence_number（32bits）：该MFU媒体数据所属媒体片段的顺序编号。

sample_number（32bits）：该MFU媒体数据所属媒体样本的顺序编号。

offset（32bits）：MFU媒体数据在所属媒体样本内的偏移。

priority（priority：8bits）：相同CEU中MFU媒体数据间的优先级。Priority的值域为‘0’到‘255’，数值越大优先级越高。

dependency_counter（dep_counter：8bits）：指示依赖该MFU媒体数据来进行媒体处理的数据单元个数。

Item_ID（32bits）：作为MFU一部分的item标识符。

7.4.3　信令消息模式

7.4.3.1　概述

SMTP 信令消息模式用于定义信令消息的传输。信令消息可用其它格式编码，如二进制格式或 XML 格式。因此在传输层能够快速访问与过滤信令消息很重要，且在过滤时希望尽量避免解析信令消息。

信令消息负载格式提供分块与整合功能以支持高效封包。

7.4.3.2　信令消息模式下 SMTP 负载头

信令消息模式下 SMTP 负载头部结构见图 14。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| f_i |  res  | H | A |  frag__counter  |   MSG__length (16+16*H)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          MSG_ payload   ……                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图14　信令消息模式下 SMTP 负载头部结构

7.4.3.3　语义

Fragmentation_Indicator（f_i：2 bits）：指示SMTP负载中信令消息的分片信息。有效值见表6。

表6　片段指示值

值	描述
00	负载包含一或多个完整信令消息。
01	负载包含信令消息的第一个片段。
10	负载包含非第一个，也非最后一个部分的信令消息片段。
11	负载包含信令消息的最后一个片段。

RES（res：4bits）：该字段所有位置‘0’，保留以后使用。

H（1bits）：表示指示信令消息长度的附加16位是否使用。

aggregation_flag（A：1bit）：该字段置‘1’表示负载中包含2个或以上信令消息。

fragmentation_counter（frag_counter：8bits）：该字段表示此信令消息片段后包含同一信令消息片段的SMTP负载个数。当aggregation_flag字段置‘1’时该字段置‘0’。

MSG_length（16+16*Hbits）：该字段表示此字段后信令消息长度。当aggregation_flag置‘1’时，该字段出现次数与整合进负载中的信令消息个数相同，并出现在每个信令消息前面。

MSG_playload：信令消息负载。

7.5　基于负载结构的传输操作

7.5.1　一般操作

一个SMTP会话由一个SMTP传送流组成。SMTP传送流的定义是：来自于一个或多个SMT发送实体，被传送到同一个目的地的包流。在IP情况下，包流的目的地是IP地址和端口。

一个单独的数据包可能需要一个或多个SMTP包流进行传输。

一个单独的SMTP传送流中的数据可来自多个数据包。

一个SMTP传送流可携带多个媒体资源，在SMTP会话中每一个媒体资源都与一个唯一的包标识符对应。SMTP提供一个最优化的流格式（CEU格式）。媒体资源作为一系列相关媒体数据被定义为媒体数据流进行传输。媒体数据可能是一个CEU，文件或者信令消息。

SMTP 包子流是一个SMTP包流的包的子集，这些子流共享同一个包标识符。

该媒体数据流作为SMTP子流进行传输。

CEU模式支持CEU的封包流化传输。

SMTP适用于单播和多播。为了保证多播/广播环境的可靠性，SMTP主要依靠FEC而不是将数据包重发。

在加入SMTP会话前，SMT接收实体需获得足够信息以确保被传送数据的接收。最少需要信息见7.3.3。

SMTP需要SMT接收实体能够识别并解复用属于特定媒体数据流的SMTP包。

7.5.2　传输 CEU

7.5.2.1　概述

CEU模式在发送实体和接收实体间传送CEU。

7.5.2.2　SMT 发送实体操作

7.5.2.2.1　时序媒体数据

CEU分包后应携带CEU元数据，或媒体片段元数据，或MFU。产生的包不能够携带超过两种不同类型的数据单元。CEU元数据由‘ftyp’盒子，‘cceu’盒子，‘moov’盒子以及其他适用于CEU的盒子组成。媒体片段元数据由‘moof’盒子和‘mdat’盒子头部组成（除了媒体数据）。携带媒体片段元数据的SMTP负载的FT字段置0x01。‘mdat’盒子中的媒体数据分成一个或多个MFU。携带MFU的SMTP负载的FT字段置0x02。

图15描述了CEU与时序媒体负载的关系。

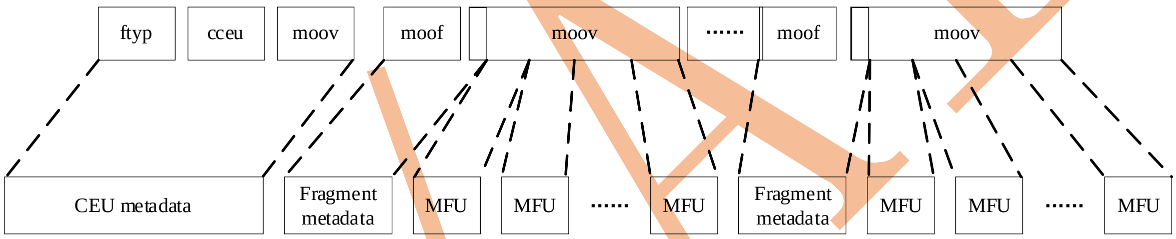


图15　时序媒体的有效负载产生

7.5.2.2.2　非时序媒体数据

CEU分包后应携带CEU元数据或MFU。CEU元数据由‘ftyp’盒子，‘moov’盒子，‘meta’盒子以及其他适用于CEU的盒子组成。携带CEU元数据的SMTP负载的FT字段置0x01。每一个MFU数据单元包括一个非时序媒体的item。携带MFU的SMTP负载的FT字段置0x02。

图16描述了CEU与非时序媒体负载的关系。

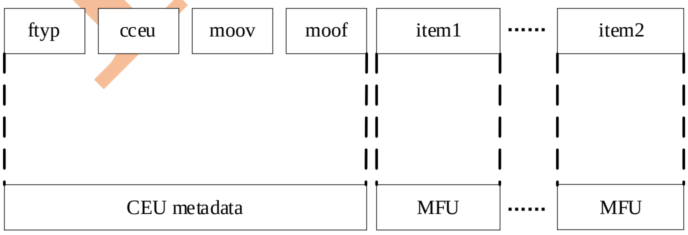


图16　非时序媒体有效负载的产生

7.5.2.3 SMT 接收实体操作

在SMT接收实体上执行解包，重组CEU。根据不同应用场景，可选择但不限于如下几种解包模式：

——CEU模式：在CEU模式下，接收实体重组完整的CEU后再传输给应用程序。该模式适用于时间要求不严格应用场景，例如CEU呈现时间充分滞后于接收时间；

——媒体片段模式：在媒体片段模式下，接收实体重组完整的媒体片段后再传输给应用程序。该模式适用于延时敏感的应用场景，其中延时时间受限但足够恢复一个完整的媒体片段；

——数据单元模式：在数据单元模式中，接收实体解包后立即传输给应用程序。该模式适用于延时非常低的应用场景。该模式支持数据单元的乱序传输。

8 信令

8.1 概述

本条详细说明了SMT的信令功能区。信令功能区中包含了一整套消息格式传达用于传输和消费SMT数据包的必要信令消息，分别称为传输信令和消费信令。本规范详细介绍了承载信令表、描述符或者传输相关信息的消息格式。信令表包含特定信令消息的元素与属性集，也可以包含描述符来携带更多细节信息。

8.2 信令消息格式

8.2.1 概述

SMT信令消息的一般格式包含三个通用字段、一个特殊字段以及一个消息负载。消息负载用来承载信令消息。

一般信令消息格式的语法和语义分别在8.2.2和8.2.3中定义。

8.2.2 语法

表7列出了SMTP信令消息的一般格式的语法。

表7 信令消息一般格式的语法

语法	值	比特位数
signalling_message () {		
message_id		16
version		8
if(message_id!=PA_message){		
length		16
}		
else{		
length		32
}		
extension		
message_payload {		
}		
}		

8.2.3 语义

message_id：信令消息的标识符。

version：信令消息的版本。SMTP发送和接收实体能够验证一个接收到的消息是否有新版本。新消息版本号值较大，SMT实体可以检查接收到的消息的版本号，确定收到消息是否为最新版本。

length：信令消息的长度，对于PA消息来说占4字节，MPT message是两字节。

extension：该字段提供信令消息中需要扩展的信息。该字段的内容和长度由信令消息规定。

message_payload：信令消息的有效负载。该字段的格式能够被message_id字段的值识别。

8.3 媒体数据包消费信令消息

8.3.1 概述

本条详细说明了与SMT数据包消费相关的信令消息。用于媒体数据包消费的信令消息可能包含信令表，其包含一组特定信令消息的元素和属性。各个信令表携带着媒体数据包相关信息，比如服务接入、内部媒体资源属性、呈现信息描述等。信令消息与信令表之间的关系见图17。

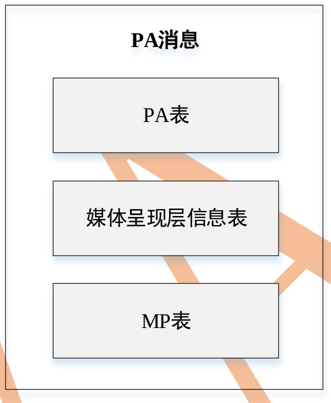


图17 信令消息与信令表之间的关系

由此可知，一个PA消息应当包含一个PA表、一个MP表和一个媒体呈现层信息表。

8.3.2 PA 消息

8.3.2.1 概述

一条PA消息包含一个PA表，该PA表包含所有其他包的信令表。一条PA消息也包含一个MP表，用来传输处理媒体数据包的其他消息。

SMTP 接收实体应当在处理其他信令消息之前处理PA消息。

8.3.2.2 语法

表8定义了PA消息的语法。

表8 PA 消息语法

语法	值	比特位数
PA_message () {		
message_id		16
version		8
length		32
extension {		
number_of_tables	N1	8
for (i=0; i<N1; i++) {		

表 8 （续）

语法	值	比特位数
table_id		8
table_version		8
table_length		16
}		
}		
message_payload {		
for (i=0; i<N1; i++) {		
table()		
}		
}		
}		

8.3.2.3 语义

message_id：指示PA消息的标识。

version：指示PA消息的版本。

length：指示PA消息的长度，以字节为单位。该长度的计算是从下一个字段的开头至PA消息的最后一字节。

number_of_tables：指示PA消息中包含的信令表的数量。

table_id：指示PA消息中包含的表识别符。这是表中包含在PA消息的有效负载中的table_id 字段的一个副本。

table_version：指示PA消息中所包含的表的版本。这是包含在PA消息的有效负载中的表的版本字段的一个副本。

table_length：包含在PA消息的有效负载中的表的长度字段的一个副本。

table()：一个SMTP信令表实体。在有效负载中的该表与扩展域中table_id出现的顺序相同。一个PA表可以作为一个table()的实例。

8.3.3 MPT 消息

8.3.3.1 概述

MPT 消息包含一整个MP表或MP表的子集，MP表的各个子集可以由不同的MPT消息传输。

不同MP表子集有不同的MP表table_id. 从1开始的14个连续的数用来表示MP表的table_id。table_id值为15的表示完整的MP表。

8.3.3.2 语法

表9定义了MPT消息的语法。

表9 MPT 消息语法

Syntax	Value	No. of bits	Mnemonic
MPT_message () {			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
extension {			
}			
message_payload {			
MP_table()			

表 9 （续）

Syntax	Value	No. of bits	Mnemonic
}			
}			

8.3.3.3 语义

message_id：标识 MPT 消息的 ID。该字段为 16 比特。

version：标识 MPT 消息的版本，新消息版本号大于旧消息版本号，SMT 实体可以检查接收到的消息的版本号，确定收到的消息是否为最新版本。

length：标识 MPT 消息的长度，该字段长度为 16 比特。MPT 消息的长度是从长度的下一个字段开始到该消息的最后一个字节的所有字节数。对于该字段来说 0 值是无效的。

MP_table()：MP 表的定义见8.3.5。

8.3.4 PA 表

8.3.4.1 概述

PA 表可以提供用于消费媒体数据包的其他信令表的所有信息。接收端在接入系统时会首先收到 PA 消息，通过 PA 消息解析 PA 表以后可以得到其他信令表的信息。

8.3.4.2 语法

表10定义了PA表的语法。

表10 PA 表的语法

语法	值	比特位数	备注
PA_table() {			
table_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
information_table_info {			
number_of_tables	N1	8	uimsbf
for (i=0; i<N1; i++) {			
signalling_information_table_id		8	uimsbf
signalling_information_table_version		8	uimsbf
location {			
SMT_general_location_info()			
}			
reserved	‘1111111’	7	bslbf
alternative_location_flag		1	bslbf
if (alternative_location_flag == 1) {			
alternative_location {			
SMT_general_location_info()			
}			
}			
}			
}			
reserved	‘1111111’	7	bslbf
private_extension_flag		1	bslbf
if (private_extension_flag == 1)			
private_extension {			
}			
}			
}			

8.3.4.3 语义

table_id：PA表的标识符。

version：PA表的版本。当它一被接收，更新的版本就覆盖旧版。

length：包含了以字节计算的PA表的长度，即从下一字段起直到MP表最后一个字节的长度。‘0’值在此字段无效。

number_of_table：为当前PA表提供信息的表的个数。

signalling_information_table_id：为当前PA表提供信息的表的标识符。

signalling_information_table_version：为当前PA表提供信息的表的版本。

SMT_general_location_info：提供资源位置的信息。

alternative_location_flag：值为‘1’表示接收实体可以获取可替代资源的地址。

SMT_general_location_info_alternative_location：可替代资源的位置信息。

location_type：指示SMT数据流的位置信息类型。

8.3.5　MP 表

8.3.5.1　概述

一个完整的MP表具有关于包含所有媒体资源的媒体数据包的信息。一个子MP表包含部分完整MP表的信息。另外，MP表子集0包含媒体数据包消费所需求的最小信息。

8.3.5.2　语法

表11定义了MP表的语法。

表11　MP 表的语法

语法	值	比特位数	备注
MP_table() {			
table_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
reserved	’11111’	6	bslbf
MP_table_mode		2	bslbf
if (table_id == SUBSET_0_MPT_TABLE_ID) {			
SMTP_package_id {			
SMTP_package_id_length	N1	8	uimsbf
for (i=0; i<N1; i++) {			
SMTP_package_id_byte		8	uimsbf
}			
}			
MP_table_descriptors {			
MP_table_descriptors_length	N2	16	uimsbf
for (i=0; i<N2; i++) {			
MP_table_descriptors_byte		8	uimsbf
}			
}			
}			
number_of_assets	N3	8	uimsbf
for (i=0; i<N3; i++) {			
Identifier_mapping()			
asset_type		64	char
asset_size		32	bslbf
reserved	‘1111111’	7	bslbf
asset_clock_relation_flag		1	bslbf
if (asset_clock_relation_flag == 1) {			

表 11 （续）

语法	值	比特位数	备注
asset_clock_relation_id		8	uimsbf
reserved	‘1111111’	7	bslbf
asset_timescale_flag		1	bslbf
if (asset_time_scale_flag == 1) {			
asset_timescale		32	uimsbf
}			
}			
asset_location{			
location_count	N6	8	uimsbf
for(i=0;i<N6;i++){			
SMT_general_location_info()			
}			
}			
asset_descriptors {			
asset_descriptors_length	N5	16	uimsbf
for (j=0; j<N5; j++) {			
asset_descriptors_byte		8	uimsbf
}			
}			
}			
}			

8.3.5.3 语义

table_id：MP表的标识符。一个完整的MP表及其各子集应采用不同的表标识符。MP表的子集编号由该字段隐形表示。由于table_id的值是连续分配的，因而MP表的子集数量可以从该字段推导得出，即，MP表的子集数量等于该字段减去基础MP表的table_id。MP表的子集编号提供了该MP表的子集编号。数字‘0’ 表示MP基表，数字‘1’ 到‘14’ 表示MP表子集。数字‘15’有特殊含义因为它反映了一个完整的MP表。

version：MP表的版本。当它一被接收，更新的版本就覆盖旧版。如果table_id表示一个完整的MP表，如果子集-0的MP表具有该字段（当MP_table_mode为1时）相同版本值，或者如果所有带有次子集编号的MP表的子集具有该字段（当MP_table_mode为0时）相同版本值，或者如果MP表子集的进程是独立的（当MP_table_mode为2时）。如果MP子集表－0 有一个更新的版本，之前所有储存在SMTP接受实体中的更大的直到数字14的MP表子集都将废弃，除非MP_table_mode是独立的模式。当MP表子集非0且MP_table_mode为‘1’时，应忽略与存储于SMTP接收实体的子集－0 MP表版本不同的MP表子集内容。此外，当MP表子集数字非0且MP_table_mode为0时，应忽略与存储于SMTP接收实体中的低级子集MP表子集数不同的MP表子集版本内容。每次版本变更应按照256模式增值。

length：包含了以字节计算的MP表的长度，即从下一字段起直到MP表最后一个字节的长度。‘0’值在此字段无效。

MP_table_mode：其表明了在MP表子集使用时，MP表子集处理的模式。在“按顺序的处理模式” 以及此MP表的表子集编号非0时，SMTP接受实体应在处理该MP表子集前接收所有与这个MP表子集有相同版本的更低的子集编号的MP表子集。例如，如果SMTP接收实体还没有收到相同版本的子集2的MP表，那么它就不能处理子集3的MP表。在“与顺序不相关的处理模式”中，当该MP表子集的子集数设定为非零时，在接收到MP表子集后一旦储存在一个SMTP接收实体中的子集－0MP表具有该MP表子集的相同版本，SMTP接收实体应立即处理MP表子集。在“独立处理模式”中，各MP表子集的版本得以单独管理。在分段的MP表里，每个MP表子集通过多重SMTP发送实体中的一个被传输，且在该模式中得以改编。MP表子集的独立模式可用于多通道实例化，即从子集－0到子集－N的MP表子集被指定为从Ch－0到Ch－N的逻辑通道。

