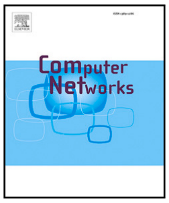
Computer Networks 223 (2023) 109576

Contents lists available at ScienceDirect

Computer Networks

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



Scalable and interoperable edge-based federated learning in IoT contexts

Claudia Campolo [a,b,*], Giacomo Genovese [a,b], Gurtaj Singh [a,b], Antonella Molinaro [a,b,c]

[a] University Mediterranea of Reggio Calabria, Italy
[b] Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Italy
[c] Laboratoire des Signaux et Systémes, CentraleSupélec, Université Paris-Saclay, France

ARTICLE INFO

Keywords:
Federated learning
MQTT
OMA LwM2M

ABSTRACT

The analysis of data coming from massively deployed Internet of Things (IoT) devices pave the way to a myriad of intelligent applications in several vertical domains. Federated Learning (FL) has been recently proposed as a prominent solution to train Machine Learning (ML) models directly on top of devices (FL clients) generating data, instead of moving them to centralized servers in charge of training procedures. FL provides inherent benefits, mainly in terms of privacy preservation and reduction of network congestion for datasets exchange. Despite the huge recent research efforts, it still faces challenges for a practical implementation effectively targeting low communication footprint, robustness and interoperability. To fill this gap, in this work we propose a novel comprehensive framework built upon the Message Queue Telemetry Transport (MQTT) publish/subscribe messaging protocol and the Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) semantics to facilitate FL operations and make them more suited to handle IoT devices acting as FL clients. The viability of the proposal as well as its communication efficiency compared to a literature solution are evaluated through a realistic Proof-of-Concept (PoC) under different link settings and for different datasets.

1. Introduction

In the recent years Internet of Things (IoT) devices proliferated at a dramatic pace. According to Ericsson [1], there will be 24 billion interconnected devices by 2050. Through the application of Machine Learning (ML) techniques, the big amount of data generated by such devices can unlock intelligence into different domains, spanning from transportation to industries, from healthcare to agriculture.

Centralized ML functions typically placed in a cloud server may incur critical scalability limitations given the IoT data explosion. Moreover they suffer from users' privacy leakage because of the need to transfer the end-devices data to a third party server for training purposes. Indeed, data to be collected exist in the form of isolated islands and may contain confidential information, e.g., in the case of healthcare applications, and hence, data sharing may not be possible [2]. Such threat may be posed even within the same enterprise in the case of Industrial IoT (IIoT) applications [3].

The concept of Federated Learning (FL) has been recently proposed to address the aforementioned issues [4,5]. Unlike traditional ML techniques, FL enables on-device training of an ML model, by coordinating multiple devices, acting as *FL clients*, with a central server, referred to as *FL aggregator*, without sharing actual datasets.

The population of FL clients involved in the training procedures may encompass heterogeneous devices, with different processing capabilities and experiencing different channel conditions when exchanging model/parameters with the FL aggregator. Randomly selecting the FL clients may result in poor performance in terms of training convergence time and/or model accuracy [6]. Devices either constrained in terms of computing/battery resources and/or slow in exchanging data with the FL aggregator may slow down the overall training. Instead, devices owning poor datasets may compromise the accuracy of the model. Nonetheless the huge recent literature proposing smart client selection schemes, e.g., [6–11], a few works have focused on the procedures needed to discover the FL clients capabilities and to enable judiciously selecting them. Retrieving information about the status of the potential FL clients is even more relevant when IoT devices, such as smartphones, vehicles, laptops or tablets, are involved [12,13]. An IoT client may not be always able to perform on-device computation to reach the target convergence within an expected time. Moreover, more exacerbated challenges need to be faced in terms of communication, computation, storage, power, and energy utilization, e.g., straggler issue [13].

* Corresponding author at: University Mediterranea of Reggio Calabria, Italy.
E-mail addresses: claudia.campolo@unirc.it (C. Campolo), giacomo.genovese@unirc.it (G. Genovese), gurtaj.singh@unirc.it (G. Singh), antonella.molinaro@unirc.it (A. Molinaro).

https://doi.org/10.1016/j.comnet.2023.109576
Received 31 October 2022; Received in revised form 6 January 2023; Accepted 11 January 2023
Available online 14 January 2023

In [7] potential clients notify about their resources (e.g., processing capabilities, dataset, etc.) the FL aggregator. The latter one leverages the retrieved information to estimate the time needed to build the global model, starting from updates of locally trained models, and to update it. Then, clients are selected according to the performed estimates. Similarly in [11], the FL aggregator transmits a participant request message to all the potential clients. Among them, those wishing to participate in the training send a response message including their data distribution and the quantity of private data. However, the aforementioned works do not discuss the protocol leveraged to exchange such information.

In [14] we proposed to leverage: *(i)* the Message Queue Telemetry Transport (MQTT) protocol [15] for FL clients-FL aggregator interactions aimed to retrieve the capabilities of the potential clients and *(ii)* the Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) data models [16] for the semantic description of such capabilities. The solution has been early implemented confirming its viability.

Besides interactions for the sake of discovering clients capabilities, which are neglected in the majority of works targeting client selection, FL requires extensive communication between the selected FL clients and the FL aggregator. As early argued in [17], the impact of the communication protocol leveraged for such purpose on the FL performance (e.g., in terms of the overall time taken to train the model) and on the resulting communication footprint, has not been adequately investigated.

In [18] the FL aggregator acts as a MQTT subscriber and the devices, which want to communicate their state (e.g., being ready for training a model), become the MQTT publishers. MQTT is also exploited in [19] to orchestrate the model parameters exchange, given its supremacy in terms of bandwidth efficiency and lower latency compared to sockets [17] and its suitability for one-to-many interactions, as needed during the distribution of the global model.

This work treasures the aforementioned solutions and goes beyond them, by providing the following main original contributions:

- We extend our preliminary work in [14], which focused on the client discovery procedures only, with a more comprehensive lightweight edge-based FL framework covering all stages of the FL training procedure in the presence of IoT devices.
- We design communication primitives based on MQTT, with relevant pre-defined topics and messages payloads, augmented with the OMA LwM2M semantics, to target interoperability and specifically cope with IoT challenges.
- We assess the performance of the proposal through a Proof-of-Concept (PoC) with virtualized and physical FL clients, when compared to a benchmark FL implementation, under different link settings and for different datasets.

The remainder of this work is organized as follows. Section 2 provides background and motivations for the work. The proposal is discussed in Section 3. The main procedures foreseen by the conceived framework are presented in Section 4, Section 5, and Section 6. Section 7 discusses the main benefits of the conceived framework. Achieved results are reported in Section 8, before concluding in Section 9 with hints on future work.

2. Background and motivations

In this section, we briefly introduce how federated learning works in general and the relevant open issues which motivated our investigation.

2.1. Federated learning: an overview

The FL paradigm has been proposed in 2016 [4] to boost Artificial Intelligence (AI) solutions thanks to the distributed and privacy-enhancing nature.

In FL, a shared global model is trained by iteratively aggregating model updates from multiple devices over multiple rounds. In each round, the FL clients download the latest global model from the FL aggregator, train the model on their local datasets, and report their respective model updates to the FL aggregator. The latter one combines all local model updates and builds a new improved global model. Model aggregation can be performed either at the edge or at the cloud [12]. The process is iterated until the global training is complete.

By leveraging the computing resources of distributed learners, the FL aggregator can enhance the training quality while minimizing user privacy leakage, since the raw data are not required for the training at the FL aggregator. As a further consequence, the latencies due to data offloading are reduced and network resources for data exchange saved.

In addition, by exploiting different datasets from different devices, FL can achieve accuracy performance comparable to those provided by a centralized approach.

2.2. FL limitations

The aforementioned advantages make FL a suitable candidate to build a myriad of intelligent and privacy-enhanced IoT applications, in domains ranging from smart healthcare to smart transportation [20].

Notwithstanding, FL has to face several challenges, among which: device heterogeneity (e.g., in terms of computational resources, available power), wireless channel uncertainties, and data disparity, due to unbalanced local datasets and non-independent and identical distribution (non-IID) data among devices [12].

The random selection of the FL clients participating in local training procedures, as in the original FL implementation [4], might provide long convergence time. This can be detrimental for synchronous federated learning where the FL aggregator waits for the updated local weight parameters uploaded by selected FL clients at each round. For instance, selecting the nodes either with the low computational resources or experiencing poor wireless channel conditions results in an increase in the overall FL training time. IoT devices may fail in completing the training task if they either run out of battery or switch to sleep mode, and accordingly affect the model accuracy.

Several more sophisticated client selection schemes have been devised in the literature, e.g., [6–11] just to name a few. They differ for the criterion/set of criteria considered to select FL clients. The effectiveness of such schemes heavily rely on the awareness about the resources and capabilities of potential clients that the FL aggregator can build. However, the aforementioned works typically neglect to discuss how such information can be retrieved. Instead, a protocol is needed to enable potential FL clients and the FL aggregator to exchange such data with a low communication overhead and in a uniform manner to cope with the inherent heterogeneity of involved devices.

