Internet of Things (IoT) with CoAP and HTTP Protocol: A Study on Which Protocol Suits IoT in Terms of Performance

Mohammad Aizuddin Daud(✉) and Wida Susanty Haji Suhaili

School of Computing and Informatics, Universiti Teknnologi Brunei, Bandar Seri Begawan, Brunei Darussalam
phpfreakz.rodriguez@gmail.com,
wida.suhaili@utb.edu.bn

Abstract. Behind Internet of Things (IoT) system, there are constrained devices and protocols that handle all the communication in the system. Constrained devices are equipped with sensor and communication capabilities to allow them to send data over the network. There are limitations on these constrained devices as they have limited resources such as processing power, memory and power consumption. In order to fit the needs for IoT systems, different types of protocols have been developed. These protocols lie in a communication protocol stacks that are from the Application layer, Transport layer, Network Layer and Network access layer. These protocols are designed to cater for the need of the systems to run smoothly with the limited resources available for the constrained devices. For this paper, the focus lies on two (IoT) protocols on the application layer: Hypertext Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP). It extends how the protocol structures the message format, communication establishment and how request is handled from the client. The study is designed on different test beds based on performance factor to meet the requirement of the device's resources. The results and analysis of this study contributes to the findings of the performance where CoAP is faster than HTTP with smaller data. The study strengthens the use of CoAP for constrained devices in relation to the limited resources mentioned before thus contributing to how data are managed in any IoT environment.

Keywords: Constrained devices · CoAP · IoT · HTTP · Protocols

1 Introduction

Home Security System, Home Automation system, Disaster Management are some of the systems developed using the idea of IoT. For this project, the main focus is on how communication of these systems was influenced by the protocol that handles the IoT system. Over the internet, communication between devices is handled by protocols where in each layer a solid communication protocol mechanism for all the IoT building blocks. This is required to create efficient and reliable communication over the IoT systems [1]. Furthermore devices used for the IoT system have constrained resources which are usually associated with limited processing power, limited storage and often

S. Phon-Amnuaisuk et al. (eds.), *Computational Intelligence in Information Systems*, Advances in Intelligent Systems and Computing 532, DOI 10.1007/978-3-319-48517-1_15

runs on battery [2]. These devices are used as nodes for data collected from the sensors. Hence a RESTful approach is used being in favour of low power embedded networks [3]. Researchers in [4] compare CoAP to HTTP in terms of performance and CoAP suitability in constrained scenarios. This paper aims to extend the work to answer the questions: (1) how specific area of communication is managed and (2) how to ensure the connectivity of each of the device are efficient to meet the need of the constrained devices. Therefore the focus of this project is on which communication protocol out of these two is most favourable to IoT systems.

2 HTTP and CoAP

HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems [5]. HTTP is a TCP/IP based protocol that is used to deliver data on the World Wide Web with port 80 as the default port. The 3 basic features of HTTP of connectionless, media independent and stateless make it simple but powerful protocol [6].

While CoAP is one of the latest application layer protocols developed by IETF that facilitates the integration of the embedded network with Web technologies [3]. This protocol is developed specifically for the IoT system which was developed based on the idea of HTTP protocols. CoAP runs over UDP to keep the overall implementation lightweight. It is the aim of CoAP development to keep the overall implementation lightweight since dealing with devices of limited resources.

It uses Restful Architecture which is similar to HTTP. In Restful Architecture, the commands GET, POST, PUT, and DELETE are used to provide resource-oriented interactions in client-server architecture [3]. The CoAP is developed not only for the communications and transferring data, but is also developed along with DTLS. DTLS is a chatty protocol that requires numerous message exchanges to establish a secure session [7]. CoAP uses DTLS for security transaction in the transport layer.

2.1 Constrained Devices

Constrained devices are devices that have limited processing and storage capabilities usually powered by batteries [2]. Constrained devices often used as nodes in the IoT system. Nodes are point of data collection where sensor will collect the data and pass it to the nodes and from the nodes; data will be stored and sent over the internet to the server [8]. Constrained devices also deal with the request and response message from and to the server. Efficiency of the device in processing the messages depends on the size of the message sent.

Here, CoAP have been developed to fit with the constrained devices. With the property of CoAP, IoT system are able to work efficiently with constrained devices since CoAP are designed to be lightweight which put less stress on the constrained devices processing capabilities.

