A Qualitative Comparison Model for Application Layer IoT Protocols

Syed Rameez Ullah Kakakhel
Department of Future Technologies
University of Turku
Turku, Finland
srukak@utu.fi

Tomi Westerlund
Department of Future Technologies
University of Turku
Turku, Finland.
tovewe@utu.fi

Masoud Daneshtalab
Department of Embedded Systems
Mälardalen University
Västerås, Sweden
masoud.daneshtalab@mdh.se

Zhuo Zou
Micro-Nano System Center
Fudan University
Shanghai, China
zhuo@fudan.edu.cn

Juha Plosila
Department of Future Technologies
University of Turku
Turku, Finland
juplos@utu.fi

Hannu Tenhunen
Department of Future Technologies
University of Turku
Turku, Finland
hannu.tenhunen@utu.fi

***Abstract*— Protocols enable things to connect and communicate, thus making the Internet of Things possible. The performance aspect of the Internet of Things protocols, vital to its widespread utilization, have received much attention. However, one aspect of IoT protocols, essential to its adoption in the real world, is a protocols' feature set. Comparative analysis based on competing features and properties are rarely if ever, discussed in the literature. In this paper, we define 19 attributes in 5 categories that are essential for IoT stakeholders to consider. These attributes are then used to contrast four IoT protocols, MQTT, HTTP, CoAP and XMPP. Furthermore, we discuss scenarios where an assessment based on comparative strengths and weaknesses would be beneficial. The provided comparison model can be easily extended to include protocols like MQTT-SN, AMQP and DDS.**

Keywords—IoT Protocols, qualitative comparison, HTTP, MQTT, CoAP, XMPP.

I. INTRODUCTION

Internet of Things (IoT) protocols have been thoroughly studied over the last decade. Works have ranged from adopting current protocols for IoT (µXMPP [1]) to the development of more specific ones (CoAP, MQTT-SN). A very conservative google scholar query ('IoT protocol' *title only*) on the topic gives more than 300 results. The diverse nature, use-cases and features have resulted in the need for cross-comparison studies. Majority of the protocol comparisons and use-case evaluations have been based on computational and network performance analysis. This is an efficient and quantifiable method that allows us to define the engineering needs. Needs such as selecting the level of devices' computational capability, bandwidth and latency requirements. Quantifiable statistics also allow us to come up with better battery life estimates.

What these strictly performance-based evaluations ignore is assessing the need for reliability, scalability, interoperability and security demanded from IoT protocols. Such qualities are inherent to the protocol and not elaborated by the performance results. The selection of a protocol is also not limited to one criterion and is mostly a tradeoff of one parameter over the other. This tradeoff selection of performance vs features vs compatibility might be left to the designer or just be based on the chosen platform's support. In broader contexts, such as smart cities, digital healthcare, business continuity and growth, a higher-level overview of the requirements and features is necessary to bring the multiple IoT stakeholders together. Hence, we see the need for a broader protocol comparison model that is more than just performance metrics.

Furthermore, a better understanding of the features and services provided by IoT protocols will enable the designers to make more informed choices. Choices that would affect both current and future needs. There can be no one protocol which can fulfil all the needs, in all different cases. Thus, it is essential to have a methodology to compare and classify them based on the main features that are needed for IoT applications. As of this writing, no such comprehensive criteria or model exists in the literature. In this paper, we provide such a qualitative comparison model. This model can be utilized by urban planners, multiple IoT stakeholders, and for other need-based evaluations.

II. RELATED WORK

The closest work to ours is by Niccolo et al. [2] and Edielson et al. [3]. They offer a qualitative comparison between MQTT and COAP. The comparisons are thorough but expressive, thus not easily extendable beyond the two protocols they assess. Their comparison is also limited to a subset of the attributes we have collected and identified (nine and seven respectively).

Vasileios et al. [4] and Ala et al. [5] provide a detailed description of major application layer IoT protocols. However, they do not provide criteria for cross comparison. Yuang et al.

[6] offer a quantitative comparison of DDS, MQTT and CoAP in the context of eHealth. Paridhika et al. [7] provide a performance analysis in the context of smart parking. Antonio et al. [8], look at semantic and syntactic interoperability in the context of HTTP, CoAP and MQTT. Jens Dede and Anna Forster [9] offer a qualitative analysis of connectivity protocols such as Wi-Fi Direct and BLE for opportunistic communication. Kabeer et al. [10] provide a qualitative comparison of routing protocols.

