Poster Abstract: Design of an Energy-Efficient End-to-End Messaging Protocol for Smart Cities

Shang Chih Chung
Dept. of Computer Science and Information Engineering
National Taiwan Normal University
Taipei City, Taiwan R.O.C.
60847091S@ntnu.edu.tw

Chao Wang
Dept. of Computer Science and Information Engineering
National Taiwan Normal University
Taipei City, Taiwan R.O.C.
cw@ntnu.edu.tw

ABSTRACT

Internet-of-Things (IoT) applications at the city scale require a reliable and energy-efficient messaging service. In order to save energy consumption at each embedded IoT instrument, thus to prolong the application lifetime, it is best that the service be free of redundant messages. In this paper, we analyze the MQTT messaging protocol and show that there are three problems that may either cause message losses or produce redundant messages. For each problem we propose a solution, and we show that our design as a whole may be implemented efficiently. We have been implementing the design, and it is our hope that the messaging protocol as a result would be a good support to many smart-city applications.

CCS CONCEPTS

• **Networks** → *Network protocol design.*

KEYWORDS

Internet of Things, Messaging Middleware, Energy Efficiency

ACM Reference Format:
Shang Chih Chung and Chao Wang. 2021. Poster Abstract: Design of an Energy-Efficient End-to-End Messaging Protocol for Smart Cities. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '21), May 18–21, 2021, Charlottesvle, VA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450268.3453508

1 INTRODUCTION

Energy efficiency plays a key role in the Internet-of-Things (IoT) applications, because many embedded IoT devices are only powered by batteries or energy-harvesting modules. Typically, as wireless transmissions dominate the overall energy consumption of an IoT device, it is critical to improve the reliability of IoT message delivery and reduce unnecessary message retransmissions. Smart city applications in particular, with its vast number of embedded IoT devices deployed at urban and suburban areas, call for the need for a reliable and energy-efficient messaging service. In this paper, we report our on-going development of such a service.

An example smart city application is smart trash cans [3], where IoT devices attached to the bins would report the current load of the bin via a messaging service. Accordingly, garbage trucks and/or back-end servers that subscribe to the message topic may plan a better route for garbage collection (Figure 1). In applications like this, most IoT devices run a relatively low duty cycle, and the sending of messages would account for the major energy consumption of the device. For example, a wireless device at the sending of message may take hundreds of milliwatts, while at the sleep mode it may take fewer than ten milliwatts [1]. Based on this observation, it is appealing if an IoT messaging service can provide a reliable message delivery without incurring redundant message retransmissions.

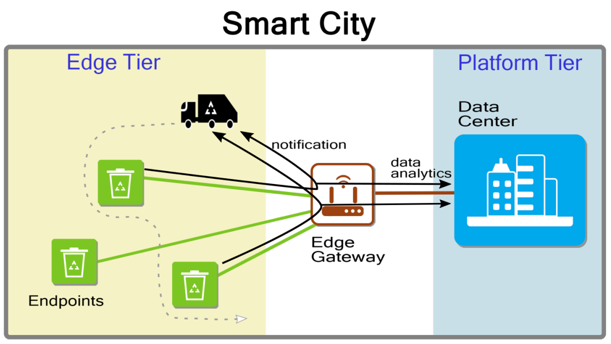

Figure 1: Example smart city messaging scenario.

MQTT is a widely used publish/subscribe messaging protocol for IoT applications [2]. By communicating via a messaging server, also known as a *messaging broker*, each MQTT client does not need to know the existence of others and can still send messages through the broker. But, as we will show in the following analysis, such a broker-based messaging protocol may lead to message losses or unnecessary retransmissions.

2 PROTOCOL ANALYSIS AND DESIGN

We organize our findings into three categories, and in each one we first present our analysis of a potential problem, followed by our solution proposal. Putting together, the main goal is to reduce unnecessary message retransmissions to save energy. We call the resulting protocol *E4*, which is energy-efficient and offers end-to-end quality of service.

Problem 1: Pairwise QoS. In order to meet the quality of service (QoS) needed by different applications, MQTT offers three levels of QoS for message delivery: *at most once* (QoS 0), *at least once* (QoS 1), and *exactly once* (QoS 2). Publishers and subscribers of the same topic may request different QoS levels, and the MQTT service will consolidate the requests. But because the MQTT specification only requires a pairwise enforcement of QoS levels (i.e., from

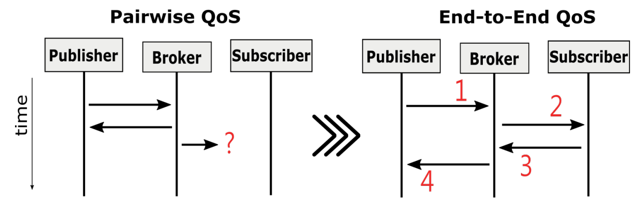

Figure 2: Pairwise QoS vs. end-to-end QoS.

