Increasing the pervasiveness of the IoT: fog computing coupled with pub&sub and security

Sabrina Sicari
Dip. di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
sabrina.sicari@uninsubria.it

Alessandra Rizzardi
Dip. di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
alessandra.rizzardi@uninsubria.it

Alberto Coen-Porisini
Dip. di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
alberto.coenporisini@uninsubria.it

***Abstract*—People are increasingly surrounded by a connected world, where they can gather and share information everywhere, at anytime, and by means of a variety of devices, belonging to the so-called Internet of Things (IoT) network. IoT technologies and applications are spreading in different scenarios, ranging from every-day life activities to business ones. The presence of a huge amount of data, continuously transmitted over the network, brings relevant issues in terms of scalability. Hence, a proper network infrastructure must be put in action, in order to efficiently manage the information. A middleware layer could be a potential solution for overcoming such an issue, and to cope with interoperability. In literature, many architectures have been proposed in the last years, but little attention has been paid to how to decentralize as much as possible all the network's components and tasks, in order to cover a wider area, avoiding single points of failure, while guaranteeing efficiency. Moreover, the interest of different stakeholders is not adequately considered yet. In this sense, fog computing represents a viable approach, which is adopted in this paper for the realization of a highly distributed and security-aware IoT middleware, aimed at operating without the need of a central coordinating unit and at allowing the participation of multiple stakeholders in the same IoT infrastructure. The proposed solution exploits the functionalities provided by the MQTT protocol, and its potentialities, besides the architectural features, are evaluated by means of a simple yet real test-bed, in terms of computing effort and latency.**

***Index Terms*—Internet of Things, Fog Computing, Security, Privacy, Publish&Subscribe**

I. Introduction

The advent of Internet of Things (IoT) technologies and applications in a large variety of application's domains have a great impact on the network's infrastructures, which must be able to efficiently manage the huge amount of data, continuously provided by IoT devices. Note that heterogeneous technologies are involved (e.g., WSN, RFID, NFC, actuators) and they communicate by means of different standards and protocols. 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 [1]. 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 the final processing and sharing with the interested parties. What does not totally 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 more pervasive.

In this paper, the raised challenge is faced by adopting the *fog computing* principles [2]. In fact, *fog computing* mainly consists of 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), situated between the data sources and a cloud [3]. *Fog computing* is promoted by the *OpenFog Consortium* [1], which encourages many initiatives all over the world about its diffusion in the IoT. The advantages of adopting such a vision are the following: (i) avoiding single points of failure (e.g., the cloud); (ii) reducing the amount of data transmitted to a central entity, which represents a sort of bottleneck; (iii) reducing the delays of information retrieval, since data are closer to the final consumers. The main contribution of this paper is the integration of such features within a security and privacy-aware IoT-based middleware platform, named NetwOrked Smart object (NOS) [4]. Such an architecture has been chosen for two main reasons. Firstly, the authors own the test-bed, so as to be able to carry out an accurate and concrete performance analysis. Secondly, NOS is already integrated with the following security functionalities, also summarized in Figure 1:

- A novel algorithm for security level's assessment has been defined; it is able to perform an automatic evaluation of the information by inferring to the data sources behavior [4]

[1] https://www.openfogconsortium.org/

- Proper key management systems are adopted, for securing the communications among NOSs and involved parties with the distribution of well-defined encryption keys [5]
- An enforcement framework, based on sticky policies, 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 [6]
- The enforcement mechanism, just presented, has been integrated with *AUPS (AUthenticated Publish&Subscribe system)*, a protocol able to effectively manage publications and subscriptions through Message Queue Telemetry Transport (MQTT)[2] interactions, securing the information sharing with parties interested in the services provided by the IoT platform [7].

