CoRE Working Group
Internet-Draft
Obsoletes: 7390 (if approved)
Updates: 7252, 7641 (if approved)
Intended status: Standards Track
Expires: October 1, 2020

E. Dijk
IoTconsultancy.nl
C. Wang
InterDigital
M. Tiloca
RISE AB

March 30, 2020

# Group Communication for the Constrained Application Protocol (CoAP) draft-ietf-core-groupcomm-bis-00

## Abstract

This document specifies the use of the Constrained Application Protocol (CoAP) for group communication, using UDP/IP multicast as the underlying data transport. Both unsecured and secured CoAP group communication are specified. Security is achieved by use of the Group Object Security for Constrained RESTful Environments (Group OSCORE) protocol. The target application area of this specification is any group communication use cases that involve resource-constrained networks. The most common of such use cases are also discussed. This document replaces [RFC7390] and updates [RFC7252] and [RFC7641].

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 1, 2020.

## Copyright Notice

Table of Contents

## 1. Introduction

This document specifies group communication using the Constrained Application Protocol (CoAP) [RFC7252] together with UDP/IP multicast. CoAP is a RESTful communication protocol that is used in resource-constrained nodes, and in resource-constrained networks where packet sizes should be small. This area of use is summarized as Constrained RESTful Environments (CoRE).

One-to-many group communication can be achieved in CoAP, by a client using UDP/IP multicast data transport to send multicast CoAP request messages. In response, each server in the addressed group sends a response message back to the client over UDP/IP unicast. Notable CoAP implementations supporting group communication include the framework "Eclipse Californium" 2.0.x [Californium] from the Eclipse Foundation and the "Implementation of CoAP Server & Client in Go" [Go-OCF] from the Open Connectivity Foundation (OCF).

Both unsecured and secured CoAP group communication over UDP/IP multicast are specified in this document. Security is achieved by using Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm], which in turn builds on Object Security for Constrained Restful Environments (OSCORE) [RFC8613]. This method provides end-to-end application-layer

security protection of CoAP messages, by using CBOR Object Signing and Encryption (COSE) [RFC7049][RFC8152].

All guidelines in [RFC7390] are updated by this document, which replaces and obsoletes [RFC7390]. Furthermore, this document updates [RFC7252], by adding security for CoAP group communication and updates [RFC7641], by adding the multicast usage of CoAP Observe.

All sections in the body of this document are normative, while appendices are informative. For additional background about use cases for CoAP group communication in resource-constrained devices and networks, see Appendix A.

## 1.1. Scope

For group communication, only solutions that use CoAP over UDP/IP multicast are in the scope of this document. There are alternative methods to achieve group communication using CoAP, for example Publish-Subscribe [I-D.ietf-core-coap-pubsub] which uses a central broker server that CoAP clients access via unicast communication. These methods may be usable for the same or similar use cases as are targeted in this document.

Furthermore, this document defines Group OSCORE [I-D.ietf-core-oscore-groupcomm] as the default group communication security solution for CoAP. Security solutions for group communication and configuration other than Group OSCORE are not in scope. General principles for secure group configuration are in scope.

## 1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification requires readers to be familiar with CoAP terminology [RFC7252]. Terminology related to group communication is defined in Section 2.1.

Furthermore, "Security material" refers to any security keys, counters or parameters required to participate in secure group communication with other devices that share the same security material.

## 2. General Group Communication Operation

The general operation of group communication, applicable for both unsecured and secured operation, is specified in this section by going through the stack from top to bottom. First, different group types are defined in Section 2.1. Group configuration (e.g. group creation and maintenance which are usually done by an application, user or commissioning entity) is considered next in Section 2.2. Then the use of CoAP for group communication including support for protocol extensions (block-wise transfer, Observe) follows in Section 2.3. How CoAP group messages are carried over various transport layers is the subject of Section 2.4. Finally, Section 2.5 covers the interworking of CoAP group communication with other protocols that may operate in the same network.

### 2.1. Group Definition

Three types of groups and their mutual relations are defined in this section: CoAP group, application group, and security group.

A CoAP group is defined as a set of CoAP endpoints, where each endpoint is configured to receive CoAP multicast messages that are sent to the group's associated IP multicast address and UDP port. An endpoint may be a member of multiple CoAP groups by subscribing to multiple IP multicast groups. Group membership(s) of an endpoint may dynamically change over time. A device sending a CoAP multicast message to a group is not necessarily itself a member of this group: it is a member only if it also has a CoAP endpoint listening to the group's associated IP multicast address and UDP port. A CoAP group can be encoded within a Group URI, i.e. a CoAP URI that has the "coap" scheme and includes in the authority part either an IP multicast address or a group hostname (e.g., a Group Fully Qualified Domain Name (FQDN)) that can be resolved to an IP multicast address. A Group URI also contains an optional UDP port number in the authority part. Group URIs follow the regular CoAP URI syntax (see Section 6 of [RFC7252]).

Besides CoAP groups, that have relevance at the level of IP networks and CoAP endpoints, there are also application groups. An application group is a set of CoAP endpoints that share a common set of CoAP resources. An endpoint may be a member of multiple application groups. An application group has relevance at the application level - for example an application group could denote all lights in an office room or all sensors in a hallway. There can be a one-to-one or a one-to-many relation between a CoAP group and application group(s). An application group is optionally identified explicitly in the path component or query component of a Group URI.

If not explicitly identified, the application group is specified implicitly in a Group URI by choice of CoAP group and resource path.

For secure group communication, a security group is required. A security group is a group of endpoints that share the same security material, such that they can mutually exchange secured messages and verify secured messages. An endpoint may be a member of multiple security groups. There can be a one-to-one or a one-to-many relation between security groups and CoAP groups. Also, there can be a one-to-one or a one-to-many relation between security groups and application groups. Any two application groups associated to the same security group do not share any resource. A special security group named "NoSec" identifies group communication without any security at the transport layer and/or application layer.

Using the above group type definitions, a CoAP group communication message sent by an endpoint can be represented as a tuple that contains one instance of each group type:

(application group, CoAP group, security group)

Figure 1 summarizes the relations between the different types of groups described above in UML class diagram notation. The items in square brackets are optionally defined.

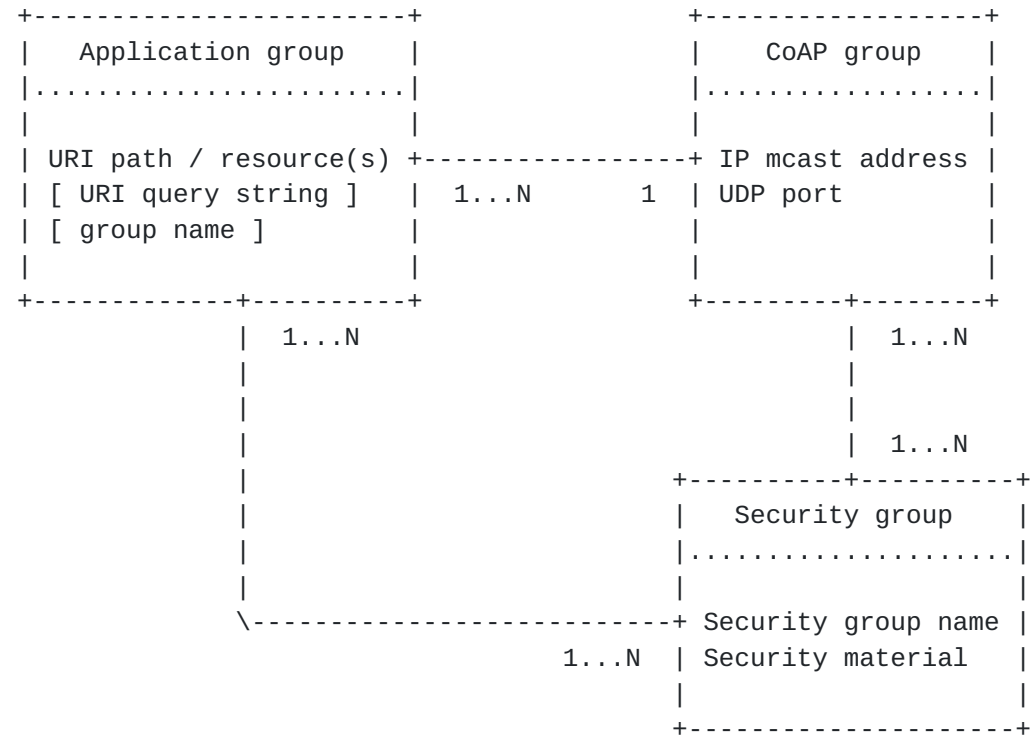


Figure 1: Relation Among Different Group Types

Figure 2 provides a deployment example of the relations between the different types of groups. It shows six CoAP servers (Srv1-Srv6) and their respective resources hosted (/resX). There are three application groups (1, 2, 3) and two security groups (1, 2). Security Group 1 is used by both Application Group 1 and 2. Three clients (Cli1, Cli2, Cli3) are configured with security material for Security Group 1. One cient (Cli4) is configured with security material for Security Group 2. All the shown application groups use the same CoAP group (not shown in the figure), i.e. one specific multicast IP address and UDP port on which all the shown resources are hosted for each server.

