Near Optimal Placement of Virtual Network Functions

Rami Cohen
Empow
ramic@empownetworks.com

Liane Lewin-Eytan
Yahoo Labs, Haifa, Israel
liane@yahoo-inc.com

Joseph (Seffi) Naor
Computer Science Dept., Technion
naor@cs.technion.ac.il

Danny Raz
Bell Labs Israel and Technion
danny@cs.technion.ac.il

***Abstract*—Network Function Virtualization (NFV) is a new networking paradigm where network functions are executed on commodity servers located in small cloud nodes distributed across the network, and where software defined mechanisms are used to control the network flows. This paradigm is a major turning point in the evolution of networking, as it introduces high expectations for enhanced economical network services, as well as major technical challenges. In this paper, we address one of the main technical challenges in this domain: the actual placement of the virtual functions within the physical network. This placement has a critical impact on the performance of the network, as well as on its reliability and operation cost. We perform a thorough study of the NFV location problem, show that it introduces a new type of optimization problems, and provide near optimal approximation algorithms guaranteeing a placement with theoretically proven performance. The performance of the solution is evaluated with respect to two measures: the distance cost between the clients and the virtual functions by which they are served, as well as the setup costs of these functions. We provide bi-criteria solutions reaching constant approximation factors with respect to the overall performance, and adhering to the capacity constraints of the networking infrastructure by a constant factor as well. Finally, using extensive simulations, we show that the proposed algorithms perform well in many realistic scenarios.**

I. Introduction

In recent years we are observing several evolution trends that have a considerable impact on the networking domain. The first is the ongoing transition into all-IP networks (VoIP, LTE, IP DSLAMs); and the second is the introduction of software defined networks (SDN) ([4], [8]) and the separation of the control and data planes. The third critical evolution is the shift towards the cloud and the emergence of new applications that are becoming the main way of communication for more and more people (e.g., chat applications, replacing traditional voice calls and email). The most recent development is the introduction of network function virtualization (NFV) ([11], [12]), an architecture in which network functions are executed over commodity servers rather than on dedicated servers.

The combination of these developments creates a major shift in network design, which is being recognized both by manufactures ([15], [16], [17]) and operators ([14]). The new NFV architecture, which is currently still in the making, will be based on executing network functions on commodity servers located in small cloud nodes that are distributed across the network, as well as on the use of software defined mechanisms for controlling the network flows.

Partly Supported by ISF grant 954/11 and BSF grant 2010426.

The new networking paradigm, sometimes referred to as "distributed cloud networking", introduces, in addition to the high expectations for better and more economical network services, also major technical challenges. In this paper we address one of the main technical challenges in this domain: the actual placement of virtual functions within the physical network. This placement has a critical impact on the performance of the network, its operational cost, and on network reliability.

We adopt the terminology used in the NFV forum documentation (see original white paper [11]), where a service is composed of one or more network functions. The execution of each of the functions requires resources (computing, memory, storage, etc.) in one of the distributed server locations. In some cases, the service is associated with a specific network node and the network functions must be provided at this node. This is the case, for example, for IMS services where the SIP functionality must be performed at the edge of the operators network. However, for some services such as DPI (Deep Packet Inspection), the exact network location is not critical as long as the flow is directed into a location where the service can be performed. Moreover, there is no need to execute the entire function on a single server. Several virtual machines (VMs) can be usedł while the distribution and the routing of traffic between them is performed using the SDN control mechanism. The problem is then to allocate server resources to network functions at various locations, such that all flows requiring a service are satisfied, and the operational cost is minimized.

While this seems to be an appropriate model, in order to study it rigorously one must define the exact metrics used to determine operational costs. We follow the well-known paradigm of facility location problems ([10]) and use two cost measures: (1) the distance from clients to locations where services are provided; and (2) the setup cost of services. The overall cost is defined as the sum of the setup costs, reflecting the cost of having VMs that execute a function, and the cost of diverting traffic into the location of these servers. Note that when the network is represented by a graph with a distance metric between the nodes, the shortest distance from a given path in that graph to a node is a metric as well. Thus (in analogy with facility location terminology), we use the term "client" to refer to a path of a relevant flow. Clearly, in our simulations we measure the actual network distance between a flow path and the locations where functions are executed.

Our goal here is to provide a profound understanding of the issues related to the location of network functions within the underlying network, while considering global network measures and constraints. To this end we define and study

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

a general model that captures the most important aspects of network function virtualization, and then evaluate the advantages of using our methodology in various practical use-cases. In general, there is a set of clients that needs to be served by a set of network functions, where each client requires a subset of the functions. Locating the functions is performed in a practical environment in which servers have limited space for allocating functions (in a graph we refer to this constraint as the node size), and our goal is to locate network functions in a way that minimizes the overall network cost, while adhering to the allocation size of the servers. We refer to this problem as the *NFV Location Problem*.

The NFV location problem captures two well known NP-hard optimization problems problems - the *facility location problem* and the *generalized assignment problem* (GAP), both of which received much attention in the last 50 years [10], [2]. The facility location problem asks for opening a set of facilities to serve a set of clients in a graph. There is a distance function defined over the graph and each client pays a *connection cost* equal to the distance from the facility which provides its service. Opening a facility at a node requires paying a *setup cost*. The goal is to minimize the sum of the connection costs and the setup costs. In some variants of the facility location problem there is a *capacity* associated with each facility, bounding the number of clients that can get service. In the case of *soft* capacities more than one copy of a facility can be opened at a node. The second optimization problem is the generalized assignment problem in which we are given a collection of jobs to be assigned to a set of machines. There are size constraints on the machines and each job needs to be assigned on exactly one machine. Each job is associated with a size and a cost, and both are machine dependent. The goal is to find an assignment of jobs to machines of minimum total cost satisfying size constraints.

Coming back to the NFV location problem, suppose that there are no size constraints at the nodes. This means that the functions can be located at the nodes independently of each other, and thus the problem of locating each function becomes an instance of the facility location problem. In contrast, suppose that all distances between clients and nodes are equal. In the case where there are no capacity constraints on the functions there is no need to open a function in more than one location. Thus, we get an instance of the generalized assignment problem in this case. To summarize, even the simplest version of the NFV location problem, where there are no capacity constraints on the functions, combines two classic NP-hard problems.

We provide an in-depth theoretical analysis of the NFV location problem. We start with the uniform demand ,case where all clients have the same demand (unit demand without loss of generality). First, we consider the basic version where the functions are uncapacitated, i.e., a function at a node can serve any number of clients. Then, we consider the capacitated version. However, To accommodate more clients, several copies of a function can be located at the nodes. We assume that the capacity of a function is node independent. Finally, we consider the most general version where demands can vary between clients. All of our algorithms provide bicriteria approximation factors. In all the different versions of the problem we approximate the objective function by a constant factor, while violating size constraints by a constant factor. The specific constants vary between the versions.

