(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 April 2017 (20.04.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/063189 A1

(51) International Patent Classification:
H04L 29/06 (2006.01) *H04N 7/04* (2006.01)
H04L 29/08 (2006.01)

(21) International Application Number:
PCT/CN2015/092095

(22) International Filing Date:
16 October 2015 (16.10.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventors; and
(71) Applicants *(for US only)*: **STOCKHAMMER, Thomas** [DE/DE]; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **ZHU, Xipeng** [CN/CN]; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **WALKER, Gordon Kent** [US/US]; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: **LEE AND LI - LEAVEN IPR AGENCY LTD.**; Unit 2202, Tower A, Beijing Marriott Center, No. 7, Jian Guo Men South Avenue, Dongcheng, Beijing 100005 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

(54) **Title**: DEADLINE SIGNALING FOR STREAMING OF MEDIA DATA

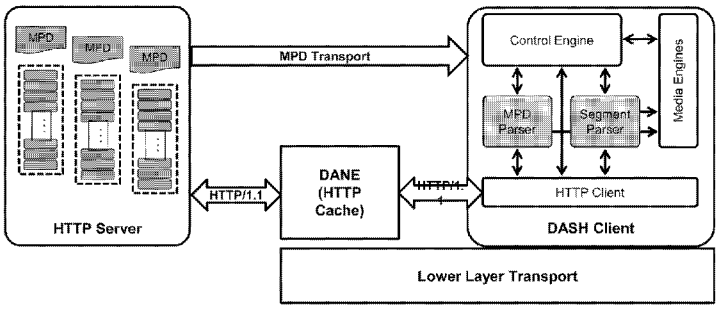

FIG. 8

(57) **Abstract**: In one example, a client device for retrieving data with real-time constraints includes means for determining times during which the data will be available for download, means for determining a time at which the data is needed to fulfill the real-time constraints, and means for sending a request for the data and deadline information representative of the time at which the data is needed to fulfill the real-time constraints. In another example, a network element for transporting data with real-time constraints includes means for receiving deadline information representative of a time at which the data is needed by a client device to fulfill the real-time constraints, means for retrieving the data; and means for delivering the data to the client device such that the data is available to the client device at or before the time at which the data is needed.

WO 2017/063189 A1

DEADLINE SIGNALING FOR STREAMING OF MEDIA DATA

TECHNICAL FIELD

[0001] This disclosure relates to transport of media data.

BACKGROUND

[0002] Digital media capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, video teleconferencing devices, and the like.

[0003] Digital media may be compressed prior to transmission. Video data, for example, may be compressed using video compression techniques that perform spatial prediction and/or temporal prediction to reduce or remove redundancy inherent in video sequences.

[0004] After media data has been encoded, the media data may be packetized for transmission or storage. The media data may be assembled into a media file conforming to any of a variety of standards, such as the International Organization for Standardization (ISO) base media file format and extensions thereof. The media data may further be transmitted using a computer-based network via a streaming protocol, such as Dynamic Adaptive Streaming over HTTP (DASH).

SUMMARY

[0005] In general, this disclosure describes techniques for signaling deadline information for media data. That is, a client device may signal data representative of a time at which a media file, such as a DASH segment, must be received. This time may represent the time by which the media file must be received in order to ensure smooth, continuous playout by the client device.

[0006] The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0007] FIG. 1 is a block diagram illustrating an example system that implements techniques for streaming media data over a network.

[0008] FIG. 2 is a block diagram illustrating an example set of components of a retrieval unit.

[0009] FIG. 3 is a conceptual diagram illustrating elements of example multimedia content.

[0010] FIG. 4 is a block diagram illustrating elements of an example video file, which may correspond to a segment of a representation.

[0011] FIG. 5 is a conceptual diagram illustrating a common understanding of HTTP adaptive streaming.

[0012] FIG. 6 is a conceptual diagram illustrating responsibilities of "smart" client devices.

[0013] FIG. 7 is a conceptual diagram illustrating the scope of MPEG DASH.

[0014] FIG. 8 is a conceptual diagram illustrating an example architecture according to the techniques of this disclosure.

[0015] FIG. 9 is a conceptual diagram illustrating a simple client model.

[0016] FIG. 10 is a conceptual diagram illustrating segments from the perspective of a server and a client.

[0017] FIG. 11 is a conceptual diagram illustrating one example for video object deadline aware scheduling.

[0018] FIG. 12 is a conceptual diagram of an example implementation according to the example of FIG. 11.

[0019] FIG. 13 is a conceptual diagram illustrating another example solution based on video object deadline aware scheduling.

[0020] FIG. 14 is a conceptual diagram illustrating an example implementation according to the techniques of FIG. 13.

[0021] FIG. 15 is a conceptual diagram illustrating an example of packetization in accordance with the techniques of this disclosure.

[0022] FIG. 16 is a conceptual diagram illustrating a series of segments, including an initialization segment (IS) and a plurality of media segments (MSs).

[0023] FIGS. 17A and 17B are conceptual diagrams representing a playout curve for a media segment.

[0024] FIG. 18 is a conceptual diagram of media delivery events (MDEs) for delivering data of segments.

[0025] FIG. 19 is a conceptual diagram illustrating another example that may be used as a potential simplification.

[0026] FIG. 20 is a conceptual diagram of an example delivery architecture in accordance with the techniques of this disclosure.

[0027] FIG. 21 is a conceptual diagram of a receiver model in accordance with the techniques of this disclosure.

DETAILED DESCRIPTION

[0028] In general, this disclosure describes techniques for signaling deadline information during streaming of media data using hypertext transfer protocol (HTTP). Such streaming techniques are also referred to herein as HTTP streaming. In particular, as explained below, this disclosure describes techniques by which a streaming client of a client device may signal deadline information to a streaming aware network element, to attempt to ensure prompt delivery of segments by respective deadlines. Similarly, the streaming aware network element may use these techniques to deliver segments to client devices such that the segments reach the client devices by their respective deadlines.

[0029] In HTTP streaming, frequently used operations include HEAD, GET, and partial GET. The HEAD operation retrieves a header of a file associated with a given uniform resource locator (URL) or uniform resource name (URN), without retrieving a payload associated with the URL or URN. The GET operation retrieves a whole file associated with a given URL or URN. The partial GET operation receives a byte range as an input parameter and retrieves a continuous number of bytes of a file, where the number of bytes correspond to the received byte range. Thus, movie fragments may be provided for HTTP streaming, because a partial GET operation can get one or more individual movie fragments. In a movie fragment, there can be several track fragments of different tracks. In HTTP streaming, a media presentation may be a structured collection of data that is accessible to the client. The client may request and download media data information to present a streaming service to a user.

[0030] In the example of streaming 3GPP data using HTTP streaming, there may be multiple representations for video and/or audio data of multimedia content. As explained below, different representations may correspond to different coding

characteristics (e.g., different profiles or levels of a video coding standard), different coding standards or extensions of coding standards (such as multiview and/or scalable extensions), or different bitrates. The manifest of such representations may be defined in a Media Presentation Description (MPD) data structure. A media presentation may correspond to a structured collection of data that is accessible to an HTTP streaming client device. The HTTP streaming client device may request and download media data information to present a streaming service to a user of the client device. A media presentation may be described in the MPD data structure, which may include updates of the MPD.

[0031] A media presentation may contain a sequence of one or more periods. Periods may be defined by a *Period* element in the MPD. Each period may have an attribute *start* in the MPD. The MPD may include a *start* attribute and an *availableStartTime* attribute for each period. For live services, the sum of the *start* attribute of the period and the MPD attribute *availableStartTime* may specify the availability time of the period in UTC format, in particular the first Media Segment of each representation in the corresponding period. For on-demand services, the *start* attribute of the first period may be 0. For any other period, the *start* attribute may specify a time offset between the start time of the corresponding Period relative to the start time of the first Period. Each period may extend until the start of the next Period, or until the end of the media presentation in the case of the last period. Period start times may be precise. They may reflect the actual timing resulting from playing the media of all prior periods.

[0032] Each period may contain one or more representations for the same media content. A representation may be one of a number of alternative encoded versions of audio or video data. The representations may differ by encoding types, e.g., by bitrate, resolution, and/or codec for video data and bitrate, language, and/or codec for audio data. The term representation may be used to refer to a section of encoded audio or video data corresponding to a particular period of the multimedia content and encoded in a particular way.

[0033] Representations of a particular period may be assigned to a group indicated by an attribute in the MPD indicative of an adaptation set to which the representations belong. Representations in the same adaptation set are generally considered alternatives to each other, in that a client device can dynamically and seamlessly switch between these representations, e.g., to perform bandwidth adaptation. For example, each representation of video data for a particular period may be assigned to the same

adaptation set, such that any of the representations may be selected for decoding to present media data, such as video data or audio data, of the multimedia content for the corresponding period. The media content within one period may be represented by either one representation from group 0, if present, or the combination of at most one representation from each non-zero group, in some examples. Timing data for each representation of a period may be expressed relative to the start time of the period.

[0034] A representation may include one or more segments. Each representation may include an initialization segment, or each segment of a representation may be self-initializing. When present, the initialization segment may contain initialization information for accessing the representation. In general, the initialization segment does not contain media data. A segment may be uniquely referenced by an identifier, such as a uniform resource locator (URL), uniform resource name (URN), or uniform resource identifier (URI). The MPD may provide the identifiers for each segment. In some examples, the MPD may also provide byte ranges in the form of a *range* attribute, which may correspond to the data for a segment within a file accessible by the URL, URN, or URI.

[0035] Different representations may be selected for substantially simultaneous retrieval for different types of media data. For example, a client device may select an audio representation, a video representation, and a timed text representation from which to retrieve segments. In some examples, the client device may select particular adaptation sets for performing bandwidth adaptation. That is, the client device may select an adaptation set including video representations, an adaptation set including audio representations, and/or an adaptation set including timed text. Alternatively, the client device may select adaptation sets for certain types of media (e.g., video), and directly select representations for other types of media (e.g., audio and/or timed text).

[0036] DASH enables object-based real-time streaming delivery. In the basic operation mode, the client requests data from the server and schedules the playout. The DASH client uses buffers in order to optimize the playout and avoid buffer underruns. Also the client schedules the requests for Segments properly in order to ensure that the Segments arrive at the client in order to ensure proper playout. In the basic operation, all control and timing is with the client.

[0037] However, in certain scenarios, specifically, in the case considered in Server and Network Assisted DASH (SAND), the server and network cooperate with and assist the client in order to optimize the delivery primarily in terms of network efficiency as well

as user experience. In addition, as the HTTP requests are typically handled as stateless and timeless requests in the network, the client may support the network in the delivery of objects over HTTP, especially if the network is aware of deadlines for the delivered. Such technologies are particularly relevant in cases where the network can make use of such deadlines in the delivery.

[0038] In this context, this disclosure proposes the addition of the following messages in the context of SAND:

- A status message to provide an absolute deadline (wall-clock) for the requested object in the receiver.
- A status message to provide an maximum RTT (duration) for the requested object
- A PED message to provide the relative deadline of the different byte ranges of the segment
- A status message to provide the relative deadline of the different byte ranges of the segment

[0039] These messages may be sent by a client device to a network element, such as a streaming aware network element, a DASH aware network element, or a DASH server to cause the DASH server to forward the messages to a streaming aware network element or a DASH aware network element.

[0040] The techniques of this disclosure may yield certain advantages. For example, the techniques of this disclosure may provide information to a network element, such as a DASH aware network element (DANE) or a media aware network element (MANE), of timing requirements of a client device for data to be delivered to the client device. This may ensure that the client device receives the data when the data is needed by the client device, which may satisfy real-time constraints of the client device for the data. For example, in the context of DASH, media data may be required at certain times to avoid buffer underflow (that is, the client device consuming all buffered data). Buffer underflow would generally result in needing to await reception of additional data, which may cause an undesirable pause in playback. These techniques may avoid such a pause, by avoiding the buffer underflow.

[0041] FIG. 1 is a block diagram illustrating an example system 10 that implements techniques for streaming media data over a network. In this example, system 10 includes content preparation device 20, server device 60, and client device 40. Client

device 40 and server device 60 are communicatively coupled by network 74, which may comprise the Internet. In some examples, content preparation device 20 and server device 60 may also be coupled by network 74 or another network, or may be directly communicatively coupled. In some examples, content preparation device 20 and server device 60 may comprise the same device.

[0042] Content preparation device 20, in the example of FIG. 1, comprises audio source 22 and video source 24. Audio source 22 may comprise, for example, a microphone that produces electrical signals representative of captured audio data to be encoded by audio encoder 26. Alternatively, audio source 22 may comprise a storage medium storing previously recorded audio data, an audio data generator such as a computerized synthesizer, or any other source of audio data. Video source 24 may comprise a video camera that produces video data to be encoded by video encoder 28, a storage medium encoded with previously recorded video data, a video data generation unit such as a computer graphics source, or any other source of video data. Content preparation device 20 is not necessarily communicatively coupled to server device 60 in all examples, but may store multimedia content to a separate medium that is read by server device 60.

[0043] Raw audio and video data may comprise analog or digital data. Analog data may be digitized before being encoded by audio encoder 26 and/or video encoder 28. Audio source 22 may obtain audio data from a speaking participant while the speaking participant is speaking, and video source 24 may simultaneously obtain video data of the speaking participant. In other examples, audio source 22 may comprise a computer-readable storage medium comprising stored audio data, and video source 24 may comprise a computer-readable storage medium comprising stored video data. In this manner, the techniques described in this disclosure may be applied to live, streaming, real-time audio and video data or to archived, pre-recorded audio and video data.

[0044] Audio frames that correspond to video frames are generally audio frames containing audio data that was captured (or generated) by audio source 22 contemporaneously with video data captured (or generated) by video source 24 that is contained within the video frames. For example, while a speaking participant generally produces audio data by speaking, audio source 22 captures the audio data, and video source 24 captures video data of the speaking participant at the same time, that is, while audio source 22 is capturing the audio data. Hence, an audio frame may temporally correspond to one or more particular video frames. Accordingly, an audio frame

corresponding to a video frame generally corresponds to a situation in which audio data and video data were captured at the same time and for which an audio frame and a video frame comprise, respectively, the audio data and the video data that was captured at the same time.

[0045] In some examples, audio encoder 26 may encode a timestamp in each encoded audio frame that represents a time at which the audio data for the encoded audio frame was recorded, and similarly, video encoder 28 may encode a timestamp in each encoded video frame that represents a time at which the video data for encoded video frame was recorded. In such examples, an audio frame corresponding to a video frame may comprise an audio frame comprising a timestamp and a video frame comprising the same timestamp. Content preparation device 20 may include an internal clock from which audio encoder 26 and/or video encoder 28 may generate the timestamps, or that audio source 22 and video source 24 may use to associate audio and video data, respectively, with a timestamp.

[0046] In some examples, audio source 22 may send data to audio encoder 26 corresponding to a time at which audio data was recorded, and video source 24 may send data to video encoder 28 corresponding to a time at which video data was recorded. In some examples, audio encoder 26 may encode a sequence identifier in encoded audio data to indicate a relative temporal ordering of encoded audio data but without necessarily indicating an absolute time at which the audio data was recorded, and similarly, video encoder 28 may also use sequence identifiers to indicate a relative temporal ordering of encoded video data. Similarly, in some examples, a sequence identifier may be mapped or otherwise correlated with a timestamp.

[0047] Audio encoder 26 generally produces a stream of encoded audio data, while video encoder 28 produces a stream of encoded video data. Each individual stream of data (whether audio or video) may be referred to as an elementary stream. An elementary stream is a single, digitally coded (possibly compressed) component of a representation. For example, the coded video or audio part of the representation can be an elementary stream. An elementary stream may be converted into a packetized elementary stream (PES) before being encapsulated within a video file. Within the same representation, a stream ID may be used to distinguish the PES-packets belonging to one elementary stream from the other. The basic unit of data of an elementary stream is a packetized elementary stream (PES) packet. Thus, coded video data generally

corresponds to elementary video streams. Similarly, audio data corresponds to one or more respective elementary streams.

[0048] Many video coding standards, such as ITU-T H.264/AVC and the High Efficiency Video Coding (HEVC) standard, define the syntax, semantics, and decoding process for error-free bitstreams, any of which conform to a certain profile or level. Video coding standards typically do not specify the encoder, but the encoder is tasked with guaranteeing that the generated bitstreams are standard-compliant for a decoder. In the context of video coding standards, a "profile" corresponds to a subset of algorithms, features, or tools and constraints that apply to them. As defined by the H.264 standard, for example, a "profile" is a subset of the entire bitstream syntax that is specified by the H.264 standard. A "level" corresponds to the limitations of the decoder resource consumption, such as, for example, decoder memory and computation, which are related to the resolution of the pictures, bit rate, and block processing rate. A profile may be signaled with a profile_idc (profile indicator) value, while a level may be signaled with a level_idc (level indicator) value.

[0049] The H.264 standard, for example, recognizes that, within the bounds imposed by the syntax of a given profile, it is still possible to require a large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. The H.264 standard further recognizes that, in many applications, it is neither practical nor economical to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile. Accordingly, the H.264 standard defines a "level" as a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively, these constraints may take the form of constraints on arithmetic combinations of values (e.g., picture width multiplied by picture height multiplied by number of pictures decoded per second). The H.264 standard further provides that individual implementations may support a different level for each supported profile.

[0050] A decoder conforming to a profile ordinarily supports all the features defined in the profile. For example, as a coding feature, B-picture coding is not supported in the baseline profile of H.264/AVC but is supported in other profiles of H.264/AVC. A decoder conforming to a level should be capable of decoding any bitstream that does not require resources beyond the limitations defined in the level. Definitions of profiles and levels may be helpful for interpretability. For example, during video transmission, a

pair of profile and level definitions may be negotiated and agreed for a whole transmission session. More specifically, in H.264/AVC, a level may define limitations on the number of macroblocks that need to be processed, decoded picture buffer (DPB) size, coded picture buffer (CPB) size, vertical motion vector range, maximum number of motion vectors per two consecutive MBs, and whether a B-block can have sub-macroblock partitions less than 8x8 pixels. In this manner, a decoder may determine whether the decoder is capable of properly decoding the bitstream.

