Constrained Application Protocol of Internet of Things

Dr Banala Saritha
National Institute Of Technology Silchar
32 PUBLICATIONS **130** CITATIONS

International Journal of Engineering &Technology, 7 (1.8) (2018) 244-246

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Constrained Application Protocol of Internet of Things

V. Thirupathi[1]*, Banala Saritha[2], Sridevi Chitti[3]

[1]Department of Computer Science and Engineering, S R Engineering College, Warangal Urban, Telangana, India
[2,3]Department of Electronics and Communication Engineering, S R Engineering College, Warangal Urban, Telangana, India
**Corresponding author E-mail: thirupathi_vadluri@srecwarangal.ac.in*

Abstract

Constrained Application Protocol (CoAP) is an application layer protocol of Internet of Things (IoT) protocol stack. It allows tiny devices to communicate with other tiny devices or constrained devices which uses similar protocols. CoAP is mainly developed for devices like low power and constrained lossy networks. These tiny devices may built using 8 bit microcontrollers with limited Random Access Memory (RAM) , Read only memory (ROM) and constrained networks like 6LoWPANs (IPV6 low power wireless personal area networks). This paper explains constrained application protocol and its applications.

Keywords: *CoAP; IoT; 6LoWPAN; RAM; ROM.*

1. Introduction

The Internet of things (IOT) is group of objects which are interconnected via internet. The objects may be animals, people, mechanical instruments or digital instruments connected through internet with their unique identities. CoAP uses the client/server communication model as the HTTP protocol. In machine-to-machine interactions CoAP client request is similar to HTTP client request, it specifies universal resource identifier (URI) on a server to get a unique resource. CoAP establishes asynchronous communication between machine-to-machine over a user datagram protocol (UDP). CoAP has four kinds of message codes as Confirmable, Non-confirmable, Acknowledgement and Reset. CoAP use these message codes to acknowledge their peers [1].

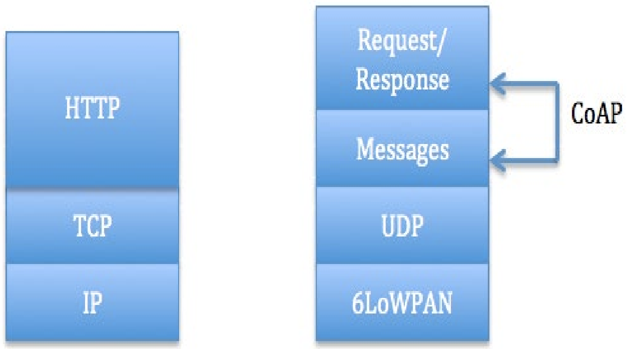


Fig.1: HTTP and CoAP protocol stack

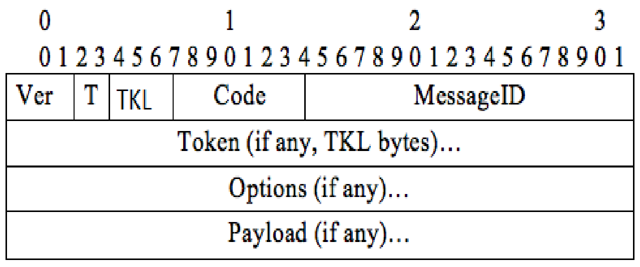
Figure.1 shows HTTP and CoAP protocol stack. HTTP works on Transport control protocol (TCP) is a reliable protocol. But CoAP works on user datagram protocol (UDP) unlike HTTP. In an existing web services applications utilization of CoAP protocol is a good option than other protocols. It is more convenient for constrained environments like energy-limited devices, low-bandwidth links and lossy or congested networks. On congested networks MQTT/TCP works but CoAP/UDP even could not complete handshake protocol. Multicast and Broadcast networks use CoAP protocol. CoAP is nothing but for Constrained Application Protocol is one of the best widespread Machine to Machine (M2M) and IoT principles. It is meant for web transmission protocol and perfect for usage in constrained networks. It's constructed for multicast provision, low overhead. Both CoAP and HTTP were constructed on the REST model [7]. CoAP is appropriate for construction some application where a device is installed in "report only" mode. Once installed, device sends information back to a server. CoAP protocol is most appropriate for smart home applications, which operate on low cost and lightweight resources [6]. The CoAP protocol is closely associated to the traditional web stack based on HTTP. CoAP operates with UDP protocol instead of TCP at transport layer level. Binary encoding used by CoAP whereas HTTP uses textual encoding. But both follows request/response model [6].

CoAP protocol has some standards, following are few of them.
CoAP protocol has some standards, following are few of them.

