



- (51) **International Patent Classification:**
H04W 4/00 (2009.01) *H04L 12/841* (2013.01)
H04L 29/08 (2006.01)
- (21) **International Application Number:**
PCT/US2016/050134
- (22) **International Filing Date:**
2 September 2016 (02.09.2016)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
62/213,353 2 September 2015 (02.09.2015) US
- (71) **Applicant:** CONVIDA WIRELESS, LLC [US/US]; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US).
- (72) **Inventors:** LI, Xu; 28 Ashford Drive, Plainsboro, NJ 08536 (US). DONG, Lijun; 6085 African Holly Trail, San Diego, CA 92130 (US). LY, Quang; 115 Sterling Drive, North Wales, PA 19454 (US). CHEN, Zhuo; 1397 Parish Avenue, Claymont, DE 19703 (US). RAHMAN, Shamim, Akbar; 6704 Conklin Road, Cote St. Luc, QC H4V 1B8 (CA). LU, Guang; 1 Mendel Crescent, Thornhill, ON L4J 9B6 (CA). WANG, Chonggang; 9 Carlyle Court, Princeton, NJ 08540 (US).
- (74) **Agents:** SAMUELS, Steven, B. et al.; Baker & Hostetler LLP, Cira Centre, 12th Floor, 2929 Arch Street, Philadelphia, 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,

[Continued on next page]

(54) **Title:** ENABLING TIME FLEXIBILITY FOR BLOCK TRANSFER IN COAP PROTOCOL

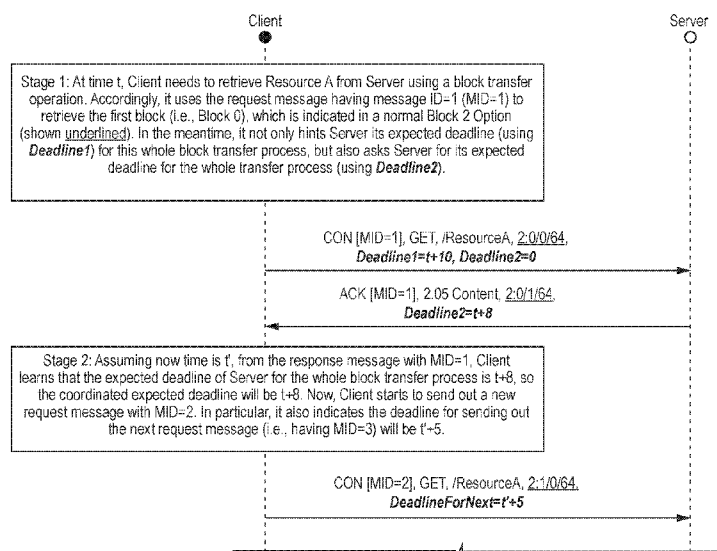


FIG. 10A

(57) **Abstract:** Modifications are made to the existing CoAP block transfer mechanism to minimize waste associated with excess load and congestion. In one aspect, two new Block Options are introduced to indicate client and server time requirements with regard to the current block transfer process. In another aspect, two other new Block Options are introduced to indicate when the next block should be transferred.



GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ,
TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU,
TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,
DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT,
LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE,

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

Published:

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

ENABLING TIME FLEXIBILITY FOR BLOCK TRANSFER IN COAP PROTOCOL

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims benefit under 35 U.S.C. §119(e) of U.S. Provisional Patent Application No. 62/213,353, filed on September 2, 2015, and titled “Enabling Time Flexibility for Block Transfer in CoAP Protocol,” the content of which is incorporated by reference herein in its entirety.

BACKGROUND

[0002] The Internet Engineering Task Force (IETF) Constrained RESTful Environments (CoRE) Working Group (IETF CoRE) has developed Constrained Application Protocol (CoAP) (RFC7252). CoAP is an application protocol for use with constrained nodes and constrained networks, such as wireless-sensor networks and other Internet of Things (IoT) systems.

[0003] The interaction model of CoAP is similar to the client/server model of Hypertext Transfer Protocol (HTTP). CoAP is a single protocol that uses a two-layer approach, as shown in FIG. 1: a CoAP Messages Layer for handling UDP, though with different reliability requirements than typical UDP; and a Request/Response Layer that uses Methods – GET, PUT, POST, DELETE – and Request/Response codes for handling the asynchronous nature of the network interactions. In implementation, both the messages layer and the request/response layer are features of the CoAP header.

[0004] CoAP defines four types of messages: CON, NON, ACK, and RST. A Confirmable message (CON) is retransmitted using a default timeout and exponential back-off between retransmissions until the recipient sends back an Acknowledgement message (ACK) with the same Message ID. FIG. 2 shows an example of a CON with ACK of the same Message ID 0x7d34. A Non-confirmable message (NON) does not require reliable transmission. For example, each single measurement out of a stream of sensor data can be sent as a NON. NONs are not acknowledged, but still have a Message ID for duplicate detection. An example of a NON with Message ID 0x01a0 is shown in FIG. 3. An ACK is used to acknowledge receipt of a CON. A Reset message (RST) may be used when a recipient is not able to process a NON.

[0005] CoAP defines four methods: GET, POST, PUT, and DELETE.

[0006] The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request Uniform Resource Identifier (URI). Upon success, a 2.05 (Content) or 2.03 (Valid) response code should be present in the response.

[0007] 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. POST usually results in a new resource being created or the target resource being updated.

[0008] 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.

[0009] The DELETE method requests that the resource identified by the request URI be deleted.

[0010] Regarding the CoAP Request/Response Layer shown in FIG. 1, CoAP request and response semantics are carried in CoAP messages. A Token is used to match responses to requests independently from the underlying messages. Typically, there are two cases: a synchronous case, and an asynchronous case.

[0011] FIG. 4 shows an example of the synchronous case. In the synchronous case, when a request is carried in a CON, the response to this request will be directly carried in the resulting ACK. This is called a “piggy-backed response.” If the CoAP Client that issued the request in a CON does not receive an ACK within an ACK timer, it will retransmit the request using an exponential back-off algorithm and double the ACK timer to wait for CoAP Server’s Response. If a request is sent in a NON, then the response is sent using a new NON.

[0012] FIG. 5 shows an example of the asynchronous case. In the asynchronous case, when a request is carried in a CON, the client will receive an ACK without a payload. Later, when the requested content becomes available, the server will issue a new CON that includes the response to the client’s initial CON request. In FIG. 5, the token “0x73” is used to associate the response with the previous request.

[0013] CoAP defines a number of options that may be included in a message. 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, an unsigned integer, or a string. 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.

[0014] Options are either "critical" or "elective". An Option is identified by an Option Number, which also provides some additional semantics information. For example, 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. Upon reception, unrecognized Options of class "elective" must be silently ignored. Unrecognized Options of class "critical" that occur in a CON request must cause the return of a 4.02 (Bad Option) response. Unrecognized Options of class "critical" that occur in a CON response, or piggy-backed in an ACK, must cause the response to be rejected (according to RFC7252, rejecting a Confirmable message is effected by sending a matching Reset message and otherwise ignoring it). Unrecognized Options of class "critical" that occur in a NON must cause the message to be rejected.

[0015] Table 1 displays the properties of CoAP Options using two examples defined in the current CoAP block-transfer specification: Block1 and Block2.

Table 1. Example--Block Option Numbers

No.	C (Critical)	U (Unsafe)	N (NoCacheKey)	R (Repeatable)	Name	Format	Length	Default
23	X	x	-	-	Block2	Unit	0-3 B	(none)
27	x	x	-	-	Block1	Unit	0-3 B	(none)

[0016] The CoAP Options are maintained by an Internet Assigned Numbers Authority (IANA) registry. The IANA policy for future additions to this sub-registry is split into three tiers: range [0-255] is reserved for Options defined by the IETF; range [256-2047] is reserved for commonly used Options with public specifications; and range [2048-64999] is for all other Options, including private or vendor-specific Options.

[0017] CoAP is a Representational State Transfer (RESTful) transfer protocol for use with constrained nodes and networks. Basic CoAP messages work well for small data transfers, such as temperature sensors, light switches, and similar building-automation devices. Occasionally, however, applications need to transfer larger payloads. For example, in an e-health application, it may be desirable to update the firmware of a watch from time to time so that the watch can work properly with newly added features.

[0018] Whereas protocols like HTTP may rely on transport protocols like TCP to perform the low-level work, CoAP is based on simpler datagram transports like UDP. HTTP relies on TCP to divide large payloads into multiple packets and ensure that all packets arrive and are handled in the correct order. In contrast, UDP has a size limit on resource representations that may be transferred without excessive fragmentation, and although UDP may support larger payloads through IP fragmentation, IP fragmentation is limited to a size of 64 KiB and is not well-suited for constrained applications and networks.

[0019] To avoid problems with IP fragmentation, existing CoAP defines “Block” Options for transferring multiple blocks of information from a resource representation, and the transferring occurs using multiple request-response pairs. For example, if a CON from a client to a server includes a GET request, the server will immediately send back an ACK with the requested payload piggy-backed in the message. In such a manner, this existing block transfer mechanism can be thought of as part of the Messages Layer of CoAP where the Messages Layer is exactly aligned with the Request/Response Layer.

[0020] Accordingly, the concept of “request-response” pairs as defined in block transfers refers to the messaging paradigm of the CoAP Messages Layer: a “request message” typically refers to a CON with an included GET, PUT, or POST request; and a “response message” typically refers to a response piggy-backed in the corresponding ACK. Further, the term “payload” refers to the actual content of a single CoAP message, while the term “body” refers to the entire resource representation being transferred in block-wise fashion.