[0051] In the example of FIG. 1, encapsulation unit 30 of content preparation device 20 receives elementary streams comprising coded video data from video encoder 28 and elementary streams comprising coded audio data from audio encoder 26. In some examples, video encoder 28 and audio encoder 26 may each include packetizers for forming PES packets from encoded data. In other examples, video encoder 28 and audio encoder 26 may each interface with respective packetizers for forming PES packets from encoded data. In still other examples, encapsulation unit 30 may include packetizers for forming PES packets from encoded audio and video data.

[0052] Video encoder 28 may encode video data of multimedia content in a variety of ways, to produce different representations of the multimedia content at various bitrates and with various characteristics, such as pixel resolutions, frame rates, conformance to various coding standards, conformance to various profiles and/or levels of profiles for various coding standards, representations having one or multiple views (e.g., for two-dimensional or three-dimensional playback), or other such characteristics. A representation, as used in this disclosure, may comprise one of audio data, video data, text data (e.g., for closed captions), or other such data. The representation may include an elementary stream, such as an audio elementary stream or a video elementary stream. Each PES packet may include a stream_id that identifies the elementary stream to which the PES packet belongs. Encapsulation unit 30 is responsible for assembling elementary streams into video files (e.g., segments) of various representations.

[0053] Encapsulation unit 30 receives PES packets for elementary streams of a representation from audio encoder 26 and video encoder 28 and forms corresponding network abstraction layer (NAL) units from the PES packets. In the example of H.264/AVC (Advanced Video Coding), coded video segments are organized into NAL units, which provide a "network-friendly" video representation addressing applications such as video telephony, storage, broadcast, or streaming. NAL units can be categorized to Video Coding Layer (VCL) NAL units and non-VCL NAL units. VCL

units may contain the core compression engine and may include block, macroblock, and/or slice level data. Other NAL units may be non-VCL NAL units. In some examples, a coded picture in one time instance, normally presented as a primary coded picture, may be contained in an access unit, which may include one or more NAL units.

[0054] Non-VCL NAL units may include parameter set NAL units and SEI NAL units, among others. Parameter sets may contain sequence-level header information (in sequence parameter sets (SPS)) and the infrequently changing picture-level header information (in picture parameter sets (PPS)). With parameter sets (e.g., PPS and SPS), infrequently changing information need not to be repeated for each sequence or picture, hence coding efficiency may be improved. Furthermore, the use of parameter sets may enable out-of-band transmission of the important header information, avoiding the need for redundant transmissions for error resilience. In out-of-band transmission examples, parameter set NAL units may be transmitted on a different channel than other NAL units, such as SEI NAL units.

[0055] Supplemental Enhancement Information (SEI) may contain information that is not necessary for decoding the coded pictures samples from VCL NAL units, but may assist in processes related to decoding, display, error resilience, and other purposes. SEI messages may be contained in non-VCL NAL units. SEI messages are the normative part of some standard specifications, and thus are not always mandatory for standard compliant decoder implementation. SEI messages may be sequence level SEI messages or picture level SEI messages. Some sequence level information may be contained in SEI messages, such as scalability information SEI messages in the example of SVC and view scalability information SEI messages in MVC. These example SEI messages may convey information on, e.g., extraction of operation points and characteristics of the operation points. In addition, encapsulation unit 30 may form a manifest file, such as a media presentation descriptor (MPD) that describes characteristics of the representations. Encapsulation unit 30 may format the MPD according to extensible markup language (XML).

[0056] Encapsulation unit 30 may provide data for one or more representations of multimedia content, along with the manifest file (e.g., the MPD) to output interface 32. Output interface 32 may comprise a network interface or an interface for writing to a storage medium, such as a universal serial bus (USB) interface, a CD or DVD writer or burner, an interface to magnetic or flash storage media, or other interfaces for storing or transmitting media data. Encapsulation unit 30 may provide data of each of the

representations of multimedia content to output interface 32, which may send the data to server device 60 via network transmission or storage media. In the example of FIG. 1, server device 60 includes storage medium 62 that stores various multimedia contents 64, each including a respective manifest file 66 and one or more representations 68A–68N (representations 68). In some examples, output interface 32 may also send data directly to network 74.

[0057] In some examples, representations 68 may be separated into adaptation sets. That is, various subsets of representations 68 may include respective common sets of characteristics, such as codec, profile and level, resolution, number of views, file format for segments, text type information that may identify a language or other characteristics of text to be displayed with the representation and/or audio data to be decoded and presented, e.g., by speakers, camera angle information that may describe a camera angle or real-world camera perspective of a scene for representations in the adaptation set, rating information that describes content suitability for particular audiences, or the like.

[0058] Manifest file 66 may include data indicative of the subsets of representations 68 corresponding to particular adaptation sets, as well as common characteristics for the adaptation sets. Manifest file 66 may also include data representative of individual characteristics, such as bitrates, for individual representations of adaptation sets. In this manner, an adaptation set may provide for simplified network bandwidth adaptation. Representations in an adaptation set may be indicated using child elements of an adaptation set element of manifest file 66.

[0059] Server device 60 includes request processing unit 70 and network interface 72. In some examples, server device 60 may include a plurality of network interfaces. Furthermore, any or all of the features of server device 60 may be implemented on other devices of a content delivery network, such as routers, bridges, proxy devices, switches, or other devices. In some examples, intermediate devices of a content delivery network may cache data of multimedia content 64, and include components that conform substantially to those of server device 60. In general, network interface 72 is configured to send and receive data via network 74.

[0060] Request processing unit 70 is configured to receive network requests from client devices, such as client device 40, for data of storage medium 62. For example, request processing unit 70 may implement hypertext transfer protocol (HTTP) version 1.1, as described in RFC 2616, "Hypertext Transfer Protocol – HTTP/1.1," by R. Fielding et al, Network Working Group, IETF, June 1999. That is, request processing unit 70 may be

configured to receive HTTP GET or partial GET requests and provide data of multimedia content 64 in response to the requests. The requests may specify a segment of one of representations 68, e.g., using a URL of the segment. In some examples, the requests may also specify one or more byte ranges of the segment, thus comprising partial GET requests. Request processing unit 70 may further be configured to service HTTP HEAD requests to provide header data of a segment of one of representations 68. In any case, request processing unit 70 may be configured to process the requests to provide requested data to a requesting device, such as client device 40.

[0061] Additionally or alternatively, request processing unit 70 may be configured to deliver media data via a broadcast or multicast protocol, such as eMBMS. Content preparation device 20 may create DASH segments and/or sub-segments in substantially the same way as described, but server device 60 may deliver these segments or sub-segments using eMBMS or another broadcast or multicast network transport protocol. For example, request processing unit 70 may be configured to receive a multicast group join request from client device 40. That is, server device 60 may advertise an Internet protocol (IP) address associated with a multicast group to client devices, including client device 40, associated with particular media content (e.g., a broadcast of a live event). Client device 40, in turn, may submit a request to join the multicast group. This request may be propagated throughout network 74, e.g., routers making up network 74, such that the routers are caused to direct traffic destined for the IP address associated with the multicast group to subscribing client devices, such as client device 40.

[0062] As illustrated in the example of FIG. 1, multimedia content 64 includes manifest file 66, which may correspond to a media presentation description (MPD). Manifest file 66 may contain descriptions of different alternative representations 68 (e.g., video services with different qualities) and the description may include, e.g., codec information, a profile value, a level value, a bitrate, and other descriptive characteristics of representations 68. Client device 40 may retrieve the MPD of a media presentation to determine how to access segments of representations 68.

[0063] In particular, retrieval unit 52 may retrieve configuration data (not shown) of client device 40 to determine decoding capabilities of video decoder 48 and rendering capabilities of video output 44. The configuration data may also include any or all of a language preference selected by a user of client device 40, one or more camera perspectives corresponding to depth preferences set by the user of client device 40, and/or a rating preference selected by the user of client device 40. Retrieval unit 52

may comprise, for example, a web browser or a media client configured to submit HTTP GET and partial GET requests. Retrieval unit 52 may correspond to software instructions executed by one or more processors or processing units (not shown) of client device 40. In some examples, all or portions of the functionality described with respect to retrieval unit 52 may be implemented in hardware, or a combination of hardware, software, and/or firmware, where requisite hardware may be provided to execute instructions for software or firmware.

[0064] Retrieval unit 52 may compare the decoding and rendering capabilities of client device 40 to characteristics of representations 68 indicated by information of manifest file 66. Retrieval unit 52 may initially retrieve at least a portion of manifest file 66 to determine characteristics of representations 68. For example, retrieval unit 52 may request a portion of manifest file 66 that describes characteristics of one or more adaptation sets. Retrieval unit 52 may select a subset of representations 68 (e.g., an adaptation set) having characteristics that can be satisfied by the coding and rendering capabilities of client device 40. Retrieval unit 52 may then determine bitrates for representations in the adaptation set, determine a currently available amount of network bandwidth, and retrieve segments from one of the representations having a bitrate that can be satisfied by the network bandwidth.

[0065] In general, higher bitrate representations may yield higher quality video playback, while lower bitrate representations may provide sufficient quality video playback when available network bandwidth decreases. Accordingly, when available network bandwidth is relatively high, retrieval unit 52 may retrieve data from relatively high bitrate representations, whereas when available network bandwidth is low, retrieval unit 52 may retrieve data from relatively low bitrate representations. In this manner, client device 40 may stream multimedia data over network 74 while also adapting to changing network bandwidth availability of network 74.

[0066] Additionally or alternatively, retrieval unit 52 may be configured to receive data in accordance with a broadcast or multicast network protocol, such as eMBMS or IP multicast. In such examples, retrieval unit 52 may submit a request to join a multicast network group associated with particular media content. After joining the multicast group, retrieval unit 52 may receive data of the multicast group without further requests issued to server device 60 or content preparation device 20. Retrieval unit 52 may submit a request to leave the multicast group when data of the multicast group is no

longer needed, e.g., to stop playback or to change channels to a different multicast group.

[0067] Network interface 54 may receive and provide data of segments of a selected representation to retrieval unit 52, which may in turn provide the segments to decapsulation unit 50. Decapsulation unit 50 may decapsulate elements of a video file into constituent PES streams, depacketize the PES streams to retrieve encoded data, and send the encoded data to either audio decoder 46 or video decoder 48, depending on whether the encoded data is part of an audio or video stream, e.g., as indicated by PES packet headers of the stream. Audio decoder 46 decodes encoded audio data and sends the decoded audio data to audio output 42, while video decoder 48 decodes encoded video data and sends the decoded video data, which may include a plurality of views of a stream, to video output 44.

[0068] Although not shown in FIG. 1, network 74 may further include a streaming aware network element, such as a DASH aware network element (DANE) or a media aware network element (MANE). Retrieval unit 52 may implement the techniques of this disclosure, described in greater detail below, to advertise deadline information to the streaming aware network element of network 74.

[0069] Video encoder 28, video decoder 48, audio encoder 26, audio decoder 46, encapsulation unit 30, retrieval unit 52, and decapsulation unit 50 each may be implemented as any of a variety of suitable processing circuitry, as applicable, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic circuitry, software, hardware, firmware or any combinations thereof. Each of video encoder 28 and video decoder 48 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined video encoder/decoder (CODEC). Likewise, each of audio encoder 26 and audio decoder 46 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined CODEC. An apparatus including video encoder 28, video decoder 48, audio encoder 26, audio decoder 46, encapsulation unit 30, retrieval unit 52, and/or decapsulation unit 50 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0070] Client device 40, server device 60, and/or content preparation device 20 may be configured to operate in accordance with the techniques of this disclosure. For purposes of example, this disclosure describes these techniques with respect to client device 40

and server device 60. However, it should be understood that content preparation device 20 may be configured to perform these techniques, instead of (or in addition to) server device 60.

[0071] Encapsulation unit 30 may form NAL units comprising a header that identifies a program to which the NAL unit belongs, as well as a payload, e.g., audio data, video data, or data that describes the transport or program stream to which the NAL unit corresponds. For example, in H.264/AVC, a NAL unit includes a 1-byte header and a payload of varying size. A NAL unit including video data in its payload may comprise various granularity levels of video data. For example, a NAL unit may comprise a block of video data, a plurality of blocks, a slice of video data, or an entire picture of video data. Encapsulation unit 30 may receive encoded video data from video encoder 28 in the form of PES packets of elementary streams. Encapsulation unit 30 may associate each elementary stream with a corresponding program.

[0072] Encapsulation unit 30 may also assemble access units from a plurality of NAL units. In general, an access unit may comprise one or more NAL units for representing a frame of video data, as well audio data corresponding to the frame when such audio data is available. An access unit generally includes all NAL units for one output time instance, e.g., all audio and video data for one time instance. For example, if each view has a frame rate of 20 frames per second (fps), then each time instance may correspond to a time interval of 0.05 seconds. During this time interval, the specific frames for all views of the same access unit (the same time instance) may be rendered simultaneously. In one example, an access unit may comprise a coded picture in one time instance, which may be presented as a primary coded picture.

[0073] Accordingly, an access unit may comprise all audio and video frames of a common temporal instance, e.g., all views corresponding to time X. This disclosure also refers to an encoded picture of a particular view as a "view component." That is, a view component may comprise an encoded picture (or frame) for a particular view at a particular time. Accordingly, an access unit may be defined as comprising all view components of a common temporal instance. The decoding order of access units need not necessarily be the same as the output or display order.

[0074] A media presentation may include a media presentation description (MPD), which may contain descriptions of different alternative representations (e.g., video services with different qualities) and the description may include, e.g., codec information, a profile value, and a level value. An MPD is one example of a manifest

file, such as manifest file 66. Client device 40 may retrieve the MPD of a media presentation to determine how to access movie fragments of various presentations. Movie fragments may be located in movie fragment boxes (moof boxes) of video files.

[0075] Manifest file 66 (which may comprise, for example, an MPD) may advertise availability of segments of representations 68. That is, the MPD may include information indicating the wall-clock time at which a first segment of one of representations 68 becomes available, as well as information indicating the durations of segments within representations 68. In this manner, retrieval unit 52 of client device 40 may determine when each segment is available, based on the starting time as well as the durations of the segments preceding a particular segment.

[0076] After encapsulation unit 30 has assembled NAL units and/or access units into a video file based on received data, encapsulation unit 30 passes the video file to output interface 32 for output. In some examples, encapsulation unit 30 may store the video file locally or send the video file to a remote server via output interface 32, rather than sending the video file directly to client device 40. Output interface 32 may comprise, for example, a transmitter, a transceiver, a device for writing data to a computer-readable medium such as, for example, an optical drive, a magnetic media drive (e.g., floppy drive), a universal serial bus (USB) port, a network interface, or other output interface. Output interface 32 outputs the video file to a computer-readable medium, such as, for example, a transmission signal, a magnetic medium, an optical medium, a memory, a flash drive, or other computer-readable medium.

[0077] Network interface 54 may receive a NAL unit or access unit via network 74 and provide the NAL unit or access unit to decapsulation unit 50, via retrieval unit 52. Decapsulation unit 50 may decapsulate a elements of a video file into constituent PES streams, depacketize the PES streams to retrieve encoded data, and send the encoded data to either audio decoder 46 or video decoder 48, depending on whether the encoded data is part of an audio or video stream, e.g., as indicated by PES packet headers of the stream. Audio decoder 46 decodes encoded audio data and sends the decoded audio data to audio output 42, while video decoder 48 decodes encoded video data and sends the decoded video data, which may include a plurality of views of a stream, to video output 44.

[0078] FIG. 2 is a block diagram illustrating an example set of components of retrieval unit 52 of FIG. 1 in greater detail. In this example, retrieval unit 52 includes eMBMS middleware unit 100, DASH client 110, and media application 112.

[0079] In this example, eMBMS middleware unit 100 further includes eMBMS reception unit 106, cache 104, and server unit 102. In this example, eMBMS reception unit 106 is configured to receive data via eMBMS, e.g., according to File Delivery over Unidirectional Transport (FLUTE), described in T. Paila et al., "FLUTE—File Delivery over Unidirectional Transport," Network Working Group, RFC 6726, Nov. 2012, available at http://tools.ietf.org/html/rfc6726. That is, eMBMS reception unit 106 may receive files via broadcast from, e.g., server device 60, which may act as a BM-SC.

[0080] As eMBMS middleware unit 100 receives data for files, eMBMS middleware unit may store the received data in cache 104. Cache 104 may comprise a computer-readable storage medium, such as flash memory, a hard disk, RAM, or any other suitable storage medium.

[0081] Local server unit 102 may act as a server for DASH client 110. For example, local server unit 102 may provide a MPD file or other manifest file to DASH client 110. Local server unit 102 may advertise availability times for segments in the MPD file, as well as hyperlinks from which the segments can be retrieved. These hyperlinks may include a localhost address prefix corresponding to client device 40 (e.g., 127.0.0.1 for IPv4). In this manner, DASH client 110 may request segments from local server unit 102 using HTTP GET or partial GET requests. For example, for a segment available from link http://127.0.0.1/rep1/seg3, DASH client 110 may construct an HTTP GET request that includes a request for http://127.0.0.1/rep1/seg3, and submit the request to local server unit 102. Local server unit 102 may retrieve requested data from cache 104 and provide the data to DASH client 110 in response to such requests.

[0082] FIG. 3 is a conceptual diagram illustrating elements of example multimedia content 120. Multimedia content 120 may correspond to multimedia content 64 (FIG. 1), or another multimedia content stored in storage medium 62. In the example of FIG. 3, multimedia content 120 includes media presentation description (MPD) 122 and a plurality of representations 124A–124N (representations 124). Representation 124A includes optional header data 126 and segments 128A–128N (segments 128), while representation 124N includes optional header data 130 and segments 132A–132N (segments 132). The letter N is used to designate the last movie fragment in each of representations 124 as a matter of convenience. In some examples, there may be different numbers of movie fragments between representations 124.

