Towards Efficient and Secure Cloud Architectures and Models : Application on Smart Grid, IoT and SDN Vehicular Networks

Djabir Abdeldjalil Chekired

Thèse
de doctorat
de l'UTT

Djabir Abdeldjalil CHEKIRED

Towards Efficient and Secure Cloud Architectures and Models: Application on Smart Grid, IoT and SDN Vehicular Networks

Champ disciplinaire :
Sciences pour l'Ingénieur

2019TROY0024 **Année 2019**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

EN SCIENCES POUR L'INGENIEUR

Spécialité : INGENIERIE SOCIOTECHNIQUE DES CONNAISSANCES, DES RESEAUX ET DU DEVELOPPEMENT DURABLE

présentée et soutenue par

Djabir Abdeldjalil CHEKIRED

le 16 septembre 2019

Towards Efficient and Secure Cloud Architectures and Models: Application on Smart Grid, IoT and SDN Vehicular Networks

JURY

M. P. LORENZ	PROFESSEUR DES UNIVERSITES	Président
M. R. LANGAR	PROFESSEUR DES UNIVERSITES	Rapporteur
Mme H. LABIOD	PROFESSEURE	Examinatrice
Mme A. MOLINARO	PROFESSORE ASSOCIATO	Examinatrice
M. L. KHOUKHI	PROFESSEUR UTT	Directeur de thèse
M. H. MOUFTAH	DISTINGUISHED UNIVERSITY PROFESSOR	Directeur de thèse

RÉSUMÉ

Les technologies du Cloud computing ont attiré l'attention des gouvernements, industriels et chercheures, car elles offrent un potentiel énorme en termes d'efficacité de traitement, de capacité de stockage et de disponibilité. Cependant, elles présentent des défis particuliers qui entravent l'intégration du Cloud dans des nouvelles technologies émergentes. L'objectif principal de ce projet de recherche est de concevoir et d'évaluer les performances des nouvelles architectures et modèles Cloud pour : la gestion de l'énergie des réseaux électriques intelligents, l'Internet des objets industriel et les réseaux véhiculaires basés sur SDN et 5G. Pour atteindre cet objectif, cette thèse propose des architectures et modèles efficaces et sécurisés basés sur les technologies de Cloud computing.

Ce travail présente trois applications principales du cloud computing. La première application concerne la gestion de l'énergie des réseaux électrique intelligents. On propose trois contributions principales: 1) en considérant le traitement des demandes de chargement des véhicules électriques (EV) comme étude de cas, nous proposons un nouveau modèle pour la planification de plug-in des EVs basé sur une architecture cloud, 2) afin de réduire les coûts de la production d'énergie et les pics de consommation, nous proposons un modèle de tarification dynamique en temps réel, 3) pour sécuriser l'infrastructure de comptage intelligent de la consommation d'électricité contre les attaques d'injection de fausse informations, nous proposons un nouveau système de sécurité, appelé SecMetering. La deuxième application est conçue pour le traitement des données de l'Internet des objets industriels ; nous proposons une nouvelle architecture décentralisée basée sur un fog hiérarchique. La troisième application concerne les réseaux véhiculaires. Nous proposons deux nouvelles architectures : 1) une architecture fog hiérarchique basée sur SDN, appelé HSVF, 2) une architecture qui utilise le réseau sans fil 5G basé sur SDN pour améliorer les performances de la conduite autonome.

Mots clés : Informatique dans les nuages, Réseaux électriques intelligent, Internet des objets, Réseau défini par logiciel, Réseaux ad hoc de véhicules, Cyber-Sécurité, 5G.

ABSTRACT

Cloud computing technologies have seized the attention of different governments, industries, and academic institutions as they offer tremendous potential for new business opportunities in terms of scheduling efficiency, storage capacity, and availability. Yet, they have particular challenges that hinder cloud integration in new emerging technologies and systems. The main objective of this research project is to design and evaluate the performance of new Cloud computing architectures and models for: Smart Grid energy management, Industrial Internet of Things, and Vehicular Network based on SDN and 5G technologies. To accomplish this goal, this dissertation proposes efficient and safe architectures and models based on Cloud and decentralized fog computing.

This work presents three principals applications of Cloud computing. The first application issue is for smart grid energy management. It proposes three principal contributions: 1) considering the scheduling of electric vehicles (EVs) energy demands for charging and discharging as a case study, we propose a new energy scheduling model for smart grid based on Cloud computing, 2) in order to reduce the energy cost and energy peak loads, we propose a real-time dynamic pricing policy, 3) to secure energy smart metering infrastructure against false data injection attacks in smart grid environment, we propose a new security scheme, labeled SecMetering. SecMetering is based on hierarchical and distributed intrusion detection system. The second application is designed for scheduling industrial internet of things data; we propose a new hierarchical and decentralized Fog architecture. The third application concerns vehicular networks. We propose two new architectures based on distributed SDN: 1) a hierarchical SDN-based wireless vehicular Fog, labeled HSVF, 2) a framework that uses 5G networks based SDN to enhance the performance of autonomous driving applications.

Keywords: Cloud computing, Smart power grids, Internet of things, Vehicular ad hoc Networks, Cyber-Security, 5G.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PaaS	Platform as a Service
SaaS	Software as a Service
IaaS	Infrastructure as a Service
SG	Smart Grid
AMI	Advanced Metering Infrastructure
HEMS	Home Energy Management System
DoS	Denial-of-Service
OCC	Operations Control Center
FDIA	False Data Injection Attack
IoT	Internet of Things
IIoT	Industrial Internet of Things
ICT	Information and Communication Technology
VANET	Vehicular Ad hoc Network
DSRC	Dedicated Short-Range Communication
SDN	Software Defined Networking
NFV	Network Function Virtualization
WDN	Wireless Distributed Networks
MAC	Medium Access Control
HD-IDS	Hierarchical and Distributed Intrusion Detection System
HSVF	Hierarchical SDN for Vehicular Fog
EV	Electric Vehicle
EPSS	Electric Public Supply Station
EV_CPA	EV Calendar Priority Attribution algorithm
EV_RPA	EV Random Priority Attribution algorithm
BSS	Best Station Selection algorithm
RB_EM	Renewable Building Energy Management algorithm
EV_OCM	EV Optimal Calendar Management algorithm

(dedication) To my late FATHER.

"He taught me to persevere and prepared me to face the challenges with faith and humility".

ACKNOWLEDGMENTS

I would like to thank all people who have encouraged me, in various ways, to pursue my goal. This dissertation would not have been possible without them.

My deep appreciation goes to Prof. Lyes Khoukhi and Prof. Hussein Mouftah for giving me the opportunity to work with them on one of the promising technologies that can make our world better. Their exemplary supervision, great support, valuable advices and constructive criticism have helped me become the researcher I am today.

Special thanks go to my friends Hocine and Amine for their guidance and encouragements during my first year as a Ph.D student. Their explanations were easy to grasp and their feedback was highly appreciated. I am very glad to have worked with them.

I would like also to thank all my colleagues at ERA Research Laboratory for the beneficial discussions, research collaboration, and continuous exchange of knowledge. No doubt all ERA members are great people.

In addition, I would sincerely like to thank all my friends in Troyes, France, for their continuous support and encouragements. Special thanks for my brother Karim Mekkouci, who made my stay in Troyes comfortable and very enjoyable.

Last but not least, I would like to thank my mother, my two brothers and my sister Yasmine for all their love and encouragements. In particular, I am grateful to my mother who raised me with a love of science and supported me in all my career, and for all of the sacrifices that she has made on my behalf. And most of all for my loving, supportive, encouraging, and patient wife Manel; she was always my support in the moments when there was no one to answer my queries.

CHAPTER 1

INTRODUCTION

In this chapter, we start by presenting some basic definitions, concepts and applications related to our thesis (Cloud, smart grid, IoT and vehicular networks, and 5G). Then, we present our contributions and objectives, and finally we conclude the chapter by illustrating our thesis organization.

1.1 Research Context

1.1.1 Cloud Computing Architecture

Cloud computing is an emerging computation model that provides on-demand facilities, and shared resources over the Internet. Cloud computing, based on large storage and computational devices, acts as a utility provider [1]. Implementation and maintenance costs, and system complexity of cloud computing are reduced with the utilization of information technology. With cloud computing, computations are achieved over the Internet. Resources such as software and information are shared through the network and retrieved by computers and devices on demand. Cloud computing provides three distinct types of services: Platform as a Service (PaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS) [2].

- *Infrastructure as a service:* IaaS is the infrastructure service model that includes storage and virtual machines. Load balancing in cloud computing is performed using IaaS. Users can install access to required software through virtual machines. These virtual devices provide on-demand facility to the customers [3]. The IaaS service offers hardware platform to the users on-demand basis. Therefore, users can access the online hardware platform as on-demand basis to fulfill their requirements. Additionally, the IaaS service also supports virtualization of resources on which a guest user can run his/her own operating system.
- *Platform as a service:* PaaS is responsible for the development and delivery of programming models to IaaS. Users can access such programming models through cloud and execute their programs. PaaS is responsible for the runtime execution of users' given task. Therefore, the PaaS service completes the requirements of building and delivering of Web-applications without downloading and installing required software as well.
- *Software as a service:* SaaS supports all the applications in the cloud environment. This feature of cloud computing is accessible through Web-browsers. The SaaS service provides

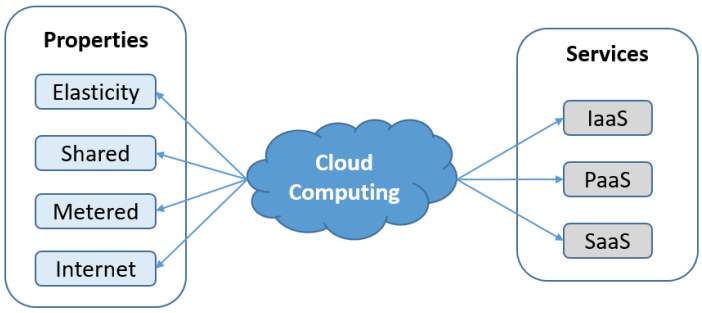


Figure 1.1- Cloud computing infrastructure

the modeling of software deployment where users can run their applications without installing it on his/her own computer. However, this service is limited to the users, i.e., only existing set of services is available to the customers.

On the other hand, Cloud can also be categorized, depending on the deployment models, as: private, public, community, and hybrid [3].

- *Private cloud:* The cloud is owned by a private organization, and information is shared only within the organization. The purpose of this type of cloud application is to serve its own business applications.
- *Public cloud:* On the other hand, public cloud is owned by a service provider, and used by public for their purposes.
- *Community cloud:* Community cloud is similar to the private cloud with some additional features to provide services to a group of organizations who have similar type of requirements.
- *Hybrid cloud:* Hybrid cloud is the extension of cloud computing with private, public, and community cloud computing techniques. The private, public, and community clouds are integrated together to perform several tasks which are capable of handling the requirements of private, public, and community organizations.

In Figure 1.1, cloud services and properties are shown. The advantages of using a cloud computing model are as follows [4]:

- *Elastic nature:* Cloud computing supports elastic nature of storage and memory devices. It can expand and reduce itself according to the demand from the users, as needed.

- *Shared architecture:* Cloud computing also supports shared architecture. Information can be shared among the users after meeting the privacy issues, and, thereby, reducing service costs [5].
- *Metering architecture:* Cloud computing offers metering infrastructure to customers [36]. In the metering system, cost optimization mechanisms are offered to users, enabling them to provision and pay for their consumed resources only.
- *Internet services:* Cloud computing can be implemented in the existing Internet service system. Thus, it supports the existing network infrastructure.

1.1.2 Smart Grid Architecture

A smart grid (SG) can be conceptualized as an integration of electric power grid with the bidirectional communication network system [6]. With the integration of information and communication technology, modern smart grid is capable of providing electricity to the end users in an increasingly efficient manner. A smart grid architecture spans primarily three different technical domains generation system, transmission side, and distribution side. The generation side consists of traditional power plant generation. The transmission side is responsible for delivering electricity to the distribution side (customers). An important characteristic of a smart grid is controlling electricity consumption at the customers' ends by establishing different optimization methods [7]. To achieve this goal, smart metering and micro-grid are the most important components that have been incorporated in the smart grid architecture.

Smart metering: Smart metering is one of the most emerging technologies used in smart grid to obtain information about customers' real-time energy consumption. It is also capable of controlling the advanced metering infrastructure (AMI) systems. AMI is supported with bidirectional communication mechanism to obtain real-time energy consumption at the customers' ends remotely. A smart meter is a device deployed at the distribution-end, and capable of recording the energy consumption by the customers. Customers and utilities are benefited with smart metering infrastructure. For example, a customer can estimate his/her energy consumption during the whole day for cost-optimization, and utility is able to maintain real-time monitoring for the supply-demand curve.

Micro-grid: In the concept of the smart grid architecture, the power distribution side is divided into subgroups similar to the step-down transformers. These subgroups have self-generation capacity such as combined heat power, wind generation, and solar generation.

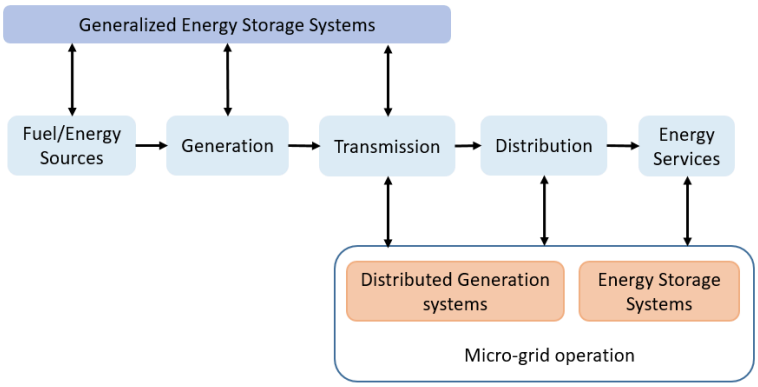


Figure 1.2- Smart Grid architecture with centralized and decentralized generation

Additionally, they can control and distribute electricity to the end-users. These subgroups are known as micro-grids. A micro-grid is the integration of low-voltage electricity systems with self-generating facilities. A subgroup has an independent power supply and control system to provide electricity to the end users [8]. In the presence of an intrusion in the entire system, a micro-grid acts in the islanding mode. In such a situation, a micro-grid is able to control the power flow autonomously. Due to the fluctuation in renewable energy sources, the primary distributed energy resources are converted into distributed generation using the micro-grid operation. Some of the advantages of a micro-grid are listed as follows:

- Electricity distribution facility is converted to decentralized from centralized.
- A micro-grid increases the local reliability, and energy management mechanisms.
- Improved real-time monitoring system can be achieved.
- In the presence of any kind of intrusion, it acts in the islanding mode, and supplies electricity in secure and efficient ways to the end users.
- All the micro-grids can interact with one another and exchange energy, when there is an excess or deficit of it.

The smart grid conceptual model, identified by the National Institute of Standards and Technology (NIST) [9], gives the characteristics, requirements, operations and services that should be provided by a SG. It also specifies communication ways from top level to lower levels for SG applications. The conceptual model includes seven domains such as bulk generation, transmission, distribution, customer, service provider, operations and markets.

The conceptual model begins with bulk generation. In this domain, electricity generation and protection procedures are realized. The second domain is the markets which perform load balancing by analyzing and optimizing energy pricing to help control energy consumption of

customers. Business processes of energy producers, customers and transmission companies are performed by a service provider, which is the third domain of the SG conceptual model. Operations in the network such as monitoring of network operation, network control, fault detection and reporting are realized with the fourth operations domain. Transportation of electricity from sources to distribution are achieved by using the transmission domain. Service providers optimize flows by the agency of the transmission domain and connect with customers via the distribution domain which achieves real time monitoring of electricity consumption. The last SG domain is the customers who let their energy usage be managed. All of these domains makeup the SG architecture and result in many benefits including efficiency, low cost, fault tolerance and renew-able energy generation.

Figure 1.2 illustrates a smart grid architecture with the generation, transmission, and distribution components. The traditional power plants, and renewable energy sources are treated as generation side of a smart grid. Electricity is transmitted to the distribution side from the generation side using transmission lines. Finally, the distribution side is responsible for distributing it to the end users.

1.1.3 Cloud Computing Application for Smart Grid Energy Management

Energy management is a major concern in smart grid environments. In the past several years, researchers addressed this issue by incorporating the implementation of different components such as Home Energy Management System (HEMS), building energy management system, dynamic pricing, and load shifting [10]. Therefore, the objective of the smart grid is to support cost-effective and reliable energy management in real-time. On the other hand, The SG offers many features and applications to consumers, however it needs to be improved to handle more secure, efficient and scalable systems. This can be carried out by using of Cloud computing capacities. Operations can be done at low cost with cloud computing because sharing and automation are widespread within these systems. Also, real time response is very important for SG applications for giving immediate demand response. In cloud platforms, when a client request comes to the cloud operation system, a response is sent to the client in real time. In addition, cloud computing provides a power efficient self-healing system which is crucial for SG applications to recover from faults and give instant response to customers. Communication in the SG is achieved via the Internet, therefore there should be no Internet outages to provide consistent transmission. Cloud computing has many characteristics that can yield improved SG applications. These cloud characteristics are listed below:

- Reconstruction of SG technological infrastructure is provided with the agility characteristic of cloud computing [11].
- Communication between the machine and cloud software is done via cloud computing application programming interface (API).
- Public cloud delivery model provides lower cost for SG customers.
- Any device by a consumer/customer that needs to access the systems can do this from anywhere via the Internet. This is the device and location independence feature of cloud computing.
- Maintenance and virtualization are other properties of cloud computing. Installation is not required for running applications and performing computations. This provides easy access to cloud computing services from anywhere and at any time. Servers and storage devices can be shared and carried easily from one server to another using the virtualization property of cloud computing data centers.
- Large number of users use shared resources, which are located in a pool. This is enabled by the cloud computing multitenancy characteristic. In this way, infrastructure is centralized with lower cost, without needing extra device load capacity and utilization increase.
- Cloud computing provides disaster recovery capability and this assures reliability.
- Some architectures are constructed for increasing system performance in cloud computing platform by using web services.
- Reliability is improved by using private cloud platforms which prevents connection losses.

1.1.4 Cloud Computing Application for Smart Grid Security

A smart grid can be conceptualized as a cyber-physical system that connects physical electricity systems and cyber-infrastructure, with the integration of the Internet. This service can communicate with the consumer appliances and also provide the backbone for service providers to absorb contents and control operations. With the presence of online connectivity, it is a big challenge to prevent cyber-attacks in the smart grid that can potentially disrupt the power supply [12]. One of the important issues is power theft by consumers. This can be done by hacking a smart meter or modifying the real-time information through accessing communication channel to change the reported electricity usage. Additionally, data manipulation is also one of the most security concerns in the smart grid. To overcome these issues, we need to implement proper security for secure and reliable smart grid architecture.

Security can be implemented on the consumer side, transmission side, and generation side. The main security aspects of smart grid are as follows:

- With the increase of the grid system complexity and their integration, it is difficult to track interactions among business systems securely.
- The smart grid architecture is more complex than the one for the traditional power grid. The implementation of a smart sensor network, wireless communication, and smart meters increases the complexity in the protection of the information security system.
- With the implementation of millions of smart meters, the network is distributed to the end-user systems. So, the capacity of the protection at the end-users needs further enhancement.
- The denial-of-service (DoS) attack to affect the stability of the applications for the SG.
- Utility and third-party can access the user data as well as private information, thereby affecting the privacy of the users.

1.1.4.1 False Data Injection Attacks in Smart Grid

In smart grid, enormous amount of real-time energy information is collected and reported to operations control center (OCC) for appropriate monitoring and analyzing the state of power grid. To achieve this, real-time energy consumption is measured from the electric appliances in residential users' home. The appliances are connected to a central element called smart meter; this latter measures and reports the electricity consumption of appliances to the OCC via an advanced metering infrastructure (AMI) for further analysis and scheduling (e.g., for real-time energy pricing, detecting energy fraud, predict the energy supply, etc.). However, AMI is confronted with various cyber security threats [13]. The costs of security events may be benign disruptions or deliberate acts of sabotage. The latter can menace lives of citizens and even national security. For example, in December 2015, more than 225,000 people in Ukraine suffered from a blackout due to a devastating cyber-attack (i.e., mainly false data injection attack (FDIA)) on a power station [14].

In power grid system, the metered data has an important economic value (e.g., when the data is used for dynamic pricing calculation). For instance, a malicious attacker who gets access to the crypto key may remotely disconnect massive smart meters, which causes a heavy financial loss of smart grid utility [15]. Moreover, users' privacy is of supreme importance to ensure correct operation and protection of users personal data: it has been shown that users electricity measurements can be used to outline their behavior and even to determine which household appliances are being used [16]. Therefore, from the perspective of users, protecting its data is

of a primary concern. For example, a relative low and static daily consumption of a household may indicate that no one is at home (this may facilitate robbing actions, etc.). Thus, there is a critical need to secure the information exchanged within AMI environment.

Different security fears in smart grid energy metering using smart meters may come from several attackers that can use different techniques (e.g., physical attacks, cyber-attacks, etc.). The attackers may be external and/or local. Local attackers may be the users, the gateways or the operation center. Generally, these attackers are honest-but-curious [16]; they attempt various techniques to seek and deduce knowledge about victims, using for example their energy consumption measurements. It is stated that the determination of users' behavior figures in the second most serious attacks consequences on smart grid environment in 14 fears types defined by Electronic Privacy Information Center [16].

On the other hand, an external attacker may compromise a smart meter by injecting false data or spying on the communication channel to violate user's privacy. The false data injection attack (FDIA), first proposed by Liu et al. [17], is a cyber-attack where power system state estimation outputs are corrupted by injecting false data into meter measurements in a carefully coordinated fashion. The defining feature of a successful FDIA is that the state estimation residual falls below a hypothesis test threshold despite the presence of corrupted measurements; the attack thereby may evade detection.

Yet, smart meters, even they have protection against physical damage using sophisticated microcontrollers, they are still vulnerable to electricity theft and false data injection attacks. Furthermore, as a fully trusted party, a trusted authority should be implemented in the AMI system model; it is responsible to insure random secret keys to household users and operation center through a secure communication channels. Hence, some issues like how to design an effective privacy preserving, secure and fault tolerant smart metering scheme, and AMI architecture resisting to power theft and detecting false data injections in smart grid, still merit more investigations and effort by the both academia and industry.

1.1.5 Cloud Computing Application for Industrial IoT

Internet of Things (IoT) has been gaining attraction in many industry areas, such as logistics, manufacturing, retailing, and pharmaceutics [18]. On the other hand, with the advances in wireless communication and sensor network technologies, more and more networked things or smart objects are being involved in industrial IoT (IIoT). As a result, IIoT-related technologies

have also made a large impact on information and communication technology (ICT) and industrial systems technologies [19].

Recently, the manufacturing industry has gained another transformation sparked by smart connectivity and advanced analytics. This is referred to as advanced manufacturing in North America and Industry 4.0 in Europe [20]. Because of its highly adapted properties and reduced delivery time, this smart connectivity and analytics based on IIoT are seen as key enablers to future industry. On the one hand, this transformation facilitates flexibility and scalability, particularly with respect to the high generation of data and information. On the other hand, it also permits the delivery of products with improved quality at the same or lower cost by leveraging the data collected by various elements of a connected assembly line.

Consequently, a large amount of data is generated from IIoT devices installed over different physical factory components as assembly lines, human machine interface, and logistic structures. Besides, the data collected brings transparency about the machines' operations, the materials utilized, the facility logistics, and even about the human operators. However, much of the data collected are used for direct real-time feedback control and for forensic purposes. As presented in a case study from Bosch group in [20], during the manufacturing process, information about the status of the process, the states of machines, tools, and parts produced are uninterruptedly relayed, scheduled and stored in a real-time manner. The capacity, scale, and regularity of production in this traditional case study (i.e., Bosch group case) is so high that it requires the use of a big data tools scheduling to store, reprocess, and join the data. In these regards, an efficient service management for IIoT needs to be implemented, and the traditional industrial network should be enhanced and adapted in order to ensure real time analytic process, to minimize the end-to-end delay and to increase the data storage capacity.

In recent research works, cloud computing has acted as a key approach for big data processing with its ample computing, storage capacity and distributed aspects [21], [22]. However, most interaction between IIoT devices and back-end servers in the traditional system is done through large scale cloud data centers. Nevertheless, because of the delays induced on Wide Area Networks (WANs), and being far from IIoT devices, cloud-based IIoT systems face several challenges, including especially a high response time due to the high cost of communication bandwidth, and heavy loads on cloud servers due to the high redundancy of data.

1.1.5.1 The Need of Fog Computing for Scheduling IIoT Data

In order to overcome the weaknesses of cloud computing, a local cloud computing architecture, called fog computing, has been introduced recently. According to Cisco [23], fog computing extends the cloud computing away from the cloud computing data centers and towards the edge of the network. Fog computing, considered as a new decentralized computing aspect of cloud architecture, is one of challenging industry topics for IoT. Fog computing is a newly introduced concept that aims to put the cloud services closer to the end users (things) for better quality of service (QoS). Fog is an intelligent server's layer sitting between cloud and IIoT that brings low latency, location awareness, and widespread geographical distribution for IoT [24]. Getting main concepts of cloud computing, fog provides computation and storage capacities to end users, but at the edge of the network. An open issue consists of exploring the enhancement of an optimal end-to-end delay network by having an intelligent layer of fog servers between IIoT devices and cloud data centers.

1.1.6 SDN and 5G for Vehicular Networks

Recent advances in wireless technologies have enabled vehicles to share information between each other. Such information could significantly improve the user's driving practice in terms of both safety and efficiency. Nowadays, a vehicle can exchange information with other vehicles (V2V), surrounding infrastructure (V2I), and a back-end cloud server, enabling the vehicle to everything communication (V2X). According to Cisco internet business solutions group report [25], there will be 400 million gigabytes of data generated by 300 million passenger vehicles, interconnected via various wireless technologies. This data is generated by various types of applications, including route planning, traffic statistics, road conditions as well as media streaming and online gaming. These applications have different QoS requirements, which impose significant challenges to the existing networking infrastructure.

To mitigate these inconveniences, car manufacturers and research institutions have been working jointly to develop V2X communication technologies to capture future commercial opportunities. A key technology enabler of V2X is vehicular ad hoc network (VANET), which is based on the dedicated short-range communication (DSRC). Apart from VANET, cellular networks, particularly 4G/5G, are also considered as a promising technology to enable V2X communication as it can provide real-time, highly reliable, and actionable information flows [26]. For instance, the 5G cellular network can achieve a data rate of up to 7.5 Gb/s in a stationary environment and 1.2 Gb/s when a vehicle is moving at a speed of 100 km/h based on

28 GHz spectrum [27]. Although these technologies can ensure reliable and universal mobile coverage, several salient features of VANETs introduce new challenges, such as unbalanced traffic flows in a multi-path topology, high response delay and inefficient network resources utilization [28]. Thus, flexible and programmable architectures are key requirements for wireless vehicular networks.

1.1.6.1 SDN for VANET

Software Defined Networking (SDN) is an emerging network paradigm that has been increasingly incorporated into different types of network systems. In SDN, the control plane and data plane are decoupled, and network resources are managed by a logically centralized controller. Furthermore, devices from various vendors can communicate with each other via a standardized interface, which significantly simplifies the network management and offers a programmable and flexible network architecture. Originally, SDN was designed and deployed in wired network environments with high-speed switches, such as data centers and campus networks. Yet, recently, researchers have started using SDN enhancing many forms of wireless distributed networks (WDNs), including mobile ad hoc networks, wireless sensor networks, device-to-device networks, Wi-Fi networks, and vehicular networks [28]. Indeed, SDN brings new intuitions and high potential to improve the flexibility, programmability and efficiency of wireless networks [29] despite the new challenges that it may introduce.

1.1.6.2 Fog Computing for VANET

Other technologies can also be considered to enhance VANET's performance. For instance, recent research studies have shown that interconnecting VANET with cloud services can improve the network's performance [30]. In addition, fog computing can also be considered as a fundamental solution for accommodating the flow of mobile traffic and reducing latency, both known to be pivotal problems of cloud computing. Fog services, including computation, storage, and distributed networking, are deployed near end users (i.e., the edge of the network); thus, timely and reliable access is provisioned to delay-sensitive mobile applications. Still, these fog services can get overwhelmed, in some cases, due to the growing number of demands from customers. This can be observed particularly during peak hours, subsequently resulting in a critical performance degradation.

1.1.6.3 5G Technology and Service Slicing for VANET

The emerging 5G technology is foreseen as the promising solution to improve network performance and management while ensuring high data rate and enhanced QoS. It is expected to support a variety of vertical industries such as manufacturing, automotive, healthcare, energy, media and entertainment [31]. Such verticals will spark different use cases, imposing new set of requirements (e.g., scalability, latency, reliability and availability) that the currently deployed networks, which are based on the "one-fit-all" conceptual paradigm, are not able to meet [32, 33]. Accommodating the new applications while supporting existing services imply having a programmable and flexible network infrastructure that can be shared by different network technologies (e.g., IoT, cellular and vehicular). This shared infrastructure should endorse a software-based architecture that uses open interfaces to allow access to different users (e.g., individuals or businesses) while matching their service requirements.

One way to achieve such a design is a concept called network slicing. It consists of splitting the physical network infrastructure into multiple logical networks, referred to as slices. These slices are controlled and managed independently by the slice owners, which can be of two types: Over-The-Top (OTT) service providers and Virtual Mobile Network Operators (VMNO) [34]. Each slice is allocated a set of network functionalities that are selected from the shared network infrastructure. These functionalities can be virtualized using technologies such as SDN and NFV. Indeed, an SDN-based architecture was proposed in [35] to enable efficient and scalable network slicing. It deploys a controller between slice owners and the shared network infrastructure that provides an abstract view of the network resources to the slice owners while enabling them to transmit their service requirements to the network infrastructure using northbound interfaces.

1.2 Thesis Objectives

In a time where data traffic and energy consumption is constantly booming in smart cities, Cloud computing is becoming the most plausible solution to cope with the conventional smart city networks' shortcomings. Most of these conventional networks have rigid performances requirements (i.e., energy saving, delay, scalability, bandwidth and throughput). Yet, meeting these requirements is not straightforward, given the particular characteristics of smart cities networks (e.g., high energy consumption, high mobility, sporadic connectivity, high data traffic, and security threats). Thus, the objective of this dissertation is to contribute to the design and the evaluation of new solutions that ensure smart city networks requirements. This work

was carried out on the medium access control (MAC) sublayer, the network layer and the application layer. On the one hand, it proposes a new efficient and safe models and architectures based on cloud and SDN technologies for: 1) smart grid energy managements, 2) industrial IoT data scheduling, and 3) connected vehicles. The proposed solutions take performance evaluation parameters, such as energy saving, response time, and latencies into account. On the other hand, it specifies, evaluates, and implements mechanisms for managing networks resources, in order to offer differentiated services to further enhance the performance of different smart city networks (i.e., smart grid network, IoT network, and VANET).

The first part of this thesis addresses the problem of smart grid energy saving considering three principals issues: 1) EVs energy plug-in management for batteries charging and discharging, 2) Dynamic and real-time pricing for energy saving and cost optimization, 3) Secure energy smart metering against false data injection attacks. Several approaches have been proposed in the literature to deal with the previous issues (e.g., [38], [39], [43], [48], [49], [60]). Despite their good performance, exiting schemes sustain major shortcomings. These models are based on conventional centralized architectures that only acquire local and centralized network topology in smart grid environment, making them prone to local maximum, data congestion problems, and high response delay which affects drastically the energy saving in smart grid especially for scalable smart city networks.

The second part of this dissertation focuses on the problem of Industrial IoT data scheduling. To efficiently integrate IoT technology into industry, the collected and sensed data from IIoT need to be scheduled in real time constraints with an optimal end-to-end delay; especially for big factories. In the literature, several schemes [63-66] have been proposed to help smart factories to schedule data and requests in real-time constraints. They can be of two types: centralized schemes based on Cloud computing, and decentralized schemes that consider distributed computation feature. Although they assure certain level of performance, the existing schemes experience various limitations. For the centralized schemes, on the one hand, most of the existing works focus on centralized computation architectures for the purpose of monitoring data and managing control processes in the industrial automation and focus only on higher levels than the field-level. On the other hand, decentralized based schemes are cost-inefficient (i.e., due to the use of multiple servers); in addition they ignore data transmission delay and synchronization time especially when using distributed computing manner over decentralized data centers.

The final part of this thesis addresses the problem of the development of an efficient VANET architecture that requires considerable efforts in information gathering and data processing to allow quick decision-making and fast feedback to all users. Thus, several novel and promising VANET network applications have been developed lately. Examples include trip planning, media sharing and Internet access. However, sufficient network resources (e.g., equipped vehicles, base stations (BSs), Road Side Units (RSUs) and other infrastructures) to support these applications are not yet available or are deployed at a slow pace. On the other hand, 5G networks are anticipated to support a plethora of innovative and promising VANET services. These services have heterogeneous performance requirements (e.g., high-rate traffic, low latency and high reliability). To meet them, 5G networks are entailed to endorse flexibility that can be fulfilled through the deployment of new emerging technologies, mainly SDN, NFV, and network slicing. Although applying SDN in a heterogeneous vehicular network is challenging, several architectures (e.g., [67], [68], [71], [72], [73]) have been proposed in the literature to address the aforementioned issues. These studies describe network architectures that can be arranged in two categories: centralized and distributed. Despite their adequate performance, schemes in both categories have limitations. On the one hand, the use of only one SDN controller could be the bottleneck of the entire system, leading to resiliency and scalability issues. In addition, adopting the conventional ad hoc mode of vehicular networks whenever connections to the SDN controller are lost may escalate the end-to-end delay. On the other hand, deploying multiple SDN controllers increases the system complexity and introduces new challenges (i.e., handover) that need to be dealt with. This may lead to higher control overhead [71, 72] and longer end-to-end delay [73]. Yet, to the best of our knowledge, none of the proposed models have considered all the aforementioned factors.

1.3 Thesis Contributions

The aforementioned research topics have yielded several results that we have published as articles in international refereed journals and conferences. The thesis consists of six contributions made from six scientific articles (4 published journals and 8 conferences).

To address the issues of integrating cloud computing in the context of smart grid for energy management and smart metering security, we propose three principal contributions: 1) we propose a new energy scheduling model for smart grid based on cloud computing; we consider the scheduling of EVs energy demands for charging and discharging as a case study. This work was published in IEEE Transactions on Industrial Informatics Journal: Special

Section on Smart Grid and Renewable Energy Resources: Information and Communication Technologies with Industry Perspective [22]. 2) We propose a real-time dynamic pricing model for EVs charging and discharging service and building energy management, in order to reduce the peak loads, and then optimize energy saving. Our proposed approach uses a decentralized cloud computing architecture. This work was published in IEEE Transactions on Industrial Informatics Journal: Special Section on Cloud Computing in Smart Grid Operation and Management [36]. 3) In order to secure smart metering infrastructure against false data injection attacks in smart grid environment, we propose a new security scheme (called SecMetering) based on hierarchical and distributed intrusion detection system (HD-IDS). The proposed HD-IDS is based on distributed fog architecture considering three hierarchical network layers. A part of this work is accepted for publication in IEEE International Conference on Communication, the other part is submitted to IEEE Transactions on Industrial Informatics Journal: Special Section on New Trends in Residential Energy Management (the paper is under Major revisions).

To address the issue of using cloud computing for industrial IoT, we propose a hierarchical fog servers' deployment at the network service layer across different tiers. Using probabilistic analysis models, we prove the efficiency of the proposed hierarchical fog computing compared with the flat architecture. We use two-priority queuing model in order to schedule and analyze IIoT data. We further introduce a workload assignment algorithm to offload peak loads over higher tiers of the fog hierarchy. This work was published in IEEE Transactions on Industrial Informatics Journal: Special Section on Fog Computing for Industrial Applications [37].

To address the issue of application cloud computing for vehicular networks, we propose two new architectures as follows: 1) we propose a new hierarchical SDN-based wireless vehicular fog architecture, called HSVF (i.e., Hierarchical SDN for Vehicular Fog architecture). HSVF is based on a hybrid SDN control plane that reinforces centralized and distributed management. This work was published in IEEE Vehicular Magazine Special Issue on Next-Generation Softwarized Networks. 2) we design a framework that uses 5G networks, SDN, NFV, and network slicing technologies to enhance the quality of service of the autonomous driving application. The framework is made of i) a distributed and scalable SDN core network architecture that deploys fog, edge and cloud computing technologies; ii) a network slicing function that maps autonomous driving functionalities into service slices; and iii) a network and service slicing system model that promotes a four-layer logical architecture to improve the

transmission efficiency and satisfy the low latency constraint. This work is submitted to IEEE Journal on Selected Area on Communications (the paper is under minor revision).

1.4 Thesis Organization

The remaining of this thesis is structured as follows. In Chapter 2, we describe the existing approaches in the literature that address the aforementioned issues. In Chapter 3, we address the application of cloud computing for scheduling EVs energy demands in smart grid. We then present the dynamic pricing model proposed for energy saving in smart grid based distributed Fog architecture in Chapter 4. In Chapter 5, we present the SecMetering scheme proposed to secure smart metering against false data injection attacks in smart grid. In Chapter 6, we present the hierarchical Fog architecture for scheduling industrial IoT data. Finally, we present the issue of decentralized cloud computing integration for vehicular networks where we propose a hierarchical SDN-based wireless vehicular fog architecture, then we present the application of 5G slicing technology for autonomous driving services based on a new SDN core network architecture.

CHAPTER 2

RELATED WORK

Following the thesis contributions, we present in this Chapter recent and relevant related work of cloud computing integration on smart grid energy management, Industrial IoT and vehicular networks, and 5G. For smart grid energy management, we study the state of the art of three relevant topics, a) the management of EVs plug-in for energy charging and discharging in smart, b) recent pricing models proposed to save smart grid energy, and 3) relevant secure smart metering solution in smart grid. Next, we present the recent proposed works on the literature about the application of cloud computing for scheduling industrial IoT data. Then, we present recent studies in the literature that demonstrate the high potential of SDN to improve data forwarding and to ensure better QoS in vehicular networks. These studies describing network architectures are arranged in two categories: centralized and distributed. Finally, we present the various 5G slicing approaches proposed in the literature for the automotive vertical.

2.1 Cloud Computing Application for Smart Grid Energy Management

For several years, researchers proposed several solution concepts for demand response and smart grid energy management. In this part, we study the state of the art of three relevant energy management issues in smart grid: a) the management of EVs charging and discharging energy demands according to the used architecture, b) pricing models for energy saving in smart grid, and c) smart metering security.

2.1.1 EV plug-in Energy Management

To deal with the impact of EVs charging on the power grid (in terms of energy overflow and real-time management), several scheduling models have been proposed. Existing works can be classified into two classes according to the used architecture:

1) Non cloud based architecture: In this class (e.g. [38], [39], [40], [41], [42]), the management of plug-in EVs at public supply stations is centralized in the micro grids, and EVs scheduling is executed by smart grid providers. The authors in [38] present an approach for scheduling EVs charging and discharging. They introduce global and local scheduling schemes where the charging power is optimized to minimize the total cost of EVs charging and discharging during the day. The authors prove by simulation that the local scheduling schema

can achieve a close performance to the global one; however, they did not show how the future base loads information are managed especially when using a centralized architecture. In [39], the authors propose a scheduling model in order to regulate the charging of EVs for minimizing load variance in household micro grid. Their model is formulated using an optimization problem and is slacked into a quadratic programming by ignoring the non-linear factors existing in the constraints and solved using in polynomial time. However, some factors have been assumed to be known in advance, such as the load-power curve of home appliances and the time periods when EVs are connected to the micro grid. In [40], the authors propose an optimal centralized scheduling method to jointly control the electricity consumption of home appliances and plug-in EVs; they mathematically formulate the scheduling method as a mixed integer linear programming (MILP) problem. However, [40] assumes just a local scheduling for micro grids. In [41], the authors formulate a multi-objective scheduling problem for the maximization of aggregator's profit, minimization of EV charging cost and maximization of EVs departing with target SOC as the multiple objectives. However, the authors did not extend their work to support the future large scale adoption of EVs into the electrical grid.

2) Cloud based architecture: Only few works (e.g. [43], [44], [45]) have proposed to integrate cloud computing in the smart grid environment. In [43] the authors propose cloud-based information infrastructure for next-generation power grid. The work introduces three specific cloud-enabled power services. The first two services validate how to develop practical compute-intensive and data-intensive power applications by utilizing different layered services provided by the state-of-the-art public cloud platforms. For the third service, the authors propose a cloud-based collaborative direct load control framework in a smart grid and show the merits of the cloud-based information infrastructure. The authors in [44] focus on the procurement of load shifting service by optimally scheduling the charging and discharging of EVs in a decentralized fashion. The optimal scheduling problem is then formulated as a mixed discrete programming (MDP) problem. However, the authors did not use cloud platforms to manage the demands of EVs users. The authors in [45] propose a decentralized algorithm to optimally schedule EV charging. The algorithm exploits the elasticity of electric vehicle loads to fill the valleys in electric load profiles. They formulate the EV charging scheduling problem as an optimal control problem. As in [41], the algorithm requires each EV to solve its local problem. Besides, this work doesn't use V2G technology.

2.1.2 Dynamic Pricing

Several models have been proposed in the context of EVs charging/discharging and pricing scheduling (e.g., [46], [47], [48], [49], [50], [51], [52], [53], [54]). A prediction-based charging strategy for EVs management is discussed in [49]. In this approach, EVs receive price information using wireless communication, and predict the market price during the charging period, in such a way that the time of charging (TOC) price is low. Different dynamic pricing policies are proposed in smart grid usage (D2P) [47], (UDP) [48], quadratic cost function (QCF) [49], and distributed demand response (D2R) [50]. In a recent work [46], the authors propose an intelligent distributed dynamic pricing (D2P) mechanism for EVs charging management. Two pricing models are proposed: home-price and roaming-price; consequently, two types of energy services are considered: home micro grids and foreign micro grids. In this work, the real-time price is decided by micro grids using standard communication architecture for SG, where the scheduling is centralized, in such a case the management of all EVs demands become very difficult especially in peak hours. The authors in [48] propose a usage-based dynamic pricing (UDP) scheme for smart grid. In this work, distributed gateways are used as proxies of the utility company to timely respond the price from customers; nevertheless, the scheduling of EVs demands is performed in a centralized way inside the utility company. In [49], the authors use neural network for piecewise quadratic cost function. In such a pricing model, the grid decides the price depending on the supply power. As a result, customers may have to pay more cost, even though the total demand is low. A distributed demand response algorithm for PHEV charging in smart grids is proposed in [50]. In such pricing models, the costs acquired by the customers directly depend on the demanded energy even though micro grids have excess energy to serve. Thus, clients may not be interested to consume more energy, and, thus, excess energy may be useless.

To schedule the electricity load, the utility company adopts the conventional direct load control (DLC) strategy [51] where smart switches are installed inside of houses such that the house appliances can be turned off during a high-demand period. The DLC enforces the customers to abandon the control of their appliances at certain conditions. Recently, in Ontario, Canada, a time-of-use (TOU) pricing strategy has been widely adopted by utility companies, e.g., Hydro One [52], Waterloo North Hydro [53]. TOU means that the electricity unit price changes according to the time of the day. The Ontario Energy Board (OEB) divides daily and seasonal TOU periods into three categories: off-peak, mid-peak, and on-peak. TOU enables the customers to view the electricity usage online and potentially influences electricity usage

behavior of the customers. Though the period settings of TOU can be updated, TOU is neither truly dynamic nor related to the real-time usage. Therefore, TOU may cause some inappropriate situation.

Yet, to the best of our knowledge, there has been only limited effort to exploit the benefits of SDN for smart grid communications. In [54], the authors studied the application of SDN as an alternative to multiprotocol label switching (MPLS) for wide area communication within the smart grid system. In [55], SDN was applied for developing self-configuration for substations automation, which was tested using emulation of a substation communication network. For smart grid, these applications are of major reputation where robustness of the communication network is a key enable for automated control.

2.1.3 Secure Smart Metering

False data injection attack (FDIA) is considered as one of the most dangerous cyber-physical attacks in smart grid environment [16, 17]. To address these issues, many secure schemes are proposed in the literature. They mainly based on cryptographic, privacy-preserving with differential private data aggregation, key distribution and management [56, 57, 58]. In [56], the authors propose a security-preserving smart metering scheme with fault tolerance for mart grid; the authors extend the lifted ElGamal encryption to aggregate users' consumption reports. The authors in [57] propose DRAFT, a differential private data aggregation with fault tolerance. DRAFT can support fault tolerance of malfunctioning smart meters efficiently and flexibly. In [58], the authors propose a fault-tolerant protocol for smart metering that can handle general communication failures. The authors use a new design of future ciphertexs, and distribute trust among the smart meters by sharing secret keys over them. However, the encryption and key management alone are not sufficient for securing smart electricity metering in smart grid; monitoring solutions are critically necessary to be taken as a counterpart. Like other centralized systems, the traditional AMI system contains schemes and implementation flaws that result in security vulnerabilities. We believe that a secure AMI system based on distributed architecture, for smart grid environment, can provide guarantees regarding the confidentiality, integrity, and availability of its objects (e.g., users and smart meters' privacy, or) [59]. Thus, an integrated strategy to improve the security of smart grid and to deal with cyber physical attacks of the AMI is extremely important.

As learned from the recent literature [60, 61, 62], intrusion detection system (IDS) is frequently adopted as the second line defense besides the first line security measures (e.g.,

encryption, authorization, authentication, firewall, etc). IDS is a monitoring system that can dynamically detect any unwanted entity into a targeted system and trigger alerts for the administrator to take appropriate measures to mitigate the risks and prevent the escalation of malicious activities. The work in [60] proposes a threat model for smart meters based on Petri networks. The authors consider the constrained computation and storage resources of a smart meter, and present a collaborative intrusion detection mechanism against false data injection attack. However, the authors did not take in consideration the users behavior. In [61], the authors present novel intrusion detection and prevention system for ZigBee-based home area networks in smart grid based on machine learning model. The detection module excerpts network features and analyzes them to decide whether the network is in a normal state. Nevertheless, the proposed IDS operates only in the home area network level, and the false data can be injected at different levels of the network; AMI may be not well secured when IDS is implemented only at the home area network.

2.2 Cloud Computing Application for IoT Data Scheduling

According to Xu et *al*. [63], the application of IoT in the industrial automation address to a very small extent. IoT solutions for the industrial automation are still evolving. As explained earlier, the fog-based approaches can meet the requirements of modern industrial systems. However, most of the existing works focus on centralized computation architectures that utilize cloud computing for the purpose of monitoring data and managing control processes in the industrial automation [64], [65]. The majority of existing approaches and solutions in the context of cloud computing in the industrial automation focus on the higher levels than the field-level. In [24], the authors investigate a prototype to explore the use of IoT devices to communicate with a cloud-based controller targeting the automation industry. Nevertheless, this work applies mitigation mechanisms to deal only with the communication delays that are caused by the networks and ignored the computation delays and capacities of cloud servers; also the mitigation model was not validated using a concrete mathematical model.

In a recent work [64], the authors derive an exact expression for the performance of IIoT by combining computation with intelligence. They design an efficient way to obtain a threshold by approximating the performance of different computing manners, and show how to apply it to practical IIoT applications. The proposed work is interesting and gives good results (i.e., we will compare the computing efficiency of our work with this work [64] in the performance evaluation section). However, the authors ignore data transmission delay and synchronization time especially when using distributed computing manner.

Only few works (e.g., [65], [66]) consider the distributed computation feature. In [65], the authors proposed a computing paradigm, named Edge Mesh, which distributes the decision-making tasks among edge devices within the network instead of sending all the data to a centralized server. Edge Mesh provides many benefits, including distributed processing, low latency and fault tolerance. However, the authors ignore the tasks allocation problem in such distributed architecture. A vision of distributed IIoT data processing using fog and cloud computing has also been discussed in a recent work [66]. The authors discuss the emerging challenges in the aspects of data processing in IIoT. They design a flexible framework by integrating the fog computing and cloud computing. Based on the time latency requirements, the collected data are processed and stored by the edge server or the cloud server.

2.3 Cloud-Based SDN Architectures for Vehicular Networks

Although applying SDN in a heterogeneous vehicular network is challenging, several studies in the literature have demonstrated the high potential of SDN to improve data forwarding and to ensure better QoS for vehicular network applications. These studies describe network architectures that can be arranged in two categories: centralized and distributed.

Centralized approaches deploy one SDN controller that is responsible for maintaining the status of all nodes in the network and making packet forwarding decisions. For instance, Ku et al. [67] proposed an SDN-based architecture to select routing paths with high connectivity. Two operational modes were deployed, depending on the control level reinforced by the SDN controller (i.e., total or hybrid). Similarly, He et al. [30] introduced SDVN, another SDN-based architecture that uses traffic information, collected through RSUs, as well as a trajectory prediction module to make forwarding decisions. SDVN also adopts a network slicing mechanism to prevent the broadcast storm problem and supports control flexibility via allowing nodes to deploy routing protocols that exquisitely fit the current context. Azizian et al. [68] proposed an architecture that is based on SDN and cloud computing to tackle the issue of vehicles' software updates dissemination. Indeed, the SDN controller, in collaboration with several cloud elements, uses beacon messages to compute relative mobility between vehicles and to construct connectivity graphs. These graphs serve two purposes: 1) computing routes and updating flow tables; and 2) assigning frequency bands to vehicles to mitigate hidden terminal and interference problems. Abolhasan et al. [69] proposed a hybrid SDN-based architecture that uses two distinct frequency bands to transmit control and data packets. It splits the complexity of route discovery between the SDN controller and the forwarding nodes to reduce significantly the signaling overhead. Finally, Yang et al. [70] proposed a cloud-assisted

architecture, inspired by SDN, to facilitate information dissemination among vehicles. It uses driving histories to build virtual social networks (VSNs) between vehicles. Within each VSN, the closeness between any two vehicles is computed, based on a voting system, and stored in the cloud. Therefore, instead of flooding the network, vehicles only get information that may be of interest to them.

Distributed schemes, on the other hand, use multiple SDN controllers along with cloud/fog services to maintain network state information and to make forwarding decisions. For example, Correia et al. [71] proposed a hierarchical SDN-based vehicular architecture that ensures data dissemination in situations where connection to the SDN controller is lost. Indeed, local SDN domains are built using vehicles clustering. Each cluster head acts as a local controller, retaining some of the network intelligence that is employed to select a set of gateways to forward data packets.

Likewise, Ge et al. [72] proposed a new vehicular network architecture to enhance QoS requirements (i.e., delay, throughput) by integrating 5G mobile communication technologies, SDN, cloud computing and fog computing. Vehicles and RSUs are responsible for collecting traffic information (e.g., speed, density, and road condition). This information is processed by fog computing clusters and transmitted to the SDN controllers, which will forward it to the cloud computing centers. This information is used to: 1) build the global information map; and 2) generate control information based on rules and strategies from the application plane. These rules are then forwarded to RSUs. Finally, Lui et al. [73] proposed another SDN-enabled network architecture assisted by Mobile Edge Computing (MEC) to provide low-latency and high reliability communications. Instead of requesting services from remote data centers, these services can be deployed in cloud computing resources that can be placed near the vehicular network. This way, local data forwarding decisions can be quickly executed, decreasing therefore the round-trip time of data packets.

Despite their adequate performance, schemes in both categories have limitations. On the one hand, the use of only one SDN controller could be the bottleneck of the entire system, leading to resiliency and scalability issues. In addition, adopting the conventional ad hoc mode of vehicular networks whenever connections to the SDN controller are lost may escalate the end-to-end delay. On the other hand, deploying multiple SDN controllers increases the system complexity and introduces new challenges (i.e., handover) that need to be dealt with. This may lead to higher control overhead [71, 72] and longer end-to-end delay [73].

Related Works	Control plane architecture	Multiple controllers	Communication	Controller failover	Resource management	Contribution
Ku et al. 2014 [67]	Centralized Flat	No	V2V	No	Yes	SDN-based VANET architecture
He et al. 2016 [30]	Centralized Flat	No	V2V, V2I	No	No	SDN-based VANET architecture
Azizian et al. 2017 [68]	Distributed Flat	No	V2V	No	No	Cloud-Based VANET SDN Updates
Correia et al. 2017 [71]	Centralized Hierarchical	Yes	V2V, V2I	Yes	No	Hierarchical SDN-based VANET
Ge et al. 2017 [72]	Distributed Flat	Yes	V2I based 5G	No	Yes based on Fog, Cloud	5G-Based SDN for VANET
Lui et al. 2017 [73]	Distributed Flat	Yes	V2V, V2I	No	Yes based on MEC	MEC-Based SDN for VANET
Proposed architecture HSVF	Hybrid Hierarchical	Yes	V2V, V2I, I2I	Yes	Yes Fog, Cloud, NFVM	Hierarchical SDN-Based VANET

Table 2.1- Comparison between HSVF and existing approaches

Table 2.1 illustrates a comparison summary between HSVF and the various existing schemes, including recent ones using different metrics as the architecture of the control plane, the number of controllers, the deployment of a controller failover system, and the ability to manage storage and computational resources considering techniques such as data offloading and resource allocations. Table 1 also shows that while HSVF supports all V2X communications (i.e., V2V, V2I, and I2I). the remaining schemes do not support I2I communications. HSVF deploys a decentralized and hierarchical Fog and Cloud architecture for resource management, and makes use of a computational capacity allocation mechanism using NFVM, which will be described in Section 3.

2.4 5G-based SDN architectures for Vehicular Networks

2.4.1 Network Slice Architetures based SDN

There exists various network slicing approaches that have been proposed in the literature. For example, Costanzo et al. [74] proposed a network slicing solution that makes use of an efficient scheduling algorithm, based on the centralized SDN architecture presented in [35], to ensure a real-time allocation of bandwidth to the different slices considering their requirements in cloud radio access network. Velasco et al. [75] proposed an architecture to enable autonomic slice networking. It has two domains: metro and core, each of which includes: i) a provisioning and reconfiguration module based on SDN; ii) a data analytics module that collects data records from nodes and analyze them; and iii) a slice manager that exports virtualized network resources in the form of network slices through a northbound interface. The approaches proposed in [74]

and [75] are based on a centralized SDN architecture, however, in our work we propose a new core network based on decentralized and scalable SDN architecture.

