Resource Identification in Fog-to-Cloud Systems: Toward an Identity Management Strategy

Alejandro Gómez-Cárdenas · Xavi Masip-Bruin · Eva Marín-Tordera · Sarang Kahvazadeh

Abstract

Fog-to-Cloud (F2C) is a novel paradigm aiming at extending the cloud computing capabilities to the edge of the network through the hierarchical and coordinated management of both, centralized cloud datacenters and distributed fog resources. It will allow all kind of devices that are capable to connect to the F2C network to share its idle resources and access both, service provider and third parties resources to expand its own capabilities. However, despite the numerous advantages offered by the F2C model, such as the possibility of offloading delay-sensitive tasks to a nearby device and using the cloud infrastructure in the execution of resource-intensive tasks, the list of open challenges that need to be addressed in order to have a deployable F2C system is pretty long. In this paper we focus on the resource identification challenge, proposing an identity management system (IDMS) solution that starts assigning identifiers (IDs) to the devices in the F2C network in a decentralized fashion using hashes and afterwards, manages the usage of those IDs applying a fragmentation technique. The obtained results during the validation phase show that our proposal not only meets the desired IDMS characteristics, but also that the fragmentation strategy is aligned with the constrained nature of the devices in the lowest tier of the network hierarchy.

Keywords Identity management · Identification · IDMS · Resource identity · Fog-to-cloud resource identification

I. Introduction

In the Internet of Things (IoT) [1] era, where any everyday object can be turned into a gadget capable to both interact with other computers and monitor its surrounding environment, the number of connected devices requesting the cloud services to process or store the data they generate is growing exponentially. This situation is originated in the fact that these devices are characterized by their multiple limitations in terms of hardware turning into the need for offloading tasks they cannot process, typically, to cloud datacenters.

Nevertheless, the rapid technological development and deployment of IoT devices [2] have led to the emergence of new use cases at the edge of the network where the use of the cloud infrastructure is not the most suitable solution, as for example, delay-sensitive applications that need to operate with a lower latency than the one offered by cloud, such as critical urban infrastructure or eHealth monitor devices. In these cases, the cloud's centralized nature and the conceptual distance between the cloud datacenter and the user/device requesting the service [3] prevent cloud to meet the low-delay requirement.

✉ **Alejandro Gómez-Cárdenas**
alejandg@ac.upc.edu

Xavi Masip-Bruin
xmasip@ac.upc.edu

Eva Marín-Tordera
eva@ac.upc.edu

Sarang Kahvazadeh
skahvaza@ac.upc.edu

Universitat Politècnica de Catalunya
Advanced Network Architectures Lab (CRAAX) located at Rambla de l'Exposició 59, Vilanova i la Geltrú, Barcelona, Spain 08800

Other cloud limitations, such as the lack of mobility support, undefined security and privacy policies, quality of service (QoS) issues and the need of a reliable Internet connectivity with sufficient bandwidth [4], have encouraged the emergence of new computing paradigms that put the focus on the devices that operate at the edge of the network.

An already existing approach came from the so-called Fog computing concept [5] [6]. Fog computing is a novel paradigm that proposes to extend the cloud capacities to the edge of the network, where data is generated, through some sort of distributed fog nodes (also called aggregator nodes) that allow to use network resources but with a reduced latency and thus, a better QoS. Key research initiatives in fog related areas are the OpenFog Consortium [7] and the mF2C [8], the latter leveraging the Fog-to-Cloud (F2C) concept defined in [9] (further discussed in section II).

Recognized the potential benefits brought by fog computing systems [10] and the new range of services and applications that it will drive, it is undoubtedly worth the study of the still open challenges (see [11]) that have to be overcome in order to have a fog framework that can be deployed.

In this paper we focus on the identity management system (IDMS) [12] challenge, specifically, in the management of the devices (resources) identities, which is one of the key functionalities of any fog computing control plane. In this sense, it is worth highlighting the fact that even when there are many IDMS proposals in the literature, as we review in section IV, they don't meet the fog paradigm requirements (as described in section III). This is due to: i) the highly dynamic network conditions expected in the fog computing environment caused mainly by the mobile devices; ii) the predominant centralized approach in the existing solutions, and; iii) the large amount of computing resources required by these proposals for a proper operation, not to be met by usually highly constrained devices at the edge of the network.

The main contributions of this paper are summarized next:

- Unlike other IDMS proposal, we assign a unique identifier (ID) to each device leveraging the model hierarchical topology instead of using the IP address as identifier.
- We assign persistent IDs that remain even if the node moves to another location.
- The fragmentation strategy during the ID management allows a more efficient use of the scarce hardware resources available in the devices.

The reminder of this paper is organized as follows. In section II we describe the F2C system model. In section III we study the relevance of the adoption of a proper identity management strategy and its most common requirements. In section IV the related work is discussed. The proposed IDMS is presented in section V. Section VI shows an illustrative use case of the adoption of the proposed IDMS. In section VII the results obtained in the validation phase are analyzed and finally, we conclude this work in section VIII.

II. F2C system model

F2C is a collaborative computing paradigm where resources are distributed in a hierarchical topology. The main difference between the original fog computing proposal and the F2C is that while in fog computing the cloud services will be used mainly for long-term data storage and data mining, in F2C it plays a more active role. For example, a service may be granulated to execute simultaneously the delay-sensitive tasks in the neighboring nodes and the non-critical tasks in the cloud.

It is said that F2C is a collaborative approach because it will allow users not only to use the available resources but also to contribute to the resources pool with idle resources in their devices, i.e., users will be able to share their idle compute capacity, memory and storage while connected to the F2C network.

In short, F2C is structured as follows: in the lower hierarchical tier the most constrained resources will be grouped (IoT devices). The middle tier will be integrated by the fog nodes playing as gateways for the devices in the lower tier, and finally, the cloud datacenter seating at the top.

For the sake of illustration Figure 1 shows the general F2C topology and communication model. Reacting to a user wiling to execute a service, the F2C system will select the best place to execute it, depending on the required data location, the maximum delay allowed, the service characteristics and some other parameters. Thus, the devices participating in the F2C infrastructure can execute tasks either in nearby devices at the same hierarchical tier, in a higher fog tier or even in the cloud datacenter according with the task requirements.

Consequently, F2C services will be able to use a combined set of resources that will not necessarily be located in the same hierarchical tier. Let's consider for example a service that can be decomposed into subtasks. The resulting subtasks can be assigned to nodes deployed throughout the network according to their specific requirements. Accordingly, the subtasks that need few compute resources can be allocated to constrained IoT devices and subtasks that require long-term storage or intensive processing to the cloud datacenters.

