Software-Defined Internet of Things for Smart Urban Sensing

With more people living in cities, urban sensing is urgently required to create a comfortable and convenient living environment. IoT is the fundamental infrastructure to realize urban sensing, it should be flexible to support various application requirements and convenient management of infrastructure. Inspired by software-defined networking, which aims to make networks more flexible, the authors propose a software-defined IoT architecture for smart urban sensing.

Jiaqiang Liu, Yong Li, Min Chen, Wenxia Dong, and Depeng Jin

Abstract

With more people living in cities, urban sensing is urgently required to create a comfortable and convenient living environment. As Internet of Things (IoT) is the fundamental infrastructure to realize urban sensing, it should be flexible to support various application requirements and convenient management of infrastructure. Inspired by software-defined networking, which aims to make networks more flexible, the authors propose a software-defined IoT architecture for smart urban sensing. This architecture decouples urban sensing applications from the physical infrastructure. Centralized controllers are designed to manage physical devices and provide APIs of data acquisition, transmission, and processing services to develop urban sensing applications. With these properties, various applications can coexist on the shared infrastructure, and each application can request controllers to customize its data acquisition, transmission, and processing on-demand by generating specific configurations of physical devices. This article discusses the background, benefits, and design details of the proposed architecture as well as open problems and potential solutions to realize it, which opens a new research direction for IoT and urban sensing.

Jiaqiang Liu, Yong Li, and Depeng Jin are with Tsinghua University.

Min Chen is with Huazhong University of Science and Technology.

Wenxia Dong is with Huawei Technologies Co. Ltd.

This work is supported by the National Basic Research Program of China (973 Program) (No. 2013CB329001), and the National Nature Science Foundation of China (No. 61301080, No. 91338203, No. 91338102, and No. 61321061).

Introduction

The number of people living in cities has increased dramatically in recent years, and the trend is expected to continue. The United Nations Population Fund estimates that by the year 2030 nearly 60 percent of the world's population will live in an urban environment [1]. More convenient and comfortable living condition, as well as more opportunities for work and career development, are the main motivations for urbanization. However, the explosion of city populations resulting from urbanization is straining existing daily and public facilities such as transportation, healthcare, and security, and creating new problems like environmental pollution. These issues need to be solved to provide sustainable urbanization.

Urban sensing is one of the most promising solutions to address the above problems. As a simple example, if the real-time traffic load is known, the efficiency of existing transportation systems can be enhanced significantly. The Internet of Things (IoT) has the potential to make urban sensing a reality. With an increasing number of various sensor devices connected to the Internet, it is possible to obtain the infrastructural and environmental data in real time that would enable an efficient approach to perceive and manage urban facilities. Many cities have deployed sensor platforms to support urban sensing. For example, London has deployed various sensor nodes to obtain traffic, environmental, and utilities data [2], and various experimental platforms of IoT have been developed for research [3]. In addition to dedicated sensor platforms, as human carried smart phones are equipped with a rich set of sensors like cameras, digital compasses, GPS, etc., they can also be exploited to realize urban sensing. This is referred to as mobile crowd sensing [4].

Currently, IoT is still in the initial stages of development and deployment. However, there is no doubt that the IoT will have an important impact on people's lives, just like Internet does today. The Internet has had great success and changed our lives, but it still faces some problems. On one hand, as the control intelligence, which is implemented by various routing and management protocols, is embedded in every router/switch and is hard to change, Internet infrastructure becomes ossified and therefore evolves slowly. Also, vendor-dependent interfaces make the infrastructure management complex and error-prone. On the other hand, it only provides best-effort service and thus prevents the development of highly personalized applications with specific requirements on service quality and user experience. The design of the future IoT architecture should avoid these problems to support sustainable evolution, convenient management, and various application requirements.

COMMUNICATIONS STANDARDS

Software-defined networks (SDNs) [5] offer the ability to address the above mentioned problems. In SDN, the control intelligence is moved from data plane devices (switches, routers) and implemented in a logically centralized controller, which interacts with data plane devices through standard interfaces. The network operator runs software programs on the controller to automatically manage data plane devices and optimize network resource usage. They can further develop up-to-date control schemes to provide different network services for applications, e.g. providing QoS guaranteed forwarding services.

Inspired by SDN, this article proposes a software-defined IoT (SD-IoT) architecture for smart urban sensing. In accordance with SDN, SD-IoT also decouples the control logic from functions of the physical devices through a logically centralized controller that manages the devices via standard interface. In particular, SD-IoT extends the spirit of the software-defined approach from network devices to sensor platforms and the cloud, and combines them to support urban sensing applications together by

 0163-6804/14/$25.00

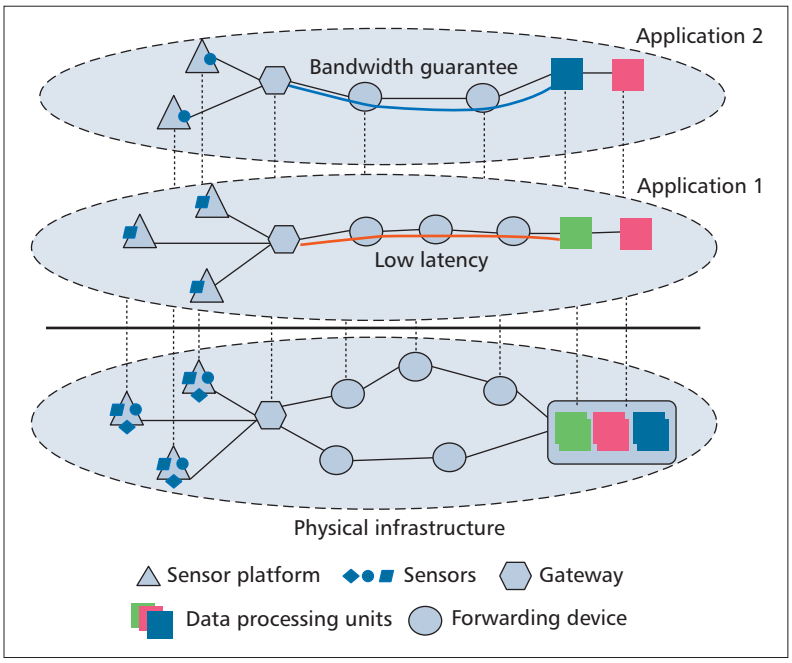


Figure 1. Illustration of software-defined IoT solution.

