Virtual Fog: A Virtualization Enabled Fog Computing Framework for Internet of Things

Jianhua Li, *Student Member, IEEE*, Jiong Jin, *Member, IEEE*, Dong Yuan, *Member, IEEE*, and Hongke Zhang, *Senior Member, IEEE*

***Abstract*—The prosperity of Internet of Things (IoT) and the success of rich Cloud services have expedited the emergence of a new computing paradigm called Fog computing, which promotes the processing of data at the proximity of their sources. Complementary to the Cloud, Fog promises to offer many appealing features, such as low latency, low cost, high multitenancy, high scalability, and to consolidate the IoT ecosystem. Although the Fog concept has been widely adopted in many areas, a comprehensive realization has yet been adequately researched. To address all these issues, in this paper, object virtualization is investigated to overcome obstacles resulting from resource constraints on sensory-level nodes while service virtualization is explored to easily create tailored applications for end users. Moreover, network function virtualization is studied to perform the flexibility of network service provisioning. Grounded on object virtualization, network function virtualization and service virtualization, a layered framework that encompasses smart objects, Fog and Cloud is presented to illustrate the realization of virtual Fog along IoT continuum. This proposed virtual Fog framework is applied to a smart living case for verification, then quantitative analysis is conducted to demonstrate the low latency, low operating expense, high multitenancy and scalability, followed by an experimental evaluation to further confirm that delay and jitter can be decreased through virtualization.**

***Index Terms*—Fog computing, Internet of Things (IoT), network function virtualization, object virtualization, service virtualization.**

I. Introduction

THE proliferation of Internet of Things (IoT) brings us an unprecedented picture of future collaboration among things and human beings, where there will be hundreds of things around each person on average [1]. Moreover, it is expected that every "thing" in the cyber world should be identified and connected even without having any direct communication interfaces. Currently, Cloud is assumed as the center to support ubiquitous IoT networks and their interactions [2], although it is far away from things that are generating data at an incredible volume. For example, heterogeneous sensory nodes (sensors, controllers, actuators, etc.) on a driverless car are estimated to generate about 1 GB data per second [3]. As the number of sensors grows, the data deluge grows out of control, leading to massive challenges if all sent and handled in a faraway Cloud. Some common ones include high bandwidth requirement, high latency [4], and high cost. Clearly, processing the vast volume of such data at their proximity can help cope with the above concerns, as compared to moving toward the Cloud, which gives the arising of Fog.

The concept of "Fog" was initially proposed in [5], where its role was defined as an extension of Cloud to support some appealing features, such as low latency, heterogeneity, and mobility. More specifically, Fog might be specified in terms of functionality as Fog edge nodes (FENs), Fog server (FS), and Foglet, where FENs and FS are hardware nodes, and Foglet is the middleware in charge of data exchange, as presented in Fig. 1 [6]. When Fog is employed as a platform for IoT, an FEN accommodates its adjacent smart objects for network access and edge computing, thus sensing, control, and interoperation could be immediately accomplished on the FEN. Different from an FEN dealing with the interaction among smart objects, an FS is focused on the interplay between FENs and Cloud. In this manner, an FS controls, manages, and coordinates FENs at their one-hop proximity, while the Foglet offers the cross-platform capability of monitoring, liaising, and organizing of Fog resources. Overall, Fog distributes computing, control, storage, and networking services along the Cloud-to-Things continuum [7], and also facilitates the collaborations among IoT components.

In addition to proximity, another important advantage of Fog is to support heterogeneity, which is considered as the most distinguished characteristic of the IoT. Often, it contains a variety of subnetworks adopting different communication technologies. For example, ZigBee network is widely used to connect home appliances, while ultrawideband (UWB) network is commonly employed in home entertainment system. Despite homogeneous sensors are deployed at the same place in the above example, the wireless sensor networks are usually application-specific [8], making it almost impossible to share sensor hardware among applications. Such an application-specific feature also tends to cause more issues, including resource underutilization, poor manageability (hard to manage the physical nodes once deployed),

Manuscript received July 31, 2017; revised September 25, 2017 and October 23, 2017; accepted November 9, 2017. Date of publication November 16, 2017; date of current version February 9, 2018. *(Corresponding author: Jiong Jin.)*

J. Li and J. Jin are with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia (e-mail: jianhuali@swin.edu.au; jiongjin@swin.edu.au).

D. Yuan is with the School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2008, Australia (e-mail: dong.yuan@sydney.edu.au).

H. Zhang is with the National Engineering Laboratory on Interconnection Technology for Next Generation Internet, Beijing Jiaotong University, Beijing 100044, China (e-mail: hkzhang@bjtu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2017.2774286

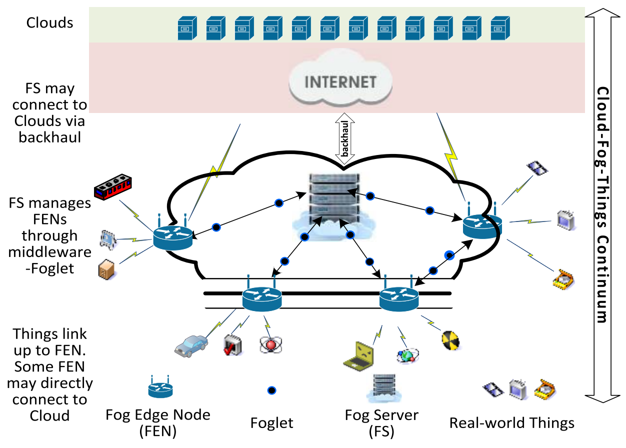
Clouds
INTERNET
FS may connect to Clouds via backhaul
backhaul
FS manages FENs through middleware -Foglet
Things link up to FEN. Some FEN may directly connect to Cloud
Fog Edge Node (FEN)
Foglet
Fog Server (FS)
Real-world Things
Cloud-Fog-Things Continuum

Fig. 1. Internetworking of Cloud and Things.

counter-productivity [9], and protocol inconsistency (IP and sensor-specific protocols, such as ZigBee). Due to the lack of unified networking protocols, the interobject operation among sensors will exaggerate the overhead of data transmission and the complexity of application development. Although using a complex gateway may tackle the inconsistency between IoT subnetworks and the IP-enabled cyber world, it meanwhile causes severe problems (e.g., the existence of single-point-of-failure and the bottleneck of functionalities) because of the inherent complexity and the lack of flexibility and scalability.

As a result of the above features, Fog has been quickly adopted in various business scenarios, and created a wide range of applications and services. Emerging development includes Fog control network, Fog access network, and Fog storage network. The combinations of Fog with IoT, content delivery networks, connected vehicle, and radio access network have also been investigated. For example, Peng *et al.* [10] presented a hybrid of Fog and radio access network (FRAN), where the benefits and challenges associated with Fog have been well elaborated. Based on FRAN, the approach of enabling device-to-device communication in cellular system has been proposed in [11]. Rather than applying the Fog concept to a specific area, this paper is focused on the realization of Fog. From this perspective, a delay-tolerant networking-based Fog framework is proposed to disseminate data at lower cost in [12], and an information-centric networking-based Fog framework is presented in [13]. In industry, Intel gives its Fog reference design overview [14], while Cisco advocates its IOx [15]. Both utilize the field-programmable gate array technology [16] to cast proprietary hardware in a chassis as a Fog node, allowing users to configure and program it after manufacturing. No surprisingly, the manufactures are more interested in the promotion of their products. More relevantly, the OpenFog Consortium releases a milestone technical whiter paper—reference architecture in earlier 2017, which provides high-medium level Fog framework and insightful vision from industry on Fog. Unfortunately, a comprehensive realization of Fog has not been adequately researched.

