



(51) International Patent Classification:
H04W 4/00 (2009.01) *H04L 29/08* (2006.01)

(21) International Application Number:
PCT/US2016/050982

(22) International Filing Date:
9 September 2016 (09.09.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/215,979 9 September 2015 (09.09.2015) US

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

(72) Inventors: **DI GIROLAMO, Rocco**; 632 De Fribourg
Street, Laval, Quebec, H7K 3Y3 (CA). **LY, Quang**; 115
Sterling Drive, North Wales, PA 19454 (US). **LI, Xu**; 28
Ashford Drive, Plainsboro, NJ 08536 (US). **WANG,**
Chonggang; 9 Carlyle Court, Princeton, NJ 08540 (US).
RAHMAN, Shamim, Akbar; 6704 Conklin Road, Cote
St. Luc, Quebec, H4V 1B8 (CA). **CHEN, Zhuo**; 1397 Par-
ish Avenue, Claymont, DE 19703 (US). **CHOYI, Vinod,**
Kumar; 200 W. Elm Street, Unit 1233, Conshohocken, PA
19428 (US). **DONG, Lijun**; 6085 African Holly Trail, San
Diego, CA 92130 (US).

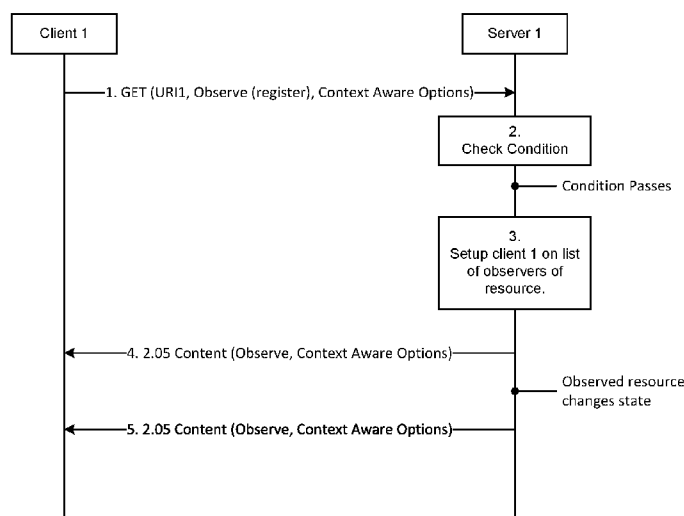
(74) Agents: **CLARE, Thomas, J.** et al.; Baker & Hostetler
LLP, Circa Centre, 12th Floor, 2929 Arch Street, Phil-
adelphia, PA 19104-2891 (US).

(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,
KW, 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.

[Continued on next page]

(54) Title: METHODS FOR ENABLING CONTEXT-AWARE COAP MESSAGING

FIG. 11



(57) Abstract: CoAP network nodes may leverage context awareness through conditional performance of CoAP methods based on options specified by clients and conditions evaluated at servers. Multiple context-aware options, and multiple parameters for each option, may be associated with conditional requests for the performance of methods. Similarly, the sending of responses and/or notifications may be based on a condition of a server context. Contexts on which context-aware options may be based include: the status of power, memory, processes, connectivity, and communications; the number resources, observers, groups; and/or the life of a cached resource. Each context-aware option may have a number of parameters, such as a specified persistence, a preferred action to be taken if the condition is not met, and a default behavior if the option is not supported.



(84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE,

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

Published:

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

METHODS FOR ENABLING CONTEXT-AWARE COAP MESSAGING

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 62/215,979, filed on September 9, 2015 entitled “Methods for Enabling Context-Aware CoAP Messaging”, the content of which is hereby incorporated by reference in its entirety

BACKGROUND

[0002] Machine-To-Machine (M2M), Internet-of-Things (IoT), and Web-of-Things (WoT) network deployments may employ group communications between nodes such as M2M/IoT/WoT servers, gateways, and devices which host M2M/IoT/WoT applications and services. Such network deployments may include, for example, constrained networks, wireless sensor networks, wireless mesh networks, mobile ad-hoc networks, and wireless sensor and actuator networks. These may use various protocols, such as Internet Engineering Task Force (IETF) RFC 7252 Constrained Application Protocol (CoAP) and IETF RFC 7390 Group Communication for the Constrained Application Protocol. CoAP is a web transfer protocol that is useful in constrained environments, e.g., environments with low-power devices and/or lossy communications. CoAP networks may include such devices as constrained devices, user mobile devices, sensor nodes, actuator nodes, medical devices, gateways, and network applications servers.

[0003] A CoAP endpoint is a logical and/or physical node in a CoAP network. Each CoAP endpoint may perform many roles. A CoAP sender is the originating endpoint of a message. A CoAP recipient is the destination endpoint of a message. A CoAP client is the originating endpoint of a request and the destination endpoint of a response. A CoAP server is the destination endpoint of a request and the originating endpoint of a response.

[0004] A CoAP resource is an object which has a type, associated data, and possibly relationships to other resources. CoAP uses a client/server model similar to Hypertext Transfer Protocol (HTTP). Resources are identified by a Uniform Resource Identifier (URI) and are operated on by a set of methods (GET, POST, PUT, and DELETE). A client requests an action by sending a message specifying the resource URI and the method. The server issues a response. The response potentially contains a representation of the resource.

[0005] An “observe” is a procedure whereby a CoAP client may receive multiple notifications as the state of a resource changes by requesting that a server register the client in a list of observers of the resource.

[0006] A CoAP management entity is a logical entity that may configure and/or manage functionality of CoAP nodes and their interaction. A CoAP proxy is an endpoint that acts both as a server towards a client, and as a client towards an origin server. A CoAP proxy forwards requests and relays back responses, possibly performing caching, namespace translation, or protocol translation in the process.

SUMMARY

[0007] Disclosed herein are methods and devices whereby CoAP network nodes may leverage context awareness through conditional performance of CoAP methods based on options specified by clients and conditions evaluated at servers. Multiple context-aware options, and multiple parameters for each option, may be associated with conditional requests for the performance of methods such as requests for resources and requests to observe resources. Similarly, the sending of responses and/or notifications may be based on a condition of a server context, where this condition is specified by the client.

[0008] Contexts on which context-aware options may be based include, but are not limited to: power status, such as charging status and remaining battery power; memory status such as amount of memory available; status of one or more processes; connectivity status, such as connectivity mode, transmission rate, and error rate; number of resources, observers of a resource, or groups; and/or life of a cached resource.

[0009] Each context-aware option may have a number of parameters, such as, but not limited to: a specified persistence, during which the associated condition is to be evaluated; a preferred action to be taken in the case that the condition is not met; and a default behavior if the option is not supported.

[0010] 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

[0011] The summary, as well as the following detailed description, is further understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there are shown in the drawings exemplary embodiments of the invention. However, the invention is not limited to the specific methods, compositions, and devices disclosed.

[0012] Figure 1 depicts abstract layers of the CoAP protocol.

[0013] Figure 2 is a call flow diagram showing examples of CoAP CON and ACK messages.

[0014] Figure 3 is a call flow diagram showing an example CoAP NON message.

[0015] Figure 4 is a call flow diagram showing an example of CoAP reliable message delivery.

[0016] Figure 5 is a call flow diagram showing an example of CoAP requests and responses.

[0017] Figure 6 shows the CoAP message format.

[0018] Figure 7 is a call flow diagram of an example of a CoAP observation registration and resource notifications.

[0019] Figure 8 is a call flow diagram showing an example of battery depletion due to non-conditional resource observation.

[0020] Figure 9 is a call flow diagram showing an example of non-conditional resource observation via cellular connectivity.

[0021] Figure 10 is a call flow diagram showing examples of using conditional requests.

[0022] Figure 11 is a call flow diagram showing an example of responding to a failure of a condition when using a conditional request.

[0023] Figure 12 is a call flow diagram showing an alternative example of responding to a failure of a condition when using a conditional request.

[0024] Figure 13 is a call flow diagram showing another alternative example of responding to a failure of a condition when using a conditional request.

[0025] Figure 14 is a flow chart of an example method for ongoing observation of a context condition.

[0026] Figure 15 is a system diagram of example graphical user interfaces operating on an M2M device and M2M gateway.

[0027] Figure 16 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.

[0028] Figure 17 is a system diagram of an example architecture that may be used within the M2M/IoT/WoT communications system illustrated in Figure 16.

[0029] Figure 18 is a system diagram of an example communication network node, such as an M2M/IoT/WoT node, gateway, or server that may be used within the communications system illustrated in Figures 16 and 17.

[0030] Figure 19 is a block diagram of an example computing system in which a node of the communication system of Figures 16 and 17 may be embodied.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0031] Disclosed herein are methods and devices whereby CoAP network nodes may leverage context awareness through conditional performance of CoAP methods based on options specified by clients and conditions evaluated at servers. Multiple context-aware options, and multiple parameters for each option, may be associated with conditional requests for the performance of methods such as requests for resources and requests to observe resources. Similarly, the sending of responses and/or notifications may be based on a condition of a server context, where this condition is specified by the client.

[0032] Contexts on which context-aware options may be based include, but are not limited to: power status, such as charging status and remaining battery power; memory status such as amount of memory available; status of one or more processes; connectivity status, such as connectivity mode, transmission rate, and error rate; number of resources, observers of a resource, or groups; and/or life of a cached resource.

[0033] Each context-aware option may have a number of parameters, such as, but not limited to: a specified persistence, during which the associated condition is to be evaluated; a preferred action to be taken in the case that the condition is not met; and a default behavior if the option is not supported.

[0034] As defined by the IETF, the standard CoAP protocol includes very limited context awareness. Specifically, certain methods may be conditioned on the presence or absence of a resource at an endpoint. The inventors observe that a rich amount of context information is

available to the CoAP layer. Such information may be exploited to alleviate network load and associated consumption of constrained resources. Through extensions of the CoAP protocol described herein, power and network bandwidth may be conserved, and endpoint processing reduced, by context-aware methods which result in the overall reduction of network traffic.

[0035] Useful context information may take many forms. It may include for example, but is not limited to, user location, time of day, neighboring users and devices, user activities, level of activity. It includes the computing environment, such as available processors, devices accessible for user input and output, network capacity, connectivity, and cost of computing. It also includes the application environment, such as location, which nodes are nearby, and social context. It includes the physical environment, such as temperature, lighting, air pressure, humidity, and noise levels, for example.

[0036] Context information may be acquired in a number of ways. For example, information may be acquired automatically via optical, chemical, electronic, or mechanical sensors, as well as by user input or the observation of network traffic.

[0037] Context-awareness may be used to determine actions taken by applications running on various network nodes such as user devices, as well as actions taken by various layers of the communication protocol stack used by any or all of the nodes of a network. This includes, for example, the application layers of various protocols, including CoAP.

[0038] Context-aware determinations may be made autonomously at a single node, or may involve interactions between nodes. For example, a smartphone may autonomously adjust the orientation of a video playback upon sensing a change in the phone's physical position. No interaction with another device is required to take this action. With orientation awareness, the smartphone is able to determine for itself whether portrait or landscape presentation is better for the user experience. In contrast, when an automobile passenger uses his or her smartphone to discover nearby Chinese restaurants via Google maps, multiple nodes may be involved in acquiring and using context information. The smartphone may, for example, provide internal GPS data to Google, which Google then applies in determining what data to send in response. Similarly, the smartphone may use a dialogue with a cell phone tower or a WiFi hotspot to better determine its position.

[0039] CoAP sits between the application and the transport protocol. As shown in Figure 1, CoAP may be considered to include two layers of operation. One CoAP layer deals with User Datagram Protocol (UDP) asynchronous messaging. The second CoAP layer deals

with request and response interactions that are carried in CoAP methods such as GET, PUT, POST, and DELETE, as well the associated response codes.

[0040] The IETF defines four types of CoAP messages. A confirmable (CON) message is retransmitted using a default timeout and exponential back-off between retransmissions. An acknowledgement (ACK) message is used to acknowledge a CON message. Figure 2 shows an example of the confirmable message with ACK of the same message ID 0x7d34. CON messages may be retransmitted, for example, until the sender receives an acknowledgement (ACK) message from the recipient which contains a message ID that matched the original message.

[0041] A non-confirmable (NON) message does not require reliable transmission. For example, each individual measurement in a stream of sensor data may be sent as a NON message. These messages are not acknowledged, but still contain a message ID for duplicate detection. An example with message ID 0x01a0 is shown in Figure 3. A reset (RST) message may be used when a recipient is not able to process a NON message.

[0042] CoAP, which relies on the UDP transport protocol, provides its own mechanism to ensure reliable message transmission. Each message includes a message ID. CON messages are retransmitted using a default timeout and exponential back-off between retransmissions, until the sender receives an ACK from the recipient with a matching message ID. Retransmission is controlled by a timeout and a retransmission counter. For each new Confirmable message, the sender chooses an initial timeout, e.g., based on the ACK_TIMEOUT transmission parameter. The sender also sets a retransmission counter to 0. If the timeout is triggered and the retransmission counter is less than some maximum, e.g., MAX_RETRANSMIT transmission parameter, then the sender: retransmits the Confirmable message with the same message ID; increments the retransmission counter; and doubles the timeout value.

