Evaluation of Hypertext Transfer Protocol (HTTP) And Message Queuing Telemetry Transport Protocol (MQTT) Based On Required Network Resources For IoT

Dr. Kalaivazhi Vijayaragavan[1], R. Giritharan[2], M. Hariharan[3], S Johnprince[4]
[1]Associate Professor, Dept of Information Technology
[2, 3, 4]Dept of Information Technology
[1, 2, 3, 4] Anjalai Ammal Mahalingam Engineering College, Thiruvarur, India

Abstract- *The standard and real-time communication technology has an unalloyed inevitability for the development of Internet of Things (IoT) applications. However, the selection of standard and effective messaging protocol is a challenging task, since it depends on the nature of the IoT system and its messaging requirements. All type of IoT system not able to support all messaging requirements. Messaging protocol is an ongoing problem for the IoT industry; consequently, it is important to understand the pros and cons of the widely accepted and emerging messaging protocols for IoT systems to determine their best-fit scenarios. Therefore, this paper presents an evaluation of the two established messaging protocols MQTT, and HTTP for IoT systems. HTTP has been widely applied for data transfer. However, in networks for IoT, this protocol causes a large overhead. This problem, can be solved by named based transfer protocols has been discussed. This paper compares the performance of HTTP with that of MQTT, a type of named based transfer protocol. Also, this paper suggests enhancements to MQTT for better performance*

Keywords- IoT, HTTP, MQTT, NDN, ICN, Performance Evaluation, Protocol Overhead

I. INTRODUCTION

The Internet of Things (IoT) is a concept aims to extend the benefits of continuous internet connection for various things such as remote control, and data monitoring. IoT is the idea of researchers who need to optimize equipment's such as sensor, radio frequency identification (RFID), wireless sensor network, and all equipment's connected to the Internet network to communicate with humans. Collect physical data using sensors such as temperature and humidity, and then send it to the server to be stored in database or be displayed on the application interface [3]. Although Internet Protocol (IP) has been adopted for most types of communication, it will have some problems when it is applied to IoT.

Presently, Internet access needs application protocols over TCP/IP or UDP/IP. One of the application protocols is Hyper Text Transfer Protocol (HTTP), which have been standardized in IETF, e.g., [2] (initial version) and [1] (the latest version), and has been applied for general communication over Internet. However, when HTTP is applied to communication in IoT, in which a huge number of petite data blocks are transferred, protocol overhead and resulting performance degradation are a serious problem.

Moreover, IP addressing depends on physical location, which causes the problem of complexity of network control. To solve these problems, name- based architectures, such as Named Data Networking (NDN), Content Centric Networking (CCN), and Information Centric Networking (ICN) have been discussed; see e.g., [4] – [10]. Some of the examples focus on adopting these architectures to IoT; see e.g., [11] − [12].

In these architectures, MQ Telemetry Transport (MQTT) is one of the protocols, as described in [13]. MQTT reduces protocol overheads and provides high efficiency communication for IoT. It also invokes "Name based routing," and mitigates IP address based routing for IoT traffic flows.

This paper discusses the possibility of considering MQTT as a candidate for the communication protocols on the IoT platform. It evaluates the performance of MQTT with that of HTTP. Moreover, it proposes new mechanisms to enhance the current MQTT specifications.

II. HTTP FOR IoT COMMUNICATION

It has been assumed that HTTP can be applied to communication for IoT. The HTTP transfer a large number of miniature packets. Protocol overhead of HTTP may cause serious problems, such as consumption of network resources and large delays.

Communication using HTTP has been configured as shown in Figure 1. Sequence charts are also shown in Figure 1.

Meanwhile HTTP is operated over TCP/IP, reliable communication is provided. TCP established connections are released on every access, since accessed data is transferred based on IP address and URL and their relationship is changed dynamically. In general, after various times of establishment of release of a connection, communication is completed. Therefore, communication for IoT causes serious overhead and consumption of network resources during the communication.