The paper is organized as follows. Section II discusses related work. The model is formally defined in Section III. Sections IV & V contain our approximation algorithms for both the uncapacitated and capacitated versions of the NFV location problem. Section V-A considers the most general case in which clients have non-uniform service demands. The simulation study is presented in Section VI. Finally, conclusions are discussed in Section VII.

II. Related Work

To the best of our knowledge, the NFV location problem presented in this paper has not been investigated before. It is the combination of two well known optimization problems: GAP (further elaborated on in Section III-B) and facility location. The facility location problem has been widely investigated over the years, and many variants of it were studied. The most basic variant of the problem is the uncapacitated case (a facility can serve any number of clients) for which a constant factor approximation is known (see e.g. [10]). The capacitated version of this problem was investigated in [1], where a constant approximation bicriteria result was given for the case where there are hard non-uniform bounds on the amount of demand that can be routed to any facility.

Another variant of this problem, closer to our setting and called the *multicommodity facility location*, was studied in [9]. This model considers a natural extension of facility location where there is a finite set of k commodities (corresponding to functions in our terminology), and each client requires a subset of those commodities. The location cost is then composed of a fixed setup cost which is node dependent, and an incremental cost which is commodity dependent. There are no allocation (size) constraints at the nodes. The work in [9] presents an $O(\log k)$ approximation algorithm for this problem.

There are very few works considering virtual functions in the context of placement and multi-function services. A work dealing with a multi-function service system is presented in [7] for a totally different environment, considering police-ambulance systems in an urban area. In the context of NFV and SDN, [3] considers a single service (Deep Packet Inspection) for which a placement heuristic is presented. A demonstration of the placement of virtualized mobile core network functions is presented in [6], where a network provider is required to provide optimal cellular coverage in case of large "mega" events. While not being close to our model, these works relate to our general motivation. We are not aware of any work modeling the NFV placement challenge as an optimization problem, as well as providing proven theoretical results.

III. Model and Preliminaries

A. General model

We are given an undirected graph (or network) $G = (V, E)$, equipped with a distance function $d(\cdot, \cdot)$ between any pair of nodes inducing a metric space. Given is a set of clients $C \subseteq V$. For a client $c \in C$, c indicates both the client and the node where it resides. There is a set of network functions F, $|F| = m$, that need to be located at the nodes of V. For each

client $c \in C$, $f(c)$ denotes the set of functions in which c is interested. We denote by $U \subseteq V$ the set of nodes where the functions of F can be located.

Each node $u \in U$ has total size $w(u)$, and each function $i \in F$ has size w_u^i which is the size it occupies in case it is located in u. The *setup cost* of locating a function i at u is denoted by p_u^i. In general, function i in node u can serve only μ^i clients. (This bound is node independent.) To accommodate more clients, several copies of function i can be located at node u, however, each copy occupies size w_u^i and pays cost p_u^i.

In the NFV location problem we want to compute an allocation of functions in F to nodes in U, as well as an assignment of each client $c \in C$ to the set $f(c)$. The allocation of functions to nodes should satisfy size constraints: the overall size of the (copies of) functions located at each node u does not exceed $w(u)$. The assignment of clients to functions should satisfy: (i) each client c is assigned to all the functions in the set $f(c)$; (ii) each copy of function i does not serve more than μ^i clients. The goal is to compute an allocation and assignment that minimizes the total system cost. This cost comprises of the sum of the setup costs of the functions and the sum of the distances between the clients and the nodes from which they get service.

We formulate the NFV location problem as an integer linear program. We denote by y_u^i the number of copies of function i located at node u, and by x_{cu}^i the variable indicating whether function i is used by client c at node u.

$$\text{Min} \quad \sum_{c\in C}\sum_{i\in f(c)}\sum_{u\in U} x_{cu}^i \cdot d(c,u) + \sum_{u\in U}\sum_{i=1}^{m} y_u^i \cdot p_u^i \qquad \text{(NFV Location-LP)}$$

s.t.

$$\text{for each client } c, \text{ function } i \in f(c): \quad \sum_{u\in U} x_{cu}^i \geq 1, \tag{1}$$

$$\text{for each client } c, \text{ node } u, \text{ function } i: \quad x_{cu}^i \leq y_u^i, \tag{2}$$

$$\text{for each node } u: \sum_{i=1}^{m} y_u^i \cdot w_u^i \leq w(u), \tag{3}$$

$$\text{for each node } u, \text{ function } i: \quad \sum_{c\in C} x_{cu}^i \leq y_u^i \cdot \mu^i, \tag{4}$$

$$\text{for each function } i, \text{ node } u: \quad y_u^i = 0 \quad \text{if } w_u^i > w(u). \tag{5}$$

The NFV demand constraint (1) guarantees that each client c gets service for each function $i \in f(c)$. The setup cost constraint (2) states that if client c is assigned to function i at node u, then function i is indeed located at u. The size constraint (3) states that the sum of the sizes of the functions located at each node cannot exceed the size of the node. The capacity constraint (4) states that each function i at node u cannot serve more clients than its capacity.

Since our problem is NP-hard, we consider the linear relaxation of the above program. This means that the assignment of a client to a function can be fractionally split between several nodes, and also a function can only be fractionally located at a node. The linear relaxation requires an additional constraint (5), stating that a function cannot be fractionally assigned to a node to which it cannot be fully allocated (which is clearly the case for any integral feasible solution).

B. Generalized Assignment Problem

A basic procedure used in our solution is an approximation algorithm for the generalized assignment problem (GAP). In GAP we are given a collection of n jobs to be assigned to ℓ machines. Each job $j = 1, \ldots, n$ is to be assigned to exactly one machine; if it is assigned to machine m then it has size w_{mj} and incurs a cost of p_{mj}. The size of each machine m is w_m. The goal is to find a feasible assignment of minimum total cost. We formulate the fractional version of GAP as a linear program, where a job can be fractionally split between several machine, and denote by λ_{mj} the fraction of job j located at machine m.

$$\text{Min} \quad \sum_{m=1}^{\ell}\sum_{j=1}^{n} \lambda_{mj} p_{mj} \qquad \text{(GAP-LP)} \tag{6}$$

s.t.

$$\text{for each job } j: \quad \sum_{m=1}^{\ell} \lambda_{mj} = 1, \tag{7}$$

$$\text{for machine } m: \quad \sum_{j=1}^{n} w_{mj}\lambda_{mj} \leq w_m, \tag{8}$$

$$\text{for each job } j, \text{ machine } m: \quad \lambda_{mj} = 0 \quad \text{if } w_{mj} > w_m. \tag{9}$$

The first constraint (7) is the job demand constraint, guaranteeing that the sum of the fractions of each job (assigned perhaps to different machines) must be one. The second constraint (8) states that the sum of the fractional sizes of the jobs allocated at each machine cannot exceed the size of the machine. Finally, the last constraint (9) states that a job cannot be fractionally assigned to a machine to which it cannot be fully allocated (which is clearly the case for any integral feasible solution).

