Journal of Network and Computer Applications 174 (2021) 102907

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Regulating data sharing across MQTT environments

Pietro Colombo *, Elena Ferrari, Engin Deniz Tümer

Department of Theoretical and Applied Science, University of Insubria, Via O. Rossi, 9 - 21100 Varese, Italy

ARTICLE INFO

Keywords:
Access control
Internet of Things
MQTT
ABAC
Bridged environments

ABSTRACT

Nowadays, due to the personal nature of the managed data, numerous Internet of Things (IoT) applications represent a potential threat to user privacy. In order to address this issue, several access control models have been specifically designed for IoT. The great majority of these proposals adopt centralized enforcement mechanisms designed to control the communication of IoT devices operating in the same environment. However, these approaches cannot regulate data exchange operated by devices connected to different environments. To the best of our knowledge, effective approaches capable of controlling these forms of communications are still missing. Therefore, in this paper, we do a step to fill this void, by focusing on applications built on top of MQTT, a widely used protocol for IoT. We propose an access control framework to regulate data sharing across bridged MQTT environments, on the basis of both access control policies and user preferences. The proposed approach regulates data exchange among IoT devices belonging to interconnected environments by means of a decentralized enforcement mechanism. Experimental analyses show the efficiency of the proposed approach.

1. Introduction

Internet of Things (IoT) applications are starting to be massively integrated into our lives (Ravidas et al., 2019) due to the indisputable benefits they bring. For instance, by exploiting the pervasivity of wearable technologies, manifold applications assist users during their daily routines (e.g., sport training or health monitoring). Due to the personal nature of the managed data, these applications have been recognized as potential threats to user privacy (e.g., see Ravidas et al., 2019).

In recent years, to cope with this issue, research has deeply analyzed the trade-off between service utility and user privacy (Lee et al., 2018), proposing several data protection solutions. In particular, for what access control is concerned, different approaches have been designed (see Section 7 for a compendium). However, the majority of these approaches aim at regulating the access to data generated and exchanged within a single environment (e.g., La Marra et al., 2017b; Colombo and Ferrari, 2018).

In contrast, an increasing number of IoT applications rely on IoT devices distributed in multiple environments. This has resulted in research work proposing frameworks on support of distributed IoT scenarios. For instance, Wang et al. (2018) propose a collaborative edge computing framework for vehicular networks, which, by means of inter-environment and intra-environment collaborations, allows data sharing among network edges. Similarly, Zhang et al. (2016) propose an edge computing framework, which, by means of virtual views of the data built by data owners for specific end-users, allows users of different environment to share data.

Distributed approaches favor better performance and allow the delivery of more advanced services to users, but, at the same time, bring serious security/privacy threats, as they extend the scope of the sensed data to multiple environments.

In this paper, we do a first step to address this issue, by proposing an Attribute Based Access Control (ABAC) framework to regulate data sharing among interconnected MQTT[1] environments. Due to the pervasivity of many IoT scenarios, the proposed approach provides support for fine grained, context based access control policies. For instance, let us consider the case of an Internet of Sport (IoS) application, say MyPersonalTrainer, developed for MyGymBrand, a brand of affiliated sport halls. MyPersonalTrainer allows gym frequenters to share data generated during their training sessions. In particular, when deployed on exercise bikes, the app allows gym frequenters to participate to cycling races, and share data, such as the current speed, and the covered distance. In such a scenario, it would be useful to specify an access control policy that regulates data sharing across gyms in order to limit the exchange of rider performances to the race duration, granting the access only to the set of users registered for the race.

In addition, to enhance user control on the data generated by the administered devices, our framework allows users to specify their

* Corresponding author.
E-mail addresses: pietro.colombo@uninsubria.it (P. Colombo), elena.ferrari@uninsubria.it (E. Ferrari), edtumer@uninsubria.it (E.D. Tümer).

[1] http://docs.oasis-open.org/mqtt/.

https://doi.org/10.1016/j.jnca.2020.102907
Received 22 May 2020; Received in revised form 7 October 2020; Accepted 3 November 2020
Available online 7 November 2020

own preferences, which might restrict the privileges granted by access control policies. For instance, rider Mary wishing to enforce a stricter privacy protection, may specify a user preference that, during a competition, filters out the identifiable data from the data shared with the riders of other gyms.

The choice of focusing on MQTT environments is motivated by the diffusion of this protocol, which is nowadays used in a wide variety of IoT scenarios (e.g., automotive, manufacturing, telecommunications).

The framework proposed in this paper extends the one proposed by us in Colombo and Ferrari (2018), where we design an ABAC model to control the communication of devices operating in a single MQTT environment.

The enhanced framework proposed in this paper has required a substantial extension of the original framework. The key contribution is the *decentralized* approach to enforce access control policies and user preferences, which supersedes the centralized enforcement mechanism proposed in Colombo and Ferrari (2018). Policies and preferences can be specified in each side of a pair of interconnected MQTT environments, and are enforced by means of the joint work of monitors deployed in each environment and along the environments bridge. More precisely, access control policies regulating messages that can enter or leave an environment are enforced by a monitor operating at the interface of the environment where these policies have been specified. In contrast, user preferences can also be enforced in environments different from the ones where the preferences have been originally specified. This is an important requirement as data originated from IoT devices could span the boundaries of the organization where they have been originally collected. For instance, Bob, who is a rider of the sport hall MyGym, may specify a user preference that allows gym coaches of any associated sport hall to only access his/her performance data. It is worth noting that the enforcement of user preferences is a quite complex task. The reasons are manifold. For instance, in case a message protected by a user preference *up* is forwarded from a local to a remote environment, the enforcement monitor of the remote environment must be made aware of the specified preference. In addition, *up* can refer to the context within which the message is going to be accessed, as well as to the publishing context. For instance, *up* could authorize the read access to a message if performed within 1 s from the publishing. To enforce a similar user preference the monitor of the remote environment must be made aware of the message publishing time in the local environment.

The enforcement monitor that regulates data sharing, operates as an MQTT broker proxy that alters the communication flow between bridged environments. The monitor can be easily integrated into existing MQTT deployments, with basic configuration activities. The efficiency of the monitor prototype has been experimentally assessed, showing a reasonably low enforcement overhead in different testing scenarios.

To the best of our knowledge, the framework proposed in this paper is among the earliest edge-based access control approaches that allow regulating data sharing across interconnected IoT environments. We are only aware of another pioneering work by Fuentes Carranza and Fong (2019), which proposes an approach to regulate the interaction of message brokers of different environments. However, Fuentes Carranza and Fong (2019) does not support context based policies, operates at coarse grained level, and does not allow users to customize data sharing on the basis of their privacy preferences.

Contributions. To summarize, the main contribution of this paper is a decentralized approach to regulate data sharing across bridged MQTT environments. This is articulated in multiple points:

– Access control policies and user preferences to regulate data sharing across bridged MQTT environments, which can be specified in any environment of the interconnected pair.
– A new enforcement monitor which regulates the messages that can enter/leave an environment.
– A decentralized enforcement mechanism, leveraging on the joint work of monitors deployed in each environment of a bridged pair, and along the bridge.
– An experimental evaluation of the framework performance

Structure. The remainder of the paper is organized as follows. Section 2 shortly presents background information related to MQTT, whereas Section 3 presents the adopted access control model. Section 4 provides an overview of the proposed solution. Section 5 introduces the enforcement mechanism. In Section 6 we present our experimental evaluation. Section 7 surveys related work. Finally, Section 8 concludes the paper.

2. Background

This section describes basic concepts related to MQTT, and core aspects of the access control framework proposed in Colombo and Ferrari (2018), which allows regulating clients communication within a single MQTT environment.

2.1. MQTT

MQTT enables peers communication under the publish/subscribe architecture. MQTT *clients* communicate with other clients through a *message broker*. A client can either request the broker to publish a message on a *topic*, or to subscribe the receiving of messages on topics matching a *topic filter* expression. Brokers forward messages by means of topic based routing criteria. On receipt of a publishing request on a topic *t*, the broker analyzes the topic filters of the active subscriptions, and forwards the message to any client who has subscribed a topic filter expression that matches *t*.

Example 1. Let us consider again the Internet of Sports (IoS) app MyPersonalTrainer, introduced in Section 1. Let us suppose that MyPersonalTrainer allows gym frequenters who are enrolled in a training session *ts* to share their running performance. In particular, suppose that the smart treadmills of the sport halls MyGym and RemoteGym, which integrate multiple sensors and a tablet, also host an instance of MyPersonalTrainer which has been configured to publish messages specifying runner performances on the following topics:

tr/performance/ts/speed, *tr/performance/ts/avgspeed*, *tr/performance/ts/length*, *tr/performance/ts/duration*, *tr/performance/ts/hearthbeats*, where *tr* and *ts* are placeholders for a treadmill and a training session identifier, respectively. The same devices have also been configured to subscribe the receiving of performance data of all users attending *ts*, specifying *+/performance/ts/+* as topic filter. In addition, the app deployed on gym coaches' tablets allows them to monitor performance data of gym frequenters. The app has been configured to subscribe the topic filter *+/performance/#*, so that coaches can receive performance data published by any treadmill within any training session.

