Pulga, a tiny open-source MQTT broker for flexible and secure IoT deployments

Jose Luis Espinosa-Aranda, Noelia Vallez, Carlos Sanchez-Bueno,
Daniel Aguado-Araujo, Gloria Bueno, Oscar Deniz
VISILAB, University of Castilla-La Mancha
ETSI Industriales, Avda Camilo Jose Cela s/n
13071, Ciudad Real, Spain
Emails: {JoseL.Espinosa, Noelia.Vallez, Carlos.SanchezBueno, Daniel.Aguado, Gloria.Bueno, Oscar.Deniz}@uclm.es

***Abstract*—The Eyes of Things (EoT) EU H2020 project envisages a computer vision platform that can be used both standalone and embedded into more complex artifacts, particularly for wearable applications, robotics, home products, surveillance etc. The core hardware will be based on a number of technologies and components that have been designed for maximum performance of the always-demanding vision applications while keeping the lowest energy consumption. An important functionality is to be able to communicate with other devices that we use everyday (say, configuring and controlling the EoT device from a tablet). Apart from low-power hardware components, an efficient protocol is necessary. Text-oriented protocols like HTTP are not appropriate in this context. Instead, the lightweight publish/subscribe MQTT protocol was selected. Still, the typical scenario is that of a device that sends/receives messages, the messages being forwarded by a cloud-based message broker. In this paper we propose a novel approach in which each EoT device acts as an MQTT broker instead of the typical cloud-based architecture. This eliminates the need for an external Internet server, which not only makes the whole deployment more affordable and simpler but also more secure by default.**

I. Introduction

Traditionally focused on factory automation, computer vision (i.e. software that automatically analyzes images to extract content and meaning) is rapidly moving beyond academic research and factories to many novel application scenarios. Vision technology allows inferring big data from reality and enables new types of interactivity. The possibilities are endless in terms of wearable applications, augmented reality, surveillance, ambient-assisted living, etc. Perhaps the best example of successful computer vision is Microsoft's Kinect, which combined sensing hardware with computer vision techniques to create a videogaming device. Kinect still stands as the fastest selling consumer electronics device. Nevertheless, these and other examples have generally involved large companies that can afford the required specific designs. While some required technologies are already mature and affordable, the fact is that no flexible open platform for mobile embedded vision is currently available. The main objective of the ongoing Eyes of Things project (EoT) [1] is to build an optimized core vision platform that can work independently and also embedded into all types of artefacts (Figure 1). The envisioned open hardware is to be combined with carefully designed APIs that maximize inferred information per milliwatt and adapt the quality of inferred results to each particular application. This will not only mean more hours of continuous operation, it will allow to create novel applications and services that go beyond what current vision systems can do, which are either personal/mobile or "always-on" but not both at the same time. The platform is targeted at OEMs.

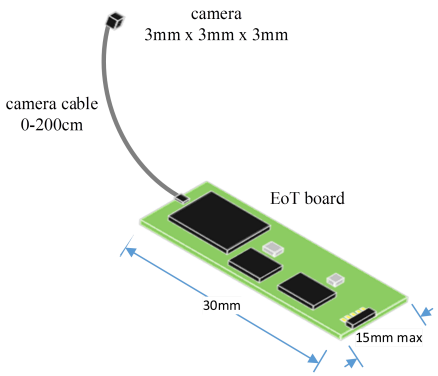

Fig. 1. EoT device

In the last few years cognitive applications and services have been acknowledged as key drivers of innovation and demand. The particular case of computer vision represents a fundamental challenge. While image analysis and inference requires massive computing power, the sheer volume of visual information that can be potentially generated by mobile devices cannot be transferred to the cloud for processing. This problem gets even worse when we consider emerging sensing technologies like 3D and hyperspectral cameras. One of the most recent attempts at alleviating this problem is the *cloudlets* approach [2], which essentially proposes offloading computation to local computers that are within one wireless hop of the mobile device. These computers would play a role similar to those in data centers. In particular, streams of image data would be processed and analyzed in those computers, providing relatively fast responses to the mobile device.

