Coopetition Between Network Operators and Content Providers in SDN/NFV Core Networks

Nir Gazit, Francesco Malandrino, and David Hay
School of Engineering and Computer Science, Hebrew University, Jerusalem 91904, Israel
{nirgazit1, francesco, dhay}@cs.huji.ac.il

***Abstract*—It is widely expected that the core of next-generation cellular networks will be (i) based on software-defined networking (SDN) and network function virtualization (NFV), and (ii) shared between multiple parties, including traditional mobile operators and content providers. Such parties are normally competing with each other; however, they can obtain significant, mutual benefits even from limited and defined cooperation. We study how such a *coopetition* relationship influences the decisions of (i) how to place virtual network functions on physical hardware and (ii) how to steer traffic between them. We present an efficient, online algorithm to make such placement and steering decisions, and study 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 by 60%.**

I. 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 (NFV)*. Today's LTE core networks are based on the Evolved Packet Core (EPC) architecture and built from propriety routers 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 [1] that next-generation networks might take is to replace these special-purpose boxes with smaller, more flexible middleboxes, each implementing a *network function*. As envisioned in [2], such core networks will have enough flexibility to efficiently process traffic flows belonging to different parties, lowering the resulting costs.

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.

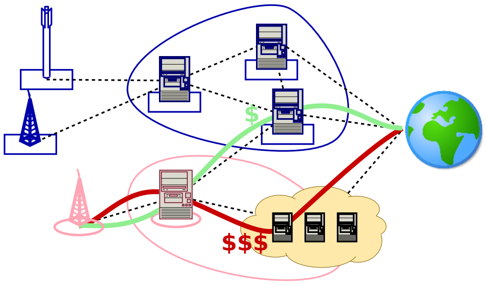


Fig. 1. A next-generation cellular network, some parts of which belong to the blue mobile operator (square shadow), and others to the pink content provider (round shadow). The content provider can have the traffic generated at its own base station (the pink one at the bottom) processed at a third-party 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 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.

This peculiar feature of content providers' core networks represents a *cooperation* opportunity we seek to exploit. On one hand, mobile operators have built complete core networks which, due to the daily fluctuations in traffic demand, are partially unused for most of the time; on the other, content providers are in need for that very same computational and network capacity. Allowing content providers to use the spare capacity available at the mobile operators' core, as shown by the green line in Fig. 1, provides significant benefits to both: mobile operators can obtain further revenue form their core networks, and content providers can save over hefty cloud fees[1]. In this paper, we will focus on the infrastructure corresponding to Virtual Network Functions (VNFs), such

[1] Larger content providers, having their own data centers, would not incur in cloud fees, but rather in higher setup/maintenance costs. Also notice that the mobile operator's computational capacity is deployed right *in* the core network, resulting in lower delays.

as commodity servers in the core network. We will study how different cooperation levels between the mobile operator and content provider yield a different preference of a VNF placement.

Specifically, our contribution is threefold. First, we derive a proper mathematical model (Section II) for the problem described above. Second, we present an efficient, online algorithm to solve this problem (Section III). 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 number of resource-consuming VNF 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 (Section V). 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.

II. System model

We envision a two-stage decision process that, without loss of generality, we describe with reference to one mobile operator and one content provider only.

In 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., minimizing the load on servers. It then announces to the content provider the amount of spare CPU and memory available at each server.

In 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, and it has the opportunity to use its own servers (if available), or servers belonging to the mobile operator that have spare capabilities, or servers obtained from a third-party cloud vendor.

Depending on the extent to which operators and content providers coordinate, we can distinguish several scenarios. In the *opportunistic* scenario, the operator ignores the content provider altogether, with the latter acting in a similar way to secondary users in cognitive radio scenarios. In the *content provider-aware* scenario, the mobile operator uses 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. In the *cooperative* scenario, operators and content providers share their demand information and make their decisions jointly, so as to maximize the mutual benefit: as we will see, the degree of saving is correlated with the degree of coopetition between both parties.

