An Empirical Study of Application Layer Protocols for IoT

Upendra Tandale, Dr. Bashirahamad Momin
Dept. of Computer Science and Engineering
Walchand College of Engineering
Sangli, India
Email: upendra.tandale@gmail.com, bfmomin@yahoo.com

Deva P. Seetharam
DataGlen Technologies, Pvt Ltd
Bangalore, India
Email: dpseetharam@dataglen.com

***Abstract*— IoT has enormous potential of growth. The application layer protocols used are key drivers of internet traffic. Appropriately chosen protocols can reduce network traffic, improve reliability. Here in this paper are discussing performance of three application layer protocols, Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT) and REST (Representational State Transport). These protocols are implemented on arm based device Raspberry Pi3 which as as Gateway. Performance is measured experimentally and compared in terms of bandwidth consumed and time taken. Two different networks are used: cellular 4G and High speed broadband connection. Results show that CoAP is best performer for small payloads and bad performer for bigger payloads.**

Keywords—MQTT, CoAP, REST, IoT, application layer protocols, performance, comparison

I. INTRODUCTION

The rapid evolution of the low cost mobile internet, compact, low cost and high performance hardware manufacturing, and high speed machine to machine (M2M) communication has enabled the enormous growth IoT technologies. A lot of Internet traffic is going to be driven by Internet of Things devices. CISCO predicts that Machine 2 Machine (M2M) traffic is going to increase to 12 billion by 2020 from 5 billion in 2015[1]. This increased traffic puts load on existing network and hardware infrastructure. Appropriately chosen protocol can easy burden on existing network and computing infrastructure. For example CoAP is specifically developed for constrained devices such as microcontrollers with less than 1 kb RAM. This comparison can help in choosing right protocol for the application requirement based on payload size and bandwidth usage.

In the TCP/IP protocol stack application layer sits on top of Transport layer. This layer provides services to application program to access network resources. Word Wide Web uses the most famous application layer protocol HyperText Transfer Protocol (HTTP). REST architecture is used to optimize the HTTP protocol. But looking at the needs of IoT such as low processing devices, little memory and power constrains this protocol is not optimized[7]. Two new protocols gained popularity which are designed specifically to cater needs of IoT which are MQTT and CoAP. MQTT was designed by IBM and CoAP by Internet Engineering Task Force(IETF). MQTT is optimized messaging application protocol designed for pub/sub based systems, CoAP is designed for constrained devices with low processing and memory devices.

Here in this paper we are providing preliminary comparison between CoAP, MQTT and REST. In this paper we will start with short description of protocols, and then the experimental setup for comparison and finally the analysis and result will be discussed.

II. PROTOCOLS

A. CoAP

Constrained application protocol (CoAP) was designed by using a subset of HTTP methods. It uses UDP instead of TCP to keep things lightweight. This protocol has been specially made to be used on constrained devices having low processing power and little RAM[8]. It follows request-response model for interaction between client and server. CoAP fulfills most of the requirements required for M2M communication such as simplicity, small header packet, and multicast. It also provides asynchronous message transfers [2].

CoAP uses two types of Quality of Service mechanisms:

- Confirmable Message: UDP is unreliable transport layer protocol as it doesn't have any inbuilt retransmission mechanism. Confirmable message provides the reliability by implementing retransmission mechanism in the application layer itself. This guarantees reliability in the communication. These types of messages use the acknowledgment method. As soon as destination receives message it returns acknowledgment or reset message.
- Non-conformable Message: It is just send and forget. There is no need to send an acknowledgment message. That means if any packets are lost they are not retransmitted.

978-1-5386-1887-5/17/$31.00 ©2017 IEEE

B. MQTT

Message Queuing Telemetry Transport (MQTT) is designed as pub/sub model uses TCP as transport layer protocol. In this model multiple clients which acts as publisher and subscriber connects to central broker. Every subscriber subscribes to a topic registered at broker. Each publisher publishes to a particular topic and broker broadcasts this message to all the subscribed clients. Clients can send two type of message PUBLISH or SUBSCRIBE [5]. Although TCP provides reliability at transport layer; MQTT has its own retransmission mechanism at application layer.