[0083] MPD 122 may comprise a data structure separate from representations 124. MPD 122 may correspond to manifest file 66 of FIG. 1. Likewise, representations 124

may correspond to representations 68 of FIG. 2. In general, MPD 122 may include data that generally describes characteristics of representations 124, such as coding and rendering characteristics, adaptation sets, a profile to which MPD 122 corresponds, text type information, camera angle information, rating information, trick mode information (e.g., information indicative of representations that include temporal sub-sequences), and/or information for retrieving remote periods (e.g., for targeted advertisement insertion into media content during playback).

[0084] Header data 126, when present, may describe characteristics of segments 128, e.g., temporal locations of random access points (RAPs, also referred to as stream access points (SAPs)), which of segments 128 includes random access points, byte offsets to random access points within segments 128, uniform resource locators (URLs) of segments 128, or other aspects of segments 128. Header data 130, when present, may describe similar characteristics for segments 132. Additionally or alternatively, such characteristics may be fully included within MPD 122.

[0085] Segments 128, 124 include one or more coded video samples, each of which may include frames or slices of video data. Each of the coded video samples of segments 128 may have similar characteristics, e.g., height, width, and bandwidth requirements. Such characteristics may be described by data of MPD 122, though such data is not illustrated in the example of FIG. 3. MPD 122 may include characteristics as described by the 3GPP Specification, with the addition of any or all of the signaled information described in this disclosure.

[0086] Each of segments 128, 132 may be associated with a unique uniform resource locator (URL). Thus, each of segments 128, 132 may be independently retrievable using a streaming network protocol, such as DASH. In this manner, a destination device, such as client device 40, may use an HTTP GET request to retrieve segments 128 or 132. In some examples, client device 40 may use HTTP partial GET requests to retrieve specific byte ranges of segments 128 or 132.

[0087] FIG. 4 is a block diagram illustrating elements of an example video file 150, which may correspond to a segment of a representation, such as one of segments 114, 124 of FIG. 3. Each of segments 128, 132 may include data that conforms substantially to the arrangement of data illustrated in the example of FIG. 4. Video file 150 may be said to encapsulate a segment. As described above, video files in accordance with the ISO base media file format and extensions thereof store data in a series of objects, referred to as "boxes." In the example of FIG. 4, video file 150 includes file type

(FTYP) box 152, movie (MOOV) box 154, segment index (sidx) boxes 162, movie fragment (MOOF) boxes 164, and movie fragment random access (MFRA) box 166. Although FIG. 4 represents an example of a video file, it should be understood that other media files may include other types of media data (e.g., audio data, timed text data, or the like) that is structured similarly to the data of video file 150, in accordance with the ISO base media file format and its extensions.

[0088] File type (FTYP) box 152 generally describes a file type for video file 150. File type box 152 may include data that identifies a specification that describes a best use for video file 150. File type box 152 may alternatively be placed before MOOV box 154, movie fragment boxes 164, and/or MFRA box 166.

[0089] In some examples, a Segment, such as video file 150, may include an MPD update box (not shown) before FTYP box 152. The MPD update box may include information indicating that an MPD corresponding to a representation including video file 150 is to be updated, along with information for updating the MPD. For example, the MPD update box may provide a URI or URL for a resource to be used to update the MPD. As another example, the MPD update box may include data for updating the MPD. In some examples, the MPD update box may immediately follow a segment type (STYP) box (not shown) of video file 150, where the STYP box may define a segment type for video file 150. FIG 7, discussed in greater detail below, provides additional information with respect to the MPD update box.

[0090] MOOV box 154, in the example of FIG. 4, includes movie header (MVHD) box 156, track (TRAK) box 158, and one or more movie extends (MVEX) boxes 160. In general, MVHD box 156 may describe general characteristics of video file 150. For example, MVHD box 156 may include data that describes when video file 150 was originally created, when video file 150 was last modified, a timescale for video file 150, a duration of playback for video file 150, or other data that generally describes video file 150.

[0091] TRAK box 158 may include data for a track of video file 150. TRAK box 158 may include a track header (TKHD) box that describes characteristics of the track corresponding to TRAK box 158. In some examples, TRAK box 158 may include coded video pictures, while in other examples, the coded video pictures of the track may be included in movie fragments 164, which may be referenced by data of TRAK box 158 and/or sidx boxes 162.

[0092] In some examples, video file 150 may include more than one track. Accordingly, MOOV box 154 may include a number of TRAK boxes equal to the number of tracks in video file 150. TRAK box 158 may describe characteristics of a corresponding track of video file 150. For example, TRAK box 158 may describe temporal and/or spatial information for the corresponding track. A TRAK box similar to TRAK box 158 of MOOV box 154 may describe characteristics of a parameter set track, when encapsulation unit 30 (FIG. 3) includes a parameter set track in a video file, such as video file 150. Encapsulation unit 30 may signal the presence of sequence level SEI messages in the parameter set track within the TRAK box describing the parameter set track.

[0093] MVEX boxes 160 may describe characteristics of corresponding movie fragments 164, e.g., to signal that video file 150 includes movie fragments 164, in addition to video data included within MOOV box 154, if any. In the context of streaming video data, coded video pictures may be included in movie fragments 164 rather than in MOOV box 154. Accordingly, all coded video samples may be included in movie fragments 164, rather than in MOOV box 154.

[0094] MOOV box 154 may include a number of MVEX boxes 160 equal to the number of movie fragments 164 in video file 150. Each of MVEX boxes 160 may describe characteristics of a corresponding one of movie fragments 164. For example, each MVEX box may include a movie extends header box (MEHD) box that describes a temporal duration for the corresponding one of movie fragments 164.

[0095] As noted above, encapsulation unit 30 may store a sequence data set in a video sample that does not include actual coded video data. A video sample may generally correspond to an access unit, which is a representation of a coded picture at a specific time instance. In the context of AVC, the coded picture include one or more VCL NAL units which contains the information to construct all the pixels of the access unit and other associated non-VCL NAL units, such as SEI messages. Accordingly, encapsulation unit 30 may include a sequence data set, which may include sequence level SEI messages, in one of movie fragments 164. Encapsulation unit 30 may further signal the presence of a sequence data set and/or sequence level SEI messages as being present in one of movie fragments 164 within the one of MVEX boxes 160 corresponding to the one of movie fragments 164.

[0096] SIDX boxes 162 are optional elements of video file 150. That is, video files conforming to the 3GPP file format, or other such file formats, do not necessarily

include SIDX boxes 162. In accordance with the example of the 3GPP file format, a SIDX box may be used to identify a sub-segment of a segment (e.g., a segment contained within video file 150). The 3GPP file format defines a sub-segment as "a self-contained set of one or more consecutive movie fragment boxes with corresponding Media Data box(es) and a Media Data Box containing data referenced by a Movie Fragment Box must follow that Movie Fragment box and precede the next Movie Fragment box containing information about the same track." The 3GPP file format also indicates that a SIDX box "contains a sequence of references to subsegments of the (sub)segment documented by the box. The referenced subsegments are contiguous in presentation time. Similarly, the bytes referred to by a Segment Index box are always contiguous within the segment. The referenced size gives the count of the number of bytes in the material referenced."

[0097] SIDX boxes 162 generally provide information representative of one or more sub-segments of a segment included in video file 150. For instance, such information may include playback times at which sub-segments begin and/or end, byte offsets for the sub-segments, whether the sub-segments include (e.g., start with) a stream access point (SAP), a type for the SAP (e.g., whether the SAP is an instantaneous decoder refresh (IDR) picture, a clean random access (CRA) picture, a broken link access (BLA) picture, or the like), a position of the SAP (in terms of playback time and/or byte offset) in the sub-segment, and the like.

[0098] Movie fragments 164 may include one or more coded video pictures. In some examples, movie fragments 164 may include one or more groups of pictures (GOPs), each of which may include a number of coded video pictures, e.g., frames or pictures. In addition, as described above, movie fragments 164 may include sequence data sets in some examples. Each of movie fragments 164 may include a movie fragment header box (MFHD, not shown in FIG. 4). The MFHD box may describe characteristics of the corresponding movie fragment, such as a sequence number for the movie fragment. Movie fragments 164 may be included in order of sequence number in video file 150.

[0099] MFRA box 166 may describe random access points within movie fragments 164 of video file 150. This may assist with performing trick modes, such as performing seeks to particular temporal locations (i.e., playback times) within a segment encapsulated by video file 150. MFRA box 166 is generally optional and need not be included in video files, in some examples. Likewise, a client device, such as client device 40, does not necessarily need to reference MFRA box 166 to correctly decode

and display video data of video file 150. MFRA box 166 may include a number of track fragment random access (TFRA) boxes (not shown) equal to the number of tracks of video file 150, or in some examples, equal to the number of media tracks (e.g., non-hint tracks) of video file 150.

[0100] In some examples, movie fragments 164 may include one or more stream access points (SAPs), such as IDR pictures. Likewise, MFRA box 166 may provide indications of locations within video file 150 of the SAPs. Accordingly, a temporal sub-sequence of video file 150 may be formed from SAPs of video file 150. The temporal sub-sequence may also include other pictures, such as P-frames and/or B-frames that depend from SAPs. Frames and/or slices of the temporal sub-sequence may be arranged within the segments such that frames/slices of the temporal sub-sequence that depend on other frames/slices of the sub-sequence can be properly decoded. For example, in the hierarchical arrangement of data, data used for prediction for other data may also be included in the temporal sub-sequence.

[0101] FIG. 5 is a conceptual diagram illustrating a common understanding of HTTP adaptive streaming. In this example, video is encoded at multiple bitrates to form multiple videos. Each of the videos is then split into small segments. Each segment is encrypted, and made available via an HTTP URL. A client device determines which segment(s) to download, acquires a license for the encrypted content, and then splices together and plays back the content.

[0102] FIG. 6 is a conceptual diagram illustrating responsibilities of "smart" client devices. In this example, a client manages one or more manifest files (e.g., one or more MPDs), HTTP transport, and one or more TCP connections. The client device monitors or measures a playout buffer, download times and throughput of segments, local resources (such as a CPU, memory, screen, and the like), and dropped frames. The client also performs bandwidth adaptation.

[0103] FIG. 7 is a conceptual diagram illustrating the scope of MPEG DASH. In particular, FIG. 7 illustrates a system including an HTTP server and a DASH client, and various elements thereof. Elements of the system within the scope of MPEG DASH are highlighted using grey shading. In this example, MPEG DASH has scope including the MPD, segments, MPD parser, and segment parser.

[0104] FIG. 8 is a conceptual diagram illustrating an example architecture according to the techniques of this disclosure. As shown in the example of FIG. 8, the architecture

shown in FIG. 8 further includes a DASH Aware Network Element (DANE), coupled to the DASH client via lower layer transport.

[0105] FIG. 8 provides a high-level architecture of the considered message flow. DASH media is stored on an HTTP server and delivered through an HTTP-CDN. The DASH client controls the session and issues HTTP requests at appropriate time. An intermediate node that terminates the HTTP connection may act as a cache if DASH unaware or as a DANE, if DASH aware. A DANE may receive information from the Server and/or the Client in order to optimize the delivery, for example, by providing such information to the lower layer transport that makes use of the information in its delivery decisions.

[0106] FIG. 9 is a conceptual diagram illustrating a simple client model. In this example, a client includes a download engine, a buffer, and a media decoder and renderer.

[0107] Following the descriptions in DASH-IF IOP v3.1, section 4.3.4, the DASH client acts as follows. A DASH client is guided by the information provided in the MPD. The simple client model is shown in FIG. 9.

[0108] Assume that the client has access to an MPD and can derive the segment availability times for each segment. For simplicity, it is assumed that the MPD only contains a single Period with period start time $PSwc[i]$ and the MPD-URL does not include any fragment parameters. The following example client behavior provides a continuous streaming experience to the user:

1. The client parses the MPD, selects a collection of Adaptation Sets suitable for its environment based on information provided in each of the **AdaptationSet** elements.
2. Within each Adaptation Set it selects one Representation, typically based on the value of the @bandwidth attribute, but also taking into account client decoding and rendering capabilities.
3. The client creates a list of accessible Segments at least for each selected Representation taking into account the information in the MPD and the current time *JOIN* in the client and in particular the segment closest to the live edge referred to as the *live edge segment*.
4. The client downloads the initialization segment of the selected Representations and then accesses the content by requesting entire Segments or byte ranges of

Segments. Typically at any time the client downloads the next segment at the larger of the two: (i) completion of download of current segment or (ii) the Segment Availability Start Time of the next segment. Note that if the @availabilityTimeOffset is present, then the segments may be downloaded earlier, namely at the adjusted segment availability start time. Based on the buffer fullness and other criteria, rate adaptation is considered. Typically the first media segment that is downloaded is the *live edge segment*, but other decisions may be taken in order to minimize start-up latency.

5. According to FIG. 9, media is fed into buffer and at some point in time, the decoding and rendering of the media is kicked off. The downloading and presentation is done for the selected Representation of each selected Adaptation. The synchronization is done using the presentation time in the Period. For synchronized playout, the exact presentation times in the media shall be used.
    a. Once presentation has started, the playout process is continuous. The playout process expects media to be present in the buffer continuously. If the **MPD**@suggestedPresentationDelay is present, then this value may be used as the presentation delay PD. If the **MPD**@suggestedPresentationDelay is not present, but the client is expected to consume the service at the live edge, then a suitable presentation delay should be selected, typically between the value of @minBufferTime and the value of @timeShiftBufferDepth. It is recommended that the client starts rendering the first sample of the downloaded media segment k with earliest presentation time EPT(k) at $PSwc[i] + (\text{EPT}(k) - o[r,i]) + PD$.
6. The client may request Media Segments of the selected Representations by using the generated Segment list during the availability time window.
7. Once the presentation has started, the client continues consuming the media content by continuously requesting Media Segments or parts of Media Segments and playing content according to the media presentation timeline. The client may switch Representations taking into updated information from its environment, but this aspect is of less relevance for the discussion in this document.
8. With the wall-clock time *NOW* advancing, the client consumes the available Segments. As *NOW* advances the client possibly expands the list of *available* Segments for each Representation in the Period.

[0109] Certain aspects are summarized in bullets 5–7 above. The client controls the scheduling of a request and the playout scheduling. At the same time, the DASH client has knowledge on the latest time the next segment of a Representation needs to be available to avoid buffer underflow. The information is available at least on a good accuracy from the MPD (using the mapping of the Period timeline to the media timeline), but on an even better accuracy once the previous Segment is downloaded. More details are shown in FIG. 10 below.

[0110] FIG. 10 is a conceptual diagram illustrating segments from the perspective of a server and a client. The server makes data available and leaves data available until a certain time. The DASH client determines this information (i.e., the times of availability) from the MPD. Once the MPD is downloaded the first time `FetchMPD(1)` the DASH client is aware that segment 1 and 2 are available, over time more and more segments get available based on the information provided in the MPD. At some point in time after some presentation delay (using the suggested presentation delay or a client selected one), the DASH client schedules the playout and starts. Once started, the client knows, based on the MPD and when a segment is requested, when the segment must be available in the DASH client in order to ensure smooth playout. At the same time, the DASH client would know when there is the latest time that the segment is still available on the server.

[0111] A DASH client may generate this information and send it back to the network in order for the network to decide on how to schedule the request – more examples are provided. Furthermore, DASH clients may be on a relatively good network time protocol (NTP) sync clock in certain circumstances. In other circumstances, the synchronization may be much looser. Additionally, availability times and scheduling are also relevant for static type content, e.g., on-demand content.

[0112] Various use cases are contemplated for using deadline information. Some examples for using such deadline information include:

- A DANE is collocated with a mobile base station (together with a PGW), for example an eNB. The information for a deadline may be used by the radio scheduler to optimize the delivery in the corresponding cell using TCP/IP.
- A DANE is collocated with a home gateway. The information for a deadline, possibly from different users, may be used to optimize the requests towards network in order to ensure timely delivery of urgent objects.

- A DANE is collocated with a mobile base station (together with a PGW), for example an eNB. The information for a deadline may be used by the radio scheduler to optimize the delivery in the corresponding cell using a different delivery protocol than TCP, for example a packet-based protocol.

[0113] In creating a relevant solution, the following criteria are considered:

- The solution may be simple.
- The solution may preferably work independent of DASH.
- The solution may be part of the SAND status message framework.
- The solution may be implementable for a DASH client.
- The solution may enable provision of the deadline information as an absolute or a relative value.
- The solution may work with regular segment requests as well as byte range requests (e.g., HTTP GET or partial GET requests).
- The solution may be an optimization and work in a backward-compatible manner.

[0114] The techniques of this disclosure are premised on the basic idea that the DASH client is fully aware of timing information. That is, the DASH client may determine from the manifest file (e.g., the MPD) the following information in wall clock time: when segments are available on the network and the time when each segment needs to be available at the receiver in order to be able to continue smooth playout. If the DASH client provides some of this information to the DANE, the DANE may then optimize the delivery in order to ensure that the Segment is available at the receiver in time. Although for purposes of explanation the techniques are described with respect to a DASH client and a DANE, these techniques can be performed by other streaming devices (e.g., any other streaming client and a streaming aware network element).

[0115] User Equipment (UE) (that is, a client device) may report the following information to a streaming aware network element:

- HTTP Streaming Playout Buffer Status
    - Buffer Level (ms)
    - Timestamp: The time when UE generates the buffer status information
    - Playout data rate: for eNodeB (eNB) to estimate the buffer level of UE, if eNB is not DASH aware
- Deadline in wall-clock time or maximum RTT for the entire segment (object)

- UE specifies this time in each HTTP GET request

- More detailed information, for example it may be that the initial part of the segment has other deadlines than the later part
  - Client reports deadline of initial packet
  - Object contains playout schedule for each byte range, such that intermediate node can schedule delivery of each byte range according to the information from DASH client and the playout curve
- Also report the reception mode: full segment or media delivery event (MDE)

[0116] The client device may additionally or alternatively report playout information.