For completeness we describe the details of the approximation algorithm for GAP ([10]) here. The approximation algorithm is based on solving (GAP-LP) and then rounding the fractional solution computed into an integral solution. Clearly, an optimal solution to (GAP-LP) is a lower bound on the optimal *integral* solution. The rounded integral solution has the property that it does not exceed the cost of an optimal fractional solution; however, it violates feasibility by allowing each machine m to be allocated to at most twice its size.

The rounding reduces to computing a matching in a bipartite graph derived from the fractional solution. We construct a bipartite graph $\mathcal{B} = (J, S, E')$ where one side consists of *job* nodes J, and the other side of *machine slots* S. The fractional solution λ assigns in total $\sum_{j=1}^{n} \lambda_{mj}$ jobs to machine m. We allocate $k_m = \lceil \sum_{j=1}^{n} \lambda_{mj} \rceil$ "slots" for machine m, and define $S = \{(m,s) : m = 1, \ldots, \ell, s = 1, \ldots, k_m\}$. Note that summing up the number of slots over all machines, we get $|S| \geq |J|$, due to constraint (7).

The set of edges E' of $\mathcal{B}$ is defined as follows. For a machine m, sort the jobs assigned to it (fractionally) in non-increasing order with respect to their size w_{mj}. For ease of presentation, we assume that $w_{m1} \geq w_{m2} \geq w_{mn}$. Now, consider the slot nodes (m, s) of machine m (where $s = 1, \ldots, k_m$) as bins of capacity 1, and consider the values λ_{mj}, $j = 1, \ldots, n$ as pieces of the n jobs to be packed in these bins. With respect to the non-increasing order, we place the pieces into the bin corresponding to slot $(m, 1)$ one by one, until the bin overflows (above 1) as a result of placing piece j. We then pack a fraction of j that equals the remaining space in $(m, 1)$, and pack the remaining fraction of j into the next bin (that is, $(m, 2)$). We proceed with this packing procedure until all pieces are packed into bins. Edge $(j, (m, s)) \in E$ if only if there is a positive value of job j packed into bin (m, s). The cost of edge $(j, (m, s))$ is set to be p_{mj}. It is easy to see that by repeating this procedure for each machine m, and adding the corresponding edges in $\mathcal{B}$, yields a bipartite graph with a fractional matching in which job nodes are completely matched. The cost of the fractional matching is equal to the optimal cost of (GAP-LP).

The rounding is performed by computing a minimum cost integral matching which includes all job nodes in $\mathcal{B}$. By matching theory, such an integral matching is guaranteed to exist. Moreover, it does not have higher cost than the fractional solution (the optimal cost of (GAP-LP)). In [10] it is shown that any integral matching that includes all job nodes of J corresponds to an assignment of jobs to machines that exceeds the size of each machine by at most a factor of 2. This is based on the following observations. Consider a slot (m, s), and focus on the jobs for which a positive value is packed into the bin corresponding to this slot; let $\max(m, s)$ denote the maximum size requirement w_{mj} among these jobs. Then, the total load assigned to machine m by any integral matching in $\mathcal{B}$ is at most $\sum_{s=1}^{k_m} \max(m, s)$. Due to constraint (8), it holds that $\max(m, 1) \leq w(m)$, and it can be shown that $\sum_{s=2}^{k_m} \max(m, s) \leq w(m)$. Thus, the total load on machine m is at most $2w(m)$.

Algorithm 1 The GAP Rounding Algorithm

1: Solve (GAP-LP), and build a bipartite graph $\mathcal{B} = (J, S, E')$ as follows:
- Allocate for each machine $k_m = \lceil \sum_{j=1}^{n} \lambda_{mj} \rceil$ slots, and define S as the set of all slots (over all machines).
- For each machine m, pack the job pieces λ_{mj} according to the non-increasing ordering of their size w_{mj}. Define an edge $(j, (m, s)) \in E'$ of cost p_{mj} if a positive value of job j is allocated to slot (m, s).

2: Compute an integral complete matching in $\mathcal{B}$ defining the assignment of jobs to machines, where job j is allocated to a single machine m.

IV. The Uncapacitated NFV Location Problem

In the uncapacitated NFV location problem each function i can serve an unlimited number of clients. The formulation of the linear program for the problem is identical to (NFV Location-LP), except that the capacity constraint (4) becomes redundant. We refer to this LP formulation as (UNFV Location-LP). We develop an $(O(1), O(1))$ bicriteria approximation algorithm for the uncapacitated NFV location problem. Specifically, our algorithm achieves a 6-approximation factor, while violating the size constraint of the nodes by at most a factor of 4.

Our algorithm first solves (UNFV Location-LP) to optimality. Let the optimal cost be $C^* = C_d^* + C_s^*$, where C_d^* denotes the sum of the distances between the clients and the nodes from which they get service, and C_s^* denotes the sum of the setup costs. We round the optimal fractional solution in two phases. In the first one, for each function i (separately) we compute a partition of the nodes in U into disjoint sets U_Λ^i, such that each set serves a disjoint set of clients Λ. The set U_Λ^i has the property that clients in Λ are oblivious to the choice of which node from U_Λ^i provides the service to them. In the second phase, the disjoint sets U_Λ^i, for all functions $i \in F$ are used to define an instance of GAP. The second phase applies the rounding algorithm for GAP so as to get an approximate solution to the (UNFV Location-LP) instance.

We begin by describing the first phase which is performed separately for each function $i \in F$. For each client $c \in C$ we can think of a fractional solution to (NFV Location-LP) as inducing a probability distribution over the locations from which c gets its service of function i. Thus, denote by $\bar{d}_c^i$ the expected distance from c to the different locations of function i providing service to it, i.e., $\bar{d}_c^i = \sum_{u \in U} x_{cu}^i \cdot d(c, u)$. Note that $C_d^* = \sum_{c \in C} \sum_{i \in f(c)} \bar{d}_c^i$.

