MQTT-RD: A MQTT based Resource Discovery for Machine to Machine Communication

Eliseu Pereira, Rui Pinto, João Reis and Gil Gonçalves
SYSTEC - Research Center for Systems and Technologies, Faculty of Engineering, University of Porto, Porto, Portugal

Keywords: M2M Communication, IoT, Resource Discovery, Resource Directory, Zero Configuration, MQTT.

Abstract: The Internet of Things (IoT) is one of the key enablers for digital businesses and economic growth. By interconnecting objects and people through diverse heterogeneous networks, using Machine to Machine (M2M) communication, IoT enables the continuous monitoring of devices its surrounding environment, proving to have a huge potential in terms of new business opportunities. One of the biggest challenges nowadays in M2M communication, is the way devices are capable to look up for other devices and their services in local networks and internet. This paper proposes a distributed resource discovery architecture (MQTT-RD) based on the MQTT protocol. The proposed architecture enables decentralized discovery and management of devices in multiple networks, by introducing plug and play capabilities to devices, contributing for a mechanism for zero-configuration networking in IoT environments. This architecture was tested in an experimental environment, composed of multiple devices, in order to test resource discovery capabilities using an MQTT based protocol. The evaluated metrics were the overall message drop in the network, the delays in the delivery of messages and the processing time of each message.

1 INTRODUCTION

Information and Communication Technology (ICT) is considered to be one of the most important areas during the last decades and upcoming years. The constant miniaturization of technology enabled information processing devices to shift from large computers towards small portable computers integrated into larger products. ICT devices embedded into larger systems led to the term ‘embedded systems’. These systems are defined as information processing systems embedded into enclosing products, characterized specially for their real-time constrains, dependability, concurrency and efficiency requirements.

These embedded systems have been equipped with sophisticated sensors and actuators, interfaces, processors, complex control loops, software agents and communication means. Since the embedded computation on these products uses closed models and algorithms to control the electronics, the link to the physical world is frequently ignored. More recently this close link between computation and real world has recently been stressed by the introduction of the term Cyber-Physical Systems (CPS). [Lee, 2008] defined CPS as the integration of computation and physical processes. Later, [Hellinger and Seeger, 2011] extended the definition of CPS by mentioning the importance of networking, control-loops and distribution. In fact, CPS differ of embedded systems in the sense that CPS operate in a much larger scale, while an embedded system is a self-contained system confined to a single device. CPS usually include many embedded systems and other devices.

The electronics of a CPS are controlled based on an open control feedback loop, where the resulting sensor information of the physical process monitoring is used to control and optimize the CPS functionality. In a CPS, sensors and actuators have been recently enabled by Wireless Sensor and Actuator Networks (WSAN). Sensors perceive the environment conditions of the physical world and these data is used as input in decision making processes. These processes are usually represented as holonic representations of the physical object in the virtual world, mostly known as Digital Twins (DT). Usually, in a CPS, several components of the system are represented by several DTs. Because centralized control is unsuitable for large-scale system integration, cooperation between networked DTs enables distributed control.

Advances in wireless technology enabled the extension of devices with network connectivity for remote monitoring and control. These solutions based

Pereira, E., Pinto, R., Reis, J. and Gonçalves, G.
MQTT-RD: A MQTT based Resource Discovery for Machine to Machine Communication.
DOI: 10.5220/0007716201150124
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 115-124
ISBN: 978-989-758-369-8

on communication between devices, also know as Machine to Machine (M2M) communication [Galetić et al., 2011], were based on closed networks, built specifically for this purpose. Later, this network connection was based on the Internet Protocol (IP), allowing for IP enabled devices to be monitored and controlled over the Internet. Connecting objects to each other over the Internet is the main idea of the Internet of Things (IoT).

The term IoT refers to uniquely identifiable physical objects ('Things'), also known as Smart Objects, and their virtual representations in an internet-like structure, namely the DT. The internet represents the global networking of connected Smart Objects, enabling them to communicate with each other by exchanging and transforming information. [Gubbi et al., 2013] define IoT as the interconnection of sensing and actuation devices, providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. It represents information, networking and knowledge being integrated into CPS.

IoT networks have several applications, at the level of factories, hospitals, cities or agriculture. Within Industrial IoT applications, M2M communication enables full automation of sensors and actuators, allowing machines to be interconnected and to communicate over a network with minimal human intervention. Human operators and stakeholders are able to access production systems and monitoring manufacturing processes, using several Human-Machine Interfaces (HMI) and remotely connected devices. This enables the collection of machine condition and diagnosis data, and control process parameters for optimizing costs and product quality. Also, M2M communication enables machines to connect with the IT infrastructure of their own manufactures, in order to request repairs and order new parts and components.

Regarding Intelligent Transportation Systems, M2M will play an important role in connecting cars, buses, traffic lights, trams, roads and emergency crews, enabling the interconnection of all objects and roads for road traffic efficiency and safety. In e-Healthcare applications, patients staying at home with medical bio-sensors can be monitored remotely by doctors in the hospital, collecting data and analyzing the patient's health condition in real time. Networked smart meters and advanced metering infrastructures are enabled by M2M communications in Smart Power Grid applications, for efficiency improvement, service quality enhancement, save cost in power generation, distribution and consumption.