Three types of Quality of Services are used by MQTT also called as Quality of Service (QoS):

- QoS0: (At most once) It sends message only once. The publisher sends data to broker, in response it doesn't wait for acknowledgement (ACK) from broker. If the data sent by publisher is not received by broker, it is lost as there are no retransmissions in this QoS.
- QoS1: (At least once) To avoid the earlier problem of data, the publisher waits for ACK (APUBACK) from broker. If the ACK is not received after a predefined time interval, data is retransmitted. This profile achieves reliability but increase the overhead.
- QoS2: (Exactly once) In this, publisher sends data to broker and wait for Publish Receive (PUBREC) message back. As soon as the PUBREC is received, it discards the reference to published data and Publish Released (PUBREL) to the broker[6]. Same procedure is followed by broker. When both publisher and broker perform their tasks, it ensures successful message delivery.

C. REST

Representational State Transfer (REST) service is an architecture which uses HTTP as application layer protocol. TCP is used by HTTP as transport layer protocol. It provides various web services for data exchange and communication. It is a client server approach where client can access and use these resources provided by the server.

REST uses methods defined by HTTP such as GET, PUT, POST, DELETE etc. GET and POST are used to retrieve and send data to server. PUT is used to update and do changes to resource data. REST uses simple request/ response model for messaging. The advantage of using REST is it is widely used architecture style and it is supported by all commercial cloud platform which support M2M. It can be easily implemented in mobile or any embedded device in any language as it uses HTTP library which is available for all Operating System Distributions and almost all programming languages.

III. EXPERIMENTAL SETUP

Raspberry Pi 3 (RPI) with onboard wifi is used for sending and receiving data. A public cloud machine is used as server (or broker in case of MQTT). The server is located in Singapore and client RPI is in Bangalore, India. Two types of internet connections are used to test different scenarios. One uses broadband link of 50Mbps and other uses 4G network with speed upto 6Mbps. In both cases the the performance is affected by congestion in the network and routing process.

For software implementation Mosquitto, aiocoap and django are used for MQTT, CoAP and REST. The RPI runs on Raspbian Jessie Lite which is minimal linux image based on Debian Jessie without GUI.

Various payloads are used to emulate sensor data. The experiment consists of sending payloads from sender to server or broker and back to receiver. Each experiment is repeated at least 1000 times. To get accurate timestamp values sender and receiver are kept on same machine.

We define following performance metrics.

1. Time: Time taken to send a payload from RPI to Server is upload time. Time from Server to RPI is added with upload time gives us Round Trip Time(RTT). For following functions time is recorded:
    a. Sender establishes connection with server
    b. Send Data and waits for acknowledgement
    c. Receiver connects with server
    d. Receives data
2. Data Used: This is actual data usage to send a payload. This consists of upload and Round Trip Bandwidth

IV. EXPERIMENTAL RESULTS

The payloads vary from 10 bytes to 1 megabytes. Two types of graphs are plotted for each performance metric. One for upload and second for total=upload + download.

1. Time:

- *Upload*:

For small payloads up to 10kb CoAP comes out be fastest compared to other protocols. Refer Fig5 and Fig6. But as the payload size increases CoAP loses its advantage and REST is best for such payloads in our experiments. Wireshark analysis shows that CoAP fragments payload in equal parts. Which is good for small payloads as no of packets transmitted and received are less. But as the payload size increases it linearly fragments payload in small fragments. CoAP uses timeout mechanism for retransmission [4], it needs to wait for acknowledgement of each packet. More the fragments more timeouts and more delay. It uses simple retransmission mechanism in order to keep it as simple as possible.

Whereas MQTT and REST fragments the payload depending on size of payload. Larger the payload size bigger the fragment size. Bigger fragments size means lesser acknowledgements and lesser request responses. MQTT is slightly better as compared with REST for small payloads.

- *Total=Upload + Download:*

Here it follows upload pattern of CoAP and REST. But MQTT2 is special case Fig 7. The MQTT2 has to take care of delivery of message without duplication of message. To achieve this MQTT2 takes very long times to download payload hence more total time. And CoAP takes big time for larger payloads due to large timeout interval for retransmission (Fig. 8).

