ETSI TR 126 939 V15.1.0 (2018-10)

Universal Mobile Telecommunications System (UMTS);
LTE;
5G;
Guidelines on the Framework for Live Uplink Streaming (FLUS)
(3GPP TR 26.939 version 15.1.0 Release 15)

Reference
RTR/TSGS-0426939vf10

Keywords
5G,LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

The present document describes the ways to use the Framework for Live Uplink Streaming to setup services that allow the end user to stream live feeds into the network or to a second party.

1 Scope

The present document describes how to use the Framework for Live Uplink Streaming (FLUS) to stream live feeds to the network or to a second party. It describes the usage of both variants: the IMS-based and the non-IMS-based framework to carry regular 2D and 360 degrees video feeds. It also describes a set of instantiations for the non-IMS-based solution as the FLUS User Plane has been left for the discretion of implementations to support a diversity of requirements that require different instantiations of the user plane.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 26.238: "Uplink streaming".

[3] ISO 14496-12: "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format".

[4] ISO 23000-19: "Information technology – Coding of audio-visual objects – Part 19: Common media application format (CMAF) for segmented media".

[5] 3GPP TS 23.401: "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access".

[6] 3GPP TS 23.501: "System Architecture for the 5G System".

[7] ISO 14496-12: "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format".

[8] ISO 23008-1:"Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)".

[9] ISO 23008-1: 2nd Edition AMD2, "Enhancements for Mobile Environments".

[10] IETF RFC 6455: "The WebSocket Protocol".

[11] IETF RFC 5234 (2008): "Augmented BNF for Syntax Specifications: ABNF", D. Crocker, P. Overell.

[12] IETF RFC 6817: " Low Extra Delay Background Transport (LEDBAT)".

[13] ISO 23000-19: "Common Media Application Format for Segmented Media (CMAF)".

[14] IETF RFC 7230: "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing".

[15] IETF RFC 7540: "Hypertext Transfer Protocol Version 2 (HTTP/2)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

FLUS session: A logical association between a source and a sink within which media content can be sent from the source to the sink.

Media session: A subset or part of a FLUS session including the duration to establish the media session, the time period during which media content can be sent from FLUS source to FLUS sink and the duration to terminate the media session.

Media stream: The content sent from a FLUS source to a FLUS sink within a media session.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

FLUS	Framework for Live Uplink Streaming
MCC	Mobile Country Code
MNC	Mobile Network Code

4 FLUS Overview

FLUS defines a FLUS source entity and a FLUS sink entity that can support point-to-point transmission of speech/audio, video, and text. It defines media handling (e.g., signalling, transport, packet-loss handling, and adaptation). The goal is to ensure a reliable and interoperable service with a predictable media quality while allowing for flexibility in the service offerings.

A FLUS source entity, which may be embedded in a single UE, or distributed among a UE and separate audio-visual capture devices, may support all or a subset of the features specified in the present document.

When used as a generic framework, only the F-C procedures for establishing the FLUS session are required to be supported by the source and sink entities, and no other feature or procedure specified in the present document is mandated. Impact on the service quality and network capacity is left to the discretion of the implementation and the service utilizing the framework. For example, configuration of media formats and codecs follows the requirements of the respective service.

When offered as part of a 3GPP IMS/MTSI service, the source and sink are required to support the IMS control plane and media plane procedures, and the service quality is determined by the MTSI service policy.

The present document provides guidelines for the usage of FLUS and describes different user plane instantiations that can be used with FLUS.

5 Guidelines for IMS-based FLUS

Guidelines for the usage of FLUS in the IMS-based operation mode are not provided in the present document.

6 Guidelines for non-IMS-based FLUS

6.1 Use Case: Sharing to a Social Network Service

6.1.1 Use Case Description

In this example scenario, a user is sharing a 360 degree video that is being captured through a VR camera and sent as a fish eye, side-by-side 2D video. The 360 video stream is shared with a FLUS Sink in the network that relays the stream to a popular social network service (SNS).

6.1.2 Potential Realization in FLUS

Note: There may be two or more realizations: (1) The FLUS Sink is a transport layer proxy and forwards encrypted traffic to the SNS. (2) The FLUS Sink offers post-processing and forwards the post processed traffic to the SNS or to distribution.

Once the user selects to start sharing, the UE discovers an appropriate FLUS Sink that supports the specific SNS and that can stitch the fish eye into a 360 video and transcode the content to match the distribution format. The UE decides to use RTMP for this session, so it also verifies that the FLUS Sink is capable of receiving RTMP streams.

For the discovery, the UE checks the FLUS OMA DM Management Object first but it fails to find a FLUS Sink that supports the required capabilities. It then uses the pre-configured FLUS Sink discovery link to send an HTTP POST request:

http://flus.mnc<MNC>.mcc<MCC>.pub.3gppnetwork.org/flus/v1.0/sinks/

In the body of the POST request, the UE includes a JSON or XML document that describes the required capabilities.

The network replies with a short list of FLUS Sinks that support the desired capabilities. The UE then randomly picks one of the FLUS Sinks and queries its capabilities using the Sink's URL and the path "/flus/v1.0/capabilities". The response is a JSON or XML document that describes all capabilities of the Sink.

The UE proceeds then to FLUS session creation, which returns a session identifier. The session creation request may contain some configuration information or the UE may do that in a separate request. As part of the configuration, the FLUS Source may include a workflow description that requests the Sink to perform VR Stitching, transcoding, and distribution to the SNS.

Upon successful session creation and configuration, the UE connects to the provided link and starts sending the RTMP stream to the Sink. The Sink will perform the requested processing and distribution on behalf of the Source. It may also request the network to allocate appropriate QoS for the lifetime of the session.

When the user presses the stop button, the FLUS Sink will send a termination request to end the session.

6.2 Use Case: Live uplink video stream from drones or moving vehicles

6.2.1 Use Case Description

The media producer for an event is using drone-mounted-360 cameras or other moving vehicles like F1 cars, sailing boats or bicycles to capture scenes from more innovative angles. The drone is flown using line of sight, i.e. the drone pilot has direct visual contact to the drone. Other vehicles may have the driver / pilot on-board.

The live video is streaming to the live ingest server and then used together with other camera feeds in a live TV broadcast.

In particular for battery powered cameras, it may be beneficial to avoid processing like 360 video stitching on the device. Instead, it may be beneficial to leverage network based post processing functions, e.g. multiple video streams are transmitted and the stitching function is executed in the network.

Use-Case example: An event-organizer plans to use multiple drone mounted-camera to capture live video from an event. All live video streams should be routed to an editing facility, where a program direct decides on the sequencing of live video into a single linear program. The media source of each drone is configured with their own target quality

(bitrate) and target delay. Each media source is configured with a unique media sink so that the program director can identify each media source.

Note, this use-case can be seen generic so that the camera is not limited to be "drone mounted" but can be mounted to any devices, including stationary objects.

6.3 Use Case: Breaking-News reporter

6.3.1 Use Case Description

A News Corporation uses 5G and mobile equipment to speed up and simplify their breaking-news operations. Either, professional cameras are equipped with 5G uplink streaming modems, or regular smartphones (with external microphones) are used for video capturing. The universally available 3GPP coverage is used to stream the live video (with configurable, low delay) from the breaking news scene into the broadcast operation studio.

A news corporation negotiates a service level agreement with an operator so that a set of reporters can do sequential or simultaneous live reports. The general frame agreement between the news corporation and the MNO foresees, that each reporter can determine its own maximal video quality (measured in bit per sec). Each reporter should set its own quality, but some reporters are allowed to provider higher quality (i.e. use higher bitrates) than others.

6.4 Use Case: Immersive media conversations

6.4.1 Description

In this scenario, streams of 360 video and multi-channel audio are transmitted from a media sender to a media receiver, as illustrated in Figure 1, which at the receiver side, are projected on a screen or a HMD, and played out with loudspeakers or a headphone. In the other direction, video bit-streams of lower quality or resolution are transmitted to show the sender how the far-end user is watching and hearing the video and audio. A session is used to provide two-way real-time voice conversation. The 360 video and multi-channel audio are synchronized but arrives slightly later than the speech frames captured at similar times.

7 FLUS User Plane Instantiations

7.1 Non-IMS-based User Plane Instantiations

7.1.1 Introduction

This clause describes a set of instantiations for the generic FLUS User Plane that is not based on IMS.

7.1.2 fMP4-based Instantiations

7.1.2.1 Introduction

All instantiations of this clause are based on the fragmented ISOBMFF [3] format which is profiled by CMAF [4]. The following description summarizes the used media format used in the present document:

1) Each media component is formatted as a CMAF Track.

2) Each CMAF Track starts with a CMAF Header followed by one or more CMAF Fragments. A CMAF Fragment contains one or more CMAF Chunks. Note that CMAF requires that only the first CMAF chunk of a CMAF fragment is constrained to be an adaptive switching point. All subsequent CMAF Chunks do not need to contain any service access point.