2.2 Process of Communication Made Between HTTP and CoAP

The characteristics of both HTTP and CoAP protocol are based on how they establish server client communication and their message format when exchanging messages between server and client. In this Section, the focus is to compare and contrast on how HTTP and CoAP established their communication between client and server and the packet format for request and response message.

Communication Establishment. The communication for HTTP started with a sequence of handshake protocol [9]. These handshake protocols are made between client and server to establish a connection between both. A synchronization packet is sent to initiate the connection to the server [10]. This packet is sent to setup a reliable session between the client and the server. Once the SYN packet is received by the server, the server will respond with SYN-ACK packet to the client. The SYN-ACK packet is a Synchronization Acknowledgement packet. This packet is sent to acknowledge the client that sessions are allowed to be initiated [10]. Then the client will sent ACK packet, which is an Acknowledgement packet in response for the SYN-ACK packet from the server. This packet will be followed by the establishment of reliable session between the client and the server [10].

The communications in CoAP are considered to be more direct since no handshake occurs between client and server. The differences between HTTP and CoAP in transport layer are CoAP which make use of UDP and HTTP uses TCP. In order to satisfy IoT requirement, devices in IoT can only have small or limited resources. This is how CoAP contributes to the definition of lightweight communication as CoAP uses UDP, UDP's properties are low overhead [11]. This contributes to the requirement of device resources, which makes CoAP suitable for IoT. CoAP communication started with CON. CON is a confirmable message that are sent from the client. CON message carry a request message to the server. Once CON is received by the server, it will elicit the ACK message that is the acknowledgement message. The ACK message carries the response message for the request that is made by the client.

2.3 HTTP and CoAP Message Format

HTTP Message Format. In HTTP, there are two different message formats for request and response: HTTP request and HTTP response message. Once the reliable session is initiated, the client will sent HTTP request. The HTTP message size ranges from 2 kb to 8 kb which is equivalent to 2000 bytes and 8000 bytes [12].

The HTTP request message format contains Method, URL, Version, Header Lines fields and an Entity body. Method that is specified in the message format refers to the request from the client [12]. The request methods used in this project are:

(1) GET - Retrieve information from the server on specified URL
(2) POST - Submit data to be processed on specified URL
(3) PUT - Edit or add existing information in the server on specified URL
(4) DELETE - Delete existing information in the server on specified URL

In the Header Lines Section, it contains Header field name and value. This is where additional information of the request and detail of the client are stored [10] with information such as Host, Connection, User-agent and Language.

Entity Body is part of the HTTP request message format that is only used with post method and response message [10]. Header field is similar to the Header Lines Section but with additional fields such as Date, Content Length and Content Type.

CoAP Message Format. Unlike HTTP, CoAP only have one message format which is used by both request and response message. Both the request and response message are using the same format. The minimum message size for CoAP is 4 bytes and the maximum is 1024 bytes [11]. The message format for CoAP consists of Version, Type, Token Length, Code, Message ID, Options and Payload.

Version in the CoAP message format refers to the version of CoAP protocol used in [11]. Types are message types that are represented by numbers. Message types are as follow along with its number representation [11].

(1) CON (Confirmable) – 0
(2) NON (Non-Confirmable) – 1
(3) ACK (Acknowledgement) – 2
(4) RST (Reset-Message) – 3

The Code in the CoAP message format refers to the Request Method. Each method is represented using numbers. The request methods are listed below [13] which are similar to those of HTTP:

(1) GET = 1
(2) POST = 2
(3) PUT = 3
(4) DELETE = 4

The Message ID in the CoAP message format is the number to identify the message sent [11]. This ID prevents message duplication. The minimum message size for CoAP is 4 bytes and the maximum is 1024 bytes contributes to the lightweight size of CoAP compared to HTTP message format [11].

3 Implementation

The implementation of the system was based on the Client Server architecture. In Client Server architecture, the client provides the user with user interface that allows user interaction and the server deals with database and processing the request from the client [5]. In order to show the differences in terms of performance, and to make the result more reliable, settings were kept constant such as running both clients and servers in the same platform.