The communication overhead incurred during the client discovery procedure represents only a small amount of the overall data exchange necessary for the distributed model training. Indeed, after the client selection, the FL aggregator and FL clients need to exchange initially the original model and then, the updated local and global model parameters over multiple rounds [4].

The choice of the communication protocol leveraged during these stages may affect the FL performance and, in turn, impact the network [17], especially for large model sizes, hence deserving a proper investigation which motivates our study.

3. The proposed FL-MQTT framework: basics

In this work, we propose a comprehensive lightweight framework, we refer to as *FL-MQTT*, to support interactions between an FL aggregator, implemented at the edge, and distributed IoT devices acting as FL clients.

The framework builds upon our preliminary work in [14], which specifically targeted the support of *FL client discovery* for the sake of implementing smart client selection procedures.

There, MQTT is leveraged to enable the discovery of FL client capabilities. Moreover, the OMA LwM2M semantics is exploited for the description of the capabilities of the devices wishing to act as FL clients.

In this work we leverage both MQTT and OMA LwM2M, thus adding to the scalability of MQTT the interoperability offered by the semantic-rich OMA LwM2M data model. Moreover, we significantly extend the previous proposal to support data exchange among FL clients and FL aggregator *throughout all the FL stages*, including the model exchange steps until the model convergence is reached.

3.1. Key building blocks

In the following, the basics of MQTT and OMA LwM2M are shortly reported.

3.1.1. MQTT

MQTT is one of the lightweight messaging protocols for the IoT that follows the publish/subscribe paradigm [15]. In MQTT a client can act both as a *publisher* and as a *subscriber*. The latter subscribes to a "topic" and receives notifications via an entity, i.e., the broker, playing the role of the MQTT server, whenever a new message is generated on that topic by a publisher. MQTT topics are structured in a hierarchy, similar to folders and files in a file system, using the forward slash, /, as a delimiter. Each level must consist of at least one character to be valid. While message publication can be done one topic at a time, the protocol allows client subscription to multiple topics. Specifically, it is possible to subscribe to multiple topics using one message containing one or both the two *wildcard* characters: # (hash character) multi level wildcard; + (plus character) single level wildcard. The exchanged message payload can contain string or file; no specific format is defined by the standard and is typically application-specific.

In the IoT domain, MQTT is commonly preferred over the web-oriented Representative State Transfer (REST) architecture, widely used in the Internet domain, which enables data transfer in the client–server model over the Hyper-text Transfer Protocol (HTTP) [21]. The overhead associated to request–response protocols as well as the larger headers of REST affect battery usage more than MQTT [22].

MQTT has been developed for contexts where clients can suffer from intermittent connectivity and data loss. Specific features have been included in the protocol to overcome the aforementioned challenges. First, three settable Quality of Service (QoS) options, from the least guaranteed (0) to the most guaranteed (2) are available. In fact, they use different acknowledgment (ack) messages between client and broker: QoS 0 uses a fire and forget model, with no ack; QoS 1 uses one PUBACK message to ensure that the message has been received at least once; QoS 3 foresees PUBREC, PUBREL, and PUBCOMP messages to ensure that a published message has been received exactly once.

When a client connects to a broker, it can use either a non persistent connection (clean session) or a persistent connection. The *cleansession* option rules the settings when the session is restored between client and server. If the option is set to true, the broker does not store any subscription information or undelivered message for the client. If it is set to *false*, it makes the broker keep topics subscriptions and retain messages that it cannot deliver until the client reconnects. The *cleansession false* option copes with intermittently connected clients by allowing them to read the messages transmitted on subscribed topics when they reconnect. Analogously, the message *retain* flag can be set by the publisher to instruct the broker to store the message on that topic in order to send it to all future subscribers. Using the *retain* flag, always and only the last message published on that topic is stored.

3.1.2. OMA LwM2M

The OMA LwM2M standard defines a lightweight client–server protocol where each device is described by semantic models, using an eXtensible Markup Language (XML) model file, that can be customized [23,24]. Basically, an Object is used to describe and control a specific software/hardware component of the device (such as sensors, antennas or device firmware) with associated resources (e.g., value, unit, maximum value, minimum value). Depending on the characteristics of the device, there could be one or more instances of the same object on the device, for example, multiple temperature sensors.

The standard requires each resource to be uniquely defined through a Uniform Resource Identifier (URI) path composed of the identifiers concatenated as follows *objectID/instanceID/resourceID*. For instance, the temperature sensor object is defined with the identifier 3303, while the resource relating to the temperature value is identified with 5700. Then, following the standard, the resource of the first temperature sensor of a given device is identified through the URI 3303/0/5700, that of the second sensor through the URI 3303/1/5700, thus creating two distinct URI paths for different *instanceIDs* of the same object.

The alliance provides a public registry of standard objects and resources. Every developer can contribute to the creation of new standard objects or use customized objects in its own environment.

OMA LwM2M was designed to provide a compact and efficient data model to reduce data transferred from/to the IoT device. More specifically, the information is labeled and encoded through the use of identifiers (e.g., 3303 for the temperature object). Such identifier is then translated and the information enriched thanks to all the information stored into the models.

3.2. Main assumptions and design choices

As illustrated in Fig. 1, we consider an edge-based FL approach, where the global model aggregation is performed at an edge server deployed in agreement with the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC) specifications [25], i.e., the ME host. The latter one hosts, as a virtualized application (ME app), an FL aggregator which manages the FL procedures.

A population of N IoT devices willing to act as FL clients connect to the FL aggregator through an edge network domain. In particular, we assume that they are connected to the Base Station (BS)/Access Point (AP) the edge server is attached to. Among the N devices, only a subset of $\tilde{N} \leq N$ devices is eligible to act as FL clients and up to $M \leq \tilde{N}$ devices can be selected by the FL aggregator to act as FL clients for the model training procedure.

Without loss of generality, we assume that synchronous FL is implemented, i.e., the server performs synchronized aggregation of the received model updates from all the selected FL clients at each round.

Both the N devices and the FL aggregator run an MQTT client, whereas the MQTT broker is deployed in the same edge server as the FL aggregator. This choice is intended to reduce the communication footprint for the forwarding of publish and subscribe messages, as already shown in our previous work in [14]. However, the proposal may work also in the presence of a remote broker.

In general, throughout the proposed FL-MQTT framework, the following main design choices are considered:

- all MQTT topics are conceived to be self-explanatory at every step of the FL model training, including the phase of FL client discovery and notification; specific prefixes and components are used that characterize the different phases;
- all exchanged MQTT publish messages carry a JavaScript Object Notation (JSON) formatted payload and the conveyed data follow the OMA LwM2M semantic standard, so that they can be understood by any FL clients (and FL aggregator), by improving interoperability;

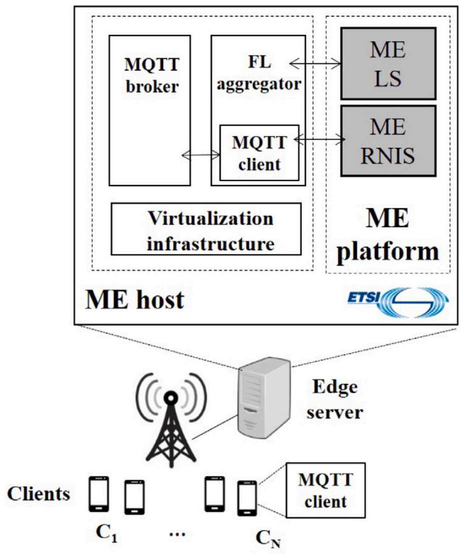


Fig. 1. The conceived FL-MQTT framework within the ETSI MEC architecture.

- when needed to support the phases of client discovery and model training, some new OMA LwM2M objects and related resources are defined by choosing their IDs among IDs dedicated to private companies or individuals, in the range 10241–26240, according to the OMA guidelines [24].

In the following Sections, the main steps of the overall FL training are discussed by describing the conceived workflows.

4. The FL client discovery

IoT devices wishing to contribute to an FL task of a given type (e.g., human activity recognition, voice recognition, image recognition) subscribe to updates from an FL aggregator looking for FL clients on the MQTT topic shortened as *topic#1,disc* and reported in full in Table 1. There, the definition of the two topics leveraged in the discovery phase is summarized along with the payloads of the relevant MQTT publish messages. Table 1 as well as the following Tables allow to figure out the semantic-rich but compact nature of the conceived topics and to give insights about what is actually exchanged between the FL clients and the FL aggregator and how interoperability is concretely achieved.

In particular, in *topic#1,disc*, the prefix *disc* indicates that such channel is dedicated to the client discovery stage; the *fl* component indicates that it refers to an FL task; and the last level of the topic tree specifies the type of FL task to be started, e.g., voice recognition.

When an FL training procedure needs to be started, the FL aggregator correspondingly publishes an MQTT message, whose payload carries a newly-defined object, named *DiscoveryFL* (with Object ID 18333).

