Virtual Mobile Edge Computing Based on IoT Devices Resources In Smart Cities

Mohammed Laroui*†‡, Hatem Ibn Khedher §¶, Hassine Moungla †‡, Hossam Afifi‡, and Ahmed E. Kamal‖
*EEDIS Laboratory, Computer Science Department, Djillali Liabes University, Sidi Bel Abbes, Algeria.
†Université de Paris, LIPADE, F-75006 Paris, France.
‡UMR 5157, CNRS, Institut Polytechnique de Paris, Telecom SudParis Saclay, France.
§Technological Research Institute - IRT SystemX, 8 avenue de la vauve, 91120, Palaiseau, France.
¶Laboratoire des Signaux et Systèmes, Université Paris Sud-CNRS-CentraleSupélec, Université Paris-Saclay, France.
‖Department of Electrical Computer Engineering, Iowa State University, Ames, IA 50011-3060, USA.
Emails: {mohammed.laroui, hassine.moungla}@parisdescartes.fr, hatem.ibnkhedher@irt-systemx.fr,
hossam.afifi@telecom-sudparis.eu, kamal@iastate.edu

***Abstract*—The emerging of the internet of things (IoT) led to increasing the computation resources required to satisfy a large number of requests from the connected devices, for this the Cloud Computing (CC) allows the processing of requests in the cloud to guarantee the efficiency of services for end-users. The main problem of the current CC architecture is the latency in real-time applications such as video streaming, which require a distributed architecture to support the future generation of applications. The Mobile Edge Computing (MEC) provides a fully distributed architecture where a part of processing executed in the edge of network which supports the requirements of IoT applications. In this paper, we propose to use the connected devices as on-demand virtual edge servers to provide computation services close to end-users where each submitted task is divided into a set of sub-tasks, each one can be executed by any other device which is a part of the virtual edge server according to the available resources in the selected device. In this context, we have formulated the partitioned and the offloading problem in MEC environment using linear programming techniques. Optimal Partitioned and Offloading (OPO) algorithm that allocates network, storage and computing resources to user application sub-tasks with respect to MEC constraints and user quality requirements is modeled, implemented, and evaluated. Results show the feasibility and efficiency of the proposed algorithms.**

***Index Terms*—IoT, Cloud Computing, Mobile Edge Computing (MEC), Partitioned and Offloading Algorithm, Optimization.**

I. Introduction

The evolution of mobile devices, vehicles, and embedded systems led to creating a smart world of connected devices that may collaborate by sensing, collecting data, and take decisions without human intervention. This system is called the internet of things (IoT) [1]. It allows the communication of different devices in the network which creates a smart environment such as smart homes and smart cities.

The main characteristic of the internet of things is the interaction of different technologies such as sensors, actuator networks, and identification networks, etc, for objective to improve the cooperation between various technologies. The connected objects generate a huge amount of data during the communication, which requires processing in a short time because most IoT applications are real-time such as video streaming and self-driving. The data generated controlled and processed in the cloud, this mechanism is called cloud computing [2].

It aims to provide high computation resources for processing and storage to connected devices, where three different services are offered by the cloud namely: *Software as a Service (SaaS) [3]* which provide an application programming interface (API) to developers in order to create their applications in the cloud, in addition, the users can not manage or control the network and the storage of data. *Platform as a service (PaaS) [4]* provides services for deploying applications, by allows users to run applications in virtualized servers in the cloud without extending their local resources. This platform consists of software that includes databases, development tools, and middleware. And finally, *Infrastructure as a Service (IaaS) [5]* that allows users to control and manage software and hardware resources at the cloud.

