The Impact of MQTT-based Sensor Network Architecture on Delivery Delay Time

Oleksandr Kovalchuk
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
me.olexandr.kovalchuk@gmail.com

Yuri Gordienko
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
yuri.gordienko@gmail.com

Sergii Stirenko
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
Kyiv, Ukraine
sergii.stirenko@gmail.com

***Abstract*—The purpose of this study is to present two new architectures of the complexes of embedded systems in the context of the Internet of Things. The proposed new schemes use the MQTT protocol but provide both an autonomous operation and an online mode. This is achieved by adding one additional layer, which consists of its own MQTT-broker and critical messages handler so that in case of the Internet access disappearance, the system could continue to respond to changes. The research of message delivery delays has been carried out for the proposed models. The comparative analysis was conducted to identify the advantages and disadvantages of both solutions.**

Keywords—Internet of Things, embedded systems, MQTT protocol, architecture of communication

I. INTRODUCTION

The Internet of Things (IoT) is a network concept consisting of interconnected physical devices that have built-in sensors. An obligatory component is also the software that allows the transmission and exchange of data between the physical world and computer systems using standard communication protocols. In addition to sensors, the network may have actuators built into physical objects and interconnected via wired or wireless networks. These interconnected devices have the ability to read and operate on the data, identify themselves and be programmed. They also exclude the need for human involvement by the use of intelligent interfaces [1].

Internet of Things is also defined as a "smart environment", based on the constant proliferation of intelligent networks, wireless sensors and mass data centers [2]

The Internet of Things is also known as the Cloud of Things (CoT). In general, IoT and CoT differ from each other, but their features often complement each other. That is why many researchers advocate their integration in order to benefit from specific scenarios and in the wider context of Cloud-Fog-Dew computing paradigm [3-5].

An important direction of IoT development is the search for a convenient way of interacting these systems, which provides reliability, convenience, and integration with many types of devices. The presented ways of the interaction of embedded systems define the novelty of this work.

II. RELATED WORKS

With the growing popularity of IoT, several protocols are being widely used to develop solutions in this area. Performance comparisons between most popular implementations of CoAP, MQTT, DDS protocols are presented in [6]. In [7] authors evaluate performance of the system, which uses MQTT, changing the quality of service (QoS) policy. With QoS 0 they experienced approximately 85% package loss, and no package loss was experienced with QoS 1 and 2. While QoS 2 guaranteed no duplicate messages was received it required 45% more time to deliver messages compared to using QoS 1.

In [8] the authors focus on handling disruptions without data loss due to unstable network links. An implementation of the bundle protocol for WSN specifically for Contiki is described. In [9] authors propose the architecture to increase the robustness of MQTT by integrating a Disruption Tolerant Network approach. The proposed architecture has been evaluated through several experiments using real devices to validate its feasibility.

A different approach is used in [10], where authors investigate the potential benefits of using QUIC compared to TCP and UDP as the underlying transport layer for MQTT. Experiment results confirmed that MQTT with QUIC as transport protocol reduces connection establishment overhead, reduces memory utilization compared to the MQTT with TCP. However, the proposed solution has some drawbacks, such as higher processor utilization, which is required for encryption, and packet processing while maintaining QUIC internal state.

III. BACKGROUND

MQTT, which stands for Message Queue Telemetry Transport, has proved itself as a reliable protocol that has a small footprint on message data size. Despite not being a new protocol, it is actively developed: the latest version of the protocol standard has been released in October 2018 [11].

MQTT is a publish/subscribe protocol, which means in systems based on this protocol there are producers, which publish data to the topics and consumers, which subscribe to the topics with the data they are interested in. There is also the broker entity in the system: MQTT server that receives messages from publishers and delivers them to corresponding subscribers based on the topics and quality of service policy. Publish/subscribe approach allows to decouple implementations of producer and consumer and add or replace components of the system as long as data format and topics names are preserved. This gives the flexibility to the system, as its components do not depend on each other's implementation, but only depend on data they provide [11].

