Optimization of IoT Service Deployment In Multi-Layered Cloud-Fog Environment

Do Dang, XuanTung Hoang, Mai Tran
dept. Communication and Computer Network
VNU University of Engineering and Technology, Vietnam
{dodv, tunghx, mai.tran}@vnu.edu.vn

Kim-Khoa Nguyen
dept. Electrical Engineering
University of Quebec, Canada
Kim-Khoa.Nguyen@etsmtl.ca

***Abstract*—Recently, fog computing, which can be done in proximity to data sources, has emerged as a solution to provide low-latency Quality-of-Service (QoS) for IoT services in complement to centralized cloud with unlimited computing resources. Optimized service deployment on both cloud and fog environments is challenging due to their heterogeneity. Prior works mainly focus on mapping service functions and dependencies directly to physical network. In this paper, we propose a multi-layer mapping mechanism that efficiently deploys multiple IoT services to the appropriate virtual networks in physical infrastructure. We design greedy-based algorithms for solving this NP-hard problem with two phases executed sequentially. Experimental results show our proposed solution can reduce upto 80% of the total service cost compared to the state-of-the-art solutions.**

***Index Terms*—IoT service deployment, fog computing, energy efficiency, optimization, virtual machine consolidation.**

I. Introduction

New IoT services with low latency, location-awareness and mobility support requirements are raising issues facing the current centralized cloud paradigm which tend to concentrate computing and storage resources in a few large data centers. Fog computing has recently emerged as a new computing paradigm which takes advantage of the extensive resources in the cloud while being able to expand computing power to the edge of the network, close to end-users [1].

Fig.1 illustrates the architecture of a Cloud-Fog system with three hierarchical layers where Virtual Network Functions (VNFs) are deployed to implement service functions. At the edgemost of the network is the device layer which contains several sensory nodes. They can be widely distributed at various public infrastructures to monitor their condition changes over time. Data generated by IoT devices can be sent to and augmented by the VNFs deployed at the fog nodes near the data sources before being sent to cloud for further processing. Each fog node is connected to and responsible for a group of IoT devices, performing data analysis in a timely manner. Thanks to the virtualization technology, the fog nodes with heterogeneous resources can provide the ability to implement IoT service functions, providing the ability to reduce network load as well as ensure service QoS constraints including latency and location-awareness. On top of the architecture is cloud layer consists of a number of powerful servers allocated in a few data centers. With its unlimited resource capacity, the cloud will analyze data, process computational-intensive tasks, store a massive amount of data and also provides the interfaces to end-users who request for a service.

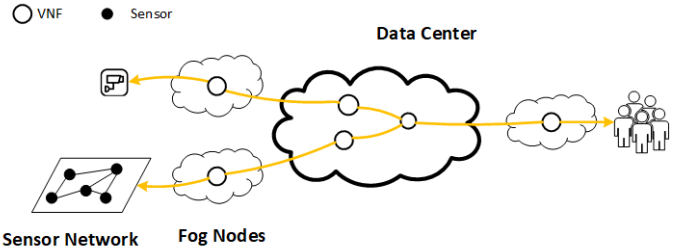

Fig. 1. IoT service in multi-layered Cloud-Fog environment

While the basic ideas and theoretical foundations of fog computing have been established, optimal deployment of IoT services onto Cloud-Fog environment is still facing many challenges. While IoT service functions prefer to be hosted at the nearby fog nodes instead of cloud to obtain the low latency and location tracking, a fog node can only host a limited number of the service functions due to its limited resource capabilities. On the other hand, the cloud is far from IoT device networks. Therefore, deploying IoT service functions onto cloud increases network load and service latency.

The problem of finding optimal deployment for virtual network requests onto a substrate network is known as virtual network embedding (VNE). [2] proposes algorithms to effectively perform generic VNE with coordinated node and link mapping. However, it doesn't take into account the nature of IoT services and multi-layered computing system. [3] presents an optimization model for VNF placement and chaining in consideration of the input traffic of VNF and specifications of connection between clouds and IoT gateways. The works in [4], [5] also introduce a mathematical formulation for the IoT services deployment in the Fog computing system and propose two approximation approaches to find the near-optimal solution for the problem. The work in [6] presents a mathematical formulation of the optimal distribution of generic IoT services over the IoT-Cloud network problem. However, the authors do not consider the problem of server sprawl and under-utilization, which is considered a major cause of energy inefficiency in data centers. Besides, all the works above had mainly focused on two-layer mapping, which

maps virtual nodes and virtual links of a VN request directly to substrate network resources.

