INVITED PAPER   *Special Section on Network Resource Control and Management for IoT Services and Applications*

Mobile Edge Computing Empowers Internet of Things*

Nirwan ANSARI[†a)] *and* Xiang SUN[†b)], *Nonmembers*

SUMMARY   In this paper, we propose a Mobile Edge Internet of Things (MEIoT) architecture by leveraging the fiber-wireless access technology, the cloudlet concept, and the software defined networking framework. The MEIoT architecture brings computing and storage resources close to Internet of Things (IoT) devices in order to speed up IoT data sharing and analytics. Specifically, the IoT devices (belonging to the same user) are associated to a specific proxy Virtual Machine (VM) in the nearby cloudlet. The proxy VM stores and analyzes the IoT data (generated by its IoT devices) in real-time. Moreover, we introduce the semantic and social IoT technology in the context of MEIoT to solve the interoperability and inefficient access control problem in the IoT system. In addition, we propose two dynamic proxy VM migration methods to minimize the end-to-end delay between proxy VMs and their IoT devices and to minimize the total on-grid energy consumption of the cloudlets, respectively. Performance of the proposed methods is validated via extensive simulations.
key words: *Internet of Things, mobile edge computing, cloudlet, semantics, social network, green energy*

1. Introduction

Internet of Things (IoTs) is enabling interconnections among a tremendous number of things such that different things can share their observations of the physical world. According to a new Gartner forecast, 26 billion things (excluding PCs, tablets, and smartphones) will be installed in 2020 [1]. Cisco predicted that 50 billion devices will be connected to the Internet by 2020 [2]. These connected things will generate a humongous volume of data, which digitally represent the states of the physical world. However, owing to the resource constrained nature, many IoT devices cannot always guarantee the interconnections, i.e., some IoT devices cannot be reachable owing to their periodical sleep schemes and intermittent wireless connections. Thus, it is important to design an efficient mechanism to facilitate resource constrained IoT devices in sharing their data over the network. Also, the resource constrained IoT devices cannot feasibly conduct complicated data access management, thus sharing IoT data over the network poses serious security challenges, i.e., unautho-

rized users/devices may easily access and misuse the shared IoT data, which may contain personal information. Therefore, designing an efficient access control mechanism tailored for IoT devices is critical to empowering the current IoT system. In addition, only providing interconnections to share raw data among IoT devices is not enough to gain the insight behind the big IoT data; the insight is more valuable for the society as a whole. Thus, it is beneficial to provision the IoT system with a comprehensive cognitive capability such that high-level knowledge can be extracted from the big IoT raw data streams by applying various types of data mining and machine learning methods [3]. The data center infrastructure has been demonstrated to provision resources flexibly and efficiently [4]; meanwhile, various parallel computing architectures and distributed storage frameworks have been designed based on a data center (e.g., MapReduce [5] and Spark [6]). Thus, it is desirable to transmit the big IoT data from the IoT devices to remote data centers via the Internet for further data analysis. Yet, this would place a heavy burden on the network to conduct data aggregation from IoT devices to a centralized data center, and thus exponentially increase the network delay for transmitting big IoT data to remote data centers, especially during the peak time. Note that delay is a key performance metric in provisioning the Quality of Service (QoS) for many IoT applications. For instance, smart grid applications have stringent requirement on latency up to 20 ms; processing automation applications (i.e., monitoring and diagnosing of industrial elements and processes) imposes latency requirements ranging from 50 ms to 100 ms [7].

The Mobile Edge Computing (MEC) concept [8] is essentially bringing the computing and storage capability from remote data centers to the mobile edge in order to reduce the network delay between end devices and computing resources. Specifically, various computing resources, attached to edge routers, wireless access points (WAPs), and smart gateways, are available for nearby mobile devices; thus, these devices can offload their workloads to computing resources at the edge, thus potentially reducing the energy consumption of the devices and accelerating the computing processes [9]. Empowering IoT with MEC can essentially improve the QoS for IoT applications. Basically, IoT applications, which try to obtain the corresponding data from different types of IoT devices and generate high-level knowledge by analyzing the acquired data based on data analytic models, would be deployed at the mobile edge, and thus the data streams generated by the IoT devices would be uploaded to

the IoT applications without traversing the mobile core network. This can significantly alleviate the traffic load in the core network and potentially speed up the IoT applications in processing big IoT data streams.

In this paper, we will design a novel Mobile Edge IoT (MEIoT) architecture by leveraging the Fiber-Wireless (FiWi) access technology, the mobile network, the cloudlet concept, and the Software Defined Networking (SDN) framework to efficiently share and analyze the big IoT data at the mobile edge. The rest of the paper is organized as follows. In Sect. 2, we describe the proposed MEIoT architecture. In Sec. 3, we illustrate the four challenges of the current IoT system and propose four potential solutions tailored for the MEIoT architecture. In Sect. 4, we evaluate the performance of the proposed solutions via simulations. We briefly delineate the future work in Sect.. 5, and present the conclusion in Sect. 6.

2. MEIoT Architecture

In order to facilitate IoT data sharing and analytics, we propose the MEIoT architecture, as shown in Fig. 1. The MEIoT architecture comprises five parts, i.e., multi-interface wireless access network, heterogeneous backhauling, distributed cloudlets, hierarchical structure of a cloudlet, and the SDN based mobile core network. We will next detail these five parts.

2.1 Multi-Interface Wireless Access Network

Various WAPs, such as WiFi access points, static Small Cells (SC) (e.g., pico cell, femto cell, etc.), dynamic SCs (e.g., drone mounted SCs), and Macro BSs (MBSs), have already been deployed in the mobile network and provide high radio coverage and network capacity. Thus, distributed WAPs have the potential to connect all IoT devices whether they are moving or static. Yet, different IoT devices have different communications requirements; that is, some energy-sensitive IoT devices (e.g., smart meters) require very low transmission date rate and some energy-insensitive devices (e.g., surveillance devices and mobile phones) need high-speed transmission to meet their embedded application requirements. The heterogeneous data transmission requirements among IoT devices effectuate different devices to adopt different wireless technologies (e.g., D2D communications, NarrowBand IoT communications, LTE communications, etc.) to share their sensed data. Thus, WAPs are equipped with multiple wireless access interfaces such that they can communicate with IoT devices by applying different wireless technologies.

2.2 Heterogeneous Backhauling

The mobile backhaul is used to carry the traffic from WAPs to the mobile core network [10]. Owing to different requirements (such as low latency, long distance transmission, mobility, and high reliability), various backhaul technologies have been proposed. The current backhaul technologies

can mainly be divided into two categories, i.e., wired and wireless backhaul solutions.

2.2.1 Wired Backhaul Solutions

Wired backhaul provides the advantages of high reliability, high data rate, and high availability. A WAP (such as MBS-1 in Fig. 1) can communicate with the mobile core network based on a wired connection (e.g., xDSL), which traverses an access switch. The access switch is connected to a cloudlet and conducts L2/L3 switching among the WAP, the cloudlet, and the mobile core network. A cloudlet [11], which comprises a number of interconnected Physical Machines (PMs), provides computing and storage resources to IoT devices with low latency.

Passive optical networks (PONs) can potentially provision cloud computing [12], and thus a WAP can also utilize the optical backhaul to achieve extra low communications delay. For example, MBS-2 in Fig. 1 is connected to an Optical Network Unit (ONU), which is further connected to an Optical Line Terminal (OLT) via an optical splitter/combiner. The function of an ONU is to aggregate the traffic from its connected WAPs and communicate with its connected OLT based on the assigned wavelength channels. The function of an OLT is to provide L2/L3 switching between mobile core network and its connected ONUs. Note that there are two different types of ONUs in the MEIoT architecture, traditional ONU and ONU-Cloudlet (ONU-C). Different from traditional ONUs, an ONU-C, which is normally connected to a local cloudlet, can not only relay the traffic between its connected OLT and WAPs but also provide the switching function to enable the local communications between its WAPs and its connected cloudlet [13]. Thus, the communications between the local WAPs and the cloudlet can be offloaded from the OLT and the mobile core network. This can significantly reduce the traffic load of the mobile core network and the OLT.

2.2.2 Wireless Backhaul Solutions

Wireless backhual technologies present the advantages of flexible deployment and low cost. Currently, many SCs (especially for mobile SCs, such as drone mounted SCs [14]) apply the wireless backhaul solutions to facilitate the communications between MBSs and SCs. The wireless backhaul solutions can be divided into two categories: in-band and out-band wireless backhaul.

