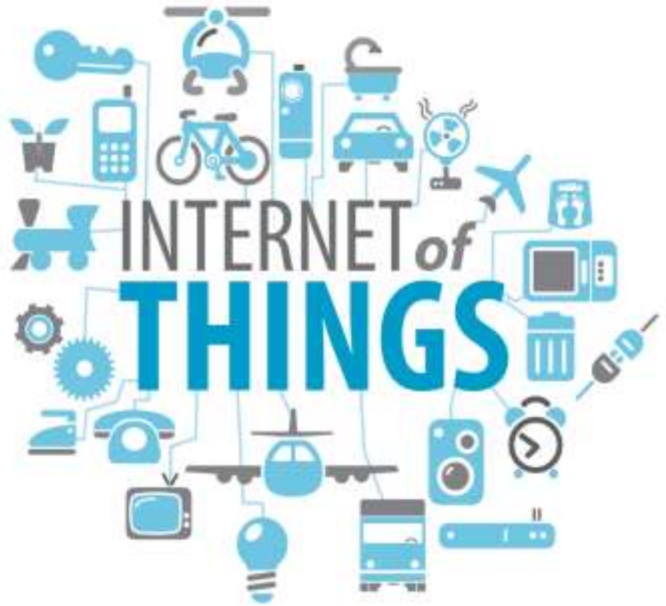


CoAP

Constrained Application Protocol
(Web Protocol for IoT)

Aniruddha Chakrabarti
Associate Vice President and Chief Architect, Digital Practice, Mphasis
ani.c@outlook.com | Linkedin.com/in/aniruddhac | slideshare.net/aniruddha.chakrabarti/ | Twitter - anchakra

Agenda

- What is CoAP
- IoT (Internet of Things) protocol stack
- CoAP - Similarity and differences with HTTP
- Comparison with other IoT protocols
- CoAP Libraries
- CoAP – sample client and server

What is CoAP

“The Constrained Application Protocol (CoAP) is a specialized **web transfer protocol** for use with **constrained** nodes and constrained networks in the **Internet of Things.**

The protocol is designed for machine-to-machine (M2M) and IoT applications such as smart energy and building automation.”

- CoAP is an application layer protocol (similar as HTTP) and follows the request-response pattern used by HTTP – CoAP has a transparent mapping to HTTP
- CoAP uses familiar HTTP stuff like Methods (Get, Post, Put, Delete), Status Codes, URIs, content type / MIME
- Think CoAP as HTTP REST for Constrained environment (low memory, low bandwidth, high rate of packet failure, low power)
- CoAP core protocol spec is specified in RFC 7252
- Like HTTP, CoAP can carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.

http://coap.technology/

IoT protocol stack (or protocol soup?)