Mentioned above is a small subset of the literature on IoT protocol comparisons. However, the theme remains the same; the studies focus on quantitative comparisons in varying contexts while offering brief qualitative comparisons. Even if provided, the qualitative assessments are limited to a subset of the protocols, expressive and not easily extendable. To the best of our knowledge, ours is the first comprehensive effort at defining standard universal criteria for comparisons based on non-performance metrics.

The rest of the paper is organized as follows: Section II describes the model parameters. Section III offers a comparison of four IoT protocols. Section IV is a discussion on the impact of such a classification. Section V concludes the paper with a discussion on its limitations and further directions.

III. Model Attributes

Our classification model is based on the content analysis of existing research and authors own assessments. The model is divided into five main categories, namely, Communication Attributes, Security Attributes, Connection Attributes, Operational Attributes and Message and Payload support. All of these have sub-categories that further expand on the main characteristics. We provide a brief definition of each category and a short portrayal of its importance.

A. Communication Attributes

This category broadly relates to how the protocol handles the principle communication. This section illustrates and differentiates via how the protocol operates, its reliance on any lower level protocol, special communication features and complexity. Sub categories for communication attributes include:

1) Communication Pattern

Communication pattern refers to which communication paradigm the protocol follows. One example of a communication pattern is a Request/Response. In Req-Resp architecture an end node sends a request to the central server for connection/data initiation (Req), and the server responds in a pre-defined pattern (Resp). This model is followed in the design of the current internet, print servers and old pre-blackberry email services. Another communication pattern is Publish-Subscribe or PubSub in short. In this design, the clients during or after connection establishment define what type of messages they are interested in (generally called topics or classes). Whenever another client (publisher) sends a message of that type (topic), it is forwarded to all the clients who had expressed interest in it (subscribers). The publishers are not concerned with the number of subscribers, nor with how do the messages are delivered. The central server (a broker in this case) handles that. No one-on-one messaging takes place. Another communication pattern is one where nodes send direct messages to each other. Queues to manage communication is becoming common in cloud-based services and is generally seen as more of a server-server communication model than a client-server model. This form of communication has its roots in the inter-process communication paradigms.

2) Transport Protocol

All application layer protocols, in the scope of this study, assume an underlying transport protocol to be present. The transport layer provides error correction & flow control. Relying on TCP/IP transport layer protocol frees the application layer protocol from providing services like data integrity and thus reduces complexity. Some protocols rely entirely on the underlying transport layer for integrity and flow control while others provide their integrity features. The choice of the transport protocol is crucial as it affects the performance, bandwidth and reliability. TCP is desired in cases where reliability is of utter importance, it offers error-checking and ordered delivery of messages. UDP offers no guarantee of ordered delivery or assurance that the packet will be delivered. If the application layer protocol has its delivery and QoS features, UDP will be a good choice as a transport layer protocol over reliable network connections. Sheng et al. offer a good discussion on this topic [11].

3) Multicast support

Sometimes it is required to send messages to multiple recipients at once, a typical scenario for IoT as thousands of devices would be present in a locality. The question would come up for the designer to investigate whether the protocol provides any support for such cases. Secondly, does the protocol offer multicast support or relies on the lower level protocols of the IP suite?

4) Reliability/QoS

Does the protocol offer any surety of delivery? Does the protocol have mechanisms to ensure the transmission of messages in uncertain circumstances or during unreliable connections? How much control does the designer, or the user have over the reliability of the transmission? The answer to these questions would explain the dependability of the protocol. If a situation arises that demands absolute reliability or no reliability at all, does the protocol have the flexibility to allow the user to choose.

5) Congestion control

Congestion occurs when the systems become overloaded. Congestion results in loss of packets or loss of service altogether. An example of this is the denial of service attacks. For this comparison we are not looking at sophisticated methods of mitigating attacks but whether the protocol can handle basic packet floods. Does the protocol offer queuing, or the messages are simply discarded?