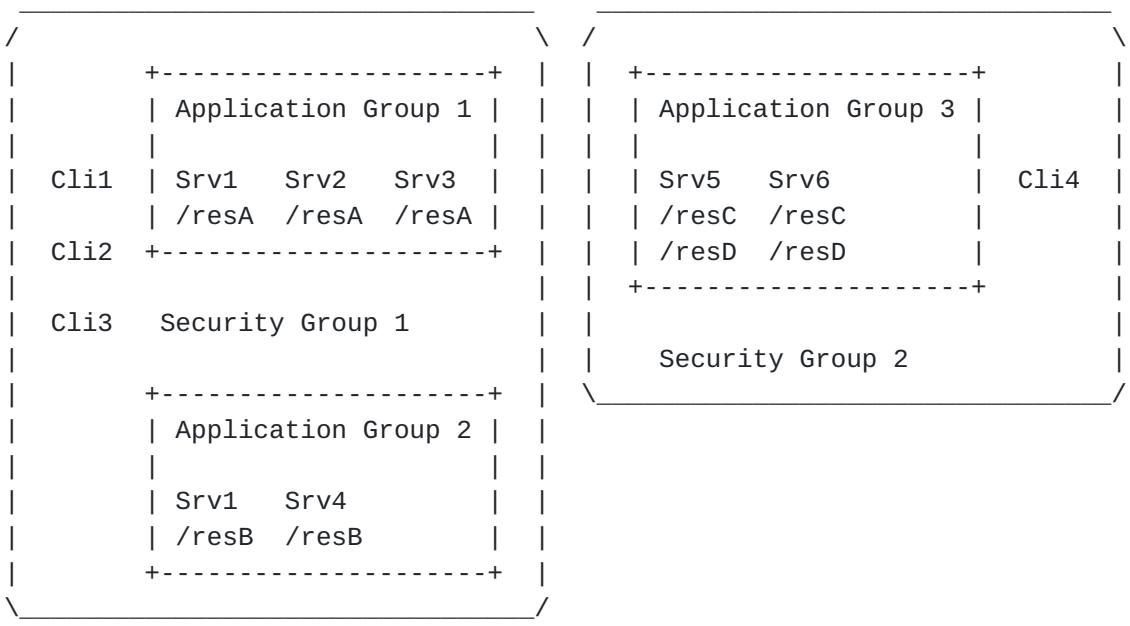


Figure 2: Deployment Example of Different Group Types

## 2.2. Group Configuration

### 2.2.1. Group Naming

A CoAP group is identified and named by the authority component in the Group URI, which includes host and optional port number. It is recommended to configure an endpoint by default with an IP multicast address literal, instead of a hostname. This is because DNS infrastructure may not be deployed in many constrained networks. In case a group hostname is configured, it can be uniquely mapped to an IP multicast address via DNS resolution - if DNS client functionality is available in the clients and the DNS service is supported in the network. Some examples of hierarchical CoAP group FQDN naming (and scoping) for a building control application are shown in Section 2.2 of [RFC7390].

An application group can be named in many ways through different types of identifiers, such as numbers, URIs or other strings. An application group name or identifier, if explicitly encoded, is typically included in the path component or query component of a Group URI. Appendix A of [I-D.ietf-core-resource-directory] shows registration of application groups into a Resource Directory, along with the CoAP group it maps to.

A security group is identified by a stable and invariant string used as group name, which is generally not related with other kind of group identifiers, specific to the chosen security solution. The "NoSec" security group is typically identified by the absence of any name or identifier, and of any security-related data structures in the CoAP message.

### 2.2.2. Group Creation and Membership

To create a CoAP group, a configuring entity defines an IP multicast address (or hostname) for the group and optionally a UDP port number in case it differs from the default CoAP port 5683. Then, it configures one or more devices as listeners to that IP multicast address, with a CoAP endpoint listening on the group's associated UDP port. These endpoints/devices are the group members. The configuring entity can be, for example, a local application with pre-configuration, a user, a software developer, a cloud service, or a local commissioning tool. Also, the devices sending CoAP requests to the group in the role of CoAP client need to be configured with the same information, even though they are not necessarily group members. One way to configure a client is to supply it with a CoAP Group URI. The IETF does not define a mandatory, standardized protocol to accomplish CoAP group creation. [RFC7390] defines an experimental protocol for configuration of group membership for unsecured group communication, based on JSON-formatted configuration resources.

To create an application group, a configuring entity may configure a resource (name) or set of resources on a CoAP endpoint, such that a request sent by a configured CoAP client with a configured URI path will be processed by one or more CoAP servers that have the same URI path configured - i.e. the application group members.

To create a security group, selected CoAP endpoints are configured with the same security material in case communication is secured within the group. The part of the process that involves secure distribution of group keys MAY use standardized communication with a Group Manager as defined in Section 4. For unsecure group communication using the "NoSec" security group, any CoAP endpoint may become a group member at any time: there is no (central) configuring entity that needs to provide the security material for this group.

This means that group creation and membership cannot be tightly controlled for the "NoSec" group.

The configuration of groups and membership may be performed at different moments in the life-cycle of a device; for example during product (software) creation, in the factory, at a reseller, on-site during first deployment, or on-site during a system reconfiguration operation.

### 2.2.3. Group Discovery

It is possible for CoAP endpoints to discover application groups as well as CoAP groups, by using the RD-Groups usage pattern of the CoRE Resource Directory (RD), as defined in Appendix A of [I-D.ietf-core-resource-directory].

In particular, an application group can be registered to the RD, specifying the reference IP multicast address, hence its associated CoAP group. The registration is typically performed by a Commissioning Tool. Later on, CoAP endpoints can discover the registered application groups and related CoAP group, by using the lookup interface of the RD.

When secure communication is provided with Group OSCORE (see Section 4), the approach described in [I-D.tiloca-core-oscore-discovery] and also based on the RD can be used, in order to discover the security group to join.

In particular, the responsible OSCORE Group Manager registers its own security groups to the RD, as links to its own corresponding resources for joining the security groups [I-D.ietf-ace-key-groupcomm-oscore]. Later on, CoAP endpoints can discover the registered security groups and related application groups, by using the lookup interface of the RD, and then join the security group through the respective Group Manager.

### 2.2.4. Group Maintenance

Maintenance of a group includes any necessary operations to cope with changes in a system, such as: adding group members, removing group members, changing group security material, reconfiguration of UDP port and/or IP multicast address, reconfiguration of the Group URI, renaming of application groups, splitting of groups, or merging of groups.

For unsecured group communication (see Section 3), addition/removal of CoAP group members is simply done by configuring these devices to start/stop listening to the group IP multicast address, and to start/

stop the CoAP server listening to the group IP multicast address and UDP port.

For secured group communication (see Section 4), the protocol Group OSCORE [I-D.ietf-core-oscore-groupcomm] is mandatory to implement. When using Group OSCORE, CoAP endpoints participating in group communication are also members of a corresponding OSCORE security group, and thus share a common set of cryptographic material. Additional related maintenance operations are discussed in Section 4.1.

