



# **The Constrained GRASP(cGRASP)** **(draft-ietf-anima-constrained-grasp-02)**

**Longwei Zhu (Presenter), Sheng Jiang , Cheng Sheng**

ANIMA@IETF 125@Shenzhen

Beijing University of Posts and Telecommunications

Huawei Technologies



## cGRASP (History & Overview)

- With the rapid development, IoT is urgent for self-management capabilities enabled by GRASP (but is heavy and resource-consuming due to its TCP-based feature).
- cGRASP replaces TCP with the lightweight CoAP protocol. It reduces message overhead while leveraging CoAP's reliability mechanisms (e.g., Confirmable messages, Tokens) to provide reliable signaling services without TCP.
- The draft has already been adopted by ANIMA as the draft-ietf-anima-constrained-grasp, and version -02 has already been submitted.



## cGRASP (History & Overview)

- cGRASP is defined as an application layer service built on top of CoAP.
  - ❑ reuses CoAP's transport and reliability mechanisms, while encoding cGRASP messages as CoAP payloads.
  - ❑ cGRASP mechanisms are mapped onto CoAP request/response exchanges
    - ✓ Discovery and Flooding → non-confirmable CoAP multicast requests (e.g., FETCH or POST) using the no-response option.
    - ✓ Negotiation and Synchronization → series of confirmable CoAP POST exchanges, where each negotiation round corresponds to a CoAP request/response pair with consistent tokens ensuring session correlation.



## cGRASP(Updates since IETF 124)

- Mechanism improvements:
  - ❑ Designed and optimized the multicast relay mechanism for cGRASP.
- Draft writing:
  - ❑ Formally clarified the typical application scenarios of cGRASP (e.g., Industrial IoT edge coordination) in the text of the draft.
- Prototype development (TCP-based graspy to CoAP-based):
  - ❑ The unicast cGRASP interaction process is almost done.
  - ❑ The multicast relay mechanism (todo)



# Key Update: cGRASP Relay over CoAP

- Since CoAP does not provide hop-by-hop multicast forwarding across multiple links, cGRASP must maintain a relaying function to expand the effective scope of discovery and flooding.
- Introduction of Relay Cache
  - ❑ The relaying instance MUST maintain a "relay cache" using (message\_type, session-ID, initiator-Locator) as the unique key to prevent loops and reflooding.
  - ❑ Cache Entry Contents: Includes incoming interface and expiration time.
  - ❑ For Discovery Messages: It specifically records the Upstream transport endpoint (source IP and UDP port) and the Upstream CoAP Token (to forward responses later)



# Key Update: cGRASP Relay over CoAP

## ➤ Processing & Forwarding Steps

- ❑ Loop-count Check: Decrement loop-count by 1; discard the message if it reaches 0.
- ❑ CoAP Encapsulation: Encapsulate the relayed message into a new CoAP request with fresh metadata (CoAP MID and Token). Send as a Non-confirmable (NON) multicast message with the No-Response option.
- ❑ Response Return (Discovery): Upon receiving an M\_RESPONSE, lookup the relay cache, and forward the result to the upstream initiator via a new unicast CoAP POST request, utilizing the cached Upstream Token.
- ❑ Rate Limiting: Enforce a fixed threshold or Trickle-like mechanism to mitigate DoS risks and excessive multicast traffic



## cGRASP(prototype implementation)

- A prototype of cGRASP based on graspy<sup>1</sup> is ongoing to be developed to facilitate community testing and interoperability verification. The unicast cGRASP interaction process is almost done.
  - ❑ Prolonged negotiations involving M\_WAIT break the simple CoAP request/response lifecycle. If the M\_WAIT sender acts solely as a CoAP-server, it cannot easily push subsequent messages when resources become available.
  - ❑ For this reason, every cGRASP-enabled node must act as both CoAP Client and Server simultaneously.



# cGRASP(prototype implementation)

- Handling M\_WAIT via Role Reversal (todo)
  - ❑ When a node needs to resume a suspended negotiation (after sending/receiving M\_WAIT), it actively switches to the CoAP Client role.
  - ❑ It transmits subsequent cGRASP messages (e.g., M\_NEGOTIATE or M\_END) by initiating a New CoAP Request (POST) to the peer's listening port.
  - ❑ And the combination of the persistent CoAP Token and cGRASP session-id bridges the gap between disconnected CoAP contexts



## cGRASP(next step)

- Draft Refinement (Towards -03):
  - ❑ Add explicit implementation guidelines for mapping P2P cGRASP to C/S CoAP.
  - ❑ Provide concrete interaction examples for bidirectional context handling (e.g., CoAP Client/Server role-reversals after M\_WAIT).
- Prototype Implementation:
  - ❑ Implement the P2P-to-C/S role-reversal logic.
  - ❑ Implement the newly defined multicast relay mechanism



**THANKS!**