ANIMA
Internet-Draft
Intended status: Standards Track
Expires: 24 July 2026

L. Zhu
S. Jiang
BUPT
C. Sheng
Huawei Technologies

20 January 2026

Constrained GeneRic Autonomic Signaling Protocol
draft-ietf-anima-constrained-grasp-02

Abstract

This document proposes the Constrained GeneRic Autonomic Signaling Protocol (cGRASP), a constrained and lightweight variant of the GeneRic Autonomic Signaling Protocol (GRASP, or the standard GRASP). cGRASP reduces message overhead and replaces TCP with CoAP as the transport protocol. By leveraging CoAP’s reliability features and deployment maturity, cGRASP can provide reliable signaling services without relying on TCP, making it suitable for IoT, where lightweight and resource-constrained devices dominate. Furthermore, this document also discusses the potential approaches to adapting the cGRASP to work on the network without IP connectivity.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In IoT that has developed rapidly in recent years, the traditional centralized and human-centered network management methods have gradually shown defects such as low efficiency and high operating costs due to the growth in the number, heterogeneity, diversity, and the increasingly uncertain distribution of devices. Autonomic Network[RFC8993] empowers networks and devices with self-management capabilities, enabling them to self-configure, self-optimize, self-recover, and self-protect without human intervention, effectively improving the stability and reliability of the network, which meets the development needs and trends of IoT and is essential for implementing IoT applications such as smart homes, smart cities, and industrial IoT.

As a new network management solution for TCP/IP networks, the Autonomic Network does not intend to break the existing IP-based network architecture. So does the GRASP[RFC8990], the signaling protocol in the Autonomic Network. While located between the transport layer and the application layer, GRASP provides reliable and efficient services for nodes in the Autonomic Network, like parameter discovery, exchange, and negotiation, based on the TCP/IP protocols. Since it does not provide reliability mechanisms such as error detection, retransmission, and flow control[RFC8990], GRASP must depend on the reliability mechanisms provided by the transport layer, particularly its synchronization and negotiation procedures based on one or more round(s) of message interaction. It is specified in [RFC8990] that GRASP unicast messages MUST use the reliable transport layer protocol, e.g., TCP.

However, the reliability provided by TCP is not free. GRASP must tolerate the inevitable additional latency, control overhead, and memory consumption caused by complex reliability mechanisms of TCP, e.g., the resource consumption and control overhead associated with establishing, maintaining, and closing TCP connections. In addition, the size of the TCP/IP stack on which GRASP relies and the memory resources required to run it are not negligible, e.g., running a standard full TCP/IP stack requires at least tens to hundreds of KBs of data and code memory, and even TCP/IP stacks specifically designed and implemented for resource-constrained devices require tens of KBs of memory. However, the resource-constrained device typically has only about 50KB of memory[RFC7228]. Obviously, in the IoT networks dominated by resource-constrained devices with limited CPU, memory, and power resources, the resource footprint of the TCP/IP stack and its execution, especially the TCP, is likely to be a limiting factor in the deployment of the Autonomic Network and GRASP. Therefore, making GRASP lightweight and removing its dependence on TCP or even IP is of great significance for the deployment and promotion of GRASP in the IoT. In addition, considering the generally short length of interaction messages between IoT nodes, it is also necessary to shorten the length of GRASP messages with the best efforts, especially the control fields, which can also reduce the overhead of nodes in processing, parsing, and sending GRASP messages.

CoAP[RFC7252] is a lightweight, RESTful protocol designed for resource-constrained IoT devices, featuring reliability mechanisms such as acknowledgements, retransmissions, and basic congestion control. It enables efficient communication in low-power and low-bandwidth networks, driving its wide adoption in IoT. Considering the demand for self-management and the resource-constrained nature of IoT devices, as well as the wide adoption and mature ecosystem of CoAP[RFC7252] in low-power and low-bandwidth networks, this document proposes a constrained and lightweight variant of GRASP (cGRASP), a constrained and lightweight variant of GRASP that replaces TCP with CoAP. Additionally, some works on extending CoAP messaging to work over non-IP network scenarios have been proposed, such as its adaptation to Bluetooth Low Energy (BLE) via CoAP over GATT[CoAPoverGATT], which are of great help for the future cGRASP IP-independent extension.

The cGRASP simplifies the objective option and retains GRASP multicast relay mechanism. By reducing message encoding size and processing overhead, and by leveraging the reliability mechanism provided CoAP, cGRASP enables reliable signaling services without relying on TCP. cGRASP is specifically targeted at constrained nodes as defined in [RFC7228], particularly class C1 and C2 devices that are capable of supporting CoAP but not TCP. A representative targeted scenario is Industrial IoT edge coordination: distributed edge controllers and smart equipment(C2 devices) collaboratively manage configuration and status data within a local CoAP/UDP domain, where devices use cGRASP discovery and synchronization messages to autonomically advertise capabilities, propagate firmware-state updates, and align configuration parameters without a central server. This enables lightweight, scalable device management and state synchronization without dependence on TCP-based protocols or centralized configuration services. While Class C0 devices typically lack IP connectivity, the possible IP-independent extension is also discussed, which can extend the use of cGRASP to non-IP networks. By extending Autonomic Network Infrastructure (ANI) signaling capabilities to constrained devices, cGRASP helps realize lightweight, scalable, and self-managing IoT systems.