However, M2M communication presents some challenges, namely [Song et al., 2014, Datta and Bonnet, 2015]: 1) Communication interoperability, due to the heterogeneity of the communication technologies used by different devices, e.g., Bluetooth, Zigbee, Modbus, etc; 2) Semantic interoperability, due to different data models used to exchange sensor measurements and actuator commands; 3) Resource discovery, i.e., lack of capabilities to detect devices and services offered by these devices in a CPS or IoT environment, e.g., sensor measurements and/or actuation commands; 4) Self-management, i.e., lack of system capabilities to configure network topology and manage a large number of smart devices across multiple networks.

Several M2M architectures were developed by different standard organizations to tackle some of these challenges, such as the European Telecommunications Standards Institute (ETSI) [Boswarthick and Mulligan, 2009], 3rd Generation Partnership Project (3GPP) [Kunz et al., 2012] and one Machine to Machine (oneM2M) [Chang et al., 2011]. In this paper, the authors tackle the resource discovery challenge, by proposing a distributed resource discovery approach, which enables zero-configuration networking in an IoT environment. This approach introduces to M2M networked devices capabilities to discover other devices and services in the same local network and in remote networks, without relying in a centralized entity to manage devices and communications. The proposed solution, named MQTT-RD, explores the MQTT protocol by using the Constrained RESTful Environments Resource Directory (CoRE RD) principles. To validate the proposed solution, scalability tests were carried out in an experimental environment. The experimental environment contains 2 machines, one runs the main component of the MQTT-RD (*Sniffer*) while the other machine simulates multiple devices, which are successively added to the network. Through the scalability tests it is possible to infer about the overall message drop in the network, the delays in communication between components and the processing time of new messages by the *Sniffer*. From these metrics is estimated the limit number of devices per *Sniffer*, and when this limit is reached, *Sniffer* overloads message management, i.e, at this point *Sniffer* can only receive messages, not being able to forward them to the subscribers devices.

The rest of the paper is organized as follows. Section 2 presents the background and surveys the state-of-the-art regarding M2M protocols and methodologies that enable zero-configuration for resource discovery. Section 3 presents and describes the proposed architecture, highlighting the novel aspects of the work. Section 4 characterizes a testbed for testing the overall functionalities of the proposed solution and presents a discussion of the obtained results. Finally, in Section 5, the conclusions are presented, while future directions of the work are identified.

2 BACKGROUND & RELATED WORK

Regarding M2M communication and according to [Fysarakis et al., 2016], there are three main approaches for communication protocols used in IoT environments, namely service-oriented approaches, message-oriented approaches and resource-oriented approaches.

Protocols used in service-oriented architectures (SOA) are oriented for Web services, where a service is a discrete unit of functionality that can be accessed and updated remotely. The Devices Profile for Web Services (DPWS) [Driscoll et al., 2009] and the Universal Plug and Play (UPnP) [Jeronimo and Weast, 2003] are examples of such protocols, where the DPWS was introduced by Microsoft as a successor to UPnP. Nowadays, the DPWS is an open standard of the Organization for the Advancement of Structured Information Standards (OASIS), being widely used for large-scale enterprise deployment, such as industrial automation systems, for exchange soft real-time data [Cucinotta et al., 2009]. Despite being suited for residential networks without enterprise-class devices, for sharing digital media among multimedia devices, UPnP was also used in industrial applications [Gonçalves et al., 2014]. Recently, since DPWS is characterized by large memory requirements and message overhead, the OPC Foundation developed the Open Platform Communications - Unified Architecture (OPC-UA) [Mahnke et al., 2009], oriented for industrial usage [Cândido et al., 2010].

Message-oriented protocols typically focus on providing asynchronous messaging between distributed devices, focusing on reliable messaging, including Quality of Service (QoS) mechanisms. The Message Queue Telemetry Transport (MQTT) [Banks and Gupta, 2014], the Advanced Message Queuing Protocol (AMQP) [Kramer, 2009] and Extensible Messaging and Presence Protocol (XMPP) are examples of such protocols. While XMPP is an IETF standard, MQTT e AMQP are currently standardized by OASIS. AMQP is also a standard of the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC). Both MQTT and AMQP were designed as a lightweight publish-subscribe communication model, where a broker is responsible for handling and organizing all communications between the various devices. Messages are published with specific topics, which can be subscribed by different devices. AMQP provides a richer set of messaging scenarios and supports different acknowledgment use cases and transactions across message queues. On the other side, despite Google stopped supporting XMPP, due the lack of worldwide support, lately the protocol has re-gained a lot of attention as a communication protocol suitable for IoT environments.

Resource-oriented protocols refer to protocols following the Representational State Transfer (REST) architecture, which is widely known within Internet application, since typically uses the Hypertext Transfer Protocol (HTTP). Because HTTP is not suitable for IoT environments, considering the resource, bandwidth and energy restrictions of devices, the Constrained Application Protocol (CoAP) [Shelby et al., 2014] was introduced by the Internet Engineering Task Force (IETF), suitable for constrained nodes and networks. Table 1 provides a comparison between the different M2M communication protocols identified, where it is considered that a constrained environment is a device with few computational resources (e.g. embedded device).

Regarding resource discovery, [Datta et al., 2015] defend that mechanisms for automatic discovery of resources, their properties and capabilities, as well as the means to access them, are fundamental to realize the M2M communication between devices and, consequently, the vision of IoT. Traditional Web resource discovery models use Web crawlers to discover resources across the Internet. However, according to [Vandana and Chikkamannur, 2016], these traditional models are not suited for IoT applications. First, since most of IoT devices are energy constrained, they remain most of the time inactive, until and event triggers them to perform a specific function. In traditional Web resource discovery models, Web crawlers can only discover Web servers that are always active. Second, most of IoT devices are deployed in semi-closed networks, which have firewalled gateways. On the other hand, Web crawlers are suitable to discover resources in a typical Web infrastructure.