While the cloudlets approach is certainly an efficient way to manage increasing demands of computing power, it falls short in a number of aspects. First, streaming of raw sensor data out of the mobile device is still being assumed. Power efficiency then becomes a major issue, since wirelessly transmitting data for remote computation can cost up to one million times more energy per operation compared to processing locally

Accepted for publication at SP15, 1st IEEE Workshop on Security and Privacy in the Cloud, Florence (Italy), September 30, 2015

in a device. Second, it also assumes that the end-user will have to purchase and manage the local computer. Another scenario is when this computer is part of some service provided in the premises (say, within a Hospital), but then the problem becomes one of security, for raw sensor data would be streamed to an externally-managed device. The philosophy behind EoT is precisely focused on maximizing the mobile device's processing power vs energy consumption ratio as well as ensuring secure use by individual users.

Within the overall aim of optimizing energy consumption, an important component of the EoT device is the low-power WiFi chip. The specific model selected for EoT[1] provides basic TCP/IP communication. A firmware in the device allows sending/receiving of images, metadata and control/config commands. Apart from hardware, an efficient communication layer protocol is necessary. This protocol shall be used for sending the results of computer vision processing, including text, images or other types of data. HTTP is widely used, but its text-oriented nature is not appropriate for resource-limited devices. Instead, the MQTT protocol was selected early on [3]. MQTT is a lightweight publish/subscribe protocol designed for use over TCP/IP networks which provides an efficient 1-to-n communication mechanism. MQTT has been designed for low bandwidth and unreliable or intermittent connections, thus being a strong candidate in the Internet of Things scenario. MQTT-enabled devices can open a connection and keep it open using very little power.

The typical scenario is that of an embedded client which connects to an MQTT server (message broker) in the cloud [4] (Fig. 2). Again, in this scenario the brokering service has to be purchased by the user, or else it has to be installed on a locally-managed server. In this work, we propose a novel architecture in which each EoT device can act as a broker. This way no external server will be required, and data will not be initially sent through the Internet. In fact, the configuration device[2] and the EoT node do not need to be in a WiFi network infrastructure, since an ad hoc network is created by the EoT device by default. This allows setting up applications in which only the EoT and another device (say, a tablet) are involved. That is in fact the default mode upon boot, with the additional possibility of connecting to an existing WiFi network. As a result, depending on the final application an EoT device can be configured to work at 3 levels: 1) Single device mode (with the only requirement of a configuration device, typically a smartphone, tablet or laptop, connecting to the EoT-generated ad-hoc WiFi), 2) Home, i.e. EoT device connecting to a local WiFi infrastructure, and 3) EoT device connecting to the cloud (through the WiFi infrastructure).

This paper is organized as follows. Section II describes the approach followed while Section III shows its implementation. Finally Section IV summarizes the conclusions of the article.

[1]CC3100 from Texas Instruments

[2]the 'configuration device' is another device of every day use (smartphone, tablet or PC) that is first used to configure and control the EoT-based device or artefact