[0043] If the retransmission counter equals MAX_RETRANSMIT, the message transmission is cancelled and the application may be notified. The default values for ACK_TIMEOUT and MAX_RETRANSMIT are 2 seconds and 4, respectively. An example is shown in Figure 4, where the ACK for message (Message ID = 0x7d37) is lost and the client times out and is forced to retransmit.

[0044] The list of relevant parameters used to control message transmission and limit congestion are shown in Table 1. Also included are the default values for these parameters.

Table 1
Message Transmission Parameters

Name	Description	Default Value
ACK_TIMEOUT	Base retransmission timeout value	2 seconds
ACK_RANDOM_FACTOR	Retransmission timeout multiply factor. Initial retransmission timeout = random between ACK_TIMEOUT and (ACK_TIMEOUT * ACK_RANDOM_FACTOR)	1.5
MAX_RETRANSMIT	Maximum number of retransmissions for a confirmable message	4
NSTART	Number of simultaneous outstanding interactions that a client maintains to a given server	1

[0045] Note that the specification for standard CoAP allows dynamically changing these values to better match the intended application. However it does not describe any method for accomplishing this.

[0046] 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.

[0047] A request is carried in a confirmable (CON) or a non-confirmable (NON) message. If immediately available, the response to a request is carried in the resulting acknowledgement (ACK) message, as shown in Figure 5. This is called a piggy-backed response. If a request is sent in a non-confirmable message, then the response is sent using a new non-confirmable message.

[0048] CoAP messages are encoded in a simple binary format, as 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 is 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.

[0049] The fields in the header are as follows. Version (Ver) is a 2-bit unsigned integer indicating the CoAP version number. Type (T) is 2-bit unsigned integer indicating whether the message is Confirmable (0), Non-confirmable (1), an Acknowledgement (2), or a Reset (3). Token Length (TKL) is a 4-bit unsigned integer indicating the length of the variable-length Token field. Normally TKL is 0-8 bytes. Token lengths of 9-15 bytes are reserved. If TKL shows a Token field length of 9-15 bytes, it must be processed as a message format error.

[0050] Code is an 8-bit unsigned integer, split into a 3-bit class (the most significant bits) and a 5-bit detail (the 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 the case of a request, the code field indicates the request method. In the case of a response, the code field indicates a response code.

[0051] CoAP code values are included in the CoAP code registry, as shown in Table 2.

Table 2
CoAP Code Values

Code Value	Description
0.00	Indicates an Empty message
0.01	Indicates a GET request
0.02	Indicates a POST request
0.03	Indicates a PUT request
0.04	Indicates a DELETE request
0.04-0.31	Unassigned Method request codes
1.00-1.31	Reserved
2.00	Unassigned response code
2.01	Response Code – Created
2.02	Response Code – Deleted

Code Value	Description
2.03	Response Code – Valid
2.04	Response Code - Changed
2.05	Response Code - Content
2.06-2.31	Unassigned response code
3.00-3.31	Response Code – reserved for future use
4.00	Response Code – Bad Request
4.01	Response Code – Unauthorized
4.02	Response Code – Bad Option
4.03	Response Code – Forbidden
4.04	Response Code – Not Found
4.05	Response Code – Method Not Allowed
4.06	Response Code – Not Acceptable
4.07-4.11	Unassigned response code
4.12	Response Code – Precondition Failed
4.13	Response Code – Request Entity Too Large
4.14	Unassigned response code
4.15	Response Code – Unsupported Content-Format
4.16-4.31	Unassigned response code
5.00	Response Code – Internal Server Error
5.01	Response Code – Not Implemented
5.02	Response Code – Bad Gateway
5.03	Response Code – Service Unavailable
5.04	Response Code – Gateway Timeout
5.05	Proxying Not Supported
5.06-5.31	Unassigned response code
6.00-7.31	Reserved

[0052] Message ID is a 16-bit unsigned integer in network byte order. A message ID is used for the detection of message duplication, and to match messages of type Acknowledgement/Reset to messages of type Confirmable/Non-confirmable.

[0053] The fields after the header are the token field and the options. The token is 0 to 8 bytes, as given by the token length field. The token value is used to correlate requests and responses. An option can be followed by the end of the message, by another option, or by the payload marker (1111 1111) and the payload.

[0054] 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. The option value can be empty, opaque, Uint (a non-negative integer), or a string.

[0055] 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. 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. For example, odd option numbers indicate a critical option, while even option numbers indicate an elective option. The difference between these is how an unrecognized option is handled by an endpoint. Upon reception, unrecognized options of class "elective" must be silently ignored. Unrecognized options of class "critical" that occur in a confirmable request must cause the return of a 4.02 (Bad Option) response. Unrecognized options of class "critical" that occur in a Confirmable response, or piggy-backed in an Acknowledgement, must cause the response to be rejected. Unrecognized options of class "critical" that occur in a non-confirmable message must cause the message to be rejected.

[0056] 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 part of the cache-key. If all NoCacheKey bits are 1, it is not.

[0057] An option that is repeatable may be included one or more times in a message. Table 3 shows two examples of properties of CoAP options, Proxy-scheme and Size1.

Table 3
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)

[0058] 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 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.

[0059] CoAP defines four methods, GET, POST, PUT and DELETE. The GET method retrieves a representation for the information that currently corresponds to a resource identified by the request URI. Upon success a 2.05 (Content) or 2.03 (Valid) response code should be present in the response. 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.

[0060] 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. The DELETE method requests that the resource identified by the request URI be deleted. 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.

[0061] The request-response basis of the CoAP core protocol does not work well when a client is interested in having the resource representation over a period of time. The IETF CoRE

Working Group draft standard regarding observing resources in CoAP, draft-ietf-core-observe-16, suggests extending the CoAP core protocol with a mechanism for a CoAP client to "observe" a resource on a CoAP server. CoAP observe is a subscribe-notification mechanism where one request results in multiple responses. The client registers its interest in a resource by issuing an extended GET request to the server. This request causes the server to add the client to the list of observers of the resource.

[0062] Figure 7 shows an example of a CoAP client registering its interest in a resource called "/temperature". The server sends a response with the current state of the resource upon registration. Subsequently, upon every state change of the resource, the CoAP server sends a notification, i.e., an additional CoAP response, to the CoAP client, with the new representation.

[0063] The client correlates the notifications in response to the original request through the token carried in the response message header. A client remains on the list of observers until it either: deregisters (cancels its observe registration); rejects a notification from the server; or fails to respond to a confirmable notification message.

[0064] In draft-li-core-conditional-observe-05, the IETF CoRE Working Group has addressed the notion of a conditional observe, whereby the server sends notifications only if a condition passes. In this scheme, the condition supplied by the client may be based on the value/content of the resource, e.g., resource value >x, resource value <x, resource value =x, $x_1 < \text{resource value} < x_2$, change in resource value exceeds x, etc. Similarly the condition may be based on a time interval between notifications, e.g., greater than a minimum, or less than a maximum.

[0065] Some limited context-awareness is included in the standard CoAP base protocol. In particular, standard CoAP allows for conditional request operations, where a client can ask a server to perform the request only if certain conditions specified by the client are fulfilled. If the given condition is not fulfilled, the server must not perform the requested method. Instead, the server must respond with the 4.12 (Precondition Failed) response code. On the other hand, if the condition is fulfilled, the server performs the request method as if the conditional request were not present.

[0066] The "condition" is conveyed to the server via a set of options. Standard CoAP defines two such conditional request options: If-Match and If-None-Match. Using If-Match, the request is conditioned on the current existence or value of an entity-tag (ETag) for one or more representations of a target resource. That is, the condition is fulfilled if the ETag of the target resource matches the supplied ETag in the condition request. Note that the client may specify an

empty If-Match, in which case the condition is fulfilled if any representation exists for the target resource.

[0067] Using If-None-Match, the request is conditioned on the non-existence of the target resource. If the target resource does exist, then the condition is not fulfilled.

[0068] Figure 8 shows a use case where a client is trying to observe a resource from a server. The client is not interested in every resource state change, and is willing to forego a notification, particularly if the server battery status is “poor”. At time t1, the server battery status changes from “good” to “poor”. During subsequent changes in the observed resource, the notification transmissions significantly drain the server battery further. Note that the battery drain is even more significant if the observed resource has a large data size and is sent using a block transfer between the server and the client.

[0069] Significantly, there is no way for the client to inform the server that the server need only send the response if its battery status is “good.” Further, as part of the current CoAP observe specification, a server may decide to autonomously skip a notification, but only if it knows that it will send another notification soon. The server decision is not based on the battery status. Further still, there is no way for the server to inform the client that a response has not been sent due to the battery status.

[0070] Figure 9 shows a case where a client is trying to GET a large resource from a server. The server is capable of connectivity through cellular and through WiFi. By default, the server will always try to connect through WiFi, but it will occasionally revert to cellular if it does not find a WiFi access point. The client is concerned about cost and power consumption, and would prefer that the server either use WiFi to transfer retrieved resources that are not deemed critical, or refrain from transferring these resources, if this is not possible.

[0071] At time T1, the server is not able to connect through WiFi, and relies on cellular to communicate. The client issues a GET request to obtain a resource that it deems critical, and the server transfers this resource using a block transfer. At time T2, the client issues a GET request for a resource that it deems non-critical. As the server is using cellular, it would prefer that the server not respond. Unfortunately the server transfers the large resource using a block transfer.

[0072] As a result, there are unwanted transmissions from both the client and the server, e.g., for the block transfer of Resource2. The situation occurs since there is no way for the client to inform the server that it only wants the resource if the server is using WiFi

connectivity, and there is no way for the server to inform the client that a response has not been sent because the server is using cellular.

[0073] Significantly, a large amount of context information is available to the CoAP layer. The CoAP layer is aware of connectivity through the method of communication used by the endpoint, such as cellular, WPAN, WiFi, and the like. The CoAP layer has the kinds of system context knowledge that is typically available through the operating system, such as application status, memory status, battery status, time, etc. The CoAP layer is aware of the context of resources, such as the number of resources, the size of resources, the age (freshness) of a cached resource, the number of group resources, the group memberships, etc. Similarly, the CoAP layer is aware of such parameters as the transmission rate at the CoAP layer.

[0074] Failure to leverage such context information at servers and proxies may lead to inefficient use of power, processing time, and network bandwidth. For example, a method may be performed at a server even when the status of a prerequisite condition known to the CoAP server is not met. In such a case, the server is only taking advantage of its awareness of the presence or absence of a resource to conditionally perform a method. Similarly, a server may generate observe notifications even when the status of a prerequisite condition known to the CoAP server is not met. In such a case, the server may generate notifications at inopportune times, for instance when battery is low, memory is low, connectivity is poor, etc. This leads to the server sending notifications when it is not necessary.

[0075] Greater efficiency may be achieved by performing requested methods conditionally based on conditions specified by a client and verified through evaluation of context information at a server versus the specified conditions. Similarly, greater efficiency may be achieved by sending observe notifications based on a condition tied to a server context specified by the client. These may be achieved by extending CoAP to include context-aware options that allow conditional observe and conditional requests, whereby a client may specify: the condition to check; the default behavior if the condition is not supported; and/or a persistence of the condition. Such context-aware options may include, but are not limited to, battery status, connectivity, and transmission rate, for example.

[0076] Below, context-aware operations are often described as occurring between client and server. However, it will be appreciated that these techniques may be applied to any endpoint. For example, the operation of proxies may be made context-aware based on these principles.

[0077] The general procedure for enhanced context-aware conditional methods and enhanced context-aware observe are the same. First, a client issues a request with context-aware option to a server. Along with the request, the client sends some default action to be taken if context-awareness is not supported by the recipient. Next, if the recipient supports context-awareness, the recipient verifies the context, and processes the method accordingly. If the recipient does not support context-awareness, the recipient may notify the client and/or perform some default action.

[0078] It will be appreciated that any form of context awareness that is applicable and available to the CoAP protocol may be used. As mentioned above, for example, a CoAP endpoint generally has information available related to connectivity, system status, resources, and CoAP transmissions.

[0079] Similarly, default behaviors and persistence requirements may take many forms. For example, a conditional request may include a default behavior flag. If a server cannot or will not support the context awareness functionality of a given request, and the default behavior flag is set to false, then the server should not execute the method. On the other hand, if the default behavior flag is set to true, then the server should nonetheless execute method, even though the server is unable to test the context condition.

[0080] The persistence of context-aware conditions may be specified by a client. For example, a client may specify how long the server persists in checking the condition. If the server initially evaluates the condition as false, it may continue checking the condition for a period of time, e.g., a “persistence.” If the condition becomes true during this persistence time, then the requested method may then be executed.