表12 MP_table_mode 的值

值	描述
00	按顺序的处理模式
01	与顺序不相关的处理模式
10	独立处理模式
11	保留

SMTP_package_id：该字符是SMTP数据包的一个独有的标识符。

SMTP_package_id_length：以字节形式表示的SMTP数据包ID字符串长度，不包括终止空字符。

SMTP_package_id_byte：SMTP数据包标识符的一个字节。当SMTP_数据包_标识符_字节是字符串时，终止空字符不应包含在这些字符串中。

asset_id ：媒体资源资源标识符。

asset_type：媒体资源的类型。这在被注册在MP4REG（http://www.mp4ra.org）上的四字符编码（“4CC”）类型中得到描述。

asset_size：描述媒体资源大小。

MP_table_descriptors：为MP表提供描述符。

MP_table_descriptors_length：包含描述符语法循环的长度。长度计算是从下一个字段开始，到描述符的语法循环结束。几个描述符可以被插入到该循环语法中。例如，额外_数据包_信息_URL描述符可以包含在循环语法中，其作用是为该数据包提供数据包信息网页的URL。

MP_table_descriptors_byte：表示描述符循环中的一个字节。

number_of_assets：指示由MP表提供信息的资源数。

packet_id：指示SMTP包报头中SMTP会话的标识符。

asset_clock_relation_flag：指示某项资产是否使用NTP时钟或其它时钟系统作为时钟参考。若该标记为‘1’，则包含asset_clock_relation_id字段在内。若该字段为‘0’，则该资产使用NTP时钟。

asset_clock_relation_id：向该媒体资源提供一个时钟关系标识符。这部分以资源CRI-说明符传输的时标关系为参考。该字段的值是由CRI描述符所提供的计时器关系标识符的值之一。

asset_timescale_flag：表明是否提供“asset-timescale”信息。如果该标记设为‘1’，则包含资源时间量程字段，如该标记设为‘0’，则资源时间量程为90000（90kHz）。

asset_timescale ：为所有的时间戳提供单位时间信息，这些时间戳用于表示资源在一秒之内的单元数量。

location_count：提供资源定位信息的数量。当一个资源是通过一个地点传送时，设置‘1’。当分发完成时，在一个资产中包含的微处理机装置是通过多种渠道传送的，而不是设置‘1’。当一个资源被发送到多个位置时，SMTP接收实体应从所有指示的位置接收该资源的所有CEU。

SMT_general_location_info: 提供资源位置的信息。

asset_descriptors_length：字节数量的计算，从下一字段开始，到资源描述符语句循环结束。

asset_descriptors_byte：资源描述符中的一个字节。

8.3.6 分块视频关系表

8.3.6.1 概述

分块视频关系表包含的信息用于描述原视频Asset和分块视频Asset之间的关系。在空间上，原视频和不同的分块视频之间拥有不同的asset_id；在时间上，同一时间的不同分块视频和原视频拥有同样的ceu_sequence_number。

8.3.6.2 语法

表13定义了分块视频关系表的语法。

表13 分块视频关系表语法

语法	值	比特数	类型
Block_association_table() {			
table_id		8	uimsbf
version		8	uimsbf
length		32	uimsbf
table_payload {			
partitioned_asset_number	N1	8	uimsbf
for (i=0; i<N1; i++) {			
asset_id()			
original_height		16	uimsbf
original_width		16	uimsbf
reserved		4	
block_number	N2	8	
for (j=0; j<N2; j++) {			
block_height_top		16	uimsbf
block_width_left		16	uimsbf
block_height		16	uimsbf
block_width		16	uimsbf
asset_id()			
}			
}			
}			
}			

8.3.6.3 语义

table_id：分块视频关系表的标识符。

version：分块视频关系表的版本。新的版本所携带的信息将覆盖任何之前的旧版本。

length：包含了以字节计算的分块视频关系表的长度，即从下一字段起直到分块视频关系表最后一个字节的长度。‘0’值在此字段无效。

partitioned_asset_number：指示被分块的原视频Asset的个数。

asset_id：指示被分块的原视频的asset_id。

original_height：指示原视频的高度，以像素点为单位。

original_width：指示原视频的宽度，以像素点为单位。

block_number：指示对应于某个原视频Asset，其分块视频的个数。

block_height_top：指示分块视频的CEU上边沿相对于原视频CEU上边沿的距离，以像素点为单位。

block_width_left：指示分块视频CEU左边沿相对于原视频CEU左边沿的距离，以像素点为单位。

block_height：指示分块视频CEU的高度，以像素点为单位。

block_width：指示分块视频CEU的宽度，以像素点为单位。

asset_id：指示某个分块视频的asset_id。

8.4 媒体资源描述符

8.4.1 概述

在MP表中，提供了媒体资源描述符（Asset_descriptor）的信息，为解决SMT灵活传输与个性化消

费的需求，提出设计了以下几种媒体资源描述符。

8.4.2 MUR 描述符

8.4.2.1 概述

为了满足个性化消费的需求，在一个媒体资源中，可以根据媒体内容类型、重要性等特征对媒体资源进行分级，因此提出了MUR(media unit relationship)描述符，该描述符的具体介绍如下。

8.4.2.2 语法

MUR描述符的语法见表14。

表14 MUR 描述符语法

语法	值	比特数	类型
MUR_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
edit_id			
edit_id_number	N1	16	uimsbf
for(j=0; j<N1; j++) {			
ceu_sequence_number		32	uimsbf
}			
}			

8.4.2.3 语义

descriptor_tag：标识符，用于标志descriptor的类型。

descriptor_length：标识符的长度，单位为字节。

edit_id：指示媒体资源分级的标志。

edit_id_number：该媒体资源分级下包含的CEU数目。

ceu_sequence_number：ceu的序列号，指示对应edit id标志的ceu。

8.4.3 AT 描述符

8.4.3.1 概述

AT(available time)描述符，用于指示当前媒体资源的可获取时间，通过对当前网络状况的估计，系统可以对媒体资源的可获取时间进行估算，并将该时间告知接收端，这样既可以保证传输质量，同时得到最佳的接收端缓冲区。

8.4.3.2 语法

AT描述符的语法见表15。

表15　AT 描述符语法

语法	值	比特位数	备注
AT_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		32	uimsbf
available_time_count	N1	8	uimsbf
for (i=0; i<N1; i++) {			
location_index		8	uimsbf
available_begin		32	uimsbf
available_end		32	uimsbf
}			
}			

8.4.3.3　语义

descriptor_tag：标识符，用于标志descriptor的类型。

descriptor_length：标识符的长度，单位为字节。

available_time_count：计数值，用以统计媒体资源可获取时间信息的位置（location）的数量。

location_index：索引值，该索引对应的是SMT_general_location_info()的媒体资源location的序号。

available_begin：时间值，该时间是上述索引的location所对应的媒体资源的最早可获取时间。

available_end：时间值，该时间是上述索引的location所对应的媒体资源的最晚可获取时间。

根据available_begin和available_end这两个属性的值，有不同的情况来获取可访问时间窗口的范围表见表16。

表16　可获取时间属性的值和对应的情况

‘available_begin’ 值	‘available_end’ 值	描述
全 ‘0’	全‘1’	内容一直可被获取
UTC	全 ‘1’	内容从‘available_begin’时间点之后可一直被获取
UTC1	UTC2	内容在‘available_begin’到‘available_end’的时间段内可被获取

8.4.4　Asset 组描述符

8.4.4.1　概述

Asset组描述符，用于指示同一个SMT Package中Asset之间的关联关系，以指导客户端进行正确解码、自适应切换及个性化呈现。

8.4.4.2　语法

Asset组描述符的语法见表17。

表17　Asset 组描述符语法

语法	值	比特位数	备注
Asset_group_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
reserved	‘1111’	4	
dependency_flag		1	blsbf
composition_flag		1	blsbf
equivalence_flag		1	blsbf
similarity_flag		1	blsbf
if(dependency_flag) {			
num_dependencies	N1	8	uimsbf
for(i = 0; i <N1; i++) {			
asset_id()			
}			
}			
if(composition_flag) {			
num_compositions	N2	8	uimsbf
for(i = 0; i <N2; i++) {			
asset_id()			
}			
}			
if(equivalence_flag) {			
equivalence_selection_level		8	uimsbf
num_equivalences	N3	8	uimsbf
for(i = 0; i <N3; i++) {			
asset_id()			
equivalence_selection_level		8	uimsbf
}			
}			
if(similarity_flag) {			
similarity_selection_level		8	uimsbf
num_similarities	N4	8	uimsbf
for(i = 0; i <N4; i++) {			
asset_id()			
similarity_selection_level		8	uimsbf
}			
}			
}			

8.4.4.3 语义

descriptor_tag：用于指示此类型描述符的标签值。

descriptor_length：指示此描述符的字节长度，从下一个字段计算至最后一个字段。

dependency_flag：指示在此描述符中是否有依赖关系的其他Asset。值‘0’意味着没有,不需要添加。

composition_flag：指示在此描述符中是否有组合关系的其他Asset。值‘0’意味着没有,不需要添加。

equivalence_flag：指示在此描述符中是否有等同关系的其他Asset。值‘0’意味着没有,不需要添加。

similarity_flag：指示在此描述符中是否有相似关系的其他Asset。值‘0’意味着没有,不需要添加。.

num_dependencies：指示此描述符所描述的Asset所依赖的Asset的数目。

asset_id：指示此描述符所描述的Asset所依赖的Asset的ID，格式如9.5.1所示。此描述符中提供的Asset ID顺序与其内部编码依赖层次相对应。

num_compositions：指示与此描述符所描述的Asset有组合关系的Asset的数目。

asset_id：指示与此描述符所描述的Asset有组合关系的Asset的ID，格式如9.5.1所示。

equivalence_selection_level：指示所对应的Asset在等同关系组中的呈现等级。‘0’值表示该Asset被默认呈现。当默认Asset无法被选择时，拥有呈现等级较小的Asset会作为替代被选择和呈现。

num_equivalences：指示与此描述符所描述的Asset有等同关系的Asset的数目。

asset_id：指示与此描述符所描述的Asset有等同关系的Asset的ID，格式如9.5.1所示。

similarity_selection_level：指示所对应的Asset在相似关系组中的呈现等级。‘0’值表示该Asset被默认呈现。当默认Asset无法被选择时，拥有呈现等级较小的Asset会作为替代被选择和呈现。

num_similarities：指示与此描述符所描述的Asset有相似关系的Asset的数目。

asset_id：指示与此描述符所描述的Asset有相似关系的Asset的ID，格式如9.5.1所示。

8.4.5　CEU 时间戳描述符

基于‘elst’box中的时间偏移信息，CEU时间戳描述符提供了对应CEU的第一个AU的呈现时间。若对应的媒体数据呈现时间已经过期，则忽略该描述符。

8.4.5.1　语法

CEU时间戳描述符的语法见表18。

表18　CEU 时间戳描述符语法

语法	值	比特位数	备注
CEU_timestamp_descriptor () {			
descriptor_tag		16	uimsbf
descriptor_length		8	uimsbf
for (i=0; i<N; i++) {			
ceu_sequence_number		32	uimsbf
ceu_presentation_time		64	uimsbf
}			
}			

8.4.5.2　语义

descriptor_tag：标识符，用于标志descriptor的类型。

descriptor_length：标识符的长度，单位为字节。

ceu_sequence_number：表示描述符所对应的CEU的序列号。

ceu_presentation_time：表示指定CEU的第一个AU的呈现时间，用64位NTP时间戳格式表示。

8.5　语法元素组

8.5.1　资源 ID

8.5.1.1　概述

资源ID语法元素组用来提供Asset标识符。如果存在‘cceu’盒子，此语法元素组的值应当等于该盒子中Asset标识符。如果不存在，Asset标识符的赋值超出本规范范围。

8.5.1.2　语法

资源ID的语法见表19。

表19　资源 ID 语法

语法	值	比特位数	备注

asset_id() {			
asset_id_scheme		32	uimsbf
asset_id_length	N	8	uimsbf
for(i = 0; i <N;i++) {			
asset_id_byte		8	blsbf
}			
}			

8.5.1.3 语义

asset_id_scheme：提供Asset标识符机制，具体取值待定义。

asset_id_length：提供asset_id的字节长度。

asset_id_byte：asset_id中的一个字节。

8.5.2 智能媒体通用位置信息（SMT_general_location_info）

8.5.2.1 概述

智能媒体传输通用位置信息语法元素组用于向指定的数据项（item）的负载数据提供位置信息，它指出了数据流中所携带的数据类型以及传输方式，例如单向广播网络、IP网络以及相应的端口号等信息。

8.5.2.2 语法

SMT_general_location_info的语法见表20。

表20　SMT_general_location_info 语法

语法	值	比特位数	备注
SMT_general_location_info() {			
location_type		8	uimsbf
if (location_type == 0x00) {			
packet_ id		16	uimsbf
} else if (location_type == 0x01) {			
ipv4_src_addr		32	uimsbf
ipv4_dst_addr		32	uimsbf
dst_port		16	uimsbf
packet_ id		16	uimsbf
} else if (location_type == 0x02) {			
ipv6_src_addr		128	uimsbf
ipv6_dst_addr		128	uimsbf
dst_port		16	uimsbf
packet_ id		16	uimsbf
} else if (location_type == 0x03) {			
network_id		16	uimsbf
MPEG_2_transport_stream_id		16	uimsbf
reserved	‘111’	3	bslbf
MPEG_2_PID		13	uimsbf
} else if (location_type == 0x04) {			
ipv6_src_addr		128	uimsbf
ipv6_dst_addr		128	uimsbf
dst_port		16	uimsbf
reserved	‘111’	3	bslbf
MPEG_2_PID		13	uimsbf
} else if (location_type == ‘0x05’) {			
URL_length	N1	8	uimsbf
for (i=0; i<N1; i++) {			
URL_byte		8	char
}			
} else if (location_type == ‘0x06’) {			
length		16	uimsbf
for (i=0;i<N2;i++) {			
byte		8	uimsbf
}			
} else if (location_type == ‘0x07’) {			
} else if (location_type == ‘0x08’) {			
message_id		8	uimsbf
} else if (location_type == ‘0x09’) {			
packet_ id		16	uimsbf
message_id		8	uimsbf
} else if (location_type == ‘0x0A’) {			
ipv4_src_addr		32	uimsbf
ipv4_dst_addr		32	uimsbf

表 20（续）

语法	值	比特位数	备注
dst_port		16	uimsbf
packet_id		16	uimsbf
message_id		8	uimsbf
} else if (location_type == ‘0x0B’) {			
ipv6_src_addr		128	uimsbf
ipv6_dst_addr		128	uimsbf
dst_port		16	uimsbf
packet_ id		16	uimsbf
message_id		8	uimsbf
} else if(location_type == ‘0x0C’) {			
ipv4_src_addr		32	uimbsf
ipv4_dst_addr		32	uimbsf
dst_port		16	uimbsf
reserved	‘111’	3	bslbf
MPEG_2_PID		13	uimbsf
}			
}			

8.5.2.3　语义

语法结构中相应字段的含义如下。

location_type：位置类型字段，指示SMT数据流的位置信息，其取值见表21。

表21　location_type 取值

值	描述
0x00	当前 SMTP 包流中， packet_id 为指定值的子流对应的媒体资源（Asset）
0x01	UDP/IPv4 传输的 SMTP 包流中 packet_id 为指定值的子流对应的媒体资源
0x02	UDP/IPv6 传输的 SMTP 包流中 packet_id 为指定值的子流对应的媒体资源
0x03	广播网络传输的 MPEG-2 TS 流中一个基本流（ES）
0x04	IPV6 广播（组播）网络传输的 MPEG-2 TS 流中的一个基本流
0x05	统一资源定位符（URL）
0x06	保留做私有位置信息用，由用户自己定义
0x07	承载本 SMT_general_location_info 的当前信令消息
0x08	消息 ID（见表 24）为 message_id 类型的信令消息， 且在当前信令消息所对应的 SMTP 子流中（数据包的 packet_id 与当前数据包的相同）
0x09	消息 ID 为指定类型的信令消息，且在当前 SMTP 包流中 packet_id 为指定值的子流中
0x0A	消息 ID 为指定类型的信令消息，且在 UDP/IPv4 传输的 SMTP 包流中 packet_id 为指定值的子流中
0x0B	消息 ID 为指定类型的信令消息，且在 UDP/IPv6 传输的 SMTP 包流中 packet_id 为指定值的子流中
0x0C	IPV4 广播（组播）网络传输的 MPEG-2 TS 流中的一个基本流
0x0D~0x9F	保留做 ISO 用
0xA0~0xFF	保留做私用

packet_id：SMTP包首部中的包标识符，用于区分携带不同媒体资源（Asset）数据的SMTP包。

ipv4_src_addr：一个IP应用数据流对应的IPv4源地址。SMT支持异构网络环境下数据流的传输，因此应当支持IP网络。

ipv4_dst_addr：一个IP应用数据流对应的IPv4目的地址。

dst_port：一个IP应用数据流对应的目的端口号。IP地址和端口号唯一标识一个通信进程。

ipv6_src_addr：一个IP应用数据流对应的IPv6源地址。考虑到IP网络的发展趋势将会向IPv6演进，SMT应当向后兼容IPv6。

ipv6_dst_addr：一个IP应用数据流对应的IPv6目标地址。

network_id：携带MPEG-2 TS传输流的广播网络标识符。此字段的取值由具体使用的广播网络决定。

MPEG_2_transport_stream_id：MPEG-2 TS传输流流标识符。stream_id标识了传输流中各个基本流（Elementary Stream，ES）的类型，其映射关系在相关协议中另有规定。

