



(43) International Publication Date
29 December 2016 (29.12.2016)

(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/US2016/038987

(22) International Filing Date:
23 June 2016 (23.06.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/183,490 23 June 2015 (23.06.2015) US

(71) Applicant: CONVIDA WIRELESS, LLC [US/US]; 200
Bellevue Parkway, Suite 300, Wilmington, DE 19809-
3727 (US).

(72) Inventors: LU, Guang; 1 Mendel Crescent, Thornhill,
Ontario, L4J 9B6 (CA). WANG, Chonggang; 9 Carlyle
Court, Princeton, NJ 08540 (US). RAHMAN, Shamim,
Akbar; 6704 Conklin Road, Sote St. Luc, Quebec, H4V
1B8 (CA). DONG, Lijun; 6085 African Holly Trail, San
Diego, CA 92130 (US). LY, Quang; 115 Sterling Drive,
North Wales, PA 19454 (US). LI, Xu; 5313 Hunters Glen
Drive, Plainsboro, NJ 08536 (US). CHEN, Zhuo; 1397
Parish Avenue, Claymont, DE 19703 (US). CHOYI,

Vinod, Kumar; 1201 Minutemen Lane, Norristown, PA
19403 (US).

(74) Agents: SAMUELS, Steven, B. et al.; Baker & Hostetler
LLP, Cira Centre, 12th Floor, 2929 Arch Street, Phil-
adelphia, PA 19104-2891 (US).

(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,

[Continued on next page]

(54) Title: MECHANISMS TO SUPPORT ADAPTIVE CONSTRAINED APPLICATION PROTOCOL (COAP) STREAMING
FOR INTERNET OF THINGS (IOT) SYSTEMS

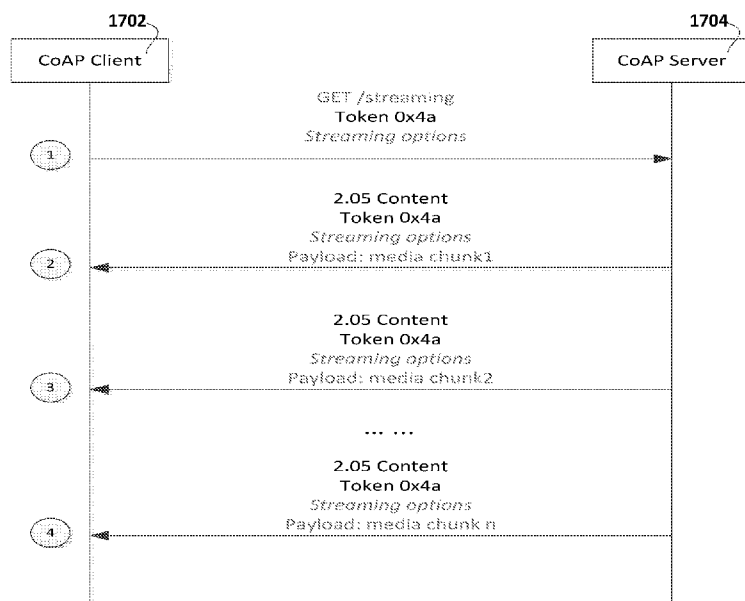


FIG. 17

(57) Abstract: The CoAP base protocol can be enhanced to support CoAP streaming. Streaming can use a reserved "/streaming" URI and current CoAP methods can be used towards the "/streaming" location, which will trigger or terminate streaming operations. Streaming can use a new STREAM method. Alternately, the current Observe mechanism can be enhanced to support streaming. Streaming operation can be combined with existing CoAP block transfer operations.



SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, —
GW, KM, ML, MR, NE, SN, TD, TG).

*before the expiration of the time limit for amending the
claims and to be republished in the event of receipt of
amendments (Rule 48.2(h))*

Published:

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

Mechanisms to Support Adaptive Constrained Application Protocol (CoAP) Streaming for
Internet of Things (IoT) Systems

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This Application claims the benefit of U.S. Provisional Patent Application Serial No. 62/183,490 filed June 23, 2015, the disclosure of which is hereby incorporated by reference as if set forth in its entirety.

BACKGROUND

[0002] Streaming media is multimedia that is received by and presented to a receiver while being delivered by a provider. There are two classes of streaming media: on-demand streaming and live streaming.

[0003] With on-demand streaming, also called video on-demand, the media has been previously recorded and likely compressed. The media files are stored at the server and delivered to one or multiple receivers when requested (on-demand). Popular sites provide streaming of stored audio and video today, including YouTube and CNN. Video on-demand allows users to select and watch/listen to video or audio content when they choose to, rather than having to watch at a specific broadcast time.

[0004] With live streaming, the media is captured, compressed and transmitted on the fly. Live streaming requires more computing resources and often specific hardware support. Live streaming also requires a form of source media (e.g. a video camera, an audio interface, screen capture software), an encoder to digitize the content, a media publisher, and a content delivery network to distribute and deliver the content.

[0005] Streaming media usually has stringent requirements on various performance metrics, for example, in terms of delay, jitter, loss, resolution, file size and network bandwidth.

[0006] The Real-time Transport Protocol (RTP), the RTP Control Protocol (RTCP) and the Real Time Streaming Protocol (RTSP) were common streaming protocols used before the Hypertext Transfer Protocol (HTTP) adaptive streaming became popular. RTP defines a standardized packet format for delivering audio and video over IP networks. RTP runs over the User Datagram Protocol (UDP). RTP is used in conjunction with the RTCP. While RTP carries the media streams (e.g., audio and video), RTCP is used to monitor transmission statistics and Quality of Service (QoS) and aids synchronization of multiple streams. RTP can carry a range of multimedia formats (such as H.264, MPEG-4, MPEG, etc.)

[0007] RTSP is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The transmission of streaming data itself is not a task of the RTSP protocol. Most RTSP servers use RTP in conjunction with RTCP for media stream delivery. However some vendors implement proprietary transport protocols.

[0008] RTSP defines control sequences useful in controlling multimedia playback. RTSP messages are Request/Response based, defining operations such as PLAY, PAUSE, SETUP, TEARDOWN, etc. RTSP is used when viewers communicate with a unicast streaming server. RTSP uses stateful session for its streaming. An RTSP session is a complete RTSP "transaction", for example, the viewing of a movie. A session typically consists of a client setting up a transport mechanism for the continuous media stream (SETUP), starting the stream with PLAY or RECORD, and closing the stream with TEARDOWN. Both the RTSP client and server maintain a state machine. Below is an example of the PLAY description in RTSP:

```
C->S: PLAY rtsp://example.com/media.mp4 RTSP/1.0
      CSeq: 4
      Range: npt=5-20
      Session: 12345678

S->C: RTSP/1.0 200 OK
      CSeq: 4
      Session: 12345678

RTP-Info:
url=rtsp://example.com/media.mp4/streamid=0;seq=9810092;rtptime=3450012
```

[0009] HTTP adaptive streaming refers to the methods to break the media file in small chunks and the client can download the chunks using the HTTP protocol. HTTP adaptive streaming can support media to be played in near real time.

[0010] There are several variations of HTTP adaptive streaming using similar approaches. Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is an adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. A similar solution is Apple's HTTP Live Streaming (HLS) solution as illustrated in more details below.

[0011] HTTP Live Streaming is a method to use HTTP to deliver streaming-like multimedia. It was implemented by Apple on their products such as QuickTime and iOS. HTTP live streaming is achieved by the following.

[0012] A multimedia presentation is specified by a URI to a Playlist file, which is an ordered list of media URIs and informational tags. The URIs and their associated tags specify a

series of media segments. Each media URI in a Playlist file specifies a media segment which is part of the overall presentation.

[0013] To play the presentation, the client first obtains the Playlist file and then obtains and plays each media segment in the Playlist. Each media segment has its own URI. It reloads the Playlist file as described in this document to discover additional segments.

[0014] Figure 1 illustrates the architecture and flow of operations for Apple's HTTP Live Streaming. There are three steps for HTTP live streaming.

[0015] An input stream is encoded/transcoded: The input 102 can be a satellite feed or any other type of input. The video and audio source is encoded (or transcoded) in encoder 104 to an MPEG-2 transport stream container, with H.264 video and Advanced Audio Coding (AAC) audio, which are the codecs Apple devices currently support.

[0016] Output profiles are created: Typically a single input stream will be transcoded to several output resolutions/bit rates, depending on the types of client devices that the stream is destined for. For example, an input stream of H.264/AAC at 7 Mbps could be transcoded to four different profiles with bit rates of 1.5Mbps, 750K, 500K, and 200K. These would be suitable for devices and network conditions ranging from high-end to low-end, such as an iPad, iPhone 4, iPhone 3, and a low bit rate version for bad network conditions.

[0017] The streams are segmented: The streams contained within the profiles all need to be segmented in stream segmenter 106 and made available for delivery to an origin web server or directly to a client device over HTTP. The software or hardware device that does the segmenting (the segmenter) also creates an index file/Playlist which is used to keep track of the individual video/audio segments.

[0018] The client 108 downloads the index file via a URL that identifies the stream. The index file tells the client where to get the stream chunks (each with its own URL). For a given stream, the client then fetches each stream chunk in order. Once the client has enough of the stream downloaded and buffered, it displays it to the user.

[0019] Constrained Application Protocol (CoAP) is being developed by IETF CoRE Working Group [RFC 7252, The Constrained Application Protocol (CoAP)] CoAP is a web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks.

[0020] The interaction model of CoAP 206 is similar to the client/server model of HTTP. One could think of CoAP 206 logically as using a two-layer approach as shown in Figure 2, a CoAP messaging layer 202 used to deal with UDP and the asynchronous nature of the

interactions, and the request/response interactions 204 using Methods (GET, PUT, POST, DELETE) and Response codes.

[0021] CoAP 204 defines four types of messages, Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), and Reset (RST).

[0022] A Confirmable (CON) message 302 is retransmitted using a default timeout and exponential back-off between retransmissions, until the recipient sends an Acknowledgement (ACK) message with the same Message ID. Figure 3 shows an example of the confirmable message 302 with ACK 304 of the same message ID 0x7d34.

[0023] An Acknowledgement (ACK) 304 is used to acknowledge the Confirmable message. When a recipient is not able to process a Non-confirmable message, it may reply with a Reset message (RST).

[0024] A Non-confirmable (NON) message 402 that does not require reliable transmission, for example each single measurement out of a stream of sensor data, can be sent as a Non-confirmable message 402. These are not acknowledged, but still have a Message ID for duplicate detection. An example with message ID 0x01a0 is shown in Figure 4.

[0025] CoAP request and response semantics are carried in CoAP messages, which include either a Method code or Response code, respectively. Optional (or default) request and response information, such as the URI and payload media type are carried as CoAP options. A Token is used to match responses to requests independently from the underlying messages.

[0026] A request is carried in a Confirmable (CON) or Non-confirmable (NON) message, and if immediately available, the response to a request carried in a Confirmable message is carried in the resulting Acknowledgement (ACK) message. This is called a piggy-backed Response, shown in Figure 5. If a request is sent in a Non-confirmable message, then the response is sent using a new Non-confirmable message.

[0027] CoAP messages are encoded in a simple binary format, shown in Figure 6. The message format starts with a fixed-size 4-byte header. This is followed by a variable-length Token value which can be between 0 and 8 bytes long. Following the Token value comes a sequence of zero or more CoAP Options in Type-Length-Value (TLV) format, optionally followed by a payload which takes up the rest of the datagram.

[0028] The fields in the header are defined as follows:
Version (Ver): 2-bit unsigned integer. Indicates the CoAP version number.
Type (T): 2-bit unsigned integer. Indicates if this message is of type Confirmable (0), Non-confirmable (1), Acknowledgement (2) or Reset (3).

Token Length (TKL): 4-bit unsigned integer. Indicates the length of the variable-length. Token field (0-8 bytes). Lengths 9-15 are reserved, are not sent, and are processed as a message format error.

Code: 8-bit unsigned integer, split into a 3-bit class (most significant bits) and a 5-bit detail (least significant bits), documented as c.dd where c is a digit from 0 to 7 for the 3-bit subfield and dd are two digits from 00 to 31 for the 5-bit subfield. The class can indicate a request (0), a success response (2), a client error response (4), or a server error response (5). All other class values are reserved. As a special case, Code 0.00 indicates an Empty message. In case of a request, the Code field indicates the Request Method; in case of a response it indicates a Response Code.

[0029] CoAP code registry is as follows:

0.00 Indicates an Empty message

0.01-0.31 Indicates a request. Values in this range are assigned by the "CoAP Method Codes" sub-registry

1.00-1.31 Reserved

2.00-5.31 Indicates a response. Values in this range are assigned by the "CoAP Response Codes" sub-registry

6.00-7.31 Reserved

Message ID: 16-bit unsigned integer in network byte order. Used for the detection of message duplication, and to match messages of type Acknowledgement/Reset to messages of type Confirmable/Non-confirmable.

[0030] The fields after the header in the rest of the message are defined as follows:

Token: 0 to 8 bytes, as given by the Token Length field. The Token value is used to correlate requests and responses.

[0031] Options: An Option can be followed by the end of the message, by another Option, or by the Payload Marker and the payload.

[0032] CoAP defines a number of options which can be included in a message. Both requests and responses may include a list of one or more options. For example, the URI in a request is transported in several options, and metadata that would be carried in an HTTP protocol header is supplied as options as well. Each option instance in a message specifies the Option Number of the defined CoAP option, the length of the Option Value and the Option Value itself. Option value can be empty, opaque, Uint (A non-negative integer) or String.

[0033] Both requests and responses may include a list of one or more options. CoAP defines a single set of options that are used in both requests and responses.

[0034] Options fall into one of two classes: "critical" or "elective". An Option is identified by an option number, which also provides some additional semantics information: e.g., odd numbers indicate a critical option, while even numbers indicate an elective option. The difference between these is how an option unrecognized by an endpoint is handled:

[0035] Upon reception, unrecognized options of class "elective" is silently ignored.

[0036] Unrecognized options of class "critical" that occur in a Confirmable request cause the return of a 4.02 (Bad Option) response.

[0037] Unrecognized options of class "critical" that occur in a Confirmable response, or piggy-backed in an Acknowledgement, cause the response to be rejected.

[0038] Unrecognized options of class "critical" that occur in a Non-confirmable message cause the message to be rejected.

[0039] Options are also classified based on how a proxy is to deal with the option if it does not recognize it. For this purpose, an option can either be considered Unsafe to Forward (Unsafe is set) or Safe-to-Forward (Unsafe is clear). In addition, for an option that is marked Safe-to-Forward, the option number indicates whether it is intended to be part of the Cache-Key in a request or not; if some of the NoCacheKey bits are 0, it is, if all NoCacheKey bits are 1, it is not. An option that is repeatable may be included one or more times in a message.

[0040] Table 1 shows the properties of CoAP options with two examples defined in the current CoAP specification called Proxy-scheme and Size1.

Table 1 Properties of Options

No.	C	U	N	R	Name	Format	Length	Default
Number	Critical	Unsafe	NoCacheKey	Repeatable				
39	x	x	-		Proxy-scheme	String	1-255	(none)
60			x		Size1	Uint	0-4	(none)

[0041] The CoAP Options are maintained by an IANA registry. The IANA policy for future additions to this sub-registry is split into three tiers as follows. The range of 0..255 is reserved for options defined by the IETF. The range of 256..2047 is reserved for commonly used options with public specifications (Specification Required). The range of 2048..64999 is for all other options including private or vendor specific ones.

[0042] CoAP defines four methods, GET, POST, PUT and DELETE.

[0043] The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request URI. Upon success, a 2.05 (Content) or 2.03 (Valid) response code should be present in the response.

[0044] The POST method requests that the representation enclosed in the request be processed. The actual function performed by the POST method is determined by the origin server and dependent on the target resource. It usually results in a new resource being created or the target resource being updated.

[0045] The PUT method requests that the resource identified by the request URI be updated or created with the enclosed representation. If a resource exists at the request URI the enclosed representation should be considered a modified version of that resource, and a 2.04 (Changed) response code should be returned. If no resource exists then the server may create a new resource with that URI, resulting in a 2.01 (Created) response code.

[0046] The DELETE method requests that the resource identified by the request URI be deleted.

[0047] The CoAP base protocol indicates that methods beyond the basic four can be added to CoAP in separate specifications. New methods do not necessarily have to use requests and responses in pairs.

[0048] CoAP core protocol is Request-Response based and does not work well when a client is interested in having the resource representation over a period of time. CoAP Observe (Observing Resources in CoAP, draft-ietf-core-observe-15) extends the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server.

[0049] Figure 7 shows an example of a CoAP client 702 registering its interest in a resource at a server 704. The client can retrieve a representation of the resource and request this representation be updated by the server as long as the client is interested in the resource.

[0050] Observe is a subscribe-notification mechanism for one Request to get multiple Responses. It can be viewed as a form of session, for the purpose of notifications.

[0051] To reduce IP fragmentation, CoAP defines a pair of "Block" options, for transferring multiple blocks of information from a resource representation in multiple request-response pairs. A pair of CoAP options (called "Block1" and "Block2") has been defined to enable block-wise transfer in CoAP. In general, both Block1 and Block2 options can be present both in request and response messages. In either case, the Block1 Option pertains to the request payload, and the Block2 Option pertains to the response payload.

[0052] The Block Option defines:

The size of the block (SZX);

Whether more blocks are following (M);

The relative number of the block (NUM) within a sequence of blocks with the given size.

[0053] Figure 8 shows an example of using the Block 2 option for the response message.

[0054] The current CoAP base protocol defines CoAP as a connection-less protocol. Figure 9 is a high level generic call flow indicating the establishment and release of a CoAP connection in one embodiment. There are detailed mechanisms about how to enable CoAP connections that are not shown here.

[0055] This draft proposed a simple CoAP based media streaming method using the CoAP Observe mechanism. It proposed a new CoAP option “Streaming”. The main purpose of this option is to indicate when a message will break into chunks of known size.

[0056] The basic cycle of an application using CoAP streaming is as follows, and shown in Figure 10:

[0057] The client 1002 registers itself with a resource at a server 1004 by performing a GET request that includes an Observe Option.

[0058] If the observation relationship is established between the server 1004 and the client 1002, the server 1004 sends a CoAP streaming response to the client, including the Observe Option, whenever some new media chunk is available.

[0059] The server 1004 sets the value of the Observe Option in each notification to the 16 least-significant bits of a strictly increasing sequence number that contain no gap. If there is no gap in the Observe Option between two sequence notifications a client 1002 is receiving, no chunk has been lost or delayed by the network. The Option number can be viewed the same as “sequence number” of RTP and be used to reorder chunks that may arrive out of order.

[0060] A client can discover different codecs a resource (i.e. videocamera) on a server 1004 supports, using the Core Resource Discovery mechanism (RFC 6690 Constrained RESTful Environments (CoRE) Link Format). The client then chooses a proper codec to observe by selecting the right URI. Figure 11 shows the operations for a client 1002 to discover supported codecs by the server.

[0061] In many M2M scenarios, direct discovery of resources is not practical due to sleeping nodes, disperse networks, etc. These problems can be solved by employing an entity called a Resource Directory (RD) 1202, which hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources. The CoRE Resource Directory draft (CoRE Resource Directory (draft-ietf-core-resource-directory-02)) specifies the web interfaces that a Resource Directory supports in order for web servers to discover the RD 1202

and to register, maintain, lookup and remove resource descriptions. End Points (EP) 1204 are assumed to proactively register and maintain resource directory entries on the RD 1202. RD 1202 supports RESTful web interfaces for Clients 1210 to look up and discover resources.

[0062] Discovery is performed by sending either a multicast or unicast GET request to `"/.well-known/core"` and including a Resource Type (rt) parameter with the value `"core.rd"` in the query string.

[0063] After discovering the location of an RD Function Set, an endpoint may register its resources using the registration interface. The registration request interface is specified as follows:

- Interaction: EP -> RD
- Method: POST
- URI Template: `/{"+rd"} {"?ep,d,et,lt,con}"`
- rd := RD Function Set path (mandatory). This is the path of the RD Function Set, as obtained from discovery.
- ep := Endpoint (mandatory). The endpoint identifier or name of the registering node, unique within that domain. d := Domain (optional). The domain to which this endpoint belongs.
- et := Endpoint Type (optional). The semantic type of the endpoint.
- lt := Lifetime (optional). Lifetime of the registration in seconds.
- con := Context (optional). This parameter sets the scheme, address and port at which this server is available in the form `scheme://host:port`.
- Content-Type: `application/link-format`

SUMMARY

[0064] To enable a constrained device with streaming, the constrained protocol can be given the streaming capability, so the constrained device can support it, and does not need to support another streaming protocol.

[0065] CoAP is a protocol developed specifically for constrained devices typical of IoT sensors and gateways. Procedures are described that enable streaming content through constrained devices beyond the limited and insufficient methods defined in CoAP Streaming (draft-loreto-core-coap-streaming-00), March 27, 2012.

[0066] Detailed mechanisms enhance the CoAP base protocol to support CoAP streaming. The mechanisms can include :

- Enable streaming using a reserved “/streaming” URI. In this mechanism, current CoAP methods can be used towards the “/streaming” location, which will trigger or terminate streaming operations

- Enable streaming using a new STREAM method
- Enhance the current Observe mechanism to support streaming
- Combine streaming operation with existing CoAP block transfer
- A list of streaming options to be used to define and configure streaming operations. These options can be used in combination to enable adaptive streaming.

[0067] Examples are given that illustrate how CoAP streaming mechanisms described above can support CoAP adaptive streaming. The following focuses on the impacts to the CoAP protocol to support streaming operations. It does not repeat commonly known methods for streaming such that the server has video files in chunks. On demand streaming and live streaming are discussed together and not differentiated. The mechanisms can apply to both. Video and audio streaming are discussed together.

[0068] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to limitations that solve any or all disadvantages noted in any part of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0069] A more detailed understanding may be had from the following description, given by way of example in conjunction with the accompanying drawings wherein:

[0070] Figure 1 is a diagram illustrating HTTP Live Streaming (HLS).

[0071] Figure 2 is a diagram illustrating the Abstract Layering of CoAP.

[0072] Figure 3 is a diagram illustrating a CoAP Confirmable Message.

[0073] Figure 4 is a diagram illustrating a CoAP Non-Confirmable Message.

