Edge Computing: Vision and Challenges

Weisong Shi, *Fellow, IEEE*, Jie Cao, *Student Member, IEEE*, Quan Zhang, *Student Member, IEEE*, Youhuizi Li, and Lanyu Xu

***Abstract*—The proliferation of Internet of Things (IoT) and the success of rich cloud services have pushed the horizon of a new computing paradigm, edge computing, which calls for processing the data at the edge of the network. Edge computing has the potential to address the concerns of response time requirement, battery life constraint, bandwidth cost saving, as well as data safety and privacy. In this paper, we introduce the definition of edge computing, followed by several case studies, ranging from cloud offloading to smart home and city, as well as collaborative edge to materialize the concept of edge computing. Finally, we present several challenges and opportunities in the field of edge computing, and hope this paper will gain attention from the community and inspire more research in this direction.**

***Index Terms*—Edge computing, Internet of Things (IoT), smart home and city.**

I. Introduction

CLOUD computing has tremendously changed the way we live, work, and study since its inception around 2005 [1]. For example, software as a service (SaaS) instances, such as Google Apps, Twitter, Facebook, and Flickr, have been widely used in our daily life. Moreover, scalable infrastructures as well as processing engines developed to support cloud service are also significantly influencing the way of running business, for instance, Google File System [2], MapReduce [3], Apache Hadoop [4], Apache Spark [5], and so on.

Internet of Things (IoT) was first introduced to the community in 1999 for supply chain management [6], and then the concept of "making a computer sense information without the aid of human intervention" was widely adapted to other fields such as healthcare, home, environment, and transports [7], [8]. Now with IoT, we will arrive in the post-cloud era, where there will be a large quality of data generated by things that are immersed in our daily life, and a lot of applications will also be deployed at the edge to consume these data. By 2019, data produced by people, machines, and things will reach 500 zettabytes, as estimated by Cisco Global Cloud Index, however, the global data center IP traffic will only reach 10.4 zettabytes by that time [9]. By 2019, 45% of IoT-created data will be stored, processed, analyzed, and acted upon close to, or at the edge of, the network [10]. There will be 50 billion things connected to the Internet by 2020, as predicted by Cisco Internet Business Solutions Group [11]. Some IoT applications might require very short response time, some might involve private data, and some might produce a large quantity of data which could be a heavy load for networks. Cloud computing is not efficient enough to support these applications.

With the push from cloud services and pull from IoT, we envision that the edge of the network is changing from data consumer to data producer as well as data consumer. In this paper, we attempt to contribute the concept of edge computing. We start from the analysis of why we need edge computing, then we give our definition and vision of edge computing. Several case studies like cloud offloading, smart home and city as well as collaborative edge are introduced to further explain edge computing in a detailed manner, followed by some challenges and opportunities in programmability, naming, data abstraction, service management, privacy and security, as well as optimization metrics that are worth future research and study.

The remaining parts of this paper are organized as follows. Section II discusses the need for edge computing as well as gives the definition of edge computing. In Section III, we show some edge computing case studies. Section IV presents the possible challenges and opportunities. Finally, this paper concludes in Section V.

II. What Is Edge Computing

Data is increasingly produced at the edge of the network, therefore, it would be more efficient to also process the data at the edge of the network. Previous work such as micro datacenter [12], [13], cloudlet [14], and fog computing [15] has been introduced to the community because cloud computing is not always efficient for data processing when the data is produced at the edge of the network. In this section, we list some reasons why edge computing is more efficient than cloud computing for some computing services, then we give our definition and understanding of edge computing.

A. Why Do We Need Edge Computing

1) Push From Cloud Services: Putting all the computing tasks on the cloud has been proved to be an efficient way for data processing since the computing power on the cloud outclasses the capability of the things at the edge. However, compared to the fast developing data processing speed, the bandwidth of the network has come to a standstill. With the growing quantity of data generated at the edge, speed of data

Manuscript received February 25, 2016; revised April 13, 2016; accepted June 02, 2016. Date of publication June 9, 2016; date of current version September 8, 2016.

The authors are with the Department of Computer Science, Wayne State University, Detroit, MI 48201 USA (e-mail: weisong@wayne.edu; jiecao@wayne.edu; quan.zhang@wayne.edu; huizi@wayne.edu; xu.lanyu@wayne.edu).

Digital Object Identifier 10.1109/JIOT.2016.2579198

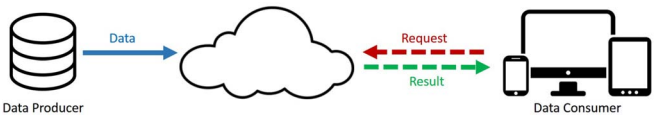


Fig. 1. Cloud computing paradigm.

transportation is becoming the bottleneck for the cloud-based computing paradigm. For example, about 5 Gigabyte data will be generated by a Boeing 787 every second [16], but the bandwidth between the airplane and either satellite or base station on the ground is not large enough for data transmission. Consider an autonomous vehicle as another example. One Gigabyte data will be generated by the car every second and it requires real-time processing for the vehicle to make correct decisions [17]. If all the data needs to be sent to the cloud for processing, the response time would be too long. Not to mention that current network bandwidth and reliability would be challenged for its capability of supporting a large number of vehicles in one area. In this case, the data needs to be processed at the edge for shorter response time, more efficient processing and smaller network pressure.

2) Pull From IoT: Almost all kinds of electrical devices will become part of IoT, and they will play the role of data producers as well as consumers, such as air quality sensors, LED bars, streetlights and even an Internet-connected microwave oven. It is safe to infer that the number of things at the edge of the network will develop to more than billions in a few years. Thus, raw data produced by them will be enormous, making conventional cloud computing not efficient enough to handle all these data. This means most of the data produced by IoT will never be transmitted to the cloud, instead it will be consumed at the edge of the network.