The main problem of the IoT-Cloud architecture is the response time, because the rapid increase of IoT devices which will reach to billions or trillions of devices [6] lead to generating a huge amount of data [7], [8] that require a short time for control and process, for this a new architecture called edge computing where a part of data processing decentralized from the cloud to the edge of the network. The edge computing called also fog computing [9]–[12] aims to be the future IoT solution that solves different issues such as time-constrained and computation-based applications. Besides, edge computing offers large benefits for future network architectures [13]–[15] such as reduce the communication latency and networking load [16], reduce the energy consumption of mobile end-users as well as provide privacy protection and system security.

Different edge computing architectures based on fixed, mobile, or mixed edge servers have been proposed each one offers benefits to current and future network generation. In this paper, we propose a new architecture for edge computing where IoT devices resources are used as a virtual edge server for data processing, each device in the network receive a part of the task according to its available resources for processing, where a group of devices forms a virtual edge server.

978-1-7281-5089-5/20/$31.00 ©2020 IEEE

II. Related Work

Different types of edge computing architectures have been used to enhancing the efficiency of the provided services such as video streaming and task scheduling.

A. Fixed Edge Servers-based Solutions

Fixed edge servers in edge computing are generally installed in fixed positions inside the cities (supermarket, big building, roadside...etc) and aim to provide services to mobile end users. The following efforts have been proposed for edge computing with fixed edge servers.

Caminha *et al.* [17] proposed a smart city architecture using fog system and buses, called SensingBus. it composed of three levels: (i) a sensing level composed of sensing nodes installed in buses, allows to collect data about the city and save the geographic GPS coordinates. (ii) the fog level that is responsible for data pre-processing and forwarding this data to the cloud, it is generally installed inside the city. and finally, (iii) cloud level that is composed of nodes, each one contains a set of virtual machines. The performance analysis proves the enhancing of CPU utilization and memory. However, the proposed architecture is limited because the buses can not provide a wide covering inside the city.

Abedin *et al.* [18] proposed a joint resource allocation and user association scheme that evaluates the application Quality of Service (QoS) requirements and considers the priority of the different QoS parameters for IoT applications in a dynamic Fog Network environment. Then, they provided an Analytic Hierarchy Process (AHP) to decompose the QoS management problem into sub-problems for objective to prioritize the QoS parameters and requirements. In addition, by using AHP they formulated a two-sided matching game in order to initiate a stable association between the IoT devices and fog network infrastructure. Finally, the best-fit resource allocation strategy is applied to solve the network resource allocation problem. The simulation results prove the stability of the association between the fog devices and IoT in a dynamic and scalable network.

B. Mobile Edge Servers-based Solutions

Mobile edge servers in edge computing allow mobile devices to connect to mobile edge servers, and generally are installed in taxis, drones, buses, etc, to receive data or process tasks. Relevant works have been proposed in different domains and presented as follows.

Ye *et al.* [19] proposed a service offloading using buses as fog servers to accomplish some tasks offloaded by cloudlets and provide services to end-users. They used a Genetic Algorithm (GA) as an allocation strategy that aims to minimize the cost of tasks offloading. The proposed strategy minimize data transmission cost. However, the risk of communication failure between bus servers and end-users caused by obstacles and congestion is not considered in this work.

Hu *et al.* [20] proposed a multi-access edge computing framework and a communication protocol in vehicular networks based on the integration of millimeter-wave, licensed Sub-6 GHz band and IEEE 802.11p. They used a cluster-based approach to select the efficient gateway nodes based on a fuzzy logic algorithm to maximize the network throughput. Further, they used a V2V communication-based IEEE 802.11p to share the relevant information between smart vehicles. The proposed solution achieves better performance. Nevertheless, the solution is not deployed in large-scale networks.

Zeng *et al.* [21] proposed a bus-oriented V2X infrastructure where the buses are considered as fog nodes (sink nodes) that fulfill communication, resource allocation, and convergence computing task. They used different simulators such as Matlab for path loss calculation and ns3 for network simulation to investigate the performance of the proposed solution. Simulation results show that the proposed solution achieves high communication efficiency and decreases the frequency of service migration. However, this work did not consider the risk of communication link failure caused by obstacles.