- Playout schedule may be:
  - Defaulted to a step function, i.e. the entire segment needs to be available
  - Reported as auxiliary information from the media/DASH server as a playout curve (byte range over time)
  - Send back from the client if the client does have such information (e.g. by the segment index) to the DANE which can be used in scheduling.
- Such information may be used by the scheduler to optimize the delivery of the byte ranges

[0117] FIG. 11 is a conceptual diagram illustrating one example for video object deadline aware scheduling. In this example, the client device delivers deadline information via HTTP to a DASH server. In particular, the UE reports deadline information to the DASH server via HTTP. The DASH server then sends the deadline information to a packet gateway (PGW) of a DANE.

[0118] In one example, there is a tunnel between the PGW and the DASH Server/proxy. The deadline information may be carried over a header of the tunnel protocol.

[0119] In another example, the PGW may perform selective deep packet inspection (DPI) on packets received from the DASH server. That is, the HTTP Server/Proxy includes deadline information in the HTTP message. The PGW performs DPI for the packets from the DASH Server to retrieve the deadline information.

[0120] The PGW may include deadline information in a GPRS Tunneling Protocol-U (GTP-U) header of each downlink packet sent to the eNB (that is, the base station). The base station then schedules transmission of these packets per the deadline information.

[0121] In another example, the UE may report the deadline information to the DANE via HTTP. The DANE may also interpret the playout curve of each Segment. The

DANE may then make this information available to the scheduler of the eNB. The eNB may then use this information to optimize playout scheduling (e.g., packet delivery).

[0122] In HTTP based deadline information delivery, information from the streaming client to the DANE may include deadline information. The deadline information may include syntax and protocol options. These options may be included in an HTTP header extension and/or as part of query parameters in a request. Additionally or alternatively, the deadline information may be mapped to existing deployments, e.g., to HTTP/1.1 based delivery or HTTP/2.0 based delivery. Information from the Server to the DANE and usage of the deadline information in this example may include a Real-Time Object Delivery over Unidirectional Transport (ROUTE) transport protocol and MDE.

[0123] In various examples, information exchanged between these various devices may define solutions to various issues. For example, semantic issues of what information ins sent from the client to the network, which may include: deadline in wall-clock time for the entire segment (object), maximum RTT in milliseconds for the entire segment (object), currently available buffer in the client, and/or more detailed information (for example, it may be that the initial part of the segment has other deadlines than the later part). Similarly, syntax and protocol options may be exchanged in an HTTP Header extension in a request, as part of a query parameters in the request, or other control channels between the DASH client and the DANE. Furthermore, there may be a mapping of such information to existing deployments, such as HTTP/1.1 based delivery, HTTP/2.0 based delivery, or ROUTE transport protocol and MDE.

[0124] FIG. 12 is a conceptual diagram of an example implementation according to the example of FIG. 11. In this example, there is an HTTP Cache above the gateway (GW). Information may define an interface between the HTTP Proxy and the PGW. Deadline based scheduling techniques may be used for deadline reporting in HTTP from the UE to the HTTP Proxy. Deadline information may be exchanged over the radio access network (RAN) from the HTTP proxy to the PGW in GTP-U headers to the eNB.

[0125] FIG. 13 is a conceptual diagram illustrating another example solution based on video object deadline aware scheduling. In this example, the UE delivers DANE specific deadline information. More particularly, the UE reports the deadline information to the eNB via RAN signaling, e.g., using packet data convergence protocol (PDCP) or radio resource control (RRC) protocol. The DASH client may report the deadline information to a modem via an application programming interface (API).

[0126] The eNB may then schedule transmissions for the HTTP streaming session per the deadline information.

[0127] Deadline information may be associated to the downlink (DL) packets in various ways. In one example, the HTTP proxy may be logically integrated into the eNB. In another example, a dedicated bearer may be established for the HTTP streaming session.

[0128] FIG. 14 is a conceptual diagram illustrating an example implementation according to the techniques of FIG. 13. In this example, HTTP caching is performed by the eNB, referred to as RAN caching. The eNB may logically cache the content. Physically, the HTTP Proxy and the Cache may be outside of the eNB (per FIG. 15 below). If not physically cached locally, the eNB fetches the content from the Internet content delivery network (CDN) or content server via the PGW. This avoids exposing the eNB to the Internet directly, and also ensures service continuity by HTTP. Mobile network operators may also push content to the eNB based on predictions. Deadline information may be reported to the eNB via PDCP, HTTP, or RRC.

[0129] FIG. 15 is a conceptual diagram illustrating an example of packetization in accordance with the techniques of this disclosure. In this example, playout curve metadata can be used by the sender to packetize and schedule data properly.

[0130] FIG. 16 is a conceptual diagram illustrating a series of segments, including an initialization segment (IS) and a plurality of media segments (MSs). The IS: contains all relevant metadata as well the media data, for one component; provides a random access point on File Format level; and provides a size in bytes, an earliest presentation time (within ISO BMFF timeline relative to start of period) in the file format as well as a duration, i.e., the duration of samples it spans (within a "Period/MPU"). The in-order-concatenation of segments results in a conforming bitstream for the ISO BMFF.

[0131] FIGS. 17A and 17B are conceptual diagrams representing a playout curve for a media segment. The playout curve expresses the amount of bytes necessary to present up to a certain time. The shape of the curve is determined by the media encoder/preparation. Examples include sample boundaries and linear increase in time. The complete segment is necessary for playout. The playout curve can be used by the transport for delivery. More metadata may be added or used for transport.

[0132] In some examples, for proper playout, an entire Segment needs to be available at the receiver. Such a model is appropriate for certain receiver implementations, but in many cases, Segments can be played "progressively," i.e., by playing and downloading at the same time. Such features are particularly relevant for low delay scenarios. The

information may be provided from the client to the DANE as an extension to the simple deadline status message or it may be provided from the DASH server to the DANE as a PED.

[0133] A Media Segment may be subdivided into playable prefixes. Assuming an earliest presentation time in a segment, the playout curve expresses the amount of bytes necessary to present up to a certain time. In the "worst case," all bytes are necessary to play the earliest presentation time. In this case, only once the entire segment is available the data may be played. However, typically, with only a prefix of the entire segment, the segment can start being presented, and with continuously more bytes being added to the prefix, more and more presentation time can be played. The shape of the playout curve is determined by the media encoder/preparation. Some examples include:

- Sample boundaries and linear increase in time (e.g., as shown in FIG. 17A)
- Complete segment is necessary for playout
- More complex structures for decoding and presentation orders are not identical, for example, when hierarchical B-pictures are used.

[0134] Two different playout curves are shown in FIG. 17B. The x-axis shows the presentation time (PT) minus the earliest presentation time (EPT) of the segment. To play the EPT, a certain amount of bytes are necessary. Whereas the dashed curve requires all bytes of the segment to play the samples with EPT, the solid curve only needs a small portion to start playing and some more data follows. The playout curve is a typical stair case.

[0135] If the network is aware of the curve, it can optimize the delivery of the related byte ranges in order to ensure timely reception of the data, assuming that the network also knows when the client needs to receive the EPT. This can be accomplished by the techniques discussed above, e.g., with respect to FIGS. 9 and 10.

[0136] FIG. 18 is a conceptual diagram of media delivery events (MDEs) for delivering data of segments. MDEs starting with a random access point (RAP) that are encapsulated in IP/UDP/ROUTE become T-MDEs starting with T-RAP.

[0137] FIG. 19 is a conceptual diagram illustrating another example that may be used as a potential simplification. In this example, the lower-layer signaling provides sufficient information to start the service without the MPD. Only if unicast is added or if richer selection is necessary, the MPD is consulted. There is still a unified MPD, but the MPD is not necessary for startup and/or broadcast-only.

[0138] FIG. 20 is a conceptual diagram of an example delivery architecture in accordance with the techniques of this disclosure.

[0139] Various examples of use cases for using progressive playout are discussed below:

- A DANE is collocated with a mobile base station (together with a PGW), for example an eNB. The information for the progressive playout may be used by the radio scheduler to optimize the delivery in the corresponding cell using TCP/IP.
- A DANE is collocated with a home gateway. The information for a progressive playout together with the deadline, possibly from different users, may be used to optimize the requests towards network in order to ensure timely delivery of urgent pieces of certain objects.
- A DANE is collocated with a mobile base station (together with a PGW), for example an eNB. The information for a deadline and progressive playout may be used by the radio scheduler to optimize the delivery in the corresponding cell using a different delivery protocol than TCP, for example a packet-based protocol. FIG. 20 shows a potential delivery architecture for such a case. The Segmenter and packetizer, in this case using ROUTE as defined in ATSC, uses the information from the sender (progressive playout), and possibly from the receiver (deadline), in order to add Target Times for the packets to be sent to/received by the receiver. This allows the network to use this information to ensure no buffer underrun.

[0140] FIG. 21 is a conceptual diagram of a receiver model in accordance with the techniques of this disclosure.

[0141] In order to address the above use cases and scenarios, the following extensions are proposed:

- A status message to provide an absolute deadline (wall-clock) for the requested object in the receiver.
- A status message to provide a maximum RTT (duration) for the requested object.
- A PED message to provide the relative deadline of the different byte ranges of the segment.
- A status message to provide the relative deadline of the different byte ranges of the segment.

[0142] An AbsoluteDeadline parameter may be specified in deadline information. This parameter may allow DASH clients to indicate to the DANE cache the absolute deadline in wall-clock time by when the Segment needs to be received.

[0143] Source and destination for the AbsoluteDeadline parameter may be as follows:

- Type : Metrics
- Sender : DASH client
- Receiver : DANE

[0144] The table below represents an example data representation for the AbsoluteDeadline parameter:

Parameter	Type	Cardinality	Description
`AbsoluteDeadline`	NTP	1	Absolute deadline for the segment to be available in the receiver.

[0145] For the format of the absolute time, basically any of the following formats may be used. Example for formats of absolute time include:

- xsdate format
- ISO timing format
- NTP format

[0146] An example way to define an extension header in the request is as follows:

X-Dash-Deadline-ISO: <time in ISO format>

[0147] An example is as follows:

X-Dash-Deadline-ISO: 2015-10-11T17:53:03Z

[0148] Additionally or alternatively, a maximum round trip time (MaxRTT) parameter may be specified in the deadline information. This parameter may allow DASH clients to indicate to the DANE cache the maximum round trip time of the request from the time when the request was issued until the requested data needs to be completely available at the DASH client. The time may be expressed in ms.

[0149] Source and destination for the MaxRTT parameter may be as follows:

- Type : Metrics
- Sender : DASH client
- Receiver : DANE

[0150] The table below represents an example data representation for the MaxRTT parameter:

Parameter	Type	Cardinality	Description

MaxRTT	Uint	1	Maximum RTT from the request until the Segments is available.

[0151] An example way to define an extension header in the request is as follows:

X-Dash-MaxRTT: <maximum Round Trip time in ms>

[0152] An example is as follows:

X-Dash-MaxRTT: 2345

[0153] This example represents that in order to avoid buffer underflow, the client requests the availability of the segment 2.345 seconds after issuing the request.

[0154] Additionally or alternatively, the deadline information may include a progressive playout profile metric (ProgressivePlayout) parameter. This parameter may allow DASH clients to indicate to the DANE cache the progressive playout profile of a segment.

[0155] Source and destination for the ProgressivePlayout parameter may be as follows:

- Type : Metrics
- Sender : DASH client
- Receiver : DANE

[0156] The table below represents an example data representation for the ProgressivePlayout parameter:

Parameter	Type	Cardinality	Description
ProgressivePlayout	Array	1	A list of tuples, each expressing the playout time and necessary bytes relative to the earliest presentation time in ms.

[0157] An example way to define an extension header in the request is as follows:

X-Dash-Progressive-Playout: <tuples of bytes and times in ms>

[0158] An example is as follows:

X-Dash-Progressive-Playout:
<4321,0;62345,200;82220,400;1010101,600;121212,800;1313131;1000>

[0159] This example represents that 4321 bytes need to be delivered to initiate playout and then the total amount of bytes for each of the playout times. Note that the steps may also be expressed as Deltas.

[0160] The ProgressivePlayout information may additionally or alternatively be delivered as an extension header along with a response.

[0161] Signaling the deadline information in this manner may ensure timely delivery of data having real-time constraints, such as media data. For example, by providing the deadline information to the DANE, the DANE can ensure that data is delivered to the client device according to the real-time constraints. Thus, with respect to DASH or other streaming of media data, the client device may avoid a buffer underflow, which may ensure continuous, smooth playout.

[0162] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code, and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0163] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc

(DVD), floppy disk and Blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0164] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term "processor," as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0165] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0166] Various examples have been described. These and other examples are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method of retrieving data with real-time constraints, the method comprising, by a real-time application of a client device:

determining times during which the data will be available for download;

determining a time at which the data is needed to fulfill the real-time constraints; and

sending a request for the data and deadline information representative of the time at which the data is needed to fulfill the real-time constraints.

2. The method of claim 1, wherein determining the times during which the data will be available comprises determining the times from at least one of a manifest file for the data or previously received data.

3. The method of claim 2, wherein the manifest file comprises a Dynamic Adaptive Streaming over HTTP (DASH) Media Presentation Description (MPD).

4. The method of any of claims 1–3, wherein the request specifies the deadline information.

5. The method of any of claims 1–4, wherein the data comprises at least one of real-time media data, streaming media data, a media segment, or a media sub-segment.

6. The method of any of claims 1–5, wherein the real-time application comprises one of a streaming client or a Dynamic Adaptive Streaming over HTTP (DASH) client.

7. The method of any of claims 1–6, wherein the real-time constraints comprise a buffer underrun for a buffer of the client device.

8. The method of any of claims 1–7, wherein sending the request comprises sending one of an HTTP GET request or an HTTP partial GET request.

9. The method of any of claims 1–8, wherein sending the deadline information comprises sending the deadline information in an HTTP extension header of the request.

10. The method of any of claims 1–8, wherein sending the deadline information comprises sending the deadline information in a query parameter of a uniform resource locator (URL) of the request.

11. The method of any of claims 1–10, wherein sending the request comprises sending the request to a streaming aware network element.

12. The method of any of claims 1–10, wherein sending the request comprises sending the request to a DASH aware network element (DANE).

13. The method of any of claims 1–10, wherein sending the request comprises sending the request to a DASH server.

14. The method of any of claims 1–10, wherein sending the request comprises sending the request to a mobile cell site.

15. The method of any of claims 1–10, wherein sending the request comprises sending the request to a WiFi access point.

16. The method of any of claims 1–15, wherein the deadline information comprises information representative of at least one of a wall-clock time by which the data specified in the request needs to be received or a maximum round-trip time from issuing the request until the data specified in the request needs to be received.

17. The method of any of claims 1–16, further comprising sending at least one of a playout curve or a subsampled version of the playout curve with the deadline information.

18. The method of any of claims 1–17, further comprising receiving the data from the streaming aware network element at or before the time at which the data is needed in response to sending the deadline information.

19. The method of any of claims 1–18, wherein the deadline information is configured to cause a streaming aware network element to optimize delivery such that the data is available at the streaming client at or before the time at which the data is needed.

20. The method of any of claims 1–19, wherein sending the deadline information comprises sending data representative of a buffer level for a buffer of the client device, a timestamp representing a time when the buffer level information was generated, or a playout data rate representing a rate at which the media data is being played by the client device.

21. The method of any of claims 1–20, wherein the data comprises a media segment, and wherein sending the deadline information comprises sending data representative of whether the data is to be received as a full segment or as a plurality of media delivery events (MDEs).

22. The method of any of claims 1–21, wherein sending the deadline information comprises sending the deadline information to a streaming server via hypertext transfer protocol (HTTP) to cause the streaming server to forward the deadline information to a streaming aware network element.

23. The method of any of claims 1–21, wherein sending the deadline information comprises sending the deadline information to a streaming aware network element via a radio access network (RAN) directly.

24. The method of any of claims 1–23, wherein sending the deadline information comprises sending data defining a mapping of at least a portion of the deadline information to an HTTP version.

25. The method of any of claims 1–24, wherein the real-time application comprises a dynamic adaptive streaming over HTTP (DASH) client, and wherein sending the request comprises sending the request to a streaming aware network element or a DASH aware network element (DANE).

26. A client device for retrieving media data, the client device comprising one or more means for performing the method of any of claims 1–25.

27. The client device of claim 26, wherein the one or more means comprise:
a memory configured to store the media data; and
one or more processors.

28. A computer-readable storage medium having stored thereon instructions that, when executed, cause a processor to perform the method of any of claims 1–25.

29. A method of transporting data with real-time constraints, the method comprising, by a network element:

receiving deadline information representative of a time at which the data is needed by a client device to fulfill the real-time constraints;

retrieving the data; and

delivering the data to the client device such that the data is available to the client device at or before the time at which the data is needed.

30. The method of claim 29, wherein receiving the deadline information comprises receiving the deadline information from the client device via a radio access network (RAN).

31. The method of claim 29, wherein receiving the deadline information comprises receiving the deadline information from a streaming server that forwards the deadline information received from the client device to the network element.

32. The method of any of claims 29–31, wherein delivering the data comprises prioritizing delivery of the data based on the deadline information.

33. The method of any of claims 29–32, wherein the deadline information comprises data representative of a buffer level for a buffer of the client device, a timestamp representing a time when the buffer level information was generated, and a playout data rate representing a rate at which the media data is being played by the client device.

34. The method of any of claims 29–33, wherein the data comprises a segment, and wherein the deadline information comprises data representative of whether the segment is to be received as a full segment or as a plurality of media delivery events (MDEs).

35. The method of any of claims 29–34, wherein the client device comprises a dynamic adaptive streaming over HTTP (DASH) client.

36. The method of any of claims 29–35, wherein the real-time constraints comprise a buffer underrun for a buffer of the client device.

37. The method of any of claims 29–36, wherein receiving the deadline information comprises receiving a request for the data that specifies the deadline information.

38. The method of claim 37, wherein the request specifies the deadline information in an HTTP extension header of the request.

39. The method of claim 37, wherein the request specifies a uniform resource locator (URL) including a query parameter that specifies the deadline information.

40. The method of any of claims 29–39, wherein the network element comprises one of a DASH aware network element (DANE), a mobile cell site element, a WiFi access point, or a home network element.