Unlike previous studies that embed a service graph to a substrate network graph [2], [3], [6], in this paper, we formulate the problem of IoT service deployment as a multi-layer mapping. Our goal aims at minimizing energy consumption and maximizing resource utilization in both fog and cloud layers. Our contributions are as follows:

- We formulate the problem of optimizing the IoT service deployment in the Cloud-Fog environment as a mixed integer linear program (MILP), and devise a multi-layer mapping mechanism that efficiently deploys IoT services to the optimal virtual network in physical infrastructure.
- We propose a greedy-based solution for the aforementioned problem which solves each phase of the deployment process sequentially. We evaluate the solution in an illustrative Cloud-Fog environment with the joint orchestration of device, access and cloud layers. Our results show that the proposed solution outperforms by reducing upto 80% of the total service cost compared to the state-of-the-art solutions.

The rest of this paper is organized as follows: Section II describes the system model. Section III introduces the IoT service deployment problem in the Cloud-Fog environment. Experiment results are presented in Section IV. Finally, Section V draws conclusions and presents the future works.

II. SYSTEM MODEL

A. Physical layer model

We model a Cloud-Fog system as a directed graph $\mathcal{G} = (\mathcal{V}^p, \mathcal{E}^p)$ with $\mathcal{V}^p$ and $\mathcal{E}^p$ is the set of physical nodes and links in the system, respectively. Each node $u \in \mathcal{V}^p$ is characterized by its processing capacity c_u (in MIPS), processing efficiency α_u (in MIPS/W) and its location $loc(u)$ on a globally understood coordinate system. Nodes are interconnected via links, each characterized by their transmission capacity, unit energy cost and transmission delay. We use c_{uv}, α_{uv} and h_{uv} to denote the capacity (in bps), the unit energy cost (in Watts per bps) and the transmission delay (in *ms*) of the link $(u, v) \in \mathcal{E}^p$.

B. Service model

A service is presented as a directed graph $\mathcal{S} = (\mathcal{F}, \mathcal{A})$ where $\mathcal{F}$ is a set of VNFs and $\mathcal{A}$ is a set of dependencies between the VNFs. A VNF $f \in \mathcal{F}$ can capture environment information or process the information in order to create final meaningful information for the service's end users. For example, the traffic monitoring service shown in Fig. 3 contains the following VNFs: Video Capturing, Video Compressor, Vehicle Recognition, Road Condition Detection, Congestion Detection, and Visualization. Each VNF $f \in \mathcal{F}$ is associated with a processing complexity λ_f (in millions instructions per bit) and a non-negative value D_f expressing how far the VNF f can be deployed from its preferred location $loc(f)$. The dependency between two VNFs f_i and f_j is presented through virtual link $(f_i, f_j) \in \mathcal{A}$. The link carries the output data of VNF f_i to be processed by f_j. Therefore, the set of VNFs which generate data required by VNF f is presented as children of f in the service graph $\mathcal{S}$.

C. Virtual layer model

The virtual layer built on top of the substrate network layer consists of virtual machines (VMs) and virtual links $\mathcal{G}^v = (\mathcal{V}^v, \mathcal{E}^v)$. A VM can host multiple VNFs depend on its capacity. However, one VNF can have many replicas deployed to different VMs. For example, with the traffic monitoring service, if there are no more than 10 surveillance cameras and each camera generates a video of 4 Mbps, we only need to deploy the Video compressor module to one VM which has a capacity of 50,000 MIPS. If the number of camera increases, the Video compressor module will need more computing power and it will exceed the capacity of a VM, so we have to create replicas of the module and deploy those replicas into the other's VMs. We assume that there is a limited number of VM types. We denote $\mathcal{T}$ as the set of available VM types e.g. flavors. Each flavor $t \in \mathcal{T}$ has a predefined processing capacity $c(t)$. A VM type t has a processing capacity $c(t)$.

