Lightweight Internet Protocols for Web Enablement of Sensors using Constrained Gateway Devices

Soma Bandyopadhyay
Innovation Lab
Tata Consultancy Services
Kolkata, India
Soma.bandyopadhyay@tcs.com

Abhijan Bhattacharyya
Innovation Lab
Tata Consultancy Services
Kolkata, India
abhijan.bhattacharyya@tcs.com

***Abstract*— Lightweight Internet protocols are increasingly being used in ubiquitous environment in order to optimize the resource usage of a constrained device like a smart mobile gateway. This paper presents a study on the various such protocols to optimize the usage of energy, and network resources, computation cost of a constrained gateway device. Comprehensive analysis and feature wise categorization of existing dominant protocols, namely MQTT (message queue telemetry transport), CoAP (constrained application protocol) are provided here to achieve improved understanding of the existing issues and gaps in this domain. The present work further identifies the best suited application areas for each protocol based on the results corresponding to typical resource requirements and performance attributes. Finally our vision on the research scope is presented here mainly for optimization of energy usage.**

Keywords-constrained device; ubiquitous computing; protocol;lightweight

I. INTRODUCTION

The Internet of Things (IoT) consisting of ubiquitous smart devices and objects with embedded sensors requires energy and bandwidth efficient interfacing of resource-constrained devices with other devices, backend applications, servers and other cloud hosted systems. Interoperation with a wide variety of sensors and smart devices, semantic interoperability, collection and analysis of sensor data, context detection, predictive analytics and generation of control and actuation outputs are some of the important aspects of IoT applications. IoT applications require access to sensor data in real-time as per their needs. The gateway device is one of the most important components required for realization of the IoT. It connects the sensor / device domain to backend application/web service domain. In this paper we consider those gateways that are low cost battery powered embedded systems and have limited processing power, memory and storage. Fig. 1 depicts the generic system overview.

The gateway dynamically aggregates the real time sensor data and shares the sensor data with the destination server/application over Internet and at the same time manages and controls the various diverse sensing-devices/sensors. An example of a resource constrained gateway is the smart mobile phone based gateway that interfaces sensors embedded in the smart phone such as GPS, accelerometer, microphone, etc. with external Internet facing system. Another example is a mobile healthcare-gateway which aggregates the various sensor data such as blood pressure, blood sugar, ECG, etc. from portable sensors. Sensor data is aggregated, filtered, pre-processed, analyzed and then posted to the backend applications in the cloud. In resource constrained gateway energy consumption, CPU cycles, memory usage, networking resources and bandwidth usage must be used as efficiently as possible.

In this paper, we study existing lightweight application protocols like CoAP (Constrained Application Protocol) and MQTT (Message Queue Telemetry Transport). They are compared considering various features such as energy consumption, bandwidth utilization, reliability, etc. Based on that analysis, these protocols are mapped to the best suited application areas. As CoAP supports both request-response and resource-observe (a variant of pub-sub), this paper illustrates how the adaptive feature of CoAP for request-response and publish-subscribe mode, can be leveraged with a common interface using sequence diagram and sate machine. Further, existing issues and gaps, and future research scopes are analyzed and discussed based on the above review.

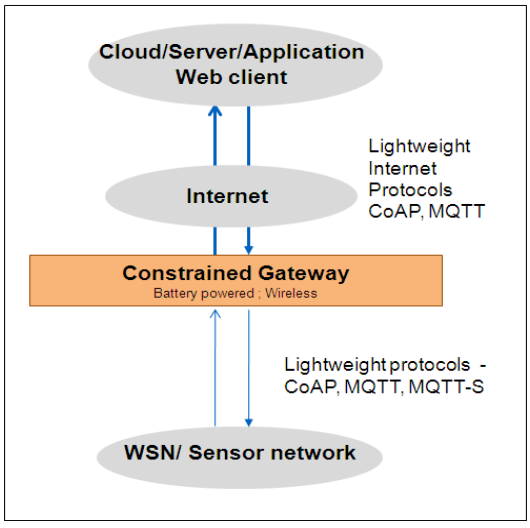


Figure 1. A generic architecture for connecting Sensor to the Web.

