Azure IoT for ESP32

Gonçalo Pereira
Distributed Systems, MIEEC
Fac. Engineering, University of Porto
Porto, Portugal
goncalo.p@fe.up.pt

João Pedro Santos
Distributed Systems, MIEEC
Fac. Engineering, University of Porto
Porto, Portugal
up201504545@fe.up.pt

Paula A Graça
Distributed Systems, MIEEC
Fac. Engineering, University of Porto
Porto, Portugal
paula.graca@fe.up.pt

***Abstract*—This project focuses on the implementation of an MQTT communication system between ESP32 endpoint devices and Microsoft Azure IoT. Afterward, experiments were executed in order to analyze several predefined benchmarks, namely round-trip time, payload length and number of devices, and their impact on the overall performance of the system. Finally, it was concluded that the real environment tests executed are not sufficient to extract definite conclusion about the effective system response, when taking full advantage of Azure IoT capabilities.**

I. Introduction

The increasingly necessity of turning everything accessible and automated nowadays, comes with different interests while developing systems. Internet of Things is one of the most epidemic subjects of the technological world and the advantages of this concept are well recognized. Therefore, the number of IoT devices are growing rapidly which brings concerns about security, scalability and robustness.

IoT devices are meant to be used for specific communications, which can cause some critical security threats. It is possible to use firewalls and software to protect the equipment, but the whole idea behind Internet of Things is that these low power, no-frills devices are what is being deployed, so there is not a big effort to take care of issues related to that. Azure IoT Hub implements a service assisted communication methodology and this mediates interaction between backend systems and devices. This way, there is a bi-directional, trust worthy communication set up. IoT Hub's capabilities allow building scalable, full-featured IoT solutions such as managing industrial equipment used in manufacturing, tracking valuable assets in healthcare, and monitoring office building usage.

The purpose of this project is to implement a system that allows several ESP32 endpoint devices to communicate with each other, by establishing a connection with the Azure IoT Hub. To do so, the MQTT protocol is used, which is highly adequate for resource-constrained IoT networks and uses a publisher-subscriber pattern. The performance of this system was evaluated taking into account some benchmarks, such as, round trip time, payload and number of devices. The increase in payload was found to have no significant impact and the device number experiment was inconclusive.

This document is organized in four sections: section 2 contains theoretical principles and implementation details; section 3 is related to the experiment results and finally, conclusions are outlined in section 4.

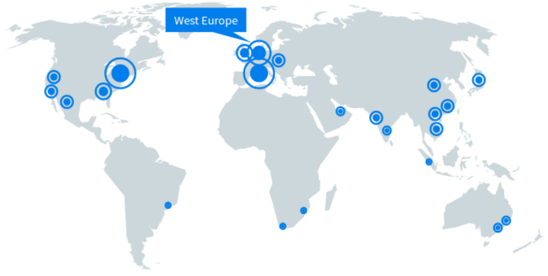

Fig. 1. Azure IoT Hub servers' locations

II. Theoretical and implementation description

A. *Microsoft Azure Iot Hub*

Microsoft Azure is a cloud service that is used to build, deploy and manage applications. IoT Hub is a managed service, hosted in the cloud, that acts as a central message hub for bi-directional communication between your IoT application and the devices it manages. The IoT Hub behaves on the implemented system as a broker since it is its function to organize, transform and route messages from publishers to subscribers.

In order to ensure minimum latency, the hub located in West Europe was chosen (Fig. 1). However, although distance is one of the main decision factors of the hub to be used, the nearest hub is not always chosen due to legislative restrictions for certain applications.

B. *Publisher-Subscriber*

A distributed system is composed of two main components: the communication system and its computing nodes. In order to exchange information in the system, the computing nodes can use a range of different cooperation models. One example of this is the Publisher-Subscriber model. In this model, communication is performed by means of topics, in which:

- One or more publishers generate information and trigger transactions (i.e. disseminates information);
- One or more subscribers consume the information sent by the publisher.

MQTT protocol is a publisher-subscriber middleware designed to be lightweight in order to run on devices with lower resources such as the ESP32. Also MQTT is a binary protocol, which result in more compact payloads than HTTPS.