MPEG_2_PID：携带基本流的MPEG-2 TS包的标识符。包标识符用于标识携带不同基本流数据的TS包。

URL_length：以字节为单位的URL长度。结尾的空字节（null）不应被计数。SMTP是应用层协议，类似于HTTP等协议，可以使用统一资源定位符进行数据拉取。

URL_byte：一个字节的URL数据。结尾的空字节（null）不应被计数。

length：指示用户私用位置信息字节长度，具体类型由用户决定。

message_id：SMT信令消息标识符，用于区分不同种类的信令。message_id与信令消息一一对应，其映射关系在SMT协议中另有规定。

8.5.3　标识符映射（Identifier mapping）

8.5.3.1　概述

本语法组元素提供了一种内容标识符和SMTP包子流之间的映射。其中，SMTP包子流是指拥有相同packet_id的SMTP包流的子集，内容标识符可用不同格式提供，例如媒体资源标识符、统一资源定位符等。

8.5.3.2　语法

标识符映射的语法见表22。

表22　标识符映射语法

语法	值	比特位数	备注
Identifier_mapping (){			
identifier_type		8	uimsbf
if (identifier_type == 0x00){			
asset_id()			
}else if (identifier_type == 0x01){			
url_count	N1	16	uimsbf
for(i=0;i<N1;i++){			
url_length[i]	N2	16	uimsbf
for(j=0;j<N2;j++){			
url_byte[i]		8	uimsbf
}			
}			
}else if (identifier_type == 0x02){			
regex_length	N3	16	uimsbf
for(i=0;i<N3;i++){		8	uimsbf
regex_byte			
}			
}else if (identifier_type == 0x03){			
representation_id_length	N4	16	uimsbf
for (i=0;i<N4;++) {			
representation_id_byte		8	uimsbf

表 22 （续）

语法	值	比特位数	备注

}
}else{
private_length
for(i=0;i<N5;i++){
private_byte
}
}
}
	
	N5
		
		16
		
		8
			
			uimsbf
			
			uimsbf

8.5.3.3 语义

identifier_type：占8位，指出用于映射packet_id的标识符的类型。允许映射的类型见表23。

表23　允许映射的标识符类型

值	描述
0x00	作为Asset id的内容标识符
0x01	作为一系列拥有相同packet_id映射的URL的内容标识符
0x02	作为RegEx字符串的内容标识符，其中，RegEx字符串用于匹配一个或多个URL，URL可识别具有相同packet_id映射的文件。
0x03	作为DASH Representation@id的内容标识符，其中，DASH Representation@id可识别所有具有相同呈现的DASH 分段。
0x04~0xFF	此值范围是为私有标识符保留的。

url_count：占16位，URL计数器，指出URL的数量。
url_length：占16位，指出下一URL的字节长度。
url_byte：占8位，是格式化为GB/T 18030-2005标准规定的Unicode字符串的URL的一个字节。
regex_length：占16位，标识符是一个匹配一组URL的正则表达式，本字段指出RegEx字符串的长度。
regex_byte：占8位，是格式化为GB/T 18030-2005标准规定的Unicode字符串的RegEx的一个字节。
representation_id_length：占16位，作为DASH Representation@id的内容标识符，本字段指出Representation@id的长度。
representation_id_byte：占8位，是Representation@id字符串的一个字节。
private_length：占16位，指出了私有标识符的字节长度。
private_byte：占8位，是私有标识符中的一个字节。

8.5.4 mime 类型（mime_type）

8.5.4.1 概述

mime_type用来指出媒体资源的类型，以便对其分类，从而对不同格式的媒体资源高效传输。

8.5.4.2 语法

mime类型的语法见表24。

表24　mime 类型语法

Syntax	Value	No. of bits	Mnemonic
mime_type(){			
mime_type_length	N1	8	uimsbf
for(i=0;i<N1;i++) {			
mime_type_byte		8	uimsbf
}			
}			

8.5.4.3　语义

mime_type_length：占8位，指出mime_type的长度。

mime_type_byte：占8位，是mime_type的一个字节，格式为GB/T 18030-2005标准规定的Unicode字符串。

8.6　ID 值

8.6.1　消息 ID

消息ID的值见表25。

表25　消息 ID 值

值	描述
0x0000	PA 消息
0x0001 ~ 0x0010	保留
0x0011 ~ 0x0020	MPT 消息
0x0021 ~ 0x01FF	保留
0x0200	时钟关系信息消息
0x0201	设备性能信息消息
0x0202	保留
0x0203	自适应前向纠错编码消息
0x0204	假设接收机缓存模型消息
0x0205 ~ 0x0208	保留做 ISO 用
0x0209	媒体资源传输特性消息
0x020A~0x7FFF	保留做 ISO 用
0x8000	资源请求响应消息
0x8001	交互反馈消息
0x8002	会话控制信令消息
0x8003	同步请求消息
0x8004	同步响应消息
0x8005 ~ 0xFFFF	保留做私用

8.6.2　表 ID

表ID的值见表26。

表26　表 ID 值

值	描述
0x00	PA 表
0x01 ~ 0x10	保留
0x11 ~ 0x1F	子集-0 MP 表 ~ 子集-14 MP 表
0x20	结束 MP 表
0x21~0x7F	保留做 ISO 用
0x80 ~ 0xFF	保留做私用

8.6.3　描述符 ID

描述符ID的值见表27。

表27　描述符 ID 值

值	描述
0x0000	时钟关系信息描述符
0x0001	CEU 时间戳描述符
0x0002-0x0004	保留
0x0005	AT 描述符
0x0006	Asset 组描述符
0x0007	MUR 描述符
0x0008	CEU 消费描述符
0x0009-0x3FFF	保留（8-bit 长度描述符）
0x4000-0x6FFF	保留（16-bit 长度描述符）
0x7000-0x7FFF	保留（32-bit 长度描述符）
0x8000-0xFFFF	保留

8.7　资源请求响应消息

8.7.1　概述

资源请求响应消息（Resource Request/Response Message, 3R_message）提供SMT服务器与客户端之间的请求与响应格式。当客户端与服务器之间需要交互请求与响应信息时，使用此消息进行会话。

8.7.2　语法

资源请求响应消息的语法见表28。

表28　资源请求响应消息语法

语法	值	比特位数	备注
3R_message(){			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload {			
reserved	‘1111111’	7	bslbf
method		1	bslbf
else if(method == REQUEST) {			
request_serial_number		8	uimsbf
mime_type()			
data_length	N1	16	uimsbf
for(j == 0; j < N1; j++) {			
data_byte		8	uimsbf
}			
}			
else if(method == RESPONSE) {			
response_serial_number		8	uimsbf
status_number		8	uimsbf
if(status_number == 0x02) {			
mime_type()			
content_length	N2	16	uimsbf
for(j == 0; j < N2; j++) {			
content_byte		8	uimsbf
}			
}			
}			
}			
}			

8.7.3　语义

message_id：实时交互消息的消息标识字段，标识此传输数据用于请求响应交互数据的传输。

version：使用消息中的版本表示当前时间数据在该时间内的序列号。

length：消息长度标识字段。

reserve：保留字段。

method：使用原保留字段其中两位来标志当前用户请求资源的方法，类型之与描述见表29。

表29　method 取值与描述

值	描述
0	“REQUEST”
1	“RESPONSE”

request_serial_number：指示客户端发送的请求的序列号。当发送消息时，它模256递增。

mime_type：标识一般化文件MIME类型的字段，用以指示服务器或客户端按照相应文件格式解析数据。

data_length：标识数据段长度的字段。

data_byte：标识数据段字段。

response_serial_number：指示服务器发出的响应信息的序列号。当发送信息时，它以模256递增。当该值与request_serial_number值相等时，请求和响应消息相互匹配。

status_number：包含描述服务器返回状态的字段，其值与描述见表30。

表30　status_number 取值与描述

值	描述
0x00	请求失败，请求所希望得到的资源未被在服务器上发现，或上传数据失败
0x01	请求已成功
0x02	请求已成功，请求所希望的响应头或数据体将随此响应返回
0x03~0x7F	Reserved for ISO
0x80~0xFF	Reserved for private use

8.8　交互反馈消息

8.8.1　概述

交互反馈消息(Interaction Feedback Message)提供沉浸式媒体消费时，服务器与客户端之间的交互反馈。当沉浸式媒体消费中的服务器与客户端之间需要发送交互反馈信息时，使用此消息进行会话。

交互反馈消息主要由交互目标、交互类型、交互内容三部分构成。随着用户具体交互行为的变化，其中封装的内容会动态的变化，因此在该表中未指定具体的交互内容。

8.8.2　语法

交互反馈消息的语法见表31。

表31　交互反馈消息语法

语法	值	比特数	备注
interaction_feedback_message() {			
message_id		16	uimsbf
version		8	uimsbf
length		32	uimsbf
message_payload {			
message_source		1	bool
reserved		7	uimsbf
asset_id()			
interaction_num	N	8	uimsbf
for (i=0; i<N; i++) {			
timestamp		32	uimsbf
interaction_target		8	uimsbf
interaction_type		8	uimsbf
interaction_content{			
interaction_content_length		32	uimsbf
}			
}			
}			
}			

8.8.3 语义

message_id：交互反馈消息的标识符。

version：交互反馈消息的版本。新的版本所携带的信息将覆盖任何之前的旧版本。

length：包含了以字节计算的交互反馈消息的长度，即从下一字段起直到交互反馈消息最后一个字节的长度。‘0’值在此字段无效。

message_source：指示消息源，0表示交互反馈消息是客户端发往服务器，1表示交互反馈消息是服务器发往客户端。

reserved：保留字节位。

asset_id：指示客户端当前消费内容的asset_id。

interaction_num：指示当前信令中包含的交互的数目。

timestamp：指示当前交互产生的时间，使用UTC时间。

interaction_target：交互目标，指示客户端当前交互的目标，包括头盔设备当前状态、用户关注目标、用户当前状态等；交互目标的取值见表32。

表32　交互目标取值

类型	取值	描述
Null	0	交互目标为空，即没有特定交互目标
HMD_status	1	交互目标为头盔设备当前状态
Object of interests	2	交互目标为用户关注区域当前状态
User_status	3	交互目标为用户当前状态

interaction_type：交互类型，指示客户端对于当前的interaction_target，进行的交互行为的类型，此字段的取值与interaction_target相关，包括追踪、注视、触碰等，交互类型的取值见表33。

表33　交互类型取值

类型	取值	说明
Null	0	交互类型为空，即没有特定交互类型
Tracking	1	交互类型为追踪
Gaze	2	交互类型为注视
Touch	4	交互类型为触碰

interaction_content_length：指示客户端对于当前的交互行为，其具体交互内容的长度。该字段取值以字节为单位，为从该字段下一字节起到当前交互内容结束的长度。

8.9 会话控制信令

8.9.1 概述

SMT接收端可以利用SMT消息建立和控制会话。SMT接收端向发送端发送SC消息来控制媒体的传输。该消息主要提供会话的开始、停止以及跳转的功能。

8.9.2 语法

会话控制信令的语法见表34。

表34　会话控制信令语法

语法	值	比特数	备注
SC_message() {			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload {			
command_code		32	uimsbf
if (command_code == 0x01){			
start_time		32	uimsbf
}			
else if(command_code == 0x02){			
}		32	uimsbf
else if(command_code == 0x03){			
current_presentation_time		64	uimsbf
seek_time		64	simsbf
progress_point		32	uimsbf
}			
number_of_assets	N	8	uimsbf
for(i=0;i<N;++i) {			
packet_id		8	uimsbf
}			
}			
}			

8.9.3　语义

message_id：标识SC消息的ID。该字段为16比特。

version：标识SC消息的版本。

length：标识SC消息的长度。该字段长度为16比特。

command_code：标识一种会话控制操作，其值与对应的描述见表35。

表35　command_code 的值

值	描述
0x01	播放
0x02	停止
0x03	跳转
0x04~0xFF	保留

start_time标识呈现的开始时间。由SMT接收实体的请求时间设置该字段的值。当SMT发送实体接收到该字段时，会选择呈现时间最接近该字段所指示时间的CEU发送。当两个CEU的呈现时间都同样接近于该字段所指示的时间时，选择呈现时间更早的CEU发送，并根据start_time和CEU的持续时间生成时间线。start_time基于NTP时间。在点播应用中，start_time作为更新CEU呈现时间的基准线。

current_presentation_time：标识当前的呈现时间。current_presentation_time基于NTP时间。

seek_time：标识从当前时间到跳转目标时间的时间。该字段长度为64比特，其值可取正或负，正值表示向前跳转，负值表示向后跳转。

progress_point：标识从开始时间到当前播放时间在整个呈现时间所占的比例。其单位为百分数。

number_of_asset：标识该消息控制的资源数。

packet_id：SMTP中的packet_id。

8.10 设备性能信息

8.10.1 设备性能消息（DCI 消息）

8.10.1.1 概述

设备性能消息（DCI消息）所传输的设备性能信息表（DCI表）提供了数据包消费所需的设备性能信息。

8.10.1.2 语法

表36定义了DCI消息的语法。

表36 DCI 消息的语法

语法	值	比特位数	备注
DCI_message () {			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
extension {			
}			
message_payload {			
DCI_table()			
}			
}			

8.10.1.3 语义

语法结构中相应字段的含义如下。

message_id：表示DCI消息的标识符。这一字段的长度为16位。

version：表示DCI消息的版本。SMT接收实体可通过这一字段检测接收消息的版本。这一字段的长度为8位。

length：包含了以字节计算的DCI消息的长度，即从下一字段起直到DCI消息最后一个字节的长度。‘0’值在此字段无效。

DCI_table()：提供数据包消费所需的设备性能信息。DCI表的具体内容在下一小节定义。

8.10.2 设备性能信息表（DCI 表）

8.10.2.1 概述

设备性能信息表（DCI表）中包含了用于数据包消费所需的设备性能信息。基于视频，音频或下载（可用于非时序数据）这几种不同的媒体资源mime类型，DCI表提供了几组不同的信息用于支持数据包的传输和消费。

8.10.2.2 语法

表37定义了DCI表的语法。

表37 DCI 表语法

语法	值	比特位数	备注
DCI_table() {			
table_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
number_of_assets	N1		
for (i=0; i<N1; i++) {			
asset_id()			
mime_type()			
reserved	‘1111111’	7	bslbf
codec_complexity_flag		1	bslbf
if (codec_complexity_flag == 1) {			
if (top_level_mime_type == VIDEO) {			
video_codec_complexity {			
video_average_bitrate		16	uimsbf
video_maximum_bitrate		16	uimsbf
horizontal_resolution		16	uimsbf
vertical_resolution		16	uimsbf
temporal_resolution		8	uimsbf
video_minimum_buffer_size		16	uimsbf
}			
}			
else if (top_level_mime_type == AUDIO) {			
audio_codec_complexity {			
audio_average_bitrate		16	uimsbf
audio_maximum_bitrate		16	uimsbf
audio_minimum_buffer_size		16	uimsbf
}			
}			
}			
else {			
download_capability {			
required_storage			
}			
}			
}			
}			

8.10.2.3 语义

语法结构中相应字段的含义如下。

table_id：表示DCI表的标识符。

version：表示DCI表的版本。接收到DCI表后新版本将立即覆盖旧版本。

length：包含了以字节计算的DCI表的长度，即从下一字段起直到DCI表最后一个字节的长度。‘0’值在此字段无效。

number_of_assets：表示的媒体资源（Asset）的数量。

asset_id：提供了媒体资源（Asset）的标识符。

top_level_mime_type：表示媒体资源的顶层资源类型，MIME type由mime_type()字段定义。

codec_complexity_flag：果这个标识位置‘1’，表示提供了编解码器复杂度信息。

video_codec_complexity：表示视频解码器需要处理的复杂度。

video_maximum_bitrate：表示整个视频媒体资源的最大比特率，单位为千位每秒(kilo-bit/s)。

video_average_bitrate：表示整个视频媒体资源的平均比特率，单位为千位每秒(kilo-bit/s)。

horizontal_resolution：表示视频的水平分辨率，单位为像素(pixel)。

vertical_resolution：表示视频的垂直分辨率，单位为像素(pixel)。

temporal_resolution：表示视频的平均时间分辨率，单位为帧每秒(frames per second)。

video_mimimum_buffer_size：表示所需的视频解码器缓冲区大小的最小值，单位为千字节(kilo-bytes)。

audio_codec_complexity：表示音频解码器需要处理的复杂度。

audio_average_bitrate：表示整个音频媒体资源的平均比特率，单位为千位每秒(kilo-bit/s)。

video_average_bitrate：表示整个音频媒体资源的最大比特率，单位为千位每秒(kilo-bit/s)。

video_mimimum_buffer_size：表示所需的音频解码器缓冲区大小的最小值，单位为千字节(kilo-bytes)。

download_capability：表示用于下载所需的性能。

required_storage：表示用于下载所需的容量的大小，单位为千字节(kilobytes)。

8.11　时钟关系信息

8.11.1　时钟关系信息消息（CRI 消息）

8.11.1.1　概述

时钟关系信息消息携带了与时钟关系相关的信息，用于NTP时间戳与MPEG-2 STC时间间的映射。

为了获得使用NTP时间戳的媒体资源与使用MPEG-2呈现时间戳（PTS）的MPEG-2 ES的同步，需要周期性的传输同一时间点的NTP时间戳样本和STC样本的值，供SMT接收实体获取NTP时间戳和MPEG-2 STC的映射关系。如果有多个使用不同MEPG-2 STC的MPEG-2 ES，则将使用多个CRI消息。

8.11.1.2　语法

表38定义了CRI消息的语法。

表38　CRI 消息语法

语法	值	比特位数	备注
CRI_message () {			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
extension {			
}			
message_payload {			
CRI_table()			
}			
}			