[0074] Figure 5 is a diagram illustrating a CoAP Request/Response.

[0075] Figure 6 is a diagram illustrating a CoAP Message Format.

[0076] Figure 7 is a diagram illustrating CoAP Observe.

[0077] Figure 8 is a diagram of an example of Block 2 Transfer.

[0078] Figure 9 is a diagram illustrating a generic CoAP Connection Call Flow.

[0079] Figure 10 is a diagram illustrating a CoAP Streaming using Observe.

- [0080] Figure 11 is a diagram illustrating a Media Type Discovery in CoAP.
- [0081] Figure 12 is a diagram illustrating the Resource Directory Architecture.
- [0082] Figure 13 is a diagram illustrating eHealth Home Monitoring System.
- [0083] Figure 14 is a diagram illustrating non-IoT Streaming.
- [0084] Figure 15 is a diagram illustrating IoT Streaming.
- [0085] Figure 16 is a diagram illustrating the use of a RD for Streaming Capability Discovery.
- [0086] Figure 17 is a diagram illustrating CoAP Streaming using /streaming URI (Request-Response).
- [0087] Figure 18 is a diagram illustrating CoAP Streaming using /streaming Location (Push).
- [0088] Figure 19 is a diagram illustrating using STREAM Method (Request-Response).
- [0089] Figure 20 is a diagram illustrating using STREAM Method (Push).
- [0090] Figure 21 is a diagram illustrating CoAP Streaming in Combination with CoAP Block Transfer.
- [0091] Figure 22 is a diagram illustrating triggering of CoAP Streaming.
- [0092] Figure 23 is a diagram illustrating Adaptive CoAP Streaming.
- [0093] Figure 24 is a diagram illustrating an exemplary user interface.
- [0094] Figure 25 is a diagram that illustrates an exemplary graphical user interface for use with embodiments.
- [0095] Figure 26A is a diagram of an example machine-to machine (M2M) or Internet of Things (IoT) communication system in which one or more disclosed embodiments of IoT event management systems and methods may be implemented.
- [0096] Figure 26B is a system diagram of an example architecture that may be used within the M2M/IoT communications system illustrated in Figure 26A.
- [0097] Figure 26C is a system diagram of an example M2M / IoT terminal or gateway device that may be used within the communications system illustrated in Figure 26A.
- [0098] Figure 26D is a block diagram of an example computing system in which aspects of the communication system of Figure 26A may be embodied.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0099] Figure 13 shows an eHealth home monitoring system 1300. A patient has multiple sensors 1302 on his body. Some sensors 1302 send data readings, for example, the patient's heart beat and body temperature. The system 1300 supports a voice system that the remote health care workers can use to interact with the patient. The system 1300 has a video monitoring function 1308. The function 1308 can be triggered by different ways, such as by the remote health care worker, by a timer, or by abnormal data from the sensors. All the equipment connects to an eHealth gateway 1304 at home, and the gateway 1304 connects to the health care system 1306 in a hospital. In this use case example, the devices used in the home monitoring system 1300 vary a lot in their capacities, from very constrained device used for sensor data, to devices supporting voice and video functions. If only sensor data is transported, CoAP can be used for constrained devices. However, to support voice and video in addition to data, the home monitoring system 1300 needs to support another media protocol at the eHealth gateway 1304. If the devices for voice and video run another protocol rather than CoAP, these devices cannot interoperate with constrained devices running CoAP within the same home monitoring system, unless they communicate via the eHealth gateway 1304. The gateway 1304 would need to translate between different devices in this small deployment. This would increase the complexities for devices, in particular the eHealth gateway 1304.

[0100] It is understood that the functionality illustrated in Figure 13, may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, an apparatus of an M2M network (e.g., a server, gateway, device, or other computer system), such as one of those illustrated in Figures 26C or 26D described below.

[0101] Traditionally, streaming is largely used for entertainment and involves human interactions. There were limited IoT streaming use cases, and they usually used proprietary protocols, such as video surveillance.

[0102] The characteristics of non-IoT streaming are very different from IoT streaming as shown in Figure 14 and Figure 15 respectively. Non-IoT streaming shown in Figure 14 is usually triggered by a person, when the person performs certain actions on a device or on a website, and the streaming can last long. For IoT streaming, it might be triggered by an IoT device as shown in Figure 15 (case a). Or it might be pushed from the media source for a period of time (case b). It may be automated for many streaming sessions, and each lasts certain period of time (case c). The streaming duration can vary from very long to very short, depending on how the streaming is used.

[0103] It is understood that the entities performing the steps illustrated in Figures 14-15 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figures 14-15 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figures 14-15. It is also understood that any transmitting and receiving steps illustrated in Figures 14-15 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0104] M2M/IoT deployments require light weight protocols to be used due to the constrained nature of IoT devices. CoAP is developed for IoT devices and the currently CoAP Streaming Internet Draft, does not have well developed mechanisms to effectively support streaming.

[0105] As illustrated in the use case above, it is common that in the IoT deployment there will be mixed types of media types and mixed devices (constrained and not constrained). If the IoT devices for streaming are not constrained, they may be able to support existing streaming protocols such as RTP/RTCP or HTTP. However, this means that there could be multiple protocols, such as HTTP and CoAP, even in a small scale deployment. It can largely increase the complexity of a system and limits the deployment options. For example, in the use case above, the eHealth home gateway may be a constrained device itself and not suitable to support multiple protocols or translate between different protocols.

[0106] Therefore it is desirable to have a light weight IoT protocol to support streaming and CoAP is a good candidate.

[0107] Since CoAP is specifically designed for IoT devices, it will be largely adopted for M2M communication. In particular, the current existing functions defined in the CoAP base protocol, and additional CoAP features like CoAP block transfer, can be a good basis to be leveraged and enhanced for supporting CoAP streaming. Such moderate modifications on existing CoAP functions can avoid significant development or re-deployment costs.

[0108] The current CoAP streaming proposal based on Observe is very preliminary and does not provide all functions needed and resolve issues using Observe.

[0109] Observe is basically a Subscribe-Notify mechanisms, not designed sufficiently for streaming, e.g. it does not support the basic operations for streaming, such as Play, Pause.

[0110] Observe follows a best-effort approach for sending the current representation to the client after a state change (see Figure 7) so it cannot support the stringent QoS requirements for streaming

[0111] A CoAP server is the authority for determining under what conditions resources change their state and thus when observers are notified of new resource states. A CoAP server may chose not to send notifications at all. Therefore, the Observer (the CoAP client), cannot fully determine what it can get.

[0112] CoAP Observe needs to be modified to support streaming function effectively.

[0113] The following sections discuss new mechanisms for the CoAP protocol to support streaming. Since CoAP is designed mainly for IoT deployments, the mechanisms are meant to be light-weight to minimize the overhead.

[0114] CoAP is designed as a light weight protocol for IoT devices. The principle goal of CoAP streaming should also be light weight so the streaming operation does not add unnecessary overhead. To increase the flexibility and capability of CoAP, it should support discoverable configurations from very simple streaming mechanisms to more complicated streaming mechanisms. Table 2 shows the summary of mechanisms discussed in this paper. One or more of the mechanisms can be supported by a CoAP endpoint. However, the mechanisms can be all supported and used together for streaming operations. Each mechanism is described below IoT streaming options to support different streaming operations.

[0115] Note that since many IoT devices support single functionality (i.e. there is only a single source of media), the mechanisms support streaming from a default source, and the requestor of the streaming does not always have to choose the streaming source. It is preferred that CoAP non-confirmable messages are used for streaming. This can reduce the number of messages and keep the streaming operation lightweight. CoAP streaming does not mandate the use of CoAP connection since streaming can be achieved without any state information, however it can work together with CoAP connection (see the use case illustrated by Figure 21).

Table 2 Summary of CoAP Streaming Mechanisms

CoAP Streaming Mechanisms	Descriptions	Discovery
using "/streaming" URI (see 5.1.1 for details)	This is a mechanism re-using the current CoAP base protocol for CoAP streaming. The client can trigger the streaming operation by directly operating on the	The client needs to discover that the server supports this operation, and the media

	reserved “/streaming” URI, not on any specific media resources. The client may stop the streaming explicitly or indicate a timer of the duration. An example use case that is suitable for this mechanism is that the client triggers a video camera to start streaming for a certain period of time. The client can also indicate the interval and size of the video chunks.	types the server supports.
Using STREAM method (see 5.1.2 for details)	This is a mechanism dedicated for CoAP streaming by introducing a new STREAM method.	The client needs to discover that the server supports this operation, and the media types the server supports.
using CoAP Observe (see 5.1.3 for details)	This mechanism uses the CoAP Observe mechanism so the client observes a specific media resource.	The client needs to discover that the server has observable media resources that the client can Observe.

[0116] Discovery between the CoAP client and server can be achieved by the exchange of streaming options between a CoAP client and a CoAP server. CoAP Resource Directory (RD) 1202 may also be used for discovery. Figure 16 shows an example how RD 1202 can be used for a CoAP streaming server (CoAP End Point 1602) to register its streaming related configurations as resources in RD 1202 to be discovered by CoAP clients. In this example a CoAP endpoint 1602 which is also a streaming server has a name “camera1”. It registers its supported streaming options as resources at RD 1202, using the RD registration interface.

[0117] Step 1 of Figure 16, the CoAP End Point sends the POST request to RD, using the RD registration interface specified in CoRE Resource Directory (draft-ietf-core-resource-directory-02). The POST request provides its end point name as “camera1”. In the payload, the POST request specifies that the content type (ct) is application/XML type, the resource type (rt) is “StreamingOptions”.

[0118] Request has two new link attributes to support CoAP streaming options, one is called StreamingType with “STREAMMethod” as the supported streaming type, and one is StreamingMediaType with “mpeg” as the supported streaming media type. The interface (if) is defined as “StreamingServer”. More streaming options can be included using the same method.

[0119] In Step 2 of Figure 16, the resulting location /rd/1045 is just an example of an RD generated location. A client as defined in the RD draft (CoRE Resource Directory (draft-ietf-core-resource-directory-02)) can retrieve streaming options from RD.

[0120] It is understood that the entities performing the steps illustrated in Figure 16 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s)

illustrated in Figure 16 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 16. It is also understood that any transmitting and receiving steps illustrated in Figure 16 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0121] Figure 17 illustrates the mechanism of CoAP streaming using the reserved “/streaming” URI. In Figure 17, it is assumed that the CoAP client already discovered that the CoAP server has streaming capability.

[0122] In Step 1 of Figure 17, the CoAP client 1702 triggers the streaming operation by performing a GET operation to the reserved “/streaming” URI. Optionally, the client may also include one or multiple options to define the parameters of the streaming. For example, the client may indicate the “StreamingMediaType” it supports. Or the client may indicate the duration of the streaming so the streaming session will be automatically ended. The details of streaming options are defined below.

[0123] There can be various ways for a CoAP client 1702 to address the streaming URI. Note that in the following methods, “streaming”, “video” and “audio” are used as reserved terms. The use of these terms results in the streaming operation.

[0124] /streaming: This is the common use for simple streaming cases, and the server will be streaming from its own media resource. The client does not need to have the knowledge of the media resource that the server has.

[0125] /streaming/video: This directory can be reserved for video streaming. The server streams from its video media resource. The client does not need to know where the media resource is stored on the server.

[0126] /streaming/audio: This directory can be reserved for audio streaming. The server streams from its audio media resource. The client does not need to know where the media resource is stored on the server.

[0127] /streaming/myownvideo/videofile: This mechanism does not exclude that the client operates on a specific media resource. Note that the path of the video file in this example is a combination of the reserved URI and the file’s location. The file is located under /myownvideo/videofile on the server, and “/streaming” is not a real directory.

[0128] In Step 2 of Figure 17, since the GET operation is on the reserved “/streaming” URI, the CoAP server 1704 knows to interpret the operation as triggering streaming. The CoAP

server 1704 provides the streaming contents in the format of small file chunks. The server 1704 may exchange streaming options with the client 1702 by including streaming options to let the client 1702 know about the current streaming capabilities and preferences of the server. By exchanging streaming options, the client and server 1704 can adapt the streaming operations. For example, the server may indicate the total size of the video file that the server has. By knowing that, the client 1702 can change its video options, for example, to limit the total size of the chunks it wants to receive.

[0128] In Step 3- Step 4 of Figure 17, the streaming operation will continue until either the specified time has expired, or the CoAP server 1704 has finished sending all the chunks. The Requests and Responses are mapped using the same Token.

[0129] The CoAP client 1702 can explicitly stop the streaming operation by sending a DELETE /streaming Request to the server 1704 . The CoAP client 1702 can also use the “StopMode” to stop a streaming operation.

[0130] It is understood that the entities performing the steps illustrated in Figure 17 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 17 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 17. It is also understood that any transmitting and receiving steps illustrated in Figure 17 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0131] Figure 18 shows the push operation using the /streaming URI. The figure shows the case using CoAP Non-confirmable message model so no ACK is sent for each of the request messages. If reliability is a key factor, CoAP confirmable message can be used. The figure shows using PUT for the push of contents. POST might be used as well.

[0132] In Step 2 – Step 3 of Figure 18, since the PUT operation is on the reserved “/streaming” location, the CoAP server knows to interpret the operation as pushing stream media. The CoAP server 1704 will treat this PUT operation differently from other non-streaming related PUT operations. If the client did not include any specific URI, the server will process the chunks based on its own logic, for example, displaying it using an application. The server 1704 may or may not choose to store it in its own location. If the server stores the contents locally,

such contents can also be retrieved by another CoAP client using the call flow shown in Figure 17. The server may not compose and send a Response message.

[0133] The streaming operation will continue until either the specified time expired, or the CoAP client 1702 sends all the chunks. The CoAP client 1702 can also explicitly stop the streaming operation by sending a DELETE /streaming Request to the server. If at any time the server 1704 decides that it would not like to receive the streaming content anymore, for example, due to the limit of storage or processing capacity at the server side, the server 1704 can send a Response message and include the “PauseMode” or “StopMode” option to pause or stop the streaming operation.

[0134] It is understood that the entities performing the steps illustrated in Figure 18 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 18 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 18. It is also understood that any transmitting and receiving steps illustrated in Figure 18 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0135] The CoAP base protocol indicates that “Methods beyond the basic four can be added to CoAP in separate specifications. New methods do not necessarily have to use requests and responses in pairs.”

[0136] Streaming is a very different type of operation comparing to data exchanges using CoAP. Therefore a new STREAM method can support this specific operation.

[0137] The STREAM method triggers the streaming operation. If no payload is included, it is a Request to trigger the streaming operation at the CoAP server. Optionally the CoAP client can include a URI pointing to the media resource requested. If no URI is included, the CoAP server uses its default media resource. The CoAP server should start streaming upon receiving the request. This scenario is illustrated in Figure 19.

[0138] If there is payload included in the STREAM message, it is a push message to the server and the CoAP client is the streaming source. This scenario is illustrated in Figure 20.

[0139] To stop streaming, the streaming option “StopMode” should be included in the streaming Option. Alternatively, there could be a “STREAMON” and a “STREAMOFF” methods respectively for starting and stopping streaming.

[0140] In Step 1 of Figure 19, the CoAP client triggers the streaming operation by sending a CoAP request message using the STREAM method. The client 1702 may include a URI pointing to the resource where the streaming media is stored. When the client 1702 and server 1704 are both constrained and there can be just one default streaming source location, and the client 1702 is not aware of the location. In this case, the client 1702 does not specify the streaming URI and the server 1704 will use the default streaming source. Optionally, the client may also include one or multiple options to define the parameters of the streaming.

[0141] In Step 2 of Figure 19, upon receiving the request, the CoAP server 1704 provides the streaming contents in the format of small file chunks. The server 1704 may also optionally include streaming options to let the client 1702 know about the current streaming capabilities and preferences of the server 1704.

[0142] Error codes defined in the CoAP base protocol can be used for error cases. In cases that the server does not support the STREAM method, the server 1704 should return 4.05 Method Not Allowed. In cases that no media streaming file is located at the URI specified by the client, the server should return 4.04 Not Found.

[0143] In Step 3- Step 4 of Figure 19, the streaming operation will continue until either the specified time expired, or the CoAP server sends all the chunks. The Requests and Responses are mapped using the same Token. The CoAP client 1702 can also explicitly stop the streaming operation by sending a STREAM request with the option sets as “StopMode”.

[0144] It is understood that the entities performing the steps illustrated in Figure 19 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 19 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 19. It is also understood that any transmitting and receiving steps illustrated in Figure 19 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0145] In Step 1 of Figure 20, the CoAP client 1702 initiates the streaming push operation by sending a CoAP request using the STREAM method. The client 1702 includes media chunks in the payload. Optionally, the client 1702 may also include one or multiple options to define the parameters of the streaming. Prior to this step of operation, the CoAP client 1702 may need to exchange streaming options with the server 1704, for example, the CoAP client 1702 needs to indicate the “StreamingMediaType” it supports.

[0146] In Step 2 – Step 3 of Figure 20, upon receiving the streaming push request, the CoAP server 1704 can process the payload based on its own logic. For example, the server 1704 may provide the media chunk to an application to display the media, and/or the server 1704 may choose to store the media chunk locally.

[0147] It is understood that the entities performing the steps illustrated in Figure 20 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 20 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 20. It is also understood that any transmitting and receiving steps illustrated in Figure 20 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0148] CoAP Observe was not designed for streaming purpose. It is a mechanism to support subscription and notification. A CoAP client 1702 can register itself with the CoAP server 1704 for interests in the state change of a resource over a period of time by issuing a GET request with the Observe option. Therefore to support streaming, some modifications are needed for Observe.

[0149] In the first enhancement, the CoAP server 1704 needs explicit indication that the Observe request is for streaming purposes and understands that it needs to push the streaming media to the client 1702. For live streaming, the state change would be constant. Therefore the CoAP server 1704 in Observe needs to be able to identify the constant state change of a resource and sends notifications. For on-demand streaming, on the contrary, the media chunks may be already available at the server 1704. Therefore from the server’s perspective, there is no state change of the resource.

[0150] The CoAP streaming options can be used to resolve these issues. For example, if a CoAP client 1702 initiates a registration request for a resource, it indicates that it is for streaming by including a PlayMode option. The client 1702 can also include other streaming options to define the streaming operation.

[0151] In the second enhancement, the CoAP Observe draft also defines that the CoAP server 1704 follows a “best-effort approach for sending the current representation to the client after a state change”. “The server will do its best to keep the resource state observed by the client as closely in sync with the actual state as possible”. However, a client 1702 cannot rely on observing every single state that a resource might go through. For example, if the network is congested or the state changes more frequently than the network can handle, the server can skip notifications for any number of intermediate states.” Basically the Observe method provide “best effort” QoS. However streaming requires different level of QoS. To resolve this issue, the CoAP client 1702 should include QoS related streaming options in its Observe registration request.

[0152] Table 3 compares the different CoAP streaming mechanisms discussed above. Comparing to using CoAP Observe, dedicated streaming mechanisms provide more flexibility for CoAP to support the streaming feature since Observe was not designed for streaming purposes. However, for CoAP end points that only support Observe, it might still be used with modifications of its behavior.

[0153] Note the destination of the streaming operation can be a multicast address using the multicast method defined in RFC 7252, The Constrained Application Protocol (CoAP) and RFC 7390 Group Communication for the Constrained Application Protocol (CoAP). The streaming request messages can be addressed to the group URI. For example, if the “/streaming” URI is used, the streaming destination will be “/group multicast URI/streaming”. The streaming Options will apply to all group members.

Table 3 Comparison of Different CoAP Streaming Mechanisms

CoAP Streaming Mechanisms	Using “/streaming” URI	Using STREAM method	Using Enhanced CoAP Observe described herein
Designed for streaming operations	Yes	Yes	No. Not designed for streaming, but may be used for streaming purposes with modifications
Impact to the current CoAP base protocol	Use existing method with new Options	Introduced a new method	Use existing Observe method with new Options and new behavior of Observe
Target of a Request message	By default operates on a reserved URI; may include	By default operates on the default streaming resource.	Target a specific resource to Observe

	the URI of a specific media resource	May include the URI of a specific media resource	
Suitable streaming operations	Any, live streaming and on-demand streaming	Any, live streaming and on-demand streaming	Intermittent streaming since Observe pushes updates of a resource based on changes

[0154] The current CoAP block transfer mechanism can be combined together with the streaming mechanisms mentioned above. The current CoAP block transfer mechanism can be used to break down a media file into smaller media chunks and sent from one CoAP endpoint to another one. Figure 21 illustrates the procedure involving Block transfer. One of the streaming mechanisms is used for illustration purpose and the concept applies to other streaming mechanism as well.

[0155] The new mechanisms supporting streaming are indicated in Figure 20. These mechanisms can also apply transmitting resources of big sizes, not limited to streaming.

- Use the “/streaming” URI. Other mechanisms disclosed herein can be used as well.
- Use the streaming options so the client and server can define and exchange streaming specific parameters
- Trigger push operations. In the current CoAP Block operation, each time the client needs to retrieve a block, it needs to send a Request message to the server. In the combined operation, push operation is enabled as shown in Figure 21.

[0156] In Step 1 of Figure 21, the procedure is the same as illustrated in Figure 17. In the streaming request, the client indicates its request of media chunk size using the Block 2 option.

[0157] Step 2 – Step 4 of Figure 21 are the same as Steps 2-4 of Figure 17, the CoAP server 1704 sends media chunks. It uses the Block options to indicate the size, sequence, and the end of the media file. Note that the CoAP server 1704 will push media chunks to the client since it is streaming operation.

[0158] It is understood that the entities performing the steps illustrated in Figure 21 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 21 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 21. It is also

understood that any transmitting and receiving steps illustrated in Figure 21 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0159] Table 4 and Table 5 define exemplary new CoAP Options of one embodiment introduced to support CoAP streaming, following the format of the CoAP base protocol. The definitions of the Options are divided into two tables. Table 4 defines the Options and how they are used. Table 5 defines the format of the Options. When presented both in the request and response messages, the CoAP endpoints involved in the streaming operation can negotiate the settings so the streaming operation can be adaptive to the need of both endpoints. The Options in the following table are not exhaustive and can be extended or modified. The new options listed below define the essential parameters to be used to support streaming operations in one embodiment. More examples on how these Options can be used are discussed below Table 4.