[0081] Figure 10 is a call flow of an example use of an enhanced conditional request. Steps 1-4 apply to the case where the context-aware condition evaluates to true, while steps 5-10 apply to a case where the context-aware condition initially evaluates to false. The client issues a method request message 1 to server 1 which contains one or more context-aware options. For example, the method request may be conditioned on the battery status of server 1. The client may further specify the default behavior flag is false, and allow a persistence of 150 sec. For example, message 1 may contain the following:

Battery-Status = (e.g. “good”)

Battery-Status-Default = (e.g. “False”)

Battery-Status-Persistence = (e.g. 150 sec)

[0082] In step 2, server 1 checks whether the condition is met. In this case the conditions are met. In step 3 the method is executed. Server 1 then sends a method response message 4 back to the client. Message 4 contains the appropriate response code to the client. In addition, server 1 may optionally return one or more details of the context information at server 1 to the client. For instance, server 1 may return the percentage of battery power remaining, e.g., 75%, or an estimate of the time until battery depletion, e.g., 321 days.

[0083] The client issues a new method request message 5 to server 2. Message 5 contains one or more context-aware options. For example, message 5 may be conditioned on the CoAP transmission rate towards the client. Furthermore, the client may specify the default behavior flag as true, and allow for a persistence of 60 sec. For example, message 5 may contain the following:

Min-Transmission-Rate = (e.g., 10 Kbps)

Min-Transmission-Rate-Default = (e.g., "True")

Min-Transmission-Rate-Persistence = (e.g., 60 sec)

[0084] In step 6, server 2 checks whether the conditions are met. In this case, the condition is not presently met. Next server 2 sends an acknowledgement message 7, which acknowledges the request and provides an indication of the related context information of server 2. For example, server 2 may include a measured transmission rate of 7.5 kbps in message 2.

[0085] In step 8, server 2 starts a timer with a duration according to the parameter Transmission-Rate-Persistence to wait for the condition to be met. In the example of Figure 10 the condition is met before the timer expiry. In step 9, server 2 stops the timer, performs the request method. Accordingly, server 2 then sends a method response message 10 back to the client. Message 10 may include context information about server, e.g., server 2 may return a measured transmission rate of 11 kbps. Not shown in Figure 10, server 2 may then purge the method request.

[0086] Figure 11 is a call flow of an example case where the condition of an enhanced conditional observe procedure evaluates to true. Client 1 issues an extended GET message 1 to server 1. Message 1 specifies that client 1 wishes to conditionally observe a requested resource. For example the condition may be based on the size of the resource. The client also specifies the default behavior flag as true. As a result, message 1 may contain the following:

Resource-Size = (e.g. 750 bytes)

Resource-Size-Default = (e.g. "True")

[0087] In step 2, server 1 checks whether the condition is met. In the example of Figure 11, the condition is met. Therefore in step 3, server 1 includes client 1 on the list of observers for the resource. Server 1 then sends a 2.05 content response message 4 to client 1. Message 4 may include an indication of the context, e.g., that the size of the requested resource is 500 bytes. Later, upon observing a state change in the observed resource, server 1 sends a second 2.05 content response, message 5, to client 1. Message 5 may also contain an indication of the context status.

[0088] Figures 12, 13, and 14 are call flows of example cases where the condition of an enhanced conditional observe procedure evaluates to false, illustrating three different ways a server may respond to such a condition.

[0089] Figure 12 is a call flow of an example case where the condition of an enhanced conditional observe procedure evaluates to false, where the server does not include the client in a list of the observers of a resource, and the server notifies the client of its decision. Client 1 issues an extended GET message 1 to server 1. Message 1 specifies that client 1 wishes to conditionally observe a requested resource. For example, the condition may be based on the minimum number of resources at the server. The client also specifies the default behavior flag as true. For example, message 1 may contain the following:

Number-Resources = 30

Number-Resources-Default = True

[0090] In step 2, server 1 checks whether the condition is met. The condition is not met. In this example, server 1 does not include the client in the list of observers of the requested resource. Server 1 responds to client 1 with an acknowledgement message 3. Message 3 includes an indication that the observe was cancelled/denied. Message 3 also includes an indication of the context at server 1, e.g., that the number of resources is 18, not 30 as requested.

[0091] Figure 13 is a call flow of an example case where the condition of an enhanced conditional observe procedure evaluates to false, where server includes the client in the list of observers to the resource, but it also notifies the client about the status of the context. The client is then free to cancel the observe. As in the example of Figure 12, client 1 issues an extended GET message 1 to server 1, specifying that client 1 wishes to observe a certain resource on the condition that there are at least 30 resources at the server, and further specifying a default behavior flag set to true. In step 2, server 1 checks and determines that the condition is not met.

[0092] In step 3, server 1 includes client 1 in a list of observers of the requested resource. Server 1 then sends a 2.05 content message 4 back to client 1. Message 4 may include

the state of the relevant context at server 1, e.g. 18. In response to message 4, client 1 may elect to cancel the observation of the resource. In the example of Figure 13, based on the returned context status in message 4, client 1 sends a method request message 5 to cancel the observation.

[0093] Figure 14 is a call flow of an example case where the condition of an enhanced conditional observe procedure evaluates to false, where the server includes the client in the list of observers to the resource, but only sends notifications to the client when the condition evaluates to true. Again as in the example of Figure 12, client 1 issues an extended GET message 1 to server 1, specifying that client 1 wishes to observe a certain resource on the condition that there are at least 30 resources at the server, and further specifying a default behavior flag set to true. In step 2, server 1 checks and determines that the condition is not met. In step 3, server 1 includes client 1 in a list of observers of the requested resource. Server 1 then sends an acknowledgement 4 back to client 1, which may include an indication of the context condition at server 1, e.g., that the number of resources is 18. ACK message 4 may also include an observe option.

[0094] In each of steps 5 and 6, server 1 observes a change of state on the observed resource, but the server does not send a notification to the client because the context-aware condition still evaluates to False.

[0095] Subsequently the context of server 1 changes in step 7. More than 30 resources are now present. In response to a change of state of the observed resource, server 1 sends a 2.05 content response message 8 to client 1, since the context-aware condition now evaluates to true. Later, after detecting another change of state of the observed resource (in step 9), server 1 sends another 2.05 content response message, message 10, to client 1, since the condition is still true.

[0096] Note that the alternatives described above may also be implemented in the form of one or more observe options exchanged between the client and server. For example, the client may use the initial GET method to further request that the server use a preferred alternative for handling the case the condition fails.

[0097] Furthermore, note that the examples of Figures 11-14 assume that the actions upon receiving the observe registration depend on the initial evaluation of the context-aware condition, as shown as step 2 in each figure. Alternatively, step 2 may be omitted, whereby a server may always accept a conditional observe request and will always put the client on the list of observers for that resource. However, the server may monitor the condition thereafter, only send notifications when the resource changes state and the condition is met.

[0098] Table 4 lists examples of potential new CoAP options to support context-awareness, and how these options may be used. The options in Table 4 are not exhaustive and may be extended. It should be understood that equivalent options may be defined for all forms of context awareness. Furthermore note that the Options introduced in Table 4 can be used individually, unless indicated herein to the contrary, or in combination.

Table 4
New CoAP Options to support Context-Aware Solutions

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Battery-Status	Condition associated with the battery status: Values: good/poor (or percentage remaining, or time until depletion)	Battery status condition to check at recipient endpoint	Status of battery at receiving endpoint
Battery-Status-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support battery status awareness True – execute the method if the endpoint does not support the battery status awareness	Default behavior at receiving endpoint if it does not support battery status awareness	Not Applicable
Battery-Status-Persistence	Duration of time the endpoint persists in checking the condition on battery status	Signal to receiving endpoint to check the battery status condition for the persistence time	Not Applicable
Memory-Status	Condition associated with the free memory status: Values: high/medium/low (or percentage)	Memory status condition to check at recipient endpoint	Status of free memory at receiving endpoint

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Memory-Status-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support memory status awareness True – execute the method if the endpoint does not support the memory status awareness	Default behavior at receiving endpoint if it does not support memory status awareness	Not Applicable
Memory-Status-Persistence	Duration of time the endpoint persists in checking the condition on memory status	Signal to receiving endpoint to check the memory status condition for the persistence time	Not Applicable
Process-Status	Condition associated with the presence of a running process at server (value is process ID or process name)	Process name or Process ID to check	Not Applicable
Process-Status-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support process status awareness True – execute the method if the endpoint does not support the process status awareness	Default behavior at receiving endpoint if it does not support process status awareness	Not Applicable
Process-Status-Persistence	Duration of time the endpoint persists in checking the condition on process status	Signal to receiving endpoint to check the process status condition for the persistence time	Not Applicable

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Connectivity-Status	Condition associated with the connectivity status of the server (value WiFi, Cellular, WPAN, Wired)	Condition to check for connectivity	Actual connectivity in use by server
Connectivity-Status-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support connectivity status awareness True – execute the method if the endpoint does not support the connectivity status awareness	Default behavior at receiving endpoint if it does not support connectivity status awareness	Not Applicable
Connectivity-Status-Persistence	Duration of time the endpoint persists in checking the condition on connectivity status	Signal to receiving endpoint to check the connectivity status condition for the persistence time	Not Applicable
Min-Number-Resources	Condition associated with the minimum number of resource at a server (value integer)	Check if server has at least Min-Number-Resources	Actual number of resources at server
Min-Number-Resources-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support resource awareness True – execute the method if the endpoint does not support resource awareness	Default behavior at receiving endpoint if it does not support resource awareness	Not Applicable

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Min-Number-Resources-Persistence	Duration of time the endpoint persists in checking the condition on minimum number of resources	Signal to receiving endpoint to check the number of resources against condition for the persistence time	Not Applicable
Max-Number-Resources	Condition associated with the maximum number of resource at a server (value integer)	Check if server has at most Max-Number-Resources	Actual number of resources at server
Max-Number-Resources-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support resource awareness True – execute the method if the endpoint does not support resource awareness	Default behavior at receiving endpoint if it does not support resource awareness	Not Applicable
Max-Number-Resources-Persistence	Duration of time the endpoint persists in checking the condition on maximum number of resources	Signal to receiving endpoint to check the number of resources against condition for the persistence time	Not Applicable
Number-Resources (See Note 1)	Condition associated with the number of resource at a server equaling a certain value (Value: integer)	Check if server has exactly Number-Resources	Actual number of resources at server

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Number-Resources-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support resource awareness True – execute the method if the endpoint does not support resource awareness	Default behavior at receiving endpoint if it does not support resource awareness	Not Applicable
Number-Resources-Persistence	Duration of time the endpoint persists in checking the condition on number of resources	Signal to receiving endpoint to check the number of resources against condition for the persistence time	Not Applicable
Min-Life	Condition associated with the minimum life of a cached resource (value integer)	Check if resource at server has at least Min-Life seconds before expiry	Actual age of resource at server
Min-Life-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support cache age awareness True – execute the method if the endpoint does not support cache age awareness	Default behavior at receiving endpoint if it does not support cache age awareness	Not Applicable
Min-Life-Persistence	Duration of time the endpoint persists in checking the condition on resources cache age	Signal to receiving endpoint to check resource cache age for the persistence time	Not Applicable

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Min-Observers (See Note 2)	Condition associated with the number of observers to a resource (Value: Integer)	Check at recipient endpoint to see number of observers of list of observers > Min-Observers	Current number of observers on list of observers for resource
Min-Observers-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support resource awareness True – execute the method if the endpoint does not support resource awareness	Default behavior at receiving endpoint if it does not support resource awareness	Not Applicable
Min-Observers-Persistence	Duration of time the endpoint persists in checking the condition on number of observers	Signal to receiving endpoint to check the number of observers condition for the persistence time	Not Applicable
Max-Groups (See Note 1, Note 2)	Condition associated with the number of multicast groups at the server (Value: integer)	Check at recipient endpoint to see if number of groups < Max-Groups	Current number or groups at the server
Max-Groups-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support resource awareness True – execute the method if the endpoint does not support resource awareness	Default behavior at receiving endpoint if it does not support resource awareness	Not Applicable

CoAP Options	Descriptions	Meaning in a Request	Meaning in a Response
Max-Groups-Persistence	Duration of time the endpoint persists in checking the condition on number of groups	Signal to receiving endpoint to check the number of groups condition for the persistence time	Not Applicable
Min-Transmission-Rate (See Note2)	Condition associated with the minimum CoAP transmission rate	Transmission rate condition to check at recipient endpoint	Current transmission rate at receiving endpoint
Min-Transmission-Rate-Default	Default behavior at the server if the context awareness is not supported. Values: False - do not execute the method if endpoint does not support transmission rate awareness True – execute the method if the endpoint does not support the transmission rate awareness	Default behavior at receiving endpoint if it does not support transmission rate awareness	Not Applicable
Min-Transmission-Rate-Persistence	Duration of time the endpoint persists in checking the condition on transmission rate	Signal to receiving endpoint to check the transmission rate condition for the persistence time	Not Applicable

Notes

Note 1: For this conditional option, the recipient endpoint knows that the condition targets the server and not a specific resource on the server. For instance a conditional option “Number-Resources” asks the target to check the number of resources at the server.

Note 2: Only one variation of this conditional option is shown (Min-* or Max-*). It will be appreciated many variations of the option are possible, such as those shown, for example, for Min-Number-Resources, Max-Number-Resources, and Number-Resources.