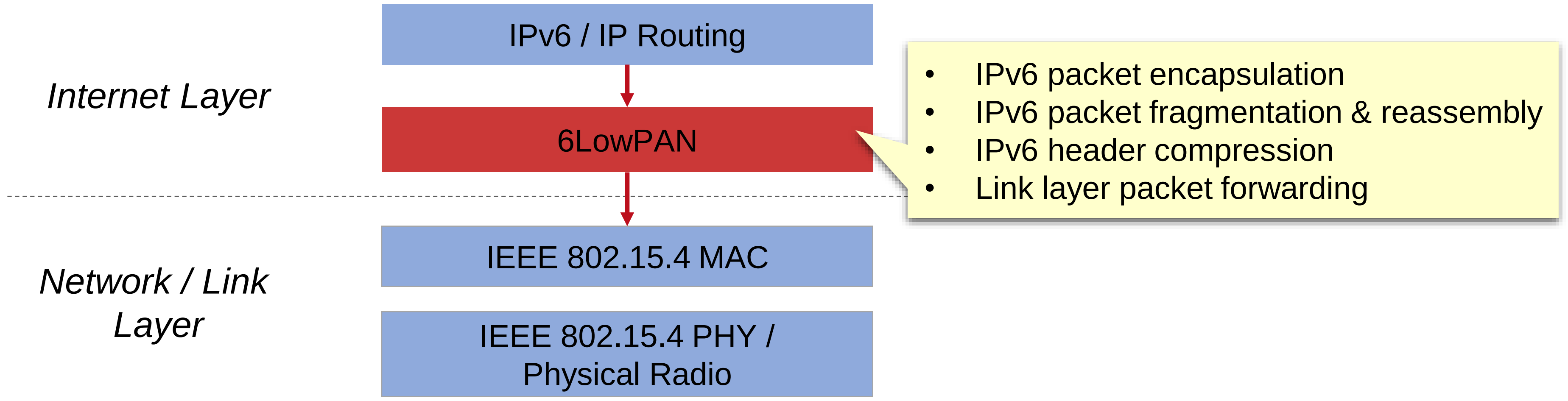
TCP/IP Model	**IoT stack**	**Internet / Web App stack**
	IoT Applications	Web Applications
	Device Management	
Data Format	Binary, JSON, CBOR	HTML, XML, JSON
Application Layer	**CoAP**, MQTT, XMPP, AMQP	HTTP, DHCP, DNS, TLS/SSL
Transport Layer	UDP, DTLS	TCP, UDP
Internet Layer	IPv6 / IP Routing	IPv6, IPv4, IPSec
	6LowPAN	
Network / Link Layer	IEEE 802.15.4 MAC	Ethernet (IEEE 802.3), DSL, ISDN, Wireless LAN (IEEE 802.11), Wi Fi
	IEEE 802.15.4 PHY / Physical Radio	

IEEE 802.15.4 (Network / Link Layer)

- Standard for wireless communication that defines the Physical Layer (PHY) and Media Access Control (MAC) layers.
- Standardized by the IEEE (Institute for Electrical and Electronics Engineers) – similar to IEEE 802.3 for Ethernet, IEEE 802.11 is for wireless LANs (WLANs) or Wi Fi.
- 802.15 group of standards specifies a variety of wireless personal area networks (WPANs) for different applications - For instance, 802.15.1 is Bluetooth.
- IEEE 802.15.4 focusses on communication between devices in constrained environment with low memory, low power and low bandwidth.

6LowPAN (Internet Layer)

- Secret sauce that allows larger IPv6 packets to flow over 802.15.4 links that support much smaller packet size.
- Acronym of IPv6 over Low power Wireless Personal Area Networks - an adaptation layer that allows to transport IPv6 packets over 802.15.4 links. Without 6LowPAN IPv6 and internet protocols would not work in these Low power Wireless Personal Area Networks that uses IEEE 802.15.4.
- 6LoWPAN is an open standard defined in RFC 6282 by the Internet Engineering Task Force (IETF), the standards body that defines many of the open standards used on the Internet such as UDP, TCP and HTTP to name a few.
- As mentioned previously, an IPv6 packet is too large to fit into a single 802.15.4 frame. What 6LowPAN does to fit an IPv6 packet in 802.15.4 frame is -
  - **Fragmentation and Reassembly -** Fragments the IPv6 packet and send it through multiple smaller size packets that can fit in a 802.15.4 frame. On the other end it reassembles the fragmented packets to re-create the IPv6 packet.
  - **Header Compression** – Additionally it also compresses the IPv6 packet header to reduce the packet size.



