Mobile operators and content providers in next-generation SDN/NFV core networks: Between cooperation and competition

Mobile Operators and Content Providers in Next-Generation SDN/NFV Core Networks: between Cooperation and Competition

Nir Gazit, Francesco Malandrino, David Hay
The Hebrew University of Jerusalem, Israel

Mobile Operators and Content Providers in Next-Generation SDN/NFV Core Networks: between Cooperation and Competition

Nir Gazit, Francesco Malandrino, David Hay
The Hebrew University of Jerusalem, Israel

Abstract

Much of the traffic processing that today takes place at cloud servers on the Internet will be performed *within* the cellular core network. In such a scenario, different network entities, e.g., switches and servers, will belong to different parties, including traditional mobile operators and over-the-top content providers. The relationship between them is often termed *coopetition*: on the one hand, they are natural competitors; on the other hand, they can reap significant benefits from limited and defined cooperation. Our main purpose in this paper is to model these heterogeneously-owned next-generation networks and solve, within such a novel and complex scenario, the traditional VNF placement and traffic steering problems. Specifically, (i) we present an efficient, online algorithm to make placement and steering decisions, and (ii) evaluate its performance in a realistic scenario. We find that our algorithm would allow mobile operators and content providers to reduce their reliance on third-party vendors – hence the associated costs – by as much as 60%.

1. Introduction

Several evolution trends in cellular networks are now converging and combining with each other.

The first trend concerns the core network and is represented by the emergence of software-defined networking (SDN) and network function virtualization. Today's LTE core networks, based on the Evolved Packet Core (EPC) architecture [1] are built from propriety switches and middleboxes. These include, for example, a Serving Gateway (S-GW) that manages user handovers and billing, a Packet Data Network Gateway (PDN-GW) that handles connectivity to external networks, and cellular endpoints (eNodeB) that provide also encryption and wireless channel management for the network operator. It is important to notice that the EPC architecture already provides a clear distinction between user- and data-plane protocols and protocol entities, however it still uses proprietary and expensive hardware. Thus, a natural direction [2] that next-generation networks might take is to replace these special-purpose boxes with smaller, more flexible middleboxes, each implementing a *network function*.

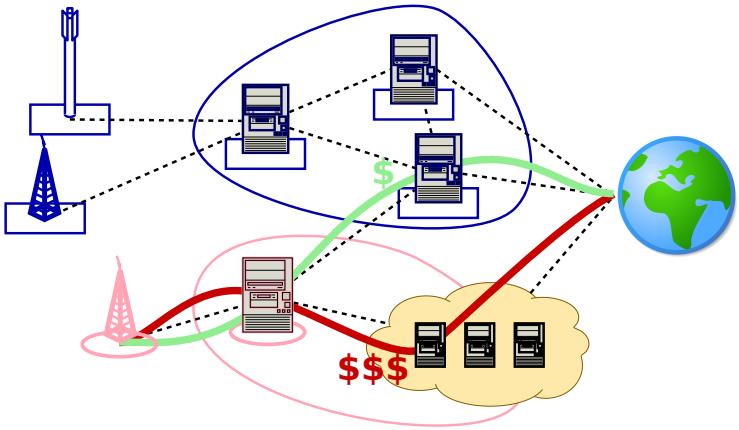


Figure 1: A next-generation cellular network, composed of both base stations and servers, wherein traffic processing takes place. Both base stations and servers have *owners*: some belong to the blue mobile operator (square shadow), and others to the pink content provider (round shadow). In addition to its own servers, the content provider can have the traffic generated at its own base station (the pink one at the bottom) processed at a third-party cloud vendor (bottom, dark red line) or at servers in the mobile operator's core network, encircled in the blue ellipse (top, light green line), if spare capacity is available therein.

The second trend mostly concerns the access network, and is represented by *heterogeneity* [3, 4]: present-day LTE networks are already composed of different kinds of infrastructure, from macro base stations to femtocells; in the future, LTE-Advanced and 5G networks will integrate multiple radio access technologies, including Wi-Fi [5] and millimeter-wave antennas [6]. In parallel, the traffic served by such networks will also become increasingly heterogeneous, coming from different applications (web browsing, real-time gaming etc.), each requiring different service levels [7] and traversing a different chain of network functions.

The third, and perhaps most disruptive, trend is that, for a variety of reasons, most of which are non-technical [8, 9], content providers such as Google, Facebook or Netflix are starting to deploy their own networks [10]. The purpose of such networks is to serve some of the content provider demand directly, and, by that, enhance the available capacity where needed. Unlike the networks deployed by mobile operators, content providers' networks will have a spotty coverage, typically concentrated in densely populated areas. In a similar way, content providers will not build a complete core network, but rather rely on third-party cloud vendors, as depicted in Fig. 1.

However, there is another party that could process the content providers' traffic – mobile operators themselvs. Recall that, in our scenario, mobile operators have built complete core networks capable of processing traffic through VNFs. Furthermore, such networks must be dimensioned so as to meet the *peak* demand mobile operators will face. It follows that, due to daily fluctuations in the traffic demand, they will be underutilized for most of the time.

This spare capacity can be leased to content providers, as shown by the green line in Fig. 1, with significant benefits to both parties: mobile operators can

obtain further revenue form their core networks, and content providers can save over hefty cloud fees. As the time evolution of different types of demand tends to be different [11], mobile operators are likely to have spare capacity available for content providers exactly when they need it. Furthermore, if no spare capacity is available, content providers can always fall back to cloud providers, as shown by the red line in Fig. 1. In this paper, we will focus on the infrastructure corresponding to VNFs (e.g., commodity servers in the core network), and study how different cooperation level between the mobile operator and content provider yields a different preference of a VNF placement.

Specifically, our contribution is threefold. First, we derive a proper mathematical model for the problem described above. Second, we present an efficient, online algorithm to solve this problem. In online settings, where we need to adapt to changes in demand, our algorithm tries to simultaneously minimize the number of placement changes (thus leading to a small amount of resource-consuming migrations) and maximize the policies satisfied. Finally, we assemble a realistic network trace, using real-life topology and demand information, and verify that the traffic load of content providers and mobile operators is indeed sufficiently different to make a cooperative approach viable. We find that optimal placements can allow mobile operators and content providers to reduce their reliance on third-party providers by 93%. The online algorithm, on the other hand, is able to obtain 60% reductions and scales well for large networks.

It is important to point out that our reference scenario only depicts the simplest possible interaction between mobile operators, content providers, and cloud vendors. More complex interactions, e.g., where prices are dynamically adjusted as the demand evolves, are possible and indeed likely in the real world. By focusing on a simple scenario and on basic interactions, our study is able to capture the essential, qualitative features of heterogeneously-owned networks, while also representing a good starting point for further, more sophisticate works.

At the same time, while next-generation mobile networks are the main motivation and primary reference scenario of our study, the problems we explore – most significantly, VNF placement and traffic steering – are also found in other kinds of networks [12]. Therefore, our solution concept can be successfully applied in any workload placing scenario, where two or more entities with different (and potentially conflicting) objectives have the option to cooperate in order to optimize the overall placement.