978-1-7281-2065-2/19/$31.00 ©2019 IEEE

The simplified scheme of interaction in the MQTT-based systems can be described as follows. Producer or publisher connects to MQTT broker. Consumers, which are also called subscribers connect to the same MQTT broker and subscribe to the topics they are interested in. When the producer has information to share, it publishes a message to the broker, which then shares the information to all subscribers of this topic.

Message Queue Telemetry Transport also provides quality of service (QoS) policies, which provide different guarantees at the cost of delivery slowdown and network traffic increase. Slowdowns are minimal when using QoS 0 (at max once), however, this policy does not guarantee that message will be delivered after it was sent. QoS 1 (at least once) guarantees that message will be delivered to the destination at least once, however, duplicate messages may occur. The most strict policy is QoS 2, which guarantees exactly one message delivery, which comes at cost of at least three additional messages being sent over the network, and thus larger delays. The latter is used in the system, where delivery is critical, and message duplicates are not allowed [7]. In this investigation, QoS 1 is used as the policy, which provides an optimal balance between delivery reliability and footprint it has for most sensor networks.

IV. Experimental and Computational Details

One of the ways to maintain the autonomy of the system is to transfer part of business logic and communication to the sensor network itself instead of using only a public broker. In this case, the part of the functionality should still be present in the external network, so that it can be accessed from the outside of the sensor network (for example, for users to receive statistics and alerts on their smartphones). Having external brokers also allow more powerful computational resources to access sensor networks data for processing.

A. Proposed Alternative Architectures

According to the above, we propose two alternative architectures to consider. Both proposed architectures have an additional layer of MQTT broker and critical message handler inside the sensor networks. Proposed architectures allow keeping system operating even when Internet access is lost. First of the two proposed architectures has a lightweight copy of the whole system in every sensor network (Fig. 1). This allows keeping sensor nodes logic clean and handling network loss separately.

On the other hand, the second of the proposed architectures (Fig. 2) requires sensor nodes to check for connectivity and handle Internet access loss by themselves. This means sensor nodes should keep both connections to a local broker and external broker. This also means sensor nodes should also carry about which data is critical for system operations, and which is not. As the solution does not indicate the connection between the local client and the remote broker, the messages sent to the local broker at the time the channel is broken will not go to the remote one. While this aspect mean data loss, the presence of local broker allows keeping sensor network operational at the cost of losing some additional functionality provided by the external clients.

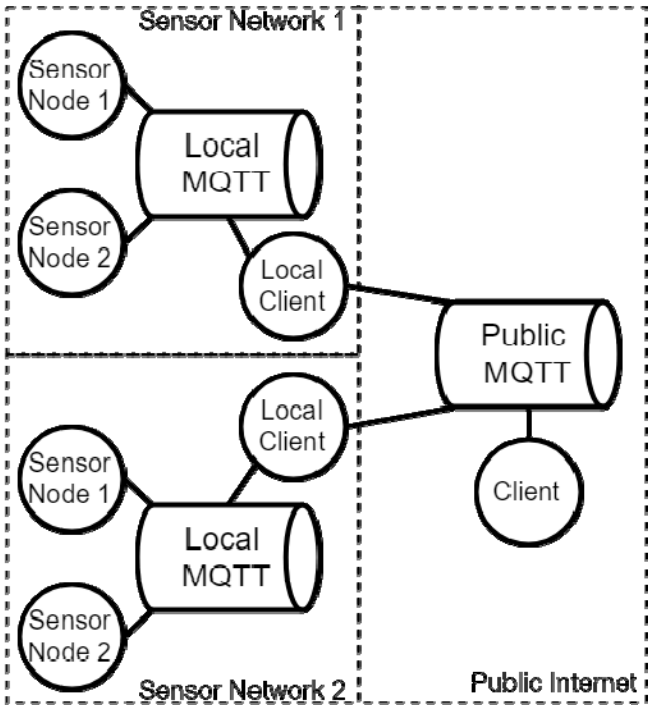

Fig. 1. Proposed alternative two-layer architectures which retransmit data to the public cloud.