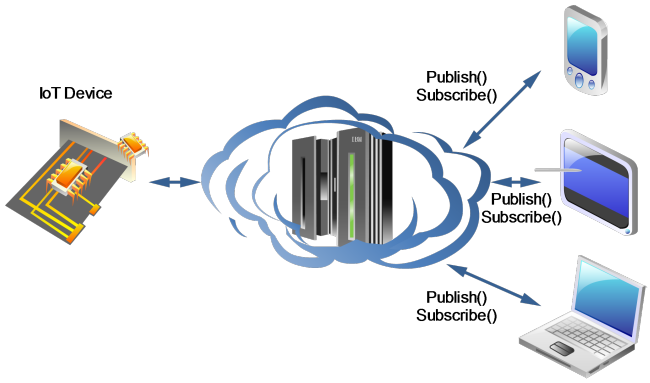

Fig. 2. Typical IoT MQTT communication model

II. Approach

By default, EoT devices create an ad-hoc WiFi. This is necessary to allow connection to the configuration device even without WiFi infrastructure. Note also that the EoT device cannot by itself connect to an existing WiFi since it does not have means for specifying SSIDs or network passwords. Each EoT device creates a WiFi with univocal SSID and password. The configuration device will have to enter those to establish communication with it. This method allows a configuration device to connect and configure devices one by one. During configuration, an EoT device can be also made to connect to a given existing WiFi (either from other EoT or from infrastructure). This allows EoTs to connect to each other (with or without infrastructure) and also allows EoTs to connect to the Internet. Low-level security is handled by an encryption protocol used in the ad-hoc WiFi (typically WPA). Horizontal arrows in Figure 3 represent data communication to/from EoT devices to/from a) Desktop computers and mobile devices such as smartphones and tablets and b) Cloud services.

All Internet of Things implementations must consider low-powered devices which need to function for months or years without getting any power recharge. This makes the as-is use of some of the existing Internet protocols to be sub-optimal. Some protocols that are heavily used in Internet add substantial overheads and a large number of device-to-cloud network technologies and protocols are being developed by researchers and start-ups. This has led to an enormous fragmentation, described in [5] in the following terms:

> *"A number of different standardization bodies and groups are actively working on creating more interoperable protocol stacks and open standards for the Internet of Things. As we move from the HTTP, TCP, IP stack to the IoT specific protocol stack we are suddenly confronted with an acronym soup of protocols- from the wireless protocols like ZigBee, RFID, Bluetooth and BACnet to next generation protocol standards such as 802.15.4e, 6LoWPAN, RPL, CoAP etc."*

Currently efficient options exist for low-power connectivity such as Zigbee and the more recent Bluetooth LE (low-energy). The proposed approach will use TCP/IP over WiFi, since low-power features present in the latest WiFi modules prepared

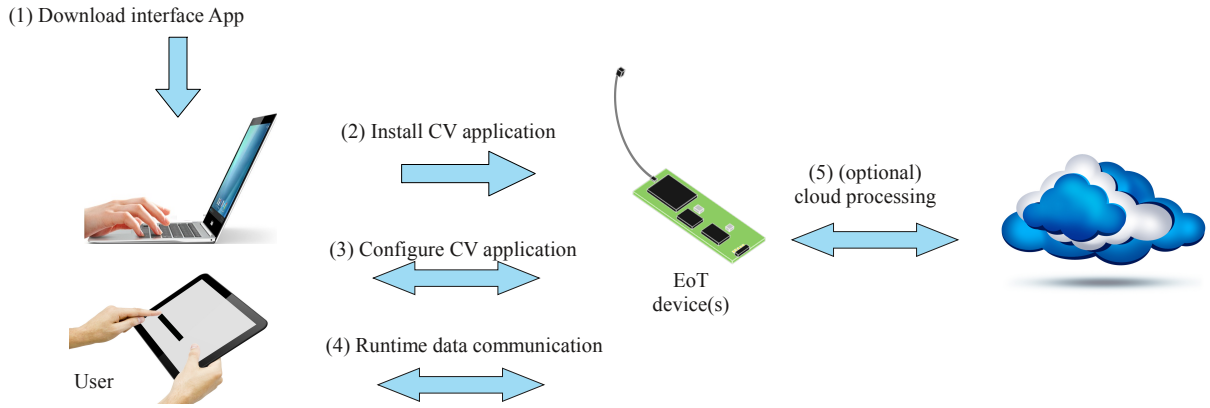

Fig. 3. User view

for the Internet of Things will be leveraged, having low-power standby modes and short wake-up times.

As mentioned above, the proposed EoT device approach uses the TCP/IP stack and the MQTT protocol [3] for communicating with other devices. MQTT-enabled devices can open a connection, keep it open using very little power and receive events or commands with as little as 2 bytes of overhead. While HTTPS is slightly more efficient in terms of establishing connection, MQTT is much more efficient during transmission, see Table I (taken from [6]).