We begin by reviewing related work in Section 2, focusing on the evolution of cellular networks and problems akin to ours. Then, in Section 3, we present an optimization formulation of our problem, which we view as a matching between virtual network functions and servers. We further discuss the complexity of such optimization, and find it to be NP-hard. Based on such a result, we turn to a heuristic approach: Section 5 describes the decisions that need to be made, and how our model is used in each of them. Section 6 presents an efficient solution strategy, suitable for real-time usage. We evaluate its performance in Section 8, based on the real-world scenario described in Section 7. Finally, Section 9 concludes the paper.

2. Related work

This work mainly considers the challenges and opportunities of transforming cellular networks into software-defined networks. We discuss recent work that has been conducted in these two important fields of research.

Next-generation cellular networks

The general consensus is that next-generation cellular networks will not be just a faster, higher-capacity version of present-day ones. Indeed, new physical-layer technologies, such as massive MIMO [13], and new types of infrastructure, such as millimiter-wave base stations [6], can support altogether new applications (e.g., proximity [14] and machine-to-machine services). *Heterogeneity* is expected to be a distinctive feature of next-generation cellular networks [4, 3], that will also integrate Wi-Fi [5] and device-to-device [15] connectivity.

An important problem in such next-generation networks concerns the ownership and management of such networks. Traditional mobile network operators — typically private companies, with little or no public support — find it increasingly difficult to cope with the capital expenditure needed to update their infrastructure [16]; network sharing agreements between operators represent an early response to this problem [17, 7].

At the same time, the availability of high-performance mobile networks is critical for the business model of such *over-the-top* content providers as Google and Netflix. This gave rise to several forms of revenue transfer from content providers to mobile operators. For example, some content providers subsidize (some) of their mobile traffic, as in the Facebook Zero [18] project and a similar Twitter program [19], effectively paying some of their users' bill. More recently, content providers have started to deploy their own infrastructure in order to complement the operators' one, as Google is doing with their Wi-Fi hotspots [10].

Ultimately, the cellular network is expected to be a *virtualized* network, where resources (e.g., spectrum and infrastructure) coming from different parties are viewed as a single pool, that can be managed in a centralized fashion [8]. The relationship between these parties, which include both traditional mobile operators and content providers, is a hybrid between cooperation and competition, and has been aptly termed coopetition [9].

Software-defined networking

As a result of the aforementioned trends, next-generation cellular networks will be substantially more complex than their present-day counterparts. Software-defined networking (SDN) provides the level of flexibility and fine-grained control needed to effectively manage them.

Early works [20, 2] advocate the adoption of SDN in cellular core networks. In particular, SoftCell [2] envisions to replace the components of traditional backhaul networks (e.g., serving gateways and packet data network gateways) with *middleboxes*, running on commodity hardware. When considering the placement of these middleboxes in a given network, one would want to decide how to *steer traffic* through them. Given a set of traffic demand flows, and the total computational processing power required by middleboxes for each such flow, the work of [21] tries to find an optimal path where: (i) edges are

not overloaded (ii) traffic passes through nodes providing enough computational power to process the entire traffic demand. The problem here is modeled as a variant of the multi-commodity flow problem, and finds a set of paths for each flow, where each paths handles a certain portion of that flow. Our algorithms provide with a more accurate placement but still outputs a set of paths for each flows. To correctly split the traffic between the sets of paths, [22] suggests using relative portions of the IP address space, with respect to the distribution of the traffic. The recent work [23] deals with the practical issues of traffic control and balancing algorithms. In particular, the authors of [23] propose a polynomial-time approximation algorithm, based on the Alternating Direction Method of Multipliers (ADMM).

A convergent trend is *Network Function Virtualization* (NFV) (see [24] and the comprehensive review [25]). The core idea is to implement middleboxes in software running as virtual machines on commodity hardware, thus obtaining lower costs and better scalability [26, 27]. In such settings, SDN's traffic steering mechanisms should also account for the physical host machines they run into and the links connecting them. Most works try to model the set of VNFs each packet has to traverse in the network as sets of policy chains. [12] for example provides a scheme for placing VNFs on physical networks given sets of general policy chains. We extend the notion of policy chains to policy Directed Acyclic Graphs (DAGs) as we find this approach more flexible given the new VNFs required by cellular networks [28, 29]. The notion of policy graphs was already introduced in [30]. It provides a framework to create these special graphs given current policies in a network. We can use the outputs of their framework as an input to our algorithm and as a strong proof for the necessity of using policy graphs instead of sets of policy chains.

The VNF placement problem was also tackled in [31], where it is modeled as a variant of the facility location problem [32, Ch. 4.5] with different types of facilities, and clients requiring subsets of these types of facilities; the network is treated as a metric space which might not be accurate for some network topologies. Algorithms and mathematical formulation for the problem of assigning virtual network functions to physical machines were considered in [21], however, this work focuses more on the traffic steering problem, which is modeled as a multi-commodity flow problem, then on the notion of placing instances of VNFs on machines, and the load balancing schemes possible. Load-balancing between different machines is extremely important in real networks, where traffic demands change constantly. Closer to our scenario, the work [33] proposes SoftAir, a novel architecture for next-generation, software defined networks leveraging the novel concepts of network function cloudification and network virtualization to achieve scalability, flexibility and resilience.

3. System model

In this section, we describe our system model. In Section 4 next, we present a MILP problem build on top of such a system model, including the decisions to make and the constraints to account for. Section 5 then discusses how network

operators and content providers formulate and solve different instances of such a problem.

We present our model in a setting with a single mobile network operator and a single content provider. Our model consists of three main elements, which will be formally defined next: (i) the physical network (including physical host servers and switches, and the links connecting them) (ii) traffic and computation demands for the two parties (iii) steering policies for the two parties. Table 1 summarizes all the symbols we use.

Network The network is modeled as a directed graph $G_N = (V_N, E_N)$. Each edge $e \in E_N$ corresponds to a *physical link*, whose *bandwidth* (or capacity) is denoted by $b(e) \in \mathbb{R}^+$ Mbytes/sec.

Similarly to [12, 2], V_N is comprised of two disjoints sets: The set of servers (denoted by $\mathcal{M}$) and the set of switches (denoted by $\mathcal{S}$). Only servers are capable of running virtual machines, and therefore, can host different VNFs. Every node $u \in V_N$ has a routing table that can hold up to $r_T(u) \in \mathbb{R}^+$ rules, a memory of $c_M(u) \in \mathbb{R}^+$ MBytes, and a CPU computation power of $c_C(u) \in \mathbb{R}^+$ Cycles/s. Naturally, for switches $s \in S$, $c_M(s) = c_C(s) = 0$; while for servers $m \in M$, we can assume that $r_T(m) = 0$.

Virtual network functions (VNFs) We are given a set $\mathcal{F}$ of *virtual network functions* (e.g., firewalls, intrusion detection systems, transcoders). One or more instances of these VNFs are deployed as virtual machines running on different servers. For each VNF $f \in \mathcal{F}$ we denote by $p(f)$ the maximum amount of traffic (in Mbit/sec) a single instance of this VNF can handle. In addition, each instance of $f \in \mathcal{F}$ requires a memory of $r_M(f) \in \mathbb{R}^+$ MBytes and a CPU computation of $r_C(f) \in \mathbb{R}^+$ Cycles/Mbit. It is important to notice that in our model the memory requirements are constant, regardless of the traffic volume the specific instance processes. The CPU requirements, on the other hand, depend on the traffic processed by the VNF. Thus, if there are k instance of some VNF $f \in \mathcal{F}$, each one processing x Mbit/sec of traffic, then the memory requirement is $r_m \cdot k$ MByte, while the incurred CPU load is $r_C(f) \cdot k \cdot x$ Cycles/s. Finally, there might be some other specific hardware requirements that further restrict where VNFs can be placed (e.g., transcoders that need to use special GPUs that only reside on some of the servers). Thus, let $\Gamma : \mathcal{F} \to 2^{V_N}$ be a *placement restriction function*, which defines a set of feasible servers on which each VNF can run. Moreover, as we shall describe shortly, this function is also useful for specifying dummy entrance/exit points for the policy chain.