Next, we present in detail the model capturing the VNF placement process. Notice that such decisions account for different input information (e.g., traffic demand) and are enacted on different networks; however, we are able to use a unified model for both.

A. Model elements and parameters

We have three main elements to capture our system model: (i) the traffic demand, and the (virtual) network functions that process it; (ii) the servers and the topology they form; and (iii) how we match the two.

The policy graph: This work extends the notation of policy chains used in previous works [11], [12] to a policy graph due to the limited descriptiveness of the former. Future policy requirements formulations, as envisioned by [13], [14] 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 VNFs. For each such 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. 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).

The network graph: The network is modeled as a directed graph $G_N = (V_N, E_N)$. Each edge corresponds to a *physical link*, whose *bandwidth* (or capacity) is denoted by $b(e)$. Similarly to [1], [11], 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 $r_T(m) = \infty$.

Finally, we have a *cost* $\kappa(m, v)$ we incur into when placing an instance of VNF v at server m. This is typically a monetary fee, where $\kappa(m, v) = \infty$ implies that one cannot place VNF v at server m.

B. Mixed Integer Linear Program Formulation

Finding the optimal VNF placement (namely, the placement with least cost in terms of κ) is NP-hard (proof, by reduction to SAT, is omitted). Nevertheless, we next show how it can be formulated as a mixed integer linear program, implying that existing tools (e.g., CPLEX) can be used to solve it optimally for moderate-size networks.

Variables: Our decision variables define 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.

For ease of presentation, we will also use the following two notations as auxiliary variables: $y(m, v)$ denotes the amount of traffic of VNF v that is processed by server m, and $n(m, v)$ denotes the number of instances of VNF v, running on server m. Specifically,

$$y(m,v)= \sum_{l\in\mathcal{M},u\in\mathcal{F}} x(l,m,u,v) - \sum_{n\in\mathcal{M},u\in\mathcal{F}} x(m,n,u,v), \qquad \forall m \in \mathcal{M}, v \in \mathcal{F} \quad (1)$$

where the first term in (1) is the traffic coming to server m and meant to be processed by any instance of VNF v and the second term is the amount of said traffic leaving server m. The number of V's instances on server m is simply $n(m, v) = \left\lceil \frac{y(m,v)}{p(v)} \right\rceil$. Note that the instances of VNF v on server m consume a total of $n(m, v)r_M(v)$ memory and $y(m, v)r_C(v)$ CPU capabilities.

Constraints: To ensure the feasibility of the placement and the flow, we define the following constraints. First, we have to ensure the demands that are defined in the policy graph are fully satisfied:

$$\sum_{m_1,m_2\in\mathcal{M}} x(m_1, m_2, v_1, v_2) \geq t(v_1, v_2), \qquad \forall (v_1, v_2) \in E_D, \quad (2)$$

where the left-hand side implies that the amount of all traffic that was processed by v_1 and then by v_2 (on any pair of servers) is at least as specified by the corresponding edge in the policy graph.

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

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

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

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

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

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

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

We can model switches as servers with zero capabilities; hence, (6) enables us to account for the limited capacity of the forwarding tables at switches [1], [11].

Objective: The objective function is explicitly defined by the cost matrix κ:

$$\min \sum_{m\in\mathcal{M}} \sum_{v\in\mathcal{F}} \kappa_{m,v} \cdot n(m, v). \quad (7)$$

III. Online algorithm

As our placement problem is NP-hard, it is not expected to scale for large networks and frequent changes in demand (each such change requires solving the optimization problem from scratch). Moreover, as Section II deals with one-shot solution to the problem, it does not take into account the number of *VNF migrations* occurring if two subsequent solutions significantly differ in their VNF placement.

Algorithm 1 is an online algorithm that addresses both issue. It starts with an initial solution $\bar{x}$, as defined in Section II. 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, the algorithm adapts the placement and traffic steering variables. We have a *threshold* parameter $0 \leq \alpha \leq 0.5$; the algorithm will try to free up servers with load higher than $1 - \alpha$, and shut down servers with load lower than α.