2. Requirements

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.

3. cGRASP enhancements to the GRASP

cGRASP redefines the standard GRASP Objective option as the cGRASP Objective option, by using a numeric objective identifier to minimize encoding overhead. This is particularly beneficial for constrained nodes with limited processing or transmission capabilities. cGRASP retains the discovery, negotiation, synchronization, and flooding procedures, as well as the defined messages and options (except for the Objective option) of the standard GRASP. In addition, cGRASP still adheres to the High-Level Deployment Model and High-Level Design defined for GRASP, ensuring compatibility with the overall signaling architecture and objectives of the protocol. To distinguish cGRASP from standard GRASP instances, cGRASP instances SHOULD listen for incoming messages on a new well-known port, CGRASP_LISTEN_PORT (TBD1). This section focuses on the cGRASP-specific enhancements to the GRASP.

3.1. cGRASP Objective option

Like standard GRASP, cGRASP messages continue to be transmitted in Concise Binary Object Representation (CBOR)[RFC8949] and be described using Concise Data Definition Language (CDDL)[RFC8610]. In fragmentary CDDL, a cGRASP Objective option follows the pattern:

```
cGRASP objective = [objective-num, objective-flags, loop-count,
                ?objective-value]
objective-num = 0..255
objective-value = any
loop-count = 0..255
objective-flags = uint .bits objective-flag
objective-flag = &(
    F_DISC:    0; valid for discovery
    F_NEG:     1; valid for negotiation
    F_SYNCH:   2; valid for synchronization
    F_NEG_DRY: 3; negotiation is a dry run
)
```

Instead of using a text string (i.e., objective-name) as the unique identifier for the Objective option, the cGRASP Objective option uses an unsigned integer in the range 0 - 255 (i.e., objective-num) as its identifier. The remaining fields of the cGRASP Objective option remain consistent with those defined in the standard GRASP. Objective numbers in the range 0 - 191 are reserved for generic cGRASP Objectives, while numbers in the range 192 - 255 are reserved for privately defined cGRASP Objectives. Each generic cGRASP Objective MUST be assigned a unique objective number and be publicly registered for use by all cGRASP nodes. When a private cGRASP Objective is defined, it MUST also be assigned a uniquely distinguishable objective number and be made public within the specific private domain.

In cGRASP, the identifier of the cGRASP Objective option is changed from a text string to an unsigned integer in the range 0 - 255. This can minimize the size of the encoded cGRASP Objective option, avoid the additional communication cost caused by excessively long objective-name text strings, and eliminate the processing cost of byte-by-byte comparison required in standard GRASP.

3.2. cGRASP constants

* CGRASP_LISTEN_PORT(TBD1)

  A well-known user port that every cGRASP-enabled network device MUST listen to for cGRASP messages.

In addition, the constants for cGRASP also contain the ALL_CGRASP_NEIGHBORS, CGRASP_DEF_TIMEOUT, CGRASP_DEF_LOOPCT, CGRASP_DEF_MAX_SIZE, whose definitions and values are respectively same as the ALL_GRASP_NEIGHBORS, GRASP_DEF_TIMEOUT, GRASP_DEF_LOOPCT, GRASP_DEF_MAX_SIZE in GRASP[RFC8990].

4. CoAP-based transmission for cGRASP

Considering the growing demand for cGRASP and the mature ecosystem of CoAP, utilizing CoAP would significantly benefit the deployment of cGRASP in current IoT networks. This section focuses on the exchange of CoAP-based cGRASP.

4.1. CoAP-based cGRASP overview

To access the cGRASP service over CoAP, this document defines the well-known URI "grasp-coap" (to be assigned by IANA). The /.well-known/grasp-coap URI is used with "coap", "coaps", "coap+tcp", "coaps+tcp", "coaps+ws", or "coap+ws".

CoAP maintains two logical sublayers: the request/response sublayer and the message sublayer. However, the request/response mechanism of CoAP conflicts with the interaction procedures of cGRASP. In particular, it’s challenging to map the multiple rounds of negotiation-related cGRASP messages directly to the CoAP request-response. For this reason, this document utilizes confirmable CoAP messages as carriers for reliable cGRASP message distribution. To minimize modifications to CoAP, cGRASP over CoAP reuses CoAP messages but does not invoke their associated methods. In CoAP-based cGRASP, the cGRASP messages MUST be encapsulated as CoAP payloads with the content-format identifier application/cbor[RFC8949]. Upon receipt of the request with the /.well-known/grasp-coap URI, the CoAP instance MUST parse out the payload and forward it to the cGRASP instance, bypassing associated resource processing. The cGRASP instance SHOULD handle messages from CoAP according to its specification and SHOULD transmit subsequent messages via separate CoAP responses or new requests.