3) When CMAF Fragments contain more than one CMAF chunk, it is beneficial that the first CMAF Chunk of the CMAF Fragment is preceded by a SegmentTypeBox that includes the compatible_brands 'cmfl', 'cmff'.

Note that the present document only considers the CMAF file format specific features.

7.1.3 fMP4 over MMTP Instantiation

7.1.3.1 General

MMTP is a transport protocol that supports the streaming of fragment ISOBMFF-formatted content using a dedicated payload format, the MPU payload format. Media data is streamed as a CMAF Header, followed by CMAF chunks for each media component separately. The CMAF Header is conformant to the MPU Header format and the CMAF Chunk is conformant to the MPU Fragment as specified in [7].

The FLUS Source and FLUS Sink use the MMTP protocol [7] over UDP, over DTLS/UDP or over WebSocket [10].

This instantiation is identified in the SDP by the protocol identifier: "MMTP/UDP", "MMTP/DTLS/UDP", or using a WebSocket URL ("ws" or "wss") respectively, as specified in [9].

This instantiation is also identified in the F-C configuration by the following urn: "org:3gpp:flus:2018:instantiations:mmtp" or "org:3gpp:flus:2018:instantiations:mmtp-ws", depending on whether using mmtp over UDP or over WebSocket.

Exactly one MMTP flow is used and each media component is sent using the MPU-mode, where each MPU conforms to the restrictions in clause 7.1.2.1.

The FLUS Source is required to maintain NTP synchronization, with a tolerance of ±20 ms, when setting the MPU_timestamp_descriptor and the MMTP delivery timestamp.

An fMP4 over MMTP/UDP or MMTP/DTLS/UDP session is described through an SDP file according to the constraints in [9]. The SDP is sent from the FLUS Sink to the FLUS source and includes exactly one media line that describes an MMTP flow in ***receive-only*** mode and with the target UDP port on which the MMTP flow is to be received at the FLUS Sink.

If the FLUS Sink is behind a NAT or Firewall, it has to ensure that the port is open (with correct NAT translation in place). It may also use known NAT traversal techniques, such as hole punching to establish the port binding at the NAT.

The source uses the MP table to describe the different components of the MMTP flow. The packet_id identifier is used to reference and map each component to the FLUS system.

7.1.3.2 MMTP Signaling

The content carried by an MMTP flow is described using the MP table, which is carried in an MPT message. The MP table with the relevant fields is provided by this table:

Table 7.1.3.2-1

Syntax	No. of bits
MP_table() {	
0xF	**8**
0x0	**8**
length	**16**
10111111b	**6**
number_of_assets	**8**
for (i=0; i<number_of_assets; i++) {	
Identifier {	
0x01	8
URL()	
}	
asset_type (4CC code)	**32**
0x0	**6**

Syntax	No. of bits
1b	**1**
asset_location {	
0x01	**8**
Location {	
0x00	8
packet_id	16
}	
}	
asset_descriptors {	
MPU_timestamp_descriptor()	
}	
}	
}	

The MP table is a compact description of the MMTP flow. It is used to identify all components of the content and the MMTP sub-flows that carry them. It also carries asset descriptors, which are used to describe the different assets to which the actual components belong to.

7.1.3.3 Synchronization

FLUS supports different types of sources. In one scenario, the FLUS Source captures the content and streams it to the FLUS Sink. In another scenario, the content is coming from multiple devices but multiplexed at the FLUS Source and streamed to the FLUS Sink. Yet in another scenario, the content is coming from multiple FLUS Sources and is synchronized and multiplexed at the FLUS Sink.

In the former 2 scenarios, the FLUS Sink can assume that synchronization of the different components of the media stream(s) is taken care of by the FLUS Source. This can be achieved by setting the PTS appropriately in the fMP4 for each component and by using a common time baseline where all fMP4 tracks from all components have the same 0 origin.

In case the content is multiplexed at the source but actually created by other devices, a clock drift may result from different clocks at the generating devices and the clock at the FLUS Sink. The MMTP timestamp may be used by the FLUS Sink to estimate such drift. The MMTP packet timestamp is a the NTP short format and reflects the time of transmission of the MMTP packet and reflects the time when the MMTP stack queues the packet for transmission. It is generally used to measure estimate delay and delay jitter during the transmission of the MMTP packets.

If content is created by different FLUS Sources, synchronization needs to be performed at the FLUS Sink. The FLUS Sink uses the MMTP MPU_timestamp_descriptor() from each of the components, which aligns the start of a CMAF track to UTC time, which enables the FLUS Sink to align media from different FLUS Sources. The MPU_timestamp_descriptor provides a NTP 64-bit timestamp that describes the presentation time of the first media sample in presentation order in the CMAF Track.

The MPU_timestamp_descriptor is carried as part of the Asset descriptors in the MP table and it has the following format:

Table 7.1.3.2-1

Syntax	Value	No. of bits	Mnemonic
MPU_timestamp_descriptor () {			
descriptor_tag		**16**	**uimsbf**
descriptor_length	**N*12**	**8**	**uimsbf**
for (i=0; i<N; i++) {			
mpu_sequence_number		**32**	**uimsbf**
mpu_presentation_time		**64**	**uimsbf**
}			
}			

In addition to the mpu_presentation_time, a field that carries the mpu_sequence_number, which provides an identifier of the current CMAF Track. This is important as the different sources may be forced to initiate a new CMAF Track during the transmission, e.g. after recovering from a failure.

7.1.3.4 Session Initiation and Description

Media Session initiation is triggered through the F-C procedure, where the session creation results in defining and returning the Entrypoint URL to the FLUS Sink. If MMTP/WebSockets is used then the Entrypoint URL points to a WebSocket URL and the MMTP sub-protocol is expected to be used. If MMTP/UDP is used, the Entrypoint URL points to an SDP file that is created by the FLUS Sink and used by the FLUS Source to start streaming data to the FLUS Sink. The SDP is generated by the FLUS Sink and includes exactly one FLUS media session, with recvonly attribute that indicates that media data is flowing uplink only.

An example of such SDP may look as follows:

```
v=0
o=user 6431641313 1 IN IP4 10.10.52.13
s=FLUS Session
t=1411639200 1427277600
a=source-filter: incl IN IP4 * 10.10.52.13
m=application 12345 MMTP/UDP 100 101 102
a=of:100 flowid=0 Signaling/PA
a=of:101 flowid=7623 MPU
a=of:102 flowid=7624 MPU
a=recvonly
```

7.1.4 fMP4-based Instantiation with HTTP Delivery.

7.1.4.1 General Description

This clause describes a FLUS Media Plane instantiation using a continuous sequence of CMAF Chunks [13] with HTTP Delivery, e.g. HTTP 1.1 [14] Chunked Delivery or HTTP 2.0 [15] Delivery.

This instantiation is identified in the F-C configuration by the following urn: "org:3gpp:flus:2018:instantiations:fmp4". An additional F-C configuration option determines, whether HTTP 1.1 with Chunked Transfer Encoding, HTTP2.0 with TCP or HTTP 2.0 with other transport protocols such as QUIC should be used for the continuous upload of CMAF Chunks. Note, the HTTP version and the transport protocol may also be negotiated at connection setup. When using QUIC as transport protocol, the FLUS Source can fall back to TCP, when the QUIC session setup fails.

The CMAF Track used for this instantiation starts with the CMAF Header, followed by a sequence of CMAF Chunks.

The CMAF Header contains information around the number of tracks, the used codec, codec configuration and optionally static metadata for the upstreamed movie file.

When the FLUS session contains multiple media components, each component is formatted as a CMAF Track and upstreamed separately. A common presentation timeline is used across the different media components.

It is assumed that the FLUS Source provides only a single quality per media components to the FLUS Sink. It is assumed that this single quality is the highest quality and the FLUS Sink could create additional quality representations using a transcoder. However, it is in principle also possible that the FLUS Source provides multiple quality representations per media components.

The CMAF Chunks of the CMAF Track are continuously appended to a larger resource. The FLUS source is generally not adding 'styp' boxes, except if the immediately succeeding CMAF Chunk contains a service access point.

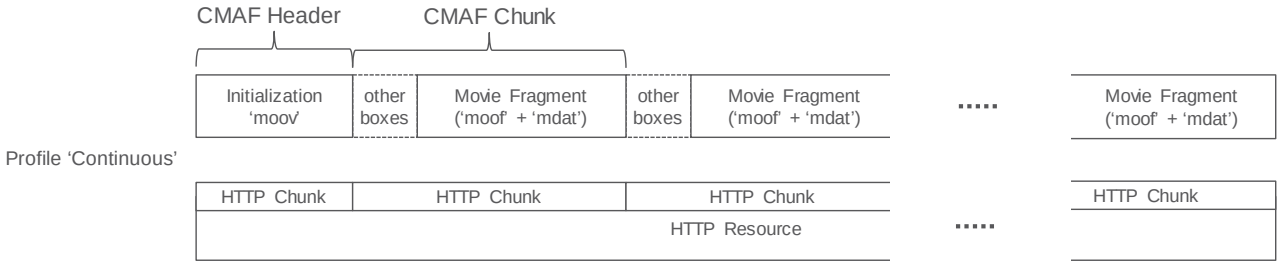

Figure 1: Illustration of Continuous Chunk Profiles (with mapping to HTTP 1.1 resources)

When a single FLUS Source streams multiple CMAF Tracks, a common media time line (i.e. decoding and composition timestamps) should be used across all CMAF Tracks. The FLUS source may insert wall clock timestamps using the Producer Reference Time Box ('pfrt'box) into the CMAF Chunk stream. This may be beneficial, when streams from multiple FLUS Sources should be jointly post processed. The FLUS Source should be properly time-synchronized with the network, e.g. using EPS time synchronization derived from SIB16 (See TS 36.331) or NTP or other appropriate mechanisms.