Algorithm 1 Our online algorithm.

```
Require: current load $t$, initial decisions $\bar{x}$
                        ▷ We use also the auxiliary variables $\bar{y}, \bar{n}$
compute $\lambda(v)$ as defined in (8)
while $\max_{v\in\mathcal{F}} \lambda(v) > 1 - \alpha$ do
    $v^\star \leftarrow \arg\max_{v\in\mathcal{F}} \lambda(v)$
    $\tilde{x} \leftarrow$ solve LP relaxation (2)–(7),(9)–(10)
    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 (11)
        update $\bar{x}, \lambda$
while $\min_{v\in\mathcal{F}} \lambda(v) < \alpha$ do
    $v^\star \leftarrow \arg\min_{v\in\mathcal{F}} \lambda(v)$
    $\tilde{x} \leftarrow$ solve LP relaxation of problem (2)–(7),(12)
    $m^\star \leftarrow \arg\min_{m\in\mathcal{M}} \left(\bar{y}(m, v^\star) - \tilde{y}(m, v^\star)\right)$
    $\bar{n}(m^\star, v^\star) \leftarrow \bar{n}(m^\star, v^\star)-1$
    re-balance $\bar{y}$ using (11)
    update $\bar{x}, \lambda$
return $\bar{x}$
```

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

$$\lambda(v) = \frac{\sum_{u\in\mathcal{F}} t(u,v)}{\sum_{m\in\mathcal{M}} \min\left(p(v)n(m,v), \frac{\phi(m,v)}{r_C(v)}\right)}, \qquad (8)$$

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.

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.

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 II with objective (7), integrated with the following constraints:

$$\tilde{y}(m,v) > \big(\bar{n}(m,v) - 1\big) \cdot p(v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \qquad (9)$$

$$\begin{aligned} &\tilde{y}(m,v) \leq \bar{n}(m,v) \cdot p(v), \\ &\forall m \in \mathcal{M}, v \in \mathcal{F} \colon r_M(v) + \sum_{u\in\mathcal{F}} \bar{n}(m,u) r_M(u) > c_M(m). \end{aligned} \qquad (10)$$

Constraint (9) 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 (10) 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:

$$\bar{y}(m,v) = t(v) \frac{\tilde{y}(m,v)}{\sum_{m'\in\mathcal{M}} \tilde{y}(m',v)} \qquad (11)$$

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 (9), imposing that the relaxed solution is a subset of the original one:

$$\tilde{y}(m,v) \leq \bar{n}(m,v) \cdot p(v), \quad \forall m \in \mathcal{M}, v \in \mathcal{F}. \qquad (12)$$

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 (11)) 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 (9) and (10). 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.

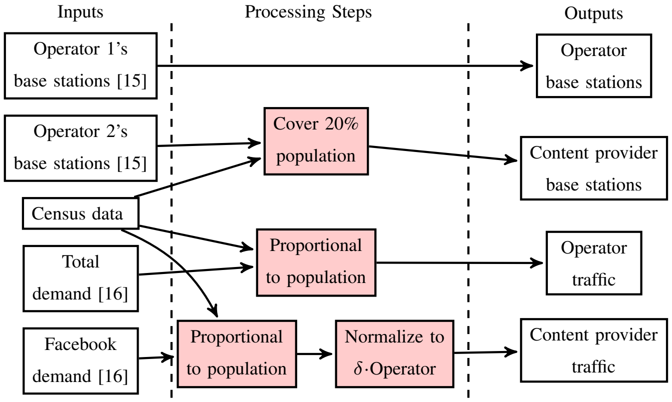


Fig. 2. The construction of our reference scenario.

IV. 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 [15];
- census data from the same country [17];
- a measurement paper [16], 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. 2 summarizes how we process our data. The full dataset of [15] includes demand (voice and data) and deployment (location, technology power class) information about a total around 5,000 base stations serving several thousands users, over a period of two weeks.