## 2.3. CoAP Usage

### 2.3.1. Request/Response Model

A CoAP client is an endpoint able to transmit CoAP requests and receive CoAP responses. Since the underlying UDP transport supports multiplexing by means of UDP port number, there can be multiple independent CoAP clients operational on a single host. On each UDP port, an independent CoAP client can be hosted. Each independent CoAP client sends requests that use the associated endpoint's UDP port number as the UDP source port of the request.

All CoAP requests that are sent via IP multicast MUST be Non-confirmable (Section 8.1 of [RFC7252]). The Message ID in an IP multicast CoAP message is used for optional message deduplication by both clients and servers, as detailed in Section 4.5 of [RFC7252].

A server sends back a unicast response to the CoAP group request - but the server MAY suppress the response if the server chooses so and if permitted by the rules in this document. The unicast responses received by the CoAP client may be a mixture of success (e.g., 2.05 Content) and failure (e.g., 4.04 Not Found) codes, depending on the individual server processing results.

The CoAP No-Response Option [RFC7967] can be used by a client to influence the default response suppression on the server side. It is RECOMMENDED for a server to implement this option only on selected resources where it is useful in the application context. If the Option is supported on a resource, it MUST override the default response suppression of that resource.

Any default response suppression by a server SHOULD be performed in a consistent way, such that if a request on a resource produces a Response Code and this response is not suppressed, then a later request on the same resource that produces a response with the same Response Code is also not suppressed.

A CoAP client MAY repeat a multicast request using the same Token value and same Message ID value, in order to ensure that enough (or all) group members have been reached with the request. This is useful in case a number of group members did not respond to the initial request and the client suspects that the request did not reach these group members. However, in case one or more servers did receive the initial request but the response to that request was lost, this repeat does not help to retrieve the lost response(s) if the server(s) implement the optional Message ID based deduplication (Section 4.5 of [RFC7252]).

A CoAP client MAY also repeat a multicast request using the same Token value and a different Message ID, in which case all servers that received the initial request will again process the repeated request since it appears within a new CoAP message. This is useful in case a client suspects that one or more response(s) to its original request were lost and the client needs to collect more, or even all, responses from group members, even if this comes at the cost of the overhead of certain group members responding twice (once to the original request, and once to the repeated request with different Message ID).

The CoAP client can distinguish the origin of multiple server responses by the source IP address of the UDP message containing the CoAP response and/or any other available application-specific source identifiers contained in the CoAP response, such as an application-level unique ID associated to the server. If secure communication is provided with Group OSCORE (see Section 4), additional security-related identifiers enable the client to retrieve the right security material for decrypting each response and authenticating its source.

While processing a response, the source endpoint of the response is not exactly matched to the destination endpoint of the request, since for a multicast request these will never match. This is specified in Section 8.2 of [RFC7252]. In case a single client has sent multiple group requests and concurrent CoAP transactions are ongoing, the responses received by that client are matched to a request using the Token value. Due to UDP level multiplexing, the UDP destination port of the response MUST match to the client endpoint's UDP port value, i.e. to the UDP source port of the client's request.

For multicast CoAP requests, there are additional constraints on the reuse of Token values at the client, compared to the unicast case. In the unicast case, receiving a response usually frees up its Token value, since no more responses to the same request will follow. Therefore, such value would become available for reuse. Note that [I-D.ietf-core-echo-request-tag] updates the Token processing of [RFC7252], so that clients do not use Tokens in a way that risk

associating responses with a wrong request. This holds especially when using a security protocol that does not provide bindings between requests and responses, e.g. DTLS [RFC6347][I-D.ietf-tls-dtls13] and TLS [RFC5246][RFC8446]. In such a case, a client should not reuse a (freed up) Token value within a secure connection, until this has been rekeyed.

However, for multicast CoAP, the number of responses is not bound a priori. Therefore, the client cannot use the reception of a response as a trigger to "free up" a Token value for reuse. Moreover, reusing a Token value too early could lead to incorrect response/request matching on the client, and would be a protocol error. Therefore, the time between reuse of Token values used in multicast requests MUST be greater than:

```
MIN_TOKEN_REUSE_TIME = (NON_LIFETIME + MAX_LATENCY +
                        MAX_SERVER_RESPONSE_DELAY)
```

where NON_LIFETIME and MAX_LATENCY are defined in Section 4.8 of [RFC7252]. This specification defines MAX_SERVER_RESPONSE_DELAY as in [RFC7390], that is: the expected maximum response delay over all servers that the client can send a multicast request to. This delay includes the maximum Leisure time period as defined in Section 8.2 of [RFC7252]. However, CoAP does not define a time limit for the server response delay. Using the default CoAP parameters, the Token reuse time MUST be greater than 250 seconds plus MAX_SERVER_RESPONSE_DELAY. A preferred solution to meet this requirement is to generate a new unique Token for every new multicast request, such that a Token value is never reused. If a client has to reuse Token values for some reason, and also MAX_SERVER_RESPONSE_DELAY is unknown, then using MAX_SERVER_RESPONSE_DELAY = 250 seconds is a reasonable guideline. The time between Token reuses is in that case set to a value greater than 500 seconds.

When securing Group CoAP communications with Group OSCORE [I-D.ietf-core-oscore-groupcomm], secure binding between requests and responses is ensured (see Section 4). Thus, a client may reuse a Token value after it has been freed up, as discussed above for the multicast case and considering a reuse time greater than MIN_TOKEN_REUSE_TIME. If an alternative security protocol for Group CoAP is defined in the future and it does not ensure secure binding between requests and responses, a client MUST follow the Token processing requirements for the unicast case discussed above, as defined in [I-D.ietf-core-echo-request-tag].

Another method to more easily meet the above constraint is to instantiate multiple CoAP clients at multiple UDP ports on the same host. The Token values only have to be unique within the context of

a single CoAP client, so using multiple clients can make it easier to meet the constraint.

Since a client sending a multicast request with a Token T will accept multiple responses with the same Token T, there is a risk that the same server sends multiple responses with the same Token T back to the client. For example, this server might not implement the optional CoAP message deduplication based on Message ID, or it might be a malicious/compromised server acting out of specification. To mitigate issues with multiple responses from one server bound to a same multicast request, the client has to ensure that, as long as the the CoAP Token used for a multicast request is retained, at most one response to that request per server is accepted, with the exception of Observe notifications [RFC7641] (see Section 2.3.5).

To this end, upon receiving a response corresponding to a multicast request, the client MUST perform the following actions. First, the client checks whether it previously received a valid response to this request from the same originating server of the just-received response. If the check yields a positive match and the response is not an Observe notification (i.e., it does not include an Observe option), the client SHALL stop processing the response. Upon eventually freeing up the Token value of a multicast request for possible reuse, the client MUST also delete the list of responding servers associated to that request.

### 2.3.2. Port and URI Path Selection

A server that is a member of a CoAP group listens for CoAP messages on the group's IP multicast address, usually on the CoAP default UDP port 5683, or another non-default UDP port if configured. Regardless of the method for selecting the port number, the same port number MUST be used across all CoAP servers that are members of a group and across all CoAP clients performing the requests to that group. The URI Path used in the request is preferably a path that is known to be supported across all group members. However there are valid use cases where a request is known to be successful for a subset of the CoAP group, for example only members of a specific application group, while those group members for which the request is unsuccessful (for example because they are outside the application group) either ignore the multicast request or respond with an error status code.

One way to create multiple CoAP groups is using different UDP ports with the same IP multicast address, in case the devices' network stack only supports a limited number of IP multicast group memberships. However, it must be taken into account that this incurs additional processing overhead on each CoAP server participating in at least one of these groups: messages to groups that are not of

interest to the node are only discarded at the higher transport (UDP) layer instead of directly at the network (IP) layer.

Port 5684 is reserved for DTLS-secured CoAP and MUST NOT be used for any CoAP group communication.

For a CoAP server node that supports resource discovery as defined in Section 2.4 of [RFC7252], the default port 5683 MUST be supported (see Section 7.1 of [RFC7252]) for the "All CoAP Nodes" multicast group as detailed in Section 2.4.

