# Toward Smart Urban Development Through Intelligent Edge Analytics


**Mahmoud Abu Zaid, Mohamed Faizal, R. Maheswar, and Osamah Ibrahiem Abdullaziz**

## 1 Introduction

Urban population of the world has seen a rapid growth, from 751 million in 1950 to 4.2 billion in 2018 [1]. According to UN 55% of all world's population lives in urban areas and this figure is predicted to go up, by as much as 68% by 2050. The net effect of urbanization according to projections in the growth of global population is another 2.5 billion people will be added to urban areas by 2050, with close to 90% of this demographic expected to be in Asia and Africa [1]. All of this means the world has been becoming urban and the trend is poised to continue in the future.

An increase in urban population comes with its own set of associated challenges in several areas, some of which include: an increase in environmental pollution, managing increasingly complex transportation system, making healthcare accessible for growing population, making government services accessible to all citizens, and providing safety and security to all population. All of these has led to the increase in intelligence in cities and a trend toward smart urban development by taking advantage of existing IoT technologies to mitigate most of the abovementioned issues; we will explore a few solutions in this chapter.

M. A. Zaid · O. I. Abdullaziz
Department of Electrical Engineering & Computer Science, National Chiao Tung University, Hsinchu City, Taiwan
e-mail: abozedmn.03g@g2.nctu.edu.tw; yabolahan.04g@g2.nctu.edu.tw

M. Faizal (✉)
Department of Electronics Engineering, National Chiao Tung University, Hsinchu City, Taiwan
e-mail: mohamedfaizal.ee05g@nctu.edu.tw

R. Maheswar
School of Electrical & Electronics Engineering (SEEE), VIT Bhopal University, Bhopal, Madhya Pradesh, India



S. Rani et al. (eds.), *Integration of WSN and IoT for Smart Cities*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-38516-3_8

The current technological framework is primarily based upon cloud and local computing with increasing reliance on edge and fog in the recent years. With the expected growth in gathering large amounts of data ("big data") in highly populated urban areas, cloud-based technologies suffer from major shortcomings, which we will highlight in this chapter; we will also present the technological trends to mitigate the same.

In moving toward the next generation of innovative and more capable applications, IoT big data analytics comes as an important cornerstone. With the current wave of the rapidly increasing data volumes, which are generated by many sensors, actuators, and devices, the need for decision-making and extracting knowledge out of this data is a necessity. IoT big data analytics plays an important role in achieving the same. The real-time response requirement stated by emerging applications, such as connected vehicles, arises a new challenge. Therefore, edge computing came into the picture to overcome the delay of processing the data entirely on the cloud. In this chapter, we show the current design approaches, protocols, and technologies that are being proposed for IoT big data analytics on both the cloud and the edge in the context of smart urban development. Also, we present some of the use cases that fit the context of smart cities and urban development.

With the recent boom in IoT and related technologies the trend in urban development has been toward increasing intelligence, i.e., it's common to see "smart cities" where sensors are deployed in key areas and the data collected is processed using intelligence analytics. Efficient machine learning algorithms are used to enable previously unavailable services and making available services more accessible; this is the whole paradigm of smart city development [2].

More recently with the explosion in the fields of IoT, big data analytics, and artificial intelligence, there is a convergence among all these fields in the context of urban development; this paradigm shift has led to what we call "cognitive smart cities" [3]. All of this will be further elaborated in this chapter.

## 2 Edge Networking for Internet of Things

Wireless networks will eventually become key enablers of ultra-reliable and low-latency applications. The driving force for wireless systems is the demand for high-quality applications which they provide to users. While industry pilots such as automotive, intelligent automation, telemedicine, and entertainment applications offer new opportunities for operators, they present new challenges in terms of reliability, latency, and cost requirements. For example, augmented reality applications will change the entertainment industry as they improve the user experience through realism. Providing a realistic and user-friendly experience requires minimal round-trip time to act and react [4].

To meet these challenging needs, edge computing and network function virtualization (NFV) has become a solution for bringing cloud services to the proximity of the user. On the one hand, multiple access edge computing (MEC) and fog

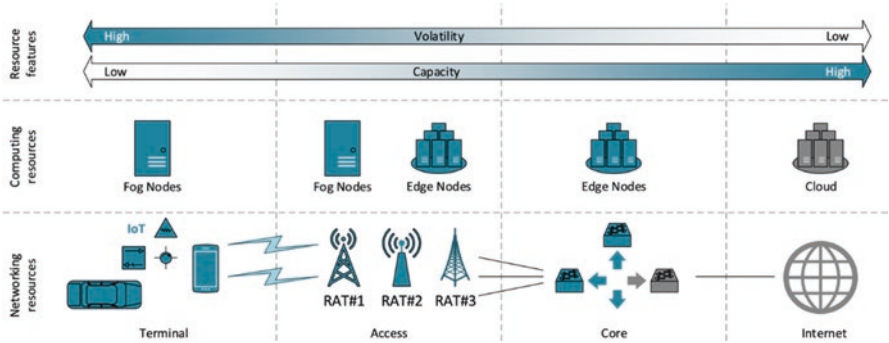


**Fig. 1** Cloud, edge, and fog resources and characteristics [6]

virtualize network edge applications, reducing end-to-end latency. On the other hand, NFV separates network functions and applications from the underlying hardware and allows them to be implemented as software. Mobile services can now be deployed in a distributed manner independent of hardware, and software-defined networking (SDN) can provide a dynamic responsive network for new services. SDN decouples the control and data planes, and thus it benefits MEC and NFV by simplifying management, enabling programmability, and enhancing performance.

These technologies work together to provide operators with effective means to coordinate and scale their infrastructures. An innovative joint of these technologies is presented by 5G-CORAL [5] which exploits edge solutions to provide low-latency and enhanced QoS across multiple-RAT environment (see Fig. 1). It adopts ETSI NFV and MEC standards as well as considers mobile and volatile resources.