In-band wireless backhaul means that the wireless bands being applied to the communications between an SC and its mobile users are the same as those being applied to the wireless backhaul between the SC and its MBS [15]. For instance, SC-1 in Fig. 1 uses the same band of f_1 to communicate with both the MBS and the mobile users. The in-band wireless backhaul achieves high frequency utilization, but it requires efficient scheduling to reduce the interference between the wireless backhaul channels and the downlink/uplink channels between an SC and its mobile users.
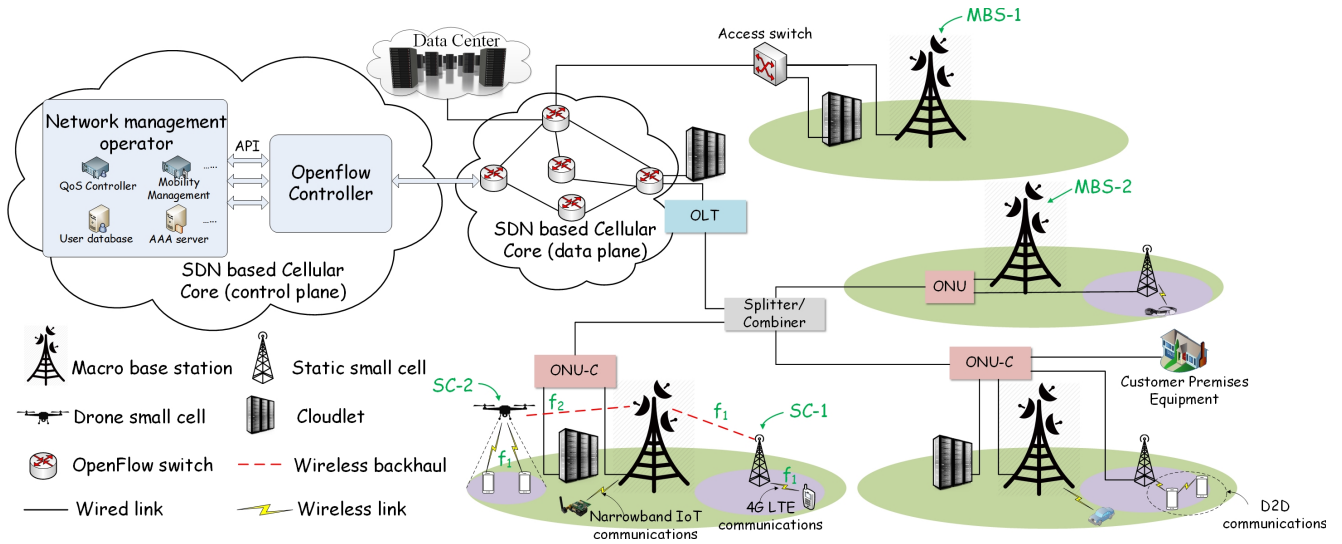
Fig. 1 The MEIoT architecture.

Out-band wireless backhaul implies that the wireless backhaul of an SC applies bands different from those used for the communications between the SC and its mobile users. For instance, SC-2 in Fig. 1 applies f_2 band to communicate with the MBS and uses f_1 band to communicate with its mobile users. Currently, two technologies have been proposed for the out-band wireless backhaul, i.e., millimeter wave (mmWave) [16] and sub-6 GHz wireless backhaul. mmWave offers high capacity and reliability based on line-of-sight communications between an MBS and an SC, and sub-6 GHz wireless backhaul provides high data rate based on non-line-of-sight communications between an MBS and an SC [17].

2.3 Distributed Cloudlets

Cloudlets are deployed at the mobile edge to provision computing and storage resources to IoT devices with low latency. The data generated by various IoT devices can be stored and analyzed in the corresponding cloudlets in real time. The deployment of cloudlets is flexible, i.e., a cloudlet can connect to an ONU-C/access switch such that the IoT devices, which are associated with the WAPs (which are directly connected to the ONU-C/access switch), can utilize the computing and storage resources of the cloudlet without traversing OLTs and the mobile core network. Also, a cloudlet can be deployed at the edge of the mobile core network or connected to an OLT such that more WAPs can share the computing and storage resources in the same cloudlet. Note that geographical distributed cloudlets can share their resources with each other via the mobile core network.

Remote data centers are located at remote sites (which are mostly directly connected to the core network) to provide the scalability and availability of the system. Specifically, the computing and storage capacities of the local cloudlets are limited, and thus they may not have enough capacities to

efficiently store and analyze IoT data streams. Data centers, which supply sufficient and flexible resource provisioning, can be considered as backup units to store and analyze IoT data streams.

It is possible that either mobile network operators or cloud vendors are willing to deploy cloudlets in the network to facilitate their business. For instance, Nokia has established a Multi-access Computing platform to attach a cloudlet to an LTE MBS. Based on the platform, a use case named "connected cars" has been developed, where each local cloudlet analyzes the data at the point of capture and feeds back the insights to the vehicles within the cloudlet's coverage with extremely low latency (less than $20\,ms$) in order to improve road safety [18]. Also, Microsoft has shown great interest in deploying cloudlets at the network edge to reduce the latency of mobile devices in accessing computing resources and improve the battery life of mobile devices. It has also suggested to build an extensive infrastructure of micro data centers (i.e., cloudlets) including 1-10 servers with several terabytes of storage (which may cost $20K-$200K per cloudlet) and place them everywhere [19]. Building a large number of cloudlets at the mobile edge may incur a huge capital expenditure; however, it would potentially generate huge revenue by renting out local computing and storage resources to users and application providers.

Different cloudlet providers may own different cloudlets, and sharing computing and storage resources among cloudlets (which are owned by different providers) can facilitate the resource provisioning to IoT devices. Thus, a fair pricing model needs to be established to achieve resource sharing among different cloudlet providers.

2.4 Hierarchical Structure of a Cloudlet

As shown in Fig. 2, there are two logical layers in a cloudlet, i.e., Proxy VM layer and Application VM layer. Proxy VM
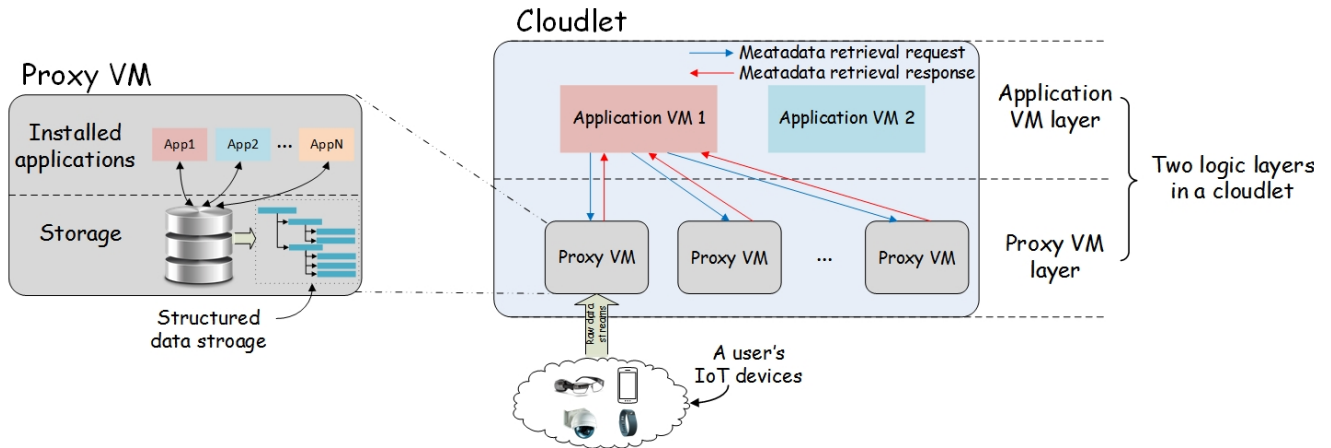
Fig. 2 The hierarchical structure within a cloudlet.

layer comprises a number of the proxy VMs. A proxy VM is considered as a private VM associated with the IoT devices, which belong to the same user[†]. Specifically, IoT devices of a user would be registered to the user's proxy VM. After the registration, these IoT devices would upload their sensed data to the proxy VM periodically or upon requests. The proxy VM converts these raw data into structured data, stores them in the local storage space, conducts the access control policies to protect stored data, and/or pre-processes the structured data upon requests. The application VM layer comprises a number of application VMs, which are deployed by the application providers, to retrieve metadata from proxy VMs, analyze the received data to generate high-level knowledge, and provide the corresponding service to users.

Here, we provide the terrorist detection application to illustrate how the hierarchical cloudlet architecture works. First, if some users are interested in the service provided by the terrorist detection application, they can install the corresponding app (e.g., App1 in Fig. 2) in their proxy VMs. Local devices upload their captured photos/videos to their proxy VMs over time. If the terrorist detecting application VM tries to locate a specific terrorist by conducting face matching over the captured photos/videos, instead of having each proxy VM transmit its captured photos/videos to the application VM, the terrorist detecting application VM would send a metadata retrieval request containing the terrorist's photo to the proxy VMs (which have installed the corresponding app) among all the cloudlets. After receiving the request, the installed app in the proxy VM would retrieve the videos/photos in the local storage, and conduct the face matching algorithm by comparing these videos/photos with the received terrorist's photo. If a match is detected, the proxy VM would respond to the application VM with the related metadata, i.e., the location information and time stamps of the corresponding photos/videos.

The hierarchical cloudlet architecture can also facilitate other applications, such as ParkNet [20] and FaceDate [21].

- ParkNet helps users locate available parking spots in the urban area. Specifically, the proxy VMs collect the sensed data streams from their smart cars, which are considered as IoT devices and are registered to the corresponding proxy VMs. Note that each smart car is equipped with a GPS receiver and a passenger-side-facing ultrasonic rangefinder to generate the location and parking spot occupancy information. Each proxy VM analyzes the information and generates the metadata, which identify the available parking spots, and forwards the metadata to the application VM. The application VM will inform and assign the available parking spots to the smart cars upon requests.

- FaceDate is to find and date nearby people based on their face preference in real-time. Specifically, each user uploads a profile photo and provides basic information (such as date of birth, gender, and a brief write-up) about himself/herself into its proxy VM. In addition, each user uploads a set of preference photos (i.e., the photos of a boy/girl whom she/he wants to date. For instance, if a man wants to date a woman who resembles Marilyn Monroe, he would upload the photos of Marilyn Monroe into its proxy VM) to identify his dream date partner. If a man tries to find a nearby date partner, his proxy VM (i.e., request proxy VM) would send a request, which contains its preference photos, to the application VM. The application VM forwards the request to other proxy VMs, which conduct the face recognition algorithm by comparing the preference photos in the request with their users' profile photos. If the photos are highly matched, the proxy VMs (i.e., response proxy VMs) would respond to the application VM with the metadata (similarity of the photos) as well as the preference photos in these response proxy VMs. The application VM would forward these preference photos (from the response proxy VMs) to the request proxy VM, which conducts the face recognition algorithm by comparing the received preference photos with its pro-

---

[†]A user can be a person who owns various private IoT devices, an entity/company that deploys a set of IoT devices in the area (such as the surveillance cameras), or a group of users who trust each other and share the same proxy VM.

file photo and responds to the application VM with the metadata (similarity of the photos). Finally, the application VM would pick the best matched candidate and enable the chatting accordingly.