Ma et al. [76] proposed a management architecture for 5G service-based core network based on SDN and NFV to provide distributed and on-demand deployment of network functions, service guaranteed network slicing, flexible orchestration of network functions and optimal workload allocation. It has two layers: i) service management which is responsible for providing services such as authentication, orchestration and security; and ii) the infrastructure management layer that contains two different SDN controllers: the core SDN which is responsible for deployment and management of network functions, and the flow SDN controller which handles efficient traffic dispatch of the core network. In this way, the core network functions can be deployed in a distributed fashion, easing the deployment of mobile edge computing technology to alleviate the workload on the core network. However, this work focus only on the deployment of core network service function and doesn't give an analysis study of the 5G wireless network, compared with our proposed solution. In Addition, the authors use two SDN layers with two different SDN controllers. In our architecture, we use a hierarchical SDN architecture with three layers (Fog, Edge and Cloud) and four SDN controllers.

Kuklinski et al. [77] proposed DASMO, a distributed autonomous slice management and orchestration framework that addresses the management scalability problem. It uses the ETSI management and orchestration framework (MANO) [78] combined with in-slice management approach, which is a variant of In-Network Management (INM) concept [79], to efficiently manage and orchestrate systems with number of slices. Oladejo et al. [80] proposed a mathematical model to efficiently allocate radio resources in a two-level hierarchical network considering transmit power and allocated bandwidth constraints. The model is based on prioritizing network slices to cater for different users in a multi-tenancy network. The priority of each slice is determined by VMNOs based on which resources are allocated to the different slice while guaranteeing the minimum requirements per slice to ensure user satisfaction.

2.4.2 5G network slice architetures for automative

Few schemes have been proposed to support 5G network slicing for the automotive vertical. For instance, Soenen et al. [81] proposed an SDN-based slicing architecture that addresses the issue of reliability when broadcasting safety messages in vehicular ad hoc networks (VANET). Instead of rebroadcasting a received safety message, a vehicle forwards it to its respective base station. Once received, the centralized SDN controller uses the dedicated slice, based on edge computing, to instruct that particular base station along with all its neighboring base stations to

forward the safety message to all the vehicles in their vicinity. Nevertheless, the use of one centralized SDN controller may provide several and serious problems for VANET. As a mobile wireless network, VANET needs a decentralized deployment of the core network infrastructures. The use of 5G slicing technology to broadcast safety message in this work is good, however, the authors didn't give a latency propagation study of broadcasting VANET safety message using 5G.

Khan et al. [82] proposed a network slicing based communication model for VANET. In this model, an autonomous driving slice deploys a relaying approach that aims at improving the performance of low signal-to-interference-plus-noise-ratio (SINR) video streaming vehicles. This is achieved through a distance-based similarity matrix which adds new access points (i.e., vehicles) to the slice considering link quality (i.e., both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I)) and the neighborhood size (i.e., number of autonomous vehicles in the area).

Campolo et al. [83, 84] proposed a framework that is made of a set of network slices representing different Vehicle-to-Anything (V2X) use case categories. The framework makes use of NFV and SDN technologies to provide different functionalities while meeting the requirements of the various applications. One of these slices is dedicated to autonomous driving. It relies on V2V communication mode as the radio access technology and uses the core network, using a centralized SDN architecture, to provide functions such as authentication, authorization and subscription management. Moreover, it deploys an application server at the network edge to help vehicles in processing 3D maps and building an augmented vision beyond their visual perception. Nevertheless, the works in [83, 84, and 85] didn't provide a mathematical framework to investigate the principal autonomous driving requirements as reliability and ultra-low latency. Moreover, the authors in [83] propose to use only one slice to handle autonomous driving services, which is insufficient due to the important modules of the autonomous driving services (localization, perception, planning, and global system management). These modules need to be virtualized and scheduled using different service slices.

From the reviewed papers, supporting the autonomous driving application in 5G given its inherent requirements (i.e., end-to-end reliability and ultra-low latency) needs further investigation in terms of core network management and orchestration, resource allocation and service differentiation. In order to fulfill the gap in the literature, we propose a framework that contains four components: i) a distributed and scalable SDN core network architecture based on vertical and horizontal partitioning of the network, where each domain is managed by a

single or multiple controllers (i.e., Fog, Edge and Core); ii) a network slicing function that maps autonomous driving functionalities into service slices; and iii) a network and service slicing system model with four logical layers. iv) an ultra-low latency mathematical framework where we used a GI/M/1 queuing system to define the handling, and the tail latencies.

2.5 Chapter Summary

In this chapter, we started by reviewing the different architectures and models proposed to addressee the issues of energy saving in smart grid environment considering the management of EVs energy plug-in as a case study, pricing models for energy cost optimization, and smart metering security against FDIA. Most of these proposed architectures and models rely on central and local topology information based on the traditional smart grid architecture. However, almost all of them endure the local maximum and/or data congestion problems, increasing therefore the end-to-end delay and the response time for scheduling data and users' energy requests (e.g., EVs energy demands for charging). We then described the various mechanisms proposed in the literature to enhance IIoT data scheduling based on cloud computing architectures. Finally, we presented the different proposed Cloud-Based SDN architectures for VANET, also we described recent and new works that propose to integrate 5G slicing technology into vehicular networks.

We will devote the rest of this thesis to our contributions that aim to contribute to the design and the evaluation of new efficient and safe cloud computing architectures that ensure smart city networks requirements (i.e., smart grid, IoT, and VANET). In Chapter 3, we address the application of cloud computing for scheduling EVs energy demands in smart grid. We then present the dynamic pricing model proposed for energy saving in smart grid based distributed Fog architecture in Chapter 4. In Chapter 5, we present the SecMetering scheme proposed to secure smart metering against FDIA. In Chapter 6, we present the proposed hierarchical Fog architecture and model for scheduling industrial IoT data. Finally, Chapter 7 outlines the proposed hierarchical SDN-based wireless vehicular fog architecture, then we present the application of 5G slicing technology for autonomous driving services based on a new SDN core network architecture.

CHAPTER 3

CLOUD-BASED EVs PLUG-IN MODEL IN SMART GRID

Smart Grid (SG) technology represents an unprecedented opportunity to transfer the energy industry into a new era of reliability, availability, and efficiency that will contribute to our economic and environmental health. On the other hand, the emergence of electric vehicles (EVs) promises to yield multiple benefits to both power and transportation industry sectors, but it is also likely to affect the SG reliability, by consuming massive energy. Nevertheless, the plug-in of EVs at public supply stations must be controlled and scheduled in order to reduce the peak load. In order to reduce the impact of charging electrified vehicles (EVs) on smart grid and optimize energy saving; in this chapter, we propose a coordinated model for planning the plug-in of electrified vehicles for charging and discharging energy.

Our contributions are summarized as follows: a) we propose a new plug-in system of EVs based on calendar planning using distributed Fog computing capacities, b) we use a multi-priority queueing system to manage the plug-in of EVs at each electric public supply station (EPSS), the proposed queuing system uses a cut-off discipline in order to enhance to plug-in process, c) in order to improve our planning scheme and maintain the energy curve stability, we propose three optimization algorithms as follows: i) two priority attribution algorithms for both the calendar's plug-in and random plug-in according to different energy constraints, ii) and an algorithm to select the optimal EPSS for each EVs to plug-in, d) we perform extensive simulations using realistic energy loads in order to improve the effectiveness of our solution; also we compare our algorithms with two recent works.

The rest of this chapter is organized as follows. Section 3.1 presents the EVs plug-in system model. Section 3.2 describes the smart grid based decentralized Fog architecture while section 3.3 presents the calendars planning of EVs plug-in based on the proposed Fog architecture. Section 3.4 describes the multi-priority queuing model for EVs plug-in while section 3.5 presents the optimal priority and EPSS attribution strategy. Section 3.6 portrays the simulations results. Finally, Section 3.7 concludes the chapter.

3.1 System Model of EVs Plug-in

In this section, we present the energy consumption system for EVs in a smart city. Primary, we express the EV plug-in problem by defining diverse parameters that contribute in this process (e.g., EVs, EPSS, users, and energy). Next, we define the energy flow between EVs and each micro grid (i.e., G2V, V2G) in order to characterize the demand-supply energy curve for each micro grid.

3.1.1 EVs Plug-in Problem Definition

We consider a geographical area that contains a number n of EVs, and a number m of charging and discharging stations. Each station has a number S of plug-in sockets. The stations are equipped with a bidirectional charge and discharge controller to insure bidirectional G2V-V2G energy flows.

The set $\mathbb{V}$ of electric vehicles ev_i, which performs charging and discharging, is represented as follows:

$$\mathbb{V} = \{ev_1, ev_2, \dots, ev_i, \dots, ev_n\}, \quad \forall\, i = 1, \dots, n \tag{3.1}$$

The set $\mathbb{V}$ involves two sets: i) the charging-only set (i.e., G2V) denoted by $\mathbb{V}^{G2V}$, ii) the discharging-only (V2G) set denoted by $\mathbb{V}^{V2G}$. So we have:

$$\mathbb{V}^{G2V} = \left\{ev_1^{G2V}, \dots, ev_i^{G2V}, \dots, ev_{n'}^{G2V}\right\}, \quad \forall\, i = 1, \dots, n',\ \forall\, ev_i^{G2V} \subset \mathbb{V} \tag{3.2}$$

$$\mathbb{V}^{V2G} = \left\{ev_1^{V2G}, \dots, ev_i^{V2G}, \dots, ev_{n''}^{V2G}\right\}, \quad \forall\, i = 1, \dots, n'',\ \forall\, ev_i^{V2G} \subset \mathbb{V} \tag{3.3}$$

$$\mathbb{V} = \ \mathbb{V}^{G2V} + \ \mathbb{V}^{V2G} \tag{3.4}$$

The set $\mathbb{S}$ of electric public supply stations $EPSS_j$ is represented as follows:

$$\mathbb{S} = \left\{EPSS_1, \dots, EPSS_j, \dots, EPSS_m\right\},\ \forall\, j = 1, \dots, m,\ \ \forall\, EPSS_j \subset \mathbb{S} \tag{3.5}$$

The charging and discharging time of an EV (i.e., service time) is superior to that of a regular vehicle. As illustrated in Figure 3.1, the charging and discharging time is composed of the waiting time to plug-in and the service time. Moreover, the waiting time at public supply stations is more important than the waiting time at home and work stations. In this regard, we study the problem of pug-in EVs at each electric public supply stations, aiming to optimize the waiting time to plug-in at public supply stations.

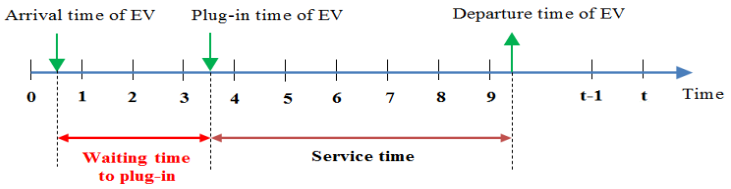


Figure 3.1- Time to plug-in diagram for an EV

3.1.2 Energy Model

We study the battery charging and discharging of EVs during the day, including peak hours, mid-peak hours and no-peak hours, at public supply stations. The day is evenly divided into a set of periods. The period set is denoted by $\mathbb{T}$. The length of a period is denoted by t . We divide the day into 288 periods, so the period length is given by $t = 5$ minutes.

Let us consider a smart grid system, with a micro grid that covers a geographical area. The micro grid provides energy and controls the power consumption of all users, including EVs. The energy demand of charging of an ev_i denoted by $E_{ev_i}^{G2V_t}$, depends on its available energy state of charge at time slot t denoted by $soc_{ev_i}^t$, and the predetermined target charging level denoted by A_i^t. Thus, we have:

$$E_{evV_i}^{G2V_t} = A_i^t - soc_{ev_i}^t \tag{3.6}$$

The total charged energy by EVs at time slot t is given by:

$$T_t^{G2V} = \sum_{i=1}^{n'} E_{ev_i}^{G2V_t} \tag{3.7}$$

Also, the energy discharging of an ev_i denoted by $E_{ev_i}^{V2G_t}$ depends on its state of charge at time slot t, and the discharging target level denoted by B_i^t. Thus, we have:

$$E_{ev_i}^{V2G_t} = soc_{ev_i}^t - B_i^t \tag{3.8}$$

The total energy discharged form EVs at time slot t is given by:

$$T_t^{V2G} = \sum_{i=1}^{n''} E_{ev_i}^{V2G_t} \tag{3.9}$$

Let E_{Sup}^t denotes the total energy supply at time slot t, which includes the energy production by the micro grid denoted by E_{Prod}^t, and the total discharging energy form EVs calculated by Equation (3.9). Thus, we have:

$$E_{Sup}^t = E_{Prod}^t + T_t^{V2G} \tag{3.10}$$

Let E_{Dem}^{t} denotes the total power demand which includes the charging energy from EVs (i.e., T_t^{G2V}), and the base load (home/work) consumption denoted by E_{BL}^{t} . Thus, we have:

$$E_{Dem}^{t} = E_{BL}^{t} + T_t^{G2V} \tag{3.11}$$

In order to characterize the energy demand-supply curve, we calculate the difference and the ratio between the total energy demand E_{Dem}^{t} calculated in Equation (3.11) and the total energy supply E_{Sup}^{t} calculated in Equation (3.10) as follows:

$$D^t = E_{Sup}^{t} - E_{Dem}^{t} \tag{3.12}$$

$$D^t = \frac{E_{Dem}^{t}}{E_{Sup}^{t}} \tag{3.13}$$

3.1.3 EVs Mobility Model

To design the mobility pattern of n EVs in a geographical area, we use the Gauss-Markov mobility model proposed in [85-86]. According to [86], a mobile agent checks its position periodically, and updates the location whenever it reaches a threshold distance. In our case study, EVs are considered as mobile agent; consequently, we use the Gauss-Markov model to implement EVs mobility patterns. The velocity of an EV is considered to be correlated over time, i.e., the location of an EV at time t depends on its location and velocity at time $(t-1)$. In the proposed scenario, the movement of each ev_i is considered to be in two-dimensional plan (x, y). Thus, based on the birth-death stochastic process, the Gaussian-Markov model is represented as in [85-47]:

$$v_t = \alpha v_{t-1} + (1-\alpha)v + \sigma\sqrt{1-\alpha^2}w_{t-1} \; \forall \; t \in T \tag{3.14}$$

$v_t = \left[v_t^x, v_t^y\right]$ denotes the velocity of EV_i at time slot t.
$\alpha = [\alpha^x, \alpha^y]$ denotes the variance of memory level.
$\sigma = [\sigma^x, \sigma^y]$ denotes the standard deviation.
$w_{t-1} = \left[w_{t-1}^x, w_{t-1}^y\right]$ is the uncorrelated Gaussian process.

From Equation (3.14), we evaluate the mobility model in two-dimensional plan (x, y), respectively as follows:

$$v_t^x = \alpha v_{t-1}^x + (1-\alpha)v^x + \sigma^x\sqrt{1-\alpha^2}w_{t-1}^x \; \forall \; t \in T \tag{3.15}$$

$$v_t^y = \alpha v_{t-1}^y + (1-\alpha)v^y + \sigma^y\sqrt{1-\alpha^2}w_{t-1}^y \; \forall \; t \in T \tag{3.16}$$

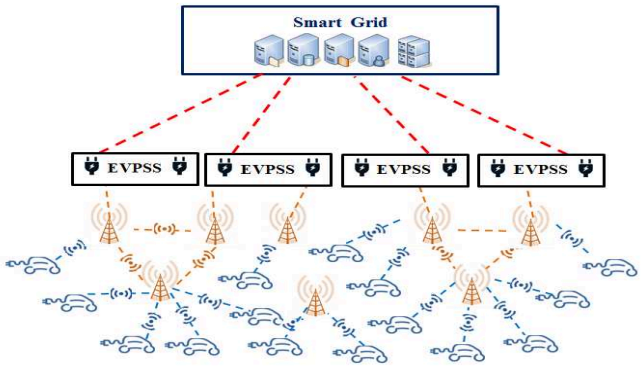


Figure 3.2- Conventional (Centralized) smart grid architecture

In the proposed model, we assume different locations of an ev_i (e.g., home, office, roadside, parking deck, and shopping mall) with three different values of α: 0, 1, and $0 < \alpha < 1$. Where α is a parameter which can reflect the randomness of Gauss-Markov process.

3.2 Smart Grid Based Decentralized Fog Architecture

We believe that the centralized scheduling of n EVs requests with their heterogeneous battery characteristics, and the management of m charging and discharging stations including base load consumption, in a geographical area, is challenging. In the centralized architecture, all information and operations are scheduled inside the micro grid. With the limited storage and computing capacity of the smart grid, also with the increasing number of EVs, the real-time demand response and energy monitoring become very problematic.

3.2.1 Traditional Smart Grid Architecture

Fundamentally, the traditional architecture of SG is designed in a centralized way where the data generated from various data sources (e.g., smart meters, EVs, EVPS) are typically examined at a centralized control center (i.e., smart grid controller) [87]. Taking EVs and EVPSSs as an example (Figure 3.2), EPPS acquires metering data and requests from EVs, and delivers them to an operation center where a meter data management system (MDMS) and scheduling applications are used to process the data and response to EVs request [88]. Using such centralized architecture, smart grid management systems receive and process big datasets and demands from all customers, which seems an easy way of managing data only in some simple situations (e.g., small-scale and non-mobile systems).

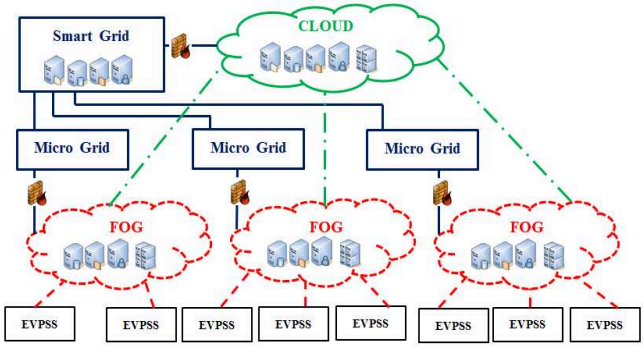


Figure 3.3- Decentralized Fog for smart grid communications

3.2.2 Why using decentralized cloud for smart grid

With the growing number of EVs fleet and electric de-vices, it is highly likely that the traditional centralized architecture reaches its physical limit of handling these big data. Such architecture can cause data congestion, serious latency, or even data loss, which may seriously damage SG services [38]. Hence, it is very critical for electric companies to define new communications standards and find the best computing infrastructure to control and manage all costumers. To address these problems, we propose a new concept of distributed computing architecture for SG, where the case study is the scheduling of EVs charging/discharging requests (see Figure 3.3). We strongly trust that a decentralized aspect of cloud computing (i.e., fog) is an encouraging solution thanks to the following advantages:

- Cloud technology provides high computing capacity to smart grid utilities. In addition, with its enormous data storage, cloud can help smart grid to improve demand side management services.
- The decentralized aspect of cloud networking, using the so-called Fog-computing, can be used to manage and control each micro grid in a distributed manner.

3.2.3 Fog-assisted vehicles demands management

We propose a decentralized fog computing architecture for an intelligent transportation system by scheduling EVs charging/discharging requests in SG environment. In the proposed scheme, multiple fog data centers are deployed in a smart city and connected to each micro grid controller; each distributed fog data center plays the role of serving one local area (i.e., one micro grid) and reporting to a centralized cloud data centers. In addition, the distributed operation center is capable of collecting and analyzing data locally, and making decisions (with localized goal settings) rather than backhauling all data to the central operation center. Such

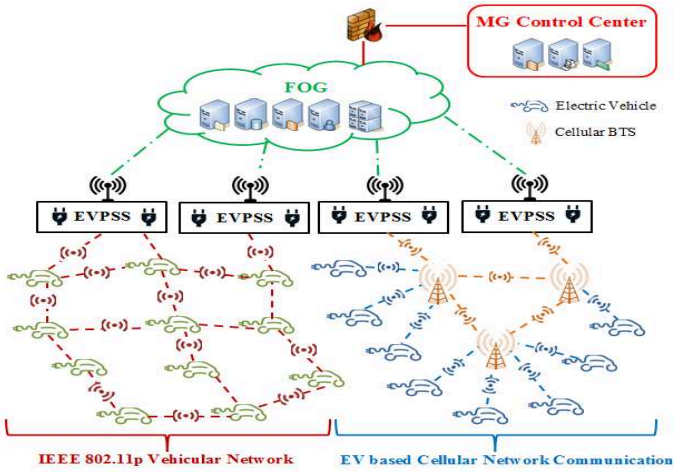


Figure 3.4- Fog architecture for EVs charging/discharging service

localized intelligence helps to reduce large volumes of raw datasets to smaller amounts of valuable and manageable data, and thus reduce communication burden and processing cost.

The proposed distributed V2G communication architecture contains two networks layers as follows:

- *Lower layer:* current cellular networks can be a good option for connecting EVs and EPPS in addition to the conventional IEEE 802.11p vehicular network can be used to route data [89] (see Figure 3.4). Additionally, cellular networks solutions also enable EPSS deployments spreading to a wide area environment, where each distributed EPSS communicates with fog data centers via wireless communication technologies (e.g., 4G, Wi-Fi and future 5G), and then the metering data and EVs requests are stored and processed locally at the distributed operation center of the appropriate micro grid.
- *Upper layer:* the distributed fog data centers communicate with the central cloud data center through a back-bone network (core network IP/MPLS). With localized management and implemented scheduling algorithms at the decentralized fog data centers, only summary information and required integrated information are transmitted to the central operation center.

Indeed, each fog data center and cloud data center can communicate respectively with the micro grid controller and the smart grid controller in order to update periodically energy parameters, such as price, total energy supply, etc. Figure 3.5 shows the proposed packet format of EVs, EVPSS and smart grid controller interactions.

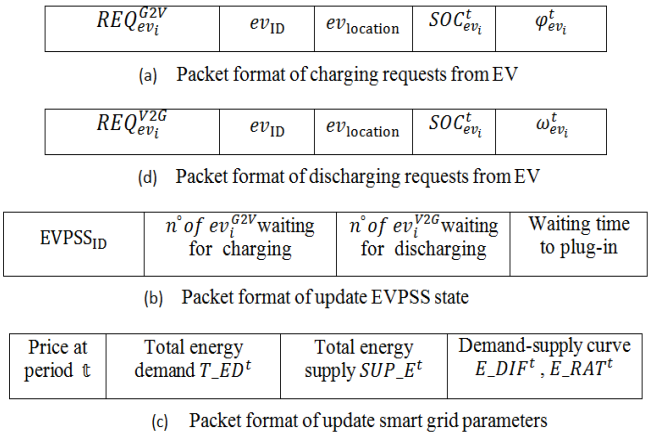


Figure 3.5- Packet format of EVs, EVPSS and smart grid communications

3.3 Calendars Planning of EVs Plug-in based on Decentralized Fog

In order to schedule the plug-in process of EVs, we use in our proposed solution the proposed architecture based on decentralized Fog (Figure 3.3), where each micro-grid has its appropriate Fog data centers. We use Fog computing capabilities (i.e., storage and processing capacities) to schedule all EVs requests. To give back this usage more efficiency and proficiency in our model, EVs users are solicited to create and save their EVs energy profiles $U_{profile}$. An $U_{profile}$ includes information about EVs users as represented below:

$$U_{Profile}\{User_{name}, User_{password}, ev_{ID}, Home_{Adress}, Battery_{type}, Battery_{efficiency}\}$$

Where ev_{ID} is the unique identifier of an electric vehicle, $User_{password}$ is used for session authentication to ensure privacy and security, $Battery_{type}$ and $Battery_{efficiency}$ are used to calculate the charging and discharging time.

EV users may create several calendars to express several charging and/or discharging requests. EV calendars for charging and discharging are represented, respectively, as follows:

$$ev_{Calendar}^{G2V}\ [ev_{ID}, ev_{location}, soc_{ev_i}^t, time\ to\ plugin\ t,\ A_i^t]$$

$$ev_{Calendar}^{V2G}\ [ev_{ID}, ev_{location}, soc_{ev_i}^t, time\ to\ plugin\ t,\ B_i^t]$$

Each EV can update several charging and discharging calendars, once accessing its profile in the cloud, by indicating its location ($ev_{location}$), state of charge $soc_{ev_i}^t$, and the estimated time to plug-in t. Moreover, it can indicate its requested energy for charging (A_i^t) and for discharging (B_i^t). Once scheduling of EVs calendars, inside each Fog computing data center, the suitable and the optimal calendars with the best waiting time to plug-in, and the best energy

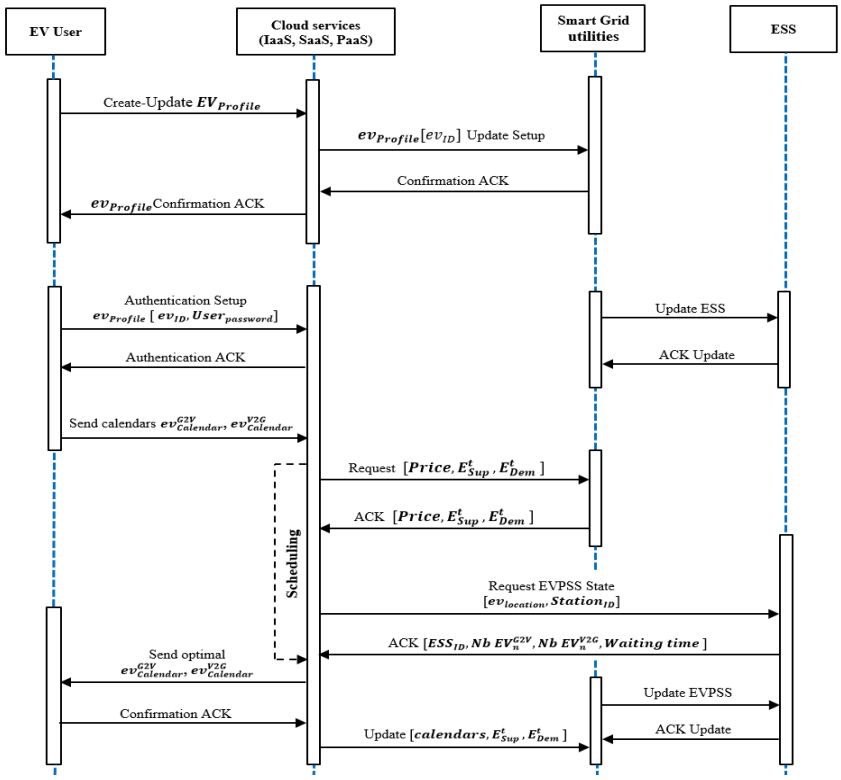


Figure 3.6- Sequence diagram of scheduling calendars

consumption profile for smart grid utilities are sent to all EVs. Figure 3.6 shows the packets format of messages exchanged between EVs, EPSS, smart grid, and cloud providers, and the schematic diagram of the proposed communication architecture based on smart grid and cloud computing interactions. It illustrates the exchange process between EVs, cloud data centers, EPSS, and smart grid to ensure the EVs charging and discharging operations. This is performed as follows:

- First, each EV user creates his personal profile; EV uses cloud as software as a service, to update its information through web-applications. The EVs profiles are saved inside cloud data centers using IaaS. The profile is confirmed in the smart grid through the communication with cloud providers.
- Now, EV users are able to create several charging and discharging calendars after having session authentication. Using the storage capacity of IaaS, all EVs calendars are saved inside cloud data centers.
- Cloud computing service (i.e., PaaS) delivers to smart grid providers a programming model to schedule all EVs calendars inside the cloud. Using PaaS, the run time execution is controlled and can be reduced and optimized. During the scheduling process, communication messages can be exchanged between cloud services, EPSS, and smart grid

utilities, in order to update and collect information in a real-time way; consequently, the best decision can be outlined.

- Once the scheduling process is performed, the optimal calendars are selected for each EV user. The user receives a notification message of the suitable calendar on his profile, where he can confirm or decline the proposed calendar. If the user confirms the charging and/or discharging calendar, he sends a notification message to the cloud in order to update information (e.g. calendars data base, demand-supply curve and price). Otherwise, the user selects his appropriate calendar.

In order to make our model more realistic, we suppose that we may have some EVs users that cannot create profiles or haven't sent calendars to plane their plug-in, we called this EVs as EVs with random plug-in. In summary, we will have four types of EVs how's perform the charging and/or discharging process. We divided the EVs into four sets as follows: 1) The set of EVs with calendar plug-in who request charging, 2) The set of EVs with calendars plug-in who request discharging, 3) The set of EVs with random plug-in who request charging, and 4) The set of EVs with random plug-in who request discharging.

3.4 Multi-Priority Queuing Model for EVs Plug-in

In this section, we study the plug-in process of EVs at each public supply station (EPSS). We use a multi-priority queueing concept with cut-off discipline. As seen earlier, we have four sets of EVs that can be in the same public EPSS at time instance t to perform charging or discharging process, thus we consider four priority levels. Each EV calendar receives its priority level to plug-in according to different metrics and condition (we will see this in the next section).

However, in order to control and manage all EVs sets, we assume that EVs with random plug-in receive their priority levels when they arrive at the EPSS. The Fog services can communicate with each ESS controller to attribute the appropriate priority levels to EVs with random plug-in according to the scheduling state.

Taking into consideration the four sets of EVs, our priority queuing theory with four priority levels is seen as a multi-priority M/M/S queuing model for each EPSS where each EPSS has a number s of plug-in sockets. In this regard, our stochastic process is presented as a Markov chain with four priorities in each state transition. We noted the priority levels $\mathcal{L}$ as follows:

$$\mathcal{L} = \{1, 2, 3, 4\} \text{ where, } (\mathcal{L} = 1) > (\mathcal{L} = 2) > (\mathcal{L} = 3) > (\mathcal{L} = 4) \qquad (3.17)$$

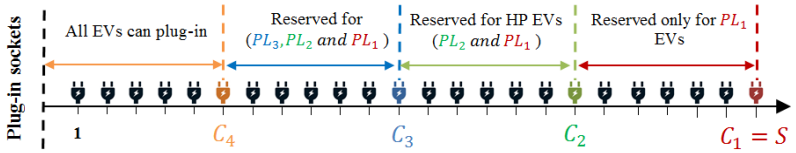


Figure 3.7- Cut-off priority mechanism for one public EPSS

By convention, $(\mathcal{L} = 1)$ designates highest priority.

3.4.1 Cut-off Discipline for M/M/S Queuing Model

In order to organize the charging and discharging process of each EV user, we use a server cut-off discipline for each EPSS. Under this discipline, we anticipate EVs with highest priorities relative to EVs with lowest priorities; in order to serve EVs arrivals with highest priorities. The cut-off discipline is shown in Figure 3.7.

Associated with each priority level $\mathcal{L}$, there is a server cut-off number $C_{\mathcal{L}}$, where $\mathcal{L} = 1, 2, 3, 4$. If a priority $\mathcal{L}$ customer finds upon arrival fewer than $C_{\mathcal{L}}$ servers busy, then he enter service immediately. Otherwise (i.e., if $C_{\mathcal{L}}$ or more servers are busy upon arrival), the customer joins an first-in-first-out (FIFO) queue of his priority class. In order to describe the relative thresholds (cut-off) of the respective priority levels, we assume that $C_{\mathcal{L}}$ are ordered as follows:

$$C_4 \leq C_3 \leq C_2 \leq C_1 \tag{3.18}$$

Where the indexes 4, 3, 2 and 1 are the priority levels.

However, if two or more priority classes have the same server cut-off number (i.e., $C_{\mathcal{L}+1} = C_{\mathcal{L}}$), then all the higher priority customers are saved in a head-of-line manner before any of lower priority customers. In this work, EVs with lowest priority receive service (i.e., to plug-in) only when the number of EVs with highest priority are waiting is null. Nevertheless, an EV with low priority that is receiving service is not interrupted, if an EV with higher priority arrives and all sockets are occupied; thus the queue discipline is no-preemptive. Let us consider a number β of busy plug-in sockets in each EPSS. As shown in Figure 3.7, the admission of each EVs category is scheduled according to the following conditions:

- For EVs with priority level 4 $(\mathcal{L} = 4)$: (i) serve an arriving EV if and only the number of occupied plug-in sockets is fewer then C_4, (i.e., $\beta < C_4$), otherwise, place it in the appropriate queue. (ii) Serve an EV from the queue whenever one of busy plug-in sockets $(< C_4)$ becomes free.

- For EVs with priority level 3 ($\mathcal{L} = 3$): (i) serve an arriving EV if and only the number of occupied plug-in sockets is fewer then C_3, (i.e., $\beta < C_3$), otherwise, place it in the appropriate queue. (ii) Serve an EV from the queue whenever one of busy plug-in sockets ($< C_3$) becomes free.
- For EVs with priority level 2 ($\mathcal{L} = 2$): (i) serve an arriving EV if and only the number of occupied plug-In sockets is fewer then C_2, (i.e., $\beta < C_2$), otherwise, place it in the according priority queue. (ii) Serve an EV from the queue whenever one of busy plug-in sockets ($< C_2$) becomes free.
- For EVs with priority level 1 ($\mathcal{L} = 1$): an EV enters service immediately unless all plug-in sockets are busy ($\beta = C_1 = S$), in which case it can be queued.

The multi-priority queuing model with cut-off discipline is motivated by the concept of charging and discharging system at public EPSS, in which it is desirable to retrain a strategic reserve of plug-in sockets for higher priority customers.

3.4.2 Workload Parameters and Steady-State Probabilities

We define the workload parameters as follows:

$$\begin{cases} \lambda = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \\ \mu = \mu_1 = \mu_2 = \mu_3 = \mu_4 \\ \rho_1 = \frac{\lambda_1}{\mu}, \rho_2 = \frac{\lambda_2}{\mu}, \rho_3 = \frac{\lambda_3}{\mu}, \rho_4 = \frac{\lambda_4}{\mu} \\ \rho = \rho_1 + \rho_2 + \rho_3 + \rho_4 \end{cases} \tag{3.19}$$

We define also the steady-state probabilities as follows:

$$\pi(N_1, N_2, N_3, N_4, \beta) = \begin{cases} \Pr[N_1 & EVs\ waiting, \\ N_2 & EVs\ waiting, \\ N_3 & EVs\ waiting, \\ N_4 & EVs\ waiting, \\ \beta & Sockets\ busy] \end{cases} \tag{3.20}$$

Where, $N_1 \geq 0, N_2 \geq 0, N_3 \geq 0, N_4 \geq 0$.

Otherwise, $$\pi(N_1, N_2, N_3, N_4, \beta) = 0 \tag{3.21}$$

And $$\pi(0,0,0,0,0) = \pi_0^c \tag{3.22}$$

The steady-state equations for our model are given by the following:

- For ($N_4 > 0$) (Figure 3.8 (a))

1) If $\beta \leq C_4$

$$(\lambda + 2\mu C_4)\pi(0,0,0,N_4,C_4) = (\lambda_1 + \lambda_2 + \lambda_3)\pi(0,0,0,N_4,C_4 - 1) + \lambda_4\pi(0,0,0,N_4 - 1,C_4) + \mu(C_4 + 1)\pi(0,0,0,N_4,C_4 + 1) + \mu C_4\pi(0,0,0,N_4 + 1,C_4) \tag{3.23}$$

2) If $C_4 + 1 \leq \beta \leq C_3 - 1$

$$(\lambda + 2\mu C_3)\pi(0,0,0,N_4,C_3) = (\lambda_1 + \lambda_2 + \lambda_3)\pi(0,0,0,N_4,C_3 - 1) + \lambda_4\pi(0,0,0,N_4 - 1,C_3) + \mu(C_3 + 1)\pi(0,0,0,N_4,C_3 + 1) + \mu C_3\pi(0,0,0,N_4 + 1,C_3) + \mu C_3\pi(0,0,1,N_4,C_3) \quad (3.24)$$

3) If $C_3 + 1 \leq \beta \leq C_2 - 1$

$$(\lambda_1 + \lambda_2 + \lambda_4 + 2\mu C_2)\pi(0,0,0,N_4,C_2) = (\lambda_1 + \lambda_2)\pi(0,0,0,N_4,C_2 - 1) + \lambda_4\pi(0,0,0,N_4 - 1,C_2) + \mu(C_2 + 1)\pi(0,0,0,N_4,C_2 + 1) + \mu C_2\pi(0,1,0,N_4,C_2) + \mu C_2\pi(0,0,1,N_4,C_2) + \mu C_2\pi(0,0,0,N_4 + 1,C_2) \quad (3.25)$$

4) If $C_2 + 1 \leq \beta \leq C_1 - 1$

$$(\lambda_1 + \lambda_4 + 2\mu C_1)\pi(0,0,0,N_4,C_1) = \lambda_1\pi(0,0,0,N_4,C_1 - 1) + \lambda_4\pi(0,0,0,N_4 - 1,C_1) + \mu C_1\pi(1,0,0,N_4,C_1) + \mu C_1\pi(0,1,0,N_4,C_1) + \mu C_1\pi(0,0,1,N_4,C_1) + \mu C_1\pi(0,0,0,N_4 + 1,C_1) \quad (3.26)$$

- For ($N_3 > 0$) (Figure 3.8 (b))

1) If $\beta \leq C_3$

$$(\lambda + 3\mu C_3)\pi(0,0,N_3,N_4,C_3) = (\lambda_1 + \lambda_2)\pi(0,0,N_3,N_4,C_3 - 1) + \lambda_3\pi(0,0,N_3 - 1,N_4,C_3) + \lambda_4\pi(0,0,N_3,N_4 - 1,C_3) + \mu(C_3 + 1)\pi(0,0,N_3,N_4,C_3 + 1) + \mu C_3\pi(0,0,N_3 + 1,N_4,C_3) + \mu C_3\pi(0,0,N_3,N_4 + 1,C_3) \quad (3.27)$$

2) If $C_3 + 1 \leq \beta \leq C_2 - 1$

$$(\lambda_1 + \lambda_3 + \lambda_4 + 3\mu C_2)\pi(0,0,N_3,N_4,C_2) = \lambda_1\pi(0,0,N_3,N_4,C_2 - 1) + \lambda_3\pi(0,0,N_3 - 1,N_4,C_2) + \lambda_4\pi(0,0,N_3,N_4 - 1,C_2) + \mu(C_2 + 1)\pi(0,0,N_3,N_4,C_2 + 1) + \mu C_2\pi(0,1,N_3,N_4,C_2) + \mu C_2\pi(0,0,N_3 + 1,N_4,C_2) + \mu C_2\pi(0,0,N_3,N_4 + 1,C_2) \quad (3.28)$$

3) If $C_2 + 1 \leq \beta \leq C_1 - 1$

$$(\lambda_1 + \lambda_3 + \lambda_4 + 3\mu C_1)\pi(0,0,N_3,N_4,C_1) = \lambda_1\pi(0,0,N_3,N_4,C_1 - 1) + \lambda_3\pi(0,0,N_3 - 1,N_4,C_1) + \lambda_4\pi(0,0,N_3,N_4 - 1,C_1) + \mu C_1\pi(1,0,N_3,N_4,C_1) + \mu C_1\pi(0,1,N_3,N_4,C_1) + \mu C_1\pi(0,0,N_3 + 1,N_4,C_1) + \mu C_1\pi(0,0,N_3,N_4 + 1,C_1) \quad (3.29)$$

- For $(N_2 > 0)$ (Figure 3.8 (c))

1) If $\beta \leq C_2$

$$(\lambda + 4\mu C_2)\pi(0, N_2, N_3, N_4, C_2) = \lambda_1\pi(0, N_2, N_3, N_4, C_2 - 1) + \lambda_2\pi(0, N_2 - 1, N_3, N_4, C_2) + \lambda_3\pi(0, N_2, N_3 - 1, N_4, C_2) + \lambda_4\pi(0, N_2, N_3, N_4 - 1, C_2) + \mu(C_2 + 1)\pi(0, N_2, N_3, N_4, C_2 + 1) + \mu C_2\pi(0, N_2 + 1, N_3, N_4, C_2) + \mu C_2\pi(0, N_2, N_3 + 1, N_4, C_2) + \mu C_2\pi(0, N_2, N_3, N_4 + 1, C_2) \quad (3.30)$$

2) If $C_2 + 1 \leq \beta \leq C_1 - 1$

$$(\lambda + 4\mu C_1)\pi(0, N_2, N_3, N_4, C_1) = \lambda_1\pi(0, N_2, N_3, N_4, C_1 - 1) + \lambda_2\pi(0, N_2 - 1, N_3, N_4, C_1) + \lambda_3\pi(0, N_2, N_3 - 1, N_4, C_1) + \lambda_4\pi(0, N_2, N_3, N_4 - 1, C_1) + \mu C_1\pi(1, N_2, N_3, N_4, C_1) + \mu C_1\pi(0, N_2 + 1, N_3, N_4, C_1) + \mu C_1\pi(0, N_2, N_3 + 1, N_4, C_1) + \mu C_1\pi(0, N_2, N_3, N_4 + 1, C_1) \quad (3.31)$$

- For $(N_1 > 0)$ (Figure 3.8 (d))

1) If $\beta \leq C_1$ (Fig. 2. (d))

$$(\lambda + 4\mu C_1)\pi(N_1, N_2, N_3, N_4, C_1) = \lambda_1\pi(N_1 - 1, N_2, N_3, N_4, C_1) + \lambda_2\pi(N_1, N_2 - 1, N_3, N_4, C_1) + \lambda_3\pi(N_1, N_2, N_3 - 1, N_4, C_1) + \lambda_4\pi(N_1, N_2, N_3, N_4 - 1, C_1) + \mu C_1\pi(N_1 + 1, N_2, N_3, N_4, C_1) + \mu C_1\pi(N_1, N_2 + 1, N_3, N_4, C_1) + \mu C_1\pi(N_1, N_2, N_3 + 1, N_4, C_1) + \mu C_1\pi(N_1, N_2, N_3, N_4 + 1, C_1) \quad (3.32)$$

Figure 3.8 shows that π_β, the probabilities that there are β servers busy, represent the total probability of being in a hyperplane orthogonal to the β-axis and passing through $\pi(0, 0, 0, 0, \beta)$. Indeed, the state transition diagrams represented in Figure 3.8 is ($\mathcal{L}$ +1)-dimensional (i.e., Five-dimensional diagram in our case with 4 priority levels). For the 4-priority problem, and using "balance-of-flow" results from M/M/S queues, it is easy to see that:

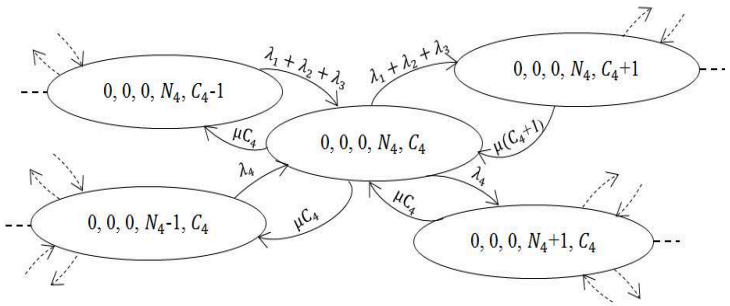
For $0 < \beta < C_4$ $$\pi_\beta = \pi_{\beta-1}\frac{\rho}{\beta} \quad (3.33)$$

Where $\rho = \sum_{\mathcal{L}=1}^{4}\frac{\lambda_{\mathcal{L}}}{\mu}$.

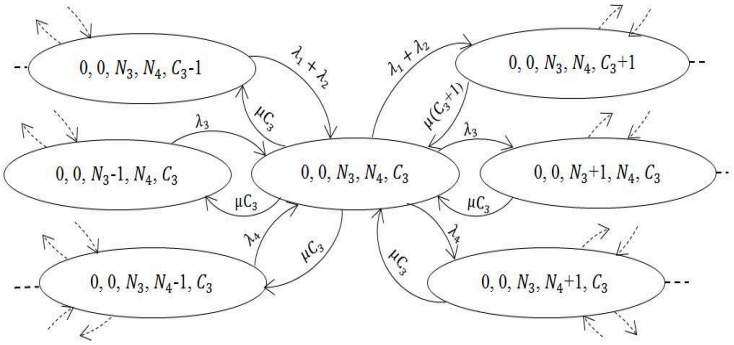
Similarly, it is easy to see by summing the global balance equations over $N_{\mathcal{L}}$ that:

For $C_{\mathcal{L}+1} < \beta < C_{\mathcal{L}}$ and $1 \leq \mathcal{L} < 4$ $$\pi_\beta^c = \pi_{\beta-1}^c\frac{\rho_{\mathcal{L}}^c}{\beta} \quad (3.34)$$

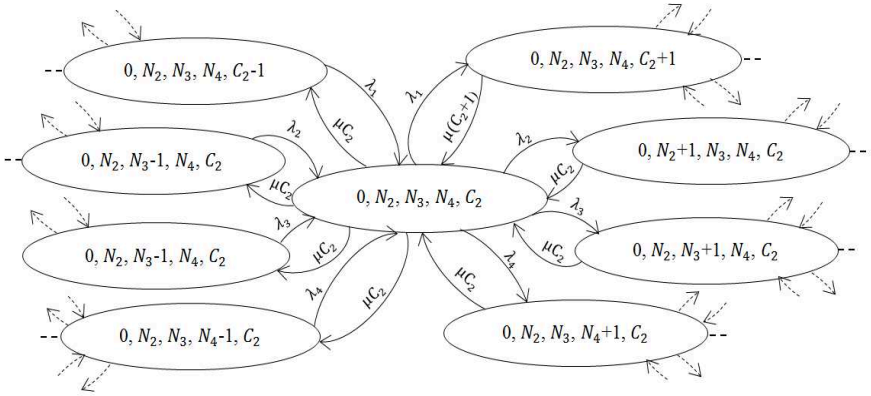
Where $\rho_{\mathcal{L}}^c = \sum_{j=1}^{\mathcal{L}}\frac{\lambda_j}{\mu}$.



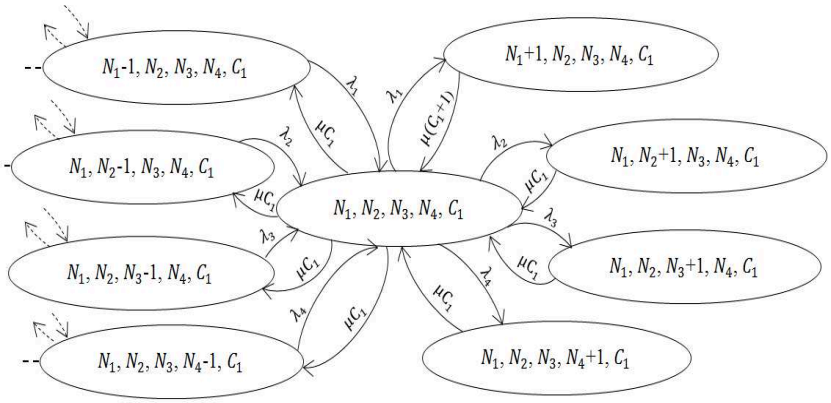
(a) For $N_4 > 0$ and $\beta \leq C_4$



(b) For $N_3 > 0$ and $\beta \leq C_3$



(c) For $N_2 > 0$ and $\beta \leq C_2$



(d) For $N_1 > 0$ and $\beta \leq C_1$

Figure 3.8- The State transition diagram for four priority levels with cut-off thresholds

Equations (3.33) and (3.34) show that, to obtain the steady state probabilities π_β, it is necessary to determine the relationship between $\pi_{c_{\mathcal{L}}-1}$ and $\pi_{c_{\mathcal{L}}}$, for $\mathcal{L} = 1, \dots, 4$. By normalizing the sum of π_β to 1 then obtaining π_0. In order to derive the probability distribution for number of busy sockets, we need to introduce some additional concepts and notations.

Let $R_\beta^{\mathcal{L}}$ denotes the first passage time from state S_n to S_{n-1} $n = (1,2, \dots, S)$ with arrival streams of priority $\mathcal{L}$. Figure 3.9, illustrates the definitions of the random variable $R_\beta^{\mathcal{L}}$, for $1 \leq \mathcal{L} \leq 4$ and $C_{\mathcal{L}+1} \leq \beta \leq C_{\mathcal{L}}$ for our model with four priority levels where $C_4 \leq C_3 \leq C_2 \leq C_1 = S$. The continued arcs show the first passage time $R_n^{\mathcal{L}}$ from one state to another. The discontinued arcs show the conditional upward transition rates for a given number of busy sockets as follow:

$$\lambda_{\mathcal{L}}^c = \sum_{k=1}^{\mathcal{L}} \lambda_k \tag{3.35}$$

3.4.3 Mean Waiting Time for Each Priority Queues

Now, let $B_{\mathcal{L}}$ (for each priority level of a customer) denotes the queue move-up time. Due to the Markovian nature of the entire system, it is clear that the $B_{\mathcal{L}}$'s are independent. The move up time is equivalent to a first passage time from one state to another. The steady state probabilities of our model with four priority levels are given by [90-91]. Using the recursive method and the derivation of the π_β that requires the knowledge of the mean first time passage time $E[R_n^{\mathcal{L}}]$ for a certain combination of indices β and $\mathcal{L}$, we derive these quantities recursively as seen in Figure 3.10.

We can now write down the π_β by inspection, in terms of the expected service times seen by the various queues. For a four priority system with arrival rates $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, individual service rate μ, and cut-offs $C_4 \leq C_3 \leq C_2 \leq C_1 = S$ the steady state probabilities are as follows:

- If $0 \leq \beta < C_4$

$$\pi_\beta^c = \pi_0^c \frac{(\rho_4^c)^\beta}{\beta!} \tag{3.36}$$

- If $C_4 \leq \beta < C_3$

$$\pi_\beta^c = \pi_0^c \left(\frac{\rho_4^c}{\rho_3^c}\right)^{C_4} \frac{(\rho_3^c)^\beta}{\beta!} \frac{1}{1-\lambda_4 E[B_4]} \tag{3.37}$$

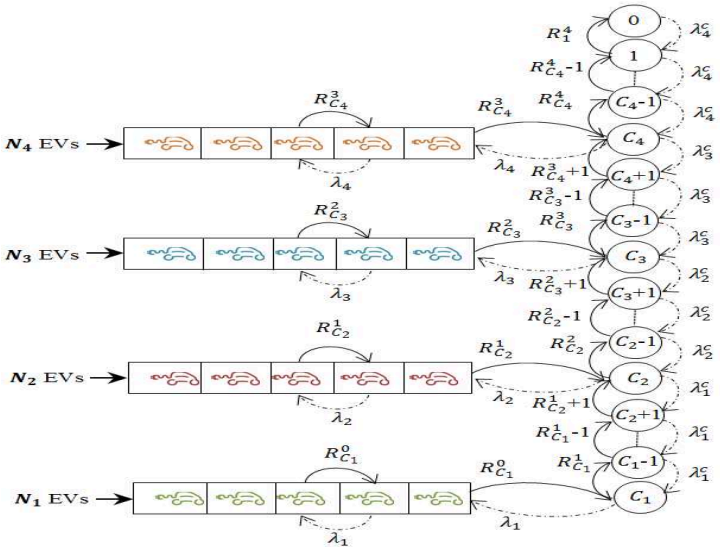


Figure 3.9 Illustration of the random variable $R_\beta^{\mathcal{L}}$, for $1 \le \mathcal{L} \le 4$ and $C_{\mathcal{L}+1} \le \beta \le C_{\mathcal{L}}$ with four priorities

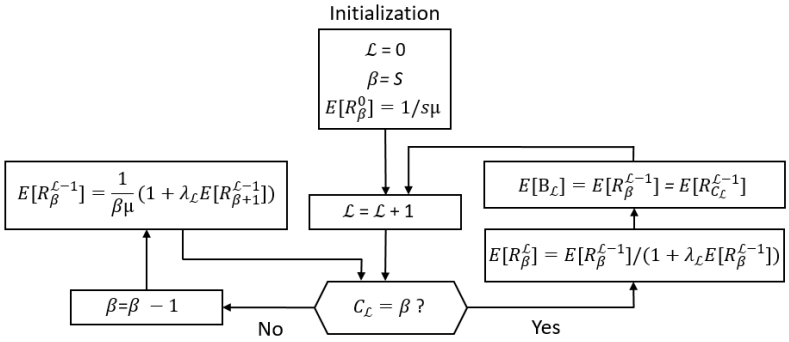


Figure 3.10 Flow Chart for Computing Mean First Passage Times and Mean Queue Move up Times

- If $C_3 \le \beta < C_2$

$$\pi_\beta^c = \pi_0^c \left(\frac{\rho_4^c}{\rho_3^c}\right)^{C_4} \left(\frac{\rho_3^c}{\rho_2^c}\right)^{C_3} \frac{(\rho_2^c)^\beta}{\beta!} \frac{1}{1-\lambda_4 E[B_4]} \frac{1}{1-\lambda_3 E[B_3]} \tag{3.38}$$

- If $C_2 \le \beta < C_1$

$$\pi_\beta^c = \pi_0^c \left(\frac{\rho_4^c}{\rho_3^c}\right)^{C_4} \left(\frac{\rho_3^c}{\rho_2^c}\right)^{C_3} \left(\frac{\rho_2^c}{\rho_1^c}\right)^{C_2} \frac{(\rho_1^c)^\beta}{\beta!} \frac{1}{1-\lambda_4 E[B_4]} \frac{1}{1-\lambda_3 E[B_3]} \frac{1}{1-\lambda_2 E[B_2]} \tag{3.39}$$

- If $\beta = C_1$

$$\pi_\beta^c = \pi_0^c \left(\frac{\rho_4^c}{\rho_3^c}\right)^{C_4} \left(\frac{\rho_3^c}{\rho_2^c}\right)^{C_3} \left(\frac{\rho_2^c}{\rho_1^c}\right)^{C_2} \frac{(\rho_1^c)^\beta}{\beta!} \frac{1}{1-\lambda_4 E[B_4]} \frac{1}{1-\lambda_3 E[B_3]} \frac{1}{1-\lambda_2 E[B_2]} \frac{1}{1-\lambda_1 E[B_1]} \tag{3.40}$$

Thus, the mean waiting time for each priority queues $\mathcal{L}$ for our four-priority M/M/S system is given by:

$$E[w_{\mathcal{L}}] = \frac{\pi_{\beta}^{c}/S\mu}{\left(1-\frac{1}{S\mu}\sum_{k=1}^{\mathcal{L}-1}\lambda_k\right)\left(1-\frac{1}{S\mu}\sum_{k=1}^{\mathcal{L}}\lambda_k\right)} \quad (3.41)$$

In order to allow a good saving of computation effort. We use for a number M of EPSS a form of block matrix notation to highlight the waiting time and the cut-off threshold vectors for all ESS, which are given as follows:

$$Vect\_E[w_{\mathcal{L}}] = \{E[w_{\mathcal{L}}]_1, \dots, E[w_{\mathcal{L}}]_j, \dots, E[w_{\mathcal{L}}]_M\} \quad (3.42)$$

$$Vect_{C_1} = C_1 = s \quad \text{For } j = 0 \dots M \quad (3.43)$$

$$Vect\_C_{2\dots4} = \{(C_{2\dots4})_1, \dots, (C_{2\dots4})_2, \dots, (C_{2\dots4})_M] \quad (3.44)$$

3.5 Optimal Priority and EPSS Attribution Strategy

3.5.1 Optimal Priority Algorithms

In this section, we study the priority assignment strategy implemented in our approach. Each priority level is attributed to an EV according to: 1) the EV type (i.e., with calendar plug-in or random plug-in), 2) the type of requests (i.e., charging or discharging), 3) the demand supply-curve state, 4) the waiting time at each EVPSS, and 5) the mobility model and EV location.
We consider four priority levels; in our priorities attribution strategy, we use the supply-demand curve characterized by Equation (3.12) as a principal constraint. Thus, the priorities attribution is given as follows:

- If $E_{Dem}^{t} \leq E_{Sup}^{t}$ (i.e., the value of Equation (3.12) is positive): EVs which request to charge their battery receive the first and the second priority levels ($\mathcal{L} = 1, \mathcal{L} = 2$) as follows: ($ev_{Calendar_{\mathcal{L}}}^{G2V} = 1$) for EVs with calendar plug-in, and ($ev_{Random_{\mathcal{L}}}^{G2V} = 2$) for EVs with random plug-in. However, EVs which request to discharge their battery receive the low priority levels ($\mathcal{L} = 3, \mathcal{L} = 4$) as follows: ($ev_{Calendar_{\mathcal{L}}}^{V2G} = 3$) for EVs with calendar plug-in and ($ev_{Random_{\mathcal{L}}}^{V2G} = 4$) for EVs with random plug-in.
- If $E_{Dem}^{t} > E_{Sup}^{t}$ (i.e., the value of Equation (3.12) is negative): EVs which request to discharge their battery receive the first and the second priority levels ($\mathcal{L} = 1, \mathcal{L} = 2$) as follows: ($ev_{Calendar_{\mathcal{L}}}^{V2G} = 1$) for EVs with calendar plug-in, and ($ev_{Random_{\mathcal{L}}}^{V2G} = 2$) for EVs with random plug-in. However, EVs which request to charge their battery receive the low priority levels ($\mathcal{L} = 3, \mathcal{L} = 4$) as follows: ($ev_{Calendar_{\mathcal{L}}}^{G2V} = 3$) for EVs with calendar plug-in, and ($ev_{Random_{\mathcal{L}}}^{G2V} = 4$) for EVs with random plug-in.