- IPv6 packet encapsulation
- IPv6 packet fragmentation & reassembly
- IPv6 header compression
- Link layer packet forwarding

CoAP – Similarities with HTTP

- Inspired from HTTP – similar to HTTP CoAP also uses a request response model.
- Can be transparently mapped to HTTP - However, CoAP also provides features that go beyond HTTP such as native push notifications and group communication.
- From a developer point of view, CoAP feels very much like HTTP. Obtaining a value from a sensor is not much different from obtaining a value from a Web API.
- **REST model for small devices** - It implements the REST architectural style
- **Made for billions of nodes with very less memory -** The Internet of Things will need billions of nodes, many of which will need to be inexpensive. CoAP has been designed to work on microcontrollers with as low as 10 KiB of RAM and 100 KiB of code space.
- Designed to use minimal resources, both on the device and on the network. Instead of a complex transport stack, it gets by with UDP on IP. A 4-byte fixed header and a compact encoding of options enables small messages that cause no or little fragmentation on the link layer.
- Like HTTP, CoAP can carry different types of payloads, and can identify which payload type is being used. CoAP integrates with XML, JSON, CBOR, or any data format of your choice.
- **Discovery integrated** - CoAP resource directory provides a way to discover the properties of the nodes (devices/things in IoT) on your network.

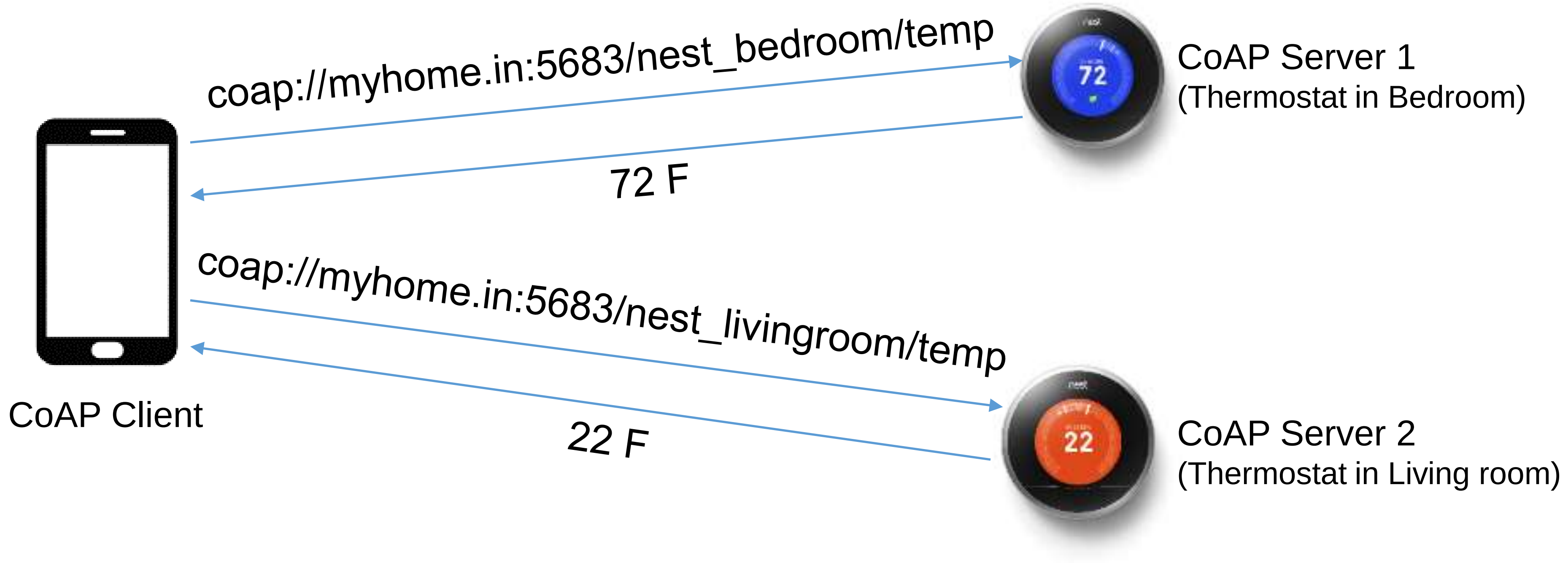
1 KiB (Kibibyte) = 1024 bytes. Kibibyte is established to replace the kilobyte in those computer science contexts in which the term kilobyte is used to mean 1024 bytes. Typically Killo represents 1000, and so causes confusion.