Fig. 1 shows the conventional cloud computing structure. Data producers generate raw data and transfer it to cloud, and data consumers send request for consuming data to cloud, as noted by the blue solid line. The red dotted line indicates the request for consuming data being sent from data consumers to cloud, and the result from cloud is represented by the green dotted line. However, this structure is not sufficient for IoT. First, data quantity at the edge is too large, which will lead to huge unnecessary bandwidth and computing resource usage. Second, the privacy protection requirement will pose an obstacle for cloud computing in IoT. Lastly, most of the end nodes in IoT are energy constrained things, and the wireless communication module is usually very energy hungry, so offloading some computing tasks to the edge could be more energy efficient.

3) Change From Data Consumer to Producer: In the cloud computing paradigm, the end devices at the edge usually play as data consumer, for example, watching a YouTube video on your smart phone. However, people are also producing data nowadays from their mobile devices. The change from data consumer to data producer/consumer requires more function placement at the edge. For example, it is very normal that people today take photos or do video recording then share the data through a cloud service such as YouTube, Facebook, Twitter, or Instagram. Moreover, every single minute, YouTube

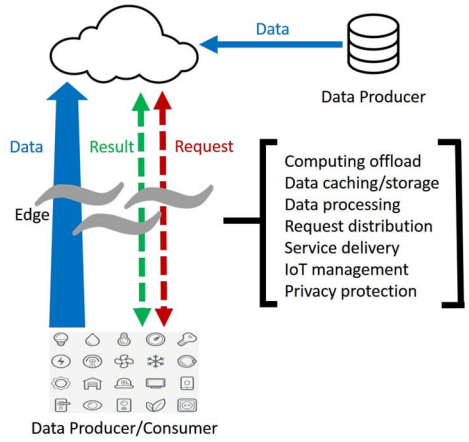


Fig. 2. Edge computing paradigm.

users upload 72 h of new video content; Facebook users share nearly 2.5 million pieces of content; Twitter users tweet nearly 300 000 times; Instagram users post nearly 220 000 new photos [18]. However, the image or video clip could be fairly large and it would occupy a lot of bandwidth for uploading. In this case, the video clip should be demised and adjusted to suitable resolution at the edge before uploading to cloud. Another example would be wearable health devices. Since the physical data collected by the things at the edge of the network is usually private, processing the data at the edge could protect user privacy better than uploading raw data to cloud.

B. What Is Edge Computing

Edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services. Here we define "edge" as any computing and network resources along the path between data sources and cloud data centers. For example, a smart phone is the edge between body things and cloud, a gateway in a smart home is the edge between home things and cloud, a micro data center and a cloudlet [14] is the edge between a mobile device and cloud. The rationale of edge computing is that computing should happen at the proximity of data sources. From our point of view, edge computing is interchangeable with fog computing [19], but edge computing focus more toward the things side, while fog computing focus more on the infrastructure side. We envision that edge computing could have as big an impact on our society as has the cloud computing.

Fig. 2 illustrates the two-way computing streams in edge computing. In the edge computing paradigm, the things not only are data consumers, but also play as data producers. At the edge, the things can not only request service and content from the cloud but also perform the computing tasks from the cloud. Edge can perform computing offloading, data storage, caching and processing, as well as distribute request and delivery service from cloud to user. With those jobs in the network, the edge itself needs to be well designed to meet the requirement efficiently in service such as reliability, security, and privacy protection.

C. Edge Computing Benefits

In edge computing we want to put the computing at the proximity of data sources. This have several benefits compared to traditional cloud-based computing paradigm. Here we use several early results from the community to demonstrate the potential benefits. Researchers built a proof-of-concept platform to run face recognition application in [20], and the response time is reduced from 900 to 169 ms by moving computation from cloud to the edge. Ha *et al.* [21] used cloudlets to offload computing tasks for wearable cognitive assistance, and the result shows that the improvement of response time is between 80 and 200ms. Moreover, the energy consumption could also be reduced by 30%–40% by cloudlet offloading. clonecloud in [22] combine partitioning, migration with merging, and on-demand instantiation of partitioning between mobile and the cloud, and their prototype could reduce 20× running time and energy for tested applications.

III. Case Study

In this section, we give several case studies where edge computing could shine to further illustrate our vision of edge computing.

A. Cloud Offloading

In the cloud computing paradigm, most of the computations happen in the cloud, which means data and requests are processed in the centralized cloud. However, such a computing paradigm may suffer longer latency (e.g., long tail latency), which weakens the user experience. Numbers of researches have addressed the cloud offloading in terms of energy-performance tradeoff in a mobile-cloud environment [22]–[26]. In edge computing, the edge has certain computation resources, and this provides a chance to offload part of the workload from cloud.

In the traditional content delivery network, only the data is cached at the edge servers. This is based on the fact that the content provider provides the data on the Internet, which is true for the past decades. In the IoT, the data is produced and consumed at the edge. Thus, in the edge computing paradigm, not only data but also operations applied on the data should be cached at the edge.

One potential application that could benefit from edge computing is online shopping services. A customer may manipulate the shopping cart frequently. By default, all these changes on his/her shopping cart will be done in the cloud, and then the new shopping cart view is updated on the customer's device. This process may take a long time depending on network speed and the load level of servers. It could be even longer for mobile devices due to the relatively low bandwidth of a mobile network. As shopping with mobile devices is becoming more and more popular, it is important to improve the user experience, especially latency related. In such a scenario, if the shopping cart updating is offloaded from cloud servers to edge nodes, the latency will be dramatically reduced. As we mentioned, the users' shopping cart data and related operations (e.g., add an item, update an item, delete an item) both can be cached at the edge node. The new shopping cart view can be generated immediately upon the user request reaching the edge node. Of course, the data at the edge node should be synchronized with the cloud, however, this can be done in the background.