The motivation of this paper is to fill the gap by proposing a layered Fog framework to better support IoT applications, encompassing all the layers along the Cloud-to-Things continuum through virtualization. In particular, the virtualization refers to the creation of hardware, operating system, storage device, network resource and event processing by abstraction, orchestration and isolation. In our context, the virtualization is further divided into object virtualization [17], network function virtualization [18], and service virtualization [19]. First, to address the protocol inconsistency, object virtualization assists physical sensors in obtaining IP capabilities without affecting its original functionalities. Moreover, the virtual objects (VOs) enhance the capabilities of their resource-limited physical counterparts by taking over some heavy workloads like security. Then, through decoupling network functions from hardware platforms to software instances, network function virtualization maps standard networking services to VOs, thus, flattening the communication process between data consumers and data producers by minimizing latency, improving security and scalability. After that, service virtualization composes the community and Cloud Apps from various vendors to serve local Fog users with high quality of experience (QoE) but at low cost. At last, Foglets are involved to seamless aggregate multiple independent virtual instances, Fog network infrastructures, and software platforms.

Our main contributions are summarized as follows.

1) A complete virtual Fog framework is developed in this paper, which concretely realizes the vision of Fog for IoT by supporting the most of appealing Fog features. It is shown to have superior performance to empower real-time applications, evidenced by both quantitative analysis and experimental evaluation.
2) A seamless integration along Cloud-Fog-Things continuum is achieved through our proposed virtual Fog, where all Fog players from customers, developers, to providers are taken into consideration, and all key elements are encompassed as shown later in Fig. 4.
3) Different from the majorities of other frameworks, merely focused on a set of functionalities, the virtual Fog is also generic enough to make it applicable in various business use cases (e.g., 5G and AI scenarios).

The remainder of this paper is organized as follows: the virtualization of smart objects, network functions, and services are presented in Sections II–IV, respectively. Section V then gives the overall virtual Fog framework, followed by case verification and quantitative analysis to demonstrate the advantages in Section VI. Furthermore, an experimental evaluation is conducted in Section VII to confirm that the latency is indeed minimized on the virtual Fog platform, as compared with Fog and Cloud. Finally, this paper is concluded in Section VIII. For the clarity purpose, Table I lists all the key abbreviations used in this paper.

II. Object Virtualization

To address the interoperability issue for diverse hardware nodes, we need a feasible solution to create a unified software VO on Fog nodes to represent heterogeneous physical entities.

TABLE I
ABBREVIATION TABLE

Abbreviation	Explanation
ANRT	average network response time
APG	application production grid
CapEx	capital expense
FEN	Fog edge node
FPGA	field-programmable gate array
FRAN	Fog and radio access network
FRD	Fog reference design overview
FS	Fog server
IVC	intermediate virtual channel
NFVM	network function virtualization manager
OpEx	operating expense
OVM	object virtualization manager
QoS/QoE	quality of service/quality of experience
RA	reference architecture
SLA	service level agreement
SVM	service virtualization manager
UAI	user application interface
UWB	ultra-wideband
VM	virtual machine
VNF	virtual network function
VO	virtual object
VPN	virtual private network
VSB	virtual service block

In this case, an abstract VO is able to interact with other hosts on the Internet. At the same time, by making full use of either wired or wireless connections available on Fog, a VO can function as a normal adjacent neighbor to a physical object.

A. Virtualization of Sensors

A VO provides a semantic description of the capabilities and features for the associated real-world object. In spite of heterogeneous functions, physical sensors are usually embedded with a processor, a small amount of memory, some external memory, such as flash memory, power supply unit, sensing module, and communication module. All these components can be possibly created and realized as virtual software instances, which are hosted on Fog nodes for actual physical object(s).

As shown in Fig. 2(a), two main streams of object virtualization, namely, OS-level virtualization and hardware-level virtualization, are explored to illustrate VO-hosting solutions. At a glance, a VO can be hosted over established operating systems (e.g., Android, Tiny Core Linux, Cisco IOx, etc.) compliant with market standards. If more advanced performance is required, dedicated hosting platforms can be specially developed by Fog players (including end users, developers, and providers) with featured modules (CPU, hardware interface, memory, etc.). Fig. 2(b) interprets the sensor virtualization framework in more details for a common physical sensor. In the first place, the "Sync Flag" stores version numbers of both collected data from physical sensors and given instructions by the virtual sensor. At this point, the version number is used for the synchronization between virtual and physical sensors. While the "Energy Manager" indicates the battery life on the physical sensor, it also triggers a "wakeup" for

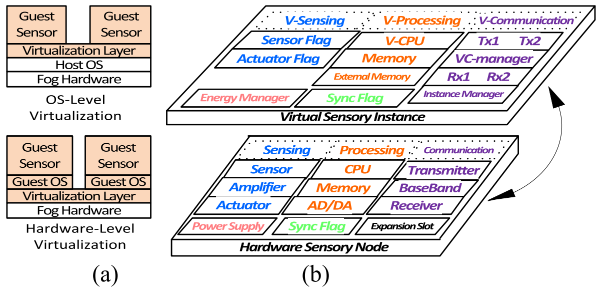

Fig. 2. Object virtualization. (a) Two streams of sensor virtualization. (b) Sensor virtualization framework.

them. Compared with physical sensors, there are V-sensing, V-processing, and V-communication modules in virtual sensors. In the left column, the "Sensor Flag" stores the collected data from physical sensor and the "Actuator Flag" stores the given instruction to the physical sensor. In the middle column, the V-processing is the counterpart of the processing on a physical sensor. Moreover, external memory can be any predefined local or network storage.

The biggest difference is the communication module as shown in the right column of Fig. 2(b). A VO is facilitated with two kinds of virtual transceivers, i.e., Tx1/Rx1 (supporting fully fledged IPv4 and IPv6 dual stack) and Tx2/Rx2 (supporting sensor-specific routing protocols). While a VO uses Tx2/Rx2 to interface with physical sensor(s), it uses Tx1/Rx1 to interact with the IP-enabled cyber world whenever required through state-of-the-art Internet protocols, regardless the physical objects are in the sleep mode or not. As the physical sensors are given more time for sleep, their battery life is much extended, and the running cost is reduced as well. When synchronization happens, the virtual and physical sensors check their version numbers to determine whether data exchange is required. If so, VOs use a variety of sensor-specific protocols to seamlessly link the physical objects via Tx2/Rx2. Besides, unicast and multicast are supported as they are in sensor networks.

Thanks to substantial capabilities of Fog, a VO dynamically discovers adjacent physical nodes and profile their attributes of function, location, and ownership. Moreover, a VO may be offloaded with some workloads that are traditionally conducted on physical sensors. For example, because a VO offers larger storage space, hardware sensors need not cache large amount of data any more. In particular, a VO could serve as a sensor network controller, situated at the proximity of actual sensors via Fog, thereby minimizing the latency between sensing and control. It is also worth mentioning, the overall IoT security might be much enhanced under this setting. One reason is obvious that the powerful but heavy-weight security technology could be now implemented on VOs, another is that the reduced distance between sensor and controller results in simplified neighbor discovery as well as less chances of eavesdropping.