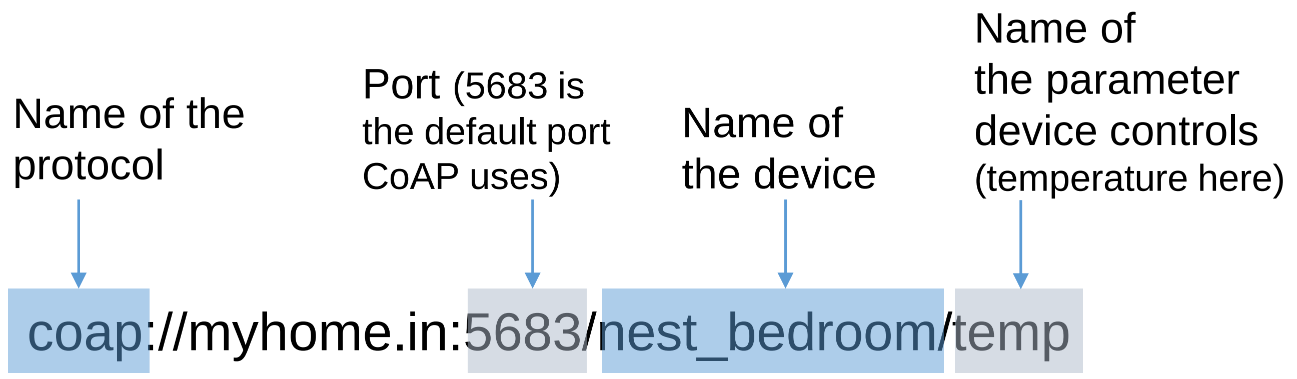
CoAP – How it’s different from HTTP

- CoAP runs over UDP and not TCP (HTTP typically uses TCP, though it can use UDP also)
- CoAP replaces the text headers used in HTTPU (HTTP Unicast) with more compact binary headers
- It reduces the number of options available in the header
- CoAP also reduces the set of methods that can be used; it allows
  - GET
  - POST
  - PUT, and
  - DELETE
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

http://www.iana.org/assignments/core-parameters/

CoAP – Request Response


coap://myhome.in:5683/nest_bedroom/temp
72 F
coap://myhome.in:5683/nest_livingroom/temp
22 F
CoAP Client
CoAP Server 1
(Thermostat in Bedroom)
CoAP Server 2
(Thermostat in Living room)


Name of the protocol
Port (5683 is the default port CoAP uses)
Name of the device
Name of the parameter device controls (temperature here)
coap://myhome.in:5683/nest_bedroom/temp

CoAP Methods

- GET
- POST
- PUT
- DELETE
- OBSERVE (Not present in Http, New in CoAP)
- Method calls can be made using confirmable & nonconfirmable message services
  - When a confirmable message is received, receiver always returns an acknowledgement. The sender resends messages if an acknowledgement is not returned within a given time.
  - When a nonconfirmable message is received, receiver does not return an acknowledgement.
- No of response code has also been reduced (to make implementation simpler)
- CoAP also broke away from the Internet Media Type scheme used in HTTP and other protocols and replaced this with a reduced set of Content-Formats.

http://www.iana.org/assignments/core-parameters/

CoAP Message Types

CON / Confirmable message

A confirmable message requires a response, either a positive acknowledgement or a negative acknowledgement. In case acknowledgement is not received, retransmissions are made until all attempts are exhausted.

NON / Non-confirmable message

A non-confirmable request is used for unreliable transmission (like a request for a sensor measurement made in periodic basis. Even if one value is missed, there is not too much impact). Such a message is not generally acknowledged.

ACK / Acknowledgement

Sent to acknowledge a confirmable (CON) message.