We define a *ball* $B(c, r)$ with center c and radius r as the set of nodes that are within distance r from c. Clearly, at least half the service of function i that c gets is located in a ball $B_c^i = B(c, 2\bar{d}_c^i)$. (E.g., it follows from Markov's inequality.) That is, $\sum_{u \in B(c, 2\bar{d}_c^i)} x_{cu}^i \geq 1/2$. We now sort the clients according to $\bar{d}_c^i$. We pick the ball $B_{c^*}^i$ with the smallest radius and multiply by at most 2 the fractions of function i inside it, so that they add up to 1. That is, we define $\tilde{x}_u^i \leq 2x_{cu}^i \leq 2y_u^i$, such that $\sum_{u \in B_{c^*}} \tilde{x}_u^i = 1$. We remove c^* together with all clients with balls intersecting $B_{c^*}^i$; each such client c is said to be *associated* with $B_{c^*}^i$. Observe that each client c associated with $B_{c^*}^i$ can get (fractionally) its full service of function i from nodes in $B_{c^*}^i$. Also, the distance of c from any node in $B_{c^*}^i$ is at most $2\bar{d}_c^i + 4\bar{d}_{c^*}^i \leq 6\bar{d}_c^i$, since $\bar{d}_c^i \geq \bar{d}_{c^*}^i$ and the ball $B(c, 2\bar{d}_c^i)$ of client c intersects with $B(c^*, 2\bar{d}_{c^*}^i)$.

We continue iteratively, until all clients either define a ball, or are associated with some ball. Denote the balls generated during the iterative process by $B_{c_1}^i, \ldots, B_{c_{n_i}}^i$. Note that the balls are disjoint, as well as the sets of clients associated with each ball. Denote by Λ_c^i the clients associated with ball B_c^i.

Consider the fractional solution contained inside the balls chosen during the iterative process, where the fraction of function i located at node u is $\tilde{x}_u^i$. It follows from the discussion above that this is a feasible fractional solution. The setup cost of this solution is at most $2C_s^*$, since $\tilde{x}_u^i \leq 2y_u^i$. As for distances, for each client c and function $i \in f(c)$, the distance between c and any node in the ball it is associated with (or it defines) is at most $6\bar{d}_c^i$.

We now turn to describe the second phase of our algorithm, where we round the above fractional solution. We define a GAP instance from the balls $B_{c_1}^i, \ldots, B_{c_{n_i}}^i$ (for all functions i) as well as a fractional assignment solution. We round the fractional assignment (as described in Section III-B) into an integral solution, which in turn defines an approximate solution to our NFV location instance.

We first define the GAP instance. The set of nodes cor-

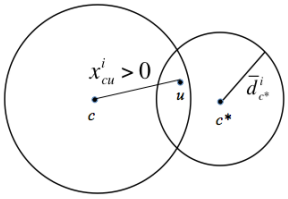

Fig. 1. $B^i_{c^*}$ with minimum $\bar{d}^i_{c^*}$. Client $c \in \Lambda^i_{c^*}$, as B^i_c intersects with $B^i_{c^*}$.

responds to the set of machines, and for each $i \in F$, each ball in $B^i_{c_1}, \ldots, B^i_{c_{n_i}}$ corresponds to a job. For each $i \in F$, $1 \leq j \leq n_i$, the fractional assignment of $B^i_{c_j}$ to each machine $u \in B^i_{c_j}$ is $\tilde{x}^i_u$. Denote by $J = \{1, \ldots, n\}$ the set of jobs (that is, $n = \sum_{i \in F} n_i$), by $M = \{1, \ldots, \ell\}$ the set of machines, and by $\tilde{x}_{mj}$ the fractional assignment of job j to machine m. Note that as a result of Phase 1 there could be several jobs assigned to the same machine u, corresponding to balls of different functions (u belonging to several balls). Recall that balls defined by the same function are disjoint. For a node $u \in U$, the size and cost of the machine corresponding to it in the GAP instance are size are w_u and p_u, respectively. Sizes and costs of jobs allocated to machines are also respectively set to the values w^i_u and p^i_u of allocating function i to node u (for all i and u). The next claim follows from the discussion.

Claim 4.1: The fractional solution defined by $\tilde{x}_{mj}$ is a feasible solution to (GAP-LP), violating machine sizes by at most a factor of 2.

Phase 2 is essentially an application of the rounding algorithm for GAP to the solution $\tilde{x}_{mj}$ (see Section III-B). We thus end up with an integral solution, where each job is assigned to a single machine. The rounding does not increase the total setup cost, but may violate feasibility by allocating each machine to jobs having total size at most twice the total size of the fractions allocated to the machine in $\tilde{x}$. Getting back to our original NFV location problem, take for example a job j corresponding to a ball B^i_c, where j is assigned in the rounded solution to a machine m corresponding to node $u \in B^i_c$. This means that function i is (integrally) located in node u. The clients belonging to Λ^i_c (i.e., associated with B^i_c) get from u their service of function i.

Summarizing our rounding procedure, the resulting approximation factors are as follows. Consider a ball B^i_c: (i) in the first phase, the distance of each client $c' \in \Lambda^i_c$ from any node in the ball B^i_c is at most $6\bar{d}^i_{c'}$; (ii) in the second phase, c' gets its service of function i from a node in B^i_c. Since $C^*_d = \sum_{c \in C} \sum_{i \in f(c)} \bar{d}^i_c$, the final distance cost is at most $6C^*_d$. In the first phase the setup cost increases to at most $2C^*_s$ (due to doubling the fractions within the balls). The setup costs are not increased anymore in the second phase. We thus achieve a total approximation factor of 6 for our objective function. The size constraints at the nodes are violated by a factor of 4, due to doubling the fractions in the first phase, and violating the size of the nodes by a factor of 2 in the second phase.

V. The Capacitated NFV Location Problem

We consider the capacitated version of the NFV location problem, where each copy of function i can provide service

Algorithm 2 The Uncapacitated NFV-Location Algorithm

1: Solve (Uncapacitated NFV Location-LP).
2: For each function i, proceed with Phase 1 as follows:
3: **Phase 1 [steps 3-4].** For each client c, compute $\bar{d}^i_c = \sum_{u \in U} x^i_{cu} \cdot d(c, u)$.
4: While there are still "unserved" clients:
- Pick the client c^* with minimum $\bar{d}^i_c$.
- For each node u within $B(c^*, 2\bar{d}^i_{c^*})$, double the fractions $x^i_{c^*u}$ and define $\tilde{x}^i_u = 2x^i_{c^*u}$.
- Define the set of clients $\Lambda^i_{c^*} = \{c^*\}$, add to $\Lambda^i_{c^*}$ all clients c with balls $B(c, 2\bar{d}^i_c)$ that intersect with $B(c^*, 2\bar{d}^i_{c^*})$, and remove these clients (as well as their fractions).

