IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 28, 2021, accepted November 8, 2021, date of publication November 10, 2021,
date of current version November 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127355

A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks

FARHOUD HOSSEINPOUR[1], (Member, IEEE), AHMAD NAEBI[2], SEPPO VIRTANEN[1], (Senior Member, IEEE), TAPIO PAHIKKALA[1], HANNU TENHUNEN[1,3], AND JUHA PLOSILA[1], (Member, IEEE)

[1]Department of Computing, University of Turku (UTU), 20500 Turku, Finland
[2]System Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China
[3]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden

Corresponding author: Farhoud Hosseinpour (farhos@utu.fi)

This work was supported by the European Institute of Innovation Technology (EIT Digital) and the Academy of Finland under Project 311273 and Project 335512. The work of Farhoud Hosseinpour was supported by the Research Awards of the Nokia and Tekniikan Edistämissäätiö (TES) Foundations.

ABSTRACT While the effectiveness of fog computing in Internet of Things (IoT) applications has been widely investigated in various studies, there is still a lack of techniques to efficiently utilize the computing resources in a fog platform to maximize Quality of Service (QoS) and Quality of Experience (QoE). This paper presents a resource management model for service placement of distributed multitasking applications in fog computing through mathematical modeling of such a platform. Our main design goal is to reduce communication between the candidate nodes hosting different task modules of an application by selecting a group of nodes near each other and as close to the source of the data as possible. We propose a method based on a greedy principle that demonstrates a highly scalable and near-optimal performance for resource mapping problems for multitasking applications in fog computing networks. Compared with the commercial Gurobi optimizer, our proposed algorithm provides a mapping solution that obtains 93% of the performance, attributed to a higher communication cost, while outperforming the reference method in terms of the computing speed, cutting the mapping execution time to less than 1% of that of the Gurobi optimizer.

INDEX TERMS Greedy, fog computing, Internet of Things, modelling, optimization, resource management.

I. INTRODUCTION AND BACKGROUND

The past decade has witnessed a wide deployment of the Internet of Things (IoT) technology in various application domains, and its pervasive role will continue to strengthen in the future [1]. The emerging IoT applications, through various sensors, generate massive amounts of data that are generally referred to as big data. Traditionally, data is not processed in the proximity of a sensor but transferred as it is to a server that might be located in the cloud. However, transferring the constantly increasing amount of information from sensors to the cloud is not feasible. To overcome the intrinsic limitation of centralized data processing in cloud computing, a new paradigm called fog computing was introduced. Fog computing is characterized by heterogeneity, dynamicity, mobility, and geographical distribution that complement cloud computing services, providing local processing and faster response for delay-sensitive applications. It is considered a derivative of cloud computing that extends its services to the network's edge [2].

Fog computing does not replace cloud computing services but rather, by a well-organized interplay, complements the cloud computing services. Fog computing reduces the delay and response time for frequent and delay-sensitive local user requests. In contrast, cloud computing provides powerful computing resources and more extensive data storage for the global data collected from a larger geographical area. Figure 1 illustrates the basic architecture of a three-layered IoT system based on cloud and fog computing.

While the effectiveness of fog computing in IoT applications has been widely investigated in various studies [3]–[7], there is still a lack of techniques to efficiently utilize

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

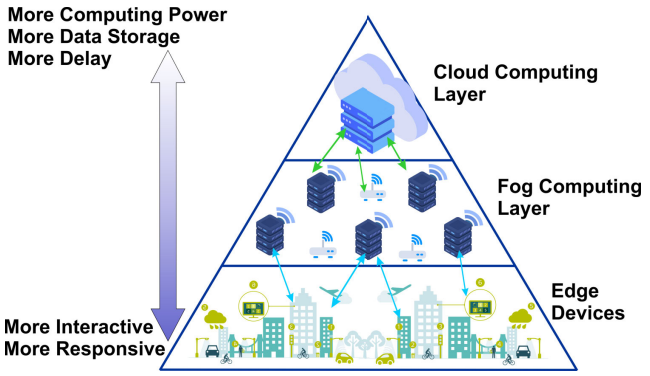

FIGURE 1. IoT architecture based on cloud and fog computing.

computing resources in this platform to maximize Quality of Service (QoS) and Quality of Experience (QoE). Inefficient scheduling and resource allocation for user services can actually result in even higher delays than sending the data for processing in the cloud [2]. Due to the aforementioned unique characteristics of fog computing, the resource allocation problem becomes more challenging and sophisticated. Understanding the nature of fog computing and the analogy between fog and cloud computing helps in systematic modeling of a fog computing framework and efficient resource management in the fog.

The fog layer is a dynamic and resource-constrained computing system, where both to-be-executed applications and executing mobile nodes can enter and leave the system at run-time, which results in high dynamicity of the workload and platform. Moreover, such applications and computing nodes are heterogeneous. Applications can be related to various services, and nodes range from switches and routers to base stations and even edge clusters or micro data centres [8], [9]. Such dynamicity and heterogeneity in both application and platform demand intelligent and agile coordination through a resource management system to achieve optimal or near-optimal performance at run-time.

A. RESOURCE MANAGEMENT

The resource management problem in fog computing has been widely investigated from different perspectives. The authors in [10] investigate the trade-off between the power consumption and delay in a cloud-fog computing system. They formulate the workload allocation problem in a cloud-fog scenario by modeling the power consumption and delay functions in the cloud and fog as well as communication delay function for dispatch. They conclude that cloud computing is more energy-efficient than fog computing while fog computing, due to the proximity to the users, can improve the performance of cloud computing by reducing communication latencies. In [11], the authors formulate a joint optimization problem for allocating the computation resources to maximize utility or the satisfaction rate and minimize carbon footprint for video streaming services in fog computing. As a solution, they propose a proximal distributed algorithm for large scale fog systems.

Communication costs between fog computing nodes are very important parameters to be optimized. Finding an optimal path to communicate the processing requests is a key task in building a robust resource allocation scheme in fog computing. The authors in [12] propose a Steiner tree based caching scheme to produce an optimal Steiner tree in a fog computing cluster to minimize the total cost of the communication path in a way that total cost of cashing resources is minimized.

The majority of the aforementioned works consider non-distributed applications running in a single device. IoT applications are typically involved with processing streams of data that are generated by sensors (i.e. data streaming applications). Stream processing involves applications that are developed as dataflow graphs that include task vertices executing some user-defined logic and streaming the messages between the tasks, i.e., distributed multi-task applications [9]. Such applications are characterized by their continuous processing requirements and computation-intensive nature. Offloading computation tasks into the fog computing or cloud computing layer, i.e., distributed processing, is an efficient solution to cope with limited resources of the edge devices.

