Toward Service Placement on Fog Computing Landscape

Quang Tran Minh, Duy Tai Nguyen,
An Van Le, Hai Duc Nguyen, Anh Truong
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology,
VNU-HCM, Vietnam
Email: quangtran@hcmut.edu.vn

Abstract—**This paper proposes an approach to optimize service placement on Fog landscape in the context of the Internet of Things (IoT). A multi-tier fog computing architecture that supports IoT service provision is devised. Based on this architecture, a novel service placement mechanism that optimizes service decentralization on Fog landscape leveraging context-aware information such as *location, time, quality of services (QoS)* has been proposed. An experiment has been conducted to evaluate the proposed approach with several simulations applying to smart grid applications. The results reveal the effectiveness of the proposed approach in terms of reducing latency, energy consumption, and network load in comparison with the conventional cloud computing model.**

Keywords *IoT, Fog/Edge computing, service placement, multi-tier architecture, context-aware*.

I. Introduction

The Internet of Things (IoT) has emerged as a revolutionary technology to offer a fully connected "smart" world. It enables the acceleration of the 4^{th} industrial revolution where thousands or millions of things in the physical world are connected with each other to share data and services. These data and services are used to specify, monitor, and manage the physical world, from which applications for smart city, smart healthcare, smart agriculture, etc., can be built to transform the way we work, play and live, improving the quality of life and the human civilization.

However, realizations of IoT services in a large scale are hindered due to the constraint of IoT devices (embedded on everyday objects such as consumer goods, enduring products, vehicles, utility components, sensors, and other physical devices) in terms of computing resources, memory capacity, energy, and communication bandwidth. Many of these issues could be resolved by employing the Cloud-assisted Internet of Things or Cloud-of-Things (CoT) technology as it offers large-scaled and on-demand networked computing resources to manage, store, process and share huge volume of IoT data.

Nevertheless, the CoT paradigm is facing increasing difficulties to handle the Big data generated by IoT services associated with beyond billions of connected devices. As these devices are frequently (e.g., in every second or shorter intervals) generating data, a large amount of data is generated every moment (exabytes of data per day). If all of such data is transferred to data centers (DCs) on the cloud for processing and storage, and then another large amount of information is returned to users from DCs, a huge volume of traffic is pumped into the network. Obviously, this process challenges system performance and robustness in terms of ensuring low latency and network bandwidth consumption, optimal utilization of computational recourses, and scalability.

In fact, most of IoT data and services are commonly generated and consumed by local users. Therefore, to cope with the aforementioned challenges, a recent trend is to devise effective Edge Computing infrastructures, termed as Edge-of-Things (EoT) computing, to allow moving computing and storage resources close to IoT devices where data is generated. This technology is known as *Edge computing* or *Fog computing* proposed by Cisco [1], [2]. Edge/Fog computing devices could be smart gateways or routers deployed at the network edge, local PCs and even mobile devices such as smart phones or tablets carried by users that can offer computing and storage capabilities. These devices play their own roles of determining what data should be stored or processed locally (for low latency and saving network bandwidth) and what needs to be sent to the Cloud for further analysis. It is clear that, EoT complements CoT paradigm in terms of high scalability, low delay, location awareness, and allowing of using local computing resources which are available at the network edge.

Although the benefit of Fog computing in the IoTs is clear and the basic ideas of this computing paradigm have been well stated in various researches [2], [3], there is still a lack of systematical modeling for Fog computing and effective solutions for optimal service placement on the Fog landscape. In this work, we propose a novel approach to **service placement on Fog landscape** with main contributions as follows:

(i) We devise a systematical model of Fog computing scheme which consists of multiple intelligent tiers for the IoT.

(ii) We propose a service placement mechanism to optimize service decentralization on Fog landscape leveraging context-aware information such as *location, time, QoS*,... hence maximize the IoT potential.

The effectiveness of the proposed approach in terms of reducing latency, energy consumption, and network load is thoroughly evaluated with simulations applying to smart grid applications. The results demonstrate the feasibility as well as the effectiveness of the proposed approach revealing its capability to maximize the IoT potential.

The rest of the paper is organized as follows: Section II reviews related work. The overall architecture and problem definition are described in Section III. The proposed service placement mechanism is presented in Section IV. Section V describes the evaluation of the proposed approach while Section VI concludes the paper.