Policy graphs This work extends the notation of policy chains used in previous works [12, 24] to a policy graph due to the limited descriptiveness of the former. Future policy requirements formulations, as envisioned by [34, 30] are better modeled as a tree or a graph rather than a set of policy chains, and translating the one to the other might result in an exponential increase in the policy requirements data size.

The policy graph is a connected directed graph $G_D = (V_D, E_D)$ with no loops. The vertices of the graph are either VNFs or dummy entrance and exit points. These dummy points correspond, for example, to gateway switches which connect to external network from where traffic arrives. Note that the

Table 1: List of notations.

Notation	Meaning	Remarks
$G_N = (V_N, E_N)$	The Network directed graph	consists of servers and switches
$b(m_1, m_2)$	Bandwidth of the link between servers m_1 and m_2	Parameter, expressed in Mbit/s
$c_C(m)$, $c_M(m)$	CPU and memory capabilities of server $m \in \mathcal{M}$	Parameters, expressed in Cycles/s and MByte
$m \in \mathcal{M}$	Physical servers (vertex on the network graph)	Set
$s \in \mathcal{S}$	Switches (vertex on the network graph)	Set
$f \in \mathcal{F}$	Virtual Network Functions	Set
$\Gamma(f)$	Allowed servers $m \in \mathcal{M}$ for placing VNF f	Set for each VNF
$p(v)$	Amount of traffic that each instance of VNF v can process	Parameter, expressed in Mbit/s
$r_C(v)$, $r_M(v)$	CPU and memory requirements of VNF $v \in \mathcal{F}$	Parameters, expressed in Cycles/Mbit and MByte
$r_T(s)$	Number of rows in the routing table of switch $s \in \mathcal{S}$	
$t(v_1, v_2)$	Traffic demand between VNFs v_1 and v_2 (flow on the policy graph)	Parameter, expressed in Mbit/s
$G_D = (V_D, E_D)$	The policy graph	consists of VNFs from $\mathcal{F}$
$\kappa_{m,v}$	Cost associated with placing an instance of $v \in \mathcal{F}$ in $m \in \mathcal{M}$	Parameter; not necessarily a monetary cost

locations of dummy exit/entrance points are restricted using the Γ function defined earlier. For each edge in the policy graph $e \in E_D$ we denote by $t(e) \in \mathbb{R}^+$ its traffic demands (meaning, the amount of traffic that must pass between the 2 VNFs on its endpoints). This naturally defines the traffic volume that needs to be processed by each VNF node $v \in V_D$ in the graph as the total amount of traffic that enters that VNF: $t(v) = \sum_{(u,v)\in E_D} t(u, v)$. It is important to notice that there is no conservation of traffic demand, as some VNFs might output less traffic than they receive (e.g., an intrusion detection system that drops packets), while others might output more bandwidth than they receive (e.g., a video decoder).

Coopetition model The Coopetition model comprises of a network infrastructure $G_N = (V_N, E_N)$, two disjoint sets of VNFs $\mathcal{F}_\mathcal{O}$ and $\mathcal{F}_\mathcal{P}$ (for the mobile operator and content provider, respectively), and two policy graphs $G_O = (V_O, E_O)$ and $G_P = (V_P, E_P)$. We note that although the mobile operator and content provider might use the same types of VNFs, the model isolates their traffic using 2 disjoint sets of VNFs. On the other hand, we have only one network infrastructure G_N, which typically consists of the mobile network operator infrastructure, but may also include some other servers, such as cloud services that were previously used by the content provider to satisfy its own demands.

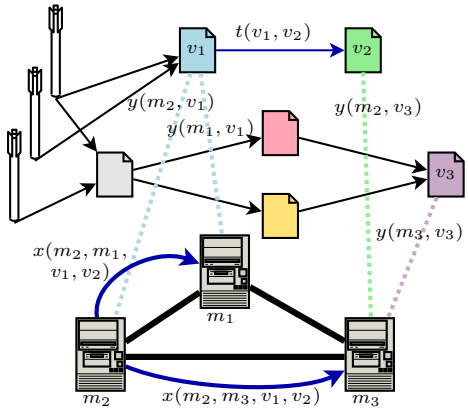


Figure 2: System model. Page icons at the top represent VNFs in $\mathcal{F}$, computers at the bottom physical servers in $\mathcal{M}$. Directed, thin edges represent the amount of traffic between VNFs, e.g., $t(v_1, v_2)$ at the top. Dotted lines represent the match between VNFs and physical servers, expressed through the y-variables; notice the many-to-many relationship. Directed, thick edges represent the flows between physical servers, expressed via the x-variables.

4. The VNF Placement Problem

Our VNF placement problem can be represented as a mixed integer linear program (MILP), where the only integer variables stem from the memory constraints. Table 2 lists all the variables in our formulation.

Table 2: List of MILP variables.

Notation	Meaning	Remarks
$y(m, v)$	amount of traffic processed by instances of VNF v running on server m	Real decision variable, expressed in Mbit/s
$n(m, v)$	Number of instances of VNF v running on server m	Auxiliary variable; equivalent to $\left\lceil \frac{y(m,v)}{p(v)} \right\rceil$
$x(m_1, m_2, v_1, v_2)$	Traffic produced by an instance of v_1 and meant to be processed at an instance of v_2, travelling between server m_1 and m_2	Real decision variable, expressed in Mbit/s

The first decision variable defines on which physical server to run each VNF instance. For each VNF $v \in \mathcal{F}$ and server $m \in \mathcal{M}$, we have a real variable $y(m, v)$, stating how much traffic is processed by instances of VNF v running on Machine m. Given a y-value, there will be $n(m, v) = \left\lceil \frac{y(m,v)}{p(v)} \right\rceil$ instances of VNF v running on Machine m, consuming a total of $n(m, v) r_M(v)$ memory and $y(m, v) r_C(v)$ CPU capabilities.

The second decision variables defines how traffic moves between physical servers. For each pair of VNFs $v_1, v_2 \in V_D$ and pair of servers $m_1, m_2 \in \mathcal{M}$ we

define a real variable $x(m_1, m_2, v_1, v_2)$, denoted the amount of traffic between physical servers m_1 and m_2 that has just been processed by VNF v_1 and need to be processed by VNF V_2. Notice that using real variables whenever possible makes our problem easier less complex to solve. Indeed, virtually all optimization algorithms and solvers perform much better with real variables than with binary or integer ones.

Constraints Next, to ensure the feasibility of VNF placement and VNF flows, we define the following constraints. First, enough VNFs must be deployed to serve all the demand:

$$\sum_{m \in \mathcal{M}} y(m, v) \geq \sum_{u \in \mathcal{F}} t(u, v), \quad \forall v \in \mathcal{F}. \tag{1}$$

Where the left-hand side of (1) is the total amount of traffic that all instances of a VNF v are processing and the right-hand side is the total amount of traffic they have to process (i.e., the sum of the traffic on all graph edges having v as a target).