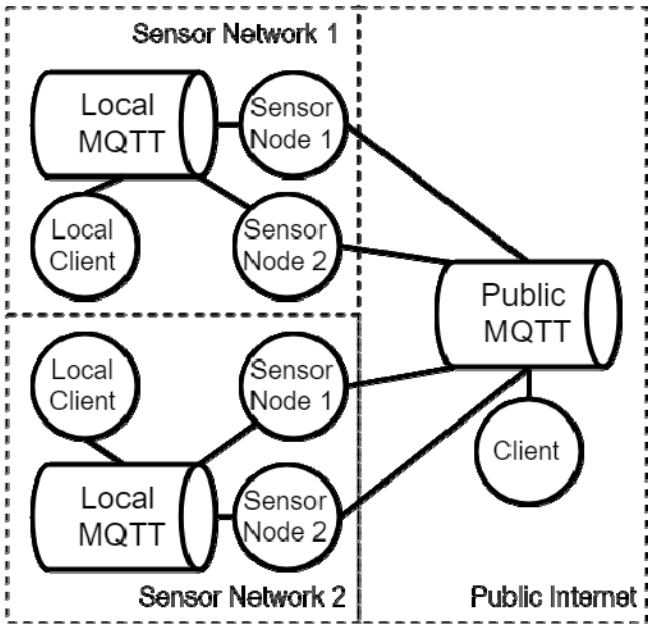

Fig. 2. Proposed alternative two-layer architectures in which sensor nodes handle connectivity checks.

B. Message delivery delay investigation

One of the key characteristics of the embedded systems is their response time. Thus, it is necessary to investigate message delivery delays in order to evaluate the effectiveness of the proposed architectures and compare them to the existing ones.

For the research and measurements, we use ESP8266 chip due to its availability and popularity. For this chip, the sensor imitation, which generates data and sends them every 5 seconds, is implemented. For the ease of investigation, we use sequential numbers as the payload of messages. This payload is “processed” with the external client, which sends

it to the different MQTT topic, to which the sensor is subscribed. Sensor node tracks the time when the message was sent and when the corresponding response has been received. Thus measuring the difference between two timestamps will give us the roundtrip time in the system.

In order to eliminate the problem of time synchronization between multiple devices and achieve the maximum accuracy, we perform all measurements with only one chip. Sensor node logic is implemented using C++ programming language and uploaded with platformio tooling.

For time measurements, built-in function micros() which return microseconds since device start is used. As mentioned above, we use QoS 1 (at least once), as this policy gives the optimal balance between message delivery guarantees and protocol data overhead. As QoS 1 allows duplicate messages to be sent/received, we measure only time between sending data and first "response". All the following duplicates are ignored.

For this investigation, we use two MQTT-brokers. External platform io.adafruit.com was used for hosting public MQTT broker. We used eclipse mosquitto (open source implementation of the broker, which supports protocol standards 3.1.1 and 3.1) for setting up local MQTT broker. The external platform uses a secure connection (over TLS) and user authentication; we do not use those in the local broker. This decision is motivated by means of ease of configuration and reducing network overhead as the unauthorized access to the broker can be prevented by properly configuring network itself.

The Internet is accessed using LTE connectivity with the download speed of 2.4Mbps and upload speed of 8.1Mbps. These network characteristics were obtained using speedtest.net service. Local connectivity was implemented using a Wi-Fi network.

For measurements, we use a simple NodeJS application, which only retransmits messages from one MQTT topic to another, thus giving the minimal impact on delays. In order to retransmit messages from local broker to public and vice-versa another NodeJS application was implemented. The latter accepts two sets of topics: ones that should be retransmitted from local broker to external and ones that should be retransmitted from external broker to local. As MQTT 3.1.1 does not support user properties in messages, there is no way to identify message sender. Due to this fact, sets of topics should not intersect; otherwise, a message retransmission flood (infinite retransmission of the same message from local to public broker and vice-versa) will occur.

V. Results

Both architectures were investigated in two different scenarios: in the first scenario there was constant good connectivity to the Internet and in the second, Internet connection was unavailable (while the local network was still working) for the time of sending a single message. For comparison, the same measurements were done for the two single-layer architectures. The first of single layer architecture uses only public MQTT broker (Fig 3), and thus cannot operate in case of Internet connection loss. The second one uses only a local broker (Fig. 4) and has no Internet connection at all, thus making it impossible to receive sensor network data from the outside. The comparison of two-layered systems performance against public-broker only solution is done to investigate the delay which is caused by introducing intermediate layer and retransmission logic into the system.