III. PROBLEM FORMULATION

A. MILP formulation.

We define the optimal services deployment in Cloud-Fog environment as a problem of finding emplacement of VNFs and routing of network flows that minimizes the overall energy consumption. The problem consists three phases:

- *Function assignment:* consists selecting an appropriate VM on a specific physical node for each VNF. Each VNF will be deployed onto a VM which created based on the VM flavor $t \in \mathcal{T}$. It is formalized as a mapping $M_F : F \to \mathcal{V}^v$ from VNFs to VMs such that: $M_F(f) \in \mathcal{V}^v, \quad \forall f \in \mathcal{F}$.
- *Node assignment:* Each VM is deployed to a substrate node by a mapping $M_N : \mathcal{V}^v \to \mathcal{V}^p$ from VMs to physical nodes such that: $M_N(w^v) \in \mathcal{V}^p, \quad \forall w^v \in \mathcal{V}^v$.
  In order to maximize the resource utilization, we have considered virtual machine consolidation when deploying VMs to a physical node.
  In Fig. 2, the service has the function mapping $\{f_1, f_2 \to VM_1$ on PS_1, $f_3 \to VM_3$ on PS_2, $f_4 \to VM_4$ on $PS_4\}$.
- *Link assignment:* Each virtual link is mapped to a substrate path (unsplittable flow) or a set of substrate paths (splittable flow) between the corresponding substrate nodes that host the end VNFs of that virtual link. It is defined by a mapping $M_E : \mathcal{E}^v \to \mathcal{P}^p$ from virtual links to substrate paths such that: $M_E(u, v) \subseteq \mathcal{P}^p(M_N(u), M_N(v)), \quad \forall (u, v) \in \mathcal{E}^v$
  The capacity and transmission delay of substrate links are taken into account in this phase. The summation of transmission delay on substrate path must less than or equal to the latency requirement of the virtual link. If the source and destination functions of a virtual link are mapped to the same physical node, the link will be ignored. For example, service in Fig. 2 has been assigned link mapping $\{(f_2, f_1) \to \emptyset, (f_3, f_2) \to (PS_2, PS_1), (f_4, f_2) \to (PS_4, PS_1)\}$

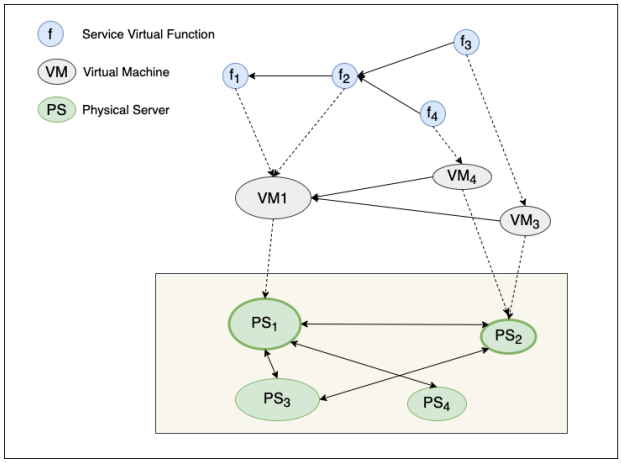

Fig. 2. Three-layer service deployment

In this paper, we assume that the number of replicas is predefined for each VNFs and we add these replicas to the service graph as individual VNFs.

To quantify the resource usage of substrate node, we use a cost function s_{u^p} is defined as the total amount of CPU capacity allocated to maintaining the operation node $u^p \in \mathcal{V}^p$ satisfying the resources requirements of VMs deployed on it.

$$s_{u^p} = \sigma_{u^p} + \sum_{u^v \to u^p} c_{u^v}$$

where σ_u denotes the amount of CPU capacity occupied by node u when it is doing nothing but power on and $x \to y$ denotes that the VM x is hosted on the substrate node y.

Similarly, the cost of a link $s(e)$ is defined as total amount of bandwidth reserved for the virtual links whose substrate paths pass through the link $e \in \mathcal{E}^p$

$$s_{e^p} = \sum_{e^v \to e^p} c_{e^v}$$

where $x \to y$ denotes that the substrate path of the virtual link x passes through the substrate link y.