8.11.1.3　语义

message_id：表示CRI消息的标识符。这一字段的长度为16位。

version：表示CRI消息的版本。SMT接收实体可通过这一字段检测接收消息的版本。这一字段的长度为8位。

length：包含了以字节计算的CRI消息的长度，即从下一字段起直到CRI消息最后一个字节的长度。‘0’值在此字段无效。此字段的长度为16位。

CRI_table()：CRI表，由下一条定义。

8.11.2　时钟关系信息表（CRI 表）

8.11.2.1 概述

CRI表由CRI消息传输，也可以由PA消息传输。

8.11.2.2 语法

表39定义了CRI表的语法。一个CRI表中可能包含多个CRI描述符。

表39　CRI 表语法

语法	值	比特位数	备注
CRI_table () {			
table_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
for (i=0; i<N; i++) {			
CRI_descriptor()		152	uimsbf
}			
}			

8.11.2.3 语义

table_id：表示CRI表的标识符。

version：表示CRI表的版本。接收到CRI表后新版本将立即覆盖旧版本。

length：包含了以字节计算的CRI表的长度，即从下一字段起直到CRI消息最后一个字节的长度。‘0’值在此字段无效。此字段的值应当为19的倍数。

CRI_descriptor()：CRI描述符，由下一条定义。

8.11.3 时钟关系信息描述符（CRI 描述符）

8.11.3.1 概述

CRI描述符用于时间同步功能，说明了NTP时间戳和MPEG-2 STC时间的映射关系。clock_relation_id字段定义了由MPEG-2 TS生成的媒体资源使用的时钟参照的值。CRI描述符由CRI表携带并传输。

8.11.3.2 语法

表40定义了CRI描述符的语法。

表40　CRI 描述符语法

语法	值	比特位数	备注
CRI_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
clock_relation_id		8	uimsbf
reserved	’111111’	6	uimsbf
STC_sample		42	uimsbf
NTP_timestamp_sample		64	uimsbf
}			

8.11.3.3　语义

descriptor_tag：标识符，用于标志descriptor的类型。

descriptor_length：包含了以字节计算的CRI描述符的长度，即从下一字段起直到CRI描述符最后一个字节的长度。‘0’值在此字段无效。

clock_relation_id：时钟关系的标识符。

STC_sample：包含与下一字段NTP_timestamp_sample值相符合的MPEG-2 STC值。STC样本值为42位格式。

NTP_timestamp_sample：与前一字段STC_sample相符合的NTP时间戳样本值。

8.12　同步请求消息

8.12.1　概述

客户端需要知道当前的网络时延状况和可用带宽来计算固定端到端时延，从而实现同步控制。

8.12.2　语法

表41定义了同步请求消息（sync_request_message）的语法。

表41　同步请求消息语法

语法	值	比特位数	备注
sync_request_message() {			
message_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload(){			
network_delay		16	uimsbf
network_bandwidth		32	uimsbf
}			
}			

8.12.3　语义

message_id：SMT信令消息标识符，用于区分不同种类的信令。message_id与信令消息一一对应，其映射关系在SMT协议中另有规定。

length：指示用户私用位置信息字节长度。

version：信令信息的版本。

message_payload: 信令信息的有效负载，在这里为network_delay和network_bandwidth。

network_delay：当前网络延迟。

network_bandwidth：当前可用带宽信息。

$$t_{tmp} = t_0 + CEU\_size/B_b + \Delta t \quad \text{......(1)}$$

同步请求消息应包含网络延时network_delay和可用宽带信息network_bandwidth，在上个公式中，CEU_size为发送的平均CEU的大小，B_b为宽带网络的可用带宽（network_bandwidth），Δt为下行宽带网络延时（network_delay），t_{tmp}为计算所得用户端收到第一个CEU的时刻。$CEU_size/B_b+\Delta t$即为固定的端到端延迟。

8.13 同步响应消息

8.13.1 概述

发送资源的同时，服务端需发送一个下行信令告知客户端发送的第一个独立可解媒体资源的序号，从而来通知用户播放的时间，同步响应消息sync_response_message如下。

8.13.2 语法

表42定义了同步响应消息（sync_response_message）的语法。

表42 同步响应消息语法

语法	值	比特位数	备注
sync_response_message() {			
message_id		8	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload()			
{			
number_of_assets	N	16	uimsbf
for(i=0; i<N; i++)			
{			
asset_id		16	uimsbf
CEU_sequence_number		32	uimsbf
}			
}			
}			

8.13.3 语义

message_id：SMT信令消息标识符，用于区分不同种类的信令。message_id与信令消息一一对应，其映射关系在SMT协议中另有规定。

length：指示用户私用位置信息字节长度。

version：信令信息的版本。

message_payload: 信令信息的有效负载，在这里为CEU_sequence_number。

number_of_assets：指示媒体资源的数目。

asset_id：指示每个媒体资源的标识。

CEU_sequence_number；来通知客户端播放的时间的由服务端发送的第一个CEU的序号。

客户端根据信令中已知的第一个媒体资源序列查表获知时间信息和平均CEU大小，缓存当前CEU，等到时间戳对应上再播放该资源，实现宽带和广播频道的同步。

8.14 媒体资源传输特性消息

8.14.1 概述

媒体资源传输特征（Asset Delivery Characteristic, ADC）消息包含与媒体资源传输相关的QoS需求以及与其关联的QoE质量信息。为了提供更准确的信息，SMT发送实体可以更新ADC消息中的参数值，并定期或不定期发送它。考虑到ADC信息的大小，可以按媒体资源或CEU进行ADC信息传递。

8.14.2 语法

表43 媒体资源传输特性消息语法

语法	值	比特位数	备注
ADC_message() {			
message_id		16	uimsbf
version		8	uimsbf
length		32	uimsbf
message_payload()			
{			
validity_start_time		32	uimsbf
validity_duration		32	uimsbf
ADC_level_flag		1	boolean
flow_label_flag		1	boolean
reserved		6	uimsbf
if(ADC_level_flag==1){			
CEU_sequence_number		32	uimsbf
}			
packet_id		16	uimsbf
qos_descriptor{			
loss_tolerance		8	bslbf
jitter_sensitivity		8	bslbf
class_of_service		1	boolean
bidirection_indicator		1	boolean
reserved		6	uimsbf
}			
qoe_descriptor{			
n_samples	N1	16	uimsbf
for(i=1;i<N1;i++){			
sample_group_index		16	uimsbf
}			
spatial_quality		16	uimsbf
temporal_quality		16	uimsbf
aggregate_rate		32	uimsbf
}			
if(class_of_service==1){			

表43（续）

语法	值	比特位数	备注
bitstream_descriptor_vbr{			
sustainable_rate		16	uimsbf
buffer_size		16	uimsbf
peak_rate		16	uimsbf
max_MFU_size		8	uimsbf
mfu_period		8	uimsbf
}			
}			
else{			
bitstream_descriptor_cbr{			
peak_rate		16	uimsbf
max_MFU_size		8	uimsbf
mfu_period		8	uimsbf
}			
}			
if(flow_label_flag==1){			
flow_label		7	uimsbf
reserved		1	uimsbf
}			
}			
}			

8.14.3 语义

message_id：媒体资源传输特征消息的消息标识字段。

version：使用消息中的版本表示当前时间数据在该时间内的序列号。

length：消息长度标识字段。

validity_start_time：用于指示更新的ADC消息开始生效的UTC时间。validity_period_duration：指示从validity_start_time起更新的ADC消息的有效期持续时间（以毫秒为单位）。 该ADC消息中此参数的值在此持续时间内一直有效。

ADC_level_flag (1 bit)：指示所包含的ADC信息是针对媒体资源还是CEU。 如果设置为“0”， 则ADC信令消息包含媒体资源的信息。 如果设置为“1”， 则它包含单个CEU的ADC信息。

flow_label_flag (1 bit)：指示是否使用flow_label。 如果该标志被设置为“1”， 则使用flow_label信息。

loss_tolerance：表示媒体资源传输的损失容忍度。 表44列出了loss_tolerance属性的值。

表44　loss_tolerance 的取值

值	描述
0	该媒体资源需要无损传输。
1	该媒体资源允许有损传输。

jitter_sensitivity：指示端到端之间媒体资源传输对网络抖动的敏感程度，表45列出了jitter_sensitivity属性的值。

表45　jitter_sensitivity 的取值

值	描述
0	此媒体资源需要SMTP数据包之间时间间隔保持不变。
1	此媒体资源不需要SMTP数据包之间时间间隔保持不变。

class_of_service：将服务分类为不同的类别，并以特定方式管理每种类型的比特流。该字段指示表46中列出的比特流属性的类型。

表46　class_of_service 的取值

值	描述
0	恒定比特率（CBR）服务类别应保证峰值比特率在任何时候都适用于媒体资源的传输。
1	可变比特率（VBR）服务类别应保证可持续的比特率，并允许媒体资源的峰值比特率在共享信道上具有延迟约束。 此类适用于大多数实时服务，例如视频电话，视频会议，流媒体服务等

Bidirection_indicator：如果设置为“1”，则需要双向传送。 如果设置为“0”，则不需要双向传送。

bitstream_descriptor_vbr：当class_of_service为“1”时，“bitstream_descriptor_vbr”应用于“Bitstream_descriptorType”。

bitstream_descriptor_cbr：当class_of_service为“0”时，“bitstream_descriptor_cbr”应用于“Bitstream_descriptorType”。

n_samples：定义与此特定工作点关联的样本。

spatial_quality：定义与符合ISOBMFF质量格式的样本相关的空间质量。

temporal_quality：定义时间质量，或与从ISOBMFF帧有效值计算出的样本相关的失真。

aggregate_rate：定义与工作点关联的比特率。

sustainable_rate：定义了媒体资源连续传输所应保证的最低比特率。sustainable_rate以每秒字节数表示。

buffer_size：定义用于媒体资源传输的最大缓冲区大小。缓冲区吸收比sustainable_rate更高的多余瞬时比特率，并且buffer_size应足够大以避免溢出。恒定比特率（CBR）资产的buffer_size应为零。buffer_size以字节表示。

peak_rate：定义媒体资源连续传输期间的峰值比特率。peak_rate是每个MFU_period期间的最高比特率。peak_rate以每秒字节数表示。

MFU_period：定义媒体资源连续传输期间的MFU期限。 MFU_period测量为两个连续MFU的第一个字节之间的发送时间的时间间隔。 MFU_period以毫秒表示。

max_MFU_size：表示MFU的最大大小，为MFU_period * peak_rate。 max_MFU_size以字节表示。

flow_label：指示流标识符。流被定义为一个比特流或一组比特流，其可以使用的网络资源根据传输特性或Package中的ADC定义。 它是从“ 0”到“ 127”的隐式序列号。 在会话期间会临时分配一个任意数字给单独的流，用以指示为其分配了处理器和可以使用的网络资源。

packet_id (16 bits)：此字段是一个整数值，可用于区分不同的媒体资源。

9　媒体呈现

9.1　概述

媒体呈现是由客户端访问、用于向用户提供流媒体服务的一组数据的集合。

根据上述SMT的内容模型、传输模型，媒体服务组织有更多的灵活性、丰富性和沉浸式，则在呈现模型中可将媒体服务的组织策略正确、高效地呈现给用户。其中需要解决的关键技术问题包括多源多网络通道传输的媒体在终端如何进行同步，媒体处理呈现如何适配于终端设备，多屏设备之间如何进行互动等等。

9.2　呈现模型

为支持多屏应用及个性化应用等多方面的呈现新需求，提出SMT的呈现模型，见图18。SMT数据经过协议解析引擎分析后，分为信令控制信息和媒体数据信息，该协议解析引擎可以配置专门的智能网关设备，为多屏终端提供接入和控制接口。媒体数据经过媒体处理模块后可分为时序媒体资源和非时序媒体资源，它们将按照信令消息的描述组合为某种媒体服务，从而有序地呈现于不同终端之上。信令控制信息描述了媒体数据的相互关系及呈现信息，一方面指导媒体数据的选择性接收或者主动获取，同时提供媒体数据在终端正确解码处理的提示信息，另一方面在空间布置和时域更新上为不同终端提供媒体呈现策略。呈现引擎根据控制信息正确渲染并呈现所接收的媒体数据，并作为控制器完成多媒体资源、多屏设备之间的同步、互动。

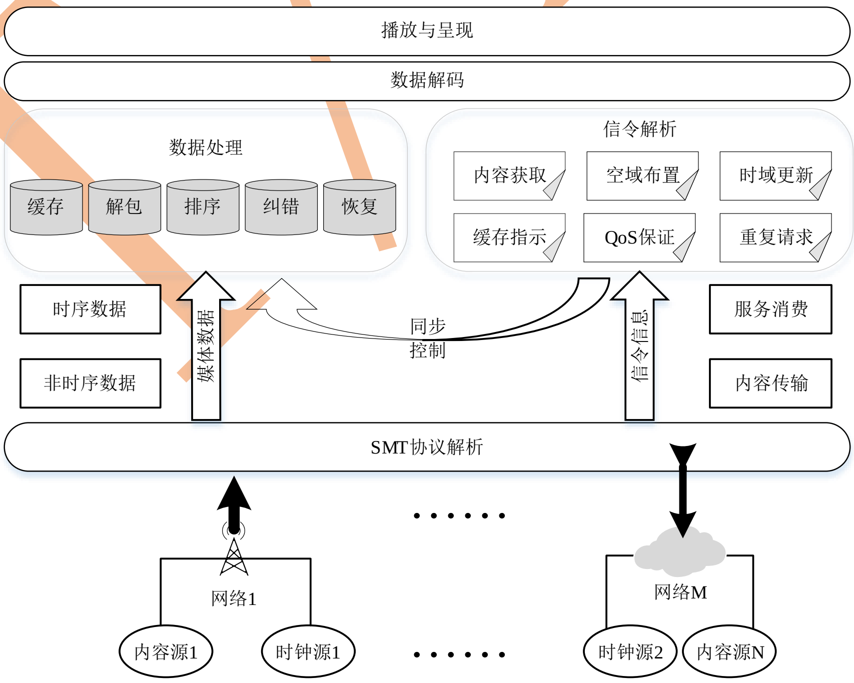


图18　SMT 呈现模型

9.3 媒体呈现信令

9.3.1 概述

本条详细介绍了用于呈现相关的信令信息。在媒体呈现过程中，需提供呈现各个层的信息和媒体内容对应层的信息，以解决SMT灵活布局呈现和个性化呈现的需求。因此，设计了以下几种呈现描述格式。其中，Layer_display_table和Layer_display_update_table在PA Message承载。

9.3.2 媒体呈现层信息表

9.3.2.1 概述

该表针对呈现中每一层的详细信息进行描述，指示了呈现的基本布局。

9.3.2.2 语法

媒体呈现层信息表的语法见表47。

表47 媒体呈现层信息表语法

语法	值	比特位数	备注
Layer_display_table () {			
table_id		8	uimsbf
version		8	uimsbf
length		16	unimbf
number_of_layer	N1	8	unimbf
for(int i = 0; i <N1;i++) {			
layer_id		8	unimbf
device_id		8	unimbf
center_x		16	unimbf
center_y		16	unimbf
width		16	unimbf
height		16	unimbf
display_order		8	unimbf
fitting_type		3	bslbf
adjust_enable_flag		1	bool
reserved	’1111’	4	bslbf
transparency		8	unimbf
}			
}			

9.3.2.3 语义

table_id：描述了该表的识别符。
version：描述了该表的版本。更新后的信令表有新的版本号，可以替换原有的信令表。
length：描述了该表的长度，从本字段下个字节开始到该表最后一个字节为止。
number_of_layer：描述了该表所描述的呈现层的数量。
layer_id：描述了该表内当前描述的呈现层的标号。

device_id：描述了该表内当前描述的呈现层所对应呈现的设备号。该号为‘0’时表示呈现在默认设备上，当为1或其他值时呈现在辅助设备或者次级/低优先级设备上，且数字越大优先级越低。

center_x：描述了该表内当前描述的呈现层中呈现媒体内容的区域的中心位置横坐标。用显示区域中心的像素在整个层水平像素中的百分比来标定。

center_y：描述了该表内当前描述的呈现层中呈现媒体内容的区域的中心位置纵坐标。用显示区域中心的像素在整个层垂直像素中的百分比来标定。

width：描述了该表内当前描述的呈现层中呈现媒体内容的区域的宽度。用显示区域的像素在整个层水平像素中的百分比来标定。

height：描述了该表内当前描述的呈现层中呈现媒体内容的区域的高度。用显示区域的像素在整个层垂直像素中的百分比来标定。

display_order：描述了该表内当前描述的呈现层在所有呈现层中的显示顺序。该顺序为‘0’时表示默认层，标号小的在底层，标号大的在上层。中间号可以空缺，但是不能重复。

fitting_type：描述了该表内当前描述的呈现层播放媒体内容时的画面适配类型。适配类型为‘0’表示铺满，即改变CEU分辨率长宽比，经过拉伸后铺满整个指定区域；适配类型为‘1’表示放大适配，即不改变CEU分辨率的长宽比，将画面从最小放大，直到宽\高与屏幕的左右\上下一个方向上适配，其余部分补黑边；适配类型为‘2’表示缩小适配，即不改变CEU分辨率的长宽比，将画面从最大缩小，直到宽\高与屏幕的左右\上下一个方向上适配，其余部分裁剪；适配类型为‘3’表示原画，即不改变CEU分辨率和长宽比，位置在指定区域正中心，若与显示区域不符，则不足部分补黑或超出部分剪裁；适配类型为‘4’表示全景，即按照全景视频的播放要求，适配到层中进行播放。适配类型不限于以上5种。见表48。

表48　画面适配类型

值	描述
000	铺满
001	放大适配
010	缩小适配
011	原画
100	全景
101-111	保留

adjust_enable_flag：描述了该表内当前描述的呈现层是否可调整的标记。该标记为‘0’时，表示用户端不可对该层进行调整；该标记为‘1’时，表示用户端可对该层进行设备、显示区域大小、位置、透明度、适配类型等进行调整。

transparency：描述了该表内当前描述的呈现层的透明程度。该字段的值表示百分号前的数值，有效范围为0~100%。

9.3.3　媒体呈现层更新信息表

9.3.3.1　概述

该表针对呈现中需要更新的层显示信息进行描述，指示了呈现布局的更新信息。该表用于对布局调整较小的情况。当布局有大范围调整时，可重新发送新版本的Layer_display_table来整体更新布局。