Application partitioning is a key technology in the development of distributed applications for distributed computing systems such as cloud and fog computing systems [13], [14]. The partitioning can be done in different levels of granularity. In coarse-grained granularity, applications are divided into a set of loosely coupled units, called application modules, that can be integrated to create larger applications. Each module of an application runs in a computing node (typically within a container or Virtual Machine (VM)) and communicates its output data to other involved modules, processing data streams collaboratively with the other modules of that application which may reside at several different computing nodes around the network [6]. In fine-grained granularity, applications or modules of applications are further divided into smaller sub-tasks or processes (e.g. threads) that can be executed on different components of a heterogeneous computing node such as Central Processing Unit (CPU), Graphics Processing Unit (GPU), and Field Programmable Gate Array (FPGA [6], [15]. Such distribution, if not properly managed, could be delay-inducing and resource-intensive in geographically distributed fog computing networks [16].

The authors in [14] propose a framework for partitioning of applications on Mobile Cloud Computing (MCC) platforms to maximize the performance of applications in terms of speed/throughput as well as optimally utilize cloud resources. The authors in [17] consider a coarse-grained application partitioning method based on virtual machines that run as in cloudlet-based MCC environments. In [18], the authors develop a resource allocation model that provides optimal partitioning and offloading the application partitions into MCC systems. They consider several system parameters such as network traffic and processing resources in mobile devices,

mobile clouds, and the cloud to optimally decide where to run each partition of the application.

From the fog computing perspective, a distributed application composed of several modules needs to be mapped onto multiple computing nodes. The candidate processing nodes communicate the data related to that application among each other according to the data flow of the application. The authors in [19] study dynamic task module mapping, i.e., deployment, in a fog computing platform. They argue that, because of limited resources in fog computing platforms, splitting the requests from the users into smaller modules is a necessity. Their main goal is to find an optimal or near-optimal way to decompose the applications into task modules and efficiently map them onto the processing nodes.

It is evident that, in distributed processing, communication among the nodes will impose a delay that may affect the system's QoS and QoE. So, a very important challenge is how to assign applications' modules to a set of fog computing nodes resulting in minimal communication and execution delays, energy consumption, and network bandwidth usage.

In an IoT-fog system, an unpredictable number of applications with different sequences may require computing resources at any time. Therefore, allocation of computing resources to the arriving service requests needs to be done dynamically at run-time instead of design time. Satisfying power and performance constraints by allocation of resources for an application consisting of several communicating modules is a complex process. Moreover, the parallel execution of the task modules in the system adds to the complexity, potentially degrading the expected performance of the fog computing system if not properly managed. Determining a set of superior fog nodes to allocate an incoming distributed application with the least possible communication cost and delay is therefore crucial for ensuring high performance of the fog computing system. At run-time, a resource allocation unit (RTRA), manages this phase in two steps: i) *node selection*, that determines a set of neighboring nodes with adequate computing and storage capacity and bandwidth to be reserved for the new application, and ii) *task module mapping*, that forwards (distributes) the task modules of a particular application to the selected fog nodes.

The authors in [20] investigate the requirements of distributed task placement in fog computing networks. They point out that modeling both tasks and computing devices is necessary for studying task placement for fog computing. Also, development of distributed applications faces more challenges than that of monolithic applications, in terms of the communication complexity among each application's components that need to be addressed. They compare different algorithms for the tasks placement problem in distributed fog computing and conclude that greedy approaches, such as the Hill-Climbing algorithm, demonstrate better performance in solving the problem (higher speed), whereas genetic algorithms provide solutions that have better performance (higher quality). The authors in [21] propose an application placement technique for concurrent applications in a fog network. They present a weighted cost model for minimizing the execution time and energy consumption of multi-task applications in the fog computing context, supported by a pre-scheduling algorithm to maximize the number of parallel executions. The authors in [22] propose a module placement algorithm called MPC4.5 using the Markov Chain process in mobile fog computing networks. They consider an application consisting of multiple modules to be placed in a set of the most suitable fog nodes. Their proposed algorithm uses parameters such as authenticity, confidentiality, availability, capacity, integrity, cost, and speed as decision parameters for placing the application module in a fog node. In [23], the authors propose an application placement algorithm based on multidimensional QoE prioritization. They prioritize the incoming offloading requests based on three main domains: the environment runtime context, application usage, and user expectations, taking into account QoE and QoS. Then they choose a set of fog computing nodes for each requesting application based on proximity to the source of data, computing capabilities of fog nodes, and expected execution time for each application.

In [24], the authors present a cost-efficient resource management model for non-distributed applications in fog computing. They develop an optimal and heuristic (near-optimal) method to minimize the cost of offloading the applications in a fog computing platform. In their model, they consider the cost of deployment of a VM for each application as well as the communication delay for a given size of data. They conclude that an appropriate set of fog nodes to host the VMs for each application is a key factor for minimizing the cost of fog computing resource management. Inspired by their work, in this paper, we propose a novel resource management model for service placement of distributed multi-task applications in fog computing through mathematical modeling of this platform.

Our main design goal is to reduce the communication cost between the candidate nodes hosting different task modules of an application by selecting a group of nodes that are near to each other and also as close to the source of the data as possible. The communication cost considered in this paper is an abstract parameter to characterize the penalty of communication in distributed application execution, in terms of the size and range of involved communication events. The rationale is that the larger (smaller) this penalty is, the less (more) optimal the mapping is performance-wise. The adopted communication cost concept reflects (predicts) the fog system's behavior with respect to more concrete communication-related parameters such as communication delay or communication energy consumption.

Figure 2 shows an overlay architecture of our resource management model in an IoT-fog computing system. The physical network underlay illustrates the sensors streaming their data to the fog nodes that are associated with them. The virtual network overlay shows an overlay network controlled by a virtual machine manager (VMM) eliminating the routers that provide physical interconnections between the

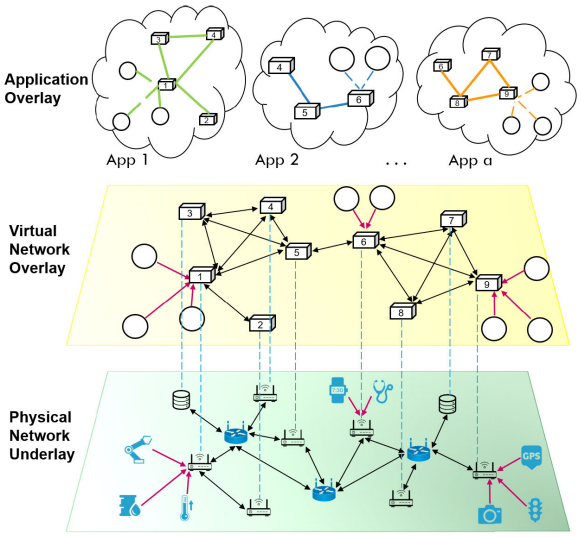

FIGURE 2. Overlay Architecture of fog computing network.

computing nodes. The task modules of each distributed application are mapped onto a set of fog computing nodes forming a separate overlay network, i.e., an application overlay, on top of the virtual network overlay.