B. Connection Attributes/Performance

The connection attributes explain the communication features related to establishing and maintaining connections and not the direct communication itself. Sub categories for connection attributes include:

1) Communication Complexity

Communication complexity in our case refers to the number of steps required to establish/setup a connection and be ready to send actual data packets. An example of communication complexity would be the number of connect and acknowledgement messages and any number of identification steps required. No numerical value can define the complexity, as different protocols require different steps. Here we will divide it into low, medium, and high complexity. Low communication complexity would mean that we can start sending messages right away or after a single connect message. Such as a request-response architecture, where a client does not authenticate but sends a request to the server, and the server responds. There is no underlying complexity here, a request was sent, and a response was received. Medium complexity is when besides the above scenario, one or two more steps are required. As an example, a PubSub protocol where the client must first register with the broker and then start sending or receiving data. High complexity is when more than the above steps are required to establish a connection and send the first data packets or serve requests.

2) Signaling Traffic Generated/Frequency of Updates

Besides the normal operation, how much extra traffic is generated that is not related to the user transmission, but the functioning of the protocol itself, e.g. ping requests to identify available clients or periodic updates to assess the network? Transmissions that are inherent to the protocol and not necessarily in the control of the user.

3) Connection establishment speed/performance

A more quantitative version of communication complexity, this one deals only with the number of packets required to establish a connection. The measurements are fast, medium and slow. Slow is when a protocol has high communication complexity and requires two or more steps to establish a connection. The scores are an out of 5, two for communication complexity while the remaining three based on the number of steps required for establishing a connection and initiating communication.

4) Session Orientation

The ability to handle sessions improves performance as a single user session can be used to send many updates, contrary to making a new connection per message. Some old protocols added this support later, e.g. HTTP. Session orientation is an essential consideration for comparison between any current and future protocols as it reduces extra traffic on the network especially in case of TCP handshakes.

C. Security Attributes

The IoT will inherit the same problems as the current internet and will amplify them because of its deep penetration and direct connection to the real world. One of these issues is security and privacy. A major differentiating factor for the choice of the protocol is the security attributes. A point to note here is that we will be talking about the features provided by the protocol (in the protocol's draft documentation) and not the ones provided via third-party add-ons (unless officially recognized) or provided via commercial implementations of the protocol. The different aspects of security are detailed below:

1) Connection Security

Connection security relates to any mechanisms provided by the protocol that enable the authentication of the nodes before any actual communication takes place. Some of the methods can include a secure key exchange between the endpoints and authentication. Connection security may rely on encryption as well. This is to ensure that the end node is not spoofed, or fake nodes are not inserted. Connection security can also be achieved via whitelisting/blacklisting or ACLs.

2) Communication Security

Communication security relates to securing communication between two nodes or endpoints. In the case of PubSub, this can be the broker and the subscriber or between a client and a server. Communication security features would ensure that a third-party interception would result in no sound or identifiable information leakage, e.g. the communication is not susceptible to eavesdropping. A conventional method of achieving this is encryption.

3) User Security

User security can mean different things in different contexts. In out context user security means whether the protocol ensures the validity of a user utilizing the system to communicate, either to the server or the broker. Authenticating the user via a username/password would be one method.

D. Operational Attributes

Secondary but essential features related to the operation of the protocol once deployed are detailed here. These include:

1) Distributed Operation/Centralized

Distributed operation means that there is no control of a single node on the operation and communication of the remaining nodes or that the communication is one-to-one where both the nodes are on equal grounds. Some protocols might request that all the data goes to a central point (a server or broker) and is then distributed to the intended recipients, in other cases direct messages would be the norm. Hybrid solutions can also exist.

2) Service/Node discovery

The ability to provide service discovery is significant in connected devices. Being able to know who and what is on the network is as important as is the ability to communicate with them. Are there ways that the protocol offers via which machines can notice peers or associated group members. Does the protocol offer any ability to auto-identify on behalf of the user?

3) Message retaining/durability

Are messages retained or discarded as soon as they are delivered? Depending on the use case this can be a

distinguishing factor for selecting a protocol. In unreliable environments or guaranteed delivery requirements, it is vital to have message retaining, although this comes at the cost of memory (as we will see in the results). In case of no durability, the developer might have to add these checks themselves which undermines the usability of the protocol. Message durability helps in maintaining a strict QoS.

4) Caching

The ability of a protocol to maintain knowledge regarding the validity of information can reduce the bandwidth usage and improve response time for any subsequent requests. If the protocol knows the freshness of the information it has available, it can serve future requests for that data by itself. Caching or similar features can be very useful when using a proxy between different servers or services/protocols.