9.3.3.2 语法

媒体呈现层更新信息表的语法见表49。

表49 媒体呈现层更新信息表语法

语法	值	比特位数	备注
Layer_display_update_table () {			
table_id		8	uimsbf
version		8	uimsbf
length		16	unimbf
layer_delete_flag		1	bool
layer_add_flag		1	bool
layer_display_order_flag		1	bool
layer_adjust_flag		1	bool
reserved1	’1111’	4	bslbf
if(layer_delete_flag){			
number_of_layer	N1	8	unimbf
for(int i = 0; i <N1;i++) {			
layer_id		8	unimbf
}			
}			
if(layer_add_flag){			
number_of_layer	N2	8	unimbf
for(int i = 0; i <N2;i++) {			

表 49 （续）

语法	值	比特位数	备注
new_layer_id		8	unimbf
device_id		8	unimbf
center_x		16	unimbf
center_y		16	unimbf
width	’1111’	16	unimbf
height		16	unimbf
display_order		8	unimbf
fitting_type		3	bslbf
adjust_enable_flag		1	bool
reserved2		4	bslbf
transparency		8	unimbf
}			
}			
if(layer_display_order_flag){			
number_of_layer	N3	8	unimbf
for(int i = 0; i <N3;i++) {			
layer_id		8	unimbf
new_layer_display_order		8	unimbf
}			
}			
if(layer_adjust_flag){			
number_of_layer	N4	8	unimbf
for(int i = 0; i <N4;i++) {			
layer_id		8	unimbf

device_id		8	unimbf
center_x		8	unimbf
center_y		8	unimbf
width		8	unimbf
height		8	unimbf
display_order		8	unimbf
fitting_type		3	bslbf
adjust_enable_flag		1	bool
reserved3	’1111’	4	bslbf
transparency		8	unimbf
}			
}			
}			

9.3.3.3 语义

layer_delete_flag：描述了是否有删除层。该标记为‘0’时表示没有删除层，该标记为‘1’时表示有删除层。如有删除层，则应给出删除层的层号。

layer_add_flag：描述了是否有增加层。该标记为‘0’时表示没有增加层，该标记为‘1’时表示有增加层。如有增加层，则应给出增加层的完整信息。

layer_display_order_flag：描述了是否有层需要改变显示顺序。该标记为‘0’时表示没有层需要改变顺序，该标记为‘1’时表示有层需要改变顺序。如有层需要改变顺序，则应给出需调整层的层号和新给出的层显示顺序。

layer_adjust_flag：描述了是否有调整层。该标记为‘0’时表示没有调整层，该标记为‘1’时表示有调整层。如有调整层，则应给出调整层调整后的参数信息。

9.3.4 CEU 消费描述符

该描述符对媒体内容CEU所应呈现的层信息进行描述，并描述了CEU可以替换呈现、复制呈现的层信息。

9.3.4.1 语法

CEU消费描述符的语法见表50。

表50　CEU 消费描述符语法

语法	值	比特位数	备注
CEU_consumption_descriptor() {			
descriptor_tag		16	uimsbf
descriptor_length		16	uimsbf
number_of_CEUs	N1	8	unimbf
for(i = 0; i <N1;i++) {			
CEU_sequence_number		32	unimbf
number_of_layer	N2	8	unimbf
for(i = 0; i <N2;i++){			
layer_id		8	unimbf
}			
layer_exchange_flag		1	bslbf
layer_copy_flag		1	bslbf
reserved	’111111’	6	bslbf
if(layer_exchange_flag){			

number_of_exchange_layer	N3	8	unimbf
for(i = 0; i <N3;i++) {			
exchange_layer_id		8	unimbf
}			
}			
if(layer_copy_flag){			
number_of_copy_layer	N4	8	unimbf
for(i = 0; i <N4;i++) {			
copy_layer_id		8	unimbf
}			
}			
}			
}			

9.3.4.2　语义

descriptor_tag：标识该描述子类型的标签名，能够定位该描述子所属的类型。

descriptor_length：标识该描述子的字节长度，从本字段下个字节开始到该描述子最后一个字节为止。

number_of_CEUs：标识当前CEU应呈现的层数量。

CEU_sequence_number：标识当前CEU的序号，Asset中的第一个CEU序号应为‘0’，后续每个CEU增加‘1’，序号在一个Asset内是唯一的。

number_of_layer：标识当前CEU应呈现的层号。

layer_exchange_flag：标识当前CEU可否在正确呈现到某层的情况下，与其他层呈现的CEU交换呈现层。该标记为‘0’时表示仅可以按照该描述子列举的层号呈现，该标记为‘1’时表示用户端可以将该CEU与其它CEU换层呈现。

layer_copy_flag：标识当前CEU可否在正确呈现到某层的情况下，复制呈现到其它层。该标记为‘0’时表示仅可以按照该描述子列举的层号呈现，该标记为‘1’时表示用户端可以将该CEU复制呈现到其他层。在复制呈现时，客户端应覆盖原层的CEU内容。

number_of_exchange_layer：标识可以与当前CEU换层呈现的层数量。

exchange_layer_id：标识可以与当前CEU换层呈现的层号。

number_of_copy_layer：表示可以被当前CEU复制呈现的层数量。

copy_layer_id：表示可以被当前CEU复制呈现的层号。

10　HTTP/2 直播

10.1　概述

本条介绍了SMT在HTTP/2直播流媒体中的应用，概念架构见图19。首先，利用HTTP Upgrade机制，在客户端与服务端之间建立起一个全双工媒体流通道；然后，客户端发送统一资源标识符和推送指令，向服务器请求媒体数据，其中推送指令用于通知服务器如何向客户端推送媒体数据；最后，服务器接收到客户端发送的请求后，先向客户端发送请求的媒体数据，并启动推送循环，根据推送策略向客户端推送媒体数据。

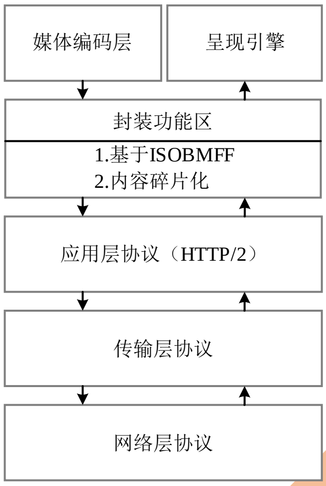


图19　SMT-HTTP/2 概念架构

SMT-HTTP/2直播的工作流见图20。在一次SMT-HTTP/2直播会话中，客户端首先请求时序媒体资源的描述(Edit_list)，然后请求媒体资源的内容碎片，并发送推送指令。在获取请求的时序媒体资源的描述(Edit_list)后，客户端开始通过指定相应内容碎片的URL和推送指令，向服务器请求内容碎片，内容碎片可以是视频片段、音频片段或字幕片段等时基媒体内容碎片。服务器响应请求，向客户端发送URL指定的内容碎片，并启动推送循环，根据推送指令指定的推送策略向客户端推送内容碎片。通常，当获取一定数量的媒体数据后，客户端开始播放视频，并循环上述过程直至整个媒体流会话结束。

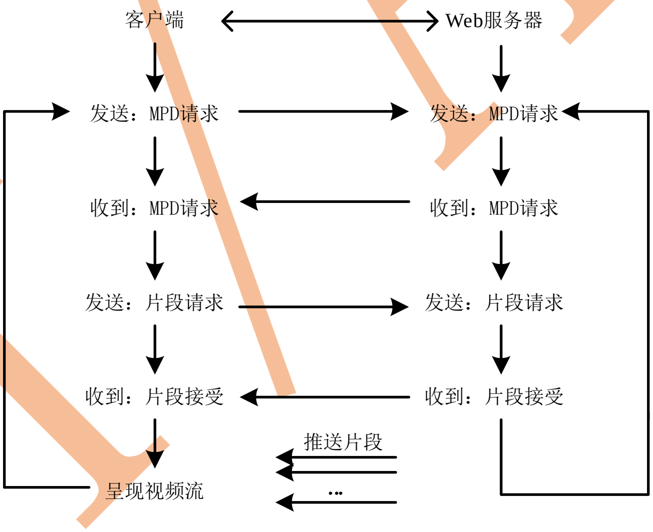


图20　SMT-HTTP/2 直播会话流程图

10.2　系统架构

基于HTTP/2的端到端SMT流媒体系统架构主要由三部分组成，见图21：

a) 源服务器——存储流媒体资源，配置一个或多个推送策略的 HTTP/2 网络服务器；

b) SMT 客户端——获取并播放视频流，可由 HTTP/2 浏览器和 SMT 播放器组成，也可由一个桌面客户端组成；

c) 内容分发网络——连接源服务器和客户端，由多个配置一个或多个推送策略的 HTTP/2 网络缓存服务器组成。

在上述系统中，有两个持久 HTTP/2 连接，分别存在于客户端和 CDN，以及 CDN 和源服务器之间。另外，在直播场景中，客户端与源服务器间应建立 HTTP/2 隧道连接。HTTP/2 流媒体服务器（源端/缓存）还可以主动向客户端（或 CDN）推送媒体数据。

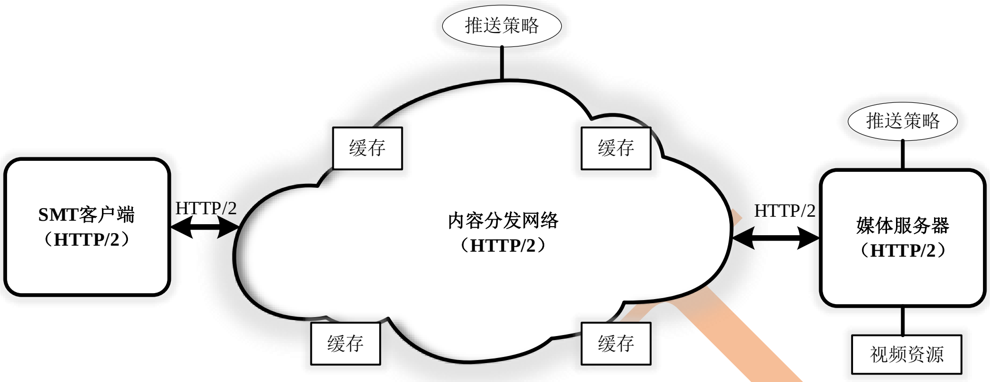


图21　SMT-HTTP/2 直播系统架构

10.3　数据类型

10.3.1　概述

本条给出基本数据类型的定义，及其在HTTP/2协议中的呈现方式。

10.3.2　通用类型

表51　本文件中数据类型的定义

数据类型	基本类型	描述
BinaryObject	N/A	二进制对象，0 或多个非类型化字节组成
Boolean	N/A	布尔对象，取值为 true/false
MPD	BinaryObject	媒体呈现描述，AVS-DASH MPD
PushDirective	String	推送指令，用于描述在本次流媒体会话中请求配置的推送策略
PushAck	String	推送确认，用于响应来自客户端的推送请求
Segment	BinaryObject	片段，AVS-DASH 初始化片段或媒体片段
String	N/A	字符串，由 GB/T 18030-2005 标准规定的 Unicode 字符组成
URI	String	统一资源标识，参见 RFC 3986
URLList	String	URL 列表，见**错误!未找到引用源。**
URLTemplate	String	推送 URL 模板及相关参数，用于客户端指定在一次推送事务中请求推送的片段列表，见 **错误!未找到引用源。**

10.3.3　推送类型

PushType用于描述一个推送策略，包括一个用于标识推送策略的类型名称以及可选的推送参数。PushType的巴克斯范式格式定义如下：

PUSH_TYPE = PUSH_TYPE_NAME [OWS ";" OWS PUSH_PARAMS]

PUSH_TYPE_NAME = DQUOTE <URN> DQUOTE

PUSH_PARAMS = PUSH_PARAM *(OWS ";" OWS PUSH_PARAM)

PUSH_PARAM = 1*PPCHAR

其中，<URN>语法在RFC 2141中定义，该URN的合法取值见表51OWS在RFC7230中定义，表示可选的空白符。

10.3.4　推送指令

PushDirective用于客户端请求服务器遵照特定推送策略推送未来一个或多个片段。PushDirective的巴克斯范式格式定义如下：

PUSH_DIRECTIVE = PUSH_TYPE [OWS ";" OWS QVALUE]

PUSH_TYPE　= < 本规范**错误!未找到引用源。**中定义>

QVALUE = < RFC 7231中定义>

当一个请求中包含多个推送指令时，客户端可使用RFC 7231中描述的用于内容协商的质量值（“qvalue”）。客户端可以为希望优先运用的推送指令赋予更高的质量值，而为其他推送指令赋予较低的质量值。

10.3.5　推送确认

服务器返回客户端PushAck，用于指示在推送会话中服务器将会遵照特定推送策略。PushAck的巴克斯范式格式定义如下：

PUSH_ACK = PUSH_TYPE

PUSH_TYPE　= < 本规范**错误!未找到引用源。**中定义>

10.3.6　URL 列表

URLList描述一组特定的URL分隔列表。 客户端可以使用URL列表来明确地表示在推送会话中请求推送的媒体片段。URL列表描述了将要在推式事务中推送的片段序列。

URLList的巴克斯范式格式定义如下：

URL_LIST = LIST_ITEM *(OWS ";" OWS LIST_ITEM)

LIST_ITEM = 1*PPCHAR

每个列表元素格式RFC 3986中定义的URL。如果URL是相对形式，则认为它相对于正在请求的媒体片段。

10.3.7　URL 模板

URLTemplate通过模板以及扩展模板所需的相应参数描述了一组特定的URL。客户端可以使用URLTemplate来明确地请求服务器在推送事务期间需要推送的片段。URLTemplate列表由一个字符串构成，其中每个URLTemplate可以被参数化为一个或多个URL值。所有URLTemplate被完全解析后得到的完整URL列表即为此次推送事务中需要推送的片段序列。

URLTemplate格式源自IETF RFC 6570中定义的“级别1”URI模板方案。此外，模板字符串可以附带一个IEEE 1003.1-2008中定义的格式标签，遵循：%0 [width] d

其中，width参数是一个无符号整数，提供最小打印字符数。如果打印值小于此数字，则结果将用零填充。而即使结果较大，该值也不会被截断。

URLTemplate的巴克斯范式格式定义如下：

URL_TEMPLATE = TEMPLATE_ITEM *(OWS ";" OWS TEMPLATE_ITEM)

TEMPLATE_ITEM = SQUOTE TEMPLATE_ELEMENT SQUOTE [OWS ":" OWS "{"OWS TEMPLATE_PARAMS OWS "}"]

TEMPLATE_ELEMENT = CLAUSE_LITERAL [CLAUSE_VAR [CLAUSE_LITERAL]]

CLAUSE_LITERAL = 1*PPCHAR

CLAUSE_VAR = "{%0" 1*DIGIT "d}" / "{}"

TEMPLATE_PARAMS = RANGE_PARAM / LIST_PARAM/ COMPRESSION_SEQUENCE_PARAM

RANGE_PARAM = 1*DIGIT OWS "-" OWS 1*DIGIT

LIST_PARAM = 1*DIGIT *(OWS "," OWS 1*DIGIT)

COMPRESSION_SEQUENCE_PARAM=START_VALUE","DIFFERENCE_VALUE":"REPEAT_NUMBER *("," DIFFERENCE_VALUE":"REPEAT_NUMBER)

每个模板元素的格式参见RFC 3986中6定义的URL构成，且最多包含一个用于参数化的宏。如果模板元素是相对形式的，则对应URL相对于正在请求的片段取值。使用所提供的模板参数值展开变量"{}"，可以指定一个特定的具有不同片段编号或片段呈现时间的URL列表或者URL取值范围。变量"{}"及模板参数值是可选的，可以仅提供一个简单的URL列表。

通过解析每个模板项中提供的模板参数可以生成URL列表。模板项定义以下模板参数，包括：范围参数、列表参数、压缩序列参数。其中，压缩序列参数格式定义为："s, d_1:r_1,d_2:r_2, …, d_i:r_i, …, d_n:r_n"，其中，s指示起始参数值，d指示两个连续参数值间的差值，r指示相同差值的持续次数，n为正整数。

10.4 推送策略

表52给出了本规范中定义的推送类型相关描述：

表52 推送类型有效值

推送类型	描述
urn:mpeg:dash:fdh:2016:push-fast-start	快速启动类型推送策略，指示在返回MPD数据同时推送初始化数据以及可选的给定数量的媒体片段。 - 接收快速启动类型推送指令，服务器可以推送部分或者所有可用的与所请求MPD相关的初始化片段以及可选的媒体片段 - 接收快速启动类型推送确认指令，客户端可获知服务器将要推送的部分或者所有可用的初始化片段以及可选的媒体片段
urn:mpeg:dash:fdh:2016:push-list	列表类型推送策略，指示推送URL列表描述的片段。 - 接收列表类型推送指令，服务器可根据URL列表确定需要推送的片段 - 接收列表类型推送确认指令，客户端可获知服务器将要推送的片段序列
urn:mpeg:dash:fdh:2016:push-next	数量类型推送策略，指示按时间顺序推送以所请求片段为初始索引的后续K个片段。 - 接收数量类型推送指令，服务器可以推送所请求片段的后续K个片段 - 接收数量类型推送确认指令，客户端可获知服务器将要推送的所请求片段的后续K个片段

urn:mpeg:dash:fdh:2016:push-none	none类型推送策略，指示不会发生推送。 - 接收none类型推送指令，服务器不应推送任何片段 - 接收none类型推送确认指令，客户端可获知服务器将不会推送任何片段
urn:mpeg:dash:fdh:2016:push-template	模板类型推送策略，指示推送URL模板描述的片段。 - 接收模板类型推送指令，服务器可根据URL模板确定需要推送的片段 - 接收模板类型推送确认指令，客户端可获知服务器将要推送的片段序列
urn:mpeg:dash:fdh:2016:push-time	时间类型推送策略，指示按时间顺序推送后续若干片段，直至片段时间（片段第1帧的呈现时间）超过时间T。 - 接收时间类型推送指令，服务器可以推送给定播放时间的片段 - 接收时间类型推送确认指令，客户端可获知服务器将要推送给定播放时间的片段序列