The proposed hierarchical cloudlet architecture exhibits the following advantages:

- *Simplified IoT devices*: Each IoT device only needs to sense the environment and upload the sensed data to its proxy VM, which converts the raw data into structured data, stores them into local storage, shares the local structured data by responding to data retrieval requests from other devices, conducts the access control, and pre-processes the structured data by converting them into metadata. Thus, associating IoT devices to their proxy VM is essentially moving most of the functionalities, which are originally executed at the IoT devices, to their proxy VM. This can significantly reduce the energy consumption of IoT devices and speed up the IoT data sharing/analytics process.

- *Accessibility of IoT data*: Cloudlets are connected with each other via wired links, implying that IoT data stored in the cloudlets are always accessible. This resolves the weak accessibility problem in the traditional IoT system, where IoT data are stored in the IoT devices and application VMs cannot retrieve IoT data from IoT devices which may periodically sleep and suffer from intermittent wireless connections.

- *Privacy preserving*: IoT data, which are generated by IoT devices of a user, are basically stored in the user's proxy VM, which is considered as a private VM to facilitate resource isolation and access control. Moreover, the proxy VM can pre-process the IoT data to share metadata (rather than raw data) by removing the user's personal information from raw data. For instance, in the terrorist detection application, each proxy VM only provides the locations and the time stamps of the matched photos/videos rather than the photos/videos.

- *Efficient distributed computing*: Each proxy VM becomes a worker node of an application VM, which acts as a master node to distribute workloads to proxy VMs, aggregate metadata from proxy VMs, and provide services to users. This distributed computing structure can fully utilize the distributed computing resources in the proxy VMs and significantly reduce the traffic load of the network as compared to the current way in which the application VM retrieves the raw data from IoT devices, analyzes them and provides services to users.

2.5 SDN Based Mobile Core Network

Instead of applying the traditional cellular core network architecture, which leads to inefficient, inflexible, and unscalable packet forwarding, the SDN based mobile core network [22]–[28] is adopted in MEIoT. The SDN based mobile core network is essentially decoupling the control plane from the switches, which only run data plane functionalities. The control plane is offloaded to a logical central controller, which transmits the control information (e.g., flow tables) to the OpenFlow switches by applying the OpenFlow protocol [29], monitors the traffic statistics of the network, and provides Application Programming Interfaces (APIs) to network management operators so that different mobile network functionalities, such as mobility management, user authentication, authorization and accounting, network virtualization, and QoS control, can be added, removed, and modified flexibly.

3. Challenges and Solutions in MEIoT

In this section, we will discuss some challenges in realizing real-time IoT data sharing/analytics and provide some potential solutions in the context of MEIoT.

3.1 Challenge 1: Interoperability Problem during IoT Data Sharing

Sharing IoT data among devices is the basic objective in the IoT system. Traditionally, if a client (i.e., an IoT device requests data from another IoT device) tries to obtain the data from a server (i.e., an IoT device generates IoT data), the client would send a request to the server, which would respond to the client with the corresponding data. In the MEIoT architecture, proxy VMs are considered as the gateways to store and manage the IoT data streams from their registered IoT devices. Normally, all the clients would send the requests to the corresponding proxy VMs (rather than the original devices, which generate IoT data) in order to retrieve their IoT data. Proxy VMs receive the raw data streams from their registered IoT devices, convert them into structured data, and respond to the data retrieval requests. Figure 3 just provides one example to illustrate the structured data provided by a temperature sensor.

However, sharing the structured data among devices provides weak interoperability because different proxy VMs and IoT devices may apply different data models and vocab-
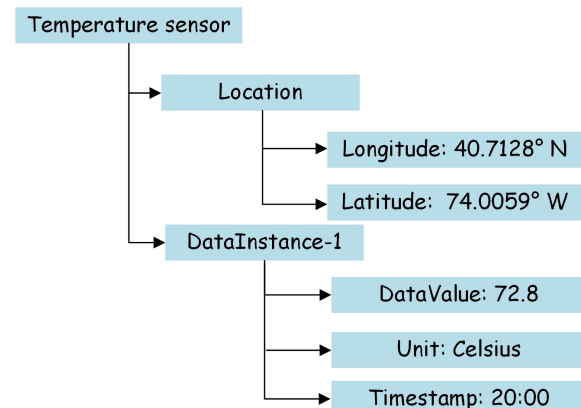

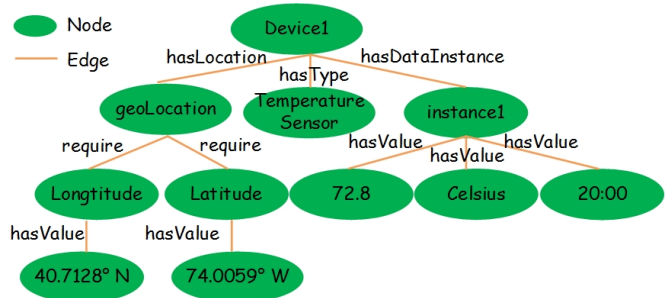**Fig. 3** One example of structured data.

(a) One example of semantic data represented by RDF triples.

(b) One example of semantic data represented by RDF graph.

Fig. 4   Semantic data.

ularies to annotate and structurize IoT data. For example, as shown in Fig. 3, the proxy VM applies "DataValue" to annotate a temperature value[†]. However, a client (who tries to retrieve the current temperature value) uses "TempValue" to annotate a temperature value, and thus the client would request to obtain the "TempValue" of the temperature sensor. Obviously, the proxy VM cannot understand the request from the client. This interoperability problem hampers the IoT data from being shared among different devices.

3.2   Solution 1: Semantic Interoperability in MEIoT

3.2.1   Basic Concept of Semantics

Semantics is a method to provide a common format for annotating data such that the new semantic data can be easily found, shared, reused, and combined by machines [30]. Introducing semantics into the IoT system can improve the interoperability among different IoT devices. In this section, we will illustrate how to leverage semantics in the MEIoT architecture.

All the vocabularies applied in the semantic IoT are defined in different ontologies. An ontology can be considered as a dictionary, which defines the concepts of all the vocabularies used in a specific domain as well as the relationships among different concepts. The ontologies are normally predefined and available for inference and reference. Based on these ontologies, IoT data can be represented as the machine-readable metadata format, such as RDF (Resource Description Framework) triples [31]. An RDF triple contains three components: subject, predicate, and object. Figure 4(a) shows one example of semantic data (corresponding to the structured data shown in Fig. 3) represented by RDF triples. The first two lines define two ontologies (whose URIs are "http://example.org" and "http://rdf.org", respectively) that have been applied by the following semantic data and Line#3–Line#11 are the semantic data represented by RDF triples. For instance, the RDF triple in

Line#3 depicts that the device type is a "TemperatureSensor". Note that "TemperatureSensor" is defined in the ontology, whose URI is "http://example.org", and thus the concept of "TemperatureSensor" can be retrieved from the URI of "http://example.org/TemperatureSensor". Similarly, the concept of "Type" in Line#3 can be obtained from the URI of "http://rdf.org/Type". Therefore, if a device cannot understand the vocabularies applied in the semantic data, it can retrieve the corresponding concepts from related ontologies.

It is worth to note that RDF triples are normally stored as RDF graphs, where the edges represent the "predicate" of RDF triples and the nodes represent "subject" and/or "object" of RDF triples. Figure 4(b) shows a RDF graph related to the semantic data in Fig. 4(a). The reason for applying RDF graphs to store semantic data is to speed up the search over a large volume of semantic data. SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF graphs. The syntax of SPARQL is detailed in [32]. Figure 5 shows an example that a device (which could be a proxy VM, an application VM, or a smart user equipment with semantic capability) sends a query to the proxy VM (which contains the RDF graph shown in Fig. 4(b)) in finding the reading of a temperature device in geolocation <40.7128° N, 74.0059° W> at 20:00[††]. The procedure is specified as follows: 1) The device would send a SPARQL query request to the proxy VM. Note that the device may not apply the same vocabularies used by the proxy VM, and thus the proxy VM may not understand the SPARQL query from the device. For instance, as shown in Fig. 5, "TemperatureMeter" and "geographicalLocation" are not applied by the proxy VM, which uses "TemperatureSensor" and "geoLocation" to annotate the corresponding context of IoT data. 2) After receiving the SPARQL query, the proxy VM would send a concept retrieval request to the remote ontology base[†††] in

---

[†]Normally, a proxy VM would use the same vocabularies that are applied by its registered devices. For example, if the temperature sensor uses "DataValue" to annotate its captured temperature value, the proxy VM would apply the same vocabulary to annotate the temperature value.

[††]Note that a device can broadcast the query to a set of proxy VMs (e.g., all the proxy VMs within a cloudlet) to search for the requested content. Alternatively, before sending the query, a device can first discover a specific proxy VM (that may contain the requested content) by sending a proxy VM discovery request to a Resource Directory (RD) [33], which acts like a DNS server containing the IDs (e.g., URIs or IP addresses) and the context information of proxy VMs. The RD would respond to the device with the ID(s) of the qualified proxy VM(s) accordingly.