Table 4 CoAP Streaming Options (part 1)

CoAP Streaming Options	Descriptions	Meaning in a Request	Meaning in a Response
StreamingType	Indicate the specific mechanism that the CoAP endpoint supports. Choices are: 1) STREAM method 2) streaming URI 3) Observe 4) None. "None" indicates the endpoint does not support streaming. All other Options should only be available if the end point supports streaming. An endpoint may support multiple mechanisms.	Indicate the streaming mechanism the endpoint that sends the Request supports	Indicate the streaming mechanism the endpoint that sends the Response supports
Duration	Time duration of the streaming operation.	If it is a request to trigger the streaming operation, it indicates how long the sender would receive the streaming operation. If it is a request to push streaming to the recipient, it indicates how long the push operation will last.	If the recipient does not agree on the Duration, it includes the Duration with a different value in the Response message to negotiate a different duration.
StartTime	The start time of a streaming operation. It can be used for delayed operation, or be used together with End Time to define the duration. If both "StartTime" and "EndTime" are used, "Duration" is not	If it is a request to trigger the streaming operation, it indicates the start time that the sender wants.	Indicates the StartTime that the sender of the Response wants.

	used.		
EndTime	The end time of a streaming operation. It is only used in pairs with the StartTime.	If it is a request to trigger the streaming operation, it indicates the end time that the sender wants.	In a Response containing media chunk, the Endtime indicates the time streaming finishes.
ChunkSize	The size of the media chunk per message contains. The server may process and produce the media in near real time so the media file was not already produced and chunk size can be changed. Or the server may have multiple media files with different chunk sizes.	Indicates the chunk size the sender of the Request supports.	Indicates the chunk size the sender of the Response supports.
TotalSize	The total size (e.g. bytes) of the media streaming. This can be used to control the overall content being streamed.	Indicates the total size the sender of the Request supports.	Indicates the total size the sender of the Response supports.
StreamingMediaType	Defines media type (codecs) used for the current streaming operation, such as wmv, mpeg.	If the sender of the Request is triggering streaming, it indicates the media type that it will support. If the Request is pushing media chunks, it indicates the type of the media chunk.	Indicates the media type of the media chunks that is sent.
PlayMode	Indicates the PLAY operation for streaming.	If Observe is used for streaming, the CoAP client includes this Option to indicate it is for streaming. It is also used to play after pause.	Used to resume playing after pause.
PauseMode	Indicates the PAUSE operation for streaming.	The sender of the request requires the pause of the streaming operation.	The sender of the response requires the pause of the streaming operation.
StopMode	Indicates the STOP operation for streaming.	The sender of the request requires the stop of the streaming operation.	The sender of the response requires the stop of the streaming operation.
FileDuration	Indicates the total duration in time (e.g. seconds) that the streaming media file the end point has. This Option is used with the "PositionTime". When an endpoint knows the FileDuration, it may specify a PositionTime to skip certain contents.	The sender of the request queries the FileDuration, or provides the value if it is a push request.	The sender of the response provides the value based on the request.
PositionTime	Indicates the position by time (e.g. in seconds) for rewind	The sender of the request to require	The sender of the response to require

	and fast forward.	playing from a specific time stamp in the media.	playing from a specific time stamp in the media.
PositionPercent	Indicates the relative position by percent (e.g 1-100) for rewind and fast forward.	The sender of the request to require playing from a specific relative position in the media.	The sender of the response to require playing from a specific relative position in the media.
Protocol	Indicates the protocols (UDP, TCP) to be used. Since UDP traffic may be blocked due to firewalls, if a device supports CoAP over TCP, TCP can be used for streaming.	The sender of the Request asks to switch to another protocol. For example, the sender may want to use CoAP/TCP for streaming.	The sender of the Response asks to switch to another protocol. For example, the sender may want to use CoAP/TCP for streaming.
Connection	Indicates if CoAP connection should be used.	The sender of the Request asks to enable CoAP connection.	The sender of the Response asks to enable CoAP connection.
Resolution	Indicates the video resolution. There can be predefined levels of resolution.	Indicates the resolution level that the sender of the request message requires.	Indicates the resolution level that the sender of the response message requires.
Quality	This is a generic parameter that defines the streaming quality. Specific levels of quality can be pre-defined.	Indicates the quality level that the sender of the request message requires.	Indicates the quality level that the sender of the response message requires.
Delay	Indicates the Delay time (e.g. in seconds) before the next chunk. Optionally or alternatively, delay variant (jitter) may be used	Indicates the Delay that the sender requires, or if it is a push operation, indicates the estimated delay the sender offers.	Same meaning as in the Request message, initiated by the CoAP server.
Reliability	This parameter indicates if reliability is required. This can trigger the CoAP client and server to use CoAP confirmable message model if reliability is required. Otherwise, CoAP non-confirmable model can be used to reduce message exchanges.	Indicates the requirement on reliability from the CoAP client.	Indicates the reliability can be offered from the CoAP server.
Intermittent	Indicates if the streaming operation is intermittent, which means the server will send the streaming content for a certain duration of time (defined by intermitInterval below). The Duration is defined as well, indicating the Duration of the streaming period.	The sender of the Request requires streaming to be sent intermittently (e.g. like the case c in Figure 15), or if it is push operation, it indicates the recipient to expect intermittent	Same meaning as in the Request message, initiated by the CoAP server.

		streaming contents.	
IntermitInterval	Indicates the time (e.g. in seconds) between each intermittent streaming. This option much be used together with the "Intermittent" option.	Indicates the interval between sesuctive intermittent streaming the sender requires, or that the sender offers in the push operation.	Same meaning as in the Request message, initiated by the CoAP server.
Buffer	Indicates if the streamed content is buffered or not.	In a Request to trigger streaming, the sender indicates if it will buffer the streamed content or not.	Not applicable
BufferSize	Indicates the buffer size (e.g. in the unit of bytes). It is only used when Buffer is set to "1 (Yes)".	Provides the buffer size to the server so that server can adapt its streaming parameters avoiding over load the sender's buffer.	Not applicable
Source	This only applies to a non-push operation. If this Option is not included, be default the source of the streaming should be the server of the Request. This Option is used to appoint a different endpoint as the source of the streaming. Note that it is assumed that the source can be addressed.	If this Option is included, the sender of the Request appoints a different endpoint as the source of the streaming.	Not applicable
Destination	This only applies to a non-push operation. The endpoint that sends the request can indicate one or more destinations (rather than itself), as the recipients of the streaming.	If this Option is included, the sender of the Request appoints one of more different destinations. So the recipient of the Request should stream to those endpoints.	Not applicable
StreamCharacteristics	This only applies to a non-push operation. This Option is used as a wild card string to describe and discover the stream file satisfying the characteristics.	The sender uses this Option as a filter to find the streaming files that match the characteristics.	Not applicable

[0160] The Options introduced in Table 4 can be used individually (unless indicated that it must be used with other Options) or combined to achieve adaptive streaming operations. This is important for IoT devices since their capacity can be limited to process streaming content and their conditions can change more often.

[0161] In one example, the CoAP client 1702 sends a streaming request with the “Buffer” Option set to “1” (Yes). After a while, the server 1704 is sending the contents very fast, with a short interval between chunks, and the buffer is getting full. The client 1702 sends a request including the Option “Delay”, which increases the interval as defined by the value of “Delay” between two messages carrying consecutive media chunks.

[0162] In another example, a CoAP endpoint residing on a constrained device receives streaming contents. At certain point, the device cannot process the streaming contents being pushed to it. The CoAP endpoint sends a request with the Option “Intermittent” set to “1” (Yes) and it also include the “IntermitInterval” and “Duration” to define how often and how long it will receive the streaming content each time. By doing so the client 1702 can regulate the streaming content it receives.

[0163] Table 4 also defines a set of play mode options. They work together for the case when the server has the media file ready for streaming (i.e. on demand, not real time). A CoAP client 1702 can start streaming by sending a request to the server, using the STREAM method. Then after a while the client 1702 may want to pause the operation for a period of time, for example, the client 1702 goes to sleep mode. The client 1702 can send another Request using the PauseMode Option. Upon receiving it, the server 1704 stops sending the media chunks, but the server 1704 will keep where the streaming was paused, awaiting for it to be resumed. The server 1704 may keep an internal timer of expiration for exception case if it does not receive a resume Request again. To resume the paused operation, the client sends a Request with “PlayMode”. In addition, the client 1702 may indicate where it wants to play by including a “PositionTime” or “PositionPercent” value.

[0164] Between the CoAP client 1702 and server 1704, they can also jump back and forth in the media content by using the “Position Time” and “PositionPercent”. These options work for the cases when the endpoint already has the media file created. The client 1702 can request to jump forward or backward in the media file by specifying a time (e.g. 10 seconds) or a relative position (e.g. 25% of the file).

[0165] Table 5 defines the Options in the same way as Table 4 in the CoAP protocol. Note that each option in Table 5 (or Table 4) can be added as a new link format or a new attribute of a resource being maintained by the Resource Directory (RD). In other words, when a CoAP endpoint (e.g. a client) registers its resources to RD, it can append those streaming-related options for the resource if it can be streamed. Later, another CoAP client can go to the RD to discover those streaming-related options for a particular resource.

Table 5 CoAP Streaming Options (part 2)

CoAP Streaming Options	Critical	Safe to forward	CacheKey	Repeat	Format	Length (byte)	Default
StreamingType	Y	Y	N	N	String	0-4	none
Duration	N	Y	N	N	Uint	0-8	none
StartTime	N	Y	N	N	String	0-8	none
EndTime	N	Y	N	N	String	0-8	none
ChunkSize	N	Y	N	N	Uint	0-4	none
TotalSize	N	Y	N	N	Uint	0-8	none
StreamingMediaType	N	Y	N	N	Unit	0-4	none
PlayMode	N	Y	N	N	String	0-1	none
PauseMode	N	Y	N	N	String	0-1	none
StopMode	N	Y	N	N	String	0-1	none
FileDuration	N	Y	N	N	Uint	0-8	none
PositionTime	N	Y	N	N	Uint	0-8	none
PositionPercent	N	Y	N	N	Uint	0-1	none
Protocol	N	Y	N	N	String	0-4	UDP
Connection	N	Y	N	N	String	0-1	0 (no CoAP connection)
Resolution	N	Y	N	N	Uint	0-4	none
Quality	N	Y	N	N	Uint	0-4	none
Delay	N	Y	N	N	Uint	0-4	none
Reliability	N	Y	N	N	Boolean	0-1	0 (indicates no reliability requirement)
Intermittent	N	Y	N	N	Boolean	0-1	0 (no intermittent streaming)
Buffer	N	Y	N	N	Boolean	0-1	0 (streamed content is not buffered)
BufferSize	N	Y	N	N	Uint	0-8	none
Source	N	Y	N	N	String	0-8	none
Destination	N	Y	N	N	String	0-8	none
StreamCharacteristics	N	Y	N	N	String	0-255	none

[0166] Figure 22 and Figure 23 illustrate examples using the above mechanism and Options to support streaming operations. The eHealth use case mentioned above with respect to Figure 13 is re-used in this example to show how the CoAP streaming mechanisms can resolve the issues mentioned earlier in this use case. The call flows also illustrate the adaptive nature of the mechanism. For example, as shown in the figures below, streaming operation can be started or ended, and the streaming parameters can be updated based on the streaming Options.

[0167] In Step 1 of Figure 22, the health center control machine 1306 supports CoAP. It initiates a CoAP Observe registration request for the home gateway 1304 at a patient's house. The home gateway 1304 acts as a CoAP proxy and sends the request to the sensor 1302 that the patient is wearing which can read the patient's heart beat.

[0168] The sensor 1302 sends notifications of multiple readings as defined by CoAP Observe. The home gateway 1304 forwards the results to the health center control machine 1306.

[0169] There is a video monitor 1308 installed at the patient's home. However, during a normal situation the monitor 1308 is in power saving mode and not used all the time. This can also reduce network traffic.

[0170] In Step 2 of Figure 22, the health center control machine 1306 (or health workers using the machine) analyzes the patient's heart beat readings over a period of time, and noticed that the heart beat is too high. The control machine 1306, based on its own logic, decides to trigger video monitoring to this patient.

[0171] In Step 3 of Figure 22, the health center control machine 1306 uses the mechanism discussed above to trigger streaming. Since it considers the patient's condition as serious, it requires streaming using CoAP connection by setting the "Connection" Option to "Yes". It also requires high resolution for video chunks by setting the Resolution Option to "High". Note that the Resolution setting may also be the absolute resolution value.

[0172] In Step 4 of Figure 22, the home gateway establishes a proxy based CoAP connection, using the mechanisms. The use of CoAP connections can improve the transmission reliability and efficiency at the CoAP layer. Note that before the home gateway can establish a CoAP connection with the video monitor, the video monitor must have waken up. It can be triggered by means of device triggering, such as by using infrared.

[0173] In Step 5 of Figure 22, the home gateway uses the mechanism discussed with respect to Figure 17 to trigger streaming from the video monitor.

[0174] In Steps 6 ~ 9 of Figure 22, the video monitor sends media chunks to the health center according to the configuration.

[0175] It is understood that the entities performing the steps illustrated in Figure 22 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 22 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 22. It is also understood that any transmitting and receiving steps illustrated in Figure 22 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0176] Figure 23 shows the procedure that health center adjusted streaming configuration and eventually terminated streaming.

[0177] In Step 1 of Figure 23, while video streaming is on (i.e. the patient is on high alert monitoring), the heart beat reader 1302 may continue to send heart beat values, or may be switched to power saving mode. Other sensors might be triggered (not shown) to enhance the overall monitoring level. The health center control machine 1306 analyzed the video chunks as well as other readings from the patient over a period of time, and made the assessment that the patient is getting better and not in a serious condition.

[0178] In Step 2 – Step 3 of Figure 23, the control machine 1306 initiates the change of streaming to low resolution. The home gateway 1304 configures the video monitor.

[0179] In Step 4 – Step 7 of Figure 23, the video monitor 1308 sends media chunks in low resolution.

[0180] In Step 8 of Figure 23, after a period of time, the streaming video results indicated that the patient is safe and no video streaming is needed.

[0181] In Step 9 of Figure 23, the control machine 1306 initiates the termination of streaming, using DELETE towards the /streaming location. Note that in an alternative way, the machine 1306 can send another GET /streaming request, and include “StopMode”

[0182] In Step 10 – 13 of Figure 23, the video streaming is deleted and the video monitor 1308 is back to power saving mode.

[0183] In Step 14 of Figure 23, since the video streaming is stopped, the home gateway 1304 tears down the CoAP connection.

[0184] It is understood that the entities performing the steps illustrated in Figure 23 are logical entities that may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of, and executing on a processor of, a network apparatus or computer system such as those illustrated in Figure 26C or Figure 26D. That is, the method(s) illustrated in Figure 23 may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as the apparatus or computer system illustrated in Figure 26C or Figure 26D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in Figure 23. It is also understood that any transmitting and receiving steps illustrated in Figure 23 may be performed by communication circuitry of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

[0185] Figure 24 shows an M2M/IoT device and gateway that each supports CoAP. In one embodiment, A GUI interface on the device or gateway can obtain CoAP operational

information via an internal API to the software stack running the CoAP endpoint. Therefore the GUI interface would be able to show the CoAP operations, for example, it can show that a STREAM request has been received, with certain streaming Options included. The GUI may use filter to select what information it will illustrate.

[0186] Interfaces, such as Graphical User Interfaces (GUIs), can be used to assist user to control and/or configure functionalities related to CoAP streaming mechanisms. Figure 25 is a diagram that illustrates an interface 2500 that allows a user to enable and monitor COAP streaming. It is to be understood that interface 2500 can be produced using displays such as those shown in Figures 26C-D described below.

Example M2M/IoT/WoT Communication System

[0187] The various techniques described herein may be implemented in connection with hardware, firmware, software or, where appropriate, combinations thereof. Such hardware, firmware, and software may reside in apparatuses located at various nodes of a communication network. The apparatuses may operate singly or in combination with each other to effect the methods described herein. As used herein, the terms “apparatus,” “network apparatus,” “node,” “device,” and “network node” may be used interchangeably.

[0188] The term “service layer” refers to a functional layer within a network service architecture. Service layers are typically situated above the application protocol layer such as HTTP, CoAP or MQTT and provide value added services to client applications. The service layer also provides an interface to core networks at a lower resource layer, such as for example, a control layer and transport/access layer. The service layer supports multiple categories of (service) capabilities or functionalities including a service definition, service runtime enablement, policy management, access control, and service clustering. Recently, several industry standards bodies, *e.g.*, oneM2M, have been developing M2M service layers to address the challenges associated with the integration of M2M types of devices and applications into deployments such as the Internet/Web, cellular, enterprise, and home networks. A M2M service layer can provide applications and/or various devices with access to a collection of or a set of the above mentioned capabilities or functionalities, supported by the service layer, which can be referred to as a CSE or SCL. A few examples include but are not limited to security, charging, data management, device management, discovery, provisioning, and connectivity management which can be commonly used by various applications. These capabilities or functionalities are made available to such various applications via APIs which make use of message formats, resource structures and resource representations defined by the M2M service layer. The CSE or SCL is a functional entity that may be implemented by hardware and/or software and that

provides (service) capabilities or functionalities exposed to various applications and/or devices (i.e., functional interfaces between such functional entities) in order for them to use such capabilities or functionalities.

[0189] Figure 26A is a diagram of an example machine-to machine (M2M), Internet of Things (IoT), or Web of Things (WoT) communication system 10 in which one or more disclosed embodiments may be implemented. Generally, M2M technologies provide building blocks for the IoT/WoT, and any M2M device, M2M gateway, M2M server, or M2M service platform may be a component or node of the IoT/WoT as well as an IoT/WoT service layer, etc. Communication system 10 can be used to implement functionality of the disclosed embodiments and can include functionality and logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500.

[0190] As shown in Figure 26A, the M2M/ IoT/WoT communication system 10 includes a communication network 12. The communication network 12 may be a fixed network (e.g., Ethernet, Fiber, ISDN, PLC, or the like) or a wireless network (e.g., WLAN, cellular, or the like) or a network of heterogeneous networks. For example, the communication network 12 may be comprised of multiple access networks that provide content such as voice, data, video, messaging, broadcast, or the like to multiple users. For example, the communication network 12 may employ one or more channel access methods, such as code division multiple access (CDMA), time division multiple access (TDMA), frequency division multiple access (FDMA), orthogonal FDMA (OFDMA), single-carrier FDMA (SC-FDMA), and the like. Further, the communication network 12 may comprise other networks such as a core network, the Internet, a sensor network, an industrial control network, a personal area network, a fused personal network, a satellite network, a home network, or an enterprise network for example.

[0191] As shown in Figure 26A, the M2M/ IoT/WoT communication system 10 may include the Infrastructure Domain and the Field Domain. The Infrastructure Domain refers to the network side of the end-to-end M2M deployment, and the Field Domain refers to the area networks, usually behind an M2M gateway. The Field Domain and Infrastructure Domain may both comprise a variety of different network nodes (e.g., servers, gateways, device, and the like). For example, the Field Domain may include M2M gateways 14 and terminal devices 18. It will be appreciated that any number of M2M gateway devices 14 and M2M terminal devices 18 may be included in the M2M/ IoT/WoT communication system 10 as desired. Each of the M2M gateway devices 14 and M2M terminal devices 18 are configured to transmit and receive signals, using communications circuitry, via the communication network 12 or direct radio link. A M2M

gateway 14 allows wireless M2M devices (e.g. cellular and non-cellular) as well as fixed network M2M devices (e.g., PLC) to communicate either through operator networks, such as the communication network 12 or direct radio link. For example, the M2M terminal devices 18 may collect data and send the data, via the communication network 12 or direct radio link, to an M2M application 20 or other M2M devices 18. The M2M terminal devices 18 may also receive data from the M2M application 20 or an M2M terminal device 18. Further, data and signals may be sent to and received from the M2M application 20 via an M2M service layer 22, as described below. M2M terminal devices 18 and gateways 14 may communicate via various networks including, cellular, WLAN, WPAN (e.g., Zigbee, 6LoWPAN, Bluetooth), direct radio link, and wireline for example.

[0192] Exemplary M2M terminal devices 18 include, but are not limited to, tablets, smart phones, medical devices, temperature and weather monitors, connected cars, smart meters, game consoles, personal digital assistants, health and fitness monitors, lights, thermostats, appliances, garage doors and other actuator-based devices, security devices, and smart outlets.

[0193] Referring to Figure 26B, the illustrated M2M service layer 22 in the field domain provides services for the M2M application 20, M2M gateway devices 14, and M2M terminal devices 18 and the communication network 12. Communication network 12 can be used to implement functionality of the disclosed embodiments and can include functionality and logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500. The M2M service layer 22 may be implemented by one or more servers, computers, devices, virtual machines (e.g. cloud/ storage farms, etc.) or the like, including for example the devices illustrated in figures 26C and 26D described below. It will be understood that the M2M service layer 22 may communicate with any number of M2M applications, M2M gateways 14, M2M terminal devices 18, and communication networks 12 as desired. The M2M service layer 22 may be implemented by one or more nodes of the network, which may comprises servers, computers, devices, or the like. The M2M service layer 22 provides service capabilities that apply to M2M terminal devices 18, M2M gateways 14, and M2M applications 20. The functions of the M2M service layer 22 may be implemented in a variety of ways, for example as a web server, in the cellular core network, in the cloud, etc.

[0194] Similar to the illustrated M2M service layer 22, there is the M2M service layer 22' in the Infrastructure Domain. M2M service layer 22' provides services for the M2M application 20' and the underlying communication network 12' in the infrastructure domain. M2M service layer 22' also provides services for the M2M gateways 14 and M2M terminal

devices 18 in the field domain. It will be understood that the M2M service layer 22' may communicate with any number of M2M applications, M2M gateways and M2M devices. The M2M service layer 22' may interact with a service layer by a different service provider. The M2M service layer 22' by one or more nodes of the network, which may comprises servers, computers, devices, virtual machines (e.g., cloud computing/storage farms, etc.) or the like.