TABLE I. MQTT VS HTTPS, SEND PERFORMANCE

	3G		WiFi	
	HTTPS	MQTT	HTTPS	MQTT
Messages/hour	1926	21685	5229	23184
% Battery/Message	0.00975	0.00082	0.00104	0.00016

MQTT v3.1.1 has become recently an OASIS Standard [7]. One of the key aspects of MQTT is an extremely efficient and scalable data distribution model. While HTTP is point-to-point, MQTT can distribute 1-1 or 1-to-n via the publish/subscribe mechanism. For these reasons MQTT is being increasingly used in mobile Apps (it is notably used by Facebook Messenger) instead of existing unreliable push notification mechanisms, see Figure 4. Devices can publish data on a topic. Other devices can subscribe to a given topic and they will receive the corresponding published data. The broker is typically hosted on an enterprise server.

MQTT is at the time of writing one of the strongest contenders in the IoT and M2M protocol wars. MQTT is a sensible option for EoT for two reasons: a) it is a low-power protocol and b) it provides an efficient 1-to-n communication mechanism. 1-to-n communication is fundamental since it allows multiple viewers, multiple (additional) processors and cooperation between sensors. The typical MQTT scenario would need a broker in the cloud. Alternatively, the smartphone/tablet/PC used for configuration could be used as a broker, but this would mean that such device (typically our personal smartphone or tablet) would have to be continuously functioning as a gateway. Thus, we propose an architecture in

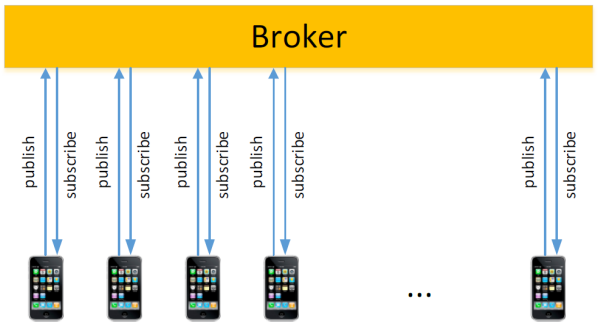

Fig. 4. MQTT publish/subscribe

which each EoT device can act as a broker. This way other EoT devices can subscribe to a given EoT. The configuration device can subscribe to the device and receive data too, although it can also 'publish' configuration commands for the device. In this architecture each EoT device can effectively act as both client and server, and the configuration device will only be woken up by the appropriate EoT (think of an alarm notification), see Figure 5.

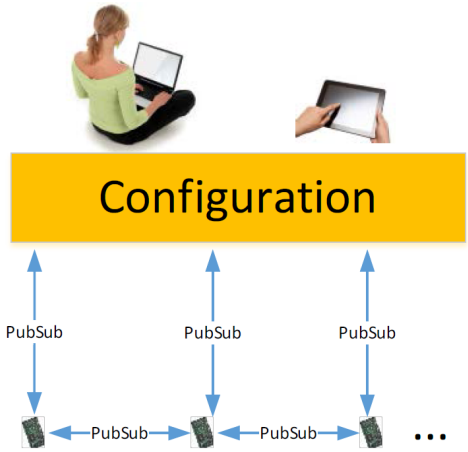

Fig. 5. Proposed EoT MQTT architecture

Apart from efficient peer-to-peer cooperation, this model allows building hierarchical processing networks. An example is extracting salient features on a first level and recognition on another. The first level of EoTs would compute salient features. The second level would be subscribed to their results and would perform object recognition. Finally, the configuration device would be subscribed to second-tier devices to get only the final result. Another possible hierarchy is object detection, followed by object recognition and object tracking. Finally, neural network-based techniques could be staged using multiple EoT devices, both in parallel and sequentially.

