VNF-B&B: Enabling Edge-based NFV with CPE Resource Sharing

He Zhu†, Changcheng Huang†
† Department of Systems and Computer Engineering
Carleton University, Ottawa, ON K1S 5B6, Canada
{hzhu, huang}@carleton.ca

***Abstract*—For embracing the ubiquitous Internet-of-Things (IoT) devices, edge computing and Network Function Virtualization (NFV) have been enabled in branch offices and homes to provide network functions on top of generic physical Customer-Premises Equipment (pCPE). While latency can be greatly reduced as most traffic does not need to be transmitted to a remote, centralized cloud, the resource limitation of a single pCPE makes it difficult for VNFs to be elastic enough upon usage surge. In this paper, we present VNF-B&B, an architecture featuring resource sharing of pCPE across the network edge. SP utilizes idle, shareable pCPE nodes as bed-and-breakfast places to deploy VNFs of other users for a certain period. By keeping the VNFs at the network edge, the cost is minimized for processing real-time data burst from IoT devices. Meanwhile, the traffic load to the core network and service delay is substantially reduced.**

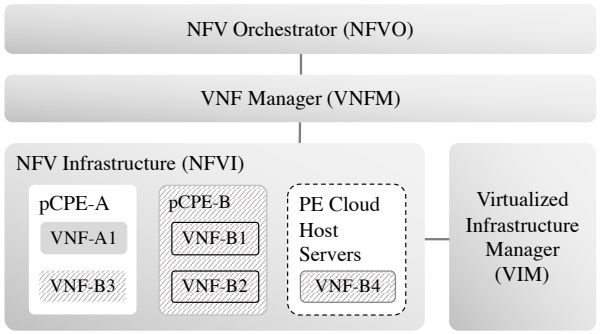


Fig. 1: The architecture of VNF-B&B based on the ETSI NFV [2] enables pCPE resources as part of NFVI along with PE cloud. E.g., when pCPE-B needs more VNFs, they can be deployed on both pCPE-A (VNF-B3) and PE (VNF-B4).

I. Introduction

In the wake of cloud computing and Network Function Virtualization (NFV), Service Providers (SPs) leverage virtual Customer Premises Equipment (vCPE) as Virtual Network Function (VNF) instances on top of generic physical Customer-Premises Equipment (pCPE), in search of rebuilding a dynamic revenue stream [11]. There may be enough resources for pCPE to deploy VNFs locally [10], while pCPE can also coordinate with the cloud if VNF scale-out is needed to accommodate heavier usage. Taking advantage of centralized cloud services in the core networks has benefits [3] because of scalable and flexible computing capabilities. However, large-scale deployments of Internet-of-Things (IoT) devices bring challenges to VNFs running in a centralized cloud, as the network traffic load would be drastically increased by transmitting data between the core and the edge of the network, causing high processing delay or even service outage due to the congestion of the network. Meanwhile, high usage of the cloud networks would jack up the price per usage, resulting in higher-than-expected operating expense (OPEX).

The concept of edge computing [8] was proposed to move the initial handling of raw data down to the Provider Edge (PE). While a single pCPE node has limited resources and typically serves a designated customer, the aggregated computing capabilities of pCPE across the network edge can be powerful. If the spare resources of pCPE can be shared within the edge, VNFs will be able to roam around. It is certainly promising to utilize hidden computing resources. However, CPE resource sharing faces challenges:

(i) Lack of models to describe how offloading to the edge can mitigate the core network throughput and minimize the OPEX.
(ii) The pCPE availability will be jeopardized if it no longer has enough resources to host VNFs.
(iii) Users need to be motivated to consent contributing their vCPEs for resource sharing. A pricing strategy is required to benefit both the SP and its end users.

In this paper, we propose an architecture to allow sharing resources of pCPE within the network edge, namely VNF-B&B. When sharable pCPE is not actively used by its owner, it will be treated as a "bed-and-breakfast" place for VNF to "stay". As shown in Figure 1, SP will have the permission to deploy VNFs for other users from the same edge network. The goal of VNF-B&B is to find the optimal strategy to deploy the VNFs with minimal cost.

We divide the contents into the following sections. The related work is illustrated in Section II. Section III formulates the problem. Section IV proposes the VNF-B&B algorithm. Simulation results are shown in Section V and Section VI concludes the paper.

