QoE-Aware Computing Resource Allocation for CDN-as-a-Service Provision

Louiza Yala*, Pantelis A. Frangoudis*, and Adlen Ksentini‡
*IRISA/University of Rennes 1, France
‡EURECOM Institute, Sophia Antipolis, France
Email: *name.surname@irisa.fr, ‡adlen.ksentini@eurecom.fr

***Abstract*—We focus on the provision of Content-Delivery-Network-as-a-Service (CDNaaS) functionality for video distribution and, in particular, on how to appropriately decide on the amount of computing resources to allocate to a CDNaaS instance to satisfy Quality of Experience (QoE) and resource capacity constraints. Our work is put in the context of a telco cloud environment: The telco cloud is open to content provides to request the dynamic deployment of a virtual CDN infrastructure on top of it, in order to cover a target user demand with a desired service quality at the regions where the telecom operator has presence. We perform testbed experiments to quantify the dependence of user experience on the load of a virtualized video service, and apply our measurement results to drive a QoE-aware virtual CPU resource allocation algorithm. Using simulation, we show our algorithm to optimally address the quality-cost tradeoff, demonstrating the benefits of QoE awareness.**

I. Introduction

Video traffic distributed over-the-top using massive content delivery infrastructures dominates the Internet. In order to enter the market and exploit their inherent network awareness and their proximity to end users, telecom operators are deploying their own Content Delivery Networks (CDN) at their Points of Presence (POP). Cloud technologies and recent advances in Network Function Virtualization (NFV) create the opportunities to offer CDN functionality as a service (CDNaaS) in a flexible and dynamic fashion. From the perspective of a content provider, such *Telco CDNs* can be used to complement traditional CDN services, combining the advantages of both: The (geographically limited) telco regional presence and the delivery efficiency it brings about can be coupled with the global service offered by a traditional CDN.

Our work targets this environment. We have proposed [1] an architecture for on-demand instantiation of a virtual CDN (vCDN) service on top of a network operator's private telco cloud, using NFV technologies and offering a northbound interface to customers (content providers), over which they can lease vCDN resources at specific regions where the operator has presence. Our scheme allows content providers to express service demand specifications and Quality of Experience (QoE) constraints, which can be used to decide on an initial virtual resource allocation for the specific vCDN instance.

We argue that QoE-awareness is critical to this end. Suboptimal allocation of virtual resources for a vCDN deployment request may lead to situations when user demand cannot be responded to with the desirable quality (e.g., when the number of CPU resources allocated is not capable of handling the load of many parallel video streams), or, conversely, to over-provisioning, when more resources than necessary are utilized, leading to increased cost for the operator.

In this direction, we make the following contributions. In order to be able to perform informed resource allocation decisions, we carry out extensive measurements of an HTTP video service running on top of a virtualization platform to measure the capabilities of the latter from a QoE-centric viewpoint (Section IV). Our measurements serve to derive an expression of user experience as a function of video server load. This relationship is then used by a measurement-driven resource allocation procedure which aims to cover a target end-user demand by minimizing the volume of the computing resources allocated, while respecting (customer) QoE and (operator) capacity constraints. Using simulations on a real POP topology, we demonstrate the advantages of QoE awareness and the importance of our empirical measurements (Section VI). Our algorithms (Section V) achieve their goals while being efficient in terms of running time.

II. Related work

Network Function Virtualization (NFV) technology came with several benefits, such as improving the flexibility of network services, and reducing capital investment and energy consumption by consolidating networking appliances [2]. One of the challenges in NFV is appropriate VNF placement. Clayman et al. [3] propose a VNF management and orchestration framework and experiment with different virtual router placement algorithms. In a similar spirit, Moens and De Turck [4] propose a theoretical model for resource allocation of virtualized networks within an NFV environment. These resources include both physical hardware and virtual instances, and the problem tackled is that of placing VNFs to handle service requests dealing with load bursts.

There are several works on Virtual Machine (VM) placement in a cloud computing environment, treating the problem from different angles, considering specific cost, availability, and performance-related criteria and constraints, which pertain

This work has been partially supported by the French FUI-18 DVD2C project.

both to the network and the rest of the cloud infrastructure. For instance, Piao and Yan [5] propose an approach for VM placement and migration to minimize the time consumed in data transfers. Rabbani et al. [6] focus on mapping Virtual Machines (VMs) to physical servers, with the aim of improving server resource (e.g., CPU or memory) utilization, overcoming the lack of space in data centers and maximizing the number of mapped VMs in one physical host. Aggarwal et al. [7] propose placement goals and constraints, which help the cloud provider address the virtual machine placement problem, and highlight some techniques and methods to solve it. Other works [8] propose to overcome the VM placement problem by grouping together VMs that have the same behavior (in terms of resource usage), using existing solutions such as a bin packing model, etc.