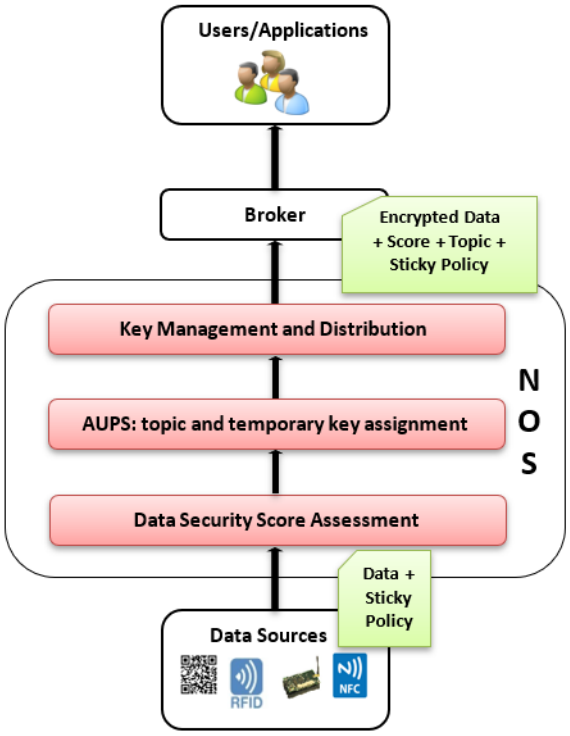

Fig. 1. NOS functionalities

The last feature is the most important for the work proposed hereby. In fact, MQTT potentialities are coupled, in a new way, with the *fog computing* paradigm, in order to obtain a highly distributed platform, composed by a network of NOSs and brokers, and able to guarantee high levels of reliability to the transmitted information, which becomes available within the IoT network, thanks to the provision of a novel sharing mechanism. Note that, in [7], the IoT platform was able to manage only a single broker, which represented a single point of failure for the system

Another fundamental point is the involvement of different stakeholders in the IoT infrastructure. This is possible only if the IoT network itself allows their participation. In this work, the brokers, acting as distributed devices in the already existing NOSs' fog layer, are in charge of transmitting the processed data, and can be owned by any organization or company interested in exploiting the functionalities of the IoT network to disclose data of interest to their consumers.

The rest of the paper is organized as follows. Section II presents the state of the art of existing IoT infrastructures. Section III describes the background on NOS's platform and MQTT functionalities. Section IV presents the proposed approach, which is then assessed in Section V, by evaluating relevant metrics. Section VI summarizes the outcomes of the conducted research.

II. Related work

With the scope of improving the quality of service (QoS), the authors, in [8], 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 single points of failure in the network architecture.

The work in [9] 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.

[10] 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 [11] [7] [12], are based on a centralized broker, which is the obstacle theu authors want to overcome in this work.

Concerning fog computing, many solutions are currently inspired to smart health scenarios [13] [14] [15], or to the Internet of Vehicles (IoV) [16] [17], or even to attacks' recognition [18] [19]. A preliminary performance analysis is included in [20], where an IoT application framework, based on fog computing principles, is integrated with MQTT and user managed access (UMA). 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 platforms.

In order to fill such a gap, the work proposed hereby aims to couple the fog computing paradigm with secure mechanisms for data sharing via MQTT, within a highly distributed network infrastructure, enabling the presence of multiple stakeholders.

[2]https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

III. Platform and motivations

In this section, the background on NOS's platform, sketched in Figure 2, and MQTT is presented.

A. Networked smart object platform

A typical IoT system includes two main entities: (i) the data sources (i.e., the nodes), which are heterogeneous devices sending data within the IoT network; (ii) the users, who make their requests within the IoT network, making use of the services made available by the IoT system itself. Note that the users usually access to the IoT services by means of mobile devices (e.g., smartphone, tablet), which are connected to the Internet. The distributed IoT platform itself is conceived as a network of NOSs, which are powerful smart devices, in charge of managing both data sources and users' requests.

Interfaces based on HTTP protocol are adopted in the communications among NOSs and nodes (i.e., the data sources). The sources can establish whether being registered or not to the IoT platform. In fact, the registration is not mandatory, but increases the level of protection of the communications, for example specifying an encryption scheme to send the information in a ciphered manner. A storage unit, named *Sources*, is responsible for storing the data concerning the registered sources. Instead, each data, provided by a node, is initially put in the *Raw Data* storage unit, where the following information are gathered: a) the type of node, that describes the type of source; b) the communication mode, that is, the way in which the data are collected (e.g., discrete or streaming communication); c) the data schema, that represents the content type (e.g., number, text) and the format of the incoming data; d) the data content; e) the reception timestamp.