978-1-5386-3531-5/17/$31.00

II. Related Work

NFV has been widely adopted by SPs and vendors to reduce cost and to provide scalable and elastic services [4]. Research on cloud-based NFV has been done to ensure that VNFs run at optimum levels in the cloud [7].

Fog computing was proposed in [1] and was anticipated to become an essential part of cloud computing with the number of Internet-of-Things (IoT) growing explosively. Vaquero et al. [12] proposed a comprehensive definition of the fog covering its features and impact, including device ubiquity, challenges on service and fog-based network management, levels of device connectivity, and privacy. Edge clouds were presented as entry points for IoT, which could be parts of the Enhanced Packet Core (EPC). The scenarios of fog computing in several domains were discussed in [9], which pointed the research direction of leveraging the edge of the network from high levels.

Virtualization in edge networks as a form of fog computing, including NFV, have been given a close look. Manzalini et al. visioned potential value chain shifts and business opportunities in [6] by emerging paradigms such as SDN and NFV. A platform called Network Functions At The Edge (NetFATE) was proposed in [5] as a proof of concept of an NFV framework at the edge of a Telco operator networks. Each CPE node was realized with a generic-purpose computer installed with a hypervisor and virtual switches.

While existing work can prove that deploying VNFs on the edge of the network is feasible, the benefits of resource sharing across different CPE nodes are not addressed and demonstrated.

III. Problem Formulation

A network edge is defined to be formulated by a PE router and all pCPE nodes under it, modeled as a directed graph $G = (V, L)$. Set V represents all pCPE nodes in the network. Set L has all connected paths between each two pCPE nodes. A pCPE node in V is denoted by v, such that $v \in V$. Define N_v as the total number of pCPE nodes in V, so that any pCPE node can be represented by $v_i \in V, \forall i \in [1..N_v]$. For any data transmitted from one node v_i to another node v_j in V, there exists a link l_{ij} via between the PE switch and the datacenter, such that $l_{ij} \in L, \forall i, j \in [1..N_v], i \neq j$. The network edge is connected to the core network via one single link, denoted by l_c.

A. VNF Types and Flavors

A VNF provisions a specific type of service. We use f to define a VNF instance. The CPU, memory, and bandwidth requirements of f is then denoted by $U(f)$, $M(f)$, and $B(f)$, respectively. We assign a to identify a specific type of network function and n_a to be the number of network function types. An instance of VNF with type a can then be represented by $f(a)$.

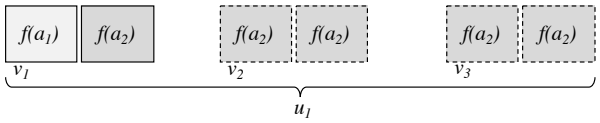


Fig. 2: An example of VNF instances grouped by u_1, while they are deployed on 3 different pCPE nodes.

B. Places to Deploy VNF instances

Regardless of where the instances are deployed, we represent a pCPE node's owner by $u_i, i \in [1..N_v]$, with the assumption that each pCPE node is owned by a unique user. Figure 2 illustrates an example of VNF instances grouped by u_1. Placement decisions are made based on the flavors of VNF instances and the resource capacities of pCPE nodes. A VNF instance of a certain user u_i, denoted by $f(a, u_i)$, can be deployed at either of the locations below.

1) B&B deployment: For a VNF instance $f(a, u_i)$, any pCPE node within the same network edge can be considered as a candidate place to deploy, also known as B&B deployments. For a B&B deployment of $f(a, u_i)$, its notation can be extended to $f(a, u_i, v_j)$, where v_j is the place to deploy f.

2) Cloud deployment: For a cloud deployment of $f(a, u_i)$, its notation can be extended to $f(a, u_i, c)$, where c is the remote cloud to deploy f. We use the set F_c to define all VNF instances deployed on cloud.

C. Factors to Impact Placement Decisions

1) pCPE Resource Capacity: Every pCPE node v has its own resource capacity to host a limited number of VNF instances. Our discussion focuses on resources of virtual CPUs (vCPUs) and memory. Let $U(v)$ denote the number of vCPUs v can provide. Let $M(v)$ be the total amount of memory for VNF instances from v.

2) Remote Function Delay: The remote function delay of a VNF instance f offloaded to the cloud is defined as the latency introduced by offloading the VNF to the cloud, denoted by $t(f)$, while $T_{max}(f)$ is the maximum allowed network delay for a specific network function instance. Let $B(l_c)$ be the total bandwidth of l_c, and $t(l_c)$ be the remote function delay of l_c. VNF instances take up the bandwidth of l_c to communicate with the pCPE nodes. The latency of the core network is therefore highly correlated to the bandwidth consumption of l_c.

During peak hours, the severity of congestion is reflected by the residue bandwidth of the path from the edge switch to the cloud, denoted by $R(l_c)$. Let F_c denote the set of VNF instances deployed in the remote cloud. $R(l_c)$ is then $R(l_c) = B(l_c) - \sum_{f \in F_c} B(f), \forall f \in F_c$. We define the delay not directly caused by the

network edge as a constant T_d. For all VNF instances offloaded to the cloud, there must be

$$t(l_c) \leq T_{max}(f), \forall f \in F_c \tag{1}$$

Equation (1) draws a limit of how much VNF offloading can be done, since a busier cloud environment would increase $t(l_c)$. We model $t(l_c)$ to be inversely proportional to $R(l_c) + b$, and proportional to T_d with b as a constant scoping the maximum remote function delay when the bandwidth of l_c is depleted. Combined with Equation (1), there is

$$\frac{T_d}{R(l_c) + b} \leq T_{max}(f), \forall f \in F_c \tag{2}$$

D. Cost of Offloading to Edge Network

By encouraging resource sharing participation, it is necessary to give incentives to users based on the amount of resource shared. We denote the unit incentive of CPU, memory and bandwidth usage for a pCPE node v as $w_U(v)$, $w_M(v)$ and $w_B(v)$, respectively. The cost of offloading a VNF instance f to any of the pCPE node, denoted by $S(f, v)$, is calculated as $S(f, v) = w_U(v)U(f) + w_M(v)M(f) + w_B(v)B(f)$.

E. Cost of Offloading to Cloud

We use c to represent the remote cloud location to deploy VNF instances. The total amounts of vCPUs, memory, and network bandwidth available in the cloud are denoted by $U(c)$, $M(c)$, and $B(c)$. Let $w_U(c)$ and $w_M(c)$ stand for the unit cost for consuming the cloud's CPU and memory resource. We model $w_U(c)$ and $w_M(c)$ to be inversely proportional to the cloud's remaining vCPUs and memory with the constant of proportionality W_U and W_M. The remaining number of vCPUs and memory are denoted by $R_U(c)$ and $R_M(c)$. The total cost of vCPUs and memory for a VNF instance f to be offloaded to the cloud, $S_U(f, c)$ and $S_M(f, c)$, are then:

$$\begin{aligned} S_U(f,c) &= w_U(c)U(f) = \frac{W_U U(f)}{U(c) - \sum_{f' \in F_c} U(f') + \delta} \\ S_M(f,c) &= w_M(c)M(f) = \frac{W_M M(f)}{M(c) - \sum_{f' \in F_c} M(f') + \delta} \end{aligned} \tag{3}$$

In Equation (3), δ is a small positive number to avoid dividing by zero. Let $w_B(c)$ denote the unit cost of the remote cloud's network bandwidth. Define $w_B(c)$ to be proportional to $t(l_c)$ with the constant of proportionality W_B. we define the total cost of bandwidth used between the VNF instance f and the cloud as $S_B(f, c)$,

$$\begin{aligned} S_B(f,c) &= w_B(c)B(f) = W_B t(l_c)B(f) \\ &= \frac{W_B T_d B(f)}{B(l_c) - \sum_{f' \in F_c} B(f') + b} \end{aligned} \tag{4}$$

From Equations (3) and (4), the cost of offloading a VNF instance f to the cloud, denoted by $S(f, c)$, is then calculated as

$$S(f,c) = S_U(f,c) + S_M(f,c) + S_B(f,c) \tag{5}$$

In Equation (5), δ is a small positive number to avoid dividing by zero.

F. Objective and 0-1 Integer Programming Formulation

Define $X(f, v)$ as a group of Boolean variables representing if each VNF instance f is deployed on the B&B node v. Define $X(f, c)$ as another group of Boolean variables representing if each VNF instance f is deployed on the remote cloud c. The objective of the optimization is to minimize the total offloading cost.

$$X(f,v) = \begin{cases} 0, & f \text{ not deployed on } v \\ 1, & f \text{ deployed on } v \end{cases} \tag{6}$$

$$X(f,c) = \begin{cases} 0, & f \text{ not deployed on cloud} \\ 1, & f \text{ deployed on cloud} \end{cases} \tag{7}$$

$$Minimize \sum_{f \in F} \sum_{v \in V} S(f,c)X(f,c) + S(f,v)X(f,v) \tag{8}$$

$$\begin{aligned} w.r.t. \quad & X(f,v), X(f,c) \\ s.t. \quad & X(f,c) + \sum_{v \in V} X(f,v) = 1, \forall f \in F \end{aligned} \tag{9}$$

$$U(c) - \sum U(f)X(f,c) \geq 0 \tag{10}$$

$$M(c) - \sum M(f)X(f,c) \geq 0 \tag{11}$$

$$\frac{T_d}{B(l_c) - \sum B(f)X(f,c) + b} \leq T_{max}(f), \forall f \in F_c \tag{12}$$

$$U(v) - \sum U(f)X(f,v) \geq 0, \forall v \in V \tag{13}$$

$$M(v) - \sum M(f)X(f,v) \geq 0, \forall v \in V \tag{14}$$

Remarks:

- Function (8), the objective function, minimizes the total cost of offloading VNFs instances.
- Constraint (9) ensures that every VNF instance $f \in F$ is only deployed at one place.
- Constraints (10) and (11) are the capacity bounds of the CPU and memory of the cloud.
- Constraint (12) sets a limit for instances offloaded to the cloud due to the residue bandwidth of l_c.
- Constraints (13) and (14) are the capacity bounds for CPU and memory of every pCPE node.

IV. VNF-B&B Placement Algorithm

From the 0-1 integer programming, we design an algorithm to achieve the minimal cost by choosing the optimal place to deploy all VNF instances.

For every request of deploying a VNF instance, we first use Algorithm 1 to check the placement eligibility of each pCPE node and the remote cloud. By calling the function GetCandidates(f), a list of candidate places

Algorithm 1 VNF-B&B Resource Eligibility Algorithm

function GETCANDIDATES(f)
  create an empty list `candidates`
  for all $v \in V$ **do**
    if ISRESOURCEENOUGH(v,f) **then**
      add v to `candidates`
  if ISRESOURCEENOUGH(c,f) **then**
    add c to `candidates`
  return `candidates`
function ISRESOURCEENOUGH(v, f)
  if v is c **then** ▷ Check delay for cloud
    `link_bw` $\leftarrow B(l_c) + b$
    for all $f' \in F_c$ **do**
      `link_bw` $\leftarrow$ `link_bw` $- B(f')$
    `delay` $\leftarrow T_d /$ `link_bw`
    if `delay` $< T_{max}(f)$ **then**
      return `false` ▷ Too much delay
  return $R_U(v) \geq U(f)$ and $R_M(v) \geq M(f)$ and $R_B(v) \geq B(f)$

will be returned from the input of a VNF instance f and the current resource level. Algorithm 2 and 3 provide implementation of the cost model from Section III and define functions to choose the place for a VNF instance f at the lowest cost.

Algorithm 2 VNF-B&B Cost Estimation Algorithm

function CLOUDCOST(f)
  `cpu_left` $\leftarrow U(c) + \delta$
  `memory_left` $\leftarrow M(c) + \delta$
  `link_bw` $\leftarrow B(l_c) + b$
  `cost` $\leftarrow 0$
  for all $f' \in F_c$ **do**
    `cpu_left` $\leftarrow$ `cpu_left` $- U(f')$
    `memory_left` $\leftarrow$ `memory_left` $- M(f')$
    `link_bw` $\leftarrow$ `link_bw` $- B(f')$
  `cost` $\leftarrow$ `cost` $+ W_U U(f) /$ `cpu_left` $+ W_M M(f) /$ `memory_left` $+ W_B T_d B(f) /$ `link_bw`
  return *cost*
function BNBCOST(f, v)
  `cost` $\leftarrow 0$
  if user of f does not own v **then**
    `cost` $\leftarrow [$`cost` $+ w_U(v)U(f) + w_M(v)M(f) + w_B(v)B(f)] \times (1 + \gamma)$
  return `cost`

V. NUMERICAL RESULTS

The numerical results based on simulations are shown in this section. Our goals are to verify the benefits of leveraging B&B nodes, compared to using a centralized cloud alone. We first choose the constants used in the algorithms: $T_d = 50000$, $W_U = W_M = W_B = 1000$, $\delta = 1$, $b = 1$, $\gamma = 1$.

We create 99 pCPE nodes with random levels of initial resources. With the cloud as an extra node, there are 100 nodes for deployments. As seen in Figure 4a, we color the cells according the resource type of the lowest level of a node.

Algorithm 3 VNF-B&B Place Selection Algorithm

function CHOOSEPLACE(f)
  `place` $\leftarrow$ `none`
  `cost` $\leftarrow +\infty$
  `candidates` $\leftarrow$ GETCANDIDATES(f)
  for all `candidate` **in** `candidates` **do**
    if `candidate` is c **then**
      `cur_cost` $\leftarrow$ CLOUDCOST(f)
    else
      `cur_cost` $\leftarrow$ BNBCOST(f, `candidate`)
    if `cur_cost` $<$ `cost` **then**
      `place` $\leftarrow$ `candidate`
      `cost` $\leftarrow$ `cur_cost`
  return `place`

TABLE I: Pre-defined flavor types for simulation

Name	U/M(GB) /B(Gbps)	T_{max} (ms)	Name	U/M(GB) /B(Gbps)	T_{max} (ms)
F1	1/1/1	1000	F6	4/4/4	1000
F2	2/2/2	100	F7	4/4/4	10000
F3	2/2/2	1000	F8	8/8/8	100
F4	2/2/2	10000	F9	8/8/8	1000
F5	4/4/4	100	F10	8/8/8	10000

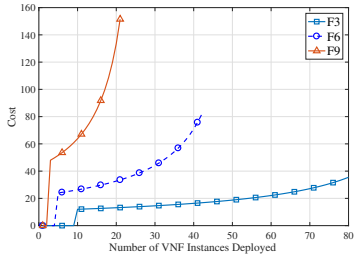
We pre-define 10 types of VNF flavors, as shown in Table I, with different requirements of resources and max delays allowed. The system can work in three modes:

1) *Local Mode*: Deploy only on the pCPE node the user owns. Cloud and B&B nodes are not allowed.
2) *Local+Cloud Mode*: Deploying locally on the pCPE node the user owns, as well as on the remote cloud.
3) *Local+Cloud+B&B Mode*: local, cloud, and B&B deployments are all enabled.

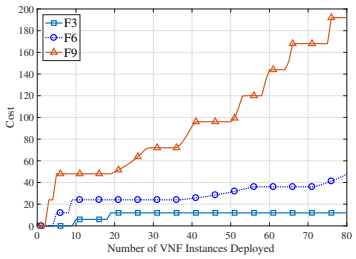
Figure 5 shows the numbers of deploying each of the 10 pre-defined flavors with the 3 modes. From the results, we learn that the numbers of instances deployed for all 10 flavors have dramatically increased by leveraging B&B nodes. The bottleneck of remote function delay is greatly relieved by the B&B nodes hosting instances, because B&B deployments do not put extra traffic to the core network.

Three flavors, F3, F6, and F9, are picked to investigate the trends of cost increase as more VNF instances are deployed in the system. Figure 3a and 3b demonstrate the changes of costs to deploy a new VNF instance on the cloud in two different modes, as the numbers of deployed instances go up. In *Local+Cloud+B&B Mode*, the costs are lower when deploying the same numbers of instances in the system. The results have demonstrated the ability of the B&B nodes to redirect the load off the cloud and to reduce the overall cost, even if offering incentives to the users.

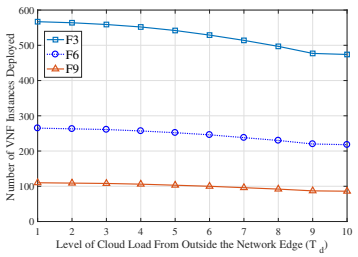
When T_d is higher, the ability of the cloud hosting VNF instances may be reduced. Under *Local+Cloud+B&B Mode* and for the three flavors F3, F6, and F9, we increase the level of T_d by 1 each time,



(a) Cost hikes when the cloud load increases: *Local+Cloud Mode*.

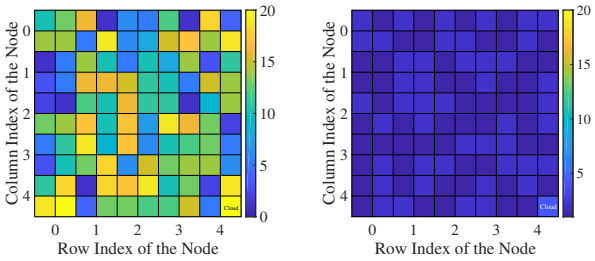


(b) Cost hikes when the cloud load increases: *Local+Cloud+B&B Mode*.



(c) No. of VNF instances deployed with three flavors under various cloud load.

Fig. 3: Numerical results of the VNF-B&B algorithm.



(a) Initial resource levels of all nodes

(b) remaining resource levels after deploying

Fig. 4: The 99 B&B nodes and the cloud node are arranged as a 10×10 matrix.

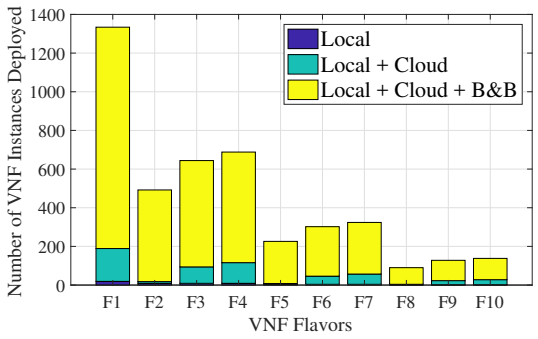


Fig. 5: Total number of VNF instances deployed for the network edge with various flavors and modes.

and repeat the deployment for 10 times, as shown in Figure 3c. From the results, the capacity of the system is affected by the increase of T_d. However, the impact becomes less significant as the level of T_d increases.

In *Local+Cloud+B&B Mode*, all B&B nodes participate in hosting VNF instances. We examine the resource levels after the system resources are depleted. When all VNF instances deployed are of the flavor F1, the resource levels after the maximum number of instances are deployed are displayed in Figure 4b. The results have demonstrated the ability of the VNF-B&B algorithm to extract the resources to deploy more instances following the best-effort basis.

VI. Conclusions

In this paper, we have presented the architecture and algorithms to share resources of pCPE nodes and deploy VNFs at the network edge for reducing delay. Future work includes factoring in the service up and down of B&B nodes and considering more factors impacting the deployment placement.

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*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[2] ETSI Industry Specification Group (ISG) NFV. ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV): Architectural Framework. 2014.

[3] N. Herbaut, D. Negru, G. Xilouris, and Y. Chen. Migrating to a nfv-based home gateway: Introducing a surrogate vnf approach. In *2015 6th International Conference on the Network of the Future (NOF)*, pages 1–7, Sept 2015.

[4] R. Jain and S. Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, 2013.

[5] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene. An open framework to enable NetFATE (network functions at the edge). In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–6. IEEE, 2015.

[6] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi. Clouds of virtual machines in edge networks. *IEEE Communications Magazine*, 51(7):63–70, July 2013.

[7] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba. Network Function Virtualization: State-of-the-art and Research Challenges. *CoRR*, abs/1509.07675, 2015.

[8] M. Satyanarayanan. The Emergence of Edge Computing. *Computer*, 50(1):30–39, Jan 2017.

[9] I. Stojmenovic and S. Wen. The Fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*, pages 1–8, Sept 2014.

[10] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz. Ease: Epc as a service to ease mobile core network deployment over cloud. *IEEE Network*, 29(2):78–88, March 2015.

[11] E. Telecom. White Paper: The Definitive Guide to vCPE. https://goo.gl/Snz5Y6. [Online; accessed 16-Aug-2017].

[12] L. M. Vaquero and L. Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, Oct. 2014.