C. Mixed Edge Servers-based Solutions

Mixed edge servers in edge computing allow connected devices to receive data from two different types of edge servers fixed and mobile. Generally, the two servers provide the same job but sometimes each server has a specific role according to the communication scenarios. In the following, we present the existing efforts in this area.

Lai *et al.* [22] proposed a Fog-based Public Vehicle System (FPVS) that includes different components of route scheduling, metadata gathering, request answering, and cost estimation. They used a heuristic algorithm to send requests and schedule routes for vehicles. The proposed FPVS framework ensures lower cost and less waiting time. Nevertheless, the heuristic algorithm requires high computation resources. The exact formulation is missed to evaluate the effectiveness of the approximation approach.

Yuan *et al.* [23] proposed a two-level automated driving solution in an edge computing environment. The autonomous vehicles submitted their service requests to the edge server which determines if the requests first processed locally or need to be handed to the cloud. They demonstrated that content sharing and content caching through vehicular networks can reduce resource utilization. However, the risk of link failure during v2v communication is not considered in this work.

Ning *et al.* [24] proposed a three-layer vehicular fog computing (VFC) architecture to enable distributed traffic management, for objective to minimize the response time. They formulated the offloading scheme as an optimization problem by leveraging parked and moving vehicles as fog nodes. Furthermore, they used a real taxi trajectory to validate the model. The performance evaluations show that the proposed solution can provide response in a short time under different network circumstances.

D. Virtual Edge Servers-based Solutions

Currently, virtual cloud/edge servers are presented as a mobile ad-hoc point of presence [25]–[27] through different

network architectures. They allow creating distributed computation resources where the smart devices work together to accomplish tasks for end-user devices. This paradigm achieves good results in terms of deployment cost, end-to-end latency, and energy consumption.

Miluzzo *et al.* [28] presented mClouds, a mobile cloud computing architecture where mobile devices (smart-phones and tablets) are used as core computing nodes to accomplish submitted tasks based on individual or distributed processing. Besides, the authors discussed the different steps of executing tasks in the devices and the management of the mClouds. However, the impact of mobility is not well discussed, and no numerical results presented in this work. Further, the exact formulation of the partitioned and offloading mechanisms are missed.

Fahim *et al.* [29] proposed a mobile device cloud (MDC) environment for tasks offloading toward mobile devices. A real measurement is used to validate the results where real mobile phones are used for tasks offloading. The performance evaluation proves the efficiency of the proposed environment where the potential gain is high in both energy and processing time. However, the impact of the mobility and the application of the proposed environment in large scale networks are not studied in this work.

Similarly, Hasan *et al.* [30] proposed a mobile ad-hoc cloud computing for tasks offloading based on IoT devices called Aura, that has three components: i) the Mobile Agents, that represent end-user devices such as smartphones. They are leveraged to run the jobs that are needed to be offloaded to the aura for execution. ii) the IoT devices, that represent the part of aura which are the devices that form the ad-hoc cloud, and iii) the controllers that represent the link between the Agents and IoT devices, which is responsible for creating and distributing sub-tasks, etc. The numerical results prove the efficiency of the aura in terms of task completion time, memory/CPU usage, and the cost of execution. However, the impact of mobility is not studied in this work.

Van *et al.* [31] proposed an offloading scheme based on deep reinforcement learning (DRL) in an ad-hoc mobile cloud that allows mobile end-user to offload tasks to the closer mobile cloudlets using the cellular network. The offloading problem is formulated using the Markov Decision Process (MDP) to determine the optimal actions for tasks processed locally and tasks offloaded to cloudlets. Then, a DRL method called Deep-Q-Network (DQN) is used to learn an efficient solution for the MDP. The simulation results validate the efficiency of the proposed scheme in terms of different performance metrics such as energy consumption and delay, etc. However, this work did not consider task requirements such as GPU and RAM.