The main contributions of this paper include:

- Mathematical modeling and presentation of distributed multi-task applications and fog computing systems.
- Formulating an optimization problem to reduce the overall communication costs in the system.
- Design and implementation of a heuristic greedy algorithm for the resource management problem for multi-tasking applications in fog computing networks.
- Performance evaluation of the proposed solution.

The rest of the paper is organized as follows. In Section 2, we present the research methodology of this study. In section 3 we present the mathematical modeling of the system and optimization problem formulation for multi-tasking resource management in the fog. In Section 4, we present our proposed algorithm for the problem. In Section 5, we provide performance evaluation for the proposed solution. Finally, in Section 6, we end with some concluding remarks.

II. METHODOLOGY

In this section, we briefly present the research methodology used in our study.

- **Modeling the applications and network:** We describe our approach for modeling an application as a composition of smaller task modules. In this study, we do not present a new method for application partitioning, but rather we consider an application comprising an ensemble of multiple task modules from the beginning. We also model our fog computing network consisting of nodes that are randomly distributed into a limited physical area.
- **Problem formulation:**We present the factors that constrain our model and formulate an optimization problem for resource management, in the context of task mapping, with the objective of minimizing the overall communication cost.
- **Heuristic method:** In order to validate our model, we propose a heuristic greedy algorithm for solving the formulated resource management problem.
- **Experimental analysis:** Finally, we investigate our proposed model from different points of view. We compare the properties of a method providing the absolute optimal mapping solution with the results obtained by the proposed heuristic greedy-based algorithm.

TABLE 1. List of notations.

Constants	
F	The fog node set
S	The sensor set
A	The application set
T	The task module set
ρ	Wireless communication range for each fog node or sensor
$\mathbf{A} \in \{0,1\}^{A\times T\times T}$	A binary-valued tensor that represents communication between task modules of an application
$\mathbf{R} \in \{0,1\}^{A\times T}$	A binary-valued incidence matrix representing the associations between task modules and applications
$\mathbf{Z} \in \{0,1\}^{S\times F}$	A binary-valued matrix that indicates if a fog node f is directly reachable from a sensor s
$\mathbf{N} \in \mathbb{N}^{F\times F}$	A distance matrix that represents the connections between fog nodes in the network.
$\mathbf{u} \in \mathbb{R}^{F}$	Vector of up-link costs of the fog nodes.
$\boldsymbol{\lambda} \in \mathbb{R}^{A}$	Vector of streaming rates of the applications.
$\mathbf{d} \in \mathbb{R}^{A}$	The length of data package uploaded to an application a
$\mathbf{c} \in \mathbb{R}^{T}$	Vector of container sizes of the task modules.
$\mathbf{h} \in \mathbb{R}^{F}$	A vector consisting of the hard disk (storage) capacities of fog nodes
Variables	
$\mathbf{M} \in \{0,1\}^{A\times T\times F}$	A binary-valued tensor variable that indicates if a task module t from an application a is mapped onto a fog node f

III. SYSTEM MODELING

In this section, we introduce our fog resource management model. For the convenience of the readers, the main notations used in this paper are listed in Table 1.

A. APPLICATION MODEL

We model an application in a fog computing system as an ensemble of task modules with inter-dependencies. Each task module $t \in T$ is a single function/portion of an application that receives input data, provided by precedent tasks, and

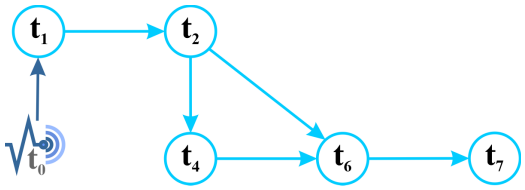

FIGURE 3. Task graph model of an application.

produces specific output data and sends it to the subsequent tasks. Hence, application mapping in a fog computing system follows a *many-to-many* function that should be calculated based on the topology of the network and the current locations of the sensors. We consider each task module developed as a bundle including a lightweight virtual machine, i.e., a container, and the source code. We define each application in the system as a directed graph.

We employ a simple mathematical model for representing applications running in the system. Let us denote by $\mathbf{A} \in \{0,1\}^{A\times T\times T}$ the tensor, whose each horizontal slice $\mathbf{A}_{a,:,:} \in \{0,1\}^{T\times T}$ is an *adjacency matrix* that represents the task graph of application a. Each entry $\mathbf{A}_{a,t_s,t_d}$ in the tensor represents communication between pairs of tasks (i.e., source and destination tasks) within the task graph of the application a that receives data from a particular sensor. To mark the first task module that receives the data from the sensor, we consider a dummy task module that we call the source node in the application's task graph, i.e., t_0, to represent the streaming sensor in $\mathbf{A}$. In this case, the task module that receives the data from t_0 will represent the first task module in the application task graph. Accordingly, an incidence matrix $\mathbf{R} \in \{0,1\}^{A\times T}$ can be obtained that represents the set of task modules composing the application. The following example (Figure 3) shows an application task graph for a given application and a set of associated sensors. The application a_1 is composed of following five task modules: t_1, t_2, t_4, t_6, and t_7, i.e., $\mathbf{R}_{a_1,:} = [1,1,1,0,1,0,1,1]$, where the involved tasks are a subset of $T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. The adjacency matrix $\mathbf{A}_{a_1,:,:}$ represents the task graph of this application in our system:

$$\mathbf{A}_{a_1,:,:} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

B. NETWORK MODEL

A fog computing network comprises heterogeneous computing nodes that communicate with each other directly or indirectly through network routers forming multi-hop paths. We denote by $\mathbf{N}$ a *distance matrix* to represent our network. Each element in this matrix represents the connection between a pair of fog computing nodes. Here, the values equal to 1 indicate a direct connection, i.e., the nodes are either physically connected or within the wireless communication range of each other. On the other hand, the values greater than 1 represent indirect connections through intermediate fog nodes or routers. In this case, the value is equal to the hop distance in the shortest path between two nodes. Two fog nodes are *reachable* from each other if there is a direct or indirect (multi-hop) connection between them. Without loss of generality, we assume that all the fog nodes in the network are reachable from each other.

1) SENSOR-NODE COMMUNICATION

Without loss of generality, we assume that all fog nodes and sensors are using wireless technology for communication. We define a radius ρ for each fog node and sensor that determines its communication range. Fog nodes $f \in F$ and sensors $s \in S$ can communicate with each other only if they are within the communication range of each other. Let us denote by $\mathbf{Z} \in \{0,1\}^{S\times F}$ a binary *incidence matrix* between sensors' set S and fog nodes' set F that indicates if a sensor $s \in S$ and a fog node $f \in F$ are within the communication range of each other, i.e., directly reachable from each other.