This FLUS media instantiation focuses on the usage of the HTTP 1.1 and HTTP 2 protocol for uplink. Usage of secure connections is possible using existing HTTP technologies.

The FLUS sink offers a simple HTTP PUT or POST interface for upload. The actual uplink stream is provided in the HTTP request body.

The FLUS sink exposes the *Push URL* element, which provides the base URL for the ingestion. All FLUS source appended sub-paths to the base URL belong to the same FLUS session.

Example, the FLUS sink offers the *Push URL* "http://sink.operator.com/sessionxyz/" via F-C. This allows the FLUS source to ingest sessions with multiple media components. Each media component is identified by a unique URL. The FLUS source appends additional path parts to complete the URL for the media. For example, the FLUS source sends audio to http://sink.operator.com/sessionxyz/audio-180130.mp4 and video to http://sink.operator.com/sessionxyz/video-180130.mp4.

When the FLUS source starts the media session, the FLUS source streams first the CMAF Header information for the movie file. After that, the FLUS source streams FLUS Chunks as the FLUS chunks become available.

In case of HTTP 1.1, the FLUS source uses HTTP chunked transfer encoding. Usage of HTTP chunked transfer encoding is indicated in the HTTP request header for the upload. The FLUS source finalizes the HTTP resource by sending a zero-size HTTP Chunk. HTTP 1.1 does not allow multiplexing of multiple simultaneous HTTP resources (aka media components), so, when multiple media components are streamed uplink, a separate TCP connection is needed for each media component.

In case of HTTP2, the FLUS source is simply omitting the Content-Length header. The FLUS source finalizes the HTTP resource by closing the HTTP2 stream using the END_STREAM flag in a frame. HTTP 2.0 allows multiplexing of multiple HTTP resources on the same transport connection (such as TCP).

When using TCP as transport, the usage of a persistent TCP connection for HTTP resource up streaming is recommended. The TCP buffer level is controlled by means of the TCP_NOTSENT_LOWAT socket option that is available in multiple operating systems.

An example of a recommended congestion control is LEDBAT [12], other congestion control schemes, which strive for a low network queue delay, are currently under development in IETF.

7.1.4.2 Rate Adaptation

The FLUS source could adapt the media bitrate to fit to the currently available link bitrate. The rate adaptation algorithms of the underlying transport protocol realization (such as TCP) are re-used. When the codec configuration parameters (e.g. picture parameter set) are not changed, the FLUS Source can change the encoding bitrate without interrupting the encoding process. A media streaming solution is preferably rate adaptive in order to cope with changing network conditions. A FLUS source, creating an fMP4 stream, can also change the bitrate as needed. In order to allow for rate adaptation, the FLUS sink offers a reception buffer (Cf. Figure 2), which delays the stream for a configurable duration. The reception buffer can be used to compensate link bitrate variations without the need of changing the media quality. The FLUS Source starts rate adaptation when the reception buffer on the FLUS Sink cannot compensate the link bitrate variations anymore. The reception buffer depth could be configurable via F-C.

The FLUS sink uses this reception queue (see figure below) to recover the encoder frame rate, i.e. to compensate network jitter. The FLUS source needs to know or needs to provision the FLUS sink delay in order to apply rate adaptation techniques for example to provide the best possible quality at minimal frame losses (i.e. due to late FLUS Sink arrival). Such a configuration is provided with the *Pipeline Description* element.

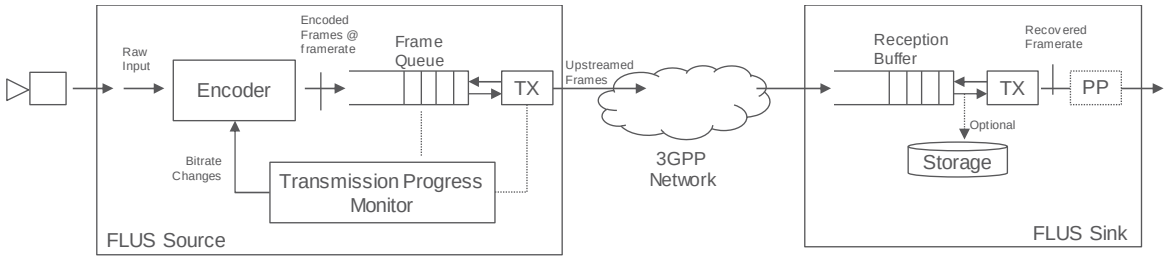

Figure 2: Rate Adaptation

A FLUS source can monitor the upstreaming progress or could listen to notifications / rate recommendations from the network. Either as an alternative and/or additional facility, the network could provide a dedicated assistance capability to boost the reception of upstream media, in case transient network throughput restrictions have caused a too high backlog at the FLUS Source. This model is particularly appropriate for approaches whereby the FLUS source expects the network to be able to upstream a particular pre-determined format or bitrate version of the media asset that is being sourced.

Existing transport protocols such as TCP employ a rate adaptation algorithm, which adjusts the TCP throughput to the available link bitrate. A rate adaptation logic can measure the bitrate at which the TCP sender is draining the frame queue. Further, the FLUS source can monitor, how quickly a frame is upstreaming (first byte of the frame until the last byte of the frame).

Any quality changes of the FLUS Source due to rate adaptation is noticeable to the audience. In some situations, it may be better to configure a larger Transmission Buffer (i.e. operate at a higher delay) to ensure a higher and sustainable quality. There is no need to standardize the detailed rate adaptation algorithm. However, the FLUS sink should support a reception queue and recovery of the encoder frame rate.

Other transport protocols such as QUIC may also be used to re-use rate control and retransmission schemes.

7.1.5 fMP4-based Instantiation using multiple segments per track

7.1.5.1 General Description

This section contains a similar description to clause 7.1.2 with the difference, that the CMAF track is subdivided into individual CMAF Segments. Depending on the FLUS Sink implementation, every CMAF Segment is identified by a unique URL or the FLUS sink derives the segment sequence from the segments itself. In the first case, all CMAF Segments belonging to the same CMAF Track are identified by the same base URL. In the latter case, that same URL is used for all CMAF Segment of the same CMAF Track.

This instantiation is identified in the F-C configuration by the following urn: "org:3gpp:flus:2018:instantiations:fmp4". An additional F-C configuration option determines, whether HTTP 1.1 (with Chunked Transfer Encoding), HTTP2.0 with TCP or HTTP 2.0 with other transport protocols such as QUIC should be used. Note, the HTTP version and the

transport protocol may also be negotiated at connection setup. When using QUIC as transport protocol, the FLUS Source can fall back to TCP, when the QUIC session setup fails.

A FLUS session may contain one or more media components. When the FLUS session contains multiple media components, each component is formatted as a CMAF Track and upstreamed separately. A common presentation timeline is used across the different media components.

The CMAF Header contains information around the number of tracks, the used codec, codec configuration and optionally static metadata for the upstreamed movie file.

Every CMAF Track starts with the CMAF Header, followed by a sequence of CMAF Segments. Every CMAF Segment may contain one or more CMAF Chunks.

Every CMAF Segment starts with an ‘styp' box.

An additional F-C configuration option allows the selection of the "Segmented” Profile. When selecting the "Segmented” Profile, some additional configuration parameters are needed. For every CMAF Track, an URL for the CMAF Header and an URL Template for CMAF segments is configured.

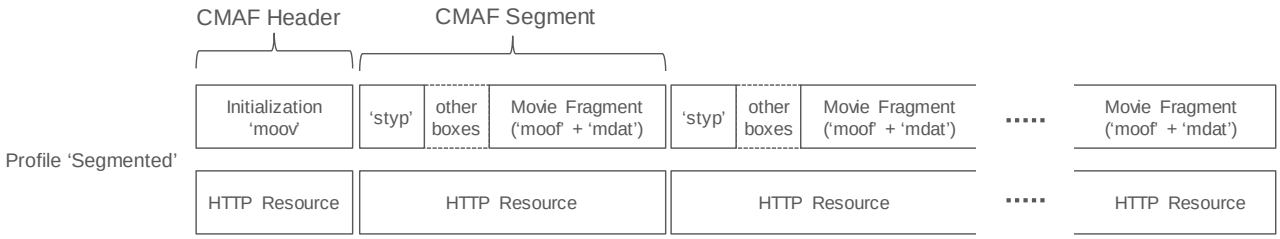

Figure 3: Illustration of Segmented Profiles with one CMAF Chunk per CMAF Segment (with mapping to HTTP resources)

The FLUS sink offers a simple HTTP PUT or POST interface for upload. The actual uplink stream is provided in the HTTP request body. The HTTP request header contains an URL, either corresponding to the CMAF header or is formatted according to the CMAF Segment URL template.

