Testing and evaluating a security-aware pub&sub protocol in a fog-driven IoT environment

Sabrina Sicari[1], Alessandra Rizzardi[1], Luigi Alfredo Grieco[2], and Alberto Coen-Porisini[1]

[1] Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, Via O. Rossi 9 - 21100, Varese, Italy (sabrina.sicari, alessandra.rizzardi, alberto.coenporisini) @uninsubria.it

[2] Department of Electrical and Information Engineering, Politecnico di Bari, Via Orabona, 4 - 70125, Bari, Italy alfredo.grieco@poliba.it

Abstract. The continuous spreading of innovative applications and services, based on the emerging Internet of Things (IoT) paradigm, leads to the need of even more efficient network architectures/infrastructures, in order to support the huge amount of information to be transmitted in real-time. Hence, new protocols and mechanisms must be conceived to allow the IoT network to be more reactive towards environmental changes and in promptly satisfying the IoT users' requests. Aiming at dealing with the emerged issues, the paper presents an efficient IoT platform, which, thanks to fog computing principles, acts as a middleware layer between data producers and consumers; it adopts a security-aware publish&subscribe protocol, based on MQTT, coupled with a network of brokers, for efficiently sharing the processed information with end-users. Transmitted data are kept secure under an enforcement framework based on sticky policies. A test campaign is conducted on a prototypical implementation of the just mentioned platform, for preliminary evaluating its efficiency, in terms of computing effort and latency.

Keywords: Internet of Things · Fog Computing · Security · Testing · Publish&Subscribe.

1 Introduction

The diffusion of Internet of Things (IoT) technologies and applications is ever increasing in a large variety of application's domains, ranging from smart buildings, e-health, smart agriculture, industry 4.0, military services, and so on. Such a continuous spreading of the IoT paradigm has a great impact on the network's infrastructure, which must be able to efficiently manage the huge amount of data, continuously provided by IoT devices. Note that heterogeneous technologies are involved, such as Wireless Sensor Networks (WSN), RFID, NFC, actuators, and they communicate by means of different standards and protocols (e.g., MQTT, CoAP, ZigBee, BLE, 6LoWPAN). Hence, two main issues naturally emerge: scalability and interoperability. To cope with such problems, many

architectures have been proposed in literature in the last years, some of them with a certain distributed nature, other ones semi-centralized, often operating with a cloud [23]. Most of them are conceived as middleware layers or gateways, able to directly interact with IoT devices, and to transmit data to proper servers or clouds, for completing the processing and sharing tasks with the interested parties.

What does not emerge from such approaches is how much such architectures are distributed, in terms of coverage area, number of managed sources, amount of data processed, possible thresholds, and so on. In fact, often the middleware or the gateways are mentioned as single entities, which presumably interact with other similar ones, in a not so clear way. Hence, little attention has been paid, until now, to how to decentralize as much as possible all the network's components (e.g., the middleware's or gateways' modules) and tasks, in order to cover a wider area and make the IoT environment both more pervasive, reliable and powerful.

Following a fog-driven approach [3] should help in facing the raised challenges. In fact, fog computing is pursuing a decentralized networking and computing infrastructure, where data, processing tasks, storage and applications are distributed in an efficient manner towards the edge of the network, in an intermediate layer (e.g., a middleware), which is situated between the data sources and a cloud [9]. Fog computing is promoted by the *OpenFog Consortium*[3], which encourages many initiatives all over the world about its diffusion in the IoT applications.

In such a perspective, the present paper describes how fog computing principles are integrated within a security and privacy-aware IoT-based middleware platform, named NetwOrked Smart object (NOS) [20], which adopts a publish&subscribe protocol, based on MQTT, for disseminating information. Such an architecture has been chosen for two main reasons. On the one hand, the authors own an already existing and running test-bed, which allows to promptly carry out an extensive performance analysis. On the other hand, NOS is already integrated with some relevant security functionalities, which will be described later in the paper. Note that more than one NOS is expected to simultaneously run within the same IoT network. As a consequence, a first fog layer includes the network of such NOSs. Instead, a second fog layer includes a network of brokers, whose role is supporting NOSs in efficiently share the acquired and processed data towards the interested end-users. A scheme of the envisioned infrastructure is sketched in Figure 1.