41. A network element device for retrieving media data, the streaming aware network element device comprising one or more means for performing the method of any of claims 29–40.

42. The apparatus of claim 41, wherein the one or more means comprise:
a memory configured to store the media data and the deadline information; and
one or more processors.

43. A computer-readable storage medium having stored thereon instructions that, when executed, cause a processor of a streaming aware network element to perform the method of any of claims 29–40.

44. A system comprising the device of any of claims 26 and 27 and the device of any of claims 41 and 42.

45. A method of transporting media data, the method comprising:
determining, from a manifest file for media data, times during which a segment of the media data will be available;
determining a time at which the segment is needed to provide smooth playout using the segment; and
sending deadline information representative of the time at which the segment is needed to a streaming aware network element.

46. A method of transporting media data, the method comprising, by a streaming aware network element:

receiving deadline information representative of a time at which a segment is needed by a client device;

retrieving the segment; and

delivering the segment to the client device such that the segment is available to the client device at or before the time at which the segment is needed.

47. A client device for retrieving data with real-time constraints, the client device comprising:

means for determining times during which the data will be available for download;

means for determining a time at which the data is needed to fulfill the real-time constraints; and

means for sending a request for the data and deadline information representative of the time at which the data is needed to fulfill the real-time constraints.

48. A device for transporting media data, the device comprising:

means for determining, from a manifest file for media data, times during which a segment of the media data will be available;

means for determining a time at which the segment is needed to provide smooth playout using the segment; and

means for sending deadline information representative of the time at which the segment is needed to a streaming aware network element.

49. A network element for transporting data with real-time constraints, the network element comprising:

means for receiving deadline information representative of a time at which the data is needed by a client device to fulfill the real-time constraints;

means for retrieving the data; and

means for delivering the data to the client device such that the data is available to the client device at or before the time at which the data is needed.

50. A streaming aware network element device for transporting data with real-time constraints, the streaming aware network element comprising:

means for receiving deadline information representative of a time at which a segment is needed by a client device;

means for retrieving the segment; and

means for delivering the segment to the client device such that the segment is available to the client device at or before the time at which the segment is needed.

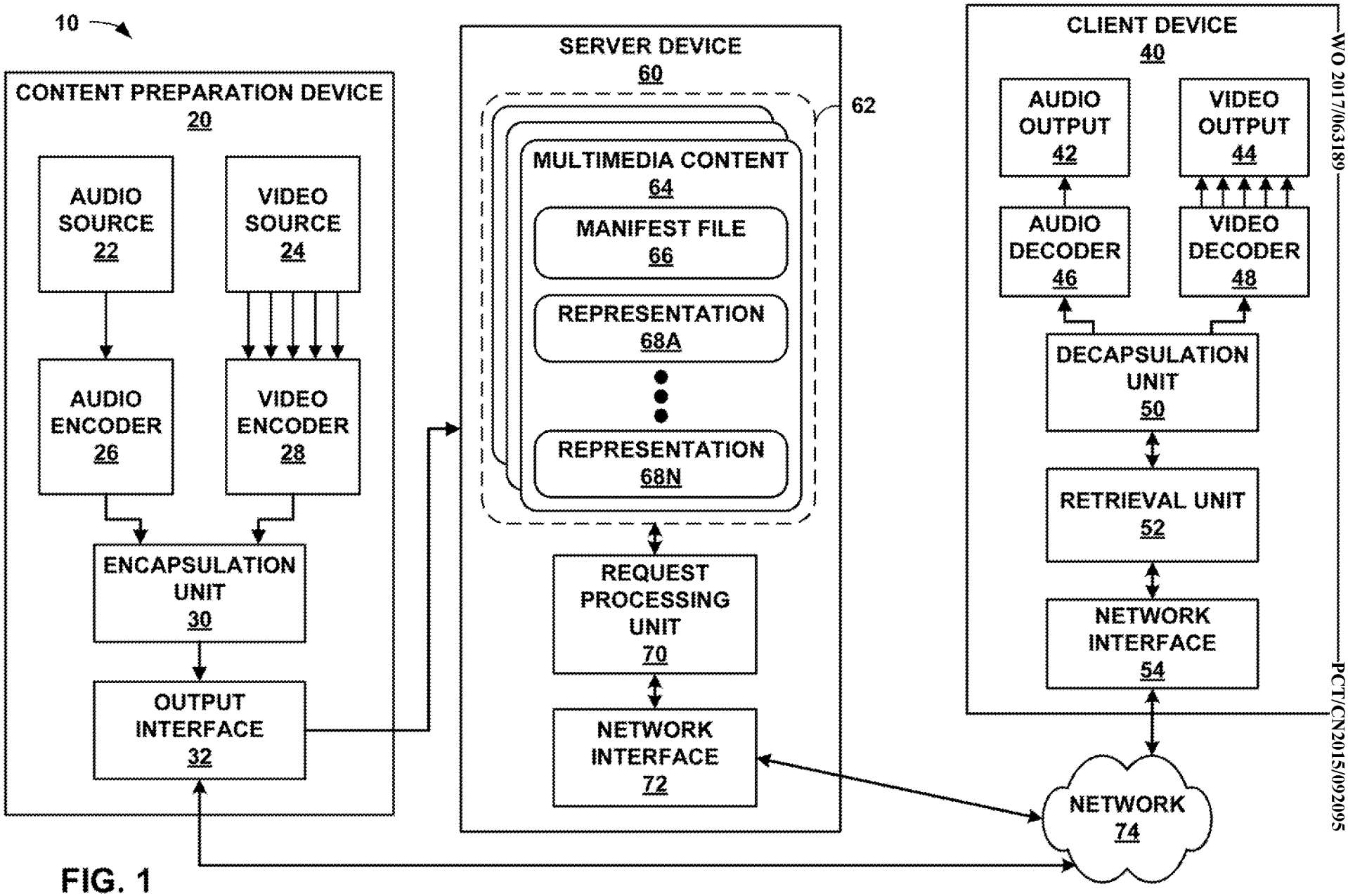

FIG. 1

WO 2017/063189
PCT/CN2015/092095

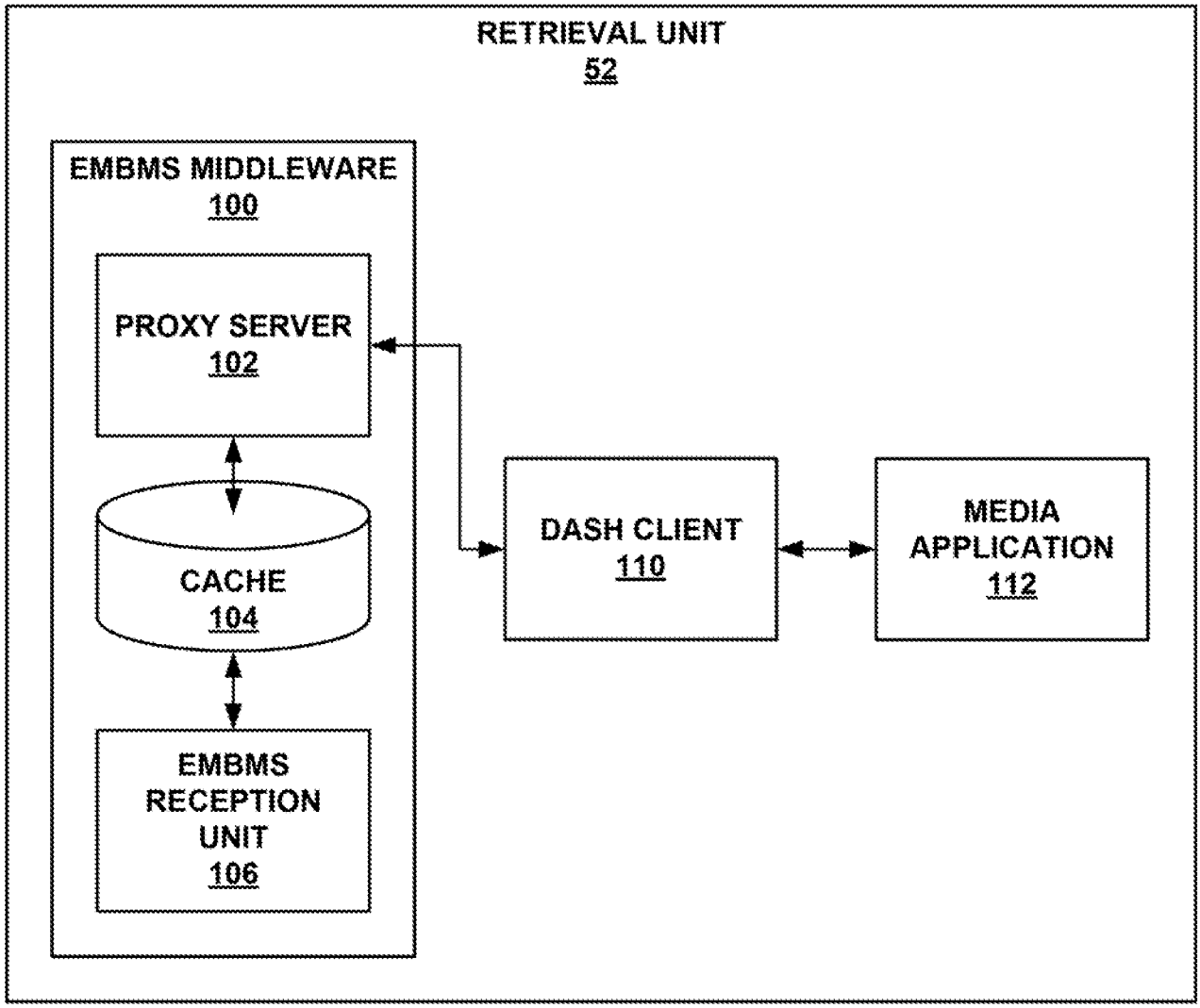
RETRIEVAL UNIT
52
EMBMS MIDDLEWARE
100
PROXY SERVER
102
CACHE
104
EMBMS
RECEPTION
UNIT
106
DASH CLIENT
110
MEDIA
APPLICATION
112

FIG. 2

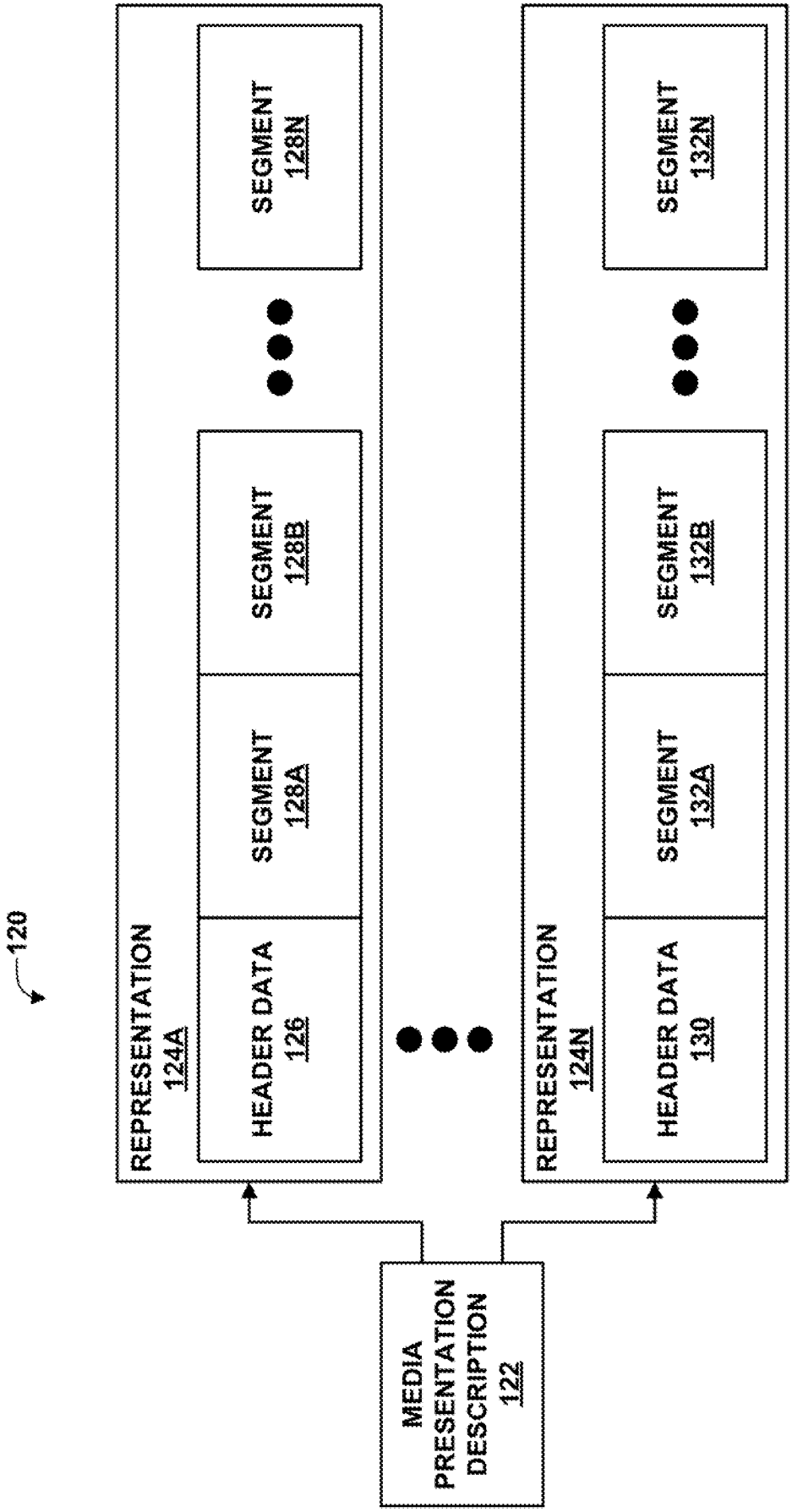
120
MEDIA PRESENTATION DESCRIPTION 122
REPRESENTATION 124A
HEADER DATA 126
SEGMENT 128A
SEGMENT 128B
SEGMENT 128N
REPRESENTATION 124N
HEADER DATA 130
SEGMENT 132A
SEGMENT 132B
SEGMENT 132N

FIG. 3

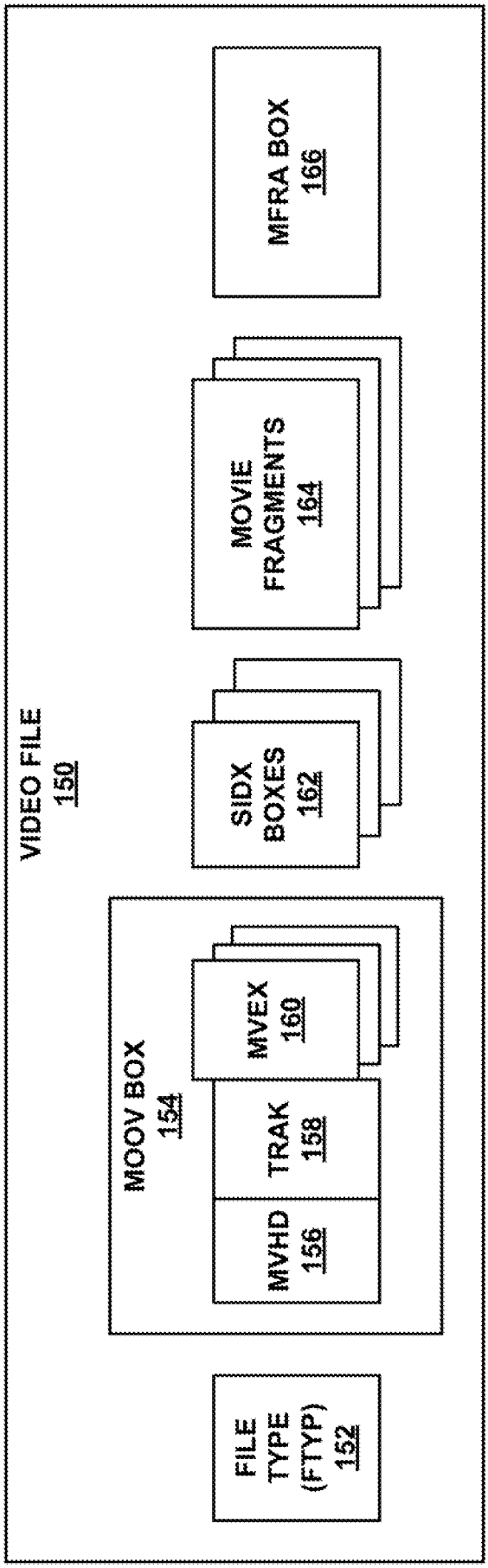
VIDEO FILE
150
FILE TYPE (FTYP)
152
MOOV BOX
154
MVHD
156
TRAK
158
MVEX
160
SIDX BOXES
162
MOVIE FRAGMENTS
164
MFRA BOX
166