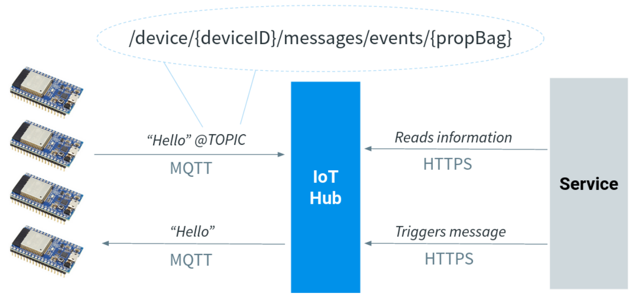

Fig. 2. Implemented solution

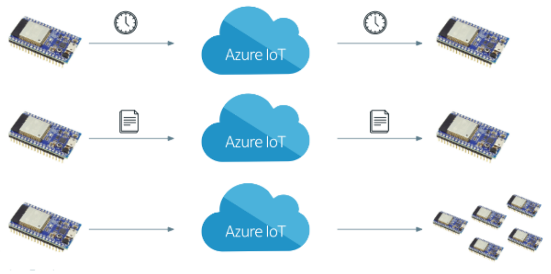

Fig. 3. Benchmarks: (top to bottom) Round-Trip time; Payload length; Number of devices

C. *Azure MQTT broker architecture*

Since Azure IoT Hub is a constrained system created by external entities to the project, it is important to define the architecture of the technology it is intended to explore. The MQTT broker on Azure IoT Hub restricts a device to publish on a specific topic bounded to himself, thus defining the notion of events. Each device is able to publish on the topic */device/deviceID/messages/events/propBag* which is defined by its ID. Therefore, Azure offers the possibility to run a service, running on the Azure or locally, in several programming languages so that message routing, filtering and any kind other kind of message treatment can be done in a costumizable manner.

D. *D2C and C2D communication*

A typical flow of a publisher-subscriber model can be described as follows: a set of devices publish a certain topic; another set of devices subscribe to that same topic; the cloud receives a message from device A and that same message is then triggered to be sent to device B, given that B is a subscriber of A's events.

In the system under study, this service was built on NodeJS using *azure-iothub* and *@azure/event-hubs* packages that allowed access to the messages sent to the cloud. By checking it's parameters, it was possible to implement a simple subscriber list and, whenever a specific device sends a new event, that message is sent to a given list of subscriber devices.

From the devices standpoint an application was developed using the *Azure C SDK* which periodically sends messages and is continually listening for incoming messages. For each message message that is received or sent the system clock was checked in order to measure delays. Since a comparison between timestamps of different devices was needed, it was key to do clock synchronization. This was done using SNTP which is similar to NTP, tailored to run on devices with lower resources.

III. Experiment Results and Performance Analysis

In order to evaluate the performance of the presented system, it is important to organize a relevant test plan so that the quantitative results can lead to pertinent conclusions about the conducted study. The experiments were executed in a defined environment and with known conditions. For instance, in every test the number of network hops was taken into consideration, consisting of a total of 9 hops from the sending device until arriving to the Microsoft.AS8075.gigapix.pt server. Additionally, all used endpoint devices were ESP32 which were clock synchronized by NTP. The three main benchmarks considered for the test experiments were the connection round-trip time, increased payload and increased number of subscriber devices, which are detailed bellow.

The measurement of transmission time for each connection consisted essentially in four main steps:

- Device publishes a message on Azure topic and stamps initial time;
- IoT Hub acts as a broker redirecting the message to a single or multiple topic's subscriber(s);
- Subscriber receives the message and stamps final time;
- Sending time and receiving times are subtracted, originating the end-to-end transmission time.

A. *Round-Trip Time*

The round-trip time of a connection, by definition, is the time it takes to execute a full transmission end-to-end plus receiving an acknowledgement back. In the system under study, this metric is slightly altered since the end-to-end communication is already basically a duplicated end-to-cloud transmission. For this reason, although it is not completely the scientifically correct definition, it was considered that the round-trip time of communication is an end-to-end transmission.
Test 1 consisted on executing end-to-end transmissions of one hundred messages of 100 Bytes each, in order to obtain a considerable mean value. The acquired measurements are represented in Figure 1. The mean time obtained was 361267 microseconds (approximately 361 ms).

In test 2, the experiment was similar to the previous described one. However in step three of the execution steps, the used subscriber was not waiting to receive the message, but instead it was busy executing other tasks to simulate a functional device. These tasks consisted on sending continuous neglected messages to the Iot Hub, just to ensure that they were actually arriving in the platform monitor. This ensured that only in between messages, the device was available to receive messages.