## *2.1 Edge and Fog Computing System*

The EFS is a logical system comprised of fog and edge resources belonging to an administrative domain. An administrative domain is a collection of isolated resources managed by a single organization. The EFS virtualizes functions and applications and can interact with EFS in other domains. The software part of EFS consists of the following:

- *Function* is a virtualized instance deployed within the EFS for networking purposes.
- *Application* is a virtualized instance deployed within the EFS for serving end users and third parties.
- *Service platform* is a data storage for telemetric data collected from the EFS environment.
- *Entity manager* is responsible for applying configuration and management policies on the EFS elements as specified by the OCS. Compared to NFV and MEC

standards, the entity manager of the EFS service platform plays the role of the MEC platform manager. For more details, please refer to 5G-CORAL deliverable D2.1 [6].

5G-CORAL considers three types of the computing layers in the continuum between user devices and core network: fog, edge, and cloud. The most commonly known computing layer today is the cloud. Cloud is a remote high-powered data center which offers virtualized computing, storage, and networking services to businesses and end users. Lately, edge (i.e., MEC) and fog emerged aiming at moving the virtualized resource closer to IoT and end users to reduce the latency. While edge architecture mainly focuses on the deployment of virtualized resources at the edge of operators' infrastructure (e.g., base station), fog architecture extends the reach of resources even closer to the user (e.g., home gateway).

## 2.2 Enabling Virtualization Technologies for IoT in the Edge

***Hypervisor-Based Virtualization*** it runs at the hardware level and provides independent and host-isolated virtual machines (VM). Each VM runs its own kernel and operating system (OS). Therefore, the hypervisor can create Windows guests on Linux host. However, isolation and host abstraction features come with a cost. Memory, disk, and CPU resources must be specified at runtime to execute VM kernel and OS. Also, hardware emulation is required for I/O operations. In the case of high-density virtualization, VM deployment becomes resource inefficient, especially for small edge and fog applications. One good example of hypervisor-based virtualization is kernel-based virtual machine (KVM).

***System-Based Containerization*** it isolates processes at the OS level and runs on top of the host kernel. There are two types of containers, namely, system container and application container. System containers (also known as machine containers) behave like a standalone Linux system. That is, the system container has its own root access, file system, memory, processes, and networking and can be rebooted independently from the host. While system containers are lightweight due to the absence of guest kernel and hardware emulation, they can only run on Linux host and are bonded to the host's kernel. Linux container (LXC/LXD) is an example of system-based containers.

***Application-Based Containerization*** it isolates an application from other applications running on top of shared kernel and shared OS. Because of sharing the same kernel and OS, application containers are lighter than system containers. The application container only encapsulates the necessary libraries, configurations, and dependencies needed to run the application. Therefore, its resource footprint is significantly lower than VM and system containers. This makes the instantiation of virtualized applications appropriate for IoT services [7]. A well-known example of an application container is Docker.

A. *State-of-the-Art Edge Network Solutions for IoT Applications*
B. *Access Migration*

In the standard IEEE 802.11, clients actively scan for available APs for association in a discovery phase. During the scanning process, the AP responding to the probe message becomes a candidate for the client. When the client selects an AP, the association occurs between the AP basic service set identifier (BSSID) and the client MAC address. During this process, the infrastructure cannot control client association decisions. In order to change the AP, the client initiates a handoff process, which takes approximately 2 seconds [8].

In order to minimize the reassociation time, many fast handoff schemes have been proposed in [9–13]. These techniques can be divided into a) scanning time minimization and b) authentication time minimization. In the process of minimizing the scanning time, the goal is to identify a target AP as soon as possible. For example, synchronization scan [9], intelligent channel scanning [10], neighboring graph [11], selective neighbor caching [12], AP prediction [13], and IEEE 802.11k are scanning time minimization techniques. In authentication time minimization, preauthentication [13] was presented and detailed in IEEE 802.11r. While the proposed technique can minimize the reassociation latency, they involve modification to client devices and require additional signaling. The fact that these techniques require changes to the client side challenges the idea of bring your own device, which states that the infrastructure must accommodate variety of user devices.

In order to move client-AP association decisions from the client to the infrastructure, a virtual access point (vAP) was introduced in [14]. A vAP is an abstraction of the network functions created to connect clients. Each client associates to a dedicated vAP with unique parameters. Based on [14], the client associates with the vAP and periodically receives beacons to know that it is still within the coverage of its AP. The received signal strength perceived by the client is encapsulated in the beacon so that neighboring APs can also learn clients signal strength. Each AP maintains two databases, namely, managed and monitored lists, which keep clients' signal strength. The managed database stores the signal levels of clients currently associated with the AP, while the monitored database stores the signal levels of clients that the AP can hear. Access migration occurs when a neighboring AP receives a beacon from the client with signal strength higher than the signal strength advertised by the serving AP. This way, the association decision is moved to the infrastructure. However, there are drawbacks. It is assumed that all APs operate on the same channel so they are able to hear the advertised signal level. This makes the solution impractical for large-scale deployment and frequency planning. In addition, since the management of the vAPs is in a distributed fashion, a global view is absent.

To advance the work presented in [14], a multichannel extension of vAP paradigm is proposed in [15]. In multichannel vAP deployment, the APs operate in different channels and communicate with each other to support client mobility. After a client connects to a vAP managed by physical AP, the AP monitors the client signal strength level. If the signal level reaches below a predefined threshold, the AP sends a scan request to the neighboring APs. As soon as a neighboring AP responds

to the request, the client is instructed to switch channels and continue communicating with the new AP. This solution defeats the interference problem caused by operating on the same channel, but still remains a distributed solution without a global view.

On the other hand, an SDN-based WiFi framework dubbed Odin is introduced in [16]. Odin incorporates SDN solutions into vAP paradigm. In other words, the programmability and global view features of SDN are used to manage clients' mobility. The Odin framework is used to migrate vAPs from an AP to another while generating game traffic on the client side in [17]. While [16, 17] enable flexible and scalable management, they still consider that all APs are running in the same channel. Lately, an approach incorporating the advantages of SDN while also operating in a multichannel is considered [18, 19].

**Containerized Application Migration**

Service migration can be divided into stateful and stateless. In a stateless migration, the state of the application is not preserved when the service is relocated to the target host. In the case of stateful migration, the state of the application is maintained when the execution of the application is continued on the target host. There are three types of stateful migration techniques, stop and copy [19], pre-copy [20], and post-copy [21]. Stop and copy freezes the application, checkpoints its state, copies the application and its state to the target, and then resumes the application. Pre-copy executes iterative state checkpoint while the application continues to run and then terminates with a shorter stop and copy. Finally, post-copy performs a short stop and copy to relocate the important execution state, then resumes the application at the target, and retrieves the rest of the data as required.