[0195] Referring also to Figure 26B, the M2M service layers 22 and 22' provide a core set of service delivery capabilities that diverse applications and verticals can leverage. These service capabilities enable M2M applications 20 and 20' to interact with devices and perform functions such as data collection, data analysis, device management, security, billing, service/device discovery etc. Essentially, these service capabilities free the applications of the burden of implementing these functionalities, thus simplifying application development and reducing cost and time to market. The service layers 22 and 22' also enable M2M applications 20 and 20' to communicate through various networks 12 and 12' in connection with the services that the service layers 22 and 22' provide.

[0196] The methods of the present application may be implemented as part of a service layer 22 and 22'. The service layer 22 and 22' is a software middleware layer that supports value-added service capabilities through a set of Application Programming Interfaces (APIs) and underlying networking interfaces. Both ETSI M2M and oneM2M use a service layer that may contain the connection methods of the present application. ETSI M2M's service layer is referred to as the Service Capability Layer (SCL). The SCL may be implemented within an M2M device (where it is referred to as a device SCL (DSCL)), a gateway (where it is referred to as a gateway SCL (GSCL)) and/or a network node or apparatus (where it is referred to as a network SCL (NSCL)). The oneM2M service layer supports a set of Common Service Functions (CSFs) (i.e. service capabilities). An instantiation of a set of one or more particular types of CSFs is referred to as a Common Services Entity (CSE) which can be hosted on different types of network nodes (e.g. infrastructure node, middle node, application-specific node). Further, connection methods of the present application can implemented as part of an M2M network that uses a Service Oriented Architecture (SOA) and/or a resource-oriented architecture (ROA) to access services such as the connection methods of the present application.

[0197] In some embodiments, M2M applications 20 and 20' may be used in conjunction with the disclosed systems and methods.. The M2M applications 20 and 20' may include the applications that interact with the UE or gateway and may also be used in conjunction with other disclosed systems and methods.

[0198] In one embodiment, the logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500 may be hosted within a M2M service layer instance hosted by an M2M node, such as an M2M server, M2M gateway, or M2M device, as shown in Figure 26B. For example, the logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500 may comprise an individual service capability within the M2M service layer instance or as a sub-function within an existing service capability.

[0199] The M2M applications 20 and 20' may include applications in various industries such as, without limitation, transportation, health and wellness, connected home, energy management, asset tracking, and security and surveillance. As mentioned above, the M2M service layer, running across the devices, gateways, servers and other nodes of the system, supports functions such as, for example, data collection, device management, security, billing, location tracking/geofencing, device/service discovery, and legacy systems integration, and provides these functions as services to the M2M applications 20 and 20'.

[0200] Generally, the service layers 22 and 22' define a software middleware layer that supports value-added service capabilities through a set of Application Programming Interfaces (APIs) and underlying networking interfaces. Both the ETSI M2M and oneM2M architectures define a service layer. ETSI M2M's service layer is referred to as the Service Capability Layer (SCL). The SCL may be implemented in a variety of different nodes of the ETSI M2M architecture. For example, an instance of the service layer may be implemented within an M2M device (where it is referred to as a device SCL (DSCL)), a gateway (where it is referred to as a gateway SCL (GSCL)) and/or a network node (where it is referred to as a network SCL (NSCL)). The oneM2M service layer supports a set of Common Service Functions (CSFs) (i.e., service capabilities). An instantiation of a set of one or more particular types of CSFs is referred to as a Common Services Entity (CSE) which can be hosted on different types of network nodes (e.g. infrastructure node, middle node, application-specific node). The Third Generation Partnership Project (3GPP) has also defined an architecture for machine-type communications (MTC). In that architecture, the service layer, and the service capabilities it provides, are implemented as part of a Service Capability Server (SCS). Whether embodied in a DSCL, GSCL, or NSCL of the ETSI M2M architecture, in a Service Capability Server (SCS) of the 3GPP MTC architecture, in a CSF or CSE of the oneM2M architecture, or in some other node of a network, an instance of the service layer may be implemented as a logical entity (e.g., software,

computer-executable instructions, and the like) executing either on one or more standalone nodes in the network, including servers, computers, and other computing devices or nodes, or as part of one or more existing nodes. As an example, an instance of a service layer or component thereof may be implemented in the form of software running on a network node (e.g., server, computer, gateway, device or the like) having the general architecture illustrated in Figure 26C or Figure 26D described below.

[0201] Further, logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500 can be implemented as part of an M2M network that uses a Service Oriented Architecture (SOA) and/or a Resource-Oriented Architecture (ROA) to access services of the present application.

[0202] Figure 26C is a block diagram of an example hardware/software architecture of a M2M network node 30, such as an M2M device 18, an M2M gateway 14, an M2M server, or the like. The node 30 can execute or include logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500. The device 30 can be part of an M2M network as shown in Figure 26A-B or part of a non-M2M network. As shown in Figure 26C, the M2M node 30 may include a processor 32, non-removable memory 44, removable memory 46, a speaker/microphone 38, a keypad 40, a display, touchpad, and/or indicators 42, a power source 48, a global positioning system (GPS) chipset 50, and other peripherals 52. The node 30 may also include communication circuitry, such as a transceiver 34 and a transmit/receive element 36. It will be appreciated that the M2M node 30 may include any sub-combination of the foregoing elements while remaining consistent with an embodiment. This node may be a node that implements the SMSF functionality described herein.

[0203] The processor 32 may be a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Array (FPGAs) circuits, any other type of integrated circuit (IC), a state machine, and the like. In general, the processor 32 may execute computer-executable instructions stored in the memory (e.g., memory 44 and/or memory 46) of the node in order to perform the various required functions of the node. For example, the processor 32 may perform signal coding, data processing, power control, input/output processing, and/or any other functionality that enables the M2M node 30 to operate in a wireless or wired environment. The processor 32 may run

application-layer programs (e.g., browsers) and/or radio access-layer (RAN) programs and/or other communications programs. The processor 32 may also perform security operations such as authentication, security key agreement, and/or cryptographic operations, such as at the access-layer and/or application layer for example.

[0204] As shown in Figure 26C, the processor 32 is coupled to its communication circuitry (e.g., transceiver 34 and transmit/receive element 36). The processor 32, through the execution of computer executable instructions, may control the communication circuitry in order to cause the node 30 to communicate with other nodes via the network to which it is connected. In particular, the processor 32 may control the communication circuitry in order to perform the transmitting and receiving steps described herein and in the claims. While Figure 26C depicts the processor 32 and the transceiver 34 as separate components, it will be appreciated that the processor 32 and the transceiver 34 may be integrated together in an electronic package or chip.

[0205] The transmit/receive element 36 may be configured to transmit signals to, or receive signals from, other M2M nodes, including M2M servers, gateways, device, and the like. For example, in an embodiment, the transmit/receive element 36 may be an antenna configured to transmit and/or receive RF signals. The transmit/receive element 36 may support various networks and air interfaces, such as WLAN, WPAN, cellular, and the like. In an embodiment, the transmit/receive element 36 may be an emitter/detector configured to transmit and/or receive IR, UV, or visible light signals, for example. In yet another embodiment, the transmit/receive element 36 may be configured to transmit and receive both RF and light signals. It will be appreciated that the transmit/receive element 36 may be configured to transmit and/or receive any combination of wireless or wired signals.

[0206] In addition, although the transmit/receive element 36 is depicted in Figure 26C as a single element, the M2M node 30 may include any number of transmit/receive elements 36. More specifically, the M2M node 30 may employ MIMO technology. Thus, in an embodiment, the M2M node 30 may include two or more transmit/receive elements 36 (e.g., multiple antennas) for transmitting and receiving wireless signals.

[0207] The transceiver 34 may be configured to modulate the signals that are to be transmitted by the transmit/receive element 36 and to demodulate the signals that are received by the transmit/receive element 36. As noted above, the M2M node 30 may have multi-mode capabilities. Thus, the transceiver 34 may include multiple transceivers for enabling the M2M node 30 to communicate via multiple RATs, such as UTRA and IEEE 802.11, for example.

[0208] The processor 32 may access information from, and store data in, any type of suitable memory, such as the non-removable memory 44 and/or the removable memory 46. For

example, the processor 32 may store session context in its memory, as described above. The non-removable memory 44 may include random-access memory (RAM), read-only memory (ROM), a hard disk, or any other type of memory storage device. The removable memory 46 may include a subscriber identity module (SIM) card, a memory stick, a secure digital (SD) memory card, and the like. In other embodiments, the processor 32 may access information from, and store data in, memory that is not physically located on the M2M node 30, such as on a server or a home computer. The processor 32 may be configured to control lighting patterns, images, or colors on the display or indicators 42 to reflect the status of an M2M service layer session migration or sharing or to obtain input from a user or display information to a user about the node's session migration or sharing capabilities or settings. In another example, the display may show information with regard to a session state. The current disclosure defines a RESTful user/application API in the oneM2M embodiment. A graphical user interface, which may be shown on the display, may be layered on top of the API to allow a user to interactively establish and manage an E2E session, or the migration or sharing thereof, via the underlying service layer session functionality described herein.

[0209] The processor 32 may receive power from the power source 48, and may be configured to distribute and/or control the power to the other components in the M2M node 30. The power source 48 may be any suitable device for powering the M2M node 30. For example, the power source 48 may include one or more dry cell batteries (e.g., nickel-cadmium (NiCd), nickel-zinc (NiZn), nickel metal hydride (NiMH), lithium-ion (Li-ion), etc.), solar cells, fuel cells, and the like.

[0210] The processor 32 may also be coupled to the GPS chipset 50, which is configured to provide location information (e.g., longitude and latitude) regarding the current location of the M2M node 30. It will be appreciated that the M2M node 30 may acquire location information by way of any suitable location-determination method while remaining consistent with an embodiment.

[0211] The processor 32 may further be coupled to other peripherals 52, which may include one or more software and/or hardware modules that provide additional features, functionality and/or wired or wireless connectivity. For example, the peripherals 52 may include various sensors such as an accelerometer, biometrics (e.g., figure print) sensors, an e-compass, a satellite transceiver, a digital camera (for photographs or video), a universal serial bus (USB) port or other interconnect interfaces, a vibration device, a television transceiver, a hands free headset, a Bluetooth® module, a frequency modulated (FM) radio unit, a digital music player, a media player, a video game player module, an Internet browser, and the like.

[0212] The node 30 may be embodied in other apparatuses or devices, such as a sensor, consumer electronics, a wearable device such as a smart watch or smart clothing, a medical or eHealth device, a robot, industrial equipment, a drone, a vehicle such as a car, truck, train, or airplane. The node 30 may connect to other components, modules, or systems of such apparatuses or devices via one or more interconnect interfaces, such as an interconnect interface that may comprise one of the peripherals 52. Alternately, the node 30 may comprise apparatuses or devices, such as a sensor, consumer electronics, a wearable device such as a smart watch or smart clothing, a medical or eHealth device, a robot, industrial equipment, a drone, a vehicle such as a car, truck, train, or airplane.

[0213] Figure 26D is a block diagram of an exemplary computing system 90 which may also be used to implement one or more nodes of an M2M network, such as an M2M server, gateway, device, or other node. Computing system 90 may comprise a computer or server and may be controlled primarily by computer readable instructions, which may be in the form of software, wherever, or by whatever means such software is stored or accessed. Computing system 90 can execute or include logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500. Computing system 90 can be an M2M device, user equipment, gateway, UE/GW or any other nodes including nodes of the mobile care network, service layer network application provider, terminal device 18 or an M2M gateway device 14 for example. Such computer readable instructions may be executed within a processor, such as central processing unit (CPU) 91, to cause computing system 90 to do work. In many known workstations, servers, and personal computers, central processing unit 91 is implemented by a single-chip CPU called a microprocessor. In other machines, the central processing unit 91 may comprise multiple processors. Coprocessor 81 is an optional processor, distinct from main CPU 91, that performs additional functions or assists CPU 91. CPU 91 and/or coprocessor 81 may receive, generate, and process data related to the disclosed systems and methods for E2E M2M service layer sessions, such as receiving session credentials or authenticating based on session credentials.

[0214] In operation, CPU 91 fetches, decodes, and executes instructions, and transfers information to and from other resources via the computer's main data-transfer path, system bus 80. Such a system bus connects the components in computing system 90 and defines the medium for data exchange. System bus 80 typically includes data lines for sending data, address lines for sending addresses, and control lines for sending interrupts and for operating the system bus. An example of such a system bus 80 is the PCI (Peripheral Component Interconnect) bus.

[0215] Memories coupled to system bus 80 include random access memory (RAM) 82 and read only memory (ROM) 93. Such memories include circuitry that allows information to be stored and retrieved. ROMs 93 generally contain stored data that cannot easily be modified. Data stored in RAM 82 can be read or changed by CPU 91 or other hardware devices. Access to RAM 82 and/or ROM 93 may be controlled by memory controller 92. Memory controller 92 may provide an address translation function that translates virtual addresses into physical addresses as instructions are executed. Memory controller 92 may also provide a memory protection function that isolates processes within the system and isolates system processes from user processes. Thus, a program running in a first mode can access only memory mapped by its own process virtual address space; it cannot access memory within another process's virtual address space unless memory sharing between the processes has been set up.

[0216] In addition, computing system 90 may contain peripherals controller 83 responsible for communicating instructions from CPU 91 to peripherals, such as printer 94, keyboard 84, mouse 95, and disk drive 85.

[0217] Display 86, which is controlled by display controller 96, is used to display visual output generated by computing system 90. Such visual output may include text, graphics, animated graphics, and video. Display 86 may be implemented with a CRT-based video display, an LCD-based flat-panel display, gas plasma-based flat-panel display, or a touch-panel. Display controller 96 includes electronic components required to generate a video signal that is sent to display 86.

[0218] Further, computing system 90 may contain communication circuitry, such as for example a network adaptor 97, that may be used to connect computing system 90 to an external communications network, such as network 12 of Figure 26A and Figure 26B, to enable the computing system 90 to communicate with other nodes of the network.

[0219] User equipment (UE) can be any device used by an end-user to communicate. It can be a hand-held telephone, a laptop computer equipped with a mobile broadband adapter, or any other device. For example, the UE can be implemented as the M2M terminal device 18 of Figures 26 A-B or the device 30 of Figure 26 C.

[0220] It is understood that any or all of the systems, methods, and processes described herein may be embodied in the form of computer executable instructions (i.e., program code) stored on a computer-readable storage medium which instructions, when executed by a machine, such as a node of an M2M network, including for example an M2M server, gateway, device or the like, perform and/or implement the systems, methods and processes described herein. Specifically, any of the steps, operations or functions described above, including the operations

of the gateway, UE, UE/GW, or any of the nodes of the mobile core network, service layer or network application provider, may be implemented in the form of such computer executable instructions. Logical entities such as CoAP client 1702, CoAP server 1704, heartbeat monitor 1302, video monitor 1308, home gateway 1304 and health center 1306 as well as logical entities to produce interfaces such as interface 2500 may be embodied in the form of the computer executable instructions stored on a computer-readable storage medium. Computer readable storage media include both volatile and nonvolatile, removable and non-removable media implemented in any non-transitory (i.e., tangible or physical) method or technology for storage of information, but such computer readable storage media do not include signals. Computer readable storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other tangible or physical medium which can be used to store the desired information and which can be accessed by a computer.

[0221] In describing preferred embodiments of the subject matter of the present disclosure, as illustrated in the Figures, specific terminology is employed for the sake of clarity. The claimed subject matter, however, is not intended to be limited to the specific terminology so selected, and it is to be understood that each specific element includes all technical equivalents that operate in a similar manner to accomplish a similar purpose.

[0222] This written description uses examples to disclose the invention, including the best mode, and also to enable any person skilled in the art to practice the invention, including making and using any devices or systems and performing any incorporated methods. The patentable scope of the invention is defined by the claims, and may include other examples that occur to those skilled in the art. Such other examples are intended to be within the scope of the claims if they have elements that do not differ from the literal language of the claims, or if they include equivalent elements with insubstantial differences from the literal language of the claims.

What is Claimed:

1. A Constrained Application Protocol (CoAP) client comprising a processor and a memory, the CoAP client further including computer-executable instructions stored in the memory of the CoAP client which, when executed by the processor of the CoAP client, cause the CoAP client to:

send a GET request to a dedicated CoAP streaming URI at a CoAP server; and
receive streaming content from the CoAP server.

2. The CoAP client of claim 1, wherein the streaming content is received in chunks.

3. The CoAP client of claim 1, wherein the CoAP streaming URI is obtained from a CoAP Resource Directory (RD).

4. The CoAP client of claim 1, wherein the GET request includes at least one option.

5. The CoAP client of claim 4, wherein the option indicates the streaming media type that the CoAP client supports.

6. A Constrained Application Protocol (CoAP) client comprising a processor and a memory, the CoAP client further including computer-executable instructions stored in the memory of the CoAP client which, when executed by the processor of the CoAP client, cause the CoAP client to:

push a first chunk of streaming media to a CoAP server using a PUT command to a CoAP streaming URI; and

push a second chunk of streaming media to the CoAP server using the PUT command to the CoAP streaming URI.

7. The CoAP client of claim 6, wherein the CoAP streaming URI is obtained from a CoAP Resource Directory (RD).

8. The CoAP client of claim 6, wherein the PUT command includes at least one option.

9. The CoAP client of claim 8, wherein the option indicates a media type.

10. The CoAP client of claim 6, wherein the CoAP server makes the streaming media available for another client.

11. A Constrained Application Protocol (CoAP) client comprising a processor and a memory, the CoAP client further including computer-executable instructions stored in the memory of the CoAP client which, when executed by the processor of the CoAP client, cause the CoAP client to:

- send a request to CoAP server using a CoAP stream method; and
- receive streaming content from the CoAP server.

12. The CoAP client of claim 11, wherein the streaming content is received in chunks.

13. The CoAP client of claim 11, wherein the request includes a CoAP streaming URI .

14. The CoAP client of claim 11, wherein the request includes at least one option.

15. The CoAP client of claim 14, wherein the option indicates the streaming media type that the CoAP client supports.

16. A Constrained Application Protocol (CoAP) client comprising a processor and a memory, the CoAP client further including computer-executable instructions stored in the memory of the CoAP client which, when executed by the processor of the CoAP client, cause the CoAP client to:

- push a first chunk of streaming media to a CoAP server using a CoAP stream method;
- and
- push a second chunk of streaming media to the CoAP server using the a CoAP stream method.

17. The CoAP client of claim 16, wherein the first chunk includes at least one option.

18. The CoAP client of claim 17, wherein the option indicates a media type.

19. The CoAP client of claim 16, wherein the CoAP server makes the streaming media available for another client.

20. A Resource Directory comprising a processor and a memory, the Resource Directory further including computer-executable instructions stored in the memory of the Resource Directory which, when executed by the processor of the Resource Directory, cause the Resource Directory to:

receive an indication of a streaming CoAP resource from a CoAP client, the indication including at least one option; and

provide the indication of the streaming CoAP resource including the at least one option to additional CoAP clients.

21. The Resource Directory of claim 20, wherein the at least one option is one of “StreamingType”, “StreamingMediaType”, “StreamCharacteristics” and “Reliability”.

22. The Resource Directory of claim 20, wherein the at least one option includes a connection option that indicates if a CoAP connection should be used.

23. The Resource Directory of claim 20, wherein the at least one option includes a connection option that indicates if a CoAP connection should be used.

24. The Resource Directory of claim 20, wherein the at least one option includes a StreamingType option that indicates a specific mechanism that the CoAP client supports.

25. The Resource Directory of claim 20, wherein the at least one option includes a reliability option that indicates if reliability is required.

26. The Resource Directory of claim 20, wherein the at least one option includes an intermittent option that indicates if the streaming operation is intermittent.

27. The Resource Directory of claim 20, wherein the at least one option includes a destination option that indicates one or more destinations other than the CoAP client.