Next we ensure that valid locations for each VNF are used:

$$y(m, v) = 0, \quad \forall v \in \mathcal{F}, m \in \mathcal{M}, m \notin \Gamma(v) \tag{2}$$

The following three constraints capture the bandwidth, CPU and memory restrictions of physical links and servers.

$$\sum_{v_1, v_2 \in \mathcal{F}} x(m_1, m_2, v_1, v_2) \leq b(m_1, m_2), \quad \forall m_1, m_2 \in \mathcal{M}. \tag{3}$$

$$\sum_{v \in \mathcal{F}} (y(m, v) \cdot r_C(v)) \leq c_C(m), \quad \forall m \in \mathcal{M}; \tag{4}$$

$$\sum_{v \in \mathcal{F}} (n(m, v) \cdot r_M(v)) \leq c_M(m), \quad \forall m \in \mathcal{M}. \tag{5}$$

It is important to notice that the definition of $n(m, v)$ includes a ceiling operator, which makes constraint (5) integral.

Additionally, there is a maximum number of active (i.e., with non-zero traffic) outgoing flows from each server:

$$\sum_{v_1, v_2 \in \mathcal{F}, l \in \mathcal{M}} \mathbb{1}_{[x(m,l,v_1,v_2)>0]} \leq r_T(s), \quad \forall s \in \mathcal{S}. \tag{6}$$

Recall that switches can be modeled as servers with zero capabilities; hence, (6) enables us to account for the limited capacity of the forwarding tables at switches [12, 2].

Finally, we need to ensure that flows defined by the x variables indeed correspond to the flows being processed. A first thing we have to ensure is that VNFs instances running at each server process the appropriate amount of traffic:

$$\sum_{l \in \mathcal{M}, u \in \mathcal{F}} x(l, m, u, v) - \sum_{n \in \mathcal{M}, u \in \mathcal{F}} x(m, n, u, v) = y(m, v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \tag{7}$$

The first term in (7) is the traffic coming to server m and meant to be processed by any instance of VNF v; the second term is the amount of said traffic leaving server m. The difference between the two must correspond to the amount of traffic processed by instances of v running at m, i.e., $y(m, v)$.

Similarly, the traffic leaving server m and meant to be processed at an instance of VNF w "downstream" v in the policy graph must either have entered server m from some other server, or been produced by instances of v running on m:

$$\sum_{n\in\mathcal{M},w\in\mathcal{F}} x(m,n,v,w) = \sum_{l\in\mathcal{M},w\in\mathcal{F}} x(l,m,v,w) + y(m,v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \quad (8)$$

Notice that (7) and (8) also imply flow conservation at servers that do not process any traffic, i.e., have $y(m, v) = 0$.

Objective

The objective function is explicitly defined by the cost values κ:

$$\min \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{F}} \kappa_{m,v} \cdot n(m,v). \quad (9)$$

5. Decision process

Our VNF placing problem problem provides a framework for specifying policies and applying them to a given network infrastructure. We now show how to use this framework in our coopetition environment; without loss of generality, we assume there is only one mobile operator and one content provider.

First and foremost, we notice that a real-life coopetition environment is not static, as traffic volumes vary over time. Thus, the decision process (namely, when our algorithms decide where to place VNFs and how to steer traffic between their deployed instances) is not a one-shot problem but a long-lived one. In the coopetition setting, the decision process consists of two stages. At the first stage, the mobile operator makes VNF placement and traffic steering decisions so as to serve all its demand at the minimum possible cost (e.g., by minimizing the load on its servers). It then announces to the content provider the amount of remaining CPU and memory available at each physical server.

At the second stage, the content provider makes its own placing and steering decisions. Its objective is still to serve its traffic while minimizing the total cost (in this case, typically a monetary one), and it has the opportunity to use its own servers (if available), servers belonging to the mobile operator that have spare capabilities, or even servers obtained from a third-party cloud vendor (typically, at a higher cost).

Notice that in this paper we do not study *how* mobile operators and content providers split the savings obtained through their cooperation. We only provide with a mechanism for both parties to reduce the total expenses imposed by their operations. As stated before, the degree of saving is correlated with the degree

of cooperation between both parties. To achieve those different degrees of cooperation, we use the κ cost figures as a way to control the specific placement each participant chooses. Depending on the extent to which operators and content providers coordinate, we consider several scenarios: (i) no cooperation, or *opportunistic* scenario; (ii) some cooperation, or *content provider-aware* scenario and (iii) *full cooperation*.

5.1. Opportunistic Scenario

Here, the operator ignores the content provider altogether, with the latter acting in a similar way to secondary users in cognitive radio scenarios.

The mobile operator starts by finding a placement for its own policy graph. For example, the cost values κ can be designed to minimize the *load* on the servers:

$$\kappa_{m,v} = \max\left\{\frac{r_C(v)}{c_C(m)}, \frac{r_M(v)}{c_M(m)}\right\} \tag{10}$$

Equation (10) correlates that the price $\kappa_{m,v}$ with the work (in term of CPU and memory) added to server m for processing VNF v. Intuitively, using loads as placement criteria has the positive effect of leaving as many servers as possible relatively free. This, in turn, gives the content provider (whose demands are not known to the mobile operator) more degrees of freedom in choosing where to place its own VNFs instances.

The mobile operator will then set the (probably monetary) cost for the content provider. The latter can then decide to use the servers provided by the mobile operator, or some external cloud provider, depending on the cost defined by each one. The cost set by the mobile operator can, for example, reflect current loads on the server, so as to keep servers as free as possible (this is especially useful when there is more than one content provider and the mobile operator wants to give to each one as many degrees of freedom as possible). One cost function that reflects this is:

$$\kappa_{m,v} = \max\left\{\frac{\sum_{j=1}^{|\mathcal{F}_\mathcal{O}|} a_{m,j} \cdot r_C(j)}{c_C(m)}, \frac{\sum_{j=1}^{|\mathcal{F}_\mathcal{O}|} ic_{m,j} \cdot r_M(j)}{c_M(m)}\right\}, \tag{11}$$

where $A_{mo} = [a_{i,j}]$ and $IC(A_{mo}) = [ic_{i,j}]$ are the placement matrix and the instance count of the mobile operator.

5.2. Content Provider-Aware Scenario

In this scenario, which is closely related to the previous one, we notice that the mobile operator can use historical information to further increase its revenue, e.g., by leaving more spare capabilities at those servers that the content provider used more in the past. There are many ways to change the costs defined by (10) to reflect that. One elegant cost setting is to use (11) with the current joint placement of both the content provider and the mobile network operator.

5.3. Cooperative Scenario

In this scenario, operators and content providers share their demand information and make their decisions jointly, so as to maximize the mutual benefit. This implies that the problem is solved only once, having a single policy graph which is the disjoint union of the mobile operator and content provider policy graphs. The cost values in this scenario is often tightly related to how the mobile network operator and the content provider split their savings, which is out of the scope of this work.

5.4. Problem complexity

In all the scenarios we examined, content providers and network operators have to solve a mixed-integer linear programming (MILP) problem, which is NP-hard[1].It follows that scalability cannot be ensured for medium- to large-scale instances. Therefore, we present an *online* algorithm, whose low computational complexity makes it suitable for real-time usage.