[0021] As mentioned above, CoAP has two Block Options defined to enable block-wise transfers: “Block1” and “Block2”. Generally both Block1 and Block2 Options may be present in both request and response messages, and their usages may vary depending on context. The Block1 Option pertains to the request payload, and the Block2 Option pertains to the response payload. For example, if a Block2 is included in the response of a GET request, it

describes the current payload in this response and is referred to as “descriptive usage.”

However, when a Block2 is included in a GET request, it provides additional control over how the payload will be formed in the response to the GET request and is referred to as “control usage.”

[0022] Each Block Option may need to include three items of information: block size (“SZX”); whether more blocks will follow the current message’s block (“M”); and the relative number of the block within a sequence of blocks with the given size (“NUM”). The structure of a Block Option is shown in FIG. 6.

[0023] The block size is represented as a three-bit unsigned integer, and the allowed values of SZX are [0-6]. The block size is found by calculating $2^{(SZX + 4)}$, making the minimum block size $2^{(0+4)} = 16$ and the maximum block size $2^{(6+4)} = 1024$.

[0024] The “more blocks” flag is represented as a single bit, meaning it may only have a value of 0 or 1. For “descriptive usage,” when unset ($M = 0$), M indicates that the payload of this message is the last block in the body; when set ($M = 1$), M indicates that there are one or more additional blocks available for transfer. When a Block1 or Block2 is used for “control usage,” then M must be set to 0.

[0025] The block number indicates the block number being requested or provided. A message having $NUM = 0$ indicates the first block of a resource body.

[0026] FIG. 7 shows an example illustrating how to use Block2 in GET requests and responses. In this example, the whole resource body has been retrieved from Server by using three request-response pairs in a block transfer process. FIG. 7 shows Block2 usage in both the descriptive usage context and the control usage context.

[0027] In the descriptive usage context with Block2 Option in a response message, $NUM = 0$ indicates that the ACK payload contains the first block, $M = 1$ indicates that more blocks need to be transferred, and $SZX = 128$ indicates that the block size is 128 bytes. In implementation, $SZX = 3$ because $2^{(3+4)} = 128$, but this has been simplified for illustration in FIG. 7. Further, because M is set, SZX must be equal to the size of the payload for compilation by the Client.

[0028] In the control usage context with Block2 Option in a request message, $NUM = 1$ indicates that Client is requesting the second block of the body, M has no function and is set to 0, and $SZX = 128$ indicates that the Client expects the block size of the next block to be 128. In

this context generally, SZX will likely be the size of the previous block, or SZX will be a suggested block size if the Client has not yet received any blocks. As shown in FIG. 7, the Server first proposed a block size of 128 in Message (MID) 1234; the Client then followed the suggestion by setting SZX = 128 in the subsequent GET requests. Similarly, FIG. 8 shows an example illustrating how to use the Block1 Option in PUT requests and responses.

[0029] FIG. 9 shows an example CoAP block transfer process. As can be seen, with the existing mechanism, each block is transmitted, by default, immediately one after another. For example, once Client successfully receives Block 0 (zero) carried in the ACK having MID=1234, Client then sends out a new request message, here MID=1235, to retrieve the next block, Block 1, from Server. This process continues until Client has received all blocks.

SUMMARY

[0030] CoAP's block transfer mechanism is needed when M2M/IoT applications intend to transfer large payloads using CoAP. During current block transfers, each block of data is transmitted immediately one after another regardless of any changing network conditions. Because network conditions in M2M/IoT systems may be dynamically changing, e.g., dropped connections, congestion, busy nodes, etc., the current CoAP block transfer process may not function in an optimal manner, wasting time and resources. Disclosed herein is a new solution to message retransmission in a CoAP block transfer. In the disclosed solution, the block transfer process may be executed in a more adaptive manner.

[0031] Embodiments disclosed herein provide modifications to the existing block transfer mechanism to minimize waste associated with excess load and congestion. In one aspect, two new Block Options are introduced to indicate client and server time requirements with regard to the current block transfer process. In another aspect, two other new Block Options are introduced to indicate when the next block should be transferred.

[0032] 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

[0033] A more detailed understanding may be had from the following description, given by way of example in conjunction with the accompanying drawings wherein:

[0034] FIG. 1 illustrates the interaction model of the CoAP protocol;

[0035] FIG. 2 shows an example message flow in accordance with the CoAP protocol;

[0036] FIG. 3 shows another example message flow in accordance with the CoAP protocol;

[0037] FIG. 4 shows another example message flow in accordance with the CoAP protocol;

[0038] FIG. 5 shows another example message flow in accordance with the CoAP protocol;

[0039] FIG. 6 illustrates the format of CoAP Block Options, Block1 and Block2;

[0040] FIG. 7 shows an example message flow in accordance with using the CoAP Block2 Option;

[0041] FIG. 8 shows an example message flow in accordance with using the CoAP Block1 Option;

[0042] FIG. 9 shows an example message flow in accordance with the CoAP block transfer process;

[0043] FIGs. 10A and 10B illustrate an example work flow in accordance with one embodiment of a method for a CoAP block transfer process using CoAP Options Deadline1, Deadline2, DeadlineForNext, and SuspendForNext;

[0044] FIG. 11A is a system diagram of an example machine-to-machine (M2M), Internet of Things (IoT), or Web of Things (WoT) communication system in which one or more disclosed embodiments may be implemented;

[0045] FIG. 11B is a system diagram of an example architecture that may be used within the M2M/IoT/WoT communications system illustrated in FIG. 11A;

[0046] FIG. 11C is a system diagram of an example communication network apparatus, such as an M2M/IoT/WoT device, gateway, or server that may be used within the communications system illustrated in FIGs. 11A and 11B;

[0047] FIG. 11D is a block diagram of an example computing system in which a node of the communication system of FIGs. 11A and 11B may be embodied; and

[0048] FIG. 12 shows an example graphical user interface (GUI) in accordance with an example embodiment.

DETAILED DESCRIPTION

[0049] CoAP's block transfer mechanism is needed when M2M/IoT applications intend to transfer large payloads using CoAP. During current block transfers, each block of data is transmitted immediately one after another until all blocks have been received. Transferring each block immediately after the previous block can become inefficient and burdensome with dynamically changing resources and networks.

[0050] The existing CoAP block transfer process does not consider any time-related flexibility or constraints. Such considerations become necessary when attempting to facilitate block transfers during dynamically changing network conditions. Accordingly, instead of conducting a transfer process in an "as-soon-as-possible" manner (which is the default approach adopted in the existing solution), it is desirable to allow for a block transfer process that can adaptively be suspended, resumed, or slowed as long as the process is still completed within deadlines set by both the client and server. To achieve this adaptive transferring, embodiments are proposed that allow the client and server to convey time-related flexibility and constraints (e.g., deadlines, etc.) to each other during a block transfer process.

[0051] There are two main aspects to the solution disclosed herein. In a first aspect, two new Block Options are introduced to indicate client and server time requirements with regard to the current block transfer process. In a second aspect, two other new Block Options are introduced to indicate when the next block should be transferred. Each of these aspects is described more fully below.

New Transfer-Deadline CoAP Block Options

[0052] In a first aspect, two new block options are used to indicate the client's and server's respective deadline requirements to complete an ongoing block transfer process: Deadline1 and Deadline2. The Deadline1 option is used for the client's expected deadlines, while Deadline2 is used for the server's expected deadlines.

[0053] In one embodiment, when the Deadline1 option is in a request message from a client, it indicates the client's expected deadline for the current block transfer process. The client may proactively assert Deadline1 to the server, or the client may provide Deadline1 at the

server's request. Further in this embodiment, when Deadline1 is included in a response message from a server, it indicates the server's request for the client's expected deadline for the current block transfer process; in this case, Deadline1 is set to zero. After the client receives the Deadline1 option (Deadline1 = 0) from the server, the client will be stimulated to include a Deadline1 option in the next request message. Optionally, the client may ignore the server's request for a deadline.

[0054] In one embodiment, when the Deadline2 option is in a response message from a server, it indicates the server's estimated deadline for the current block transfer process. When Deadline2 is included in a request message from a client, it indicates the client's request for the server's expected deadline for the current block transfer process; in this case, Deadline2 is set to zero.

[0055] By exchanging deadline information, both the client and server can specify their current timing needs or requirements for the block transfer process. In particular, the client is able to include the deadline options in messages at any time during the block transfer process so that the client can decide how best to proceed with the rest of the block transfer process, e.g., how fast to send out subsequent request messages, when to send out, etc. The specific operation behavior will depend on various scenarios and needs. Exchanging deadline information also allows the client and server to negotiate a deadline favorable to both parties.

[0056] Because the client and server may have different deadline requirements, one embodiment includes policies to select a deadline favorable to both parties. The following policies are provided as examples only and should not be construed to be limiting of the scope of the present disclosure or claims.

[0057] According to a first policy, using Deadline1 and Deadline2, both the client and server can indicate their own deadline requirements. If they do so, then the "coordinated deadline," or deadline for the entire process to conclude, should be the earlier submitted deadline. This allows the client to complete the block transfer process in a time favorable to both parties.

[0058] According to another policy, the client and server may also indicate Deadline1 and Deadline2 options multiple times throughout the block transfer process, and newly received deadline values (included in either a Deadline1 or Deadline2) will override previous deadlines values if they exist.

[0059] According to yet another policy, if the entire block transfer process is not complete by the coordinated deadline, the block transfer process will terminate. Therefore, if both the client and server wish to complete a block transfer process before termination and are currently unable to, they need to determine a new coordinated deadline.

[0060] According to other policies, additional mechanisms might be necessary for achieving proper coordination between the client and server. For example, when a server is transferring blocks to the client in an intermittent manner because both the client and server consented to a loose deadline, the client should keep all received blocks in cache as long as the block transfer process is ongoing and the deadline has not passed. The specific behaviors of the client/server relationship may be implementation-dependent. Other mechanisms and considerations will vary based upon implementation.