To ensure correct correlation between CoAP requests and responses within a cGRASP session, each cGRASP session over CoAP MUST use a unique CoAP Token generated by the initiating node. This Token MUST be used consistently across all CoAP messages related to the same cGRASP session and MUST NOT be reused across different sessions. Once a session concludes, either successfully (e.g., upon receiving a valid M_END message) or due to failure or timeout, the associated Token SHOULD be considered expired and discarded.

If message retransmission is required, the original confirmable message MUST be retransmitted with the same message ID and token, ensuring the recipient can identify and suppress duplicates appropriately. Since cGRASP does not define any internal idempotency mechanism, recipients of duplicate confirmable messages SHOULD NOT reprocess the cGRASP message logic. Instead, they SHOULD retransmit the original ACK or RST, along with any associated response, if applicable. Relaxed duplicate handling strategies permitted by CoAP for idempotent requests (e.g., GET or PUT) MUST NOT be applied to cGRASP messages unless explicitly specified in future extensions that define cGRASP-level idempotency.

4.2. cGRASP interaction procedures over CoAP

4.2.1. cGRASP negotiation and synchronization over CoAP

The cGRASP negotiation is a bidirectional multi-round procedure, whereas synchronization can be considered a single-round negotiation. The negotiation-related and synchronization-related messages over CoAP SHOULD use the confirmable CoAP POST request or their corresponding response(separate or piggybacked). The following examples illustrate a cGRASP negotiation procedure over CoAP:

```
cGRASP negotiation initiator:
 Header: POST (T=Con, Code=0.02, MID=0x7d38)
 Token: 0x53
 Uri: coap://2001:db8::1/.well-known/grasp
 Content-format: application/cbor
 Accept: application/cbor
 Payload: cGRASP M_REQ_NEG
 with cGRASP objective[objective-num=0,expected-value="A"]

cGRASP negotiation responder:
 Header: (T=ACK, Code=0.00, MID=0x7d38)

 Header: 2.04(Changed) (T=Con, Code=2.04, MID=0xad7b)
 Token: 0x53
 Content-format: application/cbor
 Payload: cGRASP M_WAIT

cGRASP negotiation initiator:
 Header: (T=ACK, Code=0.00, MID=0xad7b)

 Header: POST (T=Con, Code=0.02, MID=0x7d39)
 Token: 0x53
 Uri: coap://2001:db8::2/.well-known/grasp
 Content-format: application/cbor
 Accept: application/cbor
 Payload: cGRASP M_NEGOTIATE
 and cGRASP objective[objective-num=0,expected-value="B"]

cGRASP negotiation responder:
 Header: (T=ACK, Code=0.00, MID=0x7d39)

 Header: 2.04(Changed) (T=Con, Code=2.04, MID=0xad7c)
 Token: 0x53
 Content-format: application/cbor
 Payload: cGRASP M_END with O_ACCEPT

cGRASP negotiation initiator:
 Header: (T=ACK, Code=0.00, MID=0xad7c)
```

The following examples illustrate a cGRASP synchronization procedure over CoAP:

```
cGRASP synchronization initiator:
 Header: POST (T=Con, Code=0.02, MID=0x7d28)
 Token: 0x54
 Uri: coap://2001:db8::1/.well-known/grasp
 Content-format: application/cbor
 Accept: application/cbor
 Payload: cGRASP M_REQ_SYN
 with cGRASP objective[objective-num=0,value = ""]

cGRASP synchronization responder:
 Header: (T=ACK, Code=0.00, MID=0x7d28)

 Header: 2.04(Changed) (T=Con, Code=2.04, MID=0xad8c)
 Token: 0x54
 Content-format: application/cbor
 Payload: cGRASP M_SYNCH
 with cGRASP objective[objective-num=0,value = "A"]

cGRASP synchronization initiator:
 Header: (T=ACK, Code=0.00, MID=0xad8c)
```

4.2.2. cGRASP discovery and flooding over CoAP

A cGRASP discovery process will start with a multicast discovery message(M_DISCOVERY) on the local link, and nodes supporting the discovery objective will respond with discovery response(M_RESPONSE) messages. The cGRASP discovery message over CoAP SHOULD use the non-confirmable CoAP multicast Fetch request with the No-Response option[RFC7967] to suppress unnecessary responses and SHOULD use standard CoAP multicast addresses (e.g., 224.0.1.187 for IPv4, FF0X::FD for IPv6[RFC7252]). The discovery response over CoAP SHOULD use the CoAP unicast POST request. The following examples illustrate the cGRASP discovery and discovery response messages over CoAP:

```
cGRASP discovery initiator:
 Header: FETCH (T=Non, Code=0.05, MID=0x7d28)
 Token: 0x55
 Uri: coap://FF02::13/.well-known/grasp-coap
 Content-format: application/cbor
 Accept: application/cbor
 No-Response
 Payload: cGRASP M_DISCOVERY

 Header: FETCH (T=Non, Code=0.05, MID=0x7d59)
 Token: 0x56
 Uri: coap://224.0.1.187/.well-known/grasp-coap
 Content-format: application/cbor
 Accept: application/cbor
 No-Response
 Payload: cGRASP M_DISCOVERY

cGRASP discovery responder:
 Header: POST (T=Con, Code=0.02, MID=0xad58)
 Token: 0x55
 Uri: coap://2001:db8::1/.well-known/grasp-coap
 Content-format: application/cbor
 Accept: application/cbor
 Payload: cGRASP M_RESPONSE

cGRASP discovery initiator:
 Header: (T=ACK, Code=0.00, MID=0xad58)
```

Since the cGRASP flooding procedure performs network-wide synchronization by propagating a single flooding message, the cGRASP flooding over CoAP SHOULD use the non-confirmable CoAP multicast POST request with the No-Response option.

