© 2016 IJSRSET | Volume 2 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Resource Observation in WSN-based Constrained Networks

Farman Ullah*, Muhammad Safdar, Izaz Ahmad Khan, Zia Ur Rahman

Department of Computer Science, Bacha Khan University Charsadda, Khyber Pukhtoon Khwa, Pakistan

ABSTRACT

As we move slowly to a paradigm where a large number of machines will interact with each other without the need for human intervention, we begin to realize that we need a standard way to enable communication between these machines. Additionally, since a lot of good work has already been done in the past to setup the Internet and the IP networks, it makes sense to follow the lead and try to leverage the already existing infrastructure and build on top of it. To standardize a communication methodology for machines to independently exchange information with each other, keeping in mind that we want to build on and leverage existing knowledge and infrastructure. The Internet Engineering Task Force created a working group called Constrained RESTful Environment Group (or CoRE) group. This group was assigned the task to define a mechanism using which a large number of small, resource constrained, low power devices can communicate over low-power lossy networks. This group defined a set of specifications that is known today collectively as – Constrained Application Protocol or CoAP in short. The objective of this paper is to provide an overview of CoAP protocol and observing resource, similar to http, in the Internet of things (IoT) and wireless sensor networks (WSNs).

Keywords : M2M, RESTful Environment , CoAP , Internet of things, HTTP, JSON, XML

I. INTRODUCTION

Constrained Application Protocol is a protocol at the application level that is designed to allow message exchange between resource-constrained devices over resource constrained networks such as WSN and IoT [1, 2, 3]. Resource constrained devices are small devices that lack the processing power, memory footprint and speed that we generally expect from our computing devices. These devices often are built using 8-bit microcontrollers or low-cost, general purpose 32-bit microcontrollers. Resource constrained networks are network stacks and configurations that do not have the full capabilities of TCP/IP stack and have lower transfer rates. CoAP runs over UDP and not TCP. 6LoWPAN is an example of such a constrained network configuration setup. CoAP provides an HTTP-like request and response paradigm where devices can interact by sending a request and receiving a response. Like the web, devices are addressed using IP address and port number. Access to services exposed by the device is via RESTful

URIs. It’s very much similar to HTTP, where method type (e.g. GET, PUT), response codes (e.g. 404, 500) and content-type are used to convey information. Given the protocol’s close similarity to HTTP, it’s obvious that it was designed for easy web integration.

CoAP does not replace HTTP, instead, it implements a small subset of widely accepted and implemented HTTP practices and optimizes them for M2M message exchange. Think of CoAP as a method to access and invoke RESTful services exposed by “Things” over a network. As an example, let’s consider our temperature sensor installed in the conference room of our office as shown in figure below. The temperature sensor works like a “server” (Thing) which any CoAP based client (another Thing) can query to get the temperature.

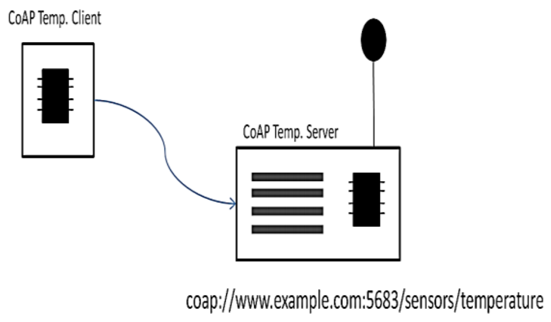

Figure 1. CoAP Client and Server

In the figure above, the server exposes the interface to query the temperature as a RESTful URL with the path as “sensors/temperature”. By this time, you would have noticed the new scheme – coap. Instead of using http as the scheme, a new scheme called “coap” is introduced. There is also a secure version, just like https, you can use coaps. In the figure above, the full provides the scheme name, the DNS name, the port number and the path. Remember, the default port suggested for CoAP is 5683. Also remember, that the communication will use UDP and not TCP. Therefore, the client needs to establish a UDP connection with the server, send a GET request to the server over the given URL path and get a response. Just like HTTP response, you can get a response in various formats (remember HTTP content-type?). The specification allows for various content formats, notable amongst them is JSON, XML and plain text. In the next section, we present the request/response interaction model for CoAP which is somewhat similar to HTTP; however, it uses extremely lightweight options, headers, metadata and tokens [4, 5].