For the sake of simplicity, we assume that each application receives data from only one sensor, and the other way around, each sensor streams the data only to one application. That means that there is one and only one application associated with each sensor and vice versa. So, there is a one-to-one relationship between an application and its corresponding sensor. This being the case, we can refer to the application and sensor interchangeably in our model. Sensors stream their data as packages with the length of $\mathbf{d}_a, \forall a \in A$, to the fog nodes $f \in F$ they are associated with, in specific intervals with an up-link cost of $\mathbf{u}_f, \forall f \in F$ and an up-link streaming rate of $\boldsymbol{\lambda}_a, \forall a \in A$. Sensors $s \in S$ can stream their data only to the nodes $f \in F$ that are directly reachable from them, i.e., to the nodes for which $Z_{sf} = 1$ holds. Each sensor $s \in S$ should be associated to one and only one fog node at the time to ensure that the network traffic is not overloaded redundantly.

C. MOTIVATION

Let us consider a toy example to demonstrate the main motivation for this work. Figure 4 illustrates how a distributed application is mapped onto a network. For simplicity and for the sake of illustration, we consider a network with a 4×4 grid-mesh network topology. A wearable ECG sensor needs to offload its data to the fog network for processing. The sensor is within the communication range of the nodes 1 and 2 in the fog network. The ECG processing application is composed of 5 task modules, i.e., $\mathbf{R}_{a,:} = (1,1,0,1,0,1,1,1)$. First, the sensor will be associated with one of the fog nodes within its range. In our example, the sensor gets associated with the fog node 1. The resource mapping algorithm needs then to select a group of fog computing nodes that can accommodate and process the containers of the task modules of the ECG processing application, providing them

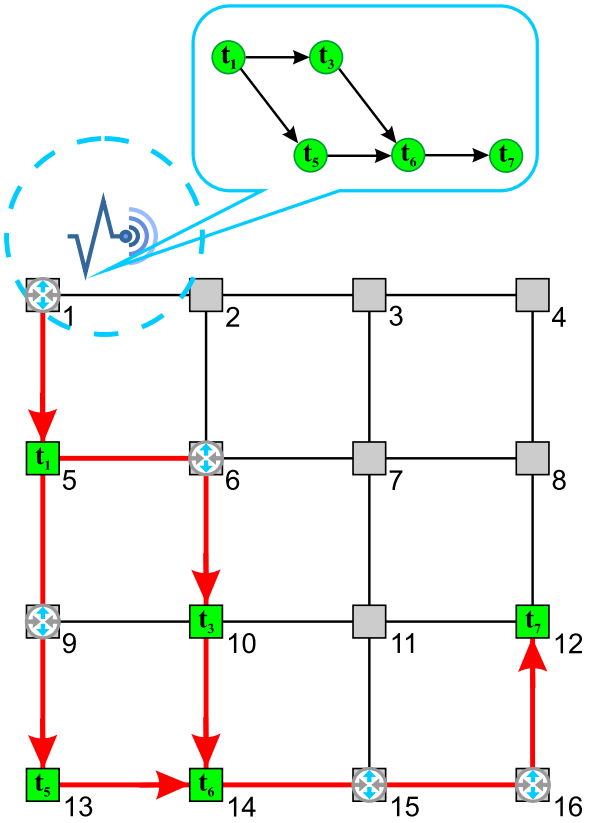

FIGURE 4. Distributed Multi-Task mapping example.

with appropriate communication paths. In our example, the fog nodes 5, 10, 12, 13, 14 are the selected nodes. Since fog nodes in general have limited resources in terms of storage capacity and processing power, some nodes cannot be chosen to host the containers of the task modules. These nodes, instead, can act as intermediate routing nodes. In the example, the nodes 6, 9, 15, 16 are such routers.

In this work, our optimization goal is to reduce the communication costs and delays between the task modules of an application by choosing the nodes that are as close as possible to the involved sensors and in the proximity of each other.

D. PROBLEM FORMULATION

In this section, we define and formulate the resource management and task mapping problem for distributed applications in fog computing systems.

An application is partitioned into smaller task modules that can be deployed as containers at fog computing nodes in the system. Each task module within an application needs to be mapped onto a selected fog node. We define a binary-valued tensor variable $\mathbf{M} \in \{0,1\}^{A\times T\times F}$, that indicates whether the task module $t \in T$ from the application $a \in A$ is mapped onto the fog node $f \in F$, i.e.,

$$\mathbf{M}_{a,t,f} = \begin{cases} 1, & \text{task module } t \text{ from application } a \text{ is mapped} \\ & \text{onto fog node } f \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

An obvious base requirement for this mapping is that the considered task module t is part of the considered application a. This condition can be expressed as:

$$\sum_{f\in F} \mathbf{M}_{a,t,f} = \mathbf{R}_{a,t}, \quad \forall a \in A,\ t \in T\ . \tag{3}$$

Since the applications are entering the fog computing system dynamically at run-time, assigning multiple task modules of an application to the nearest fog node will decrease the QoS of applications that will join later. In this case, the nearest fog nodes will soon be fully loaded with the tasks of the earlier applications, and, consequently, newly arriving applications will need to be mapped onto the fog nodes far away from their data sources, resulting in higher communication costs and delays. To avoid this condition and to effectively granulate the QoS of all involved applications, our premise is that one and only one task module of an application a can be mapped onto any given fog node. However, this constraint does not apply to a (dummy) source task t_0 representing a gateway that passes sensor data to an application. In other words, such a gateway node could also be a candidate for processing one of the computational task modules of this application. The constraint is formulated as follows:

$$\sum_{t\in T\setminus\{t_0\}} \mathbf{M}_{a,t,f} \le 1, \quad \forall a \in A,\ f \in F\ . \tag{4}$$

A sensor can be associated to a fog node, if and only if the fog node is reachable from the sensor. To ensure this, we define the following constraint:

$$\mathbf{M}_{a,t_0,f} \le \mathbf{Z}_{s_a,f} \quad \forall a \in A,\ f \in F \tag{5}$$

Mapping and deployment of containers of the task modules is possible only if there is enough disk/storage space in the candidate fog nodes. This condition can be expressed as:

$$\sum_{a\in A}\sum_{t\in T} \mathbf{M}_{a,t,f}\mathbf{c}_t + \mathbf{M}_{a,t,f}\mathbf{d}_a \le \mathbf{h}_f \quad \forall f \in F\ , \tag{6}$$

where $\mathbf{c}_t$ is the container size of a running task module t, and $\mathbf{h}_f$ is the hard disk (storage) capacity of a fog node f.