The advantages of the just presented vision are the following: (i) avoiding single points of failure, thanks to the presence of multiple distributed NOSs and brokers; (ii) reducing the amount of data transmitted to a central entity, which could represent a sort of bottleneck for the IoT network (e.g., a single NOS or a single broker); (iii) reducing the delays of information retrieval, since data are closer to the final consumers, due to the highly distributed nature of the fog layers just defined. Such outcomes are hereby evaluated by means of a test

[3] `https://www.openfogconsortium.org/`

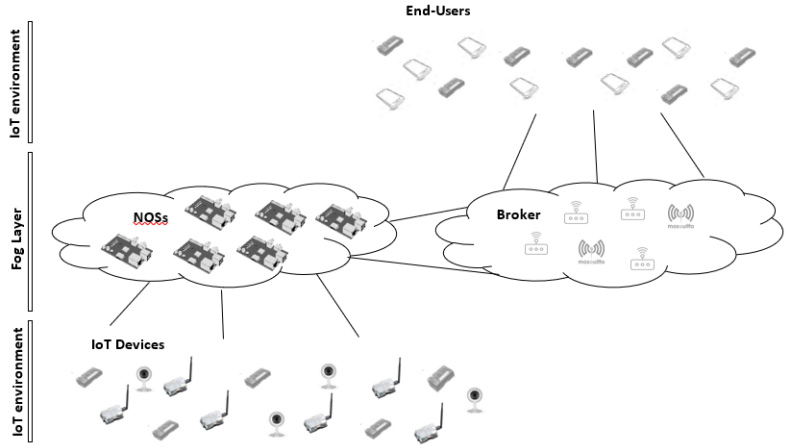

Fig. 1. IoT System composed by a dual fog layer, including multiple NOSs and brokers

campaign, which is conducted on the just mentioned test-bed. The results reveal that network's latency is reduced, since a better balancing of the data load is achieved.

The rest of the paper is organized as follows. Section 2 presents the state of the art of existing IoT infrastructures and examines how they manage the information dissemination task, which is the main focus of the proposed work. Section 3 describes the background related to NOS's platform and MQTT functionalities, useful to clearly understand the role of the introduced fog layers. Section 4.1 presents the proposed approach, which is then evaluated in Section 5. Section 6 ends the paper and draws some hints for future work.

2 Related work

The IoT environment proposed in this work envisions the coupling of fog computing paradigm and secure mechanisms for data sharing via MQTT protocol; the main purposes of the adopted network infrastructure have been just clarified in Section 1. Hereby, some related papers are described, trying to highlight their differences and weaknesses with respect to the work analyzed in the next sections.

With the final aim of improving the quality of service (QoS), the authors, in [14], present *EMMA*, an edge-enabled publish&subscribe middleware; the main weaknesses of such an approach is that it requires a controller and a broker that acts as a server for the client brokers integrated into the gateways, thus introducing a single point of failure into the network architecture.

The work in [4] proposes the adoption of a new kind of broker, named *QEST*, which is able to bridge MQTT primitives and REST interfaces, in order to ease machine-to-machine interactions. With respect to such an approach, the

target of the paper proposed hereby is a more heterogeneous and distributed IoT system, not confined to the direct communications among IoT devices, but where interactions among the different involved parties are filtered and mediated by a middleware layer, which is able to perform processing and security tasks.

[22] evaluates the performance of an edge-switch, which implements some basic MQTT broker functionalities, in a Software-Defined Networking (SDN) based system. How the different edge-switches cooperate is not clear as well as it is worth to remark that SDN still presents some centralized features.

Security and privacy requirements are not taken into account by the aforementioned solutions. Instead, works which address such issues (e.g., by means of the adoption of enforcement policies) and which make use of MQTT protocol [10] [15] [11], are based on a centralized broker, which is the obstacle the authors want to overcome in this work.

Concerning fog computing, many solutions are currently inspired to smart health scenarios [8] [13] [21], or to the Internet of Vehicles (IoV) [1] [7], or even to attacks' recognition [6] [19]. Despite such approaches deal with end-to-end secure communications, authentication and authorization, a little focus is paid on how information are effectively shared once acquired by the IoT platform.

3 IoT platform and information sharing

In this section, NOS's platform components and interactions are detailed; then, the main MQTT functionalities are explained, in order to clarify the data flow management of the envisioned IoT infrastructure.

Two main entities are typically included in an IoT system: (i) the nodes, conceived as heterogeneous data sources (e.g., WSN, RFID, NFC, actuators, etc.) which send data to the IoT platform; (ii) the users, who interact with the IoT system through services making use of such IoT-generated data, typically accessing them by means of a mobile device (e.g., smartphone, tablet) connected to the Internet (e.g., through WiFi, 4G, or Bluetooth technologies). NOSs are conceived as powerful smart devices, able to manage such entities.