More in detail, the resources of the *DiscoveryFL* object that is published on *topic#1,disc* by the FL aggregator are shown in Table 2. The object contains information about: the identifier of the involved FL entity (FL Entity ID), which is the FL aggregator ID in this case; the Task Type ID (e.g., voice recognition), and the OMA URI path, which carries the specific path to information the FL aggregator wants to know for the FL client selection (e.g., battery, CPU and memory status).

Such information can be declared using a new OMA LwM2M object, named *ClientFL* (Object ID 18332) that embeds and groups them, as illustrated in Table 3.

Then, the FL aggregator subscribes to updates from candidate FL clients.

The MQTT broker forwards the publication (from the FL aggregator) on *topic#1,disc* to the devices which subscribed to the FL task information retrieval. Such devices evaluate whether they can candidate as FL clients according to the information published by the FL aggregator in the *DiscoveryFL* object.

Table 1
Description of topics and payloads leveraged during the client discovery procedure.

#	Topic name	Publish payload
1,disc	*disc/fl/tasktype*	{“bn”:“/18833/0/”, e”:[{“n”:“26241” “v”:“AB123”} {“n”:“26249”, “v”:“voicerecog”}, {“n”:“26250”, “v”:“/18832”}]}
2,disc	*info/fl/tasktype/serverid/taskid*	{“bn":"/18832/0/", e":[{“n":“26241", “v":“EF789"}, {“n":“26242",“v":“50"}, {“n":“26243",“v":“500"}, {“n":“26244",“v":“240"}, {“n":“26245,“v":“5000"}, {“n":“26246",“v":“60000"}, {“n":“26247",“v":“150"}, {“n":“26248",“v":“6500"}]}

Table 2
OMA LwM2M DiscoveryFL object’s resources.

URI	Resource
/18833/0/26241	FL Entity ID
/18833/0/26249	Task Type ID
/18833/0/26250	OMA LwM2M URI Path

Table 3
OMA LwM2M ClientFL object’s resources.

URI	Resource	Unit of measure
/18832/0/26241	FL Entity ID	–
/18832/0/26242	Battery Level	[%]
/18832/0/26243	Battery Capacity	mAh
/18832/0/26244	CPU	MHz
/18832/0/26245	Free Memory	kB
/18832/0/26246	Dataset Size	kB
/18832/0/26247	Dataset Entries	–
/18832/0/26248	Dataset Age	s

Among the N devices, a subset of $\tilde{N}$ nodes, pass the self-eligibility check and publish the information the FL aggregator is interested to, by using a dedicated topic: *topic#2,disc*, detailed in Table 1. In such a case: the prefix *info* in the topic indicates that the message contains an information; the components *fl/tasktype/* refer to the requested task; *serverid/taskid* need to itemize the auctioneer server and its specific task identifier. The latter component *taskid* is needed since there could be multiple instances of the same task type on the same aggregator.

This MQTT publish message will carry in the payload the client identifier and its capabilities described through the OMA LwM2M semantics using the *ClientFL* object, Table 3. At this stage, without loss of generality, we assume that information about the devices’s battery, processing and memory capabilities, as well as the main characteristics of the owned dataset, is included in the *ClientFL* object.

The overall workflow for the MQTT/OMA LwM2M-supported FL client discovery phase is reported in Fig. 2.

5. The FL client selection and notification

The FL aggregator exploits the retrieved information to perform the client selection. In particular, to this purpose, the FL aggregator leverages the retrieved device-specific capabilities as well as some network-related information, which in our design is provided by the ME platform. Indeed, the *Radio Network Information Service* (RNIS) and the *Location Service* (LS) components of the ME platform natively

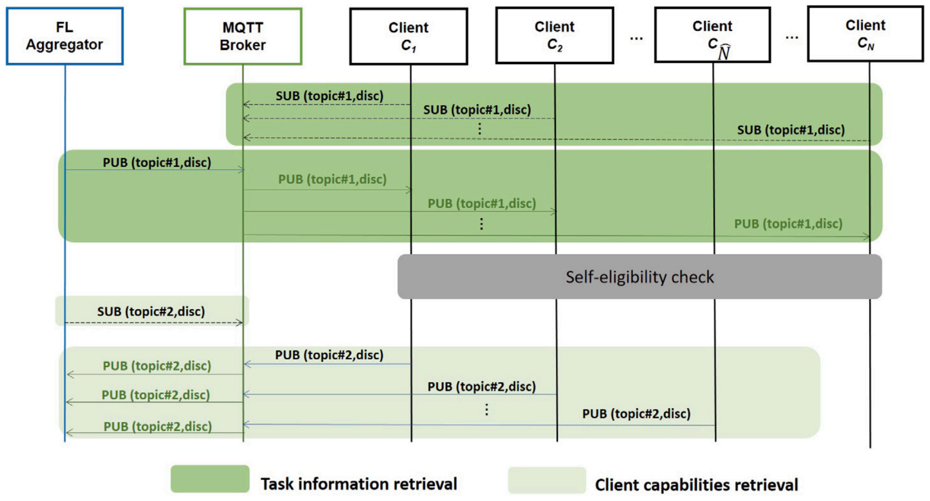


Fig. 2. FL client discovery workflow.

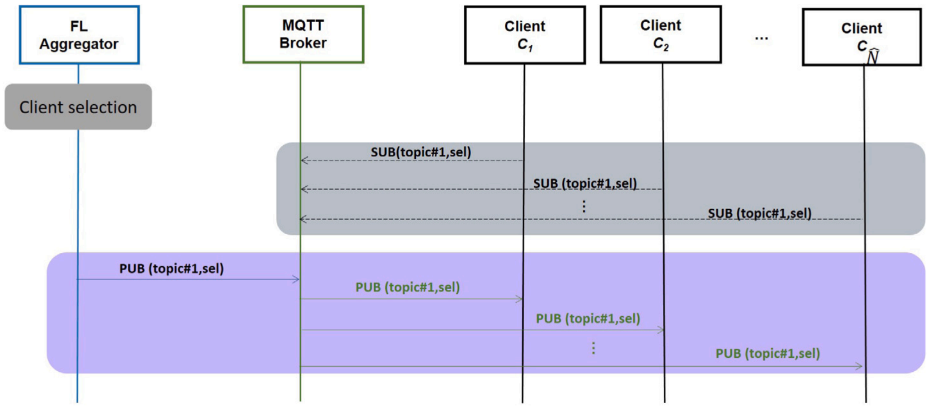


Fig. 3. Workflow of FL aggregator-FL clients messages exchange during the client notification procedure.

provide radio-related and augmented positioning information, respectively. Being the FL aggregator deployed at the ME host, it can directly retrieve the aforementioned information through interaction with such modules. The Link Quality can be exploited to estimate the time for exchanging model parameters/updates for a given IoT device, to be selected as FL client. The Position allows the FL aggregator inferring whether an IoT device, if selected as FL client, can be available for the overall training task or be disconnected before completing it.

Regardless of the specific FL client selection policy, out of the scope of the paper, the chosen M FL clients need to be notified about their participation in the training procedure.

To this aim, the FL aggregator publishes a message containing the list of selected clients (see Table 4), and this message is forwarded by the broker to all the $\tilde{N}$ devices which candidated themselves in the discovery stage as potential clients by publishing on the *topic#2,disc* and then by subscribing to the *topic#1,sel*. The relevant workflow is shown in Fig. 3.

6. The FL training procedure

Once the FL clients are selected and notified, the distributed training can start and be iterated over multiple rounds. The workflow sketched in Fig. 4 shows in detail all the messages exchanged during this stage.

The M selected FL clients perform the subscription to the global model, through the usage of a particular topic shortened as *topic#1,trai* and reported in full in Table 5, where the definition of all topics used during the training phase is summarized along with the payloads of the relevant publish messages.

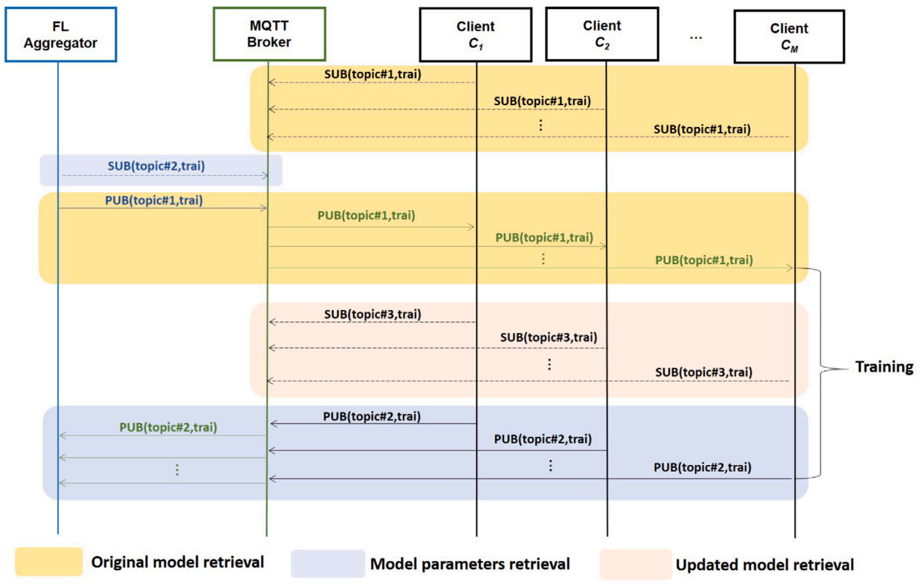
The FL aggregator in turn subscribes to *topic#2,trai* through which it declares its interest in receiving model updates from FL clients after each training round. In addition, it publishes a message whose payload contains either the original model to be trained by FL clients[1] or the Uniform Resource Locator (URL) of the repository where the model is stored and can be retrieved by the FL clients.