Compared with the state of the art, our work aims at placing cache VNFs (playing the role of local HTTP video streaming servers) following customer demand in the regions where the operator has presence, and mainly addresses the virtual resource allocation aspect per VNF instance. We therefore deal with a service dimensioning problem for dynamic CDNaaS deployment for video distribution. Our distinctive characteristic is that virtual resource allocation is demand- and QoE-driven: We derive an empirical model of the relationship between QoE and service load for an HTTP video streaming application (a popular CDN-based service), and exploit it to optimally decide on the amount of processing resources to deploy, satisfying QoE and operator capacity constraints.

III. A CDNaaS architecture for the telco cloud

In our previous work [1], we presented an architecture which allows a network operator to virtualize its CDN infrastructure and lease it to content providers on demand. We highlighted the fact that our design offers flexibility by allowing the customer to, for example, use the leased infrastructure to respond to predicted traffic surges at particular regions, enjoying the network operator's regional presence. On the network provider side, our concept allows more efficient use of its infrastructure resources, compared to a less dynamic resource reservation model with static allocation of data center resources to customers.

Our CDNaaS architecture (Fig. 1) includes various functional blocks which communicate via well-defined interfaces. This decouples their operation from any physical location, allowing the CDNaaS provider to execute any block autonomously as a virtual function over its own, or any, cloud infrastructure.

Via the Customer Interface Manager component, our system exposes a northbound RESTful API to customers (content providers), who can request to deploy a virtual CDN over the telco cloud. This API offers the customer a way to specify details on the expected end user demand for its service per region, and, importantly, a target QoE level.

As we shall show, this information is critical for service deployment, a procedure which is coordinated by the Service Orchestrator (SO) component. In particular, the SO needs to

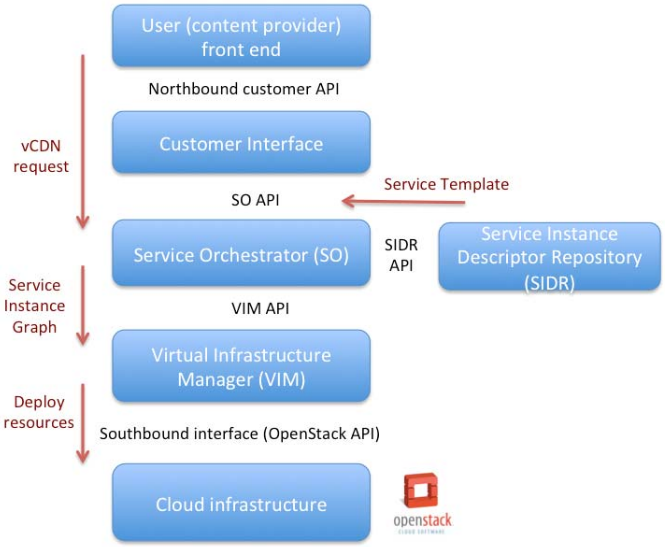
User (content provider) front end
Northbound customer API
Customer Interface
vCDN request
Service Template
SO API
Service Orchestrator (SO)
SIDR API
Service Instance Descriptor Repository (SIDR)
Service Instance Graph
VIM API
Virtual Infrastructure Manager (VIM)
Deploy resources
Southbound interface (OpenStack API)
Cloud infrastructure
openstack

Fig. 1. CDNaaS architectural components [1].

derive an appropriate VNF placement, taking into account the customer demand and quality specifications included in the service request, and its operational capacity. At the heart of this VNF placement procedure is a resource dimensioning algorithm which decides on the optimal allocation of vCPU resources across regions, aiming to satisfy the constraints dictated by the customer request (demand, quality) and the operator's capacity. In our prior work, we had not specified such algorithms, but had created the necessary support in our software architecture and our prototype for the application of a variety of them. In this work, we move on to propose such an algorithm (Section V). Its output, i.e., the amount of vCPU resources per region, is then translated to a Service Instance Graph (SIG), which maps VNF instances (VMs) and the respective resources allocated to each one of them to physical nodes. The SIG is eventually passed on to the Virtual Infrastructure Manager (VIM) for deployment on the underlying cloud infrastructure (in our case, managed by OpenStack [9]).

A CDN service is typically composed of one or more origin servers, where the content provider places original content items, and a number of caches distributed across the CDN, where content is replicated. User requests are redirected (e.g., via DNS geolocation, as in our prototype) to the optimal cache, which temporarily stores content replicas and efficiently serves them to users to cut down on delivery times and reduce the load on the origin servers. In our case, caches are implemented as VNFs.[1]

Upon receiving a customer request for the creation of a

[1] Our system supports various CDN flavors. Typically, caches, HTTP request load balancers and DNS servers are deployed as VNFs, and the origin servers are hosted outside the vCDN (e.g., in the content provider's premises, or in another public/private cloud.). In a full-fledged vCDN version, though, origin servers are also launched as VNFs in the operator's cloud infrastructure and the appropriate entry point to them is returned over the northbound customer API, so that the customer can inject content in the CDN.

vCDN, the SO runs the resource allocation algorithm (see Section V) to calculate a cache VNF instance placement and the number of vCPUs to allocate on the appropriate regional data centers in the form of a SIG, and uses the VIM API to request its deployment on the telco cloud infrastructure. The VIM is responsible for automatically configuring the DNS VNF instances for request geolocation, and for setting up cache VNF instances so that they proxy all user requests towards the content origin server which is typically outside the vCDN (location included in the customer request). The vCDN service is terminated either on demand, using the northbound customer API, or automatically, when the customer lease on the resources expires.