2. Bandwidth Consumption:

For small payloads CoAP takes very little bandwidth/data usage as compared to rest of the protocols. The bandwidth consumption increases gradually from CoAP to MQTT to REST in case of small payloads, as shown in figure 1,2,3,4. As payload size increases REST has consumes least bandwidth and CoAP consumes more bandwidth. For payload size of 100kb all protocols have almost equal bandwidth consumption.

Graphs for Broadband connection:

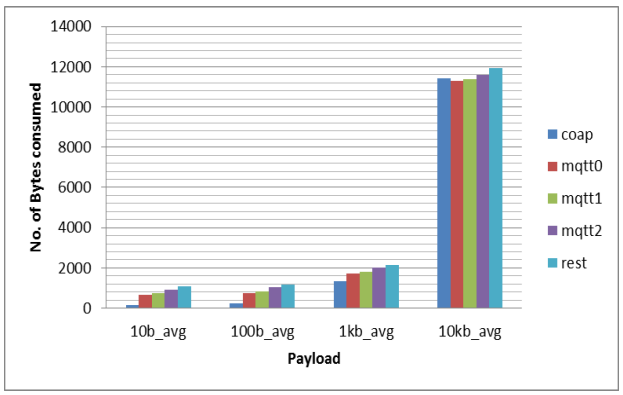

Fig 1: Upload Bandwidth (Small Payloads)

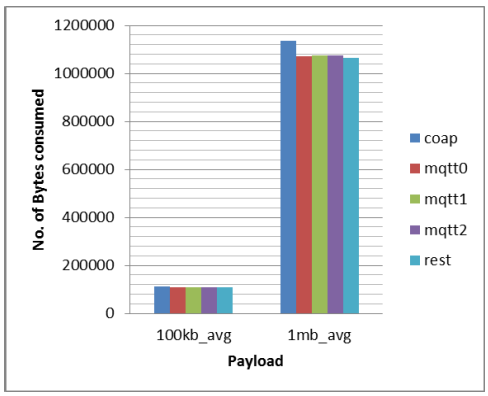

Fig 2: Upload Bandwidth (Big Payloads)

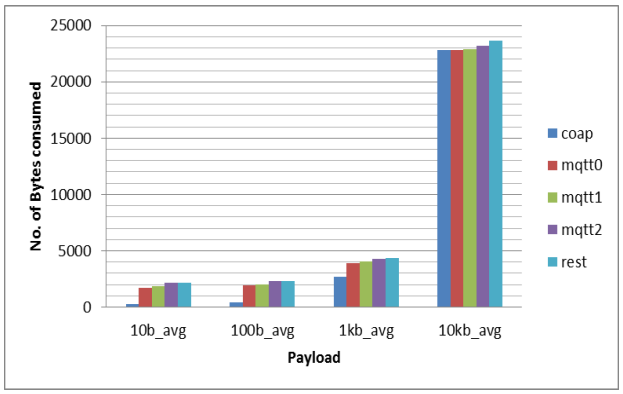

Fig 3: Total Bandwidth (Small Payloads)

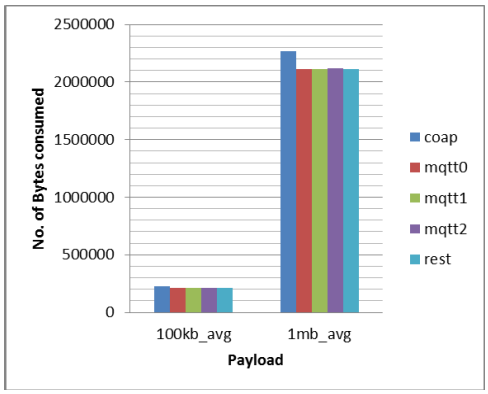

Fig 4: Total Bandwidth (Big Payloads)