Periodically (i.e., like a batch), the information collected in *Raw Data* are processed by the *Data Normalization* and *Analyzers* modules. Their goal is twofold: (i) obtaining a uniform representation of the heterogeneous data provided to the IoT platform; (ii) adding useful metadata regarding the level of the following security requirements, which are confidentiality, integrity, privacy and robustness of the authentication mechanism; (iii) adding useful metadata regarding the level of of the following data quality requirements, which are completeness, accuracy, precision, and freshness. The security and data quality assessment follows a set of rules, which are specified in the *Config* collection. Further details are available in [4]. The scope of such a task is to allow users, when accessing the IoT data, to directly filter them by themselves, according to their personal preferences.

B. MQTT protocol and motivations

MQTT protocol seems to be a viable solution, in order to allow the almost constrained devices to asynchronously communicate within an IoT network. It is lightweight and event-/message-oriented, and it is able to guarantee small bandwidth connections among the IoT devices in an easy manner. In fact, a central broker is responsible to act as an intermediary with respect to the entities that produce or consume the data, following a publish&subscribe interaction

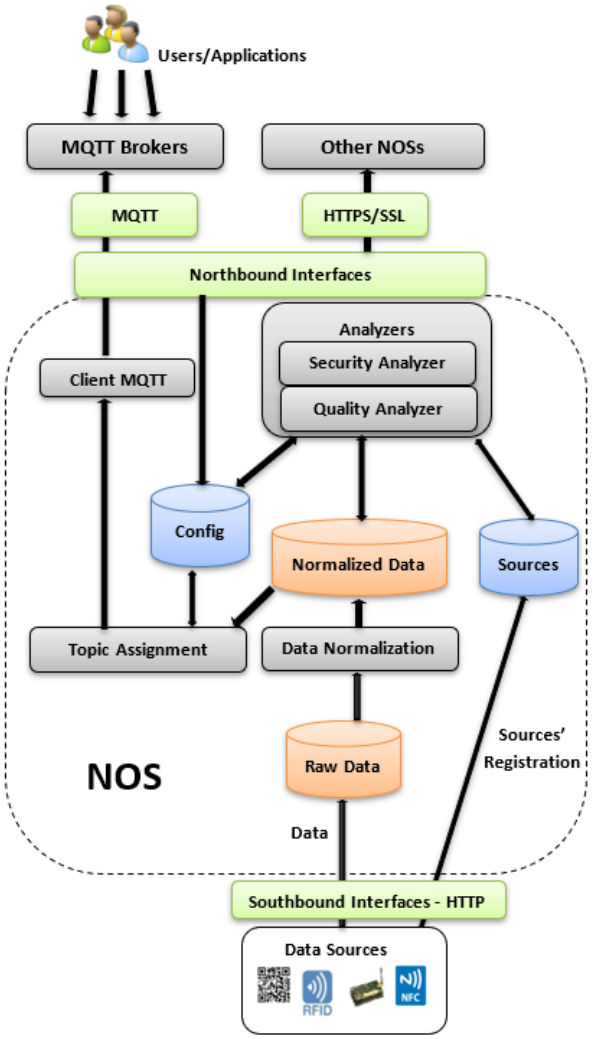

Fig. 2. Scheme of NOS architecture

pattern. Information are shared on the basis of the so-called *topics*, which represent a sort of categories of the data, made available within the IoT network. More in detail, NOSs publish some data under specific topics and users subscribe them for being notified about the information related to certain topics.

It is worth to remark that the current versions of MQTT (i.e., v5 and v3.1) do not provide a native support neither for mutual authentication mechanisms nor for ensuring data integrity and confidentiality. As a consequence, NOS platform has been integrated with a security aware MQTT based system making use of sticky policies. Please refer to [7] and [6] for further details.