28. The Resource Directory of claim 20, wherein the at least one option includes a streaming media option that indicates the streaming media type that the CoAP client supports

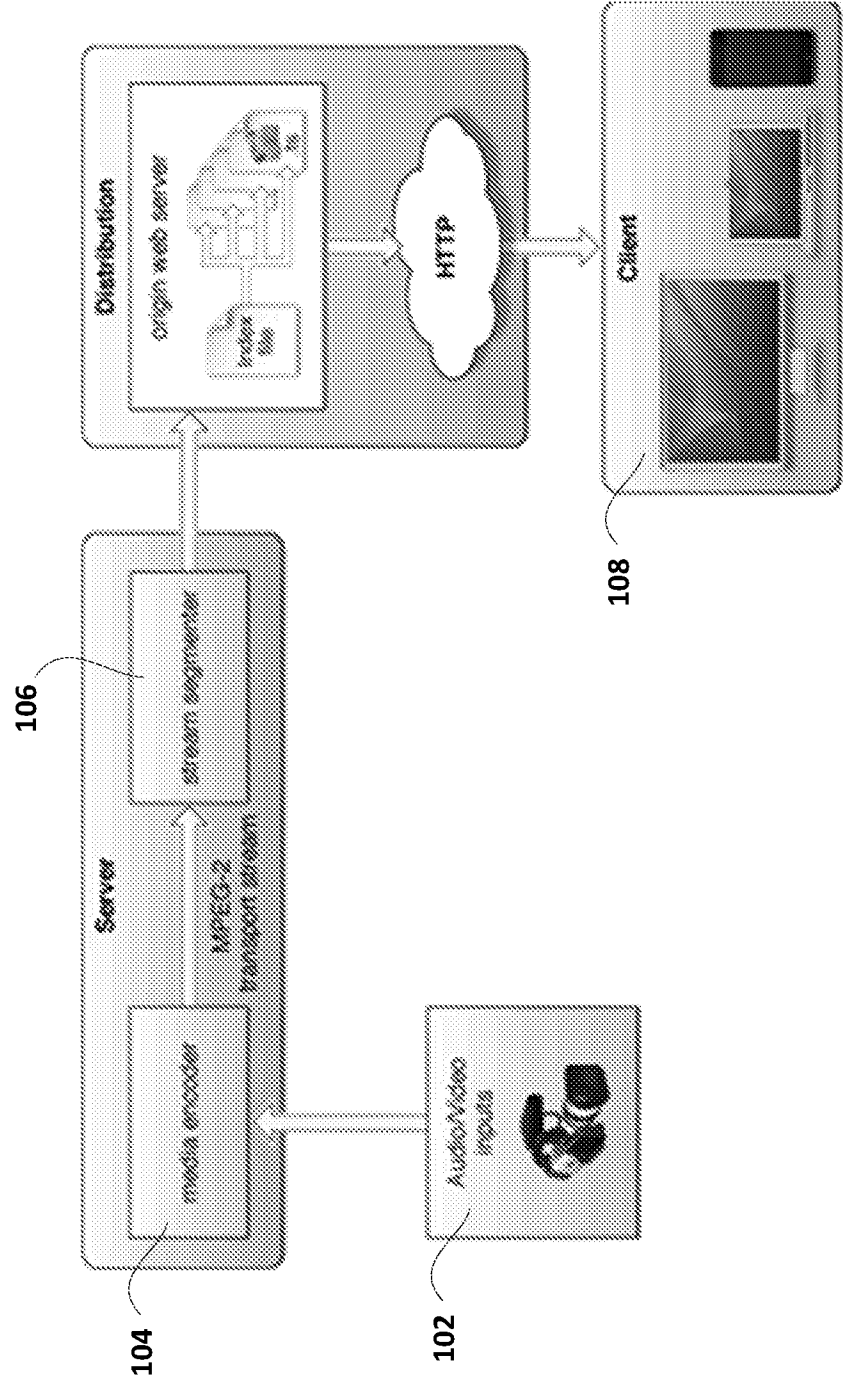


FIG. 1

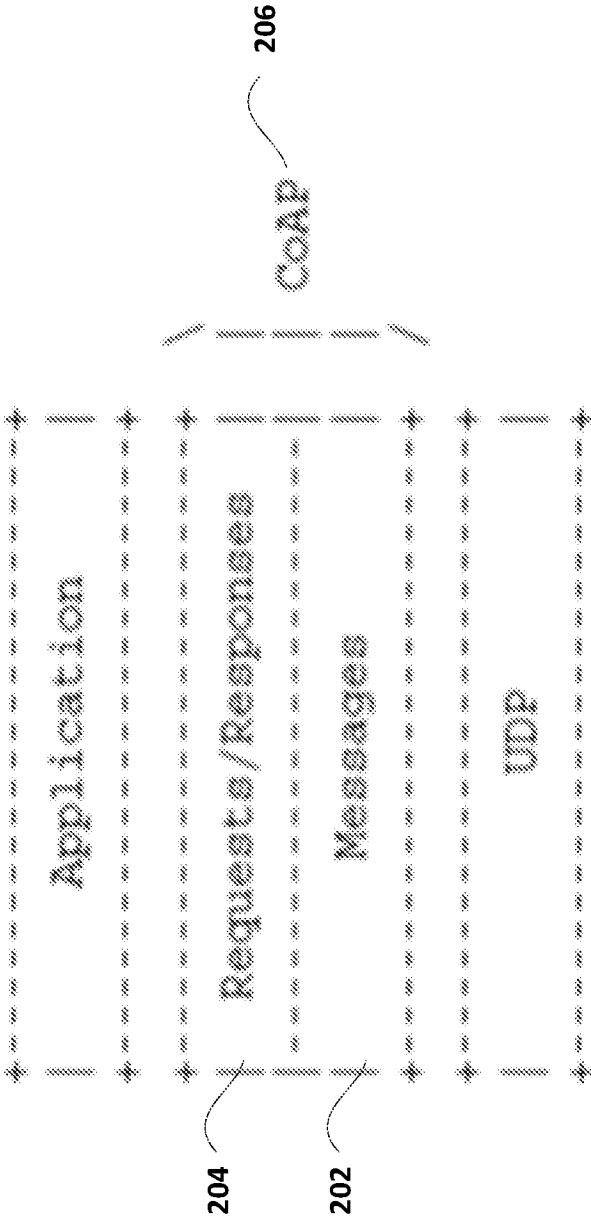


FIG. 2

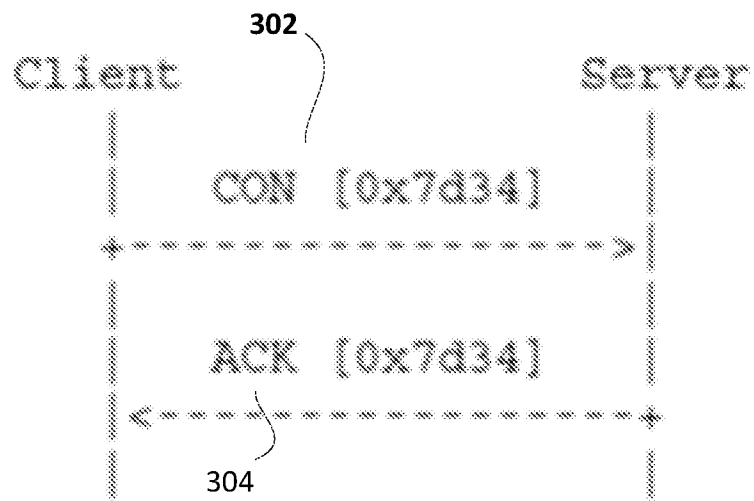


FIG. 3

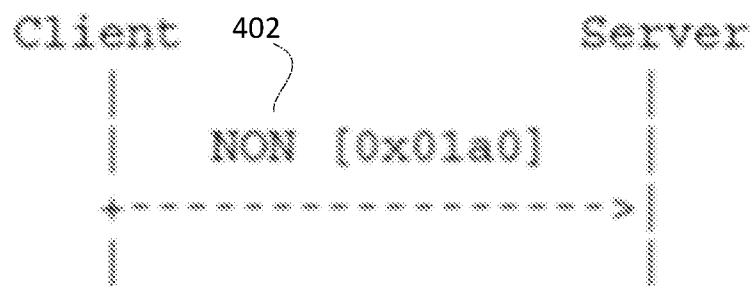


FIG. 4

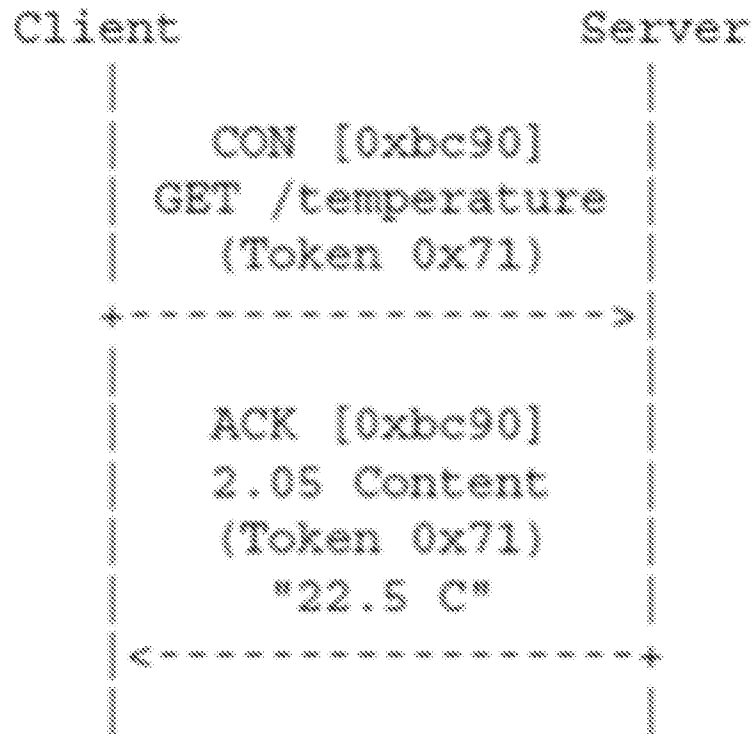


FIG. 5

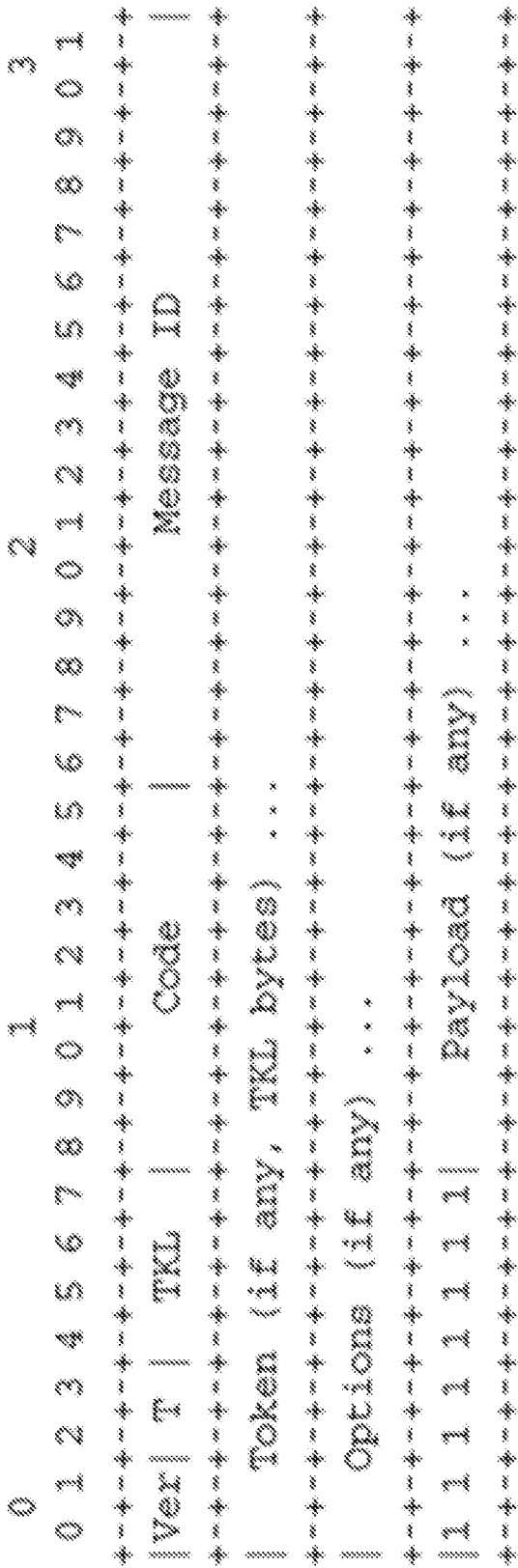


FIG. 6

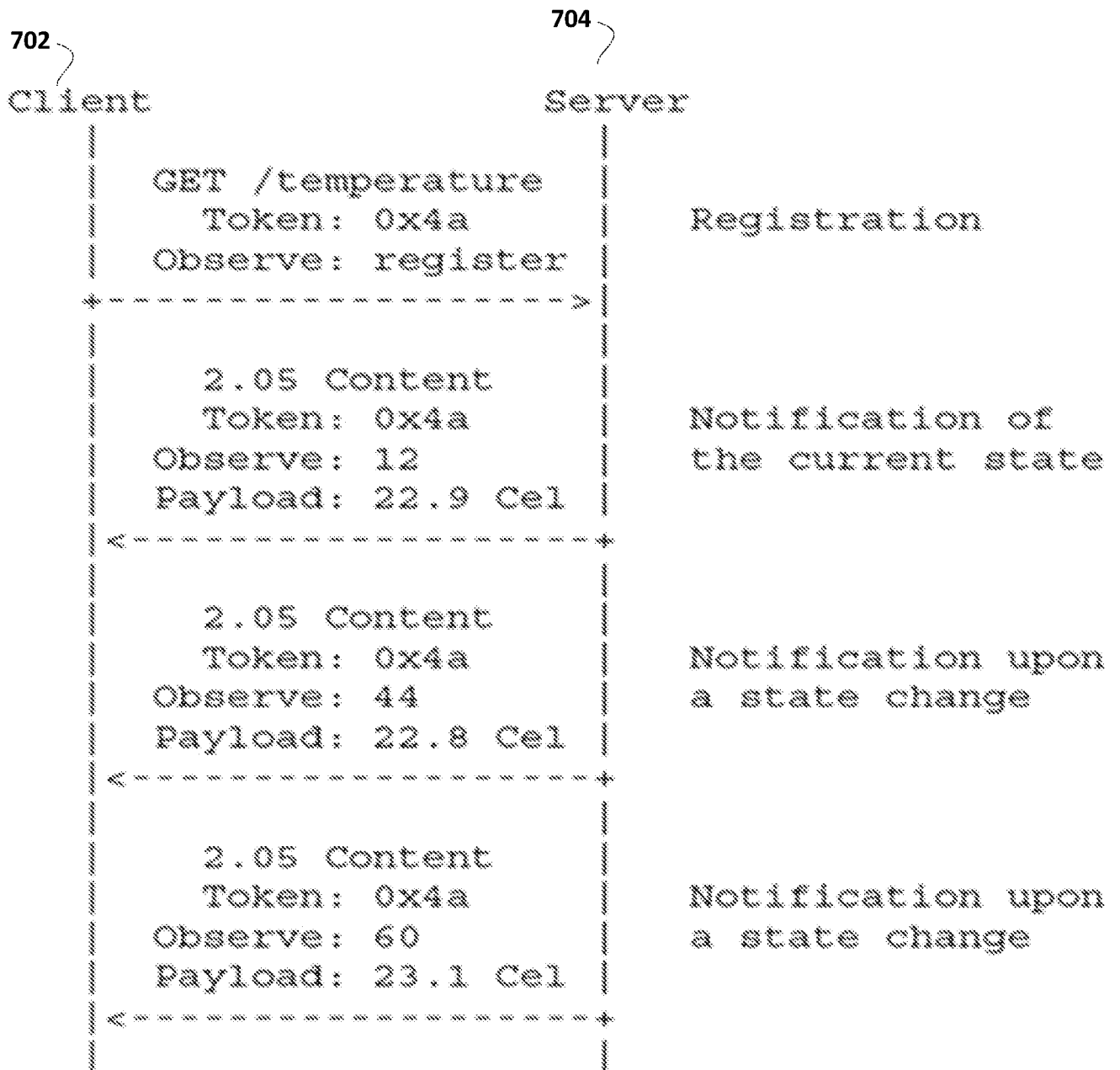
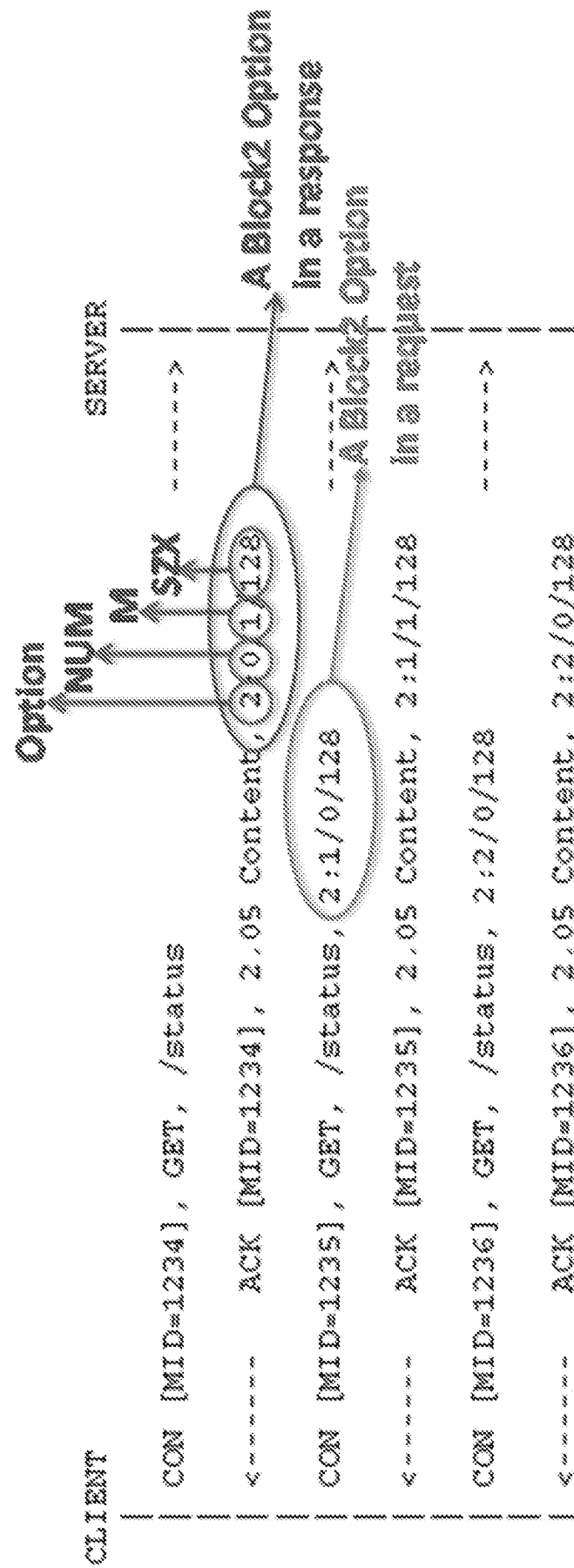