III. Proposed Architecture

In this section, we describe our proposed offloading architecture. In Figure 1, we show a generic view of the virtual MEC architecture where different IoT devices (Smart-phones, laptops, drones, cameras, etc.) are connecting in a smart city. These devices are connected to a fixed access point which is also connected directly to the central cloud. The virtual edge servers are created on demand by the access point when it receives a request from the end-user to process their tasks. The virtual edge servers are composed of a set of connected IoT devices chosen by the access point according to the requirement of the submitted task. If this task requires high computation resources that are not available, in this situation the tasks are sent directly by the access point to the cloud.

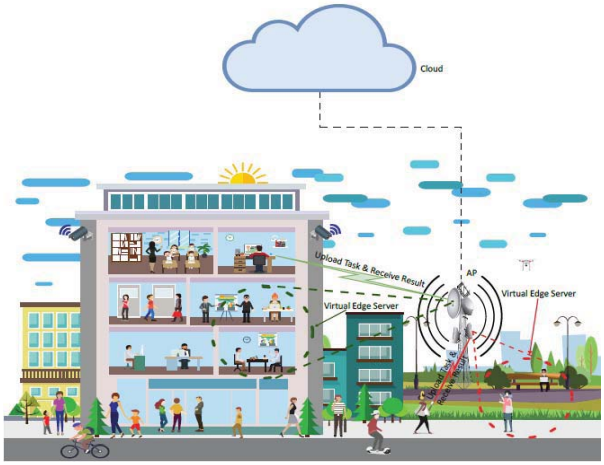

Figure 1: Virtual Mobile Edge Computing Architecture.

A. Design Goals

The objective of designing the network architecture is to provide a global overview of the communication scenarios in the network including the different components that participating in the different interactions such as smart devices, smart vehicles, etc. Thus, the proposed architecture aims to provide a distributed data processing in virtual edge servers that consisting of the different connected devices inside the smart city that by consequence, reduce the computation cost and enhance the efficiency of the network's services.

IV. VEnPA Optimization Model

In this section, we aim to design an optimal routing and task placement algorithm for the virtual applications, where the optimal criterion is to minimize capital and operational costs.

ETSI standards (MEC, NFV, and SDN research groups do not specify how virtual edge nodes including user devices) can be deployed in the larger IoT topology. Our algorithm, Optimal Virtual Edge nodes Placement Algorithm, VEnPA, given a network infrastructure, will solve this deployment problem and will give the optimal offloading/partitioning solution in terms of service instantiation graphs that simply map service sub-tasks to mobile devices. It takes as input, the global network topology consisting of :

- IoT devices (users requesting service computation).
- IoT gateways corresponding to all network elements including larger Internet.
- Cloud.

TABLE I: Summary of the most used notation in this paper.

Notations	Definition
$\mathcal{UA}$	The set of User Applications (UA).
$\mathcal{VE}$	The set of virtual edge servers.
$\mathcal{K}$	The set of user application sub-tasks.
$\mathcal{R}_{ve}$	Maximum RAM available in the edge device ve.
$\mathcal{C}_{ve}$	Maximum CPU available in the edge device ve.
$\mathcal{S}_{ve}$	Maximum Storage available in the edge device ve.
$\mathcal{G}_{ve}$	Maximum GPU available in the edge device ve.
$r_{ud,k}$	Required RAM for the application sub-task (ud, k).
$c_{ud,k}$	Required CPU for the application sub-task (ud, k).
$s_{ud,k}$	Required Storage for the application sub-task (ud, k).
$g_{ud,k}$	Required GPU for the application sub-task (ud, k).
Decision variables	**Definition**
$x^{ve}_{ud,k}$	A binary variable that assigns the sub-task $k \in \mathcal{K}$ of user application $ud \in \mathcal{N}$ to the virtual edge $ve \in \mathcal{VE}$.

A. Problem statement, constraints and main objectives

Hereafter, we first state the system hypothesizes and then present the virtual edge offloading models based on the exact Integer Linear Programming (ILP) optimization technique.