### 2.3.3. Proxy Operation

CoAP enables a client to request a forward-proxy to process a CoAP request on its behalf, as described in Section 5.7.2 and 8.2.2 of [RFC7252]. For this purpose, the client specifies either the request group URI as a string in the Proxy-URI option or it uses the Proxy-Scheme option with the group URI constructed from the usual Uri-* options. The forward-proxy then resolves the group URI to a destination CoAP group, multicasts the CoAP request, receives the responses and forwards all the individual (unicast) responses back to the client.

However, there are certain issues and limitations with this approach:

- The CoAP client component that sent a unicast CoAP request to the proxy may be expecting only one (unicast) response, as usual for a CoAP unicast request. Instead, it receives multiple (unicast) responses, potentially leading to fault conditions in the component or to discarding any received responses following the first one. This issue may occur even if the application calling the CoAP client component is aware that the forward-proxy is going to execute a CoAP group URI request.

- Each individual CoAP response received by the client will appear to originate (based on its IP source address) from the CoAP Proxy, and not from the server that produced the response. This makes it impossible for the client to identify the server that produced each response, unless the server identity is contained as a part of the response payload or inside a CoAP Option in the response.

A solution to the above issues is for the proxy to collect all the individual (unicast) responses to a CoAP group request and then send back only a single (aggregated) response to the client. However, this solution brings up new issues:

- The proxy does not know how many members there are in the group or how many group members will actually respond. Also, the proxy

does not know for how long to collect responses before sending back the aggregated response to the client. A CoAP client that is not using a Proxy might face the same problems in collecting responses to a multicast request. However, the client itself would typically have application-specific rules or knowledge on how to handle this situation, while an application-agnostic CoAP Proxy would typically not have this knowledge.

- There is no default format defined in CoAP for aggregation of multiple responses into a single response. Such a format could be standardized based on, for example, the multipart content-format [RFC8710].

Due to the above issues, it is RECOMMENDED that a CoAP Proxy only processes a group URI request if it is explicitly enabled to do so. The default response (if the function is not explicitly enabled) to a group URI request is 5.01 (Not Implemented). Furthermore, a proxy SHOULD be explicitly configured (e.g. by white-listing and/or client authentication) to allow proxied CoAP multicast requests only from specific client(s).

The operation of HTTP-to-CoAP proxies for multicast CoAP requests is specified in Section 8.4 and 10.1 of [RFC8075]. In this case, the "application/http" media type is used to let the proxy return multiple CoAP responses - each translated to a HTTP response - back to the HTTP client. Of course, in this case the HTTP client sending a group URI to the proxy needs to be aware that it is going to receive this format, and needs to be able to decode it into the responses of multiple CoAP servers. Also, the IP source address of each CoAP response cannot be determined anymore from the application/http response.

### 2.3.4. Congestion Control

CoAP group requests may result in a multitude of responses from different nodes, potentially causing congestion. Therefore, both the sending of IP multicast requests and the sending of the unicast CoAP responses to these multicast requests should be conservatively controlled.

CoAP [RFC7252] reduces IP multicast-specific congestion risks through the following measures:

- A server may choose not to respond to an IP multicast request if there is nothing useful to respond to, e.g., error or empty response (see Section 8.2 of [RFC7252]).

- A server should limit the support for IP multicast requests to specific resources where multicast operation is required (Section 11.3 of [RFC7252]).

- An IP multicast request MUST be Non-confirmable (Section 8.1 of [RFC7252]).

- A response to an IP multicast request SHOULD be Non-confirmable (Section 5.2.3 of [RFC7252]).

- A server does not respond immediately to an IP multicast request and should first wait for a time that is randomly picked within a predetermined time interval called the Leisure (Section 8.2 of [RFC7252]).

Additional guidelines to reduce congestion risks defined in this document are as follows:

- A server in a constrained network should only support group communication GET for resources that are small. This can consist, for example, in having the payload of the response as limited to approximately 5% of the IP Maximum Transmit Unit (MTU) size, so that it fits into a single link-layer frame in case IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) (see Section 4 of [RFC4944]) is used.

- A server SHOULD minimize the payload size of a response to a multicast GET on "/.well-known/core" by using hierarchy in arranging link descriptions for the response. An example of this is given in Section 5 of [RFC6690].

- A server MAY minimize the payload size of a response to a multicast GET (e.g., on "/.well-known/core") by using CoAP block-wise transfers [RFC7959] in case the payload is long, returning only a first block of the CoRE Link Format description. For this reason, a CoAP client sending an IP multicast CoAP request to "/.well-known/core" SHOULD support block-wise transfers. See also Section 2.3.6.

- A client SHOULD use CoAP group communication with the smallest possible IP multicast scope that fulfills the application needs. As an example, site-local scope is always preferred over global scope IP multicast if this fulfills the application needs. Similarly, realm-local scope is always preferred over site-local scope if this fulfills the application needs.

### 2.3.5. Observing Resources

The CoAP Observe Option [RFC7641] is a protocol extension of CoAP, that allows a CoAP client to retrieve a representation of a resource and automatically keep this representation up-to-date over a longer period of time. The client gets notified when the representation has changed. [RFC7641] does not mention whether the Observe Option can be combined with CoAP multicast. This section updates [RFC7641] with the use of the Observe Option in a CoAP multicast GET request and defines normative behavior for both client and server.

Multicast Observe is a useful way to start observing a particular resource on all members of a (multicast) group at the same time. Group members that do not have this particular resource or do not allow the GET method on it will either respond with an error status - 4.04 Not Found or 4.05 Method Not Allowed, respectively - or will silently suppress the response following the rules of Section 2.3.1, depending on server-specific configuration.

A client that sends a multicast GET request with the Observe Option MAY repeat this request using the same Token value and the same Observe Option value, in order to ensure that enough (or all) group members have been reached with the request. This is useful in case a number of group members did not respond to the initial request. The client MAY additionally use the same Message ID in the repeated request to avoid that group members that had already received the initial request would respond again. Note that using the same Message ID in a repeated request will not be helpful in case of loss of a response message, since the server that responded already will consider the repeated request as a duplicate message. On the other hand, if the client uses a different, fresh Message ID in the repeated request, then all the group members that receive this new message will typically respond again, which increases the network load.

A client that sent a multicast GET request with the Observe Option MAY follow up by sending a new unicast CON request with the same Token value and same Observe Option value to a particular server, in order to ensure that the particular server receives the request. This is useful in case a specific group member, that was expected to respond to the initial group request, did not respond to the initial request. The client in this case always uses a Message ID that differs from the initial multicast message.

In the above client behaviors, the Token value is kept identical to the initial request to avoid that a client is included in more than one entry in the list of observers (Section 4.1 of [RFC7641]).

Before repeating a request as specified above, the client SHOULD wait for at least the expected round-trip time plus the Leisure time period defined in Section 8.2 of [RFC7252], to give the server time to respond.

A server that receives a legitimate GET request with the Observe Option, for which request processing is successful, SHOULD NOT suppress the response to this request, because the client is obviously interested in the resource representation. A server that adds a client to the list of observers for a resource due to an Observe request MUST NOT suppress the response to this request.

A server SHOULD have a mechanism to verify liveness of its observing clients and the continued interest of these clients in receiving the observe notifications. This can be implemented by sending notifications occassionally using a Confirmable message. See Section 4.5 of [RFC7641] for details. This requirement overrides the regular behavior of sending Non-Confirmable notifications in response to a Non-Confirmable request.

For observing a group of servers through a CoAP-to-CoAP proxy or HTTP-CoAP proxy, the limitations stated in Section 2.3.3 apply.

### 2.3.6. Block-Wise Transfer

Section 2.8 of [RFC7959] specifies how a client can use block-wise transfer (Block2 Option) in a multicast GET request to limit the size of the initial response of each server. The client has to use unicast for any further requests, separately addressing each different server, in order to retrieve more blocks of the resource from that server, if any. Also, a server (group member) that needs to respond to a multicast request with a particularly large resource can use block-wise transfer (Block2 Option) at its own initiative, to limit the size of the initial response. Again, a client would have to use unicast for any further requests to retrieve more blocks of the resource.