II. Related Work

In the world, there are several projects that share many common points with our present work and are originated from wireless sensor networks (WSNs), without cloud computing. The HOURGLASS architecture [4], [5] was proposed for large scale of sensing data stream. The streams are optimally routed and processed by using the overlay technology in SBON [4] which is another approach in software-defined network (SDN) and network function virtualization (NFV).

As shown in the survey of [6], most of the existing IoT related projects assume the availability of centralized data centers based on a Cloud-centric approach. A typical example is described in [5] which addresses necessary components of a Cloud-centric IoT architecture. The authors proposed a federation between a private cloud (e.g., Aneka, their own system) and a public cloud (e.g., Microsoft Azure cloud) to efficiently handle sensing data from WSNs. However, in the networking aspects it focused on access networks while ignoring the core networks. Consequently, this approach could not satisfy the effectiveness required in IoT as global services are mainly computed on the cloud, and transmitted and managed over the core networks.

There are several works to reinforce the shortage of Cloud-centric IoT approach by employing localization of computing resource [7], [1]. The work in [7] describes a locality-based Utility Computing (LUC) Operating System. These utilities are distributed over the network backbones, such as local servers connecting to routers, aiming to provide the computing resources. The motivation of this work is similar to our present paper, that is instead of building bigger and bigger data centers and over-sized networks, we should bring resources near to the edges or users, where the IoT data is created. Unfortunately, the work in [7] had not utilized context-aware information such as location-awareness and other related information to improve its effectiveness, but keeps location-awareness for future works.

Cisco proposed the **Fog computing** concept [1], [2] as a direction to solve the essential issues on the Cloud-centric approach. Fog Computing is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional cloud computing data centers, typically, but not exclusively located at the edge of network.

Our proposed fog computing scheme is closely related to Cisco's Fog computing. However, one of the major differences from the Fog computing concept is that the computing resources in our system are more flexibly designed and allocated/distributed based on context-aware information, specifically *location, time, type of service, quality of the service (latency, accuracy)*, and so on to which users are expected from services. In addition, services can be optimally distributed on our proposed scheme in terms of making the highest local resources which are already available at the fog landscape while satisfying the response time of applications.

Moreover, there are also references on the passive micro-datacenter that is rechargeable datacenter (e.g., using solar energy) at a very small scale [8]. This datacenter concept can be extended to be applied in our proposed context-aware multi-tier architecture, to which not only the energy but also issues related to computing services such as computational power, data organization and indexing, functional computing, and so on are reasonably deployed on these data centers based on the collected context.

The notion of *context* has been observed in numerous areas including linguistics, knowledge discovery, artificial intelligent, information retrieval, reasoning, theory of communications [9], [10] and so forth. As a high level of abstraction, *context* is defined as *"that which surrounds, and gives meaning to, something else."* In this definition *"something"* can be an artifact, a building, a person, a computer system or even an assertion in logic as *"context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."* [11].

In the IoT environment, context includes but not limited to t*ime, location, type of service, quality of the service, service owner, etc*. Obviously, this information is very important for optimization in data organization, indexing, routing, allocating of computing resources in the IoT where there are a large number of devices, providing a huge amount of complicated services. However, as mentioned before, the existing technologies have not effectively utilized context-aware information for optimizing IoT resources to maximize its potential [7].

As fog computing is a distributed computing approach, service placement optimization is one of the essential issues. However, works for resource provisioning mechanisms, specifically for fog computing environments, are quite limited. A study in [13] proposes a simple resource provisioning strategy based on workload thresholds of different fog cells, while Aazam et al., present a more sophisticated approach using resource demand prediction [14]. These approaches, however, statically provide resources which are not adaptive to the dynamic changes of network topologies at the fog landscape. Our work is different as context-aware factors are utilized to decentralize IoT services/data, thereby the resource allocation can be effectively conducted, help to improve the quality of services, reduce the latency, and so forth.