FIG. 4

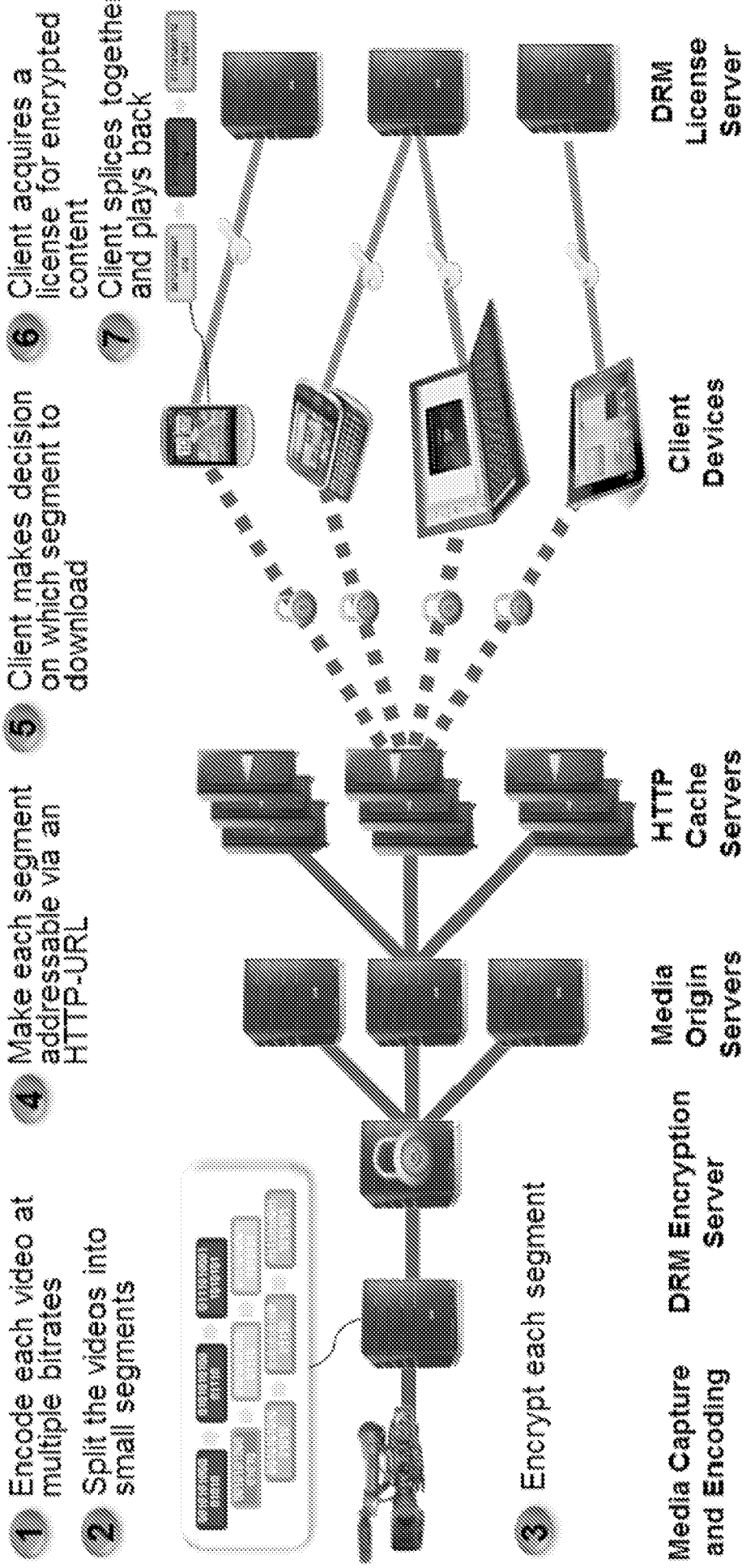
1 Encode each video at multiple bitrates
2 Split the videos into small segments
3 Encrypt each segment
4 Make each segment addressable via an HTTP-URL
5 Client makes decision on which segment to download
6 Client acquires a license for encrypted content
7 Client splices together and plays back
Media Capture and Encoding
DRM Encryption Server
Media Origin Servers
HTTP Cache Servers
Client Devices
DRM License Server

FIG. 5

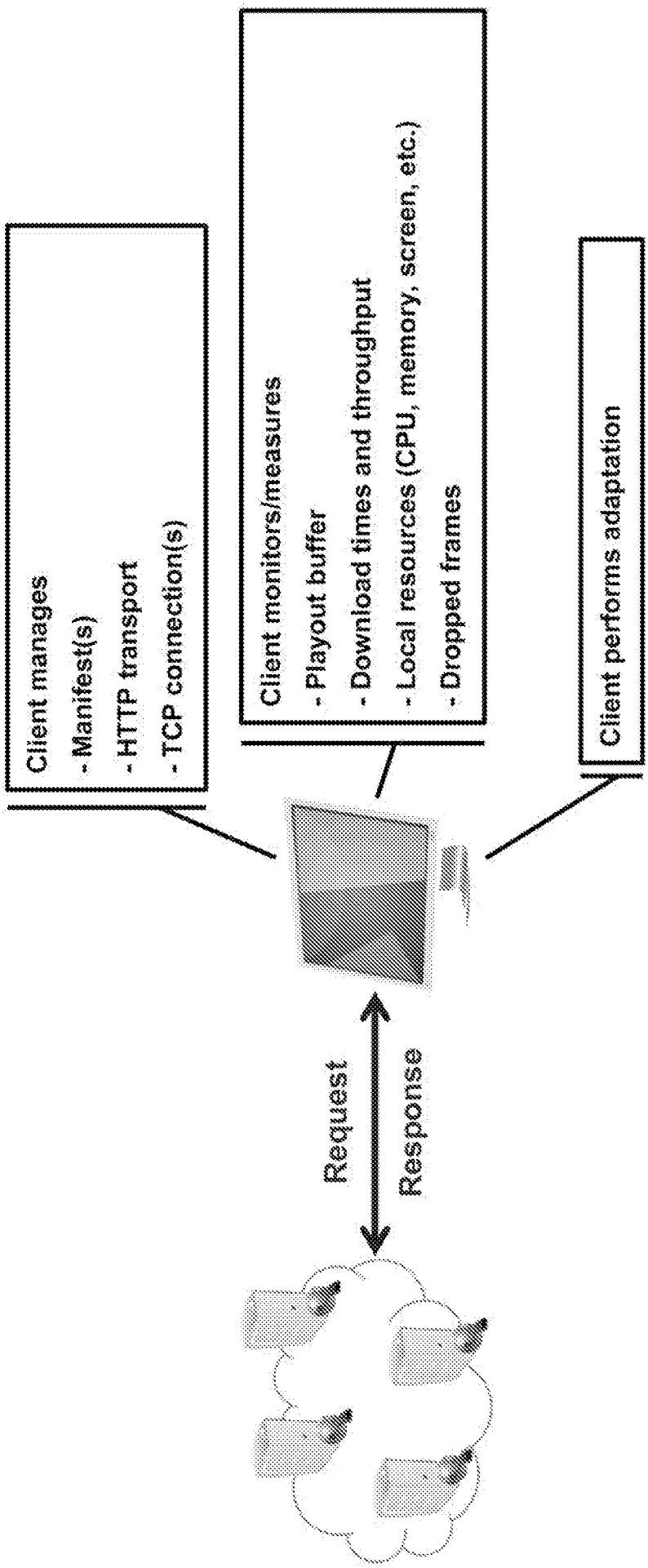
Client manages
- Manifest(s)
- HTTP transport
- TCP connection(s)
Client monitors/measures
- Playout buffer
- Download times and throughput
- Local resources (CPU, memory, screen, etc.)
- Dropped frames
Client performs adaptation
Request
Response

FIG. 6

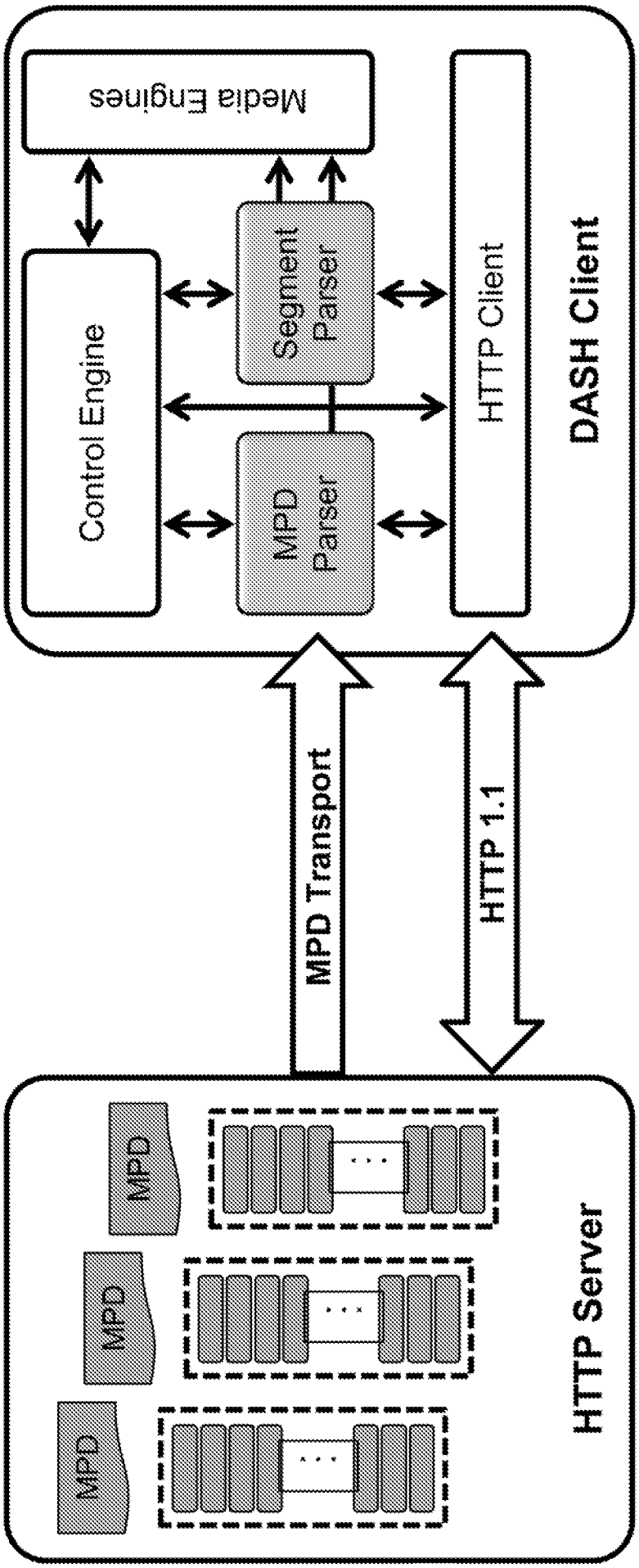
Media Engines
Control Engine
Segment Parser
MPD Parser
HTTP Client
DASH Client
MPD Transport
HTTP 1.1
MPD
MPD
MPD
HTTP Server

FIG. 7

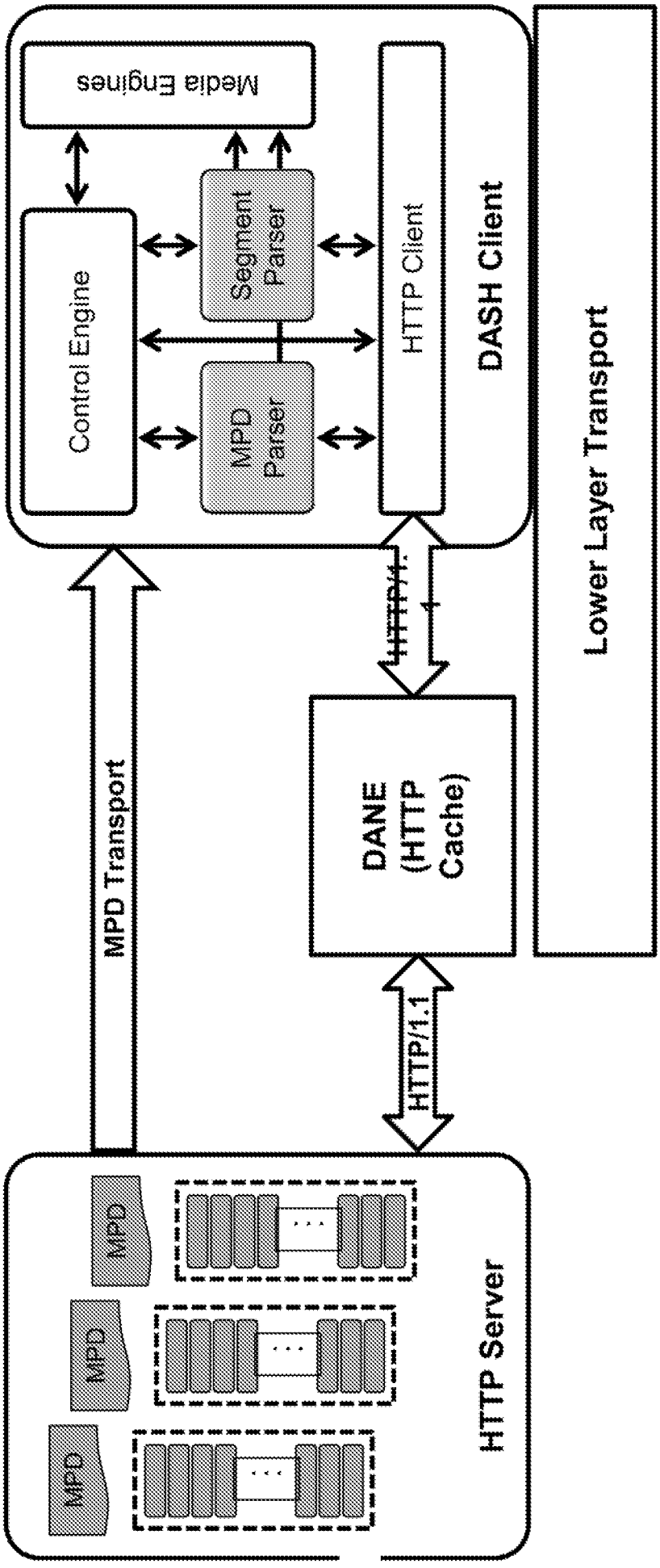
Media Engines
Control Engine
Segment Parser
MPD Parser
HTTP Client
DASH Client
Lower Layer Transport
MPD Transport
HTTP/1.1
DANE (HTTP Cache)
HTTP/1.1
MPD
MPD
MPD
HTTP Server

FIG. 8

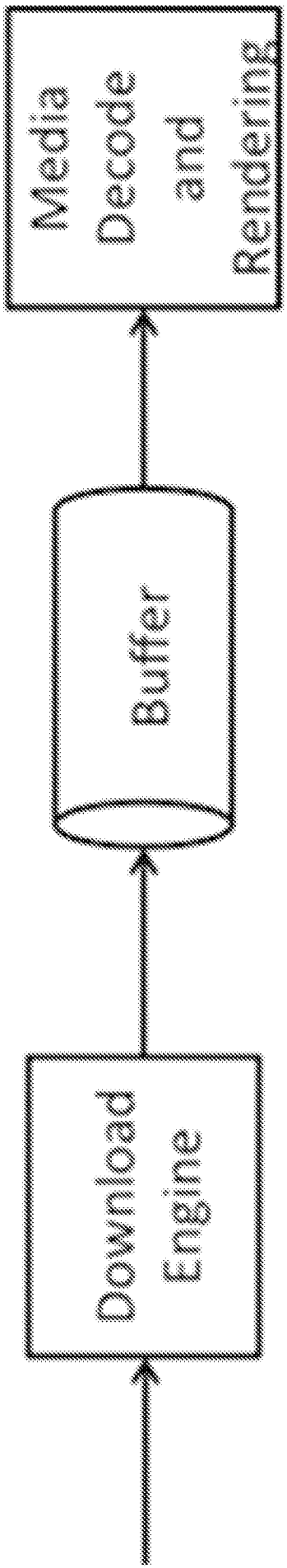

FIG. 9

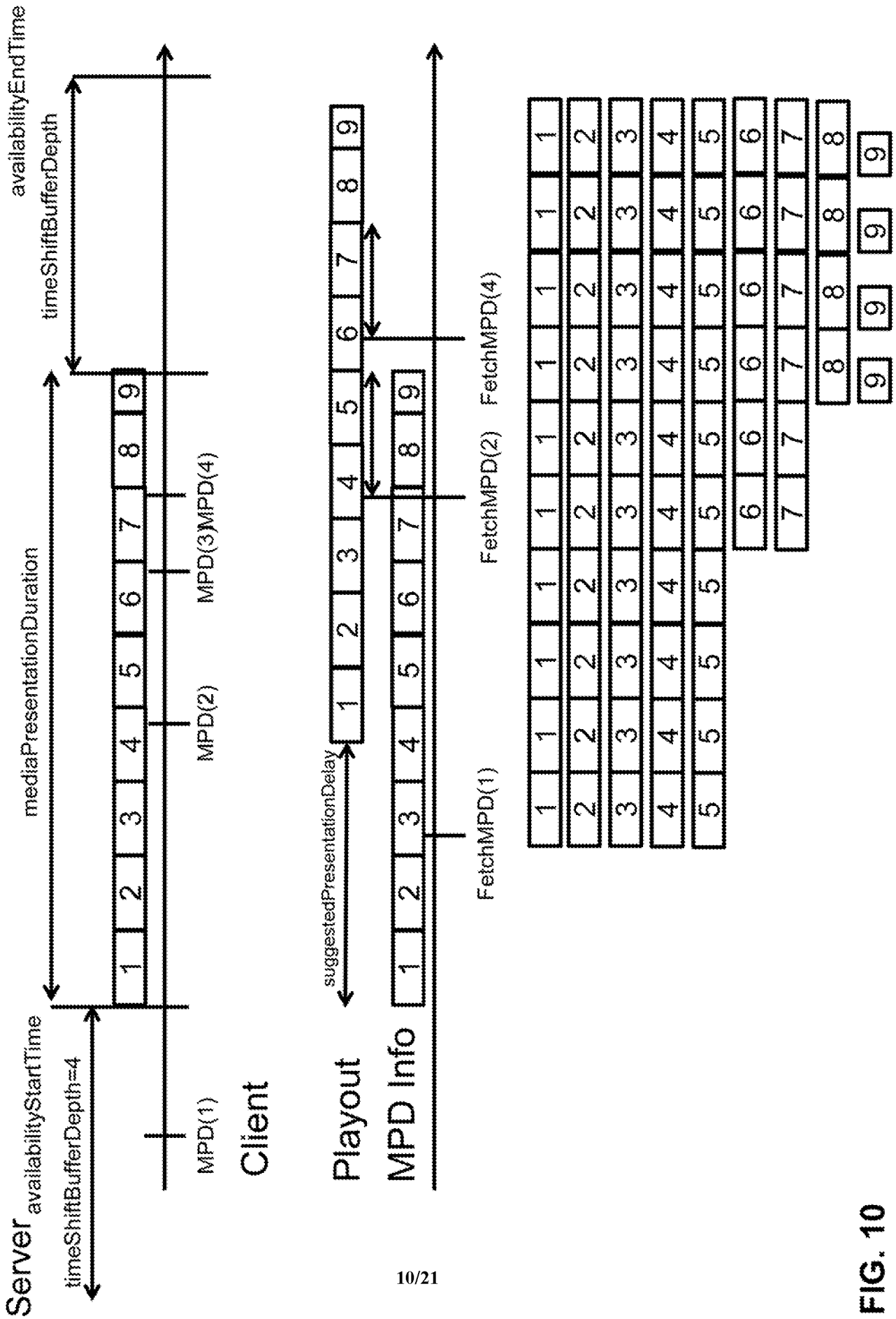
Server
availabilityStartTime
timeShiftBufferDepth=4
mediaPresentationDuration
availabilityEndTime
timeShiftBufferDepth
MPD(1)
MPD(2)
MPD(3)
MPD(4)
Client
Playout
MPD Info
suggestedPresentationDelay
FetchMPD(1)
FetchMPD(2)
FetchMPD(4)