每一推送策略仅可适用于片段请求和/或MPD请求。表53描述了可应用于特定推送策略的请求类型，以及相应的推送参数：

表53　请求类型及推送参数有效值

推送类型	请求类型	推送参数
urn:mpeg:dash:fdh:2016:push-fast-start	MPD	URLList
urn:mpeg:dash:fdh:2016:push-list	Segment	URLList
urn:mpeg:dash:fdh:2016:push-next	Segment	INTEGER
urn:mpeg:dash:fdh:2016:push-none	MPD or Segment	N/A
urn:mpeg:dash:fdh:2016:push-template	Segment	URLTemplate
urn:mpeg:dash:fdh:2016:push-time	Segment	INTEGER

10.5　HTTP/2 协议绑定

10.5.1　推送指令绑定

HTTP/2协议中，HTTP消息头Accept-Push-Policy用于携带PushDirective推送指令。PushDirective数据类型在本规范10.3.3定义。

根据HTTP协议规定，多个PushDirective推送指令可以使用上述多个HTTP头域携带，也可以将多个PushDirective推送指令作为逗号分隔列表组合到单个HTTP头域。

10.5.2　推送确认绑定

HTTP/2协议中，HTTP消息头Push-Policy用于携带PushAck推送确认指令。PushAck数据类型在本规范10.3.4定义。

根据HTTP协议规定，多个PushAck推送确认指令可以使用上述多个HTTP头域携带，也可以将多个PushAck推送确认指令作为逗号分隔列表组合到单个HTTP头域。

10.6　消息定义

10.6.1　概述

本条以客户端与服务器端之间通信消息的形式定义本文件提出的协议。在通信过程中，客户端(服务器)发送的消息将触发服务器（客户端）的操作，随后服务器（客户端）回发一个响应消息或一个错误提示。本条定义的消息集合为AVS-DASH现有功能的扩展，即若没有可选参数或扩展参数，本条定义协议的基本形式与AVS-DASH over HTTP1.x兼容。

本条假定消息为异步的，即没有同步返回值；而且，消息响应也被定义为独立的消息形式。同时，

消息流的方向也会明确给出（从服务器端至客户端，或从客户端至服务器端），见图22。在消息传输过程中，可能出现的错误或异常亦作为消息定义的一部分给出。

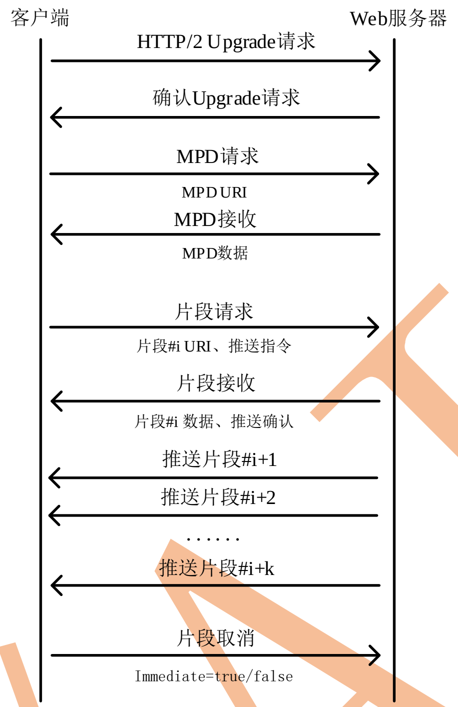


图22　消息流

10.6.2　MPD 请求

MPD请求消息发起一个DASH MPD文件的请求。在MPD请求中禁止出现推送指令。

客户端使用标准HTTP GET请求发送该消息。MPDURI参数以GET请求路径的形式提供。

表54　MPD 请求消息的定义

消息名称	get_mpd		
流动方向	Client → Server		
参数列表	参数名称	参数类型	说明
	mpd_uri	URI	请求的 MPD 文件对应的完整 URI
先决条件	无		
后置条件	MPD 请求被发起，并等待服务器端发送的所有 new-mpd 消息均到达客户端。		
异常错误	无		

10.6.3　片段请求

片段请求消息发起一个DASH 片段的请求。与其他片段请求不同，基于FDH的片段请求使用一个推送指令通知服务器端去主动推送一个或多个后续片段。

客户端使用标准HTTP GET请求发送该消息。片段URI参数以GET请求路径的形式提供。若存在推送指令，则必须以10.3.3中定义的方式发送。

表55　片段请求消息的定义

消息名称	get_segment		
流动方向	Client → Server		
参数列表	参数名称	参数类型	说明
	mpd_uri	URI	请求的视频片段对应的完整 URI
	push_directive	PushDirective	指定推送策略和需要推送的视频片段。 若值为 NULL，且已有推送指令消息发出，则保持推送策略不变，否则表明客户端要求不对视频片段进行推送。
先决条件	客户端已获取有效的 MPD 文件		
后置条件	• 片段请求被发起，并等待服务器端发送的所有 new-segment 消息均到达客户端；new_segment 消息表明服务器响应了客户端的片段请求 • 在获得请求的片段之后，客户端将可能收到一个或多个服务器端推送的片段		
异常错误	FDH 不可用。指定的推送指令被触发，但服务器检测到全双工通道不存在或失效		

10.6.4　MPD 接收

该消息表征服务器对之前收到的来自客户端的get_mpd消息的响应。

HTTP/2服务器当且仅当在响应客户端MPD请求消息时，发送该消息。

在响应客户端MPD请求消息时，与HTTP1.x DASH一致，服务器需将MPD数据作为HTTP响应报文的主体部分发送。

表56　MPD 接收消息的定义

消息名称	new_mpd		
流动方向	Server → Client		
参数列表	参数名称	参数类型	说明
	mpd	MPD	服务器返回的 MPD 文件
先决条件	客户端通过发送 get_mpd 消息请求 MPD 文件		
后置条件	客户端将解析获取的 MPD 文件		
异常错误	无		

10.6.5　片段接收

该消息表征服务器对之前收到的来自客户端的get_segment消息的响应。对于单个片段请求，根据会话中配置的推送策略，服务器可能发送多个响应。

HTTP/2服务器当且仅当在响应客户端片段请求消息时或发起片段推送时，发送该消息。

在响应客户端片段请求消息时，与HTTP1.x DASH一致，服务器需将片段数据作为HTTP响应报文的主体部分发送。

若客户端片段请求消息中指定了一个推送指令，且服务器支持该推送指令，则服务器必须按照10.3.4中定义的方式发送推送确认。

若客户端片段请求消息中未指定推送指令，或服务器不支持该推送指令，则服务器不可在响应中包含推送确认。

客户端必须假定，若服务器未返回推送确认，则服务器不会推送未请求的片段或MPD更新。

表57　片段接收消息的定义

消息名称	new_segment		
流动方向	Server → Client		
参数列表	参数名称	参数类型	说明
	segment	Segment	服务器返回的片段
	push_ack	PushAck	服务器将遵循的推送策略
先决条件	客户端通过发送 get_segment 消息请求片段		
后置条件	客户端将解析获取的片段，并处理媒体流		
异常错误	无		

10.6.6　片段取消

该消息表征客户端请求服务器取消正在进行的推送事务。若没有正在进行的推送事务，则该消息不起作用。

表58　片段取消消息的定义

消息名称	segment_cancel		
流动方向	Client → Server		
参数列表	参数名称	参数类型	说明
	immediate	Boolean	若值为 true，则服务器应立即停止当前片段数据的传送，并取消推送事务。 若值为 false，则服务器应在完成当前片段的传送后，取消推送事务。
先决条件	• 客户端已经发起推送事务 • 服务器没有完成客户端请求的推送事务		
后置条件	服务器不再维持当前推送事务，不再推送片段。		
异常错误	无		

11　自适应前向纠错编码

11.1　概述

为了在 IP 网络环境下可靠传输，SMT 提供了应用层前向纠错编码机制。该机制可视为发送模块的构建模块。CEU 分包生成的 SMT 数据包被传递到 SMT 的 FEC 模块，FEC 模块产生对应的恢复符号和 FEC 负载 ID，之后利用 SMT 协议将恢复符号与原始 SMTP 数据包发送出去。FEC 配置信息提供 FEC 编码流的识别信息并指定 FEC 编码结构。它将传递给 SMT 接收实体进行 FEC 操作，见图 23。

由SMT发送实体确定需要保护的Asset以及FEC源码流的数目。一个或多个Asset可作为一个FEC源码流进行FEC保护，FEC源码流和其配置信息将传送给SMT FEC方案进行保护。SMT FEC方案模块利用FEC编码产生恢复符号组成一个或多个FEC恢复流。恢复流与FEC负载ID一同传送给SMT协议。SMT协

议将FEC源和恢复数据包传送给接收实体。在接收实体中，SMT协议将FEC源码流和相关的恢复流一同传送给SMT FEC方案模块，该模块将恢复出SMTP包。

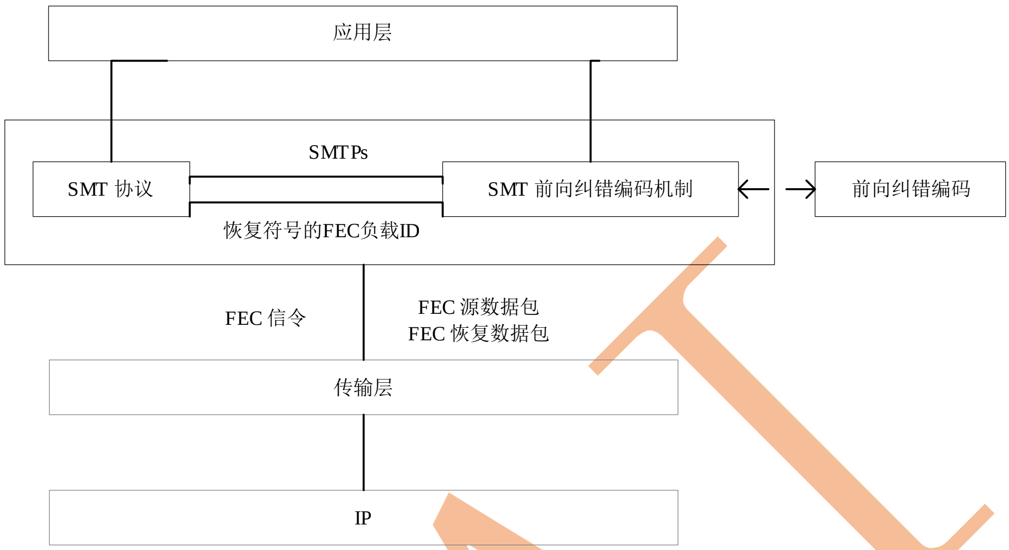


图23　SMT 应用层前向纠错编码框架

11.2　自适应前向纠错编码机制

11.2.1　概述

自适应前向纠错编码机制，基于所传输源媒体数据的性质以及当前信道的具体情况，划分相应的优先级比例，给出优先级指示，根据优先级指示和优先级比例动态选择编码矩阵，分配各数据的保护强度，使其既能保护重要数据又不会造成过大的冗余。在不同的信道条件下，都能最大程度恢复数据。其具体机制见图24，25。

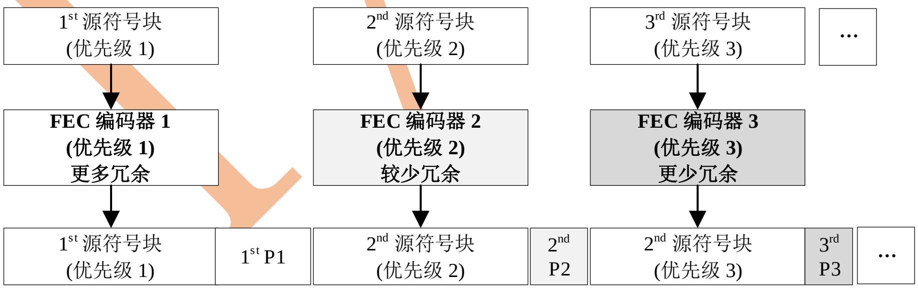


图24　不同优先级源符号块自适应前向纠错编码结构

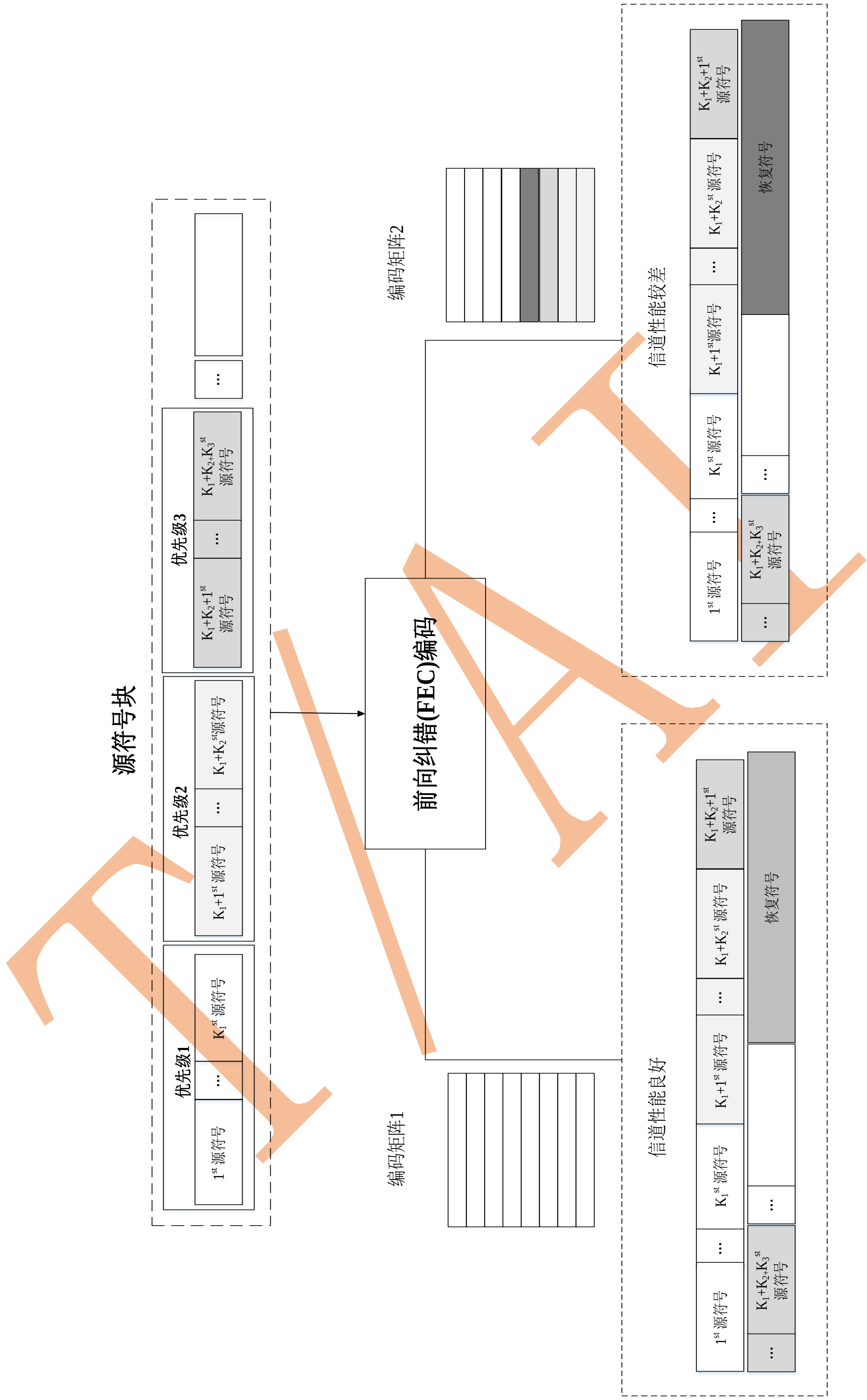


图25　同一源符号块不同源符号的不等差错保护

11.2.2 语法

自适应前向纠错编码信令消息的语法见表59。

表59 FEC 信令

语法	值	比特位数	备注
AL_FEC ( ){			
message_id		16	uimsbf
version		8	uimsbf
length		16	uimsbf
message_payload{			
fec_flag		1	bslbf
private_fec_flag		1	bslbf
reserved	‘111111’	6	bslbf
if (fec_flag==1) {			
length_of_fec_flow_descriptor		16	uimsbf
fec_flow_descriptor() {			
number_of_fec_flows	N1	8	uimsbf
for (i=0; i<N1 ; i++) {			
fec_flow_id		8	uimsbf
source_flow_id		8	uimsbf
number_of_assets	N2	8	uimsbf
for (j=0; j<N2 ;j++) {			
packet_id		16	uimsbf
}			
fec_coding_structure		4	uimsbf
ssbg_mode		2	uimsbf
reserved	‘1’	1	bslbf
length_of_repair_symbol		16	uimsbf
if (ssbg_mode == 2) {			
num_of_repair_symbol_per_packet		16	uimbsf
num_of_symbol_element_per_source_symbol		16	uimbsf
}			
if (fec_coding_structure == 0001) {			
number_of_class	N3	8	uimsbf
if (private_fec_flag == 1) {			
private_flag		1	bslbf
private_field_length	N4	7	bslbf
private_field		N4*8	uimsbf
}			
repair_flow_id		8	uimbsf
fec_code_id_for_repair_flow		8	uimbsf
maximum_k_for_repair_flow		24	uimbsf
maximum_p_for_repair_flow		24	uimbsf
protection_window_time		32	uimbsf
protection_window_size		32	uimbsf
}			
}			
}			
}			
}			
}			