Since multiple VOs are likely to be hosted on a common Fog node, these concurrent VOs must be ideally isolated to avoid task overlaps. In this regard, a Fog node may use virtual machines (VMs) to host VOs in hardware-level virtualization stream, while it may also use software containers [21], e.g., Docker, to host VOs in OS-level virtualization stream. Both VMs and software containers are facilitated with preallocated resources, so that fine-grain isolation is provided between simultaneous tasks. Next, the object virtualization manager is proposed for efficient resource monitoring and allocation.

B. Object Virtualization Manager

While a VO is a building block to base the object virtualization layer, the object virtualization manager (OVM) is responsible to manage and orchestrate virtual and physical entities as depicted in Fig. 4. To start with, an OVM performs life cycle management of corresponding VOs, including creation and termination, discovery and registration, monitoring and coordination, as well as deployment and programming. Meanwhile, an OVM keeps a VO-configuration file that is editable by local and/or remote users. Once a VO is registered to an OVM, it will download its configuration profile, so as to automatically perform self-configuration and self-reconfiguration for service deployment. To further achieve business agility, an OVM is equipped with OV-APIs, through which, Fog players can quickly interact with VOs to manage and configure physical sensors.

An efficient resource management is of paramount importance to achieve superior performance in a complicated IoT ecosystem. To this end, the OVM overlooks the resource allocation of all VOs in the Fog node, guaranteeing that every VO has enough resources to ensure the quality of service (QoS). If the container of a VO is overloaded, the OVM will migrate the VO to a new container with more resources. It can also downsize the containers for the VOs that have fewer tasks. This elastic mechanism provided by the manager improves the resource utility of the Fog nodes in handling heterogeneous sensors. Equally important, an OVM is able to orchestrate VOs among multiple tenants, as well as monitor VOs to mitigate risks carried by unhealthy ones. When an unhealthy log message is generated, an OVM may create another VO to save the business, and use this information for troubleshooting and optimization. Meanwhile, it kills the undesired VO autonomously to free up hardware resources.

If the total physical resources are limited in the Fog node, some algorithms like priority-based resource allocation according to the service level agreement (SLA) could be applied. For each VO, a queuing mechanism is implemented for the incoming tasks. Some scheduling heuristics (e.g., first-come-first-serve, round-robin, etc.) are also applied for processing the tasks with the resources allocated to the container.

It is mutually beneficial to host virtual entities, e.g., VO and OVM, on Fog nodes. Under this circumstance, the virtual entity is able to borrow the power of computing and storage, while the Fog nodes can immediately communicate with the virtual entity, through which, to manage and control physical sensors. Furthermore, the trusted and trustworthiness policy embedded in Fog nodes help maintain security in terms of authentication, authorization, and accounting.

Fig. 3 exemplifies the role of VOs played in the cyber-physical domain. Since a VO can delegate the whole sensor

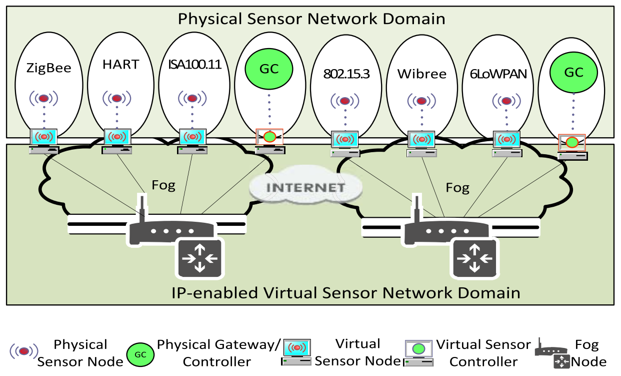
Physical Sensor Network Domain
ZigBee
HART
ISA100.11
GC
802.15.3
Wibree
6LoWPAN
GC
Fog
INTERNET
Fog
IP-enabled Virtual Sensor Network Domain
Physical Sensor Node
Physical Gateway/ Controller
Virtual Sensor Node
Virtual Sensor Controller
Fog Node

Fig. 3. Physical and virtual sensor network.

networks without interference to the actual architecture, the traditional sensor network is thus maintained and controlled as in the physical domain. Moreover, the network architecture can be reconstructed to achieve the desired functionality with multiple VOs representing actual sensors. In this scenario, a large amount of hardware sensors, such as data sinks (temporary data storage), repeaters (signal extender for remote sensor nodes), complex gateways (protocol interpreter, remote communication agent, and routing concentrator) can be omitted in Fog. No matter how much heterogeneity the sensors are, equipping with IEEE 802.15.1 Bluetooth, 802.15.2 UWB and IEEE802.15.4 ZigBee or so on, they are now dynamically managed and shared through corresponding VOs among applications and tenants.

In summary, with the help of Fog, object virtualization aids to address the issues of heterogeneity, interoperability, multitenancy, scalability, counter-productivity, mobility, and protocol inconsistency that are commonly existing in IoT.

III. Network Functions Virtualization

In order to establish the connections among VOs with network service provisioning, Fog faces networking challenges, mainly about the flexibility, scalability, and agility. In this section, network function virtualization is investigated to provide standard network services in a way of decoupling the software implementation of network functions from the location-specific and underlying proprietary hardware. By effectively leveraging virtualization technologies and generic programmable hardware (e.g., commodity servers, switches, routers, etc.), network function virtualization can also potentially reduce the energy consumption, operating expense (OpEx) and capital expense (CapEx).

A. Virtual Network Functions

In the context of network function virtualization, a virtual network function (VNF) is a software instance that contains some numbers or portions of VMs, running different processes for a network function on a commodity hardware platform. While actual network functions are often involving multiple processes, such as signaling, coding, media access control, computing, verification, the VNF must be decomposed into granular reusable components. Each one is designed as