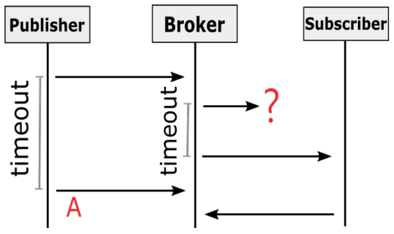

Figure 3: An example of premature timeouts.

publisher to broker and/or from broker to subscriber), depending on the implementation, a MQTT service may not provide a reliable message delivery end-to-end (i.e., from publisher to subscriber). For example, for QoS 1 enforced at both the publisher-broker pair and the broker-subscriber pair, the broker receiving a message may immediately return an acknowledgment to the publisher before sending out the message to the subscriber (Figure 2). In this case, the publisher may delete its local copy of the message and eventually the message may not be delivered to the subscriber, for example, due to networking problem between the broker and the subscriber.

Solution 1: End-to-end QoS. In the E4 protocol, we specify that the broker should send an acknowledgment only after it has received the acknowledgment from the subscriber. Figure 2 shows the sequence of this four-stage message exchange. We use timers to control the retransmission of messages. The timers' interval must satisfy the following condition:

$$\begin{cases} T_p > \sum_{i=1}^{4} t_i & \text{for a publisher;} \\ T_b > \sum_{i=2}^{3} t_i & \text{for a broker.} \end{cases} \tag{1}$$

where t_i denotes the one-way network latency for stage i; T_p, the interval of the publisher's timer; T_b, the interval of the broker's timer. In general, one should use a longer interval to prevent accidental timeouts, as it would take some time for a message receiver to respond and process the message.

Problem 2: Premature timeouts. Should the broker choose to enforce an end-to-end QoS, a delay at the broker/subscriber may cause a premature timeout at the publisher (Figure 3), which in turn may unnecessarily consume energy to perform a redundant message retransmission (marked by A in the figure). The delay may be due to a message loss or network traffic congestion at the channel between the broker and the subscriber.

Solution 2: Deadline extension. In the E4 protocol, in order to prevent the premature timeout at the publisher, the broker may either use a shorter interval for its own timer or request the publisher to temporally extend its timer interval. The latter approach may be more effective, because using a shorter timer interval at the broker might not resolve the cause of the delay developed at the

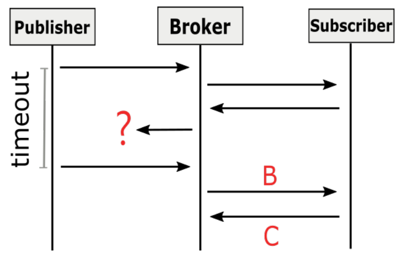

Figure 4: An example of unnecessary message exchange.

channel between the broker and the subscriber. The solution may save publisher's energy consumption since, by reducing unnecessary retransmissions, the end device may stay in the sleep mode longer.

Problem 3: Unnecessary message exchange. Sometimes, an acknowledgment from the broker may be lost and/or there may be some transient delay. As a result, upon timeout the publisher may retransmit the same message even if the broker had already delivered it (Figure 4). This would cause the subsequent message exchange unnecessary (transmissions B and C).

Solution 3: Bookkeeping. In the E4 protocol, upon receiving the same message, the broker will directly resend the acknowledgment and will not forward the duplicated message to the subscriber: from the subscriber's aspect, this message is unnecessary; from the publisher's aspect, receiving the acknowledgment early would prevent another retransmission. To identify a message duplication there are two approaches. The DUP flag in the MQTT packet header may be used for this purpose but, as noted in the MQTT specification, a packet with DUP flag set to 0 may contain a repetition of application message [2]. Instead, we propose to have the broker record the received messages and use that information to identify duplicated messages.

3 FUTURE WORK

In this paper, we described potential problems in IoT messaging as well as their solutions, which together form what we called the E4 messaging protocol, to offer energy-efficient end-to-end reliable message delivery. In order to empirically validate the E4 protocol, we will implement a smart city scenario prototype and will verify the performance of E4 through experiments and power monitoring.

ACKNOWLEDGMENTS

This work is supported by MOST grant 109-2222-E-003-001-MY3. We would like to thank reviewers for helping improve this work.

REFERENCES

[1] M. Magno, F. A. Aoudia, M. Gautier, O. Berder, and L. Benini. 2017. WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017.* 1528–1533. https://doi.org/10.23919/DATE.2017.7927233
[2] OMG. Standard. 2019. Message queuing telemetry transport, version 5. (2019). https://mqtt.org/
[3] Y. Zhu, G. Jia, G. Han, Z. Zhou, and M. Guizani. 2019. An NB-IoT-based smart trash can system for improved health in smart cities. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC).* 763–768. https://doi.org/10.1109/IWCMC.2019.8766748