FIG. 7


$$\frac{\infty}{\mathbb{G}} \mathbb{F}$$

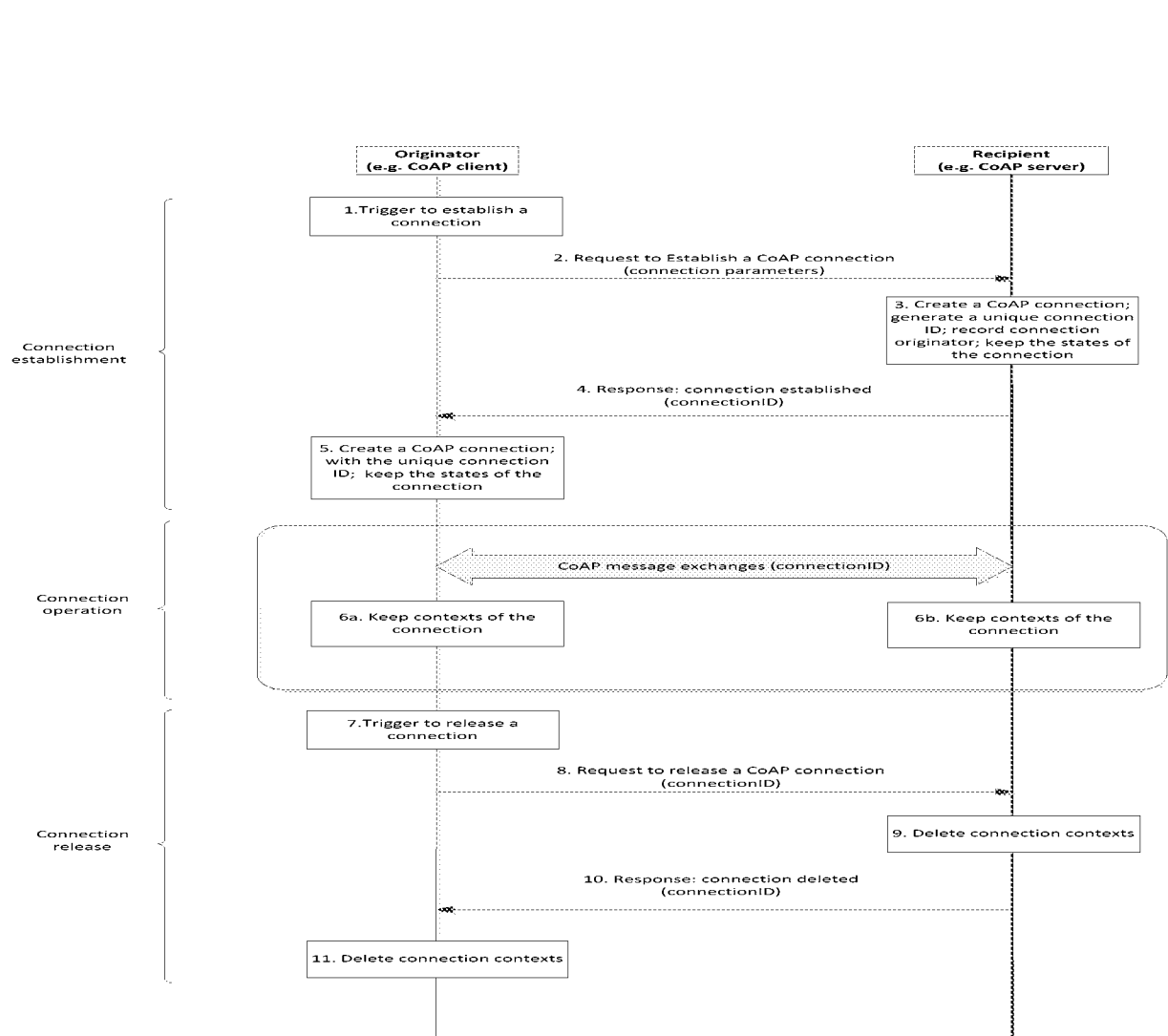


FIG. 9

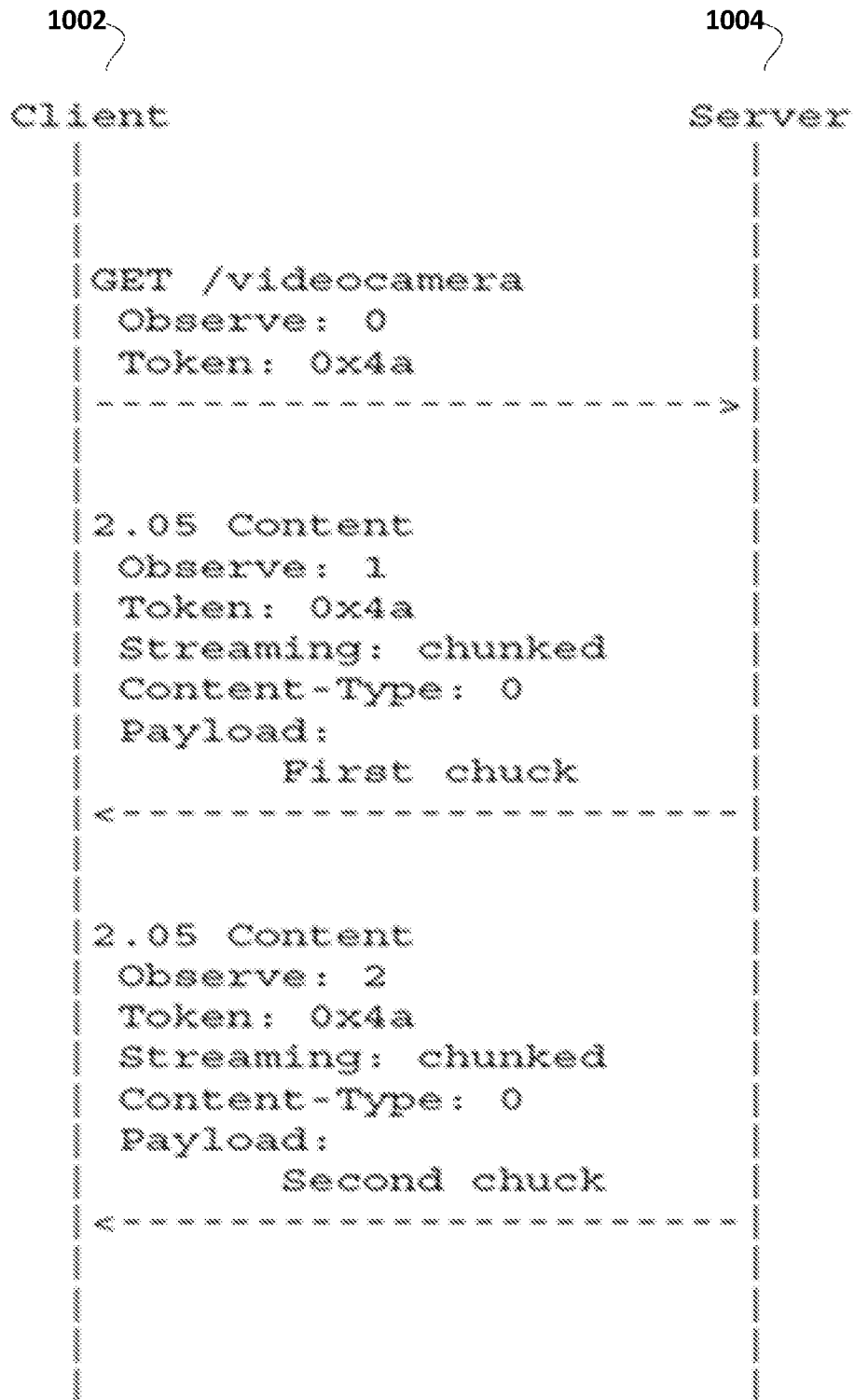


FIG. 10

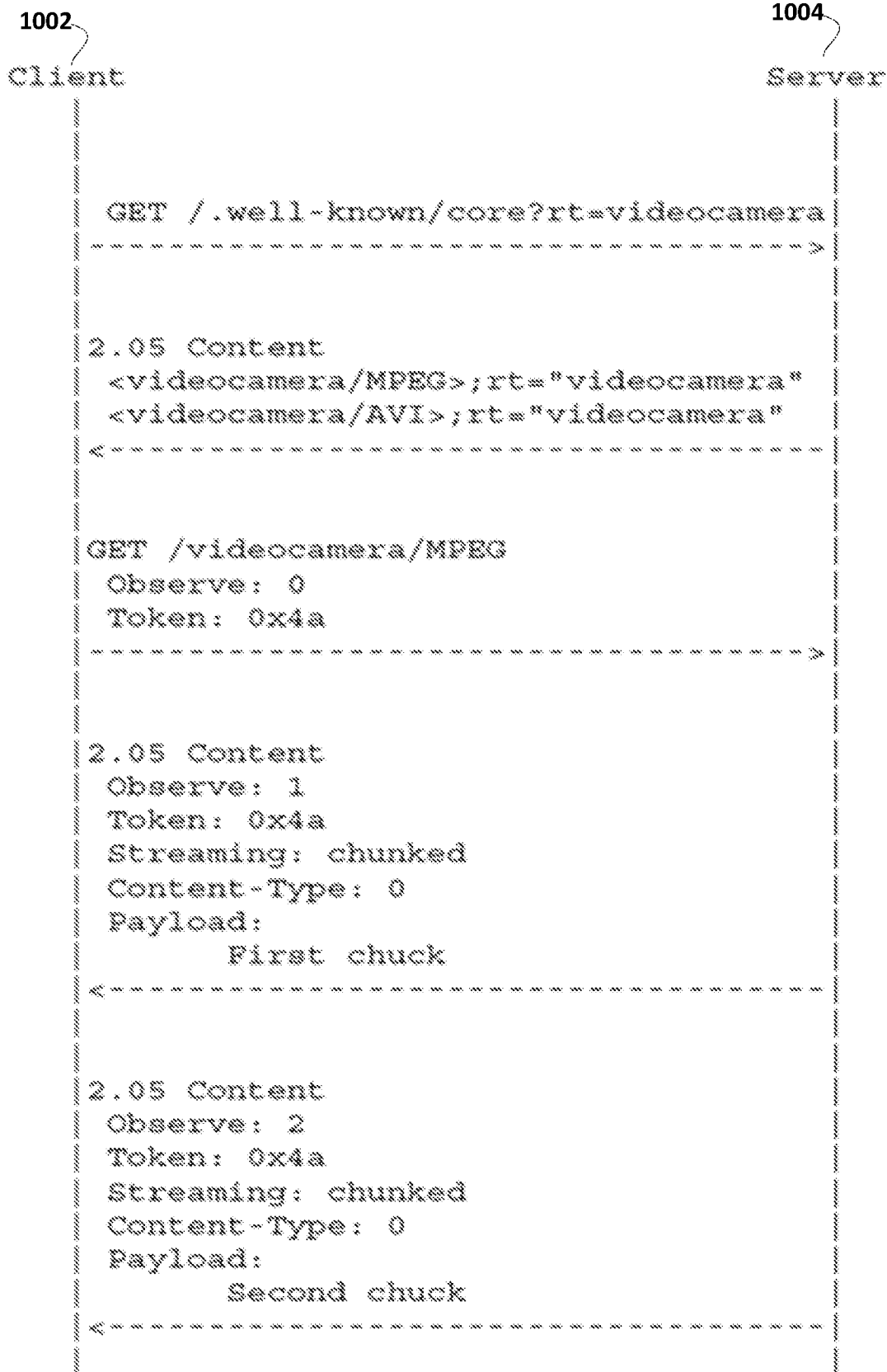


FIG. 11

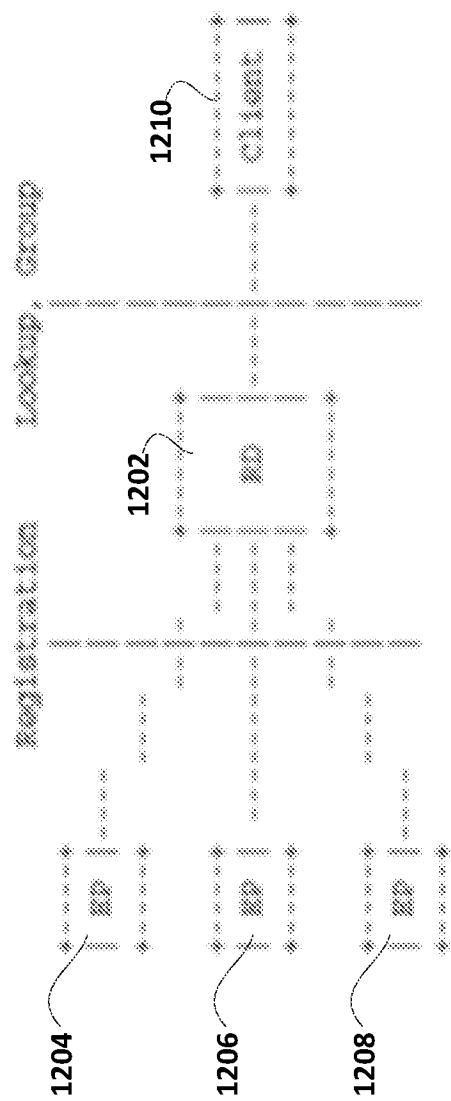


FIG. 12

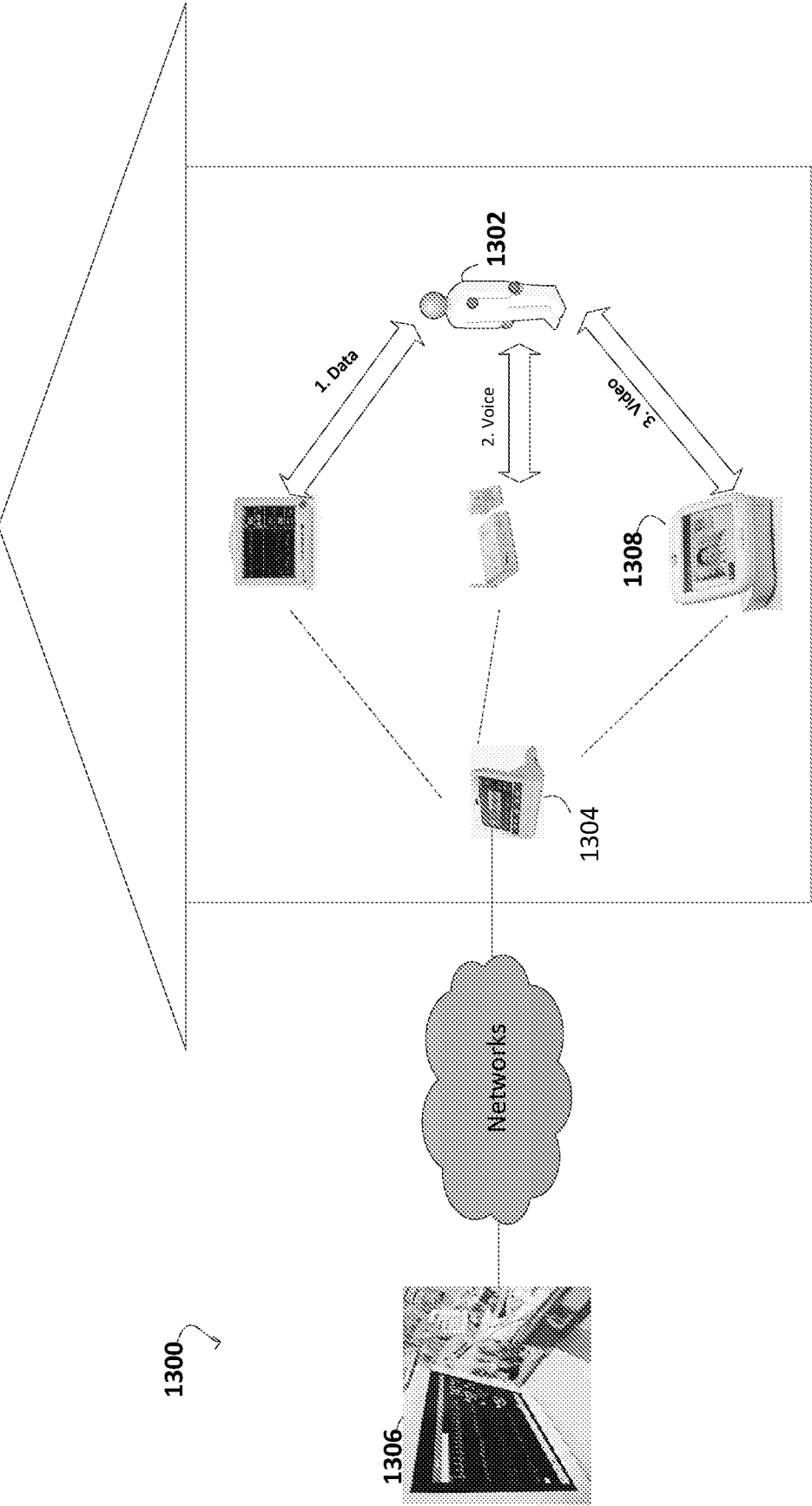


FIG. 13

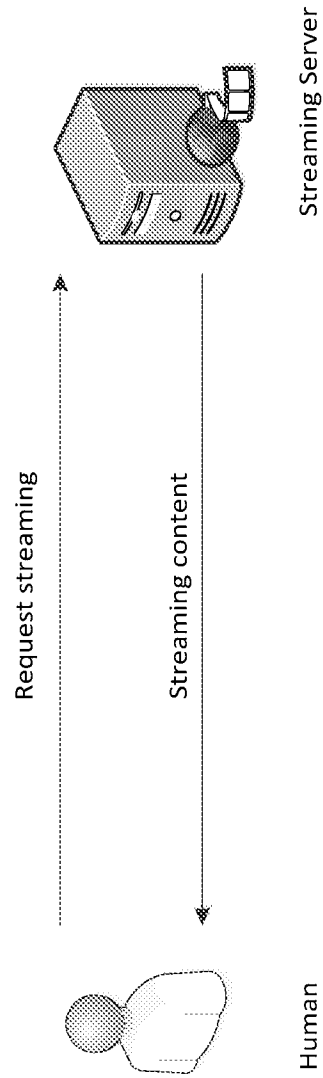


FIG. 14

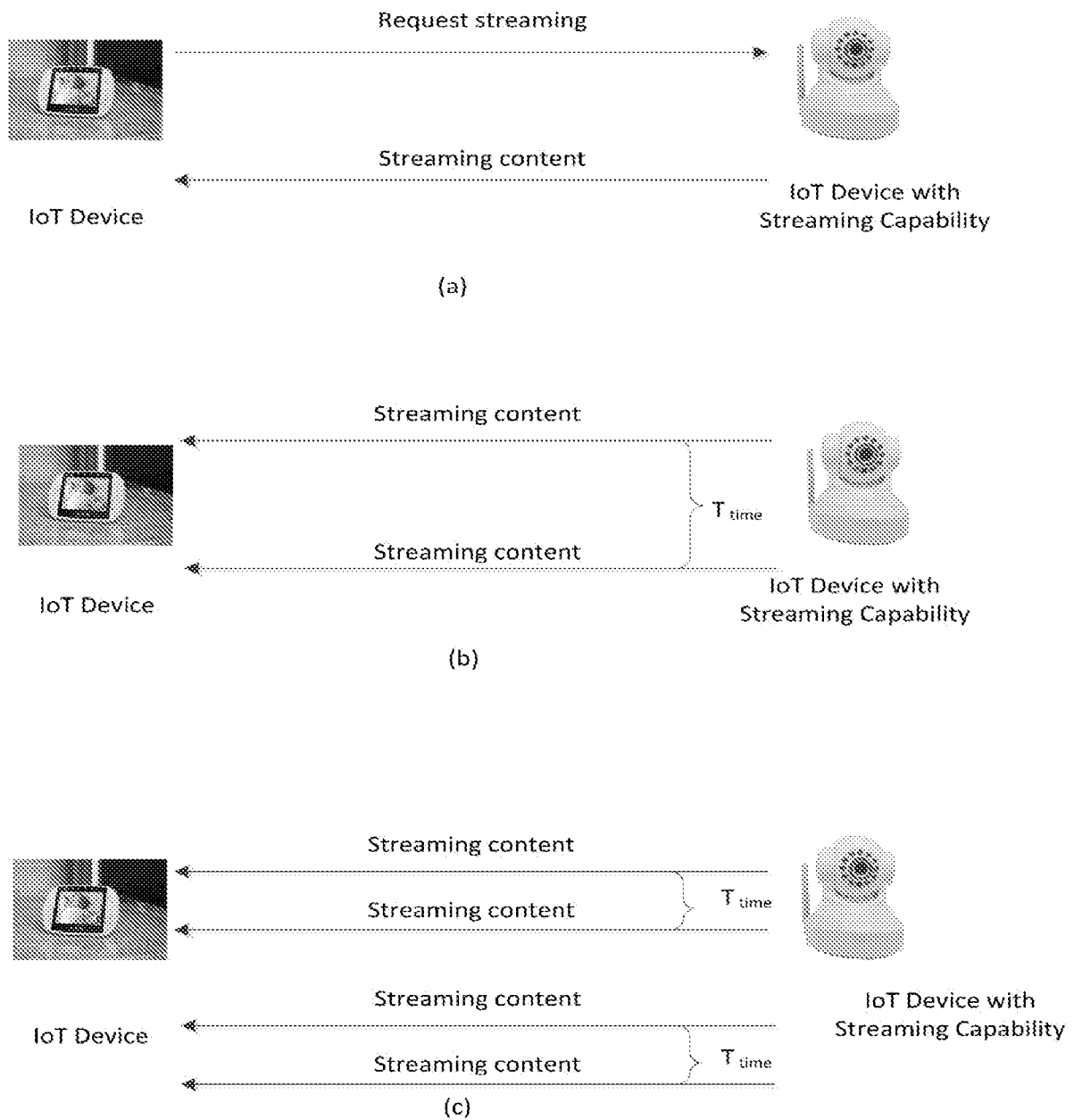


FIG. 15

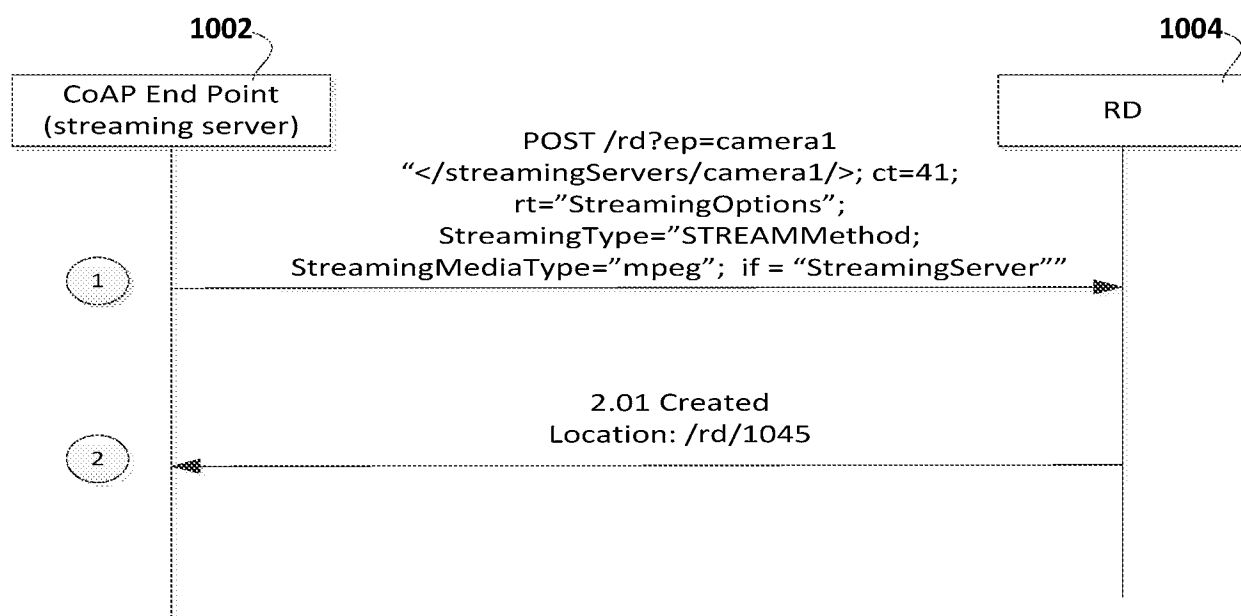


FIG. 16

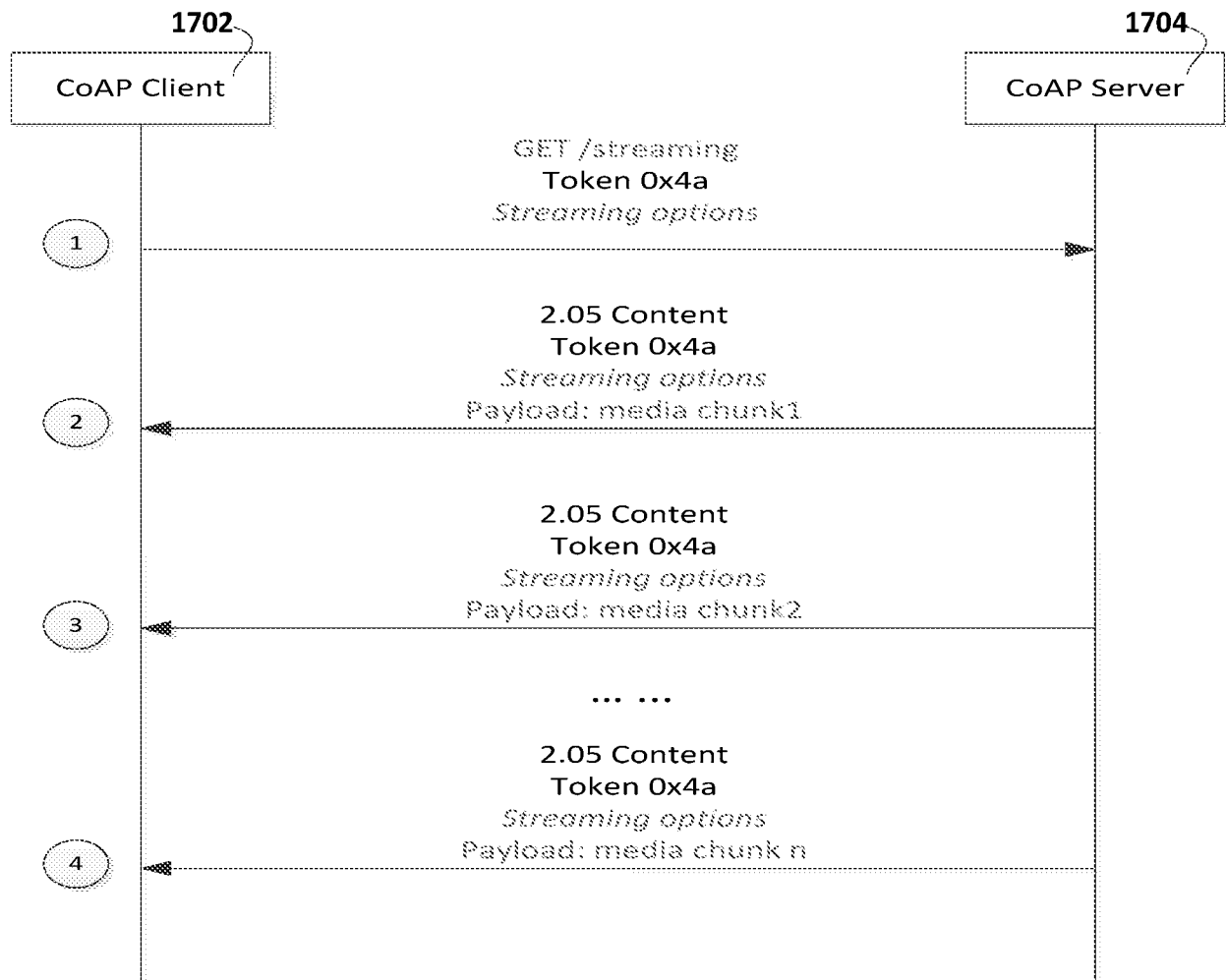


FIG. 17

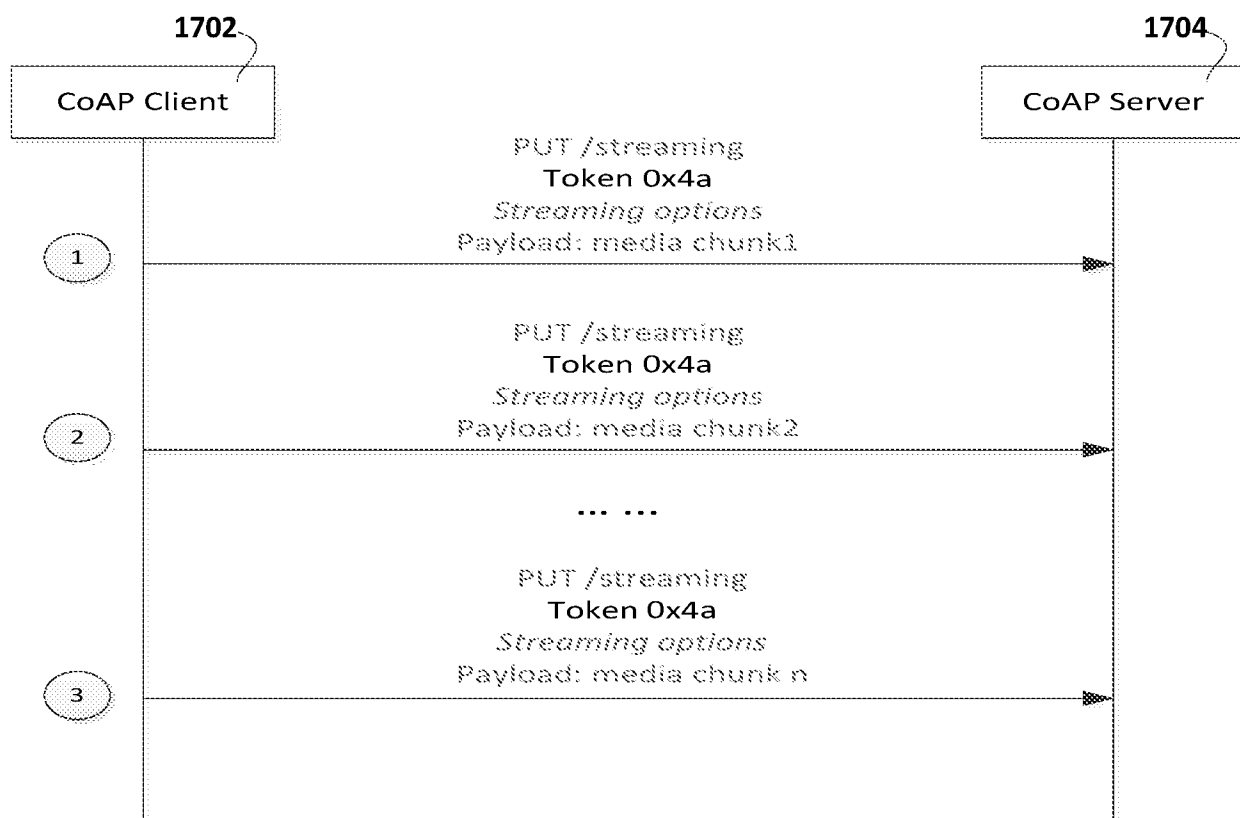


FIG. 18

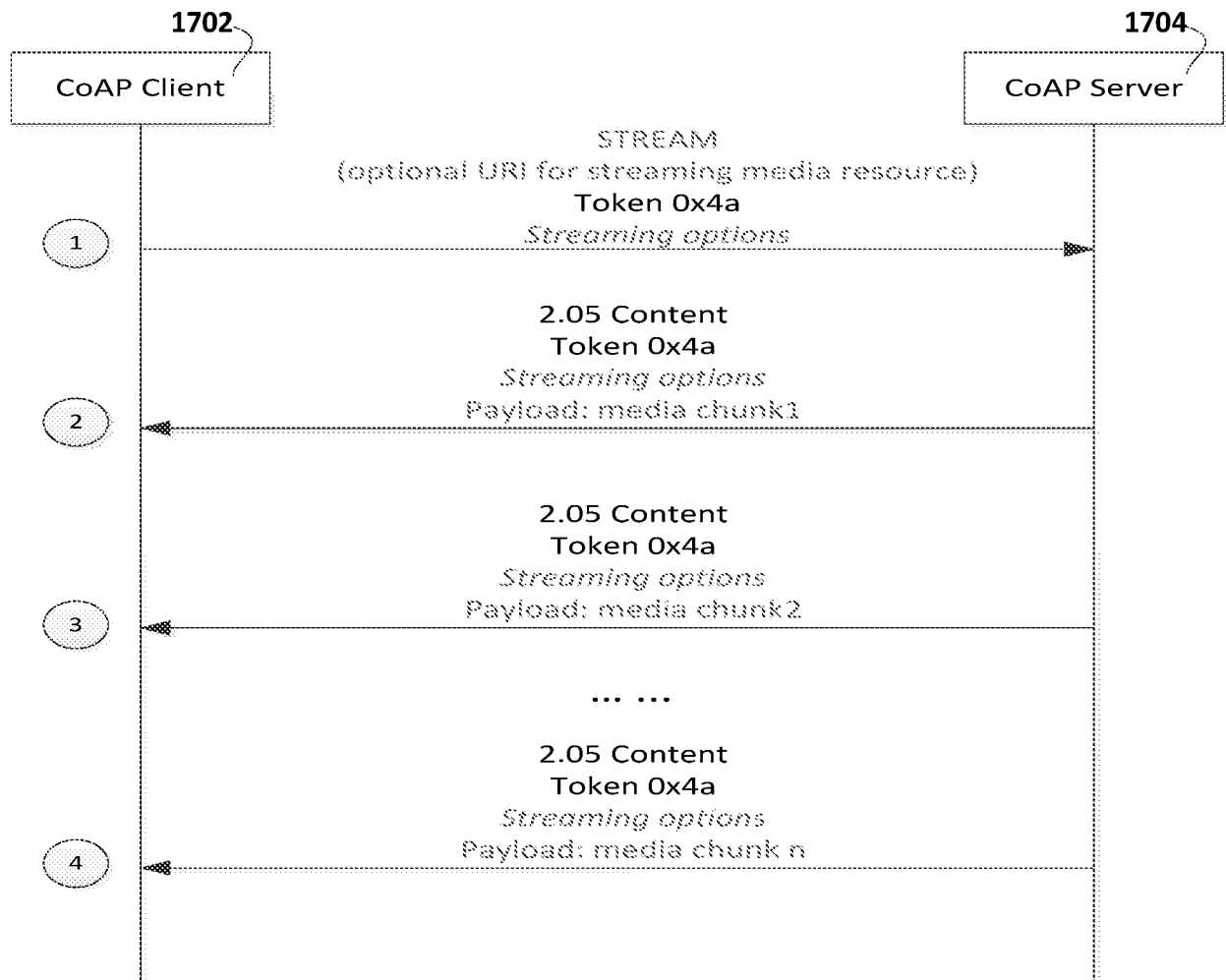


FIG. 19

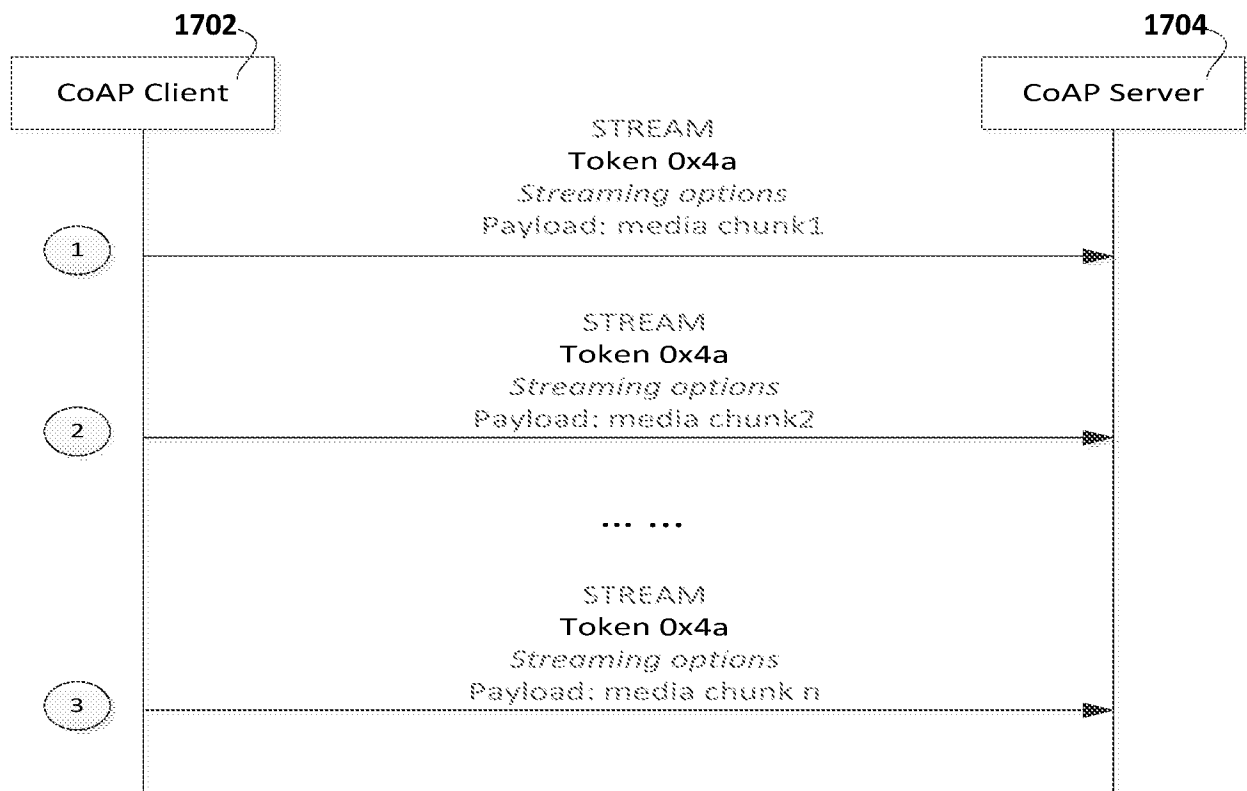


FIG. 20

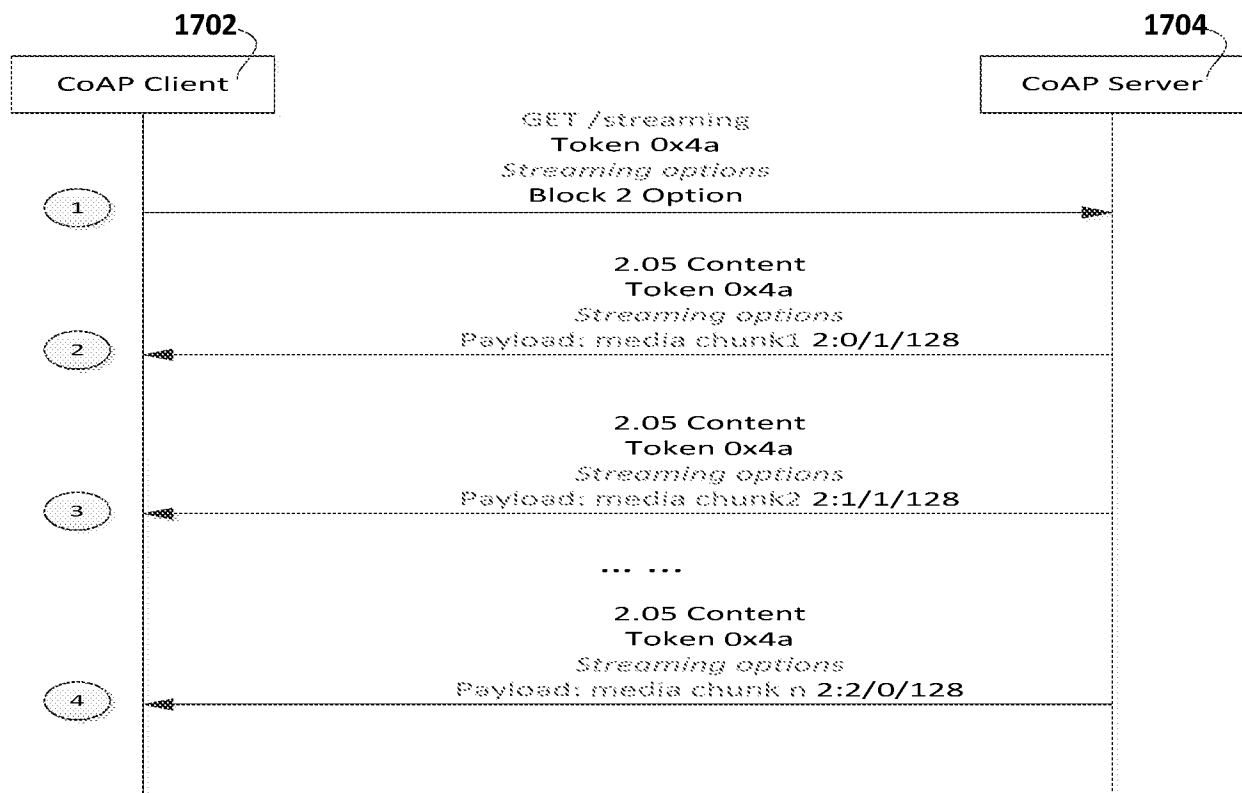


FIG. 21

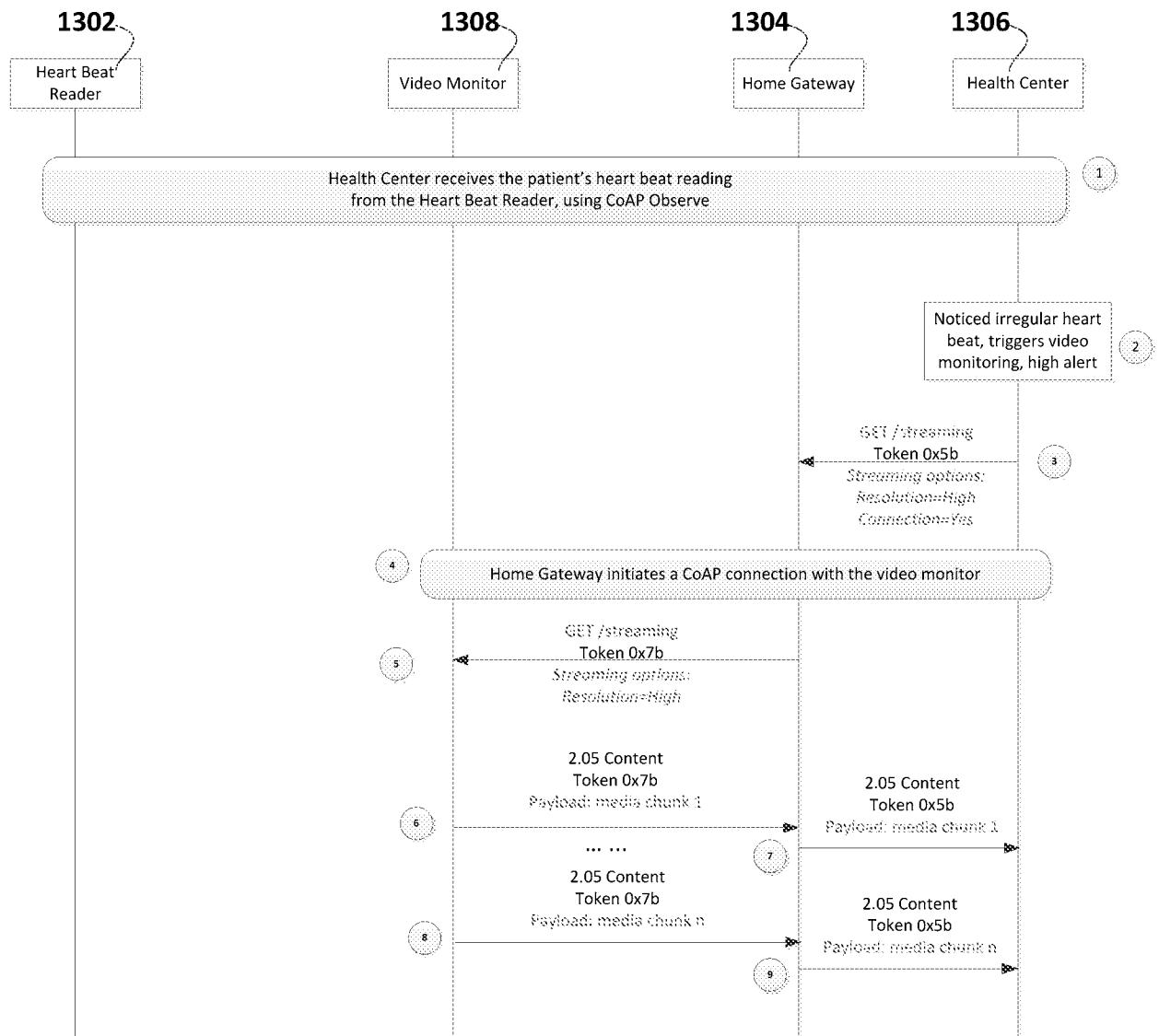


FIG. 22

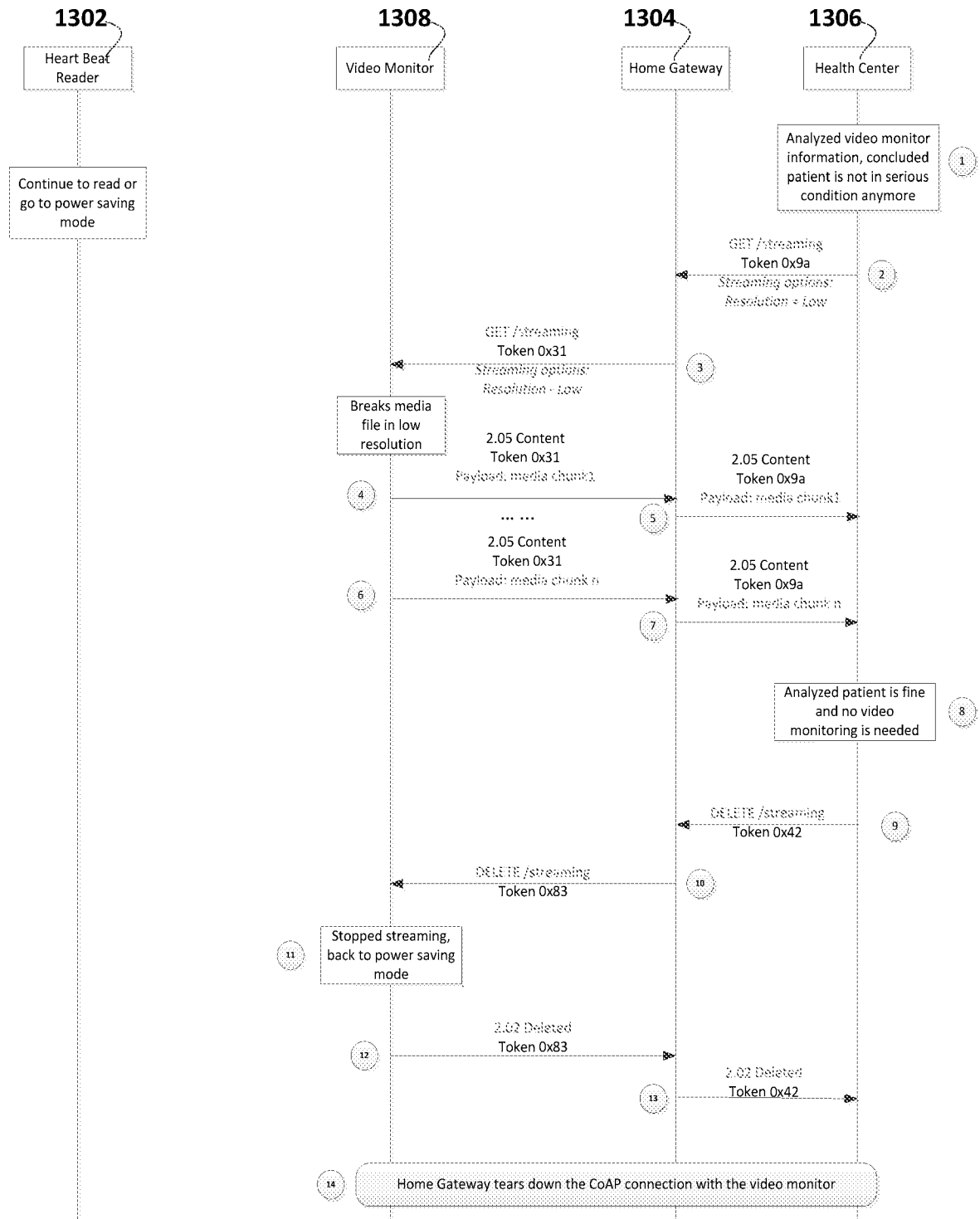


FIG. 23

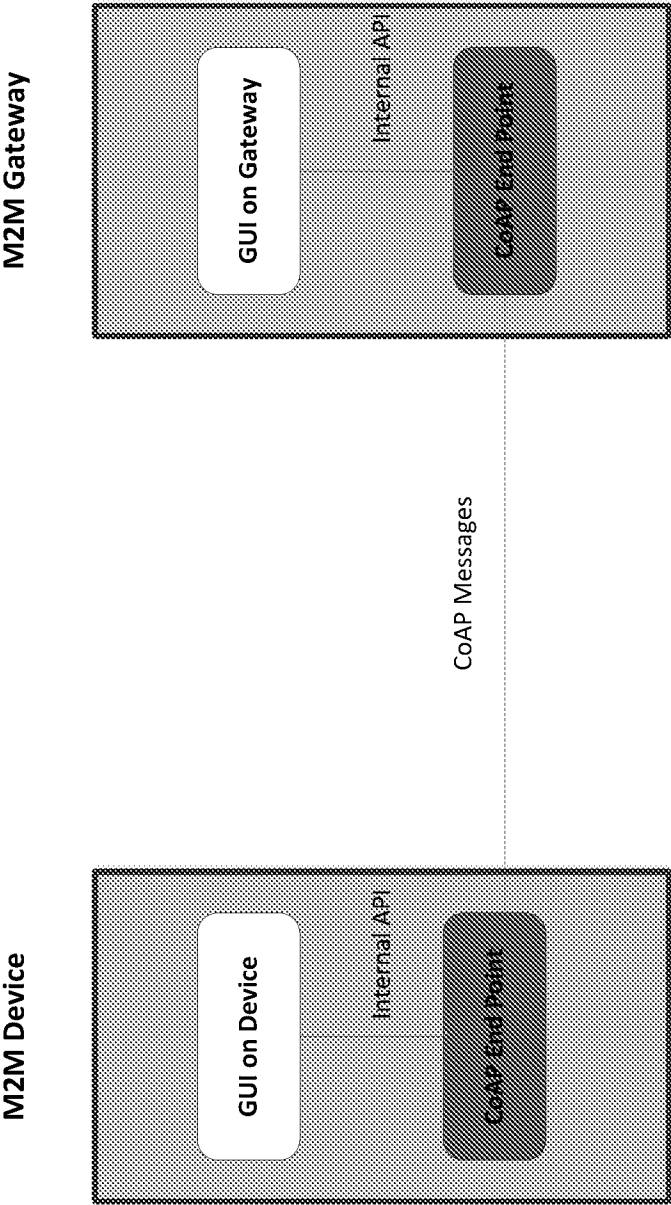


FIG. 24

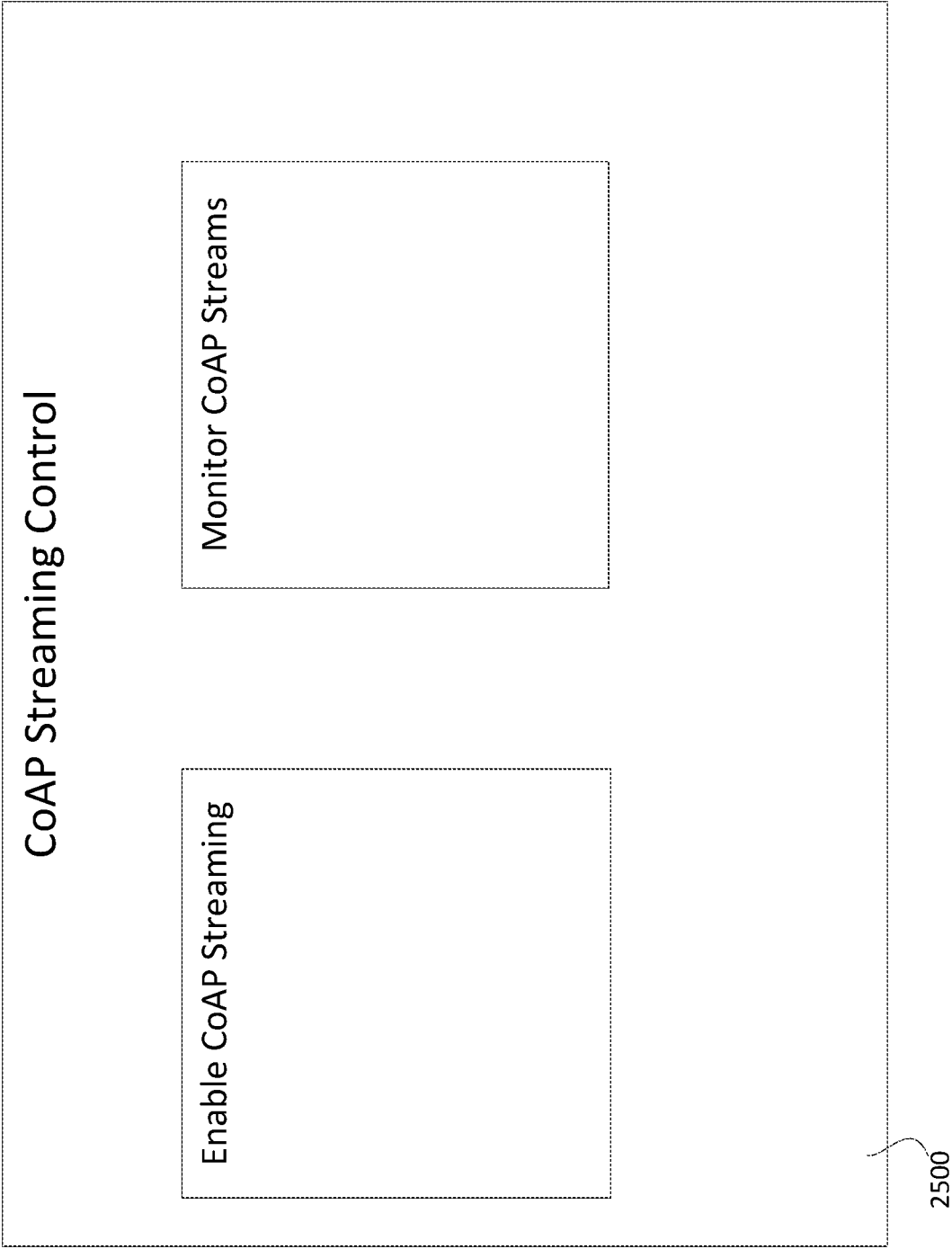


FIG. 25

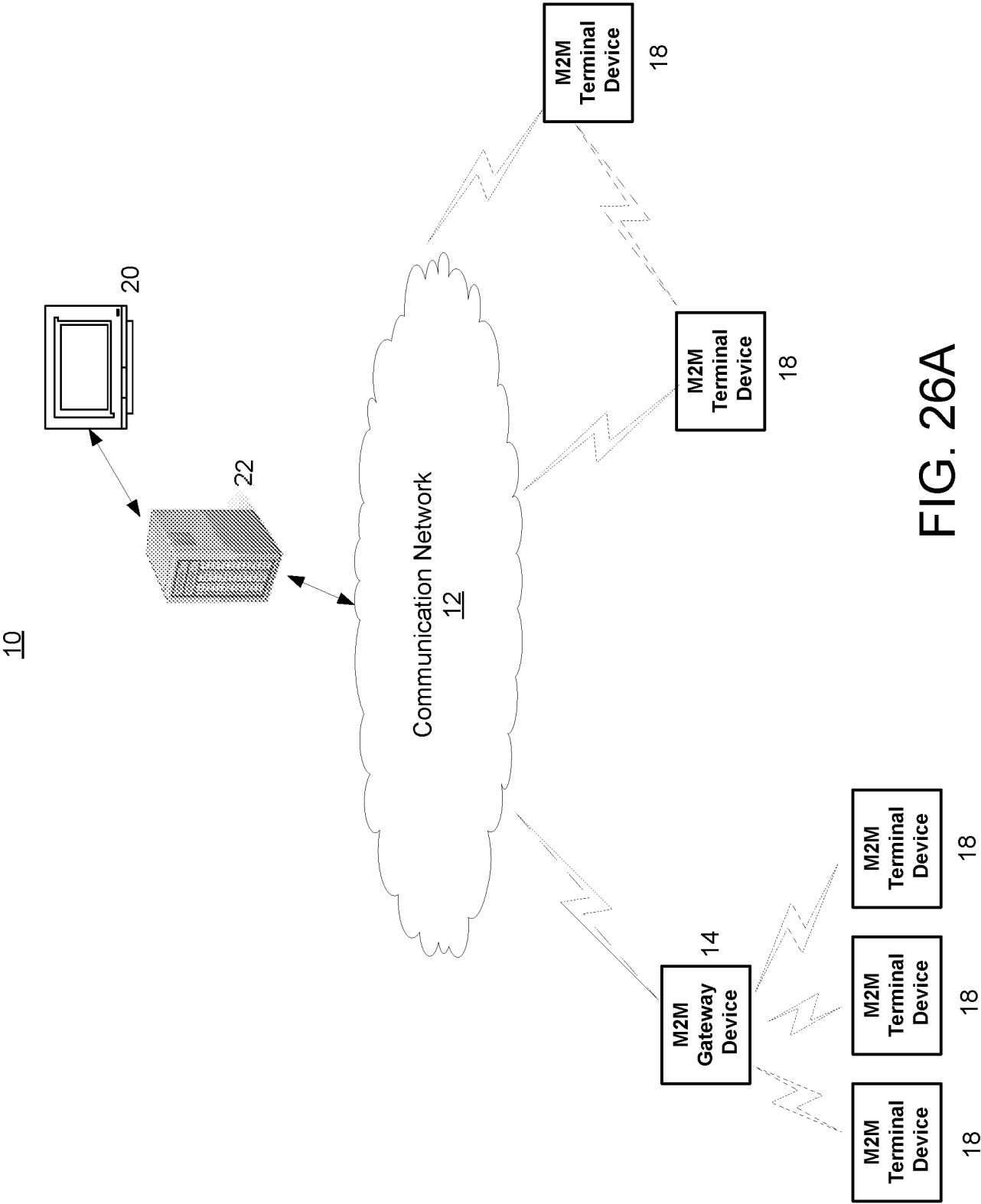


FIG. 26A

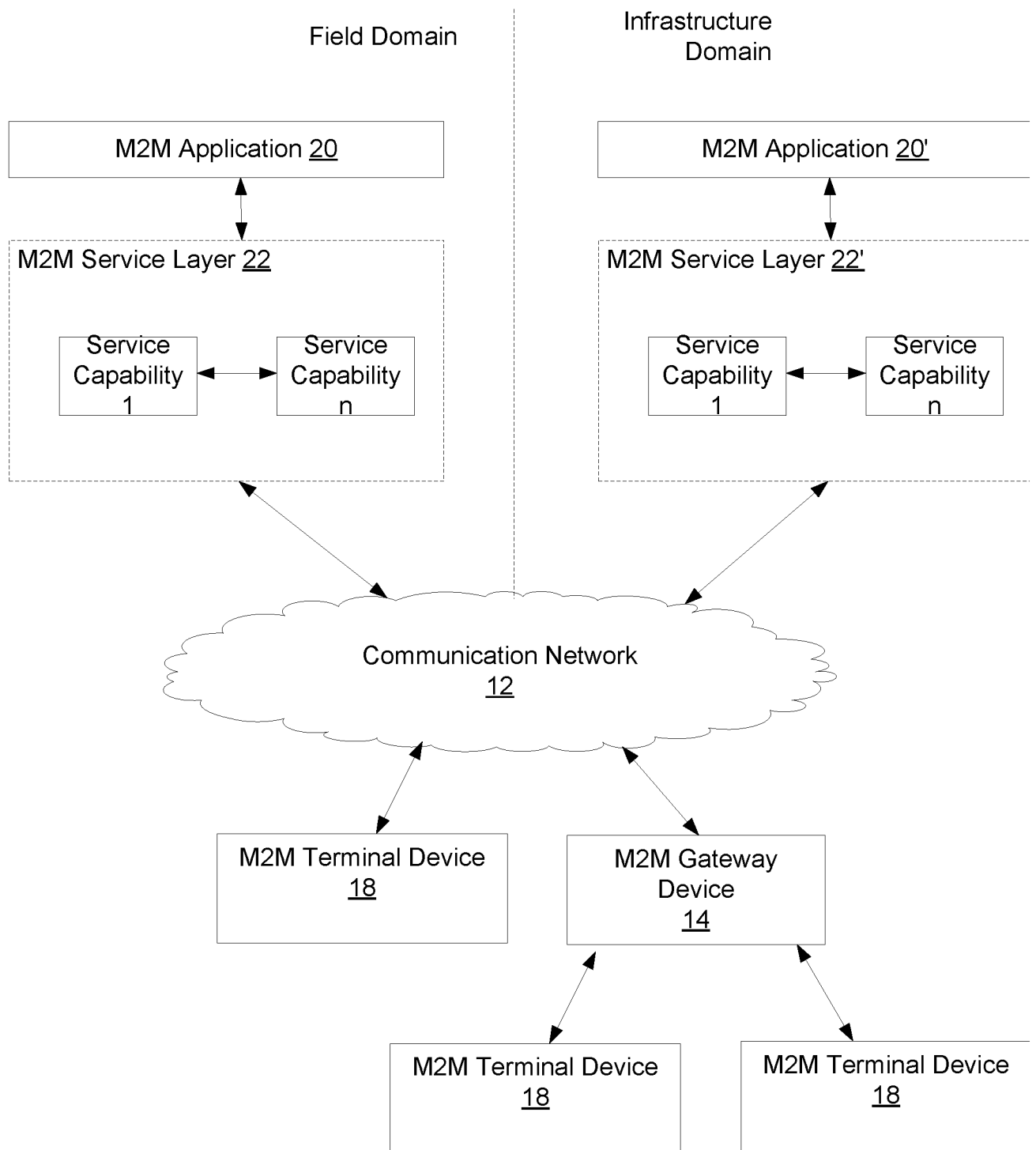


FIG. 26B

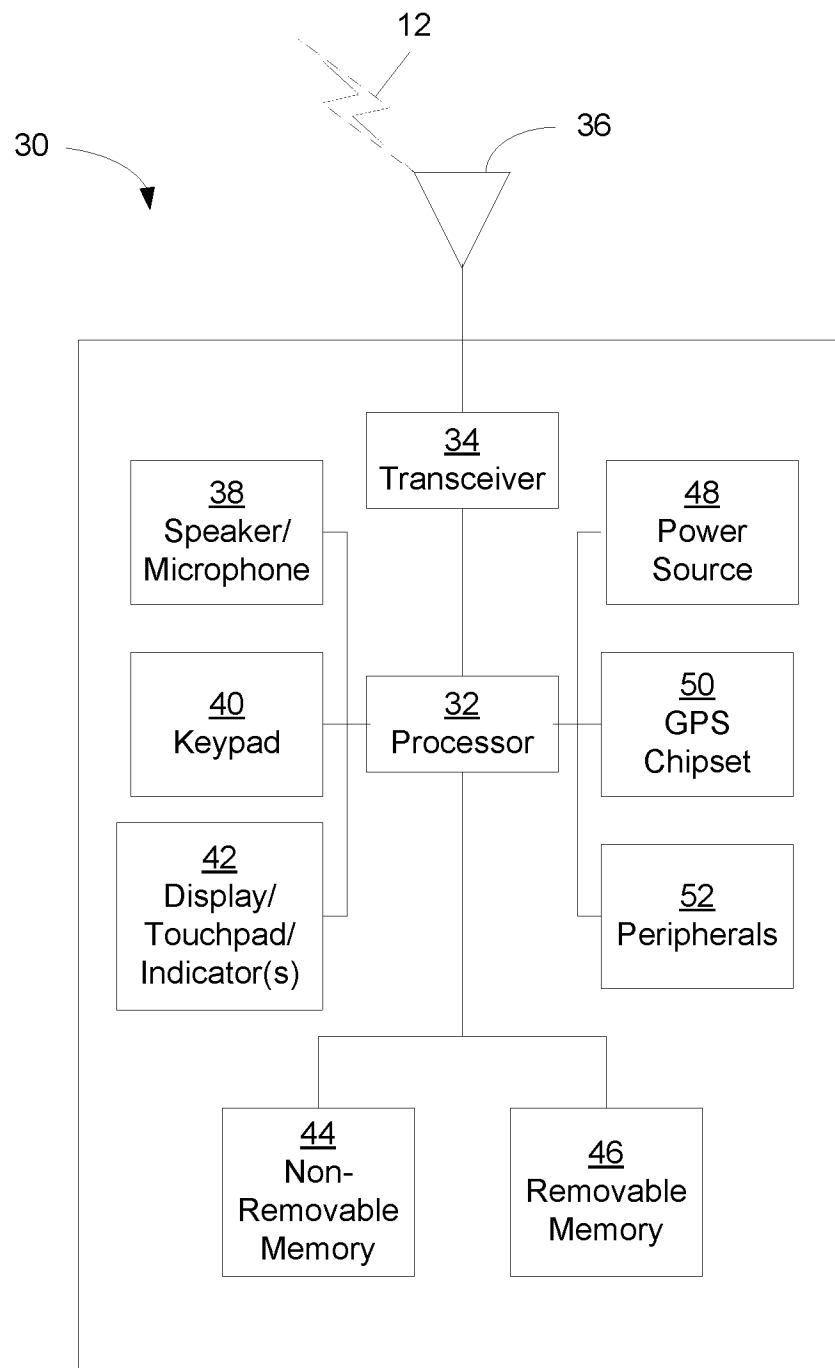


FIG. 26C

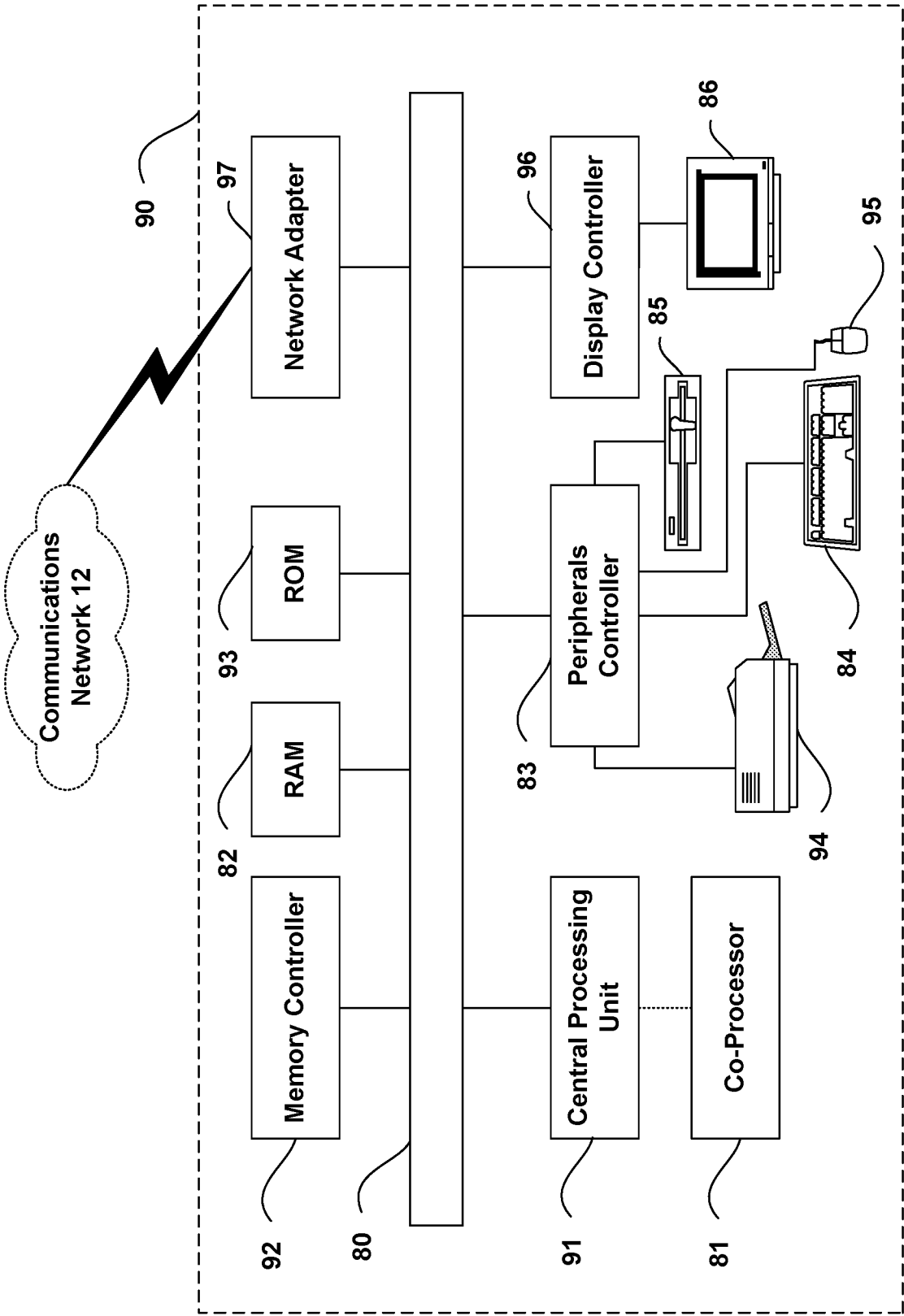


FIG. 26D