6. Online algorithm

Solving our placement and steering problem as it is stated in Section 3, i.e., optimizing (9) subject to constraints (1)–(8), is impractical for two reasons. First and most obvious, it would mean solving an NP-hard problem (we skip the full proof, based on a reduction from the boolean satisfiability problem). Perhaps more important, since decisions are made from scratch at every iteration, it can bring to a very high number of *migrations* of VNFs between servers.

We address both issues by presenting an online solution concept, summarized in Algorithm 1. Our algorithm starts with an initial solution, as defined in Section 3: the VNF placement $\bar{n}$ and $\bar{y}$ and the traffic steering decisions $\bar{x}$. This initial solution may be obtained by solving the MILP one-off, or even by some heuristic placement provided by the user. As the traffic evolves over time, we adapt the placement and traffic steering variables. We have an *aggressiveness* parameter $0 \leq \alpha \leq 0.5$. This value will be used as a threshold: the algorithm will try to free up servers with load higher then $1 - \alpha$, and shut down servers with load lower than α.

In Line 1, we compute the following λ-metric, capturing how much spare capacity we have on all servers running instances of a VNF v:

$$\lambda(v) = \frac{\sum_{u \in \mathcal{F}} t(u, v)}{\sum_{m \in \mathcal{M}} \min\left(p(v)n(m, v), \frac{\phi(m,v)}{r_C(v)}\right)}, \tag{12}$$

where $\phi(m, v) = c_C(m) - \sum_{u \neq v} \bar{y}(m, u) r_C(u)$ is the amount of CPU available at server m for instances of VNF v.

[1]Our problem is indeed NP-complete. We skip the full proof, based on a reduction from the 3-SAT problem, in the interest of simplicity and brevity.

Specifically, $\lambda(v)$ is the ratio between the total amount of traffic that currently needs to be processed by instances of VNF v and the total amount of traffic that existing instances of v can tackle. The minimum in the summation at the denominator reflect situations where a very loaded server cannot provide CPU of $p(v)$ cycles/second for all VNF v's instances that are currently running on it.

Line 2 checks if there are VNFs that require scaling out, i.e., for which the λ-value exceeds $1 - \alpha$: if such VNFs do exist, we will provision additional instance(s) for it. Notice again that lower values of the aggressiveness parameter α immediately translate into less action being taken.

If we do decide to take action, then we start by deploying an additional instance of the VNF with the highest λ-value (Line 3). To choose which server to deploy the instance at, we solve an LP relaxation of the problem in Section 3 with objective (9), integrated with the following constraints:

$$\tilde{y}(m,v) > \big(\bar{n}(m,v) - 1\big) \cdot p(v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \tag{13}$$

$$\tilde{y}(m,v) \leq \bar{n}(m,v) \cdot p(v),$$
$$\forall m \in \mathcal{M}, v \in \mathcal{F}\colon r_M(v) + \sum_{u \in \mathcal{F}} \bar{n}(m,u) r_M(u) > c_M(m). \tag{14}$$

Constraint (13) ensures that the relaxed solution is a superset of the original solution, and not a completely different one; this allows us to limit the number of changes to the network. Constraint (14) ensures we do not assign additional instances of VNF v to servers that cannot possibly host another instance of v due to memory constraints.

Assuming that we have a relaxed solution $\tilde{y}(m,v)$ (Line 4), we use that solution to choose a server $m^\star$. We select the server that maximizes $\tilde{y}(m,v^\star) - \bar{y}(m,v^\star)$, i.e., intuitively, where the relaxed solution suggested that more traffic should be handled. In Line 9, we deploy an extra instance of VNF $v^\star$ at $m^\star$. Finally, we again use the relaxed solution to re-balance the traffic across the instances proportionally to to the $\tilde{y}$ values given by the relaxed solution:

$$\bar{y}(m,v) = t(v) \frac{\tilde{y}(m,v)}{\sum_{m' \in \mathcal{M}} \tilde{y}(m',v)} \tag{15}$$

When there are no more VNFs with load $\lambda(v)$ greater than $1 - \alpha$, we move to Line 12, and start looking for VNFs with a small load, lower than α. Turning off some instances of these VNFs will reduce costs, without impairing our ability to serve all traffic. More exactly, in Line 13 we select the VNF with the lowest λ-value, and in Line 14 we solve an LP-relaxed problem to select the server at which to switch the instance off. Similar to Line 4, we use an additional restriction which is a following modified version of (13), imposing that the relaxed solution is a subset of the original one:

$$\tilde{y}(m,v) \leq \bar{n}(m,v) \cdot p(v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \tag{16}$$

In Line 15 we use the difference between the current and relaxed y-values to select the server $m^\star$, remove an instance of VNF $v^\star$ from there (Line 16), and again re-balance the y values (using (15)) and update λ-values (Line 18).

When we exit the loop, there are neither overloaded nor underloaded VNFs, and the algorithm terminates. The calculated $\bar{x}, \bar{y}$ are then sent to the *Traffic Steering Module* and the *Server Manager* accordingly which change the placement and routing rules on the network itself. The algorithm will be invoked again when the values measured by the *Traffic Monitor* indicates an increase or decrease in the traffic in the network.

Notice that Line 4 might fail to produce a relaxed solution, due to a failure to meet the constraints (13) and (14). This means that the current placement cannot be adapted to meet the new demands. Thus, we are foced to compute the initial decisions $\bar{x}, \bar{y}$ afresh by solving the full MILP problem, obtaining a new optimal solution. It is wise to also decrease the α values in such a case, in order to be able to adapt more rapidly to ongoing changes in the network.

It is easy to see that, unless the full MILP resolution is triggered, each execution of Algorithm 1 has polynomial time complexity. Both loops run for at most $|\mathcal{F}|$ iterations, and within each loop we solve a *relaxed* LP problem, which also has polynomial complexity.

7. Reference scenario

Our reference scenario includes one mobile network operator and one content provider. We build it leveraging three sources:

- the location of the base stations of two European mobile network operators, presented in [35];
- census data from the same country [36];
- a measurement paper [11], presenting the temporal evolution of both the global mobile traffic and the mobile traffic from specific websites and services.

It is worth stressing that all the information we use is public and/or published. Fig. 3 summarizes how we process our data.

Network operator infrastructure

The base stations of the mobile network operator are simply the base stations of the largest operator of our trace. The demand they serve is constructed in such a way that (i) its temporal evolution conforms to the one reported in [11] and (ii) the demand at each base station is proportional to the population it covers.

As for the core network, we assume the same three-layer topology considered in [2]. Specifically, we cluster the base stations in groups of ten, and connect each such group in a ring; these rings of base stations form the network access layer. The aggregation layer is formed by $k = 6$ *pods*, each connected to $k/2 = 3$ rings of base stations. The core layer consists of $k^2 = 36$ switches connected in full mesh.

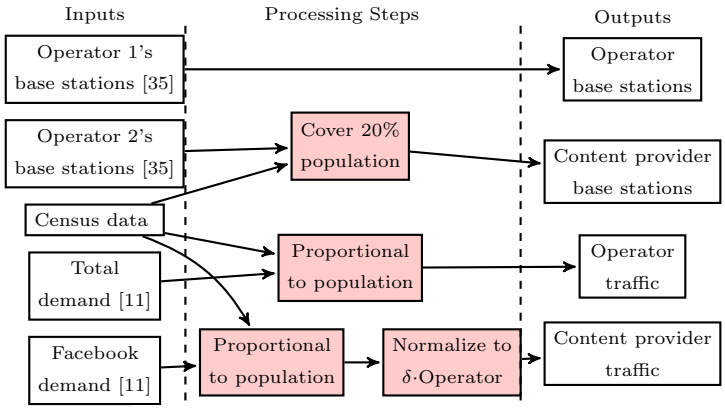


Figure 3: The construction of our reference scenario.