The main issue arises when thinking about the role of the central broker. It represents a serious drawback of the current NOSs middleware platform and, in general, of architectural solutions based on MQTT. In fact, even if the broker allows to easily share the information to the subscribed parties, it clearly represents a bottleneck in a wide IoT system, since all the notifications must be managed by a single entity, possibly

guaranteeing the service in real time. No studies have been specifically conducted in the literature on the performance of the broker involved in an IoT system, as revealed in Section II. The approach, proposed in this paper, includes the presence of a network of brokers, whose relevant metrics are evaluated with respect to the same IoT system adopting a single broker (see Section V). But, it is important to clarify the way how the fog computing concept is adopted. In fact, a sort of dual fog layer is put closer to the data sources: the former is composed by NOSs, which are in charge of acquiring data from the near IoT devices and perform the security tasks; the latter is composed by the brokers (whose number can be established considering the specific application), which interact both with NOSs and with the end-users. Hence, the broker is not yet considered a solitary central unit, closer to a cloud or a server, but a distributed technology, which contributes to the efficiency of the whole IoT system. Note that, the presence of the described fog layer should improve the overall performance of the IoT system, in terms of latency, since the middleware composed by NOSs and brokers should speed up the data processing. A scheme of the envisioned infrastructure is sketched in Figure 3.

At this point, the main issue which arises is how a data acquired by a NOS, and further transmitted to a broker, can be transferred by an end-user which is connected to another broker. Such an aspect and other practical features of the proposed approach are investigated in Section IV.

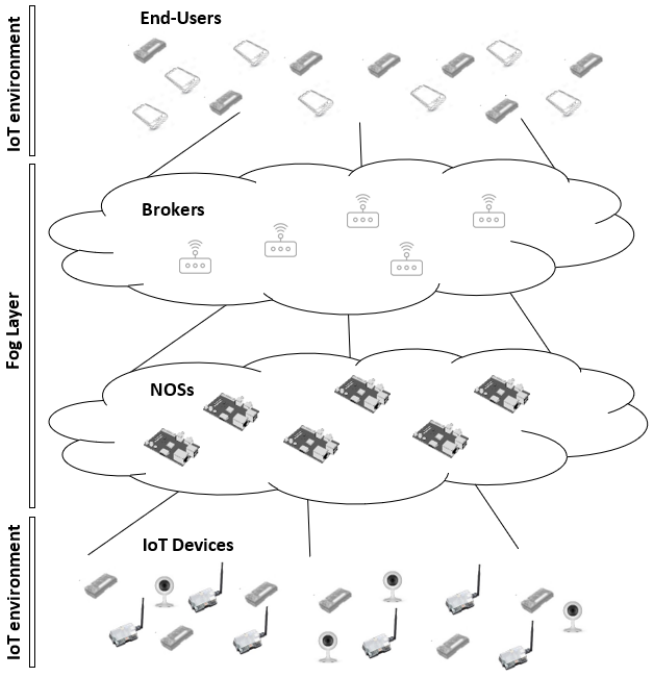

Fig. 3. IoT System composed by a dual fog layer, including multiple NOSs and brokers

IV. Integration of fog computing and MQTT

As partially revealed in the previous section, the proposed solution aims to exploit a hybrid fog layer, composed by both NOSs and brokers, in order to bring the following advantages to the IoT network: (i) turning the broker from being a single point of failure into a network of brokers; (ii) better balancing the data load to be shared with end-users; (iii) reducing the delays of information retrieval; (iv) working in a location-awareness way; (v) giving more robustness to the whole IoT system; (vi) allowing the presence of multiple stakeholders, enabled to exploit the IoT network functionalities.

