

CoAP

- Constrained Application Protocol
- specialized internet application protocol for constrained devices.
- designed to allow small, low-power devices to join the Internet of Things (IoT)
- operates over UDP and provides a request/response
- CoAP is also highly reliable for message delivery, even in cases of limited network connectivity or device power.
- web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things.
- designed to enable simple, constrained devices to join the IoT even through constrained networks with
 - low bandwidth
 - low availability
 - high congestion
 - low power consumption.
- used for machine-to-machine (M2M) applications
- The interaction model of CoAP is similar to the client/server model of HTTP.
- A CoAP request is equivalent to that of HTTP and is sent by a client to request an action (using a Method Code) on a resource (identified by a URI) on a server.
- The server then sends a response with a Response Code; this response may include a resource representation.
- CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP.

CoAP Features

RESTful Architecture

- CoAP uses a RESTful (Representational State Transfer) architecture
- follows a set of constraints that allow it to operate efficiently over a large, distributed network.
- In a RESTful system, data and functionality are considered resources, and these resources are accessed using a standard, uniform interface.
- RESTful architecture allows it to provide a high level of interoperability among different types of devices
- can use standard HTTP methods (such as GET, POST, PUT, and DELETE) to interact with resources.

Built-In Discovery

- allows devices to discover resources on other devices without requiring any prior knowledge of their existence
- built-in discovery feature in CoAP is achieved through the use of a well-known 'core' resource that provides a list of available resources on a device.
- This resource can be queried by other devices on the network, allowing them to discover what resources are available and how to interact with them.

Asynchronous Message Exchanges

- device can send a request to another device and then continue with other tasks without waiting for a response.
- The response can be processed once it arrives, even if delayed.
- This feature uses a message identifier in each CoAP message, which allows a device to match responses with requests.

Optional Reliability with Confirmable Messages

- CoAP offers optional reliability through the use of confirmable messages.
- When a device sends a confirmable message, it expects an acknowledgement from the recipient.
- If no acknowledgement is received within a certain time, the message is retransmitted.
- This feature allows CoAP to provide reliable communication in environments where network connectivity is unreliable.
- Devices can ensure that critical messages are received and processed.

Multicast Support:

- CoAP incorporates support for multicast communication, enabling a single CoAP message to be sent to multiple recipients simultaneously.
- Multicast communication reduces network traffic

Proxying and Caching:

- CoAP supports proxying, allowing intermediary devices to forward requests between clients and servers.
- It also incorporates caching mechanisms to enhance performance and reduce network traffic.

Pros of CoAP

Lightweight

- The CoAP protocol uses a simple binary header, which reduces the amount of data transmitted over the network.
- The header includes important information about the message, such as the type of message, the message ID, and the message code.
- This simplicity and compactness make the protocol efficient and better suited for resource-constrained devices and networks.
- The protocol's communication model is also lightweight. It uses a request-response model similar to HTTP, enabling straightforward communication between devices.

Fast

- The CoAP protocol operates over UDP (User Datagram Protocol).
- UDP is a simple transmission protocol that does not require the establishment of a connection before data is sent.
- This contrasts with TCP (Transmission Control Protocol), which requires a connection to be established before data can be transmitted.
- UDP is useful for IoT devices, which often need to send small amounts of data quickly and efficiently.

- With UDP, the devices can send data as soon as it is ready, without waiting for a connection to be established.

Efficient Encoding

- CoAP uses a binary encoding scheme, which is more efficient than the text-based encoding used by HTTP.
- Binary encoding reduces the size of the messages, which saves bandwidth and increases the speed of communication.
- The CoAP protocol also supports the use of compressed URIs (Uniform Resource Identifiers), which further reduces the size of the messages.
- This is particularly useful in constrained environments, where bandwidth is often limited.
- It also supports the use of separate responses, allowing a device to acknowledge a request before it has processed it.
- This improves the efficiency of the communication and allows devices to manage their resources more effectively.

Stateless Communication

- each request from a client to a server is processed independently, without any knowledge of the previous requests
- this makes the protocol more robust, as it is not affected by the failure of individual requests.
- Stateless communication also simplifies the implementation of the protocol, as it does not require the server to maintain a state for each client.
- This reduces the resource requirements of the server.
- Stateless communication also allows CoAP to support asynchronous communication, which enhances the protocol's flexibility

Cons of CoAP

Less Mature than Alternatives

- fewer resources available for developers, such as libraries and tools, which can make the development process more challenging.
- less widely adopted than its alternatives, which can result in compatibility issues.
- However, CoAP is gaining popularity, and it is expected that these issues will be resolved

NAT Traversal(Network Address Translation)

- NAT is a technique used by routers to share a single IP address among multiple devices
- Because it uses UDP, which does not establish a connection before sending data, CoAP can have issues with NAT traversal, as the router may not know where to send the response.