E. Messages and Payload Support

Knowing which protocol will easily integrate with the current systems is valuable and knowing the payload helps in that regard. Current web services and technologies will benefit from a protocol that offers support for web payloads (i.e. HTML, Plain text, UTF-8 encoded text, XML). Situations that require machine-to-machine interaction will benefit from JSON or byte-wise transmission. If the system already has a messaging platform installed, then XML will be a good choice. All the payload-related features are explained afterwards:

1) Message Overhead

In the shortest form, how many more data bytes are required to make a transmission happen? If the update required to be sent is 10 bytes, how many more bytes the protocol will add to it (in headers, checksums) for the 10 bytes to be transmitted. The overhead will be high if we will need 100 bytes to send 10 bytes of data. This is both specific to the protocol and the need of the moment. However, for our case we will look at the simplest scenario, 'the minimum number of extra bytes required to send 10 bytes of data'.

2) Design Orientation

Does the protocol define any document format or not? If the protocol defines a specific document format that makes it document-centric and if it does not, that will make the protocol data-centric. In this case, design orientation refers to the focus of the protocol towards the payload. In document-centric protocols, there can be multiple document formats supported or document identifiers, these help in expanding the use case of the protocol.

3) Fragmentation/Block Transfer

The capability to divide a large data set into smaller ones and then transfer them automatically without user intervention.

IV. QUALITATIVE COMPARISON TABLE

The table below puts the details in a compact format and acts as a one-stop resource in making a preliminary decision on the selection of an IoT protocol (before delving into performance issues). A detailed general description of these protocols can be found in the literature [5]. Where necessary, Appendix A can be consulted for the reasoning behind each classification decision.

TABLE 1 QUALTITATIVE COMPARISON

Comm. Attributes	**MQTT**	**CoAP**	**HTTP**	**XMPP**
Comm. Pattern	PubSub	PubSub & Req-Resp	Req-Resp	PubSub & Req-Resp
Transport Protocol	TCP	UDP	TCP	TCP
Multicast Support	Yes [1]	Yes [2]	No [3]	Yes [4]
Reliability /QoS	Yes [14]	Yes [15]	NIL/via TCP	NIL /via TCP
Congestion Control	Yes [5]	Yes [16]	Relies on TCP	Relies on TCP
Connection/Performance				
Comm. Complexity	Medium [6]	Minimal	Minimal	High
Signaling Traffic	Low to High [7]	Low [8]	Low	Low
Conn. Speed	4 [10]	3 [9]	3 [9]	5
Session Orientation	Yes	Yes	Yes	Yes
Security Attributes				
Conn. Security	NIL / or via TLS	NIL /via DTLS	NIL/ or SSL	TLS
Comm. Security	NIL / or via TLS	NIL / via DTLS	NIL/ or SSL	TLS
User Security	User/Passwd	Nil	User/Passwd [26]	SASL
Operational Attributes				
Distributed vs. Centralized	Centralized	Decentralized	Centralized	Both
Discovery	No [11]	Yes [12]	Yes [12]	Yes
Msg. Durability	Yes	Somewhat	Yes	Yes
Caching	Yes [13]	Yes [17]	Yes [28]	Yes
Message / Payload Support				
Msg. Overhead	Minimal [19]	Minimal [20]	High [21]	High [29]
Design Orientation	Byte-wise [22]	Payload Specified [23]	Payload Specified [24]	Payload Specified [25]
Block Transfer	No (max. msg 256MB)	Yes [18]	Yes [27]	Yes [30]

V. Discussion and Examination

Table 1 represents the most common factors between these protocols and their implementation. These competing protocols serve different purposes and have a very different feature set. Here we will look at a few different scenarios and how the different features affect protocol selection:

A. QoS

For utmost reliability and QoS, MQTT comes on top. MQTT QoS 2 has no substitute in any of the other protocols; it guarantees that duplicates are not sent to the subscriber. The most divergent of these protocols is XMPP which does not require any real-time QoS constraints but offers time-stamps for the server to utilize. MQTT QoS 0 is comparable to CoAP CON; both do not enforce reliability.

B. Payload Support

If the most diverse document format is desired, HTTP is the choice. If the user/designer wants control over the data format or there is a predefined standard, then MQTT makes sense. MQTT is entirely payload agnostic. Human-centric communication would require a human understandable payload format. For M2M communications, protocols supporting JSON is a better choice.

C. Security

The only protocol that enforces absolute security, from client-to-client and client-to-server is XMPP. CoAP and MQTT recommend TLS and DTLS respectively, but if multi-cast support is desired with CoAP, DTLS would have to be disabled.