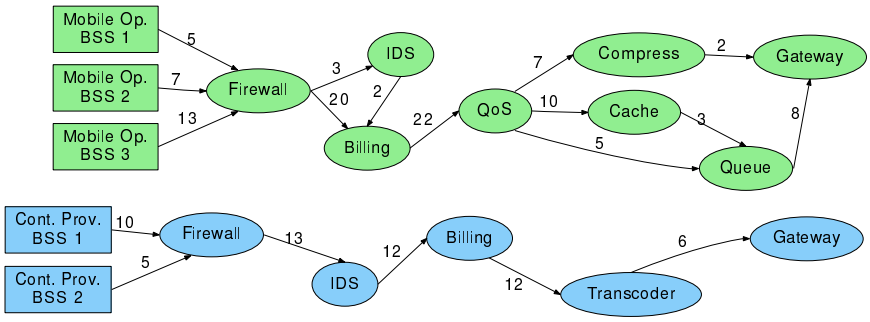


Figure 4: The policy graph we use in our test scenario. Boxes correspond to entry points, ellipses to VNFs that the traffic has to traverse. The values on the edges is the amount of data transmitted between the VNFs on their endpoints.

Servers and traffic demand

There are $|\mathcal{M}| = 54$ servers, all with normalized capabilities $c_M = c_C = 1$. As in [2], half of the servers are connected to random switches in the core layer, half to random switches in the aggregation layer.

Traffic originating from base stations belonging to the mobile operator is processed through the policy graph of Fig. 4, which is similar to the processing done in today's LTE-based mobile networks [26]. The traffic volumes on edges refer to five example base stations in our topology. Traffic from content providers' base stations is processed through the policy graph depicted Fig. 4, which includes several different VNFs for custom processing, inspired by [2, 30]. All VNFs have CPU and memory requirements randomly set between 0 and 1.

Content provider topology and demand

Because infrastructures deployed by content providers are still in their infancy, we need to use the topology of the second mobile operator in our dataset to construct the content provider's network. We make two assumptions:

1. as content providers are expected [8] to only cover the areas where the

demand for their services is most dense, we only keep the 10% most loaded base stations of the second mobile operator;

2. following recent reports [37] that Netflix's traffic accounts for over one third of the total Internet traffic, we multiply the second operator's traffic by $\delta = 0.5$.

The traffic demand follows the temporal evolution reported in [11] for Facebook[2]. The traffic is proportional to the population served by each base station, and represents a fraction δ of the operator's demand.

Each of the content provider's base stations is connected to both the cloud provider and the closest base station belonging to the mobile operator; such a link enables the two parties to cooperate. For simplicity, we assume that the content provider does not deploy any physical server of its own.

Fig. 5 describes how the traffic demand changes over space and time. Fig. 5(a) shows that, while both the demands of the mobile operator and the content provider exhibit the usual afternoon and evening peaks, neither their sizes nor their shapes match. This confirms our intuition that there is, potentially, enough spare capacity at the mobile operator's core network to accomodate a substantial fraction of the content provider's traffic. Comparing Fig. 5(b) and Fig. 5(c) provides us with insights about the location of the different demands. In the maps, lighter colors correspond to higher demand. By comparing the two maps, we can easily observe that the demand of the content provider has a stronger tendency to be localized in specific areas. This is consistent with the widespread assumption that content providers are interested in deploying their infrastructure only in some locations, but the resulting coverage will be partial and spotty.

8. Experimental results

8.1. Optimal decisions

We first tackle the question what is the most the mobile operators and content providers can save by switching from a pure competition relationship to a coopetition. Thus, we have optimally solved the problem (see Section 3), using CPLEX [38]. Fig. 6(a) and Fig. 6(b) show where the traffic is processed for the opportunistic and cooperative scenarios. Specifically, the red areas in the figures represent the traffic demand of the mobile operator, all of which is served through its own servers (cf. Fig. 5). The blue and yellow areas correspond to the demand of the content provider: the blue part is served at the mobile operators' servers and the yellow through a third-party cloud vendor. If there were no cooperation/coopetition between the mobile operator and the content provider, the latter would have to pay cloud processing fees for the traffic corresponding to both the blue and yellow areas in the plots. On the other hand, if coopetition is enabled, the content provider only has to pay such fees for the traffic depicted by yellow areas. In other words, the blue area corresponds to fees that should be paid to the cloud vendor in a pure competition scenario, and

[2]Selecting any other large content provider, e.g., Google, would have yield the same results.

are saved in a coopetition scenario: a limited and defined cooperation between mobile operators and content providers can allow them to save 93% on cloud fees if decisions are made jointly, or 73% if the decisions are opportunistic. We can further notice that cloud servers are only used during peak times, for at most 41% (58%) of the total load in the cooperative (opportunistic) scenarios. Fig. 6(c) shows the difference between the two scenarios is especially significant during peak time, when it is more likely that the deployment and steering decisions made by the mobile operator conflict with the needs of the content provider.

In Fig. 7, we move to the online case, where decisions are made through the algorithm described in Section 6. Fig. 7(a) shows how the traffic is served; comparing it with Fig. 6(b), we can immediately see that the yellow area, i.e., the amount of traffic served at the cloud vendor, is larger. In Fig. 7(b), the area below the blue curve corresponds to the savings of our online algorithm which amount to 60% of the cloud fees; the area between the yellow and the blue curves represents the savings that would have been possible if decisions were made optimally but our online algorithm cannot attain.

On the other hand, our online algorithm requires significantly less machine migrations that the optimal algorithm. Fig. 7(c) compares the number of these migrations, showing that our online algorithm reduces the number of migrations almost by an order of magnitude. This, in turn, simplifies the management of the virtualized network and reduces the associated overhead. It is also important to note that the number of migrations in the online algorithm is almost constant over time and does not depend on the network load, while the optimal algorithm incurs a surge in the migrations in response to even minor changes in traffic demand.

We also compared the utilization of servers by our online and optimal algorithm. Specifically, Fig. 8 focuses on the cooperative scenario and shows that when decisions are made optimally (Fig. 8(a)) there is more CPU time devoted to the content provider's traffic and less idle time than with the online algorithm (Fig. 8(b)).

It is perhaps more interesting to observe that even when decisions are made optimally, and even during peak hours, there is a small amount of idle CPU. This corresponds to servers that have spare CPU, but no spare memory or network capacity. While the first issue can be solved by (over)provisioning more memory, the latter is a direct consequence of the fact that we are dealing with a set of *networked* servers, and the very objective of managing them is to match the available CPU and network capacity.

8.2. Aggressiveness

In Section 6 we mentioned the aggressiveness parameter α that dictates how over- or under-loaded a VNF has to be before our algorithm decides to take action. In Fig. 9, we study how this parameter influences the behavior of our algorithm and its performance. Quite surprisingly, we find that lower aggressiveness values almost invariably translate into better performance. Intuitively, this is linked to the fact that our algorithm starts with an optimal solution, and

adjusts it over time: setting a low aggressiveness value means being less eager to make changes to the current initial solution, including unnecessary ones.