3.1 HTTP and CoAP Implementation

Both HTTP and CoAP server used the CoAPthon and Django REST Framework. The server was hosted by Ubuntu 14.04 using VMware player. Whilst for CoAP client, the Copper Extension of Mozilla Firefox browser and for HTTP client, Mozilla Firefox browser was used.

The client server architecture was set under an environment which connects to a network allowing them to communicate and send information over the network. The client will be the Mozilla Firefox browser; server will be the VMware which runs Ubuntu and host the HTTP and CoAP server. The protocol represents the link that governs the communication between both client and server over the network.

3.2 Network Performance Measures

Different types of network performance measures can be used. Each measure gave different types of result depending on what kind of performance measured. Such as:

1. Throughput - Measure of how much actual data can be sent per unit of time across a network.
2. Latency - Amount of time taken for data to travel from one location to another across the network.
3. Jitter - Variation in the delay of received packets.
4. Error Rate - Measure of the amount of error encountered during data transmission over a network. The higher the error, the less reliable the network connection is.

For this study, time was used as the measure, which is similar to latency. Time is referred to the time taken for the request and response process to complete and measured in seconds.

3.3 Testing

This implementation was tested on different scenarios to see how different network structure will affect the performance of the server. The scenarios were based on 5 different network structures: Wireless Home Network, Wired Home Network, UTB Network, Portable Wi-Fi Network and Wireless Hotspot Network. The testing process was done by sending Request from the client using four different methods that are GET, POST, PUT and DELETE. These methods triggered the server to send Response according to what have been requested by the client. Wireshark was used to record and analyse the packets sent between the client and the server.

For HTTP, time was measured from the first SYN message to the last ACK message. For CoAP, time was measured from the first CON message to the last ACK message. Data sizes were set for each method: GET 14 bytes, POST 9 bytes, PUT 9 bytes and DELETE 8 bytes.

4 Findings

For both protocols, the time taken was based on how communication was established. In each scenario, results were based on the test using the four methods of GET, POST, PUT and DELETE which were captured using Wireshark and the time taken was analysed from the moment communication was initiated until completed. The results obtained were placed according to the four methods for all the five scenarios. HTTP took longer time to complete request for each method than CoAP. The HTTP difference in terms of time taken can be clearly seen from the HTTP-REQUEST to HTTP-RESPONSE. The time taken patterns for both CoAP and HTTP in all 5 scenarios were similar for all the four methods. The HTTP requests for all methods were relatively higher for GET and POST method. As for CoAP, the time taken was similar with all the methods except for PUT method where it was slightly higher than the other but the differences can barely be seen.

4.1 Large Data

With small data the differences was barely seen for CoAP and similar for HTTP in all the methods, next was to test on a set large data of 1990 bytes and applied to the four methods, for both CoAP and HTTP to complete request with big data.

GET Large Data on Home Wireless Network. The Table 1 shows the time taken CoAP to complete the GET methods on large data for both request. The time taken for CoAP to complete the request is 24.273145 s.

Table 1. CoAP GET large data time taken

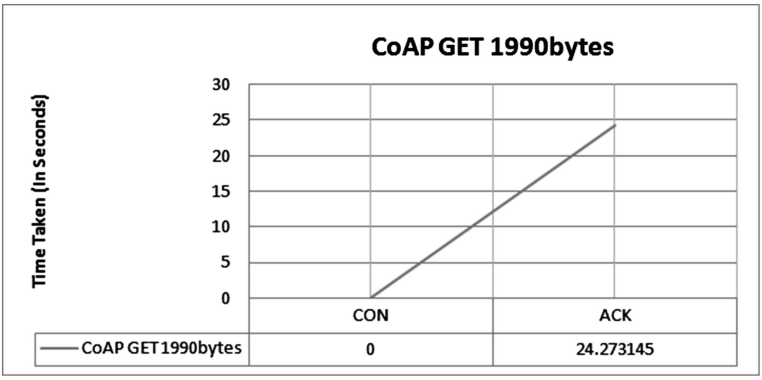
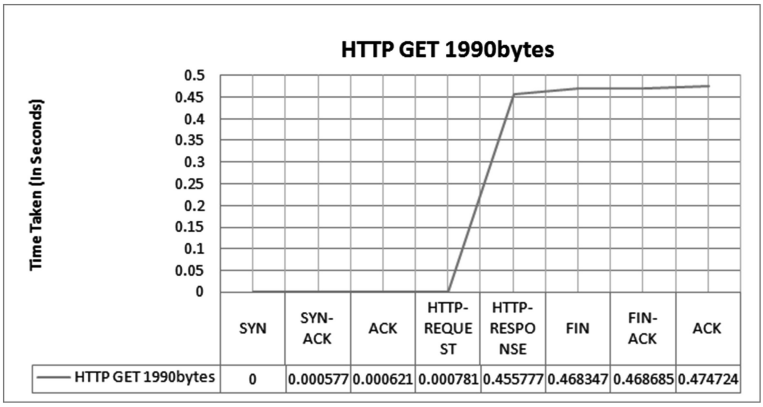