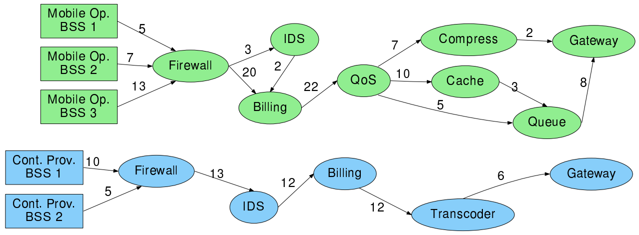


Fig. 3. The policy graph we use in our test scenario. Boxes correspond to entry points, ellipses to VNFs that the traffic has to traverse. The values on the edges is the amount of data transmitted between the VNFs on their endpoints.

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 [16] 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 [1]. 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.

Servers and traffic demand: There are $|\mathcal{M}| = 54$ servers, all with normalized capabilities $c_M = c_C = 1$. As in [1], 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. 3(a), which is similar to the processing done in today's LTE-based mobile networks. 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. 3(b), which includes several different VNFs for custom processing, inspired by [1], [14]. All VNFs have CPU and memory requirements randomly set between 0 and 1.

Content provider topology and demand: We use the topology of the second operator in our dataset to construct the content provider's network, albeit with some more manipulations. Specifically, we reduce the number of base stations, only keeping the 10% most loaded, which are sufficient to cover 20% of the population. This is consistent with the widespread belief that content providers will concentrate their coverage in the most crowded areas, instead of attempting to directly serve all their users [8].

The traffic demand follows the temporal evolution reported in [16] for Facebook. The traffic is proportional to the population served by each base station, and represents a fraction δ of the operator's demand. Unless otherwise specified, we use $\delta = 0.5$, consistent with the recent news that Netflix accounts for over one third of the total Internet traffic [18].

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.

V. Experimental results

A. Optimal decisions

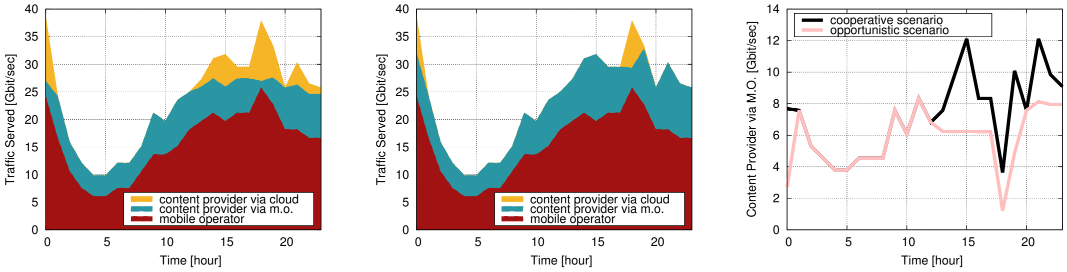
We first tackle the question what is the most the mobile operators and content providers can save by switching from a pure competition relationship to a coopetition. Thus, we have optimally solved the problem presented in Section II using CPLEX. Fig. 4(a) and Fig. 4(b) compare the opportunistic and cooperative scenarios and show that the savings mentioned above are signficant. Specifically, the red areas in the figures represent the traffic demand of the mobile operator, all of which is served through its own servers. 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. Namely, the blue area corresponds to fees that should be paid to the cloud vendor in a pure competition scenario, and 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. 4(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. 5, we move to the online case, where decisions are made through the algorithm described in Section III. Fig. 5(a) shows how the traffic is served; comparing it with Fig. 4(b), we can immediately see that the yellow area, i.e., the amount of traffic served at the cloud vendor, is larger. In Fig. 5(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.

We also compared the utilization of servers by our online and optimal algorithm. Specifically, Fig. 6 focuses on the cooperative scenario and shows that when decisions are made optimally (Fig. 6(a)) there is more CPU time devoted to the content provider's traffic and less idle time than with the online algorithm (Fig. 6(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.