III. IDMS requirements in F2C systems

It is called identity management system (IDMS) to the set of tasks, techniques and procedures used to identify uniquely an individual or an object within a given context [13]. The IDMS is a key component that should be present in every level of the F2C hierarchy. It will facilitate, among other things, to control the access to the available resources in the network and to implement the most essential security features, such as authentication and isolation of devices that incur in malicious behavior.

It is worth emphasizing the importance of the adoption of the proper identity management strategy. In F2C, mobile devices will require to be identified constantly across the network, especially when they move from one fog node coverage area to another, for example, when device A in Figure 1 moves from fog node$_1$ to fog node$_2$. In order to be able to offer a seamless handover experience, it will be imperative that the adopted identity management strategy allows nodes to be identified in the shortest possible time. According to [14], IDMS will ease the management of services, data and devices. Also, it will provide support to the service providers during the development phase while protecting the user's privacy and hardware specifications, thus it is a crucial feature of the system.

In [13], the author argues that the requirements for the IDMS design criteria are closely tied to the use cases, that is, the requirements that the IDMS should meet are not fixed but they depend on the environment where such system will be implemented. For example, a desired characteristic of the telephone management system (which may be considered as a kind of IDMS, being the phone numbers the identifiers) is that the telephone numbers are easy to remember, that is, the memorability. Nevertheless, with today's computer specifications, where even the most constrained devices can process data much faster than humans, the memorability characteristic is not a priority in environments such as Fog computing or F2C. In fact, since such characteristic may represent a security issue –such as the identity forging thread addressed in [15] [16]–, it is not recommended.

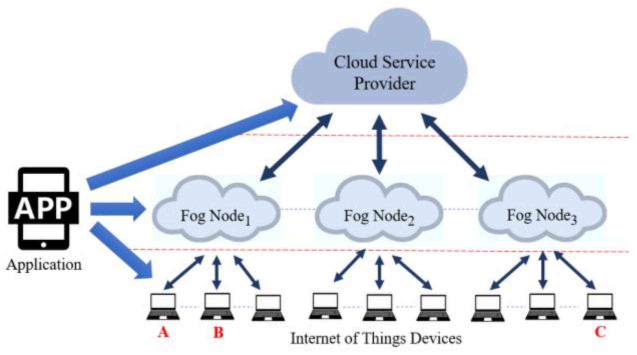


Fig. 1 Fog-to-Cloud topology and communication model.

In this section we have compiled a list of requirements that according to [13], [17] and [18] an IDMS like the one presented in section V should meet for a successful and effective management of identities in F2C systems.

- **Scalability.** The scalability is the capacity of the identity management strategy to get adapted to large volumes of changes. In F2C systems, it is essential that the IDMS component continues to work without losing quality or affecting other characteristics regardless the number of nodes in the F2C network.
- **Decentralization.** F2C is foreseen as a decentralized paradigm and thus its control functions should be decentralized as well. The distributed nature of the system not only allows it to keep operating even if a section of the network fails but also decrease the response delay by locating the key functions at the edge of the network topology.
- **Mobility.** One of the most important characteristics of the F2C paradigm is its support to mobility. Therefore, it is necessary that F2C individual functionalities, including the IDMS, provide such support to those devices on the move without degrading the QoS.
- **Uniqueness.** The identity management strategy must ensure that the resource identities are globally unique, at least in the scope of their network connections.
- **Security.** In raw words, security is responsible for enabling right individuals to access to the right resources at the right time and for the right reasons. The identity management aims at authentication process to provide data and information for authorized users. [19].
- **Privacy.** It is said that a function provides privacy when it prevents unauthorized users to consult, copy, modify or delete private information. In F2C systems, the adopted IDMS strategy must hide the end users and system sensitive information, including the data that can be used for inferring information, such as personal data, location, trajectories, behavior patterns, etc.
- **Interoperability.** This characteristic refers to the facility provided by the identity management strategy to exchange data with other service providers [20]. For example, sharing whether a device is registered in the network, its' ID and other data, with another F2C provider.

IV. Related work

Currently, in computer networks, there are two major categories in research related to identity management systems. The first category is the object oriented one, which, as its name suggests, focuses on identifying individual objects. The second category is known as user-centric and it is the most commonly implemented [21].

The OpenID is a user-centric identity technology created by an open source community that allows users to sign in to multiple sites without needing to create new passwords for each one of them [22]. The OpenID addresses the problem of having multiple user credentials, one for each web service, by relying the user authentication process to a centralized third party identity provider, which provides users with the OpenID identifier, an identity URL [23]. Although the OpenID approach is robust and widely adopted, its implementation in F2C environments to identify devices is not appropriate. For example, the centralized nature of the OpenID not only creates a single point of failure but also goes against the distributed F2C approach. This approach not only includes the distribution of resources but also control functions, including the identity management system. Even more, OpenID, as other user-centric solutions, focuses on reducing the multiple user credentials, which in F2C from the resource identification perspective isn't a problem.

In [24] the authors address another user-centric IDMS, the fingerprint. Such technique also has a version designed to uniquely identify devices, however, its effectiveness is conditioned by the need each device to be different, either in hardware, software or both [25], a condition that is unrealistic, especially in wireless sensor networks (WSNs) where a large number of basically identical devices are deployed. Even more, given the information collected from the devices, the use of this technique can be considered as a violation of the privacy of users.

In [26] the authors propose to use the identifier provided by the device vendor during the IoT devices identification process. However, as explained in [27], a standardized naming convention should be agreed in advanced by the manufacturers, which is unlikely to happen given the impact this would have on their infrastructure.

In order to avoid security risks related to the management of identities (user impersonation attacks, for example), authors in [28] propose a remote user authentication protocol that anonymize the identities in each login. Such anonymity is achieved by the use of a random nonce that encrypts the real identities and uses a dynamic identity in each session instead. The main issue with this proposal is that the performance evaluation focuses only on validating the proposal to be secure but unfortunately neglects performance aspects in constrained and legacy devices. In F2C, the constant encryption of the device ID could represent a problem for the most restricted devices, especially those mobile devices that go from one fog node to another one in a very short period of time (such as drones, intelligent vehicles, among others).

Authors in [29] propose an IDMS for the mobile cloud computing that strengthens the authentication process and identity privacy by the implementation of a two-steps authentication process (zero knowledge proof and token verification). The main issue with this strategy is that compared with other solutions, there is a communication overhead penalty of 30%. Besides that, the fog nodes overhead caused by the token generation task in highly dynamic environments due the devices mobility has not been considered.

