**Features**

1. Web protocol fulfilling M2M requirements in constrained environments
2. UDP Binding
3. Asynchronous message exchanges
4. Low header overhead and parsing complexity
5. HTTP mapping
6. Security binding to Datagram Transport Layer Security (DTLS)

**Constrained Application Protocol**

The interaction model of CoAP deals with interchanges asynchronously over UDP, but may also be used over Datagram Transport Layer Security (DTLS) or other transports such as SMS, TCP, or SCTP. UDP-lite and UDP zero checksum are not supported. The used layer of messages supports optional reliability. CoAP defines four types of messages: Confirmable, Non-confirmable, Acknowledgement, Reset.

**Request/Response Model**

CoAP request and response are carried in CoAP messages, which include either a Method Code or Response Code. Optional information are carried as CoAP options. A request is carried in a Confirmable (CON) or Non-confirmable (NON) message, and if immediately available, the response to a request carried in a CON message is piggybacked. If the server is not able to respond immediately to a request carried in a CON message, it responds with an Empty ACK message so that the client can stop retransmitting the request. When the response is ready, the server sends a separate response. If a request is sent in a NON message, then the response is sent using a new NON message. CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP. The destination endpoint of the initial request and the response's source endpoint must match.

**Message Format**

CoAP messages are encoded in a simple binary format. The message format starts with a fixed-size 4-byte header. This is followed by a variable-length Token value (client-generated), used to match an answer to a request. Following the Token value comes a sequence of zero or more CoAP Options in Type-Length-Value (TLV) format, optionally followed by the Payload Marker (0xFF) and the payload which extends to the end of the UDP datagram. The absence of the Payload Marker denotes a zero-length payload.

**0 1 2 3**

**0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1**

**+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+**

**|Ver| T | TKL | Code | Message ID |**

**+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | | Token (if any, TKL bytes) |**

**+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |**

**| Options (if any) |**

**+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |1 1 1 1 1 1 1 1| Payload (if any) |**

**+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+**

**Figure 1: Message Format**

1. Version (Ver): 2-bit unsigned integer. Indicates the CoAP version number (01 binary).

2. Type (T): 2-bit unsigned integer. Indicates if this message is of type CON (0), NON (1), ACK (2), or RST (3).

3. Token Length (TKL): 4-bit unsigned integer. Indicates the length of the variable-length Token field (0-8 bytes). Lengths 9-15 are reserved.

4. Code: 8-bit unsigned integer, split into a 3-bit class and a 5-bit detail as "c.dd" where "c" is 0 to 7 "dd" is from 00 to 31.

5. Message ID: 16-bit unsigned integer in network byte order.

**Option Format**

Each option instance in a message specifies the Option Number of the defined CoAP option, the length of the Option Value, and the Option Value itself. Instead of specifying the Option Number directly, the instances MUST appear in order of their Option Numbers and a delta encoding is used between them: the Option Number for each instance is calculated as the sum of its delta and the Option Number of the preceding instance in the message.

**0 1 2 3 4 5 6 7**

**+---------------+---------------+**

**| Option Delta | Option Length | 1 byte**

**+---------------+---------------+**

**| Option Delta | 0-2 bytes**

**| (extended) |**

**+-------------------------------+**

**| Option Length | 0-2 bytes**

**| (extended) |**

**+-------------------------------+**

**| Option Value | 0 or more bytes**

**+-------------------------------+**

**Figure 2: Option Format**

1. Option Delta: 4-bit unsigned integer. A value between 0 and 12 indicates the Option Delta. 13, 14, 15 are reserved.

2. Option Length: 4-bit unsigned integer. A value between 0 and 12 indicates the length of the Option Value, in bytes. 13, 14, 15 are reserved.

3. Value: A sequence of exactly Option Length bytes (empty, opaque, uint, string).

**Message Transmission**

As CoAP is bound to unreliable transports, messages may arrive out of order, appear duplicated, or go missing without notice. For this reason, CoAP implements a reliability mechanism that has the following features:

1. Simple stop-and-wait retransmission reliability.
2. Duplicate detection.

**Messages Transmitted Reliably**

The reliable transmission of a message is initiated by marking the message as CON in the CoAP header. A recipient MUST either

1. Acknowledge a CON message with an ACK message

2. Reject the message if it can not process the message properly

Rejecting a CON message is effected by sending a matching RST message. The ACK message MUST carry a response or be Empty. The RST message MUST be Empty. The sender retransmits the CON message at exponentially increasing intervals, until it receives an ACK (or RST message) or runs out of attempts. Retransmission is controlled by two things: a timeout and a retransmission counter.

**Messages Transmitted without Reliability**

A message can be transmitted less reliably by marking the message as NON. A NON message always MUST NOT be Empty or be acknowledged by the recipient. A recipient MUST reject the message if it lacks context to process the message properly by sending a matching RST message.

**+----------+-----+-----+-----+-----+**

**| | CON | NON | ACK | RST |**

**+----------+-----+-----+-----+-----+**

**| Request | X | X | - | - |**

**| Response | X | X | X | - |**

**| Empty | \* | - | X | X |**

**+----------+-----+-----+-----+-----+**

**Table 1: Usage of Message Types**

"\*" means that the combination is used to elicit a RST message ("CoAP ping").

**Message Correlation**

An ACK or RST message is related to a CON message or NON message by means of a Message ID along with additional address information of the corresponding endpoint.The Message ID is generated by the sender of a CON or NON message. The Message ID MUST be echoed in the ACK or RST message by the recipient.