[0099] Figure 15 shows example graphical user interfaces (GUIs) operating on an M2M/IoT/WoT CoAP device and CoAP gateway. The context-aware procedures described herein may be observed and controlled by user in a number of ways via GUIs. For example, when CoAP endpoints on the device and gateway exchange CoAP messages, a sniffer may be used to see what messages are exchanged. Alternatively, if there is a GUI interface on the device or gateway, the GUI interface may be used to display and configure options and parameters associated with the context-aware procedures of the endpoints. The GUI may use filters to select what information is displayed. Similarly, the GUI may be used to 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 context-aware CoAP operations. For example, the GUI may show that a conditional observe has been requested and/or acted upon.

[00100] 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.

[00101] As shown in Figure 16, 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 nodes (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.

[0100] Referring to Figure 17, 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 comprise servers, computers, devices, or the like. 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.

[0101] 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' in the infrastructure domain and the underlying communication network 12. 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 comprise servers, computers, devices, virtual machines (e.g., cloud computing/storage farms, etc.) or the like.

[0102] Referring also to Figure 17, 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, e.g., network 12, in connection with the services that the service layers 22 and 22' provide.

[0103] 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'.

[0104] Generally, a service layer, such as the service layers 22 and 22' illustrated in Figures 16 and 17, defines 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) (e.g., 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 18 or Figure 19 described below.

[0105] The service layer 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 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.

[0106] 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.

[0107] Figure 18 is a block diagram of an example hardware/software architecture of a node of a network, such as one of the devices illustrated in Figures 2-5, and/or 7-17 which may operate as an M2M server, gateway, device, or other node in an M2M network such as that illustrated in Figures 16 and 17. As shown in Figure 18, the 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 node 30 may include any sub-combination of the foregoing elements while remaining consistent with an

embodiment. This node may be adapted to implement aspects of conditional request and context-aware response functionality described herein.

[0108] 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 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.

[0109] As shown in Figure 18, 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, e.g., as described in reference to Figures 2-5, 7-15, and/or in the claims. While Figure 18 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.

[0110] The transmit/receive element 36 may be configured to transmit signals to, or receive signals from, other 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.

[0111] In addition, although the transmit/receive element 36 is depicted in Figure 18 as a single element, the node 30 may include any number of transmit/receive elements 36. More specifically, the node 30 may employ MIMO technology. Thus, in an embodiment, the node 30 may include two or more transmit/receive elements 36 (e.g., multiple antennas) for transmitting and receiving wireless signals.

[0112] 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 node 30 may have multi-mode capabilities. Thus, the transceiver 34 may include multiple transceivers for enabling the node 30 to communicate via multiple RATs, such as UTRA and IEEE 802.11, for example.

[0113] 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 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, for example, of an M2M condition, conditional request, or conditional response.

[0114] 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 node 30. The power source 48 may be any suitable device for powering the 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.

[0115] 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 node 30. It will be appreciated that the node 30 may acquire location information

by way of any suitable location-determination method while remaining consistent with an embodiment.

[0116] 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., finger print) sensors, an e-compass, a satellite transceiver, a sensor, 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.

[0117] 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.

[0118] Figure 19 is a block diagram of an exemplary computing system 90 which may also be used to implement one or more nodes of a network, such as the devices illustrated in Figures 2-5, and/or 7-17, which may operate as an M2M server, gateway, device, or other node in an M2M network such as that illustrated in Figures 16 and 17. 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. 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 M2M functions. This node may be adapted to implement aspects of conditional request and context-aware response functionality described herein.

[0119] 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.

[0120] 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.

[0121] 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.

[0122] 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.

[0123] 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 16 and Figure 17, 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 reference to Figures 2-5, 7-15, and/or in the claims.

[0124] 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 (e.g., program code) stored on a computer-readable storage medium. Such instructions, when executed by a machine,

such as a computer, server, M2M terminal device, M2M 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 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, CDROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other physical medium that can be used to store the desired information and that can be accessed by a computer.

[0125] 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.

[0126] 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 structural elements that do not differ from the literal language of the claims, or if they include equivalent structural elements with insubstantial differences from the literal languages of the claims.

We claim:

1. A node comprising a processor, a memory, and communication circuitry, the node being connected to a communications network via its communication circuitry, the node further comprising computer-executable instructions stored in the memory of the node which, when executed by the processor of the node, cause the node to:
 - a. receive a request comprising a CoAP method and a context-aware option;
 - b. evaluate the context-aware option based at least in part upon a condition at the node; and
 - c. conditionally perform the method if the condition at the node accords with the context-aware option.
2. The node of claim 1, wherein the computer-executable instructions cause the node to further:
 - a. respond to the request with a response comprising an indication of the condition at the node.
3. The node of claim 1, wherein the computer-executable instructions cause the node to further:
 - a. continue to evaluate the context-aware option for a period of time according to a parameter of the request.
4. The node of claim 1, wherein the computer-executable instructions cause the node to further:
 - a. include a sender of the request on a list of observers for a resource, where the request comprises a request to observe the resource.
5. The node of claim 4, wherein the computer-executable instructions cause the node to further:
 - a. if the condition does not accord with the context-aware option, to respond to the request with an ACK message, where the ACK includes an indication that the request to observe is cancelled.

6. The node of claim 4, wherein the computer-executable instructions cause the node to further:
 - a. cancel the observe according to a request to cancel the observe; and
 - b. if the condition does not accord with the context-aware option, respond to the request with an indication that context-aware option is not met by the condition.
7. The node of claim 4, wherein the computer-executable instructions cause the node to further:
 - a. report a change of state in the resource only when the condition accords with the context-aware option.
8. The node of claim 1, where the condition at the node comprises:
 - a. a power status, a memory status, a status of a process, a connectivity status, a transmission rate, a transmission error rate, a number of resources, a number of observers, a number of groups, and/or a life of a cached resource.
9. The node of claim 1, wherein the computer-executable instructions cause the node to further act upon a parameter of the context-aware option, where the request further comprises the parameter, and where the parameter comprises a specified persistence of the context-aware option, a preferred action to be taken in the case that the condition is not met, or a default server behavior requested if the option is not supported.
10. The node of claim 1, wherein the computer-executable instructions cause the node to further:
 - a. provide a graphical user interface whereby a user may see and/or configure the context-aware option and/or the context.
11. A method performed by a node of a communications network, the method comprising:
 - a. receiving a request comprising a CoAP method and a context-aware option;
 - b. evaluating the context-aware option based at least in part upon a condition at the node; and
 - c. conditionally performing the method if the condition at the node accords with the context-aware option.

12. The method of claim 11, further comprising:
 - a. responding to the request with a response comprising an indication the condition at the node.
13. The method of claim 11, further comprising:
 - a. continuing to evaluate the context-aware option for a period of time according to a parameter of the request.
14. The method of claim 11, further comprising:
 - a. including a sender of the request on a list of observers for a resource, where the request comprises a request to observe the resource.
15. The method of claim 14, further comprising:
 - a. if the condition does not accord with the context-aware option, responding to the request with an ACK message, where the ACK includes an indication that the request to observe is cancelled.
16. The method of claim 14, further comprising:
 - a. cancel the observe according to a request to cancel the observe; and
 - b. if the condition does not accord with the context-aware option, responding to the request with an indication that context-aware option is not met by the condition.
17. The method of claim 14, further comprising:
 - a. reporting a change of state in the resource only when the condition accords with the context-aware option.
18. The method of claim 11, where the condition at the node comprises:
 - a. a power status, a memory status, a status of a process, a connectivity status, a transmission rate, a transmission error rate, a number of resources, a number of observers, a number of groups, and/or a life of a cached resource.

19. The method of claim 11, further comprising acting upon a parameter of the context-aware option, where the request further comprises the parameter, and where the parameter comprises a specified persistence of the context-aware option, a preferred action to be taken in the case that the condition is not met, or a default server behavior requested if the option is not supported.
20. The method of claim 11, further comprising:
 - a. providing a graphical user interface whereby a user may see and/or configure the context-aware option and/or the context.