The *residual* or the available capacity of a node r_{u^p} is defined as the available CPU capacity of the node $u^p \in \mathcal{V}^p$. Similarly, the *residual* of a link $e^p \in \mathcal{E}^p$ is r_{e^p} defined as the available bandwidth capacity of the link.

$$r_{u^p} = c_{u^p} - s_{u^p} \qquad r_{e^p} = c_{e^p} - s_{e^p}$$

According to [2], in order to coordinate the function mapping and link mapping phases, we extend the substrate network using the location requirements of VNFs to create augmented substrate network. Each VNF $f \in \mathcal{F}$ has a set of substrate nodes $\Omega(f)$ which contains all of the substrate nodes that the VNF can be embedded in.

$$\Omega(f) = \{u^p \in \mathcal{V}^p | dis(loc(f), loc(u^p)) \leq D(f)\} \quad (1)$$

For each VNF $f \in \mathcal{F}$, we create a *meta node* $\mu(f)$ and connect $\mu(f)$ to all the substrate nodes in $\Omega(f)$ using *meta edges* with infinite bandwidth and zero cost. We combine all the meta nodes and meta edges with the substrate network $\mathcal{G}^p$ to create the augmented substrate network $\mathcal{G}^s = (\mathcal{V}^s, \mathcal{E}^s)$, where

$$\mathcal{V}^s = \mathcal{V}^p \cup \{\mu(f) | f \in \mathcal{F}\} \quad (2)$$

$$\mathcal{E}^s = \mathcal{E}^p \cup \{(\mu(f), u^p) | f \in \mathcal{F}, u^p \in \Omega(f)\} \quad (3)$$

TABLE I
NOTATIONS

Name	Description
$\mathcal{V}^p$	Set of physical nodes
$\mathcal{E}^p$	Set of physical links
$\mathcal{F}$	Set of VNFs
$\mathcal{A}$	Set of virtual links between VNFs
$\mathcal{V}^s$	Set of nodes in augmented graph
$\mathcal{E}^s$	Set of links in augmented graph
$\mathcal{T}$	Set of virtual machine's flavors
c_t	Processing capacity of virtual machine type t
p_u	Energy consumed by node $u \in \mathcal{V}^p$
p_{uv}	Energy consumed by link $(u,v) \in \mathcal{V}^p$
σ_u	Amount of processing capacity occupied by node u when it's doing nothing but power on
α_u	Processing efficiency of node u
c_u	Processing capacity of node u
λ_f	Processing complexity of VNF f
Ω_f	Set of substrate nodes that VNF f can be deployed to
c_{uv}	Transmission capacity of link $(u,v) \in \mathcal{V}^p$
h_{uv}	Transmission delay of link (u,v)
α_{uv}	Transmission cost of link (u,v)
e_i	The ith virtual link $(s_i, t_i) \in \mathcal{A}$ with source and destination VNFs s_i and t_i, respectively.
x_{uv}^i	A variable denotes the fraction of ith virtual edge in $\mathcal{A}$ embedded in link (u,v)
b_u^{tjw}	Binary variable denotes VNF w is deployed to the jth virtual machine of flavor t on physical node u
b_u^{tj}	Binary variable denotes the jth virtual machine of flavor t is run on node u
d_u	Binary variable denotes physical node u is in used

The services deployment problem now can be formulated as a mixed integer multi-commodity flow (MCF) problem. We consider each virtual link $e_i \in \mathcal{A}$ as a commodity with source and destination nodes s_i and t_i ($s_i, t_i \in \mathcal{V}^s \backslash \mathcal{V}^p$), respectively. In this setting, each flow starts from a meta node and ends in another meta node. By introducing restrictions on the meta edges, each meta node $\mu(f)$ can be forced to choose only one meta edge to connect itself to the actual substrate node in $\Omega(f)$. This effectively selects a substrate node for each meta node, i.e. maps the virtual node corresponding to that meta node to a substrate node. At the same time, all the virtual links (i.e. flows) are also mapped efficiently inside the substrate network. We present the MILP formulation in the following.