Another specific OMA LwM2M object, named Neural Network (NN) model (*NNModel*), is newly defined (Object ID 18334) to describe the model parameters to be exchanged between the FL aggregator and the selected FL clients. This object, reported in Table 6, contains information about: the round identifier (*Round ID*), the client identifier (*FL Entity ID*), the model (*Model Info*), the time when the training phase has started (*Tstart*) and the time elapsed since the start of the training (*Trel*), useful for FL clients to keep synchronization.

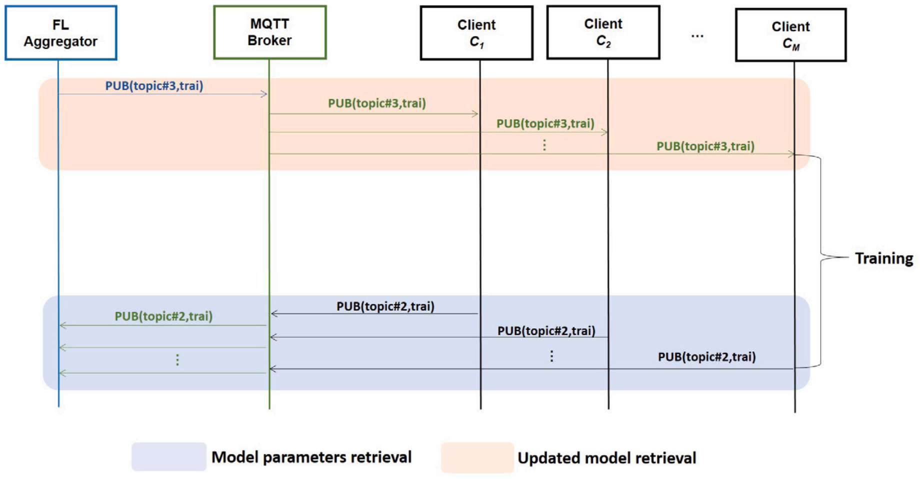
In particular, we used the HDF5 format, which is the current version of the Hierarchical Data Format (HDF) [28], to save the information about the architecture of the NN model and its weights [29].

The same OMA LwM2M object is used to describe the original model shared by the FL aggregator as well as the subsequent updates of the weights by the FL clients. They are distinguished by the Round ID

[1] Up to 256 MB bytes can be transferred in the MQTT payload [26], hence enough to carry the original model considering state-of-the-art models' size that is in order of hundreds of MB [27].



(a) Training: first round with the exchange of the original model.



(b) Training: rounds after the exchange of the original model.

Fig. 4. Workflow of the FL training procedure.

Table 4
Description of the topic and payload leveraged for the client notification procedure.

#	Topic name	Publish payload
1,sel	*modl/fl/tasktype/selection*	[{“n”:“clnts”, “v”:“$clientID_1, \dots clientID_M$”}]

resource, which is zero for the original model and greater than zero for the model updates.

Other parameters related to the training procedure that the FL aggregator could send to FL clients round by round could be the learning rate or the size of the mini-batch [30]. These parameters can be added to the object defined in Table 6 simply by defining new OMA LwM2M resources.

After locally performing the training on their own dataset, the FL clients publish on *topic#2,trai* the model parameters. The same object as in Table 6 is used to this purpose. In this case, the object also carries the *FL Entity ID* resource to track the FL client from which the model update has been received. The resource is not necessary in the publish message of *topic#1,trai*. The American Standard Code for Information Interchange (ASCII) file published as the *Model Info* resource instead contains only the information about the weights of the NN to be sent to the FL aggregator.

Then, FL clients subscribe to *topic#3,trai* to receive the updated model from the FL aggregator in subsequent rounds.

Topic#1,trai is used only in the first round to let the FL aggregator send the parameters of the model to be trained. Then, until the end of the training procedure, the FL aggregator will use *topic#3,trai* to exchange the updated parameters of the model under training.

7. Expected benefits

The conceived framework exhibits the following main strengths. Some of them are inherited from the vanilla design of the leveraged building blocks, others come from the devised individual customizations and the foreseen interplay among them.

Scalability. FL offers several advantages, among which scalability [12,13], when compared to centralized learning approaches. Scalability refers to the ability to incorporate more devices in the federated learning process acting as FL clients and distributing the training efforts. However, this feature cannot be given for granted in FL and needs

Table 5
Description of topics and payloads leveraged during the training procedure.

#	Topic name	Publish payloads
1,trai	*modl/fl/tasktype/serverid/taskid*	{{“bn":“/18834/0/",} e": [{“n":“26251",“v":“0"} {“n":“26252", “v":“fileAsciiFormat"} {“n":“26253", “v":“9/03/22-12:02:23"} {“n":“26254", “v":“200"}]}
2,trai	*modl/fl/tasktype/serverid/taskid/trained*	{“bn":“/18834/0/",} e": [{“n":“26241", “v":“AB123"}, {“n":“26251",“v":“0"} {“n":“26252", “v":“fileAsciiFormat"} {“n":“26253", “v":“9/03/22-12:02:23"} {“n":“26254", “v":“400"}]}
3,trai	*modl/fl/tasktype/serverid/taskid/update*	{ “bn":“/18834/0/", e": [{“n":“26251",“v":“1"} {“n":“26252", “v":“fileAsciiFormat"} {“n":“26253", “v":“9/03/22-12:02:23"} {“n":“26254", “v":“500"}]}

Table 6
OMA LwM2M NNModel object’s resources.

URI	Resource
/18834/0/26251	Round ID
/18834/0/26241	FL Entity ID
/18834/0/26252	Model Info
/18834/0/26253	Tstart
/18834/0/26254	Trel

to be analyzed under different perspectives, both for what concerns the overall training performance and the communication burden. In the presence of heterogeneous and potentially resource-constrained devices acting as FL clients and for synchronous FL operations, as targeted in this work, scalability may be hindered [31]. However, our proposal allows to enforce a judicious FL client selection, which may ensure scalability [12,31]. Further techniques to improve scalability are related to the optimization of networking resources [12], especially when considering hundreds or thousands of devices. By leveraging a communication protocol, like MQTT, specifically designed for resource-constrained environments and thought to work in large-scale IoT scenarios involving smart cities, power distribution grids or fleets of connected cars which may include millions of IoT devices, our approach pursues the scalability target. Decoupling FL clients and the FL aggregator allows greater scalability in the exchange of information. The FL aggregator does not need to manage the connections with the involved FL clients because this is delegated to the MQTT broker and it can focus on model aggregation procedures. This choice appears especially convenient as the number of FL clients increases. However, moving the burden to the broker may entail the need to properly deploy it in order to be equipped with the appropriate resources to cope with a high number of connected devices and message workloads. Proper vertical/horizontal scaling procedures of MQTT broker instances may be enforced by the orchestrator managing the edge platform, which is however a topic outside the scope of this work.

Flexibility. Any client selection policy can be implemented in the conceived framework provided that the needed parameters are conveyed in the aforementioned messages. Different types and number of OMA LwM2M resources can be leveraged to describe the FL clients capabilities, while easily extending the OMA LwM2M *ClientFL* object’s model.

Robustness against error-prone and intermittent connectivity. MQTT provides reliable communications by exploiting Transmission Control Protocol (TCP) at the transport layer. The reliability of messages in MQTT can be further augmented through earlier mentioned QoS levels, *cleansession*, and *retain* flags. Such features appear extremely relevant to deal with IoT FL clients connected through error-prone wireless links by providing messages buffering (*cleansession false, retain*) and client’s settings maintenance like topic subscription (*cleansession false*). Such mechanisms are crucial in scenarios where FL clients could disconnect frequently either to due to harsh propagation conditions or to sleep mode operations, common in IoT contexts.

Low communication footprint. In comparison with other reliable protocols, such as HTTP, MQTT has a lower communication footprint, thanks to its lighter header [32]. Its specific features make it a valuable messaging protocol for IoT applications [33]. Also the topic structure has been conceived to be lightweight: OMA LwM2M ensures compact and efficient data model to reduce data transferred from/to the potentially constrained FL clients. Additionally, having the FL aggregator deployed as an ME app prevents to burden the device wishing to act as FL client and the radio interface through which it is connected for the retrieval of network-related information already available at the Radio Access Network (RAN) premises.