FIG. 1

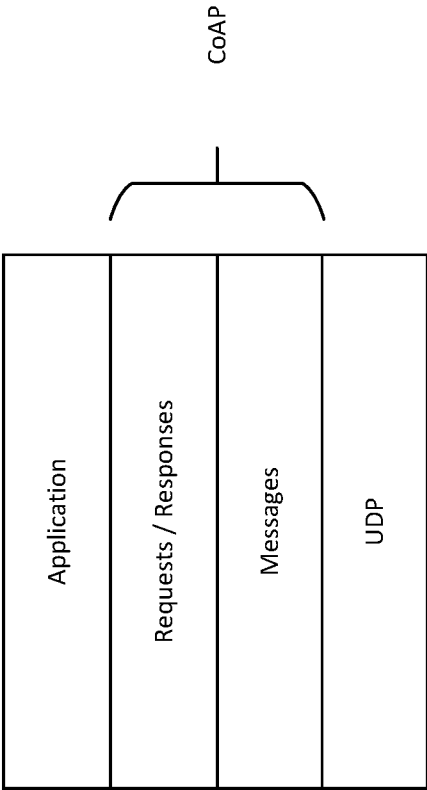


FIG. 2

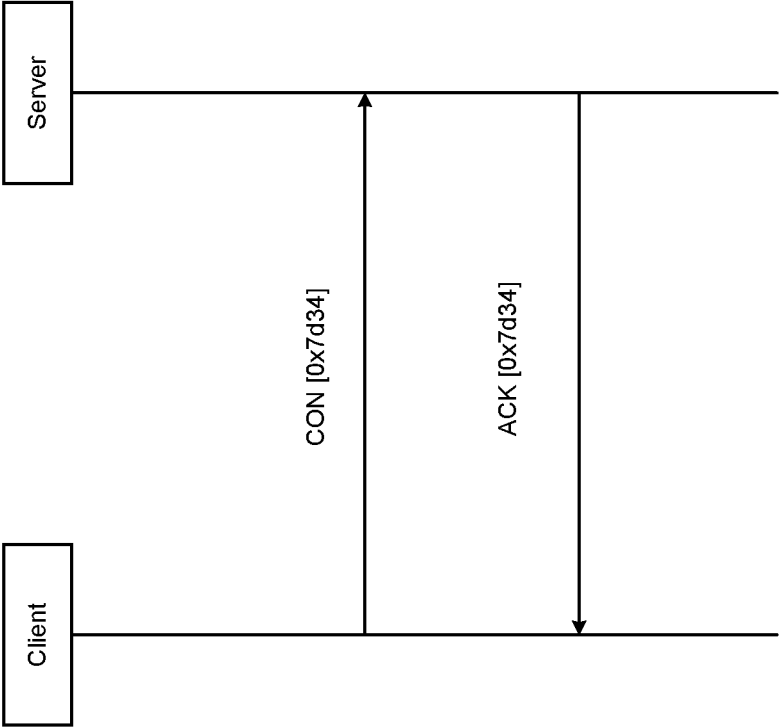


FIG. 3

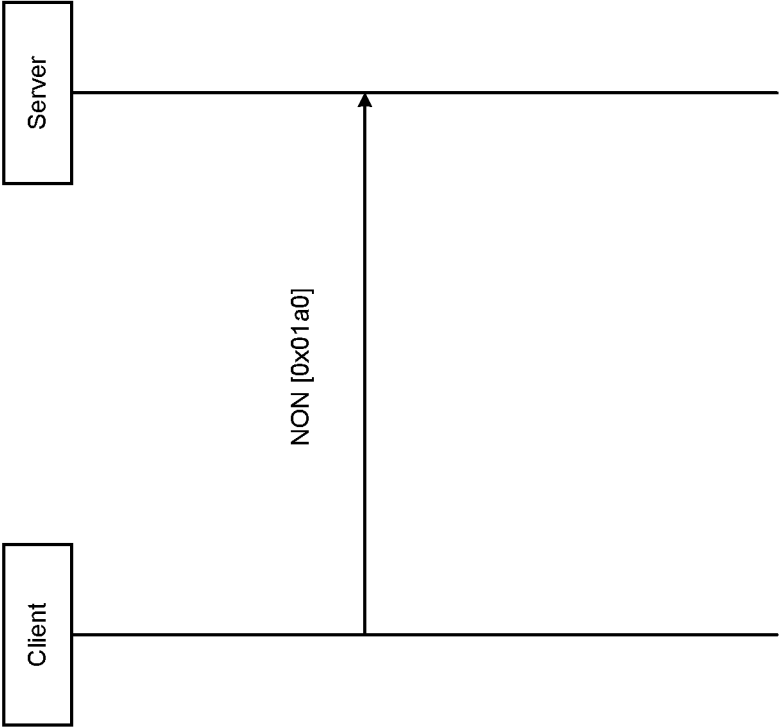


FIG. 4

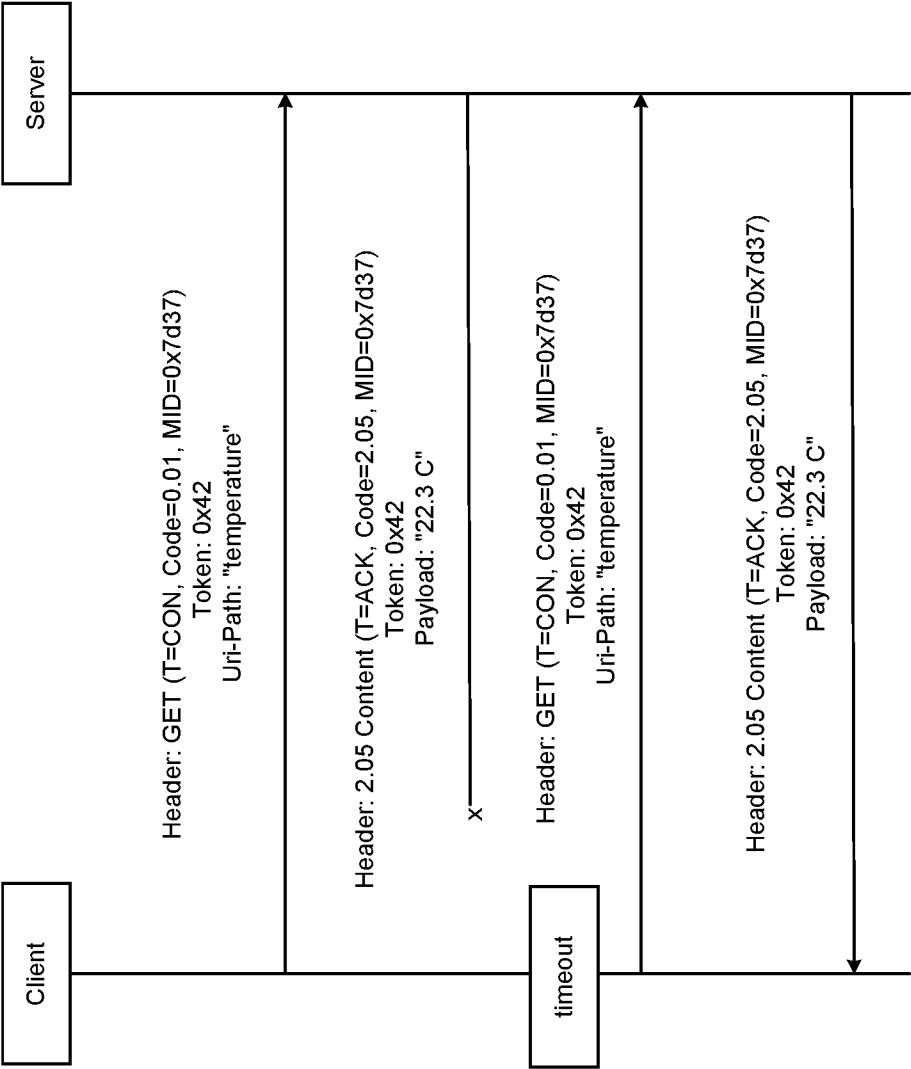


FIG. 5

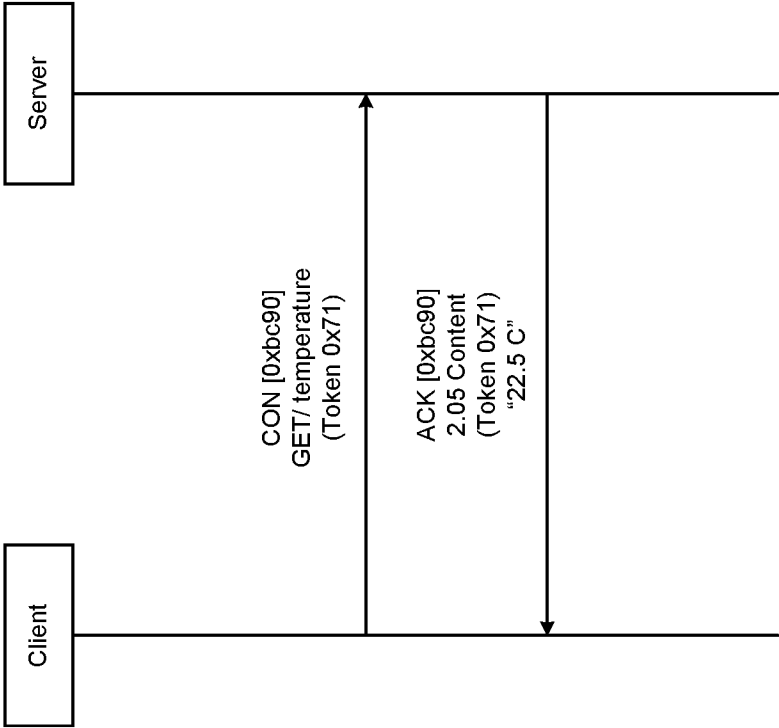