Clients interact with the broker exchanging control packets. The list of packets supported by MQTT is shown in Table 1.

On receipt of a connection request cp_{CN} from a client *c*, a broker opens the connection and acknowledges *c* of the established connection issuing a cp_{CA} packet. On receipt of cp_{CA}, *c* may request: (i) to *publish* a message, issuing a packet cp_{PB}, (ii) to *subscribe* the receiving of messages on topics that match a *topic filter*, issuing a packet cp_{SB}, (iii) to *unsubscribe* a previously subscribed topic, sending a packet cp_{US}, or (iv) to *disconnect* from the broker, issuing a packet cp_{DS}.

An MQTT broker can be configured to connect to remote brokers, enabling the communication of clients belonging to different environments. Brokers supporting these communication forms are hereafter referred to as *bridging brokers*. A bridging broker is therefore a broker configured to connect to a target remote broker with which it shares messages published by clients of the respective environments. The interaction is regulated by means of *bridging rules*. A bridging rule *br* specified for a bridging broker *bb* is a tuple ⟨*rb, tp, dr, qos, lp, rp*⟩, where

Table 1
MQTT control packets.

Control packet	Acronym	Description
CONNECT	cp_{CN}	Client requests a connection to a server
CONNACK	cp_{CA}	Acknowledge connection request
PUBLISH	cp_{PB}	message
PUBACK	cp_{PA}	Publish acknowledgment
PUBREC	cp_{PRC}	Publish received
PUBREL	cp_{PRL}	Publish release
PUBCOMP	cp_{PC}	Publish complete
SUBSCRIBE	cp_{SB}	Subscribe to topics
SUBACK	cp_{SA}	Subscribe acknowledgment
UNSUBSCRIBE	cp_{US}	Unsubscribe from topics
UNSUBACK	cp_{UA}	Unsubscribe acknowledgment
PINGREQ	cp_{PRQ}	PING request
PINGRESP	cp_{PRS}	PING response
DISCONNECT	cp_{DS}	Disconnect notification

rb denotes the remote broker representing the target of *bb*, *tp* is a topic filter expression, specifying the topics of the messages to be shared by *bb*; *dr* specifies the sharing direction (*in*, *out*), which is *in*, if *br* allows *bb* to receive messages from *rb*, or *out*, if *br* allows *bb* to send messages to *rb*; *qos* specifies the QoS level to be used for the message exchange; finally, *lp* and *rp* are local and remote prefixes, which are used by *bb* to remap the topic of outgoing and incoming messages, respectively. If *br* specifies *in* as sharing direction, *bb* prepends *tp* with *rp* and subscribes to the resulting topic on *rb*. On receipt of a message *m* from *rb* whose topic matches the subscribed topic, *bb* substitutes the remote prefix *rp* prepending the topic of *m* with *lp*, and forwards *m* to the local clients who subscribed to a matching pattern. In contrast, if *br* specifies *out* as sharing direction, *bb* prepends *tp* with *lp* and subscribes the resulting topic on the local broker. On receipt of a message *m* published by a local client, whose topic matches the subscribed topic, *bb* substitutes the local prefix with *rp*, and forwards *m* to *rb*.

Example 2. Let us consider again the scenario presented in Example 1, and suppose that MyPersonalTrainer also enables users to share performance data with a remote analysis server, that allows comparing them with previously tracked data of the same user, of users attending the same course, of frequenters of the same gym, or runners attending any sport hall of MyGymBrand. Suppose that MyGym hosts a broker, which handles the communication among its gym apparatus, as well as with brokers of the associated gym RemoteGym, and of the data analyzer environment, which hosts the remote analysis server.

Examples of bridging rules are the following:
br_1 = ⟨*Analyzer, +/performance/+/speed, out, 0, “”, MyGym*⟩, br_2 = ⟨*RemoteGym, +/performance/+/speed, in, 0, “”, “”*⟩.

br_1 enables the forwarding to the analyzer environment of messages specifying the current speed of a runner that attends a training session. The remote prefix *MyGym* denotes the provenance of the message. The remote analysis server is a client of the analyzer environment which subscribes the receiving of messages on topics that match the filter *+/+/performance/#*, which allows the server to access performance data of the frequenters of any gym. In contrast, on the basis of br_2, MyGym broker subscribes the receiving of messages from RemoteGym that refer to the speed of a runner attending any training session of the associated sport hall.

2.2. Access control enforcement within a single MQTT-based IoT environment

We now summarize the approach we proposed in Colombo and Ferrari (2018), to regulate MQTT clients communication in a single environment. The solution relies on an ABAC model for MQTT environments, and consists of an enforcement mechanism designed for the model, and implemented by an enforcement monitor, which can be easily integrated into MQTT-based environments.

2.2.1. The ABAC model

The choice of ABAC (Hu et al., 2015) has been mainly based on its diffusion, and well founded flexibility (e.g., see Jin et al., 2012). ABAC is based on the concepts of subject, object, and environment, which respectively model: (i) a subject who sends access requests, (ii) a protected resource, and (iii) the context within which an access request has been issued.

In our context, ABAC is used to regulate the reception and the publishing of messages, on the basis of access control policies and user preferences.

MQTT clients, possibly on behalf of a user, connect to a broker with the aim to exchange messages with other clients. Once connected, a client can request to publish a message on a topic, or to subscribe the receiving of messages on topics that match a topic filter (see Section 2). Clients are therefore *subjects* requiring to access messages (i.e., to publish or receive), which represent protection *objects*, within an *environment*, namely a model of the execution context from which the access request has been issued.

A subject *s* is characterized by attributes that specify properties related to the client, and possibly the user on behalf of whom the client sends access requests. Since MQTT clients requiring to connect to a broker need to specify a client identifier, *s* includes an attribute *cid* specifying the client identifier. Additional attributes can then be introduced, such as the attributes *uid* and *rid* that model the identifier and role of the user on behalf of whom the client sends publishing or subscription requests. Attributes can also refer client properties. For instance, an attribute *dev* may be used to characterize the device that hosts a client.

Application messages represent protection *objects*. Since MQTT brokers route messages on the basis of the message topics, an object *o*, modeling a message *m*, includes an attribute *tp*, that specifies the topic of *m*. However, other attributes can be introduced as well to specify additional properties of the message, such as, for instance, the category of data in the payload (e.g., sensitive, personal or generic, cfr. Colombo and Ferrari, 2015).

Depending on the considered application scenario, a message delivering context can be modeled with different sets of attributes, which may represent, for instance, time, location and purposes. For the sake of simplicity, in this paper, we consider an environment *e* characterized by a single attribute *t* that models the message delivery instant.

The model proposed in Colombo and Ferrari (2018) supports *access control policies*, specified by security administrators, as well as user defined policies, denoted as *user preferences*, which restrict the privileges granted by access control policies. Access control policies grant subjects the read or write access to messages on given topics. Read privilege represents the subject right to receive messages on a subscribed topic, whereas the write privilege allows a subject to publish a new message. User preferences enable highly customized forms of data protection, serving single user needs. Indeed, multiple users may publish data on the same topic, but each user can have a different perception of the sensitivity of the published data, and may desire to protect the access to his/her data accordingly.

Example 3. Let us consider our running example. Access control policies are used to constrain the type of data which can be published and received on behalf of MyGym frequenters, independently from a specific running session. For instance, an access control policy may authorize MyGym coaches to access, through their app, any information that is published by frequenters of the gym, and frequenters to only see the average speed, running length, and duration of other runners registered for the same gym course. However, users may wish to restrict the privileges granted by access control policies, constraining, through user preferences, the set of data which can be shared during a running session (e.g., the current and average speed), as well as the receivers of these shared data. For instance, a user preference specified by Bob, a frequenter of MyGym, may forbid John, a coach of MyGym, to access the tracked data.

Access control policies and user preferences are specified with predicates defined over subject, object and environment attributes, which constrain the contexts within which the privileges specified by access control policies and user preferences can actually be exercised.

Definition 1 (*Parametric Predicate*)**.** A parametric predicate is a boolean expression built by composition of subject, object and environment attributes, mathematical operators ($>$, $<$, $=$, $+$, $-$, $*$, $/$, %), logical operators ($\wedge$, $\vee$, $\neg$), set operators ($\in, \subset, \subseteq, \cap, \cup, \setminus$), logical quantifiers ($\forall$, $\exists$), and predefined functions allowing the processing of attributes values.

Definition 2 (*Access Control Policy*)**.** An access control policy p is a tuple $\langle s, tf, exp, pr\rangle$, where s refers the subjects constrained by p,[2] tf specifies a topic filter expression, exp is a parametric predicate,[3] whereas pr specifies the *read/write* privileges granted to s if exp is satisfied.