VM live migration is well investigated [22] and many effective solutions are commercially available. For instance, a pre-copy-based VM live migration scheme is presented in [21]. An active VM continues to run in the course of in-memory data iterative pre-copying. During a consecutive iteration, only changes in memory (dirty pages) are transferred. At last, a final state copy is performed while the VM instance is frozen and then transferred to the destination host. This way, the amount of downtime is greatly reduced when compared to a pure stop-and-copy scheme. Although the work in VM migration is mature, most of the existing solutions are tailored for data center environment where network-attached storage (NAS) and specific virtualization technology are utilized. NAS enables all the host machines in a data center to access a network-shared storage which removes the need for migrating disk storage. However, in a scenario where migration takes place between MECs, state and local disk storage has to also migrate over wide area network (WAN).

Lately, container migration has caught much attention from the research community [23, 24], especially since containerization offers many advantages, in terms of resource efficiency and performance, over traditional hypervisor-based

virtualization. This fact enables the instantiation of lightweight containerized applications suitable for IoT services [25]. In [23], container migration mechanism is developed for power efficiency optimization in heterogeneous data center. This work assumes that the source and destination hosts have access to a NAS and thus container data is not copied over WAN.

Furthermore, a framework for migrating containerized applications is presented in [24]. The proposed framework is the first to consider MEC environment for container migration. Fundamentally, the framework is a layered model which aims to reduce the downtime incurred by the migration process. While the presented results show reduction in downtime as a result of layering, the framework relies on stop-and-copy migration which is not an efficient method for containers with large in-memory state. In our proposed solution, we develop a pre-copy procedure to migrate containerized applications between edge clouds.

**Mobility Support in Edge User Application**

C. *ARNAB Double-Tier Migration*

ARNAB [26] is a novel architecture which provides transparent service continuity through access and application migrations. The term ARNAB is an Arabic word which means rabbit. ARNAB is given for the architecture since the user service exhibits the rabbit behavior hopping through the WiFi infrastructure to support user mobility. Furthermore, ARNAB is said to be transparent since there is no modification to the user device required for its operation. The main objective of proposed architecture is to deliver seamless user experience. ARNAB utilizes double-tier migration, namely, user connectivity migration and application migration. The first migration scheme uses vAP to eliminate WiFi handoff delay and relocate the association decision-making to the infrastructure. The second tier uses iterative copying scheme to minimize the downtime during application migration.

D. *Follow Me Cloud*

Follow me cloud (FMC) [27] is a novel architecture that enables cloud services (i.e., running in distributed data centers) to follow the users as they roam through the network. The FMC controller manages computing and storage resources of the data centers and decides which data center the user should be associated with. Based on FMC, a migration mechanism is developed to ensure service low latency [28]. However, the minimum reported migration downtime remains high for seamless service experience.

E. *Follow Me Fog and sFog*

Follow me fog (FMF) and seamless fog (sFog) are proposed to pre-migrate computation jobs before radio handover occurs during user mobility. This is

accomplished by constantly monitoring the received signal strength indicator (RSSI) from different fog nodes. Once the RSSI of the current node (i.e., serving the user) keeps decreasing and the RSSI of another node keeps increasing, the computing jobs are pre-migrated to the new node before the reassociation takes place. This way, FMF and sFog support user mobility by predicting the target fog node beforehand and thus reducing the waiting time for computing jobs to be available.

F. *SharedMEC*

SharedMEC [29] is an architecture which combines the standard cellular handover process with service handover. In SharedMEC, an edge platform is shared by multiple femto base stations to support user mobility. The architecture employs an algorithm to decide when to migrate user services. In addition, an analytical model is proposed to analyze the total cost of migrating user service.

## 3 Big Data Enabling Technologies

Velocity (real-time collection), volumes (large amount), and variety (different kinds) are the three data characteristics that are usually associated with the definition of big data. Traditional SQL-based database management systems fail to store and manipulate such data. Therefore, NoSQL (not relational) databases came into the picture as a solution. In this section, we present some technologies that deal with the storage and the analysis of big data.

### *3.1 Storage*

#### MongoDB

MongoDB is a general-purpose, document-based, distributed database that is suitable for IoT application [30]. It provides an Intelligent Data Platform that supports IoT Apps from Edge to the core or the cloud. Figure 2 [31] shows the architecture of MongoDB. It also provides real-time and event processing.

#### CassandraDB

Apache Cassandra is an open-source NoSQL wide-column database. It adopts a data replication mechanism on a cluster of machines. Therefore, if one or more machines fail, based on the configuration of the replication factor, it still can provide the data with no data loss. Moreover, Cassandra allows adding/removing machines to existing clusters which permits scale up or scale down capability [32].

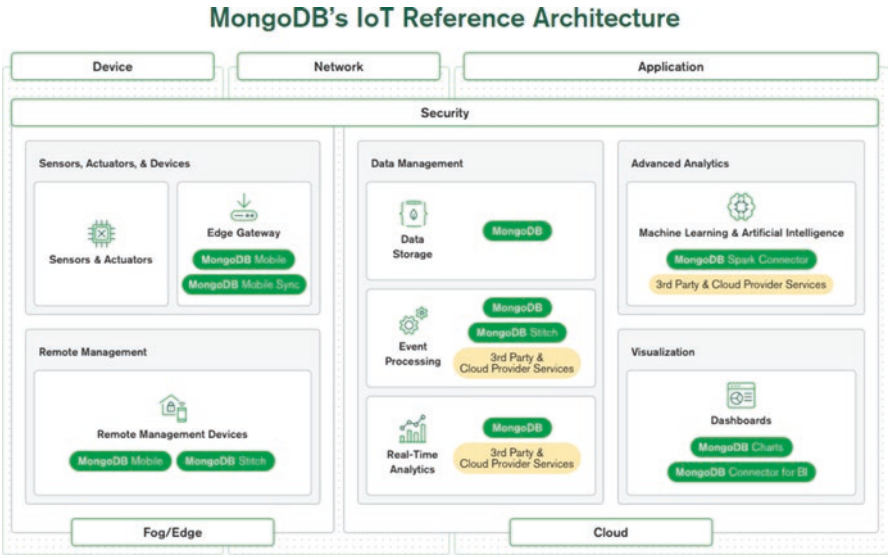


**Fig. 2** Fog/edge placement in IoT architecture

### HBase

Apache HBase is an open-source and distributed database that is created after Google's Bigtable. It presents Bigtable-like capabilities on top of Apache Hadoop and HDFS. Also, it provides a real-time read/write access and designed to host very large tables – billions of rows X millions of columns [33].