A solution for multicast block-wise transfer using the Block1 Option is not specified in [RFC7959] nor in the present document. Such a solution would be useful for multicast PUT/POST/PATCH/iPATCH requests, to efficiently distribute a large request payload as multiple blocks to all members of a CoAP group. Multicast usage of Block1 is non-trivial due to potential message loss (leading to missing blocks or missing confirmations), and potential diverging block size preferences of different members of the multicast group.

### 2.4. Transport

In this document only UDP is considered as a transport protocol, both over IPv4 and IPv6. Therefore, [RFC8323] (CoAP over TCP, TLS, and WebSockets) is not in scope as a transport for group communication.

#### 2.4.1. UDP/IPv6 Multicast Transport

CoAP group communication can use UDP over IPv6 as a transport protocol, provided that IPv6 multicast is enabled. IPv6 multicast MAY be supported in a network only for a limited scope. For example, Section 2.5.2 describes the potential limited support of RPL for multicast, depending on how the protocol is configured.

For a CoAP server node that supports resource discovery as defined in Section 2.4 of [RFC7252], the default port 5683 MUST be supported as per Section 7.1 and 12.8 of [RFC7252] for the "All CoAP Nodes" multicast group. An IPv6 CoAP server SHOULD support the "All CoAP Nodes" groups with at least link-local (2), admin-local (4) and site-local (5) scopes. An IPv6 CoAP server on a 6LoWPAN node (see Section 2.4.3) SHOULD also support the realm-local (3) scope.

Note that a client sending an IPv6 multicast CoAP message to a port that is not supported by the server will not receive an ICMPv6 Port Unreachable error message from that server, because the server does not send it in this case, per Section 2.4 of [RFC4443].

#### 2.4.2. UDP/IPv4 Multicast Transport

CoAP group communication can use UDP over IPv4 as a transport protocol, provided that IPv4 multicast is enabled. For a CoAP server node that supports resource discovery as defined in Section 2.4 of [RFC7252], the default port 5683 MUST be supported as per Section 7.1 and 12.8 of [RFC7252], for the "All CoAP Nodes" IPv4 multicast group.

Note that a client sending an IPv4 multicast CoAP message to a port that is not supported by the server will not receive an ICMP Port Unreachable error message from that server, because the server does not send it in this case, per Section 3.2.2 of [RFC1122].

#### 2.4.3. 6LoWPAN

In 6LoWPAN [RFC4944] networks, IPv6 packets (up to 1280 bytes) may be fragmented into smaller IEEE 802.15.4 MAC frames (up to 127 bytes), if the packet size requires this. Every 6LoWPAN IPv6 router that receives a multi-fragment packet reassembles the packet and refragments it upon transmission. Since the loss of a single fragment implies the loss of the entire IPv6 packet, the performance

in terms of packet loss and throughput of multi-fragment multicast IPv6 packets is typically far worse than the performance of single-fragment IPv6 multicast packets. For this reason, a CoAP request sent over multicast in 6LoWPAN networks SHOULD be sized in such a way that it fits in a single IEEE 802.15.4 MAC frame, if possible.

On 6LoWPAN networks, multicast groups can be defined with realm-local scope [RFC7346]. Such a realm-local group is restricted to the local 6LoWPAN network/subnet. In other words, a multicast request to that group does not propagate beyond the 6LoWPAN network segment where the request originated. For example, a multicast discovery request can be sent to the realm-local "All CoAP Nodes" IPv6 multicast group (see Section 2.4.1) in order to discover only CoAP servers on the local 6LoWPAN network.

## 2.5. Interworking with Other Protocols

### 2.5.1. MLD/MLDv2/IGMP/IGMPv3

CoAP nodes that are IP hosts (i.e., not IP routers) are generally unaware of the specific IP multicast routing/forwarding protocol being used in their network. When such a host needs to join a specific (CoAP) multicast group, it requires a way to signal to IP multicast routers which IP multicast address(es) it needs to listen to.

The MLDv2 protocol [RFC3810] is the standard IPv6 method to achieve this; therefore, this method SHOULD be used by group members to subscribe to the multicast group IPv6 address, on IPv6 networks that support it. CoAP server nodes then act in the role of MLD Multicast Address Listener. Constrained IPv6 networks that implement either RPL (see Section 2.5.2) or MPL (see Section 2.5.3) typically do not support MLD as they have their own mechanisms defined.

The IGMPv3 protocol [RFC3376] is the standard IPv4 method to signal multicast group subscriptions. This SHOULD be used by group members to subscribe to their multicast group IPv4 address on IPv4 networks.

The guidelines from [RFC6636] on the tuning of MLD for mobile and wireless networks may be useful when implementing MLD in constrained networks.

### 2.5.2. RPL

RPL [RFC6550] is an IPv6 based routing protocol suitable for low-power, lossy networks (LLNs). In such a context, CoAP is often used as an application protocol.

If only RPL is used in a network for routing and its optional multicast support is disabled, there will be no IP multicast routing available. Any IPv6 multicast packets in this case will not propagate beyond a single hop (to direct neighbors in the LLN). This implies that any CoAP group request will be delivered to link-local nodes only, for any scope value >= 2 used in the IPv6 destination address.

RPL supports (see Section 12 of [RFC6550]) advertisement of IP multicast destinations using Destination Advertisement Object (DAO) messages and subsequent routing of multicast IPv6 packets based on this. It requires the RPL mode of operation to be 3 (Storing mode with multicast support).

In this mode, RPL DAO can be used by a CoAP node that is either an RPL router or RPL Leaf Node, to advertise its IP multicast group membership to parent RPL routers. Then, RPL will route any IP multicast CoAP requests over multiple hops to those CoAP servers that are group members.

The same DAO mechanism can be used to convey IP multicast group membership information to an edge router (e.g., 6LBR), in case the edge router is also the root of the RPL Destination-Oriented Directed Acyclic Graph (DODAG). This is useful because the edge router then learns which IP multicast traffic it needs to pass through from the backbone network into the LLN subnet. In LLNs, such ingress filtering helps to avoid congestion of the resource-constrained network segment, due to IP multicast traffic from the high-speed backbone IP network.

### 2.5.3. MPL

The Multicast Protocol for Low-Power and Lossy Networks (MPL) [RFC7731] can be used for propagation of IPv6 multicast packets throughout a defined network domain, over multiple hops. MPL is designed to work in LLNs and can operate alone or in combination with RPL. The protocol involves a predefined group of MPL Forwarders to collectively distribute IPv6 multicast packets throughout their MPL Domain. An MPL Forwarder may be associated to multiple MPL Domains at the same time. Non-Forwarders will receive IPv6 multicast packets from one or more of their neighboring Forwarders. Therefore, MPL can be used to propagate a CoAP multicast request to all group members.

However, a CoAP multicast request to a group that originated outside of the MPL Domain will not be propagated by MPL - unless an MPL Forwarder is explicitly configured as an ingress point that introduces external multicast packets into the MPL Domain. Such an ingress point could be located on an edge router (e.g., 6LBR). The

method to configure which multicast groups are to be propagated into the MPL Domain could be:

- Manual configuration on the ingress MPL Forwarder.

- A protocol to register multicast groups at an ingress MPL Forwarder. This could be a protocol offering features similar to MLDv2.

## 3. Unsecured Group Communication

CoAP group communication can operate in CoAP NoSec (No Security) mode, without using application-layer and transport-layer security mechanisms. The NoSec mode uses the "coap" scheme, and is defined in Section 9 of [RFC7252]. The conceptual "NoSec" security group as defined in Section 2.1 is used for unsecured group communication. Before using this mode of operation, the security implications (Section 5.1) must be well understood.

## 4. Secured Group Communication using Group OSCORE

The application-layer protocol Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] provides end-to-end encryption, integrity and replay protection of CoAP messages exchanged between two CoAP endpoints. These can act both as CoAP Client as well as CoAP Server, and share an OSCORE Security Context used to protect and verify exchanged messages. The use of OSCORE does not affect the URI scheme and OSCORE can therefore be used with any URI scheme defined for CoAP.