The following example illustrates the cGRASP flood message over CoAP:

```
cGRASP flooding initiator:
 Header: POST (T=Non, Code=0.02, MID=0x7d28)
 Token: 0x58
 Uri: coap://FF02::13/.well-known/grasp-coap
 Content-format: application/cbor
 No-Response
 Payload: cGRASP M_FLOOD
```

4.2.3. cGRASP relay over CoAP

Given that CoAP does not provide hop-by-hop multicast forwarding across multiple links, cGRASP discovery and flooding over CoAP MUST maintain a relaying function consistent with the relaying behavior defined in [RFC8990], in order to expand the effective multicast scope beyond a single link. A cGRASP node that implements relaying (the "cGRASP relaying instance") MUST apply the procedures below when receiving a multicast cGRASP Discovery or Flood Synchronization message encapsulated in a CoAP request.

The relaying instance MUST maintain a "relay cache" to record an entry for each relayed discovery and flooding message. The cache key MUST be derived solely from the cGRASP payload, i.e., (message_type, session-ID, initiator-Locator).

Each relay cache entry MUST record the following information.

* Incoming interface: the cGRASP-enabled interface on which the Discovery was received.

* Expiration time: a local timeout after which the entry is invalid and MUST be removed. The timeout SHOULD be set to a suitable value because the relaying instance is unaware of the original initiator’s timeout. For a relayed M_DISCOVERY, a recommended value is proportional to the remaining loop-count (e.g., 100 ms * remaining loop-count). For a relayed M_FLOOD, the expiration timer MUST be no less than 2*CGRASP_DEF_TIMEOUT.

Each relay cache entry for discovery message MUST record the following information.

* Upstream transport endpoint: the source IP address and UDP port observed in the received CoAP request that carried the relayed M_DISCOVERY on the incoming interface.

* Upstream CoAP Token (upstream_token): the CoAP Token carried in the received Discovery request (stored for response forwarding).

A relaying instance MUST apply the following eligibility rules before relaying a received multicast cGRASP message.

1. Interface and role eligibility. The relaying instance MUST be a multi-interface cGRASP instance. A single-interface instance MUST NOT perform relaying.

2. Message type eligibility. Only multicast cGRASP messages of type M_DISCOVERY and M_FLOOD received via CoAP multicast on a cGRASP-enabled interface are eligible for relaying. Other message types MUST NOT be relayed.

3. Objective support and cached responder (Discovery only). For an eligible M_DISCOVERY, if the relaying instance supports the discovery objective on the receiving interface, it SHOULD respond locally and MUST NOT relay solely for the purpose of discovery expansion. If it does not support the objective AND it has no cached responder for that objective, it MAY relay as specified below.

Processing steps (after eligibility is met).

1. Loop-count boundary. The relaying instance MUST examine the 'loop-count' field in the cGRASP objective. If loop-count is 0, the message MUST be discarded. Otherwise, the relaying instance MUST decrement loop-count by 1 in the relayed cGRASP message. If the decremented value becomes 0, the message MUST NOT be relayed.

2. Duplicate suppression. If an eligible message with the same relay cache key (message_type, session-id, initiator-locator) has already been relayed within a configured period, the message MUST be discarded to prevent loops and reflooding.

3. Encapsulation and CoAP metadata. If the message is to be relayed, the relaying instance MUST construct a new CoAP request for transmission on each outgoing cGRASP-enabled interface except the incoming interface. The relayed CoAP request MUST use the same CoAP Method as the received CoAP request (e.g., FETCH for Discovery, POST for Flooding), and MUST carry the relayed cGRASP message as CBOR payload. The new CoAP request MUST use fresh transport-layer metadata: a freshly generated CoAP MID and a freshly generated CoAP Token. The cGRASP payload MUST remain unchanged (except for the decremented loop-count).

4. CoAP multicast constraints. The relayed CoAP request MUST be sent as a Non-confirmable (NON) multicast message. For multicast discovery/flooding message, the relayed request SHOULD include the No-Response option to suppress unnecessary CoAP-level responses.

5. Response return (Discovery results forwarding). When the relaying instance receives a M_RESPONSE, it MUST lookup check the relay cache using the key derived from the cGRASP payload. If there is no matching entry in the relay cache (expired or unknown key), the M_RESPONSE SHOULD be discarded. Otherwise, it MUST

forward the discovery result to the upstream initiator by emitting a new unicast CoAP POST request to the recorded upstream endpoint, and MUST set CoAP Token to the recorded upstream Token and a fresh CoAP MID. The forwarded payload MUST carry the received M_RESPONSE unchanged.