Note that a broker can be conceived as a piece of software, which is responsible for executing simple operations: receiving all data, filtering them, establishing who is interested in them and then publishing the data themselves to all subscribed users or applications. Hence, a broker does not require to run on powerful computers or servers, but it can be installed on devices, such as Raspberry Pi or Arduino. Such a feature makes it an ideal candidate to be part of a fog layer. The same, as just introduced in Section III, is for NOSs, which are intended to be placed within the IoT environment and to gather information from the near sources. Therefore, a clear separation is created between data acquisition, performed by NOSs (along with processing tasks), and data sharing with users, performed by brokers. In the following, 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_n, respectively. It is worth to remark that the authors decided to not integrate one broker for each NOS for three reasons: (i) preserve NOSs power consumption; (ii) avoid turning NOSs into single points of failure in the whole IoT system; (iii) enabling the participation of third-party brokers. The proportion between NOSs and brokers will depend on the specific applications involved, and it is out of the scope of this work.

A further important requirement to be considered is related to security. In fact, when a user or an application requires a service provided by the IoT system, it is supposed that a session is opened, during which the user/application, identified by $usapp_1$, $usapp_2$, ...$usapp_n$, can obtain the information provided by a NOS, taking into account the accessible resources. The resources can be accessed on the basis of the policies P_{data_1}, P_{data_2}, ...P_{data_n}, defined within NOS enforcement framework, in the format specified in [6]. All NOSs actuate the same policies, thanks to the synchronization mechanism discussed in [21]. As a consequence, all the brokers can manage the incoming information in compliance with the same associated policies, regardless of the NOSs with which they interact. The brokers must also interact with NOSs in order to establish which subscriptions to accept or deny.

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 a previous subscription) by a user/application $usapp_2$, connected to broker br_2, which does not receive any data under