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2016/038987

A. CLASSIFICATION OF SUBJECT MATTER
 INV. H04L29/06
 ADD.

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

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)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	LORETO O NOVO ERICSSON S: "CoAP Streaming; draft-loreto-core-coap-streaming-00.txt", COAP STREAMING; DRAFT-LORETO-CORE-COAP-STREAMING-00.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARDWORKINGDRAFT, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 27 March 2012 (2012-03-27), pages 1-9, XP015082193, paragraph [02.1] - paragraph [02.2] figures 2,3 -----	1-5, 11-15
X	WO 2013/127437 A1 (ERICSSON TELEFON AB L M [SE]; NOVO DIAZ OSCAR [FI]; LORETO SALVATORE []) 6 September 2013 (2013-09-06) page 8, line 11 - page 15, line 23 ----- -/-	1-5, 11-15



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

28 September 2016

Date of mailing of the international search report

28/11/2016

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040,
 Fax: (+31-70) 340-3016

Authorized officer

Iapichino, Giuliana

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/038987

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SHELBY K HARTKE C BORMANN UNIVERSITAET BREMEN TZI Z: "The Constrained Application Protocol (CoAP); rfc7252.txt", THE CONSTRAINED APPLICATION PROTOCOL (COAP); RFC7252.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARD, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 27 June 2014 (2014-06-27), pages 1-112, XP015100722, paragraph [5.8.1] paragraph [5.8.3] -----	1-5, 11-15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/038987

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-5, 11-15

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-5, 11-15

A CoAP client able to effectively stream media from a CoAP server.

2. claims: 6-10, 16-19

A CoAP client able to upload chunks of streaming media into a CoAP server.

3. claims: 20-28

A Resource Directory able to forward indications of streaming CoAP resources between CoAP clients.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2016/038987

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2013127437 A1	06-09-2013	EP 2820814 A1	07-01-2015
		US 2015350287 A1	03-12-2015
		WO 2013127437 A1	06-09-2013