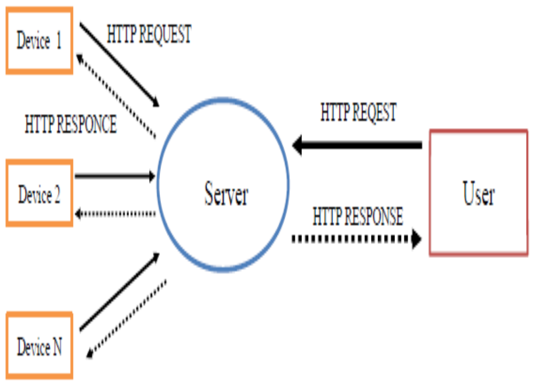

Fig. 1. System configuration using HTTP

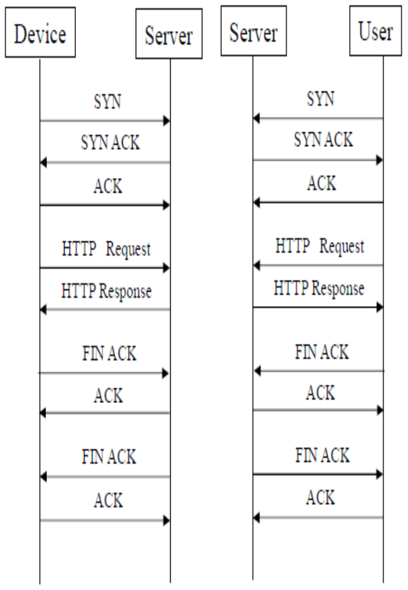

Fig. 2. Communication sequences on HTTP

III. MQTT AND ITS PERFORMANCE

MQTT moderates such protocol overheads in HTTP. This section describes sequence process by MQTT for IoT communication.

Summary of operations in MQTT

Three types of transfer modes of MQTT are based on reliability: QoS0 (Non assured transmission), QoS1 (Assured transmission), and QoS2 (Assured service on applications). QoS1 is alike to HTTP from a reliability point of view.

Whereas HTTP is a symmetric protocol, MQTT has an asymmetric architecture for lightweight. Generally communication for IoT are non-intelligent distributed devices communicate with a server with intelligent ability, asymmetric communication is provided. Due to this point, MQTT is more suitable than HTTP.

MQTT encompasses of two message sets on a connection, “Publish” and “Subscribe.” Data blocks are sent by Publish message and are received by Subscribe message. To identify the data blocks “topic” is used. Data blocks received are identified by the topics registered by Subscribe message, in advance.

The system configuration is shown in Figure 3. In this configuration, communication sequence in monitor of devices by a user is shown in Figure 3. Sequence of communication to control devices on MQTT are shown in Figure 4