8 Example FLUS Workflows

8.1 Example Workflow using F-U MMTP

The UE uses the F-C RESTful procedures to create a FLUS session. It includes the identifier: "org:3gpp:flus:2018:instantiations:mmtp-ws", to indicate that the requested F-U instantiation is fMP4 over MMTP/WebSocket. If accepted by the FLUS Sink, it includes a websocket URL in the response as the entry point to the F-U. The UE uses the provided WebSocket URL to connect to the Sink and to start the F-U session.

In this example, the session consists of one audio stream (captured from a microphone) and 2 video streams (captured from two cameras). As such, 3 different MMTP sub-flows are created and assigned packet ids 1, 2, and 3. Each sub-stream carries a different component of the media data. At the start of the session, the MPT message is sent with an MPT table that describes the 3 different sub-flows as separate assets. The MMT_general_location_info indicates the associated packet_id to each asset using location_type=0x00.

To reduce the end-to-end latency, the Source creates CMAF Chunks that are of very short duration and sends them chunk by chunk. The CMAF chunk corresponds to an ISOBMFF fragment that is generated on the fly. The Source uses the same reference clock (NTP wall clock time) to synchronize the transmission of all sub-flows. The delivery timestamp is provided as part of every MMTP packet and reflects the wall clock time at the time of generation and transmission of the MMTP packet. Optimally, an MPU_timestamp_descriptor may also be generated and included as part of the MP table.

The following figure shows the structure of the F-U stream:

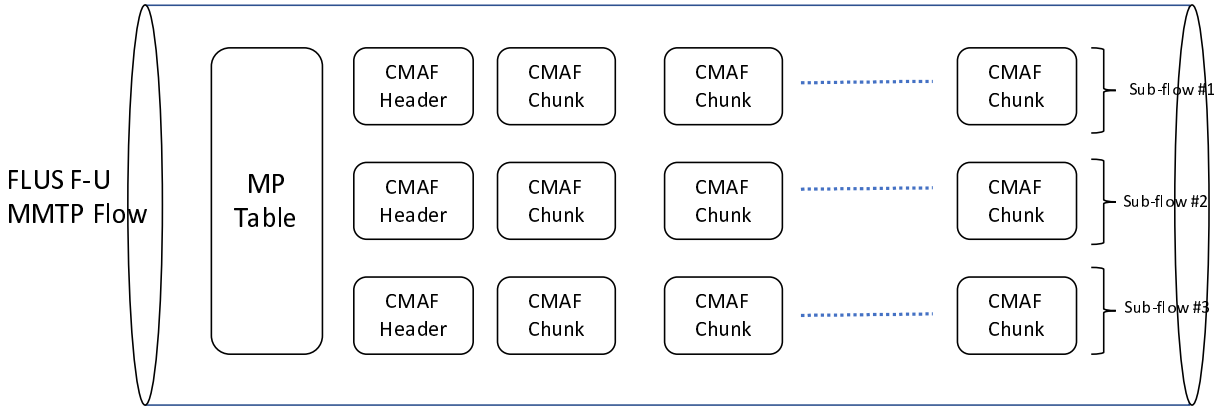

Figure 4: Structure of an MMTP flow as a FLUS F-U

The following diagram depicts the message flow in the F-C to setup a FLUS session that uses the MMTP instantiation:

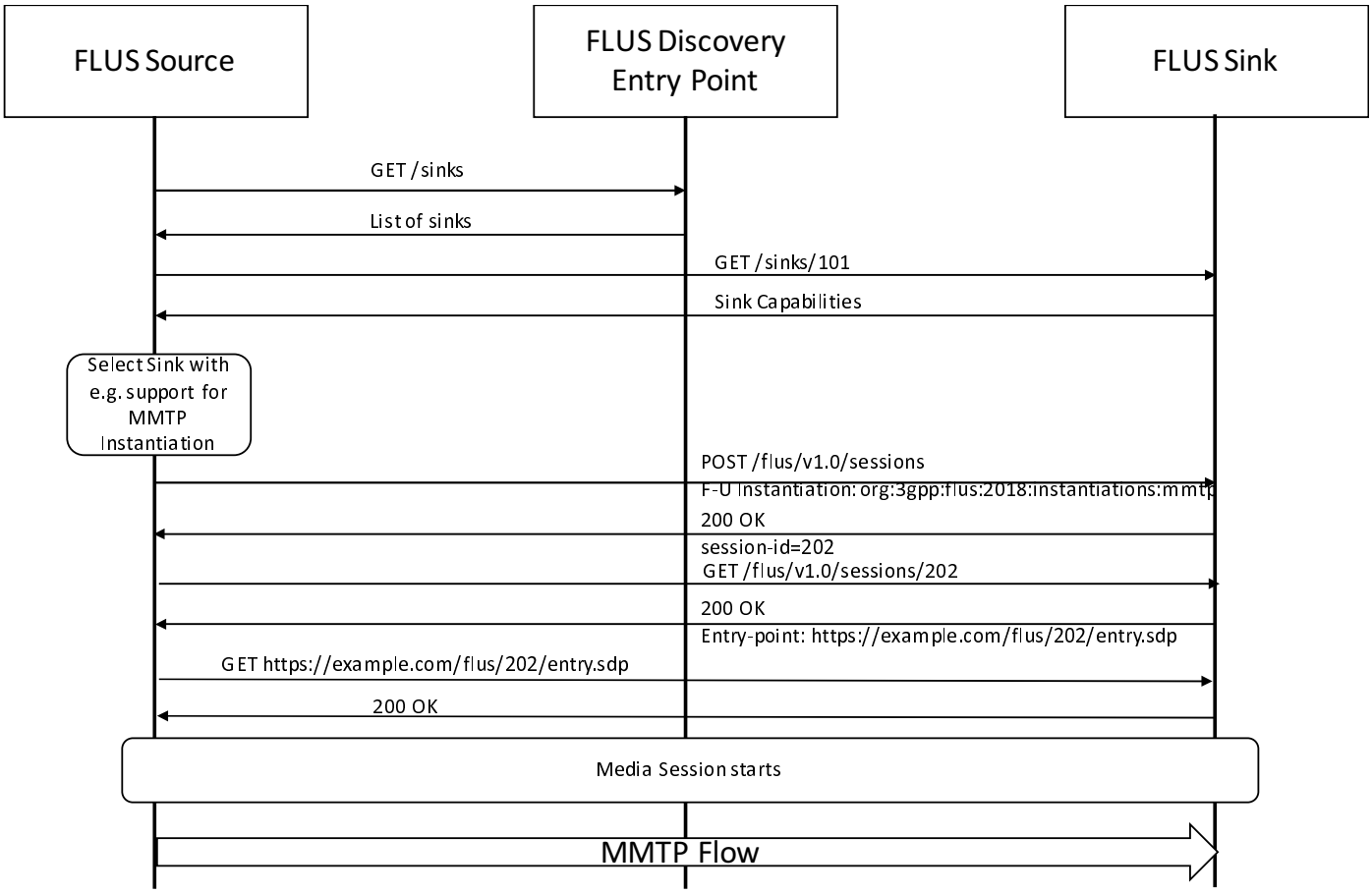

Figure 5: Message flow in the F-C to setup a FLUS session that uses the MMTP instantiation

After successful establishment of the FLUS session and fetching the SDP, the FLUS Sink will know the destination IP address and port number to which it will have to send the multiplexed MMTP Flow.

8.2 Example Call Flow for fragmented MP4 with HTTP Delivery

8.2.1 Assumptions

The FLUS session is created from the same device as the FLUS media plane is provided.

The FLUS Source selects a FLUS Sink, which supports the required Media Codecs and post processing capabilities.

The FLUS Sink provides a single HTTP Push URL so that the FLUS Source can establish one or more HTTP Sessions to the FLUS Sink. The Push URL here is http://sink.operator.com/sessionxyz/.

The FLUS Source uses the received Ingest URL and appends suffixes, so that any media component (CMAF Track) is identified by a unique URL. The FLUS Source here desires to upstream a video and an audio component.