providing well-defined service APIs in terms of data acquisition, transmission, and processing. Figure 1 conceptually illustrates the usage of this architecture. The physical infrastructure consists of sensor platforms, forwarding devices, and servers. On top of this infrastructure, multiple urban sensing applications are deployed, and each application customizes its data acquisition, transmission, and processing through the service APIs. The standard service API reduces the complexity and developing cycle for deploying a new application, while the sharing of physical infrastructure greatly reduces the capital and OAM (operation, administration and maintenance) costs. These characteristics empower SD-IoT to efficiently support various application requirements and thus enable smart urban sensing.

In the rest of this paper, we first analyze current urban sensing applications along with the problems and trends. We then introduce the proposed software-defined IoT architecture, followed by open problems and potential solutions. After that, we present a quantitative analysis to show the benefits of SD-IoT. We then conclude the paper.

IoT: State of the Art and Trends and Solutions

Urban Sensing: State-of-the-Art

Figure 2 presents the three most common urban sensing applications: temperature, noise, and PM 2.5 monitoring. These applications can be logically divided into three subsystems: data acquisition, transmission, and processing. Specifically, different sensors are deployed in the city to obtain temperature, noise, and PM 2.5 data. The obtained data are then transmitted to remote servers to be stored/processed. Usually, the sensor node first transmits the data to a gateway through a wireless sensor network (WSN) [6]. The gateway then transmits data to the remote server through wireless or wired networks. Data processing may happen during the whole course, e.g. filtering undesired data at the sensor nodes [7], compressing and encrypting data at the gateway, further analyzing acquired data at the server to obtain the statistical information, etc.

Currently, an application-oriented approach is utilized to develop these three subsystems [11]. That is, application developers customize the sensor platform, gateway, network, and remote server from slate state, according to the application requirements. Specifically, the developers need to buy or develop a sensor platform according to the application requirements, which usually includes the sensors to obtain the required data, the radio modules to transmit the data, the power supply modules, and the microcontroller to coordinate the peripheral modules and execute data processing functions. The firmware also needs to be customized for this specific application. As an example, Downes *et al.* [8] introduced the design of a platform for wireless image sensor networks. Then the developers need to take a similar path to customize the network and computing infrastructure, e.g. to determine how to access the Internet, whether to cache the data or not, where to store and process the obtain data, etc.

Problems Analysis

While the above application-oriented approach seems quite direct, it has many drawbacks. We summarize them as follows:

High Capital and Maintenance Cost: As each application needs to deploy and manage its own sensor platforms, it requires a huge investment in hardware deployment and maintenance, while in fact it is possible for many applications to share sensor platforms if they require the same type of data. Even when the required data are different, many modules in a sensor platform, like the radio, power supply, and microcontroller, can be shared to reduce the overall cost.

Inflexible for Potential Application Changes: Under this approach, the infrastructure and the application are closely coupled, i.e. the intelligence of the application is hard-wired in the sensor platform, the gateway, and the server. Any change related to an application requires redeveloping or re-customizing the physical infrastructure, which is complex, error-prone, and sometimes even impractical.

Inefficient Resource Usage: As the control logic of applications is embedded in hardware devices, it is difficult to improve resource utilization by dynamically optimizing data acquisition, transmission, and processing. For example, as there is no approach to dynamically control data collection and transmission in sensor platforms, they would continuously transmit the data to remote servers, even though such data is undesired during some time periods, and thus the energy of sensor platforms and the bandwidth of the network are wasted.

Long Development and Deployment Cycle: As each application needs to develop and deploy its own sensor platform, gateway, and remote server from scratch, the overall time to introduce a new application is long. The long development and

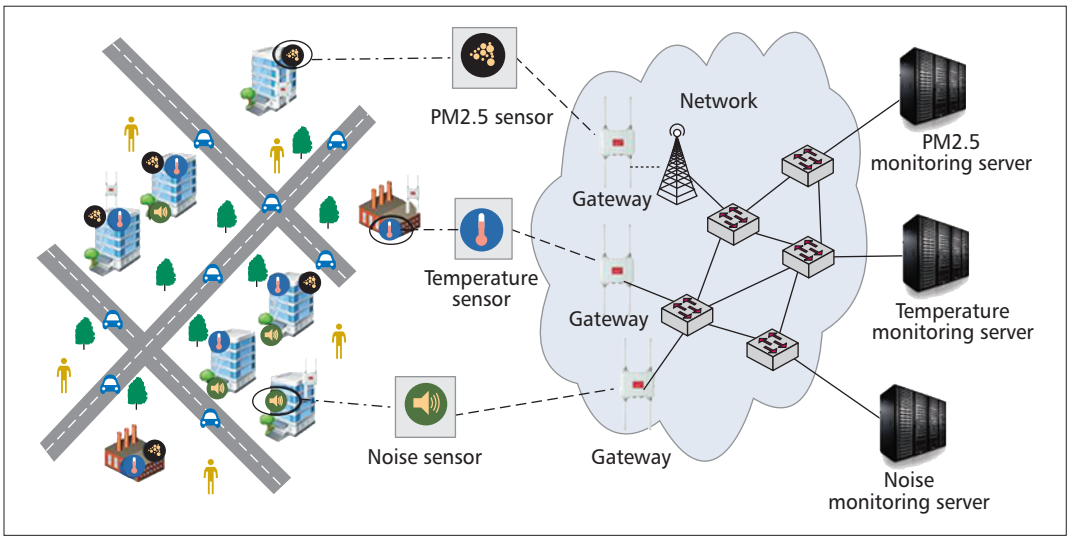


Figure 2. Illustration of urban sensing applications, which include three subsystems of data acquisition, transmission and processing.