NOSs and the data sources use RESTful interfaces, usually based on the HTTP or CoAP protocols, to communicate. Such interfaces are defined depending on the kind of IoT device. In this way, it is possible to collect the data from the IoT devices and to perform sources' registration. In fact, NOSs deal both with registered and non-registered sources. The registration is not mandatory, but it provides various advantages in terms of security, since registered sources may specify an encryption scheme for their interactions with NOSs, thus increasing the level of protection of their communications (encryption keys' distribution is made by the algorithms presented in [16]). The information related to the registered sources are put in the storage unit, named *Sources*. Instead, for each incoming data, both from registered and non-registered sources, the following information are gathered: (i) the kind of data source, which describes the type of node; (ii) the communication mode, that is, the way in which the data are collected (e.g., discrete or streaming communication); (iii) the data schema,

which represents the content type (e.g., number, text) and the format of the received data (since heterogeneous IoT devices may be connected); (iv) the data itself; (v) the reception timestamp. Hence, also non-registered sources are known to the system; the main difference with respect to the registered ones, is that non-registered sources does not agree on an encryption schema with NOSs to protect their transmitted data.

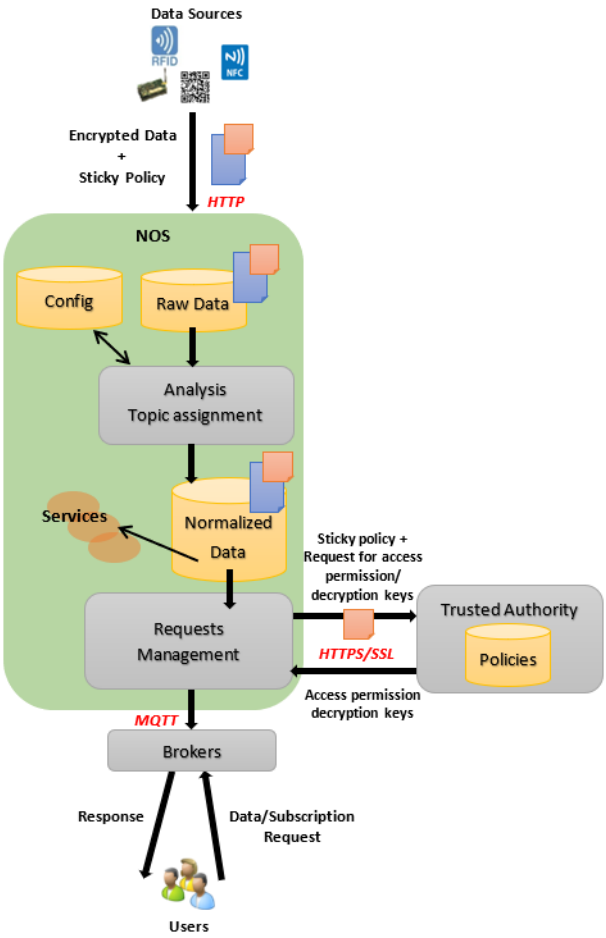

Fig. 2. Scheme of NOS architecture

Since the received data are of different types and formats, NOSs initially put them in the *Raw Data* storage unit. In such a collection, data are periodically processed, in a batch way, by the *Data Normalization* and *Analyzers* phases, in order to obtain a uniform representation and add useful metadata regarding the security (i.e., level of confidentiality, integrity, privacy and robustness of the

authentication mechanism) and data quality (i.e., level of accuracy, precision, timeliness and completeness) assessment. Such an assessment is based on a set of rules stored in a proper format in another storage unit, named *Config*, and are detailed in [20]; it allows users who access the IoT data to directly filter by themselves the data processed by NOSs, according to their personal preferences. Figure 2 sketches the NOS's components introduced hereby.

Moreover, NOS offers the following relevant security related functionalities:

- A set of algorithms, as just mentioned, for data quality and security assessment, which are able to perform an automatic evaluation of the information by inferring to the data sources behavior [20]
- Two different key management systems by Dini et al. [5] and Di Pietro et al. [12], which are adopted for protecting the communications among NOSs and data producers/consumers; note that one of them must be chosen (please refer to [16] for further details)
- An enforcement framework, which is based on sticky policies; it provides a set of general-purpose rules, aimed at regulating the access to the IoT resources and controlling the actions performed by NOSs, in order to react towards possible violation attempts [18]; this implies the presence of a *Trusted Authority*, as shown in Figure 2. Note that the *Trusted Authority* could represent a bottleneck in the system; hence, the presence of multiple coordinated trusted authorities is encouraged
- The enforcement mechanism, just presented, has been integrated with *AUPS (AUthenticated Publish&Subscribe system)*, a protocol able to securely manage publications and subscriptions through Message Queue Telemetry Transport (MQTT)[4] interactions, thus protecting the information sharing with data consumers [15].