We introduce two priority attribution algorithms: First, for EVs with calendars plug-in (i.e., EVs that have already created their charging/discharging profiles inside cloud data centers), we propose the EV Calendar Priority Attribution algorithm (EV_CPA) which is executed inside cloud platforms. All information about EVs users, EPSS and demand-supply curve are centralized in the Fog data centers. The priority levels are attributed considering the two cases of demand-supply curve (i.e., $E_{dem}^{t} \leq E_{sup}^{t}$ and $E_{dem}^{t} > E_{sup}^{t}$), and EVs calendars types (i.e., $EV_{Calendar}^{G2V}$ → charging and $EV_{Calendar}^{V2G}$ → discharging).

Algorithm 1: EV Calendar Priority Attribution (EV_CPA)

Input: $ev_{Profile}, ev_{Calendar}^{G2V}, ev_{Calendar}^{V2G}, E_{Dem}^{t}, E_{Sup}^{t}$;
Output: calendars plug-in with priority levels attribution;

```
for (each ev_Profile) do
    Read (ev_Calendar^G2V) at time slot t;
    Read (ev_Calendar^G2V) at time slot t;
    if (E_Dem^t ≤ E_Sup^t) then
        for (each ev_Calendar^G2V) do
            ev_Calendar^G2V_ℒ = 1;
            Save and confirm the (ev_Calendar^G2V);
        end
        for (each ev_Calendar^V2G) do
            ev_Calendar^V2G_ℒ = 3;
            Save and confirm the (ev_Calendar^V2G);
        end
    else
        for (each ev_Calendar^G2V) do
            ev_Calendar^G2V_ℒ = 3;
            Save and confirm the (ev_Calendar^G2V);
        end
        for (each ev_Calendar^V2G) do
            ev_Calendar^V2G_ℒ = 1;
            Save and confirm the (ev_Calendar^V2G);
        end
    end
    Update, E_Dem^t, E_Sup^t ;
end
```

Second, for EVs with random plug-in (i.e., EVs that have not yet created their charging/discharging profiles inside cloud data centers); we propose the EV Random Priority Attribution (EV_RPA) algorithm which is executed inside the station. EVs with random plug-in receive their priority levels when they arrive at EVPSS. Where, cloud services can communicate with EPSSs and attribute for each EV with random plug-in for charging $\boldsymbol{ev_{Random}^{G2V}}$ and for discharging $\boldsymbol{ev_{Random}^{V2G}}$ the appropriate priority according to the scheduling state. This algorithm is executed for each EPSS when an EV with random plug-in is arriving.

Algorithm 2: EV Random Priority Attribution (EV_RPA)

Input: $ev_{Profile}, ev_{Random}^{G2V}, ev_{Random}^{V2G}, E_{Dem}^{t}, E_{Sup}^{t}$;
Output: random plug-in with priority levels attribution;

```
for (each EPSS) do
    Read (ev^{G2V}_{Random}) at time slot t;
    Read (ev^{V2G}_{Random}) at time slot t;
    if (E^t_{dem} ≤ E^t_{sup}) then
        for (each ev^{G2V}_{Random}) do
            ev^{G2V}_{Random} = 2;
        end
        for (each ev^{V2G}_{Random}) do
            ev^{V2G}_{Random} = 4;
        end
    else
        for (each ev^{G2V}_{Random}) do
            ev^{G2V}_{Random} = 4;
        end
        for (each ev^{V2G}_{Random}) do
            ev^{V2G}_{Random} = 3;
        end
    end
    Update the ESS state;
    Send to EVPSS state information to the Cloud services;
end
```

3.5.2 EPSS Selection Strategy Algorithm

Finally, we propose a strategy to select the best EPSS for each EV with calendar plug-in using the output information from EV_CPA algorithm and some constraints related to EPSS state, the $ev_{location}$, the time and the distance to joint each EPSS. To this end, we propose a scheduling algorithm (Algorithm 3) for selecting the best station called (BSS); the algorithm is

implemented and executed inside the Fog data centers. However, for each EV calendar, the algorithm selects the optimal EVPSS that has the minimum waiting time to plug-in for each priority queue taking into consideration the mobility scenario for each EV. We note that BSS algorithm is dedicated only for EVs with calendar plug-in process. For EVs with random plug-in, we assumed earlier that they arrive randomly to EVPSS, consequently they cannot benefit of this service until they create their charging/discharging profile and calendars.

Algorithm 3: Best Station Selection (BSS)

Input: $ev^{G2V}_{Calendar}, ev^{V2G}_{Calendar}$ with priority levels, EPSS state (location, waiting time $Vect\_E[w_{\mathcal{L}}]$, $Vect\_C_{\mathcal{L}}$, Plug-in sockets number), $ev_{location}$;
Output: The optimal EPSS for each EVs.

```
1. Read ev_location;
2. Select the nearest EPSS to ev_location;
3. for (each selected EPSS) do
4.     if (ev^{G2V/V2G}_{Calendar_L} = 1) then
5.         Calculate Vect_E[w_1];
6.         Compare Vect_E[w_1] for each EPSS;
7.         Select EPSS according to min(E[w_1]);
8.         Sends location of the optimal EPSS to EV;
9.     end
10.    if (ev^{G2V/V2G}_{Calendar_L} = 3) then
11.        Calculate Vect_E[w_3];
12.        Compare Vect_E[w_3] for each EPSS;
13.        Select EPSS according to min(E[w_3]);
14.        Sends location of the optimal EPSS to EV;
15.    end
16.    Update (Vect_C_1, Vect_C_3);
17.    Update EPSS state;
   end
```

3.6 Performance Evaluation

We perform extensive simulations to evaluate and highlight the proposed scheduling approach for EV charging and discharging services. We use MATLAB to perform our simulations and SUMO-Veins [92] to implement our mobility model.

3.6.1 Simulation settings

We consider the electric load in a city, provided by a micro grid and controlled using smart grid utilities. We examine EV charging and discharging during a day (24 h) starting from 12:00 AM. The day is evenly divided into 288 intervals [38]. Each interval has a length of 5 minutes (min).

Table 3.1- Simulation Parameters

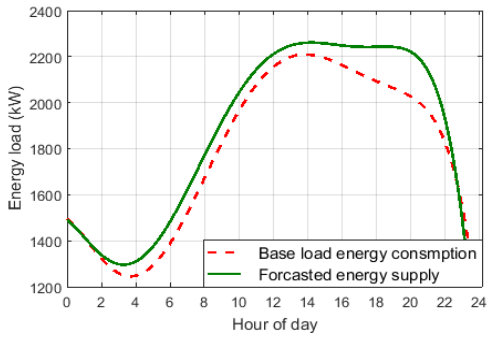
Parameters	Value
Simulation area	10 km x 10 km
Number of charging EVs	3000
Number of discharging EVs	2000
A_i^t (Charging energy)	[10 kW – 45 kW]
B_i^t (Discharging energy)	[10 kW - 40kW]
EVPSS number	20
Number of plug-in socket	10
Battery capacity	50 kW [93]
Charging and discharging rate	20 kW/h [93]

According to our proposed model we have two EVs categories: a) EVs with calendar plug-in, b) EVs with random plug-in. The arrival times of the EVs are uniformly distributed across the day, and the charging load is estimated to be high during evening peak hours (5:00 am to 22:00 am). The charging and discharging periods of EVs are uniformly distributed according to the requested energy for charging A_i^t and for discharging B_i^t. For mobility modeling, we consider a scenario of 10 roads that allow access to a city having area of 10 km x 10 km where the EPSSs are placed randomly. All vehicles are traveling with speeds that cannot exceed 60 km/h. Table 3.1 shows the various parameters used for simulation.

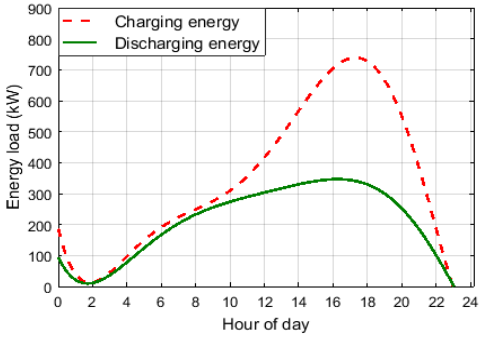
3.6.2 Performance Metrics

In order to evaluate our proposed schemes, we use different metric as follows:

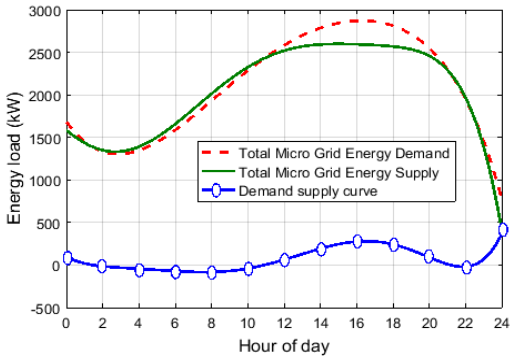
- *Demand supply curve:* is the principal metric used to inspect the performance of the proposed approach, especially during peak hours. The demand supply curve is evaluated from Equation 3.12.
- *Priority levels attribution:* we use four priority levels and compare the number of EVs categories (charging or discharging) for each priority level across the day in different case studies.



(a) Real-time supply and base load demand



(b) Real-time Charging and discharging energy



(c) Total energy load vs. supply demand curve

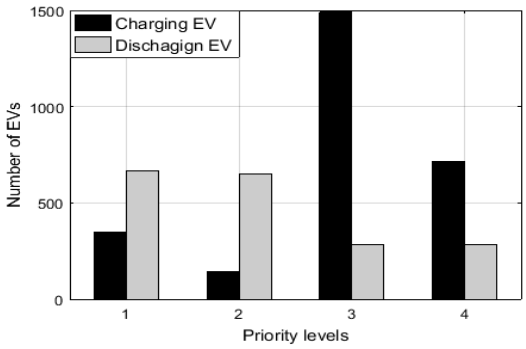
Figure 3.11- Real-time impact of EVs integration in the smart grid supply demand curve

- *Time to plug*-in: we define the time to plug-in for any EV as the sum of the waiting time to join EPSS and the cloud provider response time.

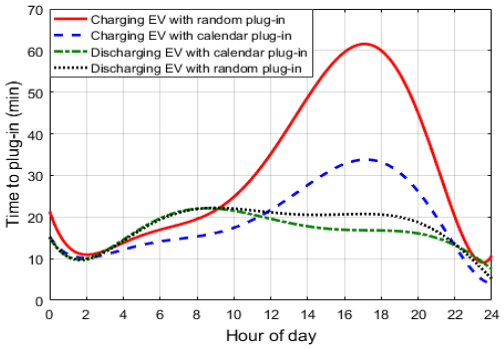
3.6.3 Results and Discussion

Three scenarios of simulations were performed as follows: 1) we perform simulations considering four priority levels for EVs where we suppose that the demand supply curve is not stable, and 2) we run simulation without considering cloud computing platforms and priority strategy, we called this scenario the standard case.

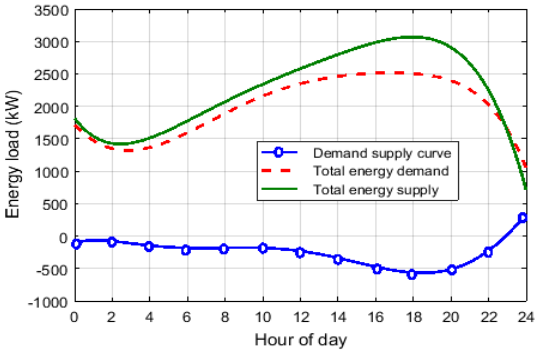
In order to keep our simulation scenarios more realistic, we use a real base load consumption, Figure 3.11(a) shows the base load consumption and the energy supply simulated by scaling the real base load and the real supply energy in Toronto by a factor of 1/1500. Figure 3.11(b) shows the real time charging and discharging energy during a day selected randomly where the charging energy is estimated to be high compared with the discharging energy. Figure 3.11(c) shows the total energy demand, the total energy supply and the demand supply curve after adding the charging and the discharging load. We can see that the demand supply curve is not



(a) EV_CPA and EV_RPA priority scheduling



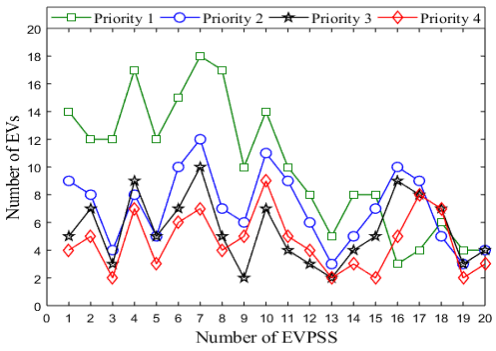
(b) Time to plug-in using EV_CPA and EV_RPA



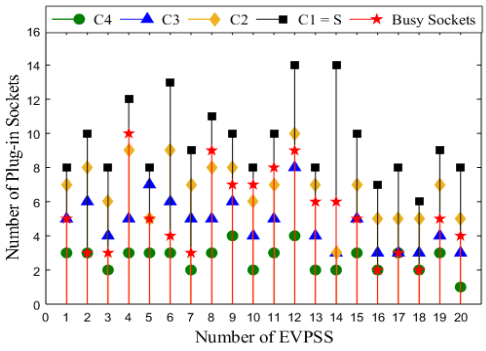
(c) Demand supply curve with scheduling

Figure 3.12- Priority attribution strategy using EV_CPA and EV_RPA scheduling and demand supply curve stabilization

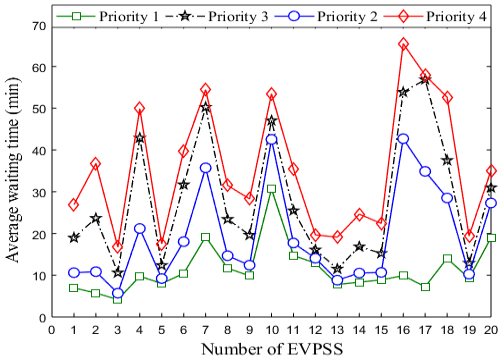
stable from [12:00 pm to 09:00 pm] because of all EVs users look to charge its batteries during this period. Now, we consider the case of non-stability of demand supply curve ($E_{dem}^{t} \leq E_{sup}^{t}$). In order to stabilize the curve during a day, we run simulations with EV_CPA and EV_RPA scheduling. Figure 3.12(a) shows the number of charging EVs and discharging EVs with different priority levels during a day. We can see that the number of discharging EVs is higher with priority levels (1 and 2) then the lowest priority levels (3 and 4). This is due mainly to the performance of the proposed priority attribution strategy implemented by EV_CPA for EVs with calendar plug-in, and EV_RPA for EVs with random plug-in. In this regard, the algorithms aim to maximize the waiting time to plug-in for EVs need to charge and minimize the waiting time to plug-in for EVs need to discharge. As represented in Figure 3.12(b) we can observe that from 03:00 am to 09:00 am, the waiting time for discharging EVs is higher than the waiting time for charging EVs, this is because the demand supply curve is stable ($E_{dem}^{t} > E_{sup}^{t}$) during this period. However, from 12:00 pm when the demand supply curve becomes not stable ($E_{dem}^{t} \leq E_{sup}^{t}$), the curves are inversed, and the waiting time to plug-in for charging



(a) Number of EVs arrivals



(b) cut-off values in each EVPSS

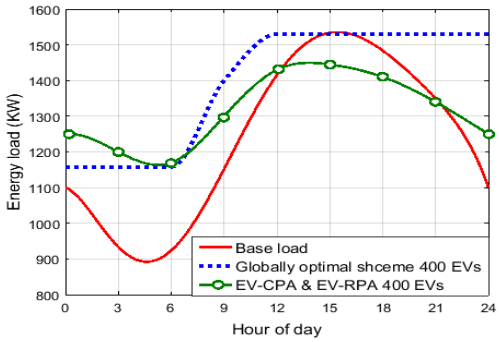


(c) Waiting time comparison

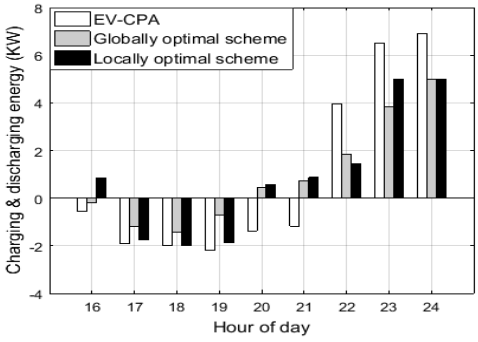
Figure 3.13- Comparison of the impact of EV arrival rate, cut-off values and number of busy sockets on waiting time for each priority

EVs become very high, especially for EVs with random plug-in. This is why EVs users are solicited to create charging and discharging calendars in order to relieve the smart grid system and select a best charging/discharging planning during a day, and especially during peak hours. As results, we can see in Figure 3.12(c) a significant stabilization of the demand supply curve compared with the standard case; we have an important energy saving (500 kwh) during evening peak hours.

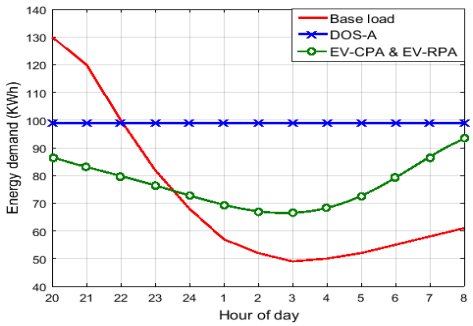
Now, we examine the efficiency of the queueing model with four priority levels and cut-off process under implementation of the proposed planning algorithms (i.e., PAC, PAR, BSS). We show in Figure 3.13 the impact of EV arrival rate, cut-off values and number of busy sockets on the waiting time for each priority level, results are performed during evening peak hours [06:00 pm to 10:00 pm]. Figure 3.13(a) shows the number of EVs arrivals into each EPSS with different priority levels, we can see that the average number of EVs with high priority level is higher than other priority levels for the most of EPSS, we can explain this by the comportment of the two algorithms PAC and PAR in order to adjust the energy curve during peak hours by maximizing the number of EVs looking for discharging energy.



(a) Total energy load comparison



(b) charging and discharging energy comparison



(c) Load shifting comparison using 10 EVs

Figure 3.14- Comparison of EV-CPA & EV-RPA performance with existing works. Total energy demand, charging/discharging and load shifting

Figure 3.13(b) and Figure 3.13(c) show respectively the update of cut-off thresholds and the waiting time in each EPSS according to the arriving rate represented in Figure 3.13(a). We see clearly the impact of the cut-off values for each priority on the waiting time for the according queue, for example EVPSS number 4, 7 and 10, $C_4 < \beta$ this is why the waiting time in the queue with priority levels $\mathcal{L} = 4$ is very high. Consequently, the arrival rate of EVs, the cut-off values and the number of busy sockets in each EVPSS have an important impact on the waiting time, the time the plug-in, and so the energy loads (i.e., demand and supply) during the charging and the discharging time period.

3.6.4 Comparison with Existing Works

Figure 3.14 illustrates the performance of the proposed scheduling approach compared with existing works, such as optimal scheduling for charging and discharging of EVs [38], and decentralized changing and discharging scheduling of plug-in EVs in smart grid (i.e., DOS_A) [44]. To do this, first we consider the same simulation parameters used in each work. In [38], the authors used 400 EVs and the battery capacity is 9 KWh, we adjust our simulation scenario with the same parameters and we run simulations. Figure 3.14(a) shows the total energy demand

using our scheduling algorithms (i.e., EV_CPA and EV_RPA) compared with the globally optimal scheme proposed in [38]. We can see that from 01:00 am to 06:00 am the total energy demand using our scheduling is higher than the total energy using the globally optimal scheme, this is because our algorithms solicit EVs users to charge their batteries during this period (i.e., no peak hours where the demand supply curve is stable) by attributing the high priority levels; in that way we reduce the waiting time in EPSS. However, during peak hours, from 12:00 pm to 21:00 pm, we can see that our algorithms are more efficient than the globally optimal scheme. Moreover, they reduce significantly the energy base load (- 100 KW), because a high number of EVs with calendars discharge during this peak period with the lowest waiting time allows a quick deliverance of energy to the grid. Next, we examine the scheduling of charging and discharging power for randomly chosen EVs during evening peak hours, for our case we select EVs with calendar plug-in. As shown in Figure 3.14(b), our algorithms discharge the energy in peak hour intervals 16:00 pm – 21 pm with a different energy load (varied according to the demand supply curve). However, the locally optimal and the globally optimal schemes discharge only during intervals 17:00 pm – 19:00 pm with a constant energy load. For the charging load, we can see that our algorithms charge energy during non-peak hour intervals 22:00 pm – 24:00 pm with a high amount of energy compared with the other schemes.

Finally, we compare our results with the work proposed in [44]. The authors in [44] use 10 heterogonous EVs and the base load curve of 100 households in the service area of Southern California Edison from 20:00 to 8:00 [44] as shown in Figure 3.14(c). We use the same parameters, and we run simulation during the same period. We observe that our proposed scheduling is more capable to regulate the total demand curve compared with DOS_A especially during peak hours 20:00 pm – 22:00 pm.

3.7 Chapter Summary

Smart charging can assist EVs to act as flexible loads and distributed storage resources that can benefit the energy system. This chapter proposes an efficient smart grid approach for EVs charging and discharging services at electric public supply stations (EPSS), based on cloud computing scheduling. The proposed approach ensures the communication between smart grid and cloud platforms, and manages the smart grid entities and operations. In order to optimize the waiting time to plug-in at EPSS, we introduced priority assignment algorithms. The simulation results proved that the proposed approach can be useful in a variety of cases, especially in peak hours when the number of EVs charging requests is high, and can also improve the grid stability.

CHAPTER 4

DYNAMIC PRICING MODEL FOR ENERGY SAVING IN SMART GRID

In this chapter, we propose a real-time dynamic pricing model for EVs charging and discharging service and building energy management, in order to reduce the peak loads. Our proposed approach uses a decentralized cloud computing architecture based on software define networking (SDN) technology and network function virtualization (NFV). We aim to schedule user's requests in a real-time way and to supervise communications between microgrids controllers, SG and user entities (i.e., EVs, electric vehicles public supply stations, advance metering infrastructure, smart meters, etc.). We formulate the problem as a linear optimization problem for EV and a global optimization problem for all microgrids. We solve the problems by using different decentralized decision algorithms. The extensive simulations and comparisons with related works proved that our proposed pricing model optimizes the energy load during peak hours, maximizes EVs utility, and maintains the microgrid stability. The simulation is based on real electric load of the city of Toronto.

The rest of the chapter is organized as follows. Section 4.1 presents the motivation behind the dynamic pricing model. Section 4.2 presents the proposed architecture and motivation for Cloud-SDN enabled SG while Section 4.3 presents the energy management model. In Section 4.4 we formulate the real-time pricing optimization model. Section 4.4 presents the optimal decision algorithms for energy management while Section 4.5 portrays the simulations results. Finally, Section 4.6 concludes the chapter.

4.1 Motivation

To improve the grid stability and make profit from the price variation, it is well known that the EVs charging process [grid-to-vehicle (G2V)] should be on off-peak demand period while the discharging process (V2G) would be on-peak load period. Nevertheless, it is expected that EVs will be deployed in large-scale environments; consequently, EVs need a real-time scheduling and monitoring. In this regard, a new distributed architecture based on flexible networks needs to be implemented to maintain the adequate balance between demand and supply in SG.

On the one hand, SG environment is suitable to accommodate several utilities, which can require huge data storage and processing [94]. For instance, it has been estimated that SG will generate 22 GB of data each day from its 2 million customers [49]. Also, the existing power

grid in the U.S. is going through a massive transformation to make it more reliable and connected with the ability to transfer data and power in two ways [95]. Moreover, the microgrids will support an exponential growth in terms of energy demands from many devices [e.g., advance metering infrastructure (AMI), smart meters, EVs, electric vehicles public supply stations (EVPSS), etc.], located at home area networks (HANs), EV area networks (EVANs), and wide area networks. Each of these networks deploys thousands of devices that need to be managed and controlled incessantly [96]. Thus, the scheduling of all microgrid entities and minimizing the network management cost becomes very problematic. On the other hand, the emerging software define networking (SDN) technology can provide good opportunities for reducing the network management cost by integrating a flexible software network controller and virtualization functions (NFV). Some of existing works in the SDN area focus on data centers virtualization and cloud computing [97]. This later, with its distributed aspect and memory capacity, can offer huge computing, storage, and networks capabilities. So, there is a need to adapt cloud-based SDN technology for various SG applications.

In this work, we introduce a new SG architecture based on a decentralized cloud computing architecture for each microgrid, and a centralized SG controller located at cloud data centers. This architecture exploits SDN paradigm in order to offer to SG new options and opportunities for dealing with networks faults and communications problems even in the case of overall outages. Also, we believe that the cloud computing with its enormous data storage, computing capacity, and its distribution aspect, can help the SG to manage efficiently users' requests and solve the proposed optimization problems in a real-time manner.

The contributions of this chapter can be summarized as follows.

1) A new dynamic pricing model, where the price is varied according to the variations of the demand–supply curve. The objective is to optimize the energy costs in the SG using V2G energy flow and renewable energy sources.
2) Renewable energy management algorithm to manage energy inside the buildings by switching the energy usage between microgrid and local renewable sources.
3) EVs charging and discharging scheduling algorithms based on two principal metrics: a) dynamic pricing and b) demand–supply curve. The objective is to ensure a well energy balancing especially on peaks hours and solve the optimization problem for EVs.
4) Centralized microgrid management algorithm. The objective is to manage and control all microgrids in different situations and solve the global optimization problem.
5) Exploration of cloud and fog computing capacities based on SDN technology. The objective is to offer EV users the best calendar (i.e., optimal reservation time) and

manage all microgrid communications networks considering real-time and energy constraints.

4.2 Distributed SG Architecture based Cloud SDN

In this section, we first describe the communication infrastructure for SG and then detail our proposed network model based on a decentralized architecture using Fog-SDN for microgrid controllers and Cloud-SDN for SG controller.

4.2.1 SG Networking Architectures

Basically, the traditional electric grid infrastructure can be divided into three major components in SG communication architecture [95]:

1) Customers layer: contains different area network as HANs, building area network (BANs), industrial area network (IAN), and EVAN. Each local network connects home, building, and industrial devices to smart meters.
2) Microgrid layer: collects the smart meters and EVPSS data from the customer layer.
3) SG layer: provides long-haul communication with the utility control centers using various communication technologies and standards.

4.2.2 SG Motivation for Cloud-SDN Enabled SG

A communication network is the important component of SG infrastructure. With the integration of advanced technologies and applications for achieving a smarter metering of energy consumption, a huge amount of data from different building appliances and EV infrastructures will be generated for analysis, update, control, and real-time pricing methods. Thus, the management of these networks is a big challenge due to the scale. Moreover, due to heterogeneity of devices and applications from different vendors, the equipment may not be interoperable. Hence, it is very critical for electric companies to define the communications standards and find the best communication infrastructure to control and manage all customers throughout the total system considering the real-time constraint. Cloud computing based SDN is an encouraging solution for the aforementioned problems, thanks to the following advantages.

1) Cloud technology provides high computing capacity to SG utilities. In addition, with its enormous data storage, cloud can help SG to improve demand side management services and customers' participation for efficient electricity usage.

2) The decentralized aspect of cloud networking using the so-called Fog-computing can be used to manage and control each microgrid in a distributed manner.
3) The SDN technology adopts standards and introduces abstraction, which provides a centralized control and management of various types of network devices and multiple vendors that are common in SG.
4) The hardware NFV through SDN solves the problem of managing different networks and reduces the energy consumption of the network equipment.
5) Programmability by operators, enterprises, independent software vendors, and users using common programming environments.

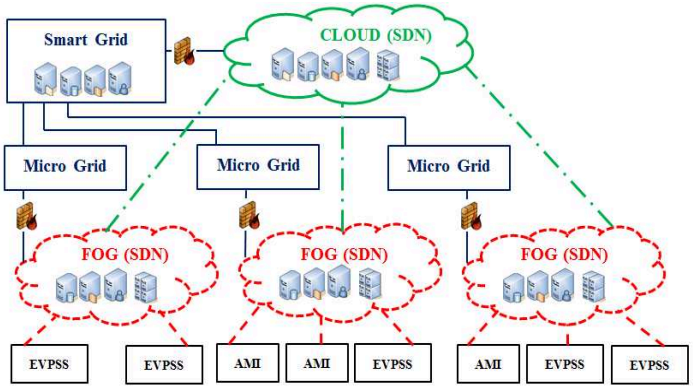
4.2.3 Proposed Network Model

As seen earlier, the traditional SG architecture contains three major components. In this regard, we need to describe the communication standards and network topologies required for information flow in a SG system for each layer.

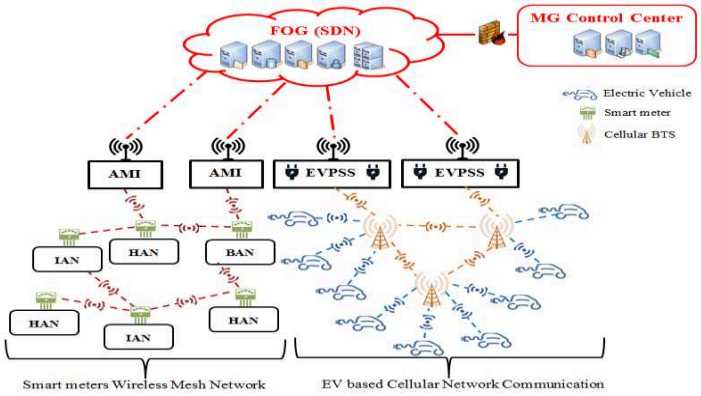
A. Customers' Area Networks (i.e., HAN, BAN, IAN): The basic metering data flow is from sensors and electrical devices to smart meters. To do this, we use ZigBee wireless communication technology that is relatively low in power usage, data rate, complexity, and cost of deployment; it is an ideal technology for smart energy monitoring and automatic meter reading [96]. Also, ZigBee integrated SG meters can communicate with the ZigBee integrated devices and control them.

B. Smart Meters Network: In order to collect the metering data from the smart meters, we use radio frequency mesh network, where we suppose that every smart meter is equipped with a radio module and each of them routes the metering data through nearby meters as presented in Fig. 1(b). Each device acts as a signal repeater until the collected data reaches the AMI (electric network access point). The mesh network technology is used by many companies for SG metering applications due to the redundancy and high availability features of this technology.

C. EVs Mobile Network: Current cellular networks can be a good option for connecting EVs and public supply stations. The existing communications infrastructure avoids electrical companies from spending operational costs and additional time for building a dedicated



(a) Cloud-SDN architecture for Micro Grids communication



(b) Decentralized Fog-SDN architecture for one micro grid

Figure 4.1- Proposed Cloud-SDN and Decentralized Fog-SDN architectures for smart grid communications

communication network. Also, the cellular network technologies (e.g., 3G, 4G) enable the mobility of EVs and the deployment of EVPSS. We suppose that an embedded mobile SIM within a cellular radio module is integrated at each EV and EVPSS to enable the communication. Many telecom operators and vendors have also agreed to put their cellular network into service of electrical companies, this can be a big challenge in terms of interoperability and privacy; we believe that the use of the SDN technology can manage and control different networks.

D. Microgrid-Based Decentralized Fog-SDN Architecture: In our proposed network architecture, we suppose that we have a number M of micro grids m, where:

$$\mathbb{M} = \{1,2,\dots,m,\dots,M\}, \forall\, m \in \mathbb{M} \tag{4.1}$$

Each micro grid contains a number S of smart meters s, a number V of EV v , and a number E of EVPSS e, where:

$$\mathbb{S} = \{1,2,\dots,s,\dots,S\}, \forall\, s \in \mathbb{S} \tag{4.2}$$

$$\mathbb{V} = \{1,2,\dots,ev,\dots,V\}, \forall\, ev \in \mathbb{V} \quad (4.3)$$

$$\mathbb{E} = \{1,2,\dots,e,\dots,E\}, \forall\, e \in \mathbb{E} \quad (4.4)$$

For each microgrid, we associate a decentralized data center (i.e., fog). A data center comprises different servers, such as database server, management server, real-time communication server, application server, and NFV and VM servers; each server provides various services for customers. Also, the data center can communicate with the microgrid control center in order to update and deliver real-time information to the company utilities (about prices for example). Hence, all communications and networks are managed and controlled using the SDN technology. Finally, the management of the fog datacenters is provided by cloud services in a centralized manner (see Fig. 1). A centralized SDN controller is implemented inside the cloud in order to manage the backhaul network.

4.3 Energy Management Model

4.3.1 Energy Demand Profiling

We consider two types of demands:

A. Building demands: represented as ED_B^t including customer's devices demands, $ed_{i_h}^t$, $ed_{i_l}^t$, $ed_{i_k}^t$, arriving at time interval t at each HAN, BAN, and IAN respectively., Thus, we have:

$$ED_B^t = \sum_{h=1}^{H}\sum_{i=1}^{dev} ed_{i_h}^t + \sum_{l=1}^{L}\sum_{i=1}^{dev} ed_{i_l}^t + \sum_{k=1}^{K}\sum_{i=1}^{dev} ed_{i_k}^t \quad (4.5)$$

Where, H, L, K are the numbers of HANs, BANs and IANs at each micro grid m, and dev is the number of devices. In our proposed model, each device has a minimum and maximum energy demand as follows:

$$ed_{i_{h,l,k}}^{min} \leq ed_{i_{h,l,k}} \leq ed_{i_{h,l,k}}^{max},\ \forall\, i_{h,l,k} \in HAN, BAN, IAN \quad (4.6)$$

B. Electric Vehicles demand: represented as ED_{EV}^t, the energy demand from mobile EVs, at public supply stations inside the micro grid m, depends on their available energy and battery capacity. Therefore, we consider the charging energy $ED_{ev_i^{G2V}}^t$ of one electric vehicle ev_i^{G2V} as follows:

$$ED_{ev_i^{G2V}}^t = \alpha_{ev_i^{G2V}}^t - soc_{ev_i^{G2V}}^t \quad (4.7)$$

$$\text{and,} \qquad \alpha_{ev_i}^t \leq cap_{ev_i} \quad (4.8)$$

Where, $\alpha_{ev_i}^t$ is the predetermined target charging level of an $ev_i^{G2V}, \forall\ ev_i^{G2V} \in \mathbb{V}$, and $soc_{ev_i^{G2V}}^t$ is the available energy state of charge of ev_i at time interval t. However, the condition in Equation (4.8), must be respected to perform charging of an EV, where $cap_{ev_i}$is the battery capacity of an ev_i.

Thus, the total energy demands $ED_{G2V_m}^t$ of a number V' of ev_i^{G2V} at the micro grid m is given by:

$$ED_{G2V_m}^t = \sum_{i=1}^{V'} ED_{ev_i^{G2V}}^t, \qquad \forall\ ev_i^{G2V} \in \mathbb{V} \tag{4.9}$$

Finally, the total energy demand of a micro grid m including all customers' devices and EVs is represented as follows:

$$ED_m^t = ED_B^t + ED_{G2V_m}^t \tag{4.10}$$

And for the total smart grid demand ED^t:

$$ED^t = \sum_{j=1}^{m} ED_j^t \tag{4.11}$$

Remark 1: If an EV perform the charging operation at home or at building areas, its energy demand for charging will be metered by the home smart meter and the EV is considered as a device, so the energy consumption will be calculated as in Equation (4.5).

4.3.2 Energy Supply Profiling

We consider that each micro grid m has renewable and non-renewable energy sources with self-generation capacity, and provides electricity in a certain region. The energy supply sources are presented as follows:

A. Building renewable energy supply: we suppose that we have a number R of building areas (HAN, BAN and IAN) equipped with renewable energy sources as solar energy equipment, so we define a new set of building energy consumption called buildings with renewable energy denoted by $\mathbb{r}$, thus we have:

$$\mathbb{r} = \{1,2,\dots,r,\dots,R\}, \forall\ r \in \mathbb{r} \tag{4.12}$$

Where:

$$R < H + L + K \tag{4.13}$$

The renewable energy supply for each building $h, l\ or\ k$ is denoted by RE_r^t; this energy is used to fulfill the self-energy consumption in the same building. We suppose that the generated

energy is stored used a domestic battery as Tesla wall battery [98]; this battery can be charged by the energy discharged from EVs that perform the discharging process at home denoted by $ES^t_{ev_r}$, so we have:

$$soc^t_{wall_r} = soc^{t-1}_{wall_r} + RE^t_r + ES^t_{ev_r}, \forall\, r \in \mathbb{r} \tag{4.14}$$

$$\text{And,} \qquad 0 \leq soc^{t-1}_{wall_r} \leq soc^t_{wall_r} \leq cap_{wall_r} \tag{4.15}$$

Where, $soc^t_{wall_r}$and cap_{wall_r} are respectively the charged level and the capacity of the wall battery at time interval t. However, the wall battery cannot be necessary sufficient to provide the total energy needed to fulfill all power demands from building devices. In this regard, we have two cases as follows:

a) If $(soc^t_{wall_r} > \sum_{i=1}^{dev} ed^t_{r_i})$, then devices of building r use the stored energy in the wall battery. The charged level of the battery will be reduced at time interval $t+1$, also the requested energy from the micro grid denoted by $ED^t_{r_m}$ is null, so we have:

$$soc^{t+1}_{wall_r} = soc^t_{wall_r} - \sum_{i=1}^{dev} ed^t_{r_i} \tag{4.16}$$

$$\text{and,} \quad soc^{t+1}_{wall_r} \geq soc^t_{wall_r} \geq \sum_{i=1}^{dev} ed^t_{r_i} > cap^{th}_{wall_r} \tag{4.17}$$

$$ED^t_{r_m} = 0 \tag{4.18}$$

b) If $(soc^t_{wall_r} < \sum_{i=1}^{dev} ed^t_{r_i})$, in this case user devices of building r can use the wall battery if only it's charged level $soc^t_{wall_r}$ is superior to the predetermined minimum threshold $cap^{th}_{wall_r}$.

The condition in Equation (4.17) must be respected.

Therefore, the requested energy from the micro grid m denoted for one building by $ED^{t+1}_{r_m}$ is not null and becomes:

$$ED^{t+1}_{r_m} = \sum_{i=1}^{dev} ed^{t+1}_{r_i} - soc^{t+1}_{wall_r} \tag{4.19}$$

Remark 2: Taking into consideration buildings characterized by a self-renewable energy generation, the total demands of all buildings inside the micro grid m represented in Eq. (5) becomes:

$$ED^t_B = \sum_{h=1}^{H-R'} \sum_{i=1}^{dev} ed^t_{i_h} + \sum_{l=1}^{L-R''} \sum_{i=1}^{dev} ed^t_{i_l} + \sum_{k=1}^{K-R'''} \sum_{i=1}^{dev} ed^t_{i_k} + \sum_{r=1}^{R} \sum_{i=1}^{dev} ed^t_{i_r} \tag{4.20}$$

$$R = R' + R'' + R''' \tag{4.21}$$

Where $R < (H + L + K)$. R', R'' and R''' are respectively the numbers of homes, buildings and industrials area networks that have a self-renewable energy generation.

Assumption 1: Buildings with self-renewable energy generation are equipped with smart meters that can switch between the wall battery and micro grid energy usage according to conditions presented in Equation (4.15) and Equation (4.17).

B. EVs discharging energy supply: In our proposed model, we use new industrial innovations used by cars and battery companies such as the energy flow from EVs batteries to grid or so-called V2G, where EVs located at micro grid m provide energy by discharging its batteries. An EV can perform the charging and discharging process at r buildings in order to charge the domestic battery as seen in Equation (4.14) or at public supply stations EVPSS in order to provide energy to micro grid storage system. Therefore, we consider the discharging energy $ES^t_{ev_i^{V2G}}$ of one electric vehicle ev_i^{V2G} as follows:

$$ES^t_{ev_i^{V2G}} = soc^t_{ev_i^{V2G}} - \beta^t_{ev_i^{V2G}} \tag{4.22}$$

Where, $\beta^t_{ev_i^{V2G}}$ is the predetermined target discharging level of an $ev_i^{V2G}, \forall\, ev_i^{V2G} \in \mathbb{V}$, and $soc^t_{ev_i^{V2G}}$ is the available energy state of charge of ev_i^{V2G} at time interval t. The total discharging energy from V'' of ev_i^{V2G} at public supply stations at time interval t is given by:

$$ES^t_{V2G_m} = \sum_{i=1}^{V''} ES^t_{ev_i^{V2G}}, \forall\, ev_i^{V2G} \in \mathbb{V}, \tag{4.23}$$

$$\text{and} \quad V = V' + V'',\ V' > V'' \tag{4.24}$$

C. Micro grid energy supply: We consider that each micro grid m has renewable $ES^t_{RW_m}$ and non-renewable energy $ES^t_{N\_RW_m}$ sources with self-generation capacity, and provides electricity in its covered region.

Assumption 2: All micro grids can communicate with the other micro grids, and can exchange energy with one another. If a micro grid has excess energy, it can provide another micro grid that has energy deficiency.

The total energy supply of one micro grid is given by:

$$ES^t_m = ES^t_{N\_RW_m} + ES^t_{RW_m} + ES^t_{V2G_m} + ES^t_{EX_{m-1}} \tag{4.25}$$

Where $ES_{V2G_m}^t$ is the energy provided from discharging EVs batteries, and $ES_{EX_{m-1}}^t$ is the energy received from another micro grid.

4.4 Real-time Pricing Optimization

4.4.1 Pricing Model based Demand-Supply Curve Balancing

In the proposed distributed dynamic pricing model, we aim to ensure the energy management by balancing the demand supply curve for each micro grid m. Therefore, real-time price is subject to different constraints such as total supply, demand, and time.

Assumption 3: We assume that the real-time price of a micro grid is not affected by the pricing policy of the other micro grids. Each micro grid has its unique price according to the local demand supply curve.

In order to characterize the demand-supply curve, we calculate the difference and the ratio between the total energy demand ED_m^t calculated in Equation (4.10) and the total energy supply ES_m^t calculated in Equation (4.25) for a particular micro grid m, as follows:

$$D_m^t = ED_m^t - ES_m^t \tag{4.26}$$

$$R_m^t = \frac{ED_m^t}{ES_m^t} \tag{4.27}$$

Each micro grid evaluates its real time price p_m^t depending on the real time difference and ratio presented in Equation (4.26) and Equation (4.24).

Theorem: The dynamic real time price for consuming one unit of electricity is represented as follows:

$$p_m^t = \{\tan^{-1}\left(e^{D_m^t}\right) + (\tan^{-1} R_m^t)^{10}\} + p_{base} \tag{4.28}$$

Where, p_{base} is a fixed base price. In Eq. (28), the part between brackets is the dynamic portion that changes the price according to the variation in R_m^t and D_m^t.

Proof: We know that $\lim_{x\to+\infty} e^x = +\infty$ and $\lim_{x\to-\infty} e^x = 0$, so the portion $e^{D_m^t}$ is always bounded between $[0, +\infty[$, $\forall D_m^t$. On the other hand, we have: $\lim_{x\to0} \tan^{-1} x = 0$ and $\lim_{x\to+\infty} \tan^{-1} x = \frac{\pi}{2}$, so the portion $(\tan^{-1} e^{D_m^t})$ is always bounded between $\left[0, \frac{\pi}{2}\right]$, but the variance of this portion is not very important. Thus, in order to keep the balance between demand and supply effects more the real time price, we add the second portion,$(\tan^{-1} R_m^t)^{10}$, varied according to the

variations in the ration $R_m^t \lim_{x \to +\infty} (\tan^{-1} x)^{10} = (\frac{\pi}{2})^{10}$. Consequently, the proposed real time price p_m^t is varied between $\left[p_{base}, \frac{\pi}{2} + (\frac{\pi}{2})^{10} + p_{base}\right]$.

Corollary: With an increase in the supply ES_m^t, the real time price p_m^t decreases while demand ED_m^t from the customers is either fixed or decreased. On the other hand, the real-time price increases with an increase in the demand, while supply is either fixed or decreased. In this regard, we aim to vary the price value in a bounded interval in order to respect the conventional prices given by electrical companies, this is why we use exponential and arctangent functions.

4.4.2 Pricing Optimization based Fog Management

A. For micro grid services: The objective of micro grid is to maximize its own utility, depending on real time demand and supply. In this regard, we formulate the real time price optimization, decided by each micro grid controller, as a maximization problem as follows:

$$\begin{aligned} &Maximize \sum_{t \in T} ED_m^t \, p_m^t \\ &\text{Subject to } \textstyle\sum_{t \in T} ED_m^t \leq \sum_{t \in T} ES_m^t \\ &\text{Where, } ED_m^t = ED_B^t + ED_{G2V_m}^t \end{aligned} \tag{4.29}$$

Equation (4.29) shows that the total demand ED_m^t to a micro grid should always be less than or equal to the total supply ES_m^t for that micro grid. In the proposed model, ES_m^t is considered as the combination of grid self-generation energy (i.e., renewable and no renewable energy), the energy from other micro grids and EVs energy which discharge their batteries at time t.

B. For buildings with renewable energy: As seen earlier, buildings with renewable energy sources have the opportunity to supply local energy. Thus, their objective is to maximize the use of stored energy in the wall battery when the price is high and the demand supply curve is not stable, so we have:

$$\begin{aligned} &Maximize \sum_{i=1}^{dev} ed_{r_i}^t \\ &\text{Subject to } ED_{r_m}^t = 0 \\ &\text{Where, } \textstyle\sum_{i=1}^{dev} ed_{r_i}^t = ES_{wall_r}^t - ES_{wall_r}^{t-1} \end{aligned} \tag{4.30}$$

$$soc_{wall_r}^{t-1} \geq ES_{wall_r}^{t} \geq \sum_{i=1}^{dev} ed_{r_i}^{t} > cap_{wall_r}^{th} \quad (4.31)$$

Equation (4.30) shows that the total demand from buildings with renewable energy sources to micro grid m should be null when the demand supply curve of that micro grid is not stable at time t, and the constraint in Equation (4.31) must be respected.

C. For EVs charging and discharging service: Based on the proposed network architecture (see Figure 4.1), we assume that EVs can communicate with their micro grid fog platforms, in order to collect information about EVPSS state, price, etc., In our model, we assume that each EV creates own profile. The EV profile is represented as follows:

$$EV_{Profile}[User_{Name}, EV_{ID}, User_{password}, Home_{Adress},$$

$$Battery_{type}, Battery_{efficiency}, SoC, Speed]$$

An EV may create several calendars to express several charging/discharging requests. We believe that the storage capacity inside the fog can be very beneficial, especially when the number of users increases. The EV calendars are scheduled inside the fog server for each micro grid m; then, the suitable calendar, in terms of price and scheduled time, will be sent to all EVs.

We consider two types of pricing optimizations for EVs users in the micro grid m ; the sale price for charging $p_{m_{G2V}}^{t}$, and the buying price for discharging $p_{m_{V2G}}^{t}$. These prices are calculated as in Equation (4.28), but the base price for charging is not equal to the base price for discharging, and it is decided according to different constraints like production cost, so we have:

$$p_{m_{G2V}}^{t} = \{ \tan^{-1}\left(e^{D_m^t}\right) + (\tan^{-1} R_m^t)^{10} \} + p_{base}^{G2V} \quad (4.32)$$

$$p_{m_{V2G}}^{t} = \{ \tan^{-1}\left(e^{D_m^t}\right) + (\tan^{-1} R_m^t)^{10} \} + p_{base}^{V2G} \quad (4.33)$$

As we can see in Equation (4.9) and Equation (4.23), EVs contribute on the supply-demand curve. In the one hand, EVs users aim to consume more energy when the sale price $p_{m_{G2V}}^{t}$ is low and discharge more energy when the buying price $p_{m_{V2G}}^{t}$ is high. Thus, the micro grid controller looks for maximizing the gain as in Equation (4.29) and this by supporting more EVs that need to be charged by selecting charging calendars when the demand supply curve is stable (i.e., $ED_m^t \leq ES_m^t$), and minimizing the number of EVs requiring the discharging, at the same time slot interval. On the second hand, when the demand-supply curve is not-stable (i.e., $ED_m^t > ES_m^t$), users prefer to discharge their EVs because the buying price is automatically high, and

the micro grid wants to maximize the number of discharging EVs by selecting discharging calendars in order to keep the demand-supply curve more stable. Our proposed model uses EVs charging and discharging calendars in order to predict the demand supply energy curve.

In this regard, we define our linear optimization problems for EVs charging and discharging process in different situations; a) the demand-supply curve is stable, and b) when they are not-stable, as follows:

a) If the supply demand curve is stable, ($ED_m^t \leq ES_m^t$):

$$Maximize \sum_{i=1}^{V'} ED^t_{ev_i^{G2V}} p^t_{m_{G2V}}$$

Where, $ED^t_{ev_i^{G2V}} = \alpha^t_{ev_i^{G2V}} - soc^t_{ev_i^{G2V}}$

Subject to $\sum_{t\in T} ED_m^t \leq \sum_{t\in T} ES_m^t$ and $\alpha^t_{ev_i} \leq cap_{ev_i}$ (4.34)

b) If the supply demand curve is not-stable ($ED_m^t > ES_m^t$):

$$Maximize \sum_{i=1}^{V'} ES^t_{ev_i^{V2G}} p^t_{m_{V2G}}$$

Where, $ES^t_{ev_i^{V2G}} = soc^t_{ev_i^{V2G}} - \beta^t_{ev_i^{V2G}}$

Subject to $\sum_{t\in T} ED_m^t \leq \sum_{t\in T} ES_m^t$ and $\beta^t_{ev_i^{V2G}} \geq cap^{th}_{ev_i}$ (4.35)

Where, $cap^{th}_{ev_i}$ is the predetermined minimum threshold that cannot be surpassed. We use this constraint to save the reserved energy in EV battery for an emergency usage.

4.5 Optimization Decision Algorithms for Energy Management

In this section, we define the utility for micro grid energy management with the consumed energy from buildings and EVs, supply energy from renewable sources and discharging EVs batteries and real time price. We discuss the decision making process for building energy management and EVs charging and discharging service by implementing algorithms to solve the optimization problems in Equation (4.29), Equation (4.30), Equation (4.34) and Equation (4.35). In order to solve the optimization problem in Equation (4.29), we start by solving the problems in Equation (4.30), Equation (4.34) and Equation (4.35).

4.5.1 Algorithm for Buildings with Renewable Energy Sources

Algorithm 1 shows the energy management inside buildings, and the switching process of energy usage between micro grid energy and the wall battery energy. The algorithm is executed inside the smart meter. The time complexity of the algorithm is $O(1)$;

Algorithm 1: Renewable Building Energy Management (RB_EM)

Input: Total supply ES_m^t, Total demand ED_m^t, $soc_{wall_r}^t$, $cap_{wall_r}^{th}$.
Output: Energy demand from the micro grid $ED_{r_m}^t$.