### OpenTSDB

It's very common in IoT solutions that several sensors generate data that monitor physical/cyber metrics over time (time series). One suitable way to store such data is the time series database OpenTSDB. Time Series Daemen (TSD) provides APIs which allows client applications to write and read data. TSD stores the time series data on HBase [34].

## *3.2 Analytics*

### Hadoop MapReduce

One of the most widely used analytic frameworks is Hadoop MapReduce. It is designed to execute batch processing analytics tasks on a large amount of data in a parallel manner. A typical Hadoop cluster could have thousands of machines that are running both the storage node (HDFS) and the analytical node (MapReduce) [35].

**Spark**

Unlike Hadoop, Apache Spark is designed to support real-time and stream processing. Also, it stores the data on Hadoop; therefore Spark has the ability to run batch processing as well. Spark provides rich features with different APIs that make it very suitable for running machine learning applications. For example, Spark supports applications written in Java, Python, and Scala [36].

## *3.3 IoT Protocols*

In this part, we demonstrate some of the used protocols, interfaces, and APIs in the realm of IoT, cloud, and big data analytics. Since the protocol stack from sensors to business in IoT is very wide, we selected the commonly used ones and applicable in the context of smart cities and urban development.

**REST**

One of the most commonly used web services is the representational state transfer (REST). As Wikipedia's definition, it is a software architectural technique that defines a set of constraints to be used for creating web services. Web services that are compliant to the REST architectural style provide interoperability between software component systems on the Internet using the http protocol. The exchanged messages could be in JSON or XML format [37].

**AMQP**

With heterogeneous platforms and systems in the IoT ecosystem. Connecting different systems with each other will become a challenge during the integration phase. Middleware technologies such as the Advanced Message Queuing Protocol (AMQP) could provide a standard way of getting the system connected. AMQP is an open standard for exchanging messages between different applications. Also, in a loosely coupled fashion, it connects systems, feeds business processes with the required information, and provides reliable transmission of the messages [38].

**MQTT**

In IoT architecture, one of the commonly used solutions to connect limited resources devices (sensors) to other elements, such as the gateway or the network server, is the MQ Telemetry Transport (MQTT). It plays the role of communication infrastructure where one device would publish the data to MQTT and another receiver would

subscribe to get it. MQTT is fitting for machine-to-machine (M2M) communication. Several products implement MQTT, for example, RabbitMQ, Mosquitos, and Erlang MQTT [39].

#### D2D

When we are considering edge computing in the fog devices, device-to-device (D2D) communication could play an important role as devices might hand off some tasks to a more capable device, to an idle device, or to a less loaded one. From another point of view, EC also could help in the utilization of the devices. Therefore, we present the D2D which is defined as direct communication between two mobile devices with no need for going through the base station (BS) or the core network [40].

The aforementioned technologies for both analyzing and storing big data are available as Docker containers. For example, Docker Hub [41] has images for MongoDB, Cassandra, HBase, OpenTSDB, and Spark. Also, a Docker image is available for [35]. Therefore, edge computing is achievable with the existence of these technologies.

### *3.4 Edge-Based and Cloud-Based Use Cases*

#### GeeLytics

Large-scale implementation of IoT solutions in different areas will lead to the need for real-time processing of stream data. Sensors like surveillance cameras, smartphones' cameras, and audio recording will generate a massive amount of streaming data. The proposed system in GeeLytics [42] attempts to provide a solution by exploiting the edge computing for processing the stream data. GeeLytics uses the cloud for offloading. Moreover, it takes into consideration the geographic location of the data stream sources and dynamically steers the processing of the tasks to the edge. These tasks could be depicted as isolated Docker containers. In summary, GeeLytics [42] could be used in different use cases, such as smart traffic, crowd prediction, and globalized smart city, which promises to provide the application with real-time processing of stream data.

#### Vehicular Fog Computing

Traffic management systems (TMS) play an important role in smart cities and urban development. However, it demands an ultralow latency for managing and monitoring the traffic. Edge computing can enable TMS to provide services that meet the aforementioned demands. The authors in [43] have proposed the vehicular

fog computing (VFC), which is a combination of exploiting fog computing (cloudlet layer) and vehicular networks. Cloudlet represents the grouping of some elements in the layer between the cloud and the vehicles. For example, routers, access points, and base stations are cloudlet layer's components. An important design principle of VFC is using both parked and moving vehicles as computing resources for data processing. Cloud computing is not totally abandoned, but it's being used for offloading. In order to minimize the response time, the authors in [43] proposed a VFC-enabled offloading algorithm for load-balancing optimization between the cloudlet fog layer and the vehicular network. The use case for the system was the real-world city map and routes of taxis in Shanghai, China. Compared to a random approach algorithm for offloading, the results show that the response time of the proposed solution was 0.6 second while the randomized approach was 4.2 seconds. In conclusion, with the use of edge computing and a novel load balancer algorithm, the system in [43] GeeLytics can provide a minimum response time for TMS.

**Recommender System**

IoT and big data analytics enable the development of smart and connected communities (SCC), where systems can make decisions such as reduce traffic congestion, fight crime, foster economic development, and manage the effects of a changing climate. The proposed architecture in [44] is composed of four layers. First is the sensing layer where data is being generated. Data sources are not only traditional sensors and open data but also personal smartphones' sensors as means of mobile crowdsensing (MCS). Second is the interconnecting layer which represents the communication infrastructure among all the layers. Third is the data layer which serves as the big data layer where data storage and analysis takes place. Finally, the fourth one is the service layer which has the APIs and the applications offered to the end users. In Trentino, Italy, the context-aware recommender system TreSight was implemented.