978-1-4673-5288-8/13/$31.00 ©2013 IEEE

The remainder of this paper is organized as follows. First, the state-of-the art in the area of lightweight Internet protocols is presented, followed by their feature-wise comparison based on the underlying transport protocol, multicast capability, reliability, etc. The next section discusses how CoAP, originally envisaged as HTTP-like request-response protocol, can work in publish-subscribe mode, followed by a discussion on the architecture to create an adaptive solution for both the request-response and publish-subscribe using common CoAP methods. Next, experimental results are presented to compare the resource requirements of CoAP and MQTT in terms of bandwidth, and energy. The final section concludes this article with future research scope, and analyzes the gaps of the existing light weight Internet protocols.

II. RELATED WORK

There has been a lot of work in the WSN domain in order to connect the sensor data to the Web to enable a ubiquitous connectivity between the Web and the sensor. Standard protocols have been designed keeping the constrained nature of the devices associated with WSN. There are two faces of the problem. Firstly, the sensors themselves are constrained in terms of resources and they are connected to the Web directly or through a gateway. So the connection between the sensor to the Web or the G/W is a concern. Here the G/W is not a resource constrained one. The other aspect of the problem unfolds when the G/W itself is a constrained device like a smart mobile phone or TAB, etc., where large numbers of sensor devices communicate through the smart mobile phone/ constrained gateway. In order to accomplish low power RF communication many sensors have adapted the ZigBee specification for low power local wireless networking. Use of IPv6 over Low Power Wireless Area Networking has given rise to the 6LowPAN standard at the link and network layer [1]. 6LowPAN enables the wireless connectivity between sensors using 802.15.4 and IPv6 together in a simple well understood way.

In order to support the constrained applications two dominant application layer protocols have been proposed to connect the sensors to the Web, MQTT from IBM [2] and Constrained Object Application Protocol (CoAP) [3] from IETF. CoAP is based on request-response model. Ref. [4] demonstrates how CoAP over UDP can be a better choice over other options like HTTP over TCP or HTTP over UDP considering the power consumption, header overhead and the basic reliability feature supported by CoAP at the application layer itself. The MQTT protocol from IBM uses hierarchical topic based [11] 'publish-subscribe' mechanism and facilitates the constrained devices by enabling 'PUSHing' [5] data from the cloud rather than polling by constrained device for the data from the server. However, in terms of communicating to the end sensors in the WSN i.e. in the sensor domain, IBM has come up with yet another protocol MQTT for Sensors (MQTT-S) [6] which is designed in such a way that the protocol is agnostic of the underlying networking services.

As can be seen in many available car, medical solutions (ex. [7], [8]), the sensor domain to G/W communication many a times uses proprietary and non-standard mechanism and that part of the chain is hidden from the developer. The open part is the communication mechanism over the link between the Constrained G/W and the applications situated in the Cloud. So, the sensor specific link-layer and network layer protocols like 6LowPAN is not considered here. Also, application protocols like MQTT-S is not relevant as MQTT-S is designed mainly for communication between the sensors and the G/W i.e. for the sensor domain.

Therefore, the study boils down to the two dominant application layer protocols CoAP and MQTT. Ref. [9] describes the use of CoAP at the G/W with a proxy to convert CoAP message to HTTP and vice-versa in order to interpret with the common language of the Web, HTTP. It also compares HTTP and CoAP in terms of power consumption.

However, there is no survey found which puts CoAP and MQTT side by side and compares them feature wise and finds the suitable areas of applications for each. The present work brings out the features of both the protocols and performs analysis based on the experimental results obtained by simulating network conditions using publish subscribe mode of both the protocols. The current article also discusses about the adaptive architecture of CoAP for two different architectures request-response and resource-observe.

III. FEATURES OF THE PROTOCOLS

This section describes the features of the dominant light weight Internet protocols as discussed in the previous section and furnishes a feature wise comparison chart.

CoAP is a Web transfer protocol based on the REST (Representational State Transfer) [10] architecture. It is very similar to HTTP. However, in order to avoid the overhead in TCP like connection oriented protocols, CoAP is originally designed to be used on top of UDP with provisions for a reduced set of reliability mechanism like TCP. The optional reliability is supported by a logical messaging layer on top of UDP. Logically, CoAP can be considered as a single protocol suit of two logical layers: request/response layer on the top and messaging layer at the bottom to interact with the UDP. It supports 4 message categories: confirmable, non-confirmable, acknowledgement and reset message. The confirmable type messages are used for reliable communication. The confirmable message is accompanied by an acknowledgement from the receiving node to provide reliability.