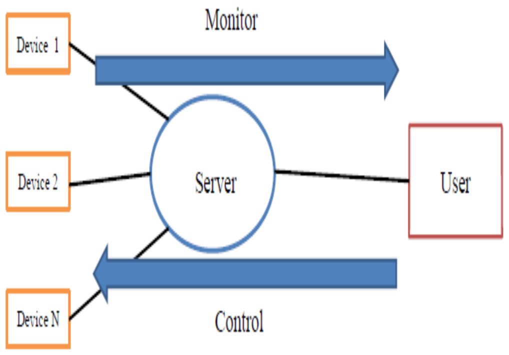

Fig3. System Configuration Using MQTT

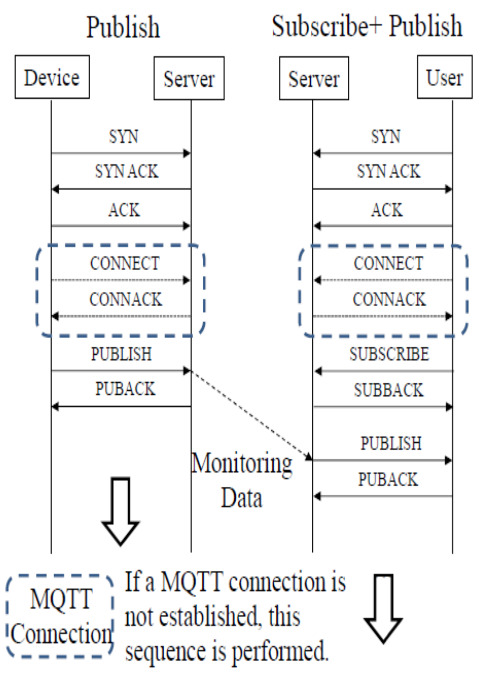

Fig. 4. Communication sequences in monitor of a device on MQTT

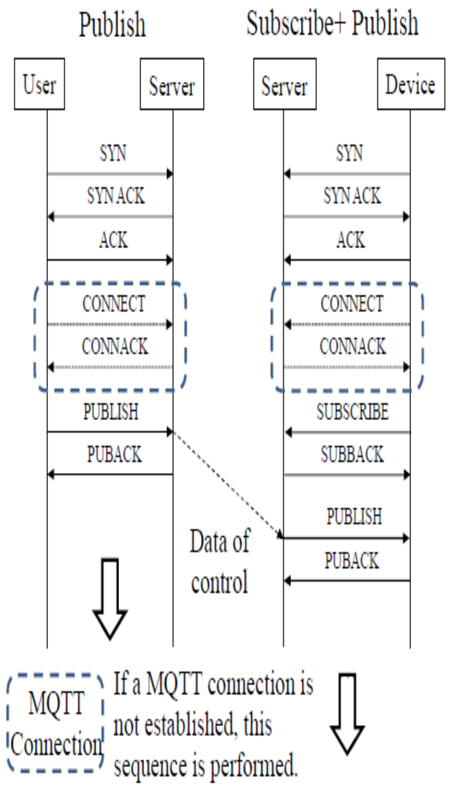

Fig. 5. Communication sequences to control a device on MQTT

IV. EXPERIMENTAL SETUP

Experiment has been carried out by collecting metrics on response time and packet size when sending identical payload through MQTT and HTTP, and by the variation of payload size and number of messages over one connection session has been done. This helps to analysis the characteristics of protocols and differences between the two protocols.

The approach used is by having a single registry in Cloud IoT Core that accepts both HTTP and MQTT connections. The device messages are routes by the registry to a single Pub/Sub topic which has one Cloud Functions endpoint as the subscriber: the Cloud Function simply writes the payload to log.

The end device is simulated in laptop, which runs both a MQTT client and a HTTP client, and then measures the response time and tracks the packets sent over the wire.

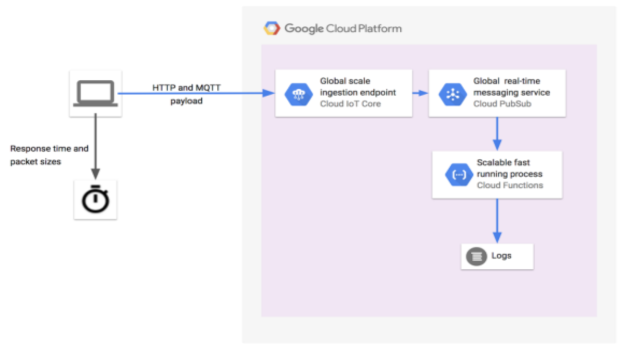
Google Cloud Platform
HTTP and MQTT payload
Response time and packet sizes
Global scale ingestion endpoint
Cloud IoT Core
Global real-time messaging service
Cloud PubSub
Scalable fast running process
Cloud Functions
Logs

Fig 6 Experiment Setup

Properties of the protocols

Before implementation , the review of MQTT and HTTP shows the influence, how the tests can setup..