RST / Reset

This represents a negative acknowledgement and means “Reset”. It generally indicates, some kind of failure (like unable to parse received data)

CoAP Status Codes

CoAP Status Code	Description
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
2.31	Continue
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option
4.03	Forbidden
4.04	Not Found
4.05	Method Not Allowed
4.06	Not Acceptable
4.08	Request Entity Incomplete
4.12	Precondition Failed
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

HTTP Status Code	Description
1xx	Informational
2xx	Successful 200 – OK 201 – Created 202 – Accepted 204 – No Content
3xx	Redirection 301 - Moved Permanently 305 - Use Proxy 307 - Temporary Redirect
4xx	Client Error 400 – Bad Request 401 – Unauthorized 403 – Forbidden 404 - Not Found 405 – Method Not Found 408 – Request Timeout
5xx	500 – Internal Server Error 501 – Not Implemented 503 – Service Unavailable 504 - Gateway Timeout

Only mostly used HTTP Status Codes are listed here

CoAP sample servers

- coap://vs0.inf.ethz.ch:5683/
- coap://coap.me:5683/

CoAP client connecting w/ remote server (JavaScript)

```
// Load coap module - similar to node http module
var coap = require('coap');

// coap.me hosts CoAP servers. The default CoAP port is 5683
var remoteServerUrl = 'coap://coap.me:5683/large';

var request = coap.request(remoteServerUrl);

request.on('response', function(response) {
    response.pipe(process.stdout);

    response.on('end', function() {
        process.exit(0);
    });
});

request.end();
```

CoAP server – running locally (JavaScript) – Option 1

```
// Load coap module - similar to node http module
var coap = require('coap');

// Create a CoAP Server object - similar to creating a Http Server object in node
// var http = require('http');
// var httpServer = http.createServer(function(req,res){ });

var coapServer = coap.createServer();   // The default CoAP port is 5683

// This function is called whenever a request arrives at CoAP port 5683
coapServer.on('request', function(req, res){
    res.write('Hello ' + req.url.split('/')[1] + '\n')
    res.end('Hello from CoAP server');
});

// Keeps the CoAP server running
coapServer.listen(function(){
    console.log('CoAP server started!');
});
```

CoAP server – running locally (JavaScript) – Option 2

```
// Load coap module - similar to node http module
var coap = require('coap');

// Create a CoAP Server object - similar to creating a Http Server object in node
// var http = require('http');
// var httpServer = http.createServer(function(req,res){ });

// The default CoAP port is 5683
var coapServer = coap.createServer(function(req, res){
    res.write('Hello ' + request.url.split('/')[1] + '\n')
    res.end('Hello from CoAP server');
});

// Keeps the CoAP server running
coapServer.listen(function(){
    console.log('CoAP server started!');
});
```

CoAP client – connecting w/ local server (JavaScript)

```
var coap = require('coap');
var localServerUrl = 'coap://localhost:5683/world';

var request = coap.request(localServerUrl);

request.on('response', function(response) {
    response.pipe(process.stdout);

    response.on('end', function() {
        process.exit(0);
    });
});

request.end();
```

CBOR

- CBOR stands for **Concise Binary Object Representation**
- CBOR is defined in an Internet Standards Document, RFC 7049.
- CBOR is based on the wildly successful JSON data model: numbers, strings, arrays, maps (called objects in JSON), and a few values such as false, true, and null.
- No predefined schema is required for CBOR, similar to JSON
- Some applications also benefit from CBOR itself being encoded in binary. This saves bulk and allows faster processing – CBOR was designed with IoT in mind.
- CBOR libraries that encode and decode data is available in all popular programming language including Java, C#, JavaScript, Python, Ruby, C, C++, Swift, Go, D, PHP, Haskell and Rust.

Resources

- Wikipedia - https://en.wikipedia.org/wiki/Constrained_Application_Protocol
- Home page - http://coap.technology/
- RFC 7252 Spec - https://tools.ietf.org/html/rfc7252
- CBOR - http://cbor.io/