Unlike HTTP, CoAP supports asynchronous message exchange. CoAP has a low header overhead and parsing complexity than HTTP. Although CoAP is originally designed for UDP, it can be used for TCP as well [3].

The basic CoAP proposal has been extended to support resource-observe mode (which is similar to publish-subscribe) in order to support features like supporting real time updates from sensors as proposed in [16] and explained in Section IV.

MQTT, as mentioned earlier, is absolutely based on topic based publish-subscribe architecture with a message-broker to bridge between the publishers and subscribers. It exploits the decoupling in space, time and flow as the core of its

working philosophy. It supports three QoS (quality of service) levels for message delivery with enhanced reliability and has a low header overhead [2]. MQTT is a connection oriented protocol as the publisher or the subscriber both needs persistent connection to the message broker. It relies on the underlying TCP layer for connection features.

Fig. 2 presents the mapping of these protocols in the OSI stack.

Table I provides the feature wise comparison of these protocols along with the potential application areas for CoAP and MQTT.

Since CoAP uses a HTTP like request-response architecture and resembles HTTP in many ways CoAP can be well used for HTTP like GET, PUT, POST, DELETE request based applications. Thus it can be used as the protocol behind browsing like applications on a constrained server. Potential applications may well use a proxy to convert CoAP requests to HTTP, the language the Web is more conversant with traditionally, as exemplified in [9].

MQTT, by virtue of its inherent publish-subscribe architecture, is very useful for 'PUSH' like application so as to free the constrained devices from resource hungry operations like 'polling' to get the updated data [5]. However, in case of constrained edge gateways this protocol can be used for 'publishing' real time aggregated sensor data to the web-server asynchronously and with reliability. Efforts have been made to integrate MQTT [12] with 'Pachube', a widely popular technology for monitoring sensors and devices using 'polling', such that 'Pachube' can take advantage of the MQTT broker by configuring triggers and can be implemented using HTTP POST.

However, CoAP seems to have a wider acceptance for constrained devices due to ease of integration with 6LowPAN, easy portability with HTTP, UDP based operation with low connection overhead, support for several features like sleeping nodes [3] (without requiring persistent connection), support for both request-response and resource-observe mode, etc. Also, apart from the IETF, CoAP is backed up by ETSI [22] and is poised for wide industry adaptation as evident from the participation list in the recent ETSI plug test of CoAP [18].

IV. CoAP AS A PUBLISH/SUBSCRIBE PROTOCOL

The request/response model does not work well for clients which are interested in having a current representation of a resource/ sensor over a period of time. Conventional method like polling also proves inefficient in terms of resource usage for constrained devices. Thus, in scenarios where real-time updates from the constrained clients are required, MQTT may seem as an acceptable solution. However, realizing the need for such real-time feature of a proposed protocol, IETF extends the CoAP core protocol with a mechanism to push resource representations from servers to interested clients, while still keeping the properties of REST [16].

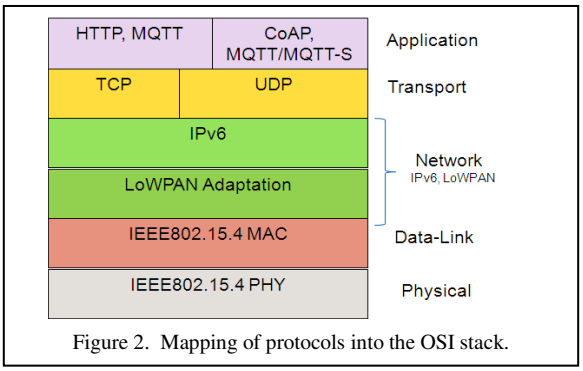


Figure 2. Mapping of protocols into the OSI stack.

In this mode of operation CoAP clients called 'observers' register their interests about a specific resource in the namespace of a known server. The observers are notified whenever the subject undergoes a change. The observers register with the subject using the GET request with a special 'observe' option activated. The subject puts the observer in the list of observers if it is allowed and responds to the observer with an immediate state of the resource. After the initial response each subsequent notification is an additional CoAP response sent by the server in reply to the GET request and includes a complete representation of the new resource state. Fig. 3 depicts an example scenario with 3 notifications.

The resource observing model can be easily extended to publish/subscribe data exchange with a publisher client, subscriber clients and a broker in between utilizing PUT and 'observable' GET request from two independent clients. Fig. 4 depicts the handshake in this case.