MQTT (Message Queuing Telemetry Transport), describes as a publisher subscriber pattern, in which clients connect to a broker and the remote devices publish messages to a shared queue. Optimization of this protocol is based on message size, for efficiency.

HTTP adheres to the standard request response model.

To have comparison between the two protocols, the steps in the authentication process (handshake) need to be taken into account. Connect and disconnect messages are measured sequentially with the actual data messages in the case of MQTT. Since there will be the overhead for the

MQTT case, we have to send a different number of data messages between one connect-disconnect cycle and the next.

Trace packets sent over wire

To have detailed view of the packet size being transmitted for both protocols, we used Wire shark.

Locust client implementation

We used Locust.io to perform load tests and to compile the metrics. Locust.io gives a simple HTTP client from which to collect your timing data, whereas for the MQTT profiling, we tested with the Eclipse Paho MQTT client package, authenticated via JWT with Cloud IoT Core. The source code for the test is available here.

Let take a closer look of the MQTT Locust client. First, an initial connect and disconnect is issued in the `on_start` function to preload the MQTT client with all the credentials it needs to connect with Cloud IoT Core, so that credentials can be reused in each measurement cycle.

```
def on_start(self):
self.client.get_client()
self.client.connect_to_server()
self.client.disconnect()
```

While publishing messages, we check the qos=1 flag to ensure that the message has beeen delivered by waiting for a pub_ack from Cloud IoT Core, which can compared to the request response cycle of the HTTP protocol. Likewise the Paho MQTT client publishes thel messages asynchronously, which forces to call the wait_for_publish() function on the MQTT Message Info object to block execution until a PUBACK response is received for each message.

```
self.client.reconnect()
self.client.loop_start()
for i in range(1, numberOfMsg+1):
 msgInfo =     self.client.publish(mqtt_topic, payload, qos=1)
msgInfo.wait_for_publish()
self.client.disconnect()
```

V. TEST CASES

MQTT

Varying the number of messages:

To measure the response time, the system uses the following scenario by sending 1, 100, and 1000 messages over a single connection cycle each, and also captured the packet sizes that were sent over the wire.

Varying the size of messages:

Here we measured the response time for sending a single message with 1, 10, and 100 property fields over a single connection cycle each, and then capture the packet size sent.

HTTP

Average response time is measured for sending a payload with 1, 10, and 100 property fields and then captures the packet size over the wire.

Results

MQTT response time

Below are the results of running both the HTTP and MQTT cases with only one simulated Locust user. The message transmitted is a simple object containing single key-value pair.

Variation in number of messages:

No.of messages over a single connection cycle	Average response time for a connection cycle (ms)	Average response time per message (ms)
1	113	113
100	4724	47
1000	40366	43

Variation in payload size:

MQTT response time:

No.of property fields in each message	Average response time for a connection cycle (ms)	Average response time per property field (ms)
1	207	207
10	212	21
100	191	2

HTTP response time

No.of property fields in each message	Average response time for a connection cycle (ms)	Average response time per property field (ms)
1	289	289
10	280	28
100	247	3