FIG. 6

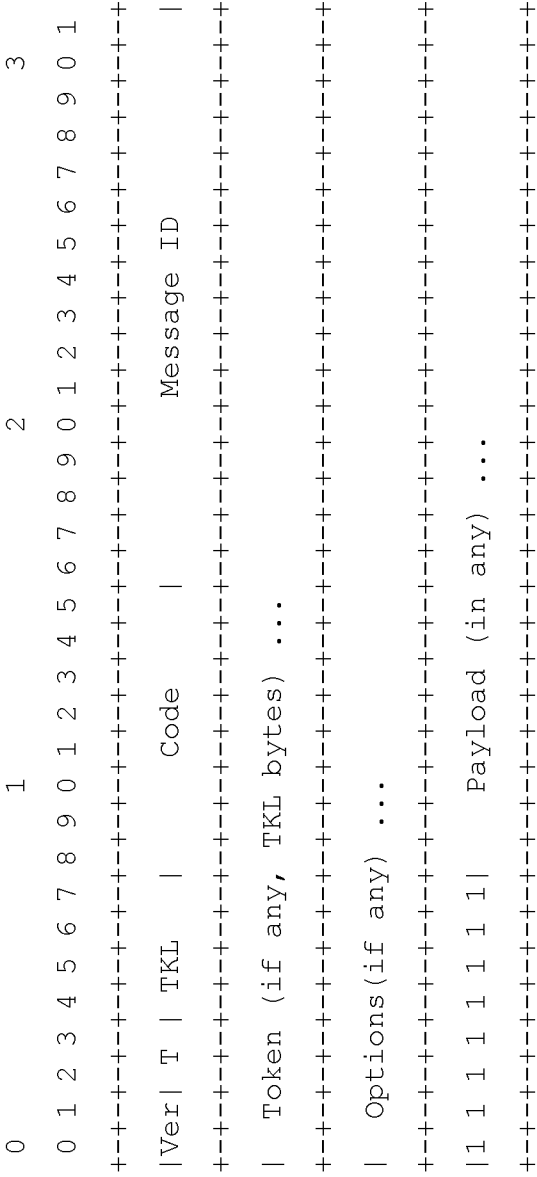


FIG. 7

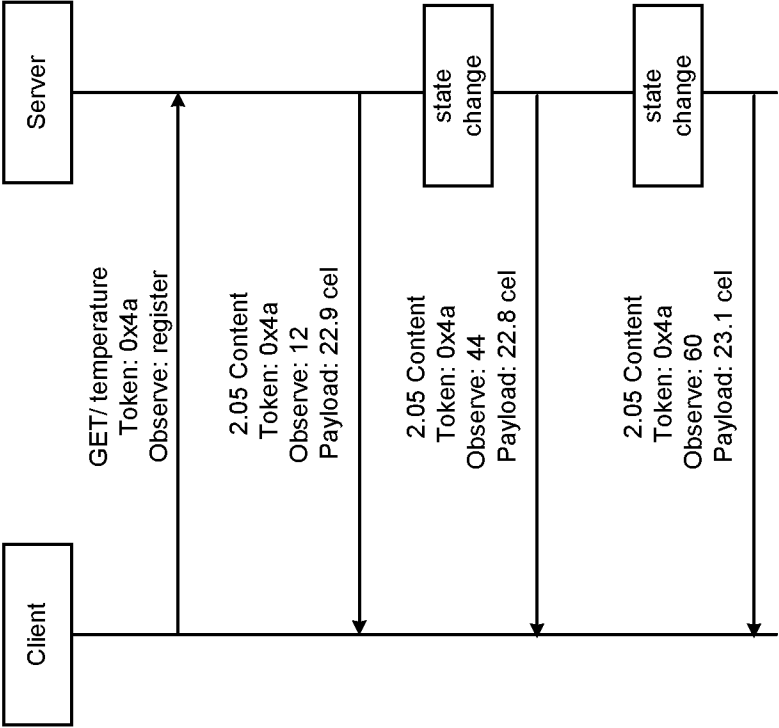


FIG. 8

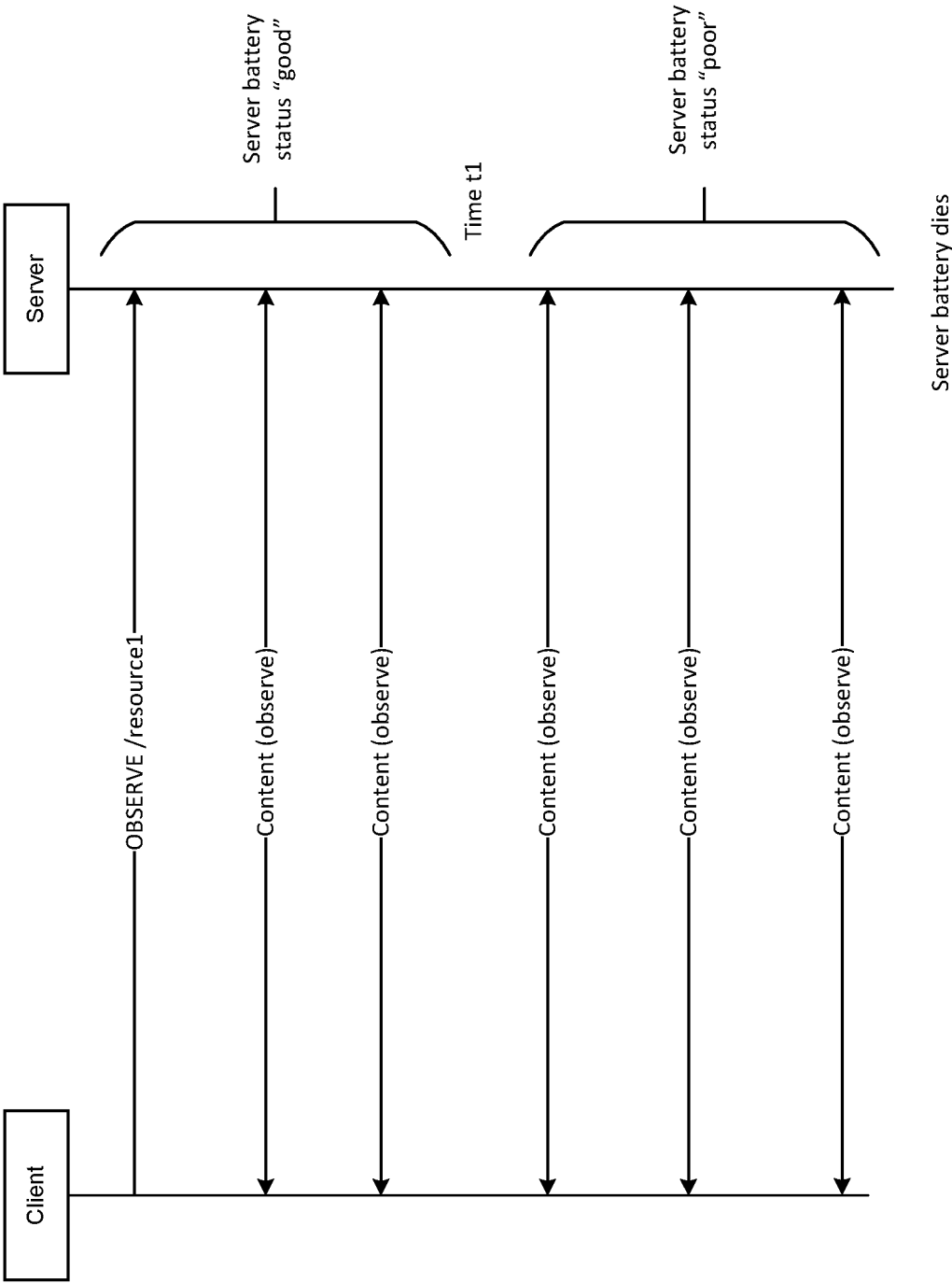


FIG. 9

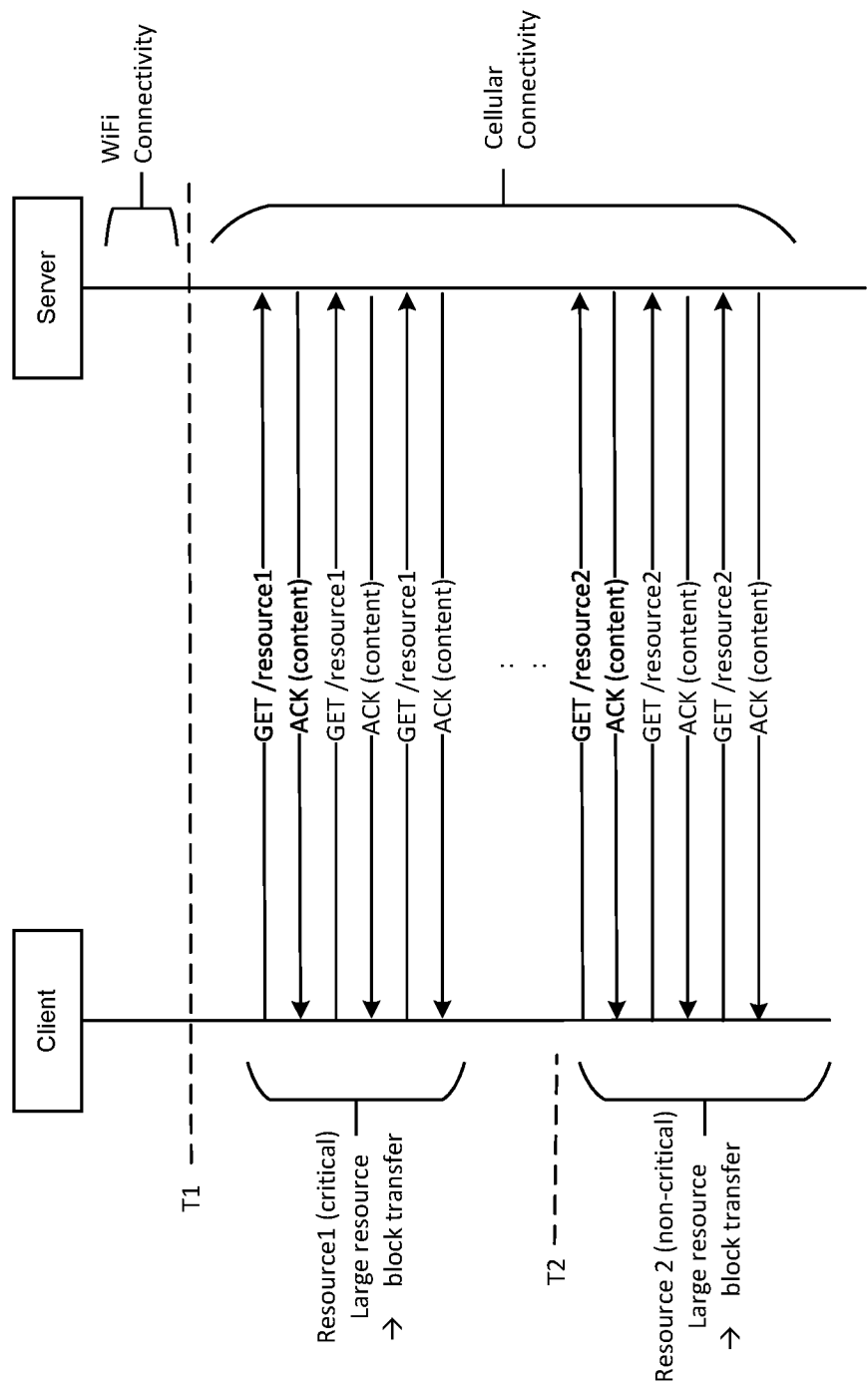


FIG. 10

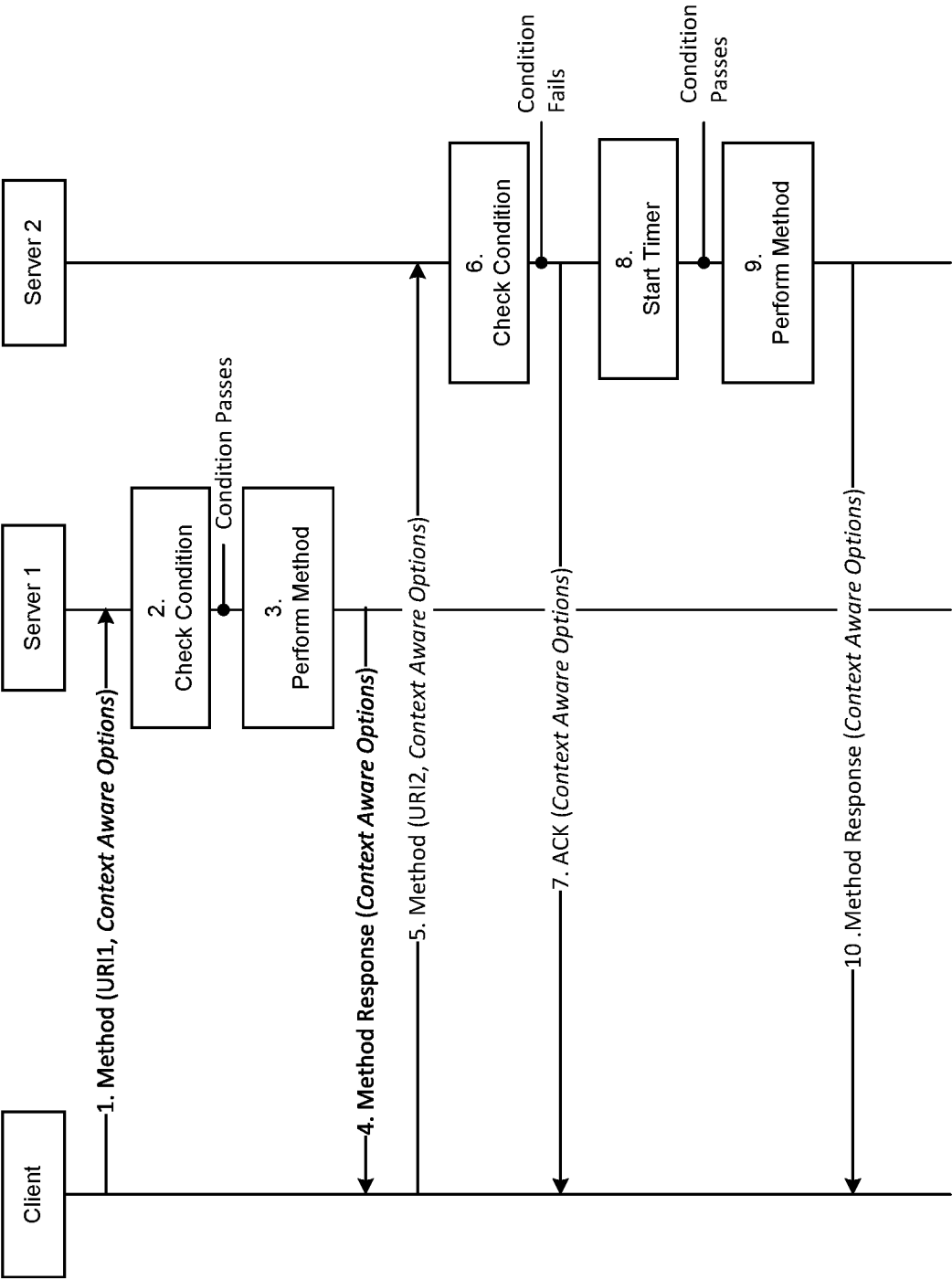


