

A Comprehensive Resource Management and Placement for Network Function Virtualization

Thi-Minh Nguyen, Serge Fdida, Tuan-Minh Pham

▶ **To cite this version:**

Thi-Minh Nguyen, Serge Fdida, Tuan-Minh Pham. A Comprehensive Resource Management and Placement for Network Function Virtualization. The 3rd IEEE Conference on Network Softwarization (IEEE NetSoft 2017), Jul 2017, Bologna, Italy. hal-01496518

HAL Id: hal-01496518

https://hal.sorbonne-universite.fr/hal-01496518

Submitted on 27 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Comprehensive Resource Management and Placement for Network Function Virtualization

Thi-Minh Nguyen
UPMC Sorbonne Universités
LIP6 Laboratory
France
Email: thi-minh.nguyen@lip6.fr

Serge Fdida
UPMC Sorbonne Universités
LIP6 Laboratory
France
Email: serge.fdida@upmc.fr

Tuan-Minh Pham
Hanoi National University of Education
Vietnam
Email: minhpt@hnue.edu.vn

***Abstract*—Network service providers have to cope with the growing on-demand need from end-users as well as the diversity of usage. The softwerization and cloudification of the network components offer an interesting solution to achieve the agility necessary to dynamically match the requirement with the level of resource consumption. This materializes with the deployment of Network Functions Virtualization (NFV) where Virtual Network Functions (VNFs) may be chained together to create network services. This paper explores important design and architectural issues related to this approach. We study the resource allocation problem in an NFV system for minimizing its cost under constraints on interconnectivity among VNFs, system resources, and service requirements. We formalize the problem in a comprehensive manner taking into account a broad set of relevant parameters. The static (offline) and dynamic (online) cases are considered. We propose and analyze three heuristic algorithms: two for handling large dimensions of the offline problem and one designed to address the online scenario.**

The evaluation shows that our solutions outperform the state of the art [1] with respect to critical performance index. Finally, we focus on the online scenario, evaluate the impact of migrating a set of running demands, and propose a simple migration technique.

***Index Terms*—NFV, Network Function Virtualization, resource allocation, optimization and migration.**

I. Introduction

Softwerization of network components is a candidate to provide the agility requested by the increasing on-demand need of customers as well as the ability to gently match those needs with the resource consumption. This trend is achieved by using a cloud approach where virtualization techniques are intensively exploited. Although this trend is prevalent, it is still necessary to better understand the right level of abstraction and hence performance that will be made possible with this approach. NFV relying upon virtualization techniques should support network operators to meet the growing customer requirements while controlling capital and operational expenditures.

In order to enable such an agile solution, a network service can be decomposed into an ordered sequence of VNFs (Virtual Network Functions), which can run on several standard physical nodes. Therefore, the resources allocated to a VNF instance will impact the capacity and performance of the network service as a whole. This raises important issues related to NFV deployment such as: 1) How and where to locate and chain VNFs? 2) How to distribute resources to VNFs? and 3) What is the cost of NFV deployment in a network? Such problems are different from the VM placement optimization in cloud computing as VNFs are associated with a single network service. In this paper, we consider a joint problem of VNFs allocation upon service requests from users and routing paths for chaining them. We develop a comprehensive solution addressing the static and dynamic scenario, when taking into account the execution order of VNFs in a service demand with regard to resource constraints for nodes and links.

The major contributions of this paper are as follows:

- We formulate the optimization problem of VNF placement as a quadratic program (QP) solving multiple objectives including optimal service location and optimal routing among VNF instances of a network service, under both resource and traffic cost constraints, simultaneously optimizing the acceptance ratio of new demands. Both the static and dynamic problems are considered.
- We propose three efficient heuristics for large dimensions of the problem within an acceptable computation time.
- We evaluate our proposed solutions in various scenarios and compare it against the ProvisionTraffic (PT) algorithm [1] that illustrates the state of the art, as well as a Random Algorithm.
- We provide a cost comparison between a network relying on NFV and a network that does not use NFV and discuss some architectural implications.
- We also propose a simple migration technique for the dynamic problem. The result provides guidelines for the percentage of traffic to be migrated in order to achieve the best gain.

The rest of this paper is organized as follows. Section II reviews the related work. In Section III, we model the NFV system under study and formalize the NFV placement problem. We present our solutions in Section IV. In Section V, we validate the algorithms and discuss the results. Section VI concludes the paper.

II. Related Work

Managing complexity is the major challenge faced by ISPs who need to focus on network efficiency, performance, cost, and resource utilization. These issues relate directly to the

resource sharing and scheduling problem. In recent years, this concern has received a lot of attention from the community and has also evolved with the emergence of the NFV concept [2]. In this section, we will discuss some of the most important research works related to the network function placement and chaining problem. We start the section by reviewing recent efforts aimed at evaluating the technical feasibility of deploying network functions on virtual networks in both static and dynamic environments. Then we highlight open issues and emphasize the competitive contributions of our work.

The NFV placement problem is related to both the virtual network embedding (VNE) [3] and the virtual data center embedding (VDCE) [4]. The survey [5], [6] provides a valuable reference for previous works related to the VNF placement problem. In [6], Herrera and Botero present a comprehensive state of the art of NFV resource allocation (NFV-RA) problem. In their novel classification, the main approaches for solving NFV-RA are described as three stages: a, VNFs - Chain Composition (VNF-CC); b, VNF - Forwarding Graph Embedding (VNF-FGE) and c, VNFs - Scheduling (VNF-SCH). In this paper, we focus on the second stage.

Recently, Rasid Mijumbi et al. proposed an online mapping and scheduling problem in an Network Function Virtual Infrastructure (NFVI) [7]. However, they did not consider links between virtual nodes while the traffic cost among virtual nodes plays an important role in distributing resources to the required services.

For the NFV location problem, Rami Cohen et al. [8] presented a near optimal approximation algorithm guaranteeing a placement with theoretically proven performance. However, they did not consider the chaining of VNFs in requested services. The same concern is observed in other papers [1], [9], [10] where the authors have considered a constraint on link capacity, but ignored the order of functions in a SFC request.

More recently, in 2016, Kuo et. al. [11] studied the joint problem of VNF placement and path selection to better utilize the network. They considered the link bandwidth required by each demand, provided feasible solutions to support all the demands with the purpose to maximize the number of accepted demands. However, they ignored the issue of serving all the arriving demands with the minimum cost.

The publications mentioned above provide a static allocation strategy that pre-computes the resource allocation for each session. There are some disadvantages to this approach such as the resource utilization or the supplied capacity when handling fluctuating demands. Therefore, it is important to dynamically adjust the demand based on the current status. In [12], [13], authors proposed a multi-objective formulation of the Virtual Machine Placement considering Service Level Agreement.

In [14], Lukovszki et al. formulated the offline (SCEP: Service Chain Embedding Problem) and online problem (OSCEP: Online SCEP). They assumed that all potential paths that can be routed through VNFs chaining for each demand are known. In addition they did not consider the resources price when deploying the network functions. PACE [15] addressed the VNF placement for unordered service chains in cloud, with the objective to satisfy as many tenant's requests as possible. However, in their model, they have to decide about the placement of middleboxes before receiving the request. In other words, the placement of middleboxes are fixed. Also, Elias et al. [16] formulated the centralized version of VNF-FGE as a non-linear integer optimization model and assume that the placement of functions is fixed. Recently, Sun et. al. investigated the offline and online solutions for the VNF placement problem with the aim of minimizing the deployment cost [17]. They took into account the chaining of VNFs, but they only considered a pre-defined set of service chaining.

In contrast, our work explicitly considers the execution order of VNFs in each requested service, integrating resource constraints for nodes and links. We solve VNFs allocation and routing in a coordinated way, that is VNFs in each customer demand are placed and routed simultaneously. Our model consists of both the OFFLINE and ONLINE case. We compare our solution with the PT algorithm to prove its efficiency for the resource cost. We also provide an evaluation of the cost efficiency of NFV deployment by comparing the overhead of an NFV system with the one of a non-NFV system. Furthermore, we propose a simple migration technique for the ONLINE case.

III. Preliminaries

We first describe a resource allocation problem in a system implementing NFV. Then, we formulate the problem as a quadratic program (QP) dealing with both optimal service location and optimal routing under constraints on service function chaining and system resources.

A. System Description

In this section, we introduce two scenarios: the static problem (OFFLINE case) and the dynamic problem (ONLINE case).

1) OFFLINE case: We study a NFV system where a network function is offered as a service. The system is composed of three components, including a set of VNFs, a set of customer demands and a virtual network that provides resources to the VNFs. VNF is a software implementation of a Network Function (NF) deployed on an NFVI. Each demand requests access to a network service (NS) that requires special functionalities. It can be composed of several VNFs in order to support advanced network connectivity, e.g. deep packet inspection, firewall, load balancer. In a NS, VNFs will need diverse resources such as CPU, memory, storage, bandwidth, etc. It is recognized that virtualized solutions are quite costly in resource consumption. Resources are distributed anywhere on the NFVI. A customer demand materializes as a packet flow from a source to a destination visiting a set of nodes where the required VNFs are deployed.

How to efficiently share resources for all demands with delay constraints or limited capacity is an important challenge for NFV providers. It is likely that the solution achieved will trade-off agility with efficiency. For instance, a NFV solution

TABLE I: Summary of Variables

Variable	Meaning
(n,m,l)	Number of virtual nodes, VNFs, and demands
(V,E)	Set of virtual nodes and links
(D_n,D_o)	Set of new demands and old demands
$D_m \subseteq D_o$	Set of old demands that will be migrated
F	Set of VNFs
R	Set of resources (CPU, STORAGE, MEMORY, etc.)
F^d	Set of ordered VNFs of demand d
ℓ_d	The number of VNFs in set F^d
ϕ_f^d	The order of function f in the chain of VNFs F^d
$w_{v_1 v_2}$	Bandwidth capacity of link between $v_1, v_2 \in V$
c_v^r	Resource capacity $r \in R$ of node $v \in V$
λ^d	SLA coefficient of demand d
$p_v(\lambda^d)$	The price of resource charged per hour of node v, according to λ^d
μ_f^r	Required resource $r \in R$ of function $f \in F$
(s^d,t^d)	Source node and destination node of demand d
b^d	Required bandwidth of demand d
(a_s^d,Δ^d)	Starting and total processing time of demand d
p_b	Price of bandwidth charged per hour
η	Migration percentage
x_{fv}^d	A binary variable that equals to 1 if and only if virtual node v hosts function f of demand d
y_{uv}^d	A binary variable that equals to 1 if and only if link (u,v) is used by demand d
χ_{uv}^d	A binary variable that equals to 1 if and only if node u is a ancestor node of v in the routing path of d
z^d	An integer variable that equals to 1 if new demand d is served, -1 if old demand d is migrated and 0 otherwise
$\hat{x}_{fv}^d$	A binary variable that equals to 1 if and only if virtual node v hosts function f of old demand d
$\hat{y}_{uv}^d$	A binary variable that equals to 1 if and only if link (u,v) is used by old demand d
(min_l,max_l)	Minimum and maximum number of demands

may take a longer path than the likely shortest path followed by a non-NFV system. However, it is still applicable if the delay lies within an acceptable range.

A virtual network is represented as an edge-weighted undirected graph denoted by $G = (V, E, w)$ where V is a set of virtual nodes, E is a set of links, and w is the bandwidth capacity assigned to each link. Each node $v \in V$ is associated with two components: 1) c_v^r that denotes a set of resource capacities with $r \in R$, where R denotes the set of resources (CPU, STORAGE, and MEMORY, etc.) and 2) $p_v(\lambda^d)$ that is the price of resource charged per hour (\$/hour) according to a Service Level Agreement (SLA) coefficient λ^d of the customer d. We denote D_n as a set of l customer demands. A demand $d \in D_n$ is represented by a set of parameters $(s^d, t^d, b^d, F^d, \Delta^d)$ where s^d is a source, t^d is a destination, b^d is a required bandwidth, $F^d = \{f_{i_1}, ..., f_{i_k}\}$ is an ordered chaining of sub-set VNFs and Δ^d is an average total processing time of demand d. We assume that each VNF $f_i \in F^d$ only appears once in each demand $d \in D_n$. We define $F = \{f_i | i = \{1, ..., m\}\}$ as a set of m VNFs. Each function $f_i \in F$ has a vector $\mu_{f_i}^r$ representing its resource requirement $(r \in R)$. In the OFFLINE model, we minimize both the computing-resource cost and the bandwidth cost under constraints on resource and bandwidth capacity with respect to VNFs chaining.

2) ONLINE case: We consider a scenario where an operational network is serving a set of customer demands D_o (called old demands) . At this stage, a set of VNFs are already deployed and the routing paths for the demands in D_o are also provisioned. Consider new demands D_n are incoming and the network operator needs to provision the required VNFs and routing paths for them. Maximizing the number of served demands and minimizing the system resource cost are the research challenges addressed in the sequel.

We consider an interval τ (according to the system configuration) where we will update the status of all demands running in the system. Upon completion, the system releases the resources allocated to a demand. At time t, we collect new demands D_n that arrived during the period $(t - \tau)$ to t.

We propose a method to select the subset of old demands to be migrated in Section-IV-B2.

The ONLINE model addresses the multiple objectives:

- Minimize resource cost on nodes to satisfy the demands
- Minimize bandwidth cost on links from source to destination passing through VNFs
- Maximize the number of accepted demands
- Minimize the penalty cost for VNF migration

B. Formal Problem Statement

We aggregate both cases and formulate the above system as a quadratic problem with the following inputs:

- Set of m functions $F = \{f_i | i = \{1, ..., m\}\}$.
  - μ_f^r: a resource requirement of function $f \in F$
- Set of old demands $D_o = \{(s^d, t^d, b^d, F^d, \Delta^d, a_s^d, \hat{x}_{vf}^d, \hat{y}_{uv}^d)\}$. Each old demand $d \in D_o$, we know a_s^d is the starting process time, $\hat{x}_{vf}^d$ and $\hat{y}_{uv}^d$ indicate their solution deployed on NFVI.
- $D_m \subseteq D_o$ is the set of old demands selected to be migrated. A migration percentage is denoted by $\eta = \frac{|D_m|}{|D_o|}$
- Set of new demands $D_n = \{(s^d, t^d, b^d, F^d, \Delta^d)\}$
- An edge-weighted undirected graph $G = (V, E, w)$ is representing the virtual network.

The notations can be found in Table I.

We define the outputs of our problem as a set of decision variables. x_{fv}^d is a binary variable that equals to 1 if and only if node v hosts function f of demand d. Similarly, the binary variables y_{uv}^d equals to 1 if and only if demand d uses link (u, v). The binary variables χ_{uv}^d equals to 1 if and only if node u is an ancestor node of v in the routing path of demand d. Finally, z^d is an binary variable that equals to 1 if a new demand d is served or an old demand d is migrated, and equals to 0 otherwise.

We aim to:

1) Minimize nodes resource cost allocated for demands

$$\begin{aligned} U_n(t) = \left(\Delta^d - t + a_s^d\right) \sum_{d \in D_m \cup D_n} z_d x_{fv}^d p_v(\lambda_d) \\ + \left(\Delta^d - t + a_s^d\right) \sum_{d \in D_o \setminus D_m} \hat{x}_{fv}^d p_v(\lambda_d) \end{aligned} \quad (1)$$

2) Minimize bandwidth cost on links from source to destination passing through VNFs

$$U_l(t) = \left(\Delta^d - t + a_s^d\right) \sum_{d \in D_m \cup D_n} p_b z_d \sum_{u,v \in V} b^d y_{uv}^d + \left(\Delta^d - t + a_s^d\right) \sum_{d \in D_o \setminus D_m} p_b \sum_{u,v \in V} b^d \hat{y}_{uv}^d \quad (2)$$

3) Maximize the number of accepted demands

$$U_{accept} = \sum\nolimits_{d \in D_n} z^d \quad (3)$$

4) Minimize the penalty cost of VNF migration

$$U_{penalty} = \sum\nolimits_{d \in D_m; u,v \in V; f \in F} x_{fu}^d \hat{x}_{fv}^d z^d M(.) \quad (4)$$

$M(.)$ is a function of $dist_{uv}$, where $dist_{uv}$ characterizes the length of the shortest path from u to v. It indicates the cost to migrate a function from node u to node v. In our simulation, we use two different cost functions to consider the migration penalty: a linear function and a square root function, discussed later in Section-V-C.

Constraints:

In order to guarantee that a node capacity is not oversubscribed by the allocation of VNFs to this node, we have:

$$\sum_{d \in \{D_m \cup D_n\}} z^d x_{fv}^d \mu_v^r + \sum_{d \in D_o} \hat{x}_{fv}^d \mu_v^r \leq c_v^r \quad \forall v \in V \quad (5)$$

Similarly for the bandwidth capacity of a link.

$$\sum_{d \in \{D_m \cup D_n\}} z^d y_{uv}^d b^d + \sum_{d \in D_o} \hat{y}_{uv}^d b^d \leq w_{uv} \quad \forall u, v \in V \quad (6)$$

Constraint (7) guarantees that each VNF in a demand has to be processed in a given preference order.

$$x_{f_1 u}^d x_{f_2 v}^d - \chi_{uv}^d \leq 0 \qquad \phi_{f_1}^d < \phi_{f_2}^d \quad (7)$$

Each VNF only appears exactly once in a demand. This constraint is expressed as follows:

$$\sum\nolimits_{v \in V} x_{fv}^d = \begin{cases} z^d & \text{if } f \in F^d \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Constraint (9) ensures that the variables y_{uv}^d are non-symmetric.

$$y_{uv}^d + y_{vu}^d \leq 1 \qquad \forall u, v \in V, d \in D_m \cup D_n \quad (9)$$

Equations (10) enforce that, for each computer path associated to a demand, it always begins at its source and terminates at its destination.

$$\sum\nolimits_{v \in V} y_{uv}^d - \sum\nolimits_{v \in V} y_{vu}^d = \begin{cases} z^d & \text{if } u = s^d \\ -z^d & \text{if } u = t^d \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

In addition, we need constraints (11), (12) and (13) to guarantee the compatibility between variables as follows:

$$\chi_{uv}^d - \sum_{i \in V, i \neq v} \chi_{ui}^d y_{iv}^d \geq 0 \qquad \forall d \in D_m \cup D_n \quad (11)$$

$$x_{fv}^d \sum_{i \in V} y_{iv}^d = x_{fv}^d \quad \forall d \in D_m \cup D_n, v \in V \setminus \{s^d, t^d\} \quad (12)$$

$$(\chi_{uv}^d - 1) y_{uv}^d = 0 \qquad \forall d \in D_m \cup D_n \quad (13)$$

Now, for the OFFLINE case, we simply set $D_o = \emptyset$.

Unfortunately, the joint problem of NFV placement and routing is harder than the integral multicommodity flow problem whose decision version is NP-complete. Indeed, our problem becomes the integral multicommodity flow problem when we consider each demand and assume that we know the location of required functions. Hence, we study heuristic that provides a solution close to the optimal with a reduced computation time. In the next section we introduce our practical solution to address this challenge.

IV. Proposed Algorithm

As our problem is NP-hard, we only apply exact algorithms to small size dimension of our system that runs within an acceptable time. So, we use the GUROBI Optimizer [18] to solve the quadratic model in order to obtain the optimal solution. In addition, we designed three heuristic algorithms to cope with large size dimensions ((Max-Min and Min-Min for the OFFLINE case, RBP for the ONLINE case). In Section-V, we compare these algorithms with the one produced by a Random Algorithm (OFFLINE case) and the PT algorithm (ONLINE case). Hence, we discuss a migration technique aiming at efficiently managing resources and balancing flows in a dynamic environment.

The three heuristics are introduced below.

A. OFFLINE case

Our heuristics (Max-Min and Min-Min algorithm) are composed of two phases: the node mapping phase and the link mapping phase. In the first phase, we rank nodes in the virtual network to order nodes able to host a VNF for a customer demand. The link mapping phase uses the result of the node mapping phase to find a path with minimum cost for each customer demand.

1) The node mapping phase: We introduce a polynomial-time heuristic finding an allocation of VNFs available on the virtual network to match customer demands. The main idea is to assign a virtual node, based on its rank in the virtual network, to a VNF. We propose two algorithms, namely the Max-Min algorithm and the Min-Min algorithm. The Max-Min algorithm allocates a "costly" function (i.e., a function whose resource requirement is large) to a cheap node that is a node whose rank is low. The Min-Min algorithm allocates a "cheap" function (i.e., a function whose resource requirement is small) to a cheap node.

We use the equation (14) of the rank for node v to find the cheapest or the best virtual node. This value is always updated after deploying a VNF on a virtual node. It accounts for the available resource capacity, the price for each resource unit and the ability to connect to other nodes. It also considers the benefit obtained by the situation where a given function is already hosted on one node and could be mutualized with another demand. We propose a formula for ranking the nodes based on the components as below:

$$rank_v = \alpha \left(\frac{p_v}{\sum_{i \in V} p_i} + \frac{1}{\sum_{f \in F} C(f,v) + 1} \right) + \beta \left(\frac{\sum_{i,j \in V} w_{ij} - \overline{y}}{\sum_{i \in V} w_{iv} - \overline{x_v}} + \frac{\sum_{d \in D_n} T(s^d, v) + \sum_{d \in D_n} T(t^d, v)}{\sum_{d \in D, i \in V} T(s^d, i) + \sum_{d \in D_n, i \in V} T(t^d, i)} \right) \quad (14)$$

where $\alpha, \beta \in [0,1]$ $(\alpha + \beta = 1)$ characterize the influence of the resource cost on nodes and the communication cost on links, respectively. In particular, these two parameters will determine the priority of our system according to the price of resources associated with nodes and communications in order to reach source and destination of customer demands. We will estimate thoroughly the impact of these parameters in Section-V. $\overline{x_v}, \overline{y}$ are the amount of bandwidth decreased for node v and other nodes after a function is selected to be hosted on node v. $T(u,v)$ is the minimum distance between two nodes $u, v \in V$. $C(f,v)$ specifies the number of occurrences of function $f \in F$ in all demands that were assigned to node $v \in V$.

Algorithm 1 Max-Min Resource Heuristic

function MAX-MIN($P(f)$)
  ▷ $P(f)$= number of occurrences of f in all demands
  $\overline{x_v} = 0\ \forall v$, $\overline{y} = 0$
  while $F \neq \emptyset$ **do**
    $f_{max} \leftarrow$ arg max$_{f \in F}\{\mu_f^r\}$
    while $P(f_{max}) > 0$ **do**
      $v_{min} \leftarrow$ arg min$_{v \in V \text{ and } v \text{ satisfies (5)}}\{rank_v\}$
      $g(f_{max}) \leftarrow g(f_{max}) \cup v_{min}$
      max_{bw} = max $\{d \in D_n | f_{max} \in F^d\}$
      $xo_{v_{min}} \leftarrow \overline{x_{v_{min}}} + max_{bw}$
      $\overline{y} \leftarrow \overline{y} + max_{bw}$
      $P(f_{max}) \leftarrow P(f_{max}) - 1$
      Update $rank_v, c^r_{v_{min}}$
    end while
    Remove f_{max}
  end while
  Compute $U_n(t)$ using equation (1).
  return g.
end function

The Pseudo-code of the Max-Min algorithm is provided in Algorithm 1. The Max-Min algorithm first selects a function $f \in F$ whose resource requirement is the highest, denoted by f_{max}. It then finds a virtual node with the minimum rank in the virtual network, denoted by v_{min}. Virtual node v_{min} is selected to host function f_{max} for a customer demand. The available resource capacity of virtual node v_{min} is updated. Finally, the function f_{max} is removed from F when f_{max} required in all customer demands is satisfied. We repeat the process until all functions are allocated.

The Min-Min algorithm is similar to the Max-Min algorithm where, f_{max} in the Max-Min algorithm is replaced by f_{min} that is a function whose resource requirement is the lowest.

2) The link mapping phase: The second phase of the heuristic is to find a path from a source to a destination with the minimum cost for each demand. Its Pseudo-code is presented in Algorithm 2. Specifically, using the result of phase 1, for each $f \in F$, we have a set of virtual nodes $g(f)$ that can support function f. We denote the ith function of the customer demand d by $F^d(i)$. For each customer demand, the algorithm first finds the shortest path between the source and a node providing $F^d(0)$. Then, for adjacent functions $(F^d(i), F^d(i+1))$, it finds the shortest path from a set of nodes $g(F^d(i))$ to a set of nodes $g(F^d(i+1))$. Finally, it finds the shortest path between the destination of the demand and a node that can provide the last function in the demand (i.e., a node in $g(F^d(\ell_d - 1))$ where ℓ_d is the number of functions for demand d).

Algorithm 2 Path Selection

function PATHSELECTION($g : F \mapsto V$)
  $\{p, p_{sol}\} \leftarrow \{\emptyset, \emptyset\}$
  for demand $d \in D_n$ **do**
    $F^d(i)$: the i^{th} VNF in demand d, $i = 0..\ell_d - 1$
    $F^d(-1) \leftarrow s^d, F^d(l_d) \leftarrow t^d$
    for a pair $F^d(i), F^d(i+1), i = -1..\ell_d - 1$ **do**
      $U_1 \leftarrow g(F^d(i)), U_2 \leftarrow g(F^d(i+1))$
      $p_i \leftarrow$ shortest_path($u_1 \in U_1, u_2 \in U_2$), satisfying (6)
      $g(F^d(i+1)) = u_2$
      $p \leftarrow p \cup p_i$
    end for
    Add p to p_{sol}
  end for
  Compute $U_l(t)$ using equation (2).
  return p_{sol}
end function

B. ONLINE case

1) Routing before Placement (RBP) algorithm: We now present an algorithm solving the ONLINE case.

In contrast to the OFFLINE case, the RBP algorithm finds the routing path prior to locating VNFs on virtual nodes. RBP will consider each demand sequentially and decide whether and how to serve it. For each demand d, we find in graph G, a shortest path p from s^d to t^d. Then we allocate one by one VNF in F^d on the path p, satisfying all resource constraints (5)→(13). If this path does not satisfy all resource constraints, we try another shortest path p' from s^d to t^d, and so on. If no

TABLE II: Scenarios

Scenario	Number of virtual nodes n	Number of VNFs m	Number of new demands size (D_n) $min_l \rightarrow max_l$	Number of old demands size (D_o)
R1	7	4	$1 \rightarrow 1$	1
R2	15	5	$3 \rightarrow 20$	5
R3	50	15	$10 \rightarrow 100$	20
R4	200	20	$50 \rightarrow 600$	30
B1	n=2,k=2	10	$2 \rightarrow 100$	10
B2	n=4,k=2	10	$2 \rightarrow 100$	10
F1	k=4	10	$2 \rightarrow 100$	10
F2	k=6	10	$2 \rightarrow 100$	10

such path exists, the demand d is rejected. The Pseudo-code of the RBP algorithm is provided in Algorithm 3.

2) Migration Technique: Another concern in the dynamic NFV problem is to decide if and how a set of old demands should be migrated. Selecting old demands to migrate will impact the current resources of our system as well as affect the system cost. Our goal is to move a flow of old demands including nodes and links to a new position to optimize the system resources. As mentioned earlier, there exist a penalty cost associated to the migration tasks (4). In this section, we propose a simple method to select a set of old demands to be migrated. The main idea of the method is to identify old demands that have the largest wasted bandwidth cost for passing through nodes that do not provide resources for any required VNFs. It results in the computation of the wasted bandwidth cost of each old demand, as follows:

$$\Gamma_d = \sum_{f_1 * f_2 = 0; v_1, v_2 \in V} \hat{x}^d_{f_1 v_1} \hat{x}^d_{f_2 v_2} \hat{y}^d_{v_1 v_2} w_{v_1 v_2} \qquad (15)$$

Then we select η % number of old demands D_o depending on the wasted resource as in (15), called D_m and re-configure this set of demands. We release all resources provisioned in D_m and find the solution for D_m using Algorithm 3. Finally, we continue this algorithm until the set of all new demands D_n has been processed.

Algorithm 3 Routing before Placement(RBP) Algorithm

```
1: function RBP(F, D_m, D_o, D_n, G(V, E, w))
2:     AcceptNo = 0;
3:     for demand d ∈ {D_m ∪ D_n} do
4:         {v_subsol, f_subsol, v_sol, f_sol} ← {∅, ∅, ∅, ∅}
5:         reject=true;
6:         while true do
7:             p ← Shortest_path(G, s^d, t^d)
8:             for f ∈ F^d, v ∈ p do
9:                 Locate f to v
10:                if satisfy constraints (5) then
11:                    v_subsol ← v_subsol ∪ v
12:                    f_subsol ← f_subsol ∪ f
13:                    Update c_v^r
14:                    reject = false; break;
15:                else
16:                    Mark p in G; continue;
17:                end if
18:            end for
19:        end while
20:        if reject== false then
21:            Add v_subsol to v_sol; f_subsol to f_sol
22:            AcceptNo ← AcceptNo +1;
23:        end if
24:    end for
25: end function
```

V. Evaluation

In this section, we evaluate the performance of our solutions according to a large set of parameters.

A. Parameter Setting

We defined the following scenario in order to evaluate the performance of our algorithms:

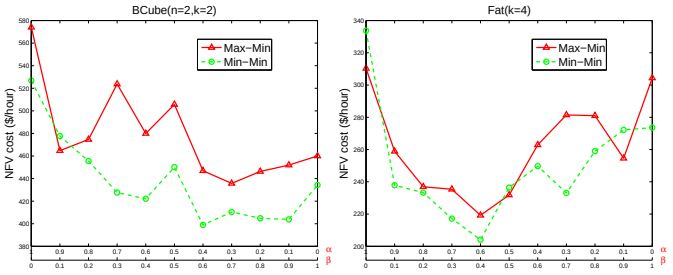
- Network Topology: In this paper, we use two types of topologies: random networks $(R1, R2, R3, R4)$ and data center networks $(B1, B2, F1, F2)$ as defined in Table II. We focused on two types of data center topologies, namely, Fat-Tree [19] and BCube [20]. For Fat-Tree networks, we generated two topologies $(F1, F2)$ with parameters $k = 4$ and $k = 6$, where k represents the number of ports in a switch. For BCube networks, we generated two topologies $(B1, B2)$ with parameters $n = 2, k = 2$ and $n = 4, k = 2$, where n represents the number of ports in a switch and k stands for the number of hierarchy of switches.
- Customer demands: We evaluated each network topology, called a scenario, using eight different tests, varying the number of demands l in a test from min_l in the first test to max_l in the last one. Each customer demand including the source node, the destination node, a chain of VNFs, the resource requirement and the cost factor of SLA, is generated randomly for each individual test.
- The price of resources (i.e. CPU, storage, memory) are collected from the Microsoft [21] and Amazon websites [22]. For example, the price of a virtual machine with 4 CPUs, 285GB storage and 7GB memory on the Microsoft cloud is 0.36 $\$/hour$.

We have simulated our proposed algorithms in Java.

- In the OFFLINE case, we evaluate the performance (**the NFV cost**) of our heuristics (Max-Min and Min-Min) with the optimal solution, and a Random algorithm (to be described later).
- In the ONLINE case, we evaluate the performance (**the average routing path length and the acceptance ratio**) of our heuristic (RBP algorithm) against the PT algorithm. We also compare the gain obtained when applying our migration strategy as a function of the percentage of old demands considered for migration.

B. OFFLINE case

We first assess the influence of the parameters α, β in equation (14). We consider different values of α, β in $[0, 1]$ and then compare the NFV cost produced by our two heuristics. Fig.1



(a) B1 (BCube, $n = 2, k = 2$) (b) F1 (Fat Tree, $k = 4$)

Fig. 1: (OFFLINE) The NFV cost with different parameters (α, β) on a data center network

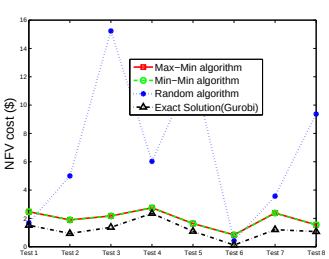


Fig. 2: (OFFLINE) The NFV cost of small scale scenarios

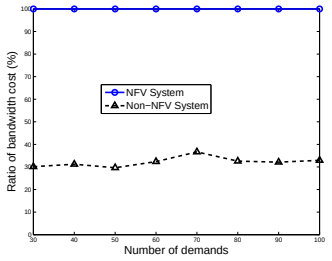
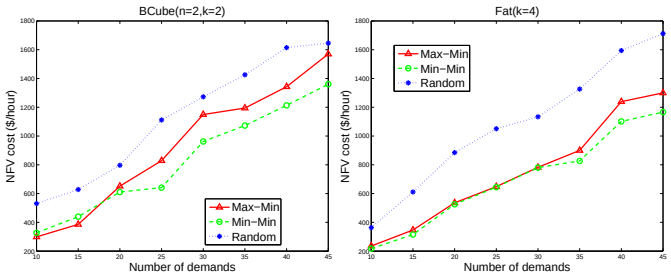


Fig. 3: (OFFLINE) Ratio between the bandwidth cost of a non-NFV system and that of a NFV system

shows the influence of the NFV cost on data center networks when selecting different value of parameters (α, β). The NFV cost is computed by adding the resource cost $U_n(t)$ as defined in equation (1) to the bandwidth cost $U_l(t)$ as in equation (2). We observe that the best values for both algorithms (Max-Min, Min-Min) depend on the network topology. For instance, the best values for a BCube topology are $\alpha < \beta$, whereas for a Fat Tree topology they are $\alpha > \beta$. When $\alpha < \beta$, the priority of nodes will mainly depend on the communication cost on links whilst if $\alpha > \beta$, the priority of nodes will mostly depend on the resource cost on nodes. In fact, a BCube topology has more links connecting switches and servers than a Fat Tree topology. Therefore, in order to select a node that can host VNFs in a BCube topology, it is more important to consider the communication cost. In the reminder of the paper, we use $(\alpha = 0.6,\ \beta = 0.4)$ with Fat Tree topologies, and $(\alpha = 0.4,\ \beta = 0.6)$ with BCube topologies for the two heuristics.

Then, we use the GUROBI Optimizer [18] to find the optimal solution for small instances where $(m = 4, l = 1)$, namely scenario 1 and introduce a simple Random algorithm to evaluate the efficiency of our solutions for larger dimensions. The Random algorithm is defined below. We first generate randomly locations for hosting all VNFs supporting all demands and satisfying constraints (5). We then find a path satisfying constraints (6) for each demand. Each path will be composed of nodes hosting the required VNFs in a given order. The obtained result constitutes a Random solution to our problem.

Fig. 2 shows the NFV cost obtained by Max-Min, Min-Min, Random algorithms and the optimal solution (GUROBI optimizer [18]) for a small size system. All tests (x-axis) in the small size scenario consider the same demand for different network topologies that are generated randomly with the number of vertices equals to 7. It is obvious that the solutions produced by the heuristic algorithms (Max-Min and Min-Min) are far better than the one provided by the Random algorithm and close to the optimal solution. For larger scenarios $(R2, R3, R4, B1, B2, F1, F2)$ in Table II (i.e. hundreds of virtual nodes, hundreds of demands), we run our heuristic algorithms and Random algorithm where the customer demand is generated randomly. Regarding the NFV cost (Fig. 4) on data center topologies $(B1$ and $F1)$, our heuristics (green line and red line) performs obviously better than Random algorithm (blue line) and become more effective when the number of demands grows. In addition, Min-Min algorithm (green line) is better than Max-Min algorithm (red line). As expected, allocating a "cheap" function (i.e. a function whose resource requirement is small) to a cheap node provides a more efficient resource management.



(a) B1 (BCube, $n = 2, k = 2$) (b) F1 (Fat Tree, $k = 4$)

Fig. 4: (OFFLINE) The NFV cost of large scale scenarios

We now consider a non-NFV system where a source node is able to host the all set of required functions. We simply use Dijkstra algorithm to find a shortest path for all pairs of demand from source to destination. We then sum the costs for all links on all the shortest paths to obtain a bandwidth cost. We use this solution to compare it against our NFV-based one and assess the overhead due to virtualizing the network functions. In Fig. 3, we compare the bandwidth required to transfer packets from a source to a destination in a NFV system and that in a non-NFV system. We use the bandwidth cost computed by equation (2). The result shows that a trade-off has to be found between efficiency and the agility provided by the virtualization of functions. It means that, in our scenario, an operator need to spend 60% more resources to be able to benefit from the agility and dynamicity of NFV. This confirms the importance for instrumenting an efficient deployment of service chaining.

We now want to evaluate the distribution of VNFs on a multi-tier network topology as illustrated in Fig. 5. We define three node categories: core nodes, aggregation nodes and access nodes. An access node is connected to an aggregation node, itself connected to core nodes, and core nodes are fully meshed. We assume that all nodes are NFVI nodes that can host VNFs. We compute the solution for the two heuristics

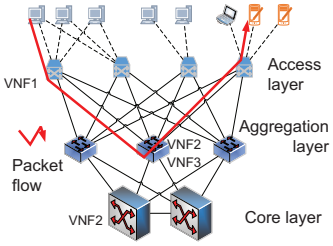


Fig. 5: Multi-tier network topology

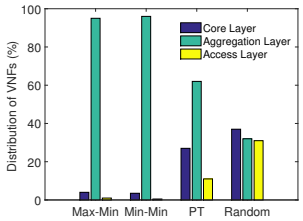


Fig. 6: Distribution of VNFs on a multi-tier network

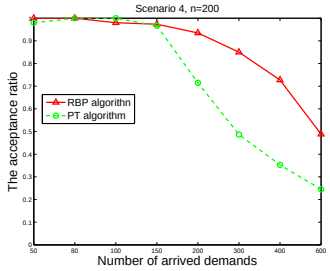


Fig. 7: (ONLINE) Comparison between the acceptance ratio of RBP and that of PT

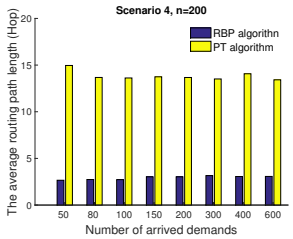


Fig. 8: (ONLINE) Comparison between the average routing path length of RBP and that of PT

and the PT algorithm in order to compare the distribution of VNFs among those algorithms. Fig. 6 shows that for the Max-Min and Min-Min algorithms, VNFs are almost similarly distributed on aggregation nodes (95%), a small part on core nodes (4%), and rarely (1%) on access nodes, while the PT algorithm distributes more equally VNFs to three node types including core nodes (21%), access nodes (17%), and aggregation nodes (62%). Finally and as expected, the Random algorithm distributes fairly VNFs to the three node types. It clearly suggests that the network functions should be located on the network edge rather than the network core. The best option will be to deploy all VNFs in every edge node but the cost might be excessive hence the importance to identify which VNFs can be mutualized and/or deployed only on some specific edge nodes whilst preserving performance constraints.

C. ONLINE case

We now evaluate our solution suited to the ONLINE case.

We acknowledge the fact that there exist an abundant literature related to the NFV placement such as [8], [9], [11]. However, we claim that our (RBT) model is comprehensive and it is therefore difficult to compare it against the existing ones. For instance, as mentioned earlier in Section II, the model in [9] does not consider the request or the customer demand as a flow from a source to a destination. Moreover, they evaluated their approaches with at most two chained network functions for each SFC request. Therefore, we decided to select the PT algorithm [1] as a comparison as we found that this is the one that share the largest set of characteristics with ours. Fig. 7 and Fig. 8 show the acceptance ratio and the average routing path length obtained by our heuristic (RBP) compared to the PT one. It shows that our solution achieve

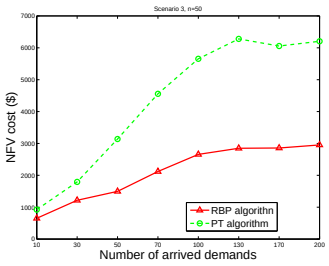


Fig. 9: (ONLINE) Comparison between the NFV cost of RBP and that of PT when no migrating

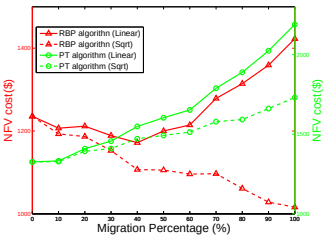


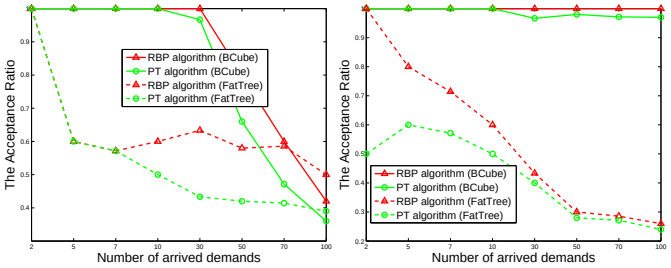
Fig. 10: (ONLINE) Different percentages of migration with two penalty cost functions

higher acceptance ratio and shorter routing paths than PT. Fig. 9 showing the NFV cost as defined by (1) and (2) provide evidence of the efficiency of our solution when no migration exists. The idea of RBP is to implement routing before placing the VNFs. This guarantees that demands always use good paths. Indeed, in Fig 3, 8, and 9, the solution obtained by RBP (red line) is obviously better than the one produced by the PT algorithm (green line). However, it has to pay the price for increasing the running time when trying all possible paths for each demand.

In addition, we also compare the acceptance ratio between RBP and PT applied on some data center network topologies $(B1, B2, F1, F2)$. Fig.11 shows the acceptance ratio as a function of the number of arriving demands, from 2 to 100. Fig. 11(a) depicts the results with a network of 32 servers (respectively BCube $B1$, FatTree $F1$) and Fig. 11(b) shows the results with a network of 128 servers (respectively BCube $B2$, FatTree $F2$). We observe that our heuristic performs better on a Bcube topology than on Fat-Tree topology. Indeed, a BCube topology provides much more links between server nodes and switch nodes than the Fat-Tree topology.

Finally, we evaluate the impact of migrating a set of old demands. We address the NFV cost using various migration percentages η from 0% (no migration) to 100% (all demands are migrated). As mentioned earlier, we select old demands eligible for migration according to the wasted bandwidth, defined in equation (15), of these demands. We also consider as a parameter, two cases for the migration penalty cost function $M(.)$ in equation (4): one linear function and one square root function.

Fig. 10 provides a comparison of the NFV cost, as a function of the migration percentage, obtained by the RBP and PT algorithms. The NFV cost is computed as the sum of the resource cost $U_n(l)$ defined by equation (1), the bandwidth cost $U_l(t)$ equation (2) and the migration penalty cost $U_{Penalty}$ (4). We observe that RBP will benefit from migrating demands whilst it is not the case for PT. The best cost is obtained for RBP with 40% migration of old demands when using a linear penalty cost. This cost will always decrease as a function of the percentage of flows being migrated for a square root function penalty cost. In summary, the migration will bring a certain benefit when we select a proper migration percentage (as 40% in our experiment with a linear penalty cost). However, if we



(a) Number of servers = 32 (b) Number of servers = 128

Fig. 11: (ONLINE) Comparison between the acceptance ratio of RBP and that of PT on a data center network

only pay a smaller penalty cost for migrating (as square root function), we obtain more benefit when updating the system resource after an interval τ. It means we have to consider both the penalty cost to move a function to a new position and the distribution of system resources at present, when determining if and how a set of old demands should be migrated.

VI. Conclusion and Future Work

In this paper, we addressed the resource allocation problem for NFV where a network service is composed of several VNF instances. Our model is generic, able to handle a large and diverse set of key parameters and can easily be customized. It takes into account the interconnectivity among VNFs, service requirements, and NFV costs (including both resource cost and traffic cost). We formulated the problem as a quadratic program for both optimal VNF location and optimal routing. We proposed two heuristic algorithms, called Max-Min and Min-Min, for the OFFLINE case and one heuristic, called RBP for the ONLINE. Our extensive evaluation shows that our heuristic algorithms always perform significantly better than the one produced by the PT solution in [1] considering various performance metrics such as the NFV cost and the acceptance ratio of arriving demands. In addition, we computed the cost uncured by a network relying on NFV to capture the overhead introduced by a solution based on the virtualization of functions. To complete our study, we investigated a simple migration technique that can efficiently manage the dynamic situation produced by a continuous flow of arriving demands. Our future plan is to exploit the value of this model in practical scenarios where measurement will be made available to parametrize our solution. In addition, we started to study the congestion issue that can arise in such an environment.

Acknowledgment

We are grateful to the anonymous reviewers for their constructive criticism. Tuan-Minh Pham was partially supported by project B2016-SPH-17 from the Vietnam Ministry of Education and Training.

References

[1] M. F. Bari, S. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *11th Int. Conf. Netw. Service Manag. (CNSM)*, Nov. 2015, pp. 50–56.

[2] ETSI, "White paper on network functions virtualization."

[3] M. T. B. H. M. A. Fischer, J. F. Botero and X. Hesselbach, "Virtual network embedding: A survey," in *IEEE Commu. Surveys Tutorials*, pp. 1888–1906.

[4] M. P. G. S. L. Z. G. Md. G.Rabbani, R. P. Esteves and R. Boutaba, "On tackling virtual data center embedding problem," in *2013 IFIP/IEEE Int. Symp. Integrated Netw. Manag.*, pp. 177–184.

[5] X. Li and C. Qian, "A survey of network function placement," in *13th IEEE Annu. Consumer Commu. & Netw. Conf., CCNC 2016, January 9-12*, pp. 948–953.

[6] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *IEEE Conf. Netw. Softwarization (NetSoft), 2015*.

[8] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conf. Computer Commu., INFOCOM, Apr. 26 - May 1*, pp. 1346–1354.

[9] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE Int. Symp. Integrated Netw. Manag. (IM)*, May, pp. 98–106.

[10] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct., pp. 171–177.

[11] T.-W. K., B.-H. L., K. C.-J. L., and M.-J. T., "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. IEEE INFOCOM 2016*.

[12] D. Ihara, F. L. Pires, and B. Barán, "Many-objective virtual machine placement for dynamic environments," in *8th IEEE/ACM Int. Conf. Utility Cloud Comput., UCC 2015, Dec. 7-10*, pp. 75–79.

[13] F. L. Pires and B. Barán, "Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach," in *IEEE/ACM 6th Int. Conf. Utility Cloud Comput., UCC 2013, Dec. 9-12*, pp. 203–210.

[14] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," *CoRR*, vol. abs/1506.04330, 2015.

[15] L. E. Li, V. Liaghat, H. Zhao, M. Hajiaghayi, D. Li, G. T. Wilfong, Y. R. Yang, and C. Guo, "PACE: policy-aware application cloud embedding," in *Proc. IEEE INFOCOM 2013, Apr. 14-19*, pp. 638–646.

[16] J. Elias, F. Martignon, S. Paris, and J. Wang, "Efficient orchestration mechanisms for congestion mitigation in nfv: Models and algorithms," *IEEE Trans. Services Comput.*, vol. PP, no. 99, pp. 1–1, 2015.

[17] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on afliation-aware vnf placement," in *Proc. IEEE GLOBECOM*, 2016.

[18] http://www.gurobi.com/.

[19] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM 2008 Conf. Applications, Technol., Architectures, Protocols for Computer Commu., Aug. 17-22*, pp. 63–74.

[20] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM 2009 Conf. Applications, Technol., Architectures, Protocols for Computer Commu., Aug. 16-21*, pp. 63–74.

[21] https://azure.microsoft.com/en-us/pricing/calculator/.

[22] https://aws.amazon.com/ec2/pricing/.