Since traditional Web discovery techniques do not produce accurate resource discovery results for IoT applications, current trends are based in network association, directory domain, peer-to-peer configuration and based on resource semantics. The most promising approaches refer to protocols specifically for resource discovery, currently proposed by IETF working group for Constrained RESTful Environments (CoRE), which are based on CoAP and Domain Name System (DNS), such as CoAP resource

Table 1: Comparison Between M2M Communication Protocols.

Protocol	**Standards**	**Constrained Environment**	**Communication Model**	**QoS**	**Resource Discovery**	**Transport Layer**
MQTT	OASIS & Eclipse Foundation	✓	Publish-Subscribe	3 levels	✗	TCP
COAP	IETF & Eclipse Foundation	✓	Request-Reply Resource-Observe	2 levels	✓	UDP
AMQP	OASIS & ISO/IEC	✗	Request-Reply Publish-Subscribe	2 levels	✗	TCP
XMPP	IETF	✗	Request-Reply Publish-Subscribe	3 levels	✓	TCP
OPC-UA	ISO/IEC	✗	Request-Reply Publish-Subscribe	✗	✓	TCP & UDP
DPWS	OASIS	✗	Request-Reply	✗	✓	TCP & UDP
UPnP	ISO/IEC	✗	Request-Reply	3 levels	✓	TCP & UDP

discovery, CoAP Resource Directory (RD) [Koster et al., 2017] and DNS-Service Discovery (DNS-SD) [Cheshire and Krochmal, 2013].

Regarding CoAP based approaches, CoAP resource discovery is a distributed approach, where a direct query is performed between devices, in order to discover resources offered by each other. A device can request the resources hosted by another device using the CoRE Link Format [Shelby, 2012]. Requests are mostly unicast, but there are the possibility to issue a multicast request, in the case that the IP address or host name of the device to be required are not known. On the other hand, CoAP RD performs resource discovery queries in a centralized manner, using a directory entity (RD). The RD is defined within a given domain, i.e., logical groups of IoT devices, instead of the entire Web. RD can be found by IoT devices using IP multicast. Then, each IoT device must register its resources to the found RD. After registration, IoT devices can discover other devices and be discovered, requesting the RD database. Discovery can occur between IoT devices located in the RD domain or may be outside of it.

[Liu et al., 2013] proposed a distributed resource directory architecture for M2M applications, referred to as *DRD4M*, which uses HTTP and CoAP protocols for resource registration and lookup in a peer-to-peer (P2P) scenario. This work focuses on designing a distributed RD and providing a real prototype. [Datta et al., 2015] proposed a resource discovery framework, based on the CoRE Link Format, which comprises a proxy layer that includes drivers for various communication technologies and protocols. [Kovatsch et al., 2014] presents a CoAP framework for IoT deployments named *Californium*, which implements RD functionality in general purpose servers. [Cirani et al., 2015] propose a Fog node named *IoT Hub*, which integrates RD functionality in order to enable a truly seamless interaction between IoT devices.

Based on the previous description of the protocols, MQTT and CoAP are supported by constrained environment devices, which allows these protocols to run almost on any device, regardless of the type of hardware. Among these two protocols, there are some differences, e.g., CoAP uses UDP while MQTT uses TCP regarding the transport layer. This difference in the transport layer gives MQTT the advantage of being more reliable in package delivery. Also, because of the higher levels of quality of service, MQTT guarantees better the delivery of messages in comparison with CoAP. Finally, regarding implementation, MQTT is easier to integrate new components when compared to more complex protocols (e.g. OPC-UA).

Although IoT networks rely greatly in the publisher-subscriber model, as supported by MQTT, request-reply models are also widely suported, namelly by protocols such as CoAP. Despite both are targeted to IoT networks, on contrary to MQTT, CoAP already supports zero configuration and resource discovery mechanisms. This capability allows the automatic configuration / discovery of new devices (hardcoded IP address not required) and the search of resources inside the *Resource Collection*. For this reason, the proposed protocol (MQTT-RD) aims to solve the limitations of the MQTT protocol, referring to resource discovery and zero configuration.

3 PROPOSED APPROACH

This section presents the MQTT-RD approach for resource discovery in M2M communication within IoT applications. The MQTT-RD uses the CoRE RD principles, but its functionalities are based on the MQTT protocol instead of CoAP. Also, the CoRE Link Format is replaced by a new defined JSON file format, named *Resource Collection*, which contains a list of all components in the network. Communication between devices is based in different message types defined in MQTT.

A typical MQTT communication scenario uses a topic-based publish-subscribe architecture. There are: 1) publishers, which are IoT devices that share data; 2) subscribers, which are IoT devices that receive shared data by the publishers; 3) brokers, which are entities that manage topics and message exchange, and can be implemented within any device in the network, including the publishers and subscribers. When new data is generated, publishers send messages to the broker, in order to update a given topic. Every time a topic is updated, the broker broadcasts the new data to every subscriber of that topic. Figure 1 represents a sequence diagram regarding the message exchange in the MQTT protocol.