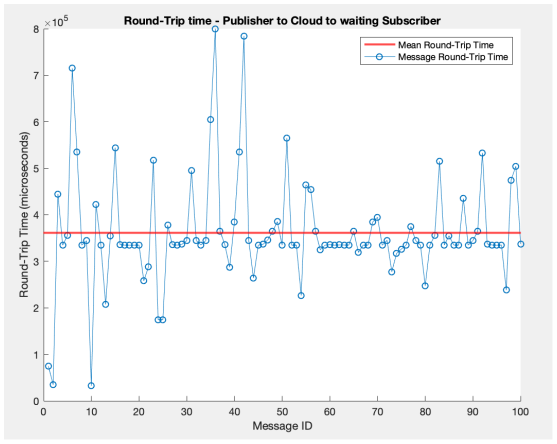

Fig. 4. Test 1: Round-Trip Time with waiting subscriber

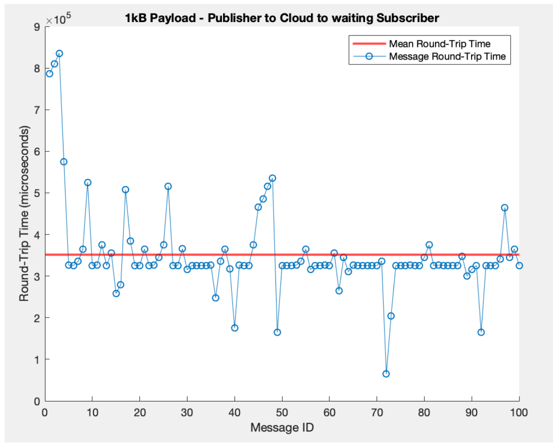

Fig. 6. Test 3: Increased payload of 1kB with waiting subscriber

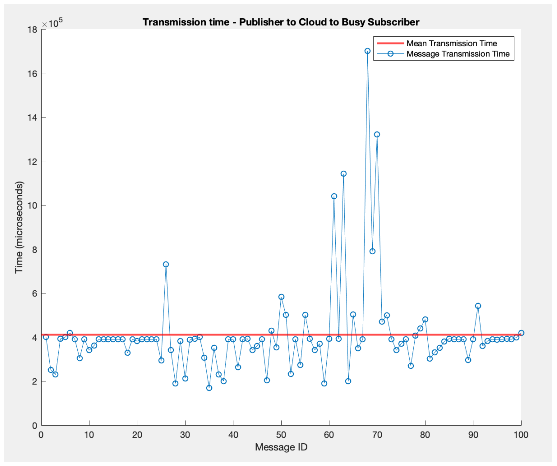

Fig. 5. Test 2: Round-Trip Time with busy subscriber

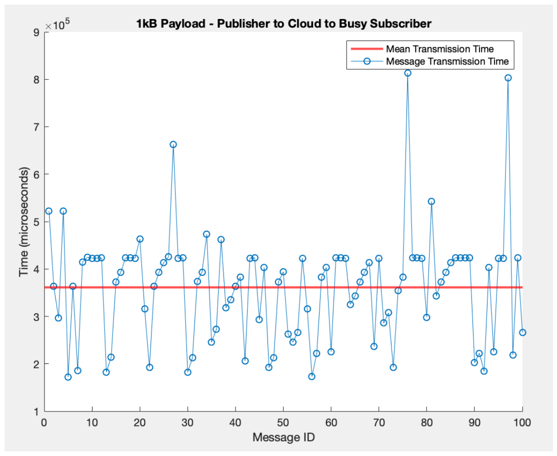

Fig. 7. Test 4: Increased payload of 1kB with busy subscriber

Test 2 consisted on executing end-to-end transmissions of one hundred messages of 100 Bytes each, in order to obtain a considerable mean value. The acquired measurements are represented in Figure 2. It is possible to deduce that the peak deviations present in some of the values can be justified as a worse communication opportunity, in which the receiver is busy executing a task and so it cannot accept the message to be delivered right away. The mean time obtained was 411472 microseconds (approximately 411 ms).

By analysing round-trip time values of tests 1 and 2, it is possible to conclude that the mean time spent transmitting messages to busy subscribers is higher than to waiting subscribers, as logically expected. However, the difference is not considerable and could be a consequence of several external factor (e.g. traffic momentaneous congestion).

B. Payload Length

When analysing benchmarks in communication systems, payload length can be a decisive factor since it defines the capability of the system to deliver the intended information to the destination.

Tests 3 use end-to-end communication from a publisher to a waiting subscriber, similarly to test 1. However, instead of using a 100 Byte message as previously, the sent message has 1KB. The acquired measurements are represented in Figure 3. It is possible to deduce that the initial more incisive peaks can be related to initial connection delays that happen to affect the overall transmission time in the first few samples. The mean time obtained was 352437 microseconds (approximately 352 ms).

Tests 4 use end-to-end communication from a publisher to a busy subscriber, similarly to test 1. However, instead of using a 100 Byte message as previously, the sent message has 1KB. The acquired measurements are represented in Figure 4. Once more, the observed high peaks' deviations present in some of the messages can be justified as a worse communication opportunity. The mean time obtained was 361348 microseconds (approximately 361 ms).