Another issue involves the collaboration of multiple edges when a user moves from one edge node to another. One simple solution is to cache the data to all edges the user may reach. Then the synchronization issue between edge nodes rises up. All these issues could become challenges for future investigation. At the bottom line, we can improve the interactive services quality by reducing the latency. Similar applications also include the following.

1) Navigation applications can move the navigating or searching services to the edge for a local area, in which case only a few map blocks are involved.
2) Content filtering/aggregating could be done at the edge nodes to reduce the data volume to be transferred.
3) Real-time applications such as vision-aid entertainment games, augmented reality, and connected health, could make fast responses by using edge nodes.

Thus, by leveraging edge computing, the latency and consequently the user experience for time-sensitive application could be improved significantly.

B. Video Analytics

The widespread of mobilephones and network cameras make video analytics an emerging technology. Cloud computing is no longer suitable for applications that requires video analytics due to the long data transmission latency and privacy concerns. Here we give an example of finding a lost child in the city. Nowadays, different kinds of cameras are widely deployed in the urban area and in each vehicle. When a child is missing, it is very possible that this child can be captured by a camera. However, the data from the camera will usually not be uploaded to the cloud because of privacy issues or traffic cost, which makes it extremely difficult to leverage the wide area camera data. Even if the data is accessible on the cloud, uploading and searching a huge quantity of data could take a long time, which is not tolerable for searching a missing child. With the edge computing paradigm, the request of searching a child can be generated from the cloud and pushed to all the things in a target area. Each thing, for example, a smart phone, can perform the request and search its local camera data and only report the result back to the cloud. In this paradigm, it is possible to leverage the data and computing power on every thing and get the result much faster compared with solitary cloud computing.

C. Smart Home

IoT would benefit the home environment a lot. Some products have been developed and are available on the market such as smart light, smart TV, and robot vacuum. However, just adding a Wi-Fi module to the current electrical device and connecting it to the cloud is not enough for a smart home. In a smart home environment, besides the connected device, cheap wireless sensors and controllers should be deployed to room, pipe, and even floor and wall. These things would report

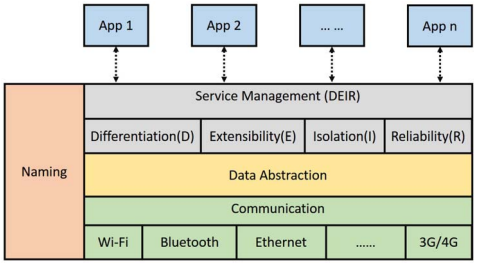


Fig. 3. Structure of edgeOS in the smart home environment.

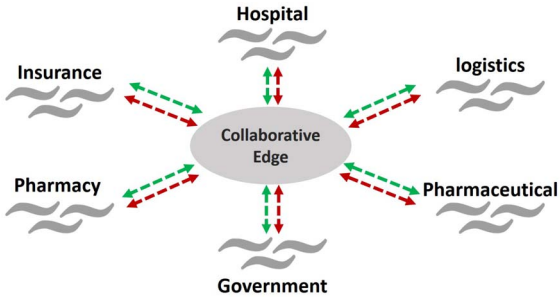


Fig. 4. Collaborative edge example: connected health.

an impressive amount of data and for the consideration of data transportation pressure and privacy protection, this data should be mostly consumed in the home. This feature makes the cloud computing paradigm unsuitable for a smart home. Nevertheless, edge computing is considered perfect for building a smart home: with an edge gateway running a specialized edge operating system (edgeOS) in the home, the things can be connected and managed easily in the home, the data can be processed locally to release the burdens for Internet bandwidth, and the service can also be deployed on the edgeOS for better management and delivery. More opportunities and potential challenges are discussed in Section IV.

Fig. 3 shows the structure of a variant of edgeOS in the smart home environment. EdgeOS needs to collect data from mobile devices and all kinds of things through multiple communication methods such as Wi-Fi, BlueTooth, ZigBee, or a cellular network. Data from different sources needs to be fused and massaged in the data abstraction layer. Detailed description of this process will be discussed in Section IV-C. On top of the data abstraction layer is the service management layer. Requirements including differentiation, extensibility, isolation, and reliability will be supported in this layer. In Section IV-D, this issue will be further addressed. The naming mechanism is required for all layers with different requirements. Thus, we leave the naming module in a cross-layer fashion. Challenges in naming are discussed in Section IV-B.

D. Smart City

The edge computing paradigm can be flexibly expanded from a single home to community, or even city scale. Edge computing claims that computing should happen as close as possible to the data source. With this design, a request could be generated from the top of the computing paradigm and be actually processed at the edge. Edge computing could be an ideal platform for smart city considering the following characteristics.

1) Large Data Quantity: A city populated by 1 million people will produce 180 PB data per day by 2019 [9], contributed by public safety, health, utility, and transports, etc. Building centralized cloud data centers to handle all of the data is unrealistic because the traffic workload would be too heavy. In this case, edge computing could be an efficient solution by processing the data at the edge of the network.

2) Low Latency: For applications that require predictable and low latency such as health emergency or public safety, edge computing is also an appropriate paradigm since it could save the data transmission time as well as simplify the network structure. Decision and diagnosis could be made as well as distributed from the edge of the network, which is more efficient compared with collecting information and making decision at central cloud.

3) Location Awareness: For geographic-based applications such as transportation and utility management, edge computing exceed cloud computing due to the location awareness. In edge computing, data could be collected and processed based on geographic location without being transported to cloud.