Fig. 9 also shows that lower aggressiveness values imply fewer VNF migrations. It is important to note that setting the right aggressiveness value in real-world scenarios will invariably involve some trial-and-error. Yet, our results suggest using low aggressiveness values.

9. Conclusion

Content providers are expected to take part in the creation of next-generation cellular networks, deploying their own base stations in those areas that are most significant to them. We envisioned that the traffic generated therein can be processed with the help of traditional mobile operators, which would open their SDN-based, virtualized core network to content providers in exchange for a fee.

We modeled the twofold problem of (i) matching virtual network functions and servers and (ii) steering traffic between them. After proving that optimizing the original problem is NP-hard, we proposed an online, scalable algorithm to solve it.

We evaluated our performance using a real-world scenario, assembled using publicly-available deployment and demand information. We found that in the optimal case, coopetition between mobile operators and content providers can allow them to save 93% on third-party cloud fees when decisions are made jointly and 73% when they are made separately. Our online algorithm is able to attain a 60% cost redution, while reducing the number of VNF migrations of almost one order of magnitude.

References

[1] 3GPP, The Evolved Packet Core, 2015, `http://goo.gl/ouqqKZ`.

[2] X. Jin, L. E. Li, L. Vanbever, J. Rexford, Softcell: Scalable and flexible cellular core network architecture, in: ACM CoNEXT, 2013.

[3] A. Ghosh, N. Mangalvedhe, R. Ratasuk, B. Mondal, M. Cudak, E. Visotsky, T. Thomas, J. G. Andrews, P. Xia, H. S. Jo, et al., Heterogeneous cellular networks: From theory to practice, in: IEEE Communications Magazine, 2012.

[4] J. Andrews, Seven ways that HetNets are a cellular paradigm shift, in: IEEE Communications Magazine, 2013.

[5] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin, M. Debbah, When cellular meets WiFi in wireless small cell networks, in: IEEE Communications Magazine, 2013.

[6] S. Rangan, T. S. Rappaport, E. Erkip, Millimiter Wave Cellular Wireless Networks: Potentials and Challenges, in: Proc. of the IEEE, 2014.

[7] P. Di Francesco, F. Malandrino, L. A. DaSilva, Mobile network sharing between operators: a demand trace-driven study, in: ACM SIGCOMM CSWS workshop, 2014.

[8] L. Doyle and J. Kibiłda and T. Forde and L. A. DaSilva, Spectrum Without Bounds, Networks Without Borders, in: Proc. of the IEEE, 2014.

[9] J. Markendahl, B. G. Mölleryd, On Co-opetition between Mobile Network Operators: Why and How Competitors Cooperate, in: ITS Biennial Conference, 2012.

[10] B. Chen, Google Confirms Plans for Wireless Service, in: New York Times, 2015.

[11] Y. Zhang, A. Arvidsson, Understanding the Characteristics of Cellular Data Traffic, in: ACM SIGCOMM Computer and Communications Review, 2012.

[12] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, SIMPLE-fying middlebox policy enforcement using SDN, in: ACM SIGCOMM Computer and Communications Review, 2013.

[13] S. Sun, T. Rappaport, R. Heath, A. Nix, S. Rangan, MIMO for millimeter-wave wireless communications: beamforming, spatial multiplexing, or both?, in: IEEE Communications Magazine, 2014.

[14] X. Lin, J. G. Andrews, A. Ghosh, R. Ratasuk, An overview of 3GPP device-to-device proximity services, in: IEEE Communications Magazine, 2014.

[15] F. Malandrino, C. Casetti, C.-F. Chiasserini, Toward D2D-enhanced heterogeneous networks, in: IEEE Communications Magazine, 2014.

[16] Credit Suisse, U.S. wireless networks running at 80% of capacity, `http://benton.org/node/81874`.

[17] J. Kibilda, L. DaSilva, Efficient coverage through inter-operator infrastructure sharing in mobile networks, in: IFIP Wireless Days, 2013.

[18] Facebook, Fast and Free Facebook Mobile Access with 0.facebook.com, `http://on.fb.me/1P3g2ja`.

[19] S. Datoo, Twitter's latest deal points to ambitions in emerging markets, in: The Guardian, 2013.

[20] L. E. Li, Z. M. Mao, J. Rexford, Toward software-defined cellular networks, in: IEEE EWSDN workshop, 2012.

[21] M. Charikar, Y. Naamad, J. Rexford, K. Zou, Multi-commodity flow with in-network processing, in: Princeton Manuscript.

[22] R. Wang, D. Butnariu, J. Rexford, et al., Openflow-based server load balancing gone wild, in: ACM Hot-ICE, 2011.

[23] S.-C. Lin, P. Wang, M. Luo, Control traffic balancing in software defined networks, Computer Networks.

[24] G. Gibb, H. Zeng, N. McKeown, Outsourcing network functionality, in: ACM HotSDN, 2012.

[25] I. F. Akyildiz, S.-C. Lin, P. Wang, Wireless software-defined networks (w-sdns) and network function virtualization (nfv) for 5g cellular systems: An overview and qualitative evaluation, Computer Networks.

[26] ETSI, Network Functions Virtualisation: an Introduction, Benefits, Enablers, Challenges and Call for Action .
URL {https://portal.etsi.org/nfv/nfv_white_paper.pdf}

[27] R. Jain, S. Paul, Network virtualization and software defined networking for cloud computing: a survey, in: IEEE Communications Magazine, 2013.

[28] 3GPP, Technical specification 23.203: Policy and charging control architecture, http://www.3gpp.org/DynaReport/23203.htm.

[29] 3GPP, Technical specification 33.401: Security architecture, http://www.3gpp.org/DynaReport/33401.htm.

[30] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, Y. Zhang, PGA: Using graphs to express and automatically reconcile network policies.

[31] L. Lewin-Eytan, J. S. Naor, R. Cohen, D. Raz, Near Optimal Placement of Virtual Network Functions, in: IEEE INFOCOM, 2015.

[32] D. P. Williamson, D. B. Shmoys, The design of approximation algorithms, Cambridge University Press, 2011.

[33] I. F. Akyildiz, P. Wang, S.-C. Lin, Softair: A software defined networking architecture for 5g wireless systems, Computer Networks.

[34] A. Bremler-Barr, Y. Harchol, D. Hay, Y. Koral, Deep packet inspection as a service, in: ACM CoNEXT, 2014.

[35] P. Di Francesco, F. Malandrino, L. DaSilva, Cellular Network Planning using Real Data, in: Submitted to IEEE Transactions on Big Data, 2015.

[36] Irish CSO census data, 2011, http://www.cso.ie/en/census/census2011_boundaryfiles/.

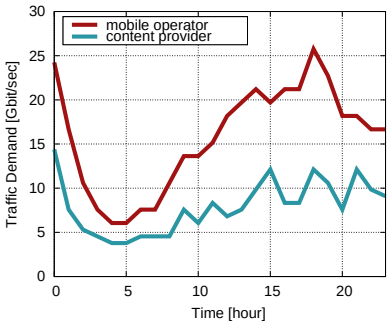
[37] N. Arce, Netflix Is Hogging 35 Percent of Peak Internet Traffic in North America: What About Others?, in: Tech Times, 2014.