D. Utilization Use Case: Smart City

Smart Cities offer a diverse and challenging environment. From transportation, to energy, emergency services, waste management and public services communications. The stakeholders in a smart city project would define their limitations, such as, control and message frequency, reliability, interoperability and integration limitations. System designers can then assess quickly via Table 1 whether there are conflicts, or all requirements can be met easily by one protocol. There is no one correct way to model for such a scenario. For the sake of elaboration, energy management requires real-time communication, therefore selection should be based on real-world performance. Public service communications require interoperability with current user paradigms, which favors HTTP. The QoS oriented focus of MQTT will favor device control, like street lights and water fountains. Waste and water management will benefit from the security and both PubSub and request-response nature of XMPP.

E. Utilization: Real-world Performance Effects

Beyond what draft documentation says, the design of a protocol has real-world performance consequences. We conducted tests on a Core i7 quad-core/eight-thread machine, running Ubuntu 14.10 with 32GB of RAM. All network communication was via localhost to avoid any potential network latencies. For the tests, the payload format was limited to plaintext. The underlying transport protocol was TCP and QoS was limited to the assured delivery offered by TCP. No security was enabled on both the HTTP and MQTT.

In our deployment, the HTTP module served 1000 clients and sent half a million messages in 12.2 seconds, consuming 100% of the CPU power at a peak memory consumption of 500KB [Figure 1]. The same was performed by the MQTT module in 3 seconds, consuming 123.3MB of RAM and 100% of the CPU power [Figure 2]. MQTT, in theory, is a much lighter protocol, but the persistent nature of messages and always open TCP connections results in substantial memory consumption while being significantly time efficient. Combining qualitative and quantitative results, the selection between MQTT and HTTP on an IoT gateway comes down to trade-offs between speed, memory consumption, QoS, service discovery and multicast support. All else being equal.

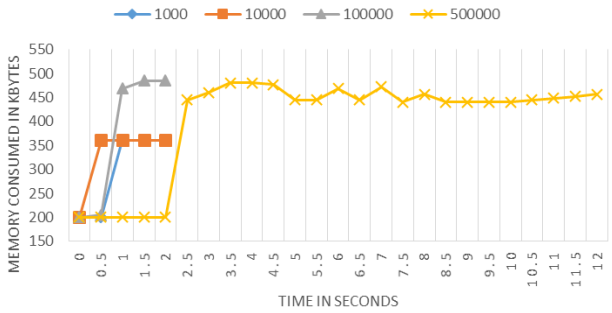

Figure 1 HTTP Memory Consumption 1000 Clients w/ different No. of messages sent.

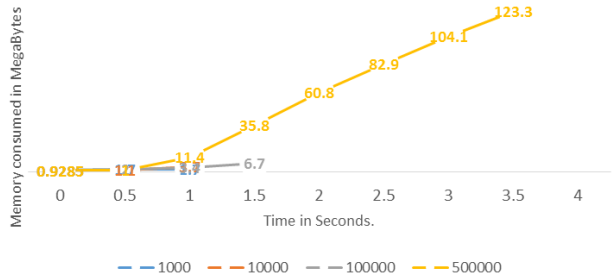

Figure 2 MQTT Memory Consumption 1000 Clients w/ different No. of messages sent.

VI. Conclusion and Limitations

Our proposed model offers a comparative analysis based on shared attributes and features. A limitation of this model is the exclusion of non-standard protocol features. For example, MQTT allows clients to register a message that is automatically sent to everyone concerned if that device goes offline (Last Will and Testament). This can be very useful in remote health monitoring scenarios (automated medical alerts). A vital goal of this discussion was to bring more attention to the non-performance metrics (soft metrics) of IoT protocols. These soft metrics play as much of a role towards a protocols' adoption as performance results. The comparison

can be further extended to include queuing protocols such as AMQP and DDS. To further the design choices and need assessments, such qualitative models should be developed for message formats (data interoperability) as well.