E. MIQP FORMULATION

In this work, we associate resource management and task mapping in fog computing with the objective of minimizing the overall communication cost in the system. In the optimum case, the data is processed in the fog nodes that are in the proximity of each other and as near as possible to the sensors providing the data streams. The total communication cost in the system is calculated as the sum of the up-link and inter-node communication costs for every application that receives data from sensors, i.e.,

$$C_{Com} = C_{Up-link} + C_{Inter-Node} \tag{7}$$

The communication cost between two points in the network is determined by the amount and rate of the data transferred between the two points as well as the hop distance between the two respective nodes. In the case of up-link

communication, since the sensors are directly connected to the receiving fog nodes, we consider the hop distance equal to 1. For inter-node communication, we use the hop distances stored in **N**. Hence, the total communication cost can be calculated as:

$$C_{Com} = \sum_{a\in A}\sum_{f\in F} \mathbf{u}_f \mathbf{M}_{a,t_0,f} \boldsymbol{\lambda}_a \mathbf{d}_a + \sum_{a\in A}\sum_{t_s\in T}\sum_{t_d\in T}\sum_{f_s\in F}\sum_{f_d\in F} \mathbf{M}_{a,t_s,f_s}\mathbf{N}_{f_s,f_d}\mathbf{M}_{a,t_d,f_d}\mathbf{A}_{a,t_s,t_d}\boldsymbol{\lambda}_a \mathbf{d}_a \tag{8}$$

Our goal is to minimize the overall communication cost by choosing the best setting for **M**. Since the distance dependence in the inter-node communication cost function is quadratic with respect to **M**, we can formulate a mixed-integer-quadratic-programming (MIQP) problem as follows:

$$\begin{aligned} &MIQP: \\ &\min: \ (8), \\ &\text{subject to}: \ (3), (4), (5), (6) \\ &\text{and}: \mathbf{M} \in \{0,1\}^{A\times T\times F} \end{aligned} \tag{9}$$

IV. SOLUTIONS

The MIQP problem defined in the previous section can be solved in different ways. To achieve an optimum solution for a given resource management task, we implement our multi-task resource management model using the Gurobi optimizer. However, since the search space grows exponentially by adding more fog computing or sensor nodes to the system, an optimum solution may not be feasible for real-world scenarios. The fog computing nodes are resource-constrained, and the dynamic decisions for resource mapping need to be done swiftly to reduce the service delivery time and to contribute to better QoS. Hence, to evaluate our model's efficiency and performance, we propose a greedy algorithm as a heuristic method that guarantees a near-optimal solution which is less computationally intensive and faster to achieve, more suitable for a resource-constrained computing environment.

A. PROPOSED GREEDY ALGORITHM

We propose a two-phase distributed greedy algorithm as a heuristic solution for our resource management model. In the first phase presented in Algorithm 1, the system greedily associates the sensors to the fog nodes considering the constraints (5) and minimizing the system's overall up-link cost. In the second phase, at each iteration, the greedy algorithm assigns a fog node with the minimum communication cost for one task module from each application considering the constraints (3), (4), (6). At each iteration, the algorithm stores the next node(s) in a queue based on the sequence defined in each application's task graph. Once all the task modules have been mapped onto fog nodes, a communication graph from the selected fog nodes for processing each multitask application is formed, and the overall cost is calculated based on Equation (8). The greedy algorithm is presented in more detail in Algorithm 1.

Most of the tensors and matrices used in our approach are very sparse, meaning that they mostly comprise zero values. Consequently, to improve the performance and simplify the complexity of the algorithm, we only consider the non-zero values in the tensor M that together represent the total number of tasks modules from all the applications that are allocated to the fog nodes. This corresponds to the total number of elements that are pushed into the queue in Algorithm 1. Similarly, we consider only the non-zero values in the matrix Z that represent the number of all possible connections that one sensor could have with the fog nodes within its range. So, if we denote τ to correspond to the number of non-zero values in the tensor M and Z where $a \leq \tau \leq at, \forall a \in A, t \in T$, the complexity of the first and second phases could be calculated as $O(\tau)$, and $O(f\tau), \forall f \in F$ respectively. So the total complexity of our algorithm in the worst case is $O(f\tau), \forall f \in F$.

Algorithm 1 Greedy Algorithm

Data: Communication radius and coordinates of fog nodes and sensors
Result: $\mathbf{M}_{a,t_0,f}$
▷ **Phase 1:** Sensor association;
for *each non-zero value in* Z_{sf} **do**
  Calculate the up-link cost;
  $f \leftarrow$FogNode with the minimum up-link cost from s;
  $\mathbf{M}_{a,t_0,f} \leftarrow 1$
  Queue.PUSH $\leftarrow$ [a, 0 and, the candidate gateway fog node]
end
▷ **Phase 2:** Task module mapping;
$L = [\,]$ // a list that contains the Fog Nodes that are already assigned for a task module of the application
while *There is an item in the Queue* **do**
  $a, t_s, f_s \leftarrow$ *Queue.POP*
  for *each unallocated task module* t_d *acting as a destination of* t_s *in application* a **do**
    $f_d \leftarrow$ find a fog node closest to f_s that is not in L
    Queue.PUSH $([a, t_d, f_d])$
    $L.append(f_d)$
    update $\mathbf{M}_{a,t_d,f_d} \leftarrow 1$
  end
end

V. EXPERIMENTAL SETUP AND EVALUATION

To evaluate the efficiency of the proposed solutions in a small-scale network of 100 × 100 meters, we consider 10 fog nodes and 3 sensor nodes in a field with a uniform distribution. The wireless coverage for the fog nodes and sensor nodes is set to 40 and 20 meters, respectively. Each sensor is associated with a multi-task application with the task graph

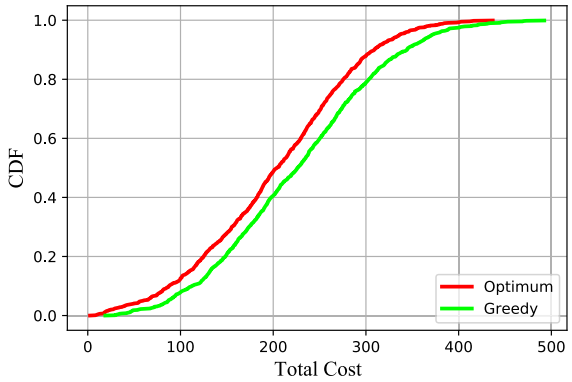

FIGURE 5. Cumulative Distribution Function of total cost for Optimum solution and Greedy algorithm.

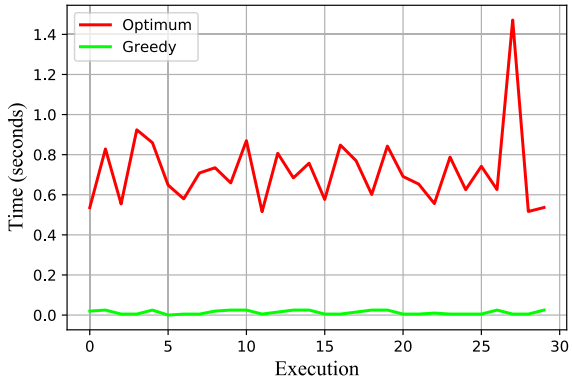

FIGURE 6. Execution time.