**Message Deduplication**

A recipient might receive the same CON message multiple times and SHOULD acknowledge each duplicate of a CON message using the same ACK or RST message but SHOULD process it only once. A recipient might receive the same NON message multiple times and SHOULD silently ignore any duplicated NON message and SHOULD process it only once.

**Message Size**

The CoAP specification itself provides only an upper bound to the message size. Good upper bounds are 1152 bytes for the message size and 1024 bytes for the payload size.

**Congestion Control**

In order not to cause congestion clients MUST limit the number of simultaneous outstanding interactions that they maintain to a given server. An outstanding interaction is either a CON for which an ACK has not yet been received but is still expected or a request for which neither a response nor an ACK message has yet been received but is still expected.

**Options**

CoAP defines a single set of options that are used in both requests and responses: Content-Format, ETag, Location-Path, Location-Query, Max-Age, Proxy-Uri, Proxy-Scheme, Uri-Host, Uri-Path, Uri-Port, Uri-Query, Accept, If-Match, If-None-Match, Size1.

1. The target resource of a request to a CoAP origin server is specified by the Uri-

Host, Uri-Port, Uri-Path, and Uri-Query Options. The Uri-Host Option specifies the Internet host of the resource being requested, the Uri-Port Option specifies the transport-layer port number of the resource, each Uri-Path Option specifies one segment of the absolute path to the resource, and each Uri-Query Option specifies one argument parameterizing the resource.

1. The Content-Format Option indicates the representation format of the message

payload. The CoAP Accept option can be used to indicate which Content-Format is acceptable to the client.

1. The Location-Path and Location-Query Options together indicate a relative URI that

consists either of an absolute path, a query string, or both. Each Location-Path Option specifies one segment of the absolute path to the resource, and each Location-Query Option specifies one argument parameterizing the resource. The Location-Path and Location-Query Option can contain any character sequence. No percent-encoding is performed. The value of a Location-Path Option must NOT be "." or "..".

1. With conditional request options, a client can ask the server to respond to the

request only in certain conditions. When multiple clients are acting concurrently on the same resource, If-Match is typically helpful for resource update requests and If-None-Match is useful for resource creation requests, such as PUT requests, as a way to prevent accidental overwrites. The If-Match Option can occur multiple times. The condition is fulfilled if any of the options match. The If-None-Match Option carries no value. If the target resource does exist, then the condition is not fulfilled.

Options are classified as either "critical" or "elective." The distinction between these is how an unrecognized option by an endpoint is handled:

* Upon reception, unrecognized options of class "elective" must be silently ignored.
* Unrecognized options of class "critical" that occur in a:
  + Confirmable request must cause the return of a 4.02 (Bad Option) response. This

response should include a diagnostic payload describing the unrecognized option(s).

* + Confirmable response, or piggybacked in an ACK, must cause the response to be

rejected.

* + NON message must cause the message to be rejected.

An Option is identified by an option number, which also provides some additional semantics information (e.g., Whether an option is elective or critical is entirely determined by whether its option number is even or odd).

**Payloads and Representations**

Both requests and responses must include a payload because of the customized packages. The payload of requests or of responses indicating success is typically a representation of a resource or the result of the requested action. Its format is specified by the Internet media type and content coding given by the Content-Format Option.

**Method Definitions**

A request with an unrecognized or unsupported Method Code must generate a 4.05 (Method Not Allowed) piggybacked response.

1. The GET method retrieves a representation for the information that currently

corresponds to the resource identified by the request URI. Upon success, a 2.05 (Content) or 2.03 (Valid) Response Code should be present in the response.

1. According to the POST method, the representation enclosed in the request must be

processed. The actual function performed by the POST method is determined by the origin server and dependent on the target resource. It usually results in a new resource being created or the target resource being updated.

1. The PUT method requests that the resource identified by the request URI be updated

or created with the enclosed representation. The representation format is specified by the media type and content coding given in the Content-Format Option, if provided.

1. The DELETE method requests that the resource identified by the request URI be

deleted.

**Response Code Definitions**

A response is identified by the Code field that indicates the result of the attempt to understand and satisfy the request.

**Success 2.xx**

This class of Response Code indicates that the clients request was successfully received, understood, and accepted.

**2.01 Created** -Used in response to POST and PUT requests.

**2.02 Deleted** - Used in response to requests that cause the resource to cease being available.

**2.04 Changed** - Used in response to POST and PUT requests.

**2.05 Content** - Used in response to GET requests. The payload returned with the response is a representation of the target resource.

**Client Error 4.xx**

This class of Response Code is intended for cases in which the client seems to have erred. These Response Codes are applicable to any request method.

**4.00 Bad Request**

**4.02 Bad Option** - The request could not be understood by the server due to one or more unrecognized or malformed options.

**4.04 Not Found**

**Server Error 5.xx**

This class of Response Code indicates cases in which the server is aware that it has erred or is incapable of performing the request. These Response Codes are applicable to any request method.

**5.00 Internal Server Error**

**CoAP URIs**

**coap URI Scheme**

coap-URI = "coap:" "//" host [ ":" port ] path-abempty [ "?" query ]

The host must NOT be empty. The port subcomponent indicates the UDP port at which the CoAP server is located. The path identifies a resource within the scope of the host and port. It consists of a sequence of path segments separated by a slash character. The query serves to further parameterize the resource. It consists of a sequence of arguments separated by an ampersand character ("&"). An argument is often in the form of a "key=value" pair.