Example 4. Let us consider a policy p_1 which grants frequenters enrolled to a gym course the privilege to publish performance data and read performance data of other trainees, and a policy p_2, which allows gym coaches to access performance data of gym frequenters during their working hours. p_1 can be specified as: $\langle$*frequenter, +/performance/ts/+, isEnrolled(s.sid),rw*$\rangle$, where *isEnrolled(s.sid)* is a function that checks whether the subject *s.sid* is enrolled to a gym course, whereas p_2 is modeled as $\langle$*gymcoach, +/performance/+, isWorkingTime(t,s.sid), r*$\rangle$, where *isWorkingTime(t,s.sid)* is a function that checks whether the time referred to by t matches the work shift of the subject.

User preferences constrain the access to messages published by a user, on the basis of parametric predicates.

Definition 3 (*User Preference*)**.** A user preference up is a tuple $\langle$*uid, tf, sub_exp*$\rangle$, where *uid* specifies the identifier of a user who wishes to protect the access to messages published by any of the clients it handles, whereas tf specifies a topic filter expression which denotes the messages, among those that have been published on behalf of *uid*, to which the preference applies, whereas *sub_exp* is a parametric predicate specifying a precondition to the receiving of a protected message to be satisfied by the subscribed clients.

Example 5. The user preference $up_1 = \langle$*Mary, +/performance/ts/+, s.rid = "coach"* $\wedge$ *s.uid = "Alice"*$\rangle$ specified by Mary, a frequenter of MyGym limits to Alice, a coach of MyGymBrand, the privilege to access Mary's performance data related to the training session *ts*.

User preferences restrict the read access privileges which are granted by access control policies. Therefore, a subscriber subject s can read, at time t, a message m published by a user u on a topic tp iff: (i) there exists an access control policy p that grants s read access to tp at time t, and (ii) there exists at least one user preference up among those specified by u with a topic filter expression that is matched by tp, which grants s read access to tp.

2.2.2. Access control enforcement

Policies and preferences are enforced through a monitor that operates as a proxy between an MQTT message broker and multiple clients, and a key–value datastore, which manages access control policies and user preferences to be enforced by the monitor. A high level view of the system architecture is shown in Fig. 1.

A key advantage of the proposed enforcement approach is that it does not rely on ad-hoc defined clients or brokers, as it supports any client and broker complying with the MQTT specification. The enforcement monitor is the only component which can connect to the server, on behalf of the client. The approach can be used in heterogeneous environments composed of different versions of clients and brokers.

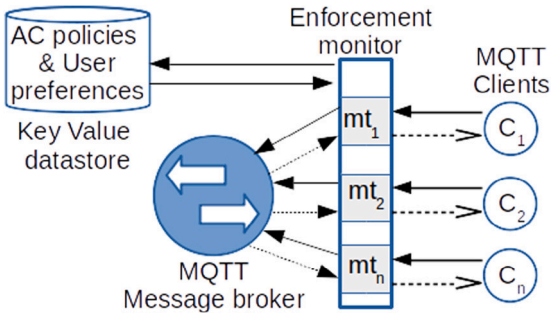

Fig. 1. A high level view of the system architecture in Colombo and Ferrari (2018).

The monitor intercepts any control packet issued by the clients and the broker, and forwards them to the respective receivers, except for packets representing connection and publishing requests, for which additional activities are executed. Connection requests trigger subject profiling activities, and the opening of communication channels with the requesting client and the broker, which are controlled by the monitor. Message publishing requests issued by clients via monitoring channels are handled by initially deriving the policies that regulate the publishing of messages by the requesting subject. If no policy is satisfied the request is blocked. In contrast, if the publishing operation complies with at least one policy, the monitor derives the preferences specified by the user on behalf of whom the publishing has been requested. The monitor then embeds the derived preferences into the payload of the publishing request, and forwards the packet to the local broker.

Publishing requests issued by the broker are handled in a similar way. Any broker request is received through a previously opened communication channel connecting the broker to a subscriber. The monitor extracts attributes and user preferences from the packet's payload, derives the subject, object and environment attributes modeling the context within which the message should be received by the subscriber, and evaluates the preferences wrt the derived attributes. If no preference is satisfied, the request is blocked, whereas if at least one preference is satisfied the monitor derives the applicable access control policies. In case the read access complies at least with one of the policies, the monitor removes from the payload all previously embedded metadata and forwards the packet to the subscriber, otherwise it blocks the request. More details can be found in Colombo and Ferrari (2018).

3. Access control across different MQTT environments

To manage data sharing across different environments, we enhance the model in Colombo and Ferrari (2018), with the ability to regulate message exchange among MQTT publishers and subscribers belonging to different environments.

In what follows, we refer to a scenario where two MQTT environments, respectively referred to as *local* and *remote*, communicate with each other by means of *interconnected brokers*. We denote two brokers as interconnected when one of them has been configured as bridging broker and specifies the other one as connection target.

The main differences with the model presented in Colombo and Ferrari (2018) are related to subject, access control policies and user preferences.

A subject s can denote : (i) a client of the local environment who connects to the local broker with the aim to publish or receive

[2] For the sake of simplicity, in this work s can refer to a client, user or role identifier. However, the approach can be extended providing support to intensional binding expressions defined by composition of subject attributes.

[3] If no restriction is required, the predicate should specify a tautology.

messages, possibly on behalf of a user,[4] (ii) it can be a local broker of one environment which aims at exchanging application messages with a remote environment, or (iii) a bridging connection through which a bridging broker communicates with a remote broker. A local broker may forward messages published by its local clients to a remote broker, as well as receive messages published in a remote environment, which then will be forwarded to the rightful subscribers of its environment.

As in Colombo and Ferrari (2018), access control policies are specified to grant subjects the read or write access to messages referring to a set of topics. Like in Colombo and Ferrari (2018), if the subject is a client, the read privilege represents the right to receive messages on a subscribed topic, whereas the write authorization specifies the right to publish a new message. In contrast, if the subject is a broker or a bridging connection, a read authorization represents the right to receive messages that have been published in a remote environment, whereas a write authorization specifies the right to forward messages to a remote environment. Policies granting read/write privileges to a broker are specified by the security administrators of the environment that hosts the broker. Although access control policies that regulate broker activities show some similarities with bridging rules, they differ from the latters in that bridging rules: (i) can only grant authorizations to bridging brokers, whereas access control policies can also target standard (non bridging) brokers, (ii) route messages only on the basis of their topics, whereas access control policies enable a context aware ABAC based forwarding regulation, and (iii) are used to configure topics remapping abilities of bridging brokers.

Definitions 1 and 2, are still applicable to formalize the concepts of parametric predicate and access control policy. The only difference is that component s of p in Definition 2 can also refer to a broker, or to a specific connection of a bridging broker. If s refers to a broker, the read/write privilege granted by p models the broker privilege to receive/forward messages. If s refers to a bridging connection, the granted privilege applies to the bridging broker that handles the connection specifying the broker communication ability for this connection.

Example 6. Let us now focus on policies regulating brokers interaction specified for our running example. The policy p_3 = ⟨*MyGym, +/performance/ts/+, isOpeningHour(t), rw*⟩ authorizes MyGym's broker to forward and receive messages encoding performance data of runners attending a training session *ts* during MyGym's opening hours. The privilege applies to any bridging connection, as no bridging connection is explicitly referred to within p_3.[5] A similar policy, say p_4, can be specified for the RemoteGym's broker as ⟨*RemoteGym, +/performance/ts/+, isOpeningHour(t), rw*⟩. In contrast, a less restricting policy can be specified for the Analyzer's broker as p_5 = ⟨*Analyzer, +/performance/#, true, r*⟩. This policy authorizes the Analyzer's broker to receive, from any MyGymBrand's gym, messages on topics referring to performance data of any training session.

Users can further constrain the access to messages published on their behalf, through user preferences. In the extended model, user preferences can either restrict the read access to published data by clients, or brokers forwarding privileges.

For instance, users may specify user preferences requiring that their running data are only shared within the sport hall where data have been sensed, without being tracked, analyzed, or forwarded to frequenters of other gyms.

[4] MQTT allows specifying a user name within CONNECT control packets (see Table 1), with the aim to support authentication mechanisms. However, user names should not be mandatorily specified.

[5] Alternatively, the pair of policies p_{3a} = ⟨*MyGym.RemoteGym, +/performance/ts/+, isOpeningHour(t), rw*⟩, and p_{3b} = ⟨*MyGym. Analyzer, +/performance/ts/+, isOpeningHour(t), w*⟩ might have been used to constrain MyGym's broker ability to communicate through its bridging connections.

