IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON COLLABORATIVE INTELLIGENCE FOR INTERNET OF VEHICLES

Received July 29, 2021, accepted August 12, 2021, date of publication August 20, 2021, date of current version August 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3106284

oneVFC–A Vehicular Fog Computation Platform for Artificial Intelligence in Internet of Vehicles

KIEU-HA PHUNG[1], HIEU TRAN[1], THANG NGUYEN[1], HUNG V. DAO[1], VINH TRAN-QUANG[1], THU-HUONG TRUONG[1], (Member, IEEE), AN BRAEKEN[2], AND KRIS STEENHAUT[2,3], (Member, IEEE)

[1]School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 100000, Vietnam
[2]Department of Engineering technology (INDI), Vrije Universiteit Brussel, 1050 Brussels, Belgium
[3]Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, 1050 Brussels, Belgium

Corresponding author: Kieu-Ha Phung (ha.phungthikieu@hust.edu.vn)

This work was supported by Vingroup Innovation Foundation (VINIF) under Project VINIF.2019.DA16.

ABSTRACT We are witnessing the evolution from Internet of Things (IoT) to Internet of Vehicles (IoV). Internet connected vehicles can sense, communicate, analyze and make decisions. Rich vehicle-related data collection allows to apply artificial intelligence (AI) such as machine learning and deep learning (DL) to develop advanced services in Intelligent Transportation Systems (ITS). However, AI/DL-based ITS applications require intensive computation, both for model training and deployment. The exploitation of the huge computational power obtained through aggregation of resources present in individual vehicles and ITS infrastructure brings an efficient solution. In this work, *oneVFC*, a tangible vehicular fog computing (VFC) platform based on oneM2M is proposed. It benefits from the oneM2M standard to facilitate interoperability as well as hierarchical resource organization. *oneVFC* manages the distributed resources, orchestrates information flows and computing tasks on vehicle fog nodes and feeds back results to the application users. On a lab scale model consisting of Raspberry Pi modules and laptops, we demonstrate how *oneVFC* manages the AI-driven applications running on various machines and how it succeeds in significantly reducing application processing time, especially in cases with high workload or with requests arriving at high pace. We also show how *oneVFC* facilitates the deployment of AI model training in Federated Learning (FL), an advanced privacy preserving and communication saving training approach. Our experiments deployed in an outdoor environment with mobile fog nodes participating in the computation jobs confirm the feasibility of *oneVFC* for IoV environments whenever the communication links among fog nodes are guaranteed by V2X technology.

INDEX TERMS Artificial intelligence, deep learning, cooperative computing, interoperability, Internet of Things, Internet of Vehicles, oneM2M.

I. INTRODUCTION

The number of vehicles used worldwide is expected to rise from one billion in 2010 to two billion in 2030. Vehicles have become sensor platforms able to sense, communicate, analyze, and make decision [1], [2]. Internet of Vehicles (IoV), a network allowing data and information exchange among vehicles, things such as roadside infrastructure, humans, and the environment is becoming a reality thanks to Vehicle to Everything (V2X) technology which is based on two pillars being 5G-LTE and Dedicated Short-Range Communications (DSRC) [3]. Those intelligent vehicles and networks give rise to Intelligent Transportation Systems (ITSs), providing services like forward accident alarms, collision avoidance, traffic congestion mitigation, platoon of vehicles, autonomous driving vehicles, etc.

Advanced ITS services are Artificial Intelligence (AI)-based applications whose efficiency is enhanced by rich data collection in the IoV environment [4]. Vehicle-related data obtained from sensors in vehicles, Global Positioning Systems (GPSs), electronic toll tags, vehicle On Board Units (OBUs) rapidly increase in volume and variety. ITS data are also collected from other sources, like ITS infrastructures such as loop detectors, infra-red sensors, ultrasonic sensors, and closed-circuit television (CCTV) cameras, travelers (who use web browsers, mobile apps, social networks).

The associate editor coordinating the review of this manuscript and approving it for publication was Celimuge Wu.

Such complex and abundant data resources help to train AI models with better accuracy. AI-based applications require rich data sets and intensive computation, both in training and deployment phase.

Previously, cloud computing played a major role in big data analytic platforms. However, the uploading of huge amounts of data to the cloud, together with the advent of latency sensitive services make fog computing which ''extends the cloud computing to the edge'', a welcome or mandatory extension [5]–[8]. Distributed and parallel processing at the network edge can provide local view-based analytics, lower data and service delivery latencies, and enhanced data privacy. It can be a perfect choice for the heterogeneous, dynamic, distributed IoV environment. However, as computing and communication workload for ITS services varies over time and location, capacity planning of fog nodes is a challenge. To enhance resource utilization efficiency, vehicular fog computing (VFC) is created to exploit a huge computational power through aggregation of individual vehicles' resources and other devices in the ITS infrastructure [9], [10]. The feasibility of leveraging computing and communication resources of slowly moving cars in cross-section regions or of parked cars for supporting advanced vehicular applications has been investigated.

Various deployment scenarios of VFC have been studied. In those scenarios, the fog nodes can be computing devices installed in buses or taxis that process data being offloaded from client vehicles, when those buses or taxis are travelling alongside those clients [6], [11]. The fog nodes can also be parked vehicles that take the role of static backbone nodes for fog computing. They can also be vehicles stuck in traffic congestion that form a cluster or computing devices combined with V2X Road Side Units (RSUs). The mobility of vehicles can be leveraged as an effective way of organizing computing resource migration [12]. Other research focuses on task assignment and resource allocation, which are essential concerns in shared resource environments. The studies in [13]–[16], show how to assign computing tasks to be parallelly computed by a set of vehicular fog nodes to satisfy objectives related to quality of service such as latency, image/video resolution under constraints related to communication bandwidth, computation capability and energy consumption of these nodes. In [17]–[19], a contract- or auction-based approach is applied to stipulate or negotiate the provided resources together with the benefits obtained in terms of parking fee reduction when leveraging the computing units of parked cars.

In this paper, our aim is to realize the management of parallel service computations on various computing devices installed in vehicles. The platform, named *oneVFC*, adopts the 3-layer VFC architecture in which the role of management and orchestration is taken up by the fog layer [20]. We propose a hierarchical structure including fog manager nodes and fog worker nodes. The manager nodes can be seen as fixed nodes like V2X-RSUs, whereas the fog worker nodes are vehicle OBUs. A manager node will manage the available computing resources of the worker nodes in its neighborhood and will direct computing tasks to them according to a task assignment algorithm. This algorithm will realize specified objectives, such as minimizing the serving processing time.

We propose to use the oneM2M standard for creating the *oneVFC* platform. oneM2M is an IoT middleware standard for realizing interoperability between heterogeneous Machine to Machine (M2M)/Internet of Things (IoT) systems active in different service domains, such as smart transportation, smart city, smart health etc. It is increasingly used in commercial deployments [21]. oneM2M is a joint effort of eight national and international standard organizations, and has more than 200 members including national telecom companies (telcos). Various big telcos have deployed oneM2M commercial platforms [22].

The choice of oneM2M is motivated as follows. Firstly, a oneM2M-based architecture for the VFC platform will facilitate the integration of already existing ITS systems, e.g., electronic toll collection systems, traffic lights, city parking systems, CCTV cameras, which are often deployed using different protocol families. Secondly, the physical nodes like vehicle OBU, V2X-RSUs, and the information exchange among nodes can be represented as resources accessible through the publish/subscribe mechanism. A oneM2M node can support more than one underlying network interface, among which innovative technologies that allow high bandwidth and low latency communication in V2X. For vehicle OBUs, both DSRC and 5G enable parallel Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communications. Based on the required Quality of Service (QoS), traffic belonging to a certain application may use DSRC network while other traffic will use a 5G network.

We decompose the aggregated service flows of AI-based applications into several ''operation primitives'' that can be used to orchestrate the computing workloads distributed over various available computing resources. Then, we propose a communication and a computation management scheme that can be used to optimize task assignment and resource allocation, considering the constraints of node capacity and availability since most of the computation devices are installed in personal cars.

We evaluate the proposed *oneVFC* platform on a lab scale testbed, for managing the deployment of AI-based applications on various machines. A significant reduction of application processing time, especially for high workloads, or for service requests arriving at high pace is obtained. The *oneVFC* platform also facilitates the deployment of AI model training in the Federated Learning approach. The data message structure and procedures in *oneVFC* allow to monitor the operation states of each computing node and to support adaptive task assignment and resource allocation. Deploying *oneVFC* on a testbed setup in university's outdoor parking space demonstrates its efficiency for managing computing units attached to travelling vehicles that take computation jobs to lower overall service processing time. The benefits brought by the *oneVFC* platform for distributing

computation over various fixed and mobile fog worker nodes are confirmed, and those benefits will be increased with the availability of DSRC's and 5G's data rates.

The paper is structured as follows. Section II presents our analysis of ITS applications and the challenges of VFC deployment in IoV environments. Section III examines the details of the *oneVFC* architecture, paying particular attention to the reasons for choosing the oneM2M standard for the VFC platform, and to the oneM2M based functional structure of manager fog nodes and worker fog nodes. Section IV shows the resources and procedures to manage information/data flows by using service primitives of the oneM2M standard. In section V, the model of task assignment and resource allocation is proposed and the applicability of a particle swarm optimization algorithm for task assignment to worker nodes is discussed. Section VI presents two use cases for evaluating *oneVFC* on the available testbeds. The final section draws a conclusion and discusses future work.

II. VEHICULAR FOG COMPUTING ARCHITECTURE FOR AI- BASED APPLICATIONS

The AI-based services process huge amounts of data generated by CCTV cameras installed along the roads in combination with data from sensors, cameras on vehicles. The service outcomes are announced to data/service users and to on-road vehicles. Recently, Deep Learning (DL) models, e.g., Yolo (You Only Look Once) [23], Convolutional Neural Networks (CNN) and their variations [24], are widely used for object detection [25]. DL model-based programs can detect cars, trucks, buses, people, etc., according to the defined object classes in the model. They become the main components for vision-based applications in many fields including traffic congestion detection and management, autonomous vehicle safety, etc. Deep Learning models require a large dataset and high computational demands for training the models, as well as high computational efforts for their real-time execution on bundles of collected images. Note that deep learning is a sub-set of machine learning, and machine learning is a subset of AI.

Generally, AI/DL-based ITS applications generate two types of computing services requests being:

1) AI-based model exploitation: this involves the real time application of AI/DL algorithms or other techniques on collected data from the surroundings to solve actual problems in traffic management, road safety, etc... A timely response must be extracted from the rich dataset. It can be beneficial to split the data in smaller pieces that each can be processed by other edge devices. Hence, *parallel data processing at various vehicular fog nodes could shorten the service response time*.
2) AI model training: predictive model training, especially with DL algorithms, requires intensive computation and data storage, and is usually performed in the cloud using popular tools such as Google Colab, Kaggle, and Azure Notebooks [26], [27]. As the demand for training has grown faster than the increase in computing resources, distributing the computation across multiple machines has become mandatory. In addition to distributed learning, a new distributed training approach, named Federated Learning (FL), is being studied intensively [28]. In FL, the global model can be found by aggregating the locally trained models. Those local models are trained with local data sets at local devices. To preserve data privacy, only model parameters, instead of data, are exchanged. In both distributed learning and federated learning, *parallel model training on various machines will increase communication efficiency and data privacy*.

To satisfy the rising demands for AI-based ITS applications, VFC is created to exploit a huge computational power through the resource aggregation of individual vehicles and other devices in the ITS infrastructure. The architecture of VFC follows a hierarchical 3-layer functional architecture consisting of device layer, fog layer and cloud layer as shown in Fig.1. The device layer includes 'data owners' being devices equipped with sensors/ actuators, that collect raw data of any kind and it also includes 'data users' being end-user devices that generate service requests. The cloud layer consists of local system servers and has a connection to the Internet cloud. The local servers are mainly used to store large amounts of raw or processed data for further analysis and are occasionally used to process instant services. The middle layer or fog layer consists of mini local servers attached to cellular-based base station (BTS) units or RSUs of DSRC, fixed fog nodes. Another kind of nodes in the fog layer are OBUs on vehicles which are mobile fog nodes. The fog layer will perform computation tasks for services that need instant response to service requests from data users.

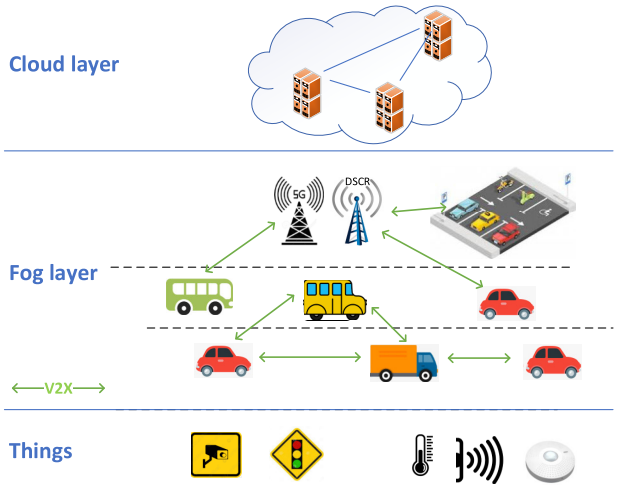

FIGURE 1. 3-layer architecture of Vehicle Fog Computing in Internet of Vehicles environment.

VFC must deal with sophisticated orchestration of heterogeneous resources considering their availability fluctuation due to resource mobility. Fog nodes may arbitrarily join or leave and offer variable communication quality in terms

of bandwidth and reliability. Moreover, the interconnections among fog nodes, data owners and data users are based on heterogeneous transmission technologies and protocols. As such, VFC's architectural design must meet the following requirements:

- Connect, interoperate, and monitor heterogeneous resources,
- Manage the availability of distributed resources for efficient usage,
- Orchestrate the information flows from data owners, the computing tasks on fog nodes as well as the feedback to data users.

In the following sections, we will analyze the suitability of oneM2M for realizing the VFC platform. We will provide the detailed design of oneM2M functions and services to build VFC functions, including the management of devices, resources, services, and data.

III. ONEM2M BASED ARCHITECTURE FOR VFC

A. WHY IS oneM2M SUITED FOR VFC IN IoV ENVIRONMENTS?

oneM2M is an IoT middleware standard for interoperability of heterogeneous M2M/IoT systems in different IoT service domains, such as smart transportation, smart city, and it is increasingly used in commercial deployments [21]. It supports a rich set of Application Programming Interfaces (APIs) for data exchange among things, machines, humans. The role of a middleware in IoT/IoV networking system is to provide storage and information processing services. A oneM2M-based architecture for VFC platform will facilitate the integration of already existing ITS systems, and leverage various authentication and authorization schemes, an important asset for providing a secure system [29].

oneM2M has classified five types of nodes, including infrastructure nodes (INs), middle nodes (MNs), application service nodes (ASNs), application dedicated nodes (ADNs) and non-oneM2M device nodes (NoDNs). A node's functions are classified into three layers, namely, the application layer, common service layer and network service layer; but not every node has all layers. Each layer has its own entities:

- application entities (AEs) stand for the applications in devices, gateways, or servers, containing the business logic of service applications,
- common service entity (CSE) is a set of common service functions for M2M/IoT services, allowing messages to be communicated coherently, regardless of the underlying network layer,
- network service entities (NSEs) are the underlying network services that are available for the CSE.

It defines two domains being the infrastructure and field domain. INs reside in the infrastructure domain while the field domain contains the other types. INs correspond to cloud platforms or servers of the application system, MNs can be field gateways which usually have sufficient resources and support both field protocols and IoT protocols. ASNs or ADNs can be smart objects or on-vehicle devices, both supporting general IoT protocols. ASNs are different from ADNs in two aspects. ASNs should have sufficiently rich resources, but ADNs have constrained resources. ASNs can relay information to other ASNs or ADNs, but ADNs cannot. Due to that, an ASN contains all AE and CSE layers whereas an ADN has only an AE layer. NoDNs correspond to sensors and actuators attached devices for which protocol conversion via gateway (MN) is required to join a oneM2M-based IoV.

oneM2M adopts the Resource Oriented Architecture (ROA) model, where all devices and related information can be handled as resources using a hierarchical structure. A resource, a uniquely addressed entity, can be transferred and manipulated using create, read, update and delete (CRUD), basic operations of RESTful architecture. Access to a resource is allowed by using different types of Common Service Functions (CSF), like Subscription and Notification, Discovery (probably with some predefined criteria for filtering). Any function/service in oneM2M is implemented as resource and procedure. A resource can be the data themselves, e.g., text files, images, energy level of a physical device, as well as the monitored states of an executed process/task, like execution time, completion time, etc. It enables a flexible representation for a wide range of data or information.

A oneM2M-based system has a tree-based architecture rooted at the infrastructure node.Information exchange between two M2M nodes will use the transport and connectivity services of the underlying networks. The routing information of a CSE, CSE-Point of Access (PoA), will depend on the characteristics of the underlying networks and will be provided by the CSE at the *registration phase*. The CSE-PoA is considered equivalent to the routable addresses of the targeted CSE. Besides, a oneM2M node can have more than one underlying network such as a low latency DSRC or 5G for outdoor environments, or a mmWAVE for indoors. For example, on vehicle OBUs, both DSRC, 5G are used to enable parallel V2V and V2I communications [30]. Multiple transportation networks can be used simultaneously to route QoS-differentiated traffic. The multiple differentiated CSE-PoAs will be defined during the development and implementation process.

IoT middleware platforms are classified into several interoperability, technical, syntactic and semantic levels [31]. Whereas oneM2M's focus is on technical and protocol interoperability with some efforts to tackle semantic interoperability, FIWARE, Oracle Fusion, Azure IoT, Amazon Web Services (AWS) [22], [32] are more focused on data models providing a semantic interoperability framework so that applications and services may easily consume information and/or trigger actions in various systems, including oneM2M-based IoT systems [31]. They are cloud-based platforms which do not fit into VFC in which data processing services are performed at edge nodes. However, they can be seen

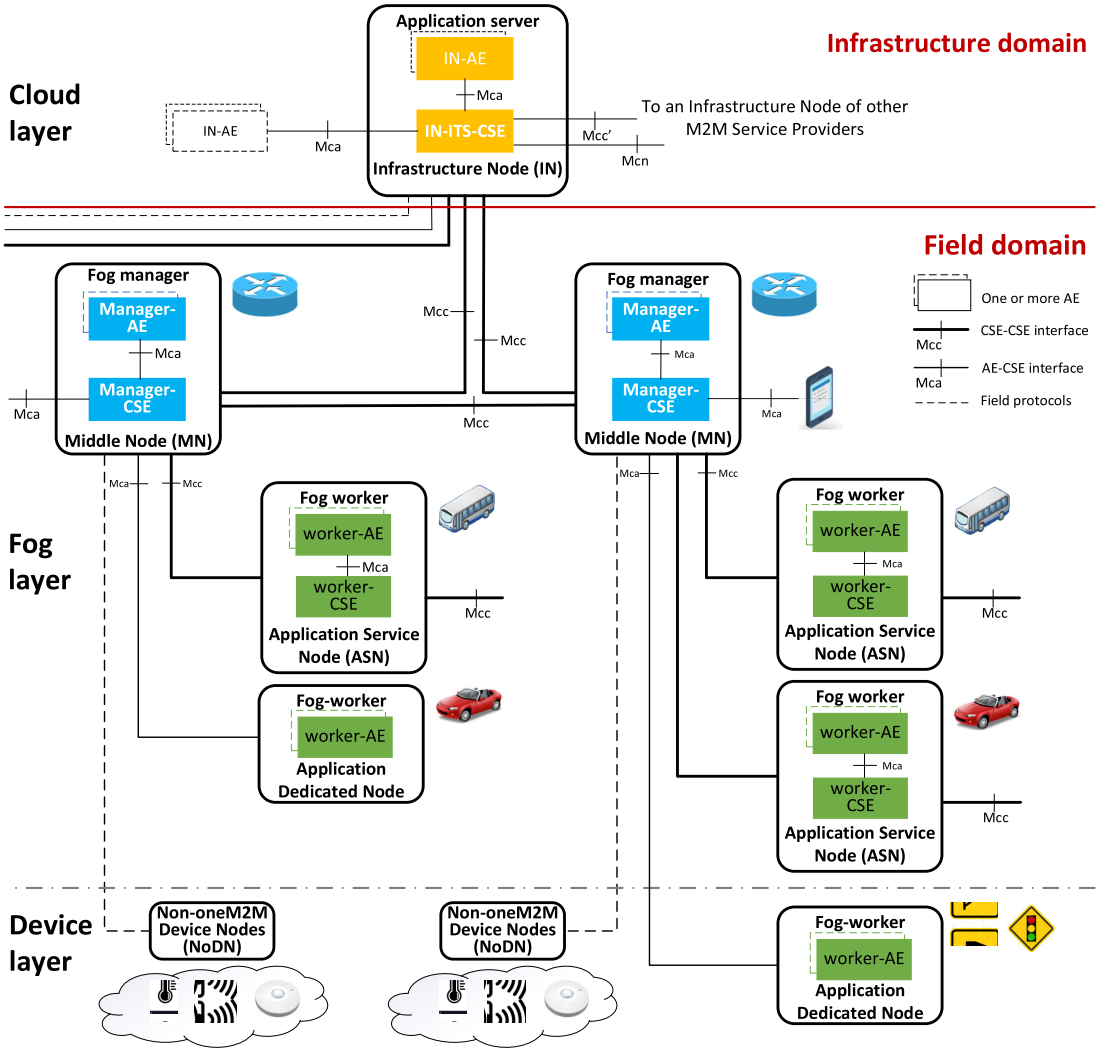

FIGURE 2. oneM2M based architecture for Vehicular Fog Computing in Internet of Vehicles environment.

as layers above the oneM2M-based systems to provide processed data results for use in a wide range of applications.

Previously, oneM2M has been studied to set up a fog channel for data exchange between two fixed fog nodes. The data channel is based on another protocol that can help to deliver data more quickly [33]. In our work, several data channels between the data owner and several worker nodes are established in parallel and are coordinated to obtain service results which are submitted to the fog manager.

B. oneVFC – oneM2M BASED ARCHITECTURE

The 3-layer VFC architecture is hierarchical. Various computing resource nodes probably in multi-hop communication fashion, are rooted to an access gateway (cellular-based BTS unit or DSRC RSU) as shown in Fig. 1. The hierarchical architecture of the proposed oneVFC is a scalable structure, and appropriate for local view-based data analytics in ITS systems. The access gateways are called *fog manager* nodes, and computing devices on vehicles are called *fog worker* nodes. A fog manager node is responsible for managing worker nodes in a given geographical area. The management of computation and communication resources, the task assignment and resource allocation are performed by the fog manager nodes. The *worker* nodes perform the computation jobs which are assigned by the manager nodes. Some worker nodes can relay information back and forth between the manager and the worker nodes. However, task deployment on far-away worker nodes is only suitable for delay tolerant services.

The VFC structure shows similarities with the architecture of oneM2M in which a middle node (MN) can manage many end-devices (ASNs or ADNs) under its responsibility, while the system server at cloud layer can act as IN. As shown in Fig. 2, we present a oneM2M-based architecture for the VFC platform. In this architecture, a *fog manager node* takes the role of MN. Vehicular computing nodes, *fog*

worker nodes, can be set up as ASNs. Since an ASN has a CSE layer, containing service functions for messages being exchanged among oneM2M nodes, a fog worker node can communicate control and data messages. Hence, a computing device attached to a bus can take the role of ASN since it travels regularly along some specified routes helping to fill the connectivity gap (by relaying information/data) in vehicle networks as well as to execute computation jobs.

Other end devices like cameras, traffic lights or IoT-based data collection systems are connected to the gateway-MN as ADNs or NoDNs to provide data for AI-based big data analytic applications. Fig. 2 presents the oneM2M-based functional structures of roadside gateways, computing devices on vehicles, data users like smartphone users, and data owners/generators like cameras, sensors, actuators in ITS applications.

Maintaining the communication links between any pair of nodes is the responsibility of the underlying networks which can be DSRC or 5G networks. To reduce the transmission latency, a fog manager node should be attached to a DSRC-RSU or a 5G BTS.

IV. RESOURCES AND PROCEDURES FOR DISTRIBUTED TASK COMPUTATION MANAGEMENT

In general, compute service requests can come from any node but only the manager node is able to process the service request to assign subtasks to a number of fog worker nodes and to return the aggregated results to the data user who requested the service. From now on, the term ''service'' and ''task'' are used interchangeably. A service request processing procedure is composed of 4 steps:

- Step 1: a data user requests a service to the manager.
- Step 2: the manager processes the service request by
  - Assigning subtasks to worker nodes: the manager will find the list of suitable computing nodes for doing the requested job. Note that this sub-step is optional if the computing service needs to be efficiently divided into subtasks for various worker nodes. In some applications, the group of worker nodes is known from the business logic of the application, e.g. in distributed learning or FL approach for AI-model training.
  - Sending notification to workers: the manager sends notifications to worker nodes about the data and the execution program.
- Step 3: the workers download the data, execute the computation task, and send back the results to the manager node. The data can be images, sensor-based measurements, or text files representing the AI model parameters, which are all considered input parameters for the computation task.
- Step 4: the manager aggregates the results and sends this aggregation to the data user who requested the service.

We design an AE-MANAGE, located at the manager node, to perform service allocation and coordination of multiple subtasks. We design an AE-COMPUTE at every worker node to call the specified service application programs whenever the information about the service is received at the worker node. The flow of control and data messages between manager and workers will be supported by the CSE layer in these nodes. Fig. 3 depicts the AEs and containers inside each manager's CSE and worker's CSE. We propose the following containers in the CSE layer:

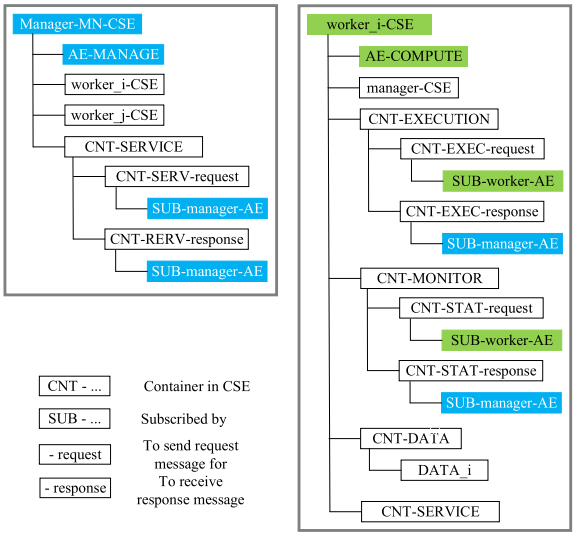

FIGURE 3. Resource trees in a manager node (left) and a worker node (right) consist of the essential containers for task execution and management, and for worker node resource monitoring.

- CNT-SERVICE: It is responsible for service-related information exchange. It receives the service requests from and feedback results to end users requesting services.
- CNT-MONITOR: It is responsible for updating the worker states to the manager, e.g., CPU, RAM, energy available or a kind of their representation in terms of time processing.
- CNT-EXECUTION: It is responsible for sending commands and data to the application programs at the worker nodes and sending back the obtained sub-results from the worker to the manager.
- CNT-DATA: It is used for data storage.

The operation of a worker node contains two phases: the initial (registration) phase and the computation phase. In the *initial phase*, the workers-CSE must successfully register to the manager-CSE to notify the manager about the workers' states. After registration, the manager AE-MANAGE will subscribe to all needed containers in the worker-CSEs (SERVICE, MONITOR, EXECUTION) through the publish/subscribe mechanism. Hence, whenever a container's content changes (e.g., service requests/responses, execution requests/responses, worker node states), *notifications* are sent to the AE-MANAGE for the purpose of task allocation and subtasks coordination. The worker AE-COMPUTE will subscribe to the CNT-EXECUTION at its CSE that it will

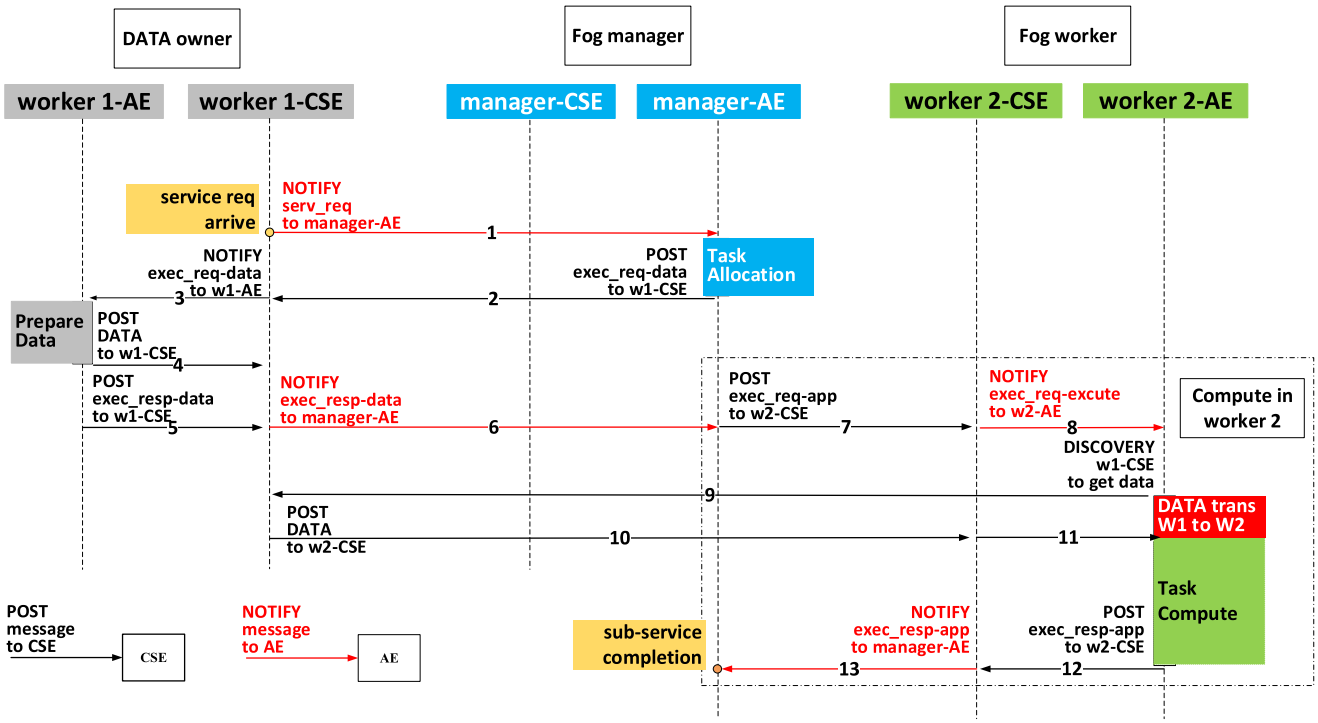

FIGURE 4. The procedures of manager node and worker nodes exchanging control/data messages to divide the computation job of a service request in subtasks and to execute the sub-service programs at worker nodes to minimize the service processing time.

be notified about the subtask data and subtask application program.

In the *computation phase*, a 4-step procedure is activated whenever a service request arrives. In this procedure, the manager and worker nodes exchange control and data messages and execute the sub-service programs at worker nodes. As shown in Fig. 4, the four steps are the following:

- the manager AE-MANAGE receives the notification about a service request from a data user AE. Based on that, it will either extract the payload or execute the task assignment and resource allocation (TARA) module to find the list of candidate computing nodes to process the subtasks and the corresponding subtask workloads. This information is sent to the data owner for data preparation.
- the manager AE-MANAGE sends to such a group of worker nodes the notifications including the data resources and the execution program resources.
- the worker AE-COMPUTE receives notifications and reads the information on the required resources for its part of the application program execution and calls the service program. The worker node can download the data from the data owner if needed.
- the worker-AE-COMPUTE executes the application program. After completion of computation job, the worker nodes notify the results to the manager-AE-MANAGE through the CNT-EXECUTION-response.

Multiple types of services addressed by pairs of *serv_ID* and *source_ID* can be processed at the AE-MANAGE. To load data for a computation job, a worker AE-COMPUTE can use a discovery mechanism with the universal resource indicator (URI) link being provided by the manager-AE-MANAGE. Our platform supports two different kinds of data transfer: i) data can be piggybacked in oneM2M packets, ii) data link resources are posted in oneM2M packets, and the receivers download the data through other protocols.

The CNT-MONITOR regularly monitors the worker node's state and will communicate updates to the manager-AE-MANAGE through the publish/subscribe mechanism. Node state includes RAM, CPU, energy level, current processing workload, service processing time, etc. Each worker node has its own policies for sharing its resources when joining the *oneVFC* platform. The MONITOR containers will make the resource capability information available to the task assignment and resource allocation (TARA) algorithm.

The TARA algorithm is executed every time the manager node receives a new service request. The TARA execution results in the subtask assignments to a group of the worker nodes. The assignments are adapted to the status of worker nodes, e.g., the availability of nodes for a group of specified services, the workload being processed, the node configuration (such as CPU speed, RAM) etc., as well as to the target being pursued with regard to service processing latency, energy consumption, etc. Different resource allocation approaches proposed in literature can be adopted in the TARA algorithm. In the following section, we propose a system model of task assignment and resource allocation to minimize service processing latency subject to the limited computing resources.

V. TASK ASSIGNMENT AND RESOURCE ALLOCATION (TARA)

A. SYSTEM MODEL FOR TASK ASSIGNMENT AND RESOURCE ALLOCATION (TARA)

A fog manager node will "manage" a set of computing resources called worker nodes, denoted by R. Those worker nodes have previously registered to the fog manager. The number of available worker nodes in R can vary because the manager must filter out worker nodes which are no longer available. The manager node will know that through regular updates on the status of worker nodes in R. The manager will try to use the available computation resources of the worker nodes under its supervision (nodes in R) to minimize the requested service processing time.

A data user k can request to the fog manager node to execute an aggregated task defined as a bundle of data W_k and a specified job to work on that data. Note that the service requesting node can be one of the worker nodes. The bundle of data can be subdivided to be processed in parallel at several nodes. Alternatively, the task can be divided into smaller subtasks $\{w_{ki} = p_{ki}W_k\}$ to be handled by the set of worker nodes $R = \{n_i\}$ managed by the manager node, and then the result will be aggregated by the manager and returned to the original requesting node k. Hence, the set $\{p_{ki}\}$ represents the partitions of the bundle of data requested by node k, which will be processed by the set of worker nodes $\{n_i\}$.

The subservice completion time at node i, called t_{ki}, will consist of three components,

$$t_{ki} = t^{ki}_{trans} + t^{ki}_{proc} + t^{ki}_{result}, \tag{1}$$

in which,

- t^{ki}_{trans}, denotes the workload transmission delay, is the time to relay data from the data owner k to the worker node i.
- t^{ki}_{proc}, denotes the subtask processing delay, is the time to process the task at the worker node i.
- t^{ki}_{result}, denotes the result response transmission delay, is the time to transfer the output analysis results to the manager or the node that requested the service.

The system performance will depend on the system configurations, e.g., bandwidth of transmission link for data/result transmission, the CPU speed, RAM storage, etc. of the computing resource. These relationships are formulated in the following analysis.

(1) The subtask workload transmission time

$$t^{ki}_{trans} = \frac{p_{ki}W_k}{B_{ki}} \tag{2}$$

The time needed for data of a subtask to be transferred from data owner k to worker node i is the ratio of the amount of data of the subtask and the transmission rate between the two nodes. The node k can also perform the entire task, then $i = k$, $B_{ki} = \infty$ so $t^{ki}_{trans} = 0$.

(2) The subtask computation time

$$t^{ki}_{proc} = \frac{p_{ki}W_k}{f_i} + \tau_{ki}\frac{N_i}{f_i} \tag{3}$$

Here, $\frac{p_iW_k}{f_i}$ represents the processing time of w_{ki} (which equals p_iW_k) data at node i with f_i being the processing speed. At any given time, node i may still be processing a certain amount of data of the previously assigned task, denoted by N_i, for that the processing time is calculated by $\tau_{ki}\frac{N_i}{f_i}$, where τ_{ki} can be 0 or 1. If $\tau_{ki} = 1$, there is a task previously scheduled and currently executed on node i. Then, the subtask computation time will include both the processing time of the newly assigned task and the task being processed. $\tau_{ki} = 0$ means there is no previous task. Hence the newly assigned task from node k can be processed immediately.

(3) The sub result transmission time

$$t^{ki}_{result} = \frac{R_{ik}}{B_{ik}} \tag{4}$$

After the subtask is completed, nodes i will send the results back to node k which is the node that originally executed the service request. This delay is calculated by the ratio of the message size R_{ik} and the transmission rate B_{ik} between two nodes.

(4) The subtask completion time

Therefore, the completion time of a subtask requested by node k to be performed by node i can be analytically presented as follows:

$$t_{ki} = \frac{p_{ki}W_k}{B_{ki}} + \left(\frac{p_{ki}W_k}{f_i} + \tau_i\frac{N_i}{f_i}\right) + \frac{R_{ki}}{B_{ik}} \tag{5}$$

Whenever the manager node receives the service request with an amount of W_k workload, the TARA module is executed to find out the number of subtasks, the subtasks' workload, and the list of suitable worker nodes. The subtasks are performed independently, mostly in parallel, the completion time of the service requested by node k is determined as follows:

$$t_k = max\,(t_{k1}, t_{k2}, \ldots, t_{kn}) \tag{6}$$

TARA, which has an objective the minimization of service processing time yields the following min-max optimization problem:

$$min\,(max\,(t_1, t_2, \ldots, t_n))\,, \tag{7}$$

Under the following constraints:

$$\sum_{i=1}^{n} p_{ki} = 1 \tag{8}$$

$$p_{ki}W_k \le \frac{W_k + \sum_{j=1}^{n} N_j}{n} \tag{9}$$

$$p_{ki}W_k \ge \alpha W_k \tag{10}$$

Constraint (8) is needed to ensure that the whole data bundle will be processed. The constraint in (9) is to avoid sending too much workload to a certain node leading to overload or

unbalance. We recommend the size of a subtask to be less than or equal to the average size of all tasks in the network. Note that, the fog manager node regularly monitors the currently executed workload $\{N_j\}$ of the computing resources.

The constraint in (10) is required to avoid that the size of a subtask is too small and needs a processing time much lower than the transmission time. This would make the task allocation ineffective, with large communication overhead and no gain in request completion time. To address that, the workload sent to a node needs to be no less than αW in which α is the parameter showing the relative order between processing time and transmission time which could be experimentally found.

Hence, the output result is set of $\{p_{ki}\}$ indicating the data bundle $\{W_{ki}\}$ assigned to the set of fog nodes managed by the manager.

B. TARA IMPLEMENTATION IN oneVFC

In the current version of *oneVFC*, we apply the particle swarm optimization algorithm [34] on the above defined min-max optimization problem and obtain the task assignments for the worker nodes. However, various approaches for the TARA model and algorithms can be implemented in the manager node as well.

The transmission rate B between any two nodes and the processing speed f (in Equation (5)) of a specified hardware have been provided by using the nominal values which in general do not reflect reality. Therefore, to estimate these parameters which are combined with specified hardware (e.g. processors, network switch, cables), we run several preliminary tests using the *oneVFC* platform. In these tests, the task with the requested workload W_k will be deployed on one worker node, and the transmission and the computation delays are measured. The approximation functions showing the relationship of the transmission and computation delay with the workload are obtained statistically. Based on the approximation functions resulting from the preliminary evaluations, the transmission rate B and the processing speed f in Equation (5) are experimentally estimated.

VI. TESTBED EVALUATION AND RESULTS

For demonstrating the well-functioning of our *oneVFC*, we deploy two use cases and analyze their performance.

- AI/DL-based model exploitation: object detection application based on CNN model,
- AI/DL model training: CNN model training process in FL approach.

The used CNN model is m-AlexNet model [35]. mAlexNet is a compact version of AlexNet [36] which is an early well-known DL model for object detection and object recognition purposes. mAlexNet has fewer convolutional layers and fewer parameters than AlexNet to trade off accuracy against computation cost. Then it is more suitable for binary object detection, two-class detection. We have chosen mAlexNet because of its reasonable computation load on Raspberry Pi hardware in our testbed.

Both computing services require two types of input components: i) data sets, i.e., the collections of images, videos; and ii) the model, generally represented in text files containing weight values and model parameters. The computing services can be distributed to various machines of the VFC platform to be processed in parallel. Afterwards, the results are collected and combined.

A. TEST-BED SETUP AND PRELIMINARY TESTS

To verify the proposed architecture and design, a testbed was built as a scale model of the reality. The testbed includes a fog manager node and five fog worker nodes. Due to the unavailability of 5G or DSRC communication, the underlying network supporting the communication among nodes is Wifi-based. The hardware and software configurations are described in Table 1. The deployment inside our laboratory room is shown in Fig.5. We also deploy several experiments in an outdoor environment (a university parking space) with WifiMESH solution.

TABLE 1. Lab-scale testbed configurations.

Components	Hardware	Software
Fog manager node	Laptop HP Elitebook 8470p, with Intel Core i5-3320M, SDRAM 8055 MB, SSD 256GB, and Wifi interface is Intel Centrino Advanced-N 6205 802.11 a/b/g/n (2x2)	Microsoft Windows 7 Ultimate, Java (TM) SE (build 1.8.0_221-b11), OM2M IoT Platform ver. 1.4.1, Python 3.7.3
Fog worker node	Raspberry Pi 4 Model B, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, 2GB SDRAM, Micro SD 32GB, Wifi 2.4 GHz and 5.0 GHz IEEE 802.11ac	Raspbian GNU/Linux 10, OpenJDK (build 1.8.0_212-8u212-b01-1+rpi1-b01), OM2M IoT Platform ver. 1.4.1, Python 3.7.3.
Network switch	CISCO, Linksys EA2700, Wireless-N, operating Bands 2.4 GHz and 5 GHz	
Wifi MESH	Aruba AP-135 wireless AP, Aruba Mobility Controller capabilities on 802.11n access points	

FIGURE 5. The testbed setup in laboratory room.

The value of the transmission rate B between any two nodes and the processing speed f (in Equation (5)) of specified hardware has been provided by the nominal values which in general do not reflect the real operation. Therefore, to better

estimate these hardware dependant parameters (e.g. processors, network switch, cables), we run several preliminary tests on our *oneVFC* platform. A specific workload is requested to a given worker node (Raspberry Pi kit), and transmission and computation delays are measured. The final results are the average over 30 identical and independent service requests. The service interarrival interval is chosen large enough to allow the completion of the image processing before a new service is requested.

Fig. 6 shows the computation time on the worker nodes and the data preparation and transmission time in function of the number of images processed by the miniAlexNet-based detection application (varying from 100 images to 500 images of about 200Kbytes). Estimates for average transmission and computation delay in function of workload are obtained. Based on those, the transmission rate B and the processing speed f in Equation (5) are experimentally estimated.

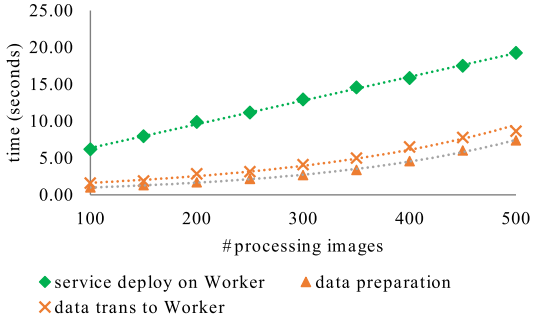

FIGURE 6. The computation time and transmission time (in seconds) versus the number of images processed by a CNN-based object detection program. The dotted lines depict the approximations for the computation time, data preparation time and data transmission time in function of the number of processed images.

B. CNN-BASED IMAGE PROCESSING APPLICATION DEPLOYMENT ON VFC

We evaluate the performance of the vision-based object detection application on the lab-scale testbed. The image data set of a thousand ∼2MBytes images is located at the worker node #1 depicted in Fig. 4. This data owner node sends its request to the manager node and the manager node runs the TARA module to divide the requested task into subtasks to be co-performed by several other worker nodes. The flowchart of request processing is depicted in Fig. 4. The procedures in a computing worker node are like the ones executed in worker #2 in Fig. 4.

The simulation scenarios are set up with the following parameters:

- The workload volume of a service request, represented as the number of processed images, varies from 100 to 500 images.
- The service request interarrival times are modelled through a uniform or exponential distribution with mean going from 5 to 20 seconds.
- The total number of worker nodes taking up the computation jobs created by a data owner (a worker node as well) varies from 1 to 5 nodes.

The performance parameter under study is the *service completion time,* calculated as the time between the moment the service request arrives at the manager and the moment of task completion at which the manager is receiving the results. The performance measures are averaged over 30 sequential service requests. The service completion time consists of several components being task assignment, data preparation control message transmission to worker node delay, as well as data transmission to worker node, service computation on worker node, and result message transmission to manager node delay. These delay components are monitored and investigated in detail thanks to *oneVFC's* functionalities.

Fig. 7 clearly shows the reduction of the *service completion time* when service requests arrive at high pace. Indeed, when the service request interarrival times are exponentially distributed with a mean of 5s and 10s, the reduction is around 84% and 60%, respectively. When the service requests arrive at low pace, with mean interarrival time of 20s, the reduction of the *service completion time* is only 20%.

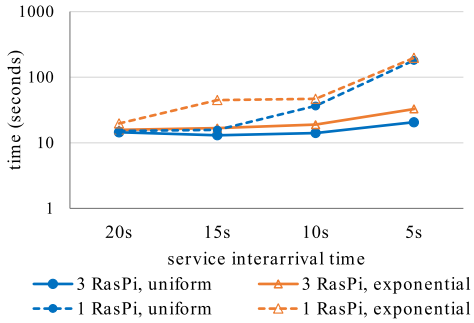

FIGURE 7. The comparison of service completion time when a CNN-based object detection program is running on 3 worker nodes and on 1 worker node versus the service request arrival rates. The results are averaged over 30 sequential service requests for processing 300 images.

The delay components of the *service completion time* in function of service request interarrival time, and in function of service workload (number of images processed) are depicted in Fig.8. The service execution delay at the worker nodes has the largest impact, whereas the control information transmission among the manager and worker nodes can be neglected. The other two delay components are the data transmission time from the data owner to the worker nodes and the time spent for data preparation (e.g., data compression before transmission). These two delay components are the so called *oneVFC* overhead delays, for a requested service to be deployed in parallel on various computing devices. Fig. 9 shows the delay components of the *service completion time* for a series of 50 requests arriving at high pace. Each service request demands an object detection for 300 images.

Fig. 8(a) shows that when the service requests arrive at high pace, the data processing and the overhead time increase, resulting in a high response delay for the requested services. Besides, if the workload of each service request is increased gradually from 100 to 500 images per request, the component delays all show an increasing trend, see Fig. 8(b).

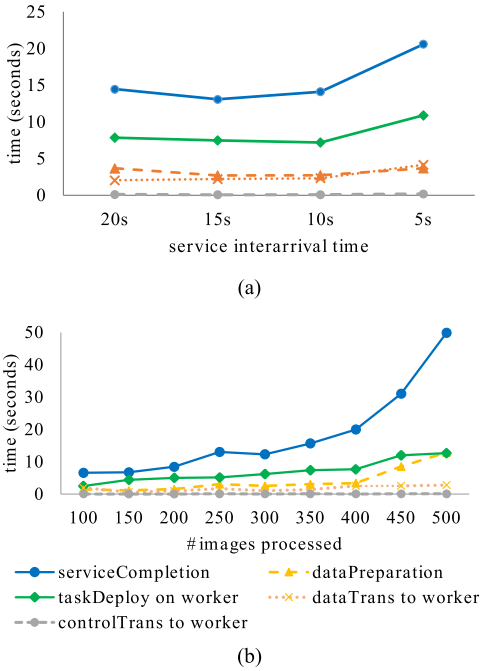

FIGURE 8. Service completion time and its components in function of (a) service request interarrival time and (b) service workload (the number of images to be processed) with service request interarrival time equal to 10 seconds. Tests are performed on a network with 1 fog manager node and 3 fog worker nodes, the underlying network in the lab is Wifi.

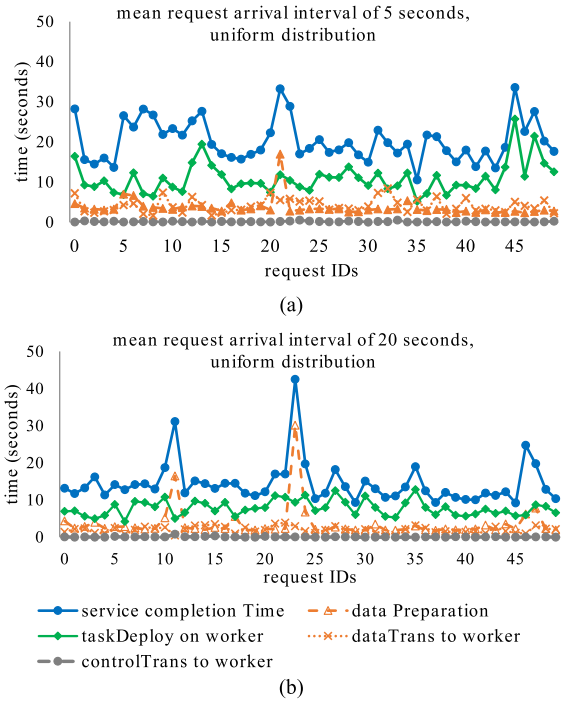

FIGURE 9. Service completion time and its delay components of each service request. The experiments are performed in two scenarios: (a) high pace, mean service request interarrival time of 5 seconds and (b) low pace, mean service request interarrival time of 20 seconds.

However, we observe that, at rather high workload (requests of 400-500 images arriving at high pace), the data preparation time has a rather sudden increment. The worker node which owns the data must do both jobs, data preparation to be sent for other worker nodes and service program execution for the subtask that is assigned to itself. Hence, the data preparation time gets higher quickly when service request workload gets higher. This effect is clear since vehicular computing devices usually have lower capability than server machines, hence this effect should be considered in the task assignment algorithms.

In our implementation, the transmission rate B and the processing speed f in (5) are experimentally estimated. Fig. 6 shows the preliminary tests to obtain these parameters, and these estimations do not consider service request arrival pace. These parameters might change when service request workload gets higher, or service requests arrive very densely. They should be estimated online to be adaptive with the current situation. This issue should be considered in task assignment and resource allocation when processing delay minimization is targeted (see section IV). Formulating the minimization problem under the assumed constant value of bandwidth and processing speed may cause non optimal results. The ability to monitor and report the states of computing nodes as well as the delay components in *oneVFC* should be exploited to determine the relationship between overhead cost and service workload and service request arrival rate.

When more nodes are willing to share their computation capabilities, the service completion time is clearly reduced as shown in Fig. 10(a). Every delay component that contributes to the service completion time is lower. However, when there are 5 computing nodes performing the subtasks, the service completion time slightly increases because data preparation and data transmission times are slightly higher. The fact is that when more computing nodes join the service computation, the data owner (worker #1) must put more effort in data preparation and transmission to other workers, resulting in a higher delay. Also, the service deployment time on worker #1 is higher, as can be seen in Fig. 10(b). It does again confirm that the interplay between transmission bandwidth, hardware processing speed and processed workload should be well investigated for achieving successful task assignment and resource allocation.

We also performed the simulation in an outdoor environment, in which two worker nodes are set up at fixed location and one is attached on a car travelling with average velocity of 20km/h in the limited area in our university campus. The underlying network is provided by Wifi MESH solution, including 4 access points operating in the 5GHz band. When a node is in the communication range of at least one access point, the communication to the other nodes connected with other access points is ensured. The outdoor measurements are confronted with the indoor Wifi ones in Fig. 11. The data transmission times to the mobile worker node increase largely, nearly 3 times in comparison to the measured values in the indoor environment. Indeed, the service completion time becomes larger than the one in the indoor setup, however, it is still much lower than the case of computing on one device. The outdoor setup is a proof of concept demonstrating

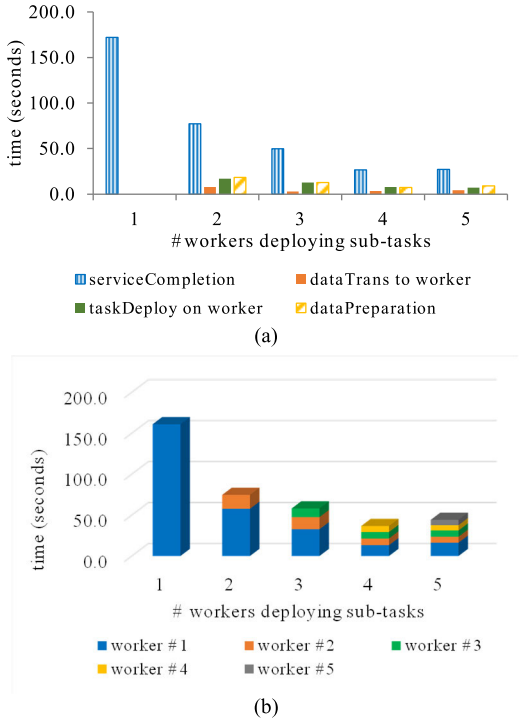

FIGURE 10. The service completion time and its delay components (a) and the subtask processing time on each worker node (b) in function of the number of worker nodes participating in computation jobs.

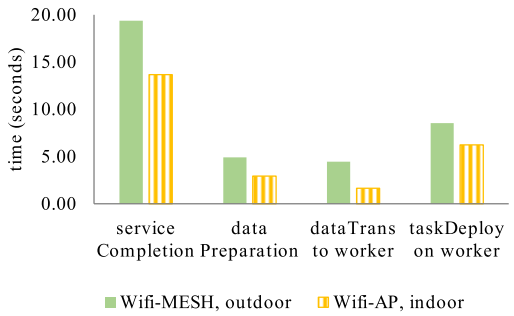

FIGURE 11. Service completion time and its delay components are measured in two scenarios, laboratory room environment and a Wifi access point to provide connections among fog nodes versus outdoor university parking space environment and a Wifi-MESH solution to provide connections for fog nodes including one mobile node.

that the *oneVFC* platform can support mobile computing nodes in outdoor environments whenever the network connections among fog nodes are guaranteed by the underlying network.

C. DEEP LEARNING MODEL TRAINING IN FEDERATED LEARNING APPROACH

Training a deep learning model can take a long time and usually requires a high end, or cloud-based server, especially when large training datasets are involved. In an FL approach, a so-called *training round,* depicted in Fig. 12(a), goes as follows: any edge node will train the model with its locally collected data to obtain the local model and send the local model parameters to the server node, the server aggregates the received local models to improve the global

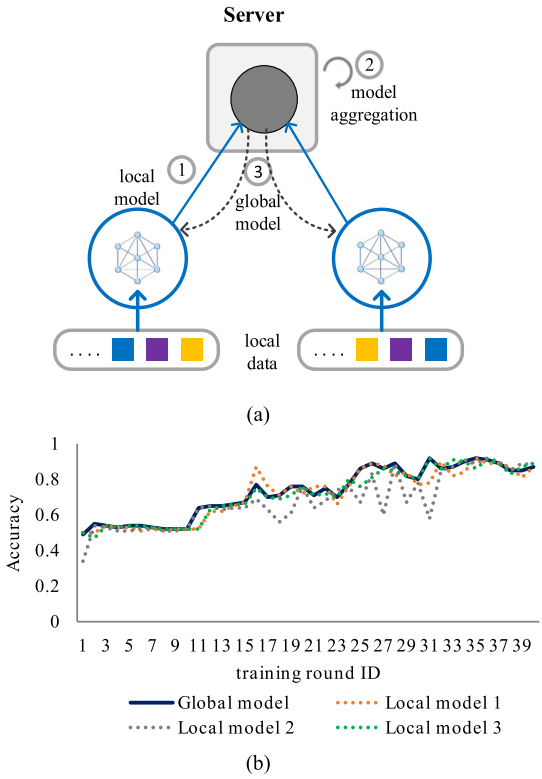

FIGURE 12. A training round of the Federated Learning approach in Deep Learning model training (a) and the learning curves of the trained global model at the server and the local models at the local devices progress through each training round (b).

model, and the server updates the global model to the local devices [28], [37], [38]. Those *training rounds* are repeated till the required model accuracy is achieved or till the maximum number of training rounds is reached.

A FL training approach does not require to transfer local data to the server, only model parameters are exchanged, preserving user data privacy which is a big concern in IoT/IoV environments. Model aggregation can follow different approaches, e.g. Federated Averaging [28], selective model aggregation depending on the data quality collected by travelling vehicles [39], [40]. These issues belonging to the business logic of each FL-based training process will be embedded in the application execution program, while the management of communication of model parameters among edge nodes and server will be performed by *oneVFC*. Possible application scenarios can involve several vehicle-worker nodes, taking the roles of edge nodes and collecting images/videos by means of their cameras while travelling on roads, and a gateway-manager working as server node.

We evaluate a mAlexNet model training with FL in which 3 worker nodes (Raspberry Pi kits) operate as local devices which store local dataset of images and have the DL model program installed. The mAlexNet model parameter file is exchanged between manager and workers. The manager node directs the FL based learning process and aggregates the feedback of the local devices to build the global model.

The request for FL training service will contain the list of worker nodes' addresses joining the training process and the dataset addresses in each node. At every training round, the execution of Python code for model training is called at the workers' AE-COMPUTE, and the Python code for model aggregation is called at the manager's AE-MANAGE. The model parameter file is exchanged among the manager-CSE and worker-CSEs through the EXECUTION containers as the procedures described in Fig.4. The parameters related to the simulation setup are shown in Table 2.

TABLE 2. Setup of federated-learning based training.

Deep Learning model		Federated Learning approach	
Model parameter file size (Kbytes)	Training data set (images) per local devices	Number of training rounds	Model aggregation policy
150	4000	40	FedAvg

The experimental measurements of the workers and the manager operations are shown in Table 3. The computation time at the worker nodes is the largest in a training round, whereas the data transmission among the worker nodes and the manager is small. The learning curves shown in Fig. 12(b) present the model accuracies of the local models and the global model aggregated at the manager node. One can see the accuracies gradually improve with increased number of training rounds.

TABLE 3. Time analysis of federated learning on various edge nodes.

Operation in manager node (seconds)		Operation in worker node (seconds)	
Model data transmission to manager	Python code execution	Model data transmission to worker	Python code execution
0.17	6.61	0.14	16.68

VII. CONCLUSION AND FUTURE WORK

In this work, we propose a vehicular fog computing (VFC) platform, in which the computation nodes, vehicle OBUs, are exploited to support ITS services. V2X technology, with the two standards- DSRC and cellular-V2X, which facilitate the real-time information exchange among cars and everything, is the main communication infrastructure for VFC. The proposed *oneVFC* platform is based on the oneM2M standard and has been designed to support AI/DL-based applications in ITS as well as to support AI/DL model training. *oneVFC* can manage the availability of distributed resources, orchestrate the information flows from data sources, control the computing tasks on vehicular nodes as well as feedback results to the application users. *oneVFC* can do real-time monitoring and overhead cost assessment which will help the task assignment algorithm to adapt to the current state of the computing machines.

In lab scale testbeds, we have demonstrated that *oneVFC* is able to manage the deployment of the AI-based applications on various machines resulting in high reduction of application processing time, especially when the workload is high, or when service requests arrive at high pace. Moreover, the *oneVFC* platform facilitates the deployment of the AI/DL model training in the Federated Learning approach. Concerning VFC, this work is the first attempt to apply the oneM2M standard to design and implement a platform to realize coordination of various computation resources on vehicles to support AI-based applications.

oneVFC has also been tested for a mobile node participating in the computation jobs in an outdoor environment. The results show that the computation job shared with the mobile worker node gets accomplished, however, the reduction of the service completion time is lower due to the larger data transmission times in the new environment and network setup. The underlying network is provided by an outdoor Wifi MESH solution due to the unavailability of DSRC or 5G-LTE. DSRC and 5G networks will soon be deployed offering data rates of about 20-30Mbps and over 100Mbps, respectively, which means that the issue of large data transmission times will be solved. Hence, the efficiency of distributed computing on various worker nodes under the support of our proposed *oneVFC* platform is confirmed.

For future work, several issues need further investigation. First, different schemes for task assignment and resource allocation should be integrated in *oneVFC*. Second, *oneVFC* should be evaluated in a professional vehicular network environment to assess the influence of variable communication conditions. Finally, the required security mechanisms to ensure data authentication and user privacy should be included and their corresponding impact needs to be evaluated.

REFERENCES

[1] Springer India-New Delhi, "Automotive revolution & perspective towards 2030," *Auto Tech Rev.*, vol. 5, no. 4, pp. 20–25, Apr. 2016, doi: 10.1365/s40112-016-1117-8.

[2] E.-K. Lee, M. Gerla, G. Pau, U. Lee, and J.-H. Lim, "Internet of Vehicles: From intelligent grid to autonomous cars and vehicular fogs," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 9, Sep. 2016, Art. no. 155014771666550, doi: 10.1177/1550147716665500.

[3] K. Kiela, V. Barzdenas, M. Jurgo, V. Macaitis, J. Rafanavicius, A. Vasjanov, L. Kladovscikov, and R. Navickas, "Review of V2X–IoT standards and frameworks for ITS applications," *Appl. Sci.*, vol. 10, no. 12, p. 4314, Jun. 2020, doi: 10.3390/app10124314.

[4] T. S. J. Darwish and K. A. Bakar, "Fog based intelligent transportation big data analytics in the Internet of Vehicles environment: Motivations, architecture, challenges, and critical issues," *IEEE Access*, vol. 6, pp. 15679–15701, 2018, doi: 10.1109/ACCESS.2018.2815989.

[5] Cisco, "Fog computing and the Internet of Things: Extend the cloud to where the things are," 2015. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[6] J. Lin, W. Yu, X. Yang, P. Zhao, H. Zhang, and W. Zhao, "An edge computing based public vehicle system for smart transportation," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12635–12651, Nov. 2020, doi: 10.1109/TVT.2020.3028497.

[7] K. S. Awaisi, A. Abbas, M. Zareei, H. A. Khattak, M. U. S. Khan, M. Ali, I. U. Din, and S. Shah, "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159100–159111, 2019, doi: 10.1109/ACCESS.2019.2950950.

[8] W. Zhang, Z. Zhang, and H. C. Chao, "Cooperative fog computing for dealing with big data in the Internet of Vehicles: Architecture and hierarchical resource management," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, Dec. 2017, doi: 10.1109/MCOM.2017.1700208.

[9] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016, doi: 10.1109/TVT.2016.2532863.

[10] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. S. Safa, and N. Zhao, "Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing," *IEEE Veh. Tech. Mag.*, vol. 12, no. 3, pp. 55–64, Sep. 2017, doi: 10.1109/MVT.2017.2667499.

[11] C. Zhu, Y.-H. Chiang, A. Mehrabi, Y. Xiao, A. Yla-Jaaski, and Y. Ji, "Chameleon: Latency and resolution aware task offloading for visual-based assisted driving," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9038–9048, Sep. 2019, doi: 10.1109/TVT.2019.2924911.

[12] S. Liao, J. Li, J. Wu, W. Yang, and Z. Guan, "Fog-enabled vehicle as a service for computing geographical migration in smart cities," *IEEE Access*, vol. 7, pp. 8726–8736, 2019, doi: 10.1109/ACCESS.2018.2890298.

[13] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11399–11403, Nov. 2019, doi: 10.1109/TVT.2019.2943647.

[14] G. M. S. Rahman, M. Peng, S. Yan, and T. Dang, "Learning based joint cache and power allocation in fog radio access networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4401–4411, Apr. 2020, doi: 10.1109/TVT.2020.2975849.

[15] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019, doi: 10.1109/JIOT.2018.2875520.

[16] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9364–9375, Sep. 2020, doi: 10.1109/TVT.2020.2970763.

[17] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019, doi: 10.1109/TVT.2019.2894851.

[18] Y. Zhang, C.-Y. Wang, and H.-Y. Wei, "Parking reservation auction for parked vehicle assistance in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3126–3139, Apr. 2019, doi: 10.1109/TVT.2019.2899887.

[19] Z. Zhou, H. Liao, X. Wang, S. Mumtaz, and J. Rodriguez, "When vehicular fog computing meets autonomous driving: Computational resource management and task offloading," *IEEE Netw.*, vol. 34, no. 6, pp. 70–76, Nov. 2020.

[20] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017, doi: 10.1109/MCOM.2017.1700322.

[21] oneM2M, "oneM2M: The interoperability enabler for the entire M2M ecosystem," 2015. [Online]. Available: https://www.onem2m.org/images/files/oneM2M-whitepaper-January-2015.pdf

[22] C. Paniagua and J. Delsing, "Industrial frameworks for Internet of Things: A survey," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1149–1159, Mar. 2021, doi: 10.1109/JSYST.2020.2993323.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[24] Labour and welfare and Ministry of Health. (2012). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. [Online]. Available: http://www.mhlw.go.jp/new-info/kobetu/roudou/gyousei/anzen/dl/101004-3.pdf

[25] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[26] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance analysis of Google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[27] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowl.-Based Syst.*, vol. 194, Apr. 2020, Art. no. 105596, doi: 10.1016/j.knosys.2020.105596.

[28] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," 2019, vol. 10, no. 2, *arXiv:1902.04885*. [Online]. Available: http://arxiv.org/abs/1902.04885

[29] S. Patonico, T.-L. Nguyen, P. Shabisha, A. Braeken, and K. Steenhaut, "Toward the inclusion of end-to-end security in the OM2M platform," *J. Supercomput.*, vol. 77, no. 4, pp. 4056–4080, Apr. 2021, doi: 10.1007/s11227-020-03415-7.

[30] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 4, pp. 1155–1165, Dec. 2020, doi: 10.1109/TCCN.2020.3002253.

[31] P. Sotres, J. Lanza, L. Sánchez, J. R. Santana, C. López, and L. Muñoz, "Breaking vendors and city locks through a semantic-enabled global interoperable Internet-of-Things system: A smart parking case," *Sensors*, vol. 19, no. 2, p. 229, Jan. 2019, doi: 10.3390/s19020229.

[32] E. Kovacs, M. Bauer, J. Kim, J. Yun, F. L. Gall, and M. Zhao, "Standards-based worldwide semantic interoperability for IoT," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 40–46, Dec. 2016, doi: 10.1109/MCOM.2016.1600460CM.

[33] S. S.-D. Xu, C.-H. Chen, and T.-C. Chang, "Design of oneM2M-based fog computing architecture," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9464–9474, Dec. 2019, doi: 10.1109/JIOT.2019.2929118.

[34] Y. Shi and R. A. Krohling, "Co-evolutionary particle swarm optimization to solve min-max problems," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, 2002, pp. 1682–1687, doi: 10.1109/CEC.2002.1004495.

[35] A. Farley, H. Ham, and Hendra, "Real time IP camera parking occupancy detection using deep learning," *Procedia Comput. Sci.*, vol. 179, pp. 606–614, Jan. 2021, doi: 10.1016/j.procs.2021.01.046.

[36] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," 2012, *arXiv:1803.01164*. [Online]. Available: https://arxiv.org/abs/1803.01164

[37] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," 2019, *arXiv:1911.02417*. [Online]. Available: http://arxiv.org/abs/1911.02417

[38] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Y. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[39] Y. He, J. Ren, G. Yu, and J. Yuan, "Importance-aware data selection and resource allocation in federated edge learning system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13593–13605, Nov. 2020, doi: 10.1109/TVT.2020.3015268.

[40] D. Ye, R. Yu, M. Pan, and Z. Han, "Federated learning in vehicular edge computing: A selective model aggregation approach," *IEEE Access*, vol. 8, pp. 23920–23935, 2020, doi: 10.1109/ACCESS.2020.2968399.

KIEU-HA PHUNG received the Bachelor of Engineering degree from the Department of Electronics and Telecommunication, Hanoi University of Technology, Vietnam, in 2001, the Master of Science degree in modeling and simulation of complex realities from the Joint Master Program between The Abdus Salam International Centre of Theoretical Physics (ICTP) and International School for Advance Studies (SISSA), Italy, in 2003, and the Doctor of Engineering degree from Vrije Universiteit Brussel, Belgium, in 2014. Since 2004, she has been a Researcher and a Lecturer at Hanoi University of Science and Technology (HUST), Vietnam. During that time, she conducted research on several topics, including traffic engineering techniques in MPLS for optical networks, energy-efficient communications in low-power wireless sensor networks, exploiting AI techniques to improve network performance, concentrating on routing protocols, medium access control protocols, link estimation, and cross-layer design. She is currently a Lecturer and a Researcher with the School of Electronics and Telecommunications, HUST. Her current research interests include wireless mesh technologies, LoRa, NB-IoT, DSCR, 5G for the Internet of Things, the Internet of Vehicles, intelligent transportation systems.

HIEU TRAN received the Bachelor of Engineering and Master of Science degrees in telecommunication engineering from Hanoi University of Science and Technology (HUST), Vietnam, in 2017 and 2019, respectively. He is currently a Research Assistant with the School of Electronics and Telecommunications, HUST. His research interests include low-power wireless transmission for the IoT and interoperability of the IoT systems.

THANG NGUYEN received the Bachelor of Engineering and Master of Engineering degrees from the Department of Electronics and Telecommunications, Hanoi University of Science and Technology (HUST), in 1996 and 1999, respectively, and the Ph.D. degree in IC fabrication from the University of Vermont, USA. Since he worked as a Researcher and a Lecturer at HUST, a Power Management IC Designer at Active Semi, a Hardware Designer at Viegrid, and a Group Leader at started up WICHIP, where he design H.264 video compression standard. He is currently a Lecturer with the School of Electronics and Telecommunication, HUST, where he has been a Faculty Member, since 1999. Recently, his research interests include machine learning, deep learning for computer vision, and intelligent transportation systems.

HUNG V. DAO was born in Hanoi, Vietnam, in February 1986. He received the B.E. degree in electronics and telecommunications and the M.S. degree in telecommunications engineering from Hanoi University of Science and Technology, Hanoi, in 2009 and 2011, respectively, and the Ph.D. degree in functional control systems from Shibaura Institute of Technology, Tokyo, Japan, in 2016. He has been working as a Lecturer with the School of Electronics and Telecommunications, Hanoi University of Science and Technology, since 2010. His current research interests include measurement and signal processing, medical instrumentation and rehabilitation robotics, and radiation detection and monitoring.

VINH TRAN-QUANG received the B.E. and M.S. degrees in electronics and telecommunications from Hanoi University of Technology, Hanoi, Vietnam, in 2000 and 2003, respectively, and the Ph.D. degree in computer science from Shibaura Institute of Technology, Japan, in 2009. He is currently an Associate Professor with the School of Electronics and Telecommunications, Hanoi University of Science and Technology. His research interests include wireless networks, *ad hoc* and sensor networks, mobile communications, and network security. He received the IEEE Section Prize Student Award, in 2008, the Osamu Omoto International Students Scholarship for Outstanding Student, in 2009, and the IEICE (ICM) English Session Award, in 2011.

THU-HUONG TRUONG (Member, IEEE) received the B.Sc. degree in electronics and telecommunications from Hanoi University of Science and Technology (HUST), Vietnam, in 2001, the M.Sc. degree in information and communication systems from Hamburg University of Technology, Germany, in 2004, and the Ph.D. degree in telecommunications from the University of Trento, Italy, in 2007. She came back to work for Hanoi University of Science and Technology as a Lecturer, in 2009, and became an Associate Professor, in 2018. Her research interests include oriented toward network security, artificial intelligence, traffic engineering in next generation networks, QoE/QoS guarantee for network services, green networking, and development of the Internet of Things ecosystems and applications.

AN BRAEKEN received the M.Sc. degree in mathematics from Ghent University, in 2002, and the Ph.D. degree in engineering sciences from the Computer Security and Industrial Cryptography Research Group (COSIC), KU Leuven, in 2006. She became a Professor with the Department of Industrial Sciences, Erasmushogeschool Brussel (currently since 2013, Vrije Universiteit Brussel), in 2007. She has cooperated and coordinated more than 15 national and international projects. She is the (co)author of over 150 publications. Her current research interests include security and privacy protocols for the IoT, cloud and fog, blockchain, and 5G security. She has been a member of the program committee for numerous conferences and workshops and an editorial board member for *Security and Communications Magazine*. In addition, she has been an expert reviewer for several EU calls, since 2015.

KRIS STEENHAUT (Member, IEEE) received the master's degree in engineering sciences, the master's degree in applied computer sciences, and the Ph.D. degree in engineering sciences from Vrije Universiteit Brussel (VUB), Belgium, in 1984, 1986, and 1995, respectively. She is currently a Professor with the Department of Electronics and Informatics (ETRO) and the Department of Engineering Technology (INDI), Faculty of Engineering, VUB. She is currently involved in several national and international the IoT/edge/cloud projects with industry and with academic partners in Europe, Vietnam, and Cuba. Her research interests include the design, implementation, and evaluation of wireless sensor and actuator networks for building automation, environmental monitoring, autonomous ground vehicle applications, mobility control and smart grids, taking into account security, and privacy aspects.