Why MQTT has been chosen as the protocol adopted for data dissemination by the NOS platform? Firstly, because MQTT is lightweight event- and message-oriented, and allows the devices to asynchronously communicate across constrained networks to reach remote systems, as happens in typical IoT scenarios. Such a protocol employs a publish&subscribe pattern, where a central broker acts as intermediary among the entities that produce and consume the messages. All the communications among brokers and producers/consumers happen via a publish&subscribe mechanism, based on the *topic* concept. A *topic* is a mean for representing the resources (i.e., the information) exchanged within the system. *Topics* are used by data producers for publishing messages and by data consumers for subscribing to the updates from other producers. A *topic* is assigned by a proper NOS' module to each processed data for regulating the information sharing itself.

Concerning the MQTT-based AUPS mechanism, it was conceived with the presence of only one broker. Such a choice, initially dictated for simplicity, reveals now to be troublesome, due to the single point of failure and the bottleneck

[4] OASIS, MQTT v5 protocol specification, `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html`

that the broker represents in the current NOS architecture. From such a consideration emerges the proposal to include a network of brokers in the IoT platform. However, such brokers must be properly managed and their tasks must be somehow coordinated, in order to really improve the IoT network's performance. No studies have been specifically conducted in the literature on the performance of the broker involved in an IoT system, as revealed in Section 2. Such an aspect will be clarified in Section 4.1.

4 Broker's network

A dual fog layer, composed by NOSs and brokers, respectively, is setup within the IoT network, to reach the following requirements: (i) better balancing the load of the data to be shared with end-users on different brokers; (ii) reducing the delays of information retrieval from the time when information are acquired by NOS, to the time when the same information is received by the end-user; (iii) working in a location-awareness way; (iv) giving more robustness to the whole IoT system, avoiding single points of failure and bottlenecks. These are the main features, characterizing the conceived solution:

- A NOS can be connected to more than one broker and vice versa; hence, a many-to-many relationship can be established among NOSs and brokers
- NOSs and brokers communicate among themselves by means of MQTT protocol, via the MQTT client installed on NOS; such interactions are kept private thanks to the adoption of *AUPS*, as said in Section 3
- All the other communications taking place within the presented IoT system are security-aware, because: (i) users/applications receive ciphered data under specific permissions, defined at the subscription phase and implemented as sticky policies [18]; (ii) NOSs, if needed, exchange information among themselves on a HTTPS/SSL channel; (iii) the brokers need not to communicate among each other (in this sense, they are agnostic of each other), since NOSs both supervision on how information is shared and coordinate the brokers' activities
- NOSs and brokers are fully decoupled, in the sense that brokers can also be owned by different organizations/companies, which are interested in exploiting the functionalities made available by NOS platform, to disclose some relevant information to their customers. As a consequence, an organization/company can deploy its own broker and connect it to NOSs; using the MQTT protocol, no issue in terms of interoperability arises. For such a reason, the brokers' instances have not been directly installed within NOSs.

Therefore, a clear separation is created between data acquisition, performed by NOSs (along with processing tasks), and data sharing with end-users, performed by brokers. Summarizing, the authors decided to not integrate one broker for each NOS for three reasons: (i) preserve NOSs' power consumption; (ii) enabling the participation of third-party brokers; (iii) there's no guarantee that the number of NOSs must be equal to that of brokers. In fact, the proportion

between the number of NOSs and brokers within the IoT network should depend on the features of the specific application domain (e.g., number of managed topics, kinds of data producers and consumers involved). In the next section, the interactions among NOSs and brokers will be explained.

4.1 Brokers' management

Suppose that the different NOSs and brokers involved in the IoT system are identified by NOS_1, NOS_2, ...NOS_n and br_1, br_2, ...br_m, respectively. When a user or an application requires a service/data provided by the IoT system, a session is opened. During such a session, the user/application, identified by $usapp_1$, $usapp_2$, ...$usapp_j$, can obtain the information provided by a NOS, taking into account the accessible resources. The resources can be accessed on the basis of the content of the sticky policies P_{data_1}, P_{data_2}, ...P_{data_k}, defined within the NOS enforcement framework, in the format specified in [18]. An advantage of such an approach is that the brokers can manage the incoming information in compliance with the associated policies, regardless of the NOSs with which they interact, since policies travel along with the corresponding data. However, the brokers must interact with NOSs in order to establish which subscriptions to accept or deny. Hence, brokers do not have a total control on the information disclosure.