Fragmentation

- when a message is too large to fit into a single packet and needs to be divided into smaller fragments.
- This can increase the complexity of the protocol and decrease its efficiency.

CoAP Terminologies

- Endpoint
 - An entity participating in the CoAP protocol.
- Sender
 - The originating endpoint of a message.
 - "source endpoint".
- Recipient
 - The destination endpoint of a message.
 - "destination endpoint".
- Client
 - The originating endpoint of a request; the destination endpoint of a response.
- Server
 - The destination endpoint of a request; the originating endpoint of a response.
- Origin Server
 - The server on which a given resource resides or is to be created.
- Intermediary
 - A CoAP endpoint that acts both as a server and as a client
- Proxy
 - An intermediary that mainly is concerned with
 - forwarding requests and relaying back responses
 - possibly performing caching
 - namespace translation
 - protocol translation in the process
 - Based on the position in the overall structure of the request forwarding,
 - there are two common forms of proxy:
 - forward-proxy
 - A "forward-proxy" is an endpoint selected by a client, usually via local configuration rules, to perform requests
 - reverse-proxy.
 - A "reverse-proxy" is an endpoint that stands in for one or more other server(s) and satisfies requests on behalf of these
- CoAP-to-CoAP Proxy
 - A proxy that maps from a CoAP request to a CoAP request
- Cross-Proxy
 - is a proxy that translates between different protocols, such as a CoAP-to-HTTP proxy or an HTTP-to-CoAP proxy.

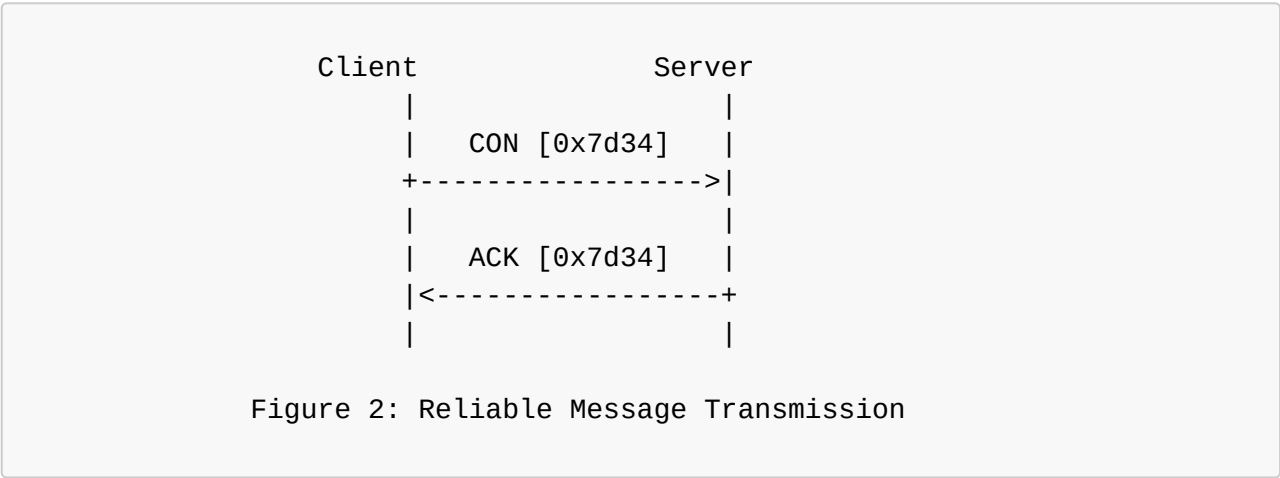
CoAP Messaging Model

Types Message Formats in CoAP

- CoAP employs structured message formats to facilitate efficient communication between IoT devices and applications.
- These formats encapsulate the necessary information for transmitting requests, responses, and control messages.
- There are four primary types of CoAP messages:

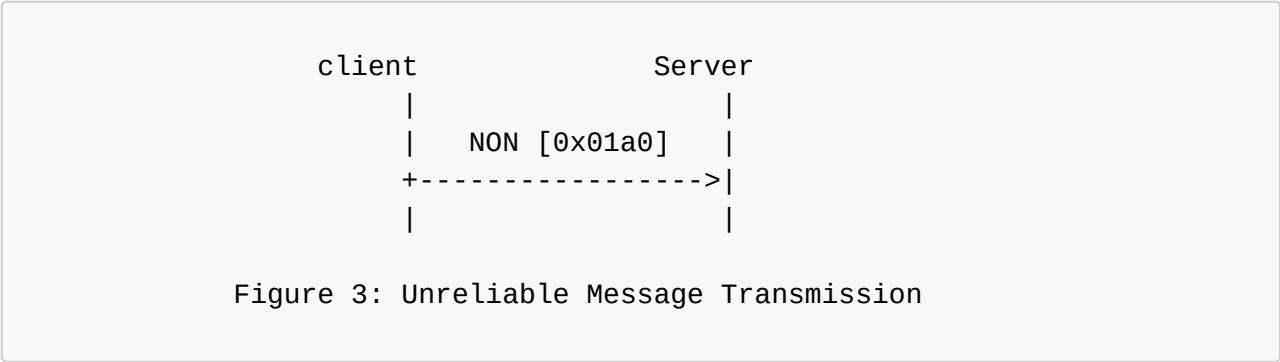
CON (Confirmable) Message:

- CON messages are used for reliable communication, ensuring that the recipient sends an acknowledgment.
- They contain a CoAP request or response and are sent by a client or server, respectively.
- The sender expects an acknowledgment (ACK) from the recipient and retransmits the message until the ACK is received.



NON (Non-Confirmable) Message:

- NON messages are used for faster communication without requiring acknowledgment.
- They are similar to CON messages but don't demand an ACK.
- NON messages are suitable for scenarios where real-time communication is prioritized over reliability.



ACK (Acknowledgment) Message:

- ACK messages are sent in response to CON messages to acknowledge their receipt.
- They indicate that the recipient has successfully received the message and is processing it.

RST (Reset) Message:

- RST messages are sent to cancel a pending CON message that hasn't yet been acknowledged.
- They are typically used when a receiver cannot or chooses not to process a pending message.

CoAP message

- It starts with a fixed 4-byte header, which contains
- CoAP version – This is set to 1, other values are reserved for future versions.
- Type – Indicates if the message is confirmable, non-confirmable, an acknowledgment or reset.
- Token length – Indicates the length of the variable-length token field.
- Code – Split into two parts, class (0-7) and detail (0-31), where class indicates request, success response or error response and detail gives additional information to the class.
- Message ID – Unique ID in network byte order, used to detect duplicates and optionally for reliability.

Request/Response Model

- CoAP request and response semantics are carried in CoAP messages, which include either a Method Code or Response Code, respectively.
- Optional (or default) request and response information, such as the URI and payload media type are carried as CoAP options.
- requests can be carried in Confirmable and Non-confirmable messages
- responses can be carried in these as well as in Acknowledgement messages
- If the server is not able to respond immediately to a request carried in a Confirmable message, it simply responds with an Empty Acknowledgement message so that the client can stop retransmitting the request.
- CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP

Message Format

- CoAP is based on the exchange of compact messages
- each CoAP message occupies the data section of one UDP datagram
- CoAP messages are encoded in a simple binary format.
- CoAP messages uses a short fixed-length binary header (4 bytes) that may be followed by compact binary options and a payload.
- Fields in header:
 - Version
 - Type
 - Token length
 - Code
 - Message ID

- This message format is shared by requests and responses.
- The protocol was designed by the Internet Engineering Task Force (IETF), CoAP is specified in IETF RFC 7252.

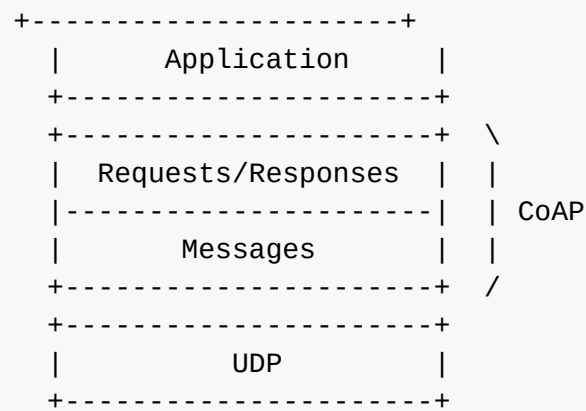


Figure 1: Abstract Layering of CoAP

- <https://www.rfc-editor.org/rfc/rfc7252>