Table 2. HTTP GET large data time taken



In Table 2, it shows the time taken HTTP to complete the GET request. The time taken for HTTP to complete the request is 0.47472 s. The pattern was similar to GET method on smaller data from the different scenarios. The one that contributed to the time taken was HTTP-REQUEST to HTTP-RESPONSE.

From previous analysis, the time taken for CoAP was relatively lower but when dealing with large data; a different set of findings was found. The time taken for CoAP is higher even compared to HTTP previous analysis. This means that large data affect the overall performance in terms of time taken for CoAP and only slight effect on HTTP.

Table 3. CoAP and HTTP overall GET time taken

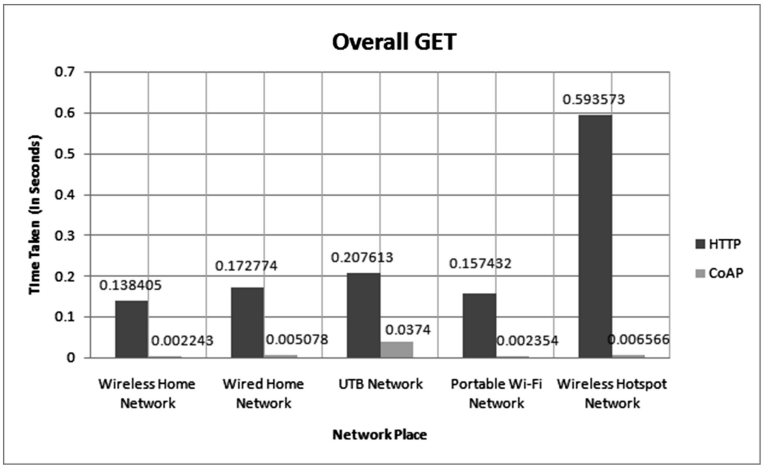
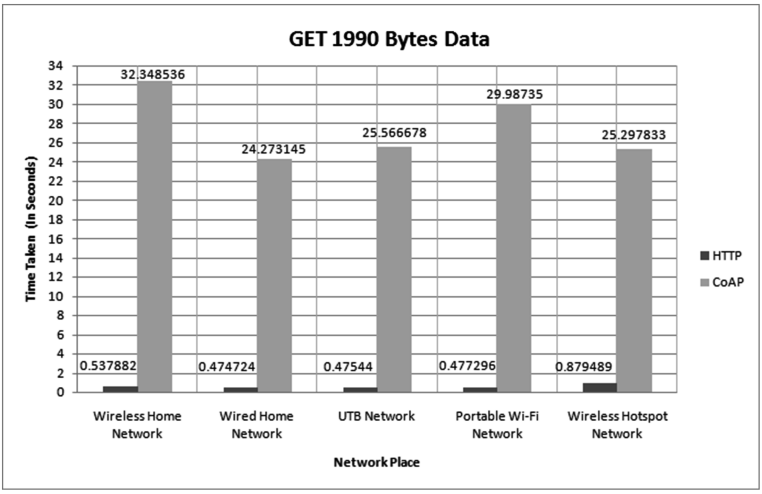


Table 4. CoAP and HTTP overall time taken large data



The Table 3 summarizes the time taken for GET method result for all of the scenarios.

The Table 4 shows the summary for CoAP and HTTP request on GET method for large data 1990 bytes.

The tables above show the huge gap of time taken between CoAP and HTTP for the five different network structures. The time taken to process large data by CoAP creates a large gap in terms of time compared to HTTP. Therefore, time performance for CoAP is affected by large data; this strengthens why CoAP is not suitable to process large data.