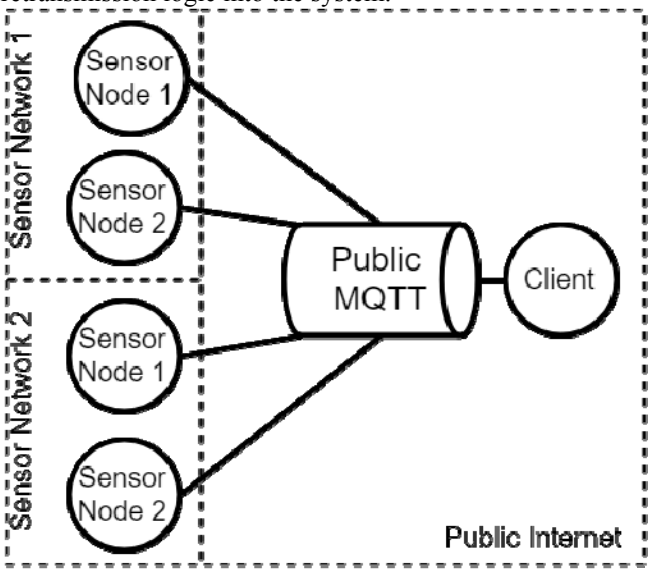

Fig. 3. Single layer architecture with public broker only.

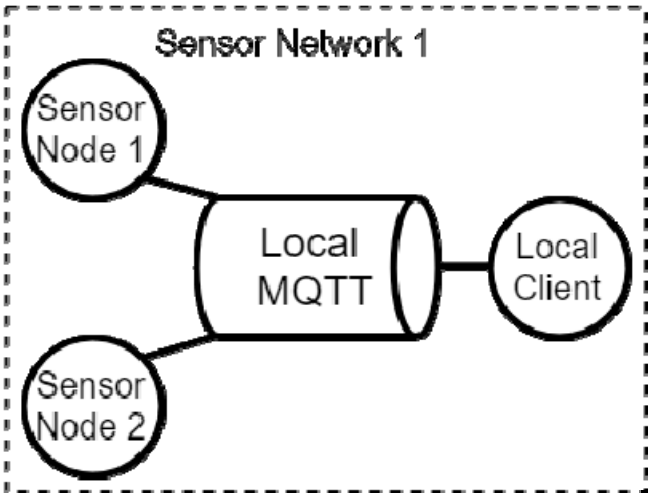

Fig. 4. Single layer architecture with local broker only.

For each case, 20 measurements were performed and average message roundtrip delays were calculated. The results of the measurements (in microseconds) are shown in Table I.

TABLE I. Measurement Results (In Microseconds)

System architecture	Measurements		
	Total time, µs	*Average time, µs*	*Standard deviation*
External broker	1.25E+07	6.23E+05	1.07E+05
No. 1	1.53E+07	7.64E+05	1.70E+05
No. 2	1.20E+07	6.01E+05	1.44E+05
No. 1, Internet connection loss	1.41E+07	7.06E+05	2.12E+05
No. 2, Internet connection loss	1.59E+07	7.97E+05	1.01E+06
Local broker	6.35E+06	3.17E+05	1.44E+05

The comparison of average message delivery delay time is shown in Fig. 5. Visualization based on values from Table I allows comparing average delays when delivering messages between the proposed two-layer architectures as well as to compare those to the single-layer architecture.

From the chart, we see that local broker has the least average delays. Both proposed architectures, in general, have bigger delays for message roundtrips than standard solutions with only one broker. While with the constant stable internet connection the proposed architecture No. 2 shows better results, than the architecture No. 1, the situation is opposite when Internet connection loss occurs.