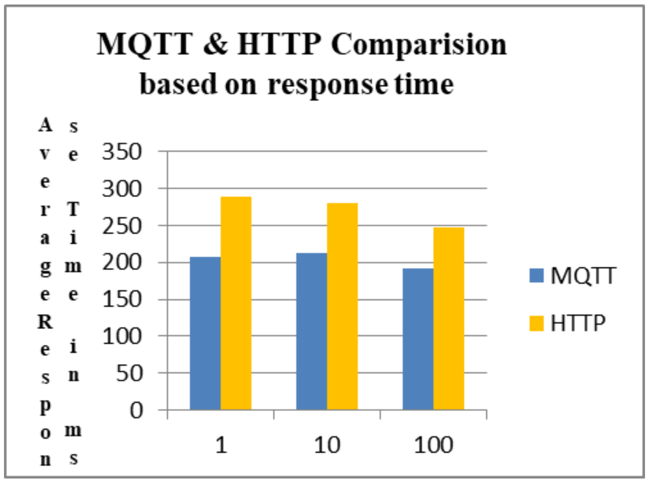

Fig 7 Response Time Comparision

Packet size capturing results

To get a more accurate view of what packets are actually being sent over the wire, we used Wire shark to capture all packets transferred from and to the TCP port used by Locust.io. The sizes of each packet were also captured to give a precise measure on the data size overhead of both protocols.

MQTT

MQTT over TLS connecting procedure log

The wire log shows the handshake process that sets up a TLS tunnel for MQTT communication. The main part of this process consists of the exchange and verification of both certificates and shared secret.

Single message publish cycle

The wire log over single message publishing cycle shows that there's a MQTT publish message from client to server, a MQTT publish ACK message back to the client, plus the client also sends back a TCP ACK for the MQTT ACK received.

Disconnect procedure log

HTTP

Handshake procedure for establishing the TLS connection

The initialization procedure for setting up the TLS tunnel is the same for the HTTP case as it is for the MQTT case, and the now established secure tunnel is re-used by all subsequent requests.

Single publish event log

The HTTP protocol is connectionless, meaning that the token is sent in the header for every publish event request and the Cloud IoT Core HTTP bridge will respond to every request.

Table summarizes the sums of the packet size sent during each of the transfer states for both MQTT and HTTP:

(Bytes)	MQTT	HTTP
Establish connection	5572	2261
Per message published	388	3285
Disconnect	376	0
Sum	6336	5546

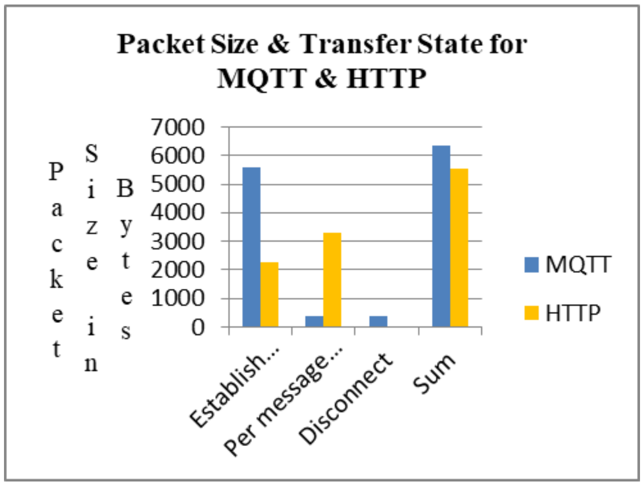

Fig 8 Packet Size and Transfer State

And this table shows how variation in payload size affects packet size over wire:

	MQTT 1 field payload	MQTT 10 fields payload	MQTT 100 fields payload	HTTP 1 field payload	HTTP 10 fields payload	HTTP 100 fields payload
Packet size transmitted over wire (Byte)	388	686	3788	1982	2332	6683
Packet size ratio compared to single field	1	1.77	9.76	1	1.18	3.37