Our objective function tries to minimize the overall running cost of Cloud-Fog system by taking into account the VM consolidation:

$$\text{minimize} \sum_{u \in \mathcal{V}^p} p_u + \sum_{(u,v) \in \mathcal{E}^p} p_{uv} \quad (4)$$

Constraints:

- *Power consumption constraints*: Every physical node consumes a certain amount of energy even it's doing nothing but power on. The total energy consumed by

a physical node is defined in (5) which contained of the consumed energy when it is doing nothing but power on and the consumed energy when processing computing tasks.

$$p_u = (d_u\sigma_u + \sum_{t,j} b_u^{tj} c_t)\alpha_u \quad \forall u \in \mathcal{V}^p \tag{5}$$

The equation (6) defines the energy consumed by a link to transmit data between nodes.

$$p_{uv} = \sum_{e_i \in \mathcal{A}} x_{uv}^i c_{e_i} \alpha_{uv} \quad \forall u, v \in \mathcal{V}^p \tag{6}$$

- *Flow-related constraints*: Constraint sets (7)-(9) refers to the flow conservation conditions, which denote that the net flow to a node must be zero except for the source node s_i and sink node t_i.

$$\sum_{v \in \mathcal{V}^s} x_{uv}^i - \sum_{v \in \mathcal{V}^s} x_{vu}^i = 0 \quad \forall e_i \in \mathcal{A}, \forall u \in \mathcal{V}^s \backslash \{s_i, t_i\} \tag{7}$$

$$\sum_{w \in \mathcal{V}^s} x_{s_i w}^i - \sum_{w \in \mathcal{V}^s} x_{w s_i}^i = 1 \quad \forall e_i \in \mathcal{A} \tag{8}$$

$$\sum_{w \in \mathcal{V}^s} x_{t_i w}^i - \sum_{w \in \mathcal{V}^s} x_{w t_i}^i = -1 \quad \forall e_i \in \mathcal{A} \tag{9}$$

- *Capacity constraints*: The summation of flows on the link (u, v) has to remain within its available bandwidth.

$$\sum_{e_i \in \mathcal{A}} x_{uv}^i c_{e_i} \leq r_{uv} \quad \forall u, v \in \mathcal{V}^s \tag{10}$$

Constraint sets (11) and (12) enforce the capacity bounds of nodes and VMs on node. The constraint (12) shows that virtual functions deployed in the VM are provided with its required computing power. The constraint (11) ensures that total computing resource required by all VMs on a physical node remains within its computing capacity.

$$\sum_{t,j} b_u^{tj}.c_t \leq r_u \quad \forall u \in \mathcal{V}^p \tag{11}$$

$$\sum_{w \in \mathcal{F}} b_u^{tjw} \lambda_w \leq c_t \quad \forall u, t, j \tag{12}$$

- *QoS constraints: Latency*: Constraint (13) refers to the latency requirement of each virtual link. The total delay of substrate links that the virtual link passes through has to less than or equal to the virtual link latency constraint.

$$\sum_{u,v \in \mathcal{V}^s} h_{uv}.x_{uv}^i \leq h_{e_i} \quad \forall e_i \in \mathcal{A} \tag{13}$$

- *Deployment restrictions*:
The set guarantees that only one substrate node is selected for each VNF and all the VNFs will be deployed to appropriate substrate node.

$$\sum_{u,t,j} b_u^{tjw} = 1 \quad \forall w \in \mathcal{F} \tag{14}$$

$$\sum_{t,j} b_u^{tjw} \geq x_{wu}^i \quad \forall e_i \in \mathcal{A}, \forall u \in \mathcal{V}^p, \forall w \in \mathcal{F} \tag{15}$$

$$\sum_{t,j} b_u^{tjw} \geq x_{uw}^i \quad \forall e_i \in \mathcal{A}, \forall u \in \mathcal{V}^p, \forall w \in \mathcal{F} \tag{16}$$

$$b_u^{tj} \geq b_u^{tjw} \quad \forall i, t, j, \forall u \in \mathcal{V}^p, \forall w \in \mathcal{F} \tag{17}$$

The node u is known as power on if there is any VM is running in it.

$$d_u \geq b_u^{tj} \quad \forall t, j, \forall u \in \mathcal{V}^p \tag{18}$$

- *Domain constraints*:

$$x_{uv}^i \in [0, 1] \quad \forall i, \forall u, v \in \mathcal{V}^s \tag{19}$$

$$b_u^{tjw}, b_u^{tj}, d_u \in \{0, 1\} \quad \forall t, j, \forall u \in \mathcal{V}^p, w \in \mathcal{F} \tag{20}$$

- We remark that the solution for IoT service deployment in Cloud-Fog system is centralized. It requires collecting information about services requirements and network resources at a centralized network controller and disseminating the solution to all network nodes.