5: **Phase 2 [steps 5-7].** Define an instance of GAP as follows:
- Define the set of balls created as output from Phase 1 (over all functions) as a set of jobs $J = \{1, \ldots, n\}$; and the set of nodes with allocated fraction within these balls as a set of machines $M = \{1, \ldots, \ell\}$.
- Each job represents a ball, and thus corresponds to a specific function i. Thus, sizes and prices of jobs allocated to machines are set to the values w^i_u (size) and p^i_u (price) of allocating functions to nodes.

6: The fractional solution $\tilde{x}_{mj}$ defines a solution to (GAP-LP).
7: Apply the rounding algorithm for GAP (Algorithm), yielding an integer assignment, where job j is allocated to a single machine m.
8: Assume that in our original NFV location problem, j corresponds to B^i_c, and m corresponds to $u \in B^i_c$. Then, function i is located at u, and clients in Λ^i_c get their function i service from u.

to at most μ^i clients. (Note that this bound is uniform over all nodes.). The fractional version of the problem is formulated in (NFV Location-LP). As in Section IV, our solution is based on rounding an optimal fractional solution, and it has three phases. (We follow the notation of Section IV in this section.)

1) For each function i, independently, disjoints sets of nodes $s^i(1), s^i(2), \ldots$ are constructed. Denote by Λ^i_j the clients getting a fraction of function i from $s^i(j)$. The sets have the following properties:
   - For each set $s^i(j)$, the total sum of fractions of function i located at nodes in the set is at least 1/4. That is, $\sum_{u \in s^i(j)} y^i_u \geq 1/4$.
   - For each client $c \in C$, at least 1/4 of its service of function i is obtained from $s^i(1), s^i(2), \ldots$
   - For each client, reassigning the service it gets between nodes belonging to the same set can only increase the cost distance by a constant factor.
2) For each function i, independently, round the fractional service that clients get from the sets, so that each client gets its full service from a *single* set.
3) Construct a GAP instance, together with a fractional solution, and round it. The rounded solution defines an integral solution to the capacitated NFV location problem.

Phase 1. For each function i we construct the disjoint sets of nodes as follows. The first set $s^i(1)$ comprises of the nodes within ball $B^i_{c_1} = B(c_1, 2\bar{d}^i_{c_1})$, where c_1 is the client with minimum expected distance $\bar{d}^i_c$. We continue inductively. Assume the sets $s^i(1), \ldots, s^i(j)$ are defined. Then, we remove all clients that have already received at least a total service of 1/4 (of function i) from nodes in the sets $s^i(1), \ldots, s^i(j)$. We next pick c_{j+1} to be the client with minimum expected distance

$\bar{d}_c^i$ among the remaining clients. The set $s^i(j+1)$ comprises of nodes within $B^i_{c_{j+1}} = B(c_{j+1}, 2\bar{d}^i_{c_{j+1}})$ that are not included in any of the other sets $s^i(1), \ldots, s^i(j)$, previously defined. Clients *associated* with $B^i_{c_{j+1}}$ are those clients that were not removed before the time $B^i_{c_{j+1}}$ was defined, and that receive a fraction of the service of function i from nodes in $s^i(j+1)$. We denote this set of clients by Λ^i_{j+1}. Overall, for each client c, we denote by α^i_c the total fraction of service received from all the sets defined. The following claims correspond to the properties stated above.

Claim 5.1: For each set $s^i(j)$, the total sum of fractions of function i located at nodes in $s^i(j)$ is at least 1/4.

The claim holds since the fractional service of function i from nodes in $s^i(j)$ received by client c_j is at least 1/4. This is true as client c_j defined ball $B^i_{c_j}$. Originally, client c_j received at least half of its service of function i from $B^i_{c_j}$. By construction, at most 1/4 of the service of function i that c_j receives can come from the sets $s^i(1), \ldots, s^i(j-1)$ (as c_j was not removed previously). Thus, the claim follows.

Claim 5.2: For each client $c \in C$, $\alpha^i_c \geq 1/4$.

The claim follows directly from the way the sets are defined. If a client is removed at some point of time, then (by construction) it has already received at least a total service of 1/4 from sets defined at that time. Else, a client defines a set. By the argument given above (for the previous claim), such a client gets service of at least $1/4$ from the set it defines.

We say that for a client $c \in \Lambda^i_j$, service is (fractionally) *reassigned* in a set $s^i(j)$ if an ε-fraction of c's service is provided by node $u' \in s^i(j)$ instead of $u \in s^i(j)$, as in the fractional solution to (NFV Location-LP). (In our setting, reassignment is always done within a set.) Reassigning of an ε-fraction of service increases the distance cost by at most $\varepsilon \cdot d(u, u')$. The next claim provides an upper bound on the increase in distance cost when reassigning service inside the sets defined by the algorithm.

Claim 5.3: Suppose that all of the service of client c is reassigned inside the sets $s^i(j)$ ($c \neq c_j$) for which $c \in \Lambda^i_j$. Then, the distance cost of c increases by at most $4\alpha^i_c \bar{d}^i_c$.

Proof: Assume c gets a fraction x^i_{cu} of service from node $u \in s^i(j)$. Since $s^i(j) \subseteq B^i_{c_j}$, the maximum distance between any two nodes in the ball $B^i_{c_j}$ is at most $4\bar{d}^i_{c_j}$ (its radius is $2\bar{d}^i_{c_j}$). Also, $\bar{d}^i_{c_j} \leq \bar{d}^i_c$ by construction (c_j was chosen over c when constructing $s^i(j)$.) Thus, summing up over all sets of function i from which c gets service, we get that the cost of rerouting is at most

$$\begin{aligned} \sum_{\{j|c\in\Lambda^i_j\}} \sum_{\{u\in s^i_j\}} x^i_{cu} \cdot 4\bar{d}^i_{c_j} &\leq 4\bar{d}^i_c \sum_{\{j|c\in\Lambda^i_j\}} \sum_{\{u\in s^i_j\}} x^i_{cu} \\ &\leq 4\alpha^i_c \bar{d}^i_c \end{aligned}$$

■

Phase 2. For each function i, we represent the solution we get from the first phase as a bipartite graph (depicted in Figure 2). The left side nodes correspond to clients, and the right side nodes correspond to our disjoint sets of nodes. A client c is connected to each set it either defined, or is associated with, in Phase 1. We now round the solution so that each client gets its full service from a single set only. First, for each set $s^i(j)$, we multiply by a factor of 4 the fraction of function i in each node $u \in s^i(j)$. This increases the total setup cost to at most $4C^*_s$ and violates node sizes by at most a factor of 4. Now, for each client c and node $u \in s^i(j)$ it is feasible to define $\tilde{x}^i_{cu} = (1/\alpha^i_c) x^i_{cu} \leq 4x^i_{cu}$. By Claim 5.2 it holds that each client now gets a total service of 1 from all the sets it is associated with.