[0061] Table 2 shows the implementation details of one embodiment of a definition for the new Deadline1 and Deadline2 CoAP Options. The proposed options may be included in the No. 33 and No. 35 CoAP block options within the headers of the request and response messages, respectively. Note that assigning Nos. “33” and “35” to these new Options is presented solely for purposes of example and should not be construed to limit the Options to these specific numbers.

Table 2. Implementation Details for Deadline1 and Deadline2 Options

No.	C (Critical)	U (Unsafe)	N (NoCacheKey)	R (Repeatable)	Name	Format	Length	Default
33	x	x	-	-	Deadline1	Unit	0-1 B	(none)
35	x	x	-	-	Deadline2	Unit	0-1 B	(none)

New Block-Deadline CoAP Options

[0062] The Deadline1 and Deadline2 options proposed in the previous section may be used for indicating expected deadlines for the whole block transfer process. Instead of or in addition to those deadlines options, a client and server may wish to determine time-related constraints for transferring individual blocks during a block transfer process. In particular, the client and server may wish to set the latest time deadline for sending out the next message, or they may wish to suspend transmission of the next message for a specified amount of time. Suspending message transmission may be beneficial if either the client or server is experiencing

temporary processing and/or storage capacity limitations that would hinder processing of a new message or block. For example, a server may signal a client not to transfer the next block within the next one minute because the server does not currently have enough cache space. Similarly, a client may signal a server not to transfer a block for a specified time because the client is about to go to sleep for that specified time. Sleeping is a common operation for constrained M2M/IoT devices, yet the existing block transfer specification does not consider any sleepy-node-related issues. The ability to set a deadline for a next transfer or suspend a transfer for a period of time allows the block transfer process to be executed more effectively than the current block transfer process when utilizing sleepy M2M/IoT nodes.

[0063] In an aspect, two new block options are used to set time-related constraints to individual block transfers: `DeadlineForNext` and `SuspendForNext`. In one embodiment, `DeadlineForNext` may be used by either the client or server to set the deadline for the next request message. `SuspendForNext` may be used by either the client or server to set a time interval before the next message should be sent. In an embodiment, `DeadlineForNext` and `SuspendForNext` refer to a number of block transfers, e.g., a group of the next ten blocks. Both options may also be used together to create a range of time when the next message should be sent. For example, when sending a message with `DeadlineForNext` indicating ten minutes from current time t and `SuspendForNext` indicating five minutes from current time t , the next request message should be sent between $[t+5, t+10]$.

[0064] In one embodiment, when the `DeadlineForNext` option is in a request message, having message ID N , from a client, it indicates the latest time that the client will issue the next request message, having message ID $N+1$. Based on this indication, the server may react accordingly. For example, if the server does not receive the next request message until after `DeadlineForNext`, it may plan to store the previously-received blocks in a low priority cache, i.e., the blocks may be deleted when the cache needs to be cleaned, because it believes that the current block transfer process has halted. Without such an indication, the server cannot tell if the ongoing block transfer process is halted or is simply interrupted in an intermittent manner (e.g., “Pause-and-Resume”).

[0065] In this embodiment, when the `DeadlineForNext` option is in a response message from a server (i.e., the response to the request message with message ID N), it indicates the latest time that the server is willing to accept the next request message (i.e., the one having message ID

N+1). The client may decide when to send out the next request message based on the server's indicated deadline.

[0066] In one embodiment, when the SuspendForNext option is in a request message, having message ID N, from a client, it indicates that the client will not issue the next request message, having message ID N+1, before the time indicated by SuspendForNext. Based on this indication, the server may devote its resources to other matters until needing to reallocate resources (CPU, Memory, etc.) to this block transfer process after the suspension period.

[0067] In this embodiment, when the SuspendForNext option is in a response message from a server (i.e., the response to the message with message ID N), it indicates that the server is unable to accept the next request message (i.e., the one having message ID N+1) until the time indicated by SuspendForNext. The client may reschedule the transmission time for the next request message based upon the suspension time, or the client may choose to go to sleep for energy-efficient reasons until a time just before the SuspendForNext time. Once awake, the block transfer process may resume.

[0068] Situations may arise where both the client and server use DeadlineForNext and/or SuspendForNext, creating conflicts. Therefore, embodiments may include policies to resolve conflicts. The following policies are provided as examples and should not be construed to be limiting of the present disclosure or claims.

[0069] According to one policy, because the client initializes request messages, if the server intends to pose its own DeadlineForNext or SuspendForNext options, these options should not conflict with those posed by the client.

[0070] According to another policy, the client and server options may be able to be coordinated. For example, both client and server can indicate their own DeadlineForNext in the request/response message (say having message ID N), respectively. If so, the "coordinated deadline" for the next request message (i.e., having message ID N+1) should be the smaller one of the two expected deadlines from two sides.

[0071] According to yet another policy, in lossy communication networks, the client and server might need to dynamically rectify or adjust values of their DeadlineForNext and SuspendForNext options to ensure the values of those options are meaningful, i.e., still valid. Any non-meaningful values will be automatically ignored.

[0072] Table 3 shows the implementation details of one embodiment of a definition for the new DeadlineForNext and SuspendForNext CoAP Options. The proposed options may be included in the No. 37 and No. 39 CoAP block options within the headers of the request and response messages, respectively. Note that assigning Nos. “37” and “39” to these new Options is presented solely for purposes of example and should not be construed to limit the Options to these specific numbers.

Table 3. Implementation Details for DeadlineForNext and SuspendForNext Options

No.	C (Critical)	U (Unsafe)	N (NoCacheKey)	R (Repeatable)	Name	Format	Length	Default
37	x	x	-	-	DeadlineForNext	Unit	0-1 B	(none)
39	x	x	-	-	SuspendForNext	Unit	0-1 B	(none)

[0073] FIGs. 10A and 10B illustrate an example workflow in which embodiments of the Deadline1, Deadline2, DeadlineForNext, and SuspendForNext options are employed, illustrating how those options may be employed in a communication scenario.

[0074] As shown in FIG. 10A, in stage 1, at time t , Client needs to retrieve Resource A from Server using a block transfer operation. Accordingly, it uses the request message having message ID 1 (MID=1) to retrieve the first block, Block 0, using a standard Block2 Option. In the same message, Client chooses to give Server an expected deadline of $t+10$, using Deadline1= $t+10$, to complete the transfer process and asks Server for Server’s expected deadline by using Deadline2=0. As mentioned above, Client or Server could indicate such information during any time of a block transfer process.

[0075] In stage 2, at time t' , Client receives Server’s response message with MID=1 and learns, via Deadline2= $t+8$, that Server’s expected deadline for the entire block transfer process is $t+8$. Using the coordinated deadline policy set out above, the coordinated deadline for both Client and Server becomes the sooner deadline of $t+8$. Client now creates a new request message with MID=2 and wishes to tell Server that Client will send out the next request message, MID=3, by time $t'+5$, using DeadlineForNext= $t'+5$. This process assumes that $t'+5 < t+8$ because the deadline for the next request message must be before the deadline for the entire block transfer process, lest the process terminate before completing.

[0076] As shown in FIG. 10B, in stage 3, Server receives Client's request message with MID=2 and learns Client plans to send the next request message by $t'+5$. However, due to resource limitations, Server cannot receive another request message before $t'+3$ and so sends a response message with SuspendForNext= $t'+3$.

[0077] In stage 4, Client receives Server's response message with MID=2 and learns that Server cannot process another message before $t'+3$. To meet both Server's request and its own deadline, Client should send out message with MID=3 between $[t'+3, t'+5]$, and Client does so at $t'+4$.

[0078] The block transfer process continues until it completes at $t+7$, as indicated in stage 5. Time $t+7$ is before the coordinated deadline of $t+8$, and both Client and Server have met their expected deadlines. It can be seen that through exchanging such useful information between Client and Server by using the newly defined options, the two parties can better schedule and coordinate the message transferring based on needs from both sides.

[0079] It is understood that the stages of FIGs. 10A and 10B may be performed by a client or server, such as the clients and servers illustrated in FIGs. 2-5 and 7-9, which may represent logical entities in a communication network and 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 such network, which may comprise one of the general architectures illustrated in FIGs. 11C or 11D as described more fully below. That is, the operations illustrated in FIGs. 10A and 10B may be implemented in the form of software (i.e., computer-executable instructions) stored in a memory of a network apparatus, such as for example the apparatus or computer system illustrated in FIGs. 11C or 11D, which computer executable instructions, when executed by a processor of the apparatus, perform the steps illustrated in the figure. It is also understood that any transmitting and receiving steps illustrated in these figures may be performed by communication circuitry (e.g., circuitry 34 or 97 of FIGs. 11C and 11D, respectively) of the apparatus under control of the processor of the apparatus and the computer-executable instructions (e.g., software) that it executes.

Example Graphical User Interface

[0080] Several new options have been proposed to enable time flexibility for CoAP block transfer operations, including new block options Deadline1 and Deadline2 as well as DeadlineForNext and SuspendForNext. Each of these new options may optionally be included

in the CoAP request/response messages. If these options are not used, a default block transfer operation may be executed. Therefore, in accordance with one embodiment, a Graphical User Interface (GUI) may be provided for a user to select (1) which options are to be used in the block transfer operations, and (2) if used, the maximal default values of those block options based on the practical application scenario. FIG. 12 illustrates one example GUI in accordance with one embodiment.

[0081] As shown in FIG. 12, in one embodiment, (1) users may select which options are to be used, and (2) for a given selected option to be used and because there are different usages for options depending on whether each option is included in a request or response message, users may input the maximal default values for each option for different cases based on specific needs. For Deadline1 and Deadline2, it can be seen that the users only need to set the maximal default value when Deadline1 is included in a request message and Deadline2 is included in a response message, respectively. This is because of the usages disclosed for Deadline1 and Deadline2; when Deadline1 is included in a response message and Deadline2 is included in a request message, respectively, their values are 0. Therefore, there is no need for a user to set values for those two cases. However, for the usages of DeadlineForNext and SuspendForNext as disclosed herein, users may set the maximal default values for those parameters when they are included in either a request message or a response message.