1. Calculate $soc_{wall_r}^t$ as in Eq.(4.14) until $soc_{wall_r}^t = cap_{wall_r}$;
2. **if** $(E_{dem}^t \geq E_{sup}^t)$ **then**
3. **if** $(soc_{wall_r}^t > \sum_{i=1}^{dev} ed_{r_i}^t)$ **then**
4. Switch to the wall battery energy usage;
5. Calculate $soc_{wall_r}^t$ as in Eq.(4.16);
6. **else**
7. **while** $(soc_{wall_r}^t > cap_{wall_r}^{th})$ **repeat**
8. Switch to the wall battery energy usage;
9. $soc_{wall_r}^{t+1} = soc_{wall_r}^t - \sum_{i=1}^{dev} ed_{r_i}^t$;
10. **until** $(soc_{wall_r}^{t+1} = cap_{wall_r}^{th})$;
11. $ED_{r_m}^{t+1} = \sum_{i=1}^{dev} ed_{r_i}^{t+1} - soc_{wall_r}^{t+1}$;
12. Request $ED_{r_m}^{t+1}$ energy from the micro grid;
13. **end**
14. **else**
15. Switch to the micro grid energy usage;

end

4.5.2 Algorithm for EVs Charging and Discharging Management

As seen earlier, EVs may create several calendars to express several charging and discharging requests. We discuss the decision making process of EVs calendars for charging and discharging process.

Algorithm 2: EV Optimal Calendar Management (EV_OCM)

Input: Total supply ES_m^t, Total demand ED_m^t, EVs charging/discharging calendars.
Output: Optimal calendar, Optimal EVPSS, Total cost $\mathcal{C}_m^{\mathcal{D}}$.

```
for (each EV_Profile ) do
    Read (EV_Calendars) at time slot t;
    if (ED_m^t > ES_m^t) then
        Select discharging calendars cal_type^t = discharging;
        Calculate utility function U_EV for discharging;
        Calculate the total cost C_m^D to discharge;
        Choose the optimal cost to maximize the utility;
        Requests EVs to discharge their battery;
    else
        Select charging calendars cal_type^t = charging;
        Calculate utility function U_EV for charging;
        Calculate the total cost C_m^D to charge;
        Choose the optimal cost, to maximize the utility;
        Requests EVs to charge their battery;
    end
      Update the energy supply ES_m^t;
      Update the energy demand ED_m^t;
      Calculate prices;
      Broadcast optimal calendars and prices to EVs users;
end
```

The proposed algorithm (i.e., EV_OCM) aims to solve the optimization problems in Equation (4.34) and Equation (4.35), by taking decisions according to the available energy, distance to the charging-discharging station, and real-time price. In this regards, we define the utility function of each EV $\mathcal{U}_{EV}$ depending on various parameters as follows:

$$\mathcal{U}_{EV}(R_m^t, p_m^t, cal_{type}^t, \mathcal{D}_m^{evpss}, soc_{ev_i}^t, \mathcal{C}_m^{\mathcal{D}}, \mathcal{Q}_m^{\mathcal{D}}) \tag{4.36}$$

Where, $R_m^t, p_m^t, cal_{type}^t, \mathcal{D}_m^{evpss}, soc_{ev_i}^t, \mathcal{C}_m^{\mathcal{D}}$ and $\mathcal{Q}_m^{\mathcal{D}}$ are the demand supply curve, price, calendar types (charging or discharging), distance to EVPSS, state of charge of ev_i battery, total cost to charge the battery and the available energy to join the EVPSS, respectively. Therefore, the decision making process of EV charging and discharging calendars considers the global information presented in the utility function $\mathcal{U}_{EV}$. We use a multi attribute decision making methodology to take the optimal decision, as explained below.

Let us consider M micro grids in particular EVs vicinity. Then, the proposed decision matrix, which is based on several decision parameters, is represented as follows:

$$\begin{pmatrix} R_1^t & \cdots & R_M^t \\ p_1^t & \cdots & p_M^t \\ \mathcal{D}_1^{evpss} & \cdots & \mathcal{D}_M^{evpss} \\ \mathcal{C}_1^{\mathcal{D}} & \cdots & \mathcal{C}_M^{\mathcal{D}} \\ \mathcal{Q}_1^{\mathcal{D}} & \cdots & \mathcal{Q}_M^{\mathcal{D}} \end{pmatrix} \tag{4.37}$$

Assumption 4: Due to the EVs mobility, an EV can charge and discharge its battery in any micro grid m, even if its calendar is not saved in the according micro grid fog. In this regard, communications between micro grids fog servers are authorized in order to ensure the roaming of EVs.

So, the decision instruction ψ(ω) with full information can be written as:

$$\psi(\omega) = argmin\ (\omega, m_{ev_i}) \tag{4.38}$$

Where, $m_{ev_i}$is the selected micro grid from which ev_i charge or discharge energy, $m_{ev_i} \in \mathbb{M}$ and $ev_i \in \mathbb{V}$, and ω is the set of full information parameters, i.e.,

$$\omega = \{R_M^t, p_M^t, \mathcal{D}_M^{evpss}, \mathcal{C}_M^{\mathcal{D}}, \mathcal{Q}_M^{\mathcal{D}}\} \tag{4.39}$$

We show in Algorithm 2 (i.e., EV_OCM), an optimal management of charging and discharging calendars based on the decision matrix represented in Equation (4.37). The algorithm is decentralized and executed inside each micro grid fog controller. The time complexity of the algorithm is $O(n)$; where n is the number of $EV_{Profile}$.

4.5.3 Algorithm for Micro Grid Energy Management

The objective of each micro grid m is to maintain the stability of the demand supply curve, and determine a real time price of energy. Therefore, we have a global maximization problem represented in Equation (4.29), where each micro grid always looks to maximize its utility. In this regards, we propose a centralized algorithm (DP_DEM) to manage and control all micro grids in different situations. Also, it uses SDN and NFV technologies to supervise the communication networks between micro grids, and control the data flow in order to maintain the total smart grid demand supply curve, by maximizing the number of stable micro grids. The algorithm is executed inside the smart grid cloud controller, as a centralized master algorithm

based on the proposed architecture. The time complexity of the algorithm is $O(m)$; where M is the number of micro grids.

Algorithm 3: Micro Grids Energy Management (DP_DEM)

Input: ES_m^t, ED_m^t and p_m^t of each micro grid m;
Output: list of stable micro grids, list of non-stable micro grids, real time price of each micro grid m;

```
1. for (m = 1 to M) do
2.     Receive ES_m^t, ES_m^t, p_m^t at time t;
3.     Update list of stable micro grids;
4.     Update list of non-stable micro grids;
5.     while (list of non − stable micro grids ≠ null) repeat
6.         Execute (RB_EM) and (EV_OCM) for micro grid m;
7.         Create virtual communications based SDN-Controller
8.         Request exceeded energy from stable micro grids;
9.         Receive Exceeded energy;
10.        Calculate energy difference D_m^t for m;
11.        if (D_m^t ≥ 0) then
12.            Remove m from list of non-stable micro grids;
13.            Add m to list of stable micro grids;
14.        end
15.    until (list of non − stable micro grids = null) ;
16.    Calculate prices
17.    Broadcast list of stable micro grids;
18.    Broadcast list of non-stable micro grids;
19.    Broadcast real time prices of each micro grid;
   end
```

4.6 Performance Evaluation

4.6.1 Simulation Settings

We implement extensive simulations to evaluate the performance of the proposed dynamic pricing model based on a decentralized Cloud-SDN architecture. We use SUMO to implement EVs mobility and MATLAB Simulink to execute the proposed energy management algorithms. We consider a real electric load in the city of Toronto, divided into 05 micro grids and managed using smart grid controller. We examine the energy load (i.e., EV charging and discharging, renewable energy supply, building consumption) during a day (24 h) starting from 12:00 AM. The day is evenly divided into 288 intervals. Each interval has a length of 5 minutes (min). For EVs mobility modeling, we consider a scenario of 10 roads that allow access to a city having

area of 10 km x 10 km where EVPSSs are placed randomly. All vehicles are traveling with speeds that cannot exceed 60 km/h. Table 4.1 shows the parameters used for simulation.

Table 4.1- Simulation parameters

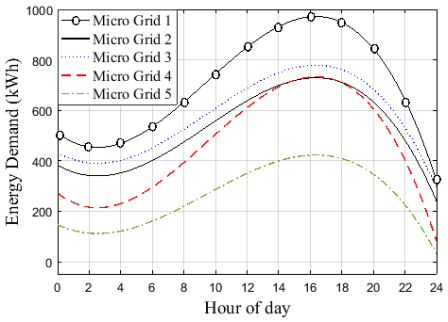
Parameter	Value
Number of EVs	1000
Initial EV SoC for charging	[1 kW – 40 kW]
Initial EV SoC for discharging	[30 kW – 50 kW]
Wall Battery capacity	[20 kWh – 80 kWh]
EV Battery capacity	60 kWh [22]
$cap^{th}_{wall_r}$, cap_{ev_i}	20 %
p^{G2V}_{base}	10 cent
p^{V2G}_{base}	5 cent

4.6.2 Performance Metrics

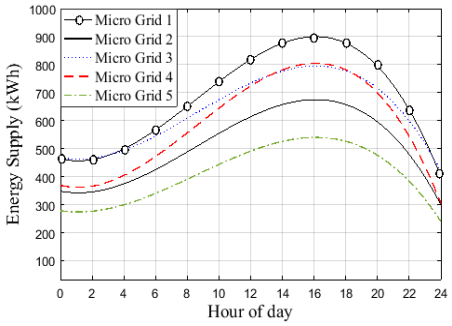
In order to evaluate our proposed model, we use different metric as follows:

- *Dynamic real time price:* the price is calculated as in Equation (4.28) for buildings energy consumption and as in Equation (4.32) and Equation (4.33) for EVs charging and discharging respectively.
- *Demand supply curve:* is the principal metric used to inspect the performance of the proposed model, especially during peak hours. The demand supply curve is evaluated from Equation (4.26) and Equation (4.27).
- *Charging delay:* is the sum of the fog provider response time, time to travel to the selected EVPSS, the waiting time in the queue and the service time.
- *EV charging cost:* is consistently related to the real time price and distance to the charging station of the selected micro grid. It is calculated as in Equation (4.38).
- *Utility for EV charging:* EV_OCM algorithm takes decisions based on the decision matrix proposed in Equation (4.37). We calculate the utility for EV as charging cost using D2P [47], D2R [50], UDP [48], QFC [49] and the proposed pricing model DP-DEM. Thus, we denote the utility function as follows:

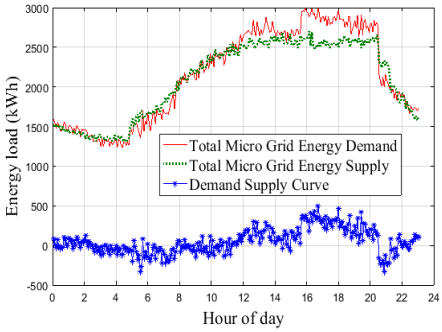
$$\mathcal{U}_{EV}\left(ED^t, p^t, \mathcal{D}_m^{evpss}\right) = \begin{cases} ED^t(p^t_{D2P} - p^t) \\ ED^t(p^t_{D2R} - p^t) \\ ED^t(p^t_{UDP} - p^t) \\ ED^t\left(p^t_{QFC} - p^t\right) \end{cases} \tag{4.40}$$



(a) Real time energy demand for micro grids



(b) Real time energy supply for micro grids



(c) Total energy supply demand curve

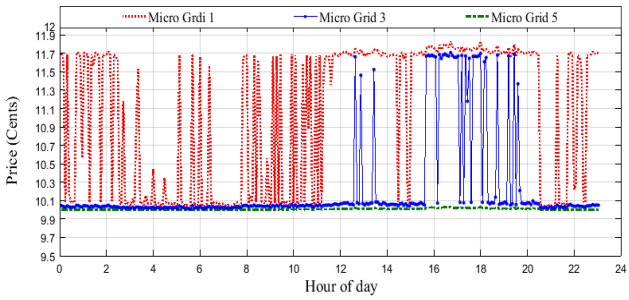
Figure 4.2- Real time supply and demand for micro grids and total supply demand curve

Where p_{D2P}^t p_{D2R}^t, p_{UDP}^t and p_{QFC}^t denote the real time prices obtained using D2P, D2R, UDP and QFC pricing policy, respectively.

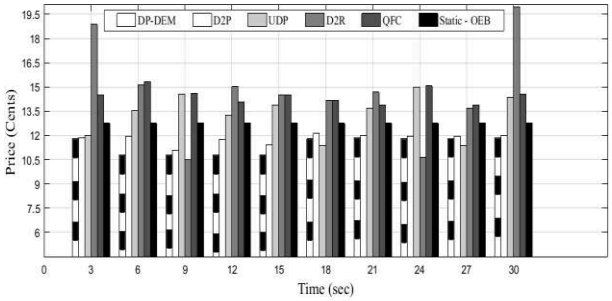
4.6.3 Results and Comparison with Exiting Works

In order to keep our simulation scenarios more realistic, we use a real load consumption, Figure 4.2(a) and Figure 4.2(b) show the energy consumption and the energy supply, respectively, for each micro grid, simulated by scaling the real energy load in Toronto city by a factor of 1/1500 and divided into 05 micro grids. Figure 4.2(c) shows the total energy demand, the total energy supply for all micro grids and the demand supply curve including the charging and the discharging load of EVs. We can see that the demand supply curve is not stable from [12:00 pm to 08:00 pm].

To evaluate the impact of the proposed dynamic pricing policy, we compare the price over different micro grids situations in Figure 4.3(a). For micro grid 1, the curve is not stable; we can observe different energy peaks during a day, consequently the price is varied according to the variations in the energy load. For micro grid 3, we have two peaks, mid-peak period [12:00 pm to 14:00 pm] and evening-peak period [17:00 pm to 20:00 pm]. We can see that the price is



(a) Energy load impact on price



(b) Comparison with related works

Figure 4.3- Impact of energy load on the proposed pricing policy and price comparaison

high (11, 7 cents) compared with other periods (10, 00 cents). Finally, the price for micro grid 5 is stable because the demand supply curve of the micro grid is regular during all the day. In Figure 4.3(b), we compare our proposed price policy DP-DEM with existing works, UDP, D2R, QFC and static pricing policy from Ontario Energy Board [99].

In simulation, we assume that each micro grid calculates the real time price every 3 seconds interval. We can see that with the implementation of DP-DEM, the cost to consume energy is less than that with the other pricing policies; the proposed distributed pricing policy gives better result than the centralized one. In the proposed scenario, we see that a reasonable pricing policy, using the proposed pricing equations, is also maintained compared with D2R price policy for example.

Figure 4.4, shows the charge that is held at 10 house batteries (we selected 10 house of a no stable micro grid). We run simulations using Renewable Building Energy Management (RB_EM) algorithm. We can observe that the charge energy in house batteries' (represented by the green plot) decrease during morning peak hours and evening peak hours, this is due because the total demand supply curve is not stable so, and according to our proposed pricing policy the price of one unit of energy increase, as results house devices use the house battery energy. However, we can see that from 12:30 pm and from 20:30 pm the charge level of the house

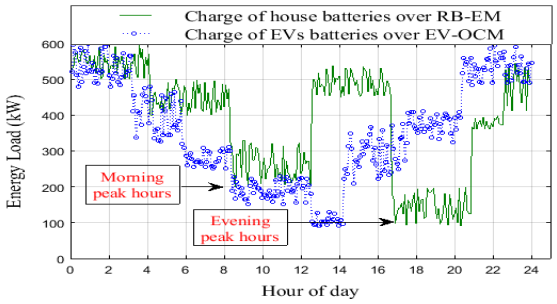


Figure 4.4- Charge of house batteries and EVs batteries

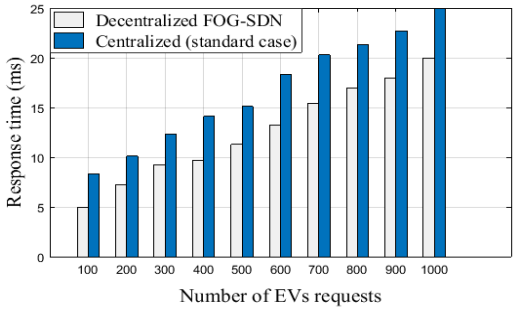
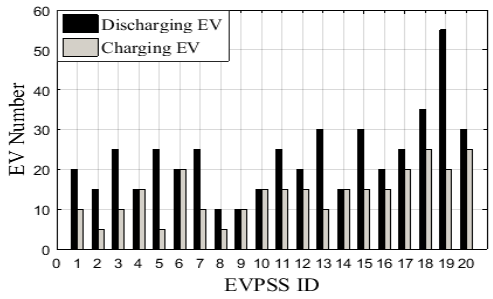


Figure 4.5- Response time with the number of EVs requests

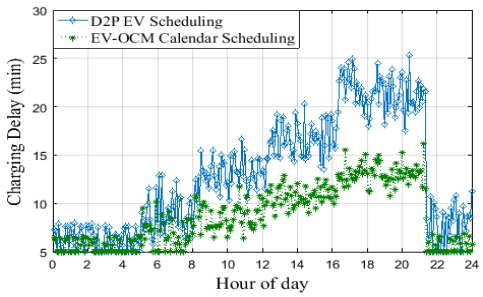
batteries increases, this is because the demand supply curve is stable so the price decrease, so the proposed scheme make profit of this situation and start charging the house batteries.

In order to specify why it is useful to use Cloud-SDN architecture in the smart grid environment, we present in Figure 4.5 the implementation of decentralized Fog-SDN architecture for micro grids, where we use response time as a performance metric (This is the time it takes for the first response to come after processing of a dataset). We implement two scenarios as follows: i) Centralized standard scenario without using SDN technology. ii) Decentralized scenario using Fog-SDN technology.

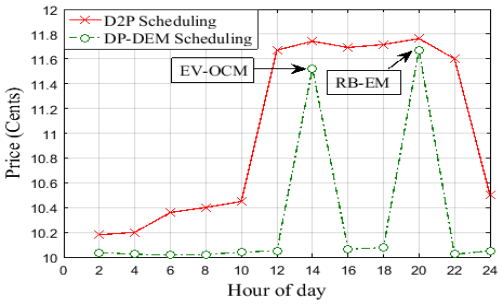
For all simulation tests, the topology shown in Figure 4.1 was used where a finite number (1000) of EVs submit their requests for charging. We have 03 data centers (i.e., each micro grid has its data center); the data centers contain different scheduling servers (i.e., represented as virtual machines VMs with 2 GO capacity in our simulation parameters). In order to evaluate to effectiveness of the proposed scheduling scheme based on a decentralized Fog-SDN architecture, we evaluate the proposed architecture using ns-2. Figure 4.5 shows the response time of end users (EVs requests) waiting for scheduling charging/discharging demands from the grid. When the number of EVs requests is increased, the proposed Fog-SDN approach yields a lower response time compared to the approach making use of the traditional centralized core



(a) EVs number in peak hours over EV-OCM



(b) Delay for discharging the EVs battery

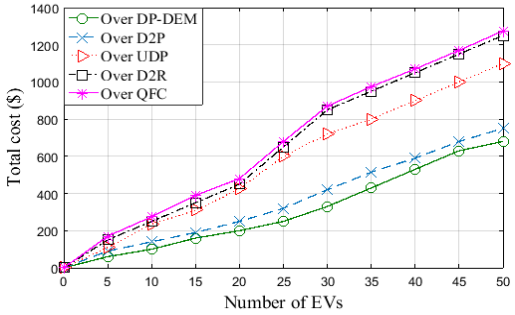


(c) Price variation and comparison during a day

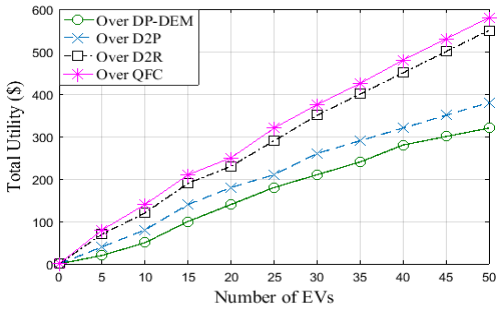
Figure 4.6- EVs prices optimization and comparison

network infrastructure. The lower response time is because when the centralized core network is used, services need to be accessed far away from remote sites that reduce the overload on the core. In the proposed decentralized scheme, the services are immediately available from the fog computing resources, thereby minimizing the response time.

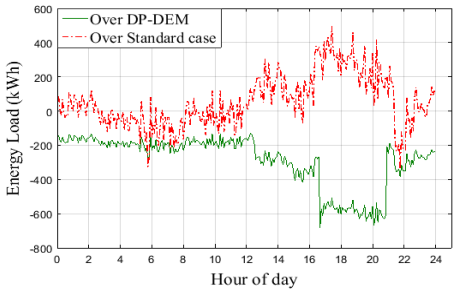
The performances of our proposed algorithms (i.e., DP-DEM, EV-OCM and RB-EM) are shown in Figure 4.6. In Figure 4.6(a), we evaluate the execution of the optimal EVs calendars management algorithm during peak hours. We can observe that the number of discharging EVs is higher than the charging EVs; this is because the demand supply curve is not stable and the EV-OCM selects discharging calendars in order to maximize the utility function $\mathcal{U}_{EV}$ for discharging by choosing the optimal cost. Also, the price in that period is high and EVs users profit to maximize their gain by discharging the EV battery. Figure 4.6(b) shows the utility function for the discharging delay for the same number of EVs in Figure 4.6(a). We can see that the delay with EV-OCM algorithm is less than that with the utility function that uses D2P pricing policy, especially during peak hours. This is because the calendars scheduling helps micro grid controller to predict the energy load and the EVPSS state at time t. On the other hand, the proposed decentralized architecture based on SDN networking and fog computing, reduces the response time by controlling all network entities (i.e., EVs, EVPSS, AMI …).



(a) Total cost for the required energy from EVs



(b) Utility comparison for EVs while charging



(c) Demand supply curve stabilization

Figure 4.7- EVs charging cost and utility comparison, and demand supply curve comparaison

In Figure 4.6(c), we evaluate the price variation during a day, and compare the DP-DEM scheduling policy with the D2P scheduling. First, we can see that our price is less than that of D2P price during all the day including peak hours and non-peak hours. Second, we see that with the proposed DP-DEM, we have two peaks during a day at 02:00 pm (mid-peak hours) and at 08:00 pm (evening peak hours); however, the price is reduced from (11.4 cents) to (10.05 cents) when the DP-DEM send notifications to all micro grids controllers located at the data centers to execute the EV-OCM algorithm (at 02:00 pm). Thus, the utility for discharging EVs is maximized as seen in Figure 4.6(a) and Figure 4.6(b). Also, at evening peak hours, DP-DEM sends notifications to all micro grids to execute the RB-EM algorithm in order to manage the energy inside the buildings and switches to the use of energy from the wall battery. DP-DEM scheduling can help each micro grid to reduce the price, especially during peak hours, by keeping the demand supply curve stable compared with the D2P scheduling.

The total charging cost for EVs is shown in Figure 4.7(a) using DP-DEM (proposed), D2P, UDP, D2R, and QCF pricing policies. Based on the energy requests from each EV, we see that with the increase in the number of EVs, DP-DEM outperforms D2P, D2R, UDP and QCF scenarios. In Figure 4.7(b), we compare the utility of EVs while charging. Using the proposed dynamic pricing policy DP-DEM, the utility of EVs increases compared to that of D2P, D2R and QFC. This is because DP-DEM algorithm supervises all micro grids energy flow

and decides when sending notifications to each micro grid to execute EV-OCM, in order to maximize the EV charging and discharging utility and minimize the cost. Finally, we evaluate in Figure 4.7(c) the demand supply curve state during a day over the proposed scheduling. We can observe a significant stabilization of the demand supply curve compared to the standard case; an important energy saving (600 kWh) is obtained during evening peak hours. Thus, our proposed model is capable to regulate the total demand curve.

4.7 Chapter Summary

In this chapter, we proposed a distributed dynamic pricing model for EVs charging and discharging scheduling and building energy management in smart grid, which considers real-time and energy demand constraints. The model is based on a decentralized cloud-SDN communication architecture. We used a linear optimization approach to achieve the price decision process, maintain the demand-supply curve stable, and insure more grid efficiency. Also, we use a multi attribute decision making methodology to take optimal decisions aiming to maximize the utility for smart grid users. We solved the optimization problems using two distributed algorithms; one for EV charging and discharging calendar, and the second for managing building renewable energy. We implemented a centralized algorithm in order to manage all micro grids. Our algorithms aim to ensure the balance between the charging and discharging requests of EVs and balancing energy usage between grid energy and renewable energy. We used dynamic pricing equations to incite users to consume their energy when the price is optimal for the smart grid and for users. As proved in simulations and comparisons with four other works, using real electric load in the city of Toronto, the proposed decentralized architecture can be useful for smart grid applications. Furthermore, the proposed pricing policy can also improve the grid stability, especially in peak hours.

CHAPTER 5

FOG-BASED SECURE SMART METERING ARCHITECTURE IN SMAT GRID

In order to secure smart metering infrastructure against false data injection attacks in smart grid environment, we propose in this chapter a new security scheme (called SecMetering) based on hierarchical and distributed intrusion detection system (HD-IDS). The proposed HD-IDS is based on distributed fog architecture considering three hierarchical network layers. The problem is formulated using stochastic modeling. Based on a Semi-Markov process, we implement an embedded Markov chain to observe the behavior of each smart meter using range-based behavior sifting policy. In addition, we propose four algorithms to analyze efficiently the behavior of smart meters and detect false data intrusion attacks in real-time way at different layers of the proposed HD-IDS architecture. We validate the effectiveness of SecMetering via extensive simulations. We evaluate the impact of SecMetering on the smart grid energy curve by illustrating real-word traces of households' electricity consumption (from the U.S. electricity system operating data-city of California). Finally, SecMetering is compared with three recent and relevant related works.

5.1 Motivation

Different security fears in smart grid energy metering using smart meters may come from several attackers that can use different techniques (e.g., physical attacks, cyber-attacks, etc.). The attackers may be external and/or local. Local attackers may be the users, the gateways or the operation center. Generally, these attackers are honest-but-curious [16]; they attempt various techniques to seek and deduce knowledge about victims, using for example their energy consumption measurements. It is stated that the determination of users' behavior figures in the second most serious attacks consequences on smart grid environment in 14 fears types defined by Electronic Privacy Information Center [16].

On the other hand, an external attacker may compromise a smart meter by injecting false data or spying on the communication channel to violate user's privacy. The false data injection attack (FDIA), first proposed by Liu et al. [17], is a cyber-attack where power system state estimation outputs are corrupted by injecting false data into meter measurements in a carefully coordinated fashion. The defining feature of a successful FDIA is that the state estimation residual falls below a hypothesis test threshold despite the presence of corrupted measurements; the attack thereby may evade detection.

Yet, smart meters, even they have protection against physical damage using sophisticated microcontrollers, they are still vulnerable to electricity theft and false data injection attacks. Furthermore, as a fully trusted party, a trusted authority should be implemented in the AMI system model; it is responsible to insure random secret keys to household users and operation center through a secure communication channels. Hence, some issues like how to design an effective privacy preserving, secure and fault tolerant smart metering scheme, and AMI architecture resisting to power theft and detecting false data injections in smart grid, still merit more investigations and effort by the both academia and industry.

Motivated by the aforementioned issues, we propose SecMetering, a novel security scheme for electricity smart metering against false data injections with range-based sifting. The proposed scheme includes a hierarchical and distributed intrusion detection system (i.e., HD-IDS) based on new distributed fog architecture for AMI. By ensuring accurate false data detection, SecMetering aims to deal with real-time consumption report failures in smart grid environment. We believe that the distributed monitoring based on fog architecture allows early detection of planned and coordinated attacks in AMI. Moreover, it allows the network administrators to localize intrusions and to take preventive measures rapidly.

To the best of our knowledge, our work is one of the first attempts to design a hierarchical and distributed intrusion detection system based on Fog architecture to secure smart metering against false data injection in smart grid energy environment. We validate our solution using real-word traces of households' electricity consumption of California city. The main contributions of this paper are as follows:

1) We propose SecMetering scheme, with the desirable features of false metering data injection detection, under hierarchical and distributed IDS (i.e., HD-IDS). The HD-IDS is designed on smart grid environment based on distributed fog architecture. The objective is to implement three protection levels over the fog layers (home area network gateway, residential area network gateway, and fog operation center).

2) We use a mathematical process based on stochastic modeling to formalize the false data metering detection problem, where we study the behavior of each smart meter. To do this, we prove that our stochastic process is a Semi-Markov process. Then, the behavior study of smart meters is represented by an embedded Markov chain under seven possible smart meter states (e.g. authentic, suspicious, malicious, etc.). Then, we define the parameters of the chain (e.g., state transition diagram, states holding times, and transit probabilities). The objective is to efficiently analyze the behavior of each smart meter and then detect the intrusion immediately.

3) To define the states of our embedded Markov chain, we use a range-based sifting policy. Where, the operation center predicts the range of electricity consumption according to historical power data, load of appliances and users profile. To classify each smart meter into the appropriate state, we use several thresholds; the thresholds can be updated periodically according to the user's energy profile.

4) We propose three algorithms to analyze the intrusions behaviors. First, for each state, we define: suspicious behavior control algorithm (SuspBehCon) for detecting suspicious smart meter, malicious behavior control algorithm (MalBehCon) for detecting malicious smart meter, and observation control algorithm (ObsCon) for smart meter placed under observation.

The remainder of this chapter is organized as follows. In Section 5.2, we describe the proposed HD-IDS architecture. Section 5.3 presents the stochastic modeling of false data injections. Section 5.4 presents the intrusion behaviors analysis algorithms. The performance evaluation of the proposed HD-IDS solution is discussed in Section 5.5. Finally, Section 5.6 concludes the paper.

5.2 HD-IDS Fog-Based Smart Metering

The proposed distributed fog architecture for smart grid electricity consumption metering has three layers as follows (Figure 5.1):

1) Cloud layer: It includes a centralized operation center. This latter interacts directly with the smart grid central operation center. *2) Fog layer:* As seen in Figure 5.1, for each micro grid, we associate a decentralized fog data center. This latter contains different servers (e.g., database server, management server, application server); each server provides various services for customers. In addition, a data center can communicate with the micro grid control center in order to update and deliver real-time information to the company utilities (for instance about prices and real-time energy consumption). *3) AMI layer:* The AMI layer for one fog (i.e., micro grid) contains different smart grid users' types (e.g., households, community residents, factories, electric vehicles …). The daily energy consumption measurements from all these users are counted periodically using smart meters according to a predefined time slots defined by the fog operation center. At the beginning of each slot, each smart meter submits real-time measurements of electricity consumption to the home gateway (i.e., HGW) via a local area network using wireless communication technologies (e.g., ZigBee which is relatively low in power usage, data rate, and cost implementation). Next, each home gateway sends the metering result (as a report) to the residential gateway (i.e., RGW) via residential area network (see Fig.

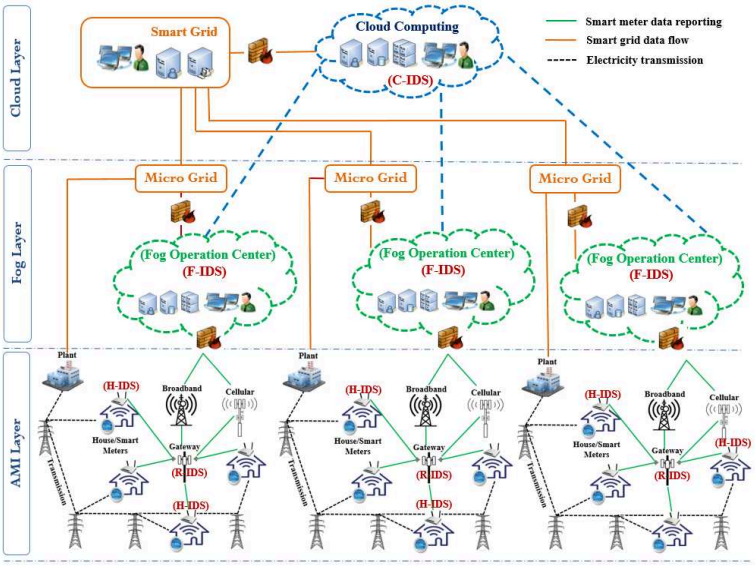


Figure 5.1- Secure AMI system architecture based on HD-IDS

1) using long-range wireless technologies (e.g., LoRa). Upon receiving the electricity consumption reports from different smart meters, RGW aggregates the reports to a compressed report (aiming to reduce the communication overhead) and forwards it to the appropriate for operation center.

To efficiently secure the metering system based on the proposed distributed fog architecture, we implement a new Hierarchical Distributed Intrusion Detection System (HD-IDS) over a layered network. In order to insure a secure electricity metering and data forwarding and aggregation, we implement three control levels, hierarchically, between the fog operation center and smart meters. To do this, we deploy three IDS agents over AMI layers as follows: a) a Home area network IDS (i.e., H-IDS) implemented inside the home gateway (HGW) b) a Residential area network IDS (i.e., R-IDS) implemented inside the residential gateway (RGW), and c) a fog network IDS (i.e., F-IDS) implemented inside the fog operation center. Each IDS agent can communicate with the operation center and other IDSs according to the hierarchy (Figure 1).

5.3 Stochastic Modeling of False Data Metering Detection

To address the aforementioned problems and enhance the AMI system architecture, we propose a stochastic modeling based on Semi-Markov processes to study the behavior of each smart meter.

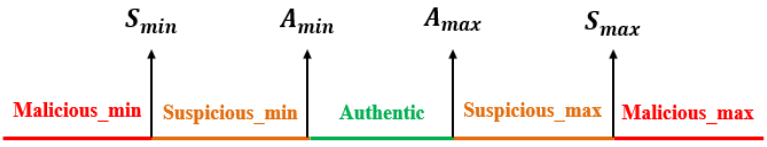


Figure 5.2- Smart meter measurements ranges with thresholds

5.3.1 Range-based measurements sifting policy

As seen earlier, we have three detection levels according to the proposed architecture; the input data into each IDS is the measurements of electricity consumption counted by each smart meter. We aim to study the behavior of each smart meter, and then classify it into five classes as shown in Figure 5.2 *authentic, suspicious-min, malicious-min*, *suspicious-max,* and *malicious-max.* The main idea of this process is to introduce five ranges with four thresholds A_{min}, A_{max}, S_{min}, and S_{max} as shown in Figure 5.3; and then, we can identify the class of each smart meter measurement. The thresholds values are determined and retrieved from the historical energy profile of each user. In our work, the thresholds are determined from real households' energy consumption datasets from the U.S. electricity system operating data [100]. Thus, we analyze the daily energy consumption of 1000 households in California during winter period (i.e., November 1 – April 30) including weekdays, weekends and statutory holidays.

5.3.2 Stochastic Modeling of Smart Meters' Behavior

The tracking of data metering behaviors of smart meter can alleviate the uncertainty caused by false data injections attacks. Therefore, we propose in this section a stochastic model to identify the stationary state of smart meter based on its previous evolution. We use the range-based measurements sifting policy defined in the previous sub-section to represent the space of possible states of each smart meter. We assume that we have, for each smart meter, seven states according to the reported measurement at a reporting time t_R denoted by $e_{id}^{t_R}$ where id is the identifier of the smart meter. The states are listed as follows:

- *Authentic state (i.e., A):* the smart meters having a measurement between A_{min} and A_{max}, $(A_{min} \leq e_{id}^{t_R} \leq A_{max})$.
- *Suspicious_max state (i.e., S_{max})*: the smart meters having a measurement between A_{max} and S_{max}, $(A_{max} < e_{id}^{t_R} \leq S_{max})$.
- *Suspicious_min state (i.e., S_{min})*: the smart meters having a measurement between A_{min} and S_{min}, $(A_{min} < e_{id}^{t_R} \leq S_{max})$.

- *Malicious_max state (i.*e., M_{max}): the smart meters having a measurement superior to S_{max}, ($e_{id}^{t_R} > S_{max}$).
- *Malicious_min state (i.*e., M_{min}): the smart meters having a measurement inferior to S_{min}, ($e_{id}^{t_R} < S_{min}$).
- *Observation state (i.e., ODS):* a smart meter can be placed in this state, if a it stays a period of time denoted by t_S, in one of the two states *Suspicious_max* and *Suspicious_min* where $t_S > t_R$*;* in other words, if a smart meter measurement value is between A_{max} and S_{max} or between A_{min} and S_{min}, during a period of time t_S or during a number of steps (each measurement report represents one step).
- *Stop state (i.e., STOP):* a smart meter can be placed in this state when the measurement value is within the malicious range (*Malicious_max or Malicious_min)*, and then the attack is confirmed. Moreover, a smart meter can be placed in this state after the observation decision (we will explain the observation process later).

In order to explain the different states transitions and behaviors, we present in Figure 5.3 a state transition diagram of smart meter with the predefined seven states. As shown in Figure 5.3, the state space is given by

$$E = \{M_{min}, S_{min}, A, S_{max}, M_{max}, OBS, STOP\} \quad (5.1)$$

From Figure 5.3, it is easy to verify that the chain is irreducible and aperiodic with a semi-Markov process.

Let $x_{k,t}$ be the state of a smart meter k, and X_t be the state of the smart meters of one residential area at a reporting time t_R. We give the stochastic process modeling $(X_{t_R})_{t_R}$ of a smart meters of a residential area as follows:

$$X_{t_R} = \bigcup_{k=1}^{n} x_{k,t_R} \quad (5.2)$$

Where n is the number of smart meters in the residential area network. The evolution of the network is represented by a Semi-Markov process, where, the process $(x_{k,t_R})_{\ t}$ is memoryless (i.e., the present, the past and the future are independent for all smart meters n). In our case study, the memoryless property of the Markov chain means that the state of a smart meter at future time $(t + \varepsilon)$ relies on the smart meter state at the current time t, and does not depend on the state at earlier time instance $(t - \varepsilon)$.

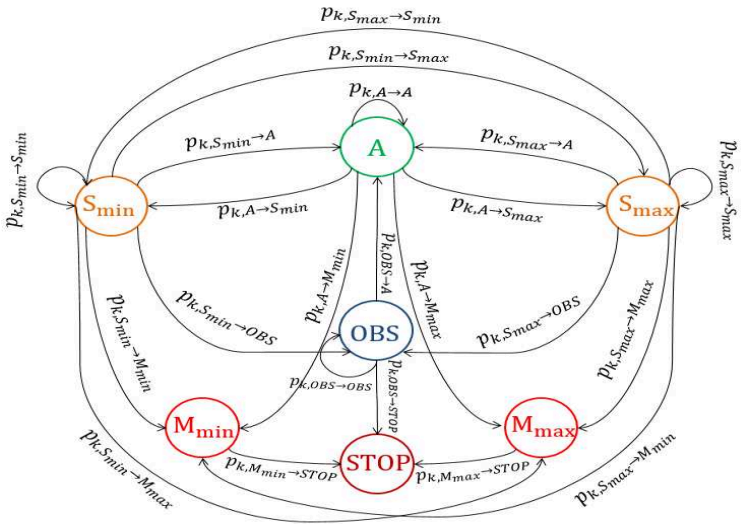


Figure 5.3- State transition diagram for one smart meter

We define the probability transition of a smart meter k from state i to state j for all $(i,j)\ \epsilon\ E^2$ as follows:

$$p_{k,ij} = p\left(x_{k,t+1} = j \middle| x_{k,} = i\right) = p(x_{k,1} = j | x_{k,0} = i) \tag{5.3}$$

After defining the states of our Markovian process and their different properties, we use a stochastic state transition matrix denoted by $\mathbb{M}_{P_k}$ to represent the evolution of a measurement of a smart meter k over the time. The matrix contains the probabilities of transition between two states (i.e., classes according to the range-based measurements sifting policy) of k, where $P_k = (p_k, ij)_{i,j\ \in E}$.

We give the probability of each matrix $\mathbb{M}_{P_k}$ entry by the following:

$$p_{k,ij} = p\left(x_{k,t+1} = j \middle| \cup_{k=1}^{n} x_{k,t} = i\right) = \frac{\alpha_{ij}}{\beta_{ij}} \tag{5.4}$$

Where α_{ij} is the number of transitions of smart meter k from current state i to another state j, and β_{ij} is the expected number of visits to state i.

Each entry $p_{k,i\to j}$ denotes the conditional probability that smart meter k moves from state i to state j. As we can see in Fig. 3, we have not transitions from states STOP to any other states. Thus, this is called an absorbing state, once entered the system stays in this state; this means that the attack is happened and an intrusion alert should be signaled to the corresponding smart meter. However, all other states $(i.e., S_{min}, A, S_{max}, OBS, STOP)$ are transient states, because starting from them, there is a positive probability that the chain will never return. Besides, they are positive recurrent where we can have a sequence of states starting and ending at i.

5.3.3 Semi-Markov Process-based Embedded Markov Chain

In this subsection, we consider a generalization of our proposed Markov chain, termed Semi-Markov process. In such process, we augment its specification by including a state holding time parameter.

Thus, our main interest is to define the *holding time* of a smart meter at each state. The *holding time* of state i, denoted by H_i, is the amount of time that passes before making a state transition from i. The value of H_i depends on the next state transition. Thus, let $H_{i,j}$ be the holding time of state i given that the next state transition is the state j. According to the law of total probability, we have

$$H_i = \sum_{j=0}^{\infty} I_{i,j} H_{i,j} \tag{5.5}$$

Where $I_{i,j}$ is an indicator random variable that equals 1 with the probability $p_{i,j}$. As we know, Markov chains have state holding times that are equal to a unit time (i.e., a step) and that are independent of the next state transition (i.e., $H_i = H_{i,j} = 1$). However, in Semi-Markov process, holding times do not have to be one-time unit nor do they have to be independent of the next state transition. A Semi-Markov process operates in the following way:

- After entering state i, the process randomly selects its next state according to a probability transition matrix.
- Given that state j is selected, the time spent in state i before jumping to state j is given by random variable $H_{i,j}$.

A Semi-Markov process selects its next state and then selects a value for its holding time. On the other hand, the sample paths for Semi-Markov processes are timed sequences of state transitions. Where, we can write a sample path by a sequence of pairs as follows:

$$(i_0, t_0), (i_1, t_1), (i_2, t_2), \dots, (i_{k-1,}\, t_{k-1}), (i_k, t_k), \dots$$

This formulation denotes a sample path where the process is in state i_k over the time period (t_k, t_{k+1}). However, if we view the process only at the times of state transition then the sample paths are identical to Markov chain, with transition Matrix that have entries as follows:

$$p_{i,j}^{e} = \begin{cases} 0, & i = j \\ p_{i,j}/(1 - p_{i;j)}, & i \neq j \end{cases} \tag{5.6}$$

Now, our main interest is to define the stationary probabilities of the embedded Markov chain. Let π_i^e be the stationary probability of our embedded chain then,

$$\pi_i^e = \sum_{j \neq i} \pi_j^e p_{j,i}^e \tag{5.7}$$

π_i^e is the probability that, at the embedded points, the process is found in state i. We give the following proposition.

The state holding time is independent of j, and is geometrically distributed with an average given by

$$E[H_i] = \frac{1}{1 - p_{i,i}}, \ \ i = 0, 1, \ldots \tag{5.8}$$

Proposition 1: *the probability of finding the Semi-Markov process of smart meter in state i at a randomly selected time is not given by π_i^e.*

Proof: to proof Proposition 1, we use a counterexample, supposing that the holding time of state x is extremely large with respect to the holding times of other states. Formerly, with a high probability, a random point in time finds the process in state x regardless of the value of π_i^e. Thus, we should define the stationary distribution of the Semi-Markov chain process with the embedded chain. Let π_i denotes the stationary probability of the Semi-Markov chain. It is clear that the value of π_i depends on the stationary probabilities of the embedded chain as well as on the holding time of each state.

Theorem 1*:* let us consider a Semi-Markov process of a smart meter with state holding time H_i and the embedded state transition probabilities **P**. We assume that the embedded Markov chain is positive recurrent and that π_i^e are its stationary probabilities. The stationary distribution of smart meter Semi-Markov process is proportional to the product $\pi_i^e E[H_i]$ and it is given by

$$\pi_i = \frac{\pi_i^e E[H_i]}{\sum_{j=0}^{\infty} \pi_j^e E[H_j]} \tag{5.9}$$

Proof: to proof Theorem 1, it is sufficient to show that the portion of time spent in state i is given by π_i. For this, let $T_i(m)$ be the portion of time spent in state i over the first m steps of the embedded chain process and let $M_i(m)$ be the number of times that state i is visited during these m steps. For $j = 1, 2, 3, \ldots$, let $L_i(j)$ be the random variable that denotes the length of time the process spends in its j^{th} visit to state i. From these definitions we have

$$T_i(m) = \frac{\sum_{j=1}^{M_i(m)} L_i(j)}{\sum_{k=0}^{\infty} \sum_{j=1}^{M_k(m)} L_k(j)} \tag{5.10}$$

As shown earlier, our Markov chain is positive recurrent, so $M_i(m) \rightarrow \infty$ as $m \rightarrow \infty$, and the strong law of large numbers implies that

$$\sum_{j=1}^{M_i(m)} \frac{L_i(j)}{M_i(m)} \rightarrow E[H_i], \ \ as \ \ m \rightarrow \infty \tag{5.11}$$

From the Markov chain property, we can see successive visits to state i as a renewal sequence (i.e., in our case study, successive visits to state *A (authentic state)* represent a renewal sequence). Thus, the expected number of steps between visits is given by

$$\frac{1}{\sum_{m=1}^{\infty} f_{i,i}^{m}} = \frac{M_i(m)}{m} \tag{5.12}$$

On the other hand, and according to the properties of the recurrent states renewal process of Markov chains, if i is recurrent and if $i \leftrightarrow j$ then

$$\frac{M_i(m)}{m} \rightarrow \pi_i^e \ \ as \ m \rightarrow \infty \tag{5.13}$$

Where,

$$\pi_i^e = \left[\sum_{m=1}^{\infty} m f_{i,i}^{m}\right]^{-1} = \frac{1}{\sum_{m=1}^{\infty} m f_{i,i}^{m}} \tag{5.14}$$

Finally, by substituting Equation (5.11), Equation (5.12), and Equation (5.14) into Equation (5.9), we obtain

$$\pi_i = \frac{\pi_i^e E[H_i]}{\sum_{j=0}^{\infty} \pi_j^e E[H_j]}$$

Therefore, theorem 1 is proved.

Now, by combining the notion of a state holding in the description of semi-Markov process, we can study the operation of the embedded chain without self-loops.

Definition 1: *a self-loop is a transition from a state immediately back to itself (with probability $p_{i,i}$ for state i).*

To model the operation of embedded Markov chains without self-loops, we distingue two equivalent cases as follows:

- *Self-loops and unit holding times:* as shown in Equation (5.3), the Markov chain changes state at each step and has self-loops. The holding time in this case is one unite of time.
- *No self-loops and geometric holding times:* in this case, we see the chain only at epochs where the chain makes transition that are not self-loops. The Markov chain operates as a Semi-Markov process. When entering state i, the chain remains there until it jump to another state j according to the transition matrix of the embedded Markov chain.

On the one hand, for the *No self-loops states*, the stationary probability of the Semi-Markov process denoted by $\pi_i^{\sim}$, $i = 0, 1, ...$, from Equation (5.8), $\pi_i^{\sim}$ can be written as follows:

$$\pi_i^{\sim} = \frac{\pi_i^e/(1-p_{i,i})}{\sum_{j=0}^{\infty}\pi_j^e/(1-p_{j,j})} \tag{5.15}$$

On the second hand, for the *self-loop states*, as proved later these states represent an irreducible Markov chain with one ergodic set, then the stationary probabilities are unique and given as follows:

$$\pi_i = \sum_{j=0}^{\infty}\pi_j p_{j,i} \tag{5.16}$$

Yet, it is clear that the two cases of the Markov chain given above have the same stationary distribution. Consequently, we can rewrite the global balance equation of Equation (5.16) for π_i as follows:

$$\pi_i = \frac{1}{1-p_{i,i}}\sum_{j\neq 1}\pi_j p_{j,i} \tag{5.17}$$

Likewise, Equation (5.7) can be rewritten a follows:

$$\pi_i^e = \sum_{j\neq i}\pi_j^e \frac{p_{j,i}}{1-p_{j,j}} \tag{5.18}$$

Finally, Equation (5.15) involves that

$$\pi_i^{\sim} = \frac{1}{1-p_{i,i}}\sum_{j\neq i}\pi_j^{\sim} p_{j,i} \tag{5.19}$$

5.4 Intrusion Behaviors Analysis Algorithms

In this section, we analyze the behavior of each smart meter measurements according to the proposed Markov chain. For this end, we propose several algorithms as follows:

5.4.1 Suspicious Behavior Control Algorithm

In this algorithm, we aim to control the behavior of smart meters placed in suspicious lists (i.e., suspicious_min $L_{S_{min}}^{t_R}$, suspicious_max $L_{S_{max}}^{t_R}$). Thus, for each element (i.e., smart meter) of the both lists, we calculate the holding times $H_{S_{min}}^{id}$ and $H_{S_{max}}^{id}$, and compare them respectively with the predefined thresholds denoted by $H_{S_{min}}^{id_{th}}$ and $H_{S_{max}}^{id_{th}}$. If the thresholds are exceeded, we place the smart meter in the observation list, otherwise, the smart meter stays in the suspicious states waiting for the next measurement to efficiently analyze its behavior. On the other hand, we should report a valid measurement $(e_{id}^{t_R})_{V_{S_{min/max}}}$ to the micro grid. For this end, we suppose to calculate the average between the reported data (i.e., $(e_{id}^{t_R})_{S_{min}}$, $(e_{id}^{t_R})_{S_{max}}$) and the historical data $(e_{id}^{t_R})_{Hist_A}$ (calculated according to the pervious authentic measurement and user energy profile).

Algorithm 1: Suspicious Behavior Control (SuspBehCon)

Input: $L_{S_{min}}^{t_R}, L_{S_{max}}^{t_R}, H_{S_{min}}^{th}, H_{S_{max}}^{th}$;
Output: $L_{OBS}^{t_R}$, $L_{S_{min}}^{t_R}$, $L_{S_{max}}^{t_R}$, $(e_{id}^{t_R})_{V_{S\_min}}$, $(e_{id}^{t_R})_{V_{S\_max}}$;

1. Read $(L_{S_{min}}^{t_R}, L_{S_{max}}^{t_R})$;
2. **for** (all $L_{S_{min}}^{t_R}$ elements $id = 1, \dots, n$ and all $L_{S_{max}}^{t_R}$ elements) **do**
3. Calculate $H_{S_{min}}^{id}$ and $H_{S_{max}}^{id}$ as in Equation (5.8);
4. **if** $(H_{S_{min}}^{id} > H_{S_{min}}^{id_{th}})$ **then** — **if** $(H_{S_{max}}^{id} > H_{S_{max}}^{id_{th}})$ **then**
5. $p_{id,\, S_{min} \to OBS} \leftarrow 1$; — $p_{id,\, S_{max} \to OBS} \leftarrow 1$;
6. $L_{OBS}^{t_R} \leftarrow (e_{id}^{t_R})_{S_{min}}$; — $L_{OBS}^{t_R} \leftarrow (e_{id}^{t_R})_{S_{max}}$;
7. Update $(L_{S_{min}}^{t_R})$; — Update $(L_{S_{max}}^{t_R})$;
8. **else** — **else**
9. $p_{id,\, S_{min} \to S_{min}} \leftarrow 1$; — $p_{id,\, S_{max} \to S_{max}} \leftarrow 1$;
10. **end** — **end**
11. $(e_{id}^{t_R})_{V_{S\_min}} = \frac{((e_{id}^{t_R})_{Hist_A} + (e_{id}^{t_R})_{S_{min}})}{2}$;
12. $(e_{id}^{t_R})_{V_{S\_max}} = \frac{((e_{id}^{t_R})_{Hist_A} + (e_{id}^{t_R})_{S_{max}})}{2}$;
13. Send $\left((e_{id}^{t_R})_{V_{S\_min}}\right), \left((e_{id}^{t_R})_{V_{S\_max}}\right)$ to the Micro Grid;
14. **end**

5.4.2 Malicious Behavior Control Algorithm

In this algorithm, we control the malicious behavior of smart meters, where we confirm the intrusion and place all smart meters in $L_{STOP}^{t_R}$ (the stop list) and then update the lists. Regarding the reported valid measurement, it is calculated using the last historical authentic measurement.

Algorithm 2: Malicious Behavior Control (MalBehCon)

Input: $L_{M_{min}}^{t_R}, L_{M_{max}}^{t_R}$;
Output: $L_{STOP}^{t_R}$, $L_{M_{min}}^{t_R}$, $L_{M_{max}}^{t_R}$ $(e_{id}^{t_R})_{V_{M\_min}}$, $(e_{id}^{t_R})_{V_{M\_max}}$;

1. Read $(L_{M_{min}}^{t_R}, L_{M_{max}}^{t_R})$;
2. **for** (all $L_{M_{min}}^{t_R}$ elements $id = 1, \dots, n$ and all $L_{M_{max}}^{t_R}$ elements) **do**
3. $p_{id,\, M_{min} \to STOP} \leftarrow 1$;
4. $p_{id,\, M_{max} \to STOP} \leftarrow 1$;
5. $L_{STOP}^{t_R} \leftarrow (e_{id}^{t_R})_{M_{min}}$; $L_{STOP}^{t_R} \leftarrow (e_{id}^{t_R})_{M_{max}}$;
6. Update $(L_{M_{min}}^{t_R})$; //delete $(e_{id}^{t_R})_{M_{min}}$ from $L_{M_{min}}^{t_R}$;
7. Update $(L_{M_{max}}^{t_R})$; //delete $(e_{id}^{t_R})_{M_{max}}$ from $L_{M_{max}}^{t_R}$;
8. $(e_{id}^{t_R})_{V_{M\_min}} \leftarrow (e_{id}^{t_R})_{Hist_A}$;
9. $(e_{id}^{t_R})_{V_{S\_max}} \leftarrow (e_{id}^{t_R})_{Hist_A}$;
10. Send $\left((e_{id}^{t_R})_{V_{S\_min}}\right), \left((e_{id}^{t_R})_{V_{S\_max}}\right)$ to the Micro Grid;
11. **end**

5.4.3 Observation Algorithm

We analyze in this algorithm the behavior of smart meters placed in the observation list (i.e., observation state). If the observation list is not empty, we calculate the holding time H_{OBS}^{id} of each smart meter id placed in the observation state, and define a holding time threshold $H_{OBS}^{id_{th}}$ for each smart meter. Note that the observation list contains both elements transited from suspicious_min and suspicious_max. Thus, for both type of elements while the thresholds are not exceeded, we compare the reported measurements with the historical ones based on the user energy profile. If the measurement is reasonable, we confirm it with the micro grid operation center, and place the concerned id smart meter in the authentic list. Rationally, we should update the thresholds $(A_{min}^{id}, A_{max}^{id}, S_{min}^{id}, S_{max}^{id})$ according to the new energy profile of the user. However, if the measurement is not confirmed, while the holding time threshold is not exceeded, the smart meter stays in the observation state waiting for next measurements. Nevertheless, when the threshold time is exceeded, the smart meter is placed immediately in the stop list, and the historical measurements are reported to the micro grid operation center.