In [30] authors present the Host Identity Protocol version 2 (HIPv2) which in order to provide support to the mobile nodes proposes to separate the host location and identifier from the IP address. However, as pointed out in [31], the HIPv2 key problem is that its implementation implies to change the TCP/IP protocols structure, which undoubtedly affects the functioning of existing networks, applications and even devices.

Authors in [32] present an approach for identifying IoT devices through the network traffic analysis. In their proposal, authors apply a set of machine learning based classifiers to a stream of sessions issued by a specific device. They claim that after a careful traffic analysis, it is possible to identify the device that generates such stream of sessions in the network. Nevertheless, this strategy presents important drawbacks that disallow to be applied into F2C systems: i) it does not identify devices individually but a classification of them (smartphone, computer, sensor, etcetera); ii) the identification of the network device is limited to a list of known devices, thus, new uncommon devices won't be recognized; iii) it is necessary to allocate compute resources to the traffic analysis task, and finally; iv) scalability may be a problem in highly heterogeneous scenarios.

In [33] an IDMS architecture that uses the cloud datacenters is introduced. Such work aims to provide device authentication and authorization among heterogeneous mobile networks. However, when implementing this proposal in a F2C environment, the QoS will be degraded by the inherent latency associated to the cloud usage during the identification, affecting mainly the mobile nodes.

Authors in [34] propose an identity management framework specifically tailored for Internet of Things scenarios, leveraging a centralized database to store the devices identities. Communication between two nodes is triggered by the requester node (client) when getting a token from the identity store, only valid for that specific connection. However, a key concern of this proposal seats on the single point of failure built by the centralized database of the identity store. A failure on the identity store would make the identity management solution not to properly work. It is worth mentioning that the required effort to manage the token will also incur in a delay that may be not negligible.

V. IDMS proposal

For a better understanding, this section has been divided into identity assignment and management. In the first one, we address the naming problem in F2C systems and in the second one we describe the identity management strategy we are proposing.

A. Identity assignments

In our previous work [35], we proposed a new hash-based identity management for combined F2C system. The proposal consists of three modules: certification, hash function and identification.

1. **Certification:** In the early stage and as a part of the F2C service subscription, users must fill in and submit a form in the F2C webpage to get their secret key. Then, users can register their devices on the F2C system using the obtained key. The certification provides secure channel between users and F2C system.
2. **Hash function:** This module uses the SHA-512 hash algorithm [36] to transform the device identification input into a 128-bytes fixed-length hash string which afterwards will be used as device ID. The device identification input are two concatenated strings: the secret key obtained during the registration phase and a random string. Once the device ID has been generated, it is stored in a distributed fashion among the F2C key nodes using Distributed Hash Tables (DHT).
3. **Identification:** The last step of the proposal is to search device's ID in the DHT. In this matter, some cases might be occurred. For example, consider figure 1. Device *A* appears for the first time in F2C system and it is in the vicinity of Fog node 1. Fog node are able to provide identity for the device and store it in the DHT over the whole F2C system. In another case, Device *C* is already registered and has the hash value (ID), when it arrives to the Fog node 3, it can easily look up in the DHT to find out the device *C*' identity. All the device's ID participants in the F2C system are stored in DHT by time stamp to track devices. One of the main advantage here is mobility facility. For example, if Device *B* already registered by fog node 1, moves to fog node 2, it can be found in the F2C system due to DHT across key components. All the device identity assignments and registration in DHT are done in hierarchical manner by nearby fog nodes or cloud.

B. Identity management

In our identity management proposal, the large global identities are partitioned into small fragments [37]. Those fragments facilitate network resources to be identified by a small fraction of their name rather than the full global identifier according to their connection layer (CL) in the F2C system.

The connection layer (CL) is the hierarchical view between different nodes in the F2C system. The CL in F2C will be given by the cloud as higher layer in F2C hierarchy. According to [38], three hierarchical levels may be considered for fog computing and thus, for F2C systems.

Although, inter-service-provider interaction must be considered as a layer in F2C scenario. Therefore, we classified the four hierarchical CLs (figure 2) in F2C system as below:

- **Edge:** In this connection layer, all connection occurred between the components (Physical or virtual) under the same fog node. In this type of CL, resources from an area at the edge of the network are geographically at vicinity of each other. For example, a shopping mall building can be considered as an area at the edge of the network.
- **Fog:** The CL of the fog is the connections between the fog nodes and the resources that they aggregate. For example, connection between a temperature sensor in one area and a laptop in another area under different fog nodes.
- **Cloud:** This CL has the overview of the all resources (might be geographically located far from each other) that established connection to the F2C system by a common cloud service provider.
- **Global:** The type of CL is the interconnection between all service providers of services globally. The resources may geographically have located far or near to each other but be connected to different network. Let's say F2C service provider A and F2C service provider B.

As illustrated in figure 3, The CLs are classified in four categories. In the F2C system, the number of layers might be changed, therefore the CLs and ID's fragmentation policy might be changed as well to be matched properly with the F2C layers. Therefore, it is worth clarifying that the four fragments policy may be considered as the general use policy and that it might change to fit other use case's needs.

When the connection layer has been defined for a system topology, the identifiers will be divided into *n* parts, where *n* is the number of CLs in the F2C system. The nodes in the F2C system will use the fraction of identifiers according to the hierarchical tier they are located for mutual identification instead of the full identifier. According to the node position in the hierarchical tier, the number of fragment of identifiers are changing. For example, assuming the network topology shown in figure 3, if the device tagged as *B* requests a storage resource located in the fog node$_2$, such CL will be tagged as fog and thus, two fragments of the full identifier will be required during the mutual identification process. Assuming identifiers as the one shown in figure 2, the full device identifier will be fractioned according to the CL as shown in the right of the figure 3.

In the topological view, in the higher CL, the more ID fragments will be used. Therefore, in the higher CL, larger ID will be used. The reason behind that is a higher node in the hierarchical F2C system has more devices (children) below itself in hierarchy. Then, for identifying each one of the children individually, longer identifiers will be used.

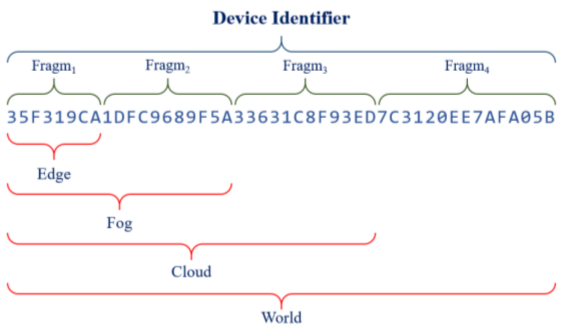


Fig. 2 Device's ID fragmented by connection layer.