IV. User experience as a function of service load

In this section, we attempt to quantify the relationship between server load and user experience. To this end, we measure the performance capabilities of an HTTP video server running on a virtualization platform, on a single CPU and in the presence of multiple parallel video sessions. We use a vanilla nginx server [10], which serves user requests for video content prepared for Dynamic Adaptive Streaming over HTTP (DASH) delivery. Under DASH technologies, the client receives a Media Presentation Description (MPD) file, which includes information on the available representations (different qualities) of the same video, which is segmented in chunks. Afterwards, the client proceeds to download the video chunk-by-chunk, potentially switching among the available video qualities (and thus bitrates) to better adapt to network conditions.

A. Measurement methodology

Our QoE metric is the Mean Opinion Score (MOS), i.e., the expected rating that a panel of users would give to the quality of the transmitted video in the 1-5 (poor-excellent) scale. To estimate it we apply the Pseudo-Subjective Quality Assessment (PSQA) approach [11]. PSQA consists in training a Random Neural Network (RNN) based on experiments with physical subjects under controlled conditions, where a set of parameters affecting quality is monitored and the ratings of users are recorded. The trained RNN classifier can then be applied in real time and output the expected MOS for specific values of the input parameters. Singh et al. [12] have applied PSQA to estimate QoE for H.264/AVC-encoded video delivered over HTTP, and we are using their tool in our experiments.

The PSQA tool operates on 16-second video segments. As our test video sequences are larger (475 s), we calculate one MOS value for each 16-second window. Since user experience at a specific time instance is not independent of the one at the previous one, we maintain a moving average of the calculated MOS, in order to take into account recency effects, according to the following equation:

$$MOS_{i+1} = 0.3MOS_i + 0.7s_{i+1},$$

where MOS_i is the moving average calculated up to window i and s_{i+1} is the MOS sample calculated for the $(i+1)$-th window.

The input parameters (appropriately normalized; see [12] for details) for the QoE estimation tool, are the following:

- The number of interruptions in the 16-second window.
- The average and the maximum interruption duration.
- The average value of the Quantization Parameter (QP) across all picture macroblocks in the measurement window. QP is an indication of picture quality (the higher the QP, the lower the video bitrate and quality).

To acquire the necessary input, we extended the open-source vlc media player [13] with specific monitoring functionality, i.e., with the ability to record QP values and playout interruptions and their duration. Furthermore, we modified the vlc DASH plugin [14] to disable video rate adaptation and always serve the video representation with a selected bitrate.

We make the following assumption with respect to QoE calculation: If at a specific 16-second window there is an interruption which is longer than 8 s (half of the measurement window time), we consider that the MOS is 1.0 (minimum value possible). This is because the RNN used shows saturation when quality is very bad and its training focused on being more accurate when quality is good, at the expense of inaccuracies when interruptions are very large or very frequent.

Our test video is a 475-second, H.264/AVC-encoded 720p sequence, with a bitrate of 1.7 Mbps, which we prepared for DASH delivery using the DASHEncoder tool [15]. Each chunk carried 2 s of video and the mean chunk size was 435 KB. To emulate multiple simultaneous viewers/streams, we generated parallel HTTP sessions using a modified version of the wrk HTTP load generator which allowed for selecting the request rate [16]. Each of the parallel wrk connections was downloading a 435 KB file at the video rate (i.e., 1 chunk/2 s).

The HTTP server was hosted on a kvm [17] virtual machine[2], which we had pinned to a single CPU core using the taskset utility. Since we were interested in measuring the pure effects of server processor load on QoE, we needed to make sure that network I/O was not a performance bottleneck. Therefore, we activated the vhost-net module, which enables in-kernel handling of the packets between the guest VM and the host. This ensured that the bandwidth available for host-to-guest communication was adequate (more than 30 Gbps aggregate TCP throughput via two virtual ethernet interfaces, as reported by the iperf tool). Simultaneously, to make sure that the CPU load on the traffic generator was not a bottleneck, we pinned it on a set of 8 CPU cores, sharing the load among 8 threads.[3]

At each experiment, repeated a number of times to acquire the necessary number of QoE samples for statistical

[2] This setup also corresponds to our OpenStack-based CDNaaS prototype.

[3] We also experimentally verified that disk I/O was not a bottleneck, since the file downloaded by the load generator sessions was served from the disk cache. We also repeated our experiments with the web server configured to serve files from a memory-mapped tmpfs file system, and our results were unaffected.

confidence, we varied the number of parallel emulated video sessions, while at the same time recording QoE-related information on a vlc instance accessing a DASH video from the server.

B. Results

With the aim of empirically associating video server load (in terms of parallel video streams) and user experience, we varied the number of parallel video sessions from 2000 to 12000. Fig. 2 presents the average MOS value observed across all 16-second video samples for each case. We observe that a vanilla nginx server can sustain up to more than 5000 parallel HD video sessions with an excellent quality.