III. Overall Architecture and Problem Definition

The natural features of an IoT system is its complicated connections between a huge number of devices, the provision of data and services are specific to application domains (e.g., healthcare, agriculture, traffic,...). IoT devices are distributed almost everywhere in the physical world to which data or

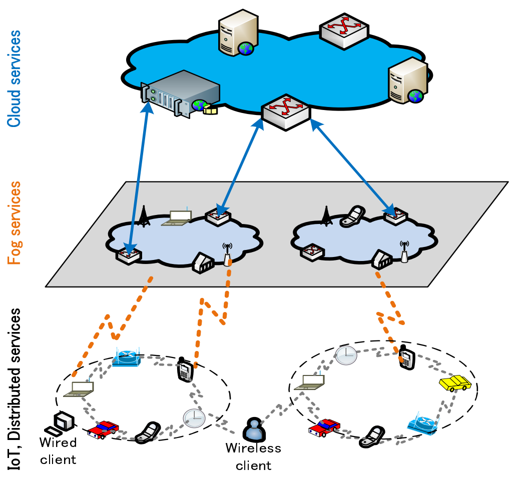

Fig. 1. A multi-tier architecture for the Internet of Things

services provided by an IoT system are mostly consumed by local users. In some other cases these data or services are consumed by global users via cloud computing systems. In general, sensor systems, distributed service computing elements are connected with each other via intermediate connection elements (e.g., local servers) and global computing elements on the cloud in order to resolve the challenges on scalability, flexibility, domain specific on IoT.

However, in which way these distributed systems can be efficiently managed in the current infrastructures, specifically the current Internet, which is not originally designed to support for the distributed computing with highly flexibility in the IoT environment as mentioned above is an essential research question which needs to be thoroughly resolved.

This paper aims at proposing a novel **multi-tier architecture for the Internet of Things**, we named it 3-tier architecture. In this architecture, IoT elements (sensors, mobile phones, laptops, vehicles, base stations, local servers, network connection and management elements, etc.) are connected in a multi-tier distributed scheme consisting of different levels of intelligence: *device/group-of-devices tier, regional tier*, and *global tier*. This architecture is depicted in Fig. 1 and is described as follows.

- *Device/group-of-devices tier* includes IoT distributed services (DSs). This tier manages distributed services generated by IoT devices (things) such as wireless sensors, vehicles, mobile devices wirelessly connected via ad-hoc or P2P modes. These services, such as pop-up advertisements shown whenever a user passes his or her favorite shop, a warning on driver's over-speed, are commonly useful for local users.
- *Regional tier* consists of IoT services that are computed at local computing servers installed at network edge forming fog landscape. In this landscape, fog nodes are gathered into fog colonies, each of which is controlled by a fog orchestration node. This tier provides services that could not be found from DSs and also serves as intermediate layer for data pre-processing or data integrations be forwarding to DCs on the cloud for further computation. This scheme helps not only to mitigate the communication and computation costs but also to reduce latency of local-based services for local users. In addition, services that best fit with the context such as introducing suitable clinics that are close to users (context about location) with appropriate speciality compared to the patients' symptoms (context about service types).
- *Global tier* provides global IoT services or cloud services which are integrated at centralized DCs or service cloud endpoints located on clouds. These global services are adequate with common context to a specific application. For example, in a healthcare system, in accordance with flu symptoms such as high fever, headache, nausea, etc., reported by different users, the preventive health function deployed on the cloud can analyze these data examining with globally historical data (e.g., *medical dictionary, descriptive data about epidemic*) to estimate whether this is a transmissive flu. If so, it will immediately provide related information and instructions to community (not only to the patients) to prevent the spreading of such a disease.

IV. Service Placement on the Fog Landscape

An inherent issue in Fog computing is how to optimize the utilization of virtualized resources on the fog landscape. This section proposes a novel method for maximizing such available resources when deploying tasks or services (hereafter, task and service are used interchangeably) in the proposed 3-tier architecture.

Problem: *Given a set of applications $A_1, A_2, .., A_m$ each of which is constrained with a deadline D_{A_k}. Suppose that each application is composed of independent tasks (i.e., tasks can be executed simultaneously), $A_k = \{\tau_1, \tau_2, .., \tau_n\}$. Let $J = \bigcup_{i=1}^{m} A_k = \{\tau_1, \tau_2, .., \tau_N\}$ be the set of tasks resulted from the decomposition of all applications that need to be deployed on the 3-tier network. This work aims to deploy the tasks mentioned above on the fog landscape and the cloud that satisfy two criteria as follows:*

C1: No application misses its deadline as described in equation (1), where R_{A_k} is the response time of A_k.

$$R_{A_k} \leq D_{A_k}, \forall A_k \tag{1}$$

C2: The number of tasks deployed on the fog landscape is maximized.