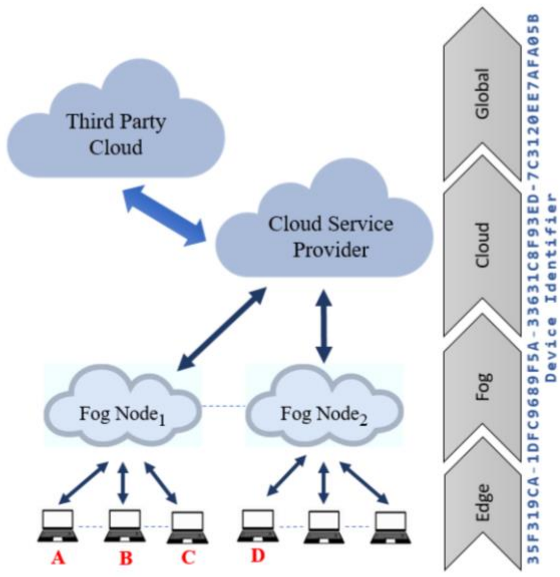


Fig. 3 Fragmentation policy according to the CL between two nodes.

Due to the use case's need and implementation, the length of ID' fragments may be different from fragment to fragment. In the lowest tier in F2C systems (IoT layer), the length of first ID's fragment depends on maximum number of resource's ID that fog node (as aggregator node) can store in cache during a specific time, that is, the identifiers cache size. If the fog node can store larger identifiers cache sizes, then a larger identifier fragments is required. In the IoT layer, small ID's fragments and cache sizes occurred due to their low computational power and resource limitation. Note that adjusting the ID's fragment length in function of the fog node cache size must be considered, otherwise, ID's collision problems will arise.

There are many different research contributions addressing the collision problem in the naming scenario (see [39] [40] [41]). Interestingly, in our identity management proposal, we define a collision occurs when two or more resources in a CL use the same identifier. Consequently, meeting what the main objective for an ID is to unambiguously identify a resource, the collision probability, as defined in (1) must be reduced as much as possible.

$$P(collision) = c \, / \, e^{l} \qquad (1)$$

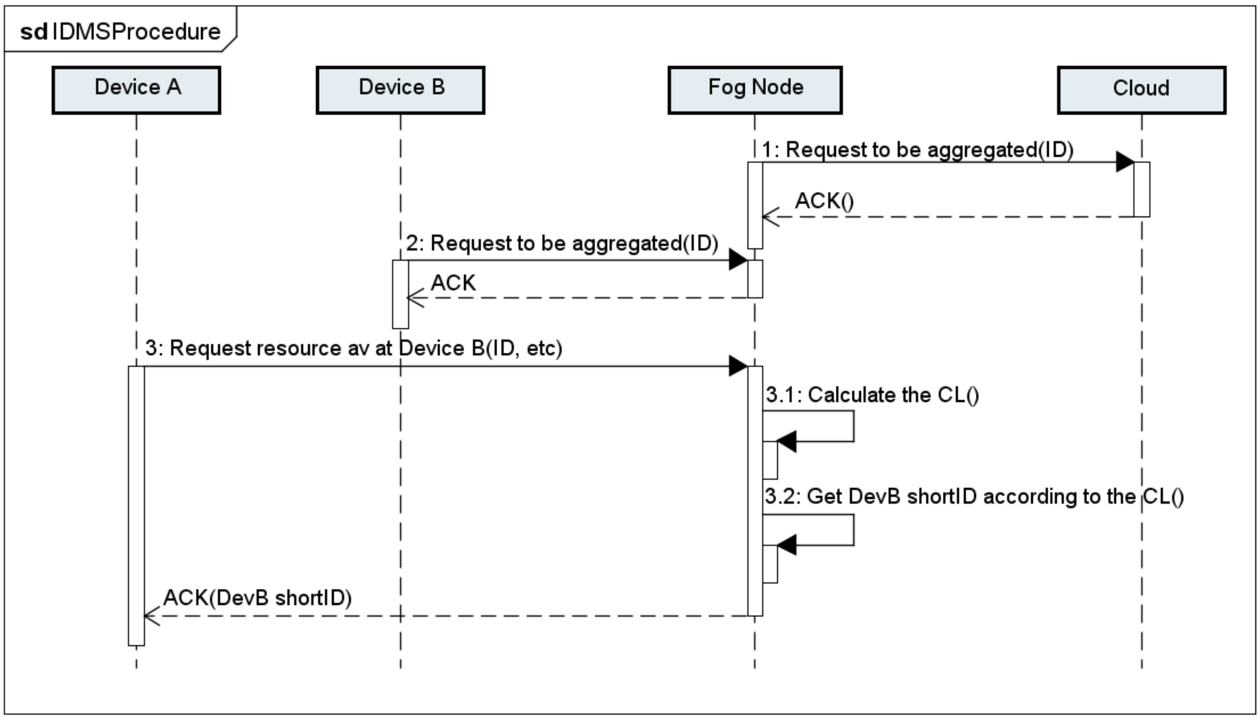

sd IDMSProcedure
Device A
Device B
Fog Node
Cloud
1: Request to be aggregated(ID)
ACK()
2: Request to be aggregated(ID)
ACK
3: Request resource av at Device B(ID, etc)
3.1: Calculate the CL()
3.2: Get DevB shortID according to the CL()
ACK(DevB shortID)

Fig. 4 IDMS procedure.

In (1):

- ***c***: It is the maximum number of IDs that the aggregator node can store in cache
- ***e***: It is the number of elements in the character set used for building the ID.
- ***l***: it is the pretended ID fragment length.

For avoiding and preventing ID collision, nodes in global connection layer use their full ID rather than a fraction of it during the identification process.

In fact, one of the main advantages of the proposed identity management is that the full resource identifier is not spreading nor storing through the whole network but its only know by: i) the resource to which the ID belongs; ii) the fog node when resources are connected through it to the F2C system and; iii) other resources in global connection layer where the full resource's ID is required for a proper identification and the decrease in collision risk.

Fog nodes play a vital role in our proposal due to be responsible for sharing the required resource ID fragments with other nodes according to the CL in the hierarchical F2C systems. For example, in the figure 3 if the node *A* requests a connection to the node *C*, the aggregator node that groups both of them will determine the CL and the number of fragments that such connection requires. In this case, only the first fragment will be used.