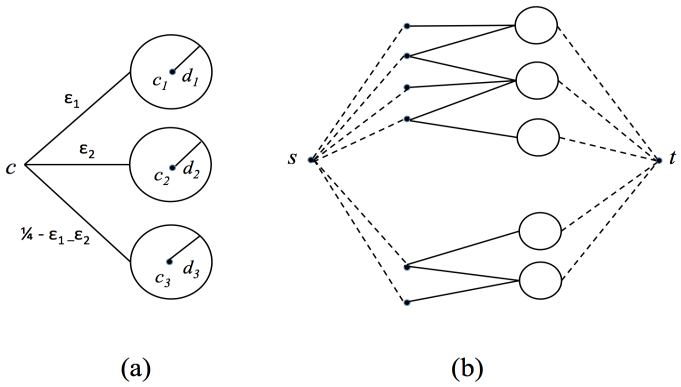

Fig. 2. (a) client c gets a total fraction $\geq 1/4$ from three different sets; (b) flow problem defined in the bipartite graph.

We define a flow problem F as follows. We connect a common source s to each left side client node by edges of capacity 1. Each right side node, corresponding to a different set $s^i(j)$, is connected to a common sink t with an edge of capacity equal to $\lceil \sum_{\{c|c\in\Lambda^i_j, u\in s^i(j)\}} \tilde{x}^i_{cu} \rceil$, i.e., the sum of all fractions assigned to clients belonging to Λ^i_j. We round up in order to get integral edge capacities, incurring at most a loss of a factor of 2, since now the sum of the fractions in each set is at least 1 (after multiplying by 4). The capacity of the edges between clients and sets is set to 1. The edge costs are set as follows. Edges adjacent to the source and sink have zero cost. The cost of an edge between a client c and a set $s^i(j)$ is equal to the maximum distance from c to a node in $s^i(j)$, so as to take into account a possible reassignment of c to another node in $s^i(j)$ (as may happen later on in Phase 3).

Claim 5.4: Consider the solution where each edge between client c and set $s^i(j)$ is assigned a flow of $\sum_{u\in s^i(j)} \tilde{x}^i_{cu}$. This solution is a feasible fractional solution to F, with total cost of at most $8\sum_{c\in C} \bar{d}^i_c \leq 8C^*_d$.

By construction the solution is feasible. The distance cost of the solution defined in the claim is the sum of two terms. The first term is the distance cost resulting from scaling up the service fractions for each client c by a factor of $1/\alpha^i_c \leq 4$. This term is at most $4C^*_d$ when summing up over all clients. The second term is the increase in distance cost by reassigning the service for each client c to the furthest away node in each set from which c gets service. By claim 5.3 this term is at most $4\alpha^i_c \cdot \bar{d}^i_c$, and it needs to be multiplied by the scaler $1/\alpha^i_c$, yielding $4\bar{d}^i_c$. Summing up over all clients we get at most $4\sum_{c\in C} \bar{d}^i_c \leq 4C^*_d$. Adding the two terms yields the bound in the claim.

Phase 2 ends by solving the min cost flow problem F. As there exists an integral optimal solution (since all edges have integral capacities), we get an integral assignment of cost at most $8C_d^*$.

Algorithm 3 Capacitated NFV Location Algorithm

1: Solve (Capacitated NFV Location-LP).
2: For each function i, proceed with Phase 1 & Phase 2:
3: **Phase 1 [steps 3-4].** For each client c, compute $\bar{d}_c^i = \sum_{u \in U} x_{cu}^i \cdot d(c,u)$. Initialize C' to be the full set of clients, initialize $j = 0$.
4: While there are clients in C':
   1) Pick c_{j+1} as the client with minimum $\bar{d}_c^i$ among clients in C'.
   2) Define the set of nodes $s^i(j+1)$ as nodes in $B^i_{c_{j+1}} = B(c_{j+1}, 2\bar{d}^i_{c_{j+1}})$, that are not included in any previous set $s^i(1), \ldots, s^i(j)$.
   3) Clients in C' associated with $s^i(j+1)$ are those getting service from nodes in $s^i(j+1)$. Denote this set of clients by Λ^i_{j+1}.
   4) Remove from C' clients that get a total fraction of function i of at least 1/4, over the sets $s^i(1), \ldots, s^i(j+1)$.
   5) Set $j = j + 1$.
5: **Phase 2 [steps 5-7].** For each set of nodes $s^i(j)$, and each client c associated with it, multiply by $1/\alpha_c^i$ the fractions of c allocated at nodes $u \in s^i(j)$; define $\tilde{x}^i_{cu} = (1/\alpha_c^i) x^i_{cu} \leq 4x^i_{cu}$.
6: Define a min cost flow problem F in a bipartite graph as follows.
   - Left side nodes are clients, right side nodes are sets. Client c is connected to the sets with which it is associated with edges of capacity 1.
   - A common source s is connected to the clients with edges of capacity 1. A common sink t is connected to the sets. The capacity of the edge from $s^i(j)$ to t is set to be $\lceil \sum_{\{c|c \to s^i(j), u \in s^i(j)\}} \tilde{x}^i_{cu} \rceil$.
   - Costs of links connected to s and t are set to 0. The cost of an edge between client c and set $s^i(j)$ is defined to be the distance from c to the furthest away node in $s^i(j)$.
7: Solve the min cost flow problem F, the output being an integral assignment of clients to sets, such that each client gets a unit of service of function i form a single set only. Note that this unit is still fractionally split within the set.
8: **Phase 3 [steps 8-11].** For each function i, set $s^i(j)$, and node $u \in s^i(j)$, define $\tilde{y}^i_u$ as the value of function i assigned to u following Phase 2 $\left(\tilde{y}^i_u = \frac{\sum_{\{c|c \to s^i(j)\}} \tilde{x}^i_{cu}}{\mu^i}\right)$. Define $\tilde{y}^i_u = (\tilde{y}^i_u \mod 1)$ and update $w(u)$ appropriately.
9: Split each set $s^i(j)$ into subsets $s^i(j,1), \ldots, s^i(j,\ell)$, s.t. the sum of the fractions $\tilde{y}^i_u$ in the nodes in each subset $s^i(j,k)$ adds up to precisely 1.
10: GAP: each set $s^i(j,k)$ is a job and the nodes belonging to it are its optional machines. The values $\tilde{y}^i_u$ define a fractional assignment to GAP.
11: Compute an integral solution to GAP by rounding the fractional assignment, the output comprising of integral copies of function i assigned to the nodes of each set $s^i(j)$.
12: For each function i and set $s^i(j)$, connect each client associated to $s^i(j)$ (according to Phase 2) to a single node in the set.