The user (tourist) will wear a bracelet IoT device and install the recommendation system on his/her smartphone. Points of interest locations in the city will have a HotSpot device that interacts with the bracelet, senses physical quantities from the environment, and provides the data to the cloud. From the connected bracelets to the HotSpot, the system can calculate the number of tourists in a particular location. The HotSpot sensors feed information about the temperature, humidity, and other physical quantities to the system. From OPenData Trentino, the system can gather weather information.

The system stores the data in MongoDB and uses Wi-Fare Cosmos for Big Data Analysis integrate with Hadoop. Based on the input data, the system makes decisions. Therefore, it can send certain knowledge to tourist-related service providers (restaurants, hotels, etc.) and recommend the user of his/her next place to visit [44].

**Digital Smart City**

The community expects smart cities to facilitate the citizens, enhance people's everyday activities, and help the authorities for better planning of the city and the provided services. To achieve that there is a need for a general system, which plays the role of an integration medium to link all existing IoT smart city systems. Things from smart homes, smart parking, vehicular networking, weather stations, and surveillance systems generate a massive amount of heterogeneous data. The goal is to connect all these systems; therefore, the proposed system in [36] suggests a central data hub for data collection from all sources. Then, it will send it to the cloud for processing.

On the cloud, Spark provides real-time processing, Hadoop for batch data processing, and Giraph over Hadoop for big city graph processing. The system will provide a set of APIs which allow the applications to consume the processed version of the data for further application-specific knowledge extraction.

## 4 Machine Learning for Smart Urban Development

The explosion in data gathering ability of cities itself is not useful, unless effective analytics are employed to extract meaningful information. Intelligent machine learning (ML) algorithms are applied on the collected data for intelligent insights. This is where the big data analytics techniques, introduced in previous section, come into play. The focus of this section is on new and efficient machine learning algorithms that have been designed to exploit the big data analytics to further enhance urban development.

When it comes to Internet of Things, we collectively refer to different sensors, actuators, and other smart objects that are essentially the "things" [2] used for data collection in smart cities. The data generated in smart cities come with a host of challenges, in terms of volume, velocity, and variety. The ML algorithms in the context of smart cities need to accommodate all the abovementioned factors to process the generated data and extract information to make intelligent decisions. The following section – a primer on machine learning – will lay a groundwork for ML algorithms in the context of smart decision-making for urban development.

### *4.1 A Primer on Machine Learning*

This section will serve as brief introduction to machine learning and various approaches in the context of smart urban development. Relevant examples are also included in this section. This is to show how the confluence of big data and machine learning advances propel smart urban development.

A pioneer in the area of machine learning is Arthur Samuel who has formulated the definition of machine learning as: "Field of study that gives computers the ability to learn without being explicitly programmed" [45]. A more workable definition was given by an expert, Tom Mitchell, whom defines a well-posed learning problem as: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E" [46]. To illustrate this definition in the context of smart urban development, we base our discussion on smart home systems, such as Google Home, Amazon's Alexa, and Apple Home, which cater to user's requests primarily through voice commands. In accordance to the definition, the following are defined:

- *Task T* – The task expected to be performed by the end user, like carrying out the desired voice command.
- *Performance measure P* – The accuracy of the voice command.
- *Experience E* – The actual execution of the voice command, both when it's interpreted correctly or otherwise. This includes vast amounts of historical data, which may also include human feedback.

In general, machine learning algorithms can be broadly classified into: supervised learning, unsupervised learning, and reinforcement learning. There are other types of algorithms like recommender systems, etc., but the aforementioned are the most relevant to the current context.

**Supervised Learning**

Algorithms of this category typically solve problems with the following characteristics:

- A dataset to train the system.
- There's a notion of expected output.
- Usually the problems solved using this algorithm have some sort of relationship between input and output.

These are typically the defining features of a supervised learning problem. And supervised learning problem can be further classified into two categories:

1. *Regression Problem*: The basic framework of supervised learning remains the same, the characteristics of regression problem is the output is a continuous data stream, i.e. input variables are mapped to a continuous function.
   *Example*: Given the sizes of different houses and the respective prices in the current real estate market, predicting the price of a house based on size is a regression problem since price is a continuous function of size here.
2. *Classification Problem*: The difference here is the output in supervised learning is a discrete quantity and depends on the input.

*Example*: Reformulating the previous example in regression problem, if the objective is to check if the selling price of house is above or below a certain value, then in this case there are just two possible outputs; this is a classification problem [47].

**Unsupervised Learning**

Algorithms in this category typically solve the problems with the following features:

- There is a dataset to train the system.
- The defining feature is the output; there is very little to no information of what the expected output would be.
- Usually the result of applying this algorithm is, a structure is derived in seemingly unrelated data.

The algorithms in this category can be classified as:

1. *Clustering Problem*: The objective here is to group the raw data based upon the similarities.
   *Example*: A supermarket chain grouping its customers based on the brands they prefer, to estimate the future demand of a brand.
2. *Non-clustering Problem*: The objective in this case is to filter data from what could be considered as irrelevant noise.
   *Example*: A voice recognition intelligent home system trying to separate voice commands in a noise-filled environment (more formally known as "the cocktail problem" in literature) [48].

**Reinforcement Learning**

Reinforced learning algorithms are defined by the following features:

- The software agents involved take actions with the sole purpose to maximize some notion of cumulative reward, which depends on the context of the problem solved.
- Unlike supervised learning it is not necessary to label the inputs and outputs, and there is no need to rectify suboptimal actions.
- The algorithms in RL tend to find a balance between exploring the unknown and exploiting the known [49, 50].

Many of the smart city applications involve RL, since there is absence of output in many cases and choosing the correct action is cumulatively rewarded, so that the desired outcome can be extracted. However, there is a drawback when it comes to smart city context. This is due to the enormous volumes of data; it's practically

impossible for humans to provide a reward feedback. A work around for this problem is to apply semi-supervised learning where data is partially labelled [3].

Most of the ML algorithms in the literature fall into one of the aforementioned categories. However, the future trend seems to shift toward deep learning (DL) and deep neural network (DNN). Briefly speaking deep learning is a subset of machine learning where multiple layers of ML algorithms are applied, such that the output of one stage is fed to the input of the next. This is implemented using specialized algorithms known as neural networks, aptly named, since they resemble the network of neurons in the human body [51].