With the increase in users and applications, Internet architecture and related infrastructure are also evolving in order to deal with encountered challenges. There are some apparent trends in this evolution, which should be considered in the design of IoT architecture.

deployment cycle, as well as the high investment required, definitely increase the barrier for deploying new applications and thus prevent potential innovations in applications.

The Trends

With the increase in users and applications, Internet architecture and related infrastructure are also evolving in order to deal with encountered challenges. There are some apparent trends in this evolution, which should be considered in the design of IoT architecture.

The Sharing of Physical Infrastructure: Sharing means that the underlying physical infrastructure simultaneously supports multiple applications belonging to multiple parties. The popularity of cloud computing best explains the trend of sharing physical infrastructure. Through cloud computing, the application developers deploy their applications in cloud data centers rather than build their own physical infrastructures. In addition to cloud computing, there is also a trend of sharing network infrastructures, such as base stations, access points, etc. As the sharing of physical infrastructure has one general benefit of reducing capital and maintenance costs, we envision that the IoT infrastructure should be shared to obtain this benefit.

The Rising of Software-Defined Architecture: SDN enables flexible network control by separating the control plane and the data plane. Inspired by these benefits, SDN has been extended to mobile access networks [9] and wireless sensor networks [10, 11]. As the physical infrastructure will become increasingly complex in the era of IoT, it is necessary to borrow the insight of SDN to realize flexible control and management of IoT infrastructure.

The Prevalence of Application Programming Interfaces: Providing application programming interfaces (API) is a growing trend to share physical infrastructure. Cloud providers like Google APP Engine have offered such APIs, and network controllers like OpenDaylight [12] also provide northbound APIs to develop control applications. In addition to enabling the sharing of physical infrastructure, APIs also hide the complexity and heterogeneity of the physical infrastructure, which significantly reduces the difficulty of application development and shortens the time to market of new applications. This trend suggests that IoT, especially sensor platforms, should provide APIs for applications to exploit their abilities in a flexible and efficient manner.

SD-IoT: Architecture Overview and System Design

Architecture Overview

In this paper we propose a software defined IoT (SD-IoT) architecture. As illustrated in Fig. 3, SD-IoT consists of three layers: a physical infrastructure layer, a control layer, and an application layer.

Physical Infrastructure Layer: This layer is composed of various kinds of physical devices, including sensor platforms, gateways, base stations, switches/routers, and servers. These devices possess the essential functions and resources to sense an urban environment, transmit data from one node to another, and process them to extract required information. However, they do not determine what to do by themselves. Instead, they leave the decision-making to the control layer by interacting with it through standard interfaces, i.e. a southbound interface named in SDN.

Control Layer: The control layer acts as the intermediary between the infrastructure layer and the application layer. On one hand, the control layer manages the physical devices with various characteristics and functions through different southbound interfaces. On the other hand, the control layer provides services to the application layer through APIs known as northbound interfaces. For urban sensing applications, the control layer will provide data acquisition, transmission, and processing service. We will explain these services in detail in the following subsections.

	State of the art	Problems	The trends
Data acquisition	Application-oriented wireless sensor platforms. The control functions are preset in the firmware.	Hard to customize in run time. Hard to implement dynamic optimization. High capital and maintenance cost.	Over the air programming to update sensor firmware.
Data transmission	Distributed protocols, such as WiFi, ZigBee, TCP/IP. The control protocols embed in each forwarding device.	Hard to control and evolve. No QoS guarantee.	Software-defined network. Network as a service with QoS guarantee.
Data processing	Each application developing data processing pipelines from the scratch.	The time cycle to develop a new application is long. Hard to share data processing resources.	Cloud based data processing to provide various data processing software, platform, and tool.

Table 1. Summary of state of the art, problems and the trends.

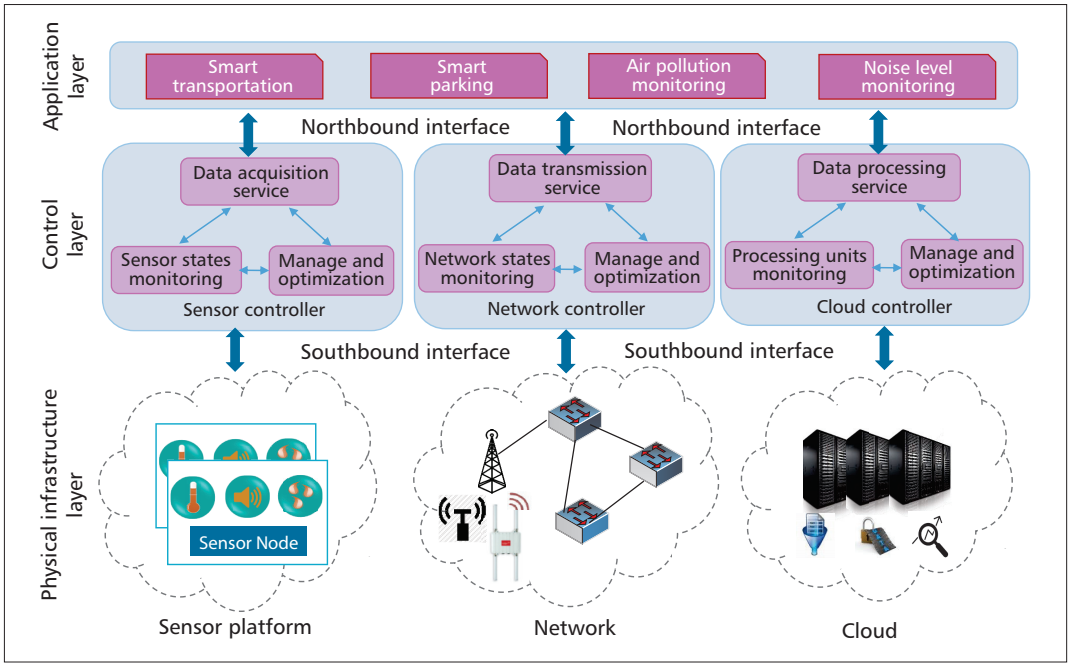


Figure 3. Architecture of software-defined IoT.