Low protocol overhead. By leveraging MQTT, which offloads to the broker part of the work necessary for the exchange of information, the proposal is suited for devices with constrained resources. Each FL client has the burden of keeping only one connection active, the one with the broker, regardless of the information it wants to receive.

Support for heterogeneous devices. The conceived framework is inherently able to support the description of potential clients exhibiting heterogeneous capabilities in terms of processing and battery lifetime as well as experiencing different network conditions over different connectivity interfaces.

Interoperability. Such challenging goal is achieved through expressive MQTT topics, able to distinguish the different stages of the FL procedures and to identify different kinds of FL tasks (e.g., object detection, speech recognition, classification), as well as through the OMA LwM2M objects carried out in the payload of publish messages to describe capabilities of heterogeneous clients in a *uniform semantic-rich* manner. The decoupling of the FL clients from the FL aggregator also paves the way for interoperability with third interested parties in learning outcomes. Any new actor, in compliance with any security mechanisms implemented, can take over the information exchange without the need for the parties interested in FL actions to make any changes to the communication/mechanisms in place. For instance, a new FL aggregator interested in obtaining an already (partially) trained model could retrieve it by subscribing to the update topic that the FL aggregator sends to the FL clients participating in the training (i.e., *topic#3,trai*). In such a way, if the message has been published with the *retain* flag, it will receive the last message published on the update topic. The type of functionality just described, can be used to implement the concept of Federated Transfer Learning (FTL) [2,3]. This paradigm, which represents an evolution of FL, involves building effective models for a target domain by leveraging the knowledge built about other domains. Considering two domains A and B, where there is an overlap in the space of the characteristics related to the datasets and/or a similarity between the FL clients, a model learned on A is transferred to B by exploiting these small overlaps. Thanks to the envisioned MQTT-based procedures that have been defined, a second FL aggregator belonging to domain B can easily obtain information on the model trained by an FL aggregator belonging to domain A. This is possible at every stage of the training procedure, through a simple MQTT subscription.

ML model-agnosticism. To compute the local learning models using local datasets, various schemes can be used at end-devices. They could be, for instance, feed-forward neural networks (FNN), convolutional neural networks (CNN), support vector machines (SVM), and long-short term memory (LSTM). The type of used local learning model strictly depends on the considered IoT application [12]. The proposal is independent of the specific ML model to be trained. Hence, a variety of models and relevant frameworks can be supported by our solution.

8. Performance evaluation

8.1. PoC description

To assess the feasibility, efficiency and effectiveness of our proposal we built a realistic small-scale PoC detailed as follows.

8.1.1. FL-MQTT implementation

A hybrid population of FL clients is considered. Some of them are deployed through Docker containers [34], whereas one is implemented on a *RaspberryPi4*. Also in this latter case, the application that implements the FL client has been containerized. Another container hosts the FL aggregator.

The Docker containers hosting the FL clients and the FL aggregator are connected through the use of the Docker bridge network. A *MikroTik* switch connects the *RaspberryPi4*, hosting one of the FL clients, to the machine hosting the virtualized FL aggregator.

Both the FL clients and the FL aggregator include *tensorflow* [35] libraries for python.

For the MQTT client implementation on both the FL aggregator and the FL clients we leverage the *Mosquitto* software [26]. The MQTT broker, also deployed through *Mosquitto*, is co-located with the FL aggregator. Mosquitto is considered one of the most popular MQTT implementation due to its simplicity in installation, operating system portability, also because done in few lines of code [36]. Hence, the choice of Mosquitto is meant to further ensure scalability, as confirmed by the study in [37].

8.1.2. Dataset

We used two different datasets: the Canadian Institute for Advanced Research, 10 classes (CIFAR-10) [38] and the modified National Institute of Standards and Technology database (MNIST) handwritten digits [39]. They are both widely used in the FL research arena [4,40,41]. Moreover, we refer to them because they exhibit complementary characteristics in terms of total size of the datasets (163 MB for CIFAR-10 and 57 MB for MNIST) and size of the individual samples (2.5 kB for CIFAR-10 and 0.6 kB for MNIST).

The first dataset is made up of 60,000 color images with a size of 32 × 32 pixels. The images are labeled with one of the 10 classes available, which are mutually exclusive. The ten classes are: airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck.

The second dataset instead contains 70,000 grayscale images of hand-drawn figures, from zero to nine. Each image has the size of 28 × 28 pixels.

Starting from them, local datasets have been assigned to the various clients. Specifically, all FL clients have a dataset of the same size and equally distributed classes. In other words, for each class, all FL clients have the same number of samples. We therefore fall into a scenario where the data is IID.

8.1.3. Training model

CNNs are used to carry out the recognition of the images for both datasets. In particular, for CIFAR-10 we used the CNN proposed in [42], instead for MNIST we used the CNN proposed in [43].

The FL aggregator combines the locally trained models it receives from the FL clients according to the *FedAvg* policy [4].

Table 7
Values of the parameters in Eqs. (1) and (2).

Parameter	CIFAR-10 [kB]	MNIST [kB]
a	0.096	0.096
b	0.174	0.174
c	0.456	0.396
d	0.101	0.101
e	0.201	0.201

8.1.4. Benchmark

The proposal is compared against Google Remote Procedure Call (gRPC) [44], a high performance framework for calling remote methods by passing parameters alike local functions. It allows to define a service that will implement the interfaces which can be called by the remote entities to execute an operation. It is widely exploited in the FL context, e.g., in [45–47], as the communication channel between the FL aggregator and all FL clients. We leverage Flower, an open-source FL framework [45], as a representative implementation of such a solution.

8.2. Communication footprint results

The first set of results aims to assess the impact of the conceived framework on the network in terms of amount of exchanged data. To this aim, we measure through Wireshark the overall amount of data (in bytes) exchanged between the FL aggregator and FL clients through the broker, and carried in MQTT messages, during each stage of the FL procedure.

8.2.1. Client discovery and notification

The client discovery stage is crucial to allow the retrieval of information about the capabilities of the potential FL clients and drive a smart selection procedure. The workflow conceived in our proposal incurs a negligible overhead both in terms of exchanged messages and in terms of latency before the distributed training starts.

The amount of data exchanged during the discovery stage can be calculated as a function of the parameters N, $\tilde{N}$, as follows:

$$ExchangedData_{disc}(N, \tilde{N}) = (a + b) \cdot N + c \cdot \tilde{N}, \qquad (1)$$

where the parameters a, b, c are defined in Table 7 for the two different datasets.

They refer respectively to the traffic (in kB) generated: by each candidate FL client to subscribe to $topic\#1, disc$; by the FL aggregator when it publishes (on the topic $topic\#1, disc$) the *DiscoveryFL* object (forwarded by the MQTT broker to each candidate FL client); and by an FL client passing the self-eligibility check to publish the *ClientFL* object (on the topic $topic\#2, disc$). We conveniently remind that the subscription by the FL aggregator on the $topic\#2, disc$ is not considered as incurred communication overhead, since the FL aggregator is co-located with the MQTT broker.

A few additional bytes are required to be carried in the publish message (on the topic $topic\#2, disc$) to convey information about the capabilities of the potential FL clients needed to enforce a smart client selection, instead of the random strategy. Indeed, please notice that even when the random selection is enforced, the presence and identity of potential FL clients need to be discovered. Without notifying its capabilities, each candidate client would publish a 221 bytes-long message against 456 bytes and 396 bytes (parameter c), for CIFAR-10 and MNIST, respectively.

In this case, we consider the entire population of FL clients and measure the traffic exchanged during the notification stage as the number of selected devices varies.[2] The results can be generalized as

[2] A simple policy is assumed which selects the FL clients exhibiting the better processing capabilities.

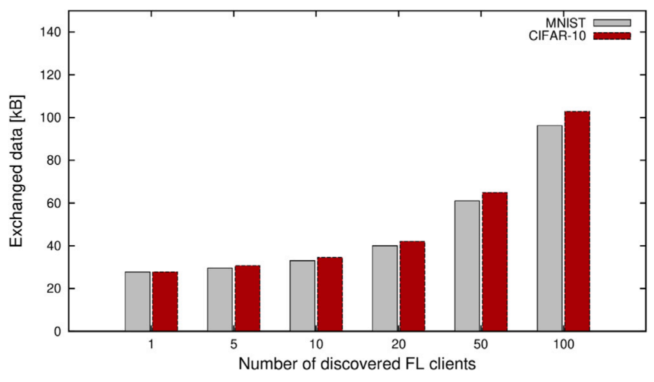


Fig. 5. Exchanged data in the discovery and notification stages when varying the number of discovered FL clients ($\tilde{N}$), $N = 100$.

a function of N, $\tilde{N}$ and M, and are respectively equal to:

$$ExchangedData_{sel}(\tilde{N}) = d \cdot \tilde{N} + e \cdot \tilde{N}. \quad (2)$$

The parameters d and e respectively represent the traffic generated by each FL client to subscribe to *topic#1,sel* and to receive the notification from the FL aggregator via the broker on the same topic. Their values are reported in Table 7. This metric is independent of the considered dataset.