FIG. 10

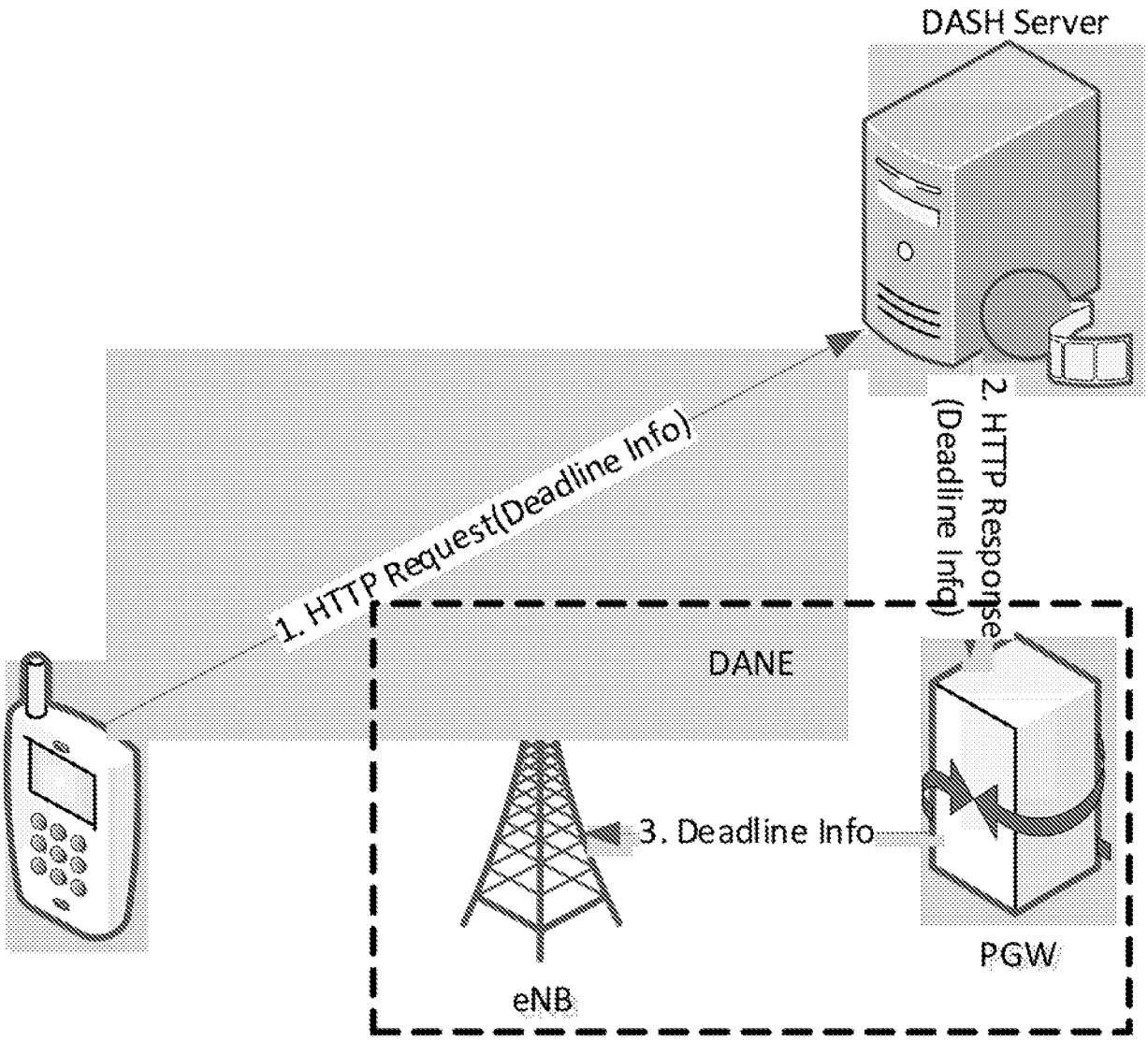
DASH Server
1. HTTP Request(Deadline Info)
2. HTTP Response (Deadline Info)
DANE
3. Deadline Info
PGW
eNB

FIG. 11

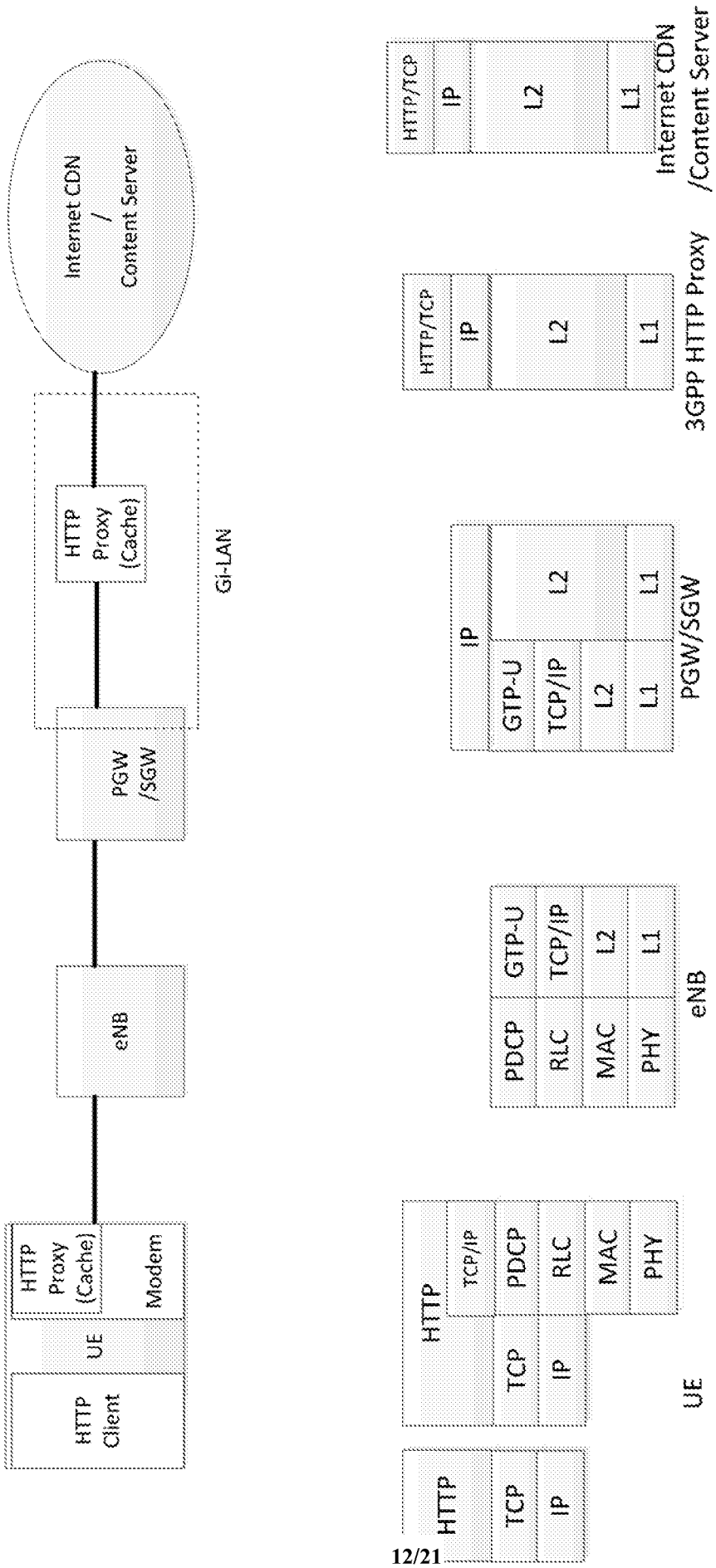
HTTP Client
UE
HTTP Proxy (Cache)
Modem
eNB
PGW /SGW
HTTP Proxy (Cache)
Gi-LAN
Internet CDN / Content Server
HTTP
TCP
IP
HTTP
TCP/IP
TCP
PDCP
IP
RLC
MAC
PHY
UE
PDCP
RLC
MAC
PHY
GTP-U
TCP/IP
L2
L1
eNB
IP
GTP-U
TCP/IP
L2
L2
L1
L1
PGW/SGW
HTTP/TCP
IP
L2
L2
L1
L1
3GPP HTTP Proxy
HTTP/TCP
IP
L2
L1
Internet CDN /Content Server

FIG. 12

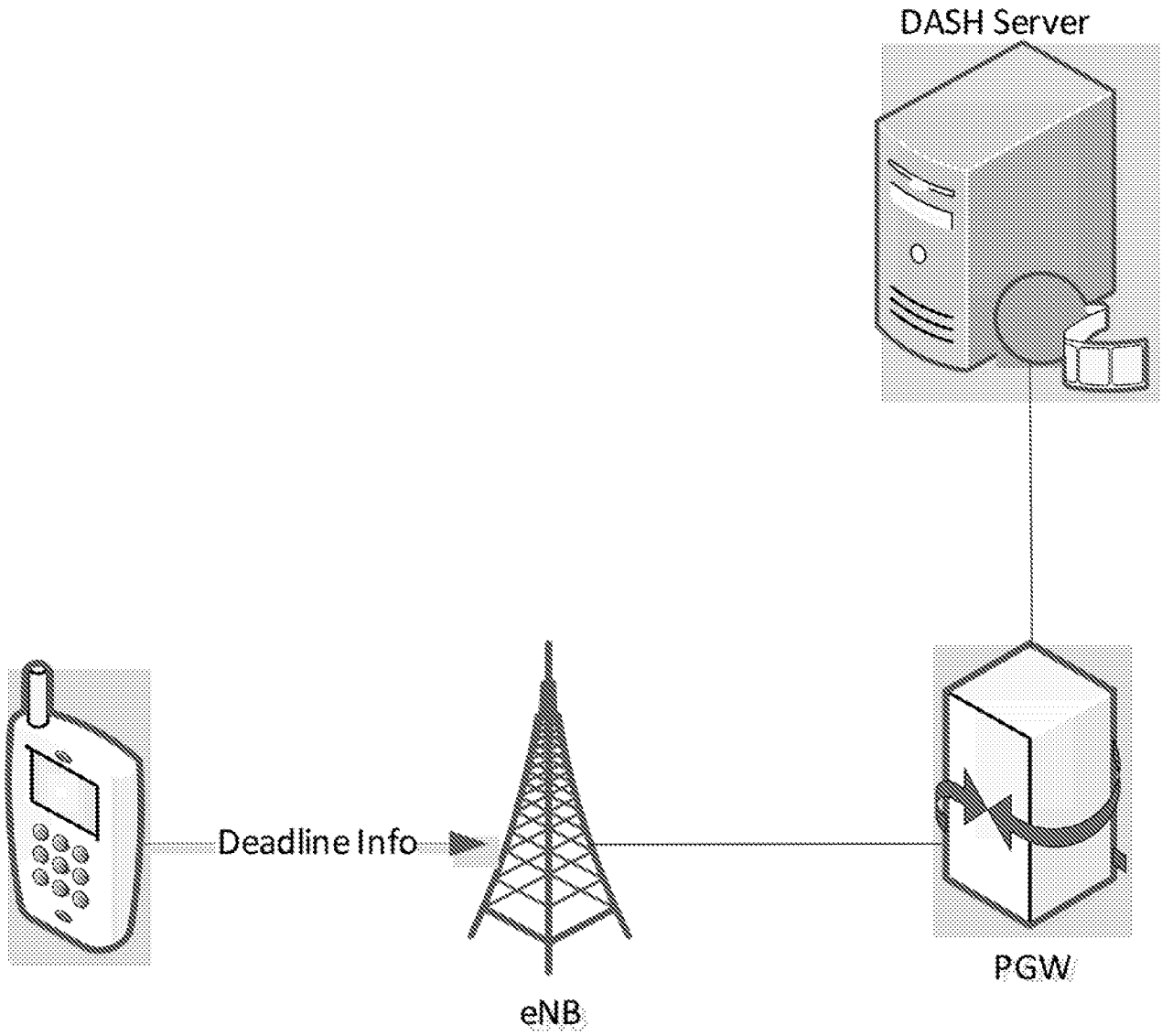

FIG. 13

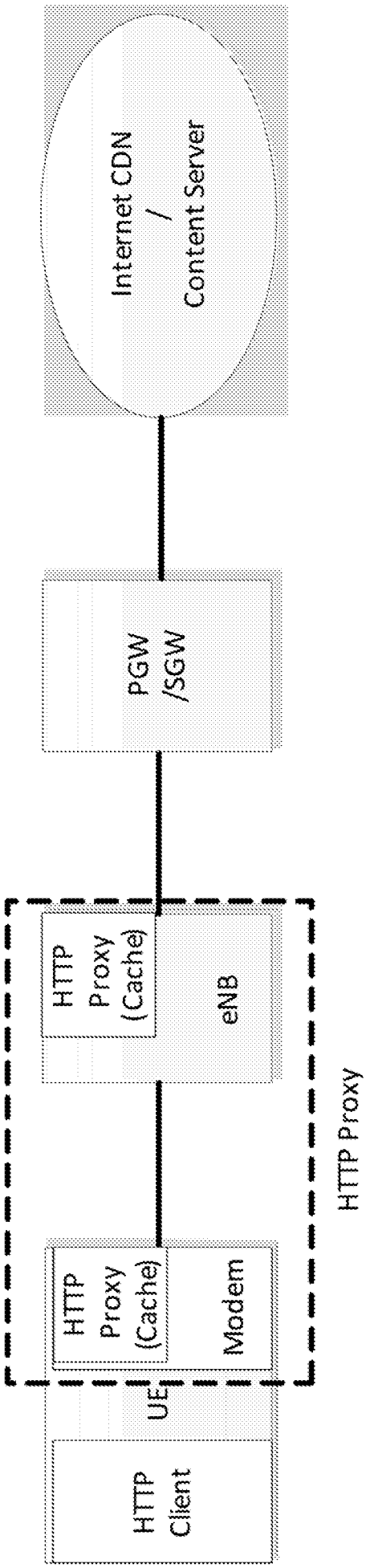
Internet CDN
/
Content Server
PGW
/SGW
HTTP
Proxy
(Cache)
eNB
HTTP Proxy
HTTP
Proxy
(Cache)
Modem
UE
HTTP
Client

FIG. 14

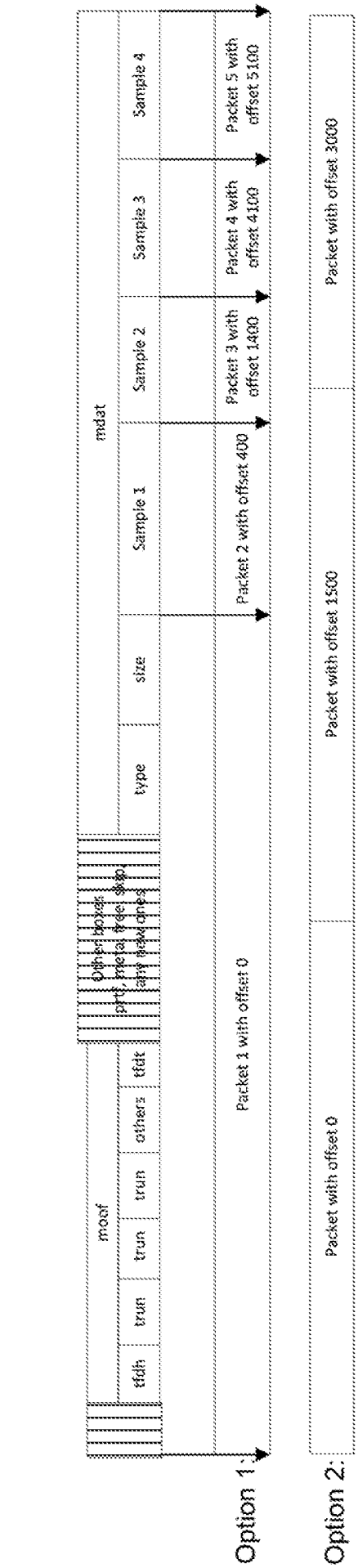
Option 1:
moof
tfdh
trun
trun
trun
others
tfdt
Other boxes
prft, meta, free, skip
data reference
mdat
type
size
Sample 1
Sample 2
Sample 3
Sample 4
Packet 1 with offset 0
Packet 2 with offset 400
Packet 3 with offset 1400
Packet 4 with offset 4100
Packet 5 with offset 5100
Option 2:
Packet with offset 0
Packet with offset 1500
Packet with offset 3000