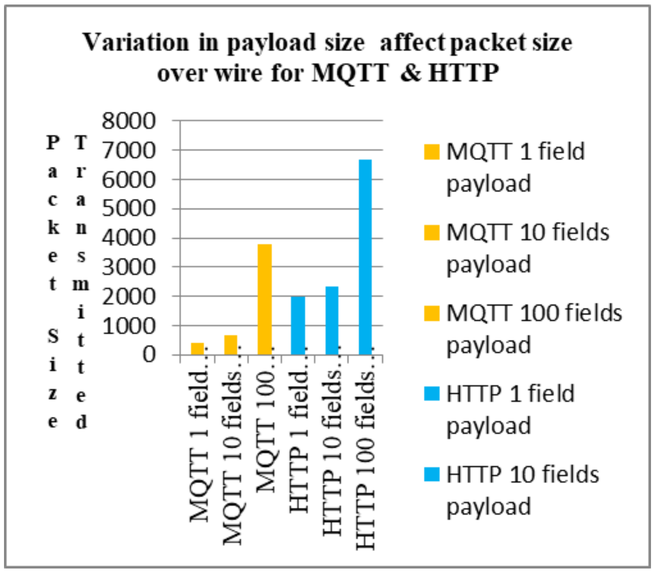

Fig 9 Variation in payload size affect packet size

Summary

(Bytes)	MQTT	HTTP
Establish connection	5572	2261
Per message published	388	3285
Disconnect	376	0
Sum	6336	5546

Case study: Measuring the amount of data received over the wire

MQTT is often called a protocol for the Internet of Things. Which means that it must be more lightweight for network usage? The experts in MQTT solutions also note that it's especially efficient in wired data transmission. Let's see what network-related data we can get from packet sniffers to compare MQTT over SSL and HTTPS.

Test 1. Comparison of protocols service part

Measured the number of bytes and packets required to establish a connection, send/receive data (simple JSON {"test":1234}) and close the connection. Here's the result:

Secure Session	Out coming bytes	Incoming bytes	Number of packets
HTTPS	1734	4186	20
MQTT over SSL (WiFi)	1274	4159	20
MQTT over SSL (Ethernet)	1186	4075	18

Inference : MQTT service part requires only **10% less traffic** than HTTP. The advantage of MQTT service part over Ethernet vs Wireless is negligible.

VI. CONCLUSION

Considering the result that compares response time for one connection cycle of MQTT, we can clearly see that the response time increases during the initial connection setup for sending single messages. The level that equals the response time of sending a one message over HTTP, which in our case rounds up to 120 ms per message. The influence in terms of data amount sent over wire is even more significant for MQTT in which about 6300 bytes is sent for one message, this is better than for HTTP, which sums up to 5600 bytes. By observing at the packet traffic log, we can see that the dominant part — more than 90% of the data transmitted is for setting up and tearing down the connection.

The benefit of MQTT over HTTP happens when the single connection is reused for sending multiple messages in this case the average response per message meets to around 40ms and the data amount per message meets to around 400 bytes. In case of HTTP, these reductions simply aren't possible.

From the result of the test for variation in payload size, we could perceive that response times were kept constant as the payload size went up. The description here is that since the payloads being sent are small, the full network capacity isn't utilized and as the payload size increases, more of the capacity is being used. Another remark we can make looking at the network packet log is that even as the amount of information packed into the payload improved by 10x and 100x, the amount of data actually transferred only increased by 1.8x respective 9.8x for MQTT and 1.2x and 3.4x for HTTP, which shows the effect of the protocol overhead when publishing messages.

The conclusion we have appeal is that when choosing MQTT over HTTP, it's really important to reuse the same connection as much as possible. If connections are set up and torn down regularly just to send individual messages, the efficiency gains are not significant compared to HTTP. The supreme efficiency can be achieved through MQTT's by increase in information density for each payload message. The most honest approach is to reduce the payload size where more data can be transmitted in each payload, which can be realized through choosing proper compression and package approaches based on the type of the data being generated.

For streaming applications, time-window bundling can upturn the number of data points sent in each message, whereby choosing the window length wisely in relation to the data generation pace and available network bandwidth, you can transmit more information with lesser latency. In most of the IoT applications, the earlier methods mentioned cannot easily be applied due to the hardware restrictions of the IoT devices. Depending on the functional requirements from case to case, a feasible solution would be the usage of gateway devices, with advanced capabilities in terms of processing and memory. The payload data is firstly delivered from the end device to the gateway, whereby different optimizing measures can be applied before added delivery to Google Cloud.