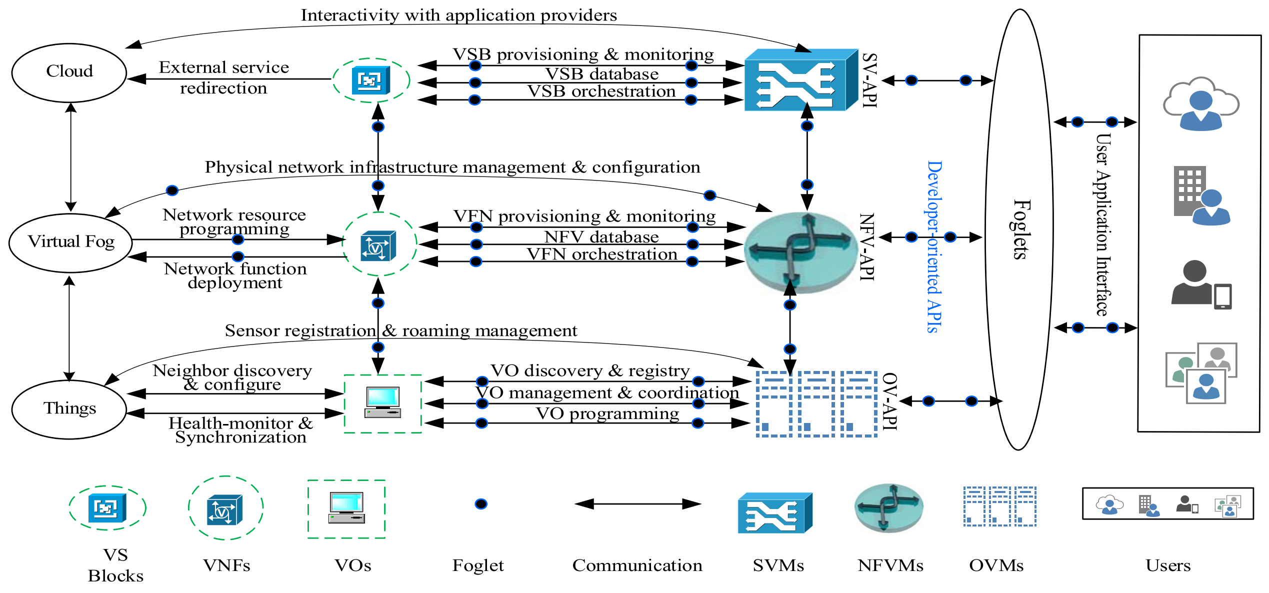
Interactivity with application providers
Cloud
External service redirection
VSB provisioning & monitoring
VSB database
VSB orchestration
SV-API
Physical network infrastructure management & configuration
Virtual Fog
Network resource programming
Network function deployment
VFN provisioning & monitoring
NFV database
VFN orchestration
NFV-API
Developer-oriented APIs
Foglets
User Application Interface
Sensor registration & roaming management
Things
Neighbor discovery & configure
Health-monitor & Synchronization
VO discovery & registry
VO management & coordination
VO programming
OV-API
VS Blocks
VNFs
VOs
Foglet
Communication
SVMs
NFVMs
OVMs
Users

Fig. 4. Virtualization enabled Fog computing framework for IoT.

a standalone and executable micro service, which could be independently deployed, configured, upgraded, and optimized. Then, such components would be packaged into micro containers to make up VNFs. On the other hand, the VNF must be resilient to recover a component failure automatically if needed. It is often instantiated on demand on existing Fog nodes to serve virtual entities. Like a VO, VNF is an integration of micro services that could be dynamically managed through APIs.

By orchestrating multiple VNFs, the network performance should be at least maintained as well as that of dedicate hardware implementation. For example, in the view of facilitating smooth communication between VOs, an immediate virtual channel (IVC) can be established by multiplexing technology, e.g., time division multiplexing, space division multiplexing, and frequency division multiplexing or their combinations. Also, various cutting-edge virtual private network (VPN) technology could be used to establish an IVC. Afterward, a highly comprehensive abstract switch forwards data between senders and receivers via the IVCs, so as to bring down the latency [22]. Furthermore, to reinforce the most demanded storage environments, the storage space could be dynamically organized to form virtual storage area network with more advanced availability and affordability.

In general, there should be no manual intervention required to deliver VNFs. Fueled by the utilization of these VNFs, it is obviously flexible to achieve demanded network functions encompassing configuration, management, QoS clinching, and service provisioning on generic devices.

B. Network Function Virtualization Manager

Referring to Fig. 4, a network function virtualization manager (NFVM) is purposely designed for managing virtualization-specific functions and services in the network function virtualization. As an illustration, an NFVM provides the functions required for the VNF provisioning and monitoring to enforce resiliency and security. Besides, NFVM coordinates the corresponding interplay between the VNFs and physical network infrastructures, upon which the functions run. For the time being, it maintains a database that stores the mapping information of physical and virtual resources, as well as service deployment information, including functions, resources, and workloads. Thanks to the NFVM's capability of managing physical network infrastructure, it conducts various VNFs provisioning on demand over appropriate hardware platforms at a lower cost of CapEx and OpEx. For this reason, Fog operators dynamically tailor network function services on a "pay-as-you-go" basis.

In brief, by decoupling various network functions from fixed vendor-specific equipment, the network function virtualization enables smart objects to be visible to available network resources, that is, standard software instances of functions run on commodity hardware (industry standard servers). This function abstraction and isolation from physical attributes (location, vendor, etc.) therefore help to enhance flexibility, scalability, affordability, mobility, multitenancy, energy efficiency, and business agility.

IV. Service Virtualization

Facilitated by VNFs with flexible network service provisioning, VOs are interconnected to ameliorate the productivity of heterogeneous sensors. Given this, the featured applications to satisfy IoT end users are the last pieces on Fog. By compositing Cloud applications and community Apps from vendors, Fog service providers will select applications on the basis of virtual service instances for local customers. In the following section, service virtualization is studied to organize local application production grid (APG) for featured service, and to provide user-friendly interfaces at endpoints for the best QoE.

Again, referring to Fig. 4, prior to registering and updating their applications to a service virtualization manager (SVM) at the service virtualization layer, each vendor profiles its products per criteria defined by Fog, so that a Fog service provider can collect featured applications for local users. Immediately, similar applications are composed to create virtual service blocks (VSBs) for instantiating specific applications, making many featured services ready. Being executable on any Fog node, a VSB is a micro service fragmented from the applications previously running on large servers. For composing Fog applications, an SVM has a database to store VSB information. Then, the VSBs are systematically orchestrated to form local APGs for service provisioning. On the other hand, the application deployment is promptly conducted once requested, through dynamically coordinating VSBs. When a user starts service execution request, the SVM will initialize an instance to check if it could execute the specific or approximate service with particular requirements. Either "acceptance" or "reject" will be replied to the user for further planning, e.g., SLA negotiation (locating and loading appropriate resource, clinching QoE, and discovering registry), service orchestration (user request interpreting, service finding, and service selecting), service deployment (service state checking, state transferring, job executing or rejection, and termination), service monitoring (user transaction monitoring, process monitoring, provider monitoring, and billing) and service optimization (log tracking, performance reporting, capacity planning, decision making, and modification applying).

Based on the service virtualization APIs associated with SVMs, Fog application developers are able to provide categorized services through a simple user application interface (UAI) that could run at all types of endpoints. For example, tens of video streaming providers may have the same program sought by a customer. The SVM then aggregates those programs with a possibility to further customize to serve one user, while the user is billed on a pay-as-you-go basis, via a standard UAI. In this circumstance, the "single-point-failure" is largely prevented, and the SVM collectively manipulates data, processing, and storage so as to guarantee the customers with the best user experiences. More importantly, as the service virtualization pushes critical application functions to a reusable modular programming environment, it fosters the competition and normalizes the process of running business on Fog for vendors. In short, service virtualization improves both the reliability and the QoE for the end users.

V. Virtualization Enabled Framework

By addressing the raised concerns in the conventional Cloud model, Fog is newly adopted to distribute powerful edge computing to strengthen IoT ecosystem. Grounded on the seamless incorporation of all the virtual entities as explored above, the virtual Fog includes all IoT elements in a unified framework to systematically enable the realization of Fog computing. Meanwhile, the virtual Fog comprehensively empowers the Cloud-to-Things continuum. In this section, the business context of virtual Fog for consolidating IoT is first presented, followed by the detailed framework with Foglets.