Fig. 5 reports the overall amount of data exchanged during the client discovery and notification stages, according to Eqs. (1) and (2), whose parameters are derived from experimental results. For $N = \tilde{N}$=100 the amount of data exchanged during the client discovery and notification stages is equal to 102.8 kB in the worst case under the considered settings (i.e., CIFAR-10 dataset). This confirms the small overhead incurred by the procedure.

Given the limited amount of additional bytes exchanged to support the discovery of clients capabilities, a negligible additional time overhead is experienced.

Overall, the time spent in the client discovery depends on the number of FL clients to be discovered, on the connectivity conditions between the potential FL clients and the FL aggregator and on the MQTT broker capability to forward publish/subscribe messages.

Fig. 6 reports the client discovery time, since the subscription of the devices wishing to act as FL clients to the notification of the selected FL clients. Unlike previous results either derived when referring to a PoC with a limited number of clients or analytically, in this case, to scale with the number of clients, we deploy a measurement campaign with the Mininet network emulator [48]. Achieved results refer to the case in which all potential FL clients, deployed as Mininet hosts, connect to the same network node through dedicated links and the network node is connected to a Mininet host acting as ME host where the MQTT broker and the FL aggregator are co-located. 100 Mbps and 10 ms are the capacity and latency, respectively, set through the traffic control (tc) utility on links through which messages are conveyed to/from the MQTT broker to the FL clients. Such settings have been selected to resemble a congested edge environment.

Even in the worst case (100 discovered clients), given the limited size of exchanged messages, the delay is below 100 ms. Moreover, the envisioned smart client selection procedure entailing the exchange of the capabilities of potential clients (curve labeled as *w capabilities*) incurs a negligible increase of the discovery time compared to the random client selection (curve labeled as *w/o capabilities*).

Not surprisingly results confirm that the client discovery time is small and represents a very limited contribution of the total FL training delay as the majority of the time is spent on the local model training and global model aggregation.

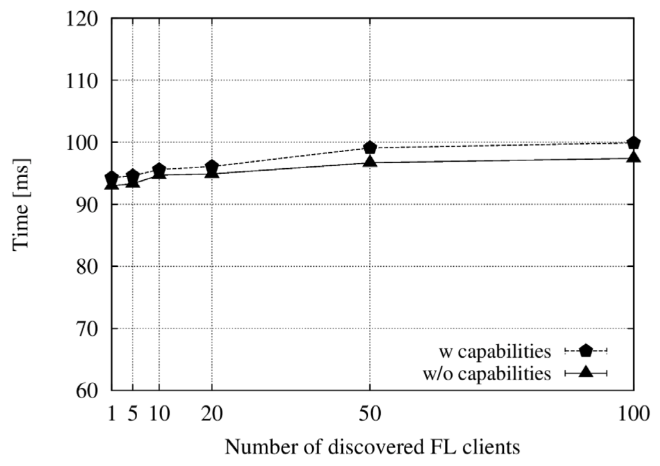


Fig. 6. Client discovery time when varying the number of discovered FL clients ($\tilde{N}$), $N = 100$, CIFAR-10 dataset.

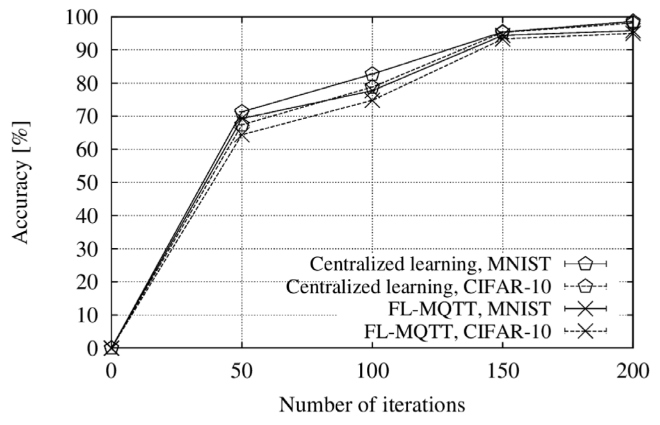


Fig. 7. Centralized Vs Federated learning: accuracy when varying the number of iterations.

8.2.2. Model training

We first compare the FL approach against a conventional centralized learning, in which a set of clients share their datasets with a server in charge of training procedures. In particular, results have been reported in terms of accuracy (see Fig. 7) for both the MNIST and CIFAR-10 datasets and compared with the accuracy achieved by the FL approach, when fixing the number of clients to 5 and when varying the number of iterations.[3]

First of all, we can observe that there are no remarkable differences between the centralized learning and FL approach. This confirms the FL effectiveness. In particular, the maximum training accuracy achieved after 200 rounds [4] by the FL global model is 95% for CIFAR-10 and 95.8% for MNIST. A training loss of 0.001 is reached for both datasets.

We then compare the amount of data exchanged by our proposal during the training stage, once FL clients are discovered, selected and notified, against the gRPC protocol.[4]

To achieve a fair comparison, the training procedure is set so to provide the same results in terms of accuracy and loss for both

[3] Please notice that iterations correspond to the global rounds performed at the aggregator in FL and to training epochs at the learner in the centralized learning.

[4] The comparison is not foreseen for the previous cases, because the Flower implementation does not natively provide primitives for the FL discovery and notification stages.

Table 8
Training settings.

Parameter	Value
Learning-rate (η)	0.25
Epoch	5
Mini-batch	50

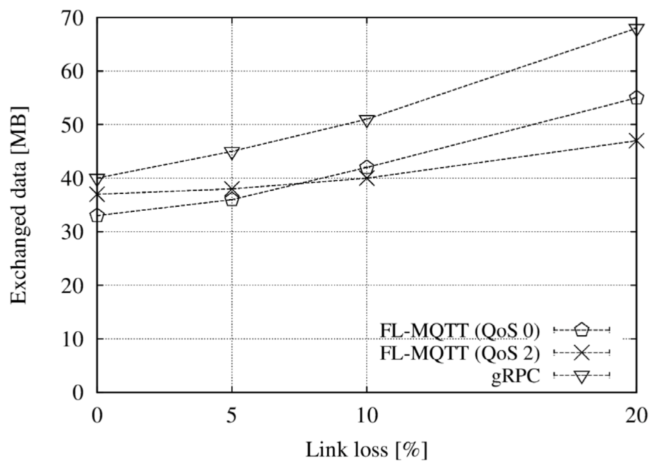


Fig. 8. FL-MQTT Vs. gRPC: exchanged data during the training procedure when varying the link loss (dataset CIFAR-10).

solutions, under the settings reported in Table 8 [4]. Measurements were carried out with 50% of the number of samples for both datasets.[5]

Fig. 8 shows the amount of exchanged data during the training procedure, when varying the loss probability over the links interconnecting the FL aggregator and 5 FL clients. The *tc* utility has been leveraged to set the packet loss probability over the aforementioned links to resemble realistic network conditions. The performance of FL-MQTT is reported both when the QoS level is set to 0 and when it is set to 2.

Results are reported for the CIFAR-10 dataset only to avoid the cluttering of the plot. However, no significant differences are noticed when varying the dataset. This is expected since the size of the original models for the two datasets are comparable: *(i)* 137 kB for the one used on the MNIST dataset and *(ii)* 175 kB for the one used on the CIFAR-10 dataset. More in detail, the size of publish messages carrying the structure of the NN sent on *topic#1,trai* is: *(i)* 143 kB for MNIST and *(ii)* 182 kB for CIFAR-10. The publish message containing the updated weight values, sent on *topic#2,trai*, has a size of: *(i)* 102 kB for MNIST and *(ii)* 122 kB for CIFAR-10.

Whatever the QoS level, the proposal allows to reduce the amount of exchanged data compared to the considered benchmark for both datasets. This is mainly because MQTT uses smaller headers compared to the gRPC-based solution which is based on the HTTP protocol [49].

Not surprisingly, the most efficient solution in absence of losses is FL-MQTT with QoS level 0, because it incurs the lower overhead.

Interestingly, as the link loss increases the improvements of FL-MQTT over the gRPC solution in terms of saved network resources for data exchange get more significant, for both datasets. This is true also when the QoS level 2 is enabled for FL-MQTT. In the latter case, for link loss equal to 20%, it is possible to exchange the lowest amount of data. This trend has to be ascribed to the fact that, thanks to the handshake enforced at the application layer, there is no need to retransmit the long publish messages in case of losses. Retransmissions are instead enforced at the transport layer by FL-MQTT (QoS 0) and gRPC.