Application Layer: In this layer, developers build urban sensing applications using the provided APIs. In particular, they can customize data acquisition, transmission, and processing without worrying about the required change of configurations in physical devices, which greatly simplifies the procedure of developing new applications. Also, as the physical infrastructure is shared by multiple applications, the overall capital and maintenance costs are reduced.

Sensor Platform and Data Acquisition Service

The data acquisition service provides APIs for applications to specify their data requirements. The controller automatically configures sensor platforms to obtain the required data. Data specification includes general attributes, such as data type, targeted geographical areas, and time duration. For example, as shown in Fig. 4a, an application would request PM 2.5 data at Tsinghua University. Applications can also specify type-dependent attributes, e.g. sampling rate can be set for PM 2.5 and noise data. The data acquisition service also provides APIs for applications to query the properties of available data such as data types and geographical areas as well as optional attributes for each type of data.

Under SD-IoT, each sensor platform is equipped with more than one sensor with the same or different types and shared by many applications. For example, a sensor platform may include a PM 2.5 sensor and a noise sensor simultaneously, significantly reducing the total number of sensor platforms that need to be deployed. As a result, the overall investment for hardware, deployment, and maintenance is also reduced. The sensor controller has a global view of the underlying sensor platforms. Specifically, it knows the location and embedded sensors of every sensor platform. Based on the global view, the sensor controller can dynamically activate/deactivate sensors and customize their configurations to satisfy application requirements and simultaneously reduce energy consumption.

Network and Data Transmission Service

The network is used to transmit data from the sensor platforms to servers in the cloud. As applications may prefer different cloud data centers, they should have the ability to specify the destination of data transmission. Also, applica-

tions may have specific performance requirement for data transmission. For example, a smart transportation application that provides path planning suggestions must be aware of current traffic load, and thus requires low latency data transmission. In contrast, a video application that provides real time street views must guarantee that the video is fluently transmitted, and thus requires reservation of bandwidth.

The data transmission service provides APIs for applications to specify their requirements, which mainly include two dimensions: destination and QoS parameters. An IP address can be used to specify destination, while several options can be provided for QoS specification: basic transmission, latency sensitive transmission, and bandwidth guaranteed transmission.

Generally, basic transmission is carried in a best-effort manner, latency sensitive transmission has high priority during traffic scheduling, and the controller reserves bandwidth for bandwidth guaranteed transmission. In addition, with the advance of network function virtualization (NFV), the network will also provide on-path data processing services, e.g. data compressing and encryption. Specifically, the data transmission service API will also allow applications to specify the service chain [13], i.e. the pipelines of virtual network functions that a specific flow needs to go through. Figure 4b illustrates two examples of requests for data transmission service.

To realize the data transmission service, the network follows a software-defined network architecture. The forwarding devices are programmable, e.g. OpenFlow-enabled, and the controller is responsible for implementing traffic steering and scheduling. Specifically, based on the collected global network view, the controller steers packets to different destinations, and dynamically schedules traffic to satisfy application requirements for network quality and optimize the usage of network resource.

Cloud Data Center and Data Processing Service

Urban sensing data is further stored and processed using resources provided by three main cloud computing models: IaaS, SaaS, and PaaS. Currently, a cloud usually uses one of these for service provisioning. However, as the application of urban sensing would require them simultaneously, we argue that they should be integrated to support data processing services, i.e. a cloud should simultaneously provide software service, platform service, and infrastructure service, and offer APIs for users to flexibly utilize them together. Figure 4c illustrates two examples. One application requires mining the received data in real time and exploiting visualization software to illustrate the mining results. Thus, it requires a data mining platform and visualization software, which can be provided by PaaS and SaaS, respectively. Another application aims to store the received data at first and then exploit their own programs for data processing. Thus, it needs storage and VMs, which can be provided by IaaS.

Data processing service APIs allow applications to specify the required resources, which includes running submitted programs on specific platforms, deploying existing software entities, and providing VMs. The cloud controller knows the state of the underlying server resource pools, such as which servers are used to support a specific platform and the residual resource in them, and maps the application's resource request to underlying server pools based on it.

Open Problems and Potential Solutions

Southbound Interface Design

To implement SD-IoT, the southbound interface should be designed for controllers to interact with the physical infrastructure. Some interfaces have been designed to address the challenge. For forwarding devices, OpenFlow is the most widely used interface that abstracts the forwarding behavior in heterogeneous switches and routers. For servers, the interface usually depends on the cloud control system. However, in general, middleware software applications are utilized to deal with device heterogeneity. Compared to forwarding devices and servers, designing a southbound interface for sensor platforms is much more difficult due to higher device heterogeneity. Besides, sensor platforms are energy limited and thus the energy consumption for interacting with the controller should be reduced as much as possible.

As an initial idea for designing a southbound interface for sensor platforms, we propose to combine the strategy of abstraction and middleware software. First, by providing an abstraction on the data collection, processing, and transmission procedure in the sensor platform, the implementation of the sensor controller is decoupled from sensor platforms. Galluccio *et al.* [11] has proposed a finite state machine based abstraction for data processing and transmission. In Fig. 4a we also show an example to abstract the data collection ability based on the included sensors and their types and IDs. Second, before the standardization of the abstraction, the actual control interface of different sensor platforms varies across each other. Middleware software then can be exploited to carry out the transformation. Particularly, to save energy at the sensor platform, middleware software can be placed in the controller, and the controller should decrease the frequency of interactions with sensor platforms when they are inactive. Despite having these benefits, the design and implementation of the abstraction and middleware software needs more discussion and study.