The obtained latency values show that a payload increment does not necessarily reflects an increment on the end-to-end transmission time. Also as previously stated, the busy subscriber continues presenting a higher transmission time

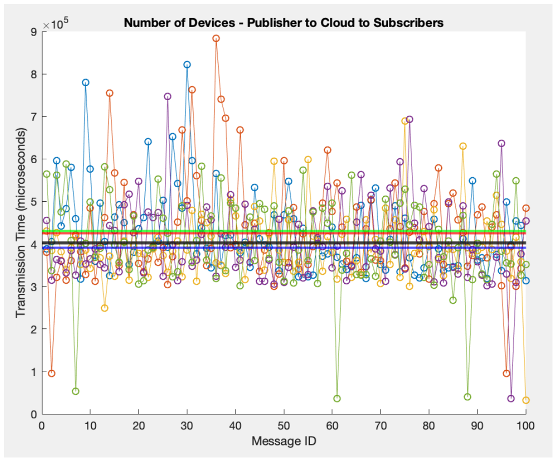

Fig. 8. Test 5: Increased number of subscribers

than the waiting subscriber, as expected.

C. *Number of devices*

The implemented publisher-subscriber system has the particular unusual characteristic of using the IoT Hub cloud platform as a broker. And so the relation between the scalability that this design allows and the transmission latency is yet unknown.
In order to evaluate this metric, test 5 was executed. Since this study was conducted in a real world environment instead of a virtual simulation, the quantity of available hardware constituted an impediment for the execution of a broader test. For that reason, a total of five subscriber ESP32 devices were used in this experiment. This test uses an end-to-end transmission from one publisher to a total of five subscriber devices (to the same specific topic) and a message length of 1kB. The acquired measurements of the receiving times for all the subscribers are represented in Figure 5. The mean time obtained was 414487 microseconds (approximately 414 ms).

Although the mean transmission time for an increased number of connected devices was slightly higher than the previous experiments, the obtained value is not representative of any substantial change in the system functionality. Since the used broker is prepared to handle millions of devices, it is predicted that the limitation of traffic should only be observable in the local network or not at all. Also, the bottleneck of transmission would be present in the used micro-service that connects the cloud and the devices.

IV. Conclusions

This paper describes the implementation of a system that enables the communication between several ESP32 endpoint devices, by establishing a connection with the Azure IoT Hub and based in a publisher-subscriber pattern. In order to evaluate the performance of the presented system, three main benchmarks were considered, namely, the connection round-trip time, payload length and number of subscriber devices involved.

The results show that the payload does not have a relevant impact on system performance, as increasing it only caused a slight improvement. Regarding the number of devices in the system, it was possible to verify that its increase resulted in a worse transmission time, and consequently a reduction of the system's performance. However, this was not considered a conclusive indicator as the IoT Hub can support up to millions of devices. Therefore this increase in transmission time may not be associated with cloud limitations but with local network congestion.

In conclusion, the real environment tests performed during the study of the developed system are not sufficient to extract definite conclusions about system's overall performance.

A. *Members Contribution*

The project can be divided into 4 main phases:

- Solution design and implementation
- Data processing
- Data analysis
- Documentation

Since none of the members had experience with any of the system required technologies and implementation steps, there was no division of tasks and all project phases were carried out by the whole team. While having the disadvantage of requiring more time for project design, this aproach allowed each element to be in touch with the challenges of each development phase. Thus the contribution of each element in the project is the same (33%).

B. *Future Research*

As Azure IoT is still a very new platform and integrates several complementary features, many diverse topics of research can be explored as part of future projects focusing on publisher-subscriber paradigms.

One interesting aspect to further examine would be the critical analysis of time measurement when trespassing the payload limitation that Azure IoT presents. Other crucial feature that could be investigated would be the connection of several tens of devices as subscribers to really understanf the influence of that metric on the system's performance.

References

[1] Nberdy. (n.d.). Introduction to Azure IoT Hub. Retrieved from https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub

[2] wesmc7777.(n.d). Send cloud-to-device messages with IoT Hub (Node.js). Retrieved from https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-c2d?fbclid=IwAR0EYOVGmvLCV_ZI0jIe7ZxcTRTTxnyTz8XB2trofb7xAesYLtInfdo0PO8

[3] robinsh.(n.d.). Communicate with your IoT hub using the MQTT protocol. Retrieved from https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support

[4] robinsh.(n.d.). Azure IoT device SDK for C. Retrieved from https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-device-sdk-c-intro