According to the system architecture described in Section III, application requests are sent to the fog orchestration node (F) of a corresponding fog colony. Then F is responsible for generating and deploying tasks over the system in accordance with two criteria (C1 and C2) mentioned above. Given a task τ_i, node F will determine to place it on one of four places:

- On itself (i.e., on F).

- On a fog cell on the colony managed by F (i.e., any $f \in Res(F)$), where $Res(F)$ denotes a set of fog cells in F's colony).

- On its neighbor colony controlled by an orchestration node N (the details of task management and execution are delegated to N).

- On the cloud denoted as R.

Let $x^f_{\tau_i}, x^F_{\tau_i}, x^N_{\tau_i}, x^R_{\tau_i} \in \{0,1\}$ be binary variables telling either the task τ_i is deployed on a fog node f ($x^f_{\tau_i} = 1$), on the fog orchestration node F ($x^F_{\tau_i} = 1$), on the neighbor colony ($x^N_{\tau_i} = 1$), or on the cloud ($x^R_{\tau_i} = 1$). Furthermore, since a task is only deployed once, the constraint in (2) is held:

$$x^f_{\tau_i} + x^F_{\tau_i} + x^N_{\tau_i} + x^R_{\tau_i} = 1, \forall \tau_i \tag{2}$$

Since our purpose is to maximize the number of task assignments on the fog landscape, with a given fog colony orchestrated by F, the objective function is formed as in (3).

$$max \sum_i^N \left(\sum_f^{Res(f)} x^f_{\tau_i} + x^F_{\tau_i} + x^N_{\tau_i} \right) \tag{3}$$

Resolving the objective function in equation (3) provides an optimal placement plan that maximizes the number of tasks deployed on the fog landscape (i.e., near to data sources and better utilize the available virtualized resources). This plan satisfies the QoS constraint presented in equation (1) to which every application is completed before a predefined deadline D_{A_k} under the available virtualized resources on the fog landscape. This means that, for a task τ_i which is planned to be deployed on a node *p* (fog cell, fog orchestration node), *p* must satisfy resources required by τ_i and all the tasks composing the application A_k must be completed before the application's deadline to satisfy the global constraint in equation (1). In this work, the resources required by a task are computation power (CPU) and storage (memory) capacity. These constraints are taken into account for a context-aware service placement that satisfies our objective function and resource and QoS constraints. Examples of evaluating computational resource constraints and estimating the global application response time are presented as follows.

Estimating response time of a task

Accomplishing a task τ_i requires four steps: *task submission, deployment, execution, and result return*. Therefore, the *response time* r_{τ_i} of such a task is calculated as follows:

$$r_{\tau_i} = w_{\tau_i} + m_{\tau_i} + d_{\tau_i} \tag{4}$$

where,

- w_{τ_i} is the *deployment time* in which data and computation resource needed by the task are prepared.

- m_{τ_i} is the *execution time* (or makespan time) in which the task actually utilizes resources on the computing node for execution.

- d_{τ_i} is the *communication time* consisting of (a) *Task submission time* which is the time it takes to move necessary information from the fog orchestration node to the node where the task will be deployed, and (b) *Result return time* which is the time it takes to return the result to the fog control node and release unused resources.

We assume that the resources on cloud is unlimited, hence when a task r_{τ_i} is submitted to the cloud, it is executed and finished immediately. Therefore, if a task is assigned to the cloud, its response time is the communication time (i.e. $r_{\tau_i} = d_{\tau_i}$).

Estimating the response time of a task running on a fog cell, in contrast, strongly depends on how the fog orchestration node distributes tasks to other nodes and the mechanisms each node uses to schedule task deployment and execution. We briefly describe those as follows.

Each fog node runs tasks in multiple *turns 1, 2, .., M*. At the beginning of a turn, the node loads *all* tasks assigned to it and deploys all of them. Once a task is deployed successfully, the node uses a part of its computation power to execute the task. Right after the task has been finished, the result will then be transferred back to its corresponding fog control node. When all of tasks have been done, the node releases all resources and marks the current turn as *"finish"*. After that, the node moves to the next turn, loads new tasks and executes them, if there is any assignment.