B. Algorithmic solutions.

Since solving the problem of simultaneous function and link mapping using MILP is practically infeasible, we propose two greedy-based strategies for solving each phase of the deployment process sequentially. The first one is presented in *Algorithm 1* which tries to deploy each of service functions into appropriate VMs first and then embed the network of the VMs onto physical network. The second strategy tries to map service functions to the satisfied physical nodes first and then applies the consolidation mechanism to pack the service functions in each node into VMs presented in *Algorithm 2*.

The *Algorithm 1* takes service deployment request as input and creates a virtual network that hosts the service. For each VNF, the algorithm create a VM based on a flavor in the list of VM flavors that has minimum processing capacity while satisfying the CPU capacity required by the VNF. Next, for each VM, the algorithm checks whether there are any possible substrate nodes within its Ω set that remaining with enough available CPU capacity. If any of the Ω sets are empty, the procedure rejects the request and stops immediately. Otherwise, for each VM w, the algorithm calculates a value ϵ_u for each substrate node $u \in \Omega_w$. ϵ_u is calculated as the fraction of the available CPU capacity r_u and the product of total capacity c_u and the processing cost α_u of the node u. Then the substrate nodes will be sorted by ϵ_u. The algorithm maps the VM w to the node with minimum ϵ_u value while satisfied with resource requirements of w to archive the minimum cost when running the VM. Once all the VMs have been mapped to suitable substrate nodes, the algorithm solves the MCF problem to map the virtual links in $\mathcal{E}^v$ onto substrate paths. Finally, it updates the residual capacities on system resources.

The *Algorithm 2* tries to map the VNFs to the substrate nodes first and then to pack the VNFs in each node into VMs. It takes the service deployment request as input. For each VNF, it checks whether there are any substrate nodes within its Ω set that remaining with enough available CPU capacity. If any of the Ω sets are empty, the algorithm rejects the

Algorithm 1 Service deployment with function-to-vm mapping first

procedure SERVICEDEPLOYMENT($\mathcal{F}$, $\mathcal{A}$)
  $\mathcal{V}^v \leftarrow \emptyset$; $\mathcal{E}^v \leftarrow \emptyset$
  for all $f \in \mathcal{F}$ **do**
    Let $t_{min} = \arg\min_{t \in \mathcal{T}}\{c_t | c_t \geq \lambda_f\}$
    Create VM w belong to type t_{min}
    $\varphi_f \leftarrow w$; $\mathcal{V}^v$ += w
  for all $e_i \in \mathcal{A}$ **do**
    Create Virtual link $(\varphi_{s_i}, \varphi_{t_i})$
    $\mathcal{E}^v$ += $(\varphi_{s_i}, \varphi_{t_i})$
  for all $w \in \mathcal{V}^v$ **do**
    if $\Omega_w \backslash \{u \in \Omega_w | r_u < c_w\} == \emptyset$ **then**
      Reject the request
      return
    for all $u \in \Omega_w$ **do**
      $\epsilon_u \leftarrow \frac{r_u}{c_u \alpha_u}$
    Let $u_{min} = \arg\min_{u \in \Omega_w}\{\epsilon_u | r_u \geq c_w\}$
    $M_N(w) \leftarrow u_{min}$
    Update residual capacities of node u_{min}
  Solve MCF to map virtual links in $\mathcal{E}^v$
  if MCF succeeded **then**
    Update residual capacities of system resources
  else
    Reject the request

request and stops immediately. Otherwise, for each substrate node u within the set Ω_f, it calculates value ϵ_u as defined in *procedure 1*. Next, the VNF f will be mapped to the node with minimum value ϵ while satisfied with resource requirements of f. Once all the VNFs have been mapped to suitable nodes, the algorithm iterates over the substrate nodes and tries to pack the mapped VNFs in the node to appropriate VMs. The algorithm uses Iterative best-fit decreasing mapping algorithm proposed in [7] to deploy the unmapped VNFs to appropriate VMs. After that, the algorithm solves the MCF problem to map the virtual links in $\mathcal{E}^v$ onto substrate paths. Finally, it updates the residual capacities on system resources.