We suppose each submitted task will be fragmented to sub-tasks that can be executed in a selected device from the set of devices that form the virtual edge server. After processing the results of sub-tasks, it combined to form the main result of the main task.

Moreover, according to the set of connected devices, a part of devices are selected to form a virtual edge server that processes the requests at the edge of the network. Each device has its available resources (RAM, CPU, GPU, Storage, Energy) that can be used to process sub-task(s).

B. Mathematical formulation

1) VEnPA: System parameters: In this subsection, we specify the parameters and the constraints that are defined and proposed in formulating the optimization/analytic model. This formulation determines the partitioning and the offloading of application services to the optimal location (i.e., virtual edge devices).

We quote in Table I the main VEnPA parameters and decision variables.

2) VEnPA: Decision variables:

- The binary variable x indicates the placement of the service sub-tasks, and its offloading from the user device ud to the optimal virtual edge device ve. It is defined as:

$$x^{ve}_{ud,k} = \begin{cases} 1 & \text{if service subtask } (ud,k) \text{ is placed} \\ & \text{on the virtual edge } ve \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

- The binary variable y that indicates the virtual edge usage. It is formulated as follows:

$$y^{ve} = \begin{cases} 1 & \text{if the virtual edge } ve \in \mathcal{VE} \text{ is used} \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

3) VEnPA: algorithm Constraints: The general formulation of the exact partitioned and offloading algorithm, **Optimal Partitioned and Offloading (OPO)** is as follows:

$$\max \sum_{ud \in \mathcal{UA}} \sum_{ve \in \mathcal{VE}} x^{ve}_{ud,k} \quad (3)$$

Subject to

$$\sum_{ve \in \mathcal{VE}} x^{ve}_{ud,k} \leq 1, \quad \forall ud \in \mathcal{UA}, k \in \mathcal{K} \quad (4)$$

$$\sum_{ud,k} r_{ud,k} x^{ve}_{ud,k} \leq \mathcal{R}_{ve}, \quad \forall ve \in \mathcal{VE} \quad (5)$$

$$\sum_{ud,k} s_{ud,k} x^{ve}_{ud,k} \leq \mathcal{S}_{ve}, \quad \forall ve \in \mathcal{VE} \quad (6)$$

$$\sum_{ud,k} c_{ud,k} x^{ve}_{ud,k} \leq \mathcal{C}_{ve}, \quad \forall ve \in \mathcal{VE} \quad (7)$$

$$\sum_{ud,k} g_{ud,k} x^{ve}_{ud,k} \leq \mathcal{G}_{ve} \quad \forall ve \in \mathcal{VE} \quad (8)$$

$$x^{ve}_{ud,k} \in \{0, 1\} \quad (9)$$

Algorithm constraints represent rules that need to be verified. Equations (5), (6), (7), and (8) ensure that the selected virtual edge device has enough amount of RAM, storage, CPU, and GPU respectively to accomplish the submitted user application sub-tasks. Besides, the constraint (4) guarantees the offloading of each subtask to a unique virtual edge device.

4) VEnPA: algorithm complexity and periodicity: It is clear that the exact formulation of the problem has an exponential number of feasible solutions. Moreover, the offloading is executed after two triggers:

- System constraints represented by CPU/GPU, RAM, and storage parameters.
- System prediction through the analysis of the available resources that are dynamically changed over time.

V. VEnPA: optimization evaluation

The performance of the virtual mobile edge computing architecture has been evaluated using Three different applications in a simulated environment. We have used EdgeCloudSim [32] to simulate the virtual edge computing communication and the offered services.

A. Simulation Setup

For each application we define a set of parameters including *CPU, GPU, RAM,* and *storage* requirements, in addition, we specify the number of *connected devices, task length, data upload, data download,* and the *required core*.