6. Rate limiting The relaying instance MUST enforce a reasonable rate limit for relay actions to mitigate denial-of-service and excessive multicast traffic. A fixed threshold or a Trickle-like mechanism may be used[RFC6206].

5. IP-independent discussion

In some IoT scenarios where the need for self-management is urgent, resource-constrained devices in it may not or choose not to support IP connectivity. Therefore, to improve the generality of cGRASP and better support the self-management requirements of the IoT, it is necessary to further discuss how cGRASP adapts to networks without the IP connection.

5.1. How cGRASP adapts to networks without IP

The GRASP and its constrained version cGRASP can only work in IP networks, due to the Locator options used by them. The Locator option is used to locate resources, services, devices, and interfaces on the network and is the basis for GRASP and cGRASP discovery, negotiation, and synchronization procedures. All the four Locator options defined in [RFC8990] have unique identification capabilities only within an IP network: O_IPv6_LOCATOR, O_IPv4_LOCATOR, O_FQDN_LOCATOR, O_URI_LOCATOR, which respectively depend on the IPv6 address, IPv4 address, Fully Qualified Domain Name (FQDN), and Uniform Resource identifier (URI) for identification and location.

Therefore, to enable the cGRASP to work without the IP connection and provide services to cGRASP-enabled nodes, it’s necessary to select an identifier(such as the MAC address in the Ethernet) based on the environment and define a new Locator option in the cGRASP to identify and locate a device, interface, resource, or service that can remove dependence of the cGRASP on IP.

Using cGRASP without the IP connection requires not only the definition of new Locator options but also the identification of cGRASP so that network nodes and devices can recognize cGRASP messages encapsulated in specific bearer protocol messages. For example, [RFC8990] designs GRASP as a user program, using a well-known port to identify GRASP messages. In practice, the protocol identification of cGRASP should be chosen and extended by the bearer protocol on which it depends, which is out of the scope of this document.

5.2. An example: Exchange cGRASP over BLE

In the IoT, where the need for self-management is more urgent, the memory, energy, and computation overheads associated with IP connectivity and transmission may be unacceptable for its resource-constrained devices. In addition, considering the episodic feature of information interactions between IoT devices, some resource-constrained devices may prefer to use low-power and low-bandwidth network connections based on technologies such as Bluetooth Low Energy and Zigbee rather than IP connections. This section discusses how to adapt cGRASP to BLE environments without IP connectivity.

The core protocol used to establish and manage communication between devices in BLE is the Generic Attribute Profile (GATT, Volume 3 PART G in [BTCorev5.4]), which defines how data is transferred between two BLE devices based on the concepts of Services and Characteristics. In BLE, data is transferred and stored in the form of Characteristics, and the Service is a collection of Characteristics, both identified by a unique numeric ID called UUID. GATT is at the top layer of the BLE stack and can provide API interfaces directly to the upper-layer applications, so it is possible to discuss the cGRASP-over-GATT to exchange cGRASP over BLE.

cGRASP-over-GATT can define and use one or more GATT Characteristic(s) to transport cGRASP messages. With the unique identification UUID of the GATT Characteristic, the device can easily recognize whether the transmitted data is a cGRASP message or not. Regarding address identification, BLE devices use a 48-bit device address as a device identifier[BTCorev5.4]. As described in Section 5.1, the cGRASP-over-GATT should define and register a new Locator option based on this identifier.

However, since the read/write semantics of the GATT characteristic do not fully match the semantics of the actions associated with the cGRASP interaction procedures, how to bridge this gap is an important step in realizing cGRASP-over-GATT. In addition, BLE provides both reliable ("write with response", "indicate") and unreliable ("write without response", "notify") data transmission, and how to choose between the two modes of data transmission for cGRASP-over-GATT needs to be carefully considered.

6. IANA Considerations

This document defines the Constrained GeneRic Autonomic Signaling Protocol (cGRASP).

As specified in Section 3.2, the IANA is requested to assign a USER PORT(CGRASP_LISTEN_PORT, TBD1) for use by cGRASP over UDP.

Like the standard GRASP, cGRASP also requires IANA to create the "Constrained GeneRic Autonomic Signaling Protocol (cGRASP) Parameters" registry. The "Constrained GeneRic Autonomic Signaling Protocol (cGRASP) Parameters" should also include two subregistries: "cGRASP Messages and Options" and "cGRASP Objective Numbers".

The "cGRASP Messages and Options" MUST retain all the entries in the "GRASP Messages and Options" subregistry assigned for the standard GRASP.

The initial numbers for the "cGRASP Objective Numbers" subregistry assigned by this document are like the following:

```
0-9 for Experimental
10-255 Unassigned
```

Considerations for IANA regarding CoAP-based transmission for cGRASP in this document are:

* Assignment of the URI /.well-known/grasp-coap

* Assignment of the media type "application/grasp-coap"

* Assignment of the content format "application/grasp-coap"

* Assignment of the resource type (rt=) "core.grasp-coap"

7. Security Considerations

As a constrained version of GRASP, cGRASP must attach importance to the security considerations of GRASP discussed in [RFC8990]. In addition, given the limited capabilities and weak tamper resistance of constrained nodes, as well as their possible exposure to insecure environments, security issues associated with constrained nodes must not be ignored, e.g., the constrained code space and CPU for implementing cryptographic primitives.