In order to allow users to restrict brokers forwarding privileges, the user preference definition (see Definition 3) has been enhanced with an additional component, denoted as *bp*, which allows specifying the forwarding preference. *bp* may refer to: (i) the name of a target environment *te*, if *up* constrains the forwarding to *te* only, (ii) *, if *up* constrains the forwarding to any remote environment, or (iii) ⊥, if *up* applies to the read access to the referred messages by subscriber clients. On the basis of *bp*, the parametric predicate *exp* specifies a precondition: (i) to the forwarding of a protected message to a remote environment, or (ii) to the receiving of the message by rightful subscribers.

Example 7. The user preference up_1 introduced in Example 5 can be redefined as: up_1 = ⟨*Mary, +/performance/ts/+, ⊥, s.rid = "coach"* ∧ *s.uid = "Alice"*⟩. It is worth noting that during Mary's training session, depending on the work shift, Alice may be working within MyGym or RemoteGym, however, the effect of up_1 is independent from the gym where Alice works when Mary is training. In addition, Mary, who does not agree to be tracked by the analysis server, specifies the preference up_2 = ⟨*Mary, +/performance/ts/+, Analyzer, false*⟩, which prohibits the sharing of her training session data with that server. Let us also consider the user preference up_3 = ⟨*Bob, +/performance/ts/+, RemoteGym, false*⟩, specified by Bob, a frequenter of MyGym. up_3 prohibits the forwarding of Bob's performance during the training session *ts* to RemoteGym.

User preferences restrict the privileges which are granted by access control policies, constraining the reading privileges of subscribed clients, and the broker ability to forward messages to a broker of a different environment. More precisely, a subscriber subject s of a given environment re can read a message m published on a topic tp by a publisher subject ps operating within re on behalf of a user u iff within re: (i) there exists an access control policy p that grants s read access to tp and (ii) either (a) there exists at least one user preference up, among those specified by u with a topic filter expression that is matched by tp, which grants s read access to tp, or (b) no user preference up specified by u refers to a topic filter expression that is matched by tp. In case s and ps belong to different environments, additional checks are required to allow the forwarding of m from the environment of ps to the one of s. Let us denote with se and pe the environments of s and ps, respectively, whereas b_{pe} and b_{se} denote the brokers of these environments. b_{pe} can forward to b_{se} a message m published on a topic tp by a subject ps of pe on behalf of a user u iff within pe: (i) there exists an access control policy p_w that grants b_{pe} the write access to tp, and (ii) either (a) there exists at least one user preference up among those specified by u with a topic filter expression that is matched by tp, which grants b_{pe} write access to tp, or (b) no user preference up specified by u has a topic filter expression that is matched by tp and grants b_{pe} write access to tp. A message m forwarded by b_{pe} can be received by b_{se} iff within se there exists at least one access control policy p_r that grants b_{se} read access to tp.

Example 8. Let us suppose that Mary and Bob are registered to a course of MyGym, and that they are attending the training session *ts*, which, within RemoteGym, is coached by Alice. Let us assume that Bob and Mary are running on the treadmills tr_1 and tr_2, which have been configured to share performance data of all runners attending the training session (see Example 1). The publishing and receiving of data is regulated by policy p_1 (cfr. Example 4), which grants any frequenter that is enrolled to a gym course and is attending a training session *ts*, the privilege to publish messages specifying his/her performances, and to receive messages over topics specifying performance data of other runners attending *ts*. The publishing by tr_1 and tr_2 of messages referring Mary and Bob performances during *ts*, complies with p_1, therefore, both tr_1 and tr_2 are authorized to publish. However, the user preference up_1 specified by Mary, constrains the receiving of messages published by tr_1 (see Example 7), as no frequenter or coach, but Alice, can access Mary's data. Therefore, on the basis of the applicable policies

and preferences, during *ts*, Mary can see Bob's performance, whereas Bob cannot see Mary's data. According to policy p_3 (see Example 6) performance data published by tr_1 and tr_2 can be forwarded by MyGym broker to any remote environment. However, the forwarding of Mary's and Bob's data is constrained by the user preferences up_2 and up_3 (see Example 7). More precisely, due to up_2, Mary's data cannot be sent to the Analyzer's broker, however, no preference prohibits the forwarding to RemoteGym. In contrast, on the basis of up_3, Bob's data cannot be forwarded to RemoteGym, but no restriction has been specified for Analyzer. The receiving of Mary's forwarded messages by the RemoteGym's broker is regulated by policy p_4 (see Example 6), which authorizes, during the opening hours of RemoteGym, the receipt of messages over performance related topics referring to the training session *ts*, which have been forwarded by MyGym's broker. Similarly, according to p_5, Analyzer can receive messages over performance related topics of any training session which have been forwarded by MyGym's broker, thus it can also receive Bob's data.

Once the forwarded message *m* is received by b_{se}, *m* can be dispatched to the rightful subscribers of *se*. More precisely, *m* can be accessed by a subscriber subject *s* of *se* iff: (1) within *se* there exists at least one access control policy p_r which grants *s* the read privilege to messages on topics that match a topic filter expression that is also matched by *tp*, and (2) the access complies with at least one of the user preferences specified by the original publisher of *m* within *pe*.

Example 9. Let us now focus on the routing, within the RemoteGym environment, of messages on performance related topics, which have been originally published by treadmill tr_1 on behalf of Mary, and then forwarded to the RemoteGym environment by MyGym's broker (see Example 8). Alice's app, has been configured to subscribe the receiving of performance data published by any device (see Example 1). However, the receipt is regulated by the applicable access control policies and user preferences. More precisely, the receiving requires the joint satisfaction of the policy *p2* (see Example 6), which grants Alice the privilege to receive messages on performance related topics during her working shift, and the user preference up_2 (see Example 7), specified by Mary, which grants the access to Alice only. Due to up_1, no client within MyGym and RemoteGym is authorized to access Mary's data, but those requesting the access on behalf of Alice.

4. Overview

As explained before, we target an application scenario where two MQTT environments, each composed of multiple MQTT clients and a local broker through which these clients communicate, need to share data. Inter-environment communication is achieved by configuring the local broker of one of the two environments as a bridging broker specifying the other broker as connection target.

The proposed framework enhances the approach introduced in Colombo and Ferrari (2018), which targeted the regulation of message flows within a single environment, with the ability to regulate the communication of MQTT clients of different IoT environments. More precisely, the enforcement monitor proposed in Colombo and Ferrari (2018), denoted in what follows as *local* monitor, has been enhanced to enforce user preferences possibly specified within a different environment. Additionally, a new enforcement monitor, denoted as *bridging monitor*, has been designed, which, on the basis of access control policies and user preferences, regulates message passing between interconnected brokers.

The proposed approach requires to interpose: (1) the local monitor between the local clients and the respective brokers; (2) the bridging monitor in between the brokers of the interconnected environments. More precisely: (1) the clients of each environment are configured to connect to the respective monitors rather than directly to their brokers, (2) the local monitors are connected to the respective local brokers, (3) the local broker, which has been configured as a bridge, specifies the bridging monitor as connection target, and (4) the bridging monitor specifies the local broker of the other environment as connection target. As such, the local monitors of the connected environments behave as proxies of the respective brokers, and are the only components that can communicate with the brokers on behalf of their clients. In contrast, the bridging monitor behaves like a proxy of the broker referred to as connection target of the local bridging broker, and is the only component enabling the communication between different environments. Fig. 2 presents an high level view of the system architecture. The designed approach does not require ad-hoc implementations of clients and brokers, and it is independent from specific client and broker versions. The local and bridging monitors can also operate within heterogeneous environments, where different versions of clients and brokers cooperate.

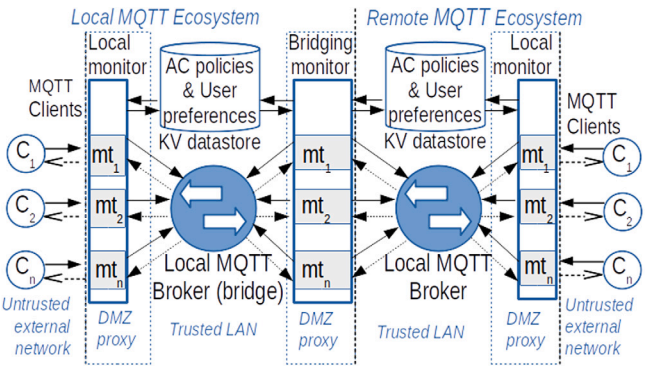

Fig. 2. A high level view of the system architecture.

According to the system architecture shown in Fig. 2: (1) local MQTT brokers are deployed in different trusted LANs, whereas (2) MQTT clients are deployed in untrusted external networks, (3) whereas the local monitors and the bridging monitor are hosted by DMZ proxies at the LAN interfaces. Firewalls placed at the interfaces of these three DMZ proxies are configured to prohibit unmediated connections with the local brokers.

The activities of the local and bridging monitors are subject to access control policies and user preferences (cfr. Section 3) regulating the communication within each environment, and between interconnected environments. Access control policies and user preferences are handled by key–value datastores, deployed in the interconnected environments. The policy set handled by each datastore allows regulating the internal communication in the respective environment, as well as the messages exchanged with the other environment. The bridging monitor accesses the policy sets of both datastores, whereas the local monitors only access the set in the datastore of the respective environment.