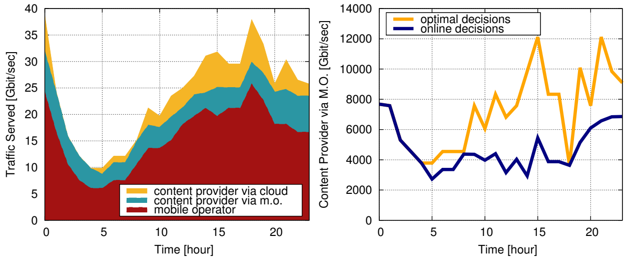
VI. Conclusion

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



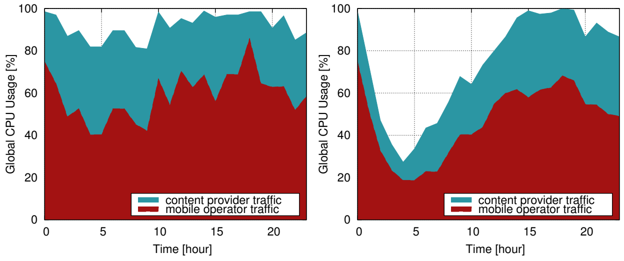
(a) (b) (c)

Fig. 4. 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)

Fig. 5. Online decisions, cooperative scenario. (a): how the traffic is served; (b): amount of content provider's traffic processed at the mobile operator's servers.



(a) (b)

Fig. 6. Cooperative scenario: CPU usage at the mobile operator's servers. (a): optimal decisions; (b): online decisions.

with the help of traditional mobile operators, which would open their SDN-based, virtualized core network to content providers in exchange for a fee.

In Section II, we modeled the twofold problem of (i) matching virtual network functions and servers and (ii) steering traffic between them. Moreover, in Section III, we proposed an online, scalable algorithm to solve such a problem.

We evaluated our performance using the real-world scenario described in Section IV. As summarized in Section V, 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 reduce the reliance of the cloud vendor by 60%, while keeping the computational complexity low.

Acknowledgement

This work was supported by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11) and European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement no. 259085.

References

[1] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *ACM CoNEXT*, 2013.
[2] T. V. de Velde. (2015) On Resource Partitioning, Network Slicing and Service Chaining. http://www.netmanias.com/ko/post/blog/8367/5g-network-slicing/on-resource-partitioning-network-slicing-and-service-chaining.
[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 Comm. Mag.*, 2012.
[4] J. Andrews, "Seven ways that HetNets are a cellular paradigm shift," in *IEEE Comm. Mag.*, 2013.
[5] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin, and M. Debbah, "When cellular meets WiFi in wireless small cell networks," in *IEEE Comm. Mag.*, 2013.
[6] S. Rangan, T. S. Rappaport, and E. Erkip, "Millimiter Wave Cellular Wireless Networks: Potentials and Challenges," in *Proc. of the IEEE*, 2014.
[7] P. Di Francesco, F. Malandrino, and 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 and 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] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *ACM SIGCOMM Comp. Comm. Rev.*, 2013.
[12] G. Gibb, H. Zeng, and N. McKeown, "Outsourcing network functionality," in *ACM HotSDN*, 2012.
[13] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *ACM CoNEXT*, 2014, pp. 271–282.
[14] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "PGA: Using graphs to express and automatically reconcile network policies."
[15] P. Di Francesco, F. Malandrino, and L. DaSilva, "Cellular Network Planning using Real Data," in *submitted*, 2015.
[16] Y. Zhang and A. Arvidsson, "Understanding the Characteristics of Cellular Data Traffic," in *ACM SIGCOMM Comp. Comm. Rev.*, 2012.
[17] "Irish CSO census data," 2011, http://www.cso.ie/en/census/census2011_boundaryfiles/.
[18] N. Arce, "Netflix Is Hogging 35 Percent of Peak Internet Traffic in North America: What About Others?" in *Tech Times*, 2014.