E. Collaborative Edge

Cloud, arguably, has become the de facto computing platform for the big data processing by academia and industry. A key promise behind cloud computing is that the data should be already held or is being transmitted to the cloud and will eventually be processed in the cloud. In many cases, however, the data owned by stakeholders is rarely shared to each other due to privacy concerns and the formidable cost of data transportation. Thus, the chance of collaboration among multiple stake-holders is limited. Edge, as a physical small data center that connects cloud and end user with data processing capability, can also be part of the logical concept. collaborative edge, which connects the edges of multiple stakeholders that are geographically distributed despite their physical location and network structure is proposed [15]. Those *ad hoc*-like connected edges provide the opportunity for stakeholders to share and cooperate data.

One of the promising applications in the near future is connected health, as shown in Fig. 4. The demand of geographically distributed data processing applications, i.e., healthcare, requires data sharing and collaboration among enterprises in multiple domains. To attack this challenge, collaborative edge can fuse geographically distributed data by creating virtual shared data views. The virtual shared data is exposed to end users via a predefined service interface. An application will leverage this public interface to compose complex services for end users. These public services are provided by participants of collaborative edge, and the computation only occurs in the participant's data facility such that the data privacy and integrity can be ensured.

To show the potential benefits of collaborative edge, we use connected healthcare as a case study. We use a flu outbreak as the beginning of our case study. The patients flow

to hospitals, and the electronic medical record (EMR) of the patients will be updated. The hospital summarizes and shares the information for this flu outbreak, such as the average cost, the symptoms, and the population, etc. A patient theoretically will follow the prescription to get the pills from a pharmacy. One possibility is that a patient did not follow the therapy. Then the hospital has to take the responsibility for rehospitalization since it cannot get the proof that the patient did not take the pills. Now, via collaborative edge, the pharmacy can provide the purchasing record of a patient to the hospital, which significantly facilitates healthcare accountability.

At the same time, the pharmacies retrieve the population of the flu outbreak using the collaborative edge services provided by hospitals. An apparent benefit is that the pharmacies have enough inventory to obtain much more profits. Behind the drug purchasing, the pharmacy can leverage data provided by pharmaceutical companies and retrieve the locations, prices and inventories of all drug warehouses. It also sends a transport price query request to the logistics companies. Then the pharmacy can make an order plan by solving the total cost optimization problem according to retrieved information. The pharmaceutical companies also receive a bunch of flu drug orders from pharmacies. At this point, a pharmaceutical company can reschedule the production plan and rebalance the inventories of the warehouses. Meanwhile, the centers for disease control and prevention, as our government representative in our case, is monitoring the flu population increasing at wide range areas, can consequently raise a flu alert to the people in the involved areas. Besides, further actions can be taken to prevent the spread of flu outbreak.

After the flu outbreak, the insurance companies have to pay the bill for the patients based on the policy. The insurance companies can analyze the proportion of people who has the flu during the outbreak. This proportion and the cost for flu treatment are significant factors to adjust the policy price for the next year. Furthermore, the insurance companies can also provide a personalized healthcare policy based on their EMR if the patient would like to share it.

Through this simple case, most of the participants can benefit from collaborative edge in terms of reducing operational cost and improving profitability. However, some of them, like hospitals in our case, could be a pure contributor to the healthcare community since they are the major information collector in this community.

IV. Challenges and Opportunities

We have described five potential applications of edge computing in the last section. To realize the vision of edge computing, we argue that the systems and network community need to work together. In this section, we will further summarize these challenges in detail and bring forward some potential solutions and opportunities worth further research, including programmability, naming, data abstraction, service management, privacy and security and optimization metrics.

A. Programmability

In cloud computing, users program their code and deploy them on the cloud. The cloud provider is in charge to decide where the computing is conducted in a cloud. Users have zero or partial knowledge of how the application runs. This is one of the benefits of cloud computing that the infrastructure is transparent to the user. Usually, the program is written in one programing language and compiled for a certain target platform, since the program only runs in the cloud. However, in the edge computing, computation is offloaded from the cloud, and the edge nodes are most likely heterogeneous platforms. In this case, the runtime of these nodes differ from each other, and the programmer faces huge difficulties to write an application that may be deployed in the edge computing paradigm.

To address the programmability of edge computing, we propose the concept of computing stream that is defined as a serial of functions/computing applied on the data along the data propagation path. The functions/computing could be entire or partial functionalities of an application, and the computing can occur anywhere on the path as long as the application defines where the computing should be conducted. The computing stream is software defined computing flow such that data can be processed in distributed and efficient fashion on data generating devices, edge nodes, and the cloud environment. As defined in edge computing, a lot of computing can be done at the edge instead of the centric cloud. In this case, the computing stream can help the user to determine what functions/computing should be done and how the data is propagated after the computing happened at the edge. The function/computing distribution metric could be latency-driven, energy cost, TCO, and hardware/software specified limitations. The detailed cost model is discussed in Section IV-F. By deploying a computing stream, we expect that data is computed as close as possible to the data source, and the data transmission cost can be reduced. In a computing stream, the function can be reallocated, and the data and state along with the function should also be reallocated. Moreover, the collaboration issues (e.g., synchronization, data/state migration, etc.) have to be addressed across multiple layers in the edge computing paradigm.

B. Naming

In edge computing, one important assumption is that the number of things is tremendously large. At the top of the edge nodes, there are a lot of applications running, and each application has its own structure about how the service is provided. Similar to all computer systems, the naming scheme in edge computing is very important for programing, addressing, things identification, and data communication. However, an efficient naming mechanism for the edge computing paradigm has not been built and standardized yet. Edge practitioners usually needs to learn various communication and network protocols in order to communicate with the heterogeneous things in their system. The naming scheme for edge computing needs to handle the mobility of things, highly dynamic network topology, privacy and security protection, as well as the scalability targeting the tremendously large amount of unreliable things.