Example Environment

[0082] FIG. 11A 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, M2M service platform or other network apparatus may be a component or node of the IoT/WoT as well as an IoT/WoT service layer, etc. Any of the client or server devices illustrated in any of FIGs. 2-5 or 7-10 may comprise a node (*i.e.*, network apparatus) of a communication system such as the one illustrated in FIGs. 11A-D.

[0083] As shown in FIG. 11A, 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.

[0084] As shown in FIG. 11A, 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 apparatuses (e.g., servers, gateways, device, and the like) of the network. For example, the Field Domain may include M2M gateways 14 and devices 18. It will be appreciated that any number of M2M gateway devices 14 and M2M devices 18 may be included in the M2M/ IoT/WoT communication system 10 as desired. Each of the M2M gateway devices 14 and M2M 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 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 devices 18 may also receive data from the M2M application 20 or an M2M 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 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. Exemplary M2M devices 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.

[0085] Referring to FIG. 11B, the illustrated M2M service layer 22 in the field domain provides services for the M2M application 20, M2M gateways 14, and M2M devices 18 and the communication network 12. It will be understood that the M2M service layer 22 may communicate with any number of M2M applications, M2M gateways 14, M2M 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 like apparatuses. The M2M service layer 22 provides service capabilities that apply to M2M 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.

[0086] 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 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' may be implemented by one or more nodes of the network, which may comprises servers, computers, devices, virtual machines (e.g., cloud computing/storage farms, etc.) or like apparatuses.

[0087] Referring also to FIG. 11B, 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.

[0088] 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 apparatuses 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'.

[0089] Generally, a service layer, such as the service layers 22 and 22' illustrated in FIGs. 11A and 11B, may be 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 Common Services Entity (CSE) or Service Capability Layer (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. 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 device SCL ("DSCL"), gateway SCL ("GSCL"), or network SCL ("NSCL") of the ETSI M2M

architecture, in a Service Capability Server (SCS) of the 3GPP MTC architecture, in a Common Service Function (“CSF”) or CSE of the oneM2M architecture, or in some other apparatus 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 apparatuses in the network, including servers, computers, and other computing devices or apparatuses, or as part of one or more existing apparatuses. As an example, an instance of a service layer or component thereof may be implemented in the form of software running on a network apparatus (e.g., server, computer, gateway, device or the like) having the general architecture illustrated in FIG. 11C or FIG. 11D described below.

[0090] Further, the methods and functionalities described herein may 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.

[0091] FIG. 11C is a block diagram of an example hardware/software architecture of an apparatus of a network, such as one of the clients or servers illustrated in FIGs. 2-5 or 7-10 which may operate as an M2M server, gateway, device, or other apparatus in an M2M network such as that illustrated in FIGs. 11A and 11B. As shown in FIG. 11C, the apparatus 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 apparatus 30 may also include communication circuitry, such as a transceiver 34 and a transmit/receive element 36. It will be appreciated that the apparatus 30 may include any sub-combination of the foregoing elements while remaining consistent with an embodiment. This apparatus may be a node that implements the time flexibility functionality described herein.

[0092] 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 apparatus in order to perform the various required functions of the apparatus. For example, the processor 32 may perform signal coding, data

processing, power control, input/output processing, and/or any other functionality that enables the apparatus 30 to operate in a wireless or wired environment. The computer executable instructions stored in the memory of the apparatus, and executed by the processor, may further cause the apparatus to perform the operations illustrated in Figure 10 described above. 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.

[0093] As shown in FIG. 11C, 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 apparatus 30 to communicate with other apparatuses 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 and illustrated herein (e.g., in FIGs. 2-5 and 7-10) and in the claims. While FIG. 11C 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.

[0094] The transmit/receive element 36 may be configured to transmit signals to, or receive signals from, other apparatuses, 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.

[0095] In addition, although the transmit/receive element 36 is depicted in FIG. 11C as a single element, the apparatus 30 may include any number of transmit/receive elements 36. More specifically, the apparatus 30 may employ MIMO technology. Thus, in an embodiment,

the apparatus 30 may include two or more transmit/receive elements 36 (e.g., multiple antennas) for transmitting and receiving wireless signals.

[0096] 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 apparatus 30 may have multi-mode capabilities. Thus, the transceiver 34 may include multiple transceivers for enabling the apparatus 30 to communicate via multiple RATs, such as UTRA and IEEE 802.11, for example.

[0097] 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 apparatus 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 communications and to provide a graphical user interface, such as the GUI illustrated in FIG. 12 and described above.

[0098] 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 apparatus 30. The power source 48 may be any suitable device for powering the apparatus 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.

[0100] 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 apparatus 30. It will be appreciated that the apparatus 30 may acquire location information by way of any suitable location-determination method while remaining consistent with an embodiment.

[0101] 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., fingerprint) 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.

[0102] The apparatus 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 apparatus 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.

[0103] FIG. 11D is a block diagram of an exemplary computing system 90 which may also be used to implement one or more apparatuses of a network, such as the clients or servers illustrated in FIGs. 2-5 and 7-10, which may operate as an M2M server, gateway, device, or other apparatus in an M2M network such as that illustrated in FIGs. 11A and 11B. 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. 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, such as, for example, performing the operations illustrated and described in FIGs. 10A and 10B and the accompanying description. 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.

[0104] 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.

[0105] 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.

[0106] 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.

[0107] 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. For example, the display 86 may be used to display the graphical user interface illustrated in FIG. 12 and described above.

[0108] 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 FIG. 11A and FIG. 11B, to enable the computing system 90 to communicate with other nodes of the network. The communication circuitry, alone or in combination with the CPU 91, may be used to perform the transmitting and receiving steps described herein (e.g., in FIGs. 2-5 and 7-10) and in the claims.

[0109] 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 an apparatus 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 may be implemented in the form of such computer executable instructions. 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 includes 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.

[0110] 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. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:
 - determine a deadline by which to complete a block transfer process to retrieve a resource comprising a plurality of blocks from an other apparatus on the network;
 - construct a CoAP message indicating a request to retrieve a first block of the resource from the other apparatus and containing information indicating a deadline for the block transfer process; and
 - send, via the communication network to the other apparatus on the network, the CoAP message to initiate the block transfer process.
2. The apparatus recited in claim 1, wherein the information indicating a deadline for the block transfer process is specified in a CoAP Option field of the CoAP message.
3. The apparatus recited in claim 2, wherein the CoAP Option field comprises one byte of the CoAP message.
4. The apparatus recited in claim 1, wherein the computer-executable instructions further cause the apparatus to:
 - if the deadline passes before the block transfer process is complete, terminate the block transfer process.
5. The apparatus recited in claim 1, wherein the computer-executable instructions further cause the apparatus to:
 - before the deadline passes, send, via the communication network to the other apparatus on the network, a CoAP message indicating a second deadline to complete the block transfer process, the second deadline replacing the deadline.

6. The apparatus recited in claim 5, wherein the computer-executable instructions further cause the apparatus to:

if the second deadline passes before the block transfer process is complete, terminate the block transfer process.

7. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:

send, via the communication network to another apparatus on the network, a CoAP message indicating a request to retrieve a first block of a resource from the other apparatus via a block transfer process and further indicating a request for a deadline from the other apparatus to complete the block transfer process.

8. The apparatus recited in claim 7, wherein the information indicating a deadline for the block transfer process is specified in a CoAP Option field of the CoAP message.

9. The apparatus recited in claim 7, wherein the computer-executable instructions further cause the apparatus to:

receive, via the communication network from the other apparatus on the network, a CoAP message including the first block of the resource and indicating the deadline to complete the block transfer process.

10. The apparatus recited in claim 9, wherein the information indicating a deadline for the block transfer process is specified in a CoAP Option field of the CoAP message.

11. The apparatus recited in claim 9, wherein the computer-executable instructions further cause the apparatus to:

if the deadline passes before the block transfer process is complete, terminate the block transfer process.

12. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:

send, via the communication network to an other apparatus on the network, a CoAP message indicating a first deadline to complete a block transfer process;

receive, via the communication network from the other apparatus on the network, a CoAP message indicating a second deadline to complete the block transfer process;

determine which of the first deadline or second deadline is a sooner deadline; and

if the sooner deadline passes before the block transfer process is complete:

terminate the block transfer process.

13. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:

send, via the communication network to an other apparatus on the network, a CoAP message indicating a first deadline to complete a block transfer process;

send, via the communication network to an other apparatus on the network, a CoAP message indicating a second deadline to complete the block transfer process; and

if the second deadline passes before the block transfer process is complete:

terminate the block transfer process.

14. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:

send, via the communication network to an other apparatus on the network, a CoAP message having a CoAP Option field set to indicate a deadline for a sending a next message;

if the deadline passes before another message is sent:

terminate the block transfer process.

15. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a communications network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to:

- receive, via the communication network from an other apparatus on the network during a block transfer process, a CoAP message having a CoAP Option field set to indicate a delay for sending a next message;

- wait for the indicated delay; and

- send, via the communication network to an other apparatus on the network, a CoAP message to continue the block transfer process.