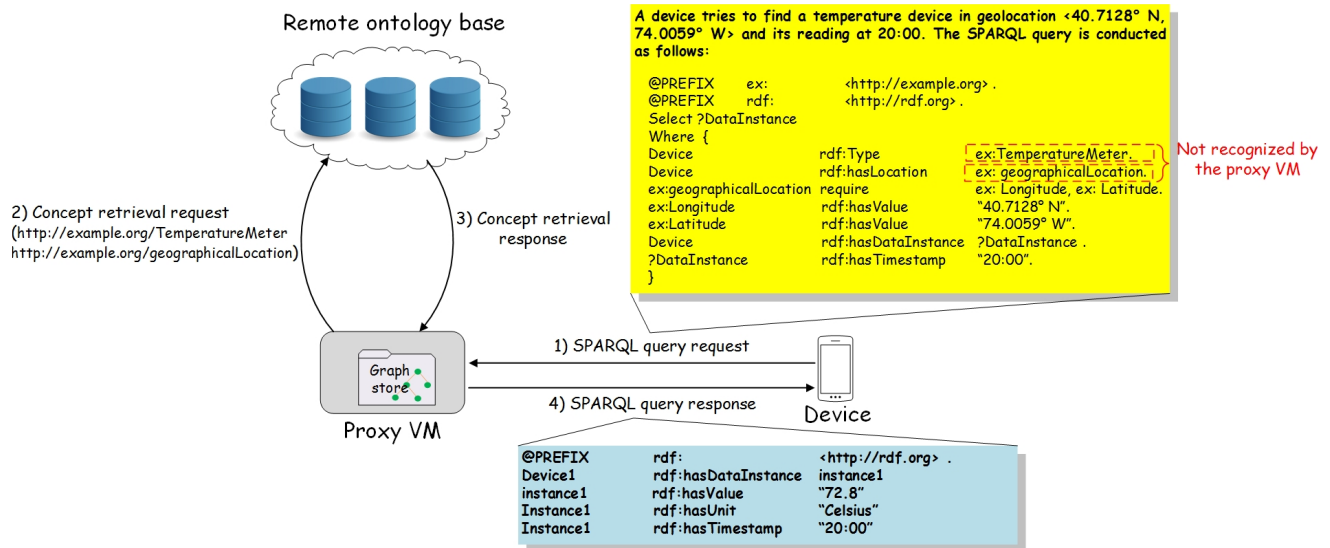[†††]The ontology base is considered as a repository to store on-

Fig. 5 One example to illustrate how a proxy VM conducts SPARQL query.

order to obtain the concepts of those vocabularies. In this example, the proxy VM would sent the concept retrieval request to the URIs of "http://example.org/TemperatureMeter" and "http://example.org/geographicalLocation". 3) The remote ontology base would respond to the proxy VM with the concepts of the vocabularies. In this example, the response message could be "*TemperatureMeter* isSameAs *TemperatureSensor*" and "*geographicalLocation* isSameAs *geoLocation*". 4) After receiving the concept retrieval response, the proxy VM would conduct the search over its local RDF graph store and return back the results in response to the SPARQL query from the device.

3.2.2 Hierarchical Ontology Base in MEIoT

Normally, the ontology base contains all the common vocabularies that are applied in the IoT system and is placed in the remote data center for global access. Proxy VMs need to frequently interact with the ontology base via the mobile core network in order to convert the raw IoT data into semantic data and respond to SPARQL queries. This may significantly increase the traffic load of the mobile core network and the response time in conducting SPARQL queries.

To reduce the traffic load of the mobile core network, we propose to construct a hierarchical ontology base structure in MEIoT by placing redundant ontologies at the mobile network edge. Specifically, there are three levels of ontology bases: proxy VM ontology base, cloudlet ontology base, and global ontology base. A proxy VM ontology base is placed in each proxy VM. It stores the concepts of all the vocabularies that are used by the proxy VM's registered IoT devices. For example, if a temperature sensor is registered to a proxy VM, then the proxy VM should download the concepts of all the vocabularies related to the temperature sensor from

tologies and provides APIs for different devices to access these ontologies.

the global ontology base and store them in its proxy VM ontology base. A cloudlet ontology base is deployed in each cloudlet. It stores the concepts of vocabularies related to some applications, which are mainly determined by the location of the cloudlet. For instance, if a cloudlet is deployed in the residential area, its ontology base should store the concepts of the vocabularies related to the smart home application. A global ontology base contains all the common vocabularies and is normally located in a remote data center.

3.3 Challenge 2: Inefficient Access Control in IoT

IoT data streams normally contain users' personal information (such as users' location traces, health status, etc.), and thus it is important to provide an efficient privacy-preserving solution for the system to identify a device's permission to access the corresponding data. One of the most common Access Control (AC) models that has been applied in the IoT system is the AC list [34], [35], where access rights/policies are listed in IoT devices. For example, if a mobile phone tries to access the data provided by a temperature sensor node, the mobile phone should send a data access request containing its identification and/or contextual information (e.g., IP address and/or geolocation of the mobile phone) to the temperature sensor, which checks its AC policies to see if the requested device has the privileges to manipulate (i.e., retrieve, update, delete, observe, etc.) the corresponding data.

The AC list method is, however, not efficient and scalable. 1) Different devices may use different identification strategies; for instance, a smart TV may use its manufacturer model number, a laptop may apply a product key of its operating system, and a temperature sensor may use its IP/MAC address to identify themselves. It is difficult to build and maintain a complete AC list to enable/disable authorized/unauthorized devices (which may adopt different identification strategies) in accessing the corresponding data.

Fig. 6 One example to illustrate an access control policy in social IoT.

2) An AC list is not automatically generated, i.e., an IoT device owner may need to manually setup/update an AC list. 3) IoT devices are mostly resource constrained, and thus cannot store and maintain a huge volume of AC lists. Hence, it is necessary to design an efficient and scalable AC mechanism to automatically generate AC policies for each device with the consideration of the resource constrained features of IoT devices.

3.4 Solution 2: Efficient Access Control Mechanism Based on Semantic Social IoT

Social IoT (SIoT) [36], [37] is to combine the IoT system with the concept of "human social network" such that IoT devices are capable of building social relationships with other IoT devices without human involvement. It is worth to note that the relationships among IoT devices do not just only rely on the human relationships, but also depend on the contextual information of those IoT devices. Farris et al. [38] summarized the relationships among IoT devices as follows: 1) *parental relationship*: devices produced by the same manufacturer; 2) *co-location relationship*: devices are located in the same places; 3) *co-work relationship*: devices collaborate together to implement the same IoT service; 4) *ownership relationship*: devices belong to the same user; 5) *social relationship*: devices communicate with each other because of the relationships among their owners.

It is thus efficient to set up the AC policies based on the social relationships among different IoT devices. For instance, if the temperature sensor in Bob's smart home can be accessed by the devices, which are owned by Bob's family members, then we can build an AC policy based on the ownership relationship. The corresponding AC policy can be specified as shown in Fig. 6, where the AC policies are represented by RDF triples. Here, Line#1–Line#3 define the applied ontologies; Line#4–Line#6 imply the type, URI, and ownership of the temperature sensor; Line#7–Line#12 depict an access control policy, i.e., policy1, associated to the device. An access control policy should define two components. 1) What operations can/cannot be performed on the device. In this example, Line#11–Line#12 describe the retrieve operation that can be conducted. 2) Who can/cannot perform these operations. In this example, Line#7–Line#10

define the family members of Bob (the owner of the device) that can retrieve the temperature value of the device. Note that applying semantics to annotate access control policies is to improve the interoperability among IoT devices and social networks.

Since most of IoT devices are resource constrained, it is not efficient to enable IoT devices to conduct access control. Instead, the access control functionalities are outsourced to proxy VMs, i.e., proxy VMs would maintain their registered devices' AC policies and handle data access requests from devices to determine whether they have the privileges to access (i.e., retrieve, update, delete, observe, etc.) the corresponding data. To better illustrate the procedure of a proxy VM in conducting access control, we provide a simple scenario in which Alice's mobile phone tries to retrieve the temperature value sensed by the temperature sensor (whose AC policy is specified in Fig. 6) located in Bob's smart home. The whole procedure comprises the following five steps.

(1) A client (i.e., Alice's mobile phone) sends a data access request to the proxy VM (which is associated to the temperature sensor). The request should contain the information on which operation is requested to be performed on which device as well as the contextual information of the client (e.g., the ownership and location of the client). For example, as shown in Fig. 7, the client sends a data access request to the proxy VM in retrieving the data provided by the temperature sensor (which is identified by the URI of "./bob/smart_home/tempSensor"). The request contains the contextual information of the client (e.g., the ownership) to identify itself. Note that the client could be any device (e.g., a mobile phone, a proxy VM, an application VM, etc.) and the data access request could be sent directly to the temperature sensor (rather than the proxy VM), which then relays this request to its proxy VM for conducting access control.

(2) After the proxy VM receives the request, the Policy Decision Maker (PDM) in the proxy VM's authorization module would try to obtain the corresponding AC policies related to the temperature sensor by performing a semantic query over its local graph store. After obtaining the AC policies, PDM would check if the client has the privilege to access the data provided by the temper-
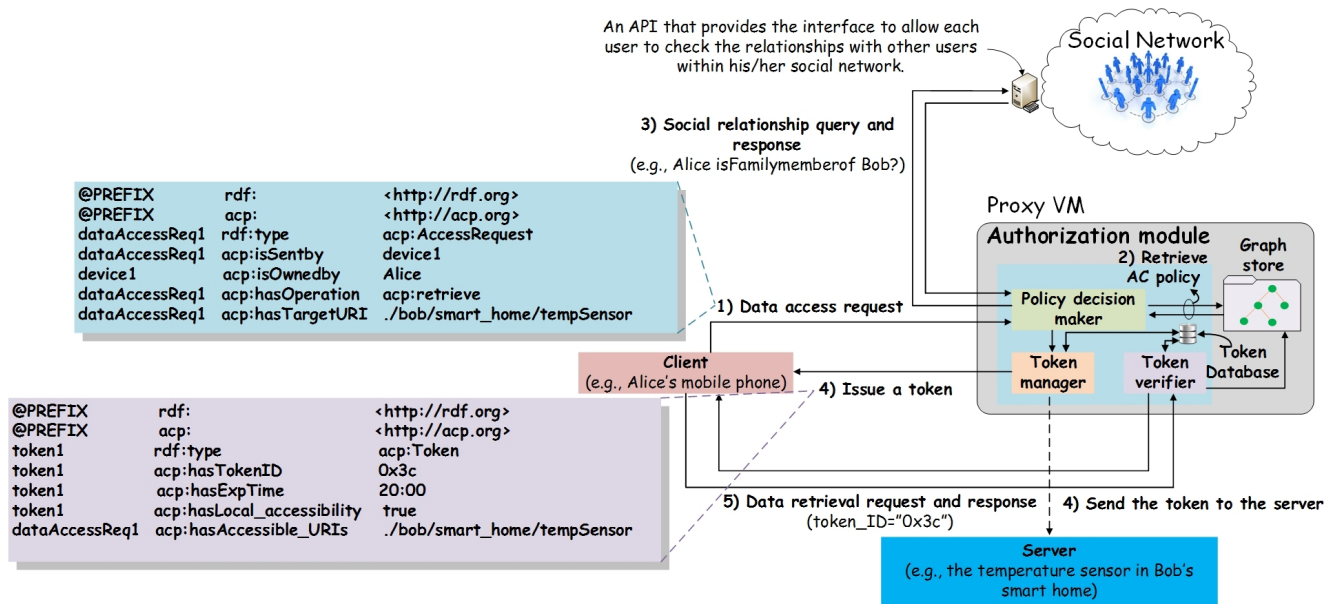
Fig. 7 The procedure of a proxy VM in conducting access control.