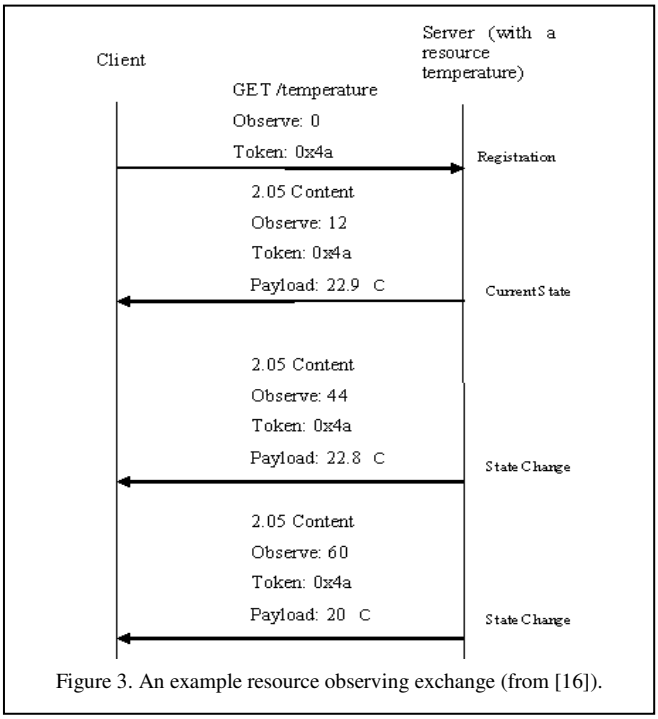


Figure 3. An example resource observing exchange (from [16]).

TABLE I.

	Transport layer	*Supported architecture*	*Payload & header structure*	*Response timeout*	*Max re-transmit*	*Multicast support*	*Reliability*	*Typical application area*
MQTT	Mainly TCP [2].	Publish-subscribe	2 bytes header. Payload treated as binary large object.	Configurable. Timeout value increases across multiple retries	Configurable	One-to-many message distribution is inherently supported through pub-sub.	With 3 QoS levels	Topic based real-time messaging kind of application using pub/sub requiring persistent connection with the server. Message broker is responsible to weave the sensors with the rest of the Web. Ex. [19].
CoAP	Mainly UDP, TCP can be used but not usually [3].	REST based request / response, Resource observe (Publish-subscribe)	4 byte header +Type Length Value (TLV). Supports payload encoding like XML	Default 2 seconds [3]. Can be configured.	4[3]. 0 for multicast	'Resource observe' mode can be used for one-to-many messaging. Dedicated multicast support exists but with no retransmission.	Simple stop-and-wait retransmission reliability with exponential back-off	Applications with sensors (possibly with defined sleeping cycle) requiring to run RESTful web-services to achieve direct connectivity to the Web (or optionally to some G/W) with HTTP like methods and URI [21] using either or both request/response or observing resource (variant of real-time pub/sub) mode. Ideal for applications requiring easy integration with the HTTP based Web. Suitable for applications like smart energy, building automation, etc. [3].

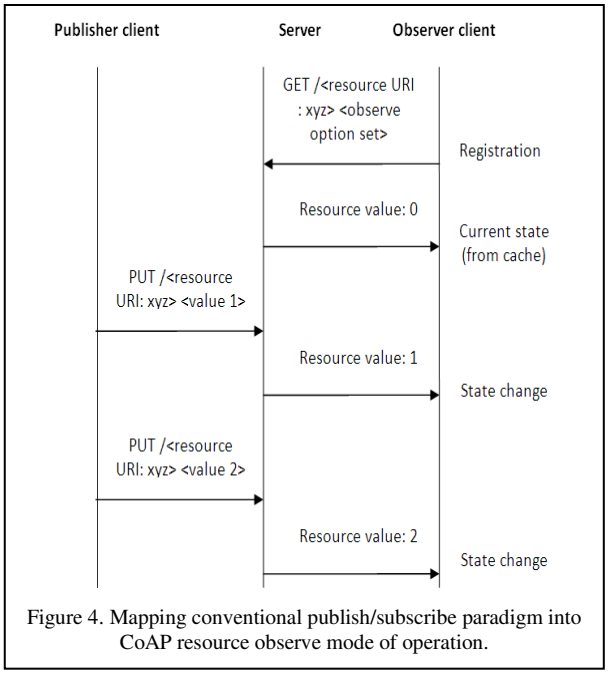


Figure 4. Mapping conventional publish/subscribe paradigm into CoAP resource observe mode of operation.