Algorithm 3: Observation control Algorithm (ObsCon)

Input: $L_{OBS}^{t_R}, H_{OBS}^{id_{th}}$;
Output: $L_{STOP}^{t_R}$, $L_{OBS}^{t_R}$, $(e_{id}^{t_R})_{V_{OBS}}$, $A_{min}^{id}, A_{max}^{id}, S_{min}^{id}, S_{max}^{id}$;

```
Read (L_OBS^{t_R});
for (all L_OBS^{t_R} elements id = 1, ..., n) do
    if (L_OBS^{t_R} = ∅) then exit;
    else
        Calculate H_OBS^{id} as in Equation(5.8);
        while (H_OBS^{id} < H_OBS^{id_th}) do
            if ((e_id^{t_R})_OBS^{S_min}) is confirmed
                p_{id, OBS→A} ← 1;
                L_A^{t_R} ← (e_id^{t_R})_OBS^{S_min};
                Update (L_OBS^{t_R});
                (e_id^{t_R})_{V_OBS} ← (e_id^{t_R})_OBS^{S_min};
                Update (A_min^{id}, A_max^{id}, S_min^{id}, S_max^{id});
            else
                p_{id, OBS→OBS} ← 1;
                (e_id^{t_R})_{V_OBS} ← (e_id^{t_R})_{Hist_A};
            end
            p_{id, OBS→STOP} = 1;
            L_STOP^{t_R} ← (e_id^{t_R})_OBS^{S_min}; L_STOP^{t_R} ← (e_id^{t_R})_OBS^{S_max};
            Update (L_OBS^{t_R});
            (e_id^{t_R})_{V_OBS} ← (e_id^{t_R})_{Hist_A};
        end
    end
end
Send ((e_id^{t_R})_{V_OBS}) to the Micro Grid;
end
```

if ((e_id^{t_R})_OBS^{S_max}) is confirmed
    p_{id, OBS→A} = 1;
    L_A^{t_R} ← (e_id^{t_R})_OBS^{S_max};
    Update (L_OBS^{t_R});
    (e_id^{t_R})_{V_OBS} ← (e_id^{t_R})_OBS^{S_max};
else
    p_{id, OBS→OBS} ← 1;
    (e_id^{t_R})_{V_OBS} ← (e_id^{t_R})_{Hist_A};
end

5.5 Performance Evaluation

In this section, we evaluate the performance of our proposed solution via extensive simulations using real-word traces of households' electricity consumption of California city. We consider the following criteria, intrusion detection rates, time cost, communication overhead, energy curve, and energy price under various system topologies. In addition, we study the impact of false measurement injection attack on smart grid environment. Finally, we compare our solution with three other relevant and recent works (i.e., [56], [57] and [58]).

5.5.1 Simulation Scenarios and Metrics

In our simulations, we used MATLAB Simulink to implement the proposed mathematical model, and to evaluate the three proposed algorithms. We use NS3 to implement the proposed AMI architecture, and VMs to represent fog servers. Each VM contains a storage capacity to store the historical energy consumption data of each household located in the appropriate fog. The thresholds are calculated from real households' energy consumption datasets from the U.S. electricity system operating data [100]. We implement two fog topologies. In the first topology, we implement our securing solution (SecMetering). The second topology is the traditional smart grid environment without security functions (called Trad). Moreover, in order to compare the effectiveness of the implemented IDSs at different levels (i.e., HAN-IDS, RAN-IDS, and Fog-IDS), we implement several topologies: a topology only with HAN-IDSs, a topology only with RAN-IDSs, and a topology only with Fog-IDS.

We evaluate our solution over different metrics as follows:

- *Intrusion detection rate:* It is the detection rate of intrusions at each level of the hierarchy.
- *Communication overhead:* It is the time needed for each smart meter to report the measurement data between the HGW and the fog operation center.
- *Computation time:* It is the time needed for each topology to detect an intrusion.
- *Energy metering:* It is the energy consumption measured by smart meters and received by the fog operation center. This data includes also false data injected by an attacker.
- *Price:* It is the electricity price proposed in [36].
- *Time cost:* It is the time needed to report a valid energy measurement to the micro grid operation center [101].

5.5.2 SecMetering performance evaluation

Figure 5.5 shows the performance evaluation of SecMetering. Figure 5.5(a) presents the number of attacks generated randomly at each reporting time slot (i.e., at each time slot of 10 minutes, smart meters' report measurements of electricity consumption) [101-102]. In order to efficiently evaluate the performance of our proposed HD-IDS implemented in SecMetering, we generate false data injection attacks at different levels of the hierarchy (i.e., HAN, RAN, and FAN). In Figure 5.6(b), we compare the intrusion detection rate of the four implemented topologies (HD-IDS, HAN-IDS, RAN-IDS, Fog-IDS). We can observe that HD-IDS topology outperforms other topologies in terms of detections rates. This is because our HD-IDS includes three detection levels (i.e., in the home area network, in the residential area network, and in the

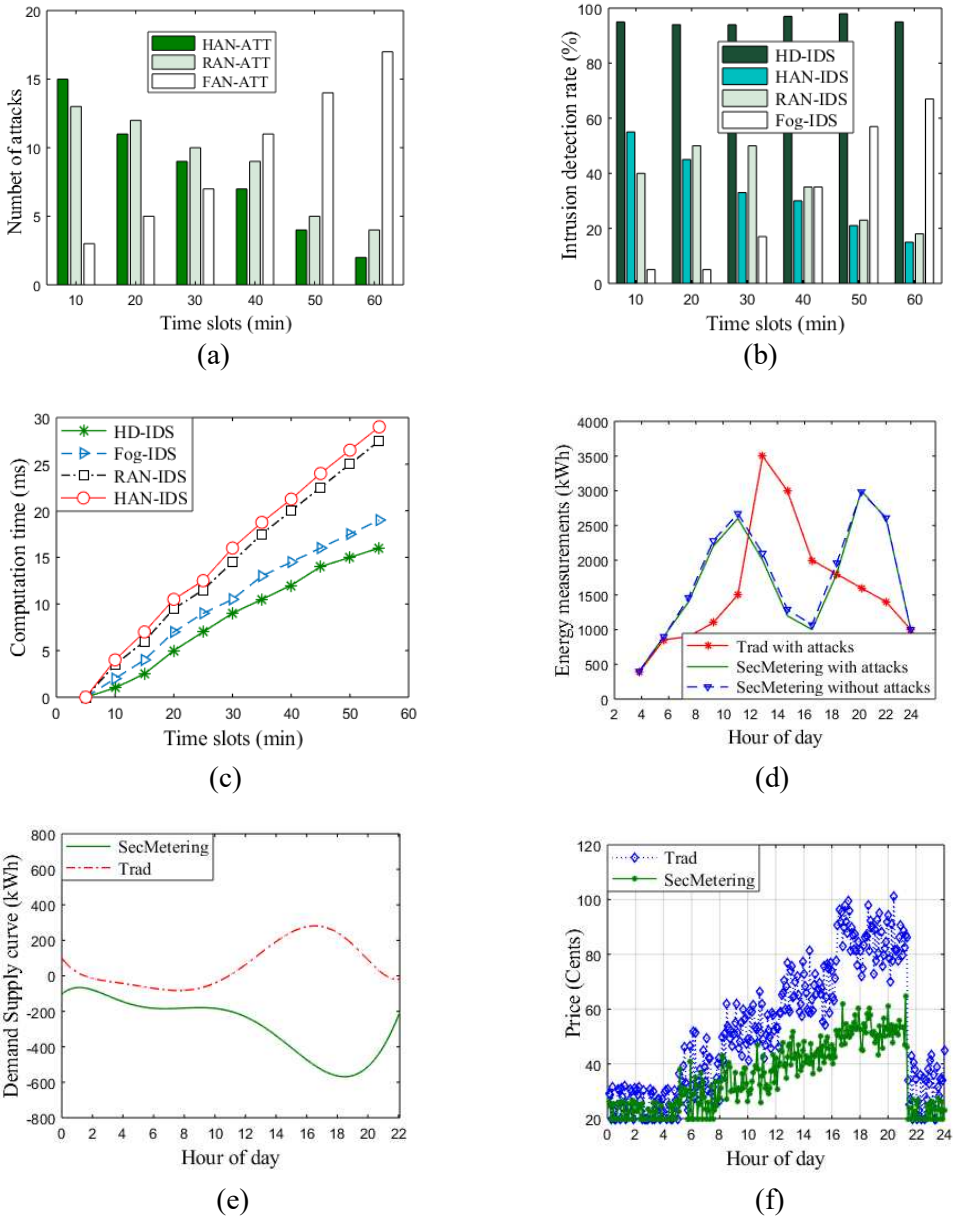


Figure 5.5- SecMetering performances evaluation. (a) Number of generated attacks. (b) intrusion detection rate. (c) computation time. (d) energy curve. (e) Energy metering. (f) Price comparison.

fog operation center), where false data injection attacks are detected efficiently at each network gateway. We compare in Figure 5.5(c) the computation time (time needed to detect an intrusion by our proposed algorithms) for each topology. We observe that HD-IDS topology outperforms other topologies by offering a smaller computation time (by around 50% for HAN-IDS, 40% for RAN-IDS, and 25% for Fog-IDS). This is because HD-IDS topology can detect intrusions in a parallel and hierarchical way.

Now we evaluate the impact of false metering data injection on smart grid energy curve. In Figure 5.5(d), we observe the values of energy consumption measurements during a day, and

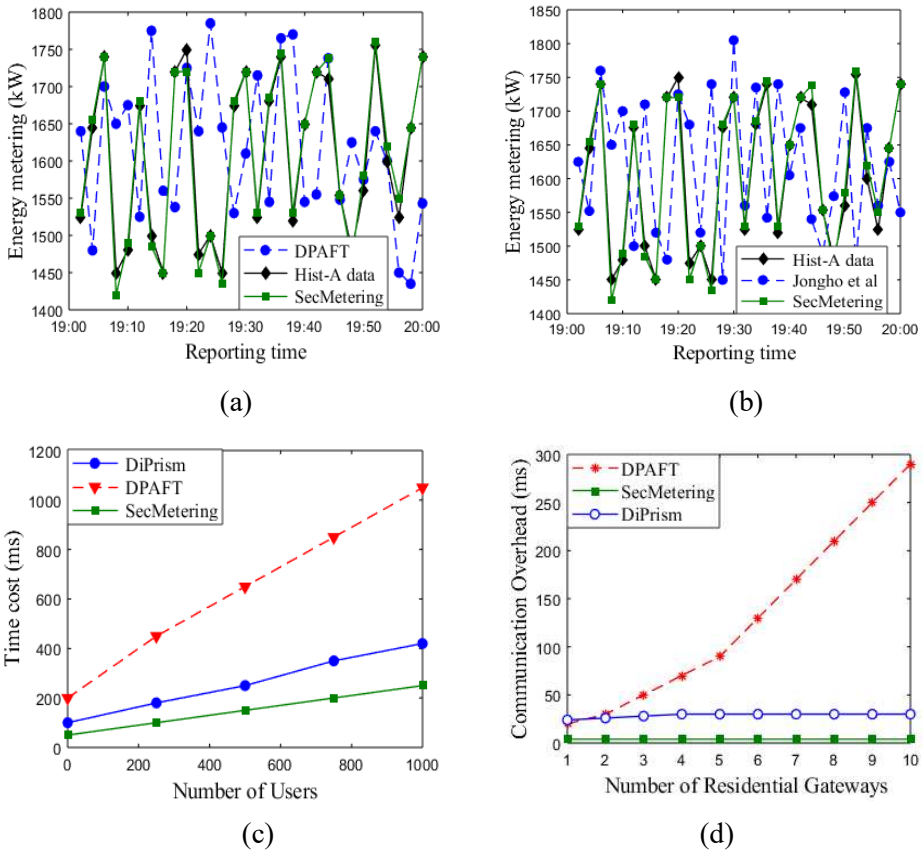


Figure 5.6- SecMetering comparison with related works. (a) Energy metering with DRAFT. (b) Energy metering with Jongho et al's scheme. (c) Time cost comparison. (d) Communication overhead.

compare three topologies (Trad with attacks, SecMetering with attacks and SecMetering without attacks). We observe clearly that the secure topology (i.e., SecMetering with attacks) has proximately the same values as the topology without attacks (i.e., SecMetering without attacks). This is explained by the efficiency of the proposed intrusions behavior analysis algorithms, where each behavior is analyzed distinctly tacking into account both historical behavior and user energy profile. In addition, the algorithms report the authentic historical data to replace immediately the false data injected when the intrusion is detected. Figure 5.5(e) shows the impact of false data metering injection on the smart rid energy curve (the difference between the energy demand and the energy supply). As we can observe, compared with the traditional architecture without any security protocol, SecMetering solution lets the curve more stable especially during evening peak hours. In Figure 5.5(f), we evaluate the impact of false measurement on electricity price; we use the dynamic pricing policy proposed in [62], where the price is calculated on real time according to the energy demand-supply curve. We observe that false data metering can affect seriously the price during the day; however, SecMetering

gives normal prices according to the real energy consumption from real and secure measurements.

5.5.3 SecMetering Comparison with Related Works

In Figure 5.6, we compare SecMetering with three recent works DiPrism [56], DPAFT [57], Jongho et al [58]. In Figure 5.6(a) and Figure 5.6(b), we compare our SecMetering scheme with DPAFT and Jongho schemes respectively. The simulations are done using the same parameters used in [56], [57] and [58], by generating realistic consumption traces during evening peak hours for 2000 households. We generate false data injection attacks at different levels and at each reporting time. As it can be seen from the figure, SecMetering, compared to DPAFT and Jongho schemes, gives approximatively the same energy metering reports as given in the historical authentic metering data (i.e., the real consumption data saved without attacks). This is because SecMetering improves the accuracy of a large scale of failures reports (caused by the data injection attack), by tacking in consideration all states of the smart meter, where the attack is confirmed after an efficient analysis of the behavior of each smart meter using the proposed algorithms. Moreover, the failure reports are replaced by the authentic historical data, thus the micro grid receives a confirmed measurement from the fog operation center.

Finally, we compare time cost, for detecting attacks and forwarding reports, of SecMetering with DiPrism and DPAFT. As shown in Figure 5.6(c), our scheme costs less time than DiPrism and DPAFT for a number k of smart meters. Since each smart meter report is represented by Markov chain and is analyzed individually over the proposed algorithm (according to the state of each smart meter) using high computing and storage capacities of fog servers, then the time cost for reporting and analyzing one smart meter is relatively low compared with the other schemes where all measurement reports are analyzed together using one ciphertexts. Figure 5.6(d) shows the comparison results of SecMetering, DiPrism, and DPAFT in terms of communication overhead considering different number of RGW. The results demonstrate the efficiency of SecMetering, in terms of report transmission between RGW and fog operation center, compared with DiPrism and DPAFT. In DPAFT, the gateways forward all reports without aggregation. In DiPrism the reports are aggregated however RGW is required to send only 996 bytes, which may cause communication overhead if the number of users increases or if RGWs receives a high number of attacks where any IDS is implemented in these schemes (DiPrism and DPAFT), compared with SecMetering.

5.6 Chapter Summary

In this Chapter, we proposed a new scheme for securing smart meters called SecMetering. SecMetering is based on a new hierarchical and distributed intrusion detection system (HD-IDS). The proposed HD-IDS operates on a distributed fog architecture considering three hierarchical network layers (i.e., home area network, residential area network, and fog operation center network). The problem is formulated using stochastic modeling based on Markov chain process. The advantage of the proposed HD-IDS solution is proved over different performance metrics and extensive simulations. Compared with other insecure architectures and recent schemes, our solution has shown better false detection rates stable energy curve, and optimal communication overhead. Finally, thanks to the distributed nature of the architecture, the proposed hierarchical distributed intrusion detection system (HD-IDS) approach can be conclusive for other types of attacks as Distributed Deny of Service (DDoS) attacks, where we can enhance the proposed embedded Markov chain to detect and localize the source of the DDoS attack, and to avoid the fail of AMI system. In addition, we aim to develop a learning process based on historical data manning to calculate the thresholds used for detection, and to analyze the behavior of each user.

CHAPTER 6

HIERARCHICAL FOG COMPUTING FOR IIOT DATA SCHEDULING

Industry 4.0 or Industrial Internet of Things (IIoT), has become one of the most talked-about industrial business concepts in recent years. Thus, to efficiently integrate Internet of Things (IoT) technology into industry, the collected and sensed data from IIoT need to be scheduled in real time constraints; especially for big factories. To this end, we propose in this chapter a hierarchical fog servers' deployment at the network service layer across different tiers. Using probabilistic analysis models, we prove the efficiency of the proposed hierarchical fog computing compared with the flat architecture.

Our contributions are as follows: a) we design a decentralized multi-tiers fog architecture for IIoT requests scheduling and data analytics. The architecture is based on hierarchical deployment of servers in the fog layer. b) we propose a probabilistic model to compare the efficiency of fog resources utilization between flat and hierarchical fog architectures. We show, using an analytical model that the hierarchical fog architecture has better efficiency requests handling in terms of minimizing computation and communication delays. c) we implement a priority queuing model for scheduling IIoT data. We divide IIoT requests into two priority levels (high and low); we favorite high priority requests (e.g., emergency requests) to be scheduled first. d) we formulate the IIoT workload assignment problem as a mixed nonlinear integer programming (MNIP) problem, and propose a branch and bound approach to solve the problem. Then, the optimal results will be aggregated for high tiers using the Simulated Annealing Algorithm in order to ensure a global optimality of IIoT workload assignment. e) we develop an IIoT offloading algorithm (IIoT_OFF) which offloads the workloads over different tiers of the hierarchy. f) finally, we compare our proposed model with relevant and recent works in the performance evaluation Section.

The remainder of this chapter is organized as follows. In section 6.1 we present a probabilistic model for hierarchical fog servers. Section 6.2 describes the priority queuing model for IIoT data scheduling. In Section 6.3, we introduce the IIoT data assignment and offloading algorithms. The evaluation of the proposed architecture and algorithms is discussed in Section 6.4. Finally, Section 6.5 concludes the paper.

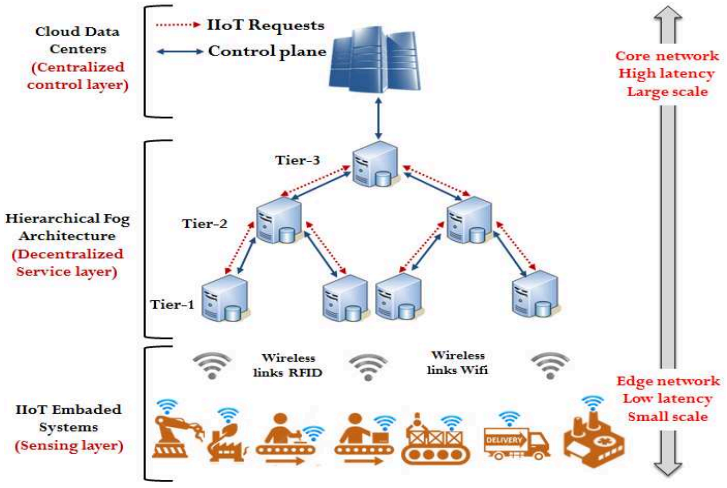


Figure 6.1- Hierarchical fog architecture for Industrial internet of things

6.1 Probabilistic Analysis Model of Fog Hierarchy

In this section, we formally prove the advantage of the hierarchical fog computing on improving the efficiency of cloud resource utilization when serving the peak load of IIoT. Without loss of generality, we first develop an analytical model for a two tier hierarchical fog. Then, the analysis results based on this two tier model are generalized to more complicated topology of the fog hierarchy.

As shown in Figure 6.2, a two tier fog hierarchy consists of s servers in tier-1, and one server in tier-2. Let c_i denotes the computational capacity of the i-th tier-1 servers, and w_i be the amount of received IIoT workload of the i-th tier-1 servers; c_i and w_i are measured in units of CPU cycles. However, in tier-2 we have only one server which has a capacity of C.

6.1.1 Probabilistic Model of Tier-1 Fog Hierarchy

We assume that $w_1, \dots, w_i, \dots, w_s$ are independent and identically distributed random variables. We denote by $\mathbb{P}(w_i \leq Cap)$ the probability that the i-th tier-1 server can successfully serve its received workload w_i when Cap is its computational capacity. The probability that all tier-1 servers can schedule their received requests from different IIoT devices and sensors is:

$$\mathbb{P}(w_1 \leq c_1,\ w_2 \leq c_2, \dots, w_s \leq c_s) = \prod_{j=1}^{s} \mathbb{P}(w_i \leq c_i) \tag{6.1}$$

According to the capacity of each server, we consider two cases as follows:

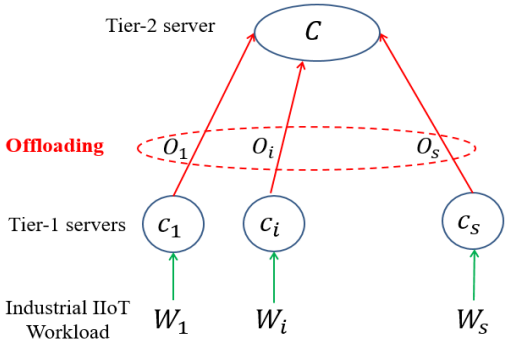


Figure 6.2- Notation in two-tier fog hierarchy for IIoT workloads

1) If $w_i > c_i$, the workload amount denoted by O_i will be offloaded on the tier-2 server (see Fig. 2). The offloaded amount O_i of one server i is given by:

$$O_i = w_i - c_i \tag{6.2}$$

2) If $w_i \leq c_i$, the offloaded workload is null (i.e. $O_i = 0$).

As results, we define the Cumulative Distribution Function (CDF) $\mathbb{F}$ of the offloaded workloads as follows:

$$\mathbb{F}_{O_i}(Cap) = \mathbb{P}(O_i \leq Cap) = \begin{cases} \mathbb{P}(w_i \leq Cap + c_i) & if\ Cap \geq 0 \\ 0 & otherwise \end{cases} \tag{6.3}$$

6.1.2 Probabilistic Model of Tier-2 Fog Hierarchy

For now, we calculate the probabilistic model for tier-2 server, systematically, the probabilistic model of tier-2 server is related to tier-1 servers. As shown in Figure 6.2, we define the total workload ω that the tier-2 server has revived from tier-1 servers as follows:

$$\omega = \sum_{i=1}^{s} O_i \tag{6.4}$$

To calculate the probability that one server in tier-2 can serve the received data from different tier-1 servers, we divide the s tier-1 servers into two groups $\mathcal{S}_1$ and $\mathcal{S}_2$ where the group $\mathcal{S}_1$includes servers from 1 to $s-1$, and $\mathcal{S}_2$ has server s (i.e., only one server).

1) We suppose that there is no workload offloaded by $\mathcal{S}_2$, so all the workloads come from $\mathcal{S}_1$, then the probability that tier-2 servers can handle their received workload ω is:

$$\mathbb{P}_{\mathcal{S}_1}(\omega) = \mathbb{P}(\sum_{i=1}^{s-1} O_i \leq Cap)\ \mathbb{P}(O_s = 0) \tag{6.5}$$

Where, $$\mathbb{P}(O_s = 0) = \mathbb{F}_{s-1}(Cap)\, \mathcal{S}_s(0) \quad (6.6)$$

2) Now, we suppose that there is a workload of amount Ω ($\Omega > 0$) offloaded by $\mathcal{S}_2$. The workloads offloaded by $\mathcal{S}_1$ cannot exceed $(C - \Omega)$, because C is the computational capacity of the tier-2 server. When Ω is fixed, the probability that the tier-2 server can serve its offloaded workload Ω is given as follows:

$$\mathbb{P}_{\mathcal{S}_2}(C) = \mathbb{P}(\sum_{i=1}^{s-1} O_i \leq C - \Omega)\, \mathbb{P}(O_s = \Omega) \quad (6.7)$$

Then, when $0 < \Omega < C$, this probability is:

$$\mathbb{P}_{\mathcal{S}_2}(C) = \int_{0^+}^{C} \mathbb{P}(\sum_{i=1}^{s-1} O_i \leq C - \Omega) d\, \mathbb{P}(O_s \leq \Omega) \quad (6.8)$$

From (6.5), (6.6), (6.7) and (6.8), we define the formula of the Cumulative Distribution Function $\mathbb{F}$ for tier-2 workload probabilistic model as follows:

Let $\mathbb{F}_s(C) = \mathbb{P}(\omega \leq C)$, and $\mathcal{S}_s(C) = \mathbb{P}(O_S \leq C)$, so we have:

$$\mathbb{F}_s(C) = \mathbb{F}_{S-1}(C)\, \mathcal{S}_s(0) + \int_{0^+}^{C} \mathbb{F}_{S-1}(C - \Omega)\, d\mathcal{S}_s(\Omega) \quad (6.9)$$

Thus, $$\mathbb{F}_s(C) = \mathbb{P}_{\mathcal{S}_1}(C) + \mathbb{P}_{\mathcal{S}_2}(C) \quad (6.10)$$

Equation (6.9) and Equation (6.10) show how to analytically derive the characteristics of ω, given the knowledge about workloads received by all tier-1 servers.

6.1.3 Analytic Comparison Model

From the above workload models, we further analyze the advantage of hierarchical fog on improving the efficiency of servers' resource utilization. As shown in Figure 2, the amount of capacity C is provisioned to the tier-2 server. Similarly, when we provision the flat fog architecture using the same amount of capacity, C will be provisioned to tier-1 servers. Moreover, we define the difference between flat (i.e., one tier) and hierarchical (i.e., multi-tier) architectures of fog servers' deployment.

To analyze the difference between flat and hierarchical designs of fog, we first have the following lemma:

Theorem 1: For any computation capacities $c_1^*, c_2^* \in [0, C], c_1^* + c_2^* = C$, we have:

$$\mathbb{P}(O_1 + O_2 \leq C) \geq \mathbb{P}(O_1 \leq c_1^*)\, \mathbb{P}(O_2 \leq c_2^*) \tag{6.11}$$

Proof: We use algebra of sets to proof this lemma. We define two sets of offloading workloads as follows:

$$\begin{cases} \mathbb{S}_1 = \{(O_1, O_2) | O_1 + O_2 \leq C\} \\ \mathbb{S}_2 = \{(O_1, O_2) | O_1 \leq c_1^*, O_2 \leq c_2^*\} \end{cases}$$

It is evident that for any element $\{\Omega_1, \Omega_2\} \in \mathbb{S}_2$, we have also $\{\Omega_1, \Omega_2\} \in \mathbb{S}_1$, so $\mathbb{S}_2 \subseteq \mathbb{S}_1$, and then we have:

$$\mathbb{P}(O_1 + O_2 \leq C) \geq \mathbb{P}(O_1 \leq c_1^*)\, \mathbb{P}(O_2 \leq c_2^*) \tag{6.12}$$

Now, we generalize the concept for all fog servers. To this, end, we extend Theorem 1 to lemma 1 as follows:

Lemma 1: For any $c_i^* \in [0, C]$ $(\forall\, i = 1,2, \dots, s)$ and $\sum_{i=1}^{s} c_i^* = C$ we have:

$$\mathbb{P}\left(\sum_{j=1}^{s} O_i \leq C\right) \geq \prod_{i=1}^{s} \mathbb{P}(O_i \leq c_i^*) \tag{6.13}$$

Proof: Lemma 1 can be proved recursively using Theorem 1:

$$\begin{aligned} \mathbb{P}\left(\sum_{i=1}^{s} O_i \leq C\right) &\geq \mathbb{P}\left(\sum_{i=1}^{s-1} O_i \leq \sum_{i=1}^{s-1} c_i^*\right) \mathbb{P}(O_s \leq c_s^*) \\ &\geq \mathbb{P}\left(\sum_{i=1}^{s-2} O_i \leq \sum_{i=1}^{s-1} c_j^*\right) \mathbb{P}(O_{s-1} \leq c_{s-1}^*) \mathbb{P}(O_s \leq c_s^*) \\ &\geq \cdots \geq \prod_{i=1}^{s} \mathbb{P}(O_i \leq c_i^*) \end{aligned} \tag{6.14}$$

From Lemma 1, we can deduct that when using a fixed amount of processing capacity, the hierarchical fog computing always has a higher chance to successfully schedule the workloads.

Corollary 1: From Theorem 1 and Lemma 1, we can generate our analytic comparison model of fog hierarchy between multi-tier and flat fog architectures. To this end, we partition the offloaded workloads $O_i, (i = 1, ,2, \dots, s)$ into K mutually groups $\{G_1, G_2, \dots, G_K\}$. Thus, for any $c_k^* \in [0, C](k = 1, 2, \dots, K)$ and $\sum_{k=1}^{K} c_k^* = C$, we have:

$$\mathbb{P}\left(\sum_{i=1}^{s} O_i \leq C\right) \geq \prod_{k=1}^{K} \mathbb{P}\left(\sum_{j \in G_k} O_k \leq c_k^*\right) \tag{6.15}$$

Equation (6.15) indicates that the efficiency of resource utilization in the 2-tier fog hierarchy will be maximized when there is only one tier-2 server. When $s = K$ in Eq. (15), the number of tier-1 and tier-2 servers are the same; as results, the flat and hierarchical designs of fog computing will be equivalent to each other. Clearly, we can also extend the results for hierarchical fog computing with more than 2-tier by repeating the above analysis recursively. For example, to construct a 3-tier fog hierarchy, we can simply use $\sum_{i=1}^{s} O_i$ as the workload of a tier-2 server.

6.2 Priority Queuing Model for Scheduling IIoT Data

As seen earlier, we consider a number of fog cells or data centers, where each data center contain servers deployed hierarchically. These servers handle the scheduling process and analysis data coming from different things installed inside the industrial factory. Incoming data, requests and information from different industrial things and sensors have different degrees of importance, e.g., emergency requests. The requests with high priority need to be scheduled and analyzed quickly.

We describe the queuing system with two priorities by making the following assumptions [103]:

Assumption 1(The input): the interval times of data of each class are mutually independent and identically distributed random variables. Thus, the input consists of a superposition of two independent renewal processes corresponding to these classes. However, it must be noted that the total input of IIoT data regardless of their class is not in general a renewal process.

Assumption 2 (The service mechanism): there are *s* servers in each fog tier. The services times of scheduling requests from IIoT are mutually independent random variables. their distribution being the same for requests in the same class, but they are different from that of other classes.

Assumption 3 (Queue discipline): the two classes are numbered 0 for low priority and 1 for high priority. Thus, a request of class 1 is always served prior to a customer of class 0. Within each class, the queue discipline is first come, first served (FIFO) with pre-emptive discipline.

In the pre-emptive case, the request of class 0 (low priority) returns to the head of the queue of his class and waits until the newly arrived request of class 1 are served.

First, we describe the queuing system with two priorities [103]. The workload parameters are defined as follows:

$$\lambda = \lambda_1 + \lambda_0 \tag{6.16}$$

$$\mu = \mu_1 = \mu_0 \tag{6.17}$$

$$\rho_1 = \frac{\lambda_1}{\mu}, \rho_0 = \frac{\lambda_0}{\mu} \text{ and } \rho = \rho_1 + \rho_0 \tag{6.18}$$

Where λ_1 and μ_1 are respectively the arrival rate and the service rate of demands with high priority. λ_0 and μ_0 are the arrival and the service rate of demands with low priority, respectively. Definitely, when arriving, IIoT requests are placed in different queues of each server; each of which has two services priority, high and low.

As a stochastic model, the queuing process theory is represented using Markov chains by incorporating information in the state description. We consider that each state in the Markov chain corresponds to the number of IIoT requests in the two queues, and the state transitions occur when a new IIoT requests arrives or an IIoT requests is served. We use the birth-death process as a stochastic model to describe the evolution of our system.

The model considers two cases: the first is when the number of waiting requests, denoted by r, in the two queues is $r < s$, where s is the number of servers in each fog tier; the overall completion rate is then $r\mu$. The second case is when the number of waiting requests is $r \geq s$; this means that the entire servers are occupied, and the completion rate is $s\mu$.

We can obtain the general stationary distribution state using recursive demonstration to distinguish two cases according whether μ_r depends on r or not, as follow:

- if $r < s$:

$$\pi_r = \pi_0 \prod_{i=0}^{r-1} \frac{\lambda_0 + \lambda_1}{(i+1)\mu} = \pi_0 \left(\frac{\lambda_0 + \lambda_1}{\mu}\right)^r \frac{1}{r!} \tag{6.19}$$

- if $r \geq s$:

$$\pi_r = \pi_0 \prod_{i=0}^{s-1} \frac{\lambda_0 + \lambda_1}{(i+1)\mu} \prod_{j=s}^{r-1} \frac{\lambda_0 + \lambda_1}{r\mu} = \pi_0 \left(\frac{\lambda_0 + \lambda_1}{\mu}\right)^r \frac{1}{s!\, s^{r-n}} \tag{6.20}$$

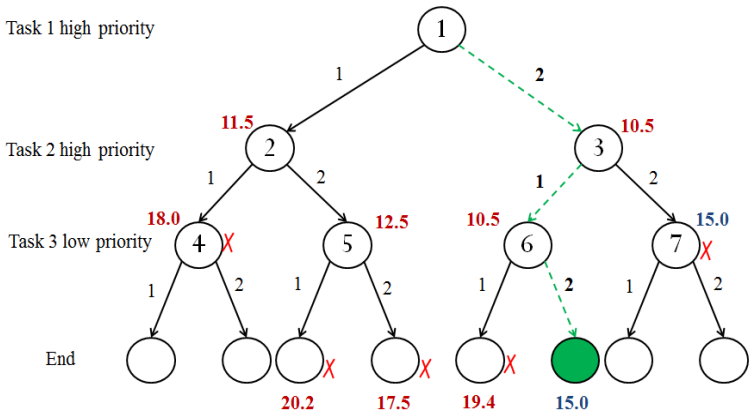


Figure 6.3- The branch and bound process based on BnB algorithm

The total probability condition is as follows:

$$\pi_0 = \left(\sum_{r=1}^{s-1} \frac{(\rho_H + \rho_L)^r}{r!} + \frac{(\rho_H + \rho_L)^s}{s!} \frac{1}{s!\, s^{r-n}} \right)^{-1} \tag{6.21}$$

Finally, from Eq. (19), (20) and (21) we define the mean waiting time (low $E(W_0)$) and (high $E(W_1)$) as follows [103]:

$$E(W_0) = \frac{\frac{\lambda_0}{{\mu_0}^2} + \frac{\lambda_1}{{\mu_1}^2}}{(1-\rho_1)(1-\rho_0-\rho_1)} \tag{6.22}$$

$$E(W_1) = \frac{\rho_1}{(1-\rho_1)\mu_1} \tag{6.23}$$

6.3 IIoT Data Assignment

In this section, we focus on determining which fog server's workloads are placed on to be scheduled. In addition, we define how much the computation capacity is provisioned to execute each request, so as to optimize the performance of executing all IIoT workloads and to take an optimal decision in a real time way to minimize the response delay.

6.3.1 Assignment Problem Formulation

We first study a scenario which considers only one server at each tier of the fog. Then, we aggregate the decisions of workload offloading at different tiers. We formulate the workload placement problem as a mixed nonlinear integer programming (MNIP) problem of fog tier which has only one server, considering integers variables to indicate the locations of things requests being placed and non-integer variables to indicate the capacity allocated to execute each request.

We assume that there are m computing tasks with amounts of computations $\{w_1, w, \dots, w_m\}$, the data size of program states $\{D_1, D_2, \dots, D_m\}$, and n servers at the fog layer with capacities $\{c_1, c_2, \dots, c_n\}$ (i.e., CPU cycles per second).

The MNIP problem is given as follows:

$$\min \mathbb{f} = \sum_{i=1}^{m} \left(\frac{D_i}{B_\psi}(\psi - 1) + \frac{w_i}{\theta_i^\psi . c_\psi} + E\left(W_{(0,1)}{}^\psi\right) \right) \tag{6.24}$$

$$\text{Subject to } \sum\nolimits_{i \in set_j} \theta_i^j = 1, j = 1, 2, \dots, n$$

Where ψ indicates the server where task i is placed on, θ_i^ψ is the percentage of server ψ's computational capacity being allocated for task i, and set_j denotes the set of computing tasks placed at server j. The rate $\frac{D_i}{B_\psi}(\psi - 1)$ measures the communication delay of transmitting task i states to server ψ, which is determined by the network bandwidth B_ψ allocated to ψ. The rate $\frac{w_i}{\theta_i^\psi . c_\psi}$ measures the computational delay of task i's execution. Finally, $E\left(W_{(0,1)}{}^\psi\right)$ measures the waiting time of task i in each ψ server's queue. (i.e., the waiting time is calculated in Equation (6.22) or Equation (6.23) according to the priority class of each task).

Theorem 2: To solve the MNIP problem in Equation (6.24), we first study the optimization problem when ψ is known to each task i. When each server ψ is fixed, the optimization problem in Eq. (24) becomes a convex optimization problem.

Proof: According to [104], function $\mathbb{f}$ in Equation (6.24) is strictly convex if its Hessian matrix H is positive definite for all the variables. So, we can prove that the Hessian matrix of function $\mathbb{f}$ is a positive definite matrix. To do this, we use θ_i to substitute θ_i^ψ, thus $\mathbb{f}$ in Equation (6.24) becomes a function of variables $\{\theta_1, \theta_2, \dots, \theta_i\}$. For each pair of (θ_i, θ_j), we have:

$$\frac{\partial^2 \mathbb{f}}{\partial \theta_i \partial \theta_j} = \begin{cases} \frac{1}{2} w_i c_\psi^{-1} \theta_i^{-3}, & if\ i = j \\ 0 & otherwise \end{cases} \tag{6.25}$$

According to [104], the Hessian matrix $\mathrm{H} = \left(\frac{\partial^2 \mathbb{f}}{\partial \theta_i \partial \theta_j}\right)_{s*s}$ of $\mathbb{f}$ is symmetric and positive definite. Therefore, since the constraint in Eq. (24) is linear, the optimization problem in Equation (6.24) is a linear convex optimization problem.

From Theorem 2, we further extend the MNIP problem as follows:

When each ψ is fixed, the optimal value of $\mathbb{f}$ is as follows:

$$\mathbb{f}_{min} = \sum_{j=1}^{n}\left[\sum_{i\epsilon set_j}\frac{(j-1)D_i}{B_\psi} + \frac{(\sum_{i\epsilon set_j}\sqrt{w_i})^2}{c_j} + \sum_{i\epsilon set_j} E\left(W_{(0,1)}{}^j\right)\right] \tag{6.26}$$

Hence, the corresponding optimal solution to the MNIP problem in Eq. (29) is:

$$\theta_i^* = \frac{\sqrt{w_i}}{\sum_{i\epsilon set_j}\sqrt{w_i}} \tag{6.27}$$

Corollary 2: when the placement for each demand is fixed, the corresponding capacity provisioning problem for scheduling these demands is deterministic and can become a combinatorial optimization problem using the transformation lemma. Hence, for each possible assignment, we can solve the problem from theorem 2 and Equation (6.26).

6.3.2 Branch and Bound Approach to Design Workload Assignment Algorithm

To determine the optimal value of $\{\theta_i\}$ and based on the above problem formulation (i.e., Equation (6.26)), we develop in this subsection IIoT workload and data assignment algorithm. As seen earlier, we have m computing tasks and n servers, so the time complexity to exhaustively search the solution space is $O(n^m)$. As we can see, the time complexity is very important. However, to further decrease the searching complexity and obtain the optimal solution space, we propose to use a branch and bound approach [105], as shown in Figure 6.3. In our work, the research space corresponds to the definition space of the optimal value of $\{\theta_i\}$. To this end, and without loss of generality, we assume that the IIoT workloads $\{w_1, w_2, \dots, w_m\}$ are successively determined to be assigned. The sub-problems of the workload assignment problem is denoted by w_y. Based on this sub-problem definition, we have the following lemma:

Lemma 2: Let $\mathbb{f}_y$ indicates the total delays for the y sub-problems having been branched, the lower bound on the total delays for the remaining $m-y$ tasks is:

$$\Upsilon_y = \mathbb{f}_y + \sum_{i=y+1}^{m}\left[\min_{1\le j\le n}\left\{\frac{w_i}{c_j} + (j-1)\frac{D_i}{B_j} + E\left(W_{(0,1)}{}^j\right)\right\}\right] \tag{6.28}$$

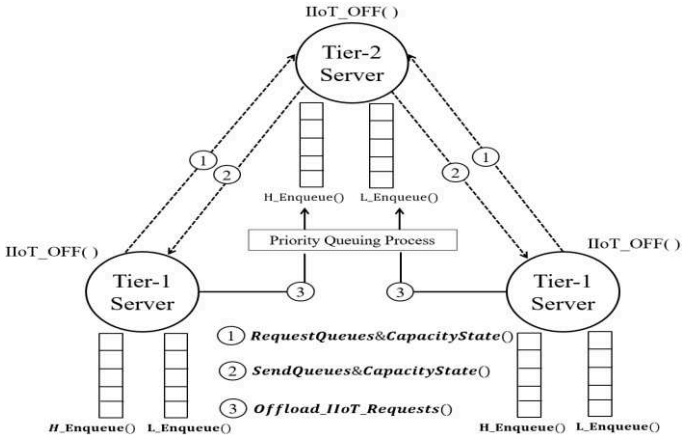


Figure 6.4- Implementation of IIoT_OFF for two tiers fog hierarchy

Proof: In Equation (6.28), all the remaining tasks ($m - y$ tasks) are provisioned with 100% capacity of a server, consequently Υ_y provides a lower bound on the total amount of delays for the remaining $m - y$ tasks.

Based on Lemma 2, the branching process is a depth-first search over subproblems placing subsets of workloads, with the ones of the smallest lower bounds being searched first. Each time when a leaf node in the search tree is reached, all other branches with larger values of lower bounds are pruned. The optimal solution will be found until all the leaf nodes have been either searched or pruned.

Example: In order to explain more the branching process using BnB algorithm, we further illustrate an example in Figure 6.3. To this end, we define the values of each variable as follows: the number of tasks $m = 3$, the number of servers $s = 2$, and the amounts of computations for each task is $\{{c_1}^H, {c_2}^H, {c_3}^L\} = \{2, 3, 4\}$ where tasks 1 and 2 are high priority tasks while tasks 3 is a low priority one. The capacity of each server is $\{c_1, c_2\} = \{1, 2\}$, and the communication delays between tier-1 and tier-2 servers (e.g., the rate $(j-1)\frac{w_i}{B_j}$) for the three tasks are $\{2, 3, 3\}$. Finally, the waiting times in high and low priority queues of the two servers are: $\{\left(\frac{1}{2}\right)^H, \left(\frac{3}{2}\right)^L\}$ for server number one, and $\{\left(\frac{1}{4}\right)^H, \left(\frac{3}{4}\right)^L\}$ for server number two.

The number indicated by the side of each node represents the lower bound of this sub-problem computed and the number on each edge indicates the decision of workload placement. For example, when task 1 is placed on tier-1, the lower bound of total delay on node 2 is $\{(\frac{2}{1}+\frac{1}{2}) + (\frac{3}{1}+\frac{1}{2}) + (\frac{4}{1}+\frac{3}{2}) = 11.5$, however, when it is placed on

tier-2, the lower bound on node 3 is $\{(\frac{2}{2}+2+\frac{1}{2})+(\frac{3}{1}+\frac{1}{2})+(\frac{4}{1}+\frac{3}{2})= 10.5$. As $10.5 < 11.5$, we further branch the next sub-problem until we reach a leave node being colored as green in Figure 6.3. Formerly, the leftmost and rightmost branches are automatically pruned because their lower bound is larger than 15.0. Therefore, the optimal solution of this example is $\{2, 1, 2\}$, with the total delay of 15.0.

6.3.3 Aggregation of Optimization Results Over High Tiers

In order to ensure a global optimality of IIoT requests assignment, we aggregate the solutions over different high-tier servers. The idea is to allocate the computational capacity of high-tier fog server on demand, when it receives offloading requests from multiple low-tier servers. Since there is one and only one optimal solution for a branch and bound searching process, the optimal allocation at a server that minimizes the cumulative delay of executing IIoT workloads is also unique.

The key challenge of aggregating the optimization results is how to allocate computational capacity of a higher-tier server to each branch of the fog hierarchy. To do this, let z denotes the capacity allocation vector of a higher-tier server to its children; a direct solution is a numerical iteration. As in [105], we adopt the Simulated Annealing Algorithm (SA) to find out the optimal allocation using numerical iterations. We describe SA in Algorithm 1. The idea is to generate a new iteration state z_{new} from the previous state z_{old}. T is the simulated temperature; at the beginning, T equals to T_{max} and decreases at each iteration by a cooling parameter $\xi (0 < \xi < 1)$.

Then, we calculate the difference of total delay, denoted as Δd, by solving the optimization problem in Equation (6.26). If $\Delta d < 0$, z_{new} is accepted; otherwise, we calculate the probability $p = e^{(-\frac{\Delta d}{T})}$ to accept z_{new}. The SA algorithm iterations terminate when $T = T_{min}$. The simulated annealing algorithm is given as follows:

Algorithm 1: Simulated Annealing Algorithm (SA)

Input: Initial solution z_{old};
Output: Optimal allocation solution;

1. Randomly generate an initial solution z_{old};
2. $T \leftarrow T_{max}$; //initialization of the simulated temperature
3. **while** $T > T_{min}$ **do**
4. Generate a new solution z_{new};
5. $\Delta d = \mathbb{f}(z_{new}) - \mathbb{f}(z_{old})$;
6. **if** $\Delta d < 0$ **then**
7. $z_{old} \leftarrow z_{new}$;
8. **end**
9. $prob = e^{(-\frac{\Delta d}{T})}$;
10. **if** $rand(0,1) < p$ **then**
11. $z_{old} \leftarrow z_{new}$;
12. **end**
13. $T \leftarrow T.\xi$; //cooling parameter $(0 < \xi < 1)$;
14. **end**

6.3.4 IIoT Offloading Algorithm Over k Tiers of Fog Hierarchy

Finally, we implement the offloading algorithm considering k tiers in the fog hierarchy. Where each tier servers have two workload queues (high priority queue and low priority queue). When a task arrives from different factory devises to the first tier level, the server invokes

functions $H\_Enqueue(IIoT\_Request)$ or $H\_Enqueue(IIoT\_Request)$ to put the workload into the appropriate queue according to its priority level.

If the queue is not empty and there is a spare computing capacity on the tier-k server, one task will be popped out from the queue and processed. Otherwise, if the tier-1 server is fully loaded, it will call $RequestQueues\&CapacityState()$ to check the availability of the $tier-(k+1)$ server. Only after having received the reply from the $tier-(k+1)$ server which invokes the $SendQueues\&CapacityState()$ function and confirmed that the $tier-(k+1)$ server has a spare computing capacity available and the optimal solution was founded, the $tier-(k+1)$ server will aggregate the optimal solution based on Algorithm 1 (i.e., SA Algorithm), and allocate the computational capacity of $tier-(k+1)$.

Algorithm 2: IIoT Offloading Algorithm (IIoT_Off)

Input: Number of tiers, Number of servers at each tier, $IIoT\_Requests(W, Priority)$;
Output: Optimal offloading solution, Optimal response delay;

```
1. Tier-1 servers receive IIoT_Requests(W, Priority);
2. k = 1;
3. if IIoT_Requests(W, Priority = 1) then
4.   for each server i of Tier-k do
5.     if (∑_{j=1}^{E_i(Q_1)} W_j ≤ C_i) then
6.       H_Enqueue(IIoT_Request);
7.     else
8.           Calculate the offloaded workloads as in Eq. (2);
9.       while ((∑_{i=1}^{S} O_i)_k ≥ (∑_{i=1}^{S} C_i)_{k+1}) then
10.        RequestQueues&CapacityState() from Tier-(k + 1);
11.        SendQueues&CapacityState() to Tier-k servers;
12.        Calculate optimal assignment using BnB;
13.      if (optimal solution founded in Tier-(k + 1)) then
14.        Aggregate the optimal solution using SA Algorithm;
15.        Allocate the computational capacity of Tier-(k + 1);
16.        Offload_IIoT_Requests();
17.        H_Enqueue(IIoT_Request) in Tier-(k + 1) servers;
18.      else
19.        k = k + 1;
20.      end
21.    end
22.  end
23. end
24. else IIoT_Requests(W, Priority = 0) low priority requests;
25.  We repeat the same instruction for low priority requests.
26. end
```

Finally, $tier - (k + 1)$ receives the offloaded tasks from the lower tier servers which the $Offload\_IIoT\_Requests()$ function invokes and the tasks will be queued in the appropriate queue according to its priority levels. However, if the optimal solution was not founded, the branch and bound research is executed for higher tiers ($k = k + 1$) until finding the optimal offloading solution with an optimal response delay for each task. The process of offloading tasks is presented in Algorithm 2 called IIoT_Off. Figure 6.4 illustrates an example execution, considering the implementation of two fog hierarchies with three servers, two of which are used as tier-1 servers and another is used as the tier-2 server.

6.4 System Performance Evaluation

In this section, we implement extensive simulations to evaluate the performance of the proposed fog architecture, and compare it (i.e., hierarchical architecture) with the flat fog architecture and relevant recent works [64], [66]. We measure the proposed architecture performance using realistic industrial data from Bosch group case study presented in [20].

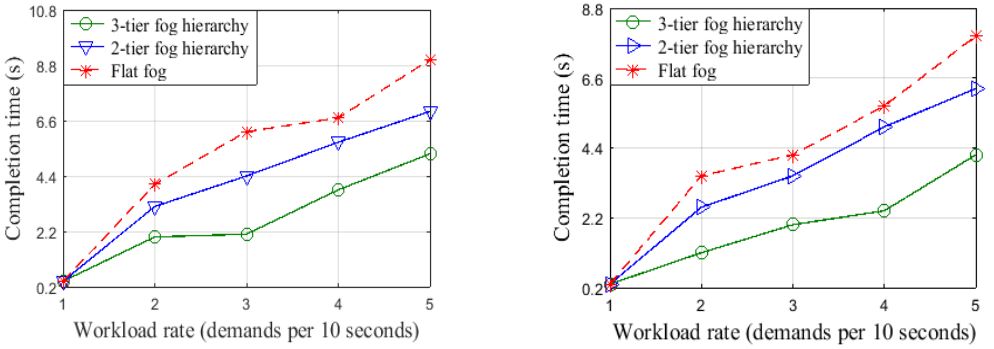
6.4.1 System Settings and Performance Metrics

We use multiple virtual machines VM as fog servers, and each VM has a CPU capacity of 1.45 GHz and 2GB RAM. We use NS-2 to implement our network topologies [36]. In our experiments, industrial IoT tasks and datasets are received by 4 tier-1 fog servers. Our experiments are conducted over different settings of the hierarchical fog with different topologies and computational capacity being provisioned. Every two fog servers are connected via a 100Mbps network link. A total amount of 20 GHz computational capacity (in terms of the amount of CPU cycles per second) is provisioned. Each VM of the network topologies is running MATLAB to execute all the proposed algorithms. We evaluate the performance of each topology using the following metrics:

- *Workload rate:* The number of tasks, which indicates the workload rate, is determined every time following the same Poisson distribution.
- *Provisioned capacity:* The computational capacity provisioned to a fog server is measured by the number of computing threads running concurrently on the server.
- *Waiting time:* It is the waiting time at each priority queue.
- *End-to-end delay:* This is the time it takes for the first response to come after processing of IIoT dataset.
- *Completion time:* It indicates the amount of computational capacity being provided by fog servers.
- *Synchronization Time:* The time interval between the time point that the IIoT devices get the readings and that of the fog servers receive these readings.

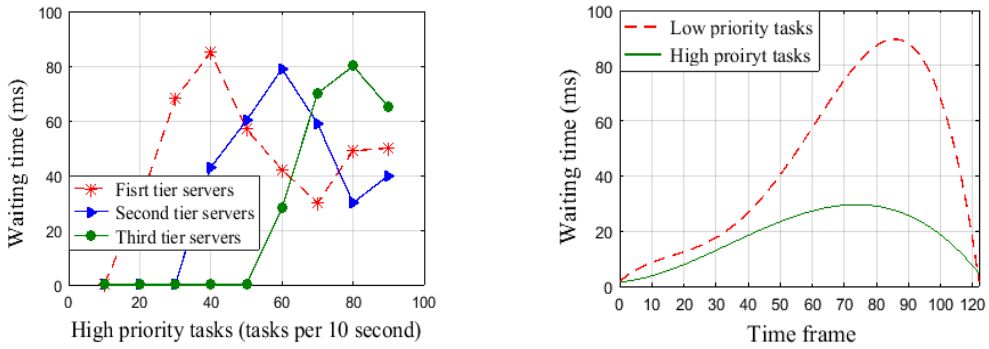
6.4.2 Experiment Results

In order to study the performance of the proposed fog architecture, we conduct simulations experiments over the three topologies of fog architecture, i.e., 3-tiers fog hierarchy, 2-tiers fog hierarchy and flat architecture. First, we evaluate the completion time performance of each topology by fixing the provisioned capacity and change the workload rate from 1 to 5 tasks per



(a) provisioned capacity = 2 (b) provisioned capacity = 4

Figure 6.5- Average completion time over different amounts of workloads

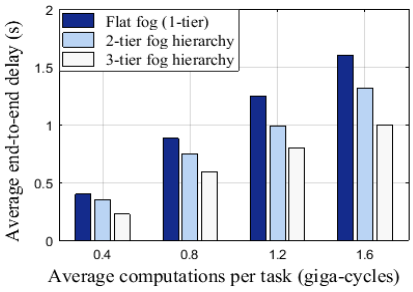


(a) Over dififfrent tiers hierarchy (b) Over dififfrent time frames

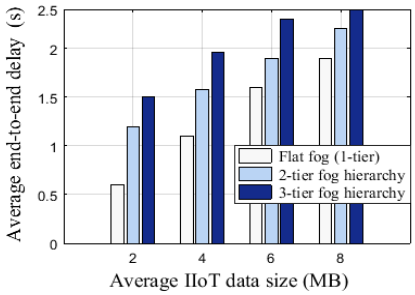
Figure 6.6- Average waiting time comparison

10 seconds. The results are shown in Figure 6.5. As we observe in Figure 6.5(a), when the provisioned capacity is 2, there is large difference between the performance of flat and hierarchical designs (e.g., 3-tier fog hierarchy and 2-tiers fog hierarchy); the 3-tiers hierarchy performs better and gives an optimal completion time. Rationally, when the provisioned capacity is 4, the completion time is more optimal as we can see in Figure 6.5(b). The major reason for this difference is that the tier-3 server is able to efficiently aggregate the peak loads from all tier-2 servers, and tier-2 servers are able to efficiently offload the IIoT peak loads from tier-1 servers. Hence, each hierarchical tier (e.g., 3-tiers and 2-tiers) utilizes the provisioned capacity to process the tasks at a much higher efficiency by executing the proposed IIoT Offloading Algorithm (IIoT_Off).