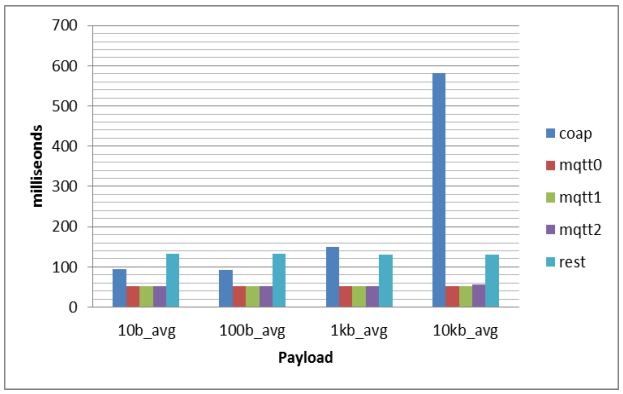

Fig 5: Upload Time (Small Payloads)

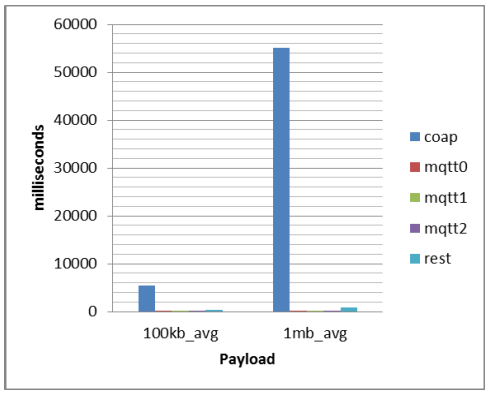

Fig 6: Upload Time (Big Payloads)

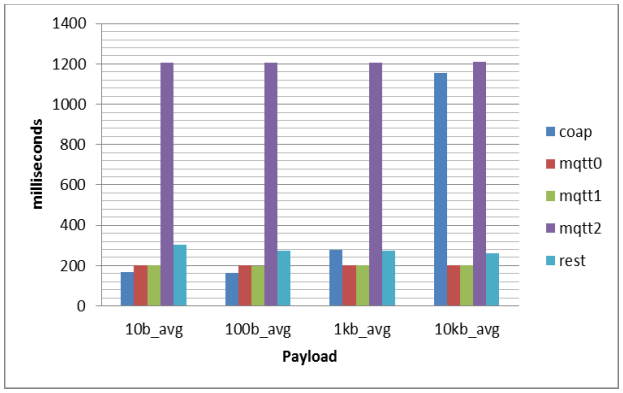

Fig 7: Total Time (Small Payloads)

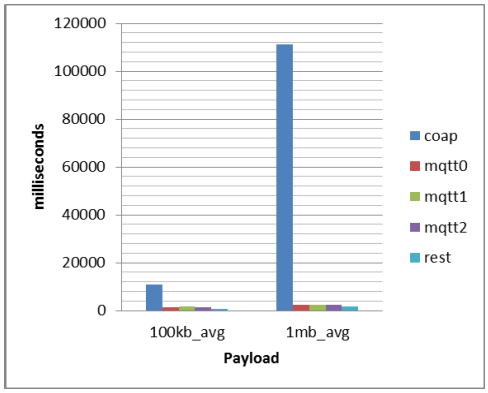

Fig 8: Total Time (Big Payloads)

Graphs for 4G connection:

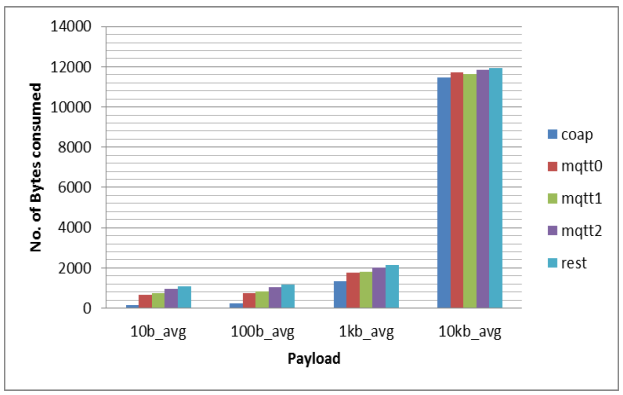

Fig 9: Upload Bandwidth (Small Payloads)