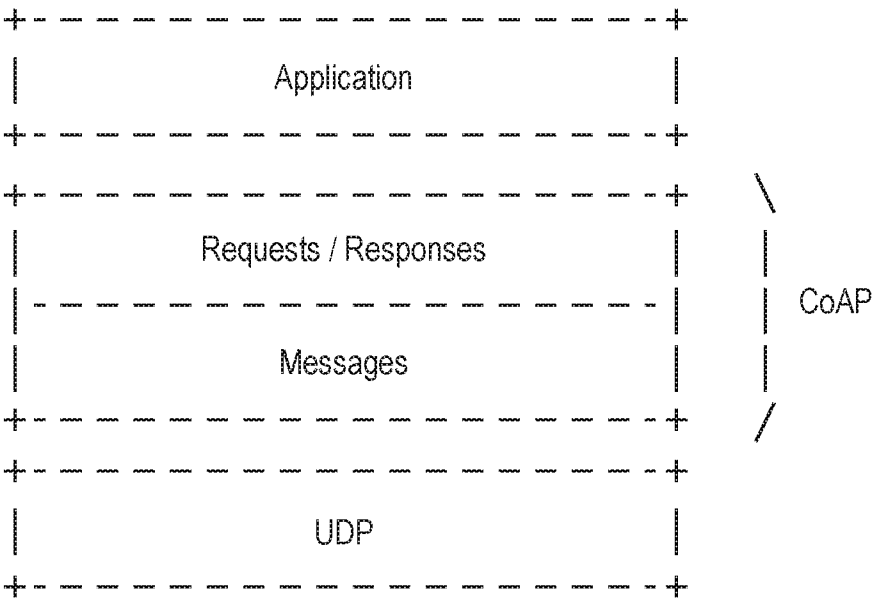


FIG. 1

2/13

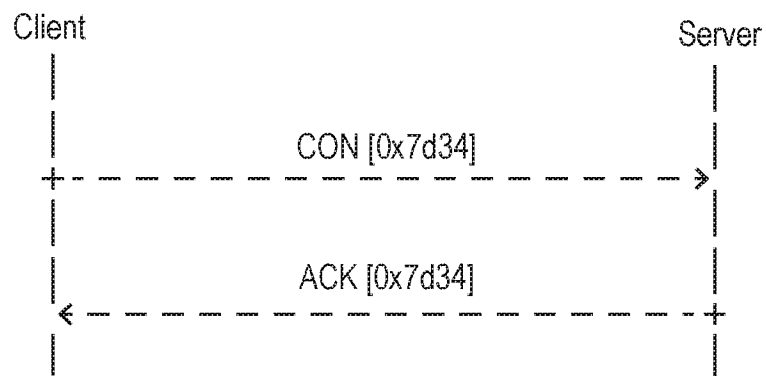


FIG. 2

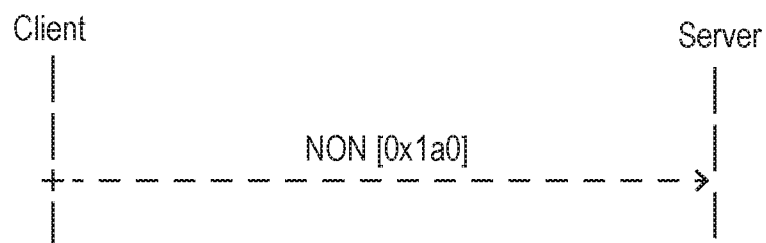


FIG. 3

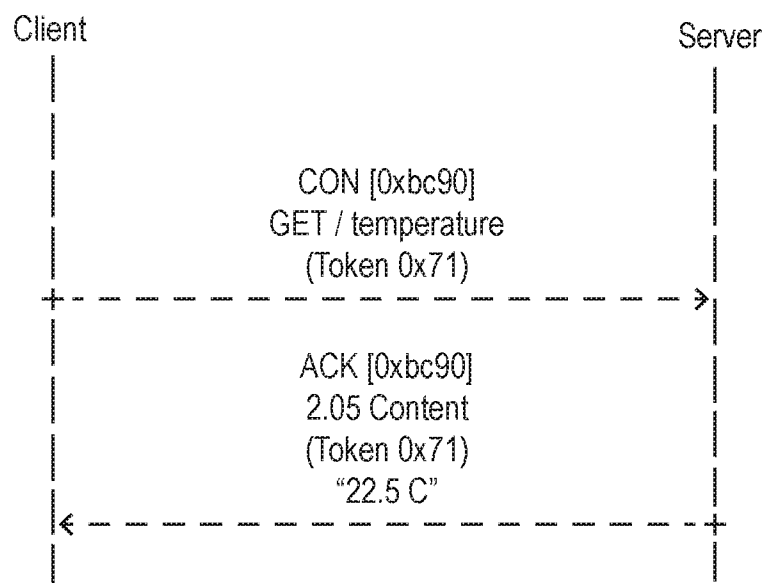


FIG. 4

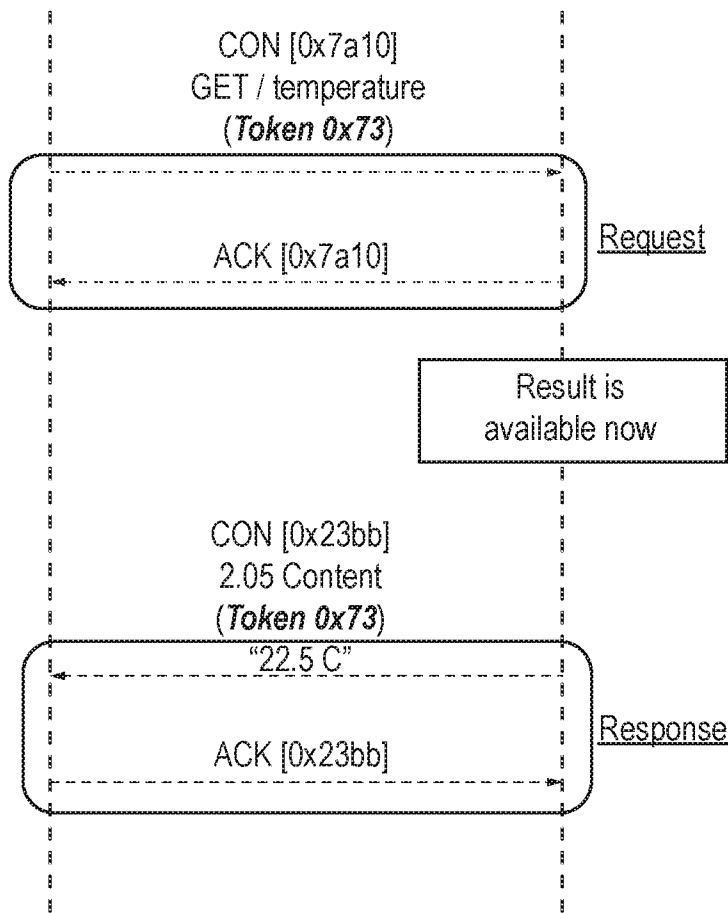


FIG. 5

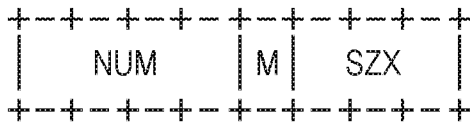


FIG. 6

4/13

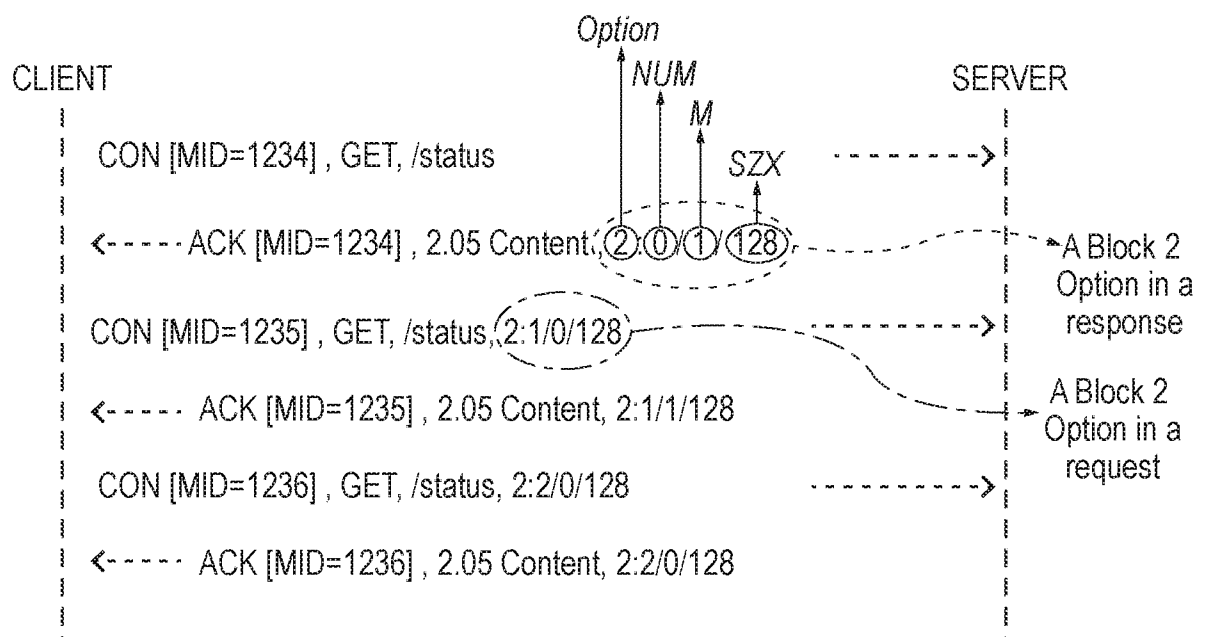


FIG. 7

5/13

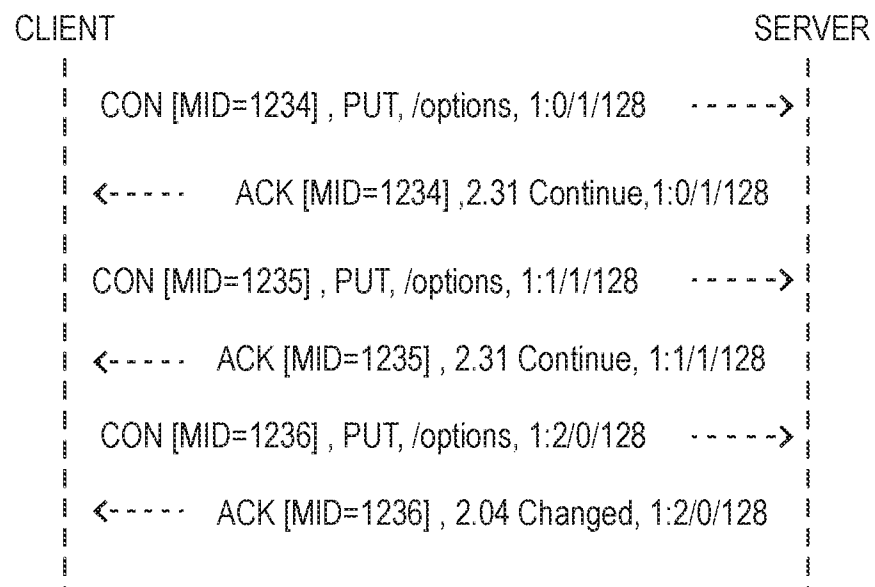


FIG. 8

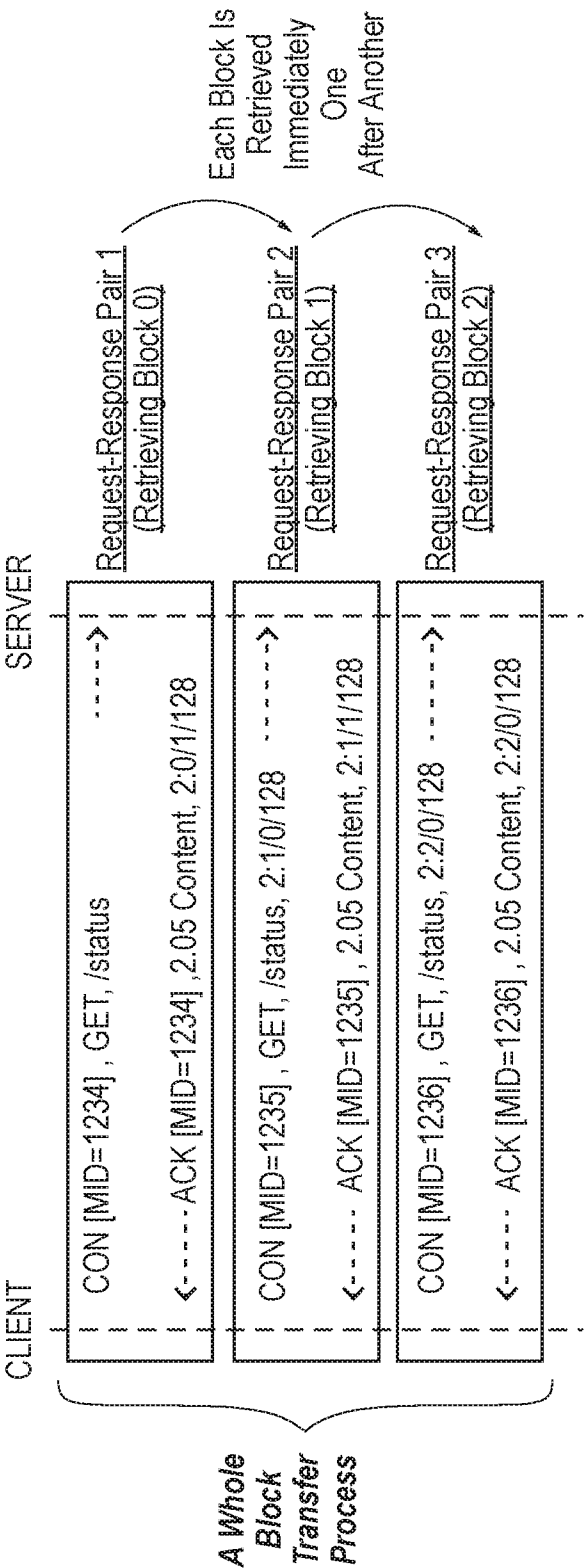


FIG. 9

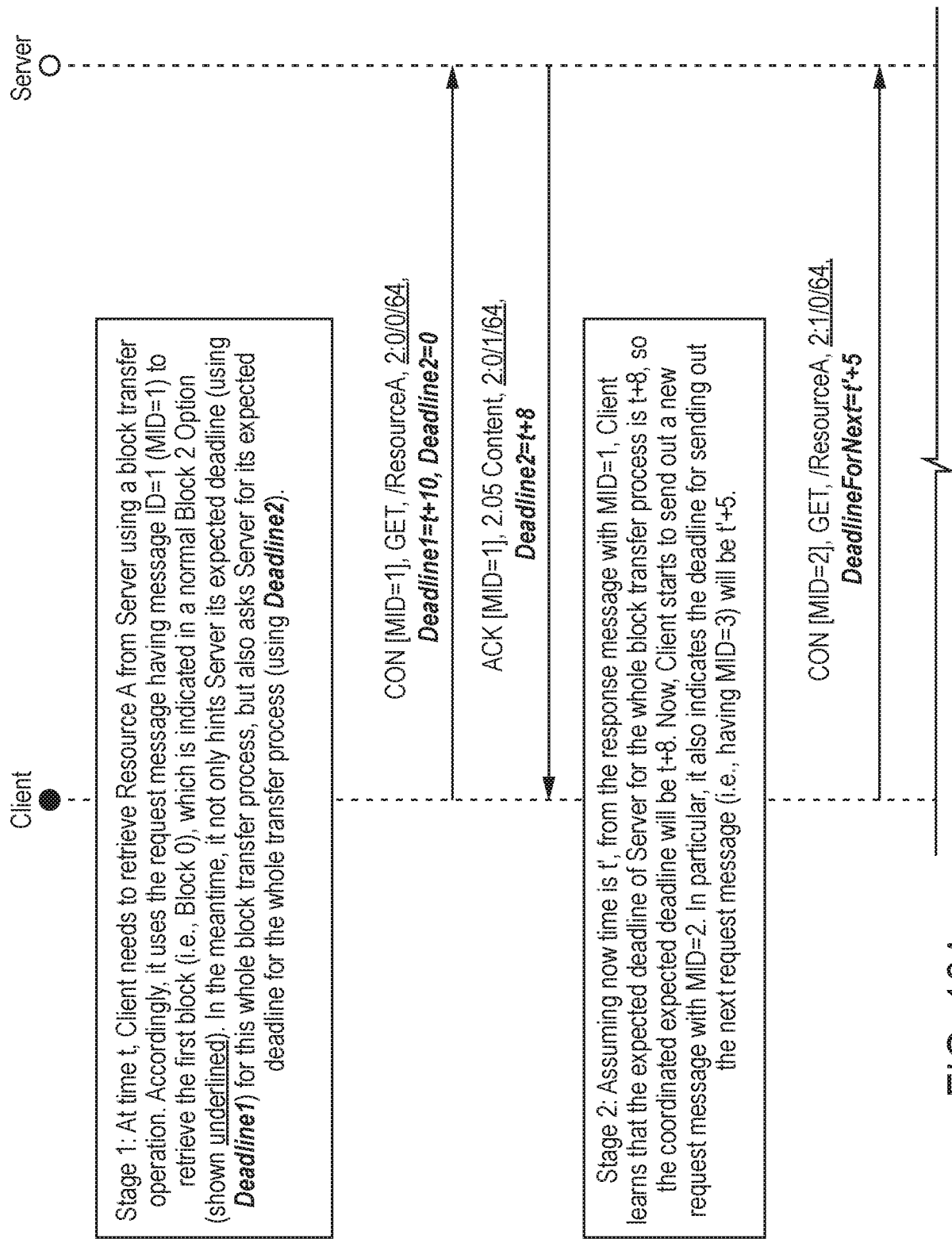


FIG. 10A

8/13

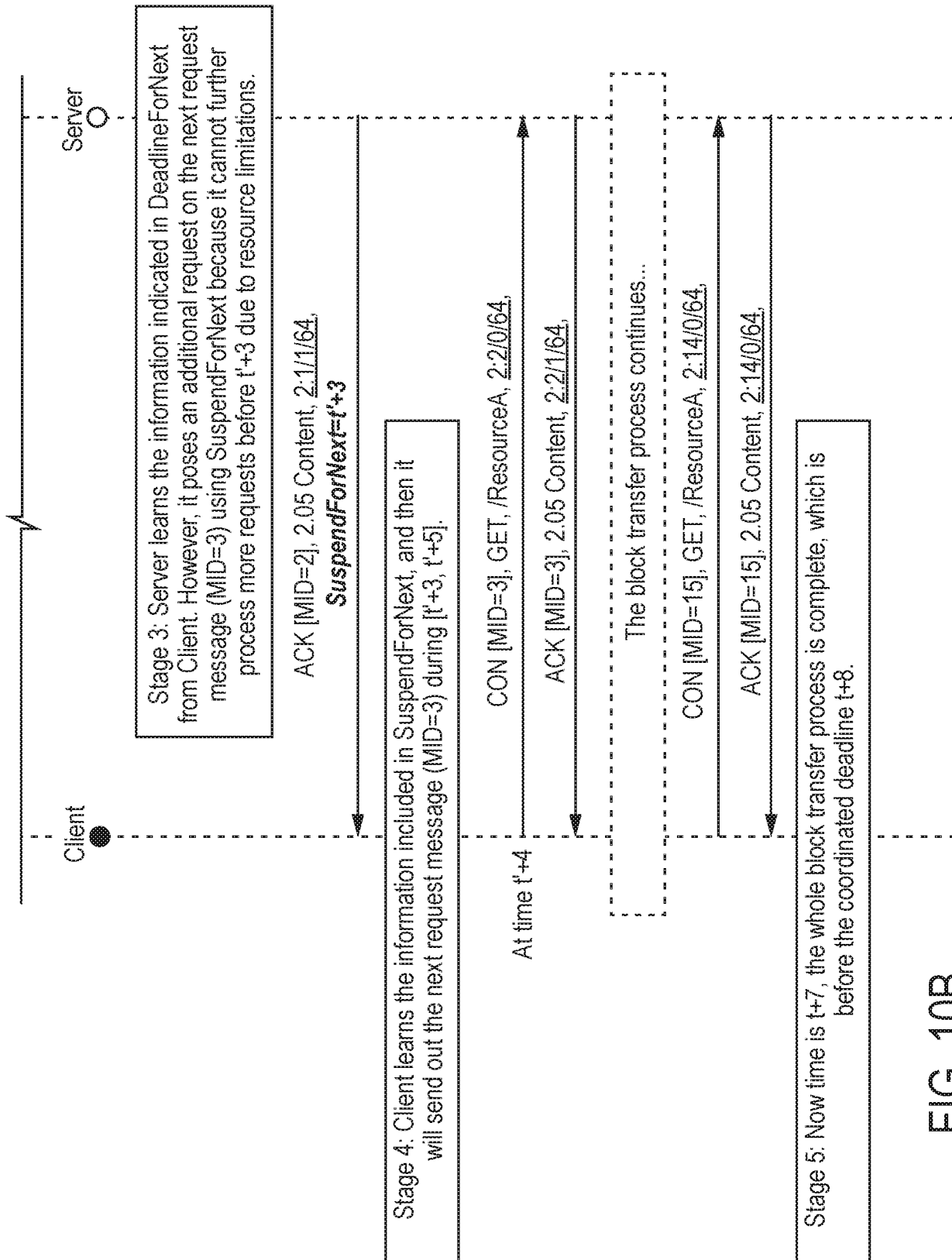


FIG. 10B

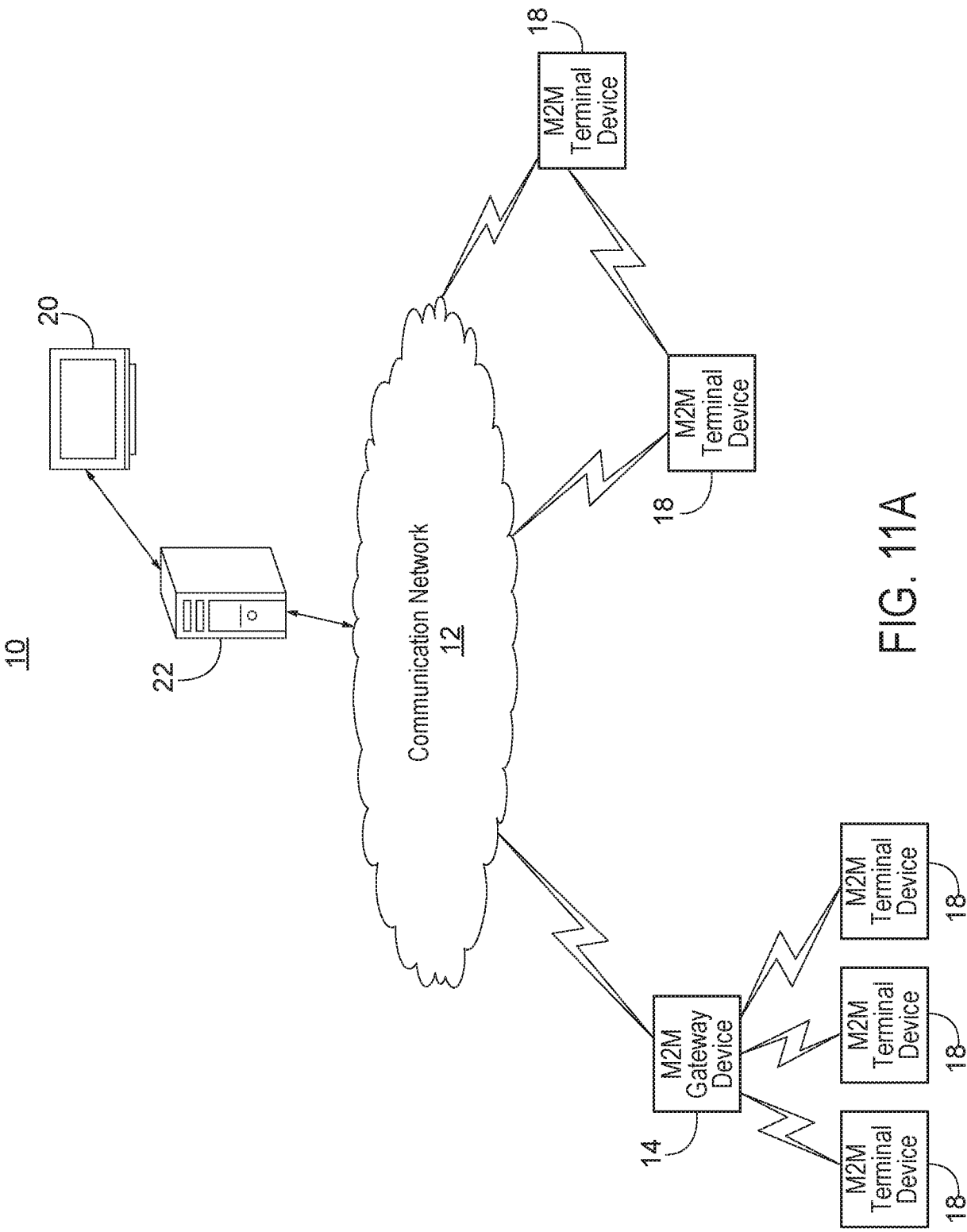


FIG. 11A

10/13

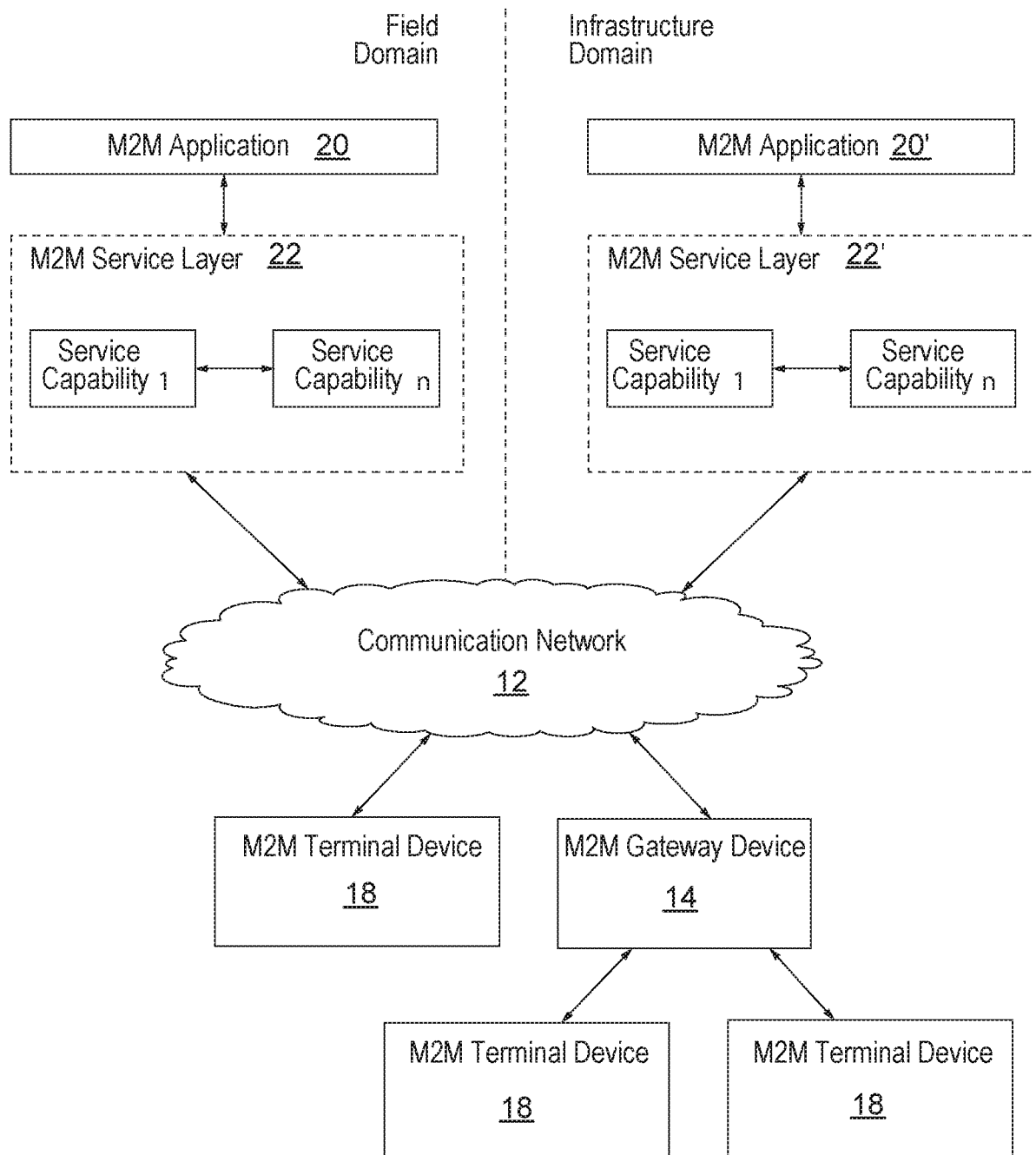


FIG. 11B

11/13

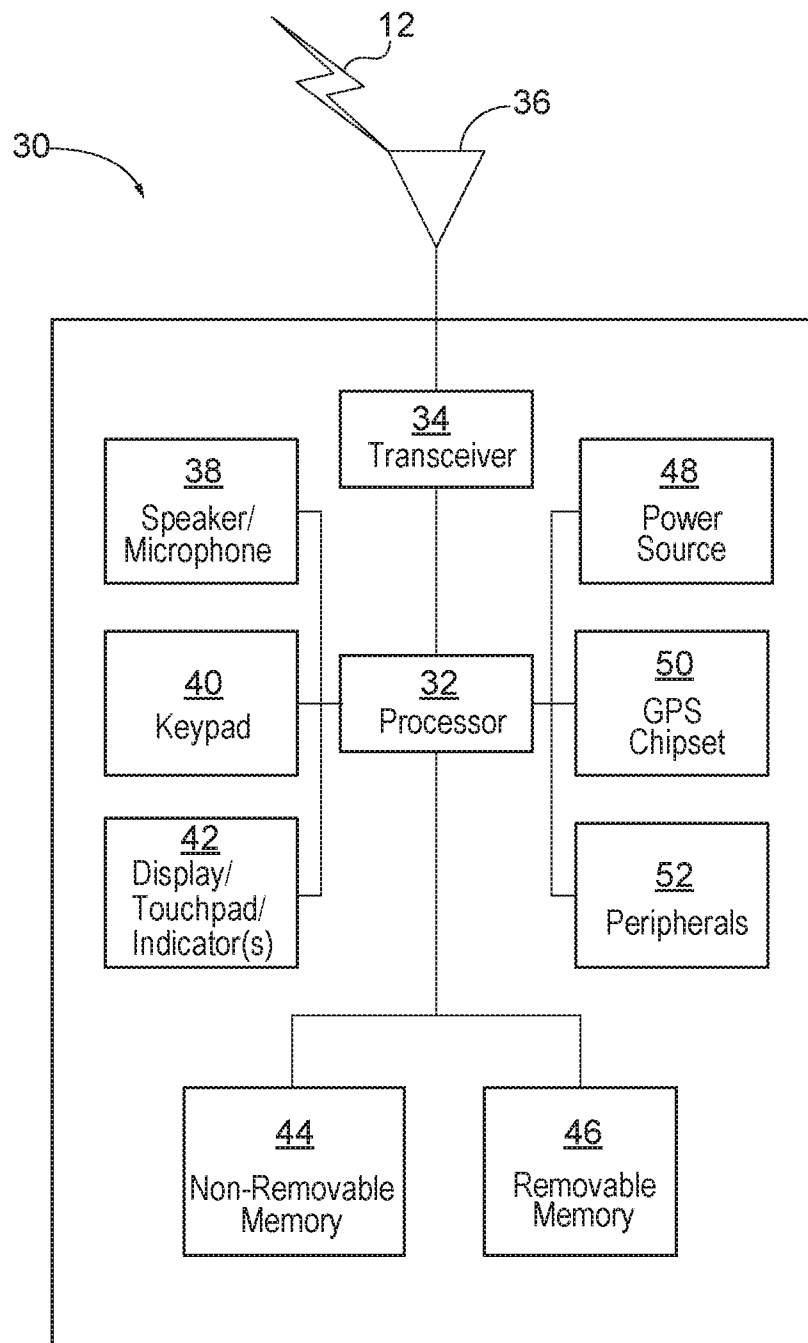


FIG. 11C

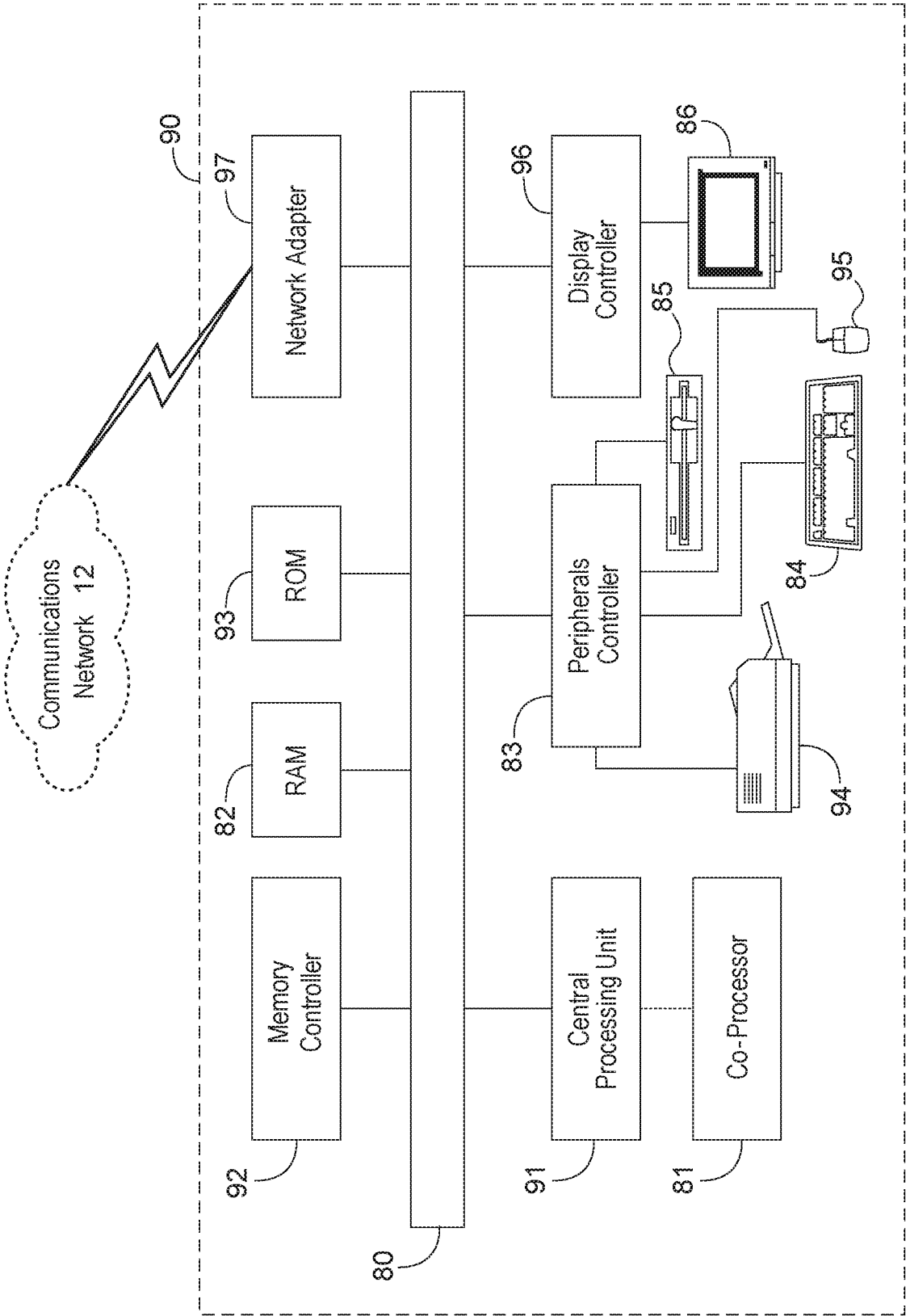


FIG. 11D

User Input Panel for

Setting Time Flexibility Related Parameters in CoAP