Moreover, for the sake of completeness, to understand the benefits of FL compared to the centralized approach, the differences in terms of exchanged data should be also considered. In the absence of losses, less than 40 MB are exchanged with FL-MQTT, for the training conducted over the CIFAR-10 dataset. This value is well below the overall dataset size which needs to be exchanged in the case of centralized learning, in the order of 163 MB.

Table 9
CPU and RAM usage.

	gRPC	FL-MQTT
CPU [%]	19.46	14.87
RAM [%]	21.84	22.60

8.3. Processing and RAM footprint

To further understand the differences between the two compared messaging schemes, results are also reported in terms of average CPU and memory usage during the data exchange. Results only refer on the FL client side, since it could be more resource-constrained. Measurements have been made over a virtual machine hosted on a physical device equipped with an Intel Core i7-9750H processor, a 16 GB DDR4 RAM and an NVIDIA GeForce GTX1050 Ti graphics card. Instead, 4 GB of RAM and one processor thread of the physical device have been dedicated to the virtual machine.

Table 9 shows that the FL-MQTT approach exhibits a significantly lower impact on the CPU usage compared to gRPC, whereas a slightly higher RAM consumption is incurred.

The achieved trends confirm the better suitability of the MQTT protocol for constrained devices compared to gRPC.

8.4. Interoperability test

As discussed in Section 7, the proposal can facilitate transfer learning among different FL domains. The last set of results is aimed to quantify the benefits in terms of training time when enabling FTL.

We consider a primary domain, $d1$, with an FL aggregator, denoted as $A1$, and a set of FL clients, as described before, and a secondary target domain, $d2$, with another FL aggregator, denoted as $A2$. The latter one works with a set of FL clients having similar but different local datasets compared to the primary scenario. In particular, we used 50% of the unused samples in the previous measurements campaign.[6]

During the training procedure in $d1$, $A2$ tries to benefit from the trained model of the primary domain and shares it with its set of clients as a (partially trained) original global model.

The conceived publish/subscribe framework and the self-explanatory topics allow $A2$ to subscribe to *topic#1,trai* and *topic#2,trai* on the broker in order to obtain information on the structure of the NN model and on the global weights in $d1$, from round to round.

Fig. 9 shows the time saved by $A2$ to train the global model in its domain. Results are reported when varying the time (in rounds) after which $A2$ takes the model from $d1$, compared to the case in which it starts with an original untrained global model.

The higher the number of rounds, the better trained is the model in $d1$, hence the larger the benefits for $A2$, for both the considered datasets. Indeed, $A2$ saves more time to train the model with its set of clients, because the latter ones, by starting with a partially trained model, perform more quickly the local training.

[5] The remaining 50% is used to carry out interoperability measurements defined in the next subsection.

[6] This is meant to ensure balanced datasets and comparable accuracy results.

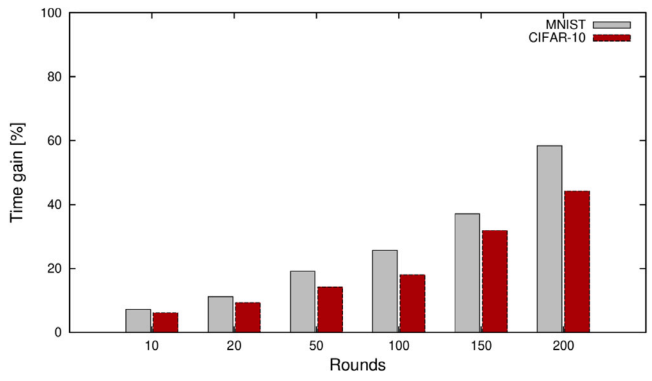


Fig. 9. Time saved when applying the FTL concept for different rounds in which the partially trained model is retrieved from the source domain for the target domain.

However, even when the complete trained model by $A1$ is retrieved by $A2$, i.e., after 200 rounds, $A2$ needs still some time to perform the training in $d2$ due to new samples in the dataset. All in all, half of the training time can be approximately saved in this case for both datasets.

9. Conclusions and future works

In this work, we have proposed a comprehensive edge-based framework to support FL operations. The proposal builds upon the MQTT protocol and the OMA LwM2M standard, reference solutions for IoT contexts. In particular, MQTT topics format as well as lighweight workflows have been defined to match the demands of the different FL stages: client discovery, client selection and model training. The payload of MQTT messages is overhauled with the OMA LwM2M semantics.

9.1. Main findings

Results collected in a realistic PoC show the superiority of the proposal compared to a popular FL implementation solution, i.e., gRPC, in terms of lower communication and computation footprint, under different datasets. This makes the solution particularly viable for IoT scenarios.

Interoperability is also proven, by assessing the benefits of letting a different set of FL aggregator and FL clients reuse partially trained models in a primary domain.

Since typically constrained IoT devices are involved, saving time in the overall training procedures, even in a small percentage, means that less resources are wasted by FL clients to perform the local training.

9.2. Open issues and future research directions

The work has room for further improvements.

Security issues need to be addressed. At this stage of research, the compared solutions have been fairly analyzed without security mechanisms in place. It is well known that in MQTT the responsibility to provide appropriate security features is left to the implementer. On the one hand, this feature makes the vanilla MQTT implementation prone to security and privacy attacks.

Indeed, MQTT uses unencrypted communication, which means that messages can potentially be intercepted and read by third parties. By default, MQTT includes weak authentication mechanisms, hence anyone can potentially connect to and publish messages to the broker. Moreover, MQTT does not provide any mechanism for verifying the integrity of messages, therefore messages could potentially be altered in transit.

On the other hand, the MQTT implementation is flexible enough to accommodate security mechanisms tailored to the considered environment.

A secure version of MQTT, MQTTS, can be leveraged which uses the Transport Layer Security (TLS) protocol to encrypt messages. The same protocol can be leveraged by gRPC. However, TLS has high computation and communication costs, not suitable for IoT devices, as targeted in our work.

Several works have been proposed to cope with specific security issues of MQTT in IoT [50–52].

Recently, the Internet Engineering Task Force (IETF) has proposed the Authentication and Authorization for Constrained Environments (ACE) [53] and Object Security for Constrained RESTful Environments (OSCORE) [54] approaches which offer authentication and authorization and end-to-end encryption and integrity, respectively, in the IoT domain. They appear promising although their coupling with MQTT has not yet fully investigated [55,56], hence paving the way for future research work.

Furthermore, the proposal can accommodate different multi-criteria client selection schemes, whose design will be a subject matter of future work. Moreover, as a further step forward we plan to assess the effectiveness and efficiency of the proposal for asynchronous [12] and decentralized [57,58] FL implementations, which better suit more dynamic and heterogeneous contexts.

CRediT authorship contribution statement

Claudia Campolo: Conceptualization, Methodology, Investigation, Writing, Reviewing, Supervision, Editing. **Giacomo Genovese:** Conceptualization, Methodology. **Gurtaj Singh:** Conceptualization, Methodology, Software, Validation, Investigation, Writing, Editing. **Antonella Molinaro:** Conceptualization, Methodology, Reviewing, Supervision.

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.

Data availability

Data will be made available on request.

References

[1] Internet of things. Ericsson, 2022, [Online]. Available at: https://www.ericsson.com/en/internet-of-things (Accessed: 07 Oct. 2022).
[2] Y. Chen, X. Qin, J. Wang, C. Yu, W. Gao, Fedhealth: A federated transfer learning framework for wearable healthcare, IEEE Intell. Syst. 35 (4) (2020) 83–93.
[3] I. Kevin, K. Wang, X. Zhou, W. Liang, Z. Yan, J. She, Federated transfer learning based cross-domain prediction for smart manufacturing, IEEE Trans. Ind. Inform. 18 (6) (2021) 4088–4096.
[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[5] O.A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1342–1397.
[6] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM, 2020.
[7] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: IEEE International Conference on Communications, ICC, 2019, pp. 1–7.
[8] J. Yao, N. Ansari, Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control, IEEE Internet Things J. 8 (5) (2020) 3438–3445.
[9] S. AbdulRahman, et al., FedMCCS: multicriteria client selection model for optimal IoT federated learning, IEEE Internet Things J. 8 (6) (2020) 4723–4735.

