JSON

- JSON stands for JavaScript Object Notation
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays
- JSON Data - A Name and a Value
  - "name":"xyz"
- JSON Objects
  - {"name":"xyz", "address":"pune"}
- JSON Arrays
  - "employees":[{"name":"xyz", "address":"pune"}, {"name":"abc", "address":"mumbai"}, {"name":"pqr", "address":"delhi"}]

CoAP (Constrained Application Protocol)

- 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

- Web Protocol Used in M2M With Constrained Requirements
- Asynchronous Message Exchange
- Low Overhead
- Very Simple To Perform Syntactic Analysis
- (URI) Uniform Resource Identifier
- Proxy and Caching Capabilities

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.

Messaging Model

- CoAP defines four types of messages:
  - Confirmable
    - Reliability is provided by marking a message as Confirmable (CON).
    - Confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions

```
     Client                Server
          |                   |
          |   CON [0x7d34]    |
          +------------------>|
          |                   |
          |   ACK [0x7d34]    |
          |<------------------+
          |                   |

   Figure 2: Reliable Message Transmission
```

- Non-confirmable
  - A message that does not require reliable transmission an be sent as a Non-confirmable message (NON).
  - These are not acknowledged, but still have a Message ID for duplicate detection

```
     client                Server
          |                   |
          |   NON [0x01a0]    |
          +------------------>|
          |                   |

   Figure 3: Unreliable Message Transmission
```

- Acknowledgement
  - recipient sends an Acknowledgement message (ACK) with the same Message ID
- Reset
  - When a recipient is not at all able to process a message, it replies with a Reset message (RST)

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.

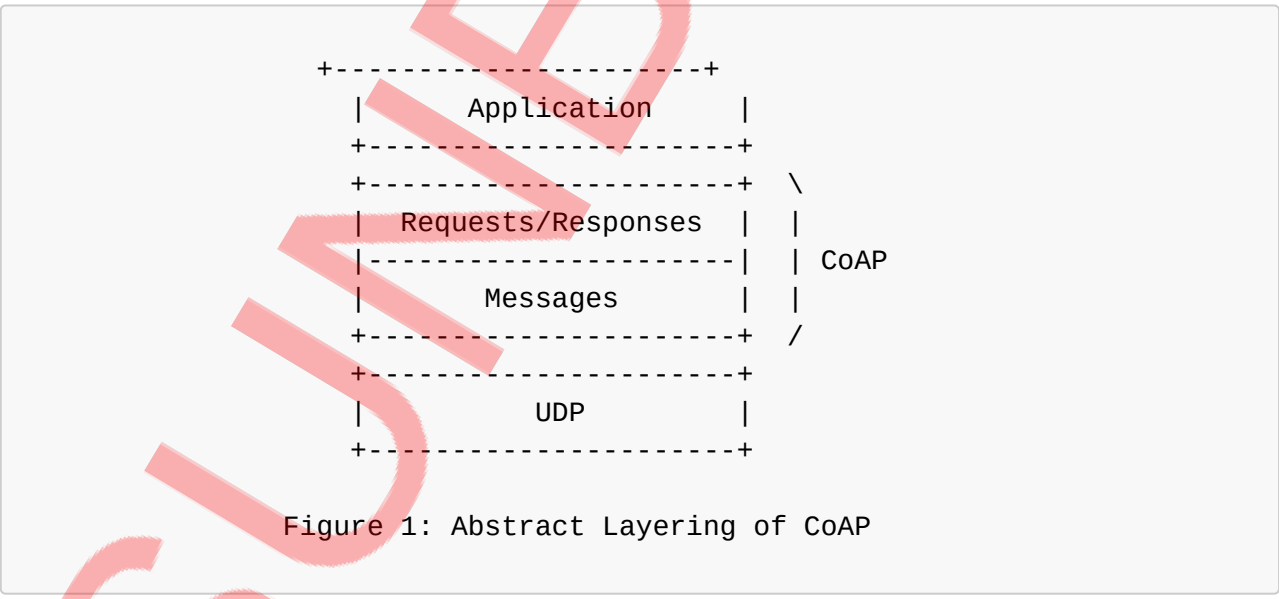

Figure 1: Abstract Layering of CoAP

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

REST Protocol and CoAP

- RESTful protocol refers to REpresentational State Transfer and is operational over HTTP.
- In its case, every entity is treated as a resource and is accessible via the mutual interface.
- REST is hugely powered by web technology but is not solely dependent on HTTP

- Suitable for IoT applications, CoAP is often called a lightweight RESTful.
- It requires less CPU resources and bandwidth on the network if we compare.
- IoT device development is a hefty task if it happens over HTTP as it involves billions of nodes.
- However, due to its nature, architecture and working, CoAP is capable of performing all of this.