TODO more security considerations.

8. References

8.1. Normative References

[BTCorev5.4] Bluetooth Special Interest Group, "BLUETOOTH CORE SPECIFICATION Version 5.4", 31 January 2023, <https://www.bluetooth.com/specifications/specs/core-specification-5-4/>.

[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>.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <https://www.rfc-editor.org/info/rfc6206>.

[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>.

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

8.2. Informative References

[CoAPoverGATT] "CoAP over GATT (Bluetooth Low Energy Generic Attributes)", <https://datatracker.ietf.org/doc/draft-amsuess-core-coap-over-gatt/>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[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>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <https://www.rfc-editor.org/info/rfc8993>.

Appendix A. UDP-based GRASP

This appendix describes an informative description UDP-based cGRASP(proposed in previous versions), which is designed to be a constrained and lightweight version of the GeneRic Autonomic Signaling Protocol(GRASP, or the standard GRASP), with shortened messages and a built-in reliability mechanism. UDP-based cGRASP can work reliably over UDP, making it suitable for IoT, where lightweight and resource- constrained devices dominate. In this appendix, cGRASP stands for UDP-based cGRASP rather than CoAP-based cGRASP.

A.1. Built-in reliability mechanism

UDP-based cGRASP is designed to avoid the additional control overhead and memory consumption caused by TCP, thus matching the capabilities of IoT nodes. Meanwhile, to ensure reliability, the UDP-based cGRASP introduces a message-oriented built-in reliability mechanism.

UDP-based cGRASP uses the 16-bit random number called Nonce to implement the acknowledgment and retransmission mechanism for messages to avoid interaction failures caused by message losses. However, as discussed in Appendix A.2.3, not all cGRASP messages require acknowledgment, such as multicast messages. The UDP-based cGRASP messages that require acknowledgment are referred to in this document as confirmable messages, and the others that do not require acknowledgment are referred to as non-confirmable messages. The transmission of confirmable messages MUST use the reliability mechanism defined in this section, while non-confirmable messages do not.

A.1.1. Reliable transmission for confirmable cGRASP messages

When sending a confirmable message, the UDP-based cGRASP sender MUST generate a 16-bit random Nonce and append the Nonce to the message. Upon receipt of a confirmable message, the receiver MUST acknowledge immediately using the same Nonce as that of the received, or wait for a post-order message in the same direction and piggyback acknowledge with this message within the CGRASP_ACK_DELAYED_TIME. The latter is the delayed acknowledgment, if there is no corresponding message to be sent within the CGRASP_ACK_DELAYED_TIME, an ACK message MUST be sent immediately. UDP-based cGRASP defines two new options, i.e., the REQ-ACK option and the ACK option. The REQ-ACK option is used to carry the Nonce generated by cGRASP for a specific confirmable message and MUST be added to this message as an option. The ACK option also contains a Nonce for acknowledging a corresponding confirmable message, which MUST be added as an option to an ACK message (immediate acknowledgment) or a post-order message in the same direction (delayed acknowledgment). The REQ-ACK option, the ACK option, and the ACK message are defined in Appendix A.2.2.2, Appendix A.2.2.3, and Appendix A.2.3, respectively.

The Nonce can be regarded as the unique identifier of a confirmable message before it is acknowledged. Thus, the cGRASP nodes MUST avoid Nonce conflicts among unacknowledged confirmable messages. Specifically, the Nonce SHOULD be generated by a pseudo-random number generator (PRNG) based on the locally generated unique seed to avoid the conflict of Nonce generated by different nodes in the same network. Meanwhile, the cGRASP instance SHOULD create and maintain a Nonce cache to record the Nonce used by confirmable messages. After generating a Nonce for a message, the cGRASP MUST check whether it conflicts with an existing entry in the Nonce cache, and if it doesn’t, it SHOULD record the Nonce in the cache. Otherwise, the Nonce for the confirmable message MUST be regenerated. After the cGRASP node receives a message with an ACK option(or more than one ACK option), it SHOULD first extract the Nonce from it and check whether there is a corresponding entry with the same Nonce value in the Nonce cache; if not, the received message SHOULD be directly ignored. Otherwise, the cGRASP node SHOULD mark the Nonce entry as acknowledged and delete it when the corresponding cGRASP session is completed. It is worth emphasizing that confirmable messages marked as acknowledged SHOULD also be considered by the aforementioned Nonce conflict detection.