POST Large Data on Home Wireless Network. An exception on the test made was discovered when the CoAP request was done for POST and PUT method on large data of 1990 bytes. CoAP was unable to complete both requests. The error encountered was the '4.08 Request Entity Incomplete', indicating that the client was unable to send the request to the server. Data limit for CoAP was 1024 bytes where the large data are 1990 bytes. This shows CoAP client was unable to send data larger than 1024 bytes. As for HTTP, the time taken was slightly higher than the previous analysis for the rest of the methods. This indicates that large data does affect the time taken performance of HTTP.

5 Conclusion

The properties of CoAP as the chosen protocol to use when dealing with IoT have been outlined in this paper. IoT system consists of a group of constrained devices which means that every device is limited in terms of resources such as RAM, memory and processing power. With the properties of CoAP and the idea of IoT system, CoAP is

efficient in delivering smaller data size as the findings suggest. This is further strengthened by comparing CoAP with the known HTTP protocol. The study shows how the light-weight feature of CoAP contribute to the short transmission time but attempt needs to be considered as this compromise the security aspect of it. With HTTP, this is taken care by the handshake before and after as well as the message format. Therefore for future enhancement for CoAP, a study on the implementation of DTLS as part of the security aspect is suggested. One known fact of HTTP is having its own ways of securing the communication. This is done by the use of HTTPS. HTTPS was introduced which is a combination of both HTTP and TLS. This combination provides an encrypted communication which is a part of security measure that has been taken by HTTP. CoAP with DTLS could also provide security to the communication. Similar test could be done to determine whether security features affect the protocol in terms of performance.

References

1. Mario, B., Candid, W.: Insecurity in the Internet of Things. In: Symantec Security Response (2015)
2. Mattern, F., Floerkemeier, C.: From the internet of computers to the internet of things. Informatik-spektrum **33**(2), 107–121 (2010)
3. Villaverde, B., Pesch, D., Alberola, R.D.P., Fedor, S., Boubekeur, M.: Constrained application protocol for low power embedded networks: a survey. In: 2012 Sixth International Conference Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 702–707 (2011)
4. Colitti, W., Steenhaut, K., De Car, N., Buta, B., Dobrota, V.: Evaluation of constrained application protocol for wireless sensor network. In: 2011 18th IEEE Workshop Local Metropolitan Area Networks (LANMAN), pp. 1– 6 (2011)
5. Fielding, R., et al.: Hypertext transfer protocol – HTTP/1.1. The Internet Society (1999). https://www.ietf.org/rfc/rfc2616.txt. Accessed 1 Sep 2016
6. Yannakopoulos, J.: Hypertext Transfer Protocol: A short course. Department of Computer Science, University of Crete, Greece (2003). http://condor.depaul.edu/dmumaugh/readings/handouts/SE435/HTTP/http.pdf. Accessed 1 Sep 2016
7. Chen, X.: Constrained application protocol for internet of things. Washington University, St. Louis (2014). http://www.cse.wustl.edu/∼jain/cse574-14/ftp/coap.pdf. Accessed 1 Sep 2016
8. Ajit A., Mininath K.: Secure CoAP using enhanced DTLS for internet of things. Int. J. Innov. Res. Comput. Commun. Eng. (2014)
9. Bardford, P., Crovella, M.: A performance evaluation of hyper text transfer protocols. In: Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 188–197 (1999)
10. Shelby, Z.: RFC 7252 - the constrained application protocol (CoAP). Internet Engineering Task Force (IETF) (2014). https://tools.ietf.org/html/rfc7252. Accessed 1 Sep 2016
11. Tod, B., Jin, Q.: The TCP handshake: practical effects on modern network equipment. Net. Protoc. Algorithm **2**(1), 197–217 (2010)

12. Fielding, R., Reschke, J.: RFC 7230 - Hypertext Transfer Orotocol (HTTP/1.1): Message Syntax and Routing. Internet Engineering Task Force (IETF) (2014). https://tools.ietf.org/html/rfc7230#section-3.2.4. Accessed 1 Sep 2016
13. Siegel, M., Sciore, E., Madnick, S.: Context interchange in a client-server architecture. J. Softw. Syst. 27(3), 223–232 (1993). http://web.mit.edu/smadnick/www/wp2/1993-07.pdf. Accessed 1 Sep 2016