V. CoAP FOR AN ADAPTIVE ARCHITECTURE

The discussion in the above section brings out an advantage of CoAP. The configuration of the GET request determines whether a client-server pair would interact in request-response or in resource-observe mode. Thus, without making any changes in the infrastructure, the same system can act both as a request/response and publish/subscribe depending on how a client has configured the GET call to the server. Thus the same architecture can be fitted into different use cases for either publish/subscribe or request/response. However, the server in the given mode of interaction should have the logic of resource-observe built into it so that it understands the 'observe' option in the GET request from the client.

Fig. 5 shows how, with a single method, one can support such an adaptive architecture. Fig. 5(a) shows the functional flow on the server side. It shows how a GET request can be served both for resource observe and request-response mode depending on the option in the GET method and the accessibility of the client with the server.

Fig. 5 (b) shows the state diagram for the client willing to observe a resource on the server. Every resource has a 'Max_age' defined for the notified value [16]. If the client finds a notification from the server which has an age greater than the defined 'Max_age' then it understands that the server is no longer keeping the client in the list of observers and the client falls back into idle mode. The client can make another observe request with a fresh token. Again, if the client has to initiate a RST message then also it falls back into non-observing mode.

VI. EXPERIMENTAL ANALYSIS AND RESULTS

Current section performs an analysis of resource requirements for CoAP and MQTT protocols based on experimental results.

The experiments were conducted using Intel X86 systems with Ubuntu installation as the nodes. Smart phone with

Android O/S is planned as our future research platform. Fig. 6(a) depicts the experimental set up.

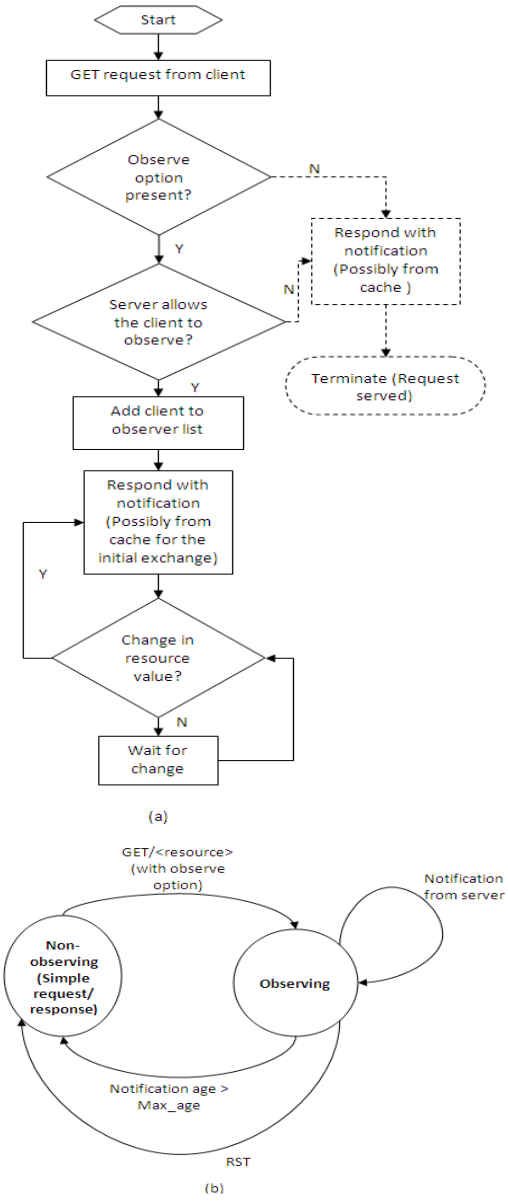


Figure 5. a) Server side flow showing how a GET request can be served both as request response (the dotted lines) and resource-observe mode. b) The state m/c for the client for an 'observe GET' request.

The connection between the nodes was routed through an intermediate router known as Wide Area Network Emulator (WANEM) [17] in order to control the network attributes for the experiment. WANEM allows controlling the physical network properties like bandwidth, packet loss, etc. to simulate constrained network, as shown in Fig. 6(b). Wireshark [20] tool was used to analyze the network traffic. We used 'libcoap' [12] and 'mosquitto' [13] as the implementation for CoAP and MQTT respectively. It is worth mentioning here that the 'libcoap' distribution supports IPv6. However, the modems used in the experiment support only IPv4. So, the 'libcoap' code was changed to support IPv4. This brings parity with the 'mosquitto' implementation which used IPv4 at the network layer.