[38] IBM CPLEX linear program solver, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/, accessed: 2015-09-30.

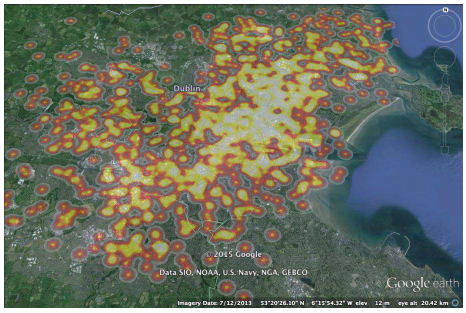
Algorithm 1 Our online algorithm.

Require: current load t, initial decisions $\bar{x}, \bar{y}, \bar{n}$

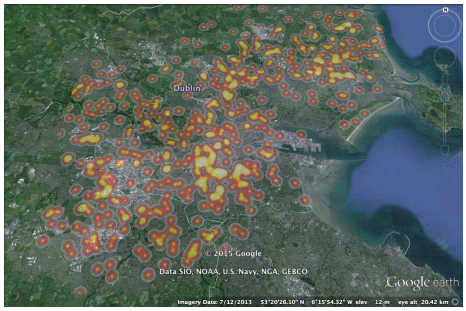
compute $\lambda(v)$ as defined in (12)
while $\max_{v \in \mathcal{F}} \lambda(v) > 1 - \alpha$ **do**
    $v^\star \leftarrow \arg\max_{v \in \mathcal{F}} \lambda(v)$
    $\tilde{x}, \tilde{y} \leftarrow$ **solve** LP relaxation (1)–(9),(13)–(14)
    if there is no feasible solution **then**
        Find a new optimal placement using the MILP.
    else
        $m^\star \leftarrow \arg\max_{m \in \mathcal{M}} \left(\tilde{y}(m, v^\star) - \bar{y}(m, v^\star)\right)$
        $\bar{n}(m^\star, v^\star) \leftarrow \bar{n}(m^\star, v^\star) + 1$
        re-balance $\bar{y}$ using (15)
        update $\bar{x}, \lambda$
while $\min_{v \in \mathcal{F}} \lambda(v) < \alpha$ **do**
    $v^\star \leftarrow \arg\min_{v \in \mathcal{F}} \lambda(v)$
    $\tilde{x}, \tilde{y} \leftarrow$ **solve** LP relaxation of problem (1)–(9),(16)
    $m^\star \leftarrow \arg\min_{m \in \mathcal{M}} \left(\bar{y}(m, v^\star) - \tilde{y}(m, v^\star)\right)$
    $\bar{n}(m^\star, v^\star) \leftarrow \bar{n}(m^\star, v^\star) - 1$
    re-balance $\bar{y}$ using (15)
    update $\bar{x}, \lambda$
return $\bar{x}, \bar{y}, \bar{n}$



(a)

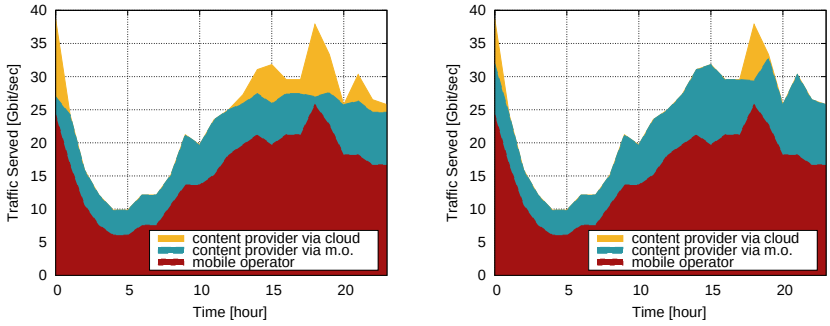


(b)



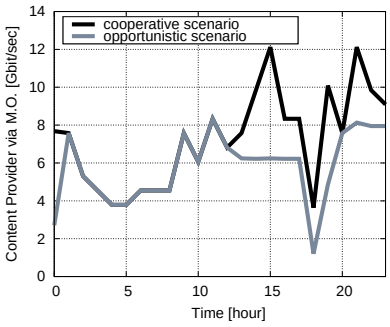
(c)

Figure 5: (a): time evolution of the total traffic demand for the mobile operator (MO) and the content provider (CP); (b), (c): location of the demand for the mobile operator and for the content provider.



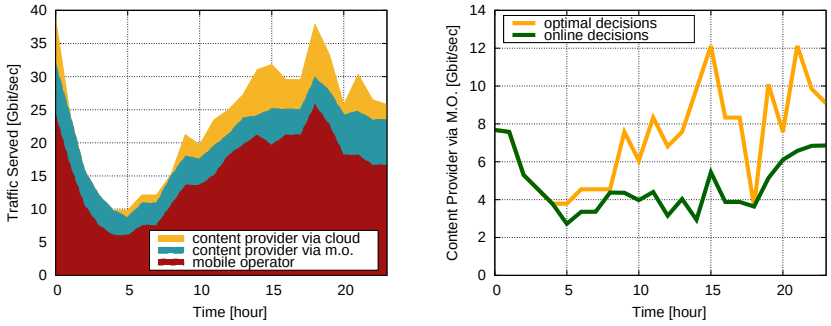
(a)

(b)



(c)

Figure 6: Optimal decisions. (a): how the traffic is served in the opportunistic scenario; (b): how the traffic is served in the cooperative scenario; (c): amount of content provider's traffic processed at the mobile operator's servers in both cases.



(a) (b)

(c)

Figure 7: Online decisions, cooperative scenario. (a): how the traffic is served; (b): amount of content provider's traffic processed at the mobile operator's servers; (c): number of VNF migrations.

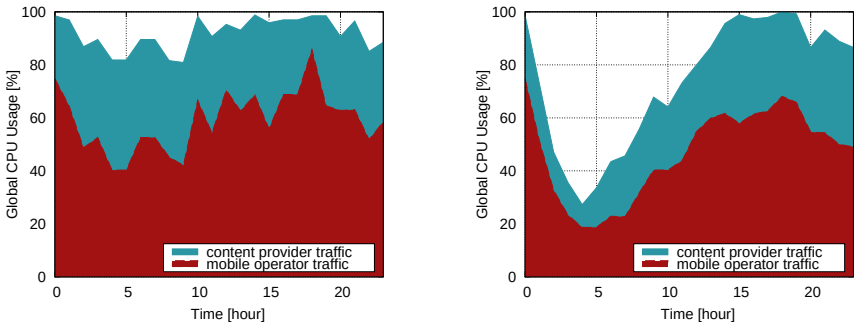


Figure 8: Cooperative scenario: CPU usage at the mobile operator's servers. (a): optimal decisions; (b): online decisions.

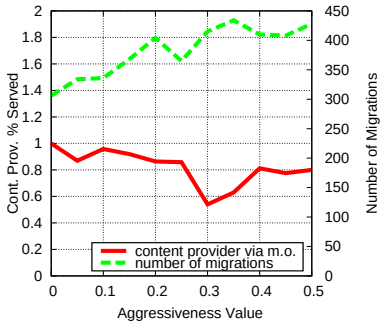


Figure 9: Online decisions, cooperative scenario. Effect of the aggressiveness α on the performance and the number of VNF migrations.