ature sensor. PDM could immediately allow/deny the request if the contextual information of the client satisfies/does not satisfy the AC policy. Also, PDM could ask the client to provide more contextual information to identify itself. In this example, the AC policy (as shown in Fig. 7) specifies that the data can be retrieved by the devices owned by Bob's family members. Thus, the proxy VM needs to check the relationship between Alice and Bob.

(3) PDM sends a social relationship query to the social network to identify the social relationship between the owner of the temperature sensor and the owner of the client. In this example, the family membership between Alice and Bob[†] should be confirmed in the Bob's social network. As a response, the social network would return a positive/negative message to confirm/deny the relationship.

(4) If the social relationship is confirmed, PDM would ask the token manager in the proxy VM's authorization module to issue a token to the client. A token is considered as a permission that allows its holder to access the data within a valid time interval [39]. A token should contain the $hasTokenID$, $hasExpTime$, $hasLocal\_accessibility$, and $hasaccessible\_URI$ attributes, where $hasTokenID$ implies the ID of the issued token; $hasExpTime$ indicates the expiration time of this token. After the expiration, the client cannot use this token to access the corresponding data and the identity of the client should be reevaluated by PDM. $hasLocal\_accessibility$ indicates whether the client can access the data in the temperature sensor

(rather than the data stored in the temperature sensor's proxy VM). If $hasLocal\_accessibility = "true"$, the proxy VM should send the token ID to the temperature sensor such that the client can access the data in the temperature sensor by applying the same token ID. $hasaccessible\_URI$ provides a list of URIs of other devices, which share the same access control policy, that could be accessed by the client. For example, if the client can retrieve the data of the temperature sensor, it can also retrieve the data of the humidity sensor in the smart home.

(5) After receiving the token, the client can retrieve the data by sending a data retrieval request containing the corresponding $token\_ID$ to the proxy VM. The token verifier in the proxy VM's authorization module would verify the $token\_ID$ and respond to the client with the temperature value (which is retrieved from the local graph store) if the $token\_ID$ has the privilege to retrieve the data from the server.

The proposed Semantic SIoT provides a flexible and efficient access control mechanism, which is to enforce access control policies based on social relationships among different devices. Also, devices, which outsource their access control functionalities to their proxy VMs, may reduce their energy consumption and accelerate the access control precess.

3.5 Challenge 3: Mobility Problem in MEIoT

In the MEIoT architecture, each IoT device is associated to a specific proxy VM located in the nearby cloudlet. The proxy VM helps its registered IoT devices to share, store, and process their generated data with low End-to-End (E2E) delay. This can substantially reduce the energy consumption
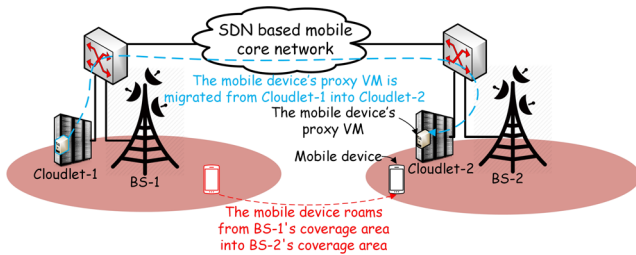

[†]Note that the identities of owners, i.e., "Alice" and "Bob", are encrypted at the device side and will be decrypted and identified at the social network side

Fig. 8 Dynamic proxy VM migration among different cloudlets.

of IoT devices and accelerate the process of IoT data sharing and analysis. However, some of the IoT devices (e.g., smart phones) are mobile and statically placing their proxy VMs in the original cloudlets may not benefit the process of IoT data sharing and analytics, i.e., mobile devices need to upload their generated data streams to their proxy VMs via the mobile core network; this may increase the traffic load of the mobile core network as well as the E2E delay between mobile devices and their proxy VMs. Note that, as mentioned in Sect. 1, delay is a critical factor that affects the QoS of many IoT applications. Thus, it is important to keep the E2E delays between mobile devices and their proxy VMs low in order to satisfy the stringent delay requirement of these IoT applications.

3.6 Solution 3: Latency Aware Proxy VM Migration among Cloudlets in MEIoT

In order to keep the E2E delays between mobile devices and their proxy VMs low, the location proxy VMs can be dynamically changed based on the locations of the mobile devices [24]. As shown in Fig. 8, if a mobile device roams from BS-1's coverage area into BS-2's coverage area, its proxy VM can be migrated to cloudlet-2. Definitely, the E2E delay between the mobile device and its proxy VM in cloudlet-2 is lower than that between the mobile device and its proxy VM in cloudlet-1. However, each cloudlet has the limitation to accommodate the number of proxy VMs, i.e., a cloudlet may not have enough space to host all the proxy VMs of the local mobile devices of the cloudlet. Thus, it is nontrivial to determine the locations of proxy VMs to minimize the total average E2E delay between mobile devices and their proxy VMs while jointly considering the capacity limitation of each cloudlet.

Denote $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ as the set of mobile devices, BSs, and cloudlets, respectively. Denote x_{ik} as the binary variable to indicate whether mobile device i's proxy VM is in cloudlet k (i.e., $x_{ik} = 1$) or not (i.e., $x_{ik} = 0$), where $i \in \mathcal{I}$ and $k \in \mathcal{K}$. Meanwhile, let τ_{jk} be the average E2E delay between BS j and cloudlet k, where $j \in \mathcal{J}$. Note that the value of τ_{jk} can be measured by the SDN controller periodically [40], [41]. We consider the E2E delay between a mobile device and its proxy VM as the E2E delay between the mobile device's associated BS and the cloudlet (which hosts the mobile device's proxy VM)[†]. Moreover, denote ψ_{ij} as the location indicator to imply whether mobile device i is in

BS j's coverage area (i.e., $\psi_{ij} = 1$) or not (i.e., $\psi_{ij} = 0$). Suppose each proxy VM has the same configuration (i.e., the same amount of CPU, memory, and hard disk resource reservations) and denote ϕ_k as the capacity of cloudlet k, i.e., the maximum number of proxy VMs that cloudlet k can accommodate. Then, we formulate the latency aware proxy VM migration problem as follows:

$$\arg \min_{\mathcal{X}} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \psi_{ij} \tau_{jk} x_{ik}, \tag{1}$$

$$s.t. \quad \sum_{k \in \mathcal{K}} x_{ik} = 1, \ \forall i \in \mathcal{I}, \tag{2}$$

$$\sum_{i \in \mathcal{I}} x_{ik} \le \phi_k, \ \forall k \in \mathcal{K}, \tag{3}$$

$$\forall k \in \mathcal{K}, x_{ik} \in \{0, 1\}, \ \forall i \in \mathcal{I}. \tag{4}$$

The objective is to minimize the total E2E delay between all the mobile devices and their proxy VMs, where $\mathcal{X} = \{x_{ik} | i \in \mathcal{I}, k \in \mathcal{K}\}$. Constraint (2) imposes each mobile device's proxy VM to be placed in one cloudlet. Constraint (3) imposes the number of proxy VMs hosted by a cloudlet not to exceed the capacity of the cloudlet. Constraint (4) means x_{ik} is a binary variable. Note that the latency aware proxy VM migration problem is an integer binary programming problem, and thus we can apply the commercial solver, i.e., CLPEX, to solve the problem.

3.7 Challenge 4: Energy Inefficiency in MEIoT

Maintaining a large number of cloudlets incurs a huge operational expenditure to the cloudlet provider by paying an expensive energy bill to the on-grid energy suppliers. Green energy can be leveraged to reduce the operational expenditure. Specifically, each cloudlet is powered by both green energy and on-grid energy. Green energy is generated from renewal resources (e.q., solar, wind, geothermal, etc.) and is considered as a "free" energy supply for the cloudlet provider; on-grid energy is pulled from the smart grid and is considered as a backup energy supply for each cloudlet (i.e., a cloudlet would consume on-grid energy only if there is no residual green energy to power the cloudlet). Detailed descriptions of the green cloudlet system can be found in [26]. It is worth to note that green energy is discouraged to be "banked" since many disadvantages have been proved in storing the superfluous green energy in batteries [42]. Therefore, if green energy is not fully utilized by cloudlets in the

---

[†]The E2E delay between a mobile device and its proxy VM comprises the E2E delay between the mobile device and its associated BS, the E2E delay between the associated BS and the cloudlet (which hosts the mobile device's proxy VM), and the E2E delay within the cloudlet. Yet, optimizing the location of the proxy VM cannot change the E2E delay between the mobile device and the BS; meanwhile, the E2E delay within the cloudlet is negligible. Hence, we consider the E2E delay between a mobile device and its proxy VM as the E2E delay between the mobile device's associated BS and the cloudlet for the rest of the paper.

current time slot, it will be wasted.