Different sample sensor data of constant size were used as payload for both the protocols. Performance of the protocols were studied by changing the packet loss condition for a given payload size.

Fig. 7 and 9 shows the comparison between MQTT and CoAP based on the actual network data transferred for transferring different size of sensor payload with 0% packet loss with CoAP in request-response mode and resource-observe mode respectively. Fig. 8 and 10 shows the comparison for 20% packet loss using CoAP in request-response and resource-observe mode respectively.

It is obvious from the graphs that MQTT consumes higher bandwidth than CoAP for transferring same payload under same network condition. The MQTT traffic was captured while MQTT publisher publishes data to the broker. Since MQTT runs on TCP, it has TCP connection overhead for connection establishment and closing. The publisher client establishes a connection with the broker first, followed by a 'PUBLISH' from the publisher. In case of QoS 0 the publisher expects no more application layer feedback from the broker and the TCP connection closes with normal transport layer exchange of ACKs. But, in case of QoS 2 the publisher performs additional exchanges with the broker to ensure 'exactly once' delivery. This causes more traffic than QoS 0 case.

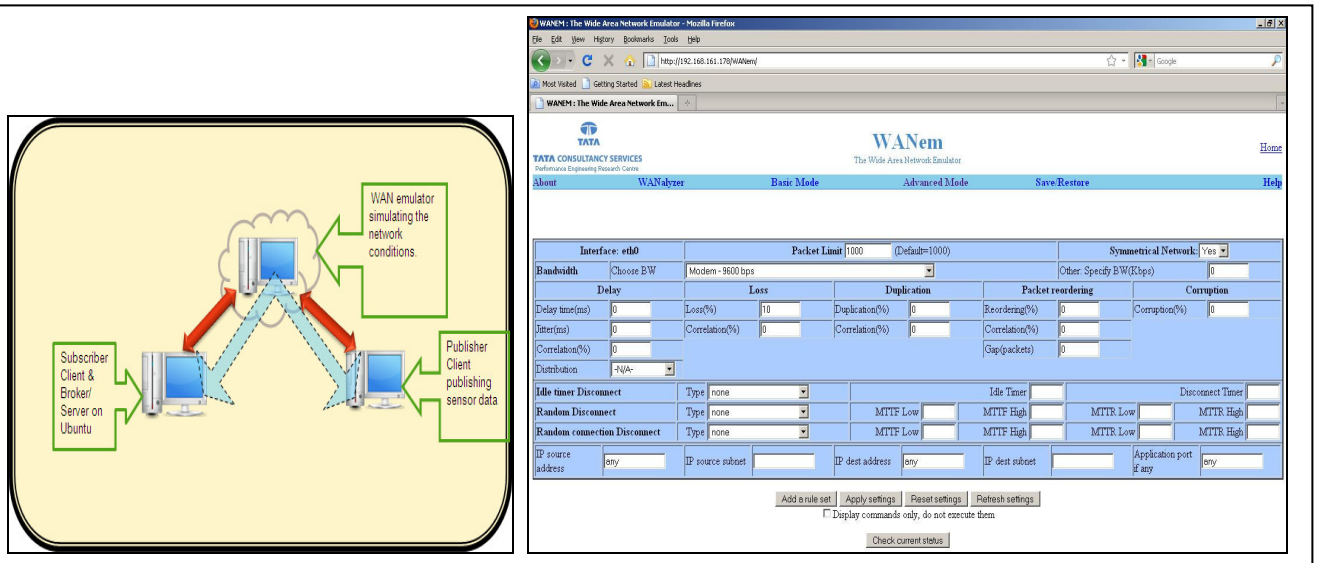
On the other hand, CoAP has no connection overhead as UDP works in 'fire and forget' basis. To ensure reliable transmission, the CoAP messages were CON (confirmable) messages with an application layer ACK message. But since no connection was established there is no question of overhead to close the connection. It has been observed that as packet loss increases, total number bytes transferred also increases because of retransmission.

It is important to note that in case of MQTT for packet loss scenarios like Fig 8. and Fig. 10, when the actual bytes sent is increased, the total bytes of data transferred dropped because of an early connection closure during TCP connection establishment or publisher broker handshaking phase.