At the initial state of the IoT network, NOSs and brokers may be associated in such a way that each broker has at least one connection to a NOS. Moreover, each broker manages the topics related to the data, which are further managed by the connected NOS. This depend on the kind of data transmitted by the sources connected to that NOS. But, what happens when a data acquired by NOS_1, assigned to a certain topic t, and transmitted by NOS_1 itself to the broker br_1, is required (due to another subscription) by a user/application $usapp_2$, connected to broker br_2, which does not receive any data under t? A mechanism for efficiently satisfying such a required information's exchange must be put in action. The simplest solution would be a sort of flooding approach: the broker br_1 notifies all the other brokers br_k of the new published data, so as to make it available in the whole IoT area, covered by the brokers; or, as an alternative, each NOS notifies all the brokers belonging to the IoT network about all the managed information. Clearly, such solutions are power-consuming and redundant, since it can be assumed that the data associated to a certain topic t are not required at all points in the network every time. Hence, a more viable approach follows the steps listed hereby, which also summarized in Figure 3.

When a user or application $usapp_i$ subscribes to a certain topic t_1 on a certain broker (e.g., br_1), the broker itself inform the connected NOS (e.g., NOS_1), which performs such tasks:

1. NOS_1 checks if $usapp_i$ is authorized to access the data published under the topic t_1 (i.e., the check is executed by contacting the Trusted Authority, which is able to verify the compliance of the sticky policy associated to the data under topic t_1 with the attributes owned by the requester)

2. If yes, br_1 is enabled to notify $usapp_i$ about the information related to topic t_1; if no, the requested resource cannot be disclosed
3. However, NOS_1 has to check if it directly manages the data assigned to topic t_1; such a check is performed by using a proper table, named *topicsMap*, which is stored in the *Config* collection (see Section 3) and contains an entry for each couple topic-NOS in the form (t_i, NOS_i)
   (a) If the couple (t_1, NOS_1) exists, the data acquired by NOS_1 and published under the topic t_1 will be notified by br_1 to the authorized $usapp_i$, but what happens if other NOSs process information related to the topic t_1, instead of NOS_1?
   (b) In such a case or in case the entry (t_1, NOS_1) is not found in NOS_1, then NOS_1 itself must find the couple or the couples (t_1, NOS_i), where i is not equal to 1, and warn the selected NOS_i (for example, NOS_2 in Figure 3) about the fact that it must begin to publish the data related to the topic t_1 towards br_1. Finally, *topicsMap* must be updated accordingly: as shown in the example of Figure 3, the couple (t_1, NOS_2) is added to the *topicsMap* on all NOSs; note that such an update is notified to all NOSs for future requests via the proper secure MQTT dedicated channel [17], in order not to compromise the *topicsMap*'s content.

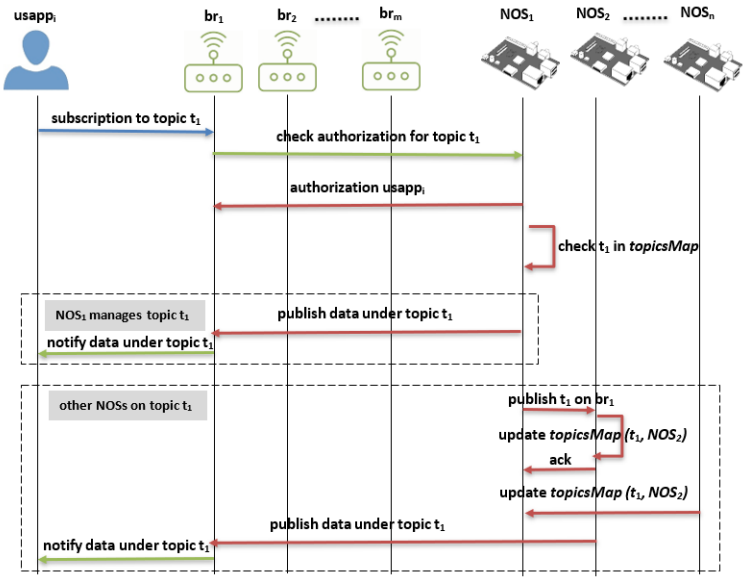

Fig. 3. Scheme of NOSs and brokers interactions

5 Performance analysis

A test campaign is hereby proposed, in order to preliminary asses the feasibility of the approach presented in Section 4.1. Firstly, the employed hardware and software tools are introduced. Secondly, obtained results are discussed.