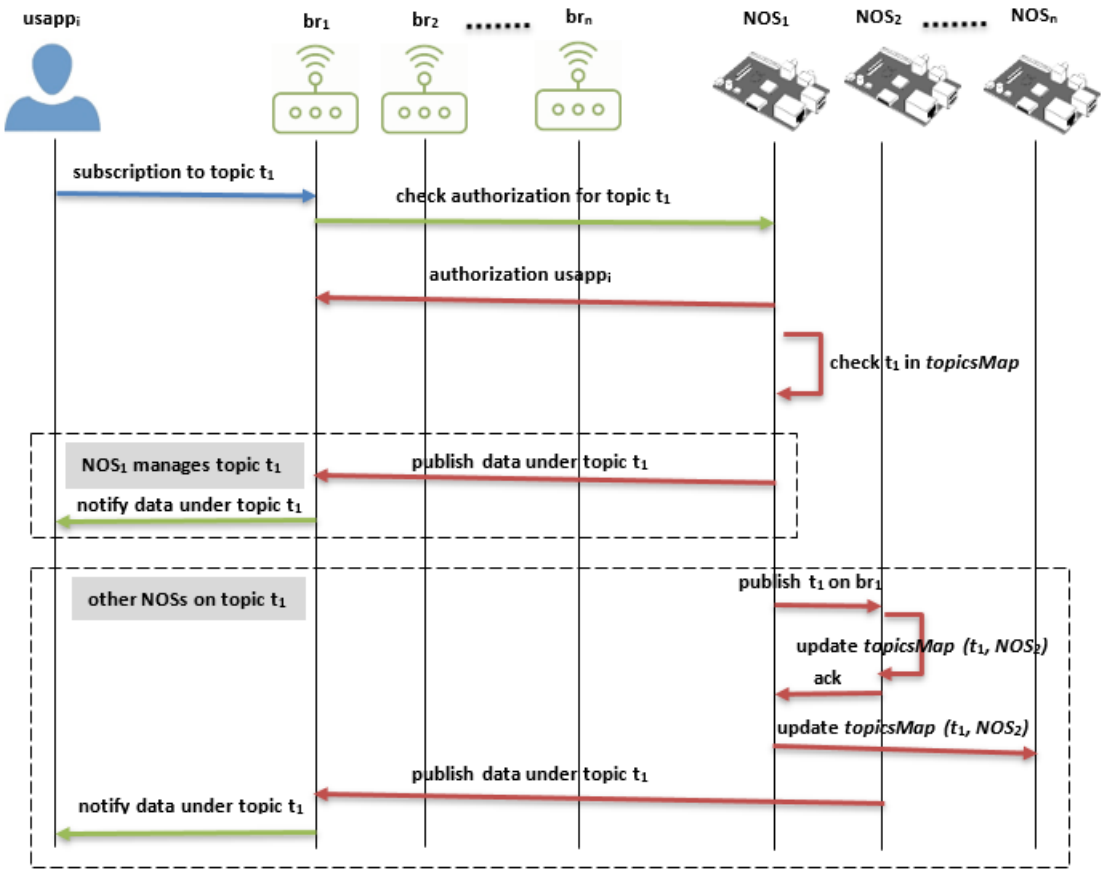

Fig. 4. Scheme related to the IoT system's behavior

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_i 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 the authors can assume 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 4:

- 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:
  - NOS_1 checks if $usapp_i$ is authorized to access the data published under the topic t_1 (i.e., the check is executed on the sticky policy associated to the data under topic t_1)
  - 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
  - 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 III) and contains an entry for each couple topics-NOSs (t_i, NOS_i)
    - If the couple (t_1, NOS_1) exists, some data acquired by NOS_1 and published under the topic t_1 will be notified by br_1 to $usapp_i$, but what happens if other NOSs process information related to the topic t_1?
    - In such a case or in case the entry (t_1, NOS_1) is not found in NOS_1, 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 4) 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 4, 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 [21], in

order not to compromise the *topicsMap*'s content.

Some important features, about the just presented mechanism, must be specified:

- NOSs and brokers must be fully decoupled, in the sense that brokers can be owned by different organizations/companies, which are interested in exploiting the functionalities made available by the IoT platform, to disclose some relevant information to their customers. As a consequence, a company or organization could deploy its own broker and connect to the NOSs' layer (i.e., the IoT platform); using the MQTT protocol, no issues in terms of interoperability arise. Instead, if the broker were installed on NOSs, the presence of external brokers would not be possible. Another opportunity for interested companies/organizations is to hire a broker, provided by another stakeholders; in this way, a broker may manage data from various parties
- All the communications taking place within the presented system are secure, because: (i) users/applications receive ciphered data under specific permissions, defined at the subscription phase and already implemented in [6], as also shown in Figure 1; (ii) NOSs exchange information among themselves on a HTTPS/SSL channel and also NOSs and brokers; (iii) the brokers need not to communicate among each other (in this sense, they are agnostic of each other), since NOSs are in charge of supervisioning how information is shared and, thus, coordinating the brokers' activities
- NOS and broker communicate by means of MQTT protocol, via the MQTT client installed on NOS (see Figure 2)
- 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.

V. Performance analysis

A preliminary analysis about the feasibility of the proposed solution is conducted by means of a test-bed, openly accessible at https://bitbucket.org/alessandrarizzardi/nos.git, composed by two instances of NOS (NOS_1 and NOS_2), running on two Raspberry Pi platforms, and by a variable number of brokers (one or two, namely br_1 and br_2) and data sources, which virtually run on separated virtual machines, installed on a personal computer. The interactions with users are simulated by means of data requests sent to the IoT platform at a certain rate.

The sources use measures from real-world smart home test-bed[3], acquired by means of installed sensors that collect electricity data every minute for the entire home [22]. 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).

[3]http://traces.cs.umass.edu/index.php/Smart/Smart

TABLE I
Test-bed parameters

Parameter	Value
NOSs	2
Brokers	[1, 2]
Sources	4
Topics	4
Data generation rate	10 pck/second
Data request rate	10 req/second
Observation time	24 hours

Wi-Fi connection is 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*[4] platform has been used for developing NOSs' core operations, *MongoDB*[5] has been adopted for the data management, and Mosquitto[6] 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 [4].

The set up of the conducted analysis is summarized in Table I. The assessed metrics are computing effort and latency. The storage overhead has just been deeply analyzed in previous works [6] [21], but 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, but such a possibility is not considered in this paper, since the main goal is the introduction of fog computing features within the NOS's platform.