5. Enforcement

In order to ensure that the internal message flows, as well as those across different environments, comply with the specified access control policies and user preferences, the local monitors analyze and possibly alter the flow of MQTT control packets exchanged by their clients, whereas the bridging monitor performs similar operations on the flow of packets exchanged by the brokers of the two environments. In the remainder of this section, we first present the rationale of the enforcement mechanism, and then we focus on its specification details.

5.1. Enforcement rationale

The message passing within a target environment is regulated by instances of the enforcement monitor proposed in Colombo and Ferrari (2018) and summarized in Section 2.2.2, which have been enhanced to

enforce user preferences possibly specified within a different environment. The enhanced version of the monitor differs from the original one for the management of security metadata, instrumental to access control. Indeed, different from Colombo and Ferrari (2018), during the analysis of client publishing requests, along with the user preferences, the monitor embeds into the analyzed packet's payload the subject, object, and environment attributes that model the context within which the publishing request has been issued. These embedded attributes are necessary as user preferences could refer to the context within which a message has been originally published, along with the context within which the message is accessed. For instance, in our running case a user preference specified by a gym frequenter could introduce a time to live constraint that authorizes the read access to performance data within 2 s from the publishing, thus requiring to keep track of the publishing time. Similarly, a user preference could also refer to attributes related to the specifying subject. For instance, a preference specified by gym frequenter could restrict the access to his/her performance data to frequenters of the same gym or of other gyms who are using the same type of exercise machine (e.g., a treadmill or a rowing machine). The evaluation of this preference requires the access to a subject attribute which denotes the exercise machine employed by the user who has specified the preference. Finally, a user preference could also refer to object attributes, such as the topic of a message which in turn denotes the type of data sensed by an exercise machine (e.g., step rate). Since a bridging broker could remap message topics when forwarding a message to a bridged environment (e.g., it could prepend the identifier of the environment where the message has been originally published, as explained in Section 2), object attributes that keep track of the original message topic are required to evaluate the preference within the environment that has received the forwarded message.

During the analysis of publishing requests the monitor extracts all these attributes along with user preferences from the packet's payload, deriving the context in which: (i) the message has been originally published, and (ii) should be received by the subscriber. The monitor evaluates the embedded preferences wrt all derived attributes. Finally, before issuing a message to a rightful and authorized subscriber, the monitor removes from the packet's payload all previously embedded metadata.

Let us now focus on the communication between environments, which is regulated by the bridging monitor. By assumption, the local broker of one of the two environments has been configured as a bridging broker specifying the bridging monitor as connection target.

Message sharing between the two environments is subject to the bridging rules that configure the communication abilities of the local brokers (cfr. Section 2). Hereafter, we refer to bridging rules whose component *dr* has been set to *out* as *output bridging rules*, and to bridging rules whose component *dr* has been set to *in* as *input bridging rules*. Output bridging rules specify the topics of the messages published within the local environment that can be forwarded to the broker of the other environment, and the remapping criteria with which the topic of a message is modified before the message is actually forwarded, whereas input bridging rules specify the topics of the messages published within the connected environment which the local broker subscribes to receive, and the criteria with which the topic of the received messages is remapped before they are forwarded to a rightful local subscriber. As a first step, the local broker of the environment where the bridging monitor is hosted sends a connection request to the bridging monitor, which in turn forwards the request to the local broker of the other environment, as it was the original sender of the request. If the request is accepted, a communication channel connecting the two local brokers is open, which is controlled by the bridging monitor. In what follows, we denote as *connecting broker* the local broker that sends the connection request, whereas the other broker is denoted as *target broker*. Through the established channel, the connecting broker sends a subscription request for any input bridging rule,[6] and then waits for control packets issued by local clients and by the bridging monitor.

The bridging monitor has been designed to forward any control packet received by the local brokers to the respective receiver, except packets encoding publishing requests which require additional operations.

Let us first consider publishing requests originating from the environment managed by the connecting broker. For any received publishing request issued by a local client that matches an output bridging rule, the connecting broker remaps the topics of the packet (see Section 2) and forwards the request to the bridging monitor. On the packet reception, the bridging monitor extracts from the payload: (1) the user preferences specified by the subject who has originally requested the publishing, and (2) the attributes that model the publishing request context.[7] Then, the bridging monitor selects from the extracted preferences those constraining message forwarding to the other environment, and evaluates them wrt the derived attributes. If no preference is satisfied, the message is blocked. Otherwise, if at least one preference is satisfied, the bridging monitor enforces the applicable access control policies. More precisely, it selects from the policy set of the environment managed by the connecting broker those policies regulating the forwarding of messages to external environments, and from the policy set of the environment managed by the target broker those policies regulating the receiving of messages from external environments. If the forwarding complies with at least one policy of both sets, the message is sent to the target broker, otherwise the forwarding is forbidden. In contrast, the target broker, upon receiving a publishing request from the bridging monitor, forwards, on the basis of the message topic, a copy of the received packet to communication channels that connect the broker to the rightful subscribers of the other environment, each controlled by the local monitor. For any message receiver candidate, the monitor checks whether the considered subscriber, on the basis of the preferences in the message payload and the applicable access control policies, is authorized to receive the message, and, in this case, it removes all metadata from the payload and forwards the packet.

Similarly, the target broker forwards to the bridging monitor any publishing request received from a local publisher that matches a previously subscribed topic filter.[8] The bridging monitor handles the publishing request with an approach symmetric to the one just described for the opposite message flow. The only difference is related to the local broker management of publishing requests from the bridging broker. Indeed, due to the remapping criteria specified by the input bridging rules, the topic of the message is remapped before the message is forwarded to the channels, controlled by the local monitor, which connect the broker to the rightful local subscribers.

5.2. The enforcement mechanism in details

We start to explain configuration aspects enabling broker to broker communication within the system architecture introduced in Section 4, which are instrumental to the proposed enforcement mechanism.

Let us hereafter refer to the bridging monitor as *bm*, and let us denote as *lb* the local broker that has been configured to operate as bridging broker specifying the bridging monitor *bm* as connection target, whereas we denote with *rb* the broker of the other environment.

On behalf of local clients, and on the basis of the configured bridging rules (cfr. Section 2), *lb* can publish messages to *rb*, and can subscribe the receiving of messages published within *rb*'s environment.

[6] Each subscription request specifies as topic filter the concatenation of the remote prefix (see Section 2.1) and the topic filter of the considered input bridging rule.

[7] Such data have been added to the payload by the local monitor.

[8] We remind that any subscription is achieved on behalf of the local broker on the basis of an input bridging rule.

The interaction of *lb* and *rb* starts with a connection request cp_{CN}, sent by *lb*. Upon receipt of cp_{CN}, the bridging monitor *bm* extracts the credentials of *lb* from the *CONNECT* control packet, and forwards the packet to *rb*. *rb* authenticates the subject and replies with a *CONNACK* control packet cp_{CA}, which is received by *bm* and then forwarded without any modification to *lb*. cp_{CA} specifies whether the connection request has been accepted or refused by *rb*, and, in the latter case, the cause. If cp_{CA} encodes the acceptance of the connection request, *lb* can start communicating with *rb*.

Once the connection has been established, for any input bridging rule *ibr* that has been specified for *lb*, *lb* sends a *SUBSCRIBE* control packet cp_{SB} to *rb*, specifying as topic filter the expression resulting from the concatenation of the components *rp* and *tp* of *ibr*. *bm* receives cp_{SB} and forwards it to *rb*, which keeps track of the topic filter *rp*+*tf*, and sends back a *SUBACK* packet notifying the receiving of the request. Once received by *bm*, the acknowledgment is forwarded to *lb*, which, on the basis of the specified bridging rules, is now ready to forward messages published by its local clients to *rb*, as well as to receive messages published in the *rb*'s environment.

We now focus in more details on the enforcement mechanism implemented by the bridging monitor, whose joint work with the local monitors allows regulating the communication of clients belonging to different environments.