Please select which parameters to be included in the request/response messages and their values in different cases:

☐ Deadline1

Please input the maximal default value of Deadline1 when it is included in the request message:

☐ Deadline2

Please input the maximal default value of Deadline2 when it is included in the response message:

☐ DeadlineForNext

Please input the maximal default value of DeadlineForNext when it is included in the request message:

☐ SuspendForNext

Please input the maximal default value of SuspendForNext when it is included in the request message:

☐ DeadlineForNext

Please input the maximal default value of DeadlineForNext when it is included in the response message:

☐ SuspendForNext

Please input the maximal default value of SuspendForNext when it is included in the response message:

Submit

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/050134

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04W4/00 H04L29/08 H04L12/841
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)
H04W 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	EP 2 800 339 A1 (HUAWEI TECH CO LTD [CN]) 5 November 2014 (2014-11-05)	1-6, 8-11,13
Y	paragraphs [0050], [0079], [0113], [0130]; figure 3	7,12,14, 15
Y	----- US 2014/155024 A1 (CHETLUR MALOLAN [IN] ET AL) 5 June 2014 (2014-06-05) paragraph [0037]	7
Y	----- EP 2 309 783 A1 (PANTECH CO LTD [KR]) 13 April 2011 (2011-04-13) paragraphs [0028] - [0030]	12
Y	----- EP 2 823 687 A1 (ERICSSON TELEFON AB L M [SE]) 14 January 2015 (2015-01-14) paragraphs [0062] - [0064]; figure 1	14
	----- -/-	



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

25 November 2016

Date of mailing of the international search report

01/12/2016

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

Adhari, Hakim

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2016/050134

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2014/280885 A1 (AYANDEH SIAMACK [US] ET AL) 18 September 2014 (2014-09-18) paragraph [0025] -----	15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2016/050134

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 2800339	A1	05-11-2014	CN 103227803 A 31-07-2013
			EP 2800339 A1 05-11-2014
			US 2014359133 A1 04-12-2014
			WO 2013113275 A1 08-08-2013

US 2014155024	A1	05-06-2014	NONE

EP 2309783	A1	13-04-2011	CN 102045649 A 04-05-2011
			EP 2309783 A1 13-04-2011
			JP 2011082989 A 21-04-2011
			KR 20110039627 A 20-04-2011
			US 2011086637 A1 14-04-2011

EP 2823687	A1	14-01-2015	EP 2823687 A1 14-01-2015
			US 2015055573 A1 26-02-2015
			WO 2013131562 A1 12-09-2013

US 2014280885	A1	18-09-2014	NONE