The sequence diagram shown in figure 4 describes the procedure to be followed when two nodes in the F2C need to communicate. As remark, we want to mention that in this scenario, we consider that nodes will be authenticated in the previous steps before communicating and all communication between layers, devices, users, etc. will be through secure channels. The steps 1 and 2 and their respective ACK are the initial preconditions to have the system running. In the first interaction, a node that can exercise the role of aggregator node asks the cloud agent to add it to the list of aggregators available in the fog tier. The cloud agent responds with an ACK. This process is repeated in the second interaction, with the difference that the node that requests to be added is in the lower tier of the network hierarchy, and the aggregation request is made to an aggregator node in the fog tier, instead of the cloud agent.

Let's assume that the Device *B* in figure 4 is offering a resource -hardware such as processor or storage, a service or data collected by sensors- that the Device *A* is requiring and that Device *A* is also part of the F2C network. The Device *A* will request to the network for the desired resource in a secure fashion and when it finds the resource available in an aggregator node, it will submit its ID among other information securely. Using such information, the aggregator node will calculate the CL of the interaction and based on such CL, the Device *B* identifier fragment to be shared with Device *A*.

VI. Illustrative scenario: deploying a F2C system in a smart city

This section introduces an illustrative example to understand what the envisioned procedures and assumptions related to the naming strategy when deploying a F2C system will be. For the sake of realism, we consider a city as the scenario where the F2C system will be deployed as well as some preliminary assumptions.

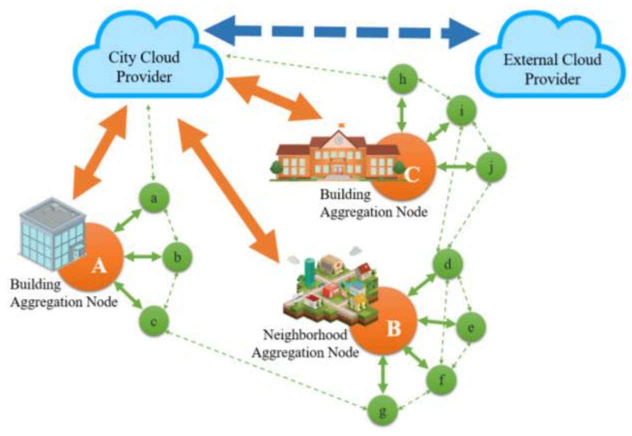

City Cloud Provider
External Cloud Provider
Building Aggregation Node
A
Building Aggregation Node
C
Neighborhood Aggregation Node
B
a
b
c
d
e
f
g
h
i
j

Fig. 5 Illustrative scenario.

Let's suppose the local government of a city, along with a regional Internet Service Provider is planning to modernize the city downtown by enriching the set of services that the city offers to the citizens. To that end, diverse IT infrastructure is deployed enabling the execution of innovative services, such as smart traffic management, urban surveillance, real time environment information (weather, pollution, noise, sunrise, sunset, etcetera), or enriched dependable e-health services, just to name a few. Assuming the fact that the deployed resources will not keep operating at full capacity all over the time, it is proposed to offer the idle city resources as an additional service to citizens, so paving the way for: i) a new collaborative model where citizens may execute other tasks and services in these idle city resources, and; ii) the creation of an environment where citizens can also contribute by sharing idle resources in their own devices.

In order to implement such resources and services management, the city manager has decided to deploy a F2C management solution, thus organizing the available resources in layers according with their capabilities, as follows (figure 5). The lower tier, also called IoT layer, is integrated by the most constrained resources, such as sensors, actuators and other basic computing devices –e.g. Raspberry Pi or Arduino boards– and is represented in figure 5 by the green circles connected to their respective fog nodes. In the second tier (orange circles marked with capital letters), the resources with medium processing power are grouped. These resources deploy control and aggregation functions for resources in the immediate lower tier and also when needed, provide their own resources to execute tasks that cannot be executed in the IoT resources. Finally, the upper tier, consists of the datacenter located at the ISP facilities. This tier is mainly used to execute demanding tasks and long-term data analysis and storage.

During the initial F2C deployment, a key decisions must be made regarding the way in which the resources participating in the network will be identified. It refers to the length in which the resource IDs will be fragmented in every tier of the network. To that end, it is worth highlighting the differences brought when considering the IoT layer. Indeed, while a F2C manager may pretty accurately know the amount of resources deployed at the cloud and fog tiers –easy decision about the length for the fragment that correspond to those tiers–, there is no clue on the maximum number of resources connected and stored in the aggregation node cache for a given period of time in the IoT layer. Hence, while an easy decision can be taken about the length for the fragment that corresponds to the cloud and fog tiers, an estimation is needed for the IoT layer. We also assume that, in the illustrative scenario for the envisioned city, the downtown area with the highest population density hosts every day up to a maximum of 1.5 million people of which 500,000 live there, 700,000 are employees that work in the zone and the rest are visitors.

Thus, the city manager must choose the optimal ID's fragment length for the IoT layer assuming that the F2C framework is preconfigured to store in every aggregation node a cache, with information of the resources connected to them during a specific period of time that the people in the area are potential service users with a single device, and that they all are managed by a single aggregation node. As a first approach, although certainly further work is needed to evaluate the impact of varying this period, we set that period of time to the last seven days.

From the aforementioned data, the city manager gets two of the three variables set in (1), namely *c* and *e*. In this case *c* is the number of people that every day visit the area multiplied by seven, plus the number of people that every day goes to the area to work, plus the number of people that live there. Due to the adopted naming schema, IDs are generated using only the hexadecimal charset, therefore the value of *e* in (1) is 16.
The third and last variable of (1) is the desired length of the ID's fragment *l*. We assume that the city manager prefers to prioritize a low ID collision probability over a short fragment. Therefore, the collision probability for fragments of 8 characters' length in hexadecimal representation is calculated.

The result (2) shows that the probability of collision is 0.00076, that is, 0.076%.

$$P(collision) = 3{,}300{,}000 \,/\, 16^{8} = 0.00076 \qquad (2)$$

Getting a smaller collision probability, would require the city manager to consider shorter fragments of 10 characters' length, as shown in (3).

$$P(collision) = 3{,}300{,}000 \,/\, 16^{10} = 3x10^{-6} \qquad (3)$$

The obtained results for the collision probability with the new parameters can be considered negligible and thus this would be the fragment length chosen by the project manager for the IDMS in the proposed F2C implementation.

Table 1. IDMS features.