Green energy generation exhibits spatial dynamics, i.e., green energy generated in different cloudlets may vary [43]. Also, different cloudlets may host different number of proxy VMs, and thus the energy demand exhibits spatial dynamics as well, i.e., different cloudlets may have different energy demands. The spatial dynamics of green energy generation and energy demands invoke the problem of unbalanced energy gap[†] among cloudlets, thus resulting in inefficient energy utilization and increasing the operational costs for the cloudlet provider.

3.8 Solution 4: Energy Driven Proxy VM Migration among Cloudlets in MEIoT

In order to balance the energy gap among cloudlets, we propose to migrate proxy VMs from the cloudlets with negative energy gap into the cloudlets with positive energy gap [25]. Specifically, the energy demand of cloudlet k equals to the sum of the energy consumption of working PMs[††]. The energy consumption of each working PM is modeled as

$$p^{PM} = \Delta T \left(\rho^s + \alpha \mu \right), \qquad (5)$$

where ΔT is the duration of one time slot, ρ^s is the static power consumption of a PM (the power consumption of a PM when it is in the standby mode), and $\alpha \mu$ is the dynamic power consumption of a PM. Here, μ is the CPU utilization of a PM, which equals to the sum of the CPU utilization of its hosted proxy VMs and α is the power coefficient that maps the CPU utilization into power consumption. If there are N_k working PMs in cloudlet k, we can derive the energy demand of cloudlet k as

$$p_k = \Delta T \left(N_k \rho^s + \alpha \sum_{i \in \mathcal{I}} \mu_i x_{ik} \right), \qquad (6)$$

where $N_k \rho^s$ and $\alpha \sum_{i \in \mathcal{I}} \mu_i x_{ik}$ are total static and dynamic power consumption of all the working PMs in cloudlet k, respectively. Here, μ_i indicates the CPU utilization of mobile device i's proxy VM. Note that the value of N_k is determined by the number of proxy VMs in cloudlet k. If each PM in a cloudlet can host ϵ number of proxy VMs, then $N_k = \left\lceil \frac{\sum_{i \in \mathcal{I}} x_{ik}}{\epsilon} \right\rceil$, where $\lceil \bullet \rceil$ is the ceiling function. By approximating $N_k \approx \frac{\sum_{i \in \mathcal{I}} x_{ik}}{\epsilon}$, we have

$$p_k = \Delta T \sum_{i \in \mathcal{I}} \left(\frac{\rho^s}{\varepsilon} + \alpha \mu_i \right) x_{ik}. \qquad (7)$$

[†]Energy gap of a cloudlet is defined as the difference between the energy demand and the green energy generation of the cloudlet. Positive energy gap means the generated green energy cannot satisfy the energy demand of the cloudlet and negative energy gap implies the cloudlet has superfluous green energy to meet its energy demand.

[††]A PM is said to be a working PM if it hosts at least one proxy VM.

Thus, the energy driven proxy VM migration problem can be formulated as follows:

$$\arg \min_{\mathcal{X}} \sum_{k \in \mathcal{K}} \max \left\{ \Delta T \sum_{i \in \mathcal{I}} \left(\frac{\rho^s}{\epsilon} + \alpha \mu_i \right) x_{ik} - g_k, 0 \right\}, \qquad (8)$$

$$s.t. \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \psi_{ij} \tau_{jk} x_{ik} \leq \gamma, \ \forall i \in \mathcal{I}, \qquad (9)$$

$$Constraints \ (2), (3), and \ (4), \qquad (10)$$

where g_k indicates the total amount of green energy generated by cloudlet k during a time slot and γ is an E2E delay threshold, which defines the maximum E2E delay between a mobile device and its proxy VM. The objective of the problem is to minimize the total on-grid energy consumption and Constraint (9) is to guarantee the E2E delay between every mobile device and its proxy VM to be less than a pre-defined threshold γ. Note that the energy driven proxy VM migration problem is a mixed integer linear programming problem, and thus we also can use CPLEX to solve it.

4. Performance Evaluation

In this section, we will evaluate the performance of Latency Aware proxy VM Migration (LAM) and Energy Aware proxy VM Migration (EAM) as compared to the Static method. The Static method means proxy VMs do not change their locations after the initial deployment.

We set up a network with 5×5 BS-cloudlet combinations. As shown in Fig. 9, each BS is connected to its cloudlet via an access switch. Also, each BS/cloudlet can communicate with other BSs/cloudlets via the SDN based cellular core network and each cloudlet is powered by both green energy and on-grid energy. The amount of green energy generated in each cloudlet is considered to be the same, i.e., $\forall k \in \mathcal{K}$, $g_k = 1000\,W$. The radio coverage area for each BS is $1\,km \times 1\,km$. In order to emulate each mobile device's movement pattern, we apply the user movement trace provided by the EveryWare Lab [44]. The trace provides the users' movement in the road network of Milan in different time slots. We select a $5\,km \times 5\,km$ area of the network and monitor the movement of the qualified users[†††] in different time slots during the monitoring period of six hours. We use the qualified users' movement trace to obtain the values of $\psi = \left\{ \psi_{ij} \mid i \in \mathcal{I}, j \in \mathcal{J} \right\}$ in different time slots. In addition, we assume that the E2E delay between a cloudlet and a BS is proportional to their distance[††††], i.e., $\tau_{jk} = \lambda d_{jk} + \beta$, where d_{jk} is the distance between BS j and cloudlet k, and λ and β are the coefficient and the offset used to map distance into delay, respectively. Moreover, the CPU utilization of each mobile device's proxy VM is randomly selected between 20% and 100%, i.e., $\mu_i = U(0.2, 1), \forall i \in \mathcal{I}$. Each cloudlet

---

[†††]The qualified users are users who only move within the selected $5km \times 5km$ area during the monitoring period.

[††††]The E2E delay between a clouldet and a BS is measured by the SDN controller in the real system.
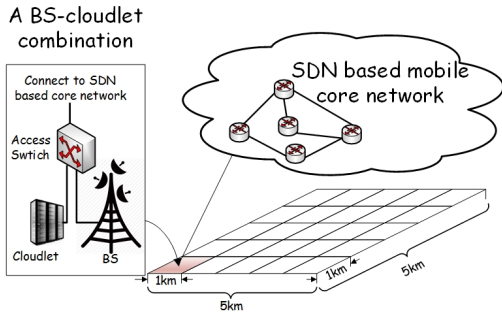
Fig. 9　The network topology.

Table 1　Values and definitions of parameters.

Parameters	Definition	Value
α	power coefficient	$0.2\,W/\%$
β	E2E delay offset	$10\,ms$
λ	E2E delay coefficient	$25\,ms/km$
γ	E2E delay threshold	$40\,ms$
ΔT	the length of a time slot	$0.5\,hour$
ρ^s	static power consumption of a PM	$80\,W$
$|\mathcal{I}|$	total number of qualified mobile devices	632
$|\mathcal{J}|$	total number of BSs	25
$|\mathcal{K}|$	total number of cloudlets	25

contains 5 PMs and each PM can host at most 6 proxy VMs, i.e., $\epsilon = 6$ and $\phi_k = 30$. Other simulation parameters are listed in Table 1.

Figure 10(a) shows the average E2E delay between mobile devices and their proxy VMs, and the average on-grid energy consumption of the network during the monitoring period (i.e., six hours). Obviously, LAM incurs the lowest average E2E delay as compared to EAM and Static because if LAM is applied, once a mobile device roams from one BS into a new BS, its proxy VM may also be migrated into the cloudlet, which has the lowest E2E delay respect to the new BS. Yet, LAM incurs the highest on-grid energy consumption because many mobile devices would move to the same BS' coverage area, and thus their proxy VMs would be migrated to the cloudlet (that is connected to the BS). Consequently, the energy demand of this cloudlet would be significantly increased, and thus the cloudlet's green energy is subsequently drained. This triggers the cloudlet to pull energy from the grid. On the other hand, since energy demands are moving to the cloudlet, other cloudlets may have superfluous green energy, i.e., small value of energy gap. This unbalanced energy gap among different cloudlets increases the total on-grid energy consumption by applying LAM. As a comparison, EAM can balance the energy gap among cloudlets by migrating proxy VMs from the cloudlets with positive energy gap into the cloudlets with negative energy gap. Thus, as shown in Fig. 10(b), although EAM incurs higher average E2E delay than LAM, it saves 39.17% and 35.74% of on-grid energy consumption as compared to LAM and Static, respectively. Therefore, we conclude that there is a tradeoff between minimizing the average E2E delay and minimizing the on-grid energy consumption. Note that

EAM can also guarantee the E2E delay between each proxy VM and its mobile devices to be less than the predefined threshold (i.e., the value of γ), as demonstrated in Fig. 10(b), where the average E2E delay violation rate[†] for EAM during the monitoring period is 0%. Yet, the average E2E delay violation rate of LAM and Static are 6.3% and 33.6%, respectively. Figure 10(b) also shows the maximum E2E delay among all the mobile devices and their proxy VMs during the monitoring period. Interestingly, although LAM incurs the lowest average E2E delay, the maximum E2E delay of LAM is larger than that of EAM.

We further investigate how the amount of green energy generation (i.e., the value of g_k) affects the performance of the three methods. As shown in Fig. 10(c), the average on-grid energy consumption by applying LAM and Static almost linearly decreases as the amount of green energy generation increases. Yet, the decrement of the average on-grid energy consumption by applying EAM is much faster than LAM and Static. This demonstrates that EAM can better utilize green energy by further balancing the energy gaps among cloudlets when more green energy is available. On the other hand, as shown in Fig. 10(d), the average E2E delay incurred by EAM is slightly increasing as the amount of green energy generation increases. This is because more available green energy in each cloudlet may result in more proxy VMs being migrated to the cloudlets (with negative gap), thus incurring long E2E delay. Note that the average E2E delay incurred by LAM and Static does not change as the amount of green energy generation increases.