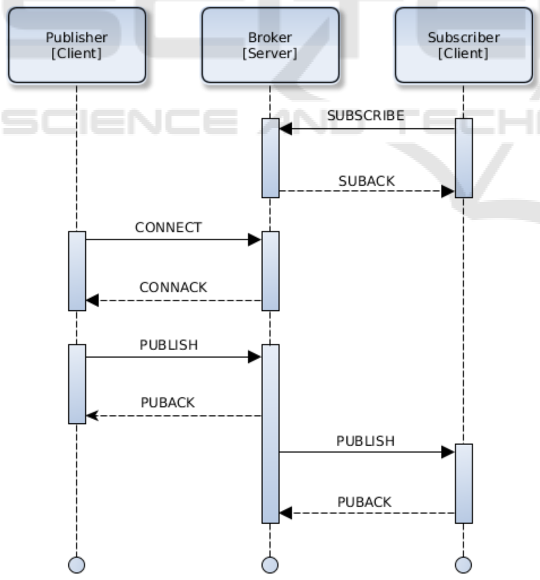


Figure 1: Message exchange in the MQTT protocol.

The *CONNECT* message is used when a device requests a connection to the broker, identifying in the payload the device ID, name and password. The broker returns a *CONNACK* message to indicate to the device success or failure in the connection. The *PUBLISH* message is associated with a topic and is used by a device to update data within a topic. The message payload contains the data to be published. After receiving a *PUBLISH* message, the broker returns a *PUBACK* message, which means that the message received can be brodcasted by all the subscribers. Facing other levels of QoS, the broker may also return a *PUBREC* message, which means that the message received can not be yet broadcasted to the subscribers. In order to subscribe to one or more topics, a *SUBSCRIBE* message can be sent to the broker, identifying the topic to be subscribed and the level of QoS. After receiving a *SUBSCRIBE* message, the broker returns a *SUBACK* message.

MQTT-RD is based on a distributed architecture where some devices are responsible for the resource discovery, i.e., identification of the IoT devices and MQTT brokers present in the network, these devices are called *Sniffers*. There is, at least, one *Sniffer* representing a LAN (i.e. *Local Sniffer*), the same way as a RD is defined within a given domain. While *Local Sniffers* are only capable of discovering resources in a specific LAN, *Internet Sniffers* can connect with other *Internet Sniffers* in other networks using the Internet, these components are present in Figure 2, which contains 2 different LANs, where each LAN contains one *Local Sniffer* and one *Internet Sniffer*. This way, the *Sniffer* is responsible off managing all the devices in the network and publications performed between them, by taking advantage of the MQTT broker functionalities. For this reason a MQTT broker is required for communication between components, so each *Sniffer* has an associated MQTT broker, while *Internet Sniffer* additionally uses another MQTT broker for communication over the internet and that's why this MQTT broker is hosted at a public address (a CloudMQTT broker was used). While the other MQTT devices may choose to have an associated MQTT broker depending on the traffic they generate or if they want an M2M communication. The interaction between *Sniffers* and devices is also presented in Figure 2, where the data flow goes from the devices to the *Sniffers*, from there the *Sniffer* forwards the data to the stakeholders, either locally or remotely.

3.1 MQTT Topics

As mention before, *Sniffers* take advantage of MQTT functionalities in order to connect with MQTT devices and other *Sniffers* present in the network. In this case, the *Sniffer* uses several specific topics, according to the type of message, namely:

- **/getlist:** This topic is present in every MQTT broker and is used by *Sniffers* to update the list of every existing *Sniffer* and MQTT device in the network, i.e., the *Resource Collection*. In this case, a

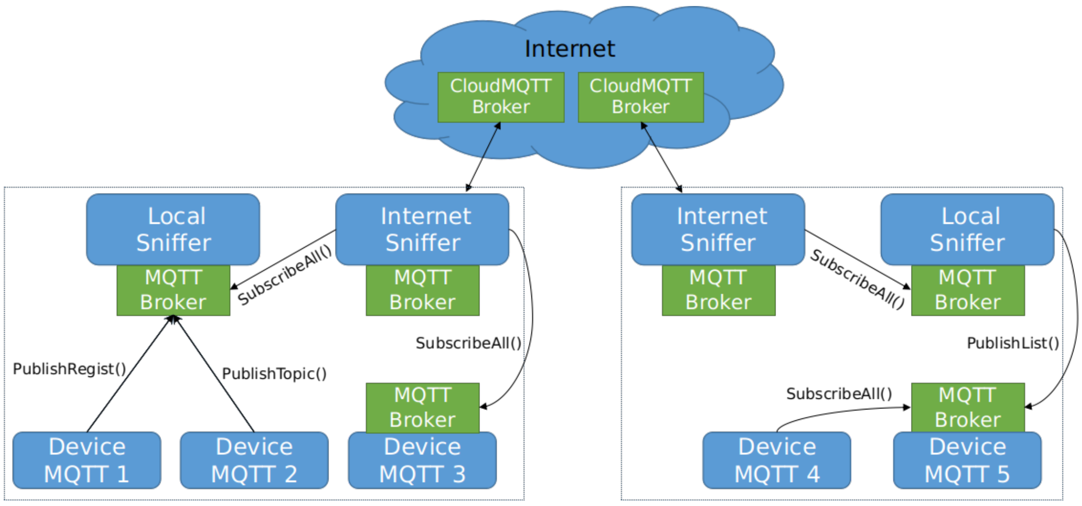


Figure 2: MQTT-RD Architecture.

new *Sniffer* or MQTT device that connects to the network should subscribe to this topic, in order to get and updated version of the *Resource Collection*.