II. METHODS AND MATERIAL

CoAP Request/Response Interaction Model

As stated before, CoAP is similar to HTTP. One party can send a request to the remote party [6, 7, 8], and the remote party may respond back. There are four kinds of message types defined by the specification:

1. CON – This represents a 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. The retransmissions use a non-linear, exponential strategy between attempts.
2. NON – This represents a 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 by the receiver, i.e., a server.

3. ACK – This represents an acknowledgement. It is sent to acknowledge a confirmable (CON) message.
4. RST – This represents a negative acknowledgement and means "Reset". It generally indicates, some kind of failure (like unable to parse received data)

A typical confirmable message exchange could look like as shown in Figure 2

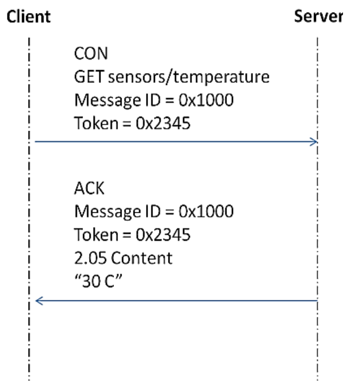

Figure 2. CON Message Exchange

The client sends a CON message to the server. The method type is GET, the path URL is "sensors/temperature". The message ID is a 16-bit number used to uniquely identify a message and help the server in duplicate detection. Token is used to correlate messages. We will soon see an example of message correlation. Once the server gets the message, it measures the temperature and returns an acknowledgement. The acknowledgement contains the same message ID and the token that was received in the request [9, 10, 11]. Along with the acknowledgement, the message also contains the temperature data (in the above figure, its 30 C). Sending response data, along with the acknowledgement is also called "piggy-backed response". Finally, the response also has a message code, in this case it's "2.05 Content". These are very similar to HTTP status codes (there is a 4.04 message code to indicate not found, like the HTTP 404 not found status code).

The previous example indicates a success, but sometimes CON message might result in failure. For example, if the path URL is incorrect, there is no way the server can serve the request. In HTTP world, if the URL is incorrect, we get a **404** as response code. In the same manner, in CoAP also, we get a "Not Found" response. Figure 3 indicates such a situation.

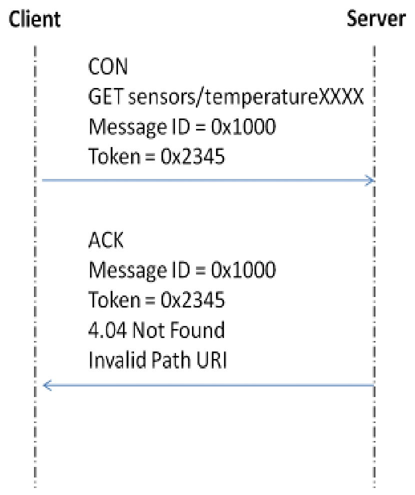

Figure 3. CON Message Exchange Failure

In the figure 3, the request is made to an unrecognized path URL"sensors/temperature XXXX". Since there is no way server can handle that path, it acknowledges the receipt of the message, however, it sets the message code as **4.04** which means “not found”. Additionally, implementations may add diagnostic message in the response payload, and in this example, the string “Invalid Path URL” was added as the diagnostic payload. Sometimes, the request is not mission critical and it’s acceptable if some values are never received. Classic example is temperature sensing request for room cooling that continues 24 x 7. Even if some queries to the temperature server are lost there would hardly be an issue. In those cases, NON (or Non-Confirmable) requests are used. Figure 4 depicts such a scenario.