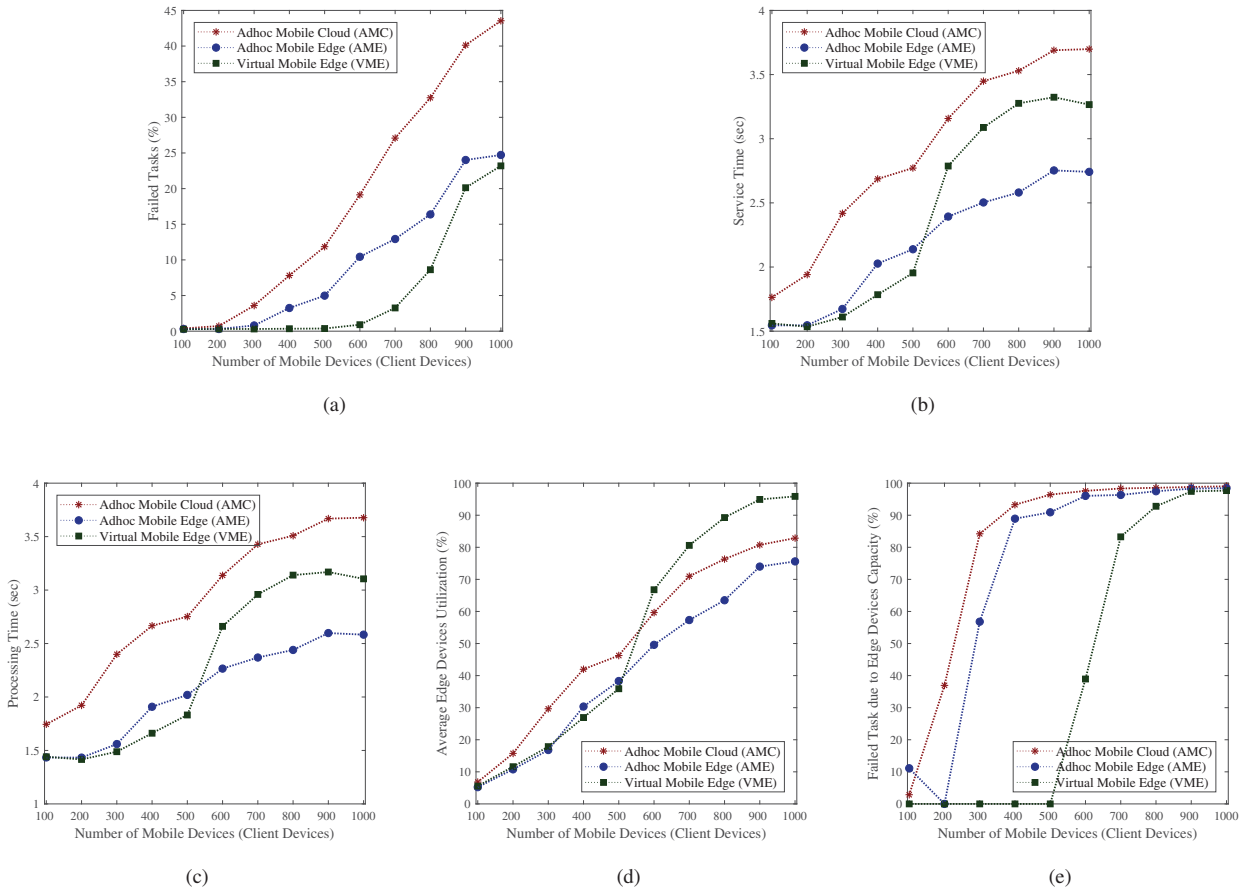

Figure 2: Performance Evaluation of the Virtual Mobile Edge Computing Based VEnPA Model.

B. Metrics

The performance of the Cloud and Edge computing systems is based on resources utilization in the proposed architecture. Therefore, in this work, we focus on the measurement of *Failed Tasks, Processing Time, Service Time, Average Edge Devices Utilization*, and *Failed Tasks due to Edge Devices Capacity*.