- **/registdevice:** This topic is used to register new MQTT devices. In this case, the new MQTT device publish to this topic a self-description document. The *Sniffer* associated to the corresponding MQTT broker, which have already subscribed to this topic, uses the self-description document to add the new MQTT device to the *Resource Collection*.
- **/device_id/attrs/object_id:** This topic is used by MQTT devices to publish data regarding their attributes, such as sensor readings or actuator state. Each device attribute is associated with one of these topics, where *device_id* represents the ID used by the MQTT device in the registration in the *Sniffer* and the *object_id* represents the ID of the considered attribute.
- **/sniffercommunication:** This topic is used for message exchange between *Sniffers*, in order to update the *Resource Collection* between *Sniffers* in the registration and removal of *Sniffers* and MQTT devices. Also, this topic can be used by *Sniffers* to detect if there are anomalies regarding other *Sniffers*.

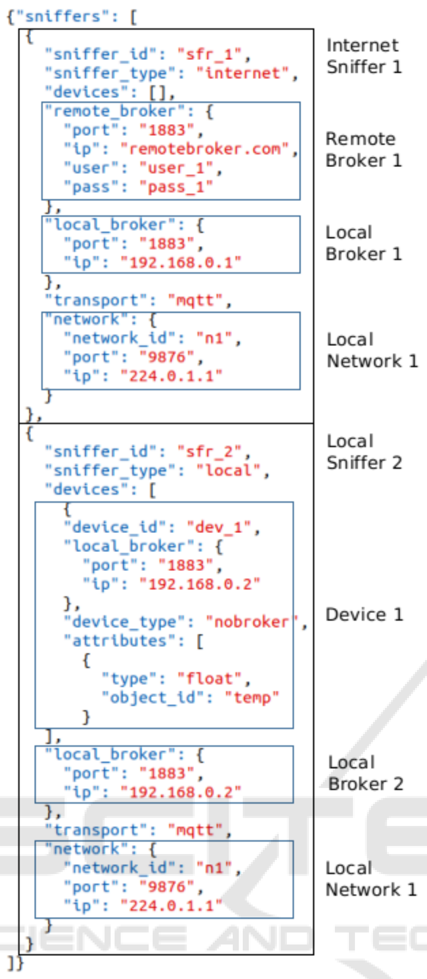
3.2 Resource Collection

The *Resource Collection* is a JSON file format that contains information regarding every *Sniffer* and MQTT device in the network. Every *Sniffer* contains a copy of the *Resource Collection*, which is updated regularly in order to keep track of existing MQTT devices and other *Sniffers* in the network, considering the registration and removal of *Sniffers* and MQTT devices in the network. Figure 3 represents the structure of the *Resource Collection* file.

The file is structured by the characterization of *Sniffers*. In this case, each *Sniffer* is represented by an unique ID, a type (in order to differentiate between local or Internet), the network location of the MQTT broker associated (local IP address and port) and the MQTT broker that enables remote connections with other *Sniffers* (public IP address, port, username and password), LAN identification and multicast IP, and a list of all MQTT devices registered in that *Sniffer*, each one containing a device ID, attributes (associated with topics, such as sensor measurements and actuator state) and device type, i.e., with or without MQTT broker associated. If a MQTT broker is associated, their parameters should be included (local IP address and port). Otherwise, parameters regarding the MQTT broker of the *Sniffer* should be considered.

3.3 Resource Registration

Before registering resources MQTT devices and *Sniffers* should know the IP address of the *Sniffer* to which they will join and for this are used multicast IP addresses. In this case, when joining the network, every MQTT device or *Sniffer* connects to the same multicast IP address. In case of one MQTT device or *Sniffer* sends a ping message to that IP, every other MQTT device or *Sniffer* connected to it are able to receive and respond to the ping message. Through this process, it is possible to add to the network new devices with or without MQTT broker and new *Sniffers*, in the case of devices with MQTT broker it is also necessary for *Sniffer* to subscribe to all MQTT broker topics.



```
{"sniffers": [
  {
    "sniffer_id": "sfr_1",
    "sniffer_type": "internet",
    "devices": [],
    "remote_broker": {
      "port": "1883",
      "ip": "remotebroker.com",
      "user": "user_1",
      "pass": "pass_1"
    },
    "local_broker": {
      "port": "1883",
      "ip": "192.168.0.1"
    },
    "transport": "mqtt",
    "network": {
      "network_id": "n1",
      "port": "9876",
      "ip": "224.0.1.1"
    }
  },
  {
    "sniffer_id": "sfr_2",
    "sniffer_type": "local",
    "devices": [
      {
      "device_id": "dev_1",
      "local_broker": {
        "port": "1883",
        "ip": "192.168.0.2"
      },
      "device_type": "nobroker",
      "attributes": [
        {
          "type": "float",
          "object_id": "temp"
        }
    ],
    "local_broker": {
      "port": "1883",
      "ip": "192.168.0.2"
    },
    "transport": "mqtt",
    "network": {
      "network_id": "n1",
      "port": "9876",
      "ip": "224.0.1.1"
    }
  }
]}
```

Figure 3: Resource Collection Information Model.

From this point forward, the MQTT device is able to register in a *Sniffer*, using the */registdevice* topic, as shown in Figure 4. On the other hand, *Sniffers* should subscribe to this topic, in order to add MQTT devices in the *Resource Collection*. In this case, a *Sniffer* can subscribe to the */registdevice* topic provided by their own MQTT brokers or the ones associated with MQTT devices, as previously mentioned. To update the *Resource Collection* on the remaining *Sniffers*, a message is sent to every *Sniffers* present on the network, this message is associated with the */sniffercommunication* topic, as reported in Figure 4. To update the *Resource Collection* on all LANs, *Internet Sniffer* must forward the register message to the remaining *Internet Sniffers* in order to update the data on their LANs.