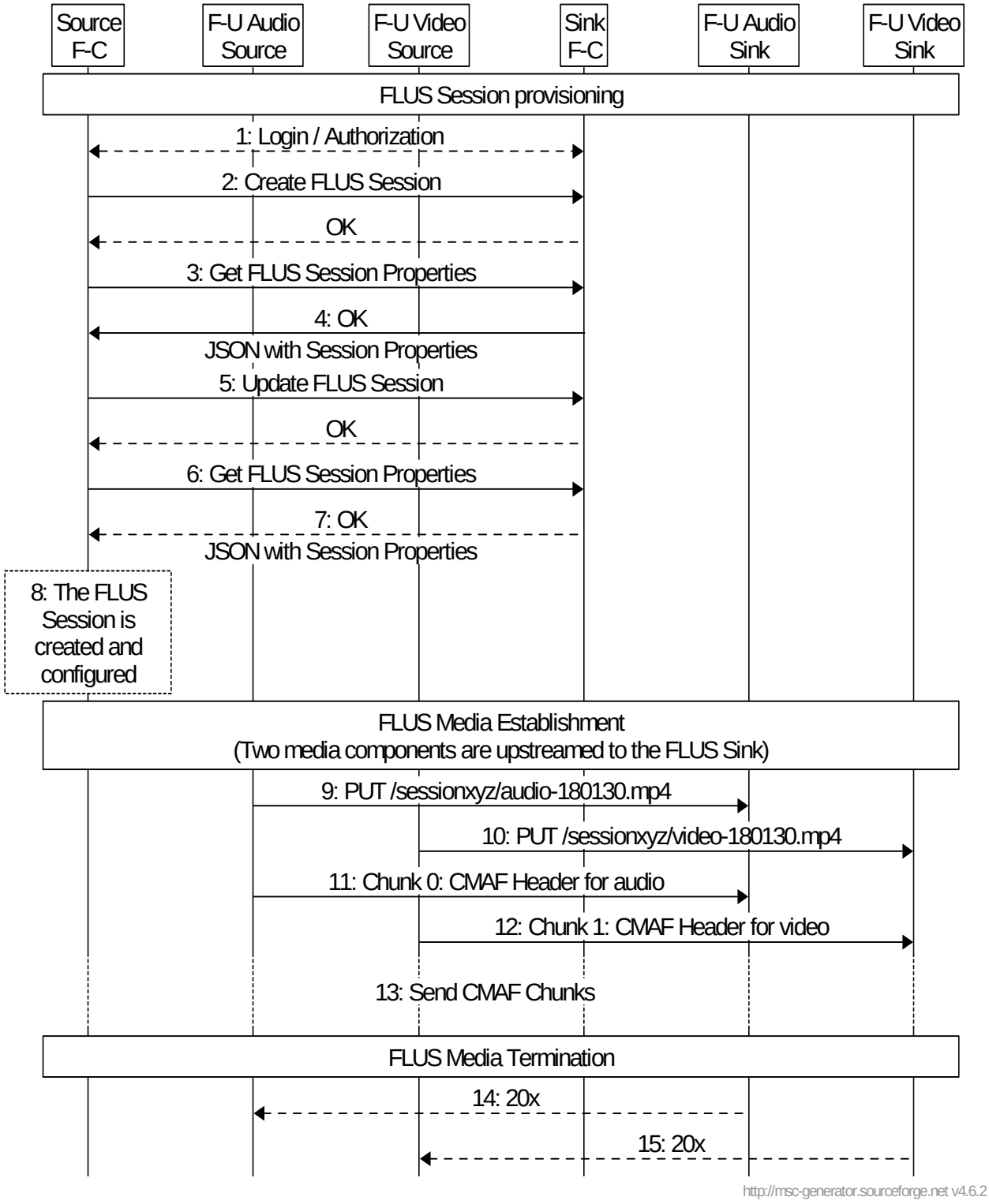

Figure 6: Example Call Flow for fragmented MP4 with HTTP Delivery

FLUS Session Provisioning using F-C

1) The User of the system (FLUS Source Control) logins into the FLUS system, e.g. using user-name and password.

2) The user creates a FLUS session using F-C. The FLUS Sink provides a unique session id to be used in subsequent transactions

3) For FLUS Session provisioning, the FLUS source first fetches the FLUS session parameters.

4) The FLUS Sink provides the FLUS session parameters with the response.

5) The FLUS Source does the needed FLUS Session modifications and applies the changes.

6) The FLUS source fetches the updated FLUS session parameters (The FLUS Sink may have done updates due to configurations).

7) The FLUS Sink provides the FLUS session parameters with the response. The FLUS Session parameters contains a Push base URL (here. http://sink.operator.com/sessionxyz/)

8) The FLUS Session is now fully provisioned and the FLUS Source has all needed information.

NOTE: The FLUS source function may be separated over different devices. A user can do the FLUS Session Configuration much earlier and potentially using a different device. When the FLUS Source re-connects to an already established, but not active session, the FLUS Source needs to authenticate towards the FLUS Sink (repetition of Step 1).

When time is due to establish the FLUS media session, the FLUS Source should establish here two HTTP Sessions for media components (each formatted as CMAF Track).

1) The FLUS Source establishes a transport connection (e.g. TCP) and sends an HTTP 1.1 Command. Here, the FLUS Source Sends an HTTP PUT command to establish the audio HTTP session.

2) The FLUS Source establishes a transport connection (e.g. TCP) and sends an HTTP 1.1 Command. Here, the FLUS Source Sends an HTTP PUT command to establish the video HTTP session.

3) The FLUS Source sends the CMAF Header file for the audio component. The FLUS sink finds detailed conduct configuration information.

4) The FLUS Source sends the CMAF Header file for the video component. The FLUS sink finds detailed conduct configuration information.

5) The FLUS Source starts appending CMAF Chunks to the established HTTP Sessions (according to the type),

When the FLUS Source is pausing the live uplink streaming session, the FLUS source may stop sending CMAF Chunk and keep the HTTP session open. When the FLUS session is continued, the FLUS Source may continue appending CMAF chunks to the session.

When time is due to terminate the FLUS media session, the FLUS Source sends a zero size HTTP Chunk.

6) Upon reception of the zero size HTTP Chunk, the FLUS Sink sends the HTTP response, indicating the creation of the audio track.

7) Upon reception of the zero size HTTP Chunk, the FLUS Sink sends the HTTP response, indicating the creation of the video track.

8.2.2 CMAF Format Example

An example from a wireshark capture is depicted below. The FLUS source uses here HTTP PUT together with HTTP chunked transfer encoding to an Apache2 server. The Apache2 server was configured with a webdav server module.

The first HTTP Chunk contains the 'ftyp'box and the initialization information. The first HTTP chunk is of size '27d'. The first FLUS chunk (containing here only 'moof'and 'mdat'boxes) is set afterwards as single HTTP chunk. The size of the second HTTP chunk is 2beb.

```
PUT /webdav/dbg-DirCam-20180119-092131.mp4 HTTP/1.1
Transfer-Encoding: chunked
Content-Type: video/mp4
```

```
User-Agent: FLUS_HTTP_User_Agent
Host: 192.168.1.141
Connection: Keep-Alive
Accept-Encoding: gzip
Scheme: http

27d
....ftypisom....isomavc1...emoov...lmvhd......[
..[
.....@.....trak... \tkhd......[
..[
.....@.....emdia... mdhd......[
..[
.....U.....%hdlr.....vide.....Tlos.....minf....vmhd.....$dinf....dref.....url
.....stbl....stsd.....|avc1.....H...H.....&avcC.B.
(....gB.(..@x....E8...h.C.....stts.....stsc.....stsz.....stco.....(mvex... trex.....
.....2beb...Pmoof....mfhd.....8traf....tfhd...8.....d..+....@....trun.....X..+.mdat..+.e...@
...&(....}.....O.. 3..;.}.....]..}.....}..}.....?.G.W.Q..... '..x....>

<cut>
..Dns.@#.v'.....8..L#.....{..G.....?."8@F.....4F...B.B.'....7.#.C..8.<p.....8...D.0G...a.LG.#.x..>.>
...X`C.^...?8....?.P....
..<. mN#.....O?V..%..:.,#....q.z...V..b....U....7.%hK.xpC..."".....x....|Gb;..7
..ui..@.0Gb..#t!..w....Y..;.z..@!a..]Z...8LF.
0

HTTP/1.1 201 Created
Date: Fri, 19 Jan 2018 08:21:42 GMT
Server: KnownServer/2.4.18
Location: http://192.168.1.141/webdav/dbg-DirCam-20180119-092131.mp4
Access-Control-Allow-Origin: *
Content-Length: 291
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>201 Created</title>
</head><body>
<h1>Created</h1>
<p>Resource /webdav/dbg-DirCam-20180119-092131.mp4 has been created.</p>
<hr />
<address>KnownServer/2.4.18 Server at 192.168.1.141 Port 80</address>
</body></html>
```

When the FLUS source terminates the HTTP Request body using a zero size HTTP chunk, the HTTP server provides the HTTP response.

9 Guidelines for QoS usage for FLUS

9.1 Use-Case introduction

For Live Uplink Streaming, e.g. for professional media production vertical, the 3GPP QoS system needs to strive to fulfill throughput requirements of the video flows beyond the guaranteed bit rate.

The Professional Media Production vertical (for example) requires fairly high media bitrates in order to achieve a decent video quality in downlink. In professional media production, uncompressed or lightly compressed video is carried often at speeds of several Gigabit per second (cf. SDI bitrates). This is of course often not feasible for mobile video production, in particular when mobility and wide-area coverage are important features (i.e. when deploying a dedicated LTE cells inside of a media production facility, it could make sense to send uncompressed or lightly compressed frames.