Note again that the configuration device and the EoT node do not need to be in a WiFi network infrastructure, since an ad hoc connection can be established. In fact, EoT devices can be integrated in a so-called MANET (Mobile ad hoc network) [8]. Note also that the huge fan-out capabilities of MQTT also make it ideal for massively parallel processing.

III. Tested implementation: Pulga

Developing an MQTT broker for an embedded device is a challenging task. The development platform was the first prototype built in the EoT project, see Figure 6. This platform is based on the Myriad 2 chip by designed by Movidius.

Fig. 6. First EoT prototype. Main parts, top-right: WiFi module, top-left: voltage level converter board, bottom: Myriad 2 development board, which includes cameras, HDMI output, dipswitch, LEDs, debug connectors, etc.

Pulga, which means flea in Spanish, is the proposed tiny MQTT broker implementation for EoT devices. Its name derives from Mosquitto [9], which is a widely-used Open Source MQTT v3.1 message broker written by Roger Light. As opposed to Mosquitto, Pulga is a lightweight broker designed to be run in embedded systems. Figure 7 shows an example of the typical configuration that the proposed approach will have.

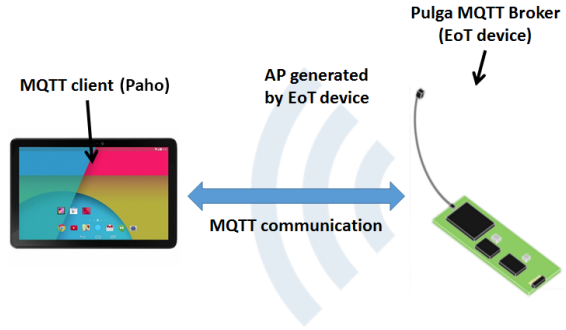

Fig. 7. Pulga broker

To develop Pulga it was necessary to start from the ground-up implementing only the minimal parts of the protocol. In particular, Pulga has the following MQTT protocol functionality implemented:

- Manage connection/disconnection of a client.
- Publish message.
- Multiple clients can connect to Pulga and subscribe/unsubscribe to topics.
- Topics/subscriptions management. Currently Pulga does not consider hierarchical topics (subtopics), only basic topics.
- Keep alive functionality through the ping request.

With respect to other typical features of a MQTT broker as defined by the protocol, there are some of them that are not implemented because of the main focus of Pulga broker. More specifically Pulga does not manage retained messages, the definition of a session as clean or durable, or the “last will” option, which allows a client to send a message that it wishes the broker to forward when it disconnects unexpectedly.

Moreover MQTT defines three levels of Quality of Service (QoS). The QoS defines how hard the broker/client will try to ensure that a message is received. There are three levels of QoS defined by the MQTT protocol, 0: The broker/client will deliver the message once, with no confirmation, 1: The broker/client will deliver the message at least once, with confirmation required, and 2: The broker/client will deliver the message exactly once by using a four step handshake. The current Quality of Service (QoS) implemented in Pulga is the QoS 0, with prevision of implementing the rest in the future.

As used in EoT, Pulga includes other functionality that a typical MQTT broker does not have. The most important is that it adds the possibility of defining ’restricted’ topics. This option will allow to include new uses for the proposed MQTT broker. When a Publish message is received Pulga first

detects if the topic is a 'restricted' one using a simple parser on the broker, changing the typical publish behaviour as the programmer defines. This approach allows the user to have additional functionality:

- Send/receive files. Considering that the MQTT protocol sends binary messages (the text of an MQTT message is always serialized) it is possible to use an MQTT message to send binary data without serializing/deserializing it. This process would allow us to upload the applications we want to run into the device or send the captured images in the EoT device to a client that requests them.
- Configuration of the EoT device. By defining a *configuration* topic, Pulga is able to select the specific preloaded executable to run[3], to manage the different EoT parameters and to configure the WiFi connection of the device. A simple parser is used to manage the messages published under the *configuration* topic (see Figure 8).
- Configuration of access to a WiFi infrastructure. The configuration device can send the required connection data to the EoT device. This data can be stored in the device and used by preloaded applications to connect to an existing WiFi.