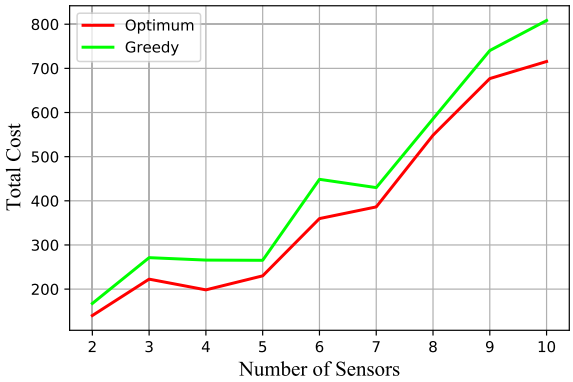

FIGURE 7. Effects of number of sensors on total cost.

presented in Figure 3. The data streaming rate from each sensor is assigned based on a Gaussian distribution with a mean and standard deviation of 1.2 and 0.8, respectively. The container size for each task module is set to 0.4, and the storage capacity of each fog node is normalized to 1. The up-link cost associated with each fog node is also assigned based on a Gaussian distribution with a mean and standard deviation of 1.2 and 0.8, respectively.

We use a commercial solver, Gurobi 9.1, for solving the MIQP problem for the optimal case. We run 1000 simulation instances with different random seeds and plot the Cumulative Distribution Function (CDF) of both algorithms' total costs in Figure 5. According to this figure, our greedy solution achieves near-optimal results, with the CDF reaching on average 93.2% of the optimal value. This validates the correctness and efficiency of our algorithm.

To validate the performance of our greedy algorithm, we also record the execution time for 30 simulations with the same setup. Figure.6 illustrates the comparison of the execution time for both optimal and greedy methods. Our greedy heuristic algorithm achieves considerably lower and more predictable execution time than the optimal method, cutting the execution time for resource mapping to 0.97% of the time required by Gurobi-based optimization on average.

We visualize the task mapping for the optimal and greedy algorithms for 3 sensor nodes with the same setting that we described above. Figure 15 visualizes the solution by our greedy algorithm, and Figure 16 visualizes the optimal solution obtained by the Gurobi optimizer.

To evaluate the effects of a network setting on the average total cost in both greedy and optimal solutions, we run 10 simulations for each network setting and obtain the average values for the total cost. Figure 7 illustrates the average total cost under different settings for a group of sensors, varying from 2 to 10 sensors. To make the model feasible for our larger experiments, we set the storage capacity of each fog node to 10, a relatively large arbitrary value. We keep the rest of the setup the same as in our earlier experiments. It is evident, based on this diagram, that the total cost generally increases when increasing the number of the sensor nodes, as can be expected. However, the somewhat surprising local decline in the diagram in the case with 4 sensors in the optimum cost curve, and with 7 sensors in the greedy cost curve, can result from a placement of sensors in an area where the density of fog node distribution is higher. In such conditions, the majority of inter-node communication is either direct involving no intermediate nodes, or indirect involving only few intermediate nodes, leading to a lower total cost due to a lower average communication cost between task modules of applications associated with the sensors.

Figure 8 illustrates the average total cost with respect to varying rates of arriving data. In this experiment, the number of fog nods and sensors are set to 10 and 3, respectively. As can be seen, the average total cost for the greedy algorithm is very near to the optimal solution and increases with the arrival data rate.

To investigate the scalability of our model, we run a set of experiments to examine the CPU time and maximum used memory under different network settings. Figures 9 and 10 illustrate the CPU time and maximum used memory, respectively, with the number of sensors varying from 10 to 100. In this experiment, we set the number of fog nodes to 50. It is evident that our greedy algorithm outperforms the Gurobi solver, that computes the optimal solution, by requiring

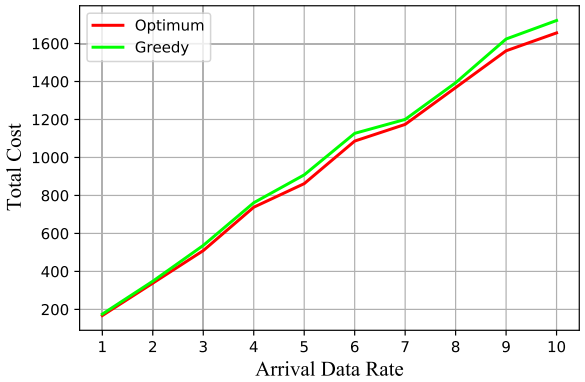

FIGURE 8. Effects of arrival data rate on total cost.

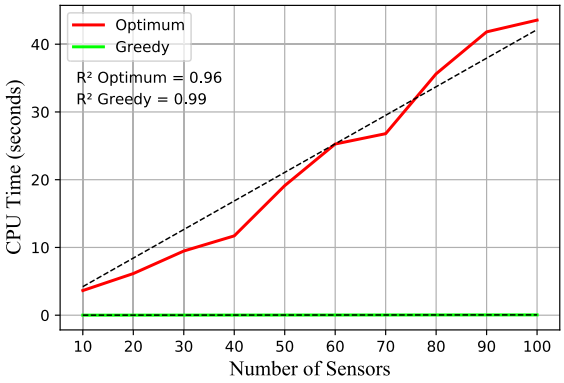

FIGURE 9. Effects of number of sensors on CPU execution time.

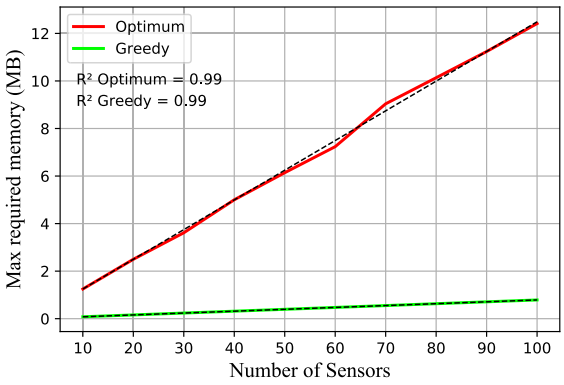

FIGURE 10. Effects of number of sensors on maximum required memory.