After registration, every *Sniffer* gets the *Resource Collection* with updated data regarding existing *Sniffers* and MQTT devices in the network. The *Resource Collection* allows *Sniffers* and MQTT devices to learn the location of every component in the network and the attributes of each MQTT device. With this approach, the *Resource Collection* simplifies the operations of MQTT devices regarding topic subscription. Thus, as shown in Figure 4, after searching at the *Resource Collection* for the resource, the MQTT device can subscribe to this resource through the */device_id/attrs/object_id* topic.

3.4 Device Anomaly Detection

The MQTT-RD provides a mechanism to detect anomalies in *Sniffers*, in order to reconfigure the network accordingly. In this case, in order to evaluate the malfunction of a *Sniffer*, two methods were implemented, namely 1) the exchange of alive messages and 2) detection of connectivity lost with the MQTT broker.

Exchange of alive messages consists in a mechanism implemented in the *Sniffers*, where a *ping* message is broadcasted regularly. Within a give time period, if a *ping* message is not received by a given *Sniffer* that was present in the network, a request to remove that *Sniffer* from the *Resource Collection* is issued to all *Sniffers* in the network, as illustrated in Figure 4.

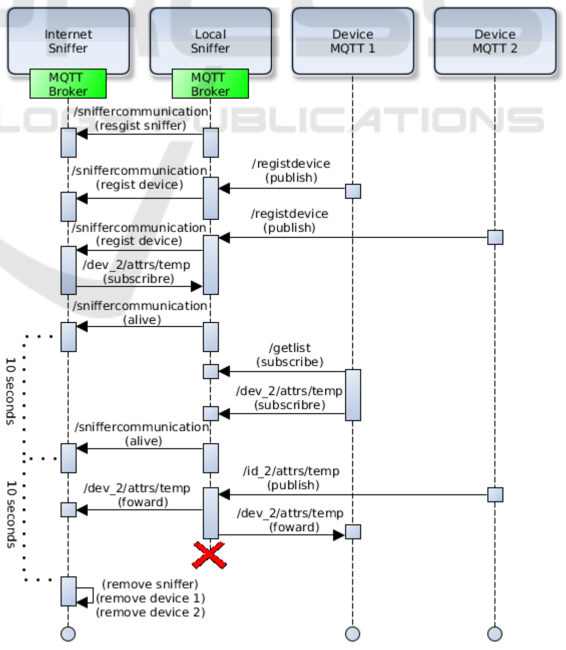

Internet Sniffer
Local Sniffer
Device MQTT 1
Device MQTT 2
MQTT Broker
MQTT Broker
/sniffercommunication (resgist sniffer)
/sniffercommunication (regist device)
/registdevice (publish)
/sniffercommunication (regist device)
/registdevice (publish)
/dev_2/attrs/temp (subscribre)
/sniffercommunication (alive)
/getlist (subscribe)
/dev_2/attrs/temp (subscribre)
/sniffercommunication (alive)
10 seconds
/id_2/attrs/temp (publish)
/dev_2/attrs/temp (foward)
/dev_2/attrs/temp (foward)
10 seconds
(remove sniffer)
(remove device 1)
(remove device 2)

Figure 4: Message exchange in the MQTT-RD protocol.

On the other end, evaluate malfunction *Sniffers* is possible by identifying the lost of connection of a given *Sniffer* to the respective MQTT broker. If the *Sniffer* cannot reconnect to the MQTT broker after a couple of trials, then a request to remove that *Sniffer* from the *Resource Collection* is issued to all *Sniffers* in the network.

4 TESTS AND RESULTS

This section details the implemented testbed to assess the concept and evaluate the technological constraints od the proposed protocol. In this case, the MQTT-RD was tested in a laboratory environment, where it was evaluated the *Sniffer* component and overall network functionality regarding scalability, i.e., number of devices available and message exchange.

4.1 Testbed

For proof of concept and evaluation, a simple version of the system architecture represented in Figure 2 was implemented. In this case, it was considered one network, containing a *Sniffer* associated with a respective MQTT broker. Simulated MQTT devices are introduced to the network, where each of these devices is associated with the MQTT broker in the *Sniffer*. Figure 5 represents the implemented testbed.

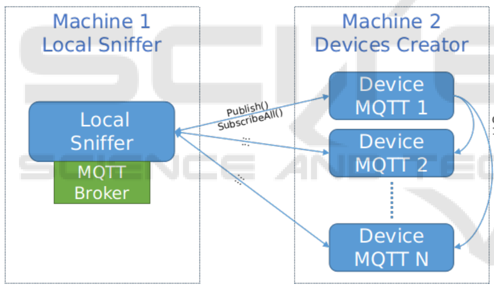


Figure 5: Testbed Architecture.

The local network was setup using a network switch, where both *Sniffer* and MQTT device components were connected via Ethernet cable. In this case, a machine (Intel i7 processor with 16GB RAM) was used to execute the *Sniffer* component and corresponding MQTT broker, while another machine (ARM Cortex-A7 processor with 1GB RAM) was used to execute a routine that simulates MQTT devices in intervals of 10 seconds. Each simulated device has only one attribute and it publish updates in the respective topic with a frequency of 1Hz. Every time a new MQTT device is created, it connects to the MQTT broker and registers in the *Sniffer*, in order to start publishing in the respective topic and subscribe to every topic of the existing MQTT devices in the network. For this, the recently registered MQTT device requests the *Sniffer* for the list of existing devices and topics, in order to subscribe to all of them.