C. Simulation Results

Figure 2 outline the performance evaluation of the proposed architecture. Beside, the figures 2a, 2b, 2c, 2d, and 2e shows a benchmark among our proposed architecture (*Virtual Mobile Edge (AME)*), *Adhoc Mobile Cloud (AMC)* that represent the architecture proposed in [28]–[31]. And finally, *Adhoc Mobile Edge (AME)* architecture that allow to use end user devices as edge servers in Adhoc mode.

Figure 2a, display the percent of failed tasks during the increase of client devices, we show that when the number of devices increases the percent of failed tasks increase in the three architectures.

In the proposed architecture the increased level is low compared to the two other architectures because the proposed architecture provides an efficiency offloading technique that can select the virtual edge devices for each submitted task, wherein the Adhoc modes (*AMC, AME*) the offloading is difficult because it takes a long time which leads to task failure.

Figure 2b, reports the service time for task execution in the three architectures, we can see that when the number of client devices is less than 500, the service time for the *VME* architecture is low compared to the *AMC* and *AME*.

However, when the number of client devices is more than 500, the service time of *VME* is lower than *AMC* and higher than *AME*, because in *AMC*, the end-user devices play the role of the cloud which leads to a risque of high level of task failure (figure 2a) due to high tasks requirements or insufficient mobile cloud devices.

Where in *AME*, the service time is lower than the *VME* because the number of failed tasks in *AME* is high compared to the *VME* (figure 2a) which mean that *VME* process tasks more than *AME* which by consequence take more time.

Figure 2c, show the processing time in *AMC, AME*, and *VME*. We can notice that the *AME* has the shortest processing time. However, the *VME* is the more efficient because is process tasks more than *AMC* and *AME* (figure 2a).

Figure 2d, represents the *Average Edge Devices Utilization*, we can see that when the number of client device increase, the edge devices utilization increase so, because of the generated tasks of client devices, besides, the *VME* has the highest percentage value of edge devices utilization because it has the lowest failed tasks (figure 2a).

Figure 2e, depicts the *Failed Tasks due to Edge Devices Capacity*, the figure show that the proposed architecture (*VME*) has the lowest failed tasks due to edge devices capacity (*RAM, CPU, GPU, Storage*) because of the proposed offloading model (VEnPA model) that provide an efficient partitioned and offloading.

VI. Conclusion & Future Work

In this work, we propose to use end-user devices as virtual edge servers for tasks offloading in an edge computing environment. An optimal partitioned and offloading (OPO) algorithm is formulated based on *Integer Linear Programming (ILP)* optimization technique to guarantee optimal tasks offloading to the selected edge server. The performance evaluation proves the efficiency and the feasibility of the proposed system in terms of *Failed Tasks, Processing Time, Service Time, Average Edge Devices Utilization*, and *Failed Tasks due to Edge Devices Capacity*. Future work we will use more IoT devices such as drones, connected vehicles, etc as a virtual edge server, besides, we will study the impact of mobility and energy consumption on the offloading decision.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] P. K. Senyo, E. Addae, and R. Boateng, "Cloud computing research: A review of research themes, frameworks, methods and future research directions," *International Journal of Information Management*, vol. 38, no. 1, pp. 128–139, 2018.

[3] M. A. Cusumano, "Cloud computing and SaaS as new computing platforms." *Commun. ACM*, vol. 53, no. 4, pp. 27–29, 2010.

[4] C. Pahl, "Containerization and the paas cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[5] J. Scheuner and P. Leitner, "Performance Benchmarking of Infrastructure-as-a-Service (IaaS) Clouds with Cloud WorkBench," in *Companion of the ACM/SPEC International Conference on Performance Engineering*. ACM, 2019, pp. 53–56.

[6] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal (IoT-J)*, vol. 3, no. 5, pp. 637–646, 2016.