Control Layer Design

The design of a logically centralized control layer should achieve three objectives: high scalability, high performance, and high robustness. First, as more and more physical devices and applications will be added over time, the control layer should scale at the same time to support them. Besides, in SD-IoT, application performance and control flexibility depends on the performance of the interaction between the control layer and the physical layer, e.g. communication delay. Further, the control layer must be robust enough to work normally under various possible failures.

Deploying multiple controllers is a general approach to achieve these objectives. On one hand, the controller can be replicated to increase

The cloud controller knows the state of the underlying server resource pools, such as which servers are used to support a specific platform and the residual resource in them, and maps the application' s resource request to underlying server pools based on it.

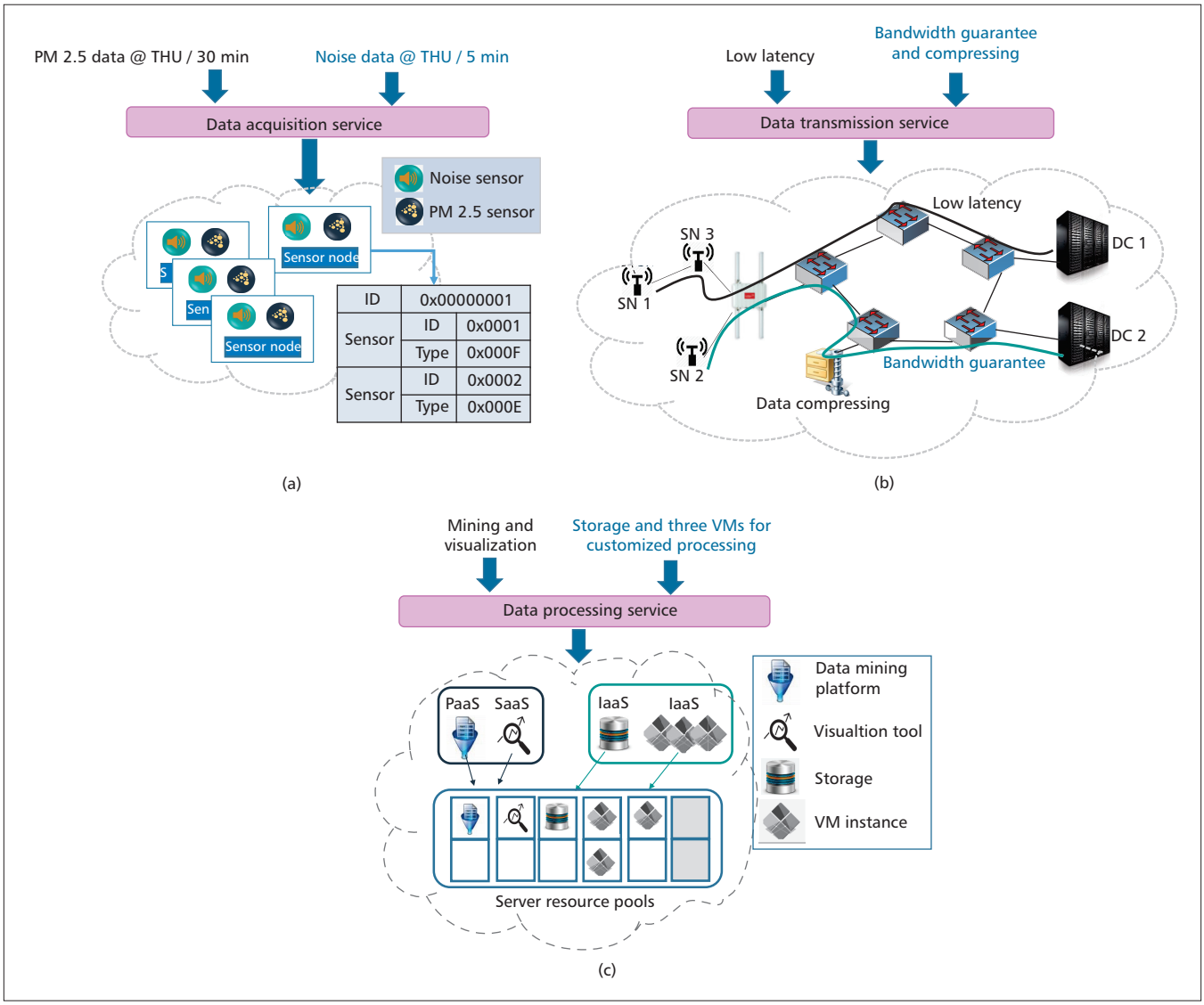


Figure 4. Illustration of data acquisition, transmission and processing services:a) data acquisition service; b) data transmission service; and c) data processing service.

its robustness. On the other hand, each controller can manage only part of the devices and thus the control layer can be scaled by increasing the number of controllers. Further, the controllers can be placed at different locations to reduce the average communication delay to the physical devices. Ahmed and Boutaba [14] proposed to exploit a vertical approach to organize multiple controllers for software defined wide area networks, which can also be extended to the control of sensor platforms and clouds. For example, we can use one controller to control one WSN and use an orchestration controller to coordinate them.

Mobility Management

In SD-IoT, the control plane needs to implement mobility management to support the mobile sensor platform handover from one gateway to another. Particularly, as the gateways are controlled by multiple physical controllers for scalability, the control plane needs to coordinate these controllers to implement the mobility management function.

Wu *et al.* [15] have introduced a solution to mobility management by maintaining a controller network based on structured overlay when the physical controllers are distributed. In addition, we can also employ an orchestration controller to coordinate the mobility management by recording the controller currently managing each mobile sensor platform. When a mobile sensor platform attaches to the gateway managed by a new controller, the controller reports the event to the orchestration controller, which then coordinates the original controller and the new controller to carry out the handover.