Let us start to consider the forwarding of messages published within *lb*'s environment. The payload of any publishing request cp_{PB} received by *lb* includes: (1) the user preferences, if any, specified by the user, on behalf of whom the message has been published, and (2) the subject, object, and environment attributes which model the context within which the publishing request has been issued. If the topic *tp* referred to by cp_{PB} is matched by an output bridging rule *obr* of *lb*, *lb* remaps the topic, prepending to *tp* the remote prefix *rp* of *obr*. The resulting control packet, referred to as $cp_{PB'}$, is then forwarded to *bm*, which, by assumption, has been specified as connection target of *lb*. Upon the receipt of $cp_{PB'}$, *bm* extracts from the packet's payload the user preferences and the subject, object, and environment attributes. For any included user preference *up*, *bm* checks whether *up* constrains the forwarding of $cp_{PB'}$ to the environment managed by *rb*, and, in such a case, it evaluates the parametric predicate *exp* of component *bp* of *up* (see Section 3) with respect to the extracted subject, object, and environment attributes. If the predicate of at least one of the user preferences that constrain the forwarding is satisfied, *bm* checks whether there exists at least one access control policy among those specified in the environment of *lb* that authorizes the forwarding, and at least one in the policy set of the environment of *rb* that authorizes the import. If the check fails, the forwarding is blocked. The selection of the applicable policies in both environments is achieved by matching the topic filter of any policy *p* specified for *bm*, with the topic of $cp_{PB'}$. Policies are evaluated with respect to the attributes extracted from the payload. If the parametric predicate of at least one policy is satisfied, *bm* forwards $cp_{PB'}$ to *rb*, which in turn sends $cp_{PB'}$ to its local monitor. The local monitor then handles the publishing request as described in Section 2.2.2. The approach used by the bridging monitor to regulate the forwarding of messages published within the environment managed by *rb* is similar to the previously explained one. The only difference is related to the topic remapping task operated by *lb*. Indeed, upon receipt of a publishing request cp_{PB} from the bridging monitor *bm*, *lb*, on the basis of the specified input bridging rules, first redefines the topic of cp_{PB} removing the remote prefix, and then forwards the packet to the local monitor which will handle the request (see Section 2.2.2).

6. Performance analysis

In this section, we first shortly present core aspects related to the implementation of the bridging monitor, and than we evaluate the efficiency of the proposed enforcement mechanism with two experiment sets.

6.1. Implementation

We now shortly discuss implementation aspects of the bridging monitor.[9] A high level view of the bridging monitor architecture and of the related control flow is shown in Fig. 3. The bridging monitor *bm* has been designed to listen for connection requests from *lb*.[10] Upcoming connections requests are handled by a connection handler, which, on receipt of a new request from *lb*, opens a communication channel cc_{lb} with the local broker, and one channel cc_{rb} with *rb*, the local broker of the other environment. The handler also instantiates a monitoring task *mt*, which regulates the flow of control packets flowing through cc_{lb} and cc_{rb}. A pair of message queues are used to keep track of the messages flowing as input and output to the bridging monitor through these channels, respectively denoted in_{lb}, out_{lb}, in_{rb} and out_{rb}.[11] The monitoring task enqueues packets intended for *lb* and *rb* to out_{lb} and out_{rb}, whereas draws packets issued by *lb* and *rb* from in_{lb} and in_{rb}, respectively. A pipeline of packet handlers is used by *mt* for the marshaling of output packets, as well as for the unmarshaling of packets to be added to the input queues. Any control packet *cp* received as input is handled by *mt*, on the basis of the enforcement mechanism introduced in Section 5.2. Therefore, on receipt of a publishing request from *lb* and *rb*, the applicable policies are derived and checked. Such policies are stored within kvd_{lb} and kvd_{rb}, the key value datastores of the connected environments.

The proposed framework reuses the same criteria adopted in Colombo and Ferrari (2018) for the modeling of access control policies and user preferences within the datastores, as it has been shown that these favor a very efficient execution of the queries that derive all per request applicable policies and preferences. The interested reader can refer to Colombo and Ferrari (2018) for more details.

6.2. Experiments

Our experiments refer to an application scenario characterized by two MQTT based interconnected environments. One of the local brokers has been configured to operate as a bridge that considers the other broker as a connection target. Brokers are instances of Mosquitto v.1.4.10.[12] The choice of Mosquitto is motivated by : (1) the provided support for the bridging mechanism, (2) hardware requirements, which make this broker suited to low capacity devices (e.g., Raspberry Pi devices, which have been used for our experiments), and (3) the popularity of this MQTT broker. The instance of Mosquitto which operates as bridging broker has been configured to: (1) handle a single bridging connection; (2) use MQTT 3.1.1 as bridge protocol version; and (3) use a pair of bridging rules which allow forwarding/receiving messages on topic with a predefined prefix to/from the bridged environment, and which specify topic remapping for any message that enters or leaves the environment.

We assume that MQTT clients operating within the local environments are handled by 100 users, each managing 1 to 4 clients.

Clients are distributed between the two environments in such a way that any user who handles a client in one environment cannot handle a client in the other one. Clients distribution is carried out on the basis of three *configurations*. In the first configuration, the target environment only includes publisher clients, whereas the connecting one only subscribers. In the second configuration, the distribution criterion is inverted, whereas, in the third one, publishers and subscribers

[9] Local monitors are extended version of the enforcement monitor presented in Colombo and Ferrari (2018). Therefore, we do not report here their details, referring the interested reader to Colombo and Ferrari (2018).

[10] We remind that *lb* denotes the local broker that has been configured as a bridging broker specifying *bm* as connection target.

[11] *in* and *out* denote the verse of the flow from the monitor perspective, whereas *lb* and *rb* denote the broker associated with the queue.

[12] https://mosquitto.org/.

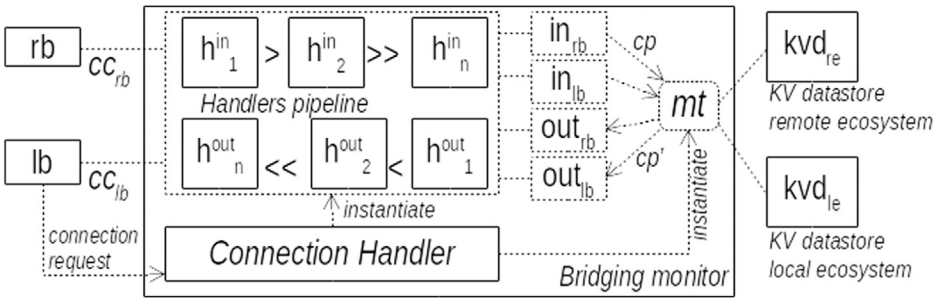

Fig. 3. Bridging monitor architecture.

Table 2
Testing scenarios.

Scenario	Connecting environment		Target environment	
	#publishers	#subscribers	#publishers	#subscribers
$S_{1,1}$	0	20	20	0
$S_{2,1}$	0	100	100	0
$S_{3,1}$	0	200	200	0
$S_{1,2}$	20	0	0	20
$S_{2,2}$	100	0	0	100
$S_{3,2}$	200	0	0	200
$S_{1,3}$	10	10	10	10
$S_{2,3}$	50	50	50	50
$S_{3,3}$	100	100	100	100

are equally distributed between the two environments.[13] For each configuration, we consider three *deployment options*, each involving a different number of clients. The first option refers to a client set of 20 publishers and 20 subscribers, the second to 100 publishers and 100 subscribers, whereas the third one involves 200 publishers and 200 subscribers. To make reference to the tested deployment settings easier, we refer to them as *scenarios*. Moreover, we use the notation $S_{i,j}$, where $i, j \in [1, 2, 3]$, to denote the scenario corresponding to the ith deployment option of the jth configuration. For instance, $S_{2,3}$ refers to a scenario where 100 publishers and 100 subscribers are equally distributed between the two environments. The considered scenarios are summarized in Table 2.

In the considered deployments, publisher clients have been configured to send 1 publishing request per second, whereas a single subscription request is issued by each subscriber client. Moreover, clients have been configured to issue publishing and subscription requests specifying all the same Quality of Service level (QoS).[14]

The routing activities of the bridging broker are regulated by the bridging monitor (cfr. Fig. 2). This monitor intercepts and possibly alters the flow of the messages exchanged by the brokers on the basis of the access control policies and user preferences which have been specified within the local environments. The specified access control policies, 80 per environment, have been defined in such a way that, 40 policies constrain the local brokers ability to forward messages to the other environment, whereas the remaining 40 constrain the local brokers ability to receive messages published in the other environment. User preferences have been defined in such a way that each user specifies at most one preference which constrains the corresponding environment's broker ability to forward messages to the other environment.

Our experiments aim at assessing the enforcement overhead, by measuring the *transmission time*, namely the time spent by a message published in one of the environments to reach a rightful subscriber, and the packets *throughput*, considered as the number of control packets which are handled per second.

Experiment 1. Our first set of experiments consider a scenario where neither of the two environments is equipped with an enforcement monitor, therefore local clients are directly connected to the respective brokers. The bridging monitor and the clients are hosted by desktop PCs equipped with i7 64-bit Quad Core CPU and 16 GB of RAM, whereas the local brokers by Raspberry Pi 3 Model B devices (equipped with a 64-bit Quad Core CPU and 1 GB of RAM).

In order to analyze the enforcement overhead, for each scenario, we compare the transmission time measured in a deployment devoid of the bridging monitor, with a deployment where the monitor is active. Two cases per scenario are considered, where all clients issue their publishing and subscription requests specifying QoS 0 and 2, respectively.