[10] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, C.G. Brinton, Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation, in: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, IEEE, 2021, pp. 1–10.
[11] J. Lee, H. Ko, S. Seo, S. Pack, Data distribution-aware online client selection algorithm for federated learning in heterogeneous networks, IEEE Trans. Veh. Technol. (2022).
[12] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for Internet of things: Recent advances, taxonomy, and open challenges, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1759–1799.
[13] A. Imteaj, U. Thakker, S. Wang, J. Li, M.H. Amini, A survey on federated learning for resource-constrained IoT devices, IEEE Internet Things J. 9 (1) (2021) 1–24.
[14] G. Genovese, G. Singh, C. Campolo, A. Molinaro, Enabling edge-based federated learning through MQTT and OMA lightweight-M2M, in: IEEE VTC Spring, 2022.
[15] A. Banks, R. Gupta, MQTT Version 3.1.1, OASIS Standard, 2014.
[16] Open mobile alliance, lightweight machine to machine technical specification core; V1_1-20180612-C, 2018.
[17] G. Cleland, D. Wu, R. Ullah, B. Varghese, FedComm: Understanding communication protocols for edge-based federated learning, 2022, arXiv preprint arXiv:2208.08764.
[18] A. Feraudo, et al., Colearn: Enabling federated learning in MUD-compliant IoT edge networks, in: EdgeSys, 2020, pp. 25–30.
[19] B.C. Tedeschini, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, L. Serio, Decentralized federated learning for healthcare networks: A case study on tumor segmentation, IEEE Access 10 (2022) 8693–8708.
[20] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for Internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1622–1658.
[21] D. Glaroudis, A. Iossifides, P. Chatzimisios, Survey, comparison and research challenges of IoT application protocols for smart farming, Comput. Netw. 168 (2020) 107037.
[22] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, J. Alonso-Zarate, A survey on application layer protocols for the Internet of things, Trans. IoT Cloud Comput. 3 (1) (2015) 11–17.
[23] OMA-LwM2M registry, 2022, [Online]. Available at: https://technical.openmobilealliance.org/OMNA/LwM2M/LwM2MRegistry.html (Accessed: 27 Oct. 2022).
[24] OMA-LwM2M object resource editor, 2022, [Online]. Available at: https://github.com/OpenMobileAlliance/ (Accessed: 27 Oct. 2022).
[25] ETSI GS MEC 003 v1.1.1. Mobile Edge Computing (MEC); Framework and Reference Architecture, ETSI, 2016.
[26] Mosquitto, MQTT implementation, https://mosquitto.org/.
[27] N. Tonellotto, A. Gotta, F.M. Nardini, D. Gadler, F. Silvestri, Neural network quantization in federated learning at the edge, Inform. Sci. 575 (2021) 417–436.
[28] B. Fortner, HDF: The hierarchical data format, Dr Dobb's J. Softw. Tools Prof. Program 23 (5) (1998) 42.
[29] Save and load models, 2022, [Online]. Available at: https://www.tensorflow.org/tutorials/keras/save_and_load?hl=en (Accessed: 07 Oct. 2022).
[30] B. Camajori Tedeschini, S. Savazzi, R. Stoklasa, L. Barbieri, I. Stathopoulos, M. Nicoli, L. Serio, Decentralized federated learning for healthcare networks: A case study on tumor segmentation, IEEE Access 10 (2022) 8693–8708, http://dx.doi.org/10.1109/ACCESS.2022.3141913.
[31] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.
[32] J. Dizdarević, F. Carpio, A. Jukan, X. Masip-Bruin, A survey of communication protocols for Internet of things and related challenges of fog and cloud computing integration, ACM Comput. Surv. 51 (6) (2019) 1–29.
[33] B. Mishra, A. Kertesz, The use of MQTT in M2M and IoT systems: A survey, IEEE Access 8 (2020) 201071–201086.
[34] D. Merkel, Docker: Lightweight linux containers for consistent development and deployment, Linux J. 2014 (239) (2014) 2.
[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., TensorFlow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.
[36] E. Bertrand-Martínez, P.D. Feio, V. de Brito Nascimento, B. Pinheiro, A. Abelém, A methodology for classification and evaluation of IoT brokers, in: LANOMS, 2019.
[37] M. Bender, E. Kirdan, M.-O. Pahl, G. Carle, Open-source MQTT evaluation, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2021, pp. 1–4.
[38] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, Citeseer, 2009.
[39] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[40] Z. Zhang, L. Wu, D. He, Q. Wang, D. Wu, X. Shi, C. Ma, G-VCFL: Grouped verifiable chained privacy-preserving federated learning, IEEE Trans. Netw. Serv. Manag. (2022).
[41] J. Li, Y. Shao, K. Wei, M. Ding, C. Ma, L. Shi, Z. Han, H.V. Poor, Blockchain assisted decentralized federated learning (BLADE-FL): Performance analysis and resource allocation, IEEE Trans. Parallel Distrib. Syst. 33 (10) (2021) 2401–2415.
[42] R.C. Çalik, M.F. Demirci, Cifar-10 image classification with convolutional neural networks for embedded systems, in: IEEE/ACS 15th International Conference on Computer Systems and Applications, AICCSA, 2018, pp. 1–2.
[43] L. Hertel, E. Barth, T. Käster, T. Martinetz, Deep convolutional neural networks as generic feature extractors, 2017, arXiv.
[44] gRPC, google remote procedure call, 2022, [Online]. Available at: https://grpc.io./ (Accessed: 07 Oct. 2022).
[45] D.J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, P.P. de Gusmão, N.D. Lane, Flower: A friendly federated learning research framework, 2020, arXiv preprint arXiv:2007.14390.
[46] P. Pinyoanuntapong, P. Janakaraj, R. Balakrishnan, M. Lee, C. Chen, P. Wang, EdgeML: Towards network-accelerated federated learning over wireless edge, Comput. Netw. (2022) 109396.
[47] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, M. Nordlund, Open-source federated learning frameworks for IoT: A comparative review and analysis, Sensors 21 (1) (2020) 167.
[48] Mininet, http://mininet.org/.
[49] T. Yokotani, Y. Sasaki, Comparison with HTTP and MQTT on required network resources for IoT, in: 2016 International Conference on Control, Electronics, Renewable Energy and Communications, ICCEREC, IEEE, 2016, pp. 1–6.
[50] H. Mrabet, S. Belguith, A. Alhomoud, A. Jemai, A survey of IoT security based on a layered architecture of sensing and data analysis, Sensors 20 (13) (2020) 3625.
[51] E. Lee, Y.-D. Seo, S.-R. Oh, Y.-G. Kim, A survey on standards for interoperability and security in the Internet of Things, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1020–1047.
[52] C.-S. Park, H.-M. Nam, Security architecture and protocols for secure MQTT-SN, IEEE Access 8 (2020) 226422–226436.
[53] F. Palombini, C. Sengul, Pub-sub profile for authentication and authorization for constrained environments (ACE), in: Internet Engineering Task Force, IETF, 2022.
[54] F.P.G. Selander, J. Mattsson, L. Seitz, Object security for constrained RESTful environments (OSCORE), in: Internet Engineering Task Force, no. RFC 8613, IETF, 2019.
[55] Z. Laaroussi, O. Novo, A performance analysis of the security communication in CoAP and MQTT, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2021, pp. 1–6.
[56] V. Seoane, C. Garcia-Rubio, F. Almenares, C. Campo, Performance evaluation of CoAP and MQTT with security support for IoT environments, Comput. Netw. 197 (2021) 108338.
[57] H.T. Nguyen, R. Morabito, K.T. Kim, M. Chiang, On-the-fly resource-aware model aggregation for federated learning in heterogeneous edge, in: 2021 IEEE Global Communications Conference, GLOBECOM, IEEE, 2021, pp. 1–6.
[58] S. Savazzi, M. Nicoli, V. Rampa, Federated learning with cooperating devices: A consensus approach for massive IoT networks, IEEE Internet Things J. 7 (5) (2020) 4641–4654.

Claudia Campolo is an associate Professor of Telecommunications at University Mediterranea of Reggio Calabria, Italy. She received a Laurea degree in Telecommunications Engineering (2007) and a Ph.D. degree (2011) from the University Mediterranea of Reggio Calabria, Italy. Before her current appointment, she was an Assistant Professor at the University Mediterranea of Reggio Calabria (2012–2020). Her main research interests are in the field of vehicular networking and 5G systems.

Giacomo Genovese received the B.Sc. and M.Sc. degrees in Telecommunication Engineering from the Mediterranean University of Reggio Calabria, Italy, in 2010 and 2014, respectively. In 2015 he has been a young research engineer at CTTC of Castelldefels, Barcelona, Spain. He served as Research Engineer in several national and European projects like the H2020 INPUT project. Currently, he is working as a young researcher at the ARTS Laboratory, at the Mediterranean University of Reggio Calabria, Italy. His research interests are in the field of IoT heterogeneous gateways design, IoT device virtualization, edge computing technologies.

Gurtaj Singh earned his Bachelor's Degree (2018) in Information Engineering and his Master's Degree in Telecommunications Engineering (2020) from the Mediterranean University of Reggio Calabria. In 2021 he worked in Alten as a telecommunications engineer and followed Huawei and Vodafone for the design of optical routes on the Italian territory. Since January 2022 he has been working as a Ph.D. student at the Mediterranean University of Reggio Calabria, Italy. His research fields are related to distributed artificial intelligence, network protocols for IoT, technologies for edge computing.

Antonella Molinaro is a full professor of telecommunications at the University Mediterranea of Reggio Calabria. Previously, she was an assistant professor with the University of Messina (1998–2001) and the University of Calabria (2001–2004), and a research fellow at the Politecnico di Milano (1997–1998). She was with Telesoft, Rome (1992–1993) and Siemens, Munich (1994–1995). Her current research focuses on 5G, vehicular networking and future Internet architectures.