Conflict Resolution and Optimization for the Sensor Platform

Under SD-IoT, the sensor platforms are shared by different applications, which may lead to potential configuration conflicts. For example, one application may request noise data in Tsinghua with a sampling rate of once every five minutes, while another application may also

request the noise data in Tsinghua but with a different sampling rate. When conflicts happen, the sensor controller needs to decide whether to accept the application's request, how to resolve the conflicts, and simultaneously minimize the sensor platform's energy consumption.

To avoid the conflicts, the controller can allocate each sensor to at most one application. While this strategy is simple, it is inefficient because sometimes the sensor can still be shared by applications with different settings. For example, if Application A wants to set the sampling rate to once every five minutes, while Application B wants to set it to once every seven minutes, the controller can make Application A and Application B share the sensor by generating a series of sampling time points, e.g. five minutes, seven minutes, 10 minutes, 14 minutes, 15 minuntes, etc. To extend this approach to a general case, the controller sets up a resolver for each sensor. The resolver records the sensor's current configuration. After receiving a service request and transforming it to the configuration of the sensor, the controller sends the configuration to the corresponding resolver. The resolver generates appropriate configurations according to the new configuration and current configuration, or it will reject the configuration request if there are unresolvable conflicts.

QoS Enabled Traffic Scheduling

In SD-IoT, the network controller can provide end-to-end QoS guaranteed data transmission. However, several challenges need to be addressed to achieve this. First, the forwarding devices have a limited number of queues for QoS enforcement, and thus it is difficult for them to support a huge number of QoS requirements. Second, it is a challenge to design efficient traffic scheduling algorithms to satisfy QoS requirements in a large-scale network.

We now explain two strategies to deal with the above challenges. The first is reducing the demand for queues in forwarding devices by quantization of QoS requirements, which can be conducted based on statistics of application requirements. The second is considering the number of available queues in each forwarding device when scheduling traffic. As this strategy further increases the complexity of the traffic scheduling problem, some approximate algorithms should be developed to solve it efficiently.

Resource Mapping in Cloud Data Centers

In SD-IoT, the cloud controller needs to decide how to map application service requests to the physical devices. For example, considering that the cloud allows applications to store data at first and then rent VMs to process them, the cloud controller needs to decide where to store the data and which servers should be used to host VMs for post-data processing. Generally, several objectives are expected, such as increasing the cloud provider's revenue by accepting more service requests, saving energy by using less servers, and balancing the server's load by equally mapping service requests to different servers. The constraints include server capacity, storage capacity, and the type of softwares/platforms/VMs a server can host.

One challenge to implement the above optimizations is that sometimes different objectives are in conflict with each other, and hence it is impossible to achieve them simultaneously. As an example, energy saving and load balancing are conflicting objectives. Efficient heuristic or approximate algorithms should be developed to achieve a trade-off between conflicting objectives. As an example, a threshold-based strategy to activate and shut down servers can be an effective trade-off between load balancing and energy saving, i.e. activate more servers when the average workload exceeds a pre-set maximum, and shut down some servers when the workload goes below a pre-set minimum.

Case Study and Quantitative Analysis

Selected Scenario

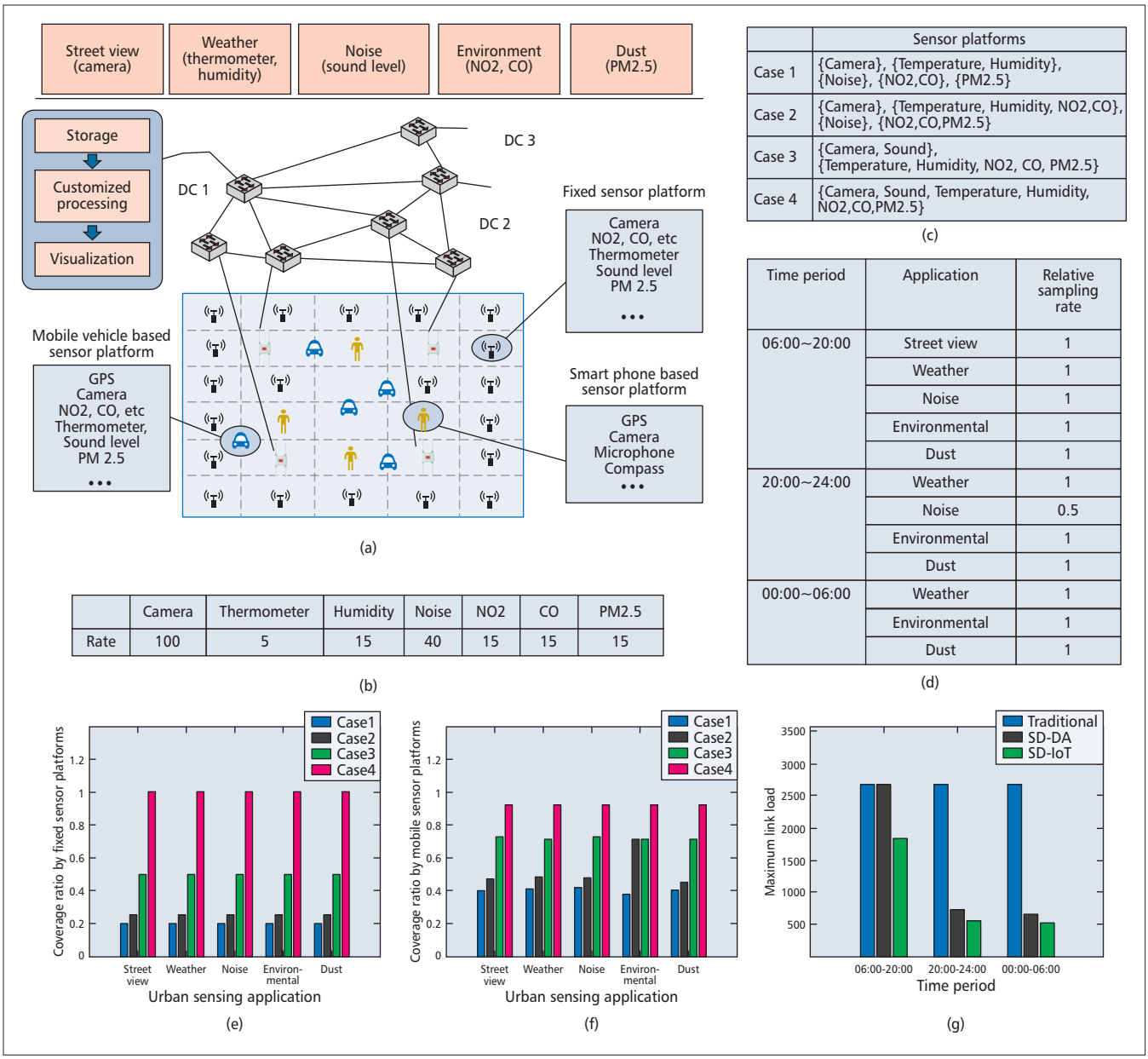
In order to further illustrate the benefits of SD-IoT, we conducted a case study and quantitative analysis, described in this section. Figure 5a illustrates the selected scenario. The target region consists of 5x6 rectangular urban areas. There are three types of sensor platforms: fixed sensor platform, user smart phone based sensor platform, and mobile vehicle based sensor platform. We considered five urban sensing applications: street view, weather monitoring, noise monitoring, environmental monitoring, and dust monitoring. The essential sensors to support each application are shown in the figure. We assume that each rectangle and vehicle can deploy at most one sensor platform. During a time period, each vehicle has a constant probability to appear in one specific rectangular area. In our simulation there are 2500 vehicles, and the constant probability is set to 1/1000. Further, there are three data centers connected by a network of seven forwarding devices. The data rate of each sensor under a standard sampling rate is shown in Figure 5b. The five applications employ a similar data processing procedure. Data is stored at first; then, VMs are used to execute customized processing; finally, the visualization software applications are exploited to illustrate the processing results.