A. Context and Benefits

Along the Cloud-Fog-Things continuum, the network can be classified as two distinct areas, i.e., fully fledged IP-enabled domain and non-IP domain. While Clouds are usually sitting in the IP domain, the majority of sensors are in the non-IP domain due to resource constraints, as seen from Fig. 3. Powered by the two types of transceivers, a VO serves the role of interface between these two, via which, the two domains are connected and the protocols are translated bidirectionally. Because Fog provides a seamless coverage and perfect mobility support to things, a VO hosted on Fog is capable of peering with any physical sensor(s), and representing things in the IP domain. Hence, Fog enhances the capabilities of VOs in computing, storage, and networking. In contrast, by delegating some of the workloads to a VO, the physical sensor could be freed to solely focus on data collection (sensing) and action-taking (actuation).

When the VOs are connected for more collaborations, flexible network functions, and services provisioning are indispensable in the dynamic Fog environment. Network function virtualization timely steps in, by providing vendor-independent VNFs. As a variety of VNFs are conducted on different geo-locations and platforms, it reshapes the network from provider-controlled mode to client-controlled mode, leading to the improvement of network efficiency and customer satisfaction. From user experience point of view, service virtualization achieves customizable service provisioning by orchestrating all levels of application resources. As a result, Fog users enjoy personalized services at their endpoints, while Fog service providers are able to create and maintain featured APGs locally. The grid is gradually constructed on VSBs that are virtual instances of aggregate applications.

Altogether, the IoT ecosystem from Cloud to Things is amalgamated by effectively integrating these three layers. Notably, the virtualization helps reduce the delay and jitter. It is because that the transmission overhead is minimized by using the uniform IP protocols, while the proximity among data requesters and data providers are enabled. This is further showcased in Sections VI and VII.

Another open issue is to determine where user application shall be deployed along the Cloud-to-Things continuum. Relying on the nature of business, some tasks are more favorable to be carried out in Fog, while others are more suitable to Cloud. The placement of what tasks should be in Fog and what should go to Cloud is application-specific. The tasks that need huge computing power or storage are usually conducted in Cloud, while others that have stringent latency requirement are often offloaded to Fog tiers. In this regard, Fog players will determine what tasks should be performed on which layers. For example, technical operations like protection and control systems, visualization systems, and real-time data analytics are naturally performed in Fog, while enterprise operations involving long-term strategies are mostly performed in Cloud. Therefore, Fog players need to consider the current network status, including network node capabilities, processor loads, link bandwidth, storage space, fault events, security consideration, cost, latency requirement, etc., to make the decision where the tasks should be allocated in order to maximize the QoE. It is worth mentioning that, this paper is however more about designing a unified framework to enable Fog through virtualization, where the Fog players can quickly deploy and redeploy their applications to meet QoE requirement and standard of SLA.

B. Framework With Foglets

The proposed virtual Fog framework consists of three layers, i.e., object virtualization layer, network function virtualization layer, and service virtualization layer, which have been described previously. Besides these core components, the overarching framework incorporates Cloud, Things, and Fog players, gluing together by Foglets as depicted in Fig. 4. By making good use of resources, Fog players are engaged with their corresponding business in the virtual Fog. In particular, the virtual Fog providers integrate, manage, and configure available resources to offer tailored IoT services on a pay as you go basis, while end users customize IoT services via UAI on their own endpoints. On the other hand, Fog developers could also take advantage of various APIs to simplify network programming (learning, consistency, compatibility, etc.). As has been noted, Foglets are used for the interconnection, service provisioning, programming, and deployment.

More specifically, the Foglet is a light-weight middleware, designed for the interplay among Fog nodes. It is to fulfill the management and orchestration functionalities, to monitor the health of Fog nodes, and to control resources and request services. As it is shown in Fig. 4, Foglets are responsible to perform life-cycle management activities, which includes standing up/down guest OSes and hypervisors, provisioning and terminating service instances of VOs, VNFs and VSBs. They are also used by network function virtualization manager to manage physical network infrastructures. Similar to a Web interface, a UAI might be installed at endpoints for convenient access of the virtual Fog-based IoT system over Foglets. Service could meanwhile be deployed and provisioned through the collective developer-oriented APIs (service virtualization APIs, network function virtualization APIs, and object virtualization APIs). Of course, the most common usage of Foglets is among virtual entities for their interconnection, intercommunication, and interoperation.

Benefit from Foglets for empowering the interactivities among virtual entities and other hardware nodes within Fog environment, a myriad of pervasive and heterogeneous networked things, data, process, and users are globally integrated along the Cloud-Fog-Things continuum via either developer-oriented APIs or UAIs. For instance, an SVM uses Foglets to generate and coordinate VSBs. In the case of provisioning network service between VOs and VSBs, an NFVM relies on Foglets to perform management and configuration of Fog network infrastructures. Therefore, users can order and access their IoT services through Foglets anywhere, anytime, at endpoints in virtual Fog networks. All in all, the three verticals together with Foglet constitute the virtual Fog framework, which indeed realizes the fog computing vision concretely.

VI. Case Verification and Quantitative Analysis

A. Scenario

In [5], we made a significant QoE improvement from Cloud to Fog, where smart living has been conceptualized to smart energy, healthcare, office, protection, entertainment, and surroundings, termed as "EHOPES." In accordance to the study of the benefits, EHOPES is selected to manifest the advantages of the virtual Fog in comparison with Cloud (Internet) and Fog. The same EHOPES applications have been separately conducted on these three platforms, whose characteristics are summarized in Table II.

It is predicted that "by 2020, there will be around 26 smart objects for every human being on Earth." Hence, three settings are assumed, in the scale of EHOPES deployment, namely, light (tens of sensor nodes), medium (100+ sensor nodes), and heavy (hundreds of sensor nodes), whose sensor numbers are presented in Table III. The three settings are applicable to an EHOPES-enabled smart community in 2020. With 20 local streets and 400 families in the IoT enabled area, the EHOPES targeted region locates in a 500×500 meters community, at the average of four people per family, residing on a 20×20 block. The community infrastructure includes four public transport stations, ten shops, two parks, two medical services, and one gym. Facilitated with 40 public IoT service points to accommodate proximate smart objects, the service providers offer dynamically customizable services for community management and inhabitants. ZigBee sensors ($1 or $2 each) and controllers ($50 each) are used in healthcare and surroundings network while Wi-Fi sensors ($3 or $5 each) and controllers ($70 or $100 each) are deployed in office and protection network. Additionally, UWB sensors ($4 each) and controllers ($60 each) are employed in entertainment networks. There are 1, 2, or 3 controllers per network per family according to the numbers of sensor nodes in the autonomous area.

Based on the characteristics shown in Table II, we assume that some repeaters and data sinks are required to link sensors for the internetworking[1] on Cloud, while it is not required in other platforms. Also, more hardware sensors can be omitted on virtual Fog, as compared with the traditional Fog platform.

B. Quantitative Analysis

1) Performance Index and Data Validation: The purpose of quantitative analysis is to illustrate the advantages of our virtual Fog framework with low latency, low cost, high multitenancy, and high scalability. The analysis is conducted based on different settings and system characteristics, at the EHOPES-enabled community, to compare their performance in the terms of the four-aforementioned metrics. To begin with, we define the *network response time* as end-to-end packet latency in seconds, which is measured through UDP-based applications. The OpEx is counted at dollars per head (dweller) averagely within the community. We assume multiple smart objects can be serviced by one IoT service point (in Fog, it refers

[1]According to wireless networking engineering experience, 10% nodes may be repeating nodes. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/33549/spec34.pdf.