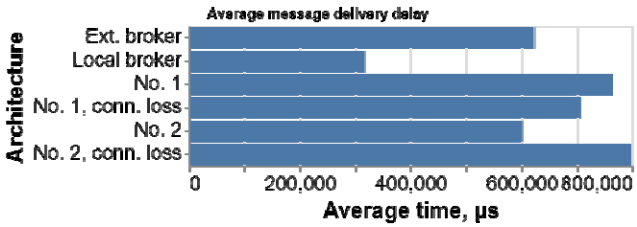

Fig. 5. Comparison diagram of average message delivery delay.

Having standard deviation calculation we can investigate actual measurements deviation from the average value (Fig. 6). As we can see, the standard deviation is the biggest for architecture No. 2 when internet connection loss occurs. This means that delays caused by Internet connection have a significant impact on the message delivery delays in this case.

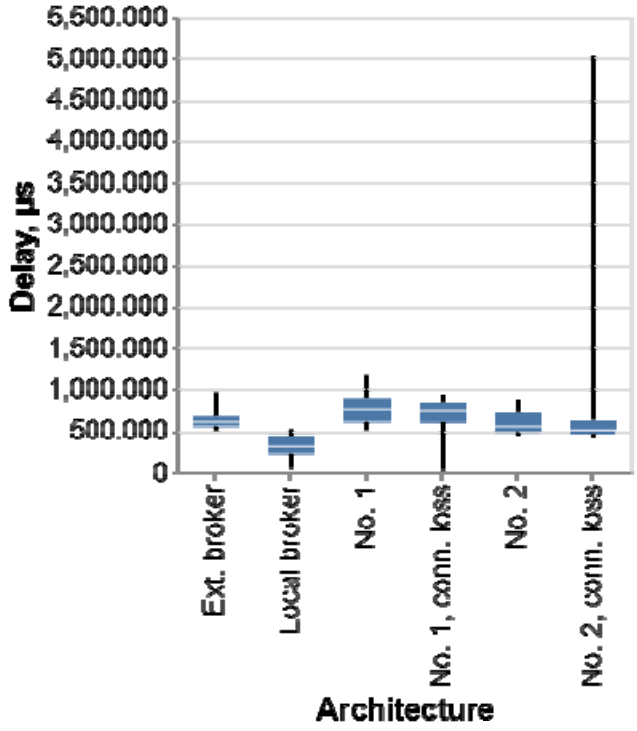

Fig. 6. Statistical analysis of the observed results.

Average message delivery delay time compared to the ones of one-layered architecture are shown in Table II. We can observe that in conditions of stable internet connection, proposed architecture No. 2 has the least delay increase compared to only local and only external broker architectures. In case of Internet connection loss, architecture No. 2 has 91 milliseconds longer delays. The local broker solution is the fastest amongst all and compared to the external broker is faster by 306 milliseconds.

TABLE II. MESSAGE DELIVERY DELAYS COMPARISON

System architecture	Average time, ms	Compared to external broker, ms	Compared to a local broker, ms
External broker	6.23E+02	0	3.06E+02
No. 1	7.64E+02	1.41E+02	4.47E+02
No. 2	6.01E+02	-2.18E+01	2.84E+02
No. 1, Internet connection loss	7.06E+02	8.28E+01	3.89E+02
No. 2, Internet connection loss	7.97E+02	1.74E+02	4.80E+02
Local broker	3.17E+02	-3.06E+02	0

The proposed architectures solve the problem of both online and offline work of the sensor network by adding an additional layer, which consists of local MQTT broker. Proposed solution No. 1 does not require sensor logic modification and solution No. 2 allows offloading computations to external network completely if it is available.

VI. DISCUSSION

Results demonstrate (Table I) that message delivery delays are the lowest for the architecture with the local broker only. This approach cannot always be used, as it does not provide the possibility to access data from outside the sensor network.

As for the delays when Internet connection loss occurs, an average delay time is increased in both two-layer architectures. It is caused by the implementation details: by default, connection status is checked every 5 seconds and this time was spent until connection loss was noticed and the system switched to the local broker. Connectivity check timeout can be configured, thus opening the field for further investigations.

The average message delay time decrease, which was observed using proposed architecture No. 2, was insignificant and probably caused by the mobile network load.