Note that a task must be completed within a *single* turn. It cannot be propagated across multiple turns. Also, a node only deploys tasks at the beginning of a turn. Therefore, if a task is assigned to a node, this node does not start the task immediately but waits for the current turn to finish (tasks are put on the node waiting queue). More formally, a node deploys new tasks if and only if:

- there exists at least one task in its waiting queue, and

- all tasks in the previous turn have been finished and the node is ready for releasing resources for deploying new tasks.

Since a node loads all tasks assigned to it in each turn and tasks are executed concurrently, and all fog nodes are controlled (i.e., they can be synchronized) by the fog orchestration node, the response time of an application r_{A_k} can be estimated as follows:

$$r_{A_i} = max_{\tau i \in A_k}(r_{\tau_i}) = max_{\tau i \in A_k}(w_{\tau_i} + m_{\tau_i} + d_{\tau_i}) \tag{5}$$

Given this mechanism, the fog orchestration node can estimate $w_{\tau_i}, m_{\tau_i}, d_{\tau_i}$ based on the available CPU, memory and communications resources of the destination node (where the task will be deployed) and the corresponding resource requirement from the task.

V. Evaluation

This section presents a preliminary evaluation to validate the feasibility and effectiveness of the proposed approach when applying to a smart grid management system we are building for Ho Chi Minh City.

Figure 2 illustrates different applications provided by the smart grid management system and are categorized into *Operational operations* (to compute the power output of the grid system); the *Alert applications* to provide alarm for abnormalities such as phase reversion, changes of current or voltage at

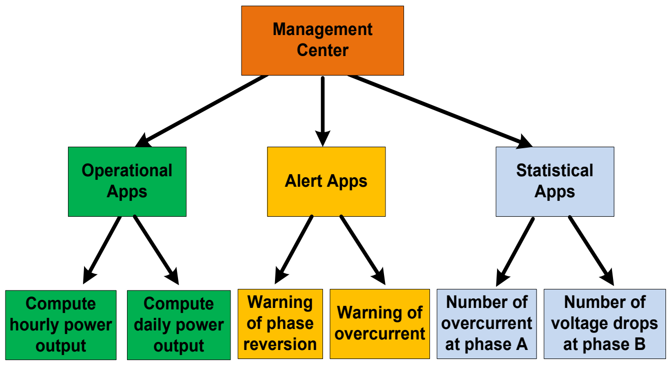
Management Center
Operational Apps
Alert Apps
Statistical Apps
Compute hourly power output
Compute daily power output
Warning of phase reversion
Warning of overcurrent
Number of overcurrent at phase A
Number of voltage drops at phase B

Fig. 2. Illustration of applications provided by a smart grid management system in Ho Chi Minh City

a considering phase, etc; the *Statistical applications* to provide statistics of the system status for a better management strategy.

Each application consists of several tasks or services. For example, the *Compute hourly power output* application consists of three tasks: i) collecting data from electricity meters at the considering region during a given time period; ii) aggregating the collected data at a computing node (e.g., a DC on the cloud or a HES - Head End System - which is a fog orchestration node in our proposed architecture); and iii) disseminating the computed result to users (e.g., staffs or managers). The proposed service placement mechanism will identify suitable computing nodes on a given network topology to maximize the numbers of services deployed on fog landscape while satisfying the requirement on response time of the applications.

We have conducted several simulations on task deployment using iFogsim [12] to compare the effectiveness, in terms of *latency, energy consumption* and *network usage*, of the proposed approach and the traditional cloud-based method. Simulations were conducted with 4 configurations described as follows:

- *Config1*: a DC (on the cloud) connects to a HES which connects to 4 modems, each of which connects to a meter.
- *Config2*: a DC connects to 2 HESs. Each HES connects to 4 modems each of which connects to a meter.
- *Config3*: a DC connects to 3 HESs. Each HES connects to 4 modems each of which connects to a meter.
- *Config4*: a DC connects to 4 HESs. Each HES connects to 4 modems each of which connects to a meter.

Figure 3 illustrates the *Config2* as an example, other configurations can be illustrated at the same way.