The cGRASP sender MUST set the retransmission timer when sending a confirmable message; see Appendix A.1.2 for details on setting the timeout. If the cGRASP confirmable message does not get an acknowledgment within the retransmission timeout, then the message MUST be retransmitted. The retransmitted message SHOULD retain the same Nonce as the original message. However, when a confirmable message has been accepted and processed by the receiver but is retransmitted due to lost acknowledgment, the cGRASP can not identify the retransmission message and will repeatedly process it, which can be dangerous. Thus, the cGRASP receiver SHOULD record and cache the Nonces of confirmable messages that have been received and processed for each cGRASP session until it is completed and check whether the Nonce of each arriving message conflicts with the cached Nonces, if it doesn’t, then accept and process it. Otherwise, which means the message is a retransmission message, cGRASP SHOULD discard it and send acknowledgment, to avoid duplicated processing of the retransmission and original messages due to the loss of the acknowledgment.

The delayed acknowledgment mechanism can reduce the communication cost caused by the ACK message, but its waiting time may cause unnecessary delay, which reduces the efficiency of communication. In the actual cGRASP implementation, users SHOULD be allowed to enable or completely disable delayed acknowledgment according to their needs.

A.1.2. Retransmission and retransmission timeout

The retransmission timeout for reliable transmission of cGRASP messages is CGRASP_RETRANS_TIMEOUT. If the cGRASP message is not acknowledged within the retransmission timeout and the number of retransmissions does not reach MAX_RETRANS, the message MUST be retransmitted and the retransmission timer SHOULD be reset, the retransmission timeout SHOULD be incremented to twice, and the number of retransmissions SHOULD be incremented by 1. If the cGRASP message is not acknowledged within the retransmission timeout and the number of retransmissions exceeds MAX_RETRANS, the retransmission MUST be discarded, and the transmission fails.

A.2. UDP-based cGRASP definition

cGRASP has made modifications to the standard GRASP by reducing the fixed fields and introducing a message-oriented built-in reliability mechanism with the acknowledgment and retransmission capability based on Nonce. To achieve this, cGRASP redefines the Objective option in standard GRASP as the cGRASP Objective option and defines a new message named ACK message, along with two new options named REQ-ACK option and ACK option. However, cGRASP does not modify the discovery, negotiation, synchronization, and flooding procedures, as well as the defined options (except for the Objective option) of the standard GRASP. In addition, cGRASP still adheres to the High-Level Deployment Model and High-Level Design defined for GRASP, so as not to affect the signaling service provided by the protocol. In order to differentiate from standard GRASP, cGRASP instances SHOULD listen for messages using a new well-known port, CGRASP_LISTEN_PORT (TBD1).

A.2.1. cGRASP message format

Like standard GRASP, cGRASP messages continue to be transmitted in Concise Binary Object Representation (CBOR)[RFC8949] and be described using Concise Data Definition Language (CDDL)[RFC8610]. The session-id in the cGRASP message is shortened from 32 bits to 16 bits to minimize the length of the message, while the meanings of the other fields are still consistent with the standard GRASP message. In fragmentary CDDL, a cGRASP message follows the pattern:

```
cgrasp-message = (message .within message-structure) / noop-message
message-structure = [C_MESSAGE_TYPE, session-id, ?initiator,
                     *cgrasp-option]
C_MESSAGE_TYPE = 0..255
session-id = 0..65535 ; up to 16 bits
cgrasp-option = any
```

A.2.2. cGRASP option

A.2.2.1. cGRASP Objective option

Same as the Section 3.1.

A.2.2.2. REQ-ACK option

The REQ-ACK option is used to indicate that the message MUST be acknowledged by the receiver. When a message needs acknowledgment (i.e., the confirmable message), the sender MUST generate the REQ-ACK option and add it to the message to request the receiver to acknowledge. The REQ-ACK option MUST NOT be allowed to appear in the non-confirmable message (like the Discovery message and the Flood Synchronization message) to avoid a large number of ACK messages in a short time. In fragmentary CDDL, a REQ-ACK option follows the pattern:

```
req-ack-option = [O_REQ_ACK, Nonce]
Nonce = 0..65535
```

Nonce is a 16-bit random number and MUST avoid local conflicts. The Nonce generation and conflict prevention mechanisms are described in Appendix A.1.1.

A.2.2.3. ACK option

cGRASP also defines an ACK option for acknowledging messages carrying a REQ-ACK option. Upon receiving a message with the REQ-ACK option, as specified in Appendix A.1.1, the cGRASP receiver MUST either promptly send an ACK message with a corresponding ACK option; or wait a while for a post-order message in the same direction to be sent and add the ACK option to that message to piggyback acknowledge. The ACK option MUST only be allowed to appear in confirmable messages, as discussed in Appendix A.2.3. In fragmentary CDDL, an ACK option follows the pattern:

```
ack-option = [O_ACK, Nonce]
Nonce = 0..65535; same as the req-ack option
```

Where, the Nonce MUST be the same as the Nonce in the corresponding REQ-ACK option.

A.2.3. cGRASP message

cGRASP reserves all the message types and values of the standard GRASP, as well as the definitions of each related field. cGRASP extends its unicast messages to allow them to carry the REQ-ACK option or the ACK option, enabling cGRASP to implement a built-in reliability mechanism.