TABLE II
EHOPES ON DIFFERENT PLATFORMS

Parameter \ Settings	Cloud based EHOPES	Fog based EHOPES	Virtual Fog based EHOPES
End-to-end Latency	High	Low	Extremely Low
Proximity	Poor	Good	Excellent
CapEx and OpEx	High	Medium	Low
Multitenancy	Poor	Good	Excellent
Real-time Interaction	Not Support	Support	Better Support
Reduced Hardware Nodes	No	Somewhat	Greatly Reduced
Fast Redeployment	Slow	Medium	Fast
Local Backup	Not Support	Support	Better Support
Vendor Lock-in	Yes	Somewhat Yes	No
All-in-one Server	No	Somewhat Yes	Yes
Reduced Wireless Interference	No	Yes	Yes
Easy to Maintain	Poor	Fair	Good
QoE	Poor	Good	Excellent

TABLE III
ASSUMED AMOUNTS OF SMART OBJECTS

Applications \ Settings	Light	Medium	Heavy
Energy generation device per family	1	1.2	1.5
Smart objects for water, electricity and gas per family	40	80	160
Healthcare sensors per head	1	2	4
Healthcare monitor and robot per family	1	1.2	2
Study and work device per family	5	10	20
Security camera per family	2	6	18
Protection robots per family	1	2	4
Video recorder per family	1	2	4
Entertainment devices per family	2	6	18
Wireless detectors per family	10	20	40
Controller per family	1	2	3

to Fog nodes). The more objects per service point holds, the higher the multitenancy it is. Then, we define *scalability* as the ability of a network to expand to meet business needs. Typically for the same number of networked nodes, the more workloads the network copes with, the higher the scalability it is. For a group of identical services, the less hardware nodes are required, the higher the scalability it is. Here, we use the minimum number of required physical sensors per head in supporting EHOPES services to assess the scalability.

It is always critical to harvest the meaningful data in the process of quantitative analysis. In this case, the OPNET Modeler, Open Flow with MININET, and MATLAB are combined in use to collect the required latency data. Particularly, OPNET Modeler is used to simulate both Cloud and Fog platforms, while MININET is employed to simulate the virtual Fog platform. Furthermore, the collected data are verified and validated by open Web tools [23] with the aid of MATLAB.

From the cost perspective, the CapEx refers to the expenditure used to purchase Fog hardware and software of FS, FEN, Foglet, and other supplementary equipment, while the OpEx is the ongoing expenditure for running Fog-based IoT services. Different from traditional network infrastructure that is often invested and maintained by service providers, many Fog network resources could be widely deployed and maintained by common users, including small companies, communities, and individuals, so as to provide Fog services at their proximity. Such resources can then be managed and configured by virtual Fog service providers to empower IoT services for end users, through virtualization technology, e.g., it has already been demonstrated in [24] that a wireless modem router could be converted to an open-flow switch. Generally speaking, the Fog provider mainly integrates these devices, rather than reinvesting all by itself. Therefore, the CapEx and OpEx (purchase, labor, energy, and maintenance) on virtual Fog deployment are much reduced. Beyond that, as a value-added service, the proposed virtual Fog allows to maximize the utilization of hardware IoT components and improve their productivity, by effectively managing and dynamically sharing these resources among users and applications. Overall, the cost of deploying virtual Fog is far less compared to the value created. We will thereafter focus on OpEx per head to evaluate the cost of the virtual Fog-based IoT services. It is beyond the scope of this paper to address the issues such as how to get permissions from customer and what should be permitted to share with other users. In the same way, we also carefully work out the capacity of IoT service point, that is, how many smart objects per service point hold to evaluate the multitenancy. The minimum required number of things to provision the three settings of services are cautiously counted to imply the scalability. Next, we present the results for comparison.

2) Comparison: Fig. 5(a) illustrates the average network response time for EHOPES services that are conducted in the three modes, each with light (L), medium (M), and heavy (H) sensor numbers. The average delay on the Cloud is 3 s, whilst on Fog, it is about 0.8 s, in sharp contrast with the value in virtual Fog mode, which is 0.3 s. The low delay is extremely vital to delay-sensitive or real-time applications. The figure confirms that the virtual Fog is the best platform to support those applications.

As shown in Fig. 5(b), in Cloud mode, the OpEx surges from 100+ to 200+ and 700+ dollars per head from the light, medium to heavy scale. In Fog mode, the cost varies from 100+ to 200+ dollars per head. However, in the virtual Fog mode, the cost is increased from 70s to 90s dollars per head. It is observed that, among the three green bars, the cost significantly drops from 700+ to 90s dollars per head in the heavy settings between the Cloud and the virtual Fog modes. Obviously, virtualization helps to make EHOPES financially more viable.

By using sensor nodes per service access point as an indicator, the capability of multitenancy is depicted in Fig. 5(c). On the Cloud, only a few nodes (less than 10) can be shared among different networks, whereas on the Fog platform, the multitenancy is slightly improved due to the elimination of

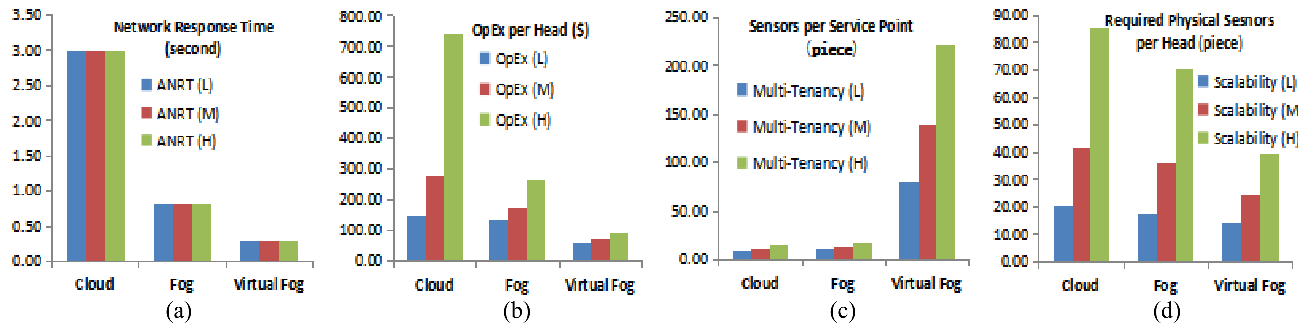

Fig. 5. Quantitative analysis. (a) Network response time. (b) Operating expense. (c) Multi-tenancy. (d) Scalability.

unnecessary sensors, such as repeaters and sinks. By contrast, in virtual Fog mode, a service access node usually accommodates hundreds of VOs. More interestingly, it is possible to further aggregate homogeneous virtual nodes on an FEN. Therefore, the virtual Fog has the best performance in multitenancy.