The computational and storage capacities as well as power consumption of computing devices in the network topology shown in Fig. 2 are presented in Table I. It should be noted that the power consumption (the last column) is shown in two levels representing the status when the device works with its maximum capacity (denoted as M) and when the device is idle (denoted as I). Table II shows the computational and storage resources in terms of CPU length (million instructions - MI) and network length (Byte).

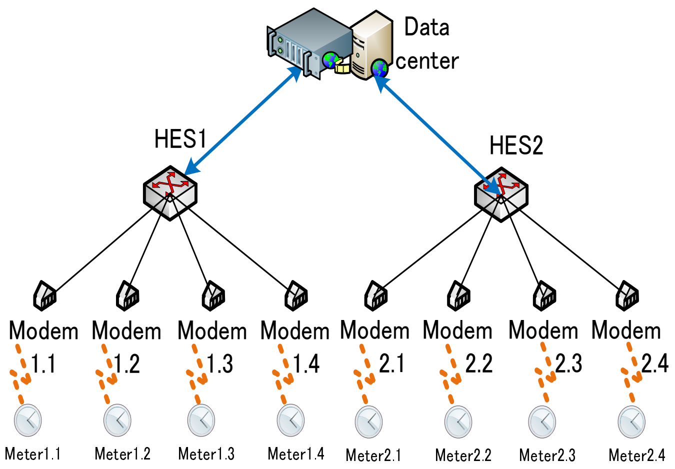

Fig. 3. The network topology of the Config2

TABLE I
DEVICES' COMPUTATIONAL AND STORAGE CAPACITIES AND POWER CONSUMPTION

Device / Capacity	**Computation capacity (MIPS)**	**Ram capacity (MB)**	**Power consumption (W)**
Modem	300	256	88.77(M) 82.7 (I)
HES	1,400	2,048	103.0(M) 83.25(I)
Cloud	10,000	10,240	4*103.0(M) 4*83.25(I)

We have deployed of services that consumed the data shown in Table II for computation both on the cloud and at appropriate nodes in the proposed fog architecture to evaluate the effectiveness of the proposed method. For the cloud approach, all the data is sent to the DC on the cloud for analysis. For the fog computing, the *electricity indicator* associated *data for billing* and the *Alert* are accumulated for computation at *modems* (right after the Meter which is a dump sensors shown in Fig. 3). Meanwhile, the data for *area-billing* and *invoice computation* is accumulated and processed at HES. The effectiveness of the proposed fog-computing method compared with the cloud-based method in terms of latency, energy consumption, and network usage is evaluated.

We have calculated the average response time of all the

TABLE II
DATA TYPE AND ITS RESOURCE REQUIREMENT

Data type	**Description**	**CPU length (MI)**	**N/W length (Byte)**
Electricity meter	Electricity meter data collected at the Meter	1000	500
Billing	Get data for billing	2000	1000
Alert	Alert data upon specific event	50	100
Area-billing	Data for billing in an area	1000	500
Invoice	For invoice computing	3000	100