FIG. 15

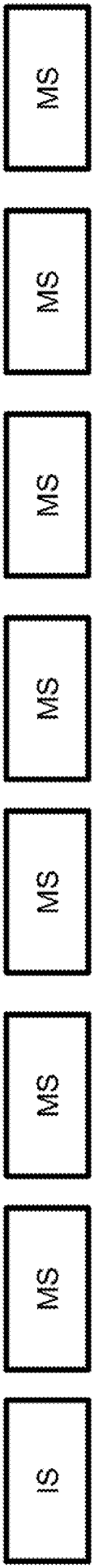

FIG. 16

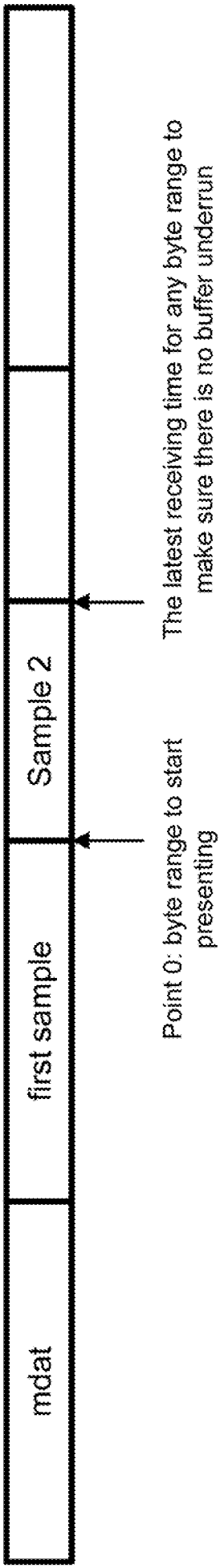

FIG. 17A

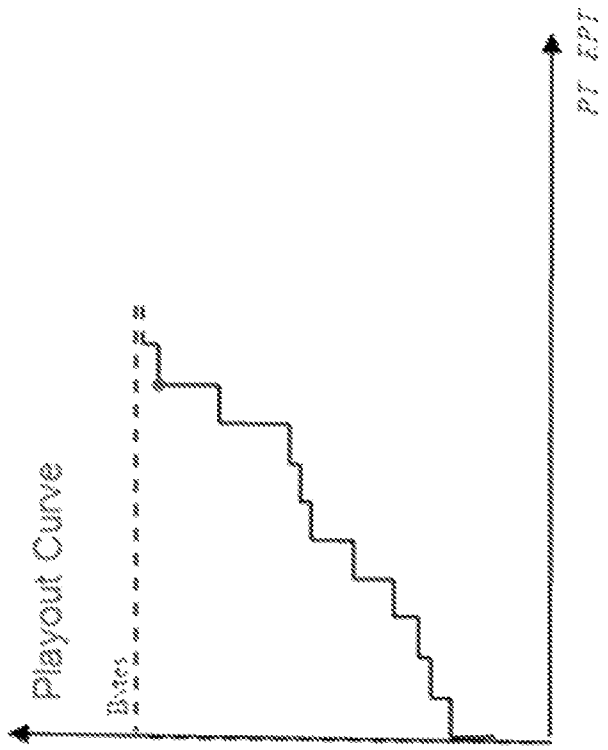

FIG. 17B

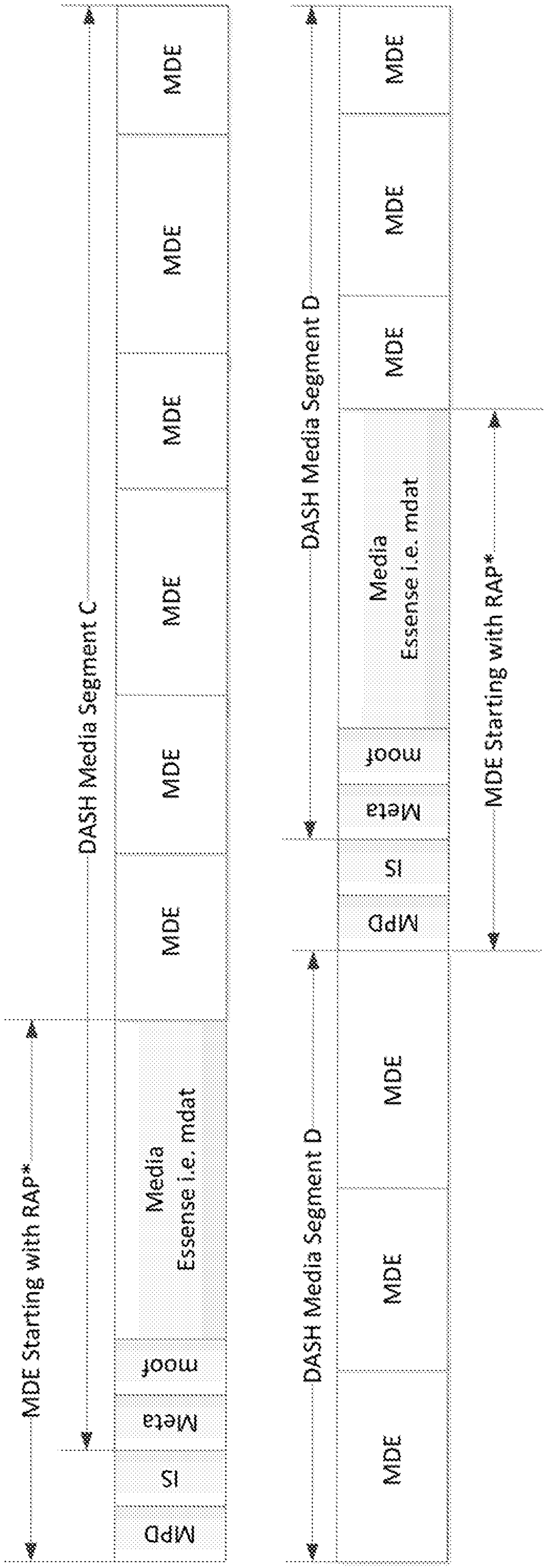
DASH Media Segment C
MDE Starting with RAP*
MPD
IS
Meta
moof
Media
Essense i.e. mdat
MDE
MDE
MDE
MDE
MDE
MDE
MDE
DASH Media Segment D
MDE
MDE
MDE
MPD
IS
Meta
moof
Media
Essense i.e. mdat
MDE Starting with RAP*
DASH Media Segment D
MDE
MDE
MDE

FIG. 18

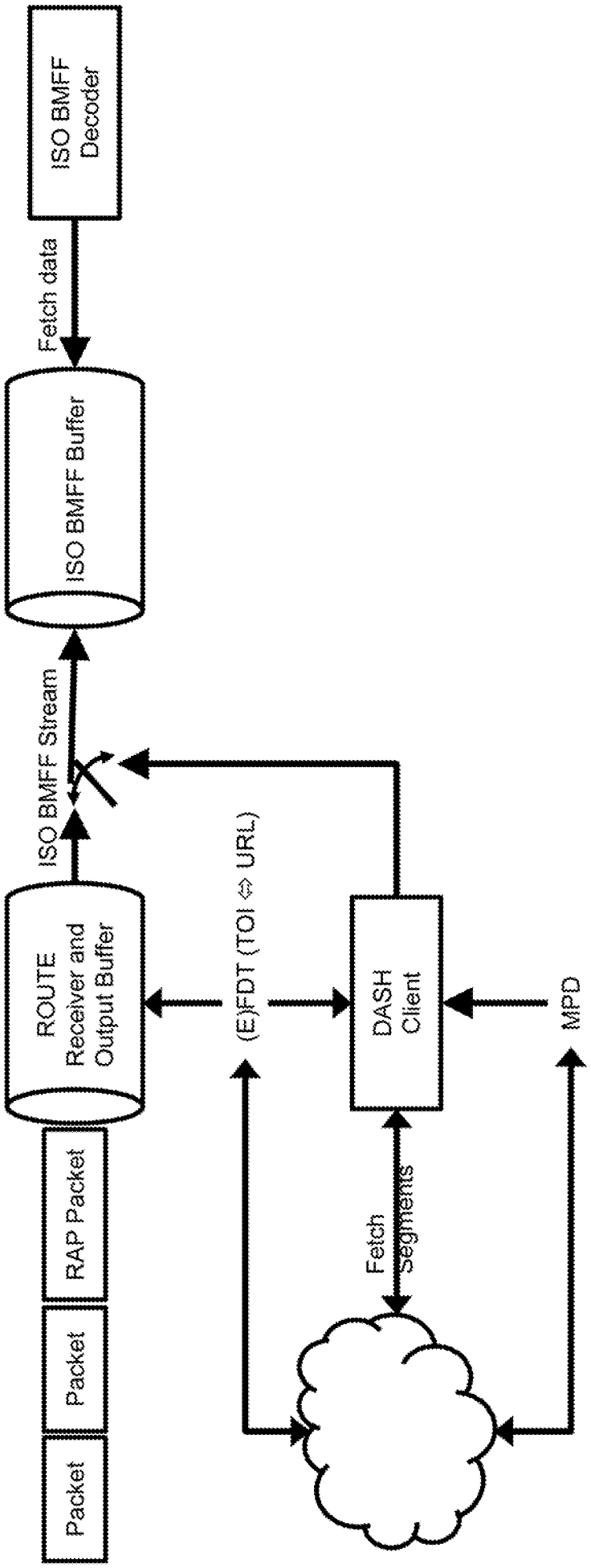
ISO BMFF Decoder
Fetch data
ISO BMFF Buffer
ISO BMFF Stream
ROUTE Receiver and Output Buffer
(E)FDT (TOI ⇔ URL)
DASH Client
MPD
Fetch Segments
RAP Packet
Packet
Packet

FIG. 19

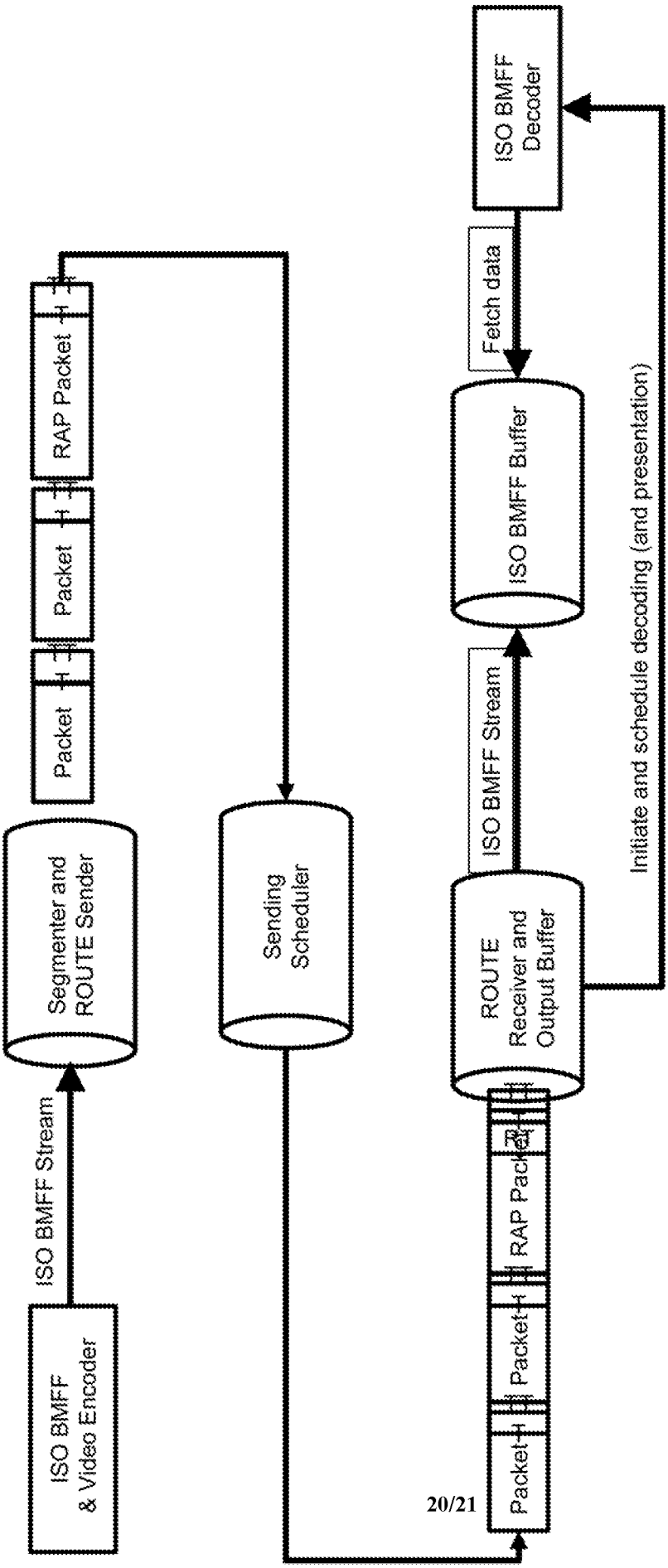
ISO BMFF & Video Encoder
ISO BMFF Stream
Segmenter and ROUTE Sender
Packet
Packet
Packet
RAP Packet
Sending Scheduler
ROUTE Receiver and Output Buffer
ISO BMFF Stream
ISO BMFF Buffer
Fetch data
ISO BMFF Decoder
Initiate and schedule decoding (and presentation)

FIG. 20

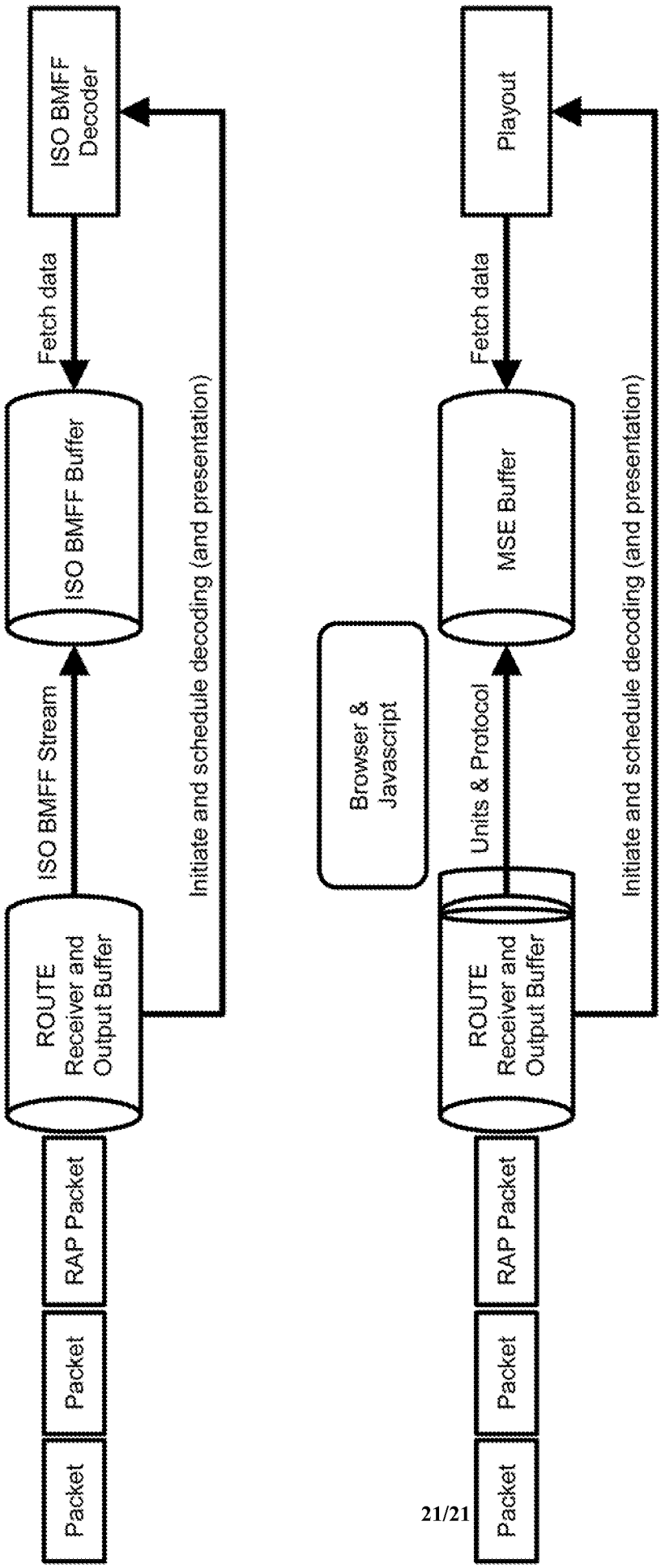
Packet
Packet
RAP Packet
ROUTE Receiver and Output Buffer
ISO BMFF Stream
ISO BMFF Buffer
Fetch data
ISO BMFF Decoder
Initiate and schedule decoding (and presentation)
Packet
Packet
RAP Packet
ROUTE Receiver and Output Buffer
Units & Protocol
Browser & Javascript
MSE Buffer
Fetch data
Playout
Initiate and schedule decoding (and presentation)

FIG. 21

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2015/092095

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/06(2006.01)i; H04L 29/08(2006.01)i; H04N 7/04(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L; H04N; H04W; G06F; H04B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS,CNTXT,CNKI,VEN: multimedia, video, audio, data, real-time, on line, download, deadline

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 103747283 A (CHINESE ACAD INST ACOUSTICS) 23 April 2014 (2014-04-23) claims 1-5, description, paragraph [009], figures 1-2	1-50
A	CN 104918072 A (ADOBE SYSTEMS INC) 16 September 2015 (2015-09-16) the whole document	1-50
A	CN 102333083 A (ZTE CORP) 25 January 2012 (2012-01-25) the whole document	1-50
A	CN 101420457 A (TENCENT TECH SHENZHEN CO LTD) 29 April 2009 (2009-04-29) the whole document	1-50
A	US 2011082914 A1 (DISNEY ENTPR) 07 April 2011 (2011-04-07) the whole document	1-50

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 June 2016	**22 July 2016**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China**	**SUN,Yufang**
Facsimile No. **(86-10)62019451**	Telephone No. **(86-10)62089463**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2015/092095

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 103747283 A	23 April 2014	None	
CN 104918072 A	16 September 2015	DE 102015002119 A1	17 September 2015
		GB 201503780 D0	22 April 2015
		US 2015264096 A1	17 September 2015
		GB 2525485 A	28 October 2015
CN 102333083 A	25 January 2012	WO 2012151865 A1	15 November 2012
CN 101420457 A	29 April 2009	CN 101420457 B	05 October 2011
US 2011082914 A1	07 April 2011	US 8484368 B2	09 July 2013
		US 2014019592 A1	16 January 2014