Phase 3. We proceed by defining a feasible instance of the general assignment problem. For each function i, we now have an integral number of clients connected to each set of nodes $s^i(j)$, where for each node $u \in s^i(j)$, we have a value of $\tilde{y}^i_u \leq 8 \cdot y^i_u$ (the 8 factor resulting from the fraction multiplication and the round up performed in Phase 2). For each node u, we subtract the maximum possible integral number from $\tilde{y}^i_u$, remaining with a fraction less than 1 (e.g., in case $\tilde{y}^i_u = 7.4$, only a fraction of 0.4 is left). The integral number removed corresponds to an integral assignment of copies of function i to node u, and we are now left with a fractional assignment that remains to be rounded. Additionally, we update the size of the node $w(u)$ by subtracting the total size of the copies of function i removed from u.

We split $s^i(j)$ to subsets of nodes as follows. We arbitrarily pick nodes, until the sum of their values $\tilde{y}^i_u$ adds up to one. In case the last node picked causes the total amount to be greater than one, we add it both to the current set as well as the next one, with different values of $\tilde{y}^i_u$ (the fraction added to the current set will bring the sum to exactly one, and the value added to the next set will be its remaining fraction).

Assume we end up with subsets of nodes $s^i(j,1), \ldots, s^i(j,\ell)$, each subset comprising of nodes with fractions adding up to precisely 1. The fractional assignment of GAP is interpreted as follows. For each function i, each subset $s^i(j,k)$ is considered a job, and the nodes within $s^i(j,k)$ are its optional machines. The fraction $\tilde{y}^i_u$ indicates the fractional assignment of the job to a machine $u \in s^i(j,k)$.

Claim 5.5: The fractional solution defined by $\tilde{y}^i_u$ is a solution to (GAP-LP) violating the machine sizes by at most a factor of 8.

The claim follows directly as $\tilde{y}$ is obtained by multiplying y by at most a factor of 8; recall that the fractional assignment y is an optimal solution to (NFV Location-LP) and thus adheres to the size constraints of the machines (nodes).

We now perform the rounding algorithm of GAP, resulting in an additional violation of node sizes by a factor of 2, while distance and setup costs are not increased. For each function i and set $s^i(j,k)$, the resulting solution assigns an integral copy of function i to a single node in the set. We now have integral copies of function i for each node in the original set $s^i(j)$ (the subtracted copies plus the integral assignment resulting from the rounding of GAP). Clients connected to this set (as output from Phase 2), can now be integrally connected to one node only, in order to get their unit of function i. Note that the potentially increase in distance cost resulting from connecting a client to a single node in a set is already taken care of by the reassignment cost.

Summarizing the three phases of our rounding procedure, the resulting approximation factors are as follows. Due to the first phase, we obtain a distance cost of at most $8C_d^*$. The second phase results in a setup cost of at most $8C_s^*$ (due to the multiplication of the allocated fractions within the sets by 4 and then rounding up the edge capacities in the min cost instance). We thus achieve a total approximation factor of 8 for our objective function. The size constraints are violated by a factor of 8 in the second phase and another factor of 2 in the third phase, yielding a total violation by a factor of 16.

A. General Demands

In the previous sections, we considered the case where clients have similar service demands of one unit with respect to a function i. We turn to consider the case where clients have general demands, denoting the demand of client c with respect to function i by r_c^i. An integral solution for the general demands problem assigns a client to a single node for each function $i \in f(c)$. A client getting function i's service from node u reduces the capacity of i remaining at u by the amount of service it gets. Assume y^i_u copies of function i are allocated at u. If client c gets its service from u, then the remaining capacity of function i at node u is $(y^i_u \cdot \mu^i - r^i_c)$. The linear

program of this version is easily formulated by updating the demand constraint (1) of (NFV Location LP) to be at least the client's demand.

The rounding algorithm of this variant is presented in [19], and is omitted here due to lack of space. As in the previous cases, it is based on rounding the LP fractional solution. In the general demands case, the integrality gap of the LP can be unbounded (see [19]). To avoid this behavior, a pre-processing phase is performed, where the demands are rounded, and original functions are mapped to new types with rounded capacities. Accordingly, Algorithm 3 is adapted to handle non-uniform demands; specifically, an unsplittable min cost flow is computed over the bipartite graph generated in the second phase. The resulting rounding algorithm is a bicriteria approximation, with increased constant factors both with respect to the objective function, as well as to the violation of the size constraints.

VI. Experimental Study

In this section we evaluate the performance of the *Capacitated NFV Location Algorithm* 3 in realistic scenarios. We consider the physical optical network of Cogent, a tier 1 Internet Service Provider [18]. This network covers 195 access locations (mostly within Europe and North America), with about 260 links and almost 40 data centers. Cogent's structure, along the fact that its topology is publicly available, make it a good candidate for testing our algorithm. We selected 200-400 random pairs of (source, destination), and determined a shortest path between each source and destination, defining the client flow. Each such flow is associated with 1-4 network functions that were chosen randomly from a set comprising of 30 different functions. The size of a network function, which represents the amount of resources it requires, corresponds in this case to the number of VMs required by the function. The capacity of each function represents the number of flows it can handle. The nodes in Cogent's topology represent data centers, and each data center is associated with a size that corresponds to the amount of VMs it can allocate. In our experiments, each data center was set a randomly chosen size between the range $100 - 500$.

In order to evaluate the performance of Algorithm 3, we compare it to a simple greedy algorithm. The greedy algorithm considers the network functions by arbitrary order, and selects for each function the best available remaining location in a greedy way. Specifically, given a function i and a node location u, the clients (i.e., the flow paths) requiring service from i are sorted according to their distance from u. The potential clients to be served in this case are the k closest clients to u, where k is the function capacity. For each function, the location selected is the location with minimum cost, where the cost is the sum of the setup cost of function i at u, and the distances between the k closest clients and u. Since our algorithm is a bi-criteria algorithm allowing violation of the size constraints of the nodes, we allow the greedy algorithm to violate the nodes sizes by an equal factor. This way, we achieve a fair comparison between the performance of the two algorithms.

Figure 3 depicts the ratio of the costs achieved by Algorithm 3 and the greedy algorithm, as a function of the average capacity of a network function in the range of 20-220 (that is, the number of flows that can be served by an instance of an average function). All experiments were performed with 400 flows (routed between the clients and a set of 30 different types of network functions). The curves in Figure 3 correspond to experiments performed with different function sizes.

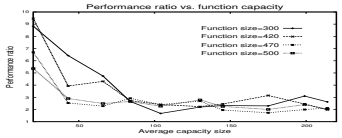

Fig. 3. Performance ratio of the *Capacitated NFV Location Algorithm* and the greedy algorithm with respect to average function capacity.