Quantitative Analysis

Data Acquisition: To illustrate the benefits of SD-IoT, we consider four deployment cases, which are illustrated in Figure 5c. The first case reflects current IoT architecture, where there are five sensor platforms, each embeding sensors required by corresponding applications. The other three cases reflect the proposed SD-IoT architecture, where some sensor platforms embed more sensors and are shared by different applications. For example, in case 3 the sensor platform equipped with camera and sound level sensors can be shared by street view and noise monitoring applications. In each deployment case, every rectangular area and participating vehicle randomly selects one sensor platform to deploy.

We then observe the average coverage ratio of each application, which is defined as the ratio of areas covered by sensors of that application over the total target area. We show the results in

When a mobile sensor platform attaches to the gateway managed by a new controller, the controller reports the event to the orchestration controller, which then coordinates the original controller and the new controller to carry out the handover.



	Camera	Thermometer	Humidity	Noise	NO2	CO	PM2.5
Rate	100	5	15	40	15	15	15

	Sensor platforms
Case 1	{Camera}, {Temperature, Humidity}, {Noise}, {NO2,CO}, {PM2.5}
Case 2	{Camera}, {Temperature, Humidity, NO2,CO}, {Noise}, {NO2,CO,PM2.5}
Case 3	{Camera, Sound}, {Temperature, Humidity, NO2, CO, PM2.5}
Case 4	{Camera, Sound, Temperature, Humidity, NO2,CO,PM2.5}

Time period	Application	Relative sampling rate
06:00~20:00	Street view	1
	Weather	1
	Noise	1
	Environmental	1
	Dust	1
20:00~24:00	Weather	1
	Noise	0.5
	Environmental	1
	Dust	1
00:00~06:00	Weather	1
	Environmental	1
	Dust	1

Figure 5. Quantitative analysis scenario and results: a) selected scenario; b) data rate of each sensor under standard sampling rate; c) sensor platforms exploited in four deployment cases; d) the relative sampling rate (compared to standard sampling rate) of each application during different time periods; e) the coverage ratio by fixed sensor platforms with four deployment cases; f) the coverage ratio by vehicle based sensor platforms with four deployment cases; and g) the maximum link load during different time periods under three different scenarios.

Fig. 5e and Fig. 5f. From the figure, we can find that the coverage ratio of each application increases from case 1 to case 4. For example, in Fig. 5f, when sensor platforms in each vehicle can be used by one application, as case 1 indicates, the coverage ratio is only about 40 percent. However, if each sensor platform can be shared by five applications, as case 4 indicates, the coverage ratio increases to 90 percent. Such enhancement is due to the increase in the number of sensor platforms that can be used by each application. As the cost to deploy a sensor platform with seven sensors is not much higher than the cost with fewer sensors, these results suggest that it is efficient for different applications to share the underlying sensor platforms.

Data Transmission: We assume there are four gateways deployed in the target area. The sensor platform first transmits the data to the nearest gateway, then the gateway transmits the data to the corresponding data center. In Fig. 5d we show the sampling rate of each application at different times of day. We consider three scenarios. The first scenario corresponds to the current IoT architecture, and we use "traditional" to refer to it, where the configuration of sensor platforms is unchangeable and the network uses the shortest path to transmit data from the gateway to the data center. The second scenario is denoted by "SD-DA," where the configuration of sensor platforms can be dynamically changed, but the network still uses the shortest path for

data transmission. The third scenario, referred as "SD-IoT," corresponds to the proposed software-defined IoT architecture, where both the sensor platform and network are software defined and thus can be exploited to dynamically optimize data transmission.

Figure 5g shows the maximum link load under the above three scenarios. For the traditional architecture, the maximum link load remains the same over time because the configuration of each sensor platform is fixed and thus the total data rate does not change. In contrast, the SD-DA scenario reduces the maximum link load at night, since it can switch off sensors when not required or lower their sampling rate to reduce the total data rate.