Traditional naming mechanisms such as DNS and uniform resource identifier satisfy most of the current networks very

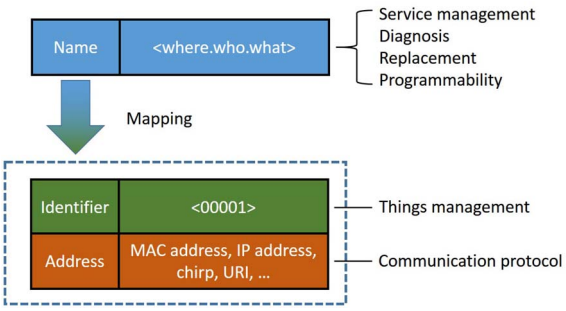


Fig. 5. Naming mechanism in edgeOS.

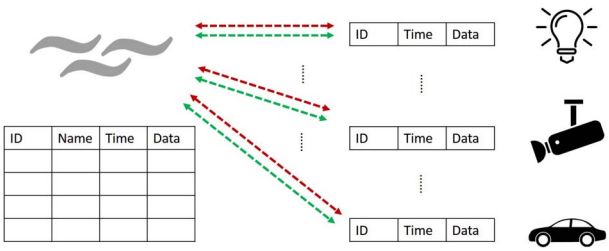


Fig. 6. Data abstraction issue for edge computing.

well. However, they are not flexible enough to serve the dynamic edge network since sometimes most of the things at edge could be highly mobile and resource constrained. Moreover, for some resource constrained things at the edge of the network, IP based naming scheme could be too heavy to support considering its complexity and overhead.

New naming mechanisms such as named data networking (NDN) [27] and MobilityFirst [28] could also be applied to edge computing. NDN provide a hierarchically structured name for content/data centric network, and it is human friendly for service management and provides good scalability for edge. However, it would need extra proxy in order to fit into other communication protocols such as BlueTooth or ZigBee, and so on. Another issue associated with NDN is security, since it is very hard to isolate things hardware information with service providers. MobileFirst can separate name from network address in order to provide better mobility support, and it would be very efficient if applied to edge services where things are of highly mobility. Neverless, a global unique identification (GUID) needs to be used for naming is MobileFirst, and this is not required in related fixed information aggregation service at the edge of the network such as home environment. Another disadvantage of MobileFirst for edge is the difficulty in service management since GUID is not human friendly.

For a relative small and fixed edge such as home environment, let the edgeOS assign network address to each thing could be a solution. With in one system, each thing could have a unique human friendly name which describes the following information: location (where), role (who), and data description (what), for example, "kitchen.oven2.temperature3." Then the edgeOS will assign identifier and network address to this thing, as shown in Fig. 5. The human friendly name is unique for each thing and it will be used for service management, things diagnosis, and component replacement. For user and service provider, this naming mechanism makes management very easy. For example, the user will receive a message from edgeOS like "Bulb 3 (what) of the ceiling light (who) in living room (where) failed," and then the user can directly replace the failed bulb without searching for an error code or reconfigure the network address for the new bulb. Moreover, this naming mechanism provides better programmability to service providers and in the meanwhile, it blocks service providers from getting hardware information, which will protect data privacy and security better. Unique identifier and network address could be mapped from human friendly name. Identifier will be used for things management in edgeOS. Network address such as IP address or MAC address will be used to support various communication protocols such as BlueTooth, ZigBee or WiFi, and so on. When targeting highly dynamic environment such as city level system, we think it is still an open problem and worth further investigation by the community.

C. Data Abstraction

Various applications can run on the edgeOS consuming data or providing service by communicating through the air position indicators from the service management layer. Data abstraction has been well discussed and researched in the wireless sensor network and cloud computing paradigm. However, in edge computing, this issue becomes more challenging. With IoT, there would be a huge number of data generators in the network, and here we take a smart home environment as an example. In a smart home, almost all of the things will report data to the edgeOS, not to mention the large number of things deployed all around the home. However, most of the things at the edge of the network, only periodically report sensed data to the gateway. For example, the thermometer could report the temperature every minute, but this data will most likely only be consumed by the real user several times a day. Another example could be a security camera in the home which might keep recording and sending the video to the gateway, but the data will just be stored in the database for a certain time with nobody actually consuming it, and then be flushed by the latest video.

Based on this observation, we envision that human involvement in edge computing should be minimized and the edge node should consume/process all the data and interact with users in a proactive fashion. In this case, data should be preprocessed at the gateway level, such as noise/low-quality removal, event detection, and privacy protection, and so on. Processed data will be sent to the upper layer for future service providing. There will be several challenges in this process.

First, data reported from different things comes with various formats, as shown in Fig. 6. For the concern of privacy and security, applications running on the gateway should be blinded from raw data. Moreover, they should extract the knowledge they are interested in from an integrated data table. We can easily define the table with id, time, name, data (e.g.,{0000, 12:34:56PM 01/01/2016, kitchen.oven2.temperature3, 78}) such that any edge thing's data can be fitted in. However, the details of sensed data have been hidden, which may affect the usability of data.

Second, it is sometimes difficult to decide the degree of data abstraction. If too much raw data is filtered out, some applications or services could not learn enough knowledge. However, if we want to keep a large quantity of raw data, there would be a challenge for data storage. Lastly, data reported by things at edge could be not reliable sometime, due to the low precision sensor, hazard environment, and unreliable wireless connection. In this case, how to abstract useful information from unreliable data source is still a challenge for IoT application and system developers.

One more issue with data abstraction is the applicable operations on the things. Collecting data is to serve the application and the application should be allowed to control (e.g., read from and write to) the things in order to complete certain services the user desires. Combining the data representation and operations, the data abstraction layer will serve as an public interface for all things connected to edgeOS. Furthermore, due the heterogeneity of the things, both data representation and allowed operations could diverse a lot, which also increases the barrier of universal data abstraction.

D. Service Management

In terms of service management at the edge of the network, we argue that the following four fundamental features should be supported to guarantee a reliable system, including differentiation, extensibility, isolation, and reliability.