11.2.3 语义

message_id：表示FEC信令的标识符，见表24信息ID值。

version：表示FEC信令的版本。SMT接收实体利用此字段废弃旧版本，采纳新版本的配置。。

length：表示FEC信令的长度。这个字段的长度是16位。以字节表示FEC消息的长度，从下一字段的第一字节到本信令的最后字节，不能为0。

fec_flag：表示是否有至少一个源码流需要进行FEC保护。如果每个源码流均不需FEC保护，则本字段后的所有字节均不出现。

表60　fec_flag 的值

值	描述
0	没有 FEC 源码流
1	至少有一个 FEC 源码流

private_fec_flag：标识私有FEC配置信息是否定义。若该字段为0， 则没有相应字段；若为1，则存在私有FEC相关字段。

length_of_fec_flow_descriptor：fec_flow_descriptor 所占字节数。

number_of_fec_flows：FEC编码流的数量。

fec_flow_id ：用于识别FEC编码流的任意整数。每个FEC编码流由有关联的一个FEC的源码流和一个或多个FEC恢复码流组成。

source_flow_id：用于识别FEC源码流的任意整数。每个FEC源码流与一个FEC编码流相关。

number_of_assets：属于同一个FEC源码流的Asset的数量。

packet_id：SMTP中包头的packet_id。

ssbg_mode：表示已提供的SSBG模式，该字段的值表示的含义见表61。

表61　ssbg_mode 的值

值	描述
b00	ssbg_mode0 (SMTP 包大小固定)
b01	ssbg_mode1 (SMTP 包大小可变)
b10	ssbg_mode2 (SMTP 包大小可变)
b11	保留

length_of_repair_symbol：以字节表示恢复符号的长度。

num_of_repair_symbol_per_packet：在使用ssbg_mode2模式时，一个FEC恢复数据包中所携带的恢复符号的数量。

num_of_symbol_element_per_source_symbol：在ssbg_mode2模式下，一个源符号所包含的源符号项的数量。

fec_coding_structure：表示对FEC源码流采用的FEC编码结构，该字段的值表示的含义见表62。

表62　fec_coding_structure 的值

值	描述
b0000	不采用 AL-FEC 编码模式
b0001	应用自适应 FEC 编码模式
b0010 ~ b1111	保留

number_of_class：用于指示划分优先级的数目。

repair_flow_id：用于识别FEC恢复码流的任意整数。它表示FEC恢复数据包的SMTP包头中packet_id。恢复码流的packet_id表示为repair_flow_id。

fec_code_id_for_repair_flow：FEC恢复码流的FEC编码标识符。

private_flag：当设置为‘1’时，表示存在私有FEC的描述信息，比如特定FEC算法的参数等。当设置为‘0’时，表示没有这样的私有字段。

表63　private_flag 的值

值	描述
b0	指定的 FEC 架构没有私有字段
b1	指定的 FEC 架构含有私有字段

private_field_length：以字节表示私有字段的长度。若private_flag等于0，则该值应置0表示没有私有字段。

private_field：该字段包含特定FEC算法参数的私有FEC信息。

maximum_k_for_repair_flow：表明与FEC恢复码流相关的源符号块中所允许的最大源符号数量。

maximum_p_for_repair_flow：表明与FEC恢复码流相关的恢复符号块中所允许的最大恢复符号数量。

protection_window_time：该字段以毫秒表示从第一个FEC源或恢复数据包块到最后一个FEC源或恢复数据包块的最大持续时间。若该值置0，将会使用“protection_window_size”。

protection_window_size：该字段表示该FEC编码流（一个FEC源码流和对应的一个或多个FEC恢复码流）的编码块（源符号块和恢复符号块）的最大字节数。若该值置0，则将会使用“protection_window_time”。

11.3　编码符号块格式

11.3.1　概述

前向纠错的编码块由源符号块和恢复符号块组成。源符号块根据给定的SSBG（source symbol block generation）模型，从源数据包中生成。相关的源符号块通过特定的FEC编码方式编码产生恢复符号块。编码符号块格式见图26。

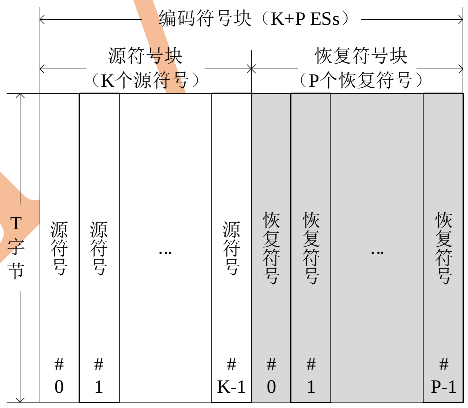


图26　编码符号块格式

11.3.2　源数据包组

为FEC要保护的数据。一个FEC源码流被划分为1个或多个源数据包组。一个源数据包组由指定数量的FEC字段值为1的SMTP包（见7.3.2.2）组成。源数据包组被转化为源符号组。

11.3.3 源符号块格式

一个源符号块由指定数量的源符号组成，源符号由源数据包生成。目前有三种生成方式（SSBG）：ssbg_mode0、 ssbg_mode1 和 ssbg_mode2。ssbg_mode0适用于固定大小的SMTP包，而ssbg_mode1和ssbg_mode2适用于可变大小的SMTP包。

采用ssbg_mode0模式时，因所有SMTP包大小相同，源符号块和源数据包组完全相同。此时源数据组中SMTP包的数量与源符号块中源符号数量一致，且对应编号相同，见图27。

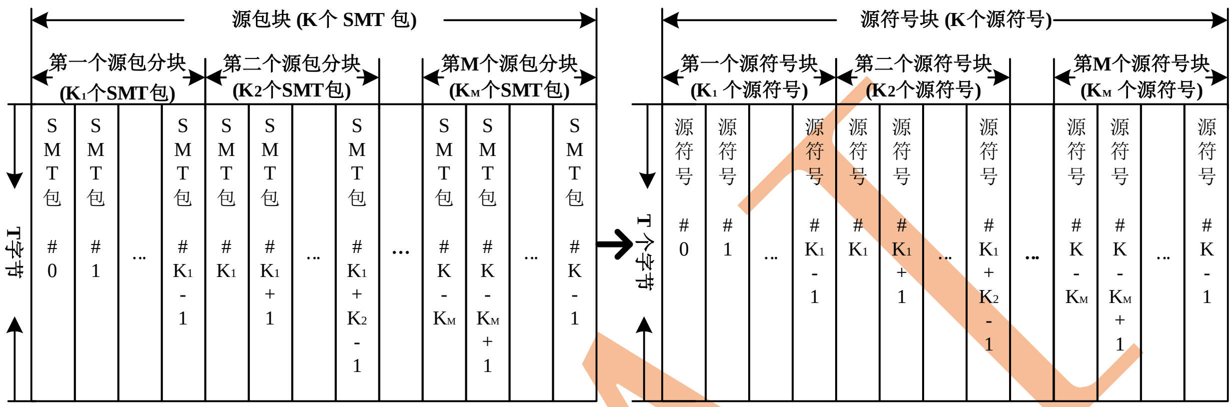


图27 不填充生成源符号块(ssbg_mode0)

ssbg_mode1模式生成源符号块时，与ssbg_mode0基本一致，源符号长度相同，但因源SMTP包大小不同，因此需要添加填充字节，此时每个源符号的开始2字节为SMTP包的大小（为网络字节序）， 接着是SMTP包的数据，之后为全0的填充字节，见图28。

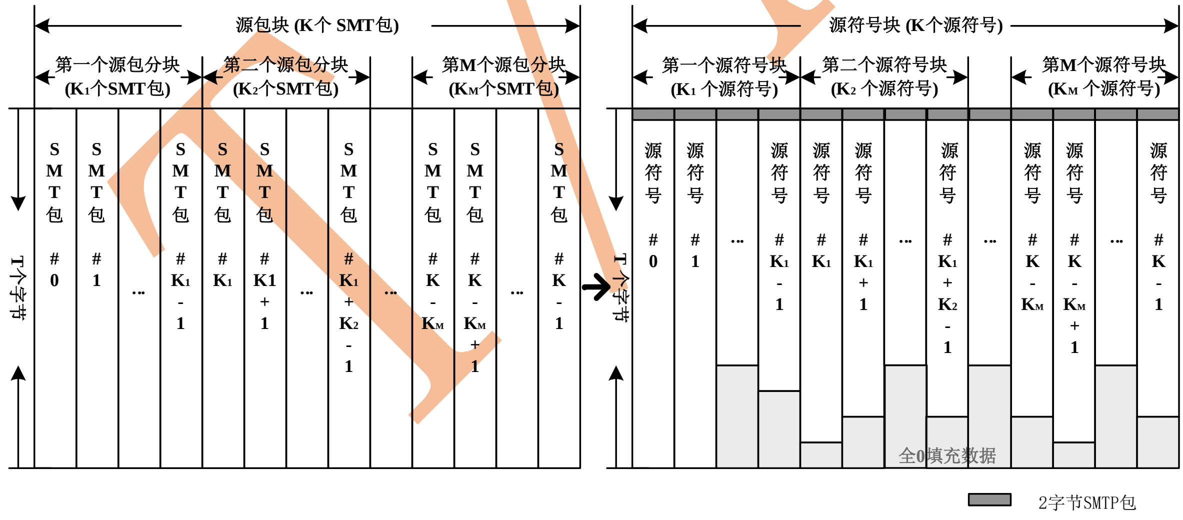


图28 填充生成源符号块（ssbg_mode1）

ssbg_mode2模式时，源符号块仍然由源数据包块进行字节填充生成，但填充方式与ssbg_mode1不同。ssbg_mode1时，源符号的大小必须大于最长的SMTP数据包的长度，且一个源SMTP数据包与一个源符号一一对应。ssbg_mode2时源符号的大小则可以小于SMTP数据包的长度，此时一个源SMTP数据包可能对应多个源符号。单个源数据包块由KSS 个源符号组成，为了减少字节填充数目，源符号进一步划分为N个等长的符号项（长度为T'）。因此源符号的长度（T）与符号项的关系为： T= T' x N；一个源

符号块中包含N*Kss个符号项。处理源数据包组的第一个SMTP数据包时，将其长度以网络字节序存放在第一个源符号的第一个符号项的前2字节中，然后存放SMTP数据包的数据（可能占据多个源符号的多个符号项），若所占的最后一个符号项没有填满，则以0填充该符号项的剩余空间；其他数据包以此类推。最后如果所占空间不足源符号长度的整数倍（也即占用的符号项的个数si， si %N != 0），则以0填充剩余空间。

图29中介绍了一个形成源符号块的例子。该例中，长度分别为34，30，56，40，48的SMTP包生成一个源符号块，其由8个源符号组成。每个源符号长度为32字节，其中含有2个符号项（T=32，N=2），每个符号项长度为16字节。

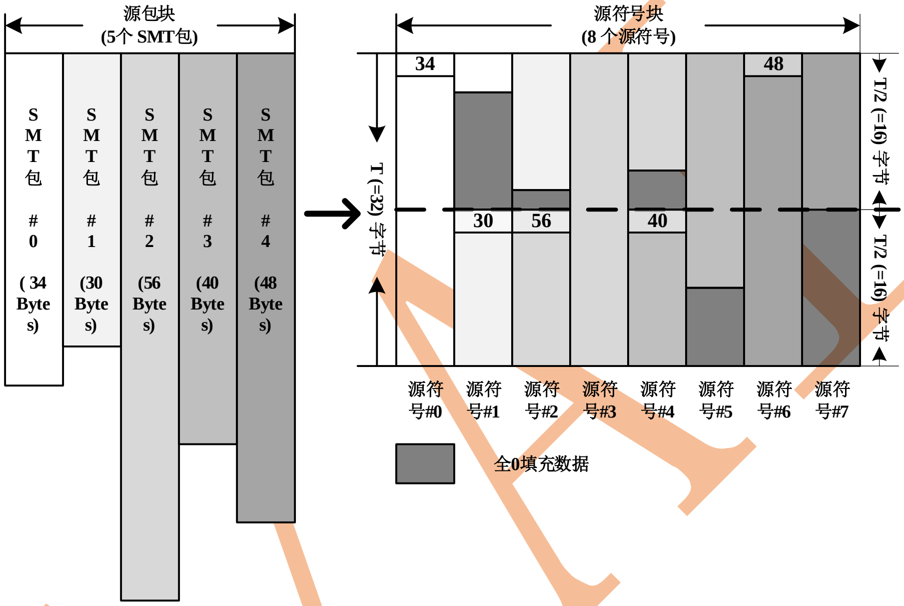


图29　带有填充的源符号块示例（ssbg_mode2）

11.4　FEC 的源数据包和恢复数据包格式

11.4.1　概述

下面将分别介绍传输SMTP包时的FEC的源数据包格式和传输恢复符号时的恢复数据包包格式，二者均为符合7.3.2.2 SMTP包结构的SMTP包。

在原始SMT数据包上添加source FEC payload ID，就形成了FEC源数据包，且需将数据包的FEC字段设为1；FEC恢复包包含一个repair FEC payload ID和恢复符号块的一个或者多个恢复符号，需将数据包的FEC字段设为2。源数据包和恢复数据包实际上都是SMTP包。FEC源数据包的source FEC payload ID标识FEC源数据包携带的源符号和符号项，FEC恢复数据包的repair FEC payload ID标识FEC恢复数据包携带的恢复符号和相应的源数据包组。

11.4.2　FEC 源数据包格式与负载 ID

FEC源数据包是为了传输受到FEC保护的SMTP包而生成的包，它由SMTP包头，SMTP负载头，负载数据和source FEC payload ID组成。

FEC源数据包的格式见图30，其与图8和图9定义的SMTP包结构一致。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SMTP packet header                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SMTP payload header                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SMTP payload data                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      source FEC payload ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图30　FEC 源数据包格式

图31规定了source FEC payload ID的格式。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            SS_ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          FFSRP_TS                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图31　source FEC payload ID 格式

SS_ID（32bits）：该序列号在FEC源数据包内识别源符号。初始值从任意值开始增加，达到最大值后归0，重新开始递增。如果ssbg_mode == 0或者ssbg_mode == 1，SS_ID每个FEC源数据包加1。如果ssbg_mode == 2，SS_ID每个符号项加1（包括最后一个源符号后的填充的符号项），且应被设置成第一个符号项的SS_ID。

FFSRP_TS（32bits）：指示值，FEC源码块或者恢复码块中发送的第一个包的SMTP包头中时间戳。

11.4.3　FEC 恢复数据包格式与负载 ID

FEC恢复数据包是用来传输一个或者多个恢复符号的包，它可以恢复相关的源码块。它由SMTP包头，repair FEC payload ID和一个或者多个恢复码组成。

FEC恢复数据包的格式见图32。其中SMTP packet header与图8和图9中定义的SMTP包结构一致，为了方便快捷地获得repair FEC payload ID，将其放置在恢复符号的前面。对于ssbg_mode0, ssbg_mode1两种模式，FEC恢复数据包应带有一个恢复符号，对于ssbg_mode2，携带一个或者多个恢复符号。此外，一个FEC源码块或者恢复码块中的所有恢复数据包中（除去最后一个），应含有相同数量的连续恢复符号。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SMTP packet header                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      repair FEC payload ID                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        repair symbol(s)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图32　FEC 恢复数据包格式

repair FEC payload ID的定义见图33。

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            SS_Start                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    RSB_length                 |     RS_ID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              RS_ID            |           SSB_length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   SSB_length  |                    ……                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     RSB_ID    |                   RSB_Start                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   RSB_Start   |                     RSBL                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           FFSRP_TS                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

图33　repair FEC payload ID 格式

SS_Start（32bits）：指出源符号块的边界。若ssbg_mode == 0或ssbg_mode == 1，它被置为相关的源符号块中第一个源符号的SS_ID。若ssbg_mode== 2，它被置为相关源符号块中第一个符号项的SS_ID。

RSB_length（24bits）：相关恢复符号块中恢复符号的数量。

RS_ID（24bits）：一个用来标示FEC恢复数据包中第一个恢复符号的整数。从0开始，对相应的恢复数据包里每一个恢复符号递增1。

SSB_length[N]（N*24位），　如果fec_coding_structure=0001，N与划分优先级的个数相关。如果ssbg_mode == 0或ssbg_mode == 1，表示源符号块中划分每个优先级下的子源符号块中源符号的数量。如果ssbg_mode == 2，表示源符号块中符号项的数量。

RSB_ID（8位），指示每个优先级下恢复符号块的序列号。

RSB_start（24位），指示第i个优先级下恢复符号块的第一个恢复符号的RS_ID。

RSBL（24位），指示第i个优先级下恢复符号的数目。

FFSRP_TS（32bits）：指示值，FEC源码块或者恢复码块中发送的第一个包的SMTP包头中时间戳。

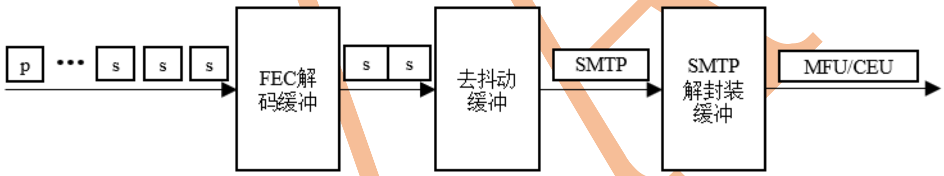
附　录　A
（规范性）
假设接收机缓存模型

A.1　概述

假设接收缓存模型是用于在固定的端到端时延和有限内存缓冲需求的情况下，确保智能媒体高效传输的一种方法。假设接收缓冲模型应被SMTP发送端所使用，以模仿接收端的行为。以下章节将详细描述。