OSCORE uses COSE [RFC8152] to perform encryption, signing and Message Authentication Code operations, and to efficiently encode the result as a COSE object. In particular, OSCORE takes as input an unprotected CoAP message and transforms it into a protected CoAP message, by using an Authenticated Encryption with Associated Data (AEAD) algorithm.

OSCORE makes it possible to selectively protect different parts of a CoAP message in different ways, while still allowing intermediaries (e.g., CoAP proxies) to perform their intended funtionalities. That is, some message parts are encrypted and integrity protected; other parts are only integrity protected to be accessible to, but not modifiable by, proxies; and some parts are kept as plain content to be both accessible to and modifiable by proxies. Such differences especially concern the CoAP options included in the unprotected message.

Group OSCORE [I-D.ietf-core-oscore-groupcomm] builds on OSCORE, and provides end-to-end security of CoAP messages exchanged between members of an OSCORE group, while fulfilling the same security requirements.

In particular, Group OSCORE protects CoAP requests sent over IP multicast by a CoAP client, as well as multiple corresponding CoAP responses sent over IP unicast by different CoAP servers. However, the same keying material can also be used to protect CoAP requests sent over IP unicast to a single CoAP server in the OSCORE group, as well as the corresponding responses.

Group OSCORE uses digital signatures to ensure source authentication of all messages exchanged within the OSCORE group. That is, sender devices sign their outgoing messages by means of their own private key, and embed the signature in the protected CoAP message.

A Group Manager is responsible for one or multiple OSCORE groups. In particular, the Group Manager acts as repository of public keys of group members; manages, renews and provides keying material in the group; and handles the join process of new group members.

As recommended in [I-D.ietf-core-oscore-groupcomm], a CoAP endpoint can join an OSCORE group by using the method described in [I-D.ietf-ace-key-groupcomm-oscore] and based on the ACE framework for Authentication and Authorization in constrained environments [I-D.ietf-ace-oauth-authz].

A CoAP endpoint can discover OSCORE groups and retrieve information to join them through their Group Managers by using the method described in [I-D.tiloca-core-oscore-discovery] and based on the CoRE Resource Directory [I-D.ietf-core-resource-directory].

If security is required, CoAP group communication as described in this specification MUST use Group OSCORE. In particular, a CoAP group as defined in Section 2.1 and using secure group communication is associated to an OSCORE security group, which includes:

- All members of the CoAP group, i.e. the CoAP endpoints configured (also) as CoAP servers and listening to the group's multicast IP address.

- All further CoAP endpoints configured only as CoAP clients, that send (multicast) CoAP requests to the CoAP group.

### 4.1. Secure Group Maintenance

Additional key management operations on the OSCORE group are required, depending also on the security requirements of the application (see Section 5.2). That is:

- Adding new members to a CoAP group or enabling new client-only endpoints to interact with that group require also that each of such members/endpoints join the corresponding OSCORE group. By doing so, they are securely provided with the necessary cryptographic material. In case backward security is needed, this also requires to first renew such material and distribute it to the current members/endpoints, before new ones are added and join the OSCORE group.

- In case forward security is needed, removing members from a CoAP group or stopping client-only endpoints from interacting with that group requires removing such members/endpoints from the corresponding OSCORE group. To this end, new cryptographic material is generated and securely distributed only to the remaining members/endpoints. This ensures that only the members/endpoints intended to remain are able to continue participating in secure group communication, while the evicted ones are not able to.

The key management operations mentioned above are entrusted to the Group Manager responsible for the OSCORE group [I-D.ietf-core-oscore-groupcomm], and it is RECOMMENDED to perform them according to the approach described in [I-D.ietf-ace-key-groupcomm-oscore].

## 5. Security Considerations

This section provides security considerations for CoAP group communication using IP multicast.

### 5.1. CoAP NoSec Mode

CoAP group communication, if not protected, is vulnerable to all the attacks mentioned in Section 11 of [RFC7252] for IP multicast.

Thus, for sensitive and mission-critical applications (e.g., health monitoring systems and alarm monitoring systems), it is NOT RECOMMENDED to deploy CoAP group communication in NoSec mode.

Without application-layer security, CoAP group communication SHOULD only be deployed in applications that are non-critical, and that do not involve or may have an impact on sensitive data and personal

sphere. These include, e.g., read-only temperature sensors deployed in non-sensitive environments, where the client reads out the values but does not use the data to control actuators or to base an important decision on.

Discovery of devices and resources is a typical use case where NoSec mode is applied, since the devices involved do not have yet configured any mutual security relations at the time the discovery takes place.

### 5.2. Group OSCORE

Group OSCORE provides end-to-end application-level security. This has many desirable properties, including maintaining security assurances while forwarding traffic through intermediaries (proxies). Application-level security also tends to more cleanly separate security from the dynamics of group membership (e.g., the problem of distributing security keys across large groups with many members that come and go).

For sensitive and mission-critical applications, CoAP group communication MUST be protected by using Group OSCORE as specified in [I-D.ietf-core-oscore-groupcomm]. The same security considerations from Section 10 of [I-D.ietf-core-oscore-groupcomm] hold for this specification.

#### 5.2.1. Group Key Management

A key management scheme for secure revocation and renewal of group keying material, namely group rekeying, should be adopted in OSCORE groups. In particular, the key management scheme should preserve backward and forward security in the OSCORE group, if the application requires so (see Section 2.4 of [I-D.ietf-core-oscore-groupcomm]).

Group policies should also take into account the time that the key management scheme requires to rekey the group, on one hand, and the expected frequency of group membership changes, i.e. nodes' joining and leaving, on the other hand.

In fact, it may be desirable to not rekey the group upon every single membership change, in case members' joining and leaving are frequent, and at the same time a single group rekeying instance takes a non negligible time to complete.

In such a case, the Group Manager may consider to rekey the group, e.g., after a minimum number of nodes has joined or left the group within a pre-defined time interval, or according to communication patterns with predictable intervals of network inactivity. This

would prevent paralizing communications in the group, when a slow rekeying scheme is used and frequently invoked.

This comes at the cost of not continuously preserving backward and forward security, since group rekeying might not occur upon every single group membership change. That is, latest joined nodes would have access to the key material used prior to their join, and thus be able to access past group communications protected with that key material. Similarly, until the group is rekeyed, latest left nodes would preserve access to group communications protected with the retained key material.

### 5.2.2. Source Authentication

CoAP endpoints using Group OSCORE countersign their outgoing messages, by means of the countersignature algorithm used in the OSCORE group. This ensures source authentication of messages exchanged by CoAP endpoints through CoAP group communication. In fact, it allows to verify that a received message has actually been originated by a specific and identified member of the OSCORE group.

Appendix F of [I-D.ietf-core-oscore-groupcomm] discusses a number of cases where a recipient CoAP endpoint may skip the verification of countersignatures, possibly on a per-message basis. However, this is NOT RECOMMENDED. That is, a CoAP endpoint receiving a message secured with Group OSCORE SHOULD always verify the countersignature.

### 5.2.3. Countering Attacks

As discussed below, Group OSCORE addresses a number of security attacks mentioned in Section 11 of [RFC7252], with particular reference to their execution over IP multicast.

- Since Group OSCORE provides end-to-end confidentiality and integrity of request/response messages, proxies in multicast settings cannot break message protection, and thus cannot act as man-in-the-middle beyond their legitimate duties (see Section 11.2 of [RFC7252]). In fact, intermediaries such as proxies are not assumed to have access to the OSCORE Security Context used by group members. Also, with the notable addition of countersignatures, Group OSCORE protect messages using the same constructions of OSCORE (see Sections 7.1 and 7.3 of [I-D.ietf-core-oscore-groupcomm]), and especially processes CoAP options according to the same classification in U/I/E classes.