Differentiation: With the fast growth of IoT deployment, we expected multiple services will be deployed at the edge of the network, such as Smart Home. These services will have different priorities. For example, critical services such as things diagnosis and failure alarm should be processed earlier than ordinary service. Health related service, for example, fall detection or heart failure detection should also have a higher priority compared with other service such as entertainment.

Extensibility: Extensibility could be a huge challenge at the edge of the network, unlike a mobile system, the things in the IoT could be very dynamic. When the owner purchases a new thing, can it be easily added to the current service without any problem? Or when one thing is replaced due to wearing out, can the previous service adopt a new node easily? These problems should be solved with a flexible and extensible design of service management layer in the edgeOS.

Isolation: Isolation would be another issue at the edge of the network. In mobile OS, if an application fails or crashes, the whole system will usually crash and reboot. Or in a distributed system the shared resource could be managed with different synchronization mechanisms such as a lock or token ring. However, in a smart edgeOS, this issue might be more complicated. There could be several applications that share the same data resource, for example, the control of light. If one application failed or was not responding, a user should still be able to control their lights, without crashing the whole edgeOS. Or when a user removes the only application that controls lights from the system, the lights should still be alive rather than experiencing a lost connection to the edgeOS. This challenge could be potentially solved by introducing a deployment/undeployment framework. If the conflict could be detected by the OS before an application is installed, then a user can be warned and avoid the potential access issue. Another side of the isolation challenge is how to isolate a user's private data from third party applications. For example, your activity tracking application should not be able to access your electricity usage data. To solve this challenge, a well-designed control access mechanism should be added to the service management layer in the edgeOS.

Reliability: Last but not least, reliability is also a key challenge at the edge of the network. We identify the challenges in reliability from the different views of service, system, and data here.

1) From the service point of view, it is sometimes very hard to identify the reason for a service failure accurately at field. For example, if an air conditioner is not working, a potential reason could be that a power cord is cut, compressor failure, or even a temperature controller has run out of battery. A sensor node could have lost connection very easily to the system due to battery outage, bad connection condition, component wear out, etc. At the edge of the network, it is not enough to just maintain a current service when some nodes lose connection, but to provide the action after node failure makes more sense to the user. For example, it would be very nice if the edgeOS could inform the user which component in the service is not responding, or even alert the user ahead if some parts in the system have a high risk of failure. Potential solutions for this challenge could be adapted from a wireless sensor network, or industrial network such as PROFINET [29].
2) From the system point of view, it is very important for the edgeOS to maintain the network topology of the whole system, and each component in the system is able to send status/diagnosis information to the edgeOS. With this feature, services such as failure detection, thing replacement, and data quality detection could be easily deployed at the system level.
3) From the data point of view, reliability challenge rise mostly from the data sensing and communication part. As previously researched and discussed, things at the edge of the network could fail due to various reasons and they could also report low fidelity data under unreliable condition such as low battery level [30]. Also various new communication protocols for IoT data collection are also proposed. These protocols serves well for the support of huge number of sensor nodes and the highly dynamic network condition [31]. However, the connection reliability is not as good as BlueTooth or WiFi. If both sensing data and communication are not reliable, how the system can still provide reliable service by leveraging multiple reference data source and historical data record is still an open challenge.

E. Privacy and Security

At the edge of the network, usage privacy and data security protection are the most important services that should be provided. If a home is deployed with IoT, a lot of privacy

information can be learned from the sensed usage data. For example, with the reading of the electricity or water usage, one can easily speculate if the house is vacant or not. In this case, how to support service without harming privacy is a challenge. Some of the private information could be removed from data before processing such as masking all the faces in the video. We think that keeping the computing at the edge of data resource, which means in the home, could be a decent method to protect privacy and data security. To protect the data security and usage privacy at the edge of the network, several challenges remain open.

First is the awareness of privacy and security to the community. We take WiFi networks security as an example. Among the 439 million households who use wireless connections, 49% of WiFi networks are unsecured, and 80% of households still have their routers set on default passwords. For public WiFi hotspots, 89% of them are unsecured [32]. All the stake holders including service provider, system and application developer and end user need to aware that the users' privacy would be harmed without notice at the edge of the network. For example, ip camera, health monitor, or even some WiFi enabled toys could easily be connected by others if not protected properly.

Second is the ownership of the data collected from things at edge. Just as what happened with mobile applications, the data of end user collected by things will be stored and analyzed at the service provider side. However, leave the data at the edge where it is collected and let the user fully own the data will be a better solution for privacy protection. Similar to the health record data, end user data collected at the edge of the network should be stored at the edge and the user should be able to control if the data should be used by service providers. During the process of authorization, highly private data could also be removed by the things to further protect user privacy.

Third is the missing of efficient tools to protect data privacy and security at the edge of the network. Some of the things are highly resource constrained so the current methods for security protection might not be able to be deployed on thing because they are resource hungry. Moreover, the highly dynamic environment at the edge of the network also makes the network become vulnerable or unprotected. For privacy protection, some platform such as Open mHealth is proposed to standardize and store health data [33], but more tools are still missing to handle diverse data attributes for edge computing.

F. Optimization Metrics

In edge computing, we have multiple layers with different computation capability. Workload allocation becomes a big issue. We need to decide which layer to handle the workload or how many tasks to assign at each part. There are multiple allocation strategies to complete a workload, for instances, evenly distribute the workload on each layer or complete as much as possible on each layer. The extreme cases are fully operated on endpoint or fully operated on cloud. To choose an optimal allocation strategy, we discuss several optimization metrics in this section, including latency, bandwidth, energy and cost.