Moreover, as software defined networks enable dynamic and global optimization of traffic forwarding, SD-IoT further decreases the maximum link load by distributing the traffic equally over multi paths. Specifically, the maximum link load is reduced by 32 percent, 25 percent, and 22.6 percent during the time periods 6:00-20:00, 20:00-24:00, and 00:00-06:00, respectively.

Data Processing: In the selected scenario, each application requires data storage, the platforms, or VMs to execute customized data processing, and visualization software to illustrate the data processing results. Currently, these resources can only be separately provided by different cloud data centers. Therefore, the data processing procedure needs to traverse multiple data centers, which results in significant overhead on the network, and non-ignorable increase in the delay of data processing. In contrast, the proposed SD-IoT aims to provide these resources in the same data center to reduce the delay and mitigate the overhead on the network.

Conclusion

This article focused on the design of a flexible IoT architecture for smart urban sensing. Specifically, we proposed a software-defined IoT architecture that decouples the applications from underlying physical infrastructures. With this architecture, urban sensing applications can customize their own data acquisition, transmission, and processing through well-defined APIs, and multiple applications coexist on the shared infrastructure to further reduce the overall capital and maintenance cost. As a result, this architecture enables flexible control and management of physical infrastructure, and accelerates application innovation.

References

[1] M. Naphade *et al.*, "Smarter Cities and Their Innovation Challenges," *Computer*, vol. 44, no. 6, 2011, pp. 32–39.
[2] D. Boyle *et al.*, "Urban Sensor Data Streams: London 2013," *IEEE Internet Comp.*, vol. 17, no. 6, 2013, pp. 12–20.
[3] A. Gluhak *et al.*, "A Survey on Facilities for Experimental Internet of Things Research," *IEEE Commun. Mag.*, vol. 49, no. 11, 2011, pp. 58–67.
[4] H. Ma, D. Zhao, and P. Yuan, "Opportunities in Mobile Crowd Sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, 2014, pp. 29–35.
[5] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, 2008, pp. 69–74.
[6] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks*, vol. 52, no. 12, 2008, pp. 2292–330.
[7] A. Papageorgiou *et al.*, "Smart M2M Data Filtering Using Domain-Specific Threholds in Domain-Agnostic Platforms," *Proc. IEEE BigData Congress*, 2013, pp. 286–93.
[8] I. Downes, L. B. Rad, and H. Aghajan, "Development of a Mote for Wireless Image Sensor Networks," *Proc. COGIS'06*, 2006.
[9] A. Gudipati *et al.*, "SoftRAN: Software Defined Radio Access Network," *Proc. 2nd ACM HotSDN*, 2013, pp. 25–30.
[10] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Commun. Lett.*, vol. 16, no. 11, 2012, pp. 1896–99.
[11] L. Galluccio *et al.*, "SDN-WISE: Design, Prototyping and Experimentation of a Stateful SDN Solution for Wireless Sensor Networks," *Proc. Infocom*, 2015, pp. 513–21.
[12] J. Medved *et al.*, "Opendaylight: Towards a Model-Driven SDN Controller Architecture," *Proc. 15th IEEE WoWMoW*, 2014, pp. 1–6.
[13] Z. A. Qazi *et al.*, "Simple-Fying Middlebox Policy Enforcement Using SDN," *ACM SIGCOMM CCR*, vol. 43, no. 4, 2013, pp. 27–38.
[14] R. Ahmed and R. Boutaba, "Design Considerations for Managing Wide Area Software Defined Networks," *IEEE Commun. Mag.*, vol. 52, no. 7, 2014, pp. 116–23.
[15] D. Wu *et al.*, "UbiFlow: Mobility Management in Urban-Scale Software Defined IoT," *Proc. Infocom*, 2015, pp. 208–16.

Biographies

Jiaqiang Liu received his B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012, and is now a Ph.D. student at Tsinghua University. His research interests include software defined network, data center network, network function virtualization, and Internet of Things, etc.

Yong Li [M'09] received the B.S. and Ph.D degree from Huazhong University of Science and Technology and Tsinghua University in 2007 and 2012, respectively. During 2012 and 2013 he was a visiting research associate with Telekom Innovation Laboratories and Hong Kong University of Science and Technology, respectively. From 2013 to 2014 he was a visiting scientist with the University of Miami. He is currently a faculty member in the Department of Electronic Engineering, Tsinghua University. His research interests are in the areas of mobile computing and social networks, urban computing and vehicular networks, and network science and future Internet. Dr. Li has served as general chair, Technical Program Committee (TPC) chair, and TPC member for several international workshops and conferences. He is currently an associate editor of the *Journal of Communications and Networking* and *EURASIP Journal of Wireless Communications and Networking*.

Min Chen [SM'09] (minchen@ieee.org) is a professor at the School of Computer Science and Technology at HUST. He was an assistant professor in the School of Computer Science and Engineering at SNU from September 2009 to February 2012. He worked as a post-doctoral fellow in the Department of Electrical and Computer Engineering at UBC for three years. Before joining UBC he was a postdoctoral fellow at SNU for one and a half years. His research focuses on Internet of Things, machine-to machine communications, body area networks, body sensor networks, e-healthcare, mobile cloud computing, cloud-assisted mobile computing, ubiquitous networks and services, mobile agents, multimedia transmission over wireless networks, and so on.

Wenxia Dong received the master's degree from Southwest Jiaotong University in 2008. She then joined Huawei, where she is now a network research engineer. Her research interests are in the areas of vehicular networks, SDN northbound interface, social networks, and application of big data in networks. She has 13 patents as the first author.

Depeng Jin received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1995 and 1999, respectively, both in electronics engineering. He is a professor at Tsinghua University and the chair of the Department of Electronic Engineering. Dr. Jin was awarded the National Scientific and Technological Innovation Prize (second class) in 2002. His research fields include telecommunications, high-speed networks, ASIC design, and future Internet architecture.

With the proposed architecture, urban sensing applications can customize their own data acquisition, transmission, and processing through well-defined APIs, and multiple applications coexist on the shared infrastructure to further reduce the overall capital and maintenance cost.