The data overhead analysis exposes the fact that topic definition is proving to be an extra overhead in case of MQTT as the topic string itself is part of the payload. However, efficient formation of topic string using wild cards [11] may reduce the topic overhead for MQTT. On the other hand, URI for locating resources forms a significant part of the CoAP packet [3] and an optimized URI path is needed for reducing overhead.

In terms of energy both the protocols are efficient. Detail energy consumption figures for MQTT are available in [15]

for Android implementation. Fig. 11 shows the average energy consumed by MQTT and CoAP with reliability based on our experimental data, where CoAP becomes most energy efficient.



(a) (b)

Figure 6. (a) The experimental set up using a private network for performing CoAP and MQTTperformance comparison. (b) Simulating the different network conditions.

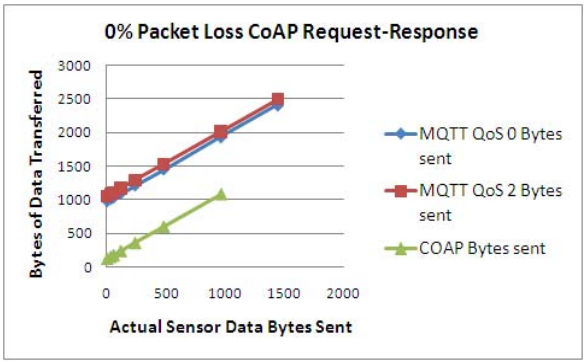


Figure 7. Data bytes sent for different sensor data posting with 0% packet loss with CoAP in request-response

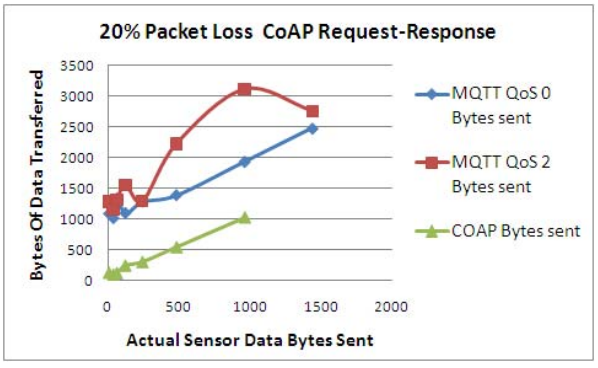


Figure 8. Data bytes sent for different sensor data posting with 20% packet loss with CoAP in request-response

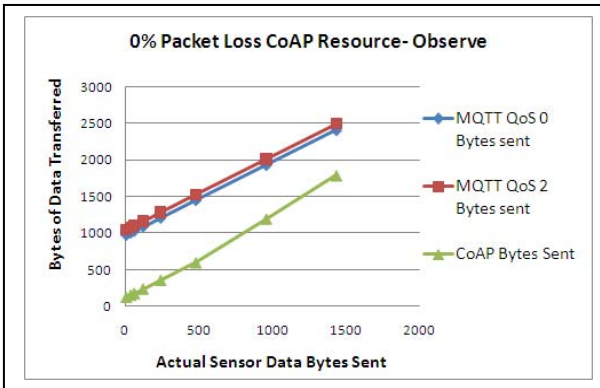


Figure 9. Data bytes sent for different sensor data posting with 0% packet loss with CoAP in resource-observe

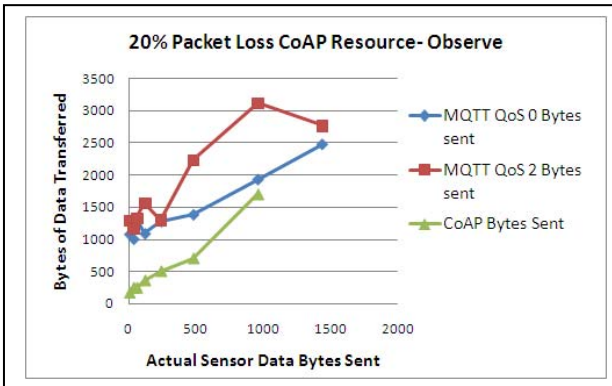


Figure 10. Data bytes sent for different sensor data posting with 20% packet loss with CoAP in resource-observe