## *4.2 Characteristics and Challenges for ML Algorithms in Smart City Ecosystem*

Smart cities as mentioned in the previous section must face three Vs – variety, volume, and velocity – to handle data and to effectively use it in ML algorithms. Another challenge comes when choosing the layer to run the ML algorithm, i.e., edge, fog, or cloud. This depends on the type of smart city application. As an example, let's take the case of autonomous vehicles – here the application demands stringent latency requirements for safety purposes. Hence, the processing is usually done in the vehicle itself, i.e., the edge node, instead of sending it to the cloud. It's imperative that all these factors are taken into consideration when implementing ML algorithms.

M. Mohammadi et al. in [3] also observe the following challenges for ML in smart city context:

- Implementing constant human feedback to enable learning would be difficult due to enormous volume of data involved.
- The rate of data generation, i.e., the velocity, also further makes it difficult for human review; hence learning should be automated.
- Since applications in smart cities tend to evolve overtime, a continuous and dynamic learning mechanism becomes a need.
- Because of the huge scale and volume, uncertainty and noise exist in thus generated data. Challenges are summarized in Fig. 3.

## *4.3 ML Smart Urban Development: A Few Use Cases from Literature*

Accurate object detection is needed for traffic control and autopilot in self-driving cars. NVIDIA has developed a state-of-the-art tool called NVIDIA Deep Learning GPU Training System (DIGITS) [52]. The link to the complete article can be found

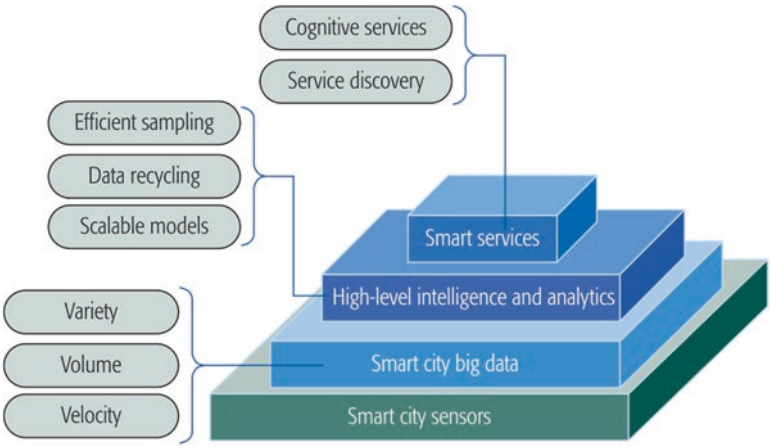


**Fig. 3** Challenges in smart cities for ML implementation [3]

in the bibliography. YOLO (current version YOLOv3) [53] is another object detection algorithm commonly used for object detection. Nagaraj et al. [54] implemented a traffic object detection using the abovementioned tools on the edge node. Edge node was chosen to keep the latency low, a common requirement for such applications. Their system could distinguish and detect objects from 14 different categories. In another case, Pacheco et al. [55] have implemented object detection in all the three different nodes, i.e., edge, fog, and cloud, for their smart classroom, thus demonstrating the versatility of these algorithms.

Nikouei et al. further have implemented surveillance as a service on the edge [56]. They have used lightweight detection tracking algorithms. The focus of their work is human object detection, further demonstrating that edge node can be used for effective implementation of ML algorithms.

In the field of crime and security for cities, Lourenco et al. [57] developed a framework called CRiMINaL (Crime patteRn MachINe Learning). The system uses historical data of various crimes like theft to assist the authorities in crime prevention. A relational machine learning approach was used in this framework. This system has been implemented by the authors.

Traffic flow prediction is another area of concern for any city. This can also be tackled by ML algorithms. Mohammed and Kianfar [58] designed and implemented ML algorithms to predict the traffic flow in Interstate 64, Missouri, USA. With such systems, proactive traffic management can be implemented for smart cities.

There are many use cases where ML techniques are increasingly being used to undertake smart urban development. In Fig. 4, the gist of data flow and levels in ML algorithms for smart city applications are summarized.

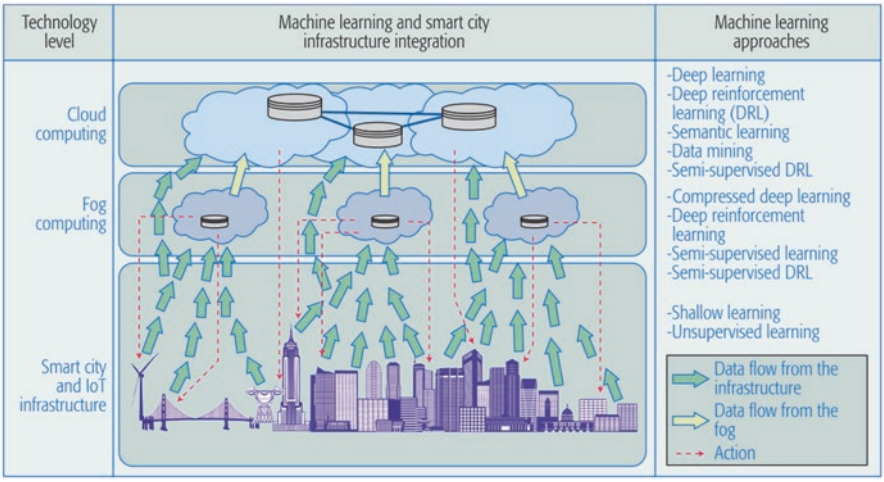


**Fig. 4** Machine learning approaches for smart city applications [3]

## 5 Conclusion

This chapter discusses the role of emerging technologies in smart urban development, specifically how edge computing empowers big data and machine learning with computational power to enable disruptive IoT applications. Also, it presents the current trending technologies in the storage and the analytics of big data. For example, NoSQL databases enable the storage of massive amounts of data. Moreover, analytics frameworks provide the necessary data analysis, processing, and visualization. Machine learning algorithms allow knowledge extraction, classification, clustering, as well as other functions to make sense of the data, thereby enabling a more intelligent decision-making system. On the other hand, from the use cases presented in this chapter, we infer that edge computing holds great potential in improving the way computational tasks are currently being processed. In applications such as waste management systems, smart traffic, and recommender systems, the use of edge computing can dramatically reduce the latency. Such improvements will expedite the development process toward smart cities.