For mobile production, the speed of setting up a live feed (i.e. speed and simplification of production) and the freedom of high mobility is likely more important than high video quality at ultra low latency. Compressed video streams can be used at expense of latency (compression efficiency increases when relaxing latency constrains). Still, the video quality should be high.

The assumption, in the following discussion, is a bitrate adaptive FLUS solution, where the FLUS source can adjust the transmission bitrate to the currently measured / estimated link bitrates. This can be achieved by influencing the encoder bitrate or by dropping frames before transmission.

The figure 7 below illustrates the desired video quality properties (and the resulting bitrates) as an example.

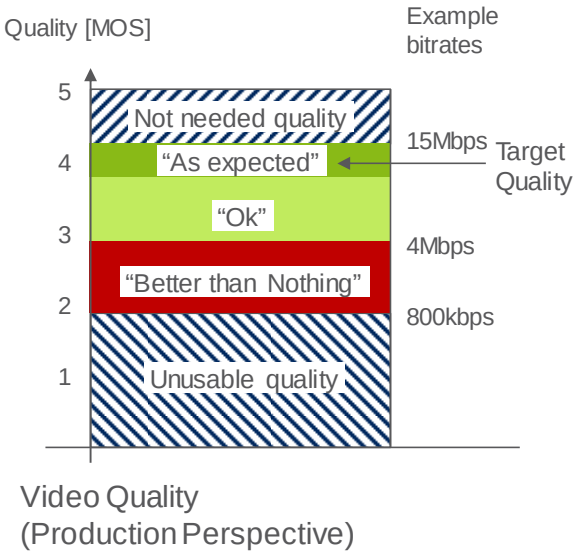

Figure 7: Quality Principles

The expectation is that the system delivers a certain target quality. Preferably, that target quality is always or as often as possible delivered and the target bitrate should be sustained by the system for a certain time duration. A higher quality as the target quality is not needed. Depending on the video codec configuration (Codec Profile, codec level and encoder features), the video quality is associated with a bitrate of the compressed stream.

When the system cannot offer the desired target bitrate, then a lower bitrate is acceptable for the video application. The video application layer (e.g. IMS / MTSI, HTTP or others) supports adaptive bitrate adaptation, i.e. it is increasing or decreasing the quality matching whatever link bitrate that is available. In the example above, a resulting video bitrate of ~15Mbps corresponds to the target video quality. The dark green color corresponds to an "as expected quality". A light green color corresponds to an "ok" quality. The resulting quality is not perfect, but still good to use.

A certain large bitrate range leads to an acceptable quality. The lower end of that bitrate range is the "better than nothing" area, where the video quality contains very obvious quality artifacts. In an example of a media production use-case, the director for the media production may still decide to use the video feed, since the captured pictures are still

"better than nothing". For example, when there was a crash or another event and there is no other video material available.

When the system cannot even offer the lowest quality (here 800kbps), the media producer will terminate the video stream, due to unusable quality. The Video source can stop sending the video stream, since the server is anyhow discarding the content.

The actual quality thresholds depend on the use-cases. The lowest unusable quality threshold is certainly lower for breaking news scenarios than for regular reports. Further, when the camera is mobile, e.g. mounted on a F1 car or a downhill racing skier, the acceptable quality is certainly different than for fixed mounted cameras.

3GPP systems offer different radio access systems. Some radio access systems are capable (depending on the deployment) to provide higher uplink data rates than others. For example, when a device is connected via the new NR radio access network, much higher data rates will be possible than using existing HSPA or GERAN radio access networks.

The figure 8 below depicts a mobility case, where a mobile uplink streaming client is either getting active in different radio access systems (nomadic mobility) or even moving between access systems with an active uplink streaming session. The different access networks have different bitrate characteristics (of course, deployment release and carrier bandwidth will have similar effects).

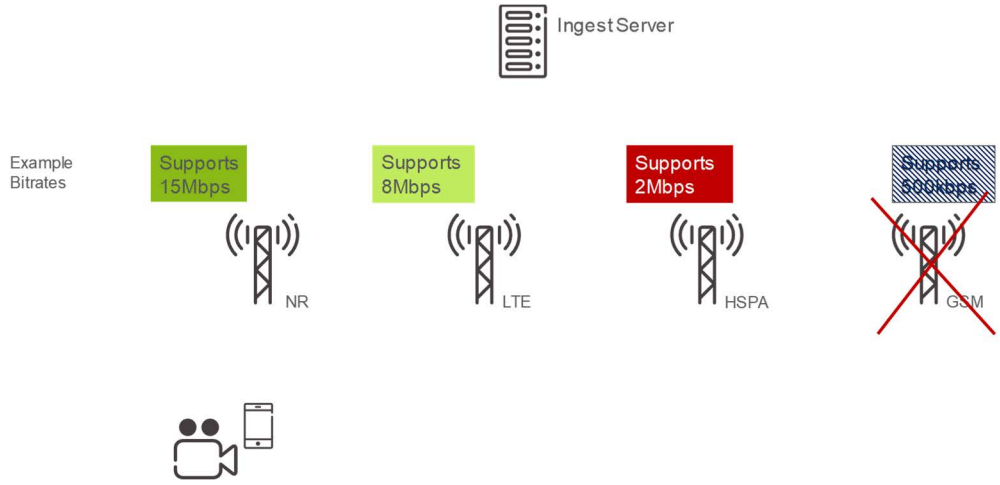

Figure 8: Mobility and example uplink bitrate expectation

As consequence, there may be handovers within one radio access network (e.g. within NR) or even between radio access networks (e.g. from NR to HSPA).

Due to inter RAT hand-over, the GBR should not be set to a too high bitrate. The UE may handover to a RAT, which does not support such high bitrate and the admission control may reject a QoS bearer. A GBR value should be found, which refers to the bare minimal acceptable bitrate so that each RAT keeps the QoS bearer and the application adapts the bitrate to the admitted parameters.

Beside the mobile media production use-case, there are several other use-cases. The devices may be stationary (e.g. stationary media production or mounted surveillance camera's) and some other may be mobile (e.g. patterns of "breaking news" reporters or vehicle mounted surveillance cameras).

9.2 Discussion of the 3GPP QoS Framework

9.2.1 Introduction

3GPP QoS framework specifies a Guaranteed (Flow) Bitrate (G(F)BR), a Maximum (Flow) Bitrate (M(F)BR), an Allocation and Retention Priority (ARP), and additional QoS Class Indicators (QCI / 5QI). Each QCI defines a priority level (PL), a maximal latency and a maximal packet loss rate for the QoS flow.

9.2.2 Architecture

In 3GPP systems, QoS bearers are requested via the PCF / PCRF. Typically network nodes interact with the PCF / PCRF for QoS.

An architecture for IMS / MTSI is depicted in Figure 9 below. The Session Border Gateway forwards the SIP INVITE (call setup message) via potentially other IMS nodes to the FLUS Sink. The SB GW extracts QoS information such as bitrate from the SIP INVITE message (paring the SDP file) and triggers the establishment of a QoS bearer / QoS flow via the Policy Control Function (PCF). The 5-Tuple(s) for the (uplink) UDP sessions are forwarded as well.

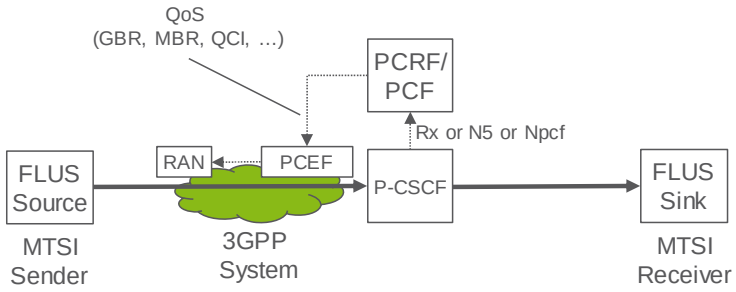

Figure 9: IMS / MTSI based architecture (considering EPS QoS terminology)

An HTTP(s) based architecture is depicted in Figure 10 below. Here, the FLUS Sink (aka HTTP Server) interacts with the Policy Control Function (PCF) to trigger the establishment of a QoS bearer / QoS flow. The FLUS sink needs to wait for the F-U establishment in order to know the 5-Tuple of the session. The FLUS Sink derives the QoS parameters from earlier provisioning steps or from the initialization information of the HTTP FLUS session (i.e. from the existing bitrate ('btrt') box in the codec configuration (e.g. the 'avcC'box for H.264 or a new box for dedicated signaling).