4.2 Solution Evaluation

Evaluating the proposed solution consisted in monitoring the overall network traffic and *Sniffer* response while the implemented proof of concept executes as described earlier. In this case, every existing topic in the network was monitored, by using the corresponding *Publish* and *Subscribe* messages sent between MQTT devices and MQTT broker. Three different metrics were used to evaluate the robustness of the solution, namely: 1) the overall message drop in the network; 2) time of travel of *Publish* messages from MQTT device to broker; and 3) delay between receiving *Publish* messages and issuing corresponding *Subscribe* messages to topic subscribers. Wireshark was used to monitor network traffic, by analyzing MQTT messages exchanged in the network. This way, 8 packet captures were obtained with an increasing number of devices between 5 and 40. For each of the packet captures, an average value was calculated in any of the metrics analyzed.

For overall message drop in the network, Wireshark was used to monitor several MQTT messages exchanged between publisher MQTT devices, MQTT broker and subscribers MQTT devices, as represented in Figure 1. Figure 6 represents the results of MQTT message drop in the network, by identifying the delivery rate of publication messages sent by the devices to the MQTT broker (*Publish* messages) and the delivery rate of messages forwarded by the MQTT broker to the devices (*Subscribe* messages).

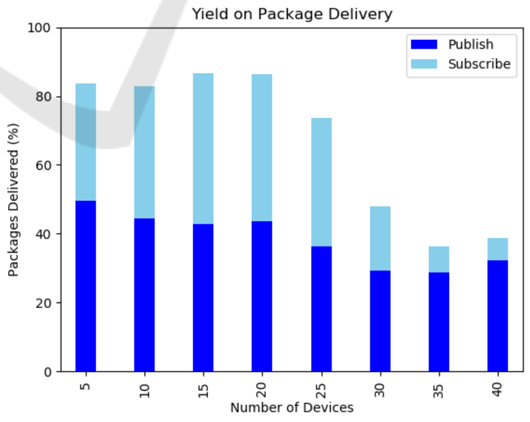


Figure 6: Yield on Package Delivery.

After the inspection of Figure 6, it is concluded that for a low number of devices, the message delivery rate is around 80-90%, which is acceptable in this case because there is a bidirectional communication with high packet traffic between the devices and the MQTT broker. The packet loss value also includes the packets that have a significant delay, i.e. packets that are not within the time interval, consequently the value of the losses increases a little. When the number of devices exceeds 20, there is a drop in the message delivery rate, more specifically in the *Subscribe* messages. This break occurs because the MQTT broker becomes saturated and is no longer able to forward messages.

Figure 7 represents the time of travel of *Publish* messages and the delay between the MQTT broker receiving *Publish* messages and issuing corresponding *Subscribe* messages to topic subscribers (MQTT broker processing delay). The time of travel of *Publish* messages from MQTT device to broker was determined by calculating the difference between the instance when MQTT devices send *Publish* messages and the instance when the MQTT broker receives them (network delay).

In this case, the number of *Subscribe* messages of a given topic increases proportionally with the number of MQTT devices in the network, because every new MQTT device in the network subscribes every topic already existing. This constant subscription of topics generates a high traffic in the network, with the increasing number of devices.

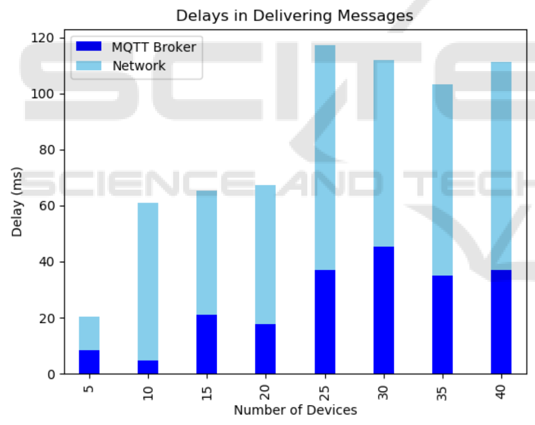


Figure 7: Delays in Delivering Messages.

For both network delay and MQTT broker processing time, their time delays tend to increase significantly when the number of devices exceeds 20. For a low number of devices the network delays range is between 10 and 60 milliseconds, which for a network with high exchange of messages is normal. Another important point is the comparison of the processing time with the network delay, in this case the processing time is lower than the network delay, which is a good indicator of the correct operation of the *Sniffer*.

5 CONCLUSION AND FUTURE WORK

As mentioned earlier, the zero configuration and resource directory mechanisms allow an easy integration of new devices into large networks. Some of the existing protocols already have these features, such as CoAP, OPC-UA or XMPP. By integrating these functionalities into a MQTT based protocol allows MQTT main advantages to be used in networks with a large number of devices, where the devices are constantly replaced. Therefore the networks become more stable, meaning less time lost in maintenance and configuration of new components.

MQTT-RD has some of the characteristics of MQTT, which distinguish it from other protocols, such as its easy integration into devices and the ability to run on components with low computational power. As improvements to MQTT, MQTT-RD allows the automatic configuration of new devices (zero configuration), the updated registration of the topics of each device (resource directory) and the search of topics by each device (resource discovery). These features enable MQTT-RD to be versatile, complete and an easy-to-use protocol.