- Group OSCORE prevents to effectively mount amplification attacks (see Section 11.3 of [RFC7252]), e.g. by injecting (small) requests over IP multicast from the (spoofed) IP address of a

victim client, and thus triggering the transmission of several (much bigger) responses back to that client. In fact, upon receiving a request protected with Group OSCORE, a server is able to verify whether the request is fresh and originated exactly by the alleged sender in the OSCORE group (see Section 7.2 of [I-D.ietf-core-oscore-groupcomm]). Furthermore, as also discussed in Section 7 of [I-D.ietf-core-oscore-groupcomm], it is recommended that servers failing to decrypt and verify an incoming message do not send back any error message.

- Group OSCORE limits the impact of attacks based on IP spoofing also over IP multicast (see Section 11.4 of [RFC7252]). In fact, requests and corresponding responses sent in the OSCORE group are encrypted and countersigned (see Sections 7.1 and 7.3 of [I-D.ietf-core-oscore-groupcomm]), and thus can be correctly generated only by legitimate group members. Within an OSCORE group, although the shared symmetric key material used for encryption strictly provides only group-level authentication (see Section 10.1 of [I-D.ietf-core-oscore-groupcomm]), countersignatures ensure source authentication of messages, as originated from the alleged, identifiable sender in the OSCORE group. Note that the server may additionally rely on the Echo option for CoAP described in [I-D.ietf-core-echo-request-tag], in order to verify the aliveness and reachability of the client sending a request from a particular IP address.

- Group OSCORE does not require group members to be equipped with a good source of entropy for generating key material (see Section 11.6 of [RFC7252]), and thus does not contribute to create an attack vector against such (constrained) CoAP endpoints. In particular, the symmetric keys used for message encryption and decryption are derived through the same HMAC-based HKDF scheme used for OSCORE (see Section 3.2 of [RFC8613]). Besides, the OSCORE Master Secret used in such derivation is securely generated by the Group Manager responsible for the OSCORE group, and securely provided to the CoAP endpoints when they join the group.

- Group OSCORE prevents to make any single group member a target for subverting security in the whole OSCORE group (see Section 11.6 of [RFC7252]), even though all group members share (and can derive) the same symmetric key material used for encrypting messages sent to the OSCORE group (see Section 10.1 of [I-D.ietf-core-oscore-groupcomm]). In fact, countersignatures computed with a node's individual private key ensure source authentication of exchanged CoAP messages, as originated from the alleged, identifiable sender in the OSCORE group.

### 5.3. Replay of Non Confirmable Messages

Since all requests sent over IP multicast are Non-confirmable, a client might not be able to know if an adversary has actually captured one of its trasmitted requests and later re-injected it in the group as a replay to the server nodes. In fact, even if the servers sent back responses to the replayed request, the client would not have a valid matching request anymore to suspect of the attack.

If Group OSCORE is used, such a replay attack on the servers is prevented, since a client protects every different request with a different Sequence Number value, which is in turn included as Partial IV in the protected message and takes part in the construction of the AEAD cipher nonce. Thus, a server would be able to detect the replayed request, by checking the conveyed Partial IV against its own replay window in the OSCORE Recipient Context associated to the client.

This requires a server to have a synchronized, up to date view of the sequence number used by the client. If such synchronization is lost, e.g. due to a reboot, or suspected so, the server should use one of the methods described in Appendix E of [I-D.ietf-core-oscore-groupcomm], such as the one based on the Echo option for CoAP described in [I-D.ietf-core-echo-request-tag], in order to (re-)synchronize with the client's sequence number.

### 5.4. Use of CoAP No-Response Option

The CoAP No-Response Option [RFC7967] could be misused by a malicious client to evoke as much responses from servers to a multicast request as possible, by using the value '0' - Interested in all responses. This even overrides the default behaviour of a CoAP server to suppress the response in case there is nothing of interest to respond with. Therefore, this option can be used to perform an amplification attack. A proposed mitigation is to only allow this Option to relax the standard suppression rules for a resource in case the Option is sent by an authenticated client. If sent by an unauthenticated client, the Option can be used to expand the classes of responses suppressed compared to the default rules but not to reduce the classes of responses suppressed.

### 5.5. 6LoWPAN

In a 6LoWPAN network, a multicast IPv6 packet may be fragmented prior to transmission. A 6LoWPAN Router that forwards a fragmented packet can have a relatively high impact on the occupation of the wireless channel and on the memory load of the local node due to packet buffer occupation. For example, the MPL [RFC7731] protocol requires an MPL

Forwarder to store the packet for a longer duration, to allow multiple forwarding transmissions to neighboring Forwarders. If only one of the fragments is not received correctly by an MPL Forwarder, the receiver needs to discard all received fragments and it needs to receive all the packet fragments again on a future occasion.

For these reasons, a fragmented IPv6 multicast packet is a possible attack vector in a Denial of Service (DoS) amplification attack. See Section 11.3 of [RFC7252] for more details on amplification. To mitigate the risk, applications sending multicast IPv6 requests to 6LoWPAN hosted CoAP servers SHOULD limit the size of the request to avoid 6LoWPAN fragmentation. A 6LoWPAN Router or multicast forwarder SHOULD deprioritize forwarding for multi-fragment 6LoWPAN multicast packets. Also, a 6LoWPAN Border Router SHOULD implement multicast packet filtering to prevent unwanted multicast traffic from entering a 6LoWPAN network from the outside. For example, it could filter out all multicast packet for which there is no known multicast listener on the 6LoWPAN network.

## 5.6. Wi-Fi

In a home automation scenario using Wi-Fi, Wi-Fi security should be enabled to prevent rogue nodes from joining. The Customer Premises Equipment (CPE) that enables access to the Internet should also have its IP multicast filters set so that it enforces multicast scope boundaries to isolate local multicast groups from the rest of the Internet (e.g., as per [RFC6092]). In addition, the scope of IP multicast transmissions and listeners should be site-local (5) or smaller. For site-local scope, the CPE will be an appropriate multicast scope boundary point.

## 5.7. Monitoring

### 5.7.1. General Monitoring

CoAP group communication can be used to control a set of related devices: for example, simultaneously turn on all the lights in a room. This intrinsically exposes the group to some unique monitoring risks that devices not in a group are not as vulnerable to. For example, assume an attacker is able to physically see a set of lights turn on in a room. Then the attacker can correlate an observed CoAP group communication message to the observed coordinated group action - even if the CoAP message is (partly) encrypted.
This will give the attacker side-channel information to plan further attacks (e.g., by determining the members of the group some network topology information may be deduced).

### 5.7.2. Pervasive Monitoring

A key additional threat consideration for group communication is pervasive monitoring [RFC7258]. CoAP group communication solutions that are built on top of IP multicast need to pay particular heed to these dangers. This is because IP multicast is easier to intercept (and to secretly record) compared to IP unicast. Also, CoAP traffic is meant for the Internet of Things. This means that CoAP multicast may be used for the control and monitoring of critical infrastructure (e.g., lights, alarms, etc.) that may be prime targets for attack.

For example, an attacker may attempt to record all the CoAP traffic going over a smart grid (i.e., networked electrical utility) and try to determine critical nodes for further attacks. For example, the source node (controller) sends out CoAP group communication messages which easily identifies it as a controller.
CoAP multicast traffic is inherently more vulnerable (compared to unicast) as the same packet may be replicated over many links, leading to a higher probability of packet capture by a pervasive monitoring system.

One mitigation is to restrict the scope of IP multicast to the minimal scope that fulfills the application need. Thus, for example, site-local IP multicast scope is always preferred over global scope IP multicast if this fulfills the application needs.

Even if all CoAP multicast traffic is encrypted/protected, an attacker may still attempt to capture this traffic and perform an off-line attack in the future.

## 6. IANA Considerations

This document has no actions for IANA.

## 7. References

### 7.1. Normative References

[I-D.ietf-core-echo-request-tag]
Amsuess, C., Mattsson, J., and G. Selander, "CoAP: Echo, Request-Tag, and Token Processing", draft-ietf-core-echo-request-tag-09 (work in progress), March 2020.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", draft-ietf-core-oscore-groupcomm-07 (work in progress), March 2020.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8075] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP)", RFC 8075, DOI 10.17487/RFC8075, February 2017, <https://www.rfc-editor.org/info/rfc8075>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

## 7.2. Informative References

[Californium] Eclipse Foundation, "Eclipse Californium", March 2019, <https://github.com/eclipse/californium/tree/2.0.x/californium-core/src/main/java/org/eclipse/californium/core>.