Feature	Description
Scalability	The fact that the aggregator nodes will provide the identity management service to the nodes which they group will allow the IDMS to scale well regardless the number of devices connected to the network.
Decentralization	In F2C the control functions will be decentralized. The identity management by the aggregator nodes that we propose is aligned with such decentralization, what eliminates the single point of failure. In fact, the service will continue operating normally even if the cloud fails.
Mobility	In our proposal, each F2C node will have its own identifier, which will allow mobile nodes not to lose their identities, even when changing their IP. Likewise, in order to provide a seamless service to mobile devices that will require to be identified constaly by the aggregator nodes, we put the focus in the reduction of the time required for identify a node in the network.
Uniqueness	Each node in the F2C system will have its own globally unique identifier. The fragmentation strategy that we propose will allow the devices to have an identifier that still is unique in the context of their connections.
Security	A key feature of the IDMS is the capability to provide security to the systems. Our proposal will allow the nodes to be identified unambigusoly using a short version of their full identifiers.
Privacy	Using a short version of the node identifiers will disallow the scalade of masquerade attacks. Likewise, the non-incorporation of prefixes to perform the partitioning will prevent sensitive information from being extracted or inferred from the ID.
Interoperability	With few modifications and agreements, the IDMS presented can be implemented in fog computing and F2C systems from different service providers.

VII. Evaluation and results

In this section we describe the evaluations carried out for the presented proposal. The rationale behind this evaluation is to qualitatively validating the fulfillment of the proposal with the expected characteristics of an IDMS (see section III) as well as analyzing the results obtained during the evaluation phase.

In this sense, it is worth emphasizing that during the evaluation of the proposal we have focused on the lowest tier of the network hierarchy, the IoT layer. This is because from the identity management perspective and due to the dynamism that characterize the lower network tier, it is the most challenging to manage in the whole F2C system.

Thus, in order to validate our naming strategy, we have studied the time required to generate an ID in two of the most typical and constrained IoT devices: an Arduino and a Raspberry Pi. Likewise, to validate our fragmentation proposal we have created a fictional scenario where a Raspberry Pi plays the aggregator (fog) node role. Such scenario will be further described in subsection C.

A. IDMS characteristics validation

The motivation that drove the development of this work was the lack of an identity management system that supports the features offered by F2C system. In this sense, we argue that the IDMS that focus on the aforementioned approach must take into consideration the requirements that the environment demands.

In this section, we return to the requirements described in section III and explain how the proposed solution meets them (table 1).

B. Naming assignment validation

Since the devices grouped in the IoT layer will be very constrained in terms of computational power, it is imperative that the functions deployed in them are in accordance with said limitations. For that reason, we have implemented and validated our naming assignment proposal in two constrained IoT devices, an Arduino UNO WiFi rev2 with an ATMEGA4809 microcontroller that operates at 16Mhz, 48 kb of flash memory and 6.14 of SRAM [42] and a Raspberry Pi v3 which incorporates a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB memory RAM [43].

In our experiment, we have measured five times the time it takes to each of the devices described above to calculate an ID using our hash-based proposal. The averaged results are shown in table 2.

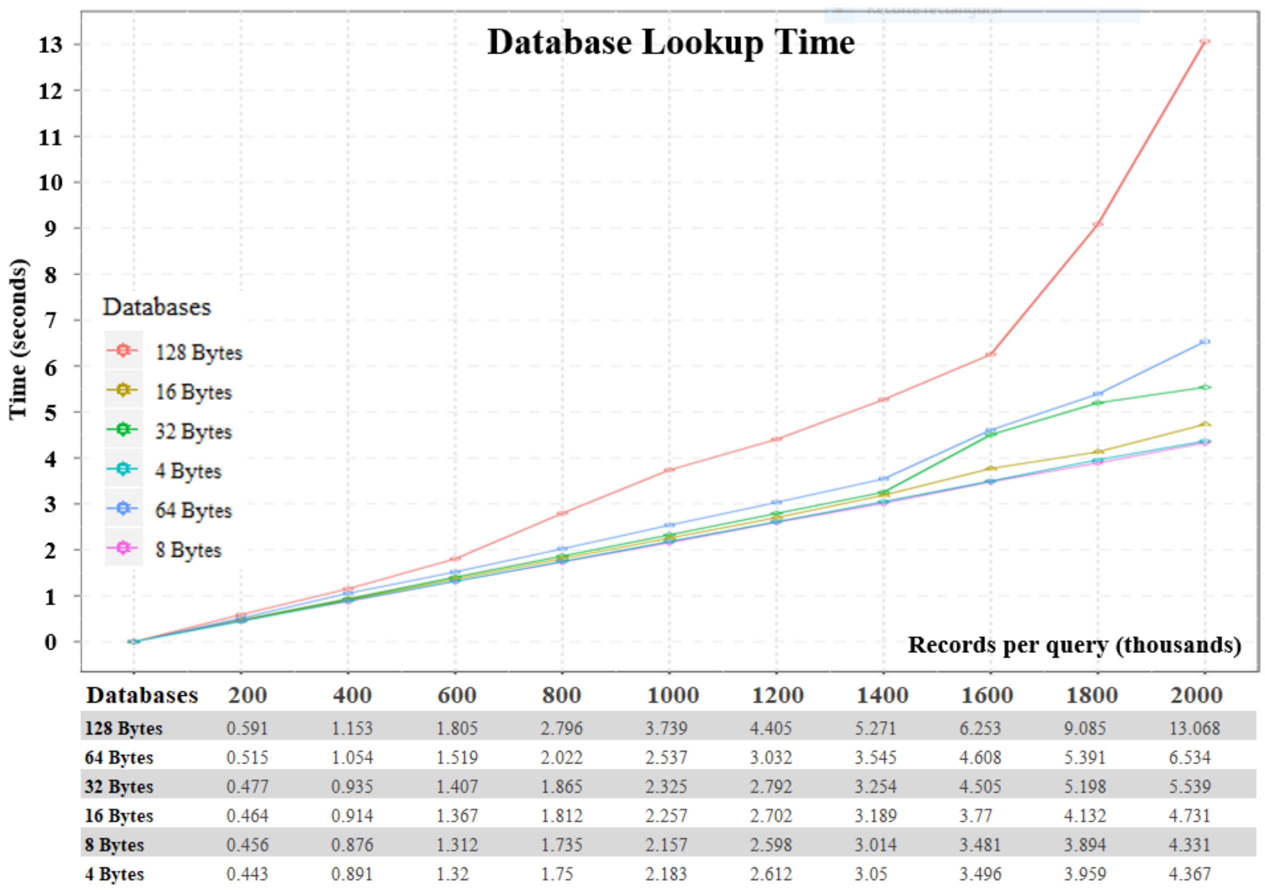
As can be expected given the specifications of both devices, the Raspberry Pi can process more bytes per second than the Arduino board. Nevertheless, the purpose of this validation is not to compare the hash rate between those devices but to prove that even the most constrained IoT devices are capable to execute our naming solution with an acceptable hash rate.