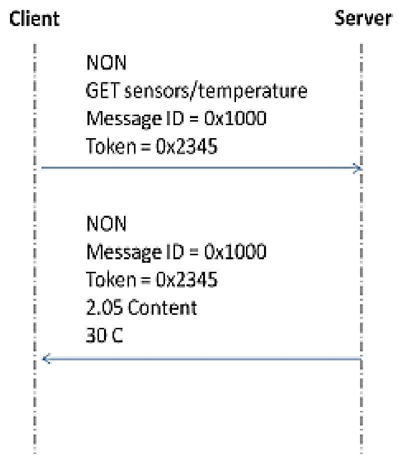

Figure 4. NON Message Exchange

The client sends a NON based GET request. The server also responds back with a NON message. In either direction, the message may get lost, but unlike a CON message, no attempt will be made to retransmit the lost message. A new NON message will simply be sent by the client when it’s due. Natural question to ask in this case would be what if NON message cannot be understood by the server (for example, the URL path is incorrect), will the server respond back? The specification states that a NON message, being non-confirmable, must not be acknowledged by the recipient. The recipient, at best, may send a RST (reset) message and must silently ignore. Therefore, if a NON message carries incorrect path, there is no guarantee that the sender will be informed of the error [12, 13, 14]. The sender may choose to re-transmit the message up to a limit, but the sender cannot expect a guaranteed response.

III. CONCLUSION

CoAP is an extremely lightweight protocol for resource observation in Internet of Things. It is a lightweight version of HTTP, however, it cannot be used as an alternative to HTTP. In this paper, we presented various types of scenarios for resource observation using CoAP protocol. Also, different messages, i.e., CON, NON, ACK and RST are studied and analyzed. Furthermore, we gave an inside to the situation where we can use such messages.

IV. REFERENCES

[1] Z. Shelby, K. Hartke, C. Bormann, "The constrained application protocol (CoAP)", 2014.

[2] M. A. Jan, P. Nanda, X. He, Z. Tan and R. P. Liu, "A robust authentication scheme for observing resources in the internet of things environment" in 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 205-211, 2014, IEEE.

[3] M. A. Jan, P. Nanda and X. He, "Energy Evaluation Model for an Improved Centralized Clustering Hierarchical Algorithm in WSN," in Wired/Wireless Internet Communication, Lecture Notes in Computer Science, pp. 154–167, Springer, Berlin, Germany, 2013.

[4] K.Hartke, "Observing resources in coap", 2014.

[5] O. Bergmann, K. T. Hillmann, S. Gerdes, A Coap-gateway for smart homes, in: 2012 International Conference on Computing,

Networking and Communications (ICNC), 2012, pp. 446–450.

[6] M. Castro, A. J. Jara, A. F. Skarm eta, "Enabling end-to-end coap-based communications for the web of things", Journal of Network and Computer Applications (2014).

[7] M. A. Jan, P. Nanda, X. He and R. P. Liu, "Enhancing lifetime and quality of data in cluster-based hierarchical routing protocol for wireless sensor network", 2013 IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC & EUC), pp. 1400-1407, 2013.

[8] M. A. Jan, P. Nanda, X. He and R. P. Liu, "PASCCC: Priority-based application-specific congestion control clustering protocol" Computer Networks, Vol. 74, PP-92-102, 2014.

[9] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer networks 54 (2010) 2787–28 05.

[10] Mian Ahmad Jan and Muhammad Khan, "A Survey of Cluster-based Hierarchical Routing Protocols", in IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC), Vol.3, April. 2013, pp.138-143.

[11] Mian Ahmad Jan and Muhammad Khan, "Denial of Service Attacks and Their Countermeasures in WSN", in IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC), Vol.3, April. 2013.

[12] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future dire ctions, Future Generation Computer Systems 29 (2013) 1645–1660

[13] M. A. Jan, P. Nanda, X. He and R. P. Liu, "A Sybil Attack Detection Scheme for a Centralized Clustering-based Hierarchical Network" in Trustcom/BigDataSE/ISPA, Vol.1, PP-318-325, 2015, IEEE.

[14] M. A. Jan, "Energy-efficient routing and secure communication in wireless sensor networks," Ph.D. dissertation, 2016.