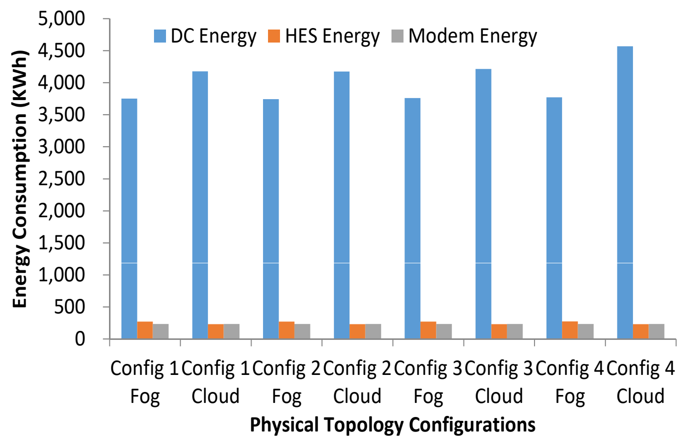

Fig. 4. Energy consumption

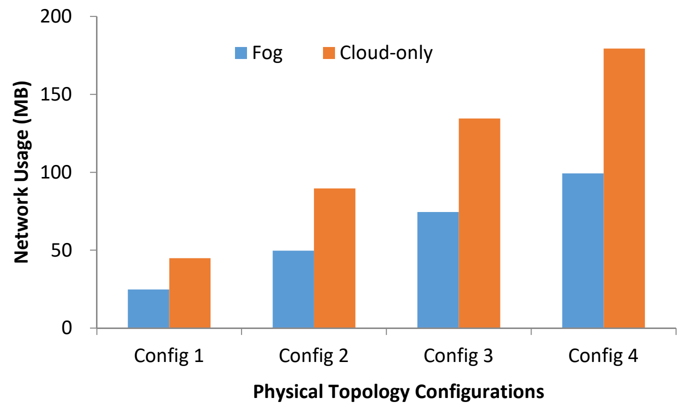

Fig. 5. Data traffic on the network

tasks in two cases: all the tasks are deployed on the cloud (for the cloud based approach) and tasks are deployed at the appropriate fog nodes in the fog computing model as discussed above. The average response time in the cloud computing is around 200ms while it is less than 30ms in the fog-computing approach.

For the energy consumption, Fig. 4 shows that the fog computing model significantly saves the energy compared with the cloud approach. Similarly, more network load (i.e., network usage) is required when applications are deployed on the cloud and when the topology becomes more complicated such as in the *Config4* as shown in Fig. 5. These results reveal that the proposed 3-tier approach can help to reduce the latency, mitigate the energy consumption and network usage (while better utilizing available virtualized resources) compared to its counterpart of traditional cloud-based computing.

VI. Conclusion

This paper proposes a novel approach to optimize task placement on fog computing paradigm. Our approach is not only cost-effective because of effectively using of virtualized resources which are already available, but also is viable to delay-sensitive applications which are common on the IoTs. The proposed approach helps to reduce energy consumption as well. We have evaluated the proposed approach via several simulations applying to smart grid applications. The results reveal that the proposed approach is effective in terms of reducing latency, energy consumption and network load in comparison with the conventional cloud computing model. We are planning to apply SDN approach to a better resource provision in fog computing paradigm.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2016.28.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepall, “Fog Computing and Its Role in the Internet of Things,” Proc. of MCC12, Helsinki, Finland, August 17, pp. 14-15, 2012.
[2] A.V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, R. Buyya, “Fog Computing: Principles, Architectures, and Applications,” Internet of Things: Principles and Paradigms, chap. 4. Morgan Kaufmann, 2016.
[3] S. Sarkar, S. Chatterjee, S. Misra, ”Assessment of the Suitability of Fog Computing in the Context of Internet of Things,” in IEEE Transactions on Cloud Computing , vol.PP, no.99, pp. 1-14.
[4] SBON - Stream-Based Overlay Network: “http://lsds.doc.ic.ac.uk/research/projects/sbon,” accessed Aug., 2017.
[5] Jayavardhana Gubbi et. al., “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems 29, pp. 1645-1660, 2013.
[6] Daniele Miorandi et. al., “Internet of things: Vision, applications and research challenges,” Ad Hoc Networks, Vol 10, 1497-1516, 2012.
[7] Adrien Lbre et. al., “Beyond The Cloud, How Should Next Generation Utility Computing Infrastructures Be Designed?”
[8] “http://parasol.cs.rutgers.edu/,” accessed Aug., 2017.
[9] V. Akman, “Context in Artificial Intelligence: A Fleeting Overview,” McGraw-Hill, Milano, 2002.
[10] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri, “Theories and uses of context in knowledge representation and reasoning,” Journal of Pragmatics, Vol. 35, No. 3 , pp. 455-484, 2003.
[11] A. K. Dey, “Understanding and Using Context,” Personal and Ubiquitous Computing Journal, Vol. 5, lss. 1, pp. 5-7 , 2001.
[12] H. Gupta, A.V. Dastjerdi, S. K. Ghosh, R. Buyya, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing En-vironments,” Tech. Rep. CLOUDS-TR-2016-2, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 2016.
[13] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwlder, B. Kold-ehofe, “Mobile Fog: A Programming Model for LargeScale Ap-plications on the Internet of Things,” In the 1st ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, pp. 15-20, 2013.
[14] M. Aazam, E. N. Huh, “Dynamic Resource Provisioning Through Fog Micro Datacenter,” In the 12th IEEE Int. Workshop on Manag-ing Ubiquitous Communications and Services, St. Louis, Missouri, USA, pp. 105-110, 2015.