All unicast messages used in the three procedures of discovery, negotiation, and synchronization of cGRASP MUST be acknowledged to ensure the reliability and operational efficiency of the interactions. At the same time, these unicast messages are allowed to carry zero or more ACK option(s) to acknowledge the confirmable message belonging to the same or different interaction session(s). In addition, Invalid messages are used to respond to invalid messages and contain related diagnostic information which if lost may affect the subsequent message interactions, thus its acknowledgment is necessary and MUST carry a REQ-ACK option. Similarly, the Invalid message can also carry zero or more ACK option(s) for acknowledgment.

The Discovery message and Flood Synchronization message that is multicast, as well as the NOOP message that does not contain actual information, are not allowed to carry the REQ-ACK option or the ACK option, i.e., non-confirmable message, whose definition is the same as the standard GRASP and will not be repeated here. The CDDL definitions for messages with extension(i.e. the confirmable message) for reliability are defined as follows:

```
response-message = [M_RESPONSE, session-id, initiator, ttl,
                    req-ack-option, *ack-option, (+locator-option
                    // divert-option), ?cGRASP objective]
ttl = 0..4294967295 ; in milliseconds

request-negotiation-message = [M_REQ_NEG, session-id, req-ack-option,
                               *ack-option, cGRASP objective]

request-synchronization-message = [M_REQ_SYN, session-id,
                                   req-ack-option,
                                   *ack-option, cGRASP objective]

negotiation-message = [M_NEGOTIATE, session-id, req-ack-option,
                       *ack-option,cGRASP objective]

end-message = [M_END, session-id, req-ack-option, *ack-option,
               cGRASP accept-option / decline-option]

wait-message = [M_WAIT, session-id, req-ack-option, *ack-option,
                waiting-time]
waiting-time = 0..4294967295 ; in milliseconds

synch-message = [M_SYNCH, session-id, req-ack-option, *ack-option,
                 cGRASP objective]

invalid-message = [M_INVALID, session-id, req-ack-option, *ack-option,
                   ?any]
```

In addition, cGRASP defines an ACK message for immediate acknowledgment. In fragmentary CDDL, an ACK message follows the pattern:

```
ack-message = [M_ACK, ack-option]
```

The Nonce in the ACK option must be the same as the corresponding REQ-ACK option.

A.2.4. cGRASP constants

* CGRASP_LISTEN_PORT(TBD1)

  A well-known UDP user port that every cGRASP-enabled network device MUST listen to for UDP-based messages.

* CGRASP_ACK_DELAYED_TIME(200 milliseconds)

  The default maximum waiting time for delayed acknowledgment.

* CGRASP_RETRANS_TIMEOUT(2000 milliseconds)

  The default timeout is used to determine that a cGRASP confirmable message needs to be resent.

* MAX_RETRANS(3)

  The default maximum times of retransmission for confirmable messages.

In addition, the constants for cGRASP also contain the ALL_CGRASP_NEIGHBORS, CGRASP_DEF_TIMEOUT, CGRASP_DEF_LOOPCT, CGRASP_DEF_MAX_SIZE, whose definitions and values are respectively same as the ALL_GRASP_NEIGHBORS, GRASP_DEF_TIMEOUT, GRASP_DEF_LOOPCT, GRASP_DEF_MAX_SIZE in GRASP[RFC8990].

A.3. IANA Considerations

This document defines the Constrained GeneRic Autonomic Signaling Protocol (cGRASP).

As specified in Appendix A.2.4, the IANA is requested to assign a USER PORT(CGRASP_LISTEN_PORT, TBD1) for use by cGRASP over UDP.

Like the standard GRASP, cGRASP also requires IANA to create the "Constrained GeneRic Autonomic Signaling Protocol (cGRASP) Parameters" registry. The "Constrained GeneRic Autonomic Signaling Protocol (cGRASP) Parameters" should also include two subregistries: "cGRASP Messages and Options" and "cGRASP Objective Numbers".

The "cGRASP Messages and Options" MUST retain all the entries in the "GRASP Messages and Options" subregistry assigned for the standard GRASP, and MUST also add three entries for the new message named "M_ACK", and the two new options named "O_REQ_ACK" and "O_ACK", whose initial values assigned by this document are like the following:

```
M_ACK = 10
O_REQ_ACK = 107
O_ACK = 108
```

The initial numbers for the "cGRASP Objective Numbers" subregistry assigned by this document are like the following:

```
0-9 for Experimental
10-255 Unassigned
```

A.4. Security Considerations

As a constrained version of GRASP, cGRASP must attach importance to the security considerations of GRASP discussed in [RFC8990]. In addition, given the limited capabilities and weak tamper resistance of constrained nodes, as well as their possible exposure to insecure environments, security issues associated with constrained nodes must not be ignored by the external secure infrastructure (e.g., the ACP) on which the cGRASP is based, e.g., the constrained code space and CPU for implementing cryptographic primitives.

Authors' Addresses

Longwei Zhu
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Haidian District, Beijing
China
Email: lwzhu@bupt.edu.cn

Sheng Jiang
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Haidian District, Beijing
China
Email: shengjiang@bupt.edu.cn

Cheng Sheng
Huawei Technologies
Q14 Huawei Campus, No.156 Beiqing Road.
Beijing
China
Email: shengcheng@huawei.com