REFERENCES

[1] M. Belshe, R. Peon, M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", IETF RFC 7540, 2015

[2] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", IETF RFC 1945, 1996

[3] Suresh P, Daniel J V, Aswathy R H 2014 A state of the art review on the Internet of Things (IoT) History, Technology and fields of deployment

[4] J. Luo, C. Wu, Y. Jiang, J. Tong, "Name Label Switching Paradigm for Named Data Networking", IEEE Communications Letters, Vol. 19, pp.335 - 338, 2015

[5] S. Eum, K. Nakauchi, Y. Shoji, N. Nishinaga, M. Murata, "CATT: Cache aware target identification for ICN", IEEE Communications Magazine, Vol. 50, No.12, pp. 60 – 67, 2012

[6] C. Fang, R. Yu, T. Huang, J. Liu, Y. Liu, "A Survey of Green Information-Centric Networking: Research Issues and Challenges", IEEE Communications Surveys & Tutorials, Vol. 17, No.3, pp. 1455 – 1472, 2015

[7] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B.Ohlman, "A survey of information-centric networking", IEEE Communications Magazine, Vo. 50, No.7, pp. 26 － 36, 2012

[8] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, G. Polyzos, "A Survey of Information-Centric Networking Research", IEEE Communications Surveys & Tutorials, Vol. 16, No.2, pp. 1024 – 1049, 2014

[9] M. Yamamoto, "Research trends on In-Network caching in content oriented networks", IEICE Technical Report., Vol. 115, No.461, pp. 23 －28, 2015

[10] Md. F. Bari, S. Chowdhury, R. Ahmed, R. Boutaba, B. Mathieu, "A survey of naming and routing in information-centric networks", IEEE Communications Magazine, Vol. 50, No.12, pp.43 －52, 2012

[11] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. Aguiar, A. Vasilakos, "Information-centric networking for the internet of things: challenges and opportunities", IEEE Network, Vol. 30, No.2, pp. 92 - 100, 2016

[12] H. Yue , L. Guo, R. Li, H. Asaeda, "Yuguang FangDataClouds: Enabling Community-Based Data-Centric Services Over the Internet of Things", IEEE Internet of Things Journal, Vol. 1, pp. 472 － 482, 2014

[13] I. Sato, T. Kurira, K. Fukuda, T. Tsuda, "A extention of information centric for IoT applications", Repeort of 3rd Technical committee on information centric networking (ICN), in IEICE, 2015

[14] [14] IBM, "MQTT V3.1 Protocol Specification", 2012, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqttv3r1.html

[15] T. Fujita, Y. Goto, A. Koike, "M2M architecture trends and technical issues", The Journal of IEICE, Vol.96, pp.305 － 312,2013

[16] "Network performance objectives for IP-based services", ITU-T Recommendation, Y.1541, 2011

[17] L. Kleinrock, "Queuing Systems, Volume 1: Theory", Wiley Inter science, ISBN 0-471-49110-1, 1980

BIBLIOGRAPHY

KalaivazhiVijayaragavan obtained her Bachelor’s degree in Electronics and Communication Engineering from REC – Trichy (Presently NIT – Trichy) in 1996 and her Master’s Degree in Computer Science and Engineering from SASTRA University, Thanjavur in 2003. She obtained her Doctoral Degree from Anna University, Chennai.She is currently working as an Associate Professor in the Department of Information Technology at AnjalaiAmmalMahalingam Engineering College, Kovilvenni. Her areas of interest include Machine Learning, Data Analytics, IoTMobile Communication and Computer Networks.

R. GIRITHARAN, Pursuing B.Tech – Information Technology (IT) Final year in ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE,Thiruvarur

M. HARIHARAN, Pursuing B.Tech – Information Technology (IT) Final year in ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE,Thiruvarur

S. JOHNPRINCE, Pursuing B.Tech – Information Technology (IT) Final year in ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE,Thiruvarur