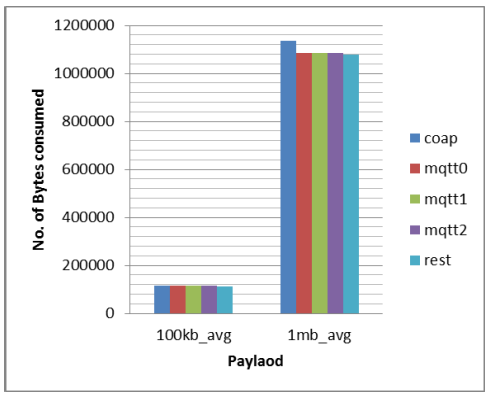

Fig 10: Uplaod Bandwidth (Big Payloads)

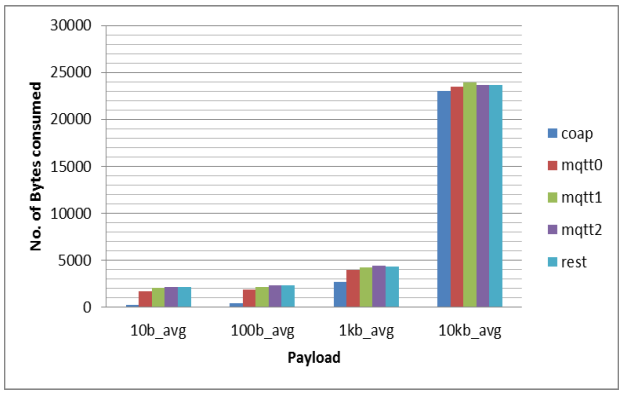

Fig 11: Total Bandwidth (Small Payloads)

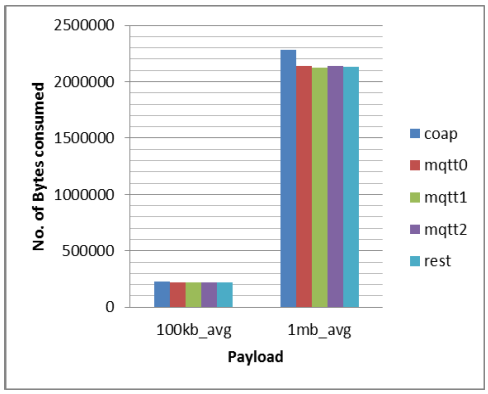

Fig 12: Total Bandwidth (Big Payloads)

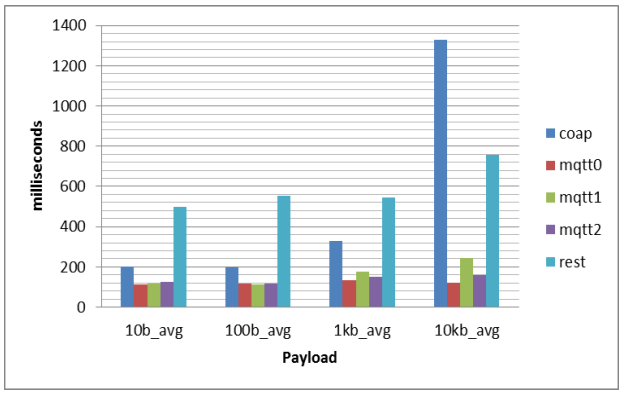

Fig 13: Upload Time (Small Payloads)

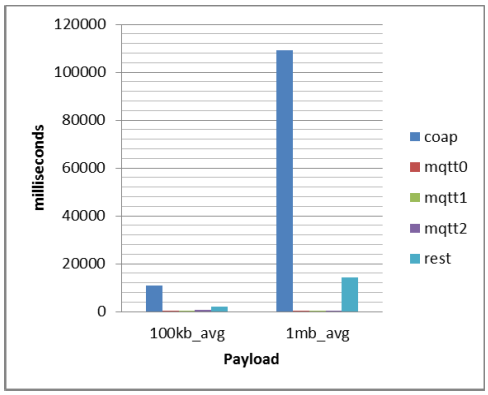

Fig 14: Upload Time (Big Payloads)