[8] M. Laroui, A. Dridi, H. Afifi, H. Moungla, M. Marot, and M. A. Cherif, "Energy management for electric vehicles in smart cities: a deep learning approach," in *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019, pp. 2080–2085.

[9] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *IEEE International Conference on Advanced Information Networking and Applications*, 2015, pp. 687–694.

[10] K. Kai, W. Cong, and L. Tao, "Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues," *the journal of China Universities of Posts and Telecommunications*, vol. 23, no. 2, pp. 56–96, 2016.

[11] M. Aazam and E.-N. Huh, "Fog computing: The cloud-iot\/ioe middleware paradigm," *IEEE Potentials*, vol. 35, no. 3, pp. 40–44, 2016.

[12] K. Monteiro, M. Marot, and H. Khedher, "Review on microgrid communications solutions: a named data networking fog approach," 06 2017, pp. 1–8.

[13] H. Khelifi, S. Luo, B. Nour, A. Sellami, H. Moungla, and F. Naït-Abdesselam, "An optimized proactive caching scheme based on mobility prediction for vehicular networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[14] B. Nour, K. Sharif, F. Li, H. Moungla, A. E. Kamal, and H. Afifi, "NCP: a near ICN cache placement scheme for IoT-based traffic class," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[15] H. Moungla, K. Haddadi, and S. Boudjit, "Distributed interference management in medical wireless sensor networks," in *IEEE annual consumer communications & networking conference (CCNC)*, 2016, pp. 151–155.

[16] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization," *IEEE Transactions on Vehicular Technology*, 2020.

[17] P. H. C. Caminha, F. F. da Silva, R. G. Pacheco, R. de Souza Couto, P. B. Velloso, M. E. M. Campista, and L. H. M. K. Costa, "SensingBus: Using Bus Lines and Fog Computing for Smart Sensing the City," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 58–69, 2018.

[18] S. F. Abedin, M. G. R. Alam, S. A. Kazmi, N. H. Tran, D. Niyato, and C. S. Hong, "Resource allocation for ultra-reliable and enhanced mobile broadband IoT applications in fog network," *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 489–502, 2019.

[19] D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2016, pp. 247–251.

[20] Q. Hu, C. Wu, X. Zhao, X. Chen, Y. Ji, and T. Yoshinaga, "Vehicular multi-access edge computing with licensed sub-6 ghz, ieee 802.11 p and mmwave," *IEEE Access*, vol. 6, pp. 1995–2004, 2018.

[21] L. Zeng, J. Zhang, Q. Han, L. Ye, Q. He, X. Zhang, and T. Yang, "A Bus Oriented Mobile FCNs Infrastructure and Intra-cluster BSM Transmission Mechanism," *IEEE Access*, 2019.

[22] Y. Lai, F. Yang, L. Zhang, and Z. Lin, "Distributed public vehicle system based on fog nodes and vehicular sensing," *IEEE Access*, vol. 6, pp. 22 011–22 024, 2018.

[23] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Network*, vol. 32, no. 1, pp. 80–86, 2018.

[24] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[25] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 3009–3036, 2015.

[26] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds," *IEEE Transactions on Cloud Computing*, 2016.

[27] F. Gu, J. Niu, Z. Qi, and M. Atiquzzaman, "Partitioning and offloading in smart mobile devices for mobile cloud computing: State of the art and future directions," *Journal of Network and Computer Applications*, vol. 119, pp. 83–96, 2018.

[28] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: mClouds-computing on clouds of mobile devices," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, 2012, pp. 9–14.

[29] A. Fahim, A. Mtibaa, and K. A. Harras, "Making the case for computational offloading in mobile device clouds," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 203–205.

[30] R. Hasan, M. Hossain, and R. Khan, "Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading," *Future Generation Computer Systems*, vol. 86, pp. 821–835, 2018.

[31] D. Van Le and C.-K. Tham, "A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 760–765.

[32] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.