FIG. 11

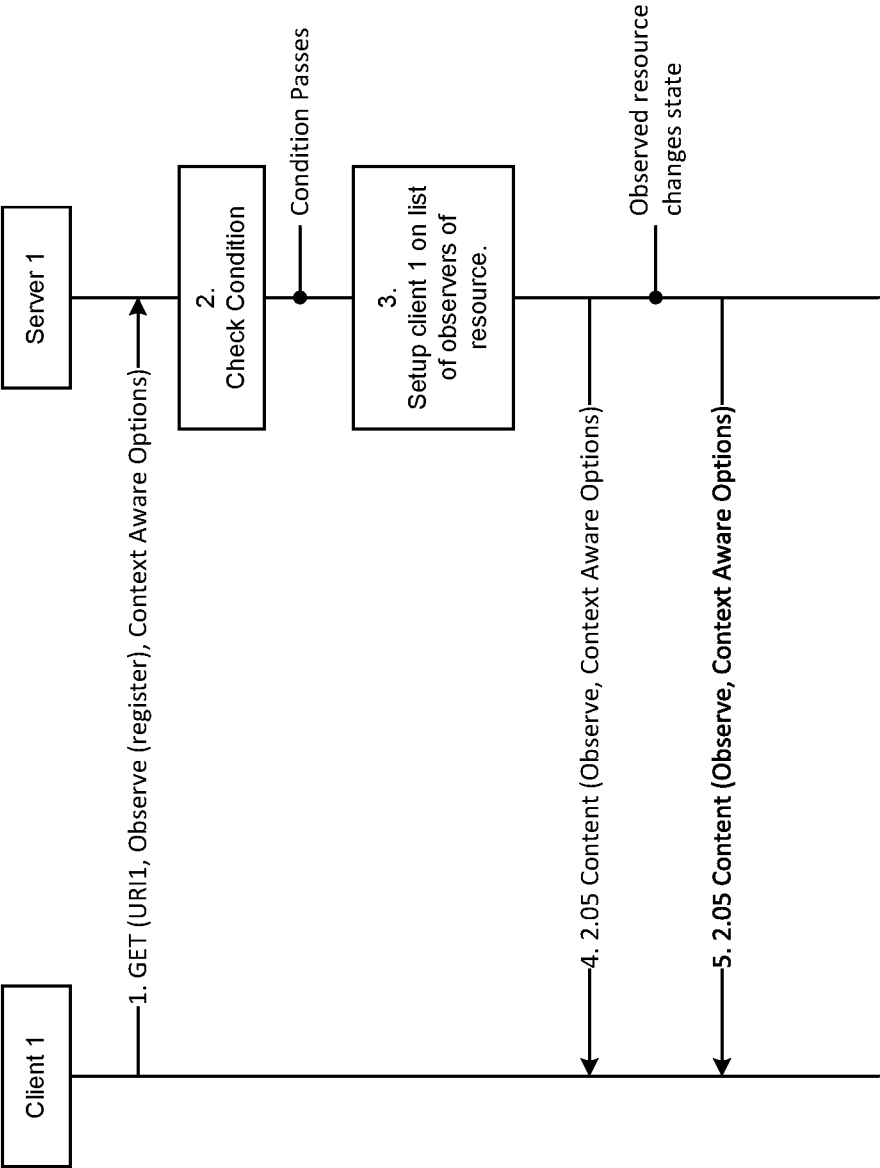


FIG. 12

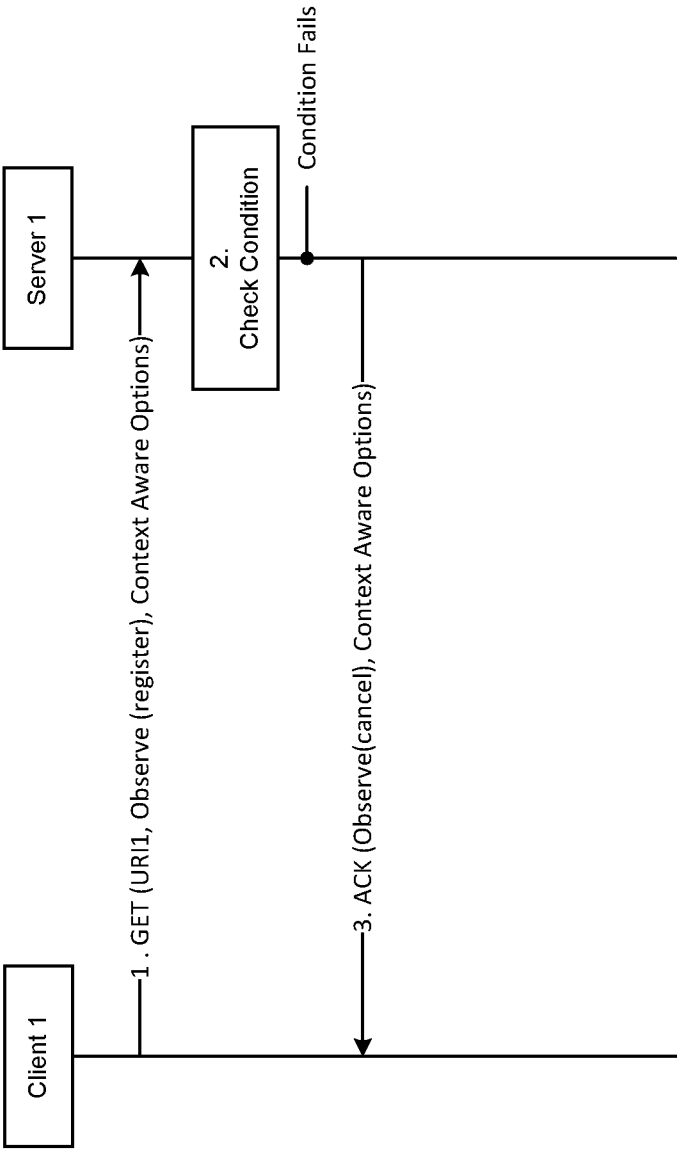


FIG. 13

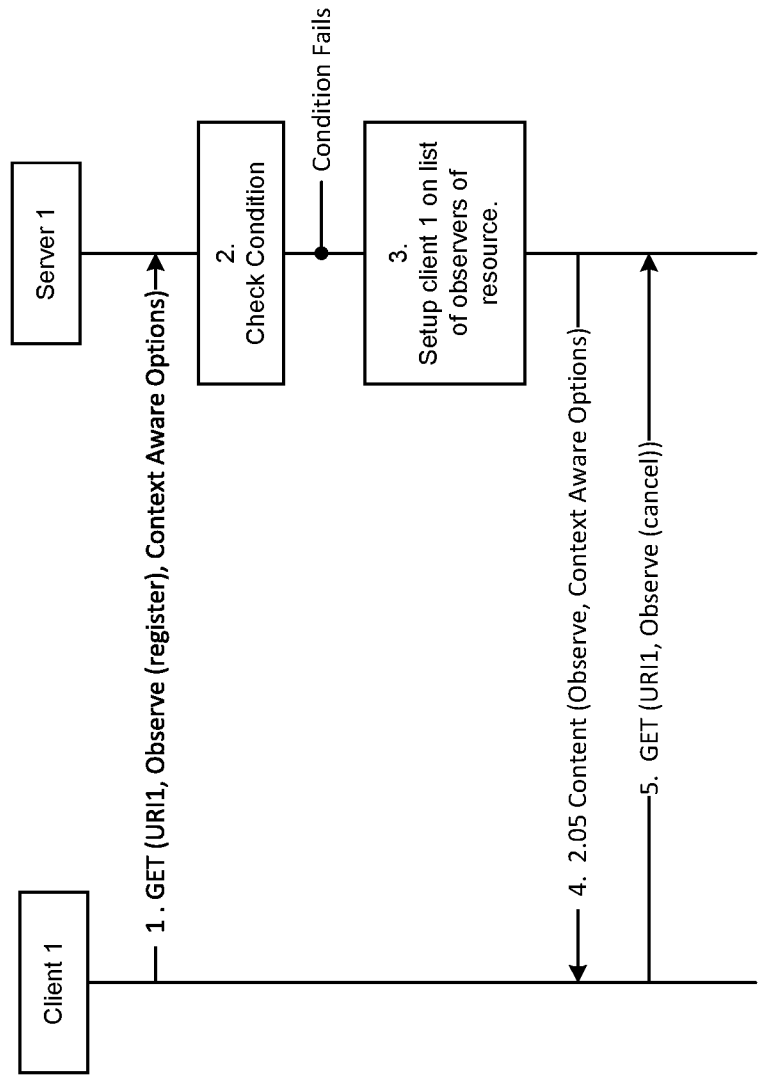


FIG. 14

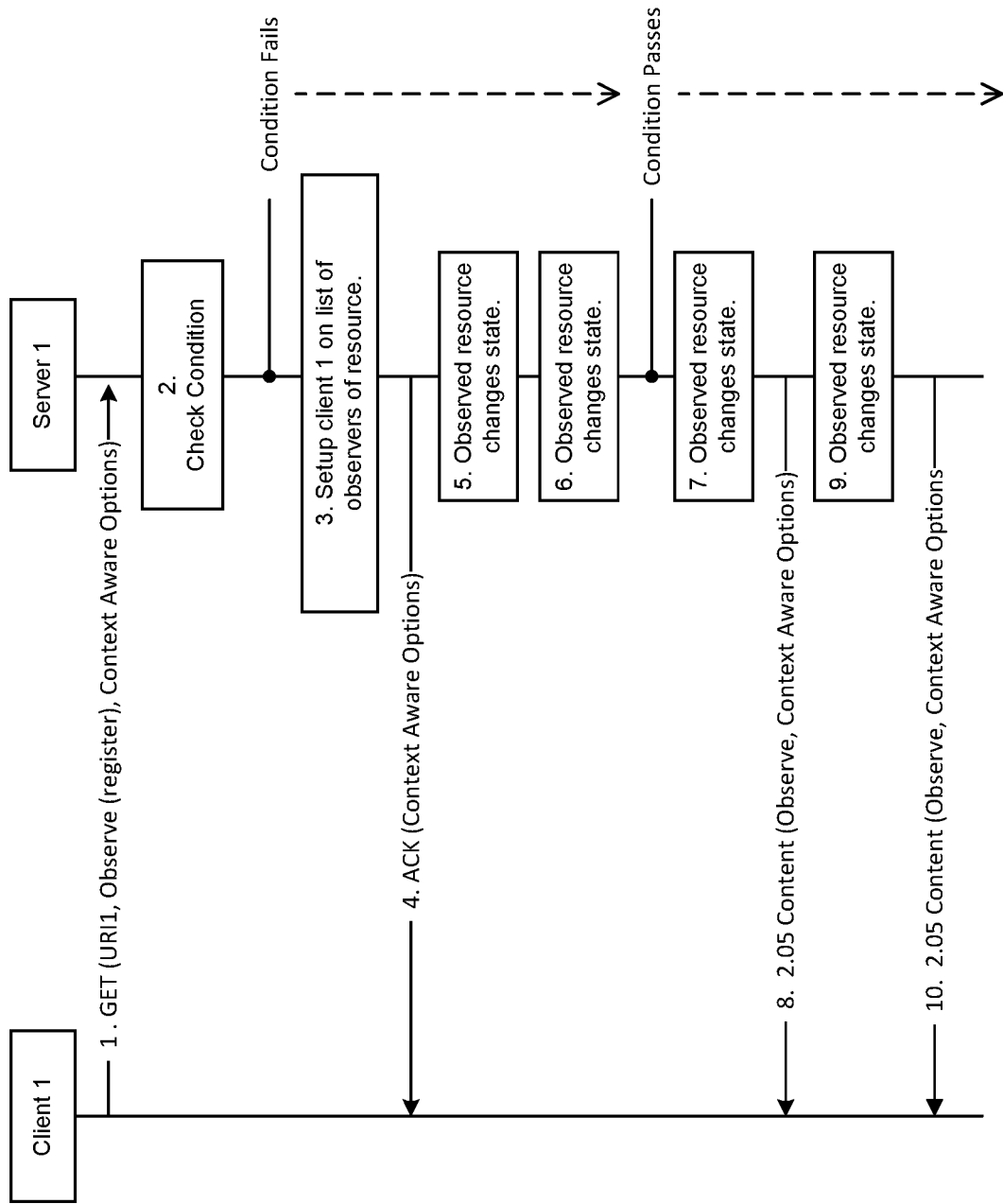
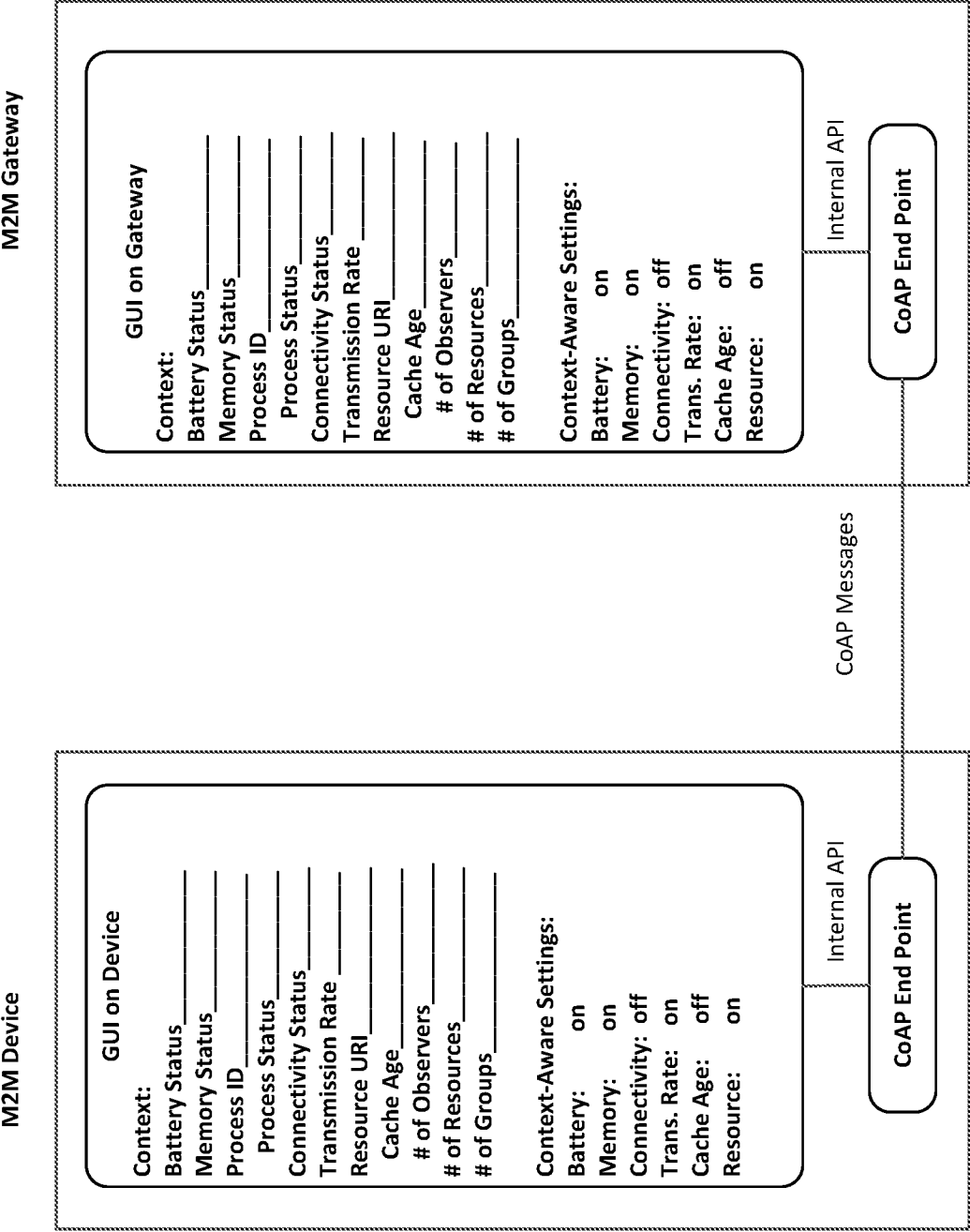