Since the majority of hardware nodes in IoT ecosystem are physical sensors, it is of paramount importance to dynamically share the hardware sensors across multiple applications for scalability. On the whole, the minimum number of required physical sensors are decreased significantly on different platforms as shown in Fig. 5(d). Particularly, in the heavy settings, it drops about 50% from the Cloud mode to the virtual Fog mode, i.e., dropping from 80 s to 40 s pieces per head for the same services. This reduction is achieved through sharing the actual sensors. To conclude, the virtual Fog has distinctive advantages to cope with latency, OpEx, multitenancy, and scalability.

VII. Experimental Evaluation

As mentioned in Section V-A, virtualization indeed bring down the end-to-end delay between VOs. The experiment is designed to verify the delay performance of virtual Fog. Assuming that the protocol inconsistency and the proximity of VO-physical objects have been tackled by object virtualization, the focus is on the proximity of inter-VO and flexible network service provisioning in virtual Fog. In more details, dynamic multipoint VPN technique is employed to simulate IVCs among FENs and FSs. Based on the realization of the proximity between sensing and processing, the protocol consistency between virtual entities, and the dedicate IVC channel, the virtual Fog has outstanding performance of latency. To evaluate, round trip time, network response time, and application response time are investigated on various platforms.

Unlike the network response time that is the end-to-end delay between Fog nodes in the virtual Fog environment, the *application response time* is considered as the time an application takes to respond to user requests that involves the transaction time of user program. It is approximately equal to two-way network response time plus application transaction time. Thanks to NMAP, TCP ping is used to simulate the value, which contains connection establishment time, ICMP processing time and connection termination time. Likewise,

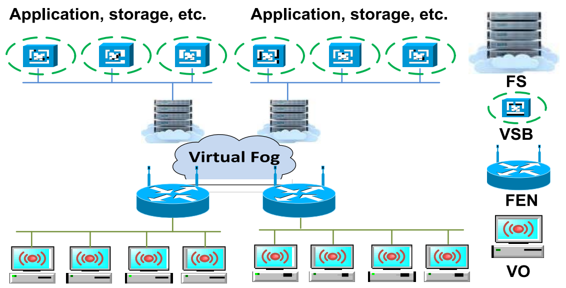

Fig. 6. Topology.

UDP ping is used to collect latency data to calculate network response time, while ICMP ping is to measure round trip time. All the latency evaluation packets are captured in Wireshark, so that their values could be worked out accordingly.

A. Simulation Environment

The simulation has been implemented in three different settings, namely, Setting-1: No IVC (neither IVC nor inter-VO proximity, Cloud is the platform); Setting-2: IVC-FS (using IVC between Fog nodes, but no inter-VO proximity, Fog is the platform); and Setting-3: IVC-FEN (IVC with inter-VO proximity, virtual Fog is the platform).

1) Topology and Device Used in the Simulation: The topology is shown in Fig. 6. The simulation has been conducted in GNS3[2] and the router images are Cisco 7204[3] IOS image. We use 13 routers to simulate the Cloud, and four routers to simulate the FSs and FENs. The virtual sensors are emulated by Virtual PC.[4]

2) Fog Traffic Emulation: It is assumed that 30% of the bandwidth are reserved for the traffic of network control and other purposes. Besides, IP, TCP, and UDP packets are generated by Ostinato,[5] which is an open-source software common for network professionals.

B. Simulation Results

Fig. 7(a) shows the round-trip time on the three settings. In Setting-1, the delays vary from 49 to 61 with an average

[2][Online]. Available: https://www.gns3.com/software

[3][Online]. Available: http://www.cisco.com/c/en/us/products/routers/7204vxr-router/index.html

[4]The virtual PC is a component integrated with GNS3.

[5][Online]. Available: http://ostinato.org/

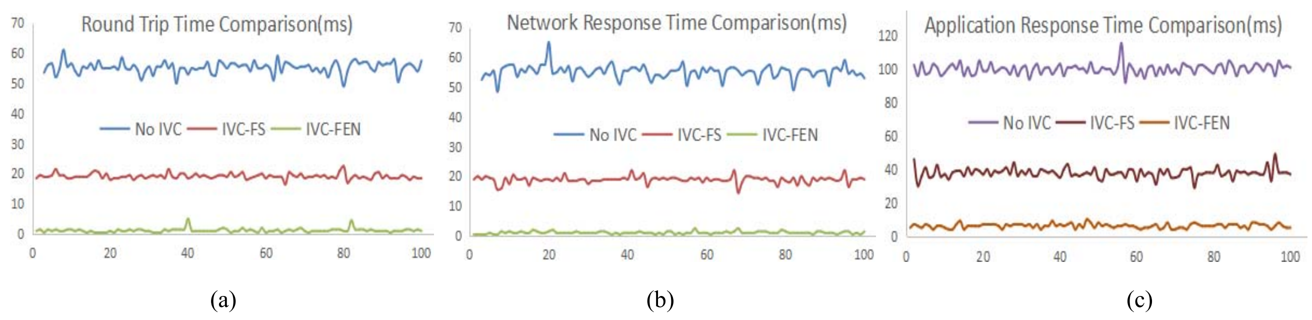

Fig. 7. Comparison of the three settings. (a) Round trip time. (b) Network response time. (c) Application response time.

at 55.3 ms and the packet jitter is 12 ms. In addition, 2% of packets are lost in this setting. As a comparison in Setting-2, the average delay is significantly dropping to 19.2 and the jitter is 6 ms. In Setting-3, the average delay is further down to 1.2 ms. It demonstrates that Fog has better performance than Cloud, whereas the virtual Fog is the best.

The network response time of the three settings is illustrated in Fig. 7(b). In Setting-1, the maximum delay is above 65 while minimum delay is about 48.5 with an average at 55 and the jitter is 16.5 ms. The random losses of packets imply the instantaneous network response time is too long to deliver the packets. Compared with Setting-1, no packet is lost on the other two. The average delay and jitter is 18.9 ms and 1.1 ms in Setting-2 and Setting-3, respectively.

The difference of application response time is shown in Fig. 7(c). In Setting-1, the instantaneous delays shift between 92 and 115.5 with an average of 92 ms. In sharp contrast, the delay in Setting-2 differs from 49 to 29 with an average of 37.8 ms. Different from Setting-1 and Setting-2, packet loss is not seen in Setting-3 and the average delay is further down to 6.1 ms. Again, virtual Fog has the best performance in terms of application response time.

Given the above observation, although Fog generally performs much better than Cloud, obviously, virtual Fog has the best performance in all aspects of round trip time, network, and application response time among these three platforms. Especially, the delay and jitter variations reveal that virtualization technology does reduce the end-to-end latency by tackling heterogeneity, proximity, and protocol inconsistency. Moreover, data are delivered without any loss on virtual Fog, which is extremely important to some data-critical applications.

C. Challenges and Open Issues

Though the virtual Fog framework has superior performance in comparison with Cloud or conventional Fog platforms, there are still some open issues worth future investigation, especially the challenges of resource procurement and utilization.

1) Resource Procurement: Along the IoT continuum, Fog resources are often spatially separated at various locations and owned by distributed owners. In order to locate, secure and orchestrate such resources prior to deployment of their applications, the providers should create a unified platform for various Fog players to share the common infrastructure. Virtual Fog is able to perform dynamic monitoring and pooling of unused resources via autonomous discovery and registration, however, those resources are heterogeneous and pervasive. Therefore, large-scale computational-efficient procurement of distributed resources is still required.