Latency: Latency is one of the most important metrics to evaluate the performance, especially in interaction applications/services [34], [35]. Servers in cloud computing provide high computation capability. They can handle complex workloads in a relatively short time, such as image processing, voice recognition and so on. However, latency is not only determined by computation time. Long WAN delays can dramatically influence the real-time/interaction intensive applications' behavior [36]. To reduce the latency, the workload should better be finished in the nearest layer which has enough computation capability to the things at the edge of the network. For example, in the smart city case, we can leverage phones to process their local photos first then send a potential missing child's info back to the cloud instead of uploading all photos. Due to the large amount of photos and their size, it will be much faster to preprocess at the edge. However, the nearest physical layer may not always be a good option. We need to consider the resource usage information to avoid unnecessary waiting time so that a logical optimal layer can be found. If a user is playing games, since the phone's computation resource is already occupied, it will be better to upload a photo to the nearest gateway or micro-center.

Bandwidth: From latency's point of view, high bandwidth can reduce transmission time, especially for large data (e.g., video, etc.) [37], [38]. For short distance transmission, we can establish high bandwidth wireless access to send data to the edge. On one hand, if the workload can be handled at the edge, the latency can be greatly improved compared to work on the cloud. The bandwidth between the edge and the cloud is also saved. For example, in the smart home case, almost all the data can be handled in the home gateway through Wi-Fi or other high speed transmission methods. In addition, the transmission reliability is also enhanced as the transmission path is short. On the other hand, although the transmission distance cannot be reduced since the edge cannot satisfy the computation demand, at least the data is preprocessed at the edge and the upload data size will be significantly reduced. In the smart city case, it is better to preprocess photos before upload, so the data size can be greatly reduced. It saves the users' bandwidth, especially if they are using a carriers' data plan. From a global perspective, the bandwidth is saved in both situations, and it can be used by other edges to upload/download data. Hence, we need to evaluate if a high bandwidth connection is needed and which speed is suitable for an edge. Besides, to correctly determine the workload allocation in each layer, we need to consider the computation capability and bandwidth usage information in layers to avoid competition and delay.

Energy: Battery is the most precious resource for things at the edge of the network. For the endpoint layer, offloading workload to the edge can be treated as an energy free method [22], [39]. So for a given workload, is it energy efficient to offload the whole workload (or part of it) to the edge rather than compute locally? The key is the tradeoff between the computation energy consumption and transmission energy consumption. Generally speaking, we first need to consider the power characteristics of the workload. Is it computation intensive? How much resource will it use to run locally? Besides the

network signal strength [40], the data size and available bandwidth will also influence the transmission energy overhead [28]. We prefer to use edge computing only if the transmission overhead is smaller than computing locally. However, if we care about the whole edge computing process rather than only focus on endpoints, total energy consumption should be the accumulation of each used layer's energy cost. Similar to the endpoint layer, each layer's energy consumption can be estimated as local computation cost plus transmission cost. In this case, the optimal workload allocation strategy may change. For example, the local data center layer is busy, so the workload is continuously uploaded to the upper layer. Comparing with computing on endpoints, the multihop transmission may dramatically increase the overhead which causes more energy consumption.

Cost: From the service providers' perspective, e.g., YouTube, Amazon, etc., edge computing provides them less latency and energy consumption, potential increased throughput and improved user experience. As a result, they can earn more money for handling the same unit of workload. For example, based on most residents' interest, we can put a popular video on the building layer edge. The city layer edge can free from this task and handle more complex work. The total throughput can be increased. The investment of the service providers is the cost to build and maintain the things in each layer. To fully utilize the local data in each layer, providers can charge users based on the data location. New cost models need to be developed to guarantee the profit of the service provider as well as acceptability of users.

Workload allocation is not an easy task. The metrics are closely related to each other. For example, due to the energy constraints, a workload needs to be complete on the city data center layer. Comparing with the building server layer, the energy limitation inevitably affects the latency. Metrics should be given priority (or weight) for different workloads so that a reasonable allocation strategy can be selected. Besides, the cost analysis needs to be done in runtime. The interference and resource usage of concurrent workloads should be considered as well.

V. Conclusion

Nowadays, more and more services are pushed from the cloud to the edge of the network because processing data at the edge can ensure shorter response time and better reliability. Moreover, bandwidth could also be saved if a larger portion of data could be handled at the edge rather than uploaded to the cloud. The burgeoning of IoT and the universalized mobile devices changed the role of edge in the computing paradigm from data consumer to data producer/consumer. It would be more efficient to process or massage data at the edge of the network. In this paper, we came up with our understanding of edge computing, with the rationale that computing should happen at the proximity of data sources. Then we list several cases whereby edge computing could flourish from cloud offloading to a smart environment such as home and city. We also introduce collaborative edge, since edge can connect end user and cloud both physically and logically so not only is the conventional cloud computing paradigm still supported, but also it can connect long distance networks together for data sharing and collaboration because of the closeness of data. At last, we put forward the challenges and opportunities that are worth working on, including programmability, naming, data abstraction, service management, privacy and security, as well as optimization metrics. edge computing is here, and we hope this paper will bring this to the attention of the community.

Acknowledgment

The authors would like to thank T. Zhang from Cisco for early discussions and W. Zhang from Alibaba for the idea of edge computing and fog computing. The example of applying a shopping cart at the edge was given by W. Zhang. The authors would also like to thank Dr. C. Wang for inviting them to submit this paper.