Fig. 4 shows the transmission time and the overhead measured for each considered case, as well as the average analysis time required by the bridging monitor to analyze the control packets issued by local brokers. The lower part of each bar shows the transmission time related to deployments lacking the bridging monitor, whereas the upper part shows the transmission time in deployments where the monitor is active, and the corresponding time overhead. The average analysis time (per control packet) is represented by horizontal lines overlying the transmission time bars.

Overall the time overhead related to cases with QoS 2 is always below 45 ms, whereas the one related to QoS 0 is below 37 ms, showing reasonably good performances of the bridging monitor. This behavior is due to the number of control packets that are exchanged per single publishing request, which, with QoS 2, is higher than with QoS 0. As a matter of fact, when QoS 2 is specified, for each publishing request, the local brokers also exchange the control packets cp_{PRC}, cp_{PRL}, and cp_{PC}. However, the analysis of these control packets requires less time than the analysis of publishing requests (i.e., cp_{PB}, namely the only ones involved in measuring cases specifying QoS 0), as these packets, once intercepted and recognized by the monitor, are directly forwarded to the respective broker. The low processing time of these packets lowers the average analysis time of each measuring case. For this reason, for each scenario, the transmission time related to QoS 2 measuring case is higher than QoS 0 case, whereas the trend is inverted for the measured average analysis time. This scheme can be observed with any analyzed scenario. The lowest transmission times have been measured in scenarios referring to the first deployment option (e.g., $S_{1,3}$). The times grow in scenarios that refer to the second deployment option (e.g., $S_{2,3}$) and even more in scenarios referring to the third option (e.g., $S_{3,3}$). This behavior is due to the number of clients involved, and the volume of messages that need to be handled by the bridging broker and the bridging monitor. In contrast, the comparison of scenarios referring to the same deployment option but different configurations show negligible time variations.

For each considered case, the red bar in Fig. 5 shows the throughput of the bridging monitor, whereas the blue bar the throughput of the

[13] We remind that the connecting environment is the one hosting the bridging broker, whereas the other one is the target environment.

[14] An MQTT client can require that specific message delivering guarantees are satisfied (see https://docs.oasis-open.org/mqtt/mqtt).

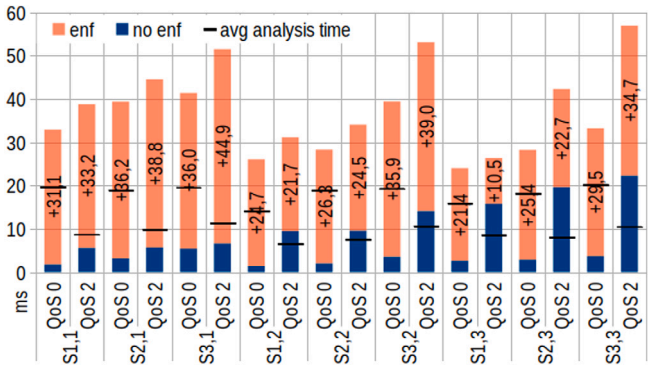

Fig. 4. First experiment: transmission time analysis.

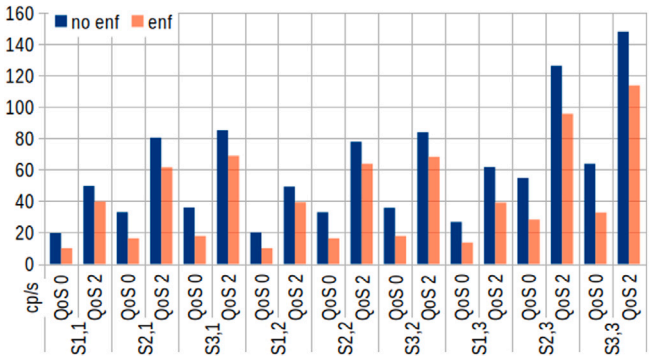

Fig. 5. First experiment: throughput analysis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

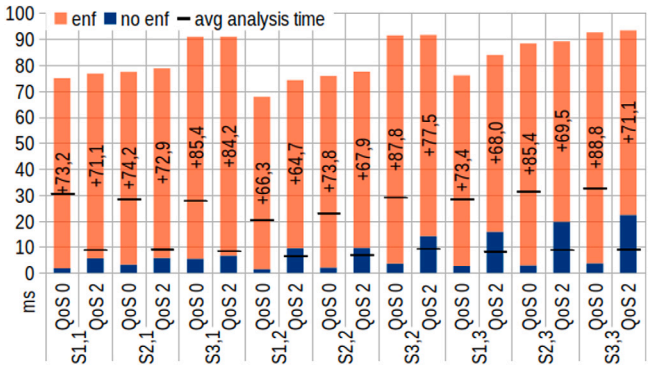

Fig. 6. Second experiment: transmission time analysis.

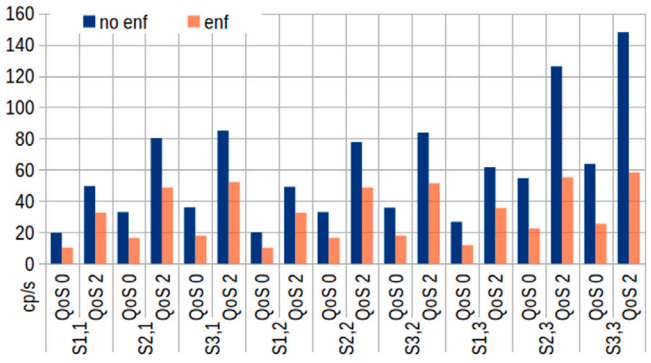

Fig. 7. Second experiment: throughput analysis. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

broker in deployments devoid of the monitor. The trends in Fig. 5 are similar to those observed for the transmission time analysis. Due to the analysis of policies and preferences that regulate the transit of any control packet, the rate of packets that is handled is lower than in scenarios with no enforcement mechanism. However, the observed rates combined with the previously considered transmission times (see Fig. 4) appear as a reasonably good result.

Experiment 2. In the second set of experiments each of the interconnected environments also includes an enforcement monitor that regulates the environment internal communication. Therefore, the considered system architecture corresponds to the one shown in Fig. 3 and discussed in Section 4. The bridging monitor and the clients are hosted by desktop PCs, equipped with i7 64 bit Quad Core CPU and 16 GB of RAM, whereas the MQTT brokers of the two environments are hosted by two Raspberry Pi 3 Model B devices (64 bit Quad Core and 1 GB RAM), which also host the local monitors.

In this second set of experiments, we aim at assessing the enforcement overhead introduced by the combined work of the local and bridging monitors.

The diagram in Fig. 6 shows, for each considered case, the transmission time related to a deployment devoid of enforcement monitors, which is matched against the transmission time measured in a deployment where all monitors are active.

Overall the transmission time is always below 94 ms, whereas the time overhead always below 89 ms. Therefore, the addition of the local monitors causes a reasonably contained growth of the transmission time and the time overhead wrt the first set of experiments. The same trends that have been observed in Fig. 4 are also visible in Fig. 6.

Finally, the diagram in Fig. 7 shows the measured throughput. For each case, the red bars show the throughput of the bridging monitor, whereas the blue bars show the throughput of the broker in deployments devoid of the monitor.

The trends visible in Fig. 7 are aligned with the ones observed with the first set of experiments (see Fig. 5). However, in this case, due to the filtering operated by the joint work of the local monitors, a lower ratio of control packets per second reaches the bridging monitor (cfr. Fig. 5).

Overall, the experiments show an enforcement overhead that is always reasonably contained, even in scenarios where data sharing across environments is regulated by three monitors.

7. Related work

The great majority of access control models which have been proposed in the literature allow regulating the communication of IoT devices operating in a single environment. For instance, the CapBAC model (Gusmeroli et al., 2013) relies on users tokens, referred to as capabilities, which specify users access privileges for specific objects. A user who aims at accessing an object needs to submit his/her access request along with his/her capabilities to a Policy Decision Point. Several extensions of RBAC have also been designed for IoT For instance, the trust-based extension proposed in Gwak et al. (2018) allows handling the privileges of a user on the basis of his/her role, and the trustworthiness of the group of users to whom the same role has been assigned. Fernandez et al. (2017) propose to complement the authorization process of RBAC with OAuth 2.0 based authentication. OAuth tokens are used to identify a subject who has issued an access request by means of an IoT device, along with his/her roles. Other extensions of RBAC aim at supporting context-aware policies, such as Ben Fadhel et al. (2018), which exploits model driven engineering technologies for policy specification and enforcement.

UCON based solutions for IoT have been less extensively studied than approaches based on CapBAC and RBAC (e.g., see Ravidas et al., 2019), and most of them target purpose specific application scenarios.

For instance, La Marra et al. (2017b) proposes a UCON framework designed for smart home applications, whereas in La Marra et al. (2017a), the same authors propose an approach to enforce UCON based access control within an MQTT environment.