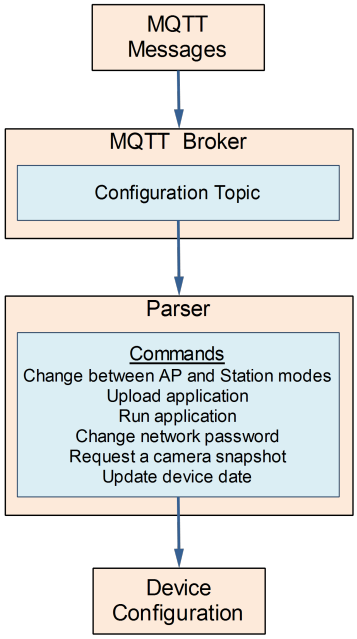

Fig. 8. Device configuration through MQTT

IV. Conclusions and Future Work

EoT focuses on developing an open platform for mobile embedded computer vision. The building elements have been all optimized for size and cost. Particularly, the device optimizes the processing power vs energy consumption ratio. Apart from hardware and architectural elements, software and protocols used have been also optimized. The publish/subscribe MQTT protocol has been selected early on because of its low-power profile. While typical scenarios involve (mobile) clients sending/receiving messages to/from a cloud-based broker, in this paper a novel architecture is proposed in which each EoT device can act as a broker itself. This provides a minimal way of communication that does not require any cloud-based broker. In this way no data is initially sent through the Internet which is also an advantage in terms of security. This basic form of communication can be in turn used to establish additional modes of communication should the application require it. It can, for example, be used to configure the device and the embedded application to run on it, including connection to an existing WiFi.

The proposed embedded MQTT broker, Pulga, offers the opportunity to install and configure applications in the EoT device using a computer or a mobile device with any MQTT client. Pulga and sample clients will be soon released as open-source software developed within the EU Eyes of Things project. The roadmap for the future includes: a) continue developing the prototype and extend it to implement QoS 1 and QoS 2 quality services and b) further develop customized MQTT clients (for Android and desktop PC) tailored to the needs of the EoT project.

[3]EoT applications are stored in either Flash memory or the embedded SD card. On boot, depending on the status of a dipswitch the EoT device will either enter 'configuration mode' (e.g. Pulga) or will directly run one of the preloaded applications.

Acknowledgements

This work has been supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 643924 (EoT Project) [1].

References

[1] "Eyes of things," Last accessed: 26th of June 2015. [Online]. Available: http://eyesofthings.eu

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[3] "Oasis standards - MQTT v3.1.1," Last accessed: 3rd of July 2015. [Online]. Available: https://www.oasis-open.org/standards

[4] L. Belli, S. Cirani, G. Ferrari, L. Melegari, and M. Picone, "A Graph-Based Cloud Architecture for Big Stream Real-Time Applications in the Internet of Things," in *Advances in Service-Oriented and Cloud Computing*, 2015, vol. 508, pp. 91–105.

[5] R. Sutaria and R. Govindachari, "Making sense of interoperability: Protocols and Standardization initiatives in IoT," in *2nd International Workshop on Computing and Networking for Internet of Things*, 2013.

[6] S. Nicholas, "Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android," 2012.

[7] "MQTT: a machine-to-machine (M2M)/internet of things connectivity protocol," Last accessed: 26th of June 2015. [Online]. Available: http://mqtt.org

[8] T. Zhuang, P. Baskett, and Y. Shang, "Managing Ad Hoc Networks of Smartphones," *International Journal of Information and Education Technology*, vol. 3, no. 5, p. 540, 2013.

[9] "Mosquitto: an open source message broker that implements the MQ telemetry transport protocol," Last accessed: 26th of June 2015. [Online]. Available: http://mosquitto.org/