C. Database lookup time

To validate our identity management proposal, we have chosen two well-defined parameters that allowed us to measure the effectiveness of our proposal by comparing the performance obtained before and after applying our solution. The used metrics are the lookup time in the database that stores the identifiers and the space in disk that the DB uses.

The reasons why we have chosen these parameters are: i) in F2C all the framework components, including the IDMS, must be able to perform their function efficiently, otherwise, an uncontrolled delay in any of them may cause a bottleneck in the system and degrade the QoS. When an aggregator node receives a connection request, it should perform a lookup in the DHT in order to validate that the device ID is authorized in the system. In all the cases, the goal is to provide a seamless experience, especially to the mobile nodes that are using or sharing a resource on the go, so then, the less time it takes to search in the database, the faster the node will be connected to the network; ii) given the limitations of the constrained IoT and legacy devices, the resources, including the storage, must be used in the most efficient way possible. It means that the space in disk that each component uses matters.

Therefore, we have developed a testbed where a Raspberry Pi v3 acts as aggregator node. In the Raspberry Pi we have installed the Ubuntu Server 16.04 as Operating System and a Database Management System (DBMS). In the DBMS we have created and filled six databases (DB), each one with 2 million of synthetic identifiers. In the first database the ID length was set as 128 bytes. In the next databases the ID length was set as 64, 32, 16, 8 and 4 bytes respectively.



Databases	200	400	600	800	1000	1200	1400	1600	1800	2000
128 Bytes	0.591	1.153	1.805	2.796	3.739	4.405	5.271	6.253	9.085	13.068
64 Bytes	0.515	1.054	1.519	2.022	2.537	3.032	3.545	4.608	5.391	6.534
32 Bytes	0.477	0.935	1.407	1.865	2.325	2.792	3.254	4.505	5.198	5.539
16 Bytes	0.464	0.914	1.367	1.812	2.257	2.702	3.189	3.77	4.132	4.731
8 Bytes	0.456	0.876	1.312	1.735	2.157	2.598	3.014	3.481	3.894	4.331
4 Bytes	0.443	0.891	1.32	1.75	2.183	2.612	3.05	3.496	3.959	4.367

Fig. 6 Queries execution times.

The database with identifiers of 128 bytes was considered as the database that stores the full devices' identifiers. The shorter IDs stored in the following databases were considered as the possible lengths of the first ID fragment, the one that corresponds to the edge CL.

After that, we executed lookup queries in all the DBs considering different volumes of data. In each DB we conducted queries with 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800 and 2000 (thousands of) records respectively in order to be able to analyze how the times behave as the volume of data increases. Each query was repeated ten times and the results were averaged and presented in figure 6.

From figure 6 we can observe that the execution time in all the DBs is so similar when the data volume is low (200 thousands) but as that volume of data increases, the graphical lines begin to break that tendency, highlighting among them the one that represents the DB with the 128 bytes identifier because it requires the most time to process the queries, especially when the data volume is too high.
On the other hand, the database that requires the least time to process the queries is the one that stores the 4-byte fragments. However, it can be seen that the differences between the queries execution times in the 4-bytes and the 8-bytes databases are minimal, so much so that in figure 6 the lines are basically overlapping. Such behavior is caused by the index processing in the DBMS engine.

Table 2. IoT devices hash rate.

Device	Bytes processed per second
Arduino	7,667.63
Raspberry Pi	42,004.48

Table 3. Databases sizes.

Database	Size (MB)	%
128 Bytes	312.80	100%
64 Bytes	164.67	52.64%
32 Bytes	110.63	35.36%
16 Bytes	79.59	25.44%
8 Bytes	64.58	20.64%
4 Bytes	55.58	17.76%

In this case, it is convenient to take the length of the first fragment of the identifier as 8 bytes instead of 4, as this will greatly reduce the probability of collision in the identifiers without significantly increasing the time required to perform queries to the DB.

D. Database size in disk

Using the scenario and DBs described in the previous subsections, we have measured the space in disk that each DB uses. The results are summarized in the table 3.

The table 3 shows that using the presented fragmentation strategy also the storage required to store the databases is markedly reduced. The database that stores the 4-bytes IDs fragments only uses the 17.76% space in disk compared with the 128-bytes IDs database. Likewise, the database with the 8-bytes IDs fragments uses 79.36% less than the DB with the full identifiers.

VIII. Conclusions and future work

F2C has been designed as a solution to efficiently manage the resource continuum from the edge up to the cloud. A F2C system brings remarkable benefits, such as the possibility of executing services and applications closer to the end users and thus, with low-latency, it also facilitates mobility and handover by distributed fog nodes, and allows users to have more control over their data. However, there are still open challenges and issues that must be addressed. One of the main challenges in F2C system is the lack of an Identity Management System that meets the environment requirements.

In this work we propose an IDMS that consists of a hash-based naming strategy and a new hierarchical identity management technique. The proposed strategy assigns to each device a unique (ID) that is partitioned into smaller fragments according to the device hierarchical position in the F2C system. The fog node can determine the connection layer and according to that, the number of required fragments for a proper mutual identification among devices.

In the evaluation part, we illustrate that database sizes and query execution times both are decrease significantly when compared to a strategy using the full resource identifier, with a very low and hence affordable increase in the collision probability. The proposed strategy facilitates the efficiently usage of the limited resources at the edge of the network in F2C systems due to the aforementioned reductions and finally the identification's time process is markedly reduced. Likewise, the qualitative analysis shows that the proposed IDMS meets the main features that the identity management system must offer in a F2C environment.

As a future work, we plan to develop and implement the proposed hierarchical IDMS in a close-to-real scenario to assess the proposed solution benefits in a F2C system.

Acknowledgment

This work is supported by the H2020 mF2C project (730929), by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund both under contract TEC2015-66220-R (MINECO/FEDER), and for Alejandro Gómez-Cárdenas by the Consejo Nacional de Ciencia y Tecnología de los Estados Unidos Mexicanos (CONACyT), under grant No. 411640.