Finally, several ABAC approaches have been recently defined. For instance, Gabillon et al. (Bruno et al., 2019; Gabillon et al., 2020) propose an ABAC model to regulate clients communication within a single MQTT environment. Although adopting different implementation strategies, the framework in Bruno et al. (2019) and Gabillon et al. (2020) is essentially aligned with the approach originally presented in Colombo and Ferrari (2018). Healthcare Plane (Ray et al., 2017) is an ABAC model specifically designed for the remote healthcare monitoring domain, which relies on the NIST NGAC Framework (Ferraiolo et al., 2016) for policy management.

However, none of the above-mentioned proposals aims at controlling data sharing among different environments, which is the focus of the current paper. We are only aware of a single attempt, by Fuentes Carranza and Fong (2019), to regulate data sharing among interconnected brokers. This approach is built on top of an event-based architecture (Fiege et al., 2002), which enables the communication of interconnected message brokers. Brokers interaction is regulated by scoping rules that constrain the range of the authorized receivers of any message to be routed. The approach relies on *brokering policies*, namely access control rules that constrain the brokers ability to propagate messages received through a channel to other channels. In Fuentes Carranza and Fong (2019), any communication channel that connects a device to a message broker, or a broker to another broker, specifies the security level required to issue a message through that channel. A message m received by a broker b through a channel c_1 can be forwarded through a channel c_2 to another broker or a device, if c_2 has at least the same security level as c_1. The proposed approach operates at coarse grained level, as the same constraints apply to all messages having the same source and destination channels. The approach has been implemented exploiting an *ad-hoc* modified version of Mosquitto.[12] Differently from Fuentes Carranza and Fong (2019), our framework:

(i) operates at a finer grained level. Indeed, in Fuentes Carranza and Fong (2019) the same access decision applies to any message issued through a communication channel, whereas in our approach, any access request issued by a subject can lead to a different decision;
(ii) supports context aware policies,
(iii) enforces user preferences,
(iv) can be used with heterogeneous MQTT brokers.

In recent years, research has also extensively studied access control solutions for cloud enabled IoT applications. Alshehri and Sandhu (2016) propose ACO, a reference architecture to integrate access control into cloud enabled IoT applications. In Bhatt et al. (2019), ACO has been used to support the enforcement of ABAC policies, whereas, Alshehri et al. (2018) shows how ACO can favor the integration of enforcement mechanisms for different access control models into AWS IoT.[15] Fernandez et al. (2019) propose a framework that allows users to specify privacy preferences regulating device level data collection, as well as access control policies that constrain data-sharing in the cloud. All these approaches rely on devices capable of managing connections with cloud services. In contrast, our framework does not rely on a cloud based infrastructure, and it is suited to devices with limited computational capacity.

8. Conclusions

In this paper, we have presented an ABAC framework to control data sharing among interconnected MQTT environments. The framework integrates an enforcement monitor specifically designed to be easily integrated into MQTT deployments. The experimental analysis has shown a reasonably low enforcement overhead in different scenarios.

Our work is progressing in several directions. To favor the adoption of our framework in large scale scenarios, we are investigating paralleling and load balancing techniques to automatically split up the enforcement mechanism among multiple monitors. We are also working at complementing the framework with tools for policy and user preference management. In addition, we are developing a monitoring tool to assess the effects of the specified policies and preferences.

CRediT authorship contribution statement

Pietro Colombo: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Elena Ferrari:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Engin Deniz Tümer:** Methodology, Software, Investigation, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has received funding from CONCORDIA, the Cybersecurity Competence Network supported by the European Union's Horizon 2020 Research and Innovation program under grant agreement No 830927, and from RAIS (Real-time analytics for the Internet of Sports), Marie Skłodowska-Curie Innovative Training Networks (ITN), under grant agreement No 813162.

References

Alshehri, A., Benson, J., Patwa, F., Sandhu, R., Access control model for virtual objects (shadows) communication for aws internet of things. In: ACM CODASPY 2018.
Alshehri, A., Sandhu, R., Access control models for cloud-enabled internet of things: A proposed architecture and research agenda. In: IEEE CIC 2016.
Ben Fadhel, A., Bianculli, D., Briand, L., Model-driven run-time enforcement of complex role-based access control policies. In: ACM ASE 2018.
Bhatt, S., Lo'ai, A.T., Chhetri, P., Bhatt, P., Authorizations in cloud-based internet of things: Current trends and use cases. In: IEEE FMEC 2019.
Bruno, E., Gallier, R., Gabillon, A., 2019. Enforcing access controls in IoT networks. In: FDSE 2019. In: LNCS, vol. 11814, Springer.
Colombo, P., Ferrari, E., 2015. Efficient enforcement of action-aware purpose-based access control within relational database management systems. IEEE TKDE 27 (8).
Colombo, P., Ferrari, E., Access control enforcement within mqtt-based internet of things ecosystems. In: ACM SACMAT 2018.
Fernandez, F., Alonso, A., Marco, L., Salvachua, J., A model to enable application-scoped access control as a service for iot using oauth 2.0. In: IEEE ICIN 2017.
Fernandez, M., Jaimunk, J., Thuraisingham, B., Privacy-preserving architecture for cloud-iot platforms. In: IEEE ICWS 2019.
Ferraiolo, D., Chandramouli, R., Hu, V., Kuhn, R., 2016. A comparison of Attribute Based Access Control (ABAC) standards for data service applications. In: NIST Special Publication 800.
Fiege, L., Mühl, G., Gärtner, F.C., 2002. Modular event-based systems. Knowl. Eng. Rev. 17 (4), Cambridge University Press.
Fuentes Carranza, J.C., Fong, P.W., 2019. Brokering policies and execution monitors for iot middleware. In: ACM SACMAT.
Gabillon, A., Gallier, R., Bruno, E., 2020. Access controls for IoT networks. SN Comput. Sci. 1 (1), Springer.
Gusmeroli, S., Piccione, S., Rotondi, D., 2013. A capability-based security approach to manage access control in the Internet of Things. Math. Comput. Modelling 58 (5), Elsevier.
Gwak, B., Cho, J., Lee, D., Son, H., Taras: trust-aware role-based access control system in public internet-of-things. In: IEEE TrustCom 2018.
Hu, V.C., Kuhn, D.R., Ferraiolo, D.F., Voas, J., 2015. Attribute-based access control. Computer 48 (2), 85–88.

[15] https://aws.amazon.com/iot.

Jin, X., Krishnan, R., Sandhu, R., 2012. Access control model covering DAC, MAC and RBAC. In: DBSec 2012. In: Springer LNCS, vol. 7371.

La Marra, A., Martinelli, F., Mori, P., Rizos, A., Saracino, A., 2017a. Improving MQTT by inclusion of usage control. In: SpaCCS 2017. In: LNCS, vol. 10656, Springer.

La Marra, A., Martinelli, F., Mori, P., Saracino, A., 2017b. Implementing usage control in internet of things: A smart home use case. In: IEEE Trustcom/BigDataSE/ICESS.

Lee, A.J., Biehl, J.T., Curry, C., Sensing or watching? Balancing utility and privacy in sensing systems via collection and enforcement mechanisms. In: ACM SACMAT 2018.

Ravidas, S., Lekidis, A., Paci, F., Zannone, N., 2019. Access control in Internet-of-Things: A survey. J. Netw. Comput. Appl. 144, Elsevier.

Ray, I., Alangot, B., Nair, S., Achuthan, K., Using attribute-based access control for remote healthcare monitoring. In: IEEE SDS 2017.

Wang, K., Yin, H., Quan, W., Min, G., 2018. Enabling collaborative edge computing for software defined vehicular networks. IEEE Netw. 32 (5).

Zhang, Q., Zhang, X., Zhang, Q., Shi, W., Zhong, H., Firework: Big Data sharing and processing in collaborative edge environment. In: IEEE HotWeb 2016.

Pietro Colombo is an assistant professor of Computer Science at the University of Insubria (Italy), where he works within the STRICT SociaLab of the Department of Theoretical and Applied Sciences. His most recent research activities are in the field of access control in modern data management systems, and Internet of Things applications. He has published more than 40 scientific papers in international journals and conference proceedings, and he is co-inventor of 2 US patents.

Elena Ferrari is a full professor of Computer Science at the University of Insubria, Italy, and scientific director of the K&SM Research Center. Her research activities are related to access control, privacy and trust. In 2009, she received the IEEE Computer Society's Technical Achievement Award for "outstanding and innovative contributions to secure data management". She received a Google Award in 2010, and an IBM Faculty Award in 2014. Since 2012 she has been an IEEE fellow, and in 2019 she has been named ACM Fellow.

Engin Deniz Tümer is a Ph.D. student in Computer Science and Computational Mathematics at the University of Insubria, Italy. He holds a Master degree (2018) in Computer Engineering at Ege University — Turkey. His main research interests are related to Machine Learning, Blockchain, Data Mining and Security. Before starting his Ph.D., he has been a Research Assistant in Computer Engineering at Celal Bayar University — Turkey.