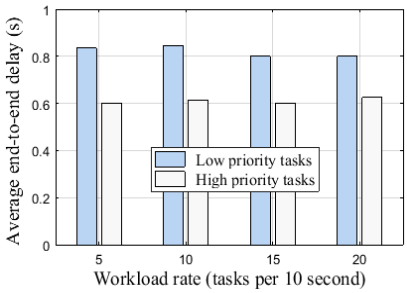
Now, we evaluate the proposed priority queuing mechanism implemented over each tier server. To do this, we examine the waiting time at each queue; the results are shown in Figure 6.6 over 3-tier hierarchy topology. Figure 6.6(a) shows the comparison of waiting time at high priority queue (i.e., dedicated for high priority tasks) over the three tier servers of the fog



(a) Over different computations tasks



(b) Over different IIoT data sizes



(c) For high and low priority tasks

Figure 6.7- Average end-to-end delay over different topologies

hierarchy. We observe that for each tier, the waiting time increases and decreases periodically; the average waiting time in tier 1 servers increases first and starts decreasing when the waiting time in tier 2 servers' increases. Similarly, when the waiting time in tier 3 server increases, the waiting time in tier 2 servers' decreases. This is explained by the offloading process (IIoT_Off algorithm) of high priority tasks over different tiers as follows: First, tier 1 servers receive the workloads from IIoT; formerly the workloads are queued in the queue, so the waiting time increases until $\sum_{j=1}^{E_i(Q_1)} w_j > c_i$ (i.e., the sum of queued workloads computation becomes superior to the capacity of each tier 1 server). As results, the next received workloads will be offloaded to tier 2 servers which leads to a decrease in the waiting time for tier 1 servers and an increase in the waiting time in tier 2 servers. In the same way, we can explain the increase/decrease relationship of waiting time between tier 2 servers and tier 3 servers.

Figure 6.6(b) shows the average waiting time comparison between high and low priority tasks over different time frames (e.g., 50% of high priority tasks and 50% of low priority tasks). In this experimentation, we suppose that the time frames from 60 to 90 contain a high amount of IIoT computation tasks where 80% of them are high priority tasks and 20% are low priority ones. We can see an increase in the waiting time for both high and low priority tasks from frames 60 to 90. However, the waiting time for high priority workloads is more stable than for

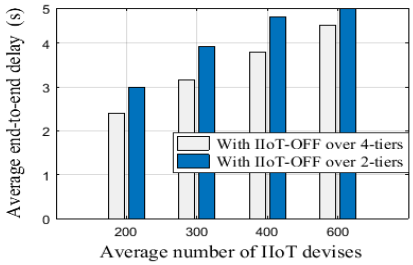
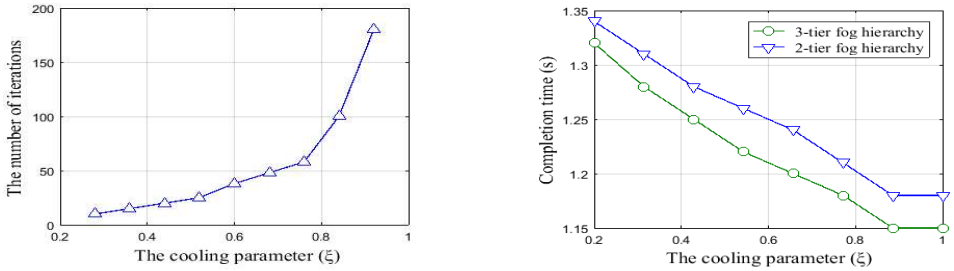


Figure 6.8- Average end-to-end delay over IIoT-OFF Algorithm for 4-tiers and 2-tiers



(a) The number of iterations (b) The completion time

Figure 6.9- The effect of the cooling parameter

low priority workloads, and the difference is very big between frame 60 and 90 in Figure 6.6(b). This is because there are around 80% of high priority tasks in this frames interval (i.e., 60 and 90), and the proposed queue discipline is pre-emptive (the low priority tasks returns to the head of the queue of its class and

waits until the newly arrived high priority tasks is served). As results, from Figure 6.6 we can conclude that the proposed priority queuing mechanism helps the IIoT_Off algorithm to balance the high workloads over all tier servers in order to minimize the waiting time at each queue.

Now, we study why the end-to-end delay is an important parameter to consider in Industry 4.0 and smart factories based IoT. We propose a comparison study between each tier topology over various computation tasks (e.g., Figure 6.7(a)), different IIoT data size (e.g., Figure 6.7(b)), and several priority levels of tasks (e.g., Figure 6.7(c)). As shown in Figure 6.7(a), the 3-tier fog architecture can reduce the end-to-end delay by about 40% compared to that of the flat architecture. In addition, the 2-tier hierarchy reduction is around 20%, this is due to its incapability of aggregating enough peak loads from lower tiers. Indeed, the 3-tier fog hierarchy provides more computational capacity on the higher tiers servers and less capacity on the first tier, leading to a more iterative aggregation of devices demands. In Figure 6.7(b), we evaluate the end-to-end delay of the proposed architecture over different IIoT data sets sizes.

As known, larger data sizes increase the communication delay for transmitting workloads to high tiers; however, our proposed scheduling scheme can still optimize the offloading of industrial things requests by reducing the total delay by 30% with different data sizes. Mainly, the 3-tier fog hierarchy can also outperform other topologies, because of its efficient iterative aggregations of the peak loads using IIoT_Off algorithm. Figure 6.7(c) shows the average end-to-end delay for high and low priority tasks. As explained earlier, the end-to-end delay for high priority requests is better than that of low priority requests; this is because the waiting in high priority queues is smaller compared to that in low priority ones (see Figure 6.6(b)).

Figure 6.8 evaluates the performance of the offloading algorithm over 2-tiers and 4-tiers hierarchy, where $k = 2$ and $k = 4$ in IIoT_Off algorithm. We increase the number of IIoT devices from 200 to 600 where each device sends the same amount of data size (2 MB). As shown in the figure, with the increase in the number of interconnected industrial devices, the end-to-end delay increases for the two topologies, however it is smaller when we offload workloads over 4-tiers than over 2-tiers. This is mainly because the 4-tier fog hierarchy provides more computational capacity on the higher tiers servers and less capacity on the first and second tiers.

Finally, we study the impact of the SA algorithm cooling parameter (ξ) on the performance and time complexity in Figure 6.9. The experiment results are shown in Figure 6.9(a). Since a larger value of ξ helps the SA algorithm process to converge, it further reduces the total delay of task completion time. On the other hand, when the value of ξ is too small (< 0.6), the iterations may not be enough for the SA algorithm to converge, incurring higher delay of task completion. Figure 6.9(b) shows that the 3-tier fog has the lowest delay compared to other 2-tier fog under different cooling parameters values, even though the result has not reached the convergent value. The reason is that although the cooling process accelerates, the 3-tier fog hierarchy can still aggregate more peak loads from lower tiers compared to the 2-tier hierarchy.

6.4.3 Comparison with exiting works

Now, we compare the proposed hierarchical fog computing and the scheduling approach of IIoT requests with two recent works (i.e., [64] and [66]), based on three principal metrics (i.e., synchronization time, accuracy, computing time, and end-to-end delay). First, we consider the same simulation parameters used in each work. Then, we adjust our simulation scenario with the same parameters and we run simulations.

Figure 6.10 illustrates the performance of the proposed hierarchical fog architecture compared with two schemes (i.e., fog-cloud scheduling scheme and cloud scheme) proposed in

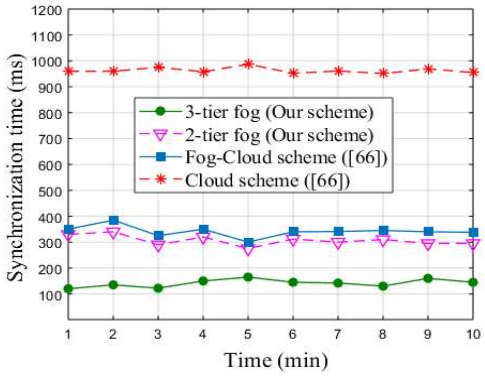
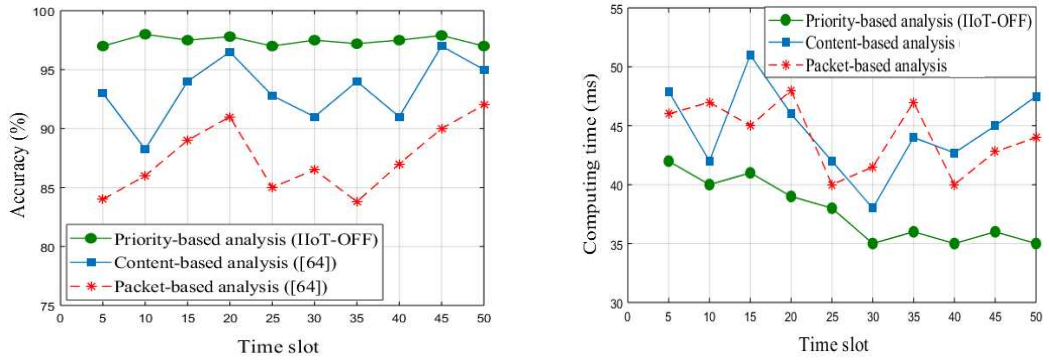


Figure 6.10- Synchronization time comparison



(a) Accuracy analysis (b) Computing time analysis

Figure 6.11- Comparison of the perfromences of the proposed scheme

[66] (see related work section). As in [66], in our proposed model, there is an interesting tradeoff between the data preprocessing time and data transmission time (due to the hierarchical deployment of the servers). We employ the synchronization time to measure the time efficiency of our model and compare it with the two schemes in [66] (i.e., traditional scheme which uses only cloud computing, and the framework that uses edge servers and cloud). We can observe that our two schemes (3-tier fog and 2-tier fog) are much more time-efficient than the traditional scheme (only cloud [66]) and the edge framework scheme [66]. This is reasonable considering that multiple links between the fog servers' hierarchy are built in our architecture, while only one link is built in [66] scheme which can provide an important overhead. Our proposed model ensures both offloading IIoT requests based on priority queuing discipline and optimal capacity allocation and aggregation over different fog tiers; this make it consumes very little time compared to [66].

Finally, we compare in Figure 6.11 the performance of our proposed scheduling scheme (i.e., IIoT_Off algorithm) with a recent work [64] (see related work section). In [64], the authors design an efficient and simple content-aware data processing rule, in order to define how to

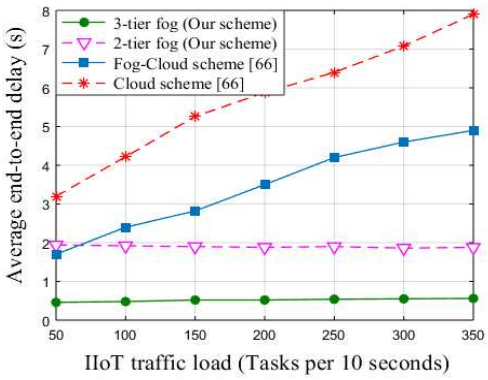


Figure 6.12- Average end-to-end delay comparison with existing work [66]

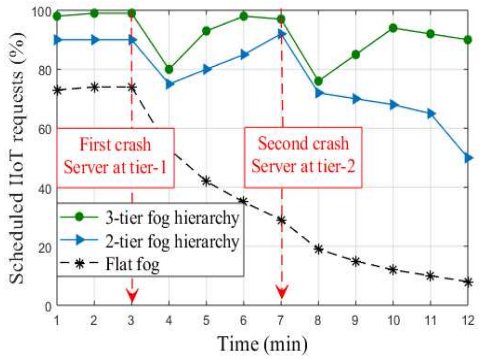


Figure 6.13- Performances of our scheme when servers' crashes at different tiers of the hierarchy

design an optimal computing manner for a general IIoT (i.e., decentralized computing or centralized computing). We first examine the correctness of the proposed priority-based analysis method compared with the content-aware analysis and packet-based analysis methods (used as the benchmark). The comparison results are shown in Fig. 11(a) from the aspects of analysis accuracy, and in Figure 6.11(b) from the aspect of computing time.

As we can see in Figure 6.11(a), the priority-based analysis using IIoT_Off algorithm can achieve at least 6% on the analysis accuracy compared to content-aware method and more than 15% compared to the conventional one. The priority curve is remarkably stable during all time slots; this is due to the optimal offloading of IIoT requests over different fog tiers using the proposed queuing system at each server. Similarly, the computing time when using the proposed scheduling scheme is more optimal, reduced to 20 (ms) compared with the content analysis scheme, and to 30 (ms) compared with the packet analysis scheme. We observe also that the computing time curve of priority analysis is stable, and is decreased over highest time slots. We can explain this by the optimal resource allocation and workloads assignment over different fog tiers using the brand and branch method to solve the assignment problem in Equation (6.26).

Figure 6.12 illustrates the performance of the proposed hierarchical fog architecture with two topologies (i.e., 3-tier hierarchy and 2-tier hierarchy) compared with two schemes (i.e., fog-cloud scheduling scheme and cloud scheme) proposed in [66] (see related work section). We aim to examine the average end-to-end delay over different amounts of IIoT traffic loads (we increase the number of tasks at each 10 seconds). As we can see in Figure 6.12, the two topologies of our proposed scheme (i.e., 3-tier, and 2-tier) outperforms the other schemes where the average end-to-end delay is stable between 0.5 and 0.7 second for 3-tier topology, and between 1.8 and 2.0 second for 2-tier topology. However, the average end-to-end delay increases significantly for fog-cloud scheme (it can reach 5 seconds) and cloud scheme (it can reach 8 seconds) when we increase the number of tasks. it is well-known that the end-to-end delay includes the computing time, the transmission time of each requests, and the synchronization time (in the case of offloading of requests). As we have seen earlier, our proposed scheme proposes an appropriate mechanism to allocate the computation capacities for each server located at different tiers of the fog hierarchy; however, the work in [66] did not propose a mechanism to reduce the computing time. On the other hand, as we compared in Figure 6.10, our two schemes are much more time-efficient (in terms of synchronization time) than the traditional scheme (only cloud [66]) and the edge framework scheme [66]. This is reasonable considering that multiple links between the fog servers' hierarchy are built in our architecture, while only one link is built in [66] scheme which can provide an important overhead, which may increase significantly the end-to-end delay.

Given these results from the comparison with relevant existing works, we find that the proposed scheduling scheme based on hierarchical fog servers' deployment and priority queuing outperforms the works [64] and [66] in terms of synchronization time (i.e., data transmission over different servers), computing time, processing accuracy, and average end-to-end delay.

6.4.4 Performances when Servers' Crashes at Different Tiers of the Hierarchy:

Finally, we examine the performance of our scheme when one of servers at different tiers of the hierarchy crashes. For this end, we study the percentage of scheduled IIoT requests during a simulation time of 12 minutes. We implement the three topologies (3-tier, 2-tier, and flat fog), and we generate the same load of IIoT traffic for each topology. Indeed, we simulate two crashes of servers; the first is caused at a server at tier-1 of each topology at 3 min, and the second is at a server located at tier-2 of each topology. The results are shown in Fig. 13; we investigate the results as follows:

- Before the first crash (from 1 to 3 min): we observe that the percentage of scheduled requests is stable for all topologies, and the 3-tier fog hierarchy topology outperforms the author topologies where it can reach 99% of scheduled requests, 91% for 2-tier hierarchy, and 72% for flat topology.
- After the first crash (from 3 to 7 min): we observe that the percentage decreases for the tier topologies from 3 to 4 min (i.e., to 80% for 3-tier, and to 70% for 2-tier). After 4 min, we observe that the percentage increases for the two topologies quickly to reach 98% for 3-tier and 90% for 2-tier. To loss of these requests can be explained by the crash of the server, where the requests scheduled in this server are not handled. However, we see that the problem is rapidly solved by offloading the incoming request to other servers of the hierarchy using IIoT_Off algorithm, and reallocating the computation capacities according to the new situation of the fog hierarchy (the proposed algorithms can be adapted rapidly to any situation change in the architecture by modifying the initial parameters as the number of servers at each tier). However, for flat architecture, we can see the percentage still increase until 35% at 7 min, because we have only one tier (we can offload the requests to other tiers).
- After the second crash (from 7 to 12 min): we observe that the percentage of the scheduled IIoT requests for 3-tier topology decreases to reach 70% at 8 min, and it increases to be stable between 90% and 95%. However, for 2-tier topology, the percentage decreases significantly to reach 50% at 12 min. This is because we have only one sever at tier-2 for 2-tier topology, and by crashing the server at tier-2, the topology becomes flat where we cannot offload the IIoT requests, also, we cannot aggregate the optimal results, this is way the percentage did not increase. Nevertheless, for 3-tier hierarchy, even a server is crashed at tier-2, we still have a hierarchical topology with three tiers. Hence, we can adapt and execute our algorithms according to this situation.

6.5 Chapter Summary

In this Chapter, we propose an efficient fog architecture for industrial internet of things applications, and introduce a new scheduling model for IIoT data processing. The proposed architecture is based on multi-tier servers' deployment at the fog layer. To efficiently schedule different industrial devices requests in a real time way, we developed an optimal workload assignment algorithm named IIoT-OFF by solving a mixed nonlinear integer programming (MNIP). Then, the optimal solution is aggregated over different tiers using a new Simulated Annealing Algorithm (SA). The advantages of the proposed hierarchical fog architecture

compared to the conventional industrial system and two relevant recent works are proved over different performance metrics and through extensive simulations using realistic industrial data from Bosch group.

CHAPTER 7

5G SLICING BASED ON SDN HIERARCHICAL VEHICULAR FOG

Several novel and promising wireless vehicular network applications have been developed lately. Examples include trip planning, media sharing and Internet access. However, sufficient network resources (e.g., equipped vehicles, base stations (BS), Road Side Units (RSUs) and other infrastructure) to support these applications are not yet available or are deployed at a slow pace. Moreover, these applications require considerable efforts in information gathering and data processing to allow quick decision-making and fast feedback to all users. This imposes significant challenges on the development of an efficient wireless vehicular communication platform.

On the other hand, 5G networks are anticipated to support a plethora of innovative and promising network services. These services have heterogeneous performance requirements (e.g., high-rate traffic, low latency and high reliability). To meet them, 5G networks are entailed to endorse flexibility that can be fulfilled through the deployment of new emerging technologies, mainly Software-Defined Networking (SDN), Network Functions Virtualization (NFV) and Network Slicing.

In this chapter, we propose two vehicular architectures a follows:

1) A hierarchical SDN-based wireless vehicular fog architecture, called HSVF (i.e., Hierarchical SDN for Vehicular Fog architecture). HSVF is based on a hybrid SDN control plane that reinforces centralized and distributed management. The proposed SDN control plane includes a trajectory prediction module to mitigate the drawbacks of the frequent handover problem between RSUs and vehicles. The proposed architecture is then evaluated using a relevant case study that addresses the scheduling of electric vehicles (EVs) energy demands.

2) A distributed and scalable SDN core network architecture that support 5G slicing functions that maps autonomous driving functionalities into service slices, the proposed architecture is called SliceScal. SliceScal deploys fog, edge and cloud computing technologies and propose a network and service slicing system model that promotes a four-layer logical architecture to improve the transmission efficiency and satisfy the low latency constraint.

7.1 HSVF: Hierarchical SDN for Vehicular Fog Architecture

SDN along with decentralized cloud architectures (i.e., Fog computing) have recently been explored to handle specific tasks and issues related to vehicular networks. By coupling these technologies with the concept of network virtualization, we propose HSVF as illustrated in Figure 7.1. From the bottom up, HSVF is divided into three planes: data plane, control plane, and application and service plane.

Data plane (RSUs, BSs, and Vehicles): The data plane includes all data transmission network devices. Functions of the data plane are focused on data collection, quantization, and data forwarding to the control plane. In HSVF, the data plane is constructed using a hierarchical overlay fog network to eliminate the heterogeneity of existing vehicular networks. All devices are abstracted as SDN switches and equipped Open vSwitch. We assume that SDN devices support virtualization and can host virtual machines (VMs) to be able to install software updates. These devices can communicate with micro-data centers, located in the hierarchical fog, to deliver different services.

Communication module (Infrastructures module of vehicles): The communication module proposed in HSVF includes V2V, V2I and I2I communication submodules. The V2V communication submodule provides wireless communication among adjacent vehicles using the WAVE standard (i.e., IEEE 802.11p). V2I communication consists of two communication types: V2RSUs based on IEEE 802.11p and V2BSs, where base stations transmit wireless signals using DSRC frequencies to provide broad coverage for vehicles. BSs also have LTE connections to allow RSUs to forward their connectivity information to their respective SDN-F controller. Note that according to the Federal Communication Commissions (FCC) amendment regarding DSRC, only four 10 MHz channels, called service channels (SCHs), are used for service data communications. Note also that WAVE supports multi-hop communication among vehicles. Since all vehicles move in an orderly fashion, a group of vehicles can be assumed to be a communication unit in a fog cell. When one vehicle within this group connects to one RSU, the whole group can be considered as connected to that RSU. This way, frequent handovers can be avoided.

Control plane of SDN-F: SDN-F are deployed in specific servers of each fog data center. Some SDN-Fs are directly connected to RSUs, BSs, and vehicles in their areas while others are only connected to SDN-Fs, making a decentralized. Hierarchical fog-based architecture. All

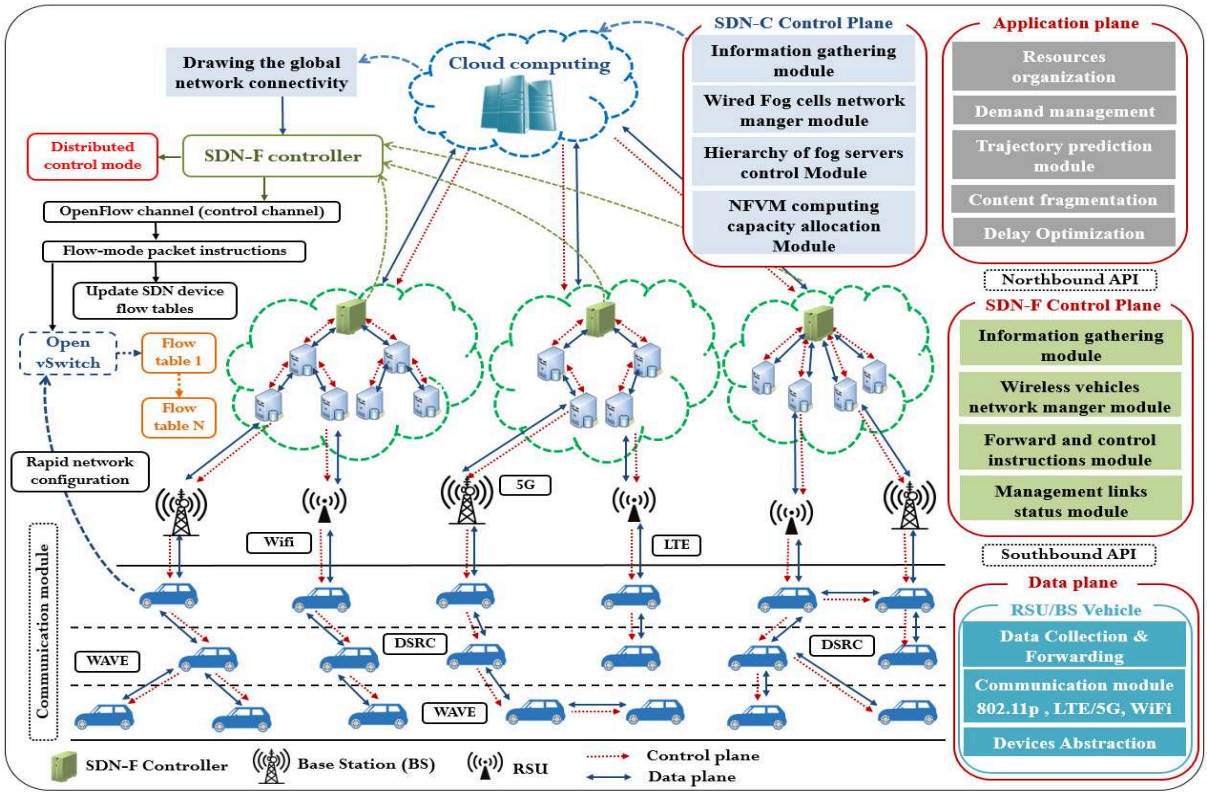


Figure 7.1- HSVF system architecture. Hybrid SDN controls management planes and hierarchical fog cell servers' deployment.

fog data centers are connected to a centralized cloud data center equipped with an SDN-Cloud (SDN-C) controller that manages the fog networks. The control plane of each SDN-F controller maintains the status of all the switches in its area and is responsible for making packet forwarding instructions based on the information received from SDN-C. Considering the high mobility and the massive traffic between RSUs and vehicles, frequent handovers may occur. To address this problem, HSVF deploys a distributed control plane over different fog cells, each of which is composed of vehicles, RSUs and LTE base stations.

To support the functions of the control plane, SDN-F has four modules:

- ***Information gathering module:*** collects information from vehicles, RSUs and BSs as well as servers within the fog cell. This information is used to construct the vehicles' connectivity network model and to allocate frequency bands. In addition, this module generates control instructions based on the SDN-C directives and forwards them to the forward and control instructions module.
- ***Wireless vehicles network manager module:*** ensures the management of different wireless networks (i.e., WAVE, LTE and Wi-Fi).

- ***Forward and control instructions module:*** based on the data received from the information gathering module, the forward and control instructions module generates control directives and forwards them to all devices in the fog cell.
- ***Management links status module:*** manages the links status communications (V2V, V2I, and I2I. It is responsible for maintaining service continuity while considering network topological changes.

Control plane of SDN-C: SDN-C is responsible for constructing the global network connectivity by aggregating information received from all SDN-Fs and generating control information based on rules and strategies from the application plane. SDN-C has four modules:

- ***Information gathering module:*** collects topological information from each SDN-F to maintain the network's global topology and controls cooperation within each fog cell (e.g., data offloading, vehicles mobility management and data migration).
- ***Wired fog cells network manager module:*** manages the wired network connecting fog cells data centers (i.e., servers). It is responsible for routing the data and analyzing the network topology of each fog data center.
- ***Hierarchy of fog servers control module:*** controls the hierarchy deployment of fog servers and draws the global information of all fog cells based on the data information received from SDN-Fs data plane; it then generates the control information of offloading demands over different tiers of the hierarchy.
- ***NFVM computing capacity allocation module:*** uses NFVM to allocate sufficient computational resources for each server in the fog tiers. The NFVM creates and configures different VMs according to the quantity of data offloaded from each server. This module provides good computation opportunities for scheduling different requests in a timely manner.

Communication interfaces: The control and data planes in HSVF communicate with each other via an integrated interface (i.e., southbound application programming interface, South-API), which includes some predefined control and statement messages. The northbound API, on the other hand, enables application and control planes to communicate. It provides applications with a high-level abstraction to communicate with SDN-Fs and SDN-C.

The southbound API is implemented by extending OpenFlow 1.2 to OpenFlow 1.3 to support multiple controllers. With OpenFlow 1.2, only fault management is targeted by a master/slave scheme. With OpenFlow 1.3, auxiliary connections can be used to connect the master controller

and the switches, allowing for a better load balancing. Moreover, per-connection event filtering is introduced in OpenFlow 1.3. This enables controllers to only subscribe to types of message in which they are interested. For example, a controller responsible for collecting statistics about the network can be attached as an auxiliary controller and will subscribe only to statistics events that are generated by the switches.

Application plane: Based on application requirements, rules and strategies of HSVF are generated by SDN-C for the SDN-Fs and by SDN-Fs for wireless vehicular network. The application plane contains a set of applications and services like resource organization, demands management, delay optimization, trajectory prediction module. The application plane interacts with each SDN-F to forward required information to the control plane. Additionally, in our proposed architecture, we consider content fragmentation module as a service that is provided at the application plane. It requires the help of the trajectory prediction module to enhance data dissemination of many infotainment applications (e.g., media streaming) in VANET.

HSVF uses a stable clustering technique based on the proposed deployment of Fog cells. This technique maintains the data transmission and control instructions to be used when the central SDN-C controller is not accessible. In addition, by distributing the control over different fog cells, HSVF aims to reduce network congestion and message control overhead. Indeed, SDN-Fs are the key component of HSVF. Each fog cell can have one, two or three layers' hierarchy of servers, as illustrated in Figure 1. These servers report periodically the network state (e.g., overhead, flow table changes) and the state of the scheduling requests (e.g., the quantity of offloading workloads, the capacity allocated) to their respective SDN-F. This later aggregates the information and generates the control instructions based on the SDN-C control directives.

7.2 Case Study: Scheduling of Electric Vehicles Energy Demands

EV fleets are expected to grow considerably as the EV technology is being adopted for public transportation in many countries to reduce CO2 emissions. With the development of high energy and power density batteries, EV market penetration is expected to flourish. Therefore, in this case study, we focus on scheduling EV charging and discharging demands using HSVF.

As EVs become more popular, numerous charging/discharging public stations (EVPSS) are made available at many public areas. However, this situation raises multiple concerns. One of the most challenging issues is how to adequately manage all EV demands considering EVPSS states in different grid situations. To address this problem, several questions are to be answered,

including: how difficult will it be for an EV driver to find a public charging station when away from home? Can the charging demands be scheduled in real-time? Can the communication architecture support the mobility of EVs by collecting information, maintaining vehicles connectivity and ensuring reliability? Designing an optimal scheduling of charging-discharging for EVs using an adequate communication architecture is the answer to all these questions.

7.2.1 Planning of EVs Demands Using Calendars Policy

We adopt the main idea proposed in our previous work (i.e., [22], [36]) where EV users may create several calendars to express several charging-discharging requests. While the scheduling of calendars in [22] is done using a centralized cloud computing architecture without the use of SDN, [36] proposes a decentralized scheduling using Fog architecture with a single SDN controller that manages only the wired network (i.e., the network that connects servers of each Fog). Unlike [22] and [36], HSVF proposes a novel hierarchical control plane for wireless vehicular network. In order to organize the demands of users based on the storage capacity of Fog data centers, users may create numerous calendars to express several charging-discharging requests.

The calendars are scheduled by the appropriate Fog data center based on vehicles location. The optimal calendar is then selected for each vehicle considering parameters like waiting time to plug-in, price at time t, distance to EVPSS and demand-supply energy curve.

7.2.2 Provisioning Computational Capacity in HSVF

The main goals of the case study are: 1) scheduling all EV demands in real time; 2) reducing the response time of scheduling all calendars; and 3) reducing the transmission delay of data (i.e., calendars, EVPSS state, traffic statistics) from vehicles-to-servers and from servers-to-vehicles. We implemented the SDN-F controller in a tier-3 server in order to centralize the control of Fog cell servers in one server. Also, all Fog servers (tier-1, tier-2, and tier-3) in HSVF are in charge of scheduling EV calendars using VMs. To this end, as shown in Figures 2, the three-tier hierarchical Fog-SDN architecture of one cell consists of S servers in tier-1, $S-1$ servers in tier-2, and one server in tier-3. This latter represents the SDN-F controller. C_i is the computational capacity while W_i is the amount of received EVs calendars of tier-i servers.

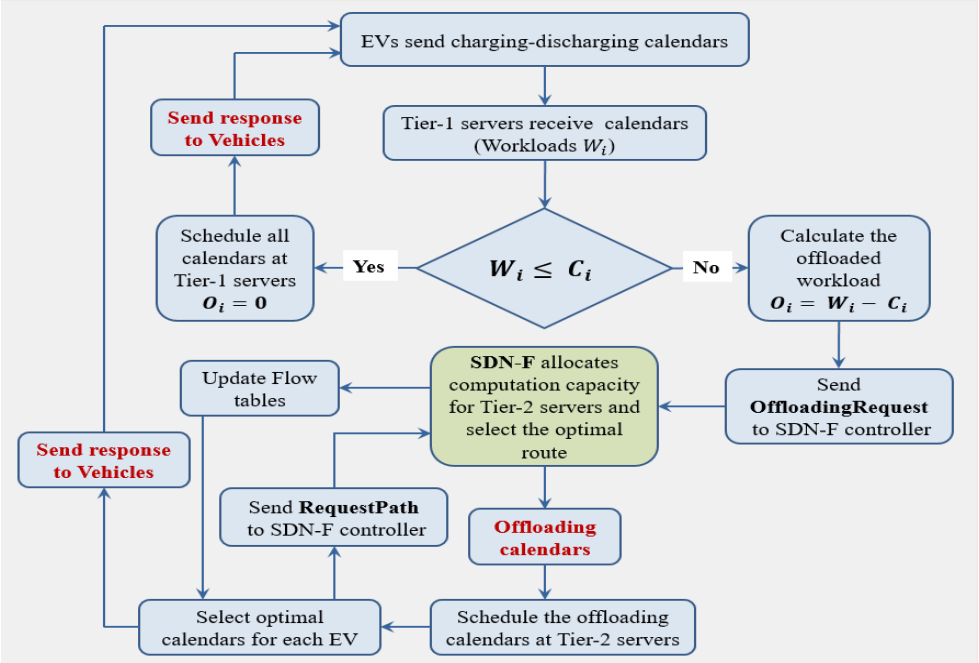


Figure 7.2- Offloading process diagram based on SDN-F controller

7.2.3 Optimal offloading process based on SDN-F controller

As shown in Figure 7.2, the offloading process is related to the capacity of tier-1 servers to handle EV requests. In case they cannot, some tasks will be offloaded to the higher tiers of the hierarchy. Figure 7.2 depicts the offloading process along with the synchronization between scheduling EV requests (selecting the optimal calendars) and the SDN-F controller (i.e., allocating the computing capacity for the offloading and updating flow tables).

7.3 SliceScal: 5G-Slicing-Enabled Distributed Scalable SDN Architecture

7.3.1 Features of 5G-Enabled SDN Architecture

Reliable and ultra-low latency of autonomous driving service functions and requests over wireless networks alone is not sufficient to guarantee the required end-to-end reliability. The core network has also to be considered for service slice management to support these requirements. Indeed, both autonomous driving and best-effort traffics will have to travel through a complex network while competing for resources during transmission, buffering, and computing. In order meet the desired requirements of autonomous driving systems, traffic generated by autonomous vehicles has to be properly marked, identified, and prioritized at all the network elements (both wireless and core network).

In this chapter, we propose a new vehicular core network based on a distributed and scalable SDN architecture called SliceScal. SliceScal uses OpenFlow, which is a communication protocol between the control plane and the network infrastructure. The SDN controllers can modify the forwarding rules for each flow that are stored in the form of flow tables. This is to maintain the priority of the critical flows with respect to other flows (e.g., best-effort traffic). In Addition, a central cloud orchestrator generates a complete isolation between different flows and service slices, and applies the necessary forwarding rules to ensure that the QoS of autonomous driving is maintained throughout the network. In a network with multiple flows requiring different levels of QoS, a distributed SDN can allow dynamical control of their Key Performance Indicators (KPIs) based on the effective demands generated by each of these flows.

7.3.2 Core Network Design

The 5G core vehicular network architecture defined in this work considers a distributed scalable SDN, it uses a stable clustering technique based on the deployment of fog cells. This technique maintains the data transmission and control instructions to be used when the central cloud orchestrator is not accessible. In addition, by distributing the control over different fog cells, we can reduce network congestion and message-control overhead.

We use 3GPP's 5G system for our vehicular network [106]; however, we introduce it as part of the global architecture for NFV virtualization. In particular, all vehicular network functions are considered to be slices that can run over standard hardware. The slices may be moved across different locations subject to the requirements of the service providers. Using this architecture, our goal is to describe the role of softwarization and service slicing from the autonomousdrving perspective. This includes software-defined traffic routing decisions in the 5G radio access and defining SDN controllers' duties in the core network. In general, service slicing rules have to be enforced in the physical infrastructure to satisfy a satisfactory level of QoS for autonomous driving systems, starting from the selection of most suitable radio access nodes and up to managing traffic and service requests in the transport vehicular network.

We believe that autonomous drivingthe use of 5G slicing based on a distributed and scalable SDN core network is crucial for autonomous driving function services to ensure appropriate traffic isolation while guaranteeing ultra-low latency and reliability. Figure 7.3 illustrates the SliceScal core network. From bottom up, our architecture includes one data access plane and three control levels: fog control plane, edge control plane, and cloud control plane. The

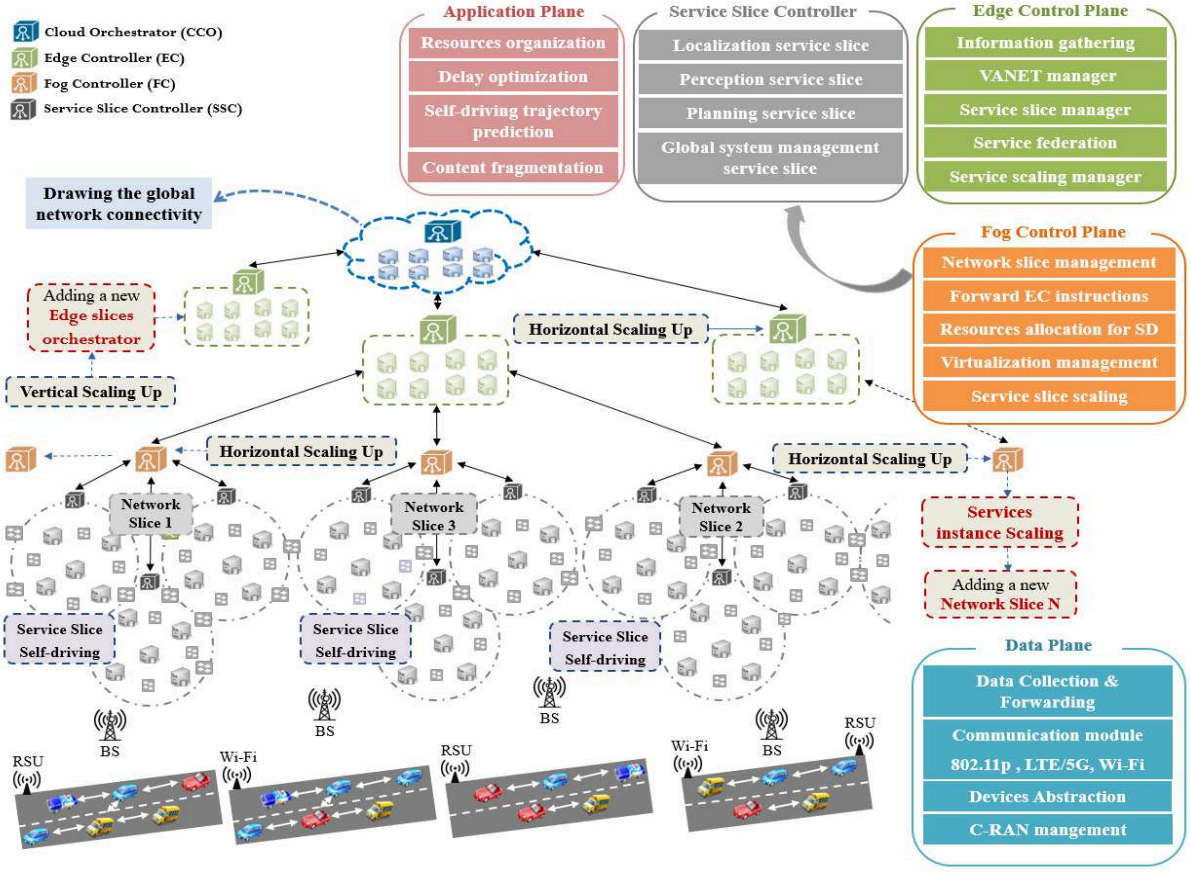


Figure 7.3- 5G-Slicing-Enabled Scalable SDN Core Network architecture (i.e., SliceScal)

southbound API between the control plane and data plane is implemented by extending OpenFlow while the northbound API is provided to the applications as a high-level abstraction to communicate with the different controllers.

Data plane (RSUs, 5G BSs, and Vehicles): The data plane includes all data transmission network devices. The main functions of the data plane are data collection, data quantization and data forwarding to the control plane through the southbound API. The data plane supports Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) and Infrastructure-to-Infrastructure (I2I) communications using different networking technologies (e.g., DSRC, 5G, Wifi, and IEEE 802.11p). The data plane is built as a hierarchical overlay fog network to eliminate the heterogeneity of existing vehicular networks. All vehicles, Road Side Units (RSUs), and 5G Base Stations (BSs) are abstracted as SDN devices and equipped with OpenFlow software switches. These software switches can be further categorized into the mobile data plane (i.e., vehicles) and stationary data plane (RSUs, 5G BSs). RSUs, BSs, and other edge equipment can communicate with micro-data centers located in the fog to deliver different services. We

assume that SDN devices are capable of virtualization and can host virtual machines (VMs) that are able to install software updates on the SDN devices.

Control planes (FC, EC, CC): SliceScal implements three control levels: edge control plane, fog control plane, and cloud control plane. The edge control plane maintains the status of all the OpenFlow software switches and is responsible for making packet forwarding instructions based on the OpenFlow control channel. The service slice management and policy enforcement functions for vehicular network are carried out by the Edge Controller (EC), which is also in charge of traffic routing and service slice scaling to route data to its intended destination. The Fog control plane is deployed to address the problem of frequent handovers that may occur considering the high mobility and the massive traffic between RSUs, BSs and vehicles . In this regard, we deploy a distributed control plane over different fog cells. Fog controllers (FC) are located in some of the servers of each fog cell. Each FC is connected with RSUs, BSs and vehicles via other OpenFlow switches and servers, which form a distributed fog based hierarchical architecture. To guarantee an appropriate planning of service slices and an adequate forwarding rules in the core network, the FC collaborate with the EC during the session initiation of each service slice. Indeed, the FC configures the fog cell transport network elements with the forwarding rules from the EC. Each fog cell is organized as clusters where each cluster covers a service slice for autonomous driving. Each cluster is controlled by a service slice controller (SSC) that manages the different flows arriving from the wireless access network. The service slice controller controls and manages different autonomous driving functions (presented in Section III): localization service, planning service, perception service, global system management. To ensure efficient and reliable service access to these functions, we propose to normalize them into service slices. Finally, the cloud control plane is responsible for constructing the global network connectivity by aggregating information received from all fog and edge controllers and generating control information based on rules and strategies from the application plane. The cloud control plane contains the SDN cloud orchestrator (CO), responsible for gathering information and constructing the global network connectivity. It can also be programmed to make automated decisions about the network in case of traffic congestion, faulty devices or security problems.

Application plane: Based on application requirements, rules and strategies are generated by CO for ECs and by FCs for the wireless vehicular network. The application plane contains a set of applications and services like resource organization, demands management, delay optimization, and trajectory prediction. The application plane interacts with each FC to forward

required information to the control plane. Additionally, in order to improve service access efficiency of autonomous vehicles, we consider the content fragmentation module as a service that is provided at the application plane. It requires the help of the autonomous-driving trajectory predection module to enhance data dissemination while offering real time information about nearby services.

7.4 Network and Service Slicing System Model

Ensuring and end-to-end service is among the main goals of 5G networks. We investigate the end-to-end performance of multi-service slicing in 5G networks based on a distributed and scalable SDN architecture. We begin by outlining the corresponding system model, then detail our function slicing model.

7.4.1 System Model

We analyze an interesting use case in vehicular networks where end-to-end communications delay is a critical metric. Indeed, we focus on autonomous driving vehicles with intensive computing tasks for real-time traffic detection and driving decision making. In this specific scenario, a softwarized 5G infrastructure enables large data transfer from various sensors on autonomous vehicles (e.g., cameras, sensors, ultrasonic radar…) to an operation center. Since the requirements for serving autonomous driving vehicles in terms of data rate and delay are hardly supported by microwave systems, the use of 5G mmWave cellular is considered here as a primary option for the autonomous vehicle. Other radio technologies are used as backup [107].

1) Environment and mobility model: We assume a typical urban setting similar to Central Paris. An urban outdoor environment is projected on a plane $\mathbb{R}^2$ where the roads are modeled using the Manhattan Poisson line process (MPLP) [108]. The road distribution model is featured by two unit-density homogeneous Poisson processes $\psi_x, \psi_y \subset \mathbb{R}^2$ along the x-axis and the y-axis, respectively. At each point of these processes, an avenue (West-East direction) or a street (North-South direction) grows infinitely along the x-axis and y-axis. The total street width is w_s and there are n lanes in each direction, each having the width of w_{Lane}. We model a straight segment of a street with left-side traffic.

We assume a test scenario with a number m of autonomous vehicles denoted by AV are traveling at a random speed S_{AV} that follows the probability density function (PDF) of $f_{S_{AV}}(x)$. The height of the vehicles is denoted by h_{AV} and all antennas are placed on their roofs for better channel conditions. Also, we assume that we have a number m' of manually driven vehicles

denoted by V assumed to be deployed randomly in the street. We denote by l the length of one manually driven vehicle V, its PDF is given by $f_{l_V}(x)$. The manually driven vehicles are traveling at a random speed S_V that follows the PDF of $f_{S_V}(x)$.

The vehicles are positioned on lane i following a Poisson process with intensity denoted by γ_i. d denotes the inter-vehicle distance in line i, $f_{d_i}(x)$ designates the PDF of d_i. All vehicles are assumed to have a constant width given by W and drive along the center of their lanes.

2) Heterogeneous wireless access network: in SliceScal, we propose to use the concept of heterogeneous vehicular access network. This concept is proposed to satisfy the communication requirements of not only autonomous driving service slices (i.e., safety message) but also best-effort traffic (i.e., non-safety message) for both autonomous driving vehicles and manually driven vehicles. Thus, due to high mobility and the dynamic change of vehicular network topology, it is difficult to provide satisfactory services through only a single wireless access network, such as dedicated short range communication (DSRC) or third generation 4G Long Term Evolution (LTE) networks or Wi-Fi [109]. In particular:

- DSRC networks are mainly designed for short-range communication without considering the pervasive communication infrastructure.
- The end-to-end delay in LTE networks can considerably fail with a large number of vehicles, while ultra-low latency for delivering real-time information for autonomous driving is required.
- The huge volume of data generated by sensors in autonomous driving system is beyond the capacity of the current vehicular networks.

Therefore, SliceScal architecture uses a heterogeneous wireless access networks which integrate four wireless technologies: (a) dedicated short range communication (DSRC), (b) cellular mmWave network, (c) cellular microwave network (LTE), and (d) non-3GPP microwave network (Wi-Fi). DSRC is used to provide V2V and V2F communications. Cellular mmWave base stations (BSs) are assumed to be deployed alternately on the left and the right sides of the buildings on a street at a constant height of h_{BS} with a distance of d_{BS} between one another. For microwave access, we assume that both LTE and Wi-Fi connectivity is always available via road side units (RSUs). Principally, the LTE network covers the entire city and maintains a constant SNR level via adequate power control. We also assume that there is always at least one Wi-Fi access point within the coverage area around the autonomous vehicle and we consider fixed-power transmissions, where the effective spectral efficiency is determined by true distance between the vehicles and the access point [110].

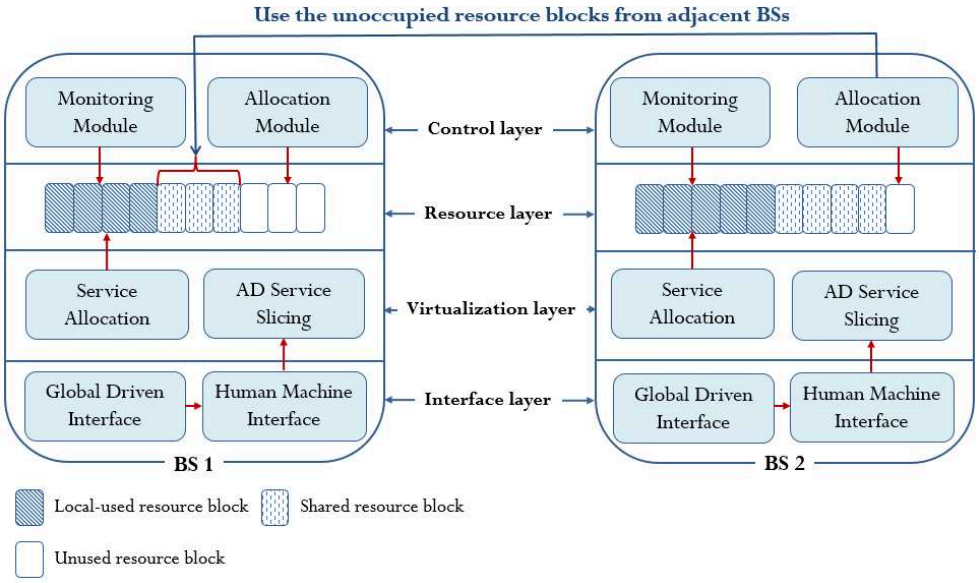


Figure 7.4- Logical layers for autonomous driving function slicing

Finally, in order to ensure high reliability and ultra-low latency when transmitting critical autonomous driving data and action-triggered messages, the autonomous vehicle should always keep a number of transmission data sessions to the nearest mmWave base station. Hence, the autonomous vehicle is in outage on the mmWave network only when it has outage on all the currently serving mmWave BSs. In this situation, the current autonomous driving service requests transmission falls back from the mmWave to the microwave segment (i.e., LTE or Wi-Fi). Based on our proposed SliceScal network configurations, the service slice controller (SSC) chose which wireless access network to use (i.e., LTE or Wi-Fi). Therefore, our SliceScal network architecture is always configured to grant high reliability and ultra-low latency for the requested data rates of autonomous driving service slice traffic even if it leads to declining the QoS of the best-effort for other vehicle (manually driven vehicles).

7.4.2 BSs Logical Layers for Function Slicing

To improve the transmission efficiency and satisfy the ultra-low latency constraint, the functions at BSs can be sliced and restricted to support the autonomous driving service slicing and resource slicing in future 5G vehicular networks. In this paper, the function slicing is realized by a new logical layers implemented at each BS. As we shown in Figure 7.4, the BS is consisted by four principal logical layers: control layer, resource management layer, virtualization layer and interface layer.

1) Control layer: includes two function modules, monitoring and allocation. The monitoring module takes charge of monitoring the number of occupied resource blocks at each BS

coverage area. In case the number of occupied resource blocks reaches the available number of blocks at the BSs and the new requirements of resource blocks are still needed by autonomous driving service slices, the allocation module will schedule the unoccupied resource blocks from adjacent BSs to be used by the corresponding BSs (see Figure 7.4). This is to improve the efficiency of resource utilization in 5G slicing vehicular network.

2) *Virtualization layer:* the main function of the virtualization layer is to organize different autonomous driving service slices and optimize the handling process for prompt response from the service slices. In addition, the virtualization layer allocates resource blocks at the location of each BS to satisfy the requirements of critical autonomous driving service slices.
3) *Resource management layer:* manages the resource blocks at the location of BSs. Originally, BSs resource blocks are allocated to a specific autonomous driving service slice. Thus, when the BSs resource management layer cannot satisfy the location requirement of service slices, the virtualization layer can provide more resource blocks from the associated control layer by sharing unoccupied resource blocks from adjacent BSs. It is the resource management layer duty to allocate these shared blocks in order to meet the requirements of different service slices.
4) *Interface layer:* provides the interface for different types of the predefined autonomous driving functions. The interface layer covers the human–machine interface (HMI) of the global system management service slice (GMSS). The main function of BS interface layer is to separate the BS resources with different types of autonomous driving service slices. Based on the BS interface layer, the relationship of different types of autonomous driving service slices is declining by separating the resource requirements with the type of service slice.