5.1 Test-bed details

A test-bed, whose software is openly accessible at `https://bitbucket.org/alessandrarizzardi/nos.git`, is setup to practically perform an analysis about computing effort and latency metrics. The testing environment, sketched in Figure 4, is composed by four instances of NOS (NOS_1, NOS_2, NOS_3 and NOS_4), running on four Raspberry Pi platforms, and by a variable number of brokers (from two to ten, namely br_1, ..., br_{10}), and data sources, which virtually run on separated virtual machines, installed on a personal computer. The sources use measures from real-world smart home test-bed[5], acquired by means of installed sensors that collect electricity data every minute for the entire home [2]. In particular, data are gathered from smart meter number 2 of *Home A*, which include, among the others, electricity consumption data of: kitchen lights, bedroom lights, duct heater HRV, and HRV furnace, published under the topics *homeA/lights/kitchen* (t_1), *homeA/lights/bedroom* (t_2), *homeA/HRV/ductheater* (t_3), and *homeA/HRV/furnace* (t_4), and so on. Wi-Fi connections are adopted for communications among the data sources, the MQTT brokers, and NOSs (i.e., the Raspberry Pi). NOSs modules interact among themselves through *RESTful* interfaces; such a feature allows the NOSs' administrators to add new modules or modify the existing ones at runtime, since they work in a parallel and non-blocking manner. Moreover, the non-relational nature of the adopted *MongoDB* database allows also the data model to dynamically evolve over the time. *Node.JS*[6] platform has been used for developing NOSs' core operations, *MongoDB*[7] has been adopted for the data management, and Mosquitto[8] has been chosen for realizing the open-source MQTT broker. Information is exchanged in *JSON* format. More details about the implementation can be found in [20].

The parameters used for simulations are summarized in Table 1. The storage overhead will be not investigated in this work, because it has just been deeply analyzed in previous works [18] [17]; it is worth to remark NOSs support a non-persistent storage of IoT-generated data, since *Raw Data* and *Normalized Data* collections are emptied as the data are transmitted to the brokers. The same is for the brokers, which do not need to persistently store IoT-data to continue their activity. If the IoT system needs to persistently store the information obtained from the IoT network, a proper infrastructure (e.g., a cloud) must be involved.

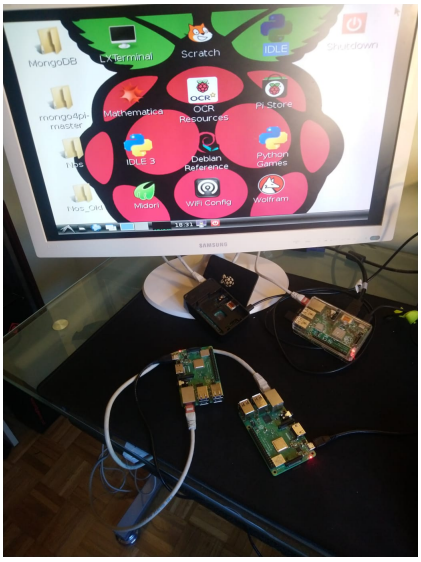

Fig. 4. Test-bed

Table 1. Test-bed parameters

Parameter	Value
NOSs	4
Brokers	[1, 10]
Sources	4
Topics	10
Data generation rate	10 pck/second
Data request rate	10 req/second
Observation time	24 hours

5.2 Results

Figures 5 and 6 show the average distribution of the CPU load on the analyzed NOSs and brokers, respectively, in three different situations: (i) the IoT system adopting only one broker (i.e., br_1); (ii) running five brokers; (iii) running ten brokers. Such a metric is important for investigating about the computational resources' consumption of the tasks performed at the middleware IoT layer. The simulated scenario is as follows: (i) sources send to NOSs data related to the aforementioned topics at a rate of $10 pck/sec$; while data requests (i.e., by hypothetical users) are simulated as well, at the same rate, and imply the notification

[5] `http://traces.cs.umass.edu/index.php/Smart/Smart`
[6] `http://nodejs.org/`
[7] `http://www.mongodb.org/`
[8] Mosquitto broker, `http://mosquitto.org`