IV. EXPERIMENT RESULTS

We compare the efficiency of our solution with: i) the IoT-Cloud solution from [6] which optimizes the placement of IoT service functions based on the flexibility of the IoT-Cloud infrastructure, and ii) the VNE model which minimizes the cost of embedding VN request, and balances the load across the substrate network resources [2].

A. Experimental configuration

We consider a Cloud-Fog system architecture composed of three main layers: i) a cloud layer with nodes located in large data centers (DC), ii) an access layer, composed of base stations (BSes) hosting fog nodes, and iii) a device layer that contains sensors and smart devices. The configurations of these nodes and links are presented in TABLE II.

Algorithm 2 Service deployment with function-to-node mapping first

procedure VIRTUALNETWORKDEPLOYMENT($\mathcal{F}$, $\mathcal{A}$)
  for all $f \in \mathcal{F}$ **do**
    if $\Omega_f \backslash \{u \in \Omega | r_u < \lambda_f\} == \emptyset$ **then**
      Reject the request
      return
    for all $u \in \Omega_f$ **do**
      $\epsilon_u \leftarrow \frac{r_u}{c_u \alpha_u}$
    Let $u_{min} = \arg\min_{u \in \Omega_w}\{\epsilon_u | r_u \geq \lambda_f\}$
    $M_F(f)$ += u_{min}
  for all $u \in \mathcal{V}^p$ **do**
    Iterative best-fit decreasing mapping.
    Update residual capacities of node u.
  Solve MCF to map virtual links
  if MCF succeeded **then**
    Update residual capacities of system resources
  else
    Reject the request

TABLE II
CLOUD-FOG SYSTEM RESOURCES

	Capacity	Efficiency
Cloud node	53.5 Million MIPS	500 MIPS/W
Fog node	6.5 Million MIPS	100-500 MIPS/W
Smart Device	5000 MIPS	MIPS/W
Sensor	1000 MIPS	MIPS/W
Optical link	4480 Gbps	12.6 nJ/bit
Wifi link	150 Mbps	300 nJ/bit
4G link (Down/Up)	72/12 Mbps	76.2/19 μJ/bit

The traffic monitoring service in which users request information about the real-time traffic status in the city illustrated in Fig. 3. In particular, we assume that there is a video of 4 Mbps generated in each of surveillance cameras. The Video Compressor module requires a computing capacity of 4000 instructions per bit to compress the input video with a compression ratio of 3:1. Both of Vehicle Recognition and Road Condition Detection modules require a capacity of 8000 instructions per bit to process the data from the Video Compressor module. The Congestion Detection module is the highest demanding component which requires 15000 instructions per bit to generate the real-time traffic status, which requested by users. The Visualization module which data from the Congestion Detection module and creates a visual map for users requires 1000 instructions per bit. In terms of network access technologies, we assume the half of the users use WiFi and the other half use 4G.

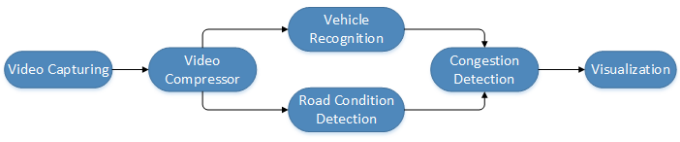

Fig. 3. Service model example: Traffic monitoring service.

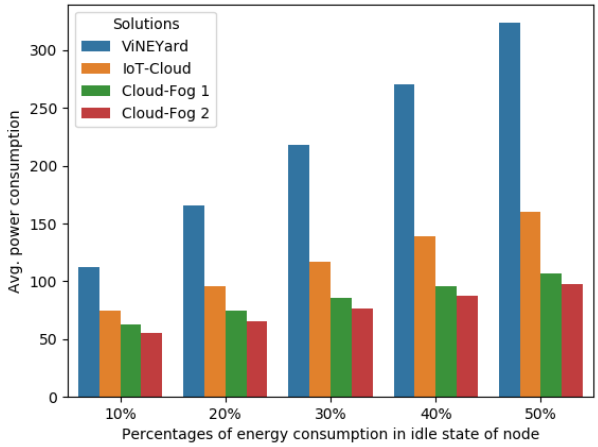

Fig. 4. Average power consumption of the traffic monitoring service for different amount of energy consumed by server nodes in idle state.