RFC 7252: It is base CoAP standard.
RFC 7959: CoAP is for transmitting slight payloads but if firmware improvements are required block-wise trans-missions option turn into useful. Data blocks were sent without preserving any state.
RFC 7641: What if we need to get alerted once resource state instabilities on the CoAP server? It would save clients status from polling the server on a consistent basis.
RFC 8323: CoAP with UDP provides reliable transmissions, simple congestion and flow control. What if we would like to have reliable transport? This RFC discusses about CoAP above TCP, TLS and WebSockets.
RFC 6690: Describes a simple arrangement for exposing resources as a base for a resource directory.
RFC 7390: Using the help of multicast property of CoAP, this RFC explains how to use CoAP for group communications [6].
Different open source libraries are available CoAP implementations. The following are few of them.
Libcoap: is a standard developed in C and it is intended to fitting in a wide range of things, from embedded things to big POSIX

devices. It provides support to the authorized RFC 7252 for the client and server side and it also affords provision for numerous extensions, i.e. Observe mode, Block-Wise transfer and Resource Directory.
SMCP: It goals to be employed in things from bare-metal sensors to Linux-based things, comprising embedded de-vices. It provides client and server sides following the RFC 7252 and it is potential to use in BSD sockets or µIP. CoAP supports the Observe mode and Multicast groups.
Californium: It is a Java library for not so constrained things. It aims back ends using JVM and it provides both client and server sides. In addition to the RFC 7252, it provides few extensions, i.e. Observe mode, Block-Wise transfer and Resource Directory. It has optional DTLS support with the Scandium project.
Node-coap: It is java script based library for node js. It supports services at both client and server sides.
CoAPthon: It is a Python library provides support both client and server sides. Along with base features, it also provides core-link format, observe mode, multicast and block-wise transmission extensions.
Microcoap: It is a CoAP standard written in C that aims tiny mi-crocontrollers. The source code offers example application for POSIX and Arduino. It uses the RFC 7252 but it does not provide full support. It offers only the server side services with limited properties. It supports GET, POST and PUT but not DELETE requests [8].

Message Format


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	MessageID
Token (if any, TKL bytes)...				
Options (if any)...				
Payload (if any)...				

Fig.2: Message format

CoAP message model using 2 bit (Ver) to mention version num-ber. T is a 2 bit unsigned to send message response as either con-firm or non-confirm. 0 is for confirmable message and 1 is for non-confirmable message. TKL is four bit token length. Code is 8 bit length message response. Message ID is a 16 bit length for message identity [1].
The main contributions and organization of this paper are summa-rized as follows: In section 2 we describe background details of CoAP protocol. The section 3 proposed work. Finally in section 4 we concluded the paper.

2. Background Work

CoAP protocol follows request/response model as HTTP protocol. But HTTP protocol used for establishing communication among computing devices which have large computing resources viz processor speed, RAM sizes. In contrast CoAP works on user datagram protocol (UDP), is for tiny devices which have con-strained resources viz less processor speed, less in RAM and ROM sizes [2]. CoAP client sends request to the server using con-firmable or non-confirmable messages. If the server accepts re-quest immediately from client, it sends an acknowledgement [3].

2.1. Methods

CoAP is used GET, POST, PUT and DELETE methods to send client request to the server and get response from a server [5].

GET: GET method is used to request to and get response from a server. It requires universal resource identifier (URI) to get a particular resource from a server.

Ex: coap: //loaclhost:8080/home?p1=image1&p2=image2.
It has a drawback i.e the URL displays in the browser's address bar, it leads security issues.

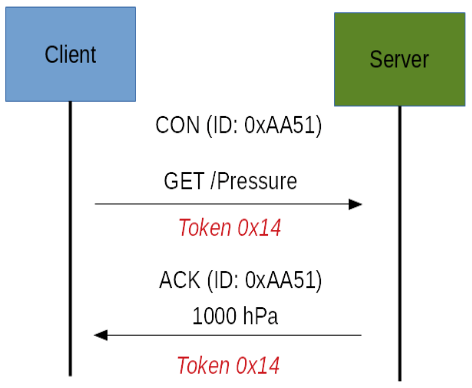


Fig.3: request/response model

POST: POST method is also similar GET method. But it does not show the URL in address bar. It provides security over GET method
Ex: coap: //loaclhost:8080/home
DELETE: DELETE is also similar to above two methods. But it is not safe as POST.
PUT: PUT method used get resources from the server by mentioning universal resource identifier (URI).

2.2. Security in CoAP protocol

CoAP provides security using DTLS (Datagram TLS), which works on user datagram protocol (UDP). It uses ad-vanced encryption standards (AES), RSA to provide security [4].

```
+----------------------+
|      Application     |
+----------------------+
+----------------------+
|  Requests/Responses  |
|----------------------|     CoAP
|       Messages       |
+----------------------+
+----------------------+
|         DTLS         |
+----------------------+
+----------------------+
|          UDP         |
+----------------------+
```

Fig.4: Abstract Layering of DTLS-Secured CoAP

3. Proposed Framework

CoAP protocol used in all automated applications like smart home systems and industrial applications. We can control all home ap-pliances remotely using CoAP. All sensor nodes collect data sends to the server [4].