7.5 HSVF Performance Evaluation

To verify the effectiveness of HSVF, we used NS-2 and SUMO to simulate our VANET. We also used Mininet 2.1.0 with OpenFlow 1.3 for the SDN data plane, as it is more scalable and provides realistic environment for testing. We implemented a distributed controller based on POX. Briefly, POX catches link change event using Link Layer Discovery Protocol (LLDP) via OpenFlow. Discover module, and sends it to the path view module. This latter sends an event to all SDN-F controllers to inform them about the changes in the network topology. It also offloads EVs workload in order to compute the optimal path using parallel computing. Once done, we get the optimal route and modify the flow tables in OpenFlow switches using

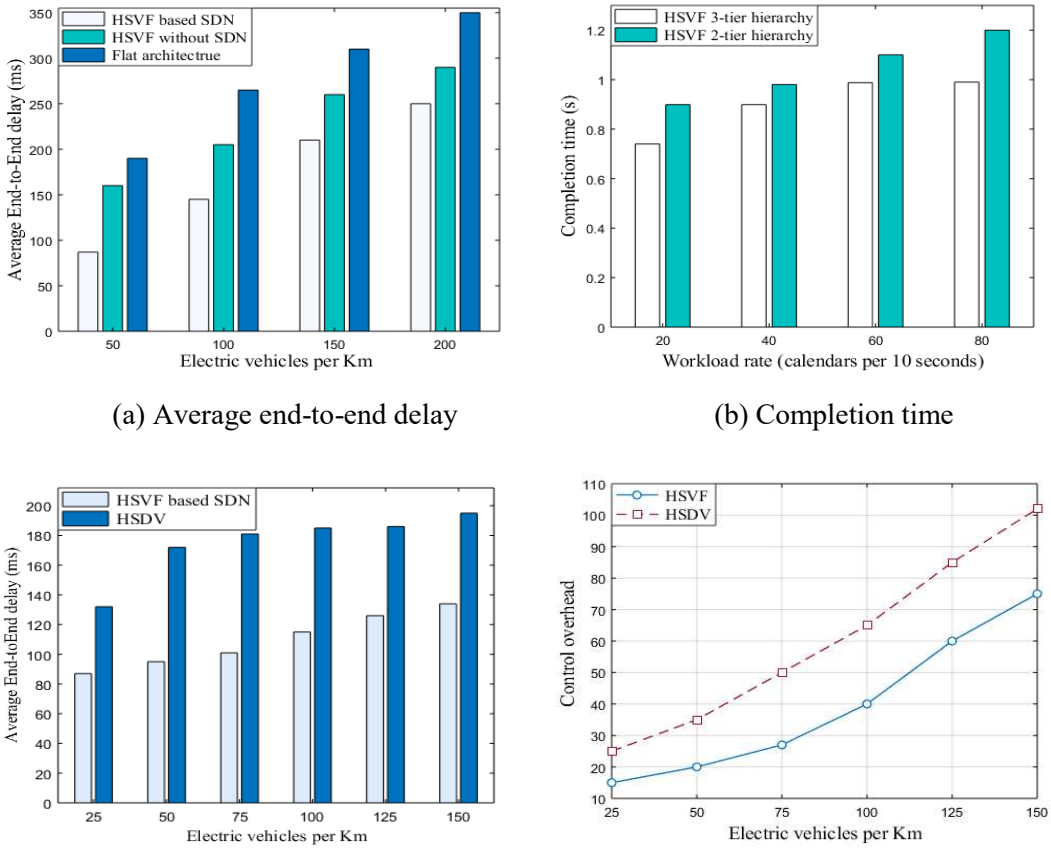
Forwarding.12_multi module. Like NS-2, POX is implemented in C++. So, the messages between the SDN network and the EVs network can be synchronized.

To make the evaluation realistic, we ran the simulations using a map of a region of the city of Paris, France. The road topology was obtained using OpenStreetMap and was filtered, formatted, and converted into a SUMO network file. Using SUMO, vehicular mobility traces were generated and used to populate the chosen area. We also deployed some RSUs along with some LTE base stations within that region following a uniform distribution. We varied the average number of vehicles between 50 and 200 per kilometer.

Table 7.1- Simulation parameters

Parameter	Value
Data rate	2Mbps (UDP)
Packet generation rate	5 – 20 packets/second
Packet size	512 byte
Transmission power	20 mW
Sensitivity	-89 dBm
Duration of a time slot	13 μs
Transmission range	200 m
Beacon rate	1 Hz
Speeds	30 and 80 km/h
Number of RSUs	10
Number of LTE BSs	5
Computation capacity	1.45 GHz and 2GB RAM
EV Calendars per second	2-5

Each car is equipped with a DSRC transceiver to enable V2V and V2I communications. During the simulation, each vehicle sends calendars as one packet using UDP protocol. We implement two hierarchical topologies (i.e., two tiers and three tiers) where we alter the number of servers at each tier between one and four according to the topology (Figure 7.2 represent an example of three tiers topology). Other simulation parameters are listed in Table 7.1.
We seek to evaluate two things: 1) the performance of HSVF when using SDN-F (HSVF-F); and 2) the offloading process of EV calendars when using the hierarchical deployment of fog cell servers. We compare HSVF-F to HSVF-W 2) (HSVF without SDN implementation) and the flat architecture of Fog cells servers. This latter can be considered as a traditional architecture of Fog servers' deployment and consists of using one tier of Fog servers (i.e., the offloading process is not supported).



(a) Average end-to-end delay

(b) Completion time

(c) Average end-to-end delay

(d) Control overhead

Figure 7.5- Simulation results comparisons

7.5.1 Delay Optimization Results

Figure 7.5(a) shows the average end-to-end delay as a function of vehicles' density. We observe that HSVF-F outperforms both HSVF-W and the flat architecture. Indeed, HSVF-F generates an average end-to-end delay that is 25% and 40% lower than the end-to-end delay incurred by HSVF-W and the flat architecture, respectively. The reason is threefold:

1) HSVF-W employs a single controller (i.e., SDN-C) to manage the entire network. As the number of vehicles increases, memory resources of the SDN-C controller might not be sufficient to handle the high number of EV requests, therefore incurring a high end-to-end delay. Unlike HSVF-W, HSVF-F partitions the network into sub-networks (i.e., fog cells), each of which is managed by a set of SDN-F controllers.; 2) HSVF-F uses vehicles' position along with their predicted trajectories to cope with the data synchronization problem and to avoid frequent handovers; and 3) Using SDN-F, HSVF-F can update flow tables periodically to reflect topology changes. Note that we repeat the experiments 30 times. Results represent the average of the runs with 95% confidence interval.

7.5.2 Completion Time Results

Figure 7.5(b) shows the completion time of different tiers in the fog architecture as a function of the workload rate. We observe that 3-tiers hierarchy outperforms 2-tier hierarchy in all the workload rates. This is 3-tiers hierarchy can efficiently offload the peak loads of calendars from lower tier servers to higher tier servers based on the offloading process depicted in Figure 3. After selecting the optimal calendars for each EV, the SDN-F selects the optimal routes according to the EV trajectory prediction.

7.5.3 HSVF Comparison with Related Work

Figures 7.5(c) and (d) depict performance results of HSVF and HSDV (i.e., Hierarchical Software-Defined VANET) [71], in terms of average end-to-end delay and control overhead. Figure 7.5(c) shows the average end-to-end delay with respect to vehicle density. We observe that HSVF outperforms HSDV in all densities. The reason is twofold: a) HSVF reduces the number of control messages in the network using a hierarchical server deployment at each fog cell along with a hybrid control plane; b) HSVF adapts the offloading mechanism of EV requests according to the capacity allocated at each tier's servers.

Figure 7.5(d) portrays the routing overhead in the wired fog cells network as a function of vehicle density. We observe that HSVF generates far less control messages compared to HSDV (i.e., on average, HSVF generates 30 % less control messages than HDSV). This is because HSVF makes use of the *wired fog cells network manager* module to: 1) ensure the proper management of the wired network connecting fog cells data centers; and 2) control the forwarding of flow table over different fog cells. HSVF also relies on SDN-C to route data packets and analyze the network topology of each fog data center. This reduces significantly the number of control messages.

7.6 SliceScal Performance Evaluation

In this section, we present a simulation-based evaluation of the proposed system. We examine the effectiveness of SliceScal and the 5G slicing model for autonomous driving. We develop an evaluation platform using network simulator NS-3, vehicles traffic simulator Veins-SUMO, and OpenFlow SDN controller. We employ a realistic urban scenario that may serve as a representative setup for a variety of possible deployment configurations. The road topology was obtained from OpenStreetMap, filtered, formatted, and converted into a SUMO network file.

7.6.1 Parametrization of Experimentation Scenario

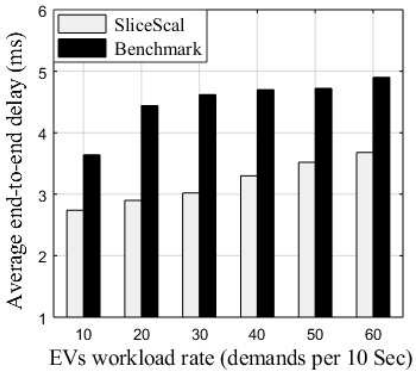
We model a 4-lane bidirectional street of 20 m width in the city of Paris. Each lane has a width of 3.65 m. We assume the Tesla Model S form factor with the dimensions of 4.979 m, 1.964 m, and 1.445 m as the autonomous vehicle. The number of AVs follow a uniform distribution from 2 to 10 vehicles. Following the data from [115], we allow the length of other vehicles, to follow a Gamma distribution with a mean of 4.5 m and a variance of 0.5 m^2. As for the height, we assume that it adheres to a Gamma distribution with a mean of 1.7 m and a variance of 0.3 m^2. The speed of vehicles (including autonomous vehicles) is uniformly distributed between 20 km/h and 80 km/h, which is representative of dense urban setting.

The mmWave radio propagation follows the default 3GPP protocol, where the case of blockage is modeled as NLoS conditions. According to 3GPP release 14, in case of blockage, the mmWave radio access is capable of establishing an alternative NLoS path that is currently non-blocked. We also adopt the typical parameters of mmWave cellular. Specifically, we set the transmission power to 25 dBm and the center frequency to 73 GHz with 2 GHz of bandwidth. Antennas' gain for BS and autonomous vehicles is assumed to be 15 dB and 10 dB, respectively. The noise floor is set to −80 dB.

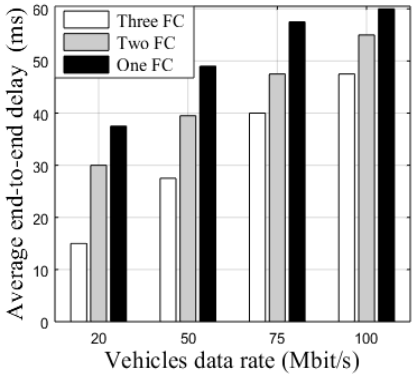
7.6.2 SliceScal Implementation

To demonstrate the ultra-low latency approach for autonomous driving advocated in this paper, we construct a simple but representative proof-of-concept implementation of SliceScal core network. In our test implementation, we use two Linux servers implemented on two DELL OptiPlex 7050 with 32GO of RAM and Intel core i7-7500T CPU 3.75GHz. Each Linux machine acts as the SDN edge controller (EC) In each Linux machine, we implement five VMs, three of them act as SDN fog controllers (FC). We use Mininet 2.1.0 with OpenFlow 1.3 for the SDN FC data plane as it is more scalable and provides a realistic environment. The two remaining VMs are used as hosts to either generate or receive data traffic. The generated UDP-flows are created with iperf [116] and represent either autonomous driving traffic or best effort traffic (other vehicles traffic.

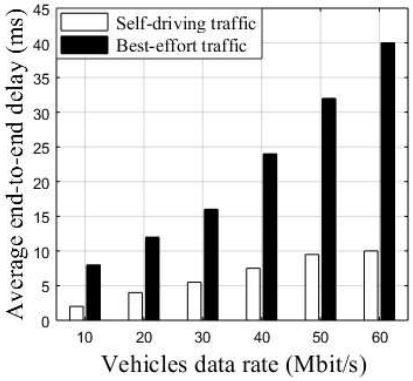
Further, we employ the OpenDaylight (ODL) SDN controller to manage FC switches. The communication between the ODL controller and the switches is based on the OpenFlow protocol. In order to have a distributed control plane, the topology of our core network has been implemented using POX [117]. To maintain the priority of the critical autonomous driving flows with respect to other flows (e.g., best-effort traffic), FC maps a flow onto a priority queue. Relying on each FC controller, we assign the data flows initiated by the VMs hosts to the



(a) SliceScal Vs Benchmark



(b) Scaling of fog controllers



(c) Autonomous driving Vs Best-effort

Figure 7.6- Performances of SliceScal core network

queues. We distinguish the critical autonomous driving traffic and the best-effort traffic using the destination IP address in the packets. The minimum (guaranteed) and maximum service rates need to be maintained in each of the queues to avoid disturbance from the best-effort traffic toward the critical autonomous driving traffic since the sum of both data flows represent the full capacity of the link.

7.6.3 Performance Results

To evaluate the average end-to-end delay, we make vehicles generate traffic at different data rates. Figure 7.6(a) illustrates the average end-to-end delay of over SlcieScal against a benchmark scheme that does not use the SDN approach. As vehicle's data traffic increases, we observe that SlcieScal outperforms the benchmark topology by (30% to 45%) in term of average end-to-end delay. We also evaluate the average end-to-end delay with repsect to generated traffic flows when scaling the SliceScal by using different number of distributed fog controllers (FCs) (i.e., One FC, two FC, three FC). Figure 7.6(b) shows that the topology with three FCs incurs the shortest average end-to-end delay compared to the other topologies. Indeed, when

using only one FC, the controller gets overwhelmed as the generated traffic per vehicle increases, leading to packet drops and high communication overhead. However, by partitioning the vehicular network into sub-networks (i.e., fog cells and clusters), where each fog cell has an SDN controller, we endorse the load balancing capability. In fact, fog controllers are responsible for only managing the vehicles located in their corresponding geographical area.

In Figure 7.6(c), we compare the average end-to-end delay of autonomous driving traffic and best-effort traffic in the core network. We can observe that autonomous driving traffic incurs is shorter end-to-end delay compared to best-effort traffic. This is because FC maintains the priority of the critical autonomous driving flows with respect to other flows (e.g., best-effort traffic) by mapping them into a priority queue. Results presented in Figure 7.6 demonstrate that the SlcieScal core network is suitable for scheduling and transmitting critical traffic (i.e., autonomous driving traffic) especially when using 5G wireless network and service slicing.

7.7 Chapter Summary

In this Chapter, we propose two architectures for vehicular network. HSVF, a hierarchical architecture for wireless vehicular networks based on a new hybrid SDN architecture. HSVF consists of decentralized SDN controllers deployed over distributed fog cells and a centralized SDN controller implemented within the cloud data centers. Simulation results show that HSVF incurs shorter end-to-end delay and completion time compared to existing approaches. Second, we introduced a distributed and scalable SDN core network architecture called SliceScal designed specifically for autonomous driving service slicing use case. We then presented the essential autonomous driving functions that need to be sliced. The simulation results have demonstrated that the SlcieScal core network is suitable for scheduling and transmitting critical autonomous driving traffic especially when using 5G wireless network and service slicing.

CHAPTER 8
CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

Cloud computing have seized the attention of different governments, industries, and academic institutions as they offer tremendous potential for new business opportunities in terms of scheduling efficiency, storage capacity, and availability. Yet, they have particular challenges that hinder the cloud computing applications for new emerging technologies and systems. The main objective of this research project is to design new cloud architectures and models for: 1) smart grid energy management, 2) industrial internet of things, and 3) vehicular network based on SDN and 5G technologies. To accomplish this goal, this dissertation presented efficient and safe architectures and models based on cloud and decentralized fog computing.

For smart grid energy management, we proposed three contributions (i.e., an EV energy plug-in model, a dynamic pricing model, and a secure architecture for smart grid energy metering against false data injection attacks). We proposed an energy saving solution in smart grid by proposing optimal scheduling models for plug-in EVs charging. The plug-in system model of EVs is based on calendar planning using distributed Fog computing capacities. We used a multi-priority queueing system to manage the plug-in of EVs at each electric public supply station; the proposed queuing system used a cut-off discipline in order to enhance to plug-in process. To improve our planning scheme and maintain the energy curve stability, we proposed three optimization algorithms as follows: 1) two priority attribution algorithms for both the calendar's plug-in and random plug-in according to different energy constraints, and 2) an algorithm to select the optimal EPSS for each EVs to plug-in.

In addition, we proposed a real-time dynamic pricing model for EVs charging and discharging service and building energy management, in order to reduce the peak loads. The proposed approach used a decentralized cloud computing architecture based on software define networking (SDN) technology and network function virtualization (NFV). We formulated the problem as a linear optimization problem for EV and a global optimization problem for all micro grids. We solved the problems by using different decentralized decision algorithms. The extensive simulations and comparisons with related works proved that our proposed pricing model optimizes the energy load during peak hours, maximizes EVs utility, and maintains the micro grid stability.

To secure smart metering infrastructure against false data injection attacks in smart grid environment, we proposed SecMetering; a new security scheme based on hierarchical and distributed intrusion detection system (HD-IDS). The proposed HD-IDS is based on distributed fog architecture considering three hierarchical network layers. The problem is formulated using stochastic modeling. Based on a Semi-Markov process, we implemented an embedded Markov chain to observe the behavior of each smart meter using range-based behavior sifting policy. In addition, we proposed four algorithms to analyze efficiently the behavior of smart meters and detect false data attacks in real-time way at different layers of the proposed HD-IDS architecture. We validated the effectiveness of SecMetering via extensive simulations.

For Industrial IoT data scheduling, we designed a decentralized multi-tiers fog architecture for IIoT requests scheduling and data analytics. The architecture is based on hierarchical deployment of servers in the fog layer. We proposed a probabilistic model to compare the efficiency of fog resources utilization between flat and hierarchical fog architectures. We show, using an analytical model that the hierarchical fog architecture has better efficiency requests handling in terms of minimizing computation and communication delay. Moreover, we formulated the IIoT workload assignment problem as a mixed nonlinear integer programming (MNIP) problem, and proposed a branch and bound approach to solve the problem. Then, the optimal results will be aggregated for high tiers using Simulated Annealing Algorithm in order to ensure a global optimality of IIoT workload assignment. Finally, we developed an IIoT offloading algorithm which offloads the workloads over different tiers of the hierarchy.

Finally, in order to enhance the QoS of vehicular network, we proposed two new vehicular architecture based on decentralized SDN and 5G technologies. First, we proposed a hierarchical SDN-based wireless vehicular fog architecture, called HSVF (i.e., Hierarchical SDN for Vehicular Fog architecture). HSVF is based on a hybrid SDN control plane that reinforces centralized and distributed management. The proposed architecture is then evaluated using a relevant case study that addresses the scheduling of electric vehicles (EVs) energy demands. Second, we proposed a distributed and scalable SDN core network architecture (SliceScal) that supports 5G slicing functions, which map autonomous driving functionalities into service slices. SliceScal deploys fog, edge and cloud computing technologies and proposes a network and service slicing system model that promotes a four-layer logical architecture to improve the transmission efficiency and satisfy the low latency constraint.

8.2 Future Work

Beyond the previously proposed architectures and models based on cloud for smart cities networks (e.g., smart grid, IoT, VANET), various future opportunities are provided hereafter with some research challenges.

Smart grid management: As we presented in Chapter 3 and Chapter 4, the heterogeneous architecture of smart grid can lead to huge amount of data generation from smart meters, and thus, cloud computing applications and decentralized fog architectures are useful for managing these data. However, there are some research challenges in smart grid, while incorporating the cloud computing applications. First, due to failures in one of the protocol used for communication, the entire system is affected in the smart grid technologies. Thus, cloud computing can be used as a solution to recover network failures. In addition, data warehouse-based cloud can be implemented in order to optimize, control, and secure smart grid communications. Second, the integration of public cloud with the private cloud infrastructure for cost-effective communication in smart grid is indispensable for new scalable smart cities networks. Public cloud can be integrated with the private cloud for large-scale development in smart grid in a cost-effective manner. However, security and privacy are two of the important issues, while allowing information exchange between private and public cloud. Third, due to heterogeneous communication architecture of smart grid when integrating cloud computing, multi-mobile agent can be used to communicate with different layers. Therefore, defining an adequate mobile strategy for the agent is required for cost-effective information management in the smart grid. Otherwise, it will be cost-expensive rather than the cost-effective one. Finally, an interactive cooperation using cloud services to support not only multiple customers, but also multiple energy sources should be considered in future works especially for large-scale development of smart grid energy management.

As we presented in Chapter 5, with the presence of online connectivity, a smart grid can be conceptualized as a cyber-physical system that connects physical electricity systems and cyber-infrastructure. However, it is a big challenge to prevent cyber-attacks in the smart grid that can potentially disrupt the power supply. One of the important issues is power theft by consumers. This can be done by hacking a smart meter or modifying the real-time information through accessing communication channel to change the reported electricity usage. Additionally, data manipulation is also one of the most security concerns in the smart grid. We have proposed in this work a mechanism (i.e., SecMetering) to secure smart metering against FDIA in smart grid.

As an extension to SecMetering, would be to implement a coordinated fault protection mechanism with the help of decentralized fog infrastructure in each micro grid. In this mechanism, different equipment are able to perform together efficiently. In addition, we aim to add a state estimation and perception method based on Machin learning for proactive and adaptive management for micro grid. Therefore, adequate strategy can be taken by service providers in order to provide reliable energy services to the end-users.

Industrial Internet of Things: The IoT has become the major disruptive technology changing software and society. Big factories, cities, and even countries must now start on this networked digitization. But focusing only on networking isn't enough. Networking depends on the software that collects the information, and that software must be designed for the IoT's unique needs. Various initiatives have already delivered IoT models, architectures, and tools. In this dissertation, we presented in chapter 6 a hierarchical fog computing architecture for scheduling IIoT data. However, a further convergence of existing approaches and industrial standards is required for simplification. This raises issues about the key IIoT enablers. One key enabler is efficient and secure communication. Many technologies exist to implement the communication stack. Low-power wireless networks that require little implementation effort are necessary, for which security is an important factor. On the other hand, device hardware suitable for implementing the IoT is readily available. However, it's questionable whether device-oriented, real-time networked operating systems will ever exist for the different application domains or whether the diversity of real-time constraints will continue.

Vehicular networks: In Chapter 7, we presented two new architectures for vehicular network based on SDN and decentralized fog computing architecture (i.e., HSVF, SliceScal). However, some research issues need yet to be addressed to take full advantage of our proposed architectures for vehicular networks. On the one hand, a mechanism to handle vehicle handovers, considered as a crucial issue in the multihop communication within a fog cell, needs to be developed. Moreover, the propagation delay of the warning messages needs to be minimized in 5G vehicular networks. Moreover, the OpenFlow standard needs to be extended to support more features in wireless vehicular communications, including: power adjustment, channel assignment and frequency allocation.

On the other hand, in a time where data traffic is constantly booming, software-defined networking based cloud computing architectures are becoming the most plausible solution to cope with the conventional networks' shortcomings. However, initial SDN deployments promoted a centralized architecture with a single controller that is responsible for managing

entire networks. Therefore, scalability is one of the major challenges faced by the logically centralized SDN architecture. In SDNs, as the network size and diameter grows, the amount of control traffic destined towards the centralized controller increases and as a result the flow setup time grows. Moreover, it is known that the capability and operation set of an OpenFlow controller is most likely limited. Yet, this design has proven to be unsalable and unreliable. Though, distributed control plane architectures have gained ground lately, they bring new concerns. A paramount question that should be investigated thoroughly is how to efficiently perform path computation and flow table updates in very large SDNs. In our future works we aim to address this inquiry by proposing new path computation models for distributed and scalable SDN networks and describes a mechanism to update path views in case of topology changes along with a failover mechanism to address the problem of isolated fog cells.

Finally, SDN-based VANET can enhance network security due to global visibility of the network state, centralized intelligence and network programmability. As such, a common distribution layer gathering information about security requirements of different services, resources and hosts, and disseminating the security establishing commands to the network elements to enforce security policies can result in robust and scalable security enforcement. However, the same key attributes of SDN (i.e., centralized intelligence and programmable network elements), make security in SDN a more challenging task. Indeed, to our understanding, there are still some security features which need to be addressed for our proposed SDN-based architectures for VANET. First, controllers should be protected all the time against DDoS attacks, particularly the centralized controller. Hence, the lack of scalability in SDN can enable targeted DDoS attacks by inundating communication between the controller and the switch to cause control plane saturation. Besides saturation of the control plane, it is demonstrated that a network can be compromised by exhausting resources of OpenFlow-enabled switches. In addition, availability is a security aspect that strongly correlates scalability with security. Most of the security threats in SDN target to compromise availability of the control plane. Thus, multiple controllers are suggested, but, simply adding multiple controllers might result in cascading failures of controllers as demonstrated. Therefore, it is necessary to correlate security and scalability in SDN to design secure SDN architectures that ensure high availability of the control plane. Thus, there is a clear need to perform more research on securing SDN-based VANET architectures because attacks not only have the potential for devastating effects to both smart cities networks and to individuals associated with them but also are expected to become even more critical in the future.

7.3 Articles Published/Submitted

The list of journal articles and conference papers produced during this thesis is as follows:

1. D. A. Chekired and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," *in IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3312-3321, Dec. 2017.

2. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Decentralized Cloud-SDN Architecture in Smart Grid: A Dynamic Pricing Model," *in IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1220-1231, March 2018.

3. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory," *in IEEE Transactions on Industrial Informatics,* vol. 14, no. 10, pp. 4590-4602, Oct. 2018.

4. D. A. Chekired, M. A. Togou and L. Khoukhi, "Hierarchical Wireless Vehicular Fog Architecture: A Case Study of Scheduling Electric Vehicle Energy Demands," *in IEEE Vehicular Technology Magazine*, vol. 13, no. 4, pp. 116-126, Dec. 2018.

5. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Multi-Level Fog Based Resource Allocation Model for EVs Energy Planning in Smart Grid," *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, Chicago, IL, USA, 2018, pp. 243-250.

6. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Queuing Model for EVs Energy Management: Load Balancing Algorithms Based on Decentralized Fog Architecture," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018.

7. D. A. Chekired and L. Khoukhi, "Multi-Tier Fog Architecture: A New Delay-Tolerant Network for IoT Data Processing," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp. 1-6.

8. D. A. Chekired and L. Khoukhi, "Optimal priority-queuing for EV charging-discharging service based on cloud computing," *2017 IEEE International Conference on Communications (ICC)*, Paris, 2017, pp. 1-6.

9. D. A. Chekired, M. Amine Togou and L. Khoukhi, "A Hybrid SDN Path Computation for Scaling Data Centers Networks," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

10. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Fog-Based Distributed Intrusion Detection System Against False Metering Attacks in Smart Grid", Accepted in *2019 IEEE International Conference on Communications (ICC).*

11. D. A. Chekired, S. Dhaou, L. Khoukhi and H. T. Mouftah, "Dynamic pricing model for EV charging-discharging service based on cloud computing scheduling," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC),* Valencia, 2017, pp. 1010-1015.

12. D. A. Chekired, D. Said, L. Khoukhi and H. T. Mouftah, "A novel pricing policy for G2V and V2G services," *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, 2017, pp. 634-635.

13. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Fog Computing Based Energy Storage in Smart Grid: A Cut-off Priority Queuing Model for plug-in Electrified Vehicles Charging" *Under Major revision in IEEE Transactions on Industrial Informatics.*

14. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Distributed Fog-Based Energy Solution for Scheduling EVs Plug-in Requests in Smart Grid", *Submitted in IEEE Transactions on Parallel and Distributed Systems.*

15. D. A. Chekired, L. Khoukhi and H. T. Mouftah, "5G Slicing of EVs Energy Services: A Latency Analysis Model Based on Decentralized SDN-Fog Architecture", *Submitted in 2019 IEEE 44rd Conference on Local Computer Networks (LCN).*

16. M. A. Togou, D. A Chekired, L. Khoukhi and G. Muntean, "A Distributed Control Plane for Path Computation Scalability in Software-Defined Networks," *2018 IEEE Global Communications Conference (GLOBECOM),* Abu Dhabi, United Arab Emirates, 2018.

17. M. A. Togou, D. A. Chekired, L. Khoukhi and G. Muntean, "A Hierarchical Distributed Control Plane for Path Computation Scalability in Large Scale Software-Defined Networks," *in IEEE Transactions on Network and Service Management.*

RÉSUMÉ EN FRANÇAIS

9.1 Introduction

9.1.1 Le Cloud Computing

Le cloud computing est un modèle de calcul émergeant qui fournit des services à la demande et des ressources partagées sur Internet. Le cloud computing, basé sur de grands dispositifs de stockage et de calcul, agit en tant que fournisseur d'utilité [1]. Les coûts de mise en œuvre et de maintenance, ainsi que la complexité du système informatique dans le Cloud, sont réduits grâce à l'utilisation des nouvelles technologies de l'information. Les ressources sont partagées sur le réseau et récupérées par des périphériques à la demande. Le cloud computing fournit trois types de services distincts : la plate-forme en tant que service (PaaS), le logiciel en tant que service (SaaS) et l'infrastructure en tant que service (IaaS) [2].

9.1.2 Le réseau électrique intelligent

Un réseau électrique intelligent peut être conceptualisé comme une intégration du réseau électrique avec le système de réseau de communication [6]. Grâce à l'intégration des nouvelles technologies de l'information et de la communication, le réseau électrique intelligent est en mesure de fournir de l'électricité aux utilisateurs finaux de manière efficace. L'architecture de réseau intelligent couvre principalement trois domaines techniques différents : système de génération, côté transmission et côté distribution.

9.1.3 Cloud computing pour la gestion d'énergie

La gestion de l'énergie est une préoccupation majeure pour le réseau électrique intelligent. Au cours des dernières années, les chercheurs ont abordé cette question en intégrant la mise en œuvre de différents composants tels que le système de gestion de l'énergie domestique, le système de gestion de l'énergie du bâtiment, la tarification dynamique et le transfert de la charge [10]. Par conséquent, l'objectif du réseau électrique intelligent est de prendre en charge une gestion de l'énergie rentable et fiable en temps réel. D'autre part, le réseau électrique intelligent offre de nombreuses fonctionnalités et applications aux consommateurs, mais il doit être amélioré pour gérer des systèmes plus évolutifs. Ceci peut être réalisé en utilisant des capacités de calcul dans le Cloud. Les opérations peuvent être effectuées à moindre coût avec le Cloud computing car le partage et l'automatisation sont répandus au sein de ces systèmes. En outre, la réponse en temps réel est très importante pour les applications du réseau électrique intelligent

afin de répondre immédiatement à la demande des utilisateurs. Sur les plates-formes de Cloud, lorsqu'une demande du client parvient au système d'exploitation, une réponse est envoyée au client en temps réel.

9.1.4 Cloud computing pour les objets connectés

L'Internet des objets (IoT) gagne de plus en plus d'attraction dans de nombreux secteurs industriels, tels que la logistique et la fabrication [18]. D'autre part, avec les progrès des technologies de communication sans fil et de réseau de capteurs, de plus en plus les objets connectés sont impliqués dans l'IoT industriel (IIoT). En conséquence, les technologies liées à l'IIoT ont également eu un impact important sur les technologies de l'information et de la communication (TIC) et les systèmes industriels [19]. Récemment, le secteur de l'industrie a connu une autre transformation déclenchée par la connectivité intelligente et l'analyse avancée des informations. C'est ce que l'on appelle l'industrie de pointe en Amérique du Nord et l'industrie 4.0 en Europe [20]. D'une part, cette transformation facilite la flexibilité et l'évolutivité, en particulier en ce qui concerne la génération élevée de données et d'informations. D'autre part, cela permet également de livrer des produits de meilleure qualité à un coût réduit, en exploitant les données collectées par divers éléments d'une chaîne de montage connectée.

Par conséquent, une grande quantité de données est générée à partir de périphériques IIoT installés sur différents composants physiques d'usine tels que les chaînes de montage. Cependant, une grande partie des données générées et collectées sont utilisées pour le contrôle en temps réel. Comme présenté dans une étude de cas du groupe Bosch dans [20], au cours du processus de fabrication, des informations sur l'état du processus, l'état des machines, des outils et des pièces produites sont relayées, planifiées et stockées de manière ininterrompue. La capacité, l'échelle et la régularité de la production décrites dans cette étude de cas traditionnelle sont si élevées qu'elles nécessitent l'utilisation d'un ordonnancement Big Data pour stocker, analyser et prendre les décisions optimales. À cet égard, une gestion efficace des services pour IIoT doit être mise en œuvre et le réseau industriel traditionnel doit être amélioré et adapté afin de garantir un processus analytique en temps réel, de minimiser le délai de bout en bout et d'augmenter la capacité de stockage de données.

Récemment, le Cloud computing, avec ses vastes capacités de calcul, la capacité de stockage et ses aspects distribués, a été considéré comme une approche clé pour le traitement de données volumineuses [21], [22]. Toutefois, dans les systèmes traditionnels, la plupart des interactions entre les périphériques IIoT et les serveurs principaux se font via des centres de données cloud

à grande échelle. Néanmoins, en raison des retards induits sur les réseaux étendus (WAN) et éloignés des dispositifs IIoT, les systèmes IIoT basés sur le cloud rencontrent de nombreux défis, notamment un temps de réponse élevé en raison du coût élevé de la bande passante de communication et des lourdes charges de travail.

9.1.5 Cloud computing basé sur SDN pour le réseau véhiculaire

Le réseau défini par logiciel (SDN) est un nouveau paradigme des réseaux qui est de plus en plus intégré dans différents types de réseau de communication. Dans un réseau SDN, le plan de contrôle et le plan de données sont découplés et les ressources du réseau sont gérées par un contrôleur centralisé de manière logique. De plus, les appareils de différents fournisseurs peuvent communiquer entre eux via une interface standardisée, ce qui simplifie considérablement la gestion du réseau et offre une architecture réseau programmable et flexible. À l'origine, SDN était conçu et déployé dans des environnements de réseau filaire avec des commutateurs à haute vitesse. Pourtant, les chercheurs ont récemment commencé à utiliser le SDN pour améliorer de nombreuses formes de réseaux distribués sans fil, y compris les réseaux ad hoc mobiles, les réseaux de capteurs sans fil, les réseaux Wi-Fi et les réseaux de véhicules [28]. En effet, le SDN apporte de nouvelles intuitions et un potentiel élevé d'améliorer l'efficacité des réseaux sans fil [29].

D'autres technologies peuvent également être envisagées pour améliorer les performances de VANET. Par exemple, des études récentes ont montré que l'interconnexion de VANET avec des services de fog computing peut améliorer les performances du réseau [30]. De plus, le cloud peut également être considérée comme une solution fondamentale pour prendre en charge l'écoulement du trafic mobile et pour réduire le temps de latence. Les services de fog computing, y compris le calcul, le stockage et les réseaux distribués, sont déployés près des utilisateurs finaux ; ainsi, un accès rapide et fiable est fourni aux applications mobiles. Néanmoins, ces services de fog peuvent être dépassés, dans certains cas, en raison du nombre croissant de demandes des clients. Ceci peut être observé en particulier aux heures de pointe, entraînant par la suite une dégradation critique des performances.

9.2 Les objectifs de la thèse

Quand le trafic de données et la consommation d'énergie sont en plein essor dans les villes intelligentes, le cloud computing devient la solution la plus plausible pour faire face aux faiblesses des réseaux de villes intelligentes classiques. La plupart de ces réseaux classiques ont des exigences de performances strictes (c'est-à-dire, l'économie d'énergie, le délai, le passage

à l'échelle, la bande passante et le débit). Cependant, il n'est pas simple de répondre à ces exigences, compte tenu des caractéristiques particulières des réseaux de villes intelligentes (forte consommation d'énergie, forte mobilité, connectivité intermittent, trafic de données important et menaces à la sécurité). L'objectif de cette thèse est donc de contribuer à la conception et à l'évaluation de nouvelles solutions répondant aux besoins des réseaux de villes intelligentes. Ce travail a été effectué sur la sous-couche de contrôle d'accès au support (MAC), la couche réseau et la couche application. D'une part, il propose de nouveaux modèles et architectures efficaces et sûrs basés sur les technologies cloud et SDN pour : 1) la gestion de l'énergie dans les réseaux électriques intelligents, 2) la gestion et le traitement des données IIoT, et 3) les véhicules connectés. Les solutions proposées prennent en compte des paramètres d'évaluation des performances, tels que l'économie d'énergie, le temps de réponse et les latences. D'autre part, elles définissent, évaluent et implémentent des mécanismes de gestion des ressources du réseau, afin de proposer des services différenciés afin d'améliorer encore les performances de différents réseaux de ville intelligente (réseau électrique intelligent, réseau IoT et VANET).

9.3 La gestion de l'énergie dans les réseaux électriques intelligents

La première partie de cette thèse aborde le problème des économies d'énergie dans les réseaux intelligents en considérant trois problèmes principaux : 1) la gestion du plug-in des véhicules électriques pour le chargement et le déchargement des batteries, 2) la tarification dynamique et en temps réel et l'optimisation des coûts, 3) la sécurisation de comptage intelligent de la consommation d'énergie contre les attaques d'injection de fausses données. Plusieurs approches ont été proposées dans la littérature pour traiter les problèmes précédents (par exemple [38], [39], [43], [48], [49], [60]). En dépit de leurs bonnes performances, les systèmes existants entretiennent des lacunes majeures. Ces modèles sont basés sur des architectures centralisées conventionnelles qui n'acquièrent qu'une topologie de réseau local et centralisé dans un environnement de réseau intelligent, ce qui les expose au risque de saturation maximale des données locales et à un délai de réponse élevé qui affecte considérablement les économies d'énergie dans le réseau électrique intelligent, en particulier pour les villes intelligentes.

9.3.1 Gestion du plug-in des véhicules électriques

L'émergence des véhicules électriques (EV) promet d'apporter de nombreux avantages aux secteurs de l'énergie verte et des transports, mais elles risquent également d'affecter la fiabilité du réseau électrique en consommant une quantité d'énergie considérable pour charger les

batteries des voitures. Néanmoins, le branchement (plug-in) des véhicules électriques dans les stations d'approvisionnement publiques (EPSS) doit être contrôlé et planifié afin de réduire la charge. Afin de réduire l'impact de la recharge des véhicules électriques sur le réseau électrique intelligent et d'optimiser les économies d'énergie ; nous proposons un modèle coordonné pour planifier le plug-in des véhicules électriques pour charger et/ou décharger de l'énergie.

Nos contributions sont résumées comme suit : a) nous proposons un nouveau système pour le plug-in des EVs basé sur la planification du calendrier en utilisant les capacités de calcul réparties dans les serveurs de Fog computing, b) nous utilisons un système de file d'attente à plusieurs priorités pour gérer le plug-in des EV. Le système de file d'attente proposé utilise une discipline de coupure (cut-off) afin d'améliorer le processus de plug-in, c) afin d'améliorer notre modèle et de maintenir la stabilité de la courbe d'énergie, nous proposons trois algorithmes d'optimisation comme suit: : i) deux algorithmes d'attribution de priorité pour le plug-in avec calendrier et le plug-in aléatoire en fonction de différentes contraintes d'énergie, ii) et un algorithme pour sélectionner l'EPSS optimal pour chaque EV à plug-in, d) nous effectuons des simulations approfondies afin d'améliorer l'efficacité de notre solution; Nous comparons également nos algorithmes avec deux travaux récents.

9.3.2 Une architecture Fog computing décentralisée pour les réseaux électriques

La planification centralisée des demandes de EV avec leurs caractéristiques de batterie hétérogènes et la gestion des EPSS, y compris la consommation de base, dans une zone géographique, est un défi. Dans l'architecture centralisée, toutes les informations et les opérations sont planifiées à l'intérieur du micro-grid. Avec la capacité de stockage et de calcul limitée du réseau électrique intelligent, ainsi que le nombre croissant de véhicules électriques, la réponse à la demande en temps réel devient très problématiques.

9.3.2.1 Pourquoi une architecture décentralisée ?

Avec le nombre croissant de parcs d'automobile, il est fort probable que l'architecture centralisée traditionnelle atteigne sa limite physique de traitement de ces données volumineuses. Une telle architecture peut entraîner une congestion des données, une latence

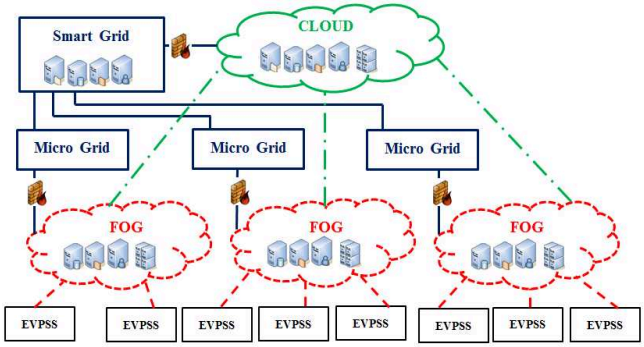


Figure 9.1- Decentralized Fog for smart grid communications

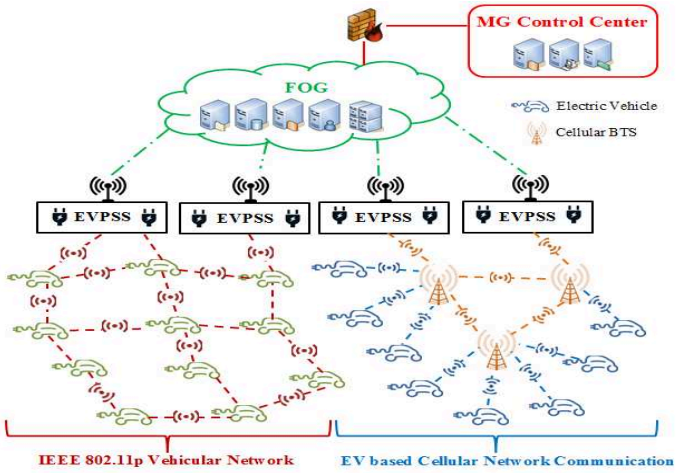


Figure 9.2- Fog architecture for EVs charging/discharging service

importante, voire une perte de données, ce qui peut sérieusement endommager les services de réseau électrique [38]. Il est donc essentiel pour les fournisseurs d'électricité de définir de nouvelles normes de communication et de trouver la meilleure infrastructure informatique pour contrôler et gérer tous les clients. Pour résoudre ces problèmes, nous proposons un nouveau concept de calcul basé sur une architecture décentraliser, dans lequel l'étude de cas est la planification des demandes de chargement/déchargement de véhicules électriques (voir Figure 9.1). Nous sommes convaincus qu'un aspect décentralisé de cloud computing (c'est-à-dire le Fog) constitue une solution encourageante grâce aux avantages suivants :

- La technologie cloud offre une capacité de calcule élevée aux utilitaires de réseau électrique intelligent. En outre, grâce à son énorme stockage de données, le cloud peut aider le réseau électrique à améliorer les services de gestion des demandes.
- L'aspect décentralisé des réseaux cloud, qui utilise ce que l'on appelle le Fog-computing, peut être utilisé pour gérer et contrôler chaque micro-grid de manière distribuée.

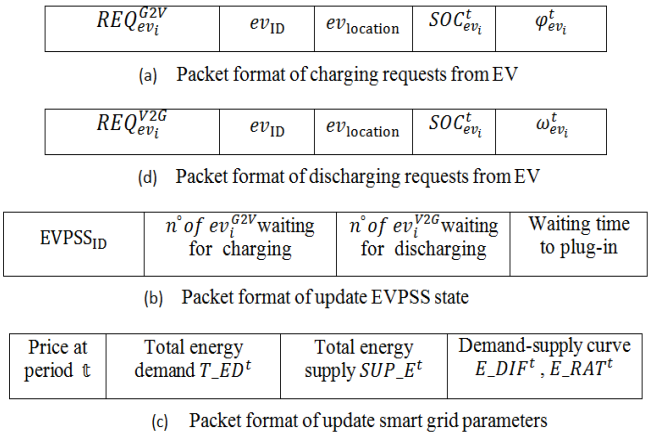


Figure 9.3- Packet format of EVs, EVPSS and smart grid communications

9.3.2.2 Gestion des demandes des véhicules électriques assistés par le fog

Dans ce travail, nous proposons une architecture fog décentralisée pour un système de transport intelligent des véhicules électriques dans l'environnement d'un réseau électrique moderne. Dans le schéma proposé, plusieurs centres de données sur le fog sont déployés dans une ville intelligente et connectés à chaque contrôleur de micro-grid ; chaque centre de données sur le fog joue le rôle de desservir une zone locale (voir Figure 9.2). En outre, le centre d'opérations décentraliser est capable de collecter et d'analyser des données localement et de prendre des décisions plutôt que de renvoyer toutes les données au centre d'opérations central (cloud).

En effet, chaque centre de données dans le fog et chaque centre de données dans le cloud peuvent communiquer respectivement avec le contrôleur de micro-grid et le contrôleur de réseau électrique intelligent afin de mettre à jour périodiquement des paramètres énergétiques, tels que le prix, la production d'énergie, etc. La figure 9.3 illustre le format de paquet échangé entre les véhicules électriques, les stations EPSS et les contrôleurs de réseau électrique intelligents.

9.3.3 Une planification basée sur les calendriers pour le plug-in des EVs

Afin de planifier le processus de plug-in des véhicules électriques, nous avons proposé une architecture basée sur le Fog décentralisé (Figure 9.1), dans laquelle chaque micro-grid est relié à un fog. Nous utilisons les capacités de calcul dans le fog pour planifier toutes les demandes des EVs. Pour rendre cette utilisation plus efficace dans notre modèle, les utilisateurs des EVs sont invités à créer et sauvegarder leurs profils énergétiques $U_{profile}$. Un fichier $U_{profile}$ contient des informations sur les utilisateurs de EVs, telles que représentées ci-dessous:

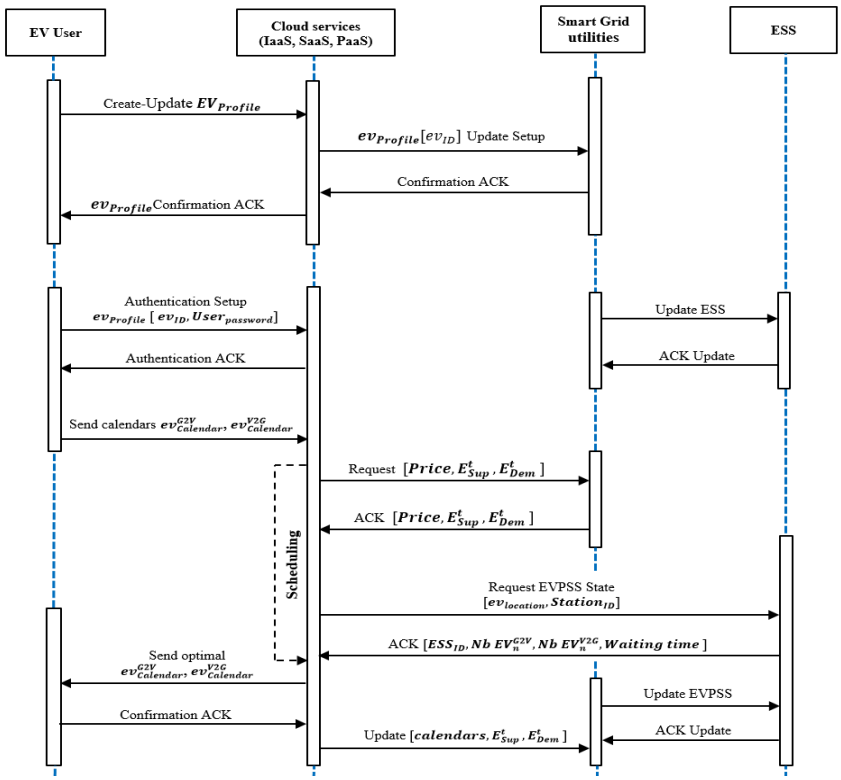


Figure 9.4- Sequence diagram of scheduling calendars

$$U_{Profile}\{User_{name}, User_{password}, ev_{ID}, Home_{Adress}, Battery_{type}, Battery_{efficiency}\}$$

Où ev_{ID} est l'identifiant unique d'un véhicule électrique, $,User_{password}$ est utilisé pour l'authentification de session afin de garantir la confidentialité et la sécurité, $Battery_{type}$ et $Battery_{efficiency}$ sont utilisés pour calculer le temps de charge et de décharge des batteries.

Les utilisateurs peuvent créer plusieurs calendriers pour exprimer plusieurs demandes de chargement et/ou de déchargement. Les calendriers d'un EV pour charger et décharger sont représentés, respectivement, comme suit :

$$ev^{G2V}_{Calendar}\ [ev_{ID}, ev_{location}, soc^t_{ev_i}, time\ to\ plugin\ t,\ A^t_i]$$

$$ev^{V2G}_{Calendar}\ [ev_{ID}, ev_{location}, soc^t_{ev_i}, time\ to\ plugin\ t,\ B^t_i]$$

Chaque véhicule électrique peut soumettre plusieurs calendriers de chargement et de déchargement, après avoir accédé à son profil dans le cloud, en indiquant son emplacement, son état de charge $soc^t_{ev_i}$, son temps estimé $time\ to\ plugin\ t$. En outre, il peut indiquer l'énergie requise pour la charge (A^t_i) et pour la décharge (B^t_i). Dans chaque centre de données fog Computing, les calendriers appropriés et optimales avec le meilleur temps d'attente pour le

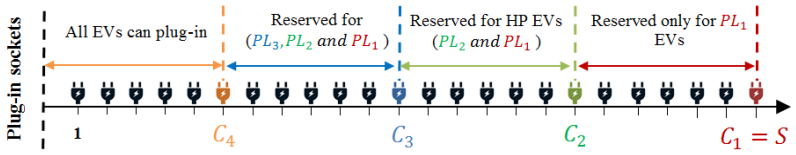


Figure 9.5- Cut-off priority mechanism for one public EPSS

plug-in et le meilleur profil de consommation d'énergie pour les utilisateurs et pour le réseau électrique intelligent seront envoyés à tous les véhicules électriques. La figure 9.4 illustre le format des paquets de messages échangés entre ainsi que le diagramme de séquence. Le diagramme illustre le processus d'échange entre les véhicules électriques, les centres de données dans le fog, EPSS et le réseau électrique intelligent pour assurer les opérations de chargement et de déchargement des véhicules électriques.

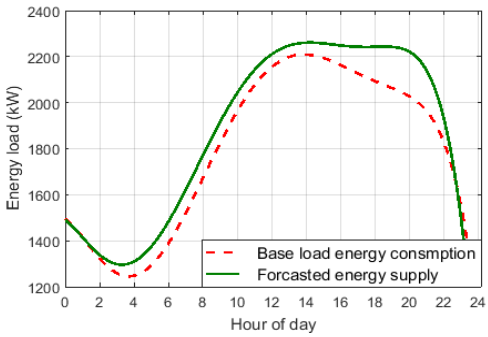
Afin de rendre notre modèle plus réaliste, nous supposons que certains utilisateurs de véhicules électriques ne peuvent pas créer de profils ou n'ont pas envoyé de calendrier pour planifier leur plug-in. Nous avons appelé ces véhicules les véhicules électriques avec un plug-in aléatoire. En résumé, nous aurons quatre types de véhicules électriques pour exécuter le processus de charge et/ou de décharge. Nous avons divisé les véhicules électriques en quatre ensembles, de la manière suivante : 1) EVs qui plug-in avec calendrier pour le chargement, 2) EVs qui plug-in avec calendrier pour le déchargement, 3) EVs qui plug-in aléatoirement pour le chargement, et 4) EVs qui plug-in aléatoirement pour le déchargement.

9.3.4 Un modèle de file d'attente multi-priorités pour le plug-in des véhicules électriques

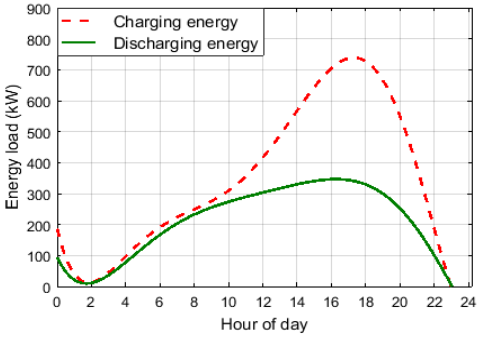
En prenant en considération les quatre ensembles de véhicules électriques, notre file d'attente avec quatre niveaux de priorité est considérée comme un modèle M/M/s à plusieurs priorités pour chaque EPSS où chaque EPSS possède un certain nombre de prises. À cet égard, notre processus stochastique est présenté comme une chaîne de Markov avec quatre priorités dans chaque transition d'état. Les niveaux de priorité $\mathcal{L}$ est donné comme suit: $\mathcal{L} = \{1, 2, 3, 4\}$. $(\mathcal{L} = 1) > (\mathcal{L} = 2) > (\mathcal{L} = 3) > (\mathcal{L} = 4)$. Par convention $\mathcal{L} = 1$ désigne la priorité la plus élevée.

9.3.5 Évaluation des performances

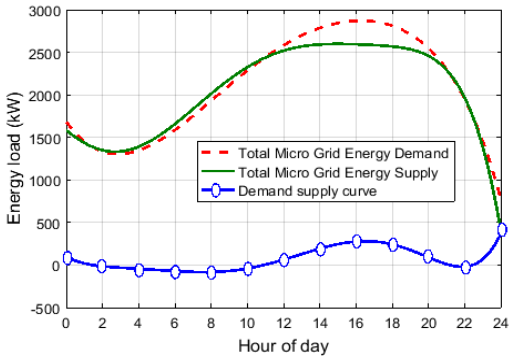
Nous effectuons des simulations approfondies pour évaluer et mettre en évidence la méthode de planification proposée. Nous utilisons MATLAB pour effectuer nos simulations et SUMO-Veins [92] pour mettre en œuvre le modèle de mobilité.



(a) Real-time supply and base load demand



(b) Real-time Charging and discharging energy



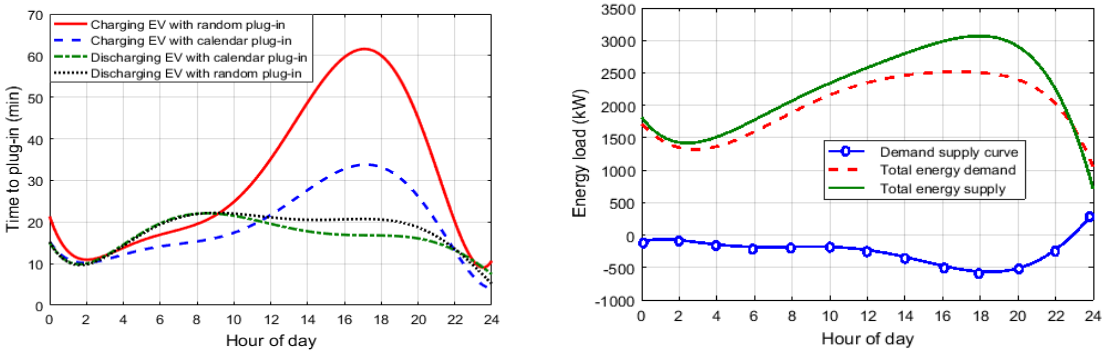
(c) Total energy load vs. supply demand curve

Figure 9.6- Real-time impact of EVs integration in the smart grid supply demand curve

9.4.1 Résultats et discussion

Trois scénarios de simulations ont été réalisés comme suit : 1) nous effectuons des simulations en considérant quatre niveaux de priorité pour les véhicules électriques lorsque nous supposons que la courbe d'offre et de la demande n'est pas stable, et 2) nous exécutons la simulation sans tenir compte de l'utilisation de cloud computing et de la stratégie de plug-in proposé.