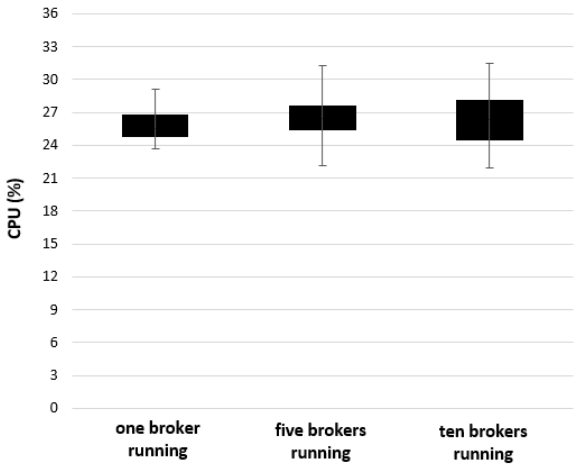

Fig. 5. Whiskers-box diagram: average CPU load on NOSs

from the brokers; (ii) at the initial stage, topic t_i is only associated with the topic t_i, but such a setting will vary during the system's running depending on the random users' requests; (iii) in fact, when an external entity requires, for example, a subscription to t_1 towards br_2, the procedure presented in Section 4.1 must be executed.

The obtained results suggest that having more brokers affects, in a relevant way, the performance of the whole IoT system. Instead, the CPU load on NOSs is approximately constant; in the three scenarios, such a behavior is due to the fact that, NOSs process the same amount of data, but, even more important is to note that the resources required by NOSs to manage the *topicsMap* is negligible. Instead, the computing effort on brokers decreases, since they share the data load, balancing the requests to be managed. Similar considerations can be made for the latency. In fact, the average time required by data from their transmission towards NOS to their reception by the subscribed entities is reduced in the three scenarios, as shown in Figure 7. This means that, in presence of more brokers, the IoT system is able to satisfy the users' requests in a shorter time, thus improving the efficiency of the whole application.

Summarizing, the presence of more brokers better balances the data load, generated by the IoT system, without demanding all the information sharing task to one centralized broker, than the previous version of the NOS platform.

6 Conclusion

The paper has presented a security-aware fog-driven IoT architecture, composed by a dual fog layer, involving smart powerful devices (i.e., NOSs), responsible

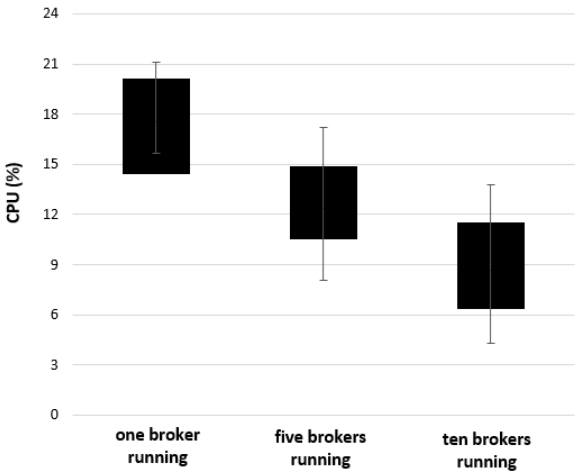

Fig. 6. Whiskers-box diagram: average CPU load on brokers

of acquiring and processing IoT-generated data, and a network of brokers, responsible for disseminating such elaborated data. Hence, MQTT protocol has been adopted, mainly due to its lightweight primitives, which fit the constraints of IoT devices. The paper has also provided a test campaign with the final aim to assess the advantages brought by the interactions among NOSs and brokers, which are conceived as fully decoupled. The outcomes revealed that network's latency is reduced, since a better balancing of the data load is achieved. Surely, a deeper analysis will be conducted in the near future, taking into account different applications scenarios and the possibility of connecting real IoT-devices as data sources, in order to conduct analysis on the power consumption of both data producers and IoT platform. Moreover, different QoS (Quality of Service) modes, related to the MQTT protocol, can be evaluated, in order to reveal how they affect the system performance.

References

1. Arif, M., Wang, G., Balas, V.E.: Secure VANETs: trusted communication scheme between vehicles and infrastructure based on fog computing. Stud. Inform. Control **27**(2), 235–246 (2018)
2. Barker, S., Mishra, A., Irwin, D., Cecchet, E., Shenoy, P., Albrecht, J.: Smart*: An open data set and tools for enabling research in sustainable homes. SustKDD, August **111**, 112 (2012)
3. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing. pp. 13–16. ACM (2012)