## References

1. 2018 Revision of World Urbanization Prospects | Multimedia Library – United Nations Department of Economic and Social Affairs [Online]. Available: https://www.un.org/development/desa/publications/2018-revision-of-world-urbanization-prospects.html. Accessed 8 Sept 2019
2. SAS Institute: A Non-Geek's A-to-Z Guide Internet of Things. https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/non-geek-a-to-z-guide-to-internet-of-things-108846.pdf
3. Mohammadi, M., Al-Fuqaha, A.: Enabling cognitive smart cities using big data and machine learning: approaches and challenges. IEEE Commun. Mag. **56**(2), 94–101 (2018)

4. Lema, M.A., et al.: Business case and technology analysis for {5G} low latency applications. IEEE Access. **5**, 5917–5935 (2017)
5. {5G-CORAL H2020} project. http://5g-coral.eu/
6. 5G-CORAL D2.1: Initial design of {5G-CORAL} edge and fog computing system. https://cordis.europa.eu/project/id/761586
7. Schulz-Zander, J., et al.: Evaluating performance of containerized {IoT} services for clustered devices at the network edge. IFIP Int. Conf. Pers. Wirel. Commun. **25**(3), 194–203 (2017)
8. Mishra, A., Shin, M., Arbaugh, W.: An empirical analysis of the {IEEE} 802.11 {MAC} layer handoff process. ACM SIGCOMM Comput. Commun. Rev. **33**(2), 93–102 (2003)
9. Ramani, I., Savage, S.: {SyncScan}: practical fast handoff for 802.11 infrastructure networks. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 675–684 (2005)
10. Kwon, K., Lee, C.: A fast handoff algorithm using intelligent channel scan for {IEEE 802.11 WLANs}. In: 6th IEEE International Conference on advanced Communication Technology, vol. 1, pp. 46–50 (2004)
11. Park, S.-H., Kim, H.-S., Park, C.-S., Kim, J.-W., Ko, S.-J.: Selective channel scanning for fast handoff in wireless {LAN} using neighbor graph. In: IFIP International Conference on Personal Wireless Communications, pp. 194–203 (2004)
12. Pack, S., Jung, H., Kwon, T., Choi, Y.: SNC: a selective neighbor caching scheme for fast handoff in {IEEE} 802.11 wireless networks. ACM SIGMOBILE Mob. Comput. Commun. Rev. **9**(4), 39–49 (2005)
13. Tseng, C.-C., Chi, K.-H., Hsieh, M.-D., Chang, H.-H.: Location-based fast handoff for 802.11 networks. IEEE Commun. Lett. **9**(4), 304–306 (2005)
14. Grunenberger, Y., Rousseau, F.: Virtual access points for transparent mobility in wireless {LANs}. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2010)
15. Berezin, M.E., Rousseau, F., Duda, A.: Multichannel virtual access points for seamless handoffs in {IEEE} 802.11 wireless networks. In: IEEE 73rd Vehicular Technology Conference (VTC Spring), pp. 1–5 (2011)
16. Schulz-Zander, J., Suresh, P.L., Sarrar, N., Feldmann, A., Hühn, T., Merz, R.: Programmatic orchestration of WiFi networks. In: USENIX Annual Technical Conference, pp. 347–358 (2014)
17. Saldana, J., de la Cruz, J.L., Sequeira, L., Fernández-Navajas, J., Ruiz-Mas, J.: Can a {Wi-Fi WLAN} support a first person shooter? In: Proceedings of the 2015 International Workshop on Network and Systems Support for Games, p. 15 (2015)
18. Sequeira, L., de la Cruz, J.L., Ruiz-Mas, J., Saldana, J., Fernandez-Navajas, J., Almodovar, J.: Building an {SDN} enterprise {WLAN} based on virtual {APs}. IEEE Commun. Lett. **21**(2), 374–377 (2017)
19. Sapuntzakis, C.P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S., Rosenblum, M.: Optimizing the migration of virtual computers. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 377–390 (2002)
20. Clark, C., et al.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 273–286 (2005)
21. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Oper. Syst. Rev. **43**(3), 14–26 (2009)
22. Medina, V., Garcia, J.M.: A survey of migration mechanisms of virtual machines. ACM Comput. Surv. **46**(3), 30 (2014)
23. Nider, J., Rapoport, M.: Cross-{ISA} container migration. In: Proceedings of the 9th ACM International on Systems and Storage Conference, p. 24 (2016)
24. Machen, A., Wang, S., Leung, K.K., Ko, B.J., Salonidis, T.: Live service migration in mobile edge clouds. IEEE Wirel. Commun. **25**(1), 140–147 (2018)
25. Morabito, R., Farris, I., Iera, A., Taleb, T.: Evaluating performance of containerized {IoT} services for clustered devices at the network edge. IEEE Internet Things J. **4**(4), 1019–1030 (2017)
26. Abdullaziz, O.I., Wang, L.-C., Chundrigar, S.B., Huang, K.-L.: {ARNAB}: transparent service continuity across orchestrated edge networks. In: 2018 IEEE Globecom Workshops (GC Wkshps), pp. 1–6 (2018)