Pour que nos scénarios de simulation restent plus réalistes, nous utilisons une consommation d'énergie réelle. La figure 9.6(a) présente la consommation de base d'électricité et la production d'électricité. La figure 9.6(b) montre l'électricité de chargement et de déchargement des EVs en temps réel. La figure 3.6(c) montre la demande totale d'énergie, la production totale d'énergie et la courbe de l'offre et de la demande après addition d'énergie de chargement et de déchargement des EVs. Nous pouvons voir que la courbe de l'offre et de la demande n'est pas stable de [12:00 pm to 21:00 pm] à cause de tous les véhicules électriques cherchent à charger ses batteries pendant cette période.



(a) Time to plug-in using EV_CPA and EV_RPA (b) Demand supply curve with scheduling

Figure 9.6- Priority attribution strategy and demand supply curve stabilization

Comme le montre la figure 9.6(a), nous pouvons observer que, de 3 h à 9 h, le temps d'attente pour décharger les véhicules électriques est supérieur au temps d'attente pour charger, car la courbe de l'offre et de la demande est stable pendant cette période. Toutefois, à partir de midi lorsque la courbe d'offre de la demande devient instable, les courbes sont inversées et le temps d'attente du plug-in pour la charge deviennent très élevés, en particulier pour les véhicules électriques avec un plug-in aléatoire. C'est pourquoi les utilisateurs sont invités à créer des calendriers de charge et de décharge afin de soulager le système de réseau électrique intelligent et de sélectionner le meilleur plan de charge/décharge au cours d'une journée, en particulier pondant les heures de pointe. Comme le montre le graphique 9.6(c), on constate une stabilisation significative de la courbe de l'offre et de la demande par rapport au cas standard ; nous avons une économie d'énergie importante (500 kWh) pendant les heures de pointe.

9.4 Gestion et traitement des données IIoT

La deuxième partie de cette thèse porte sur le problème de l'ordonnancement et de traitement des données IIoT. Pour intégrer efficacement la technologie IoT dans l'industrie, les données collectées doivent être planifiées en temps réel avec des contraintes de délai ; surtout pour les grandes usines. Dans la littérature, plusieurs solutions [63-66] ont été proposées pour aider les usines intelligentes à programmer les données et les demandes en fonction des contraintes en temps réel. Ils peuvent être de deux types : des schémas centralisés basés sur le cloud et des schémas décentralisés prenant en compte la fonctionnalité de calcul distribué. Bien qu'ils assurent un certain niveau de performance, les systèmes existants connaissent diverses limitations. Pour les systèmes centralisés, d'une part, la plupart des travaux existants se concentrent sur des architectures de calcul centralisées dans le but de contrôler les données et de gérer les processus de contrôle dans l'automatisation industrielle et se concentrent

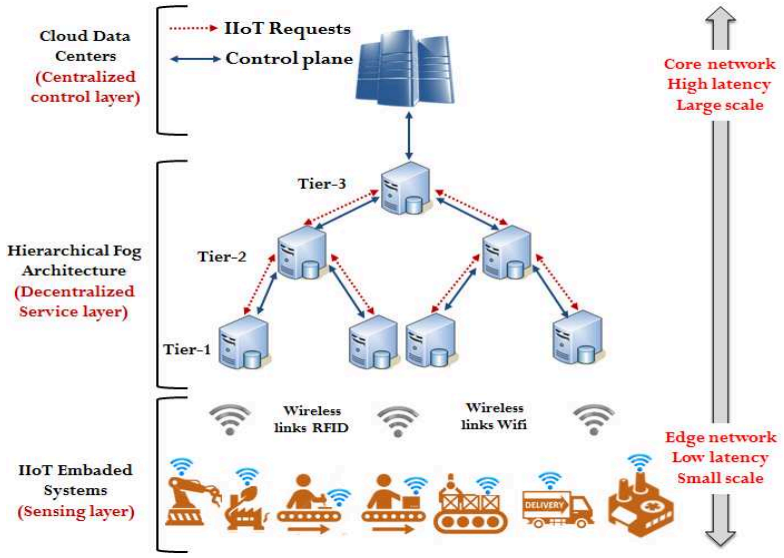


Figure 9.7- Hierarchical fog architecture for Industrial internet of things

uniquement aux niveaux supérieurs. D'autre part, les systèmes décentralisés sont peu rentables (c'est-à-dire en raison de l'utilisation de plusieurs serveurs) ; en outre, ils ignorent le délai de transmission des données et le temps de synchronisation, en particulier lors de l'utilisation du calcul distribué sur des centres de données décentralisés.

9.4.1 Modèle d'analyse probabiliste pour une architecture hiérarchique du Fog

Nous prouvons formellement l'avantage de l'utilisation d'une architecture fog hiérarchique sur l'amélioration de l'efficacité de l'utilisation des ressources du cloud lors du traitement des données et de demandes des objets connectés industriel. Nous développons d'abord un modèle analytique pour un fog hiérarchique à deux niveaux. Ensuite, les résultats d'analyse basés sur ce modèle à deux niveaux seront généralisés à une topologie plus complexe dans la hiérarchie.

9.4.2 Un modèle de file d'attente prioritaire pour la gestion de données IIoT

Nous considérons un certain nombre de centres de données de fog, où chaque centre de données contient des serveurs déployés de manière hiérarchique. Ces serveurs gèrent le processus de traitement et d'analyse des données. Les données entrantes, les demandes et les informations provenant de différents équipements industriels et capteurs ont des degrés d'importance différents, par exemple des demandes d'urgence sont des demandes hautement prioritaires qui doivent être traiter et analyser rapidement.

Nous décrivons un système de file d'attente avec deux priorités. En tant qu'un modèle stochastique, la file d'attente proposée est représentée à l'aide d'une chaîne de Markov en

incorporant des informations dans la description d'état. Nous considérons que chaque état de la chaîne de Markov correspond au nombre de demandes IIoT dans les deux files d'attente et que les transitions d'état se produisent lorsqu'une nouvelle demande IIoT arrive ou qu'une demande est servie.

9.4.3 Affectation des demandes IIoT

Nous nous concentrons sur la détermination des charges de travail pour chaque serveur dans le fog. En outre, nous définissons la capacité de calcul fournie pour exécuter chaque demande, de manière à optimiser les performances d'exécution de toutes les charges de travail et à prendre une décision optimale en temps réel pour minimiser le délai de réponse.

Nous étudions d'abord un scénario qui prend en compte un seul serveur à chaque niveau dans le fog. Ensuite, nous agrégions les décisions de déchargement de la charge de travail à différents niveaux. Nous formulons le problème de placement de charge de travail comme un problème de programmation mixte non linéaire (MNIP) d'un niveau dans le fog qui n'a qu'un serveur.

9.4.4 Évaluation des performances

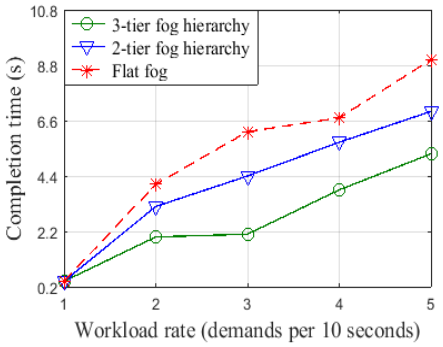
Nous mettons en œuvre de nombreuses simulations pour évaluer les performances de l'architecture proposée et nous les comparons (architecture hiérarchique, par exemple) à une architecture fog plat et aux travaux récents [64], [66].

9.4.5 Paramètres des simulations

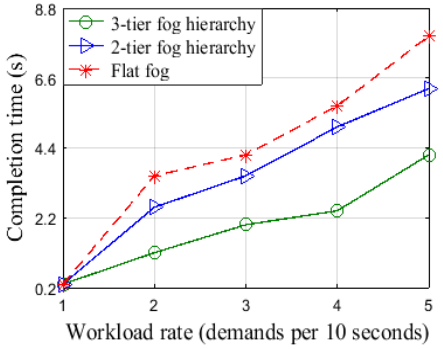
Nous utilisons plusieurs machines virtuels en tant que serveurs de fog, et chaque machine virtuel a une capacité de processeur de 1,45 GHz et 2 Go de RAM. Nous utilisons NS-2 pour implémenter nos topologies de réseau [36]. Nos expériences sont menées sur différents topologies et capacités de calcul. Chaque deux serveurs de fog sont connectés via un lien réseau à 100 Mbps. Une quantité totale de capacité de calcul de 20 GHz (exprimée en nombre de cycles de processeur par seconde) est fournie.

9.4.6 Résultats des simulations

Afin d'étudier les performances de l'architecture et du modèle proposés, nous réalisons des expériences de simulation sur les trois topologies de l'architecture de fog : une hiérarchie de à 3 niveaux, une hiérarchie à 2 niveaux et une architecture plate. Tout d'abord, nous évaluons la

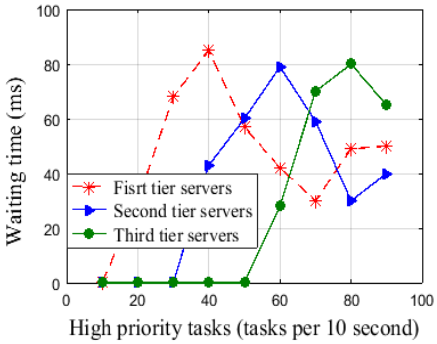


(a) provisioned capacity = 2

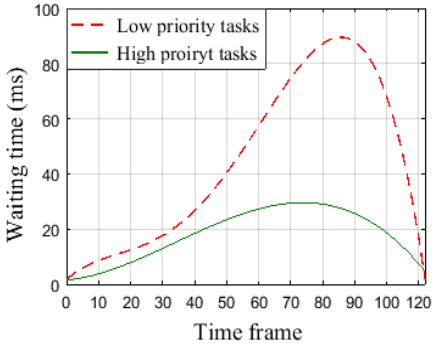


(b) provisioned capacity = 4

Figure 9.8- Average completion time over different amounts of workloads



(a) Over dififrent tiers hierarchy



(b) Over dififrent time frames

Figure 9.9- Average waiting time comparison

performance de chaque topologie en termes de temps d'exécution en fixant la capacité fournie et en modifiant le taux de charge de travail de 1 à 5 tâches chaque 10 secondes.

Les résultats sont présentés par la figure 9.8. Comme nous observons dans la figure 9.8(a), lorsque la capacité fournie est égale à 2, il existe une grande différence entre les performances de l'architecture plate et l'architecture hiérarchique ; la hiérarchie à 3 niveaux fonctionne mieux et donne un temps d'exécution optimal. Rationnellement, lorsque la capacité fournie est de 4, le temps d'exécution est plus optimal, comme le montre la figure 9.8(b). La principale raison de cette différence est que le serveur de niveau 3 est capable d'agréger efficacement les pics de charge de tous les serveurs de niveau 2. Par conséquent, chaque niveau hiérarchique utilise la capacité fournie pour traiter les tâches avec une efficacité bien supérieure.

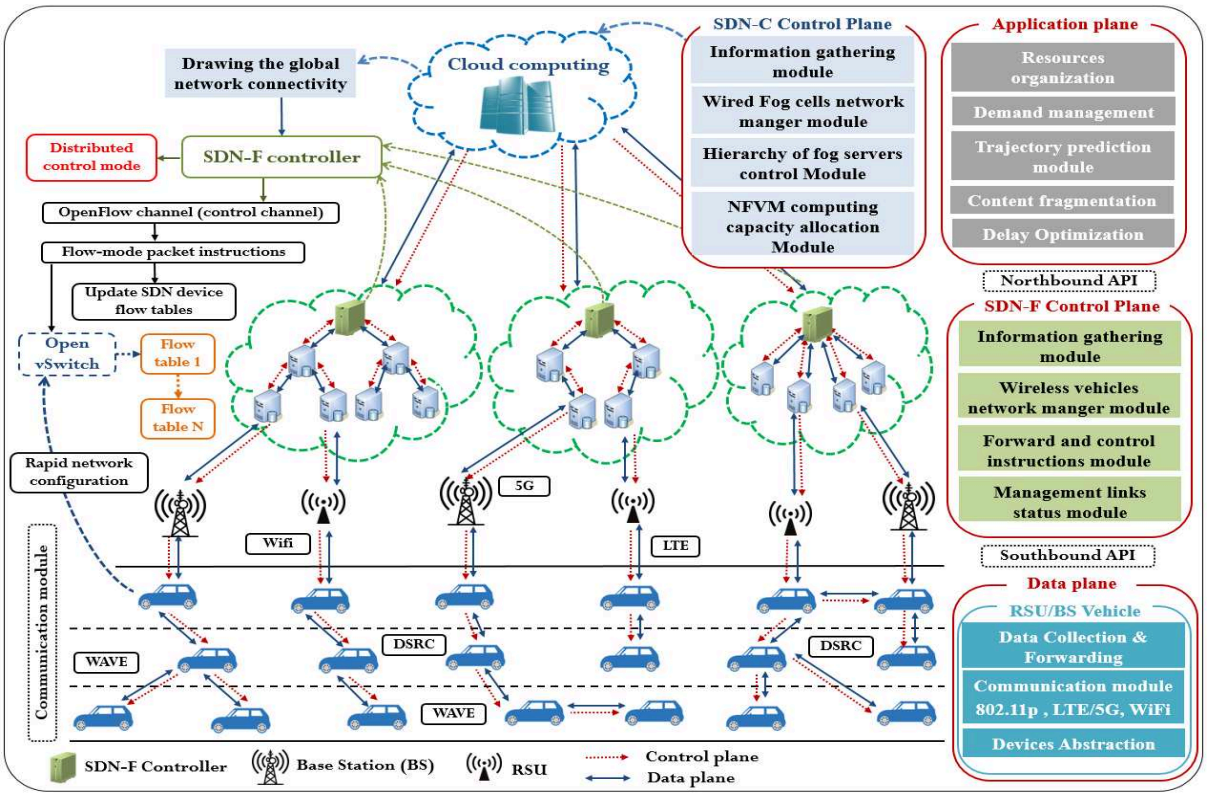


Figure 9.10- HSVF system architecture.

9.5 Une architecture SDN hiérarchique pour le réseau véhiculaire

Bien que l'application du réseau SDN dans un réseau de véhicules hétérogène pose des difficultés, plusieurs architectures (par exemple [67], [68], [71], [72], [73]) ont été proposées dans la littérature. Ces études décrivent des architectures de réseau pouvant être classées en deux catégories : centralisée et distribuée. Malgré leurs performances adéquates, les deux catégories ont des limites. D'une part, l'utilisation d'un seul contrôleur SDN pourrait constituer un goulot d'étranglement dans l'ensemble de réseau. D'autre part, le déploiement de plusieurs contrôleurs SDN accroît la complexité du système et introduit de nouveaux défis qui doivent être traités. Cela peut entraîner une surcharge de contrôle [71, 72] et un délai important de bout en bout [73].

La dernière partie de cette thèse aborde le développement d'une nouvelle architecture pour les VANETs afin de permettre une prise de décision rapide et un délai de réponse optimal.

- HSVF : Une architecture SDN hiérarchique pour les VANET basée sur le fog décentralisé. HSVF est basé sur un SDN hybride qui utilise une gestion centralisée et décentralisé en même temps. L'architecture proposée est ensuite évaluée à l'aide d'une étude de cas

pertinente traitant la planification des demandes de chargement et/ou déchargement des véhicules électriques.

9.5.1 HSVF : Une architecture SDN hiérarchique pour les VANETs

Le SDN ainsi que les architectures décentralisées du cloud (c.-à-d., le fog) ont récemment été explorés pour gérer des tâches spécifiques et des problèmes liés aux réseaux véhiculaires. En associant ces technologies avec le concept de virtualisation, nous proposons HSVF, comme illustré à la figure 9.10. De bas en haut, le HSVF est divisé en trois plans : le plan de données, le plan de contrôle et le plan d'application et de service.

- **Le plan de données :** Le plan de données comprend tous les périphériques du réseau de transmission. Les fonctions du plan de données sont axées sur la collecte, la quantification et le transfert de données vers le plan de contrôle. En HSVF, le plan de données est construit en utilisant un réseau de fog. Tous les périphériques sont résumés en tant que switch équipé par Open vSwitch. Nous supposons que les périphériques SDN prennent en charge la virtualisation et peuvent héberger des machines virtuelles (VM) pour pouvoir installer des mises à jour logicielles. Ces appareils peuvent communiquer avec des serveurs, situés dans le fog, pour fournir différents services.
- **Le module de communication :** Le module de communication proposé dans HSVF comprend les sous-modules de communication V2V, V2I et I2I. Le sous-module de communication V2V assure une communication sans fil entre les véhicules adjacents en utilisant le standard WAVE (IEEE 802.11p). La communication V2I comprend deux types de communication : les V2RSU basées sur IEEE 802.11p et les V2BS, où les stations de base transmettent des signaux sans fil utilisant des fréquences DSRC pour offrir une large couverture aux véhicules. Les BSs disposent également de connexions LTE pour permettre aux RSU de transmettre leurs informations de connectivité à leur contrôleur SDN-F respectif. Étant donné que tous les véhicules se déplacent de manière ordonnée, on peut supposer qu'un groupe de véhicules est une unité de communication dans une cellule dans le fog.
- **Le plan de contrôle SDN-F :** Les SDN-F sont déployés sur des serveurs spécifiques de chaque centre de données dans le fog. Certains SDN-F sont directement connectés à des RSUs, des BSs et des véhicules dans leur région. Tous les centres de données sur le fog sont connectés à un centre de données sur le cloud centralisé équipé d'un contrôleur SDN-Cloud

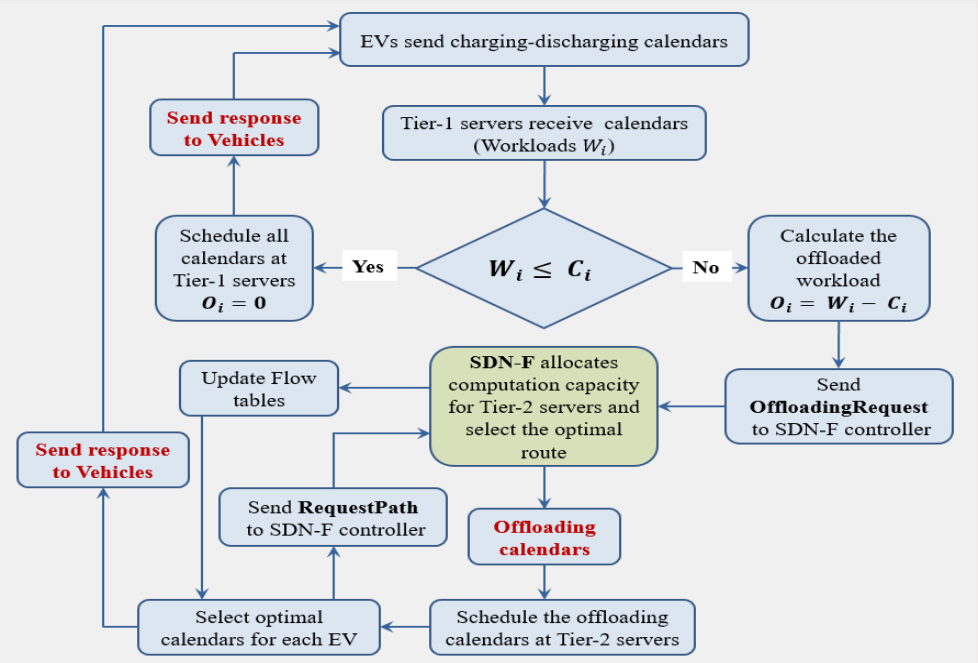


Figure 9.11- Offloading process diagram based on SDN-F controller

(SDN-C) qui gère l'ensemble des réseaux de fog. Le plan de contrôle de chaque contrôleur SDN-F maintient l'état de tous les switches dans sa zone et il est responsable des instructions de transmission de paquets sur la base des informations reçues du SDN-C. Compte tenu de la grande mobilité et du trafic massif entre les RSUs et les véhicules, des transferts de données fréquents peuvent avoir lieu. Pour résoudre ce problème, HSVF déploie un plan de contrôle distribué sur différentes cellules de fog.

- **Le plan de contrôle SDN-C :** Le SDN-C est responsable de la construction de la connectivité globale du réseau en agrégeant les informations reçues de tous les SDN-F et en générant des informations de contrôle sur la base de règles et de stratégies issues du plan d'application.
- **Le plan d'applications :** En fonction des besoins des applications, les règles et les stratégies de HSVF sont générées par SDN-C pour les SDN-F et par les SDN-F pour les réseaux sans fil d'accès. Le plan d'application contient un ensemble d'applications et de services tels que l'organisation des ressources, la gestion des demandes, l'optimisation des délais, et la prédiction de trajectoire. Le plan d'application interagit avec chaque SDN-F pour transmettre les informations requises au plan de contrôle.

9.5.2 Étude de cas : planification des demandes énergétiques des véhicules électriques

Au fur et à mesure que les véhicules électriques gagnent en popularité, de nombreuses stations publiques pour le chargement des batteries (EPSS) sont mises à disposition dans de nombreux espaces publics. Cependant, cette situation soulève de multiples préoccupations. L'un des problèmes les plus difficiles est de savoir comment gérer correctement toutes les demandes de EV. Pour résoudre ce problème, il convient de répondre à plusieurs questions, notamment : à quel point sera-t-il difficile pour un conducteur d'un EV de trouver une borne de recharge ? Les demandes de chargement peuvent-elles être traiter en temps réel ? L'architecture de communication peut-elle prendre en charge la mobilité des véhicules électriques en collectant des informations, en maintenant la connectivité des véhicules et en garantissant leur fiabilité ? Concevoir une planification optimale pour le traitement des demandes des EV en utilisant une architecture de communication adéquate est la réponse à toutes ces questions.

9.5.3 Un processus d'offloading optimal basé sur le contrôleur SDN-F

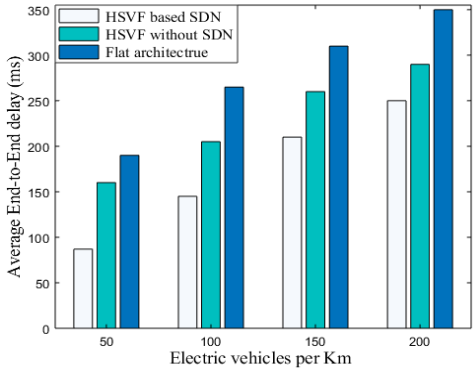
Comme le montre la figure 9.11, le processus d'offloading est lié à la capacité des serveurs de niveau 1. Dans le cas contraire, certaines tâches seront « *offloadées* » vers les niveaux supérieurs de la hiérarchie. La figure 9.11 illustre le processus d'offloading ainsi que la synchronisation entre les demandes des EV et le contrôleur SDN-F.

9.5.4 Evaluation des performances de HSVF

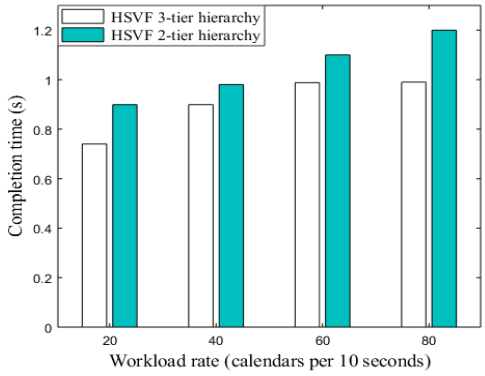
Pour évaluer les performances du HSVF, nous avons utilisé NS-2 et SUMO pour simuler le réseau véhiculaire. Nous avons également utilisé Mininet 2.1.0 avec OpenFlow 1.3 pour le plan de données SDN, car il est plus évolutif et fournit un environnement réaliste pour les tests. Nous avons mis en place un contrôleur distribué basé sur POX.

La topologie routière a été obtenue à l'aide d'OpenStreetMap et a été formatée et convertie en un fichier réseau SUMO. En utilisant SUMO, des traces de mobilité des véhicules ont été générées et utilisées pour peupler la zone choisie. Nous avons également déployé des RSU et des stations de base LTE avec une distribution uniforme. Nous avons varié le nombre moyen de véhicules entre 50 et 200 par kilomètre.

Nous cherchons à évaluer deux performances : 1) la performance de HSVF lors de l'utilisation de SDN-F ; et 2) le processus d'offloading des calendriers des EVs lors de l'utilisation du déploiement hiérarchique des serveurs dans le fog.



(a) Average end-to-end delay



(b) Completion time

Figure 9.12- Simulation results comparisons

9.5.5 Résultats des simulations

La figure 9.12(a) montre le délai moyen de bout en bout en fonction de la densité des véhicules. Nous observons que HSVF-F surpasse HSVF-W (sans SDN) et l'architecture plate. En effet, HSVF-F génère un temps de réponse moyen de bout en bout inférieur de 25% et 40%. On peut expliquer ça par : 1) HSVF-W utilise un seul contrôleur (SDN-C) pour gérer l'ensemble du réseau. Quand le nombre de véhicules augmente, les ressources du contrôleur SDN-C ne seront pas suffisantes pour gérer le nombre élevé de demandes de véhicules électriques, ce qui entraîne un délai de bout en bout élevé. Contrairement à HSVF-W, HSVF-F partitionne le réseau en sous-réseaux (c'est-à-dire des cellules fog), chacun d'entre eux étant géré par un contrôleur SDN-F. et 2) à l'aide du SDN-F, le HSVF peut mettre à jour les tables de flux périodiquement pour refléter les changements de topologie.

La figure 9.12(b) montre le temps d'exécution de différents niveaux de l'architecture fog en fonction du taux de charge de travail. Nous observons que la hiérarchie à 3 niveaux devance la hiérarchie à 2 niveaux pour tous les taux de charge de travail. Cette hiérarchie à 3 niveaux peut

efficacement « *offloader* » les charges des serveurs de niveau inférieur sur des serveurs de niveau supérieur en utilisant le processus d'offloading décrit dans la figure 9.11.

9.6 Conclusion et perspectives

Le Cloud computing a attiré l'attention de différents gouvernements, industriels et institutions universitaires, car il offre un potentiel énorme de nouvelles opportunités commerciales en termes d'efficacité du traitement, de capacité de stockage et de disponibilité. Cependant, il présent des défis particuliers qui entravent son application pour des nouvelles technologies et systèmes émergents. L'objectif principal de ce projet de recherche est de concevoir de nouvelles architectures et modèles du cloud computing pour : 1) la gestion de l'énergie d'un réseau électrique intelligent, 2) l'internet industriel des objets, et 3) et les réseaux véhiculaires basés sur les technologies SDN et 5G. Pour atteindre ces objectifs, cette thèse a présenté des architectures et des modèles efficaces et sûrs basés sur les technologies Cloud.

Au-delà des architectures et des modèles proposés dans ce projet de recherche, diverses opportunités futures sont présentées, avec quelques défis en matière de recherche.

La mise en place d'un réseaux électrique intelligent pose plusieurs problèmes de recherche tout en intégrant les applications du cloud computing. Premièrement, en raison de défaillances des protocoles utilisés pour la communication, l'ensemble du système peut être affecter. Ainsi, le Cloud peut être utilisé comme solution. En outre, un cloud basé sur un entrepôt de données peut être mis en œuvre afin d'optimiser, de contrôler et de sécuriser les communications de réseau intelligent. Deuxièmement, l'intégration du cloud public à l'infrastructure du cloud privé pour une communication rentable dans un réseau électrique intelligent est indispensable pour les nouveaux réseaux de villes intelligentes évolutives. Le Cloud public peut être intégré au Cloud privé pour un développement à grande échelle dans un réseau électrique intelligent, de manière rentable. Cependant, la sécurité et la confidentialité sont deux des problèmes importants, tout en permettant l'échange d'informations entre les Clouds privés et publics. Troisièmement, en raison de la nature hétérogène du réseau électrique intelligent lors de l'intégration du cloud computing, des agents mobiles peuvent être utilisés pour communiquer avec différents éléments de l'architecture. Par conséquent, il serait nécessaire de définir une stratégie de communication adéquate pour l'agent afin de gérer les informations de manière rentable dans le réseau électrique intelligent.

BIBLIOGRAPHIE

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: Towards a cloud definition,” ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 50–55, Jan. 2009.

[2] S. Bera, S. Misra and J. J. P. C. Rodrigues, "Cloud Computing Applications for Smart Grid: A Survey," in IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 5, pp. 1477-1494, 1 May 2015.

[3] S. Misra, P. Krishna, K. Kalaiselvan, V. Saritha, and M. S. Obaidat, “Learning automata-based QoS framework for cloud IaaS,” IEEE Trans. Netw. Serv. Manage., vol. 11, no. 1, pp. 15–24, Mar. 2014.

[4] S. Misra, S. Das, M. Khatua, and S. M. Obaidat, “QoS-guaranteed bandwidth shifting and redistribution in mobile cloud environment,” IEEE Trans. Cloud Comput., 2013.

[5] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam, “Cloudy with a chance of cost savings,” IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 6, pp. 1223–1233, Jun. 2013.

[6] F. Li, W. Qiao, H. Sun, H. Wan, J. Wang, Y. Xia, Z. Xu, and P. Zhang, “Smart transmission grid: Vision and framework,” IEEE Trans. Smart Grid, vol. 1, no. 2, pp. 168–177, Aug. 2010.

[7] S. Misra, P. Krishna, V. Saritha, and M. S. Obaidat, “Learning automata as a utility for power management in smart grids,” IEEE Comm. Mag., vol. 51, no. 1, pp. 98–104, 2013.

[8] I. Atzeni, L. G. Ordonez, G. Scutari, D. P. Palomar, and J. R. Fonollosa, “Demand-side management via distributed energy generation and storage optimization,” IEEE Trans. Smart Grid, vol. 4, no. 2, pp. 866–876, Jun. 2012.

[9] D. Von Dollen, Report to nist on the smart grid interoperability standards roadmap, in: Prepared by the Electric Power Research Institute for NIST, June 2009.

[10] R. C. Green II, L. Wang, and M. Alam, “Applications and trends of high performance computing for electric power systemsfocusing on smart grid,” IEEE Trans. Smart Grid, vol. 4, no. 2, pp. 922–931, Jun. 2013.

[11] J. Popeang, “Cloud computing and smart grids” ERP E-Business Appl. Deployment Open Source Distrib. Cloud Syst. III (2012), pp. 57-66.

[12] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, “Cyber security and privacy issues in smart grids,” IEEE Comm. Surv. Tut., vol. 14, no. 4, pp. 981–997, 2012.

[13] Y. Zhang et al., “Securing vehicle-to-grid communications in the smart grid,” IEEE Wireless Commun., vol. 20, no. 6, pp. 66–73, Dec. 2013.

[14] Hackers Caused Power Cut in Western Ukraine, BBC News, London, U.K., 2016.

[15] P. Li, Y. Liu, H. Xin and X. Jiang, "A Robust Distributed Economic Dispatch Strategy of Virtual Power Plant Under Cyber-Attacks," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4343-4352, Oct. 2018.

[16] The Smart Grid and Privacy, Electron. Privacy Inf. Center, Washington, DC, USA, 2015.

[17] Y. Liu, P. Ning, and M. K. Reiter, “False data injection attacks against state estimation in electric power grids,” ACM Trans. Inf. Syst. Security, vol. 14, no. 1, May 2011, Art. no. 13.

[18] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” IEEE Comm. Surv. Tutor.,vol. 17, no. 4, pp. 2347–2376, 2015.

[19] T. Qiu, X. Liu, L. Feng, and Y. Zhou, “An efficient tree-based self-organizing protocol for Internet of Things,” IEEE Access, vol. 4, no. 6, pp. 3535–3546, Jun. 2016.

[20] P. Lade, R. Ghosh, and S. Srinivasan, “Manufacturing Analytics and Industrial Internet of Things,” IEEE Intel. Sys. vol. 32, pp. 74 79, 2017.

[21] C. Alippi, R. Fantacci, D. Marabissi and M. Roveri, "A Cloud to the Ground: The New Frontier of Intelligent and Autonomous Networks of Things," in IEEE Commun. Mag., vol. 54, no. 12, pp. 14-20, Dec. 2016.

[22] D. A. Chekired and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," in IEEE Trans. on Indus. Inform., vol. 13, no. 6, pp. 3312-3321, Dec. 2017.

[23] Cisco white paper, “Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are,” accessed on Nov. 2017. [Online]. Available:https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[24] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold, K. Sandström, and M. Behnam, “Delay Mitigation in Offloaded Cloud Controllers in Industrial IoT,” In IEEE Access, vol. 5, pp. 4418-30, April 2017.

[25] A. Mai and D. Schlesinger. Connected Vehicles: Service Providers at a Crossroads, accessed on Octo 2017. [Online]. Available: http://www.cisco.com/c/dam/en_us/about/ac79/docs/mfg/ConnectedVehicles_Exec_Summary.pdf

[26] 5G Automotive Association, “The Case for Cellular V2X for Safety and

[27] Cooperative Driving”, [Online]. Available: http://5gaa.org/wp-content/uploads/2017/08/5GAA-whitepaper-23-Nov-2016.pdf.

[28] M. Nekovee, “Radio Technologies for Spectrum above 6 GHz — A Key Component of 5G”, Proc. 5G Radio Tech. Exploring Technical Challenges in the Emerging 5G Ecosystem, IET, 2015, pp. 1–46.

[29] Yaqoob, I., I. Ahmad, E. Ahmed, A. Gani, M. Imran, and N. Guizani. “Overcoming the Key Challenges to Establishing Vehicular Communication: Is SDN the Answer?” IEEE Commun. Mag. 55, no. 7, 2017, pp. 128–34.

[30] He, Z., J. Cao, and X. Liu. “SDVN: Enabling Rapid Network Innovation for Heterogeneous Vehicular Communication.” IEEE Network 30, no. 4, July 2016, pp.10–15.

[31] 5G-PPP, ERTICO, EFFRA, EUTC, NEM, CONTINUA and Networld2020 ETP, “5G Empowering Vertical Industries,” white paper, Feb. 2016.

[32] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," in IEEE Communications Magazine, vol. 55, no. 5, pp. 80-87, May 2017.

[33] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," in IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429-2453, thirdquarter 2018.

[34] Next Generation Mobile Networks Alliance, "5G white paper," White paper, vol. 1.0, Feb. 2015.

[35] ONF TR-526, "Applying SDN Architecture to 5G Slicing", Issue 1, Apr. 2016.

[36] D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Decentralized Cloud-SDN Architecture in Smart Grid: A Dynamic Pricing Model," in IEEE Transactions on Industrial Informatics, vol. 14, no. 3, pp. 1220-1231, March 2018.

[37] D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4590-4602, Oct. 2018.

[38] Y., B. Venkatesh, and L. Guan. "Optimal Scheduling for Charging and Discharging of Electric Vehicles," IEEE Trans. on Smart Grid, Vol. 3, no. 3, pp. 1095-1105, September 2012.

[39] L. Jian, H. Xue, G. Xu, X. Zhu, D. Zhao, and Z. Y. Shao. "Regulated Charging of Plug-in Hybrid Electric Vehicles for Minimizing Load Variance in Household Smart Microgrid," IEEE Trans. Ind. Electron., vol. 60, no. 8, August 2013.

[40] M. Tushar, H. K., C. Assi, M. Maier, and M. F. Uddin. "Smart Microgrids: Optimal Joint Scheduling for Electric Vehicles and Home Appliances," IEEE Trans. Smart Grid, vol 5, no. 1, January 2014.

[41] A. Sharma, S. Shih, and D. Srinivasan. "A Smart Scheduling Strategy for Charging and Discharging of Electric Vehicles," In IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA), 2015, 1–6, 2015.

[42] I. S. Bayram, A. Tajer, M. Abdallah and K. Qaraqe, "Capacity Planning Frameworks for Electric Vehicle Charging Stations With Multiclass Customers," in IEEE Trans. on Smart Grid, vol. 6, no. 4, pp. 1934-1943, July 2015.

[43] F. Luo, J. Zhao, Z. Y. Dong, Y. Chen, Y. Xu, X. Zhang, and K. P. Wong. "Cloud-Based Information Infrastructure for Next-Generation Power Grid: Conception, Architecture, and Applications," IEEE Trans. on Smart Grid 7, no. 4 (July 2016): 1896–1912.

[44] H. Xing, M. Fu, Z. Lin, and Y. Mou. "Decentralized Optimal Scheduling for Charging and Discharging of Plug-In Electric Vehicles in Smart Grids." IEEE Trans. Pow. Sys,. Vol. 31, no. 5, September 2016.

[45] L. Gan, U. Topcu, and S. H. Low. "Optimal Decentralized Protocol for Electric Vehicle Charging." IEEE Trans. Pow. Sys., 28, no.2, May 2013.

[46] M. Erol-Kantarci, et T. M. Hussein, "Prediction-based charging of PHEVs from the smart grid with dynamic pricing", In 2010 IEEE 35th Conference on Local Computer Networks (LCN), 1032 39, 2010

[47] S. Misra, S. Bera and T. Ojha, "D2P: Distributed Dynamic Pricing Policyin Smart Grid for PHEVs Management", IEEE Trans. on Para. and Distr. Sys. 26, no 3 (mars 2015): 702 12.

[48] X. Liang, X. Li, R. Lu, X. Lin, et X. Shen, "UDP: Usage-Based Dynamic Pricing With Privacy Preservation for Smart Grid", IEEE Trans on Smart Grid 4, no 1 (mars 2013): 141 50.

[49] J. H. Park, Y. S. Kim, I. K. Eom, and K. Y. Lee, “Economic load dis-patch for piecewise quadratic cost function using Hopfield neuralnetwork,” IEEE Trans. Pow. Sys., vol. 8, no. 3, pp. 1030–1038,Aug. 1993.

[50] Z. Fan, “A distributed demand response algorithm and its applicationto PHEV charging in smart grids,” IEEE Trans. Smart Grid,vol. 3, no. 3, pp. 1280–1290, Sep. 2011

[51] N. Ruiz, I. Cobelo, and J. Oyarzabal, “A direct load control model forvirtual power plant management,” IEEE Trans. Power Syst., vol. 24, no. 2, pp. 959–966, 2009.

[52] Hydro One [Online]. Available: http://www.hydroone.com/Pages/Default.aspx.

[53] Waterloo North Hydro [Online]. Available: http://www.wnhydro.com/

[54] Y. Guo, M. Pan, and Y. Fang, “Optimal power management of residential customers in the smart grid,” IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 9, pp. 1593–1606, Jan. 2012.

[55] A. Cahn, J. Hoyos, M. Hulse, and E. Keller, “SDN Energy Communication Networks: From Substation Automation to Fitire Smart Grids,” IEEE Intern. Conf. on Smart Grid Comm., Toronto, Canada, 2013, pp. 558-563.

[56] J. Ni, K. Zhang, K. Alharbi, X. Lin, N. Zhang and X. S. Shen, "Differentially Private Smart Metering With Fault Tolerance and Range-Based Filtering," in IEEE Transactions on Smart Grid, vol. 8, no. 5, pp. 2483-2493, Sept. 2017.

[57] H. Bao and R. Lu, "A New Differentially Private Data Aggregation With Fault Tolerance for Smart Grid Communications," in IEEE Internet of Things Journal, vol. 2, no. 3, pp. 248-258, June 2015.

[58] J. Won, C. Y. T. Ma, D. K. Y. Yau and N. S. V. Rao, "Privacy-Assured Aggregation Protocol for Smart Metering: A Proactive Fault-Tolerant Approach," in IEEE/ACM Transactions on Networking, vol. 24, no. 3, pp. 1661-1674, June 2016.

[59] L. Lyu, K. Nandakumar, B. Rubinstein, J. Jin, J. Bedo and M. Palaniswami, "PPFA: Privacy Preserving Fog-Enabled Aggregation in Smart Grid," in IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3733-3744, Aug. 2018.

[60] X. Liu, P. Zhu, Y. Zhang and K. Chen, "A Collaborative Intrusion Detection Mechanism Against False Data Injection Attack in Advanced Metering Infrastructure," in IEEE Transactions on Smart Grid, vol. 6, no. 5, pp. 2435-2443, Sept. 2015.

[61] P. Jokar and V. C. M. Leung, "Intrusion Detection and Prevention for ZigBee-Based Home Area Networks in Smart Grids," in IEEE Transactions on Smart Grid, vol. 9, no. 3, pp. 1800-1811, May 2018.

[62] Y. Zhang, L. Wang, W. Sun, R. C. Green II and M. Alam, "Distributed Intrusion Detection System in a Multi-Layer Network Architecture of Smart Grids," in IEEE Transactions on Smart Grid, vol. 2, no. 4, pp. 796-808, Dec. 2011.

[63] L. D. Xu, W. He, and S. Li, “Internet of Things in Industries: A Survey, “In IEEE Trans. on Indus. Inform., vol. 10, pp. 2233 43, Nov. 2014.

[64] L. Zhou, D. Wu, J. Chen and Z. Dong, "When Computation Hugs Intelligence: Content-Aware Data Processing for Industrial IoT," in IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1-1.

[65] Y. Sahni, J. Cao, S. Zhang and L. Yang, "Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things," in IEEE Access, vol. 5, no. , pp. 16441-16458, 2017.

[66] J. Fu, Y. Liu, H. C. Chao, B. Bhargava and Z. Zhang, "Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing," in IEEE Transactions on Industrial Informatics, vol. PP, no. 99, pp. 1-1. doi: 10.1109/TII.2018.2793350.

[67] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro and E. Cerqueira, "Towards software-defined VANET: Architecture and services," 2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), Piran, 2014, pp. 103-110.

[68] M. Azizian, S. Cherkaoui and A. S. Hafid, "Vehicle Software Updates Distribution with SDN and Cloud Computing," in IEEE Commun. Mag., vol. 55, no. 8, 2017, pp. 74-79.

[69] M. Abolhasan, J. Lipman, W. Ni and B. Hagelstein, "Software-defined wireless networking: centralized, distributed, or hybrid?," in IEEE Network, vol. 29, no. 4, July-August 2015, pp. 32-38.

[70] Q. Yang, B. Zhu and S. Wu, "An Architecture of Cloud-Assisted Information Dissemination in Vehicular Networks," in IEEE Access, vol. 4, 2016, pp. 2764-2770.

[71] S. Correia, A. Boukerche and R. I. Meneguette, "An Architecture for Hierarchical Software-Defined Vehicular Networks," in IEEE Commun. Mag., vol. 55, no. 7, 2017, pp. 80-86.

[72] X. Ge, Z. Li and S. Li, "5G Software Defined Vehicular Networks," in IEEE Commun. Mag., vol. 55, no. 7, 2017, pp. 87-93.

[73] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song and M. Qiu, "A Scalable and Quick-Response Software Defined Vehicular Network Assisted by Mobile Edge Computing," in IEEE Commun. Mag., vol. 55, no. 7, 2017, pp. 94-100.

[74] S. Costanzo, I. Fajjari, N. Aitsaadi and R. Langar, "A network slicing prototype for a flexible cloud radio access network," 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2018, pp. 1-4.

[75] L. Velasco et al., "An architecture to support autonomic slice networking," in Journal of Lightwave Technology, vol. 36, no. 1, pp. 135-141, 1 Jan.1, 2018.

[76] L. Ma, X. Wen, L. Wang, Z. Lu and R. Knopp, "An SDN/NFV based framework for management and deployment of service based 5G core network," in China Communications, vol. 15, no. 10, pp. 86-98, Oct. 2018.

[77] S. Kukliński and L. Tomaszewski, "DASMO: A scalable approach to network slices management and orchestration," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, 2018, pp. 1-6.

[78] ETSI, “Network Functions Virtualisation (NFV); Management and Orchestration” ETSI GS NFV-MAN 001 V1.1.1 (2014-12).

[79] FP7-4WARD project, “D-4.2 In-Network Management Concept” (2009-03).

[80] S. O. Oladejo and O. E. Falowo, "5G network slicing: A multi-tenancy scenario," 2017 Global Wireless Summit (GWS), Cape Town, 2017, pp. 88-92.

[81] T. Soenen, R. Banerjee, W. Tavernier, D. Colle and M. Pickavet, "Demystifying network slicing: From theory to practice," 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, 2017, pp. 1115-1120.

[82] H. Khan, P. Luoto, M. Bennis and M. Latva-aho, "On the Application of Network Slicing for 5G-V2X," European Wireless 2018; 24th European Wireless Conference, Catania, Italy, 2018, pp. 1-6.

[83] C. Campolo, A. Molinaro, A. Iera and F. Menichella, "5G Network Slicing for Vehicle-to-Everything Services," in IEEE Wireless Communications, vol. 24, no. 6, pp. 38-45, Dec. 2017.

[84] C. Campolo, A. Molinaro, A. Iera, R. R. Fontes and C. E. Rothenberg, "Towards 5G Network Slicing for the V2X Ecosystem," 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, 2018, pp. 400-405.

[85] F. Bai and A. Helmy. "A survey of mobility models," in Wireless Ad-Hoc Networks. Available: Book Chapter in "Wireless Ad hoc and Sensor Networks" Kluwer Academic, Publishers, 2004.

[86] B. Liang, Z. J. Haas, "Predictive Distance-Based Mobility Management for PCS Networks, in Proceedings of IEEE Information Communications Conference (INFOCOM '99), Apr. 1999.

[87] W. Yuan, J. Huang, Y. J. A. Zhang, "Competitive Charging Station Pricing for Plug-In Electric Vehicles," IEEE Transactions on Smart Grid, vol. 8, no. 2, March 2017.

[88] F. Ye et al., "Reliable Energy-Efficient Uplink Transmission for Neighborhood Area Networks In Smart Grid," IEEE Transactions on Smart Grid, vol. 6, no. 5, pp. 2179–88, Sept 2015.

[89] H. Peng, D. Li, K. Abboud, H. Zhou, H. Zhao, W. Zhuang, X. Shen, "Performance Analysis of IEEE 802.11p DCF for Multiplatooning Communications With Autonomous Vehicles," in IEEE Transactions on Vehicular Technology, vol. 66, no. 3, pp. 2485-2498, March 2017.

[90] C. Schaack, R. larson, "An N server cutoff priority queue", Massachusetts Institute of Technology, operations research center, 247-266, 1985.

[91] Y. Xia, D. N. C. Tse, "On the Large Deviations of Resequencing Queue Size: 2-M/M/1 Case", IEEE Tran. on Info. Theo. 54, nº 9, septembre 2008.

[92] Veins Vehicle in Network Simulations: The open source vehicular network simulation framework, Available: http://veins.car2x.org/

[93] I. S. Bayram, and I. Papapanagiotou, "A survey on communication technologies and requirements for internet of electric vehicles," EURASIP Jour. Wireless Comm. and Netw., no. 223, December 2014.

[94] Z. Ma, D. Callaway, and I. Hiskens, "Decentralized charging control for large populations of plug-in electric vehicles," in Proc. 49th IEEE Conf. Decis. Control, 2010, pp. 206–212.

[95] N. Saputro, K. akkaya, and S. Uludag, "A survey of routing protocols for smart grid communications," Comput. Netw., vol. 56, pp. 24742–2771, 2012.

[96] V. C. Gungor et al.., "Smart grid technologies: Communication technologies and standards," IEEE Trans. Ind. Informat., vol. 7, no. 4, pp. 529–539, Nov. 2011.

[97] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," IEEE Commun. Surveys Tut., vol. 18, no. 1, pp. 602–622, Jan. Mar. 2016.

[98] Powerwall, The Tesla Home Battery. Available: https://www.tesla.com/en_GB/powerwall.

[99] Ontario Energy Board, Time-of-use (TOU) Prices. Available: http://www.ontarioenergyboard.ca/OEB/Consumers/Electricity/Electricity+Prices.

[100] U.S. Electric System Operating Data, "U.S. Energy Information Administration", Available online: https://www.eia.gov/realtime_grid.

[101] M. H. Yaghmaee Moghaddam and A. Leon-Garcia, "A Fog-Based Internet of Energy Architecture for Transactive Energy Management Systems," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1055-1069, April 2018.

[102] S. Tripathi and S. De, "An Efficient Data Characterization and Reduction Scheme for Smart Metering Infrastructure," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4300-4308, Oct. 2018.

[103] N.U.Prabhu, “Fondations of Queuing Theory”, Kulmar Academic Publishers, Massachusetts, 80-112, 1997.

[104] S. Boyd, and L. vandenberghe, “Convex optimization”. Cambridge university press, 2009.

[105] J. C. Bazin, H. Li, I. S. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi, “A Branch-and-Bound Approach to Correspondence and Grouping Problems,” In IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 35, no. 7, pp. 1565 76, Jul. 2013.

[106] System Architecture for the 5G System, document TR 23.501, v1.4.0, 3GPP, 2017.

[107] “5G and e-Health,” 5G-PPP, White Paper, Feb. 2015.

[108] K. Maurice, M. Lebourges, and S. Zuyev, “Stochastic geometry and architecture of communication networks,” telecommunication systems, vol. 7, no. 1 pp. 209-227, 1997.

[109] K. Zheng, Q. Zheng, H. Yang, L. Zhao, L. Hou and P. Chatzimisios, "Reliable and efficient autonomous driving: the need for heterogeneous vehicular networks," in IEEE Communications Magazine, vol. 53, no. 12, pp. 72-79, Dec. 2015.

[110] O. Galinina, A. Pyattaev, S. Andreev, M. Dohler, and Y. Koucheryavy, “5G multi-RAT LTE-WiFi ultra-dense small cells: Performance dynamics, architecture, and trends,” in IEEE Journal on Selected Areas in Communications, vol. 33, no. 6, pp. 1224–1240, Jun. 2015.

[111] V. Petrov et al., "Achieving End-to-End Reliability of Mission-Critical Traffic in Softwarized 5G Networks," *in IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 485-501, March 2018.

[112] M. Gapeyenko et al., “On the temporal effects of mobile blockers in urban Millimeter-wave cellular scenarios,” IEEE Trans. Veh. Technol., vol. 66, no. 11, pp. 10124–10138, Nov. 2017.

[113] G. Zhang, T. Q. S. Quek, M. Kountoursi, A. Huang and H. Shan, “Fundamental of heterogeneous backhaul design analysis and optimization,” IEEE Journal on Selected Areas in Communications, vol. 64, no. 2, pp. 876-889, Feb. 2016.

[114] T. S. Rappaport et al., “Millimeter wave mobile communications for 5G cellular: It will work!” IEEE Access, vol. 1, pp. 335–349, May 2013.

[115] Car Dimensions of any Make and Model in the European Market. Accessed: Jan, 2019. [Online]. Available: http://www. automobiledimension.com/

[116] iPerf, “The ultimate speed test tool for TCP, UDP and SCTP” Accessed: Jan, 2019. [Online]. Available: https://iperf.fr/

[117] SDN Hub, POX Controller tutorials. Accessed: Jan. 18, 2018. [Online]. Available: http://sdnhub.org/tutorials/pox/

Djabir Abdeldjalil CHEKIRED

Doctorat : Ingénierie Sociotechnique des Connaissances, des Réseaux et du Développement Durable

Année 2019

Vers des architectures cloud efficaces et sécurisées : application aux réseaux électriques intelligent, IoT et SDN VANET

Les technologies du Cloud computing ont attiré l'attention des gouvernements, industriels et chercheures, car elles offrent un potentiel énorme en termes d'efficacité de traitement, de capacité de stockage et de disponibilité. Cependant, elles présentent des défis particuliers qui entravent l'intégration du Cloud dans des nouvelles technologies émergentes. Ce travail présente trois applications principales du cloud computing. La première application concerne la gestion de l'énergie des réseaux électrique intelligents. Nous proposons trois contributions principals : 1) un nouveau modèle pour la planification de plug-in des véhicules électriques EVs basé sur une architecture cloud, 2) afin de réduire les coûts de la production d'énergie, nous proposons un modèle de tarification dynamique en temps réel, 3) pour sécuriser l'infrastructure de comptage intelligent de la consommation d'électricité contre les attaques d'injection de fausse informations, nous proposons un nouveau système de sécurité, appelé *SecMetering*. La deuxième application est conçue pour le traitement des données de l'Internet des objets industriels ; nous proposons une nouvelle architecture décentralisée basée sur un fog hiérarchique. La troisième application concerne les réseaux véhiculaires. Nous proposons deux nouvelles architectures : 1) une architecture fog hiérarchique basée sur SDN, appelé HSVF, 2) une architecture qui utilise le réseau sans fil 5G basé sur SDN pour améliorer les performances de la conduite autonome.

Mots clés : informatique dans les nuages – réseaux électriques intelligents – internet des objets – réseau défini par logiciel – réseaux ad hoc de véhicules – cyber – sécurité – 5G.

Towards Efficient and Secure Cloud Architectures and Models: Application on Smart Grid, IoT and SDN Vehicular Networks

Cloud computing technologies have seized the attention of different governments, industries, and academic institutions as they offer tremendous potential for new business opportunities in terms of scheduling efficiency, storage capacity, and availability. Yet, they have particular challenges that hinder cloud integration in new emerging technologies and systems. This work presents three principals applications of Cloud computing. The first application issue is for smart grid energy management. It proposes three principal contributions: 1) considering the scheduling of electric vehicles (EVs) energy demands for charging and discharging as a case study, we propose a new energy scheduling model for smart grid based on Cloud computing, 2) in order to reduce the energy cost and energy peak loads, we propose a real-time dynamic pricing policy, 3) to secure energy smart metering infrastructure against false data injection attacks in smart grid environment, we propose a new security scheme, labeled SecMetering. SecMetering is based on hierarchical and distributed intrusion detection system. The second application is designed for scheduling industrial internet of things data; we propose a new hierarchical and decentralized Fog architecture. The third application concerns vehicular networks. We propose two new architectures based on distributed SDN: 1) a hierarchical SDN-based wireless vehicular Fog, labeled HSVF, 2) a framework that uses 5G networks based SDN to enhance the performance of autonomous driving applications.

Keywords: cloud computing – smart power grids – internet of things – vehicular ad hoc networks – SDN – cyber-security – 5G.