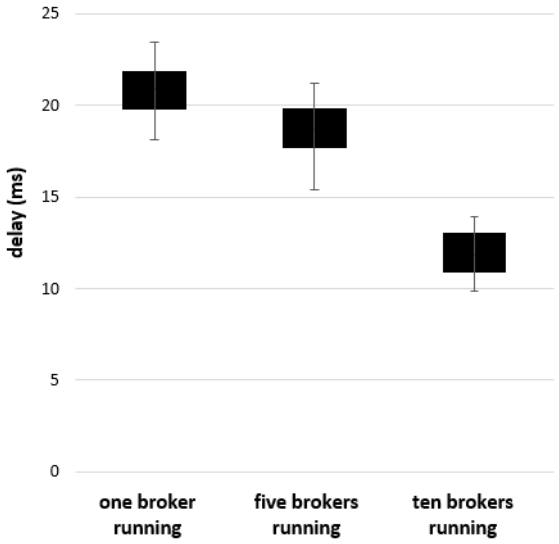

Fig. 7. Whiskers-box diagram: average end-to-end data latency

4. Collina, M., Corazza, G.E., Vanelli-Coralli, A.: Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In: IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC). pp. 36–41 (2012)
5. Dini, G., Lopriore, L.: Key propagation in wireless sensor networks. Computers & Electrical Engineering **41**, 426–433 (2015)
6. Ionita, M.G., Patriciu, V.V.: Secure threat information exchange across the internet of things for cyber defense in a fog computing environment. Informatica Economica **20**(3) (2016)
7. Kang, J., Yu, R., Huang, X., Zhang, Y.: Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles. IEEE Transactions on Intelligent Transportation Systems **19**(8), 2627–2637 (2018)
8. Moosavi, S.R., Gia, T.N., Nigussie, E., Rahmani, A.M., Virtanen, S., Tenhunen, H., Isoaho, J.: End-to-end security scheme for mobility enabled healthcare internet of things. Future Generation Computer Systems **64**, 108–124 (2016)
9. Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A.: A comprehensive survey on fog computing: State-of-the-art and research challenges. IEEE Communications Surveys & Tutorials **20**(1), 416–464 (2017)
10. Neisse, R., Steri, G., Baldini, G.: Enforcement of security policy rules for the internet of things. In: IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). pp. 165–172 (2014)
11. Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., Panya, A.: Authorization mechanism for mqtt-based internet of things. In: IEEE International Conference on Communications Workshops (ICC). pp. 290–295 (2016)

12. Pietro, R.D., Mancini, L., Jajodia, S.: Providing secrecy in key management protocols for large wireless sensors networks. Ad Hoc Networks **1**(4), 455–468 (2003)
13. Rahmani, A.M., Gia, T.N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., Liljeberg, P.: Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. Future Generation Computer Systems **78**, 641–658 (2018)
14. Rausch, T., Nastic, S., Dustdar, S.: Emma: Distributed qos-aware mqtt middleware for edge computing applications. In: IEEE International Conference on Cloud Engineering (IC2E). pp. 191–197 (2018)
15. Rizzardi, A., Sicari, S., Miorandi, D., Coen-Porisini, A.: AUPS: An open source authenticated publish/subscribe system for the Internet of Things. Information Systems **62**, 29–41 (2016)
16. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Internet of Things: Security in the keys. In: 12th ACM International Symposium on QoS and Security for Wireless and Mobile Networks. pp. 129–133. Malta (Nov 2016)
17. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Dynamic policies in internet of things: enforcement and synchronization. IEEE Internet of Things Journal **4**(6), 2228–2238 (2017)
18. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Security towards the edge: Sticky policy enforcement for networked smart objects. Information Systems **71**, 78–89 (2017)
19. Sohal, A.S., Sandhu, R., Sood, S.K., Chang, V.: A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. Computers & Security **74**, 340–354 (2018)
20. S.Sicari, Rizzardi, A., Miorandi, D., Cappiello, C., Coen-Porisini, A.: A secure and quality-aware prototypical architecture for the Internet of Things. Information Systems **58**, 43–55 (2016)
21. Thota, C., Sundarasekar, R., Manogaran, G., Varatharajan, R., Priyan, M.: Centralized fog computing security platform for iot and cloud in healthcare system. In: Exploring the convergence of big data and the internet of things, pp. 141–154. IGI Global (2018)
22. Xu, Y., Mahendran, V., Radhakrishnan, S.: Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery. In: IEEE 8th International Conference on Communication Systems and Networks (COMSNETS). pp. 1–6 (2016)
23. Yaqoob, I., Ahmed, E., Hashem, I.A.T., Ahmed, A.I.A., Gani, A., Imran, M., Guizani, M.: Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges. IEEE wireless communications **24**(3), 10–16 (2017)