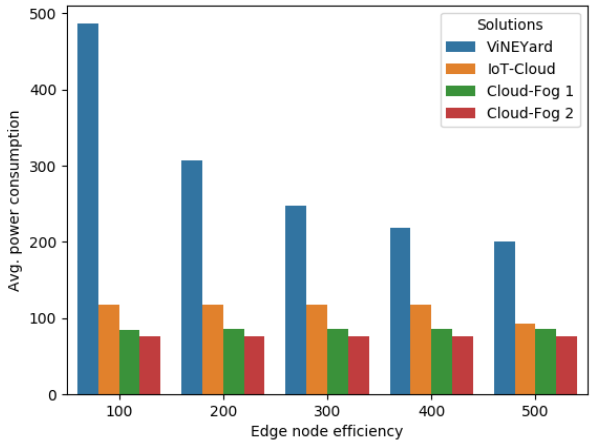

Fig. 5. Average power consumption of the traffic monitoring service for different edge node efficiencies.

B. Simulation scenarios

1) The change of energy consumed by edge nodes in idle state: Fig. 4 shows the average energy consumption of Cloud-Fog system regarding various idle states of the server nodes. In this scenario, the fog nodes efficiency is constant (400 MIPS/W). We change the amount of energy consumed by server nodes according to different idle states from 10% to 50% of the node capacity. For the cost sufficiently high, our solution allocates the functions of the service at the edge nodes instead of cloud nodes to reduce the overall energy consumption. Otherwise, our solution tends to consolidate the functions to cloud node to reduce the number of nodes being used. There is a slightly different between the results of Cloud-Fog 1 and Cloud-Fog 2 due to the consolidation mechanism. The Cloud-Fog 1 finds the appropriate VM for each VNF first, then deploys them to the physical network. It creates an unused space in each VM and increases the energy consumption while the Cloud-Fog 2 algorithm has a consolidation mechanism that consolidates VNFs into VMs reducing the unused space in VMs. We also observe that the ViNEYard solution yields the highest overall energy consumption over the solutions. This is because the ViNEYard solution tries to balance the load over the infrastructure resulting in the most servers being used. The IoT-Cloud solution tends to deploy the service functions at the lost-cost, nearby nodes to reduce the transmission cost. However, the cost to run a server is significantly high compared to the transmission cost over the network.

2) Different edge node efficiencies: In Fig. 5, we show the average energy consumption for different edge node efficiencies. In this scenario, the simulation results show that our solution chooses to consolidate the service functions in cloud data center to take advantages of high processing efficiency of cloud server and reduce the number of running nodes. The reason for the difference between the results of Cloud-Fog 1 and Cloud-Fog 2 is the same as in the scenario above. The ViNEYard solution produces the highest overall energy consumption one more time due to its deployment strategy. The IoT-Cloud solution adapts the offloading decisions according to the processing efficiency of edge nodes. With the sufficiently high of edge nodes efficiency, the IoT-Cloud offloads some functions from cloud nodes to the edge nodes resulting the increasing in number of nodes being used.

V. CONCLUSION

In this work, we study the problem of optimizing the IoT service deployment in Cloud-Fog environment, which is known as a NP-hard problem. We formulate the problem with three layers including the physical layer, virtual layer, and service layer. We also propose two greedy-based strategies for solving each phase of the deployment process sequentially. Numerical results show our solution outperforms the prior work which considers only two-layer mapping.

In the future, we plan to improve the model for dynamically deploying services according to the change in the number of users over time.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[2] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 1, pp. 206–219, 2012.

[3] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, "Placement and chaining for run-time iot service deployment in edge-cloud," *IEEE Transactions on Network and Service Management*, 2019.

[4] C. Pham, K. K. Nguyen, and M. Cheriet, "An approximation mechanism for elastic iot application deployment," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 159–165, IEEE, 2018.

[5] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, "Virtual network function placement in iot network," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1166–1171, IEEE, 2019.

[6] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "Iot-cloud service optimization in next generation smart environments," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 4077–4090, 2016.

[7] J. Kang and S. Park, "Algorithms for the variable sized bin packing problem," *European Journal of Operational Research*, vol. 147, no. 2, pp. 365–372, 2003.