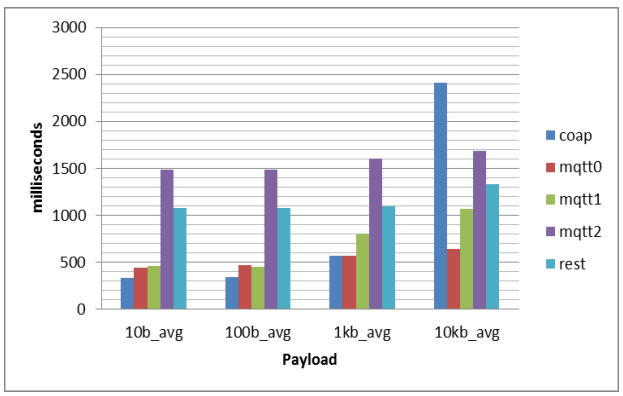

Fig 15: Total Time (Small Payloads)

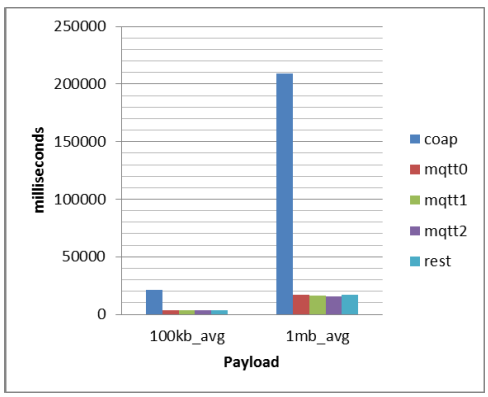

Fig 16: Total Time (Big Payloads)

V. CONCLUSION

The main objective of this paper was to analyze performance of CoAP, MQTT and REST protocols in actual internet of things environment. The Raspberry Pi acts as a Gateway device which collects data from various sensors and sends to server via internet. Two types of internet connections are considered Broadband connection with 50Mbps speed and 4G connection with 6Mbps speed. Performance was evaluated in terms of time taken and bandwidth consumed for each payload transfer in round trip. The results show that CoAP is most efficient in terms of time and bandwidth for smaller payloads and its performance deteriorates as payload size increases.

The bandwidth consumption remains almost same with change in network from broadband to 4G. But time required to upload and download changes in case of 4G network. It takes at least double time as compared to broadband connection some times 5 times. This varying time is caused by performance of underlying 4G network which is not as consistent as broadband. The MQTT2 takes a longest time in case of smaller payloads.

References

[1] Cisco Visual Networking Index: Forecast and Methodology, 2016–2021 http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[2] RFC 7252 - The Constrained Application Protocol (CoAP) - IETF Tools https://tools.ietf.org/html/rfc7252

[3] MQTT Version 3.1.1 Documentation Oasis http://docs.oasis-open.org/MQTT/MQTT/v3.1.1/os/MQTT-v3.1.1-os.html

[4] A. Betzler, C. Gomez, I. Demirkol and J. Paradells, "CoAP congestion control for the internet of things," in IEEE Communications Magazine, vol. 54, no. 7, pp. 154-160, July 2016.doi: 10.1109/ MCOM.2016. 7509394

[5] M. B. Yassein, M. Q. Shatnawi and D. Al-zoubi, "Application layer protocols for the Internet of Things: A survey," 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, 2016, pp. 1-4. doi: 10.1109/ICEMIS.2016.7745303

[6] M. Collina, M. Bartolucci, A. Vanelli-Coralli and G. E. Corazza, "Internet of Things application layer protocol analysis over error and delay prone links," 2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), Livorno, 2014, pp. 398-404. doi: 10.1109/ASMS-SPSC.2014.6934573

[7] V. Lampkin, W. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, and R. Xiang, “Building smarter planet solutions with mqtt and ibm websphere mq telemetry,” 2012.

[8] Thangavel, D., Ma, X., Valera, A., Tan, H. X., & Tan, C. K. Y. (2014, April). Performance evaluation of MQTT and CoAP via a common middleware. In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on (pp. 1-6). IEEE.