[Go-OCF] Open Connectivity Foundation (OCF), "Implementation of CoAP Server & Client in Go", March 2019, <https://github.com/go-ocf/go-coap>.

[I-D.ietf-ace-key-groupcomm-oscore] Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-05 (work in progress), March 2020.

[I-D.ietf-ace-oauth-authz] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-33 (work in progress), February 2020.

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-pubsub-09 (work in progress), September 2019.

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., Stok, P., and C. Amsuess, "CoRE Resource Directory", draft-ietf-core-resource-directory-24 (work in progress), March 2020.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-37 (work in progress), March 2020.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", draft-tiloca-core-oscore-discovery-05 (work in progress), March 2020.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <https://www.rfc-editor.org/info/rfc6092>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <https://www.rfc-editor.org/info/rfc6550>.

[RFC6636] Asaeda, H., Liu, H., and Q. Wu, "Tuning the Behavior of the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) for Routers in Mobile and Wireless Networks", RFC 6636, DOI 10.17487/RFC6636, May 2012, <https://www.rfc-editor.org/info/rfc6636>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7346] Droms, R., "IPv6 Multicast Address Scopes", RFC 7346, DOI 10.17487/RFC7346, August 2014, <https://www.rfc-editor.org/info/rfc7346>.

[RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", RFC 7390, DOI 10.17487/RFC7390, October 2014, <https://www.rfc-editor.org/info/rfc7390>.

[RFC7731] Hui, J. and R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)", RFC 7731, DOI 10.17487/RFC7731, February 2016, <https://www.rfc-editor.org/info/rfc7731>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8710] Fossati, T., Hartke, K., and C. Bormann, "Multipart Content-Format for the Constrained Application Protocol (CoAP)", RFC 8710, DOI 10.17487/RFC8710, February 2020, <https://www.rfc-editor.org/info/rfc8710>.

## Appendix A. Use Cases

To illustrate where and how CoAP-based group communication can be used, this section summarizes the most common use cases. These use cases include both secured and non-secured CoAP usage. Each

subsection below covers one particular category of use cases for CoRE. Within each category, a use case may cover multiple application areas such as home IoT, commercial building IoT (sensing and control), industrial IoT/control, or environmental sensing.

## A.1. Discovery

Discovery of physical devices in a network, or discovery of information entities hosted on network devices, are operations that are usually required in a system during the phases of setup or (re)configuration. When a discovery use case involves devices that need to interact without having been configured previously with a common security context, unsecured CoAP communication is typically used. Discovery may involve a request to a directory server, which provides services to aid clients in the discovery process. One particular type of directory server is the CoRE Resource Directory [I-D.ietf-core-resource-directory]; and there may be other types of directories that can be used with CoAP.

### A.1.1. Distributed Device Discovery

Device discovery is the discovery and identification of networked devices - optionally only devices of a particular class, type, model, or brand. Group communication is used for distributed device discovery, if a central directory server is not used. Typically in distributed device discovery, a multicast request is sent to a particular address (or address range) and multicast scope of interest, and any devices configured to be discoverable will respond back. For the alternative solution of centralized device discovery a central directory server is accessed through unicast, in which case group communication is not needed. This requires that the address of the central directory is either preconfigured in each device or configured during operation using a protocol.

In CoAP, device discovery can be implemented by CoAP resource discovery requesting (GET) a particular resource that the sought device class, type, model or brand is known to respond to. It can also be implemented using CoAP resource discovery (Section 7 of [RFC7252]) and the CoAP query interface defined in Section 4 of [RFC6690] to find these particular resources. Also, a multicast GET request to /.well-known/core can be used to discover all CoAP devices.

### A.1.2. Distributed Service Discovery

Service discovery is the discovery and identification of particular services hosted on network devices. Services can be identified by one or more parameters such as ID, name, protocol, version and/or

type. Distributed service discovery involves group communication to reach individual devices hosting a particular service; with a central directory server not being used.

In CoAP, services are represented as resources and service discovery is implemented using resource discovery (Section 7 of [RFC7252]) and the CoAP query interface defined in Section 4 of [RFC6690].

### A.1.3. Directory Discovery

This use case is a specific sub-case of Distributed Service Discovery (Appendix A.1.2), in which a device needs to identify the location of a Directory on the network to which it can e.g. register its own offered services, or to which it can perform queries to identify and locate other devices/services it needs to access on the network. Section 3.3 of [RFC7390] shows an example of discovering a CoRE Resource Directory using CoAP group communication. As defined in [I-D.ietf-core-resource-directory], a resource directory is a web entity that stores information about web resources and implements REST interfaces for registration and lookup of those resources. For example, a device can register itself to a resource directory to let it be found by other devices and/or applications.

## A.2. Operational Phase

Operational phase use cases describe those operations that occur most frequently in a networked system, during its operational lifetime and regular operation. Regular usage is when the applications on networked devices perform the tasks they were designed for and exchange of application-related data using group communication occurs. Processes like system reconfiguration, group changes, system/device setup, extra group security changes, etc. are not part of regular operation.

### A.2.1. Actuator Group Control

Group communication can be beneficial to control actuators that need to act in synchrony, as a group, with strict timing (latency) requirements. Examples are office lighting, stage lighting, street lighting, or audio alert/Public Address systems. Sections 3.4 and 3.5 of [RFC7390] show examples of lighting control of a group of 6LoWPAN-connected lights.

### A.2.2. Device Group Status Request

To properly monitor the status of systems, there may be a need for ad-hoc, unplanned status updates. Group communication can be used to quickly send out a request to a (potentially large) number of devices

for specific information. Each device then responds back with the requested data. Those devices that did not respond to the request can optionally be polled again via reliable unicast communication to complete the dataset. The device group may be defined e.g. as "all temperature sensors on floor 3", or "all lights in wing B". For example, it could be a status request for device temperature, most recent sensor event detected, firmware version, network load, and/or battery level.

### A.2.3. Network-wide Query

In some cases a whole network or subnet of multiple IP devices needs to be queried for status or other information. This is similar to the previous use case except that the device group is not defined in terms of its function/type but in terms of its network location. Technically this is also similar to distributed service discovery (Appendix A.1.2) where a query is processed by all devices on a network - except that the query is not about services offered by the device, but rather specific operational data is requested.

### A.2.4. Network-wide / Group Notification

In some cases a whole network, or subnet of multiple IP devices, or a specific target group needs to be notified of a status change or other information. This is similar to the previous two use cases except that the recipients are not expected to respond with some information. Unreliable notification can be acceptable in some use cases, in which a recipient does not respond with a confirmation of having received the notification. In such a case, the receiving CoAP server does not have to create a CoAP response. If the sender needs confirmation of reception, the CoAP servers can be configured for that resource to respond with a 2.xx success status after processing a notification request successfully.

## A.3. Software Update

Multicast can be useful to efficiently distribute new software (firmware, image, application, etc.) to a group of multiple devices. In this case, the group is defined in terms of device type: all devices in the target group are known to be capable of installing and running the new software. The software is distributed as a series of smaller blocks that are collected by all devices and stored in memory. All devices in the target group are usually responsible for integrity verification of the received software; which can be done per-block or for the entire software image once all blocks have been received. Due to the inherent unreliability of CoAP multicast, there needs to be a backup mechanism (e.g. implemented using CoAP unicast) by which a device can individually request missing blocks of a whole

software image/entity. Prior to multicast software update, the group of recipients can be separately notified that there is new software available and coming, using the above network-wide or group notification.

## Acknowledgments

The authors sincerely thank Thomas Fossati and Jim Schaad for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC.

## Authors' Addresses

Esko Dijk
IoTconsultancy.nl
_____\
Utrecht
The Netherlands

Email: esko.dijk@iotconsultancy.nl

Chonggang Wang
InterDigital
1001 E Hector St, Suite 300
Conshohocken PA 19428
United States

Email: Chonggang.Wang@InterDigital.com

Marco Tiloca
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: marco.tiloca@ri.se