3.1. Reading Sensor Value Using Node Js

Reading Sensor value using node js: Node js an open source programming language to generate dynamic web content. To work with node js, it has install. Later CoAP protocol needs to install. To install CoAP protocol use the following command.

```
npm install coap –save.
npm install coap-node –save.
npm install smartobject –save.
npm install lwm2m –save.
```

Client application reads sensor data using CoAP protocol.

```
var ss = require(“smartobject”);
var smrtobj = new SmartObject();
smrtobj.init(“temperature”,0, {sensorValue: 21, units: “C”});
smrtobj.init(“lightctrl”,0, {onoff: false});
```

The above code initializes and reads sensor data using temperature sensor.

```
var coapnd=require (“coap-node”);
var coapnode1=new Coap-Node (“LIGHT_CONTROL”,smrtobjt);
coapnode1.on(“registered”, function(){});
coapnode1.register(“172.17.236.81”, 9827, function (err, rsp){
console.log(rsp);});
```

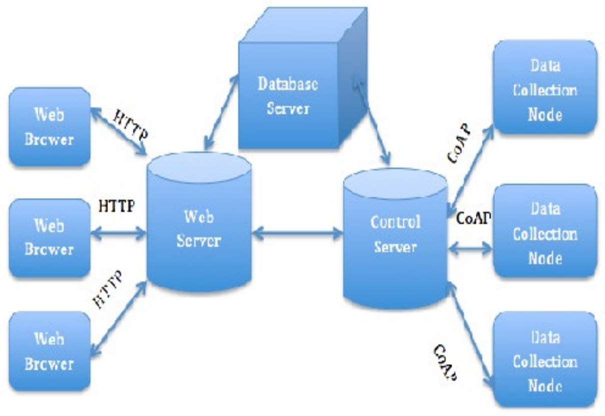


Fig.5: Smart system

The above code register sensor devices so that they are able to sense the data. CoAP protocol needs to mention IP address, port number and resource details.

```
var coapnode1 = ccserver.find(“LIGHT_CONTROL”);
coapnode1.read (“/temperature/0/sensorvalue”, function (err, rsp){
console.log(rsp)});
coapnode1.write(“/temperature/0/sensorvalue”, function (err, rsp){
console.log(rsp)});
```

The above code used for reading sensor data to control home appliance. Response sends to the client. ESP32 is a system on chip microcontroller which has built-in Wi-Fi and Bluetooth devices. In this application ESP32 microcontroller has used to receive information from server side which controls home appliances. It has a CPU of Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS with 520 KiB SRAM [6].

4. Conclusion

Internet of things is an emerging technologies which offers lot automated services. CoAP is a constrained protocol which is used to establish communication among tiny computing devices. With the help of IOT anything can communicate with each over the internet. Internet of Things (IoT) is developing technology and it allows two or more things communicate through internet. IOT architecture has a protocol with different layers via application layer, transport layer, network/ internet layer and data link or physical layer. Application layer deals with user applications which are required to transmit data among user devices. To transmit data to the end user application layer requires CoAP, MQTT, AMQP, XMPP and Web socket protocols. In future will try transfer messages between CoAP devices and MQTT devices.

Acknowledgement

Authors would like to express sincere gratitude to management and principal of SR Engineering College for their support and encouragement to carry out the research work.

References

[1] https://www.cse.wustl.edu/~jain/cse574-4/ftp/coap.
[2] https://tools.ietf.org/pdf/rfc7252.pdf
[3] Dr. D. Kothandaraman, Dr. C. Chellappan, Human Activity Detection System Using Internet Of Things, International Journal on Computer Science and Engineering (IJCSE), Vol. 9 No.11 Nov2017, e-ISSN : 0975-3397 p-ISSN : 2229-5631
[4] V. Thirupathi & C.H. Sandeep, Android Enabled Light via GSM, International Journal of Research, e-ISSN: 2348-6848 p-ISSN: 2348-795X Volume 04 Issue 10 September 2017
[5] V. Thirupathi, CH. Sandeep, G. Madhusri, WEB ENABLED LIGHT USING ARDUINO, International Journal of Research and Applications (Apr-Jun © 2015 Transactions) 2(6): 286-291, eISSN : 2349-0020 & pISSN : 2394-4544.
[6] https://devopedia.org/constrained-application-protocol
[7] https://cesanta.com/proto-coap.html
[8] Markel Iglesias-Urkia, Adrian Orive, Aitor Urbieta “Analysis of CoAP Implementations for Industrial Internet of Things: A Survey” 109C (2017) 188–195