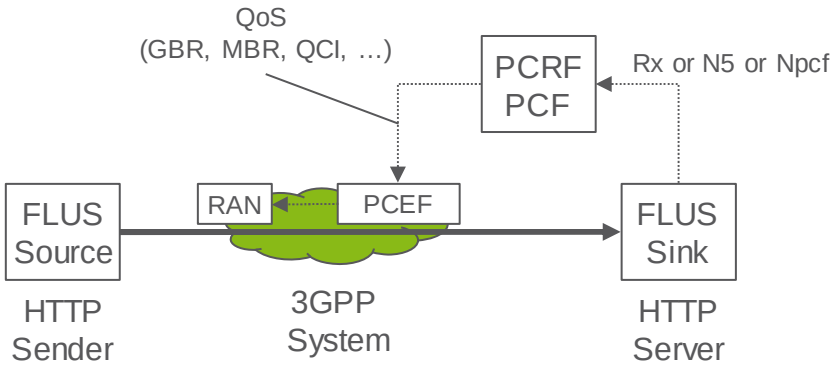

Figure 10: HTTP based Architecture (or other OTT protocols)

9.2.3 Relevant 3GPP sections

In the current Rel 15 QoS framework, the Allocation and Retention Priority defines the priority in Admission Control:

5.7.2.2 ARP

The QoS parameter ARP contains information about the priority level, the pre-emption capability and the pre-emption vulnerability. The priority level defines the relative importance of a resource request. This allows deciding whether a new QoS Flow may be accepted or needs to be rejected in case of resource limitations (typically used for admission control of GBR traffic). It may also be used to decide which existing QoS Flow to pre-empt during resource limitations.

The range of the ARP priority level is 1 to 15 with 1 as the highest level of priority. The pre-emption capability information defines whether a service data flow may get resources that were already assigned to another service data flow with a lower priority level. The pre-emption vulnerability information defines whether a service data flow may lose the resources assigned to it in order to admit a service data flow with higher priority level. The pre-emption capability and the pre-emption vulnerability shall be either set to 'yes' or 'no'.

There are two bit rate parameters available to a QoS Flow, GFBR and MFBR:

3GPP TS 23.501 V15.0.0 (2017-12)

5.7.2.5 Flow Bit Rates

For GBR QoS Flows, the 5G QoS profile additionally include the following QoS parameters:

- *Guaranteed Flow Bit Rate (GFBR) - UL and DL;*

- *Maximum Flow Bit Rate (MFBR) -- UL and DL.*

The GFBR denotes the bit rate that may be expected to be provided by a GBR QoS Flow. The MFBR limits the bit rate that may be expected to be provided by a GBR QoS Flow (e.g. excess traffic may get discarded by a rate shaping function).

The 3GPP QoS framework leaves the behavior of the scheduler above the GFBR bit rate value open to implementation:

5.7.3.3 Priority Level

The Priority level indicate a priority in scheduling resources among QoS Flows. The Priority levels shall be used to differentiate between QoS Flows of the same UE, and it shall also be used to differentiate between QoS Flows from different UEs. Once all QoS requirements are fulfilled for the GBR QoS Flows, spare resources can be used for any remaining traffic in an implementation specific manner. The lowest Priority level value corresponds to the highest Priority.

The priority level may be signalled with standardized 5QIs, and if it is received, it overwrites the default value specified in QoS characteristics Table 5.7.4.1. and similarly in 3GPP TS 23.401 V15.2.0 (2017-12):

4.7.3 Bearer level QoS parameters

[…]

Each GBR bearer is additionally associated with the following bearer level QoS parameters:

- *Guaranteed Bit Rate (GBR);*

- *Maximum Bit Rate (MBR).*

The GBR denotes the bit rate that can be expected to be provided by a GBR bearer. The MBR limits the bit rate that can be expected to be provided by a GBR bearer (e.g. excess traffic may get discarded by a rate shaping function). See clause 4.7.4 for further details on GBR and MBR.

4.7.4 Support for Application / Service Layer Rate Adaptation

[…]

The MBR of a particular GBR bearer may be set larger than the GBR.

Note, it would be possible to update the GBR value of a QoS bearer. However, the system does not trigger a renegotiation procedure before dropping a QoS bearer.

9.2.4 Usage of 3GPP QoS parameters

In the following text, the focus is on the GBR/GFBR, the MBR/MFBR and the priority level, since the aim is to get a high sustainable bitrate. Latency configuration of the QCI / 5CI may be a different issue.

The priority level which is associated to the QCI, is used to differentiate between traffic within a UE and across different UEs up to the GBR (GFBR in 5GC) value ("Once all QoS requirements are fulfilled for the GBR QoS Flows, spare resources can be used for any remaining traffic in an implementation specific manner." [5]) and it does not define a behavior for a scheduling priority to achieve a "target quality bitrate" larger than GFBR, but less than MFBR, rather only focus on a general resource distribution not related to the useful target bitrate. Moreover, in 4G, the PL parameter is only valid for flows below GBR, and the behavior of GBR bearers with bitrate above GBR is undefined. Therefore, in many 4G implementations, the GBR bearers will be treated as best effort, or worse, when the bitrate is larger than GBR.

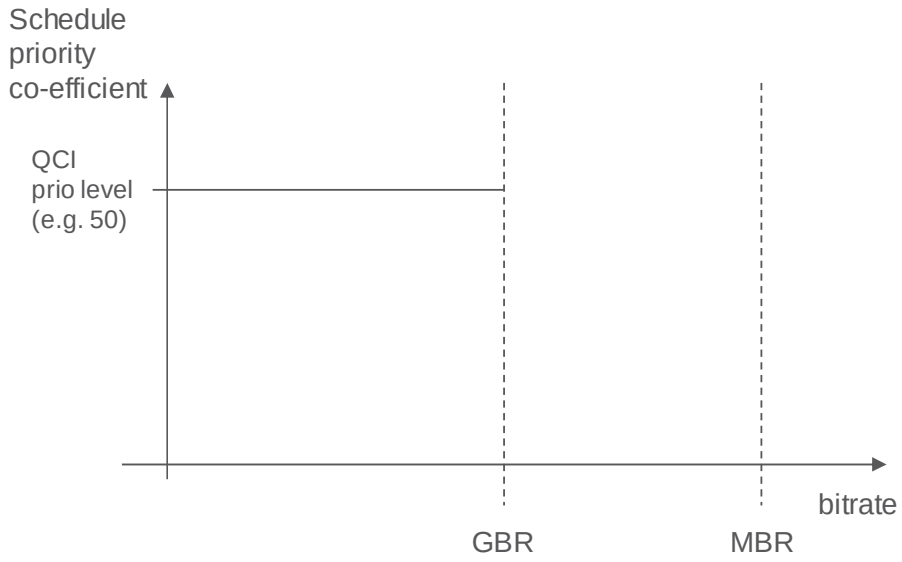

Figure 11: Today's prioritization: traffic gets priorities up to the GBR and is treated as best effort above GBR

The service as introduced in the previous section should typically operate far beyond GFBR/GBR and likely close to MFBR/MBR. If the GFBR/GBR of 3GPP flow/bearer aimed to carry the video traffic is set to the barely acceptable quality level, the scheduling priority will only prioritize the data up to the GFBR/GBR and not really be beneficial to provide bitrates close to the expected service quality. In this case, as the behavior for traffic between GFBR/GBR and MFBR/MBR is equal to best-effort MBB, then it is probably often better to skip QoS and instead use a non-GBR flow/bearer with high PL (which is likely also cheaper) for the video traffic.

If, on the other hand the GFBR/GBR value of the of 3GPP flow/bearer aimed to carry the video traffic is set to the target quality level, the scheduling priority would lead to the scheduler to prioritize the video traffic up to the target quality level at the cost of more radio resource consumption and reducing the room for the rate adaptation capabilities of the video traffic. While it is clearly desirable to use the target quality, the needed quality/cost trade-off is less optimal in this case, since the cost to guarantee the target quality at all times can easily become too high.

Further, there is an increased risk, that the system is rejecting / dropping the QoS bearer.

9.2.5 Desired QoS flow behavior

In the following, we discuss the usage of the 3GPP QoS framework.

The system admission control is going to reject / pre-empt a QoS bearer based on the GBR value. In order to get a QoS bearer accepted, the GBR value should be selected as the lowest acceptable bitrate. With increasing GBR value, also the risk is increasing that the system admission control is rejecting / pre-empty QoS bearers based on the GBR value. Note, handovers to other cells / other access networks may retrigger the admission control process.

The MBR is limiting the bitrate of the QoS bearer. In some implementations, the system is dropping traffic when the service bitrate is above MBR. Thus, due to burstiness of video traffic and when a bitrate adaptation principle is available, the MBR should be much larger than the GBR.

The (video) application layer will to tear-down the delivery of the data, when the bitrate (and the resulting quality) falls below the lower threshold, which is indicated as GBR in the figure 12 below. The preferred service operation point (called target bitrate, TBR) is much higher than the GBR and likely close to the MBR. The FLUS source may adapt the media bitrate to the current estimated link bitrate.