2) Resource Utilization: Fog nodes could use technologies like software containers to run multiple micro services on appropriate nodes. Nevertheless, how to match the variety of IoT applications with heterogeneous resources is a nontrivial task. Moreover, scheduling and synchronization of those services on resource-constrained devices with low overhead pose another key challenge.

VIII. Conclusion

With the surging development along the emergent IoT spectrum, a comprehensive virtualization enabled Fog framework is proposed in this paper. It is strongly desired by real-time applications as they seek to push powerful control and storage closer to smart objects and end users. Particularly, the three layers of virtual Fog seamlessly integrates the two distinct IoT domains, while it tackles immense challenges widely identified within IoT. This framework is simple and scalable, capable of accommodating heterogeneous objects to be dynamically managed and shared among different applications and tenants. Other aspects, including interoperability, multitenancy, flexibility, low-latency, affordability, energy efficiency, easiness to order service, and business agility are also addressed. A case verification and quantitative analysis is applied to demonstrate some of the aforementioned benefits, while the experimental evaluation has further confirmed that a use case based on our virtual Fog can mitigate delay and jitter without data loss. In conclusion, this framework provides an appealing solution for the emerging IoT ecosystem, by taking the advantages of both faraway service-rich Cloud and neighboring resource-constraint things. Some research challenges and open issues have also been pointed out as future research directions.

References

[1] D. Evans, "The Internet of Things how the next evolution of the Internet is changing everything," CISCO, San Jose, CA, USA, White Paper, pp. 1–11, Apr. 2011.

[2] J. Gubbi, R. Buyya, S. Manusic, and M. Palaniswarmi, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[3] A. D. Angelica. *Google's Self-Driving Car Gathers Nearly 1 GB/Sec.* Accessed: Dec. 6, 2016. [Online]. Available: http://www.kurzweilai.net/googles-self-driving-car-gathers-nearly-1-gbsec

[4] J. Li *et al.*, "Fog-based latency estimation for the Internet of Things," in *Proc. Telecommun. Netw. Appl. Conf. (ITNAC)*, Melbourne, VIC, Australia, Nov. 2017, pp. 218–223.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC Workshop Mobile Cloud Comput.*, Helsinki, Finland, Aug. 2012, pp. 13–16.

[6] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner, "EHOPES: Data-centered Fog platform for smart living," in *Proc. Telecommun. Netw. Appl. Conf. (ITNAC)*, Sydney, NSW, Australia, Nov. 2015, pp. 308–313.

[7] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[8] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[9] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[10] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul./Aug. 2016.

[11] S. Yan, M. Peng, M. A. Abana, and W. Wang, "An evolutionary game for user access mode selection in Fog radio access networks," *IEEE Access*, vol. 5, pp. 2200–2210, Jan. 2017.

[12] L. Gao, T. H. Luan, S. Yu, W. Zhou, and B. Liu, "FogRoute: DTN-based data dissemination model in fog computing," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 225–235, Feb. 2016.

[13] D. Nguyen, Z. Shen, J. Jin, and A. Tagami, "ICN-Fog: An information-centric fog-to-fog architecture for data communications," in *Proc. IEEE Glob. Commun. Conf. (Globecom)*, Singapore, Dec. 2017.

[14] *Intel's Fog Reference Design Overview*. Accessed: Jun. 2, 2017. [Online]. Available: http://www.intel.com/content/www/us/en/internet-of-things/fog-reference-design-overview.html

[15] *IOx and Fog Applications.* Accessed: Jun. 2, 2017. [Online]. Available: http://www.cisco.com/c/en/us/ solutions/internet-of-things/iot-fog- applications.html

[16] C. Byers and R. Swanson, "OpenFog reference architecture for fog computing," OpenFog Consortium, Fremont, CA, USA, White Paper, pp. 1–162, Feb. 2017.

[17] C. Sarkar *et al.*, "DIAT: A scalable distributed architecture for IoT," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 230–239, Jun. 2015.

[18] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2015.

[19] H. Ko, J. Jin, and S. L. Keoh, "Secure service virtualization in IoT by dynamic service dependency verification," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1006–1014, Dec. 2016.

[20] S. Jia, Y. Ai, Z. Zhao, M. Peng, and C. Hu, "Hierarchical content caching in fog radio access networks: Ergodic rate and transmit latency," *China Commun.*, vol. 13, no. 12, pp. 1–14, Dec. 2016.

[21] M. H. Syed and B. F. Eduardo, "The software container pattern," in *Proc. 22nd Conf. Pattern Languages Programs*, Pittsburgh, PA, USA, Oct. 2015, Art. no. 15.

[22] L. L. Lu, D.-P. K. Hsing, B.-C. Cheng, and T.-H. Wu, "Content-aware application switch and methods thereof," U.S. Patent 6944678, 2005.

[23] M. Leonhard. *CloudPing.info*. Accessed: May 29, 2017. [Online]. Available: http://www.cloudping.info/

[24] Y. Yiakoumis. *Pantou: Openflow1.0 for Openwrt*. Accessed: May 31, 2017. [Online]. Available: http://archive.openflow.org/wk/index.php/Pantou:OpenFlow1.0_for OpenWRT

Jianhua Li (S'15) completed the Bachelor of Computer Science degree from Shandong University, Jinan, China, and Beijing Jiaotong University, Beijing, China. He is currently pursuing the Ph.D. degree at the Swinburne University of Technology, Melbourne, VIC, Australia.

He started his career as a Network Engineer at China Telecom, Ji'nan, and has grown up with Internet development of China. He has been a CISCO Networking Academy Teacher in Melbourne since 2009 with multiple industry certificates. His current research interests include fog computing, Internet of Things, QoE, and cyber security.

Jiong Jin (M'11) received the B.E. degree (First Class Hons.) in computer engineering from Nanyang Technological University, Singapore, in 2006, and Ph.D. degree in electrical and electronic engineering from The University of Melbourne, Melbourne, VIC, Australia, in 2011.

He is currently a Senior Lecturer with the School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne. Prior to it, he was a Research Fellow with the Department of Electrical and Electronic Engineering, The University of Melbourne, from 2011 to 2013. His current research interests include network design and optimization, nonlinear systems and sliding mode control, networked robotics, Internet of Things, cyber-physical systems and applications in smart grids and smart cities.

Dong Yuan (M'12) received the B.E. and M.E. degrees from Shandong University, Jinan, China, in 2005 and 2008, respectively, and the Ph.D. degree from the Swinburne University of Technology, Melbourne, VIC, Australia, in 2012.

He is a Lecturer with the School of Electrical and Information Engineering, University of Sydney, Sydney, NSW, Australia. His current research interests include cloud computing, data management in parallel and distributed systems, scheduling and resource management, Internet of Things, business process management, and workflow systems.

Hongke Zhang (M'13–SM'16) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively.

From 1992 to 1994, he was a Post-Doctoral Researcher with Beijing Jiaotong University, Beijing, China, where he is currently a Professor with the School of Electronic and Information Engineering and the Director of the National Engineering Laboratory on Next Generation Internet Technologies. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He has authored over ten books and is the holder of over 70 patents.

Dr. Zhang is the Chief Scientist of the National Basic Research Program of China (973 Program) and has also served on the Editorial Boards of several international journals.