On the other hand, as disadvantages, MQTT-RD, such as MQTT, is not as robust as OPC-UA (regarding industrial applications), since it is a less complex protocol and only supports the publish-subscribe communication model. MQTT-RD has other handicaps that make it difficult to use, such as not being available in a programming framework and increasing MQTT overheads. By means of the experimental results, it is demonstrated that for an increasing number of devices the *Sniffer* has a high network traffic and a high processing load of the MQTT broker. To reduce the load on the *Sniffer* processing, it would be ideal to distribute the devices uniformly across all the *Sniffers* present in the network, in order to have a minimum number of devices per *Sniffer*.

Based on the disadvantages presented, future work focuses on the development of a framework, in such a way MQTT-RD can be used in other application scenarios and later compared in a more detailed way with other protocols, such as the OPC-UA. After the development and testing of the framework, another goal is to apply this solution in a real environment, in order to validate the application. In general, the MQTT-RD protocol can be used in home or industrial networks, as it provides bidirectional channels, which allows the subscription and publication of topics / messages and the discovery of resources (resource directory) and endpoints (zero configuration) in a typical IoT environment.

REFERENCES

Banks, A. and Gupta, R. (2014). Mqtt version 3.1. 1. *OASIS standard*, 29.

Boswarthick, D. and Mulligan, U. (2009). M2m activities in etsi. *Presentation Report*.

Cândido, G., Jammes, F., de Oliveira, J. B., and Colombo, A. W. (2010). Soa at device level in the industrial domain: Assessment of opc ua and dpws specifications. In *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, pages 598–603. IEEE.

Chang, K., Soong, A., Tseng, M., and Xiang, Z. (2011). Global wireless machine-to-machine standardization. *IEEE Internet Computing*, 15(2):64–69.

Cheshire, S. and Krochmal, M. (2013). Dns-based service discovery. Technical report.

Cirani, S., Ferrari, G., Iotti, N., and Picone, M. (2015). The iot hub: a fog node for seamless management of heterogeneous connected smart objects. In *Sensing, Communication, and Networking-Workshops (SECON Workshops), 2015 12th Annual IEEE International Conference on*, pages 1–6. IEEE.

Cucinotta, T., Mancina, A., Anastasi, G. F., Lipari, G., Mangeruca, L., Checcozzo, R., and Rusinà, F. (2009). A real-time service-oriented architecture for industrial automation. *IEEE Transactions on industrial informatics*, 5(3):267–277.

Datta, S. K. and Bonnet, C. (2015). A lightweight framework for efficient m2m device management in onem2m architecture. In *Recent Advances in Internet of Things (RIoT), 2015 International Conference on*, pages 1–6. IEEE.

Datta, S. K., Da Costa, R. P. F., and Bonnet, C. (2015). Resource discovery in internet of things: Current trends and future standardization aspects. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 542–547. IEEE.

Driscoll, D., Mensch, A., Nixon, T., and Regnier, A. (2009). Devices profile for web services version 1.1. *OASIS standard*, 1(4).

Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., and Katos, V. (2016). Which iot protocol? comparing standardized approaches over a common m2m application. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–7. IEEE.

Galetić, V., Bojić, I., Kušek, M., Ježić, G., Dešić, S., and Huljenić, D. (2011). Basic principles of machine-to-machine communication and its impact on telecommunications industry. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 380–385. IEEE.

Gonçalves, G., Reis, J., Pinto, R., Alves, M., and Correia, J. (2014). A step forward on intelligent factories: A smart sensor-oriented approach. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–8.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

Hellinger, A. and Seeger, H. (2011). Cyber-physical systems. driving force for innovation in mobility, health, energy and production. *Acatech Position Paper, National Academy of Science and Engineering*.

Jeronimo, M. and Weast, J. (2003). Upnp design by example.

Koster, M., Shelby, Z., Bormann, C., and Stok, P. V. d. (2017). Core resource directory.

Kovatsch, M., Lanter, M., and Shelby, Z. (2014). Californium: Scalable cloud services for the internet of things with coap. In *Internet of Things (IOT), 2014 International Conference on the*, pages 1–6. IEEE.

Kramer, J. (2009). Advanced message queuing protocol (amqp). *Linux Journal*, 2009(187):3.

Kunz, A., Kim, H., Kim, L., and Husain, S. S. (2012). Machine type communications in 3gpp: from release 10 to release 12. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 1747–1752. IEEE.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE.

Liu, M., Leppanen, T., Harjula, E., Ou, Z., Ramalingam, A., Ylianttila, M., and Ojala, T. (2013). Distributed resource directory architecture in machine-to-machine communications. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 319–324. IEEE.

Mahnke, W., Leitner, S.-H., and Damm, M. (2009). *OPC unified architecture*. Springer Science & Business Media.

Shelby, Z. (2012). Core link format, draft-ietf-core-link-format-11. Technical report, Internet draft, IETF 2012 (in progress).

Shelby, Z., Hartke, K., and Bormann, C. (2014). The constrained application protocol (coap). Technical report.

Song, J., Kunz, A., Schmidt, M., and Szczytowski, P. (2014). Connecting and managing m2m devices in the future internet. *Mobile Networks and Applications*, 19(1):4–17.

Vandana, C. and Chikkamannur, A. A. (2016). Study of resource discovery trends in internet of things (iot). *International Journal of Advanced Networking and Applications*, 8(3):3084.