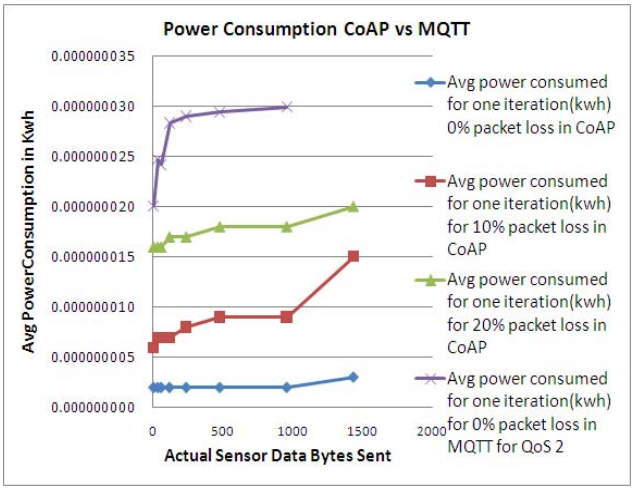


Figure 11. Average Power Consumption with CoAP in resource-observe.

VII. CONCLUSION

In this article a survey of lightweight Internet protocols is presented considering the scenario in which protocols are employed by constrained gateway devices for sharing dynamic sensor data to the applications. They are compared based on their different protocol specific characteristics as well as usage of resources like bandwidth, energy. This comparison is further supported by the results obtained and the analysis on the experiment where sensor data is transferred with different size in increasing order to the same destination over the same communication link. Link characteristics are simulated by varying network conditions. The study represents mapping of these protocols to the different application domain based on the analysis of the characteristics of these protocols. The analysis of the experimental results depicts that CoAP is most efficient in terms of energy consumption as well as bandwidth. Architecture has been discussed for adapting CoAP into two different modes, request-response as well as resource-observe.

Constrained gateway may overcome the limitation in processing speed as per the current trend, but managing energy in an efficient way is still an open issue. There is a scope for future research work to design the protocols including their message handling, state machine design, supporting both request-response and publish subscribe mode and switching between them intelligently to handle the above mentioned issue. The state machine of these protocols should handle the retransmissions and its association with over all QoS in a more efficient way, where UDP is considered as basic transport layer protocol. The protocols possess the low overhead for header parsing; however optimized encoding for payload compression is a further challenge to be resolved. Preservation of privacy while carrying the sensor data including the context information is also another open area to be addressed. Currently we are pursuing our research activity on the above mentioned open issues.

REFERENCES

[1] IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs):Overview, Assumptions, Problem Statement, and Goals, RFC 4919, IETF.

[2] MQTT V3.1 Protocol Specification (http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html).

[3] Constrained Application Protocol (CoAP) draft-ietf-core-coap-12 (http://tools.ietf.org/html/draft-ietf-core-coap-12).

[4] K. Kuladinithi, O. Bergmann, T. Pötsch, M. Becker and C. Görg, "Implementation of CoAP and its Application in Transport Logistics," in In Proc. of 'Extending the Internet to Low power and Lossy Networks' (IP+SN 2011), Chicago, USA, Apr. 11, 2011.

[5] http://dalelane.co.uk/blog/?p=938.

[6] MQTT For Sensor Networks (MQTT-S) Protocol Specification Version 1.2 (http://mqtt.org/MQTT-S_spec_v1.2.pdf).

[7] http://www.mobile-devices.fr/our-products/c4evo/.

[8] http://www.etcomm.cn/en/products_hc-001.html.

[9] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating Wireless Sensor Networks with the Web," in Extending the Internet to Low powerand Lossy Networks (IP+SN 2011), 2011.

[10] http://tools.ietf.org/pdf/draft-griffin-bliss-rest-00.pdf

[11] http://www.ibm.com/developerworks/lotus/library/expeditor-mqtt/.

[12] http://www.ibm.com/developerworks/websphere/library/techarticles/1106_maynard/1106_maynard.html?ca=drs-.

[13] http://libcoap.sourceforge.net.

[14] http://mosquitto.org/download/.

[15] http://stephendnicholas.com/archives/219.

[16] Observing Resources in CoAP draft-ietf-core-observe-05.

[17] http://wanem.sourceforge.net/

[18] Sebastian Müller, "Preliminary results of 1st CoAP Plugtest". http://www.intelligentcontainer.com/uploads/media/Preliminary_Results_1st_CoAP__Plugtest.pdf .

[19] http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger

[20] http://www.wireshark.org/

[21] http://tools.ietf.org/pdf/draft-castellani-core-http-mapping-05.pdf

[22] http://www.etsi.org/plugtests/coap/coap.htm