Figure 5 shows the average distribution of the CPU load on the analyzed NOSs and brokers in two different situations: (i) the IoT system adopting only one broker (i.e., br_1); (ii) running two brokers. 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 are simulated as well, at the same rate, and imply the notification from the brokers; (ii) t_1 and t_2 are associated to br_1, while t_3 and t_4 are associated to br_2; (iii) when an external entity requires, for example, a subscription to t_1 towards br_2, the procedure presented in Section IV must be executed only once; then, the system continues its activity without requiring further adjustments.

[4]http://nodejs.org/
[5]http://www.mongodb.org/
[6]http://mosquitto.org

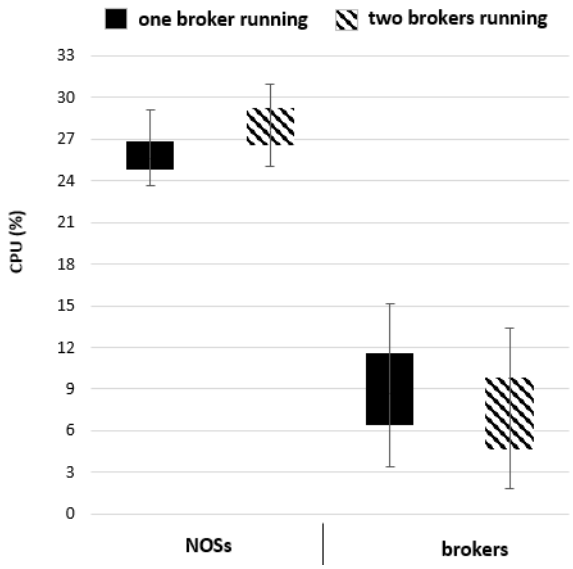

Fig. 5. Whiskers-box diagram: average CPU load on NOSs and brokers

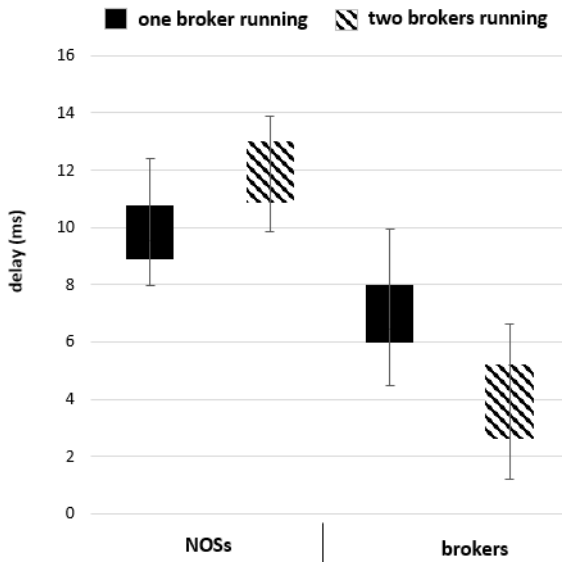

Fig. 6. Whiskers-box diagram: average end-to-end latency

Even if additional studies, covering larger deployments, are needed, the results suggest that having more brokers does not affect, in a relevant way, the performance of the whole IoT system, despite the presence of two brokers requires the execution of the tasks presented in Section IV. The CPU load on NOSs slightly increments, but bringing the advantage to potentially serving a wide IoT area, making it more pervasive; while the computing effort on brokers slightly decreases, since they share the data load.

Considerations are similar for the average time required by data from their transmission towards NOS to their reception by the subscribed entities, shown in Figure 6. Note that the presence of more than one NOS and broker better balances the data load without transfer the information sharing task to one centralized broker.