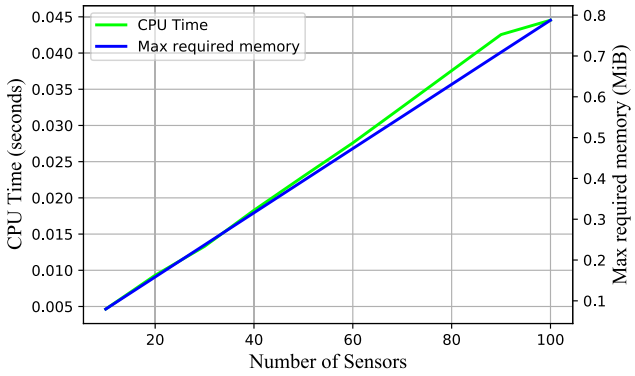

FIGURE 11. CPU time vs maximum required memory of greedy algorithm with respect to the number of sensors.

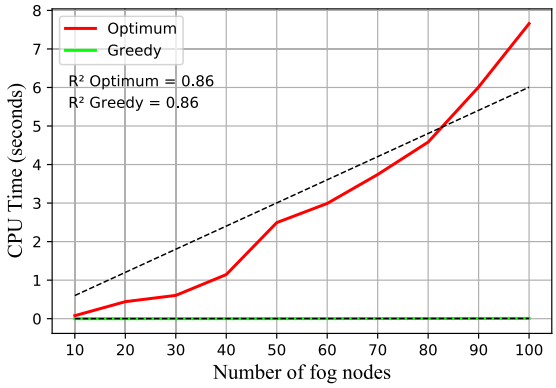

FIGURE 12. Effects of number of fog nodes on CPU execution time.

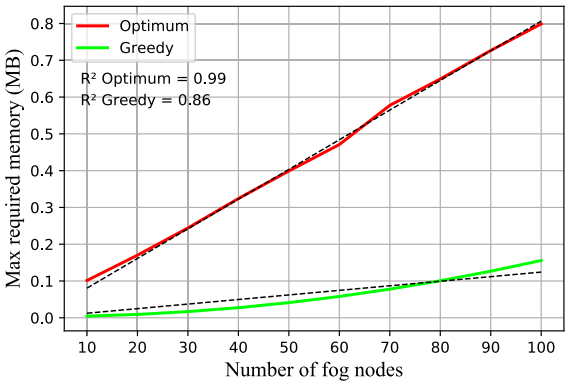

FIGURE 13. Effects of number of fog nodes on maximum required memory.

significantly less resources (i.e., on average 99.88% less CPU usage and 93.67 % less memory usage) for computing a task mapping solution. The resource usage in Gurobi-based optimization shows near-linear dependence on the number of sensors with a much steeper slope than in the case of the greedy method. Since the CPU time of the greedy method is almost constant (i.e., a horizontal line) in this experiment, we take a closer look into the resource usage of our algorithm in Figure 11, where we can see that for 10-100 sensors, the CPU time remains below 45 ms, and the memory need remains below 1 MB, indicating a relatively slow increase of the requirements. According to the experiment, for Gurobi, every additional set of 10 sensors increases the required CPU time by 4434 ms and the memory need by 1.24 MB on average. In comparison, for our greedy algorithm, these values are 4 ms and 0.078 MB, respectively. This provides evidence on the scalability of our algorithm with respect to the number of sensors in the network.

Figures 12 and 13 illustrate the CPU time and maximum used memory, respectively, with the number of fog nodes varying from 10 to 100 and the number of sensors having the constant value of 3. Also in this experiment, our greedy

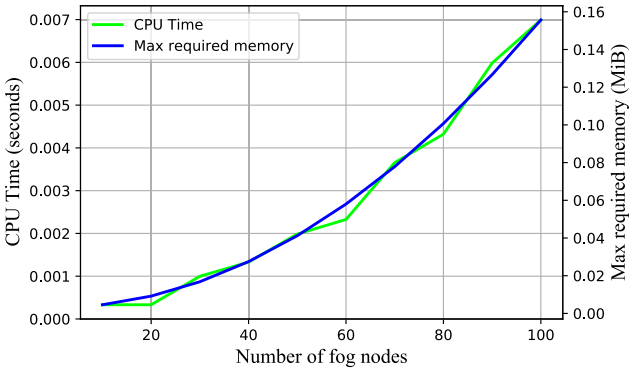

FIGURE 14. CPU time vs maximum required memory of greedy algorithm with respect to the number of fog nodes.

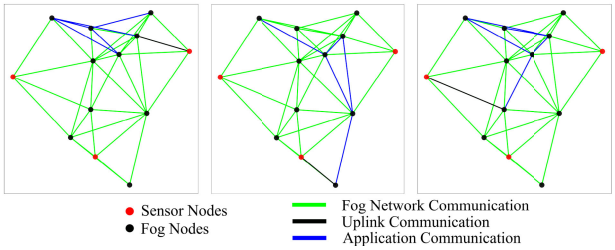

FIGURE 15. Task mapping examples (solved by our Greedy Algorithm).

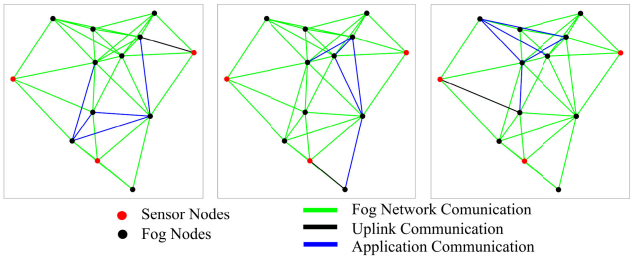

FIGURE 16. Task mapping examples (solved by Gurobi).

method clearly outperforms the Gurobi solver in terms of computing resource needs, requiring on average 99.9% less CPU time and 86.14 % less memory for solving a task mapping problem. Figure 14 shows that for 10-100 fog nodes, the CPU time and memory need do not exceed 7 ms and 0.16 MB, respectively. Moreover, for each additional group of 10 fog nodes, our greedy algorithm requires, on average, only 0.7 ms of more CPU time and 0.016 MB of more memory, whereas Gurobi requires 842 ms and 0.077 MB, respectively. Hence, compared with Gurobi, our method is highly scalable also with respect to the number of fog nodes in the network, facilitating dynamic run-time task mapping in networks of different sizes.

VI. CONCLUSION

We introduced a model for handling IoT requests with multi-tasking applications in a fog computing network and an analytical model to formulate the resource management problem from a communication cost perspective. We also proposed an algorithm based on a greedy principle to minimize the cost. Our proposed algorithm demonstrated a near-optimal efficiency, i.e., 93%, with respect to the communication cost (solution quality), while outperforming the considered optimal method in terms of the computing speed (solution latency), cutting the execution time to less than 1% of the execution time of the commercial Gurobi optimizer providing the absolute optimal solution. We showed that our proposed model is highly scalable. However, since the communication cost employed in this paper is an abstract parameter, not fully covering all communication-related cost factors in real networks, there is a need to further investigate concrete aspects such as communication delays and energy consumption when considering the resource management problem in fog computing systems. As part of the future work, we plan to extend our proposed model by considering the communication bandwidth between network nodes to calculate realistic communication delays. Moreover, we plan to take into account the deployment costs of task module containers in resource management problems.