References

[1] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.
[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, Incline Village, NV, USA, 2010, pp. 1–10.
[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, vol. 10. Boston, MA, USA, 2010, p. 10.
[6] K. Ashton, "That Internet of Things thing," *RFiD J.*, vol. 22, no. 7, pp. 97–114, 2009.
[7] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the Internet of things," vol. 20, no. 10, 2010.
[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[9] "Cisco global cloud index: Forecast and methodology, 2014–2019 white paper," 2014.
[10] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," CISCO White Paper, vol. 1, pp. 1–11, 2011.
[11] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," CISCO White Paper, vol. 1, pp. 1–11, 2011.
[12] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.
[13] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Syst. Appl. Services*, San Francisco, CA, USA, 2010, pp. 49–62.
[14] 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.
[15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of things," in *Proc. 1st Edition MCC Workshop Mobile Cloud Comput.*, Helsinki, Finland, 2012, pp. 13–16.
[16] *Boeing 787s to Create Half a Terabyte of Data Per Flight, Says Virgin Atlantic*. Accessed on Dec. 7, 2016. [Online]. Available: https://datafloq.com/read/self-driving-cars-create-2-petabytes-data-annually/172
[17] *Self-Driving Cars Will Create 2 Petabytes of Data, What are the Big Data Opportunities for the Car Industry?* Accessed on Dec. 7, 2016. [Online]. Available: http://www.computerworlduk.com/news/data/boeing-787s-create-half-terabyte-of-data-per-flight-says-virgin-atlantic-3433595/
[18] *Data Never Sleeps 2.0*. Accessed on Dec. 7, 2016. [Online]. Available: https://www.domo.com/blog/2014/04/data-never-sleeps-2-0/

[19] (2016). *OpenFog Architecture Overview. OpenFog Consortium Architecture Working Group*. Accessed on Dec. 7, 2016. [Online]. Available: http://www.openfogconsortium.org/wp-content/uploads/OpenFog-Architecture-Overview-WP-2-2016.pdf

[20] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Washington, DC, USA, 2015, pp. 73–78.

[21] K. Ha *et al.*, "Towards wearable cognitive assistance," in *Proc. 12th Annu. Int. Conf. Mobile Syst. Appl. Services*, Bretton Woods, NH, USA, 2014, pp. 68–81.

[22] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, Salzburg, Austria, 2011, pp. 301–314.

[23] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, 1998.

[24] G. C. Hunt and M. L. Scott, "The coign automatic distributed partitioning system," in *Proc. OSDI*, vol. 99. New Orleans, LA, USA, 1999, pp. 187–200.

[25] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[26] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 945–953.

[27] L. Zhang *et al.*, "Named data networking (NDN) project," Xerox Palo Alto Res. Center, Palo Alto, CA, USA, Tech. Rep. NDN-0001, 2010.

[28] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "MobilityFirst: A robust and trustworthy mobility-centric architecture for the future Internet," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 16, no. 3, pp. 2–13, 2012.

[29] J. Feld, "PROFINET—Scalable factory communication for all applications," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Vienna, Austria, 2004, pp. 33–38.

[30] J. Cao, L. Ren, W. Shi, and Z. Yu, "A framework for component selection in collaborative sensing application development," in *Proc. 10th IEEE Conf. Coll. Comput. Netw. Appl. Worksharing*, Miami, FL, USA, 2014, pp. 104–113.

[31] F. DaCosta, *Rethinking the Internet of Things: A Scalable Approach to Connecting Everything*. New York, NY, USA: ApressOpen, 2013.

[32] *WiFi Network Security Statistics/Graph*. Accessed on Dec. 7, 2016. [Online]. Available: http://graphs.net/wifi-stats.html/

[33] *Open Mhealth Platform*. Accessed on Dec. 7, 2016. [Online]. Available: http://www.openmhealth.org/

[34] K. R. Jackson *et al.*, "Performance analysis of high performance computing applications on the Amazon Web services cloud," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Indianapolis, IN, USA, 2010, pp. 159–168.

[35] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879143

[36] M. Satyanarayanan, "Mobile computing: The next decade," *SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 15, no. 2, pp. 2–10, 2011. [Online]. Available: http://doi.acm.org/10.1145/2016598.2016600

[37] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[38] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

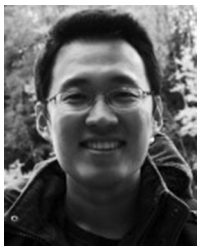
[39] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, Boston, MA, USA, 2010, p. 4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863107

[40] N. Ding *et al.*, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 29–40, Jun. 2013. [Online]. Available: http://doi.acm.org/10.1145/2494232.2466586

Weisong Shi received the Ph.D. degree in computer engineering from the Chinese Academy of Sciences, Beijing, China.

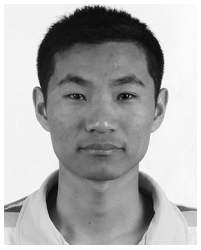
He is a Professor of computer science with Wayne State University, Detroit, MI, USA. His current research interests include big data systems, edge computing, energy-efficient computer systems, and software and mobile health.

Dr. Shi is a Senior Member of the ACM.



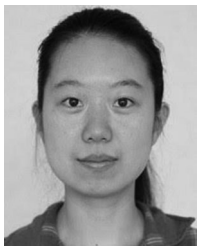
Jie Cao (S'16) received the B.S. degree in telecommunications engineering from Xidian University, Xi'an, China. He is currently pursuing the Ph.D. degree in computer science at Wayne State University, Detroit, MI, USA.

His current research interests include edge computing, smart home, mobile, and connected health.



Quan Zhang (S'16) received the B.S. degree in computer science from Tongji University, Shanghai, China. He is currently pursuing the Ph.D. degree in computer science at Wayne State University, Detroit, MI, USA.

His current research interests include distributed systems, cloud computing, and energy-efficient computing.



Youhuizi Li received the Ph.D. degree in computer science from Wayne State University, Detroit, MI, USA, in 2016.

She is an Assistant Professor of computer science with Hangzhou Dianzi University, Hangzhou, China. Her current research interests include energy-efficient computing, mobile and Internet computing, and big data systems.

Lanyu Xu received the B.S. degree in computer science from Tongji University, Shanghai, China. She is currently pursuing the Ph.D. degree in computer science at Wayne State University, Detroit, MI, USA.

Her current research interests include edge computing, smart homes, and mobile and connect health.