APPENDIX A

1. The default mode of operation is publish-subscribe which is inherently multicast.
2. The resource-observe feature can be used as a one-to-many messaging pattern. The feature of dedicated multicasting exists, but that mode does not support message retransmission.
3. Default HTTP protocol does not support multicast operations but supporting technologies like RSS and ATOM exist to provide that functionality.
4. Supports both one-to-one communication and one-to-many plus many-to-many like a PubSub protocol.
5. There is no sophisticated congestion control, but messages can be stored on the broker for later delivery in case of network errors.
6. A simple connect message is sent, and the connection is established upon reception of an acknowledgement from the broker. This is a very minimal complexity. What increases the complexity is the underlying TCP.
7. Minimal signal traffic is generated if QoS 0 is used and ping request time is set to highest. In the case of QoS 2 and lower ping times the signaling traffic generated can be higher than the actual data being sent (up to 3 extra acknowledgement messages per published message).
8. Very minimal signaling traffic is generated, especially when using the observe option and non-acknowledged messages.
9. 1 point for communication complexity as it relies on UDP. 1 to 2 points for actual messaging based on whether confirmable messages are being used or non-confirmable ones.
10. 2 points for communication complexity [i], 2 for the remaining steps required for establishing the connection (CON and CONACK messages). In case of SSL, the complexity reaches the highest level of 5.
11. No discovery mechanisms are available in MQTT. However, MQTT-SN has discovery mechanisms.
12. CoAP uses the well-known resource path /.well-known/scheme of HTTP for discovery and the Web Linking defined in RFC 5988.
13. Caching per the definition of the word is not provided but the protocol offers message retaining on the broker side, last known 'good' value can be forwarded to the new subscribers.
14. MQTT relies on three QoS levels (QoS 0, QoS 1 and QoS 2).
15. Yes, by introducing acknowledgements.
16. Yes, via retransmission if messages acknowledgements are not received.
17. If validity and freshness information is provided in a CoAP message, CoAP can use a previous message to serve future requests.
18. The Blockwise method of transmission in CoAP enables large sets of data to be transmitted in smaller chunks.
19. Only 2 bytes header size.
20. 4 bytes header size, plus a separate field called TLV (Type length value).
21. The HTTP headers size can fluctuate anywhere from 100-200 bytes to up to 2KB.
22. Payload is format agnostic and is handled as BLOB (binary large object).
23. XML based payload formatting is supported.
24. Supports Plain text, XML, HTML, JSON and many other media formats & encodings.
25. XML and EXI.
26. RFC2617, plaintext username and password.
27. Since HTTP version 1.1, called "chunked" transfer.
28. Offers delayed delivery.
29. Each message must be formatted as an XML Stanza.
30. XEP-0047: In-Band Bytestreams.

REFERENCES

[1] A. Hornsby and E. Bail, "µXMPP: Lightweight implementation for low power operating system Contiki," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*, 2009, pp. 1–5.

[2] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," *IEEE SCVT 2013 - Proc. 20th IEEE Symp. Commun. Veh. Technol. BeNeLux*, pp. 0–5, 2013.

[3] E. P. Frigieri, D. Mazzer, and L. F. C. G. Parreira, "M2M Protocols for Constrained Environments in the Context of IoT : A Comparison of Approaches," *XXXIII Brazilian Telecommun. Symp.*, no. April 2016, pp. 1–4, 2015.

[4] V. Gazis *et al.*, "A survey of technologies for the internet of things," in *IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference*, 2015.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[6] Y. Chen and T. Kunz, "Performance evaluation of IoT protocols under a constrained wireless access network," *2016 Int. Conf. Sel. Top. Mob. Wirel. Networking, MoWNeT 2016*, 2016.

[7] M. P. Kayal and H. Perros, "A Comparison of IoT application layer protocols through a smart parking implementation," pp. 331–336.

[8] H. V. Wallis and A. F. Skarmeta, "Semantic Web of Things : an analysis of the application semantics for the IoT moving towards the IoT convergence Antonio J . Jara *, Alex C . Olivieri and Yann Bocchi Markus Jung and Wolfgang Kastner," vol. 10, pp. 244–272, 2014.

[9] J. Dede and A. Förster, "Comparative Analysis of Opportunistic Communication Technologies BT - Interoperability, Safety and Security in IoT," 2017, pp. 3–10.

[10] K. Khan, A. Waris, and H. Safi, "A Qualitative Comparison of Various Routing Protocols in WSN," vol. 1, no. 1, pp. 7–13, 2016.

[11] P. H. Su, C. S. Shih, J. Y. J. Hsu, K. J. Lin, and Y. C. Wang, "Decentralized fault tolerance mechanism for intelligent IoT/M2M middleware," in *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, 2014, pp. 45–50.