References

1. Evans, D.: The Internet of Things: How the Next Evolution of the Internet is Changing Everything. (2011).
2. S. K. Datta, R. P. F. Da Costa, C. Bonnet: Resource discovery in Internet of Things: Current trends and future standardization aspects. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). pp. 542–547 (2015).
3. Y. Zhou, D. Zhang, N. Xiong: Post-cloud computing paradigms: a survey and comparison. Tsinghua Science and Technology. 22, 714–732 (2017).
4. Firdhous, M., Ghazali, O., Hassan, S.: Fog Computing: Will it be the Future of Cloud Computing? Presented at the Proceedings of the Third International Conference on Informatics & Applications, Kuala Terengganu, Malaysia (2014).
5. Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., Koldehofe, B.: Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things. In: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing. pp. 15–20. ACM, New York, NY, USA (2013).
6. H. Hong: From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom). pp. 331–334 (2017).
7. OpenFog Consortium: OpenFog Reference Architecture for Fog Computing, (2017).
8. mF2C Consortium: mF2C Project Overview, http://www.mf2c-project.eu/project-overview/, (N.D.).
9. X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, G. J. Ren: Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. IEEE Wireless Communications. 23, 120–128 (2016).
10. Ramirez, W., Masip-Bruin, X., Marin-Tordera, E., Souza, V.B.C., Jukan, A., Ren, G.-J., Dios, O.G. de: Evaluating the benefits of combined and continuous Fog-to-Cloud architectures. Computer Communications. 113, 43–52 (2017).
11. Y. Liu, J. E. Fieldsend, G. Min: A Framework of Fog Computing: Architecture, Challenges, and Optimization. IEEE Access. 5, 25445–25454 (2017).
12. W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu: Edge Computing: Vision and Challenges. IEEE Internet of Things Journal. 3, 637–646 (2016).
13. J. Leskinen: Evaluation Criteria for Future Identity Management. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications. pp. 801–806 (2012).
14. J. Cao, L. Xu, R. Abdallah, W. Shi: EdgeOS_H: A Home Operating System for Internet of Everything. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). pp. 1756–1764 (2017).
15. W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, Y. T. Hou: A Survey on Security, Privacy, and Trust in Mobile Crowdsourcing. IEEE Internet of Things Journal. 5, 2971–2992 (2018).
16. K. Zhang, X. Liang, R. Lu, X. Shen: Sybil Attacks and Their Defenses in the Internet of Things. IEEE Internet of Things Journal. 1, 372–383 (2014).
17. ETSI: Human Factors (HF): User identification solutions in converging networks, (2001).
18. Craig Webster: WebNS: Model for a Peer-to-peer Name Service, (2011).
19. Identity and Access: About - Identity and Access, https://www.identityandaccess.org/about/, (2016).
20. S. Balasubramaniam, G. A. Lewis, E. Morris, S. Simanta, D. B. Smith: Identity management and its impact on federation in a system-of-systems context. In: 2009 3rd Annual IEEE Systems Conference. pp. 179–182 (2009).
21. H. Kaffel-Ben Ayed, H. Boujezza, I. Riabi: An IDMS approach towards privacy and new requirements in IoT. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 429–434 (2017).
22. OpenID: What is OpenID?, https://openid.net/what-is-openid/, (N.D.).

23. A. Tapiador, A. Mendo: A survey on OpenID identifiers. In: 2011 7th International Conference on Next Generation Web Services Practices. pp. 357–362 (2011).
24. Jain, A., Prasad, M.V.N.K.: A novel fingerprint indexing scheme using dynamic clustering. Journal of Reliable Intelligent Environments. 2, 159–171 (2016).
25. F. Jaafar: An Integrated Architecture for IoT Fingerprinting. In: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). pp. 601–602 (2017).
26. Banda, G., Bommakanti, C.K., Mohan, H.: One IoT: an IoT protocol and framework for OEMs to make IoT-enabled devices forward compatible. Journal of Reliable Intelligent Environments. 2, 131–144 (2016).
27. Balakrichenan, S.: Why DNS should be the naming service for Internet of Things?, (2016).
28. Sharma, G., Kalra, S.: A secure remote user authentication scheme for smart cities e-governance applications. Journal of Reliable Intelligent Environments. 3, 177–188 (2017).
29. M. Suguna, R. Anusia, S. M. Shalinie, S. Deepti: Secure identity management in mobile cloud computing. In: 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2). pp. 42–45 (2017).
30. R. Moskowitz, Ed., T. Heer, P. Jokela, T. Henderson: Host Identity Protocol Version 2 (HIPv2), https://www.rfc-editor.org/info/rfc7401, (2015).
31. X. Yang, X. Ji: Host Identity Protocol—Realizing the separation of the location and host identity. In: 2008 International Conference on Information and Automation. pp. 749–752 (2008).
32. Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J.D., Ochoa, M., Tippenhauer, N.O., Elovici, Y.: ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis. In: Proceedings of the Symposium on Applied Computing. pp. 506–509. ACM, Marrakech, Morocco (2017).
33. P. Zhang, H. Sun, Z. Yan: Building up Trusted Identity Management in Mobile Heterogeneous Environment. In: 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications. pp. 873–877 (2011).
34. M. Trnka, T. Cerny: Identity Management of Devices in Internet of Things Environment. In: 2016 6th International Conference on IT Convergence and Security (ICITCS). pp. 1–4 (2016).
35. Gómez-Cárdenas, A., Masip-Bruin, X., Marín-Tordera, E., Kahvazadeh, S., Garcia, J.: A Hash-Based Naming Strategy for the Fog-to-Cloud Computing Paradigm. In: Heras, D.B., Bougé, L., Mencagli, G., Jeannot, E., Sakellariou, R., Badia, R.M., Barbosa, J.G., Ricci, L., Scott, S.L., Lankes, S., and Weidendorfer, J. (eds.) Euro-Par 2017: Parallel Processing Workshops. pp. 316–324. Springer International Publishing (2018).
36. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, (2015).
37. A. Gomez-Cárdenas, X. Masip-Bruin, E. Marin-Tordera, S. Kahvazadeh, J. Garcia: A Resource Identity Management Strategy for Combined Fog-to-Cloud Systems. In: 2018 IEEE 19th International Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM). pp. 01–06 (2018).
38. Sarkar, S., Misra, S.: Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. IET Networks. 5, 23–29 (2016).
39. Farrell, S., Dannewitz, C., Ohlman, B., Kutscher, D., Hallam-Baker, P., Keränen, A.: Naming Things with Hashes. RFC Editor (2013).
40. Bouk, S.H., Ahmed, S.H., Kim, D.: Hierarchical and hash based naming with Compact Trie name management scheme for Vehicular Content Centric Networks. Computer Communications. 71, 73–83 (2015).
41. Savolainen, T., Soininen, J., Silverajan, B.: IPv6 Addressing Strategies for IoT. IEEE Sensors Journal. 13, 3511–3519 (2013).
42. Arduino: Arduino UNO WiFi Rev2, https://store.arduino.cc/arduino-uno-wiFi-rev2, (N.D.).
43. Raspberry Pi Foundation: Raspberry Pi 3 Model B, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, (2016).