The research showed that proposed architecture No. 2 does not have overhead when a stable Internet connection is available compared to the external broker architecture. However, a significant delay occurs when the Internet connection is lost.

Some aspects, such as deeper analysis of systems' performance in lossless and lossy environments, investigating the impact of different QoS policies and durable connections usage on the delivery delay time, were out of the scope of our current work at this stage. However, these aspects could have a crucial impact on the results obtained and that is why they will be investigated on the next stage of our research.

Based on the observations the effective solution for practical application can be proposed. The smallest delays will be achieved by moving most of computations and interactions to the local network by communicating using

local MQTT broker. All data, which is required outside the sensor network can be retransmitted to the external broker, as well as all commands from the external network can be retransmitted to the local MQTT as in the proposed architecture No. 1. In general, these results support the general trend of IoT and CoT integration in the wider context of Cloud-Fog-Dew computing paradigm [4-5].

VII. Conclusions

We proposed alternative two-layer architectures and investigated them. We discovered the advantages and disadvantages of both architectures. Based on the investigation results, we proposed the more optimal two-layer architecture that combines the best parts of two investigated solutions. While having approximately 13% longer delays than public broker architecture when communicating outside the sensor network, the proposed solution has no additional overheads when all communications occur within the sensor network due to local broker usage.

It is necessary to keep in mind that sending message to local broker is always faster than sending it to the external one, thus in order to achieve better performance and lesser delays as many communications as possible should be performed using local broker only. Messages should be retransmitted to external broker in cases, when the external application or handler needs those.

References

[1] E. T. Chen,. "The Internet of Things: Opportunities, Issues, and Challenges" in The Internet of Things in the Modern Business Environment, IGI Global, 2017, pp. 167-187.

[2] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications" in Journal of Electrical and Computer Engineering, 2017

[3] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities" in IEEE Internet of Things Journal, vol 3(6), 2016, pp.854-864.

[4] Yu. Gordienko, S. Stirenko, O. Alienin, K. Skala, Z. Soyat, A. Rojbi, J. R. Benito, A. G. González., U. Lushchyk, L. Sajn, A. Llorente Coto, and G. Jervan. "Augmented Coaching Ecosystem for Non-obtrusive Adaptive Personalized Elderly Care on the Basis of Cloud-Fog-Dew Computing Paradigm" [Proceedings 40th Int. Convention on Information and Communication Technology, Electronics and Microelectronics, IEEE, pp.359-364, 2017]

[5] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable Distributed Computing Hierarchy: Cloud, Fog and Dew Computing", in Open Journal of Cloud Computing (OJCC), vol. 2, Issue 1, 2015, pp.16-24, DOI:10.19210/1002.2.1.16.

[6] Y. Chen, and T. Kunz "Performance Evaluation of loT Protocols under a Constrained Wireless Access Network", [Proceedings International Conference on Selected Topics in Mobile Wireless Networking (MoWNeT), 2016]

[7] S. R. Akbar, K. Amron, H. Mulya and S. Hanifah, "Message queue telemetry transport protocols implementation for wireless sensor networks communication—A performance review." in Sustainable Information Engineering and Technology (SIET), IEEE, November, 2017, pp.107-112.

[8] G. V. Zengen, F. Biisching, W-b. Pottner and L. Wolf (2012, September) "An Overview of DTN : Unifying DTNs and WSNs", [Proceedings 1Ith GUITG KuVS Fachgesprach Drahtlose Sensornetze, September 2012].

[9] J. E. Luzuriaga, M. Zennaro, J. C. Cano, C. Calafate, and P. Manzoni, "A disruption tolerant architecture based on MQTT for IoT applications", [Proceedings 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2017] DOI:10.1109/ccnc.2017.7983084.

[10] P. Kumar, and B. Dezfouli, "Implementation and analysis of QUIC for MQTT" in Computer Networks, vol 150, 2019, pp.28-42, DOI: 10.1016/j.comnet.2018.12.012

[11] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, "MQTT Version 5.0. Candidate OASIS Standard 01", 31 October, 2018. Accessed 2018-11-20.