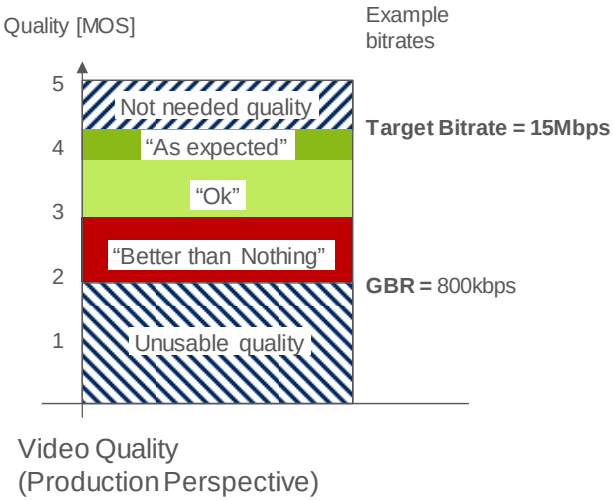

Figure 12: QoS Threshold boundaries

So, a desired behaviour would be when the priority level of a QoS flow does not fall flat to zero once the media bitrate is above GBR/GFBR bitrate. Instead, it would be preferred that the scheduling priority level should decrease gradually with the increasing bitrate. The system should prioritize the QoS flow. The level of prioritization may decrease with increasing media bitrate. As result, the traffic within the QoS bearer would still be treated better than best effort, when the media bitrate is above GBR/GFBR.

Preferable it should be possible to define the priority to get bitrates above GFBR separate from the priority to get the GFBR fulfilled. For the broadcasting media example described here it is prioritized to get high bitrates, but for other services, i.e. public safety, it might be very important to get the GFBR, while higher bitrates have low priority.

Annex A:
Immersive media signalling

A.1 General

This clause defines the parameters for signalling immersive media sessions. A media client may support all or a subset of these parameters, and the receiver of SDP may ignore the parameters it cannot understand or whose usages do not comply with the definitions. These parameters, which are associated with a media-level attribute, mediagmtr, may be used with RTP or other transport protocols. The reference coordinate system for these parameters is defined in [2].

The syntax for the attribute, following ABNF, is as follows:

media-geometry = "mediagmtr:" PT 1*2 (1*WSP ("send" / "recv") 1*WSP attr-list) 1*WSP "vp-depend=" vp-depend

PT = 1*DIGIT / "*"

attr-list = (set *(1*WSP set)) / "*"

; WSP and DIGIT are defined in [11]

set = "[" "az=" az ";" "el=" el "]"

NOTE 1: Syntax of az and el depends on the payload type they specify, due to the intrinsic differences between media. If they are used for audio, definitions in clause B.1 are used. If they are used for video, definitions in clause B.2 are used.

NOTE 2: If both azimuth and elevation angles are specified for audio, the numbers of angles are identical.

A.2 Audio

The following parameters are applicable in sessions including channel-based audio.

az: specifies the azimuth angles of audio channels in degrees, for the send or receive direction. The parameter can have a single angle or a comma-separated list of angles, and each angle is a real number greater than or equal to -180 but less than or equal to 180.

el: specifies the elevation angles of audio channels in degrees, for the send or receive direction. The parameter can have a single angle or a comma-separated list of angles, and each angle is a real number greater than or equal to -90 but less than or equal to 90.

vp-depend: Permissible values are 0 and 1. If vp-depend is 0, the audio signals are captured in fixed directions. If vp-depend is 1 and information on the direction of viewport at the receiver is available at the sender, the audio signals are captured in directions taking the direction of viewport into account.

NOTE: The audio signals are assumed to approach the origin in the specified directions.

A.3 Video

The following parameters are applicable in sessions including video.

az: specifies the range of azimuth angle for video in degrees, for the send or receive direction. The parameter can have a hyphen-separated pair of two angles (az1-az2), and each angle is a real number greater than or equal to -180 but less than or equal to 180. az1 is smaller than az2.

el: specifies the range of elevation angle for video in degrees, for the send or receive direction. The parameter can have a hyphen-separated pair of two angles (el1-el2), and each angle is a real number greater than or equal to -90 but less than or equal to 90. el1 is smaller than el2.

vp-depend: Permissible values are 0 and 1. If vp-depend is 0, the video signals are captured in fixed directions. If vp-depend is 1 and information on the direction of viewport at the receiver is available at the sender, the video signals are captured in directions taking the direction of viewport into account.

NOTE 1: The video signals are assumed to be projected on the internal surface of a spherical display.

NOTE 2: If neither azimuth nor elevation angle is specified, the video signals are assumed to be projected on a flat display whose resolution is specified by the imageattr attribute.

A.4 Examples of SDP offers and answers

A.4.1 H.264 (AVC), H.265 (HEVC), and EVS

The SDP offer includes H.264/H.265 for video and EVS for audio. In this example, direction of audio channels and range of video viewport are negotiated. Although two video codecs are offered, only H.265 is considered for a spherical display. EVS is offered as a dual-mono configuration with DTX disabled and audio bandwidth maximized to fullband, and also as a conventional configuration for super-wideband telephony. It is assumed that further information related to media handling, e.g., parameters on the projection or packing of video, is signalled using other methods.

Table A.4.1: Example SDP offer

SDP offer
m=audio 49152 RTP/AVP 97 98 b=AS:146 b=RS:0 b=RR:2000 a=rtpmap:97 EVS/16000/2 a=fmtp:97 br-send=64; bw-send=nb-fb; ch-send=2; dtx=0; max-red=220 a=mediagmtr:97 send [az=-40,40;el=45,45] vp-depend=1 a=rtpmap:98 EVS/16000/1 a=fmtp:98 br-send=5.9-24.4; bw-send=nb-swb; max-red=220 a=ptime:20 a=maxptime:240 a=sendonly m=video 49154 RTP/AVP 99 100 b=AS:15000 b=RS:0 b=RR:5000 a=rtpmap:99 H265/90000 a=fmtp:99 profile-id=1; level-id=51 a=imageattr:99 send [x=3840,y=2160] a=mediagmtr:99 send [az=-180-180;el=-90-90] vp-depend=1 a=rtpmap:100 H264/90000 a=fmtp:100 packetization-mode=0; profile-level-id=42e01f a=imageattr:100 send [x=640,y=480] a=sendonly

A maximum of 146 kbps is offered for a dual-mono configuration of EVS at 64 kbps. Two audio channels are offered in two directions that share the same elevation. Although not shown in the offer, an alternative of 42 kbps is also provided for a mono configuration at bit-rates up to 24.4 kbps. These are offered as one-way transmission from the media sender. In addition, a maximum of 15 Mbps is offered for an omnidirectional 4K video encoded with H.265. Level 5.1 of this codec supports resolution and frame rate up to 4K and 60 fps respectively. An alternative is a lower-resolution video encoded with H.264, which is expected to be projected on a flat display.

Table A.4.2: Example SDP answer

SDP answer
m=audio 49152 RTP/AVP 97 b=AS:146 b=RS:0 b=RR:2000 a=rtpmap:97 EVS/16000/2 a=fmtp:97 br-recv=64; bw-recv=nb-fb; ch-recv=2; dtx=0; max-red=220 a=mediagmtr:97 recv [az=-40,40;el=45,45] vp-depend=1 a=ptime:20 a=maxptime:240 a=recvonly m=video 49154 RTP/AVP 99 b=AS:10000 b=RS:0 b=RR:5000 a=rtpmap:99 H265/90000 a=fmtp:99 profile-id=1; level-id=51 a=imageattr:99 recv [x=3840,y=2160] a=mediagmtr:99 recv [az=-120-120;el=-90-90] vp-depend=1 a=recvonly

In the SDP answer, from the offered media configurations, a dual-mono configuration of EVS at 64 kbps and a 4K video encoded with H.265 were selected. In the case of video, the bit-rate is reduced to 10 Mbps as the range of video viewport is reduced by a third, i.e. 360 to 240 degrees, in azimuth. The directions in the attributes and parameters are all reversed. Figure B.3.1 illustrates the geometry of audiovisual media negotiated.

The user is assumed to be located inside the partial sphere. The video is projected on the internal surface of the sphere, and the two arrows represent the direction of audio channels. As both configurations were offered and answered with vp-depend=1, the media sender will take the received information on the video viewport at the media receiver, if available, into account.

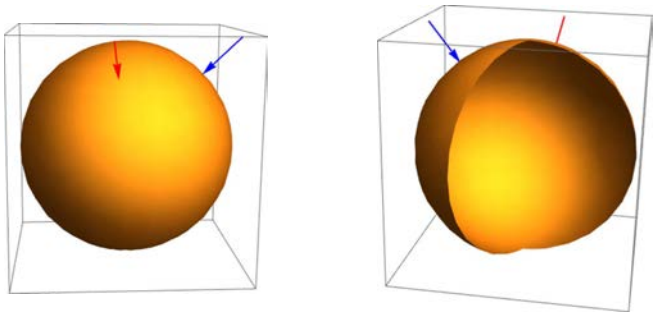

Figure A.4.1: Negotiated media geometry

Annex B:
Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
06-2018	SA#80	SP-180267				Presented to TSG SA#80 (for approval)	1.0.0
06-2018	SA#80					Approved at TSG SA#80 plenary meeting	15.0.0
09-2018	SA#81	SP-180640	0001	2	F	Corrections of fMP4 instantiation description	15.1.0

History

Document history		
V15.0.0	July 2018	Publication
V15.1.0	October 2018	Publication