FIG. 15



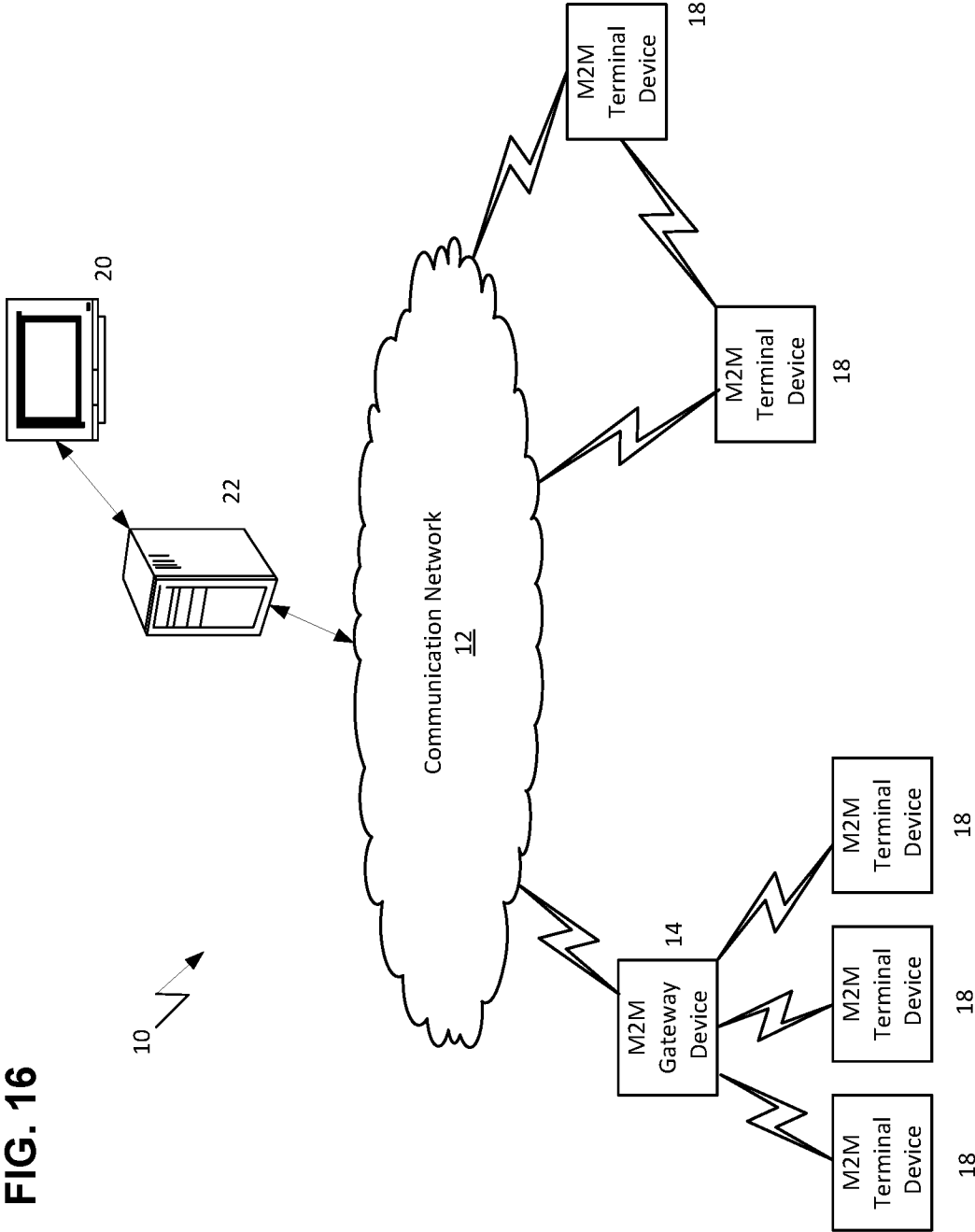


FIG. 16

FIG. 17

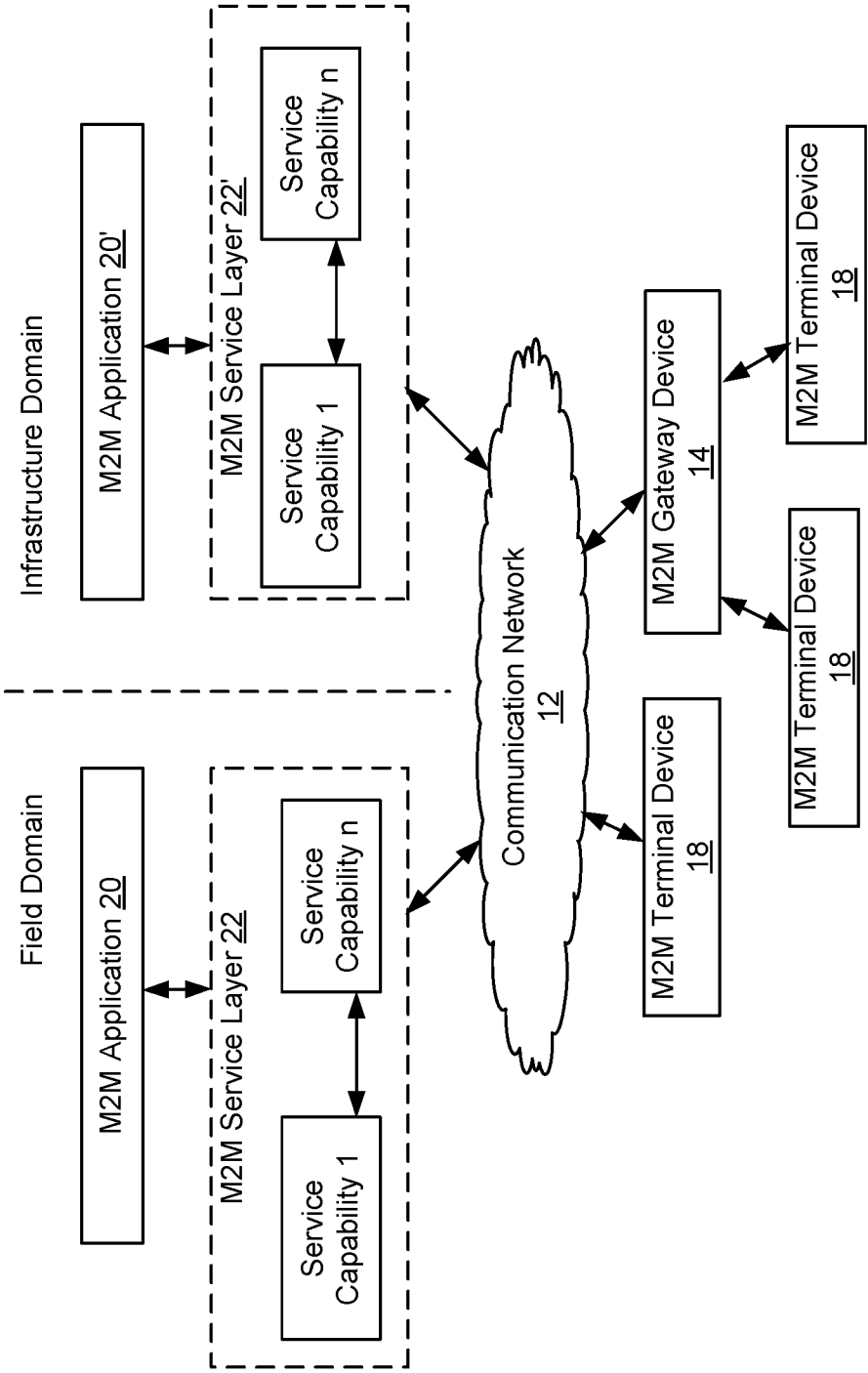


FIG. 18

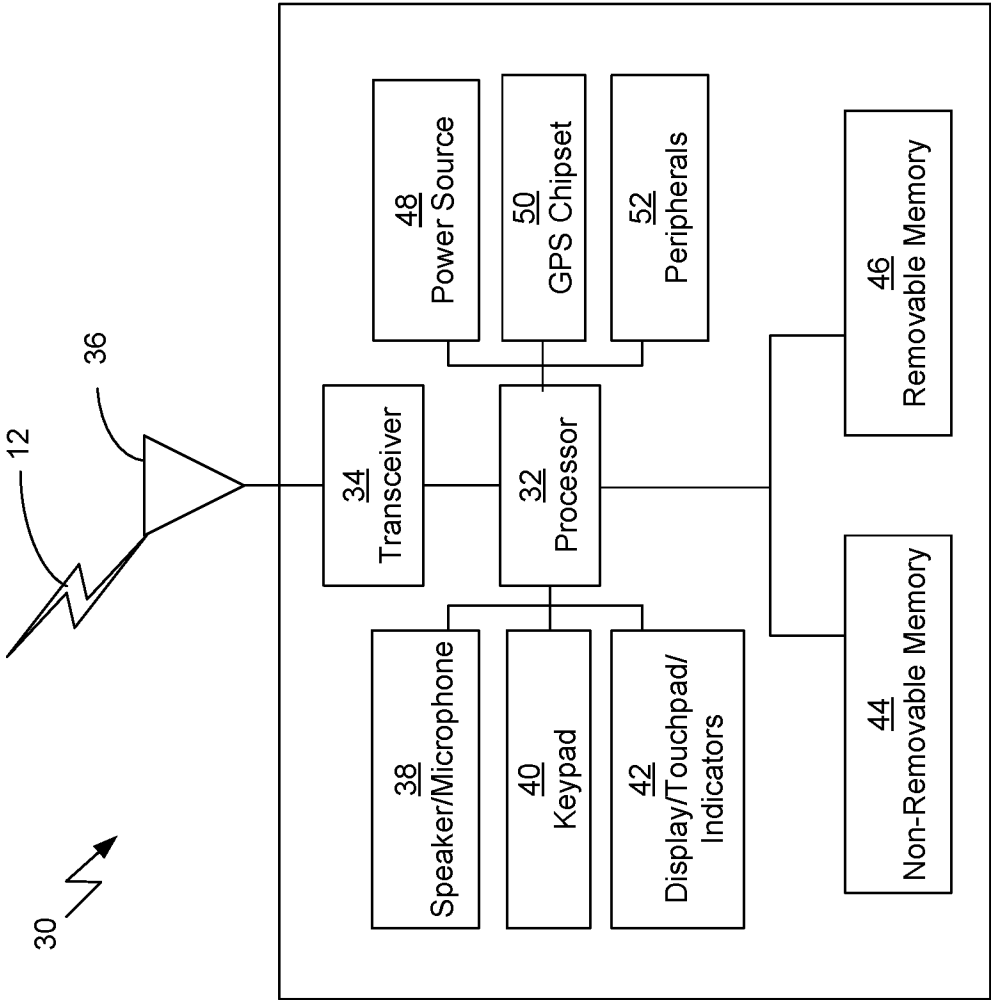
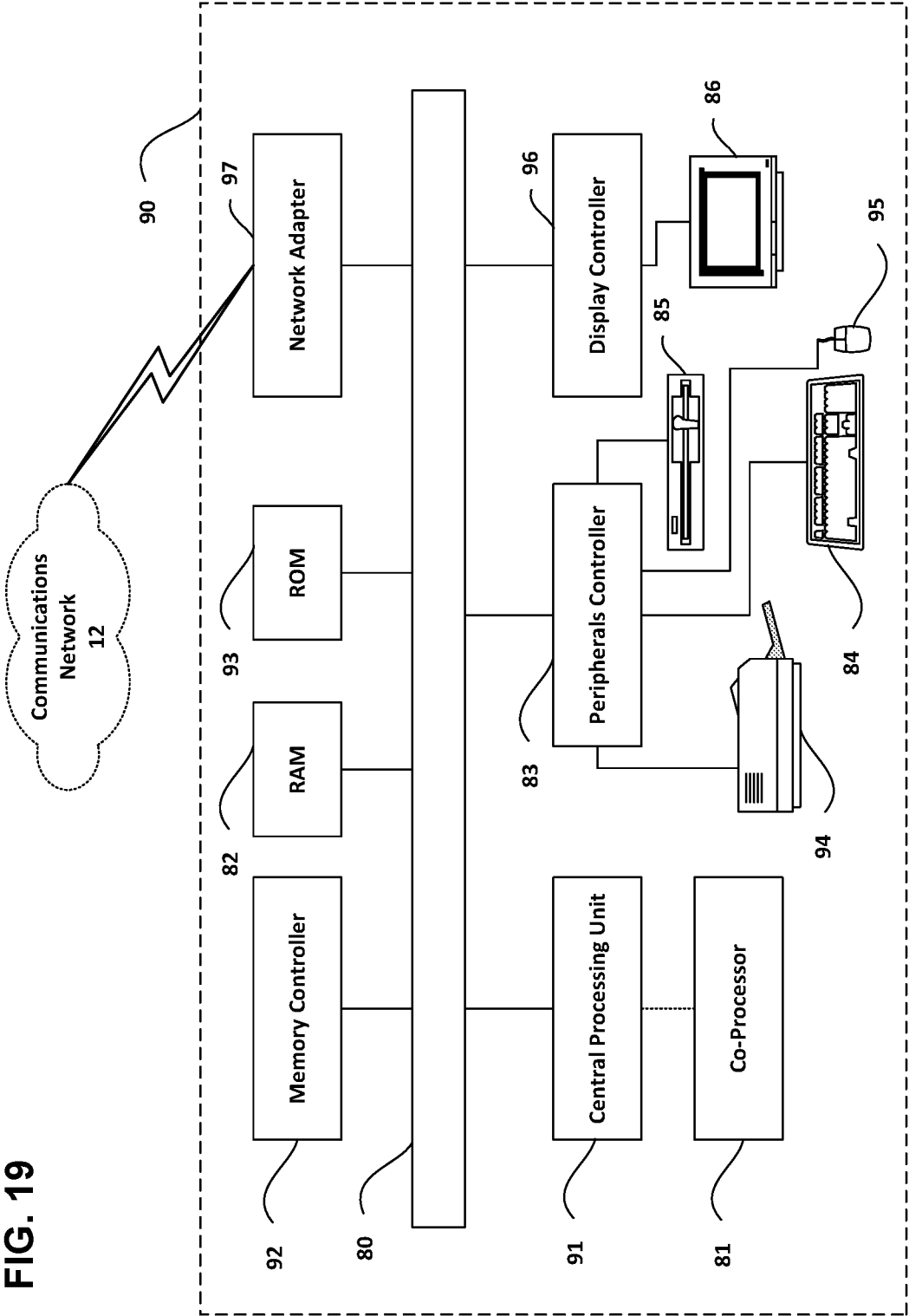


FIG. 19



INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/050982

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04W4/00 H04L29/08
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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HARTKE UNIVERSITAET BREMEN TZI K: "Observing Resources in CoAP; draft-ietf-core-observe-16.txt", OBSERVING RESOURCES IN COAP; DRAFT-IETF-CORE-OBSERVE-16.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARDWORKINGDRAFT, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 30 December 2014 (2014-12-30), pages 1-34, XP015103761, [retrieved on 2014-12-30]	1,2,4,7, 8,10-12, 14,17, 18,20
Y	page 4, lines 23-27 page 6, line 36 - page 7, line 1 page 7, lines 6-10 ----- -/--	3,5,6,9, 13,15, 16,19



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

5 December 2016

Date of mailing of the international search report

13/12/2016

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/US2016/050982

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	LI K LI HUAWEI TECHNOLOGIES J HOEBEKE F VAN DEN ABEELE IMINDS-IBCN/UGENT A JARA UNIVERSITY OF MURCIA S: "Conditional observe in CoAP; draft-li-core-conditional-observe-05.txt", CONDITIONAL OBSERVE IN COAP; DRAFT-LI-CORE-CONDITIONAL-OBSERVE-05.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARDWORKINGDRAFT, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 15 October 2014 (2014-10-15), pages 1-14, XP015102182, [retrieved on 2014-10-15] page 6, line 17 - page 7, line 2 page 4, lines 37-39 -----	3,5,6,9, 13,15, 16,19