27. Taleb, T., Ksentini, A., Frangoudis, P.: Follow-me cloud: when cloud services follow mobile users. IEEE Trans. Cloud Comput. **7**(2), 369–382 (2019)
28. AkremAddad, R., Dutra, D.L., Bagaa, M., Taleb, T., Flinck, H.: Towards a fast service migration in {5G}. In: IEEE Conference on Standards for Communications and Networking (CSCN), pp. 1–6 (2018)
29. Nasrin, W., Xie, J.: {SharedMEC}: sharing clouds to support user mobility in mobile edge computing. In: IEEE International Conference on Communications (ICC), pp. 1–6 (2018)
30. mangodb [Online]. Available: https://www.mongodb.com/. Accessed 1 Sept 2019
31. Internet of Things | MongoDB [Online]. Available: https://www.mongodb.com/use-cases/internet-of-things. Accessed 1 Sept 2019
32. Apache Cassandra [Online]. Available: http://cassandra.apache.org/. Accessed 1 Sept 2019
33. Apache HBase – Apache HBase™ Home [Online]. Available: https://hbase.apache.org/. Accessed 1 Sept 2019
34. OpenTSDB – A Distributed, Scalable Monitoring System [Online]. Available: http://opentsdb.net/overview.html. Accessed 1 Sept 2019
35. GitHub – sequenceiq/hadoop-docker: Hadoop docker image [Online]. Available: https://github.com/sequenceiq/hadoop-docker. Accessed 1 Sept 2019
36. Rathore, M.M., Paul, A., Hong, W.H., Seo, H.C., Awan, I., Saeed, S.: Exploiting IoT and big data analytics: Defining Smart Digital City using real-time urban data. Sustain. Cities Soc. **40**, 600–610 (2018)
37. REST API [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer
38. Ebert, J., Kazimierczuk, M., Kazimierczuk, M.: Class E high efficiency tuned power oscillator. IEEE J. Solid State Circuits. **16**(2), 62–66 (1981)
39. MQTT [Online]. Available: http://mqtt.org/. Accessed 1 Sept 2019
40. Asadi, A., Wang, Q., Mancuso, V.: A survey on device-to-device communication in cellular networks. IEEE Commun. Surv. Tutorials. **16**(4), 1801–1819 (2014)
41. Docker Hub [Online]. Available: https://hub.docker.com/. Accessed 1 Sept 2019
42. Cheng, B., Papageorgiou, A., Cirillo, F., Kovacs, E.: GeeLytics: Geo-distributed edge analytics for large scale IoT systems based on dynamic topology. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 565–570 (2015)
43. Ning, Z., Huang, J., Wang, X.: Vehicular fog computing: enabling real-time traffic management for smart cities. IEEE Wirel. Commun. **26**(1), 87–93 (2019)
44. Sun, Y., Song, H., Jara, A.J., Bie, R.: Internet of things and big data analytics for smart and connected communities. IEEE Access. **4**, 766–773 (2016)
45. Samuel, A.L.: Some studies in machine learning using the game of checkers. II-Recent progress. Annu. Rev. Autom. Program. **6**(PART 1), 1–36 (1969)
46. Mitchell, T.M.: Machine learning. McGraw-Hill international edit, McGraw Hill Higher Education. McGraw-Hill, New York (1997)
47. Ng, A.: Lecture note on SL (regression, optimi) STANFORD, pp. 1–30 (2000)
48. Ng, A.: Machine Learning & Machine Learning Extended (Apr 2013). http://cnx.org/content/col11500/1.4/
49. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning? A survey. J. Artif. Intell. Res. **4**(1), 237–285 (1996)
50. Kazimierczuk, M.K., Bui, X.T.: Class-E amplifier with an inductive impedance inverter. IEEE Trans. Ind. Electron. **37**(2), 160–166 (1990)
51. Dormehl, L.: What is an artificial neural network? Here's everything you need to know | Digital Trends (2018) [Online]. Available: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/
52. Tao, A., Barker, J., Sarathy, S.: DetectNet: Deep Neural Network for Object Detection in DIGITS (2016). https://devblogs.nvidia.com/detectnet-deep-neural-network-object-detection-digits/
53. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement (2018). https://pjreddie.com/media/files/papers/YOLOv3.pdf
54. Nagaraj, S., Muthiyan, B., Ravi, S., Menezes, V., Kapoor, K., Jeon, H.: Edge-based street object detection. In: 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced &

Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, pp. 1–4 (2017)
55. Pacheco, A., Cano, P., Flores, E., Trujillo, E., Marquez, P.: A smart classroom based on deep learning and osmotic IoT computing. In: 2018 Congreso Internacional de Innovación y Tendencias en Ingeniería. CONIITI 2018, pp. 1–5 (2018)
56. Nikouei, S.Y., Chen, Y., Song, S., Choi, B.Y., Faughnan, T.R.: Toward Intelligent Surveillance as an Edge Network Service (iSENSE) using lightweight detection and tracking algorithms. IEEE Trans. Serv. Comput. **PP**(c), 1 (2019)
57. Lourenco, V., Mann, P., Guimaraes, A., Paes, A., De Oliveira, D.: Towards safer (smart) cities: discovering urban crime patterns using logic-based relational machine learning. Proc. Int. Jt. Conf. Neural Netw. **2018**, 1–8 (2018)
58. Mohammed, O., Kianfar, J.: Flow prediction: a case study of interstate 64 in Missouri. In: 2018 IEEE International Smart Cities Conference, pp. 1–7 (2018)

**Mahmoud Abu Zaid** received his B.S. in Computer Science from Assiut University, Egypt, in 2011 and his master's degree from Electrical and Computer Engineering, National Chiao Tung University, Taiwan. His current research interests include software-defined networks (SDN), Internet of Things (IoT), and big data. He's also currently a research and development engineer at AnaSystem Inc., Taiwan.

**Mohamed Faizal** received his B.E. in Electronics and Communication Engineering from Anna University (Sri Krishna College of Technology), in 2011. He also has worked as a Junior Research Fellow (JRF) in Indian Institute of Technology from January 2014 to April 2016. He earned his M.S. in Electronics Engineering from National Chiao Tung University, Taiwan. His Research interests include semiconductor devices, radio-frequency circuits, EDA, and machine learning. He is currently a RF Engineer at Wealth Tech System Co., Ltd.

**R. Maheswar** has completed his B.E. (ECE) from Madras University in the year 1999, M.E. (Applied Electronics) from Bharathiar University in the year 2002 and Ph.D. in the field of Wireless Sensor Network from Anna University in the year 2012. He has about 17 years of teaching experience at various levels and presently working as an Associate Professor in the School of EEE, VIT Bhopal University, Bhopal. He has published 40 papers at International Journals and International Conferences. His research interest includes wireless sensor network, IoT, queueing theory, and performance evaluation.

**Osamah Ibrahiem Abdullaziz** received the B.S. and M.Eng.Sc. degrees from Multimedia University, Malaysia, in 2011 and 2015, respectively. He is currently a Ph.D. candidate at the department of Electrical and Computer Engineering, National Chiao Tung University, Taiwan. His current research interests include software-defined networks, multi-access edge computing, network security, and network information hiding. He's also currently a researcher at Industrial Technology Research Institute (ITRI) of Taiwan. He's an excellent researcher with several publications in reputed journals under his belt.