SMTP发送端应使用假设接收缓存模型，以确保对数据包流进行的任何处理都在接收端限制之内。SMTP发送端应决定接收端所需要的端到端时延和缓冲大小，并且通过FEC信令消息通知接收端。

在接收端，从接收到SMTP数据包到重组成CEU包含了多次缓存操作。另外，对于一些传输操作，例如前向纠错（FEC）、丢包重传等都会引起时延和抖动，这些问题都需要接收端进行缓存操作来处理。假设接收缓存模型定义了一些接收端的缓存操作来确保任何时候缓存占有率都在缓存限制要求内。图A.1描述了假设接收缓存模型的组成。详细描述将在下面章节展开。每个SMTP数据包子流都需要应用假设缓存模型（如拥有相同packet_id的数据包形成的数据包子流）。缓冲器的实际设置是基于对特定SMTP子流的发送会话的配置来决定的。



图A.1　假设接收缓存模型

A.2　前向纠错（FEC）解码缓存

前向纠错（FEC）解码缓存是可选的，只有被FEC编码的SMTP数据包子流需要设置前向纠错解码缓存（例如属于同一媒体资源的数据包）。FEC解码应用广泛，很多情况下，例如由于信道问题或者网络拥塞导致的丢包和时延，底层传输机制并不能有效的解决，而利用FEC的方式则会起到显著的效果。在进行FEC解码时，需要设置一定的缓存，确保接收足够的源数据包和修复数据包进行FEC解码。

在假设接收缓存模型中，前向纠错解码缓存是可选的，如需应用，则按如下步骤进行：

—— 初始化前向纠错解码缓存为空；
—— 接收到发送时间戳为 ts 的数据包 i 存入缓存中，如果 buffer_occupancy + packet_size < max_buffer_size，则接收这个数据包，否则丢掉；
—— 如果 FEC 作用于数据包 i；
—— 确定数据包 i 所属的源数据块 j；
—— 确定源数据块 j 中第一个数据包的到达时间 t；
—— 在 t + FEC_buffer_time 时刻，把所有属于源数据块 j 的源数据包（前向纠错结束后）存入去抖动缓存，丢掉修复数据包。

其中，发送端利用FEC_buffer_time作为FEC解码要求的缓存时间，此时间表示的是从接受到源数据块的第一个源数据包到可以对源数据块进行FEC解码的时间。这个时间值可基于FEC块大小来计算。

A.3 去抖动缓存

去抖动缓存在假设一个最大传输时延值的前提下，确保数据包从发送到从SMTP协议栈输出这个传输时延固定。其中，所有传输时延大于这个最大传输时延的数据将被接收端丢弃。

去抖动缓存将按如下步骤进行：

—— 初始化去抖动缓存为空；

—— 接收 SMT 数据包；

—— 在 ts+Δ 时刻，数据包送入解封装缓存（其中 Δ 是通过信令消息得到的固定的端到端时延）。

经过去抖动缓存后，所有SMT数据包（无论是准确接收的还是通过FEC或者重传处理）都经过了相同的端到端时延。

A.4 数据包解封装缓存

SMT数据包解封装缓存用于在把SMTP负载传入上层之前对数据包进行处理。

SMTP数据包处理（低延时操作）的输出可能是MFU负载或者一个完整的CEU。解封装（SMTP数据包和负载的头部的移除）和任何数据包所需的重组都是SMTP数据包处理的一部分。这个步骤可能需要缓冲时延或者解封装时延，用于重构这个数据，以便被传递到应用层。然而，解封装延时并不会被认为是传输时延的一部分，并且可用的用于上层消耗的数据会被发送CEU的实体所担保，而不用管解封装的延时。

解封装缓存操作将按如下步骤进行：

—— 初始化 SMT 数据包解封装缓存为空；

—— 去抖动缓存被执行后，SMTP 数据包立即送入 SMTP 数据包解封装缓冲；

—— 对于载有聚合负载的 SMTP 数据包，移除数据包和负载头，然后提取每个单个的数据单元；

—— 对于载有被分割成片段的负载的 SMTP 数据包，这个数据包就一直被保存在缓冲器中，直到所有对应的片段都被正确接收，或者直到一个不属于同一个片段数据单元的数据包被接收了；

—— 取决于客户端的操作模式，如果一个完整的 CEU 或一个单个 MFU 重组完成，则将重组的数据送入上层处理后显示。

A.5 假设接收缓存模型的使用

基于假设接收缓存模型，SMT发送端可以确定传输步骤、缓存大小和缓存时延Δ，设置最大传输时延，从而避免缓存溢出导致的丢包。在一个固定的时延且不会引起SMT发送实体缓冲到下溢或溢出后，SMT发送实体应当保证经历的传输时延小于设定的阈值的数据包将会被传送到上层。

A.6 端到端时延和缓存要求的估计

在设置固定时延时，发送端实体应估计从发送端到接收端的最大期望及容许的传输时延。如果FEC正在使用中，因为要求FEC解码能够恢复丢失的SMTP数据包，SMT发送实体应添加FEC缓冲时延，这个缓冲时延涵盖了汇聚一个源块所需的时间。最后，应当增加任何可能会由数据包片段（以及其他操作）所引起的时延应。那么，SMTP数据包发送时延的结果估计应当作为固定的端到端发送时延，以发送信号的方式传达给SMT接收实体。

固定的端到端时延=最大发送时延+FEC缓冲时间。

为了估计所产生的缓冲需求SMT发送实体应当使用固定的端到端时延，并减去到SMT接收实体的最小发送时延，然后估计缓冲大小要求缓冲器大小要求，它等于SMTP数据包最大比特率与所计算的缓冲的数据持续时间的乘积。

缓冲器大小=（最大时延-最小时延）*最大比特率。

A.7　语法

假设接收机缓存模型的语法见表A.1

表A.1　假设接收机缓存模型语法

语法	值	比特位数	备注
HRBM ( ){			
message_id		16	
version		8	
length		16	
extension{			
extension_fields_Byte			
}			
message_payload{			
max_buffer_size		32	
fixed_end_to_end_delay		32	
max_transmission_delay		32	
}			
}			

A.8　语义

message_id：表明此消息是假设接收缓存模型消息。

version：假设接收缓存模型消息的版本号。接收实体可以用来检查此消息的版本号。

length：假设接收缓存模型消息的的长度。

max_buffer_size：提供最大缓冲大小信息。

fixed_end_to_end_delay：提供发送实体与接收实体间的固定端到端时延信息。

max_transmission_delay：提供发送实体与接收实体间的最大传输时延信息。

附　录　B
（规范性）
前向纠错编码码字

B.1　FEC编码算法

如表B.1所示，表示应用于SMT中的前向纠错编码算法。

表 B.1 FEC 编码码字算法

码字编号	FEC 编码算法
0～6	保留
7	自适应前向纠错编码
7～255	保留

B.2　自适应前向纠错编码码字

B.2.1　概述

本节规定了自适应前向纠错编码码字。自适应前向纠错编码码字在标准RaptorQ码字上进行了扩展，可以实现在单层中的自适应FEC保护。

根据DU头中定义的不同优先级，可以将源符号划分为不同的类别。所有分类的符号都可以由一个FEC编码矩阵保护，并在一个流中生成源符号和修复符号。

B.2.2　编码方法

步骤1：根据DU头中定义的不同优先级，每个源符号被划分到不同编码符号块中，每个编码符号块中源符号的数目为D_1，D_2，D_3，...，D_r。这样，D_1个源符号形成第一类，接下来的D_2个源符号形成第二类，以次类推。并且，类的重要性随类索引而降低。

步骤2：根据源符号的分类，可以设计相应的编码矩阵A来产生中间码字。图1为2类优先级下的编码矩阵A的结构。

<table>
<tr><td>G_p</td><td rowspan="2">0</td></tr>
<tr><td>G_ENC1</td></tr>
<tr><td rowspan="2">0</td><td>G_p</td></tr>
<tr><td>G_ENC2</td></tr>
</table>

图 B. 1 编码矩阵结构

其中G_p矩阵由’G_LDPC,1’、’I_S’、’G_LDPC,2’、’G_HDPC’和’I_H’组成，G_ENC1和G_ENC2是LT矩阵。根据该编码矩阵A可以编码得到中间码字。

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} {A_1}^{-1} & 0 \\ 0 & {A_2}^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ D_1 \\ 0 \\ D_2 \end{bmatrix}$$

对于优先级数量超过2个的情况，其编码矩阵A和生成中间码字如下所示：

$$A = \begin{bmatrix} \mathrm{G\_p}_1 & 0 & \dots & 0 \\ G\_ENC_1 & 0 & \dots & 0 \\ 0 & \mathrm{G\_p}_2 & \dots & 0 \\ 0 & G\_ENC_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathrm{G\_p}_l \\ 0 & 0 & \dots & G\_ENC_l \end{bmatrix}$$

$$\begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_l \end{bmatrix} = \begin{bmatrix} {A_1}^{-1} & 0 & \dots & 0 \\ 0 & {A_2}^{-1} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & {A_l}^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ D_1 \\ 0 \\ D_2 \\ \dots \\ 0 \\ D_l \end{bmatrix}$$

步骤 3：采用扩展窗的方式使得优先级高的源符号所对应的中间码字具有更高的恢复概率，从而提高了优先级高的源符号的恢复概率。其编码过程如下所示。 当信道条件较差时，可以减少 G_ENC_{12} 和 G_ENC_{21} 的行数，增加 G_ENC_{11} 的行数，以增加更多重要中间码字的冗余度并保证优先级高的源符号的恢复。

$$\begin{bmatrix} D_1 \\ D_2 \\ R_1 \\ R_2 \end{bmatrix} = \begin{bmatrix} G\_ENC_1 & 0 \\ 0 & G\_ENC_2 \\ G\_ENC_{11} & 0 \\ G\_ENC_{12} & G\_ENC_2 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

其中，R_1 是由中间码字 C_1 编码产生的恢复符号，R_2 是由中间码字 C_1 和 C_2 编码产生的恢复符号。若共有 l 类源符号，则恢复符号可以如下产生：

$$\begin{bmatrix} D_1 \\ D_2 \\ \cdots \\ D_l \\ R_1 \\ R_2 \\ \cdots \\ R_l \end{bmatrix} = \begin{bmatrix} G\_ENC_1 & 0 & \cdots & 0 \\ 0 & G\_ENC_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & G\_ENC_l \\ G\_ENC_{11} & 0 & \cdots & 0 \\ G\_ENC_{12} & G\_ENC_{21} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ G\_ENC_{1l} & G\_ENC_{2(l-1)} & \cdots & G\_ENC_{l1} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_l \end{bmatrix}$$

B.2.3 解码方法

解码过程类似于一组接收到的编码符号的编码过程。

步骤 1：由于某些数据包可能在传输过程中丢失，因此需要根据 ISI 确定数据丢失情况生成编码矩阵 A。编码矩阵 A 的行应基于 ISI 删除。用于解码的编码矩阵 A 如下。可以从源符号和编码矩阵 A 逆矩阵生成中间码字。

$$\mathrm{REV\_A} = \begin{bmatrix} G\_p_1 & 0 \\ REV\_G\_ENC_1 & 0 \\ 0 & G\_p_2 \\ 0 & REV\_G\_ENC_2 \\ REV\_G\_ENC_{11} & 0 \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} \end{bmatrix}$$

矩阵 A 可以进行如下变换：

$$\mathrm{REV\_A} = \begin{bmatrix} G\_p_1 & 0 \\ REV\_G\_ENC_1 & 0 \\ REV\_G\_ENC_{11} & 0 \\ 0 & G\_p_2 \\ 0 & REV\_G\_ENC_2 \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} \end{bmatrix}$$

$$\begin{bmatrix} G\_p_1 & 0 \\ REV\_G\_ENC_1 & 0 \\ REV\_G\_ENC_{11} & 0 \\ 0 & G\_p_2 \\ 0 & REV\_G\_ENC_2 \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ REV\_D_1 \\ REV\_R_1 \\ 0 \\ REV\_D_2 \\ REV\_R_2 \end{bmatrix}$$

由于中间符号 C_1 与恢复符号 R_1 和 R_2 有关，因此具有更高的恢复概率。接收到编码后的符号，可以根据不同情况恢复得到中间码字。

情况 1：接收到的编码符号总数小于源数据生成的编码符号的个数，并且 REV_A_1 的行数小于列数时；C_1、C_2 均无法解出；

情况 2：接收到的编码符号总数小于源数据生成的编码符号的个数，且 REV_A_1 的行数大于等于列数时，但 C_1 可以成功解码，但 C_2 不能成功解码。

情况 3：接收到的编码符号总数大于源数据生成的编码符号的个数，REV_A_1 的行数大于等于列数，REV_A_2 的行数大于等于列数，C_1 和 C_2 可以使用 IETF RFC 6330 定义的方法解出。

情况 4：接收到的编码符号总数大于等于源数据生成的编码符号的个数，REV_A_1 的行数大于等于

列数，REV_A_2的行数小于列数，C_1 可以成功解码，但 C_2 无法成功解码。

情况 5：接收到的编码符号总数大于等于源数据生成的编码符号的个数，且REV_A_1的行数小于列数，REV_A_2的行数大于等于列数时，使用高斯消元法直接对矩阵 REV_A 进行求解。

对于分成两类以上的源符号，也可以根据不同的情况进行解码。对于 2 个以上优先级，根据接收到的编码符号，可以生成以下矩阵 REV_A:

$$\begin{bmatrix} G\_p_1 & 0 & \dots & 0 \\ REV\_G\_ENC_1 & 0 & \dots & 0 \\ 0 & G\_p_2 & \dots & 0 \\ 0 & REV\_G\_ENC_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & G\_p_l \\ 0 & 0 & \dots & REV\_G\_ENC_l \\ REV\_G\_ENC_{11} & 0 & \dots & 0 \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ REV\_G\_ENC_{1l} & REV\_G\_ENC_{2(l-1)} & \dots & REV\_G\_ENC_{l1} \end{bmatrix}$$

REV_A 可以做如下变换：

$$REV\_A = \begin{bmatrix} G\_p_1 & 0 & \dots & 0 \\ REV\_G\_ENC_1 & 0 & \dots & 0 \\ REV\_G\_ENC_{11} & 0 & \dots & 0 \\ 0 & G\_p_2 & \dots & 0 \\ 0 & REV\_G\_ENC_2 \dots & \dots & \dots \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & G\_p_l \\ 0 & 0 & \dots & REV\_G\_ENC_l \\ REV\_G\_ENC_{1l} & REV\_G\_ENC_{2(l-1)} & \dots & REV\_G\_ENC_{l1} \end{bmatrix}$$

由 REV_A 矩阵，可以根据其满秩的不同情况，生成中间码字。

$$\begin{bmatrix} G\_p_1 & 0 & \dots & 0 \\ REV\_G\_ENC_1 & 0 & \dots & 0 \\ REV\_G\_ENC_{11} & 0 & \dots & 0 \\ 0 & G\_p_2 & \dots & 0 \\ 0 & REV\_G\_ENC_2 \dots & \dots & \dots \\ REV\_G\_ENC_{12} & REV\_G\_ENC_{21} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & G\_p_l \\ 0 & 0 & \dots & REV\_G\_ENC_l \\ REV\_G\_ENC_{1l} & REV\_G\_ENC_{2(l-1)} & \dots & REV\_G\_ENC_{l1} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \dots \\ C_l \end{bmatrix} = \begin{bmatrix} 0 \\ REV\_D_1 \\ REV\_R_1 \\ 0 \\ REV\_D_2 \\ REV\_R_2 \\ \dots \\ 0 \\ REV\_D_l \\ REV\_R_l \end{bmatrix}$$

计算每种优先级下矩阵的秩，例如，例如，$\begin{bmatrix} G_p_1 \\ REV_G_ENC_1 \\ REV_G_ENC_{11} \end{bmatrix}$矩阵的秩可以定义为 cumRank(1)，

$\begin{bmatrix} G_p_1 & 0 \\ REV_G_ENC_1 & 0 \\ REV_G_ENC_{11} & 0 \\ 0 & G_p_2 \\ 0 & REV_G_ENC_2 \\ REV_G_ENC_{12} & REV_G_ENC_{21} \end{bmatrix}$矩阵的秩可以定义为 cumRank(2)，以此类推。如果 cumRank(i)大于从优先级 1 到优先级 i 的源符号之和，则将 cumFullRank(i)赋值等于 1。根据 cumFullRank(i)，解码过程可以分为不同的情况。

情况 1：如果 cumFullRank(l)=1, cumFullRank(1:l-1)=1，则可以使用 rfc6330 方法成功解码中间码C_1到中间码C_l。

情况 2：如果 cumFullRank(l)=1，但从 1 到 l-1 的 cumRank(i)并不总是等于 1，则可以使用高斯消元法成功解码中间码C_1到中间码C_l。

情况 3：如果 cumFullRank(l)≠1，则对 cumFullRank(l-1)进行校验，直到 cumFullRank(i)=1，并根据情况 1 和情况 2 对中间码进行解码。因此，当C_l到C_{i+1}不能解码，但与此同时C_i到C_1可以通过 rfc 6330 法或者高斯消元法进行成功解码。

步骤 2：得到中间码字后，源符号可以根据中间码字译码得到。

$$\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} = \begin{bmatrix} G\_ENC_1 & 0 \\ 0 & G\_ENC_2 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

对于优先级多于 2 个的情况，同样可以根据中间码字的恢复情况得到源符号。

$$\begin{bmatrix} D_1 \\ D_2 \\ \cdots \\ D_{i-1} \\ D_i \end{bmatrix} = \begin{bmatrix} G\_ENC_1 & 0 & \cdots & 0 & 0 \\ 0 & G\_ENC_2 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & G\_ENC_{i-1} & 0 \\ 0 & 0 & \cdots & 0 & G\_ENC_i \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \cdots \\ C_{i-1} \\ C_i \end{bmatrix}$$

步骤 3：得到源符号后，数据包可以根据 packet_sequence_number 恢复原始顺序。