VI. Conclusion

The paper has presented an IoT architecture, composed by a dual fog layer, involving smart powerful devices (i.e., NOSs), in charge of acquiring, processing and securing IoT-generated data, and brokers, responsible for disseminating information. MQTT protocol has been chosen due to its lightweight primitives, which fit the constraints of IoT devices. The middleware layer, composed by NOSs, has been kept separate from the brokers' network for reducing the latency, better balancing the data load, and allowing the participation of third-party brokers, so as to encourage interested companies/organizations in the exploitation of the IoT infrastructure. A preliminary performance evaluation has been carried out on a simple yet real prototype, but in the near future the authors are planning to test the proposed approach in a wider scenario (e.g., by increasing the number of users), trying to analyze the correlation among the number of NOSs and brokers. More in detail, the authors would investigate how many brokers are required to efficiently manage the information sharing task, in presence of a certain number of NOSs (or, also, in presence of a certain number of topics or data producers/consumers). Also, comparison with other existing approaches would help in understanding the differences between the available IoT systems. Finally, the factors influencing the energy consumption of the different network's components will be explored.

References

[1] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE wireless communications*, vol. 24, no. 3, pp. 10–16, 2017.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[3] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2017.

[4] S.Sicari, A. Rizzardi, D. Miorandi, C. Cappiello, and A. Coen-Porisini, "A secure and quality-aware prototypical architecture for the Internet of Things," *Information Systems*, vol. 58, pp. 43–55, 2016.

[5] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, "Internet of Things: Security in the keys," in *12th ACM International Symposium on QoS and Security for Wireless and Mobile Networks*, Malta, Nov 2016, pp. 129–133.

[6] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, "Security towards the edge: Sticky policy enforcement for networked smart objects," *Information Systems*, vol. 71, pp. 78–89, 2017.

[7] A. Rizzardi, S. Sicari, D. Miorandi, and A. Coen-Porisini, "AUPS: An open source authenticated publish/subscribe system for the Internet of Things," *Information Systems*, vol. 62, pp. 29–41, 2016.

[8] T. Rausch, S. Nastic, and S. Dustdar, "Emma: Distributed qos-aware mqtt middleware for edge computing applications," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 191–197.

[9] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, "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)*, 2012, pp. 36–41.

[10] Y. Xu, V. Mahendran, and S. Radhakrishnan, "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery," in *IEEE 8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1–6.

[11] R. Neisse, G. Steri, and G. Baldini, "Enforcement of security policy rules for the internet of things," in *IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2014, pp. 165–172.

[12] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, and A. Panya, "Authorization mechanism for mqtt-based internet of things," in *IEEE International Conference on Communications Workshops (ICC)*, 2016, pp. 290–295.

[13] S. R. Moosavi, T. N. Gia, E. Nigussie, A. M. Rahmani, S. Virtanen, H. Tenhunen, and J. Isoaho, "End-to-end security scheme for mobility enabled healthcare internet of things," *Future Generation Computer Systems*, vol. 64, pp. 108–124, 2016.

[14] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[15] C. Thota, R. Sundarasekar, G. Manogaran, R. Varatharajan, and M. Priyan, "Centralized fog computing security platform for iot and cloud in healthcare system," in *Exploring the convergence of big data and the internet of things*. IGI Global, 2018, pp. 141–154.

[16] M. Arif, G. Wang, and V. E. Balas, "Secure VANETs: trusted communication scheme between vehicles and infrastructure based on fog computing," *Stud. Inform. Control*, vol. 27, no. 2, pp. 235–246, 2018.

[17] J. Kang, R. Yu, X. Huang, and Y. Zhang, "Privacy-preserved pseudonym scheme for fog computing supported internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2627–2637, 2018.

[18] M.-G. Ionita and V.-V. Patriciu, "Secure threat information exchange across the internet of things for cyber defense in a fog computing environment." *Informatica Economica*, vol. 20, no. 3, 2016.

[19] A. S. Sohal, R. Sandhu, S. K. Sood, and V. Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," *Computers & Security*, vol. 74, pp. 340–354, 2018.

[20] K. S. Aloufi and O. H. Alhazmi, "Performance analysis of the hybrid iot security model of mqtt and uma," *arXiv preprint arXiv:2005.06595*, 2020.

[21] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, "Dynamic policies in internet of things: enforcement and synchronization," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2228–2238, 2017.

[22] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," *SustKDD, August*, vol. 111, p. 112, 2012.