In addition, we investigate how the amount of traffic load in the mobile core network affects the performance of the three methods. Note that increasing the traffic load in the mobile core network leads to increasing the E2E delay between a cloudlet and a BS. Thus, we use the value of λ to reflect the traffic loads of the mobile core network, i.e., larger value of λ implies heavier traffic load of the mobile core network, and vice versa. As shown in Fig. 10(e), when the value of λ is small, the average E2E delay incurred by the three methods is similar. However, as the value of λ becomes larger, the average E2E delay gap between LAM and EAM/Static becomes larger. Figure 10(f) shows the average on grid energy consumption by varying the value of λ. The energy consumption of LAM and Static does not change since the value of λ does not affect their migration strategies. Yet, the average on-grid energy consumption incurred by EAM is increasing as the value of λ increases because a larger value of λ causes less flexibility to balance the energy gap among cloudlets while satisfying Constraint (9).
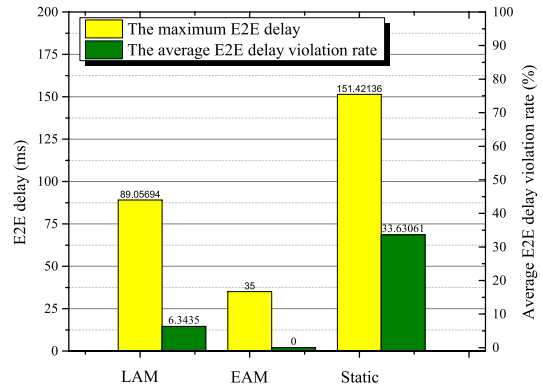
5. Future Works

Migrating proxy VMs among cloudlets can potentially re-

[†] *Average E2E delay violation rate* $= \frac{N}{|\mathcal{K}|}$, where N refers to the number of the mobile devices with E2E delay larger than γ, and $|\mathcal{K}|$ is the total number of mobile devices in the network.
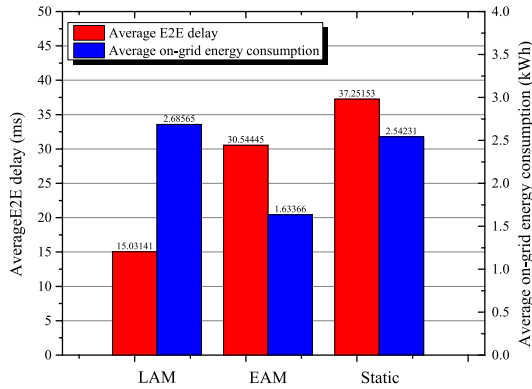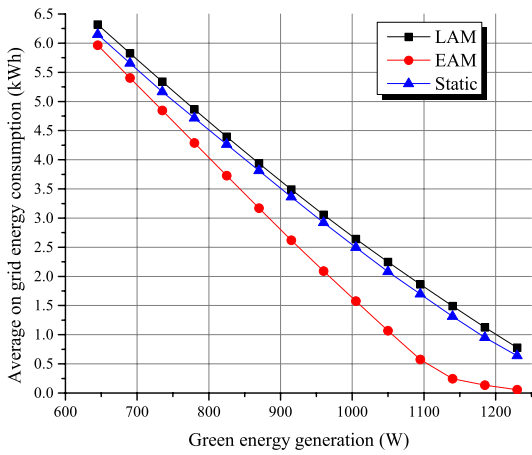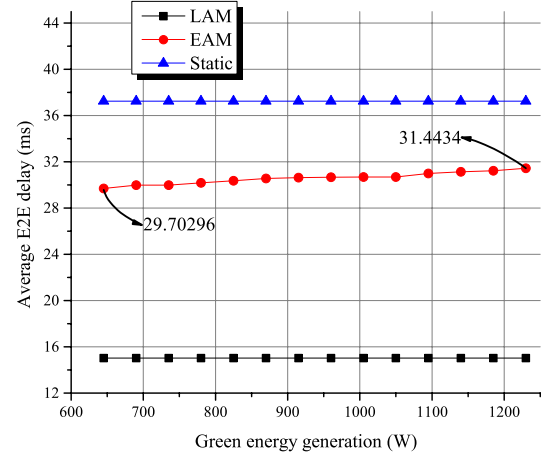
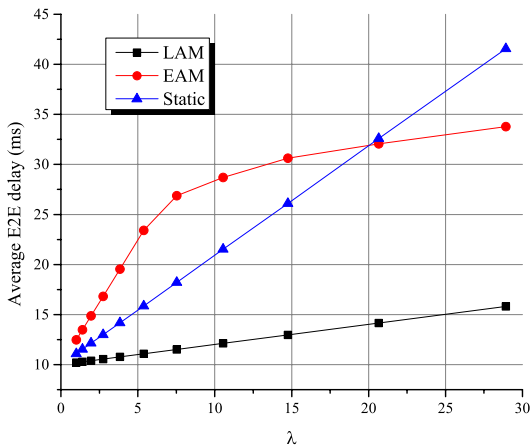(a) Average on-grid energy consumption and average E2E delay.

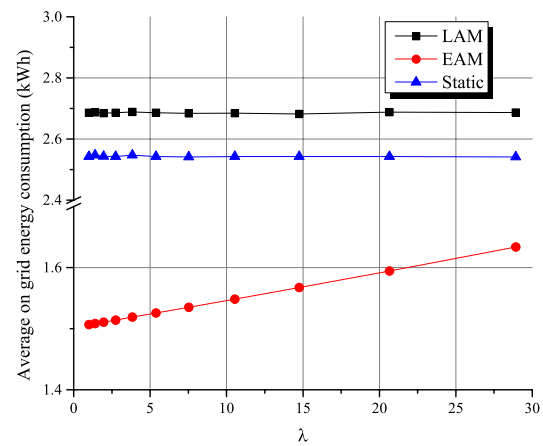(b) The maximum E2E delay and average E2E delay violation rate.

(c) Average on-grid energy consumption by varying green energy generation.

(d) Average E2E delay by varying green energy generation.

(e) Average E2E delay by varying the value of λ.

(f) Average on-grid energy consumption by varying λ.

Fig. 10   Simulation results.

duce the E2E delay between mobile devices and their proxy VMs as well as the on-grid energy consumption of the whole network. However, the following issues need to be considered in order to design a more efficient proxy VM migration strategy:

• Proxy VM decomposition: A proxy VM is associated with

a number of static/mobile IoT devices owned by the same user. Migrating the whole proxy VM when its mobile devices roam away can reduce the E2E delay between mobile devices and its proxy VM, but may increase the E2E delay between static devices and its proxy VM. Thus, before conducting migration, it is beneficial to decompose the proxy VM into two proxy VMs: one proxy VM continues

to serve the static IoT devices, and the other proxy VM migrates among cloudlets as mobile IoT devices roams away [22].

- Migration overheads: Migrating a proxy VM among cloudlets introduces extra overheads. From the networking perspective, the migration overheads incur additional proxy VM migration traffic, which is determined by the size of the proxy VM, the provisioned bandwidth, etc. [24]. From the energy consumption perspective, the migration overheads refer to the amount of energy incurred by the proxy VM migration. Migrating a proxy VM from the source cloudlet to the destination cloudlet may introduce non-negligible energy consumption from both source and destination cloudlets [45]; thus, designing a proxy VM migration strategy without considering the migration energy consumption may significantly increase the total on-grid energy consumption [46]. From the performance of the proxy VM perspective, the migration overheads indicate the performance degradation of the proxy VM [47]. Specifically, conducting data sharing and analytics in a proxy VM consumes CPU, memory, and network resource of the proxy VM; meanwhile, proxy VM migration is considered as an expensive application, which consumes a significant amount of resources in the proxy VM. Thus, proxy VM migration can decelerate the speed of data sharing and analytics, which are conducted during the migration proxy.

- Rightsizing cloudlets: The capacities of different cloudlets (i.e., the values of ϕ_k) are assumed to be the same in the simulations. However, the capacity may vary among the cloudlets to further improve the performance (i.e., the average E2E delay) of the proxy VM migration strategies. For instance, the cloudlets located in the dense areas (such as train stations) may have higher capacities (to host more proxy VMs) than the cloudlets in sparse areas. In addition, the amount of green energy generation (i.e., the value of g_k) may also vary among the cloudlets to reduce the number of proxy VM migrations, i.e., the cloudlets with higher capacities could have more green energy generator units (e.g., larger sized solar panels to produce more green energy) than the cloudlets with lower capacities. Thus, it is beneficial to rightsizing cloudlets by optimizing the capacity and green energy generation for each cloudlet.

In the future, we will design an efficient proxy VM migration strategy by considering both the proxy VM decomposition and migration overheads to maximize the profit of IoT users/network providers. Also, we will implement the proposed semantic social IoT and proxy VM migration in our MEC lab.

6. Conclusion

In this paper, we have introduced the MEIoT architecture to facilitate the IoT data sharing/analysis. We have illustrated four challenges in the current IoT system and proposed the corresponding solutions in the context of MEIoT. We have evaluated the performance of the two proposed proxy VM migration methods, i.e., LAM and EAM, via simulations. In addition, we have elicited future research directions of MEIoT.

References

[1] Newsroom, Gartner. [Online] Available: http://www.gartner.com/new-sroom/id/2844317, date of access: April 6, 2017.

[2] D. Evans, "The Internet of Things: How the Next Evolution of the Internet is Changing Everything," CISCO white paper, pp.1–11, 2011.

[3] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive Internet of things: A new paradigm beyond connection," IEEE Internet Things J., vol.1, no.2, pp.129–143, April 2014.

[4] X. Sun, N. Ansari, and R. Wang, "Optimizing resource utilization of a data center," IEEE Commun. Surveys Tuts., vol.18, no.4, pp.2822–2846, Fourthquarter 2016.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Commun. ACM, vol.51, no.1, pp.107–113, 2008.

[6] Spark 2.1.0. [Online] Available: https://spark.apache.org/docs/2.1.0/, date of access: April 6, 2017.

[7] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S.A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," IEEE Commun. Mag., vol.55, no.2, pp.70–78, Feb. 2017.

[8] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Commun. Surveys Tuts., vol.19, no.3, pp.1628–1656, 2017.

[9] X. Sun and N. Ansari, "Energy-optimized bandwidth allocation strategy for mobile cloud computing in LTE networks," 2015 IEEE Wireless Communications and Networking Conference (WCNC), pp.2120–2125, New Orleans, LA, 2015.

[10] R.N. Clarke, "Expanding mobile wireless capacity: The challenges presented by technology and economics," Telecommun. Policy, vol.38, no.8, pp.693–708, 2014.

[11] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," IEEE Pervasive Comput., vol.8, no.4, pp.14–23, Oct.–Dec. 2009.

[12] M. Taheri and N. Ansari, "A feasible solution to provide cloud computing over optical networks," IEEE Netw., vol.27, no.6, pp.31–35, Nov./Dec. 2013.

[13] B.P. Rimal, D.P. Van, and M. Maier, "Mobile edge computing empowered fiber-wireless access networks in the 5G era," IEEE Commun. Mag., vol.55, no.2, pp.192–200, Feb. 2017.

[14] C. Zhang and W. Zhang, "Spectrum sharing for drone networks," IEEE J. Sel. Areas Commun., vol.35, no.1, pp.136–144, Jan. 2017.

[15] B. Li, D. Zhu, and P. Liang, "Small cell in-band wireless backhaul in massive multiple-input multiple-output systems," 2015 IEEE International Conference on Communications (ICC), pp.1838–1844, London, 2015.

[16] S. Hur, T. Kim, D.J. Love, J.V. Krogmeier, T.A. Thomas, and A. Ghosh, "Millimeter wave beamforming for wireless backhaul and access in small cell networks," IEEE Trans. Commun., vol.61, no.10, pp.4391–4403, Oct. 2013.

[17] G. Zhang, T.Q. S. Quek, M. Kountouris, A. Huang, and H. Shan, "Fundamentals of heterogeneous backhaul design — Analysis and optimization," IEEE Trans. Commun., vol.64, no.2, pp.876–889, Feb. 2016.

[18] Nokia Multi-access Edge Computing (MEC) Solutions. [Online] Available: https://networks.nokia.com/solutions/multi-access-edge-computing, date of access: April 6, 2017.

[19] V. Bahl, "Emergence of Micro DataCenter (Cloudlets/Edges) for Mobile Computing." [Online] Available: https://www.microsoft.com/

en-us/research/wp-content/uploads/2016/11/Micro-Data-Centers-mDCs-for-Mobile-Computing-1.pdf, date of access: April 6, 2017.

[20] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet: Drive-by sensing of roadside parking statistics," Proc. 8th international conference on Mobile systems, applications, and services, pp.123–136, San Francisco, CA, June 2010.

[21] P. Neog, H. Debnath, J. Shan, N. Paiker, N. Gehani, R. Curtmola, X. Ding, and C. Borcea, "FaceDate: A mobile cloud computing app for people matching," Proc. 11th EAI International Conference on Body Area Networks. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Turin, Italy, pp.184–190, Dec. 2016.

[22] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of things," IEEE Commun. Mag., vol.54, no.12, pp.22–29, Dec. 2016.

[23] X. Jin, L.E. Li, L. Vanbever, and J. Rexford, "SoftCell: Scalable and flexible cellular core network architecture," Proc. 9th ACM conference on Emerging networking experiments and technologies, pp.163–174, Santa Barbara, CA, Dec. 2013.

[24] X. Sun and N. Ansari, "PRIMAL: PRofIt maximization avatar placement for mobile edge computing," Proc. IEEE International Conference on Communications (ICC), pp.1–6, Kuala Lumpur, Malaysia, May 2016.

[25] X. Sun, N. Ansari, and Q. Fan, "Green energy aware avatar migration strategy in green cloudlet networks," 7th IEEE International Conference on Cloud Computing Technology and Science (Cloudcom), pp.139–146, Vancouver, Canada, Nov.–Dec. 2015.

[26] X. Sun and N. Ansari, "Green cloudlet network: A distributed green mobile cloud network," IEEE Netw., vol.31, no.1, pp.64–70, Jan./Feb. 2017.

[27] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," IEEE Commun. Lett., vol.21, no.7, pp.1481–1484, 2017.

[28] X. Sun and N. Ansari, "Adaptive avatar handoff in the cloudlet network," IEEE Trans. Cloud Comput., doi: 10.1109/TCC.2017.2701794, early access.

[29] OpenFlow Switch Specification. [Online] Available: https://www.op-ennetworking.org/images/stories/downloads/sdn-resources/onf-spec-ifications/openflow/openflow-switch-v1.5.0.noipr.pdf, date of access: April 6, 2017.

[30] Semantic Web. [Online] Available: https://www.w3.org/standards/se-manticweb/, date of access: April 6, 2017.

[31] Resource Description Framework. [Online] Available: https://www.w-3.org/RDF/, date of access: April 6, 2017.

[32] SPARQL 1.1 Overview. [Online] Available: https://www.w3.org/TR/ sparql11-overview/, date of access: April 6, 2017.

[33] CoRE Resource Directory. [Online] Available: https://tools.ietf.org/pdf/draft-ietf-core-resource-directory-10.pdf, date of access: April 6, 2017.

[34] P.N. Mahalle, B. Anggorojati, N.R. Prasad, and R. Prasad, "Identity authentication and capability based access control (IACAC) for the Internet of things," J. Cyber Security and Mobility, vol.1, no.4, pp.309–348, Chicago, 2013.

[35] Y. Zhang and X. Wu, "Access control in Internet of things: A survey," arXiv preprint arXiv:1610.01065, 2016.

[36] Social Internet of Things. [Online] Available: http://www.social-iot.org/, date of access: April 6, 2017.

[37] L. Atzori, A. Iera and G. Morabito, "From "smart objects" to "social objects": The next evolutionary step of the Internet of things," IEEE Commun. Mag., vol.52, no.1, pp.97–105, Jan. 2014.

[38] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera, and G. Morabito, "Social virtual objects in the edge cloud," IEEE Cloud Comput., vol.2, no.6, pp.20–28, Nov.–Dec. 2015.

[39] O. Bodriagov and S. Buchegger, "Encryption for peer-to-peer social networks," 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Confer-ence on Social Computing, pp.1302–1309, Boston, MA, 2011.

[40] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," IEEE Trans. Veh. Technol., vol.64, no.6, pp.2577–2590, June 2015.

[41] N.L.M. Adrichem, C. Doerr, and F. Kuipers, "Opennetmon: Network monitoring in OpenFlow software-defined networks," 2014 IEEE Network Operations and Management Symposium (NOMS), pp.1–8, Krakow, Poland, May 2014.

[42] Í. Goiri, K. Le, T.D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "GreenHadoop: Leveraging green energy in data-processing frame-works," Proc. 7th ACM european conference on Computer Systems, pp.57–70, Bern, Switzerland, April 2012.

[43] N. Ansari and T. Han, Green Mobile Networks: A Networking Per-spective, Wiley-IEEE Press, ISBN: 978-1-119-12510-5, 2017.

[44] User Movement Simulations Project. Available. [Online]: http://everywarelab.di.unimi.it/lbs-datasim

[45] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?," 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp.514–521, Barcelona, 2013.

[46] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," IEEE Commun. Lett., vol.21, no.7, pp.1601–1604, 2017.

[47] A. Anand, J. Lakshmi, and S.K. Nandy, "Virtual machine placement optimization supporting performance SLAs," 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, pp.298–305, Bristol, 2013.

Nirwan Ansari is Distinguished Professor of Electrical and Computer Engineering at the New Jersey Institute of Technology (NJIT). He has also been a visiting (chair) professor at several universities. Professor Ansari has authored Green Mobile Networks: A Networking Perspective (Wiley-IEEE, 2017) with T. Han, and co-authored two other books. He has also (co-)authored more than 500 technical publications, over 200 in widely cited journals/magazines. He has guest-edited a number of special issues covering various emerging topics in communications and networking. He has served on the editorial/advisory board of over ten journals. His current research focuses on green communications and networking, cloud computing, and various aspects of broadband networks. Professor Ansari was elected to serve in the IEEE Communications Society (ComSoc) Board of Governors as a member-at-large, has chaired ComSoc technical committees, and has been actively organizing numerous IEEE International Conferences/Symposia/Workshops. He has frequently been delivering keynote addresses, distinguished lectures, tutorials, and invited talks. Some of his recognitions include IEEE Fellow, several Excellence in Teaching Awards, some best paper awards, the NCE Excellence in Research Award, the IEEE TCGCC Distinguished Technical Achievement Recognition Award, the ComSoc AHSN TC Technical Recognition Award, the ComSoc AHSN TC Outstanding Service Recognition Award, the NJ Inventors Hall of Fame Inventor of the Year Award, the Thomas Alva Edison Patent Award, Purdue University Outstanding Electrical and Computer Engineer Award, and designation as a COMSOC Distinguished Lecturer. He has also been granted 36 U.S. patents. He received a Ph.D. from Purdue University in 1988, an MSEE from the University of Michigan in 1983, and a BSEE (summa cum laude with a perfect GPA) from NJIT in 1982.

Xiang Sun received a B.E. degree in electronic and information engineering and an M.E. degree in technology of computer applications from Hebei University of Engineering, Hebei, China. He is currently working towards the Ph.D. degree in electrical engineering at the New Jersey Institute of Technology (NJIT), Newark, New Jersey. His research interests include mobile edge computing, big data networking, green edge computing and communications, Internet of Things, and cloud computing.