While in all cases our algorithm performs much better than the greedy algorithm ($60\% - 950\%$), the performance ratio is significantly increased when reducing the average function capacity. This can be explained by the fact that when the capacity decreases, more copies (instances) of each network function are needed, and thus the placement problem becomes harder and local considerations are no longer efficient.

Figure 4 depicts the ratio of the costs achieved by Algorithm 3 and the greedy algorithm, with respect to the number of different functions in the range of 4-31. Each curve in Figure 4 corresponds to simulations performed for a different value of function capacity (10, 90, 170). For each capacity value, experiments were performed over different function sizes, in the range of 200-500. Each curve was computed as the average of results for the different size values, considering a specific function capacity.

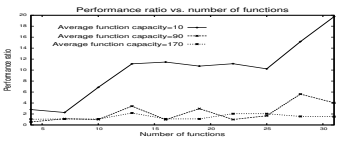

Fig. 4. Performance ratio of the *Capacitated NFV Location Algorithm* and the greedy algorithm with respect to the number of functions.

It can be seen that as the function capacity decreases, the performance ratio increases, as more function copies need to

be allocated. In addition, increasing the number of functions reduces the performance of the greedy algorithm as compared to Algorithm 3, as a local placement decision for a single function may prevent the efficient placement of other functions. The extreme ratio point is thus achieved for a function capacity of 10 and 31 different functions, leading to a performance ratio of 19 between our algorithm and greedy.

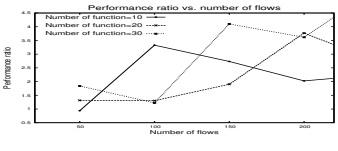

Fig. 5. Performance ratio of the *Capacitated NFV Location Algorithm* and the greedy algorithm with respect to the number of clients.

Lastly, Figure 5 depicts the ratio of the costs achieved by Algorithm 3 and the greedy algorithm, with respect to the number of clients (i.e. flows). Each curve corresponds to a different number of functions (10, 20, 30). Each curve is computed as the average of experiment results for different function capacities and sizes (range of capacities being 10-150, and range of sizes 200-400). The performance of our algorithm increases with the number of clients as compared to greedy. Again, the larger the number of clients, the harder the problem becomes as the connectivity decision taken for each client influences the others.

VII. Discusssion

The vast interest and the fast adoption of Network Function Virtualization and SDN solutions introduce a new networking paradigm with major challenges. In this paper we address one of the main technical challenges in this domain: the actual placement of the virtual functions within the network. We conducted a thorough theoretical study of the NFV location problem, for locating network functions with the goal of minimizing the overall network cost, while adhering to the size constraints of the network nodes.

We considered several variants of the problem up to its most general version, where the functions have limited capacity an can serve only a restricted number of clients, and where demands can vary between clients. We note that even the most basic version of the problem combines two classic NP-hard problems. For all variants, we provided bicriteria results, approximating the objective function by a constant factor while violating the size constraint by a constant factor as well. Our algorithms can all be implemented in the practical setting with high efficiency.

We demonstrated the practical usefulness of these results by evaluating the expected performance gain in several typical networking scenarios. The results indicate that ***.

To the best of our knowledge, our work is one of the firsts to tackle the NFV location setting. From the theoretical point of view, the modeling of our setting introduces a new type of optimization problems, combining network function setup costs, connectivity costs of the clients, and size constraints of the network nodes. Many more problems and future research directions arise in the NFV context. Examples of such are ***.

References

[1] Z. Abrams, A. Meyerson, K. Munagala and S. Plotkin. On the Integrality Gap of Capacitated Facility Location. *Carnegie Mellon University, CMU-CS-02-199.*

[2] D. Bertsimas, and R. Weismantel. Optimization Over Integers. *Dynamic Ideas*, 2005.

[3] M. Bouet, J. Leguay and V. Conan. Cost-Based Placement of Virtualized Deep Packet Inspection Functions in SDN. *Proceedings of the Military Communications Conference, MILCOM 2013 IEEE*, pp. 992-997, 2013.

[4] M. Casado, T. Koponen, R. Ramanathan and S. Shenker. Virtualizing the network forwarding plane. *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2010.

[5] Y. Dinitz, N. Garg, and M.X. Goemans. On the single-source unsplittable flow problem. *Combinatoric a*, Vol. 19, pp. 17-41, 1999.

[6] S. Gebert, D. Hock, T. Zinner, M. Hoffmann, M. Jarschel, E.D. Schmidt, R.P. Braun, C. Banse, and A. Kopsel. Demonstrating the Optimal Placement of Virtualized Cellular Network Functions in Case of Large Crowd Events. *ACM SIGCOMM*, 2014. http://www3.informatik.uni-wuerzburg.de/staff/zinner/preprints/Demo%20POCO%20SIGCOMM.pdf

[7] W.K. Hall. The Application of Multifunction Stochastic Service Systems in Allocating Ambulances to an Urban Area. *Operation Research*, Vol. 20(3), pp. 558-570, 1972.

[8] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker. Extending networking into the virtualization layer. HotNets, 2009.

[9] R. Ravi and A. Sinha. Approximation Algorithms for Multicommodity Facility Location Problems. *SIAM Journal on Discrete Math*, Vol. 24(2), pp. 538-551, 2010.

[10] D.P. Williamson and D.B. Shmoys. The Design of Approximation Algorithms. *Cambridge University Press*, 2010.

[11] Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action. ETSI, October 2012. http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[12] Network Functions Virtualization (NFV): Network Operator Perspectives on Industry Progress. ETSI, October 2013. http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[13] GS NFV 001 Network Functions Virtualization (NFV); Use Cases. ETSI, October 2013. http://www.etsi.org/deliver/etsi_gs/NFV/001_099/001/01.01.01_60/gs_NFV001v010101p.pdf

[14] AT&T Domain 2.0 Vision. *AT&T white paper.* http://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf

[15] The Programmable Cloud Network: Primer on SDN and NFV. *Alcatel Lucent white paper.* http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2013/9146-programmable-cloud-network-primer-sdn-nfv.pdf

[16] The real-time Cloud: Combining Cloud, NFV, and service provider SDN. *Ericson white paper.* www.ericsson.com/res/docs/whitepapers/wp-sdn-and-cloud.pdf

[17] Cisco Open Network Environment: Adaptable Framework for the Internet of Everything. *Cisco white paper.* http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/white_paper_c11-727538.pdf

[18] Cogent's Network Map. http://cogentco.com/en/network/network-map

[19] R. Cohen, L. Lewin-Eytan, J. Naor, D. Raz. Near Optimal Placement of Virtual Network Functions. Full paper version. https://www.dropbox.com/s/iyfrcih5fhb7a00/infocom15.pdf