REFERENCES

[1] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[2] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing," *IEEE Access*, vol. 7, pp. 115760–115773, 2019.

[3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[4] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence* (Studies in Computational Intelligence), vol. 546. Cham, Switzerland: Springer, 2014, pp. 169–186.

[5] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.

[6] 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.

[7] F. Hosseinpour, Y. Meng, T. Westerlund, J. Plosila, R. Liu, and H. Tenhunen, "A review on Fog Computing technology," *Int. J. Advancements Comput. Technol.*, vol. 8, pp. 1525–1530, Oct. 2016.

[8] F. Hosseinpour, A. S. Siddiqui, J. Plosila, and H. Tenhunen, "A security framework for fog networks based on role-based access control and trust models," in *Research and Practical Issues of Enterprise Information Systems*, A. M. Tjoa, L.-R. Zheng, Z. Zou, M. Raffai, L. D. Xu, and N. M. Novak, Eds. Cham, Switzerland: Springer, 2018, pp. 168–180.

[9] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in Internet of Things," *IEEE Access*, vol. 5, pp. 16441–16458, 2017.

[10] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[11] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2015, pp. 324–329.

[12] J. Su, F. Lin, X. Zhou, and X. Lü, "Steiner tree based optimal resource caching scheme in fog computing," *China Commun.*, vol. 12, no. 8, pp. 161–168, Aug. 2015.
[13] J. Liu, E. Ahmed, M. Shiraz, A. Gani, R. Buyya, and A. Qureshi, "Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 99–117, Feb. 2015.
[14] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Mar. 2013.
[15] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading inference for delivering applications in pervasive computing environments," in *Proc. 1st IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2003, pp. 107–114.
[16] E. Ahmed, A. Gani, M. Sookhak, S. H. A. Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 52–68, Jun. 2015.
[17] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile cloud Comput. Services (MCS)*, New York, NY, USA, 2012, pp. 29–36.
[18] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system," *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 2398–2410, Oct. 2016.
[19] H.-J. Hong, P.-H. Tsai, and C.-H. Hsu, "Dynamic module deployment in a fog computing platform," in *Proc. 18th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Oct. 2016, pp. 1–6.
[20] R. Eyckerman, S. Mercelis, J. Marquez-Barja, and P. Hellinckx, "Requirements for distributed task placement in the fog," *Internet Things*, vol. 12, Dec. 2020, Art. no. 100237.
[21] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.
[22] R. Patro, S. S. Patra, L. Barik, A. D. Prusty, and R. K. Barik, "Module placement scheme using MPC4.5 with Markov chain process for mobile fog computing environment," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Feb. 2021, pp. 304–309.
[23] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.
[24] L. Gu, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Dec. 2017.

FARHOUD HOSSEINPOUR (Member, IEEE) received the M.S. degree in information security from the University of Technology Malaysia (UTM). He is currently pursuing the Ph.D. degree in embedded computing architectures with the Department of Computing, University of Turku, Finland. He has served as a Lecturer and a Producer of several MOOCs for the European Institute of Innovation and Technology (EIT Digital), University of Turku. His main research interests include fog computing, the Internet of Things, intrusion detection systems, and modeling and optimization in embedded computing systems.

AHMAD NAEBI was born in Saray, Tabriz, Iran, in 1981. He received the B.Sc. degree from Tractor Applied Science and Technology University, Tabriz, in 2006, and the M.Sc. degree from Qazvin Islamic Azad University, Qazvin, Iran, in 2011. He is currently pursuing the Ph.D. degree with Xi'an Jiaotong University, Xi'an, China, in 2021. His research interests include programming languages, optimization, modeling, simulation, analyzing, designing, implementing, bond graph methodology, machine vision, signal processing, cognitive science, neuroscience, deep learning, and artificial intelligence.

SEPPO VIRTANEN (Senior Member, IEEE) received the D.Sc.(Tech.) degree in communication systems from the University of Turku, Finland, in 2004. He is currently an Associate Professor of cyber security engineering with the Department of Computing, University of Turku, where he is also the Vice Head of the Department. His current research interests include cyber security issues in communication and network technology, especially in the smart environment context.

TAPIO PAHIKKALA received the Ph.D. degree from the University of Turku, Finland, in 2008. He currently holds an Associate Professorship of machine learning with the University of Turku. He has led many research projects, supervised ten Ph.D. theses, held several positions of trust in academia, and served in the program committees of numerous international conferences. He has authored more than 150 peer-reviewed scientific articles and participated in the winning teams of several international scientific competitions/challenges. His current research interests include theory and algorithmics of machine learning, data analysis, and computational intelligence, as well as their applications on various different fields.

HANNU TENHUNEN received the M.Sc.EE. degree from the Helsinki University of Technology, in 1982, the Ph.D. degree from Cornell University, in 1986, and the Honorary Ph.D. degree. Since 1986, he was with the Tampere University of Technology as an Associate Professor and the coordinator of the National Microelectronics Program, from 1987 to 1991. Since January 1992, he has also been the Chair Professor of electronic system design with the KTH Royal Institute of Technology, Stockholm. He was the dean of the New School of ICT, KTH Royal Institute of Technology, from 2002 to 2005. For the period 2006–2011, he was the Director of the Turku Centre of Computer Science (TUCS), Finland. He has been a part-time Invited Professor at the University of Turku, Finland, since 2006. He has published over 940 reviewed publications and holds nine international patents. His current research interests include embedded electronics for autonomic and smart systems, and the IoT. He was the Education Director of the European Institute of Innovation and Technology, EIT Digital, for the period 2006–2011, at European level. He received the title of an Honorary Professor and the Mangolian Silver Award and Metal from the city of Shanghai for his contributions there.

JUHA PLOSILA (Member, IEEE) received the Ph.D. degree in electronics and communication technology from the University of Turku (UTU), Finland, in 1999. He is currently a Professor in autonomous systems and robotics with the Department of Computing, UTU. He is also the Head of the Autonomous Systems Laboratory (ASL) Research Group and the Smart Systems Unit, UTU. He leads the EIT Digital Master Programme in Embedded Systems at the EIT Digital Master School (European Institute of Innovation and Technology) and is a member of the Node Strategy Committee of the EIT Digital Finland Node. His research interests include adaptive multi-processing systems and platforms, intelligent multi-agent monitoring and control architectures, machine learning and optimization approaches, and application of heterogeneous energy efficient architectures to new computational challenges in the cyber-physical systems and the Internet-of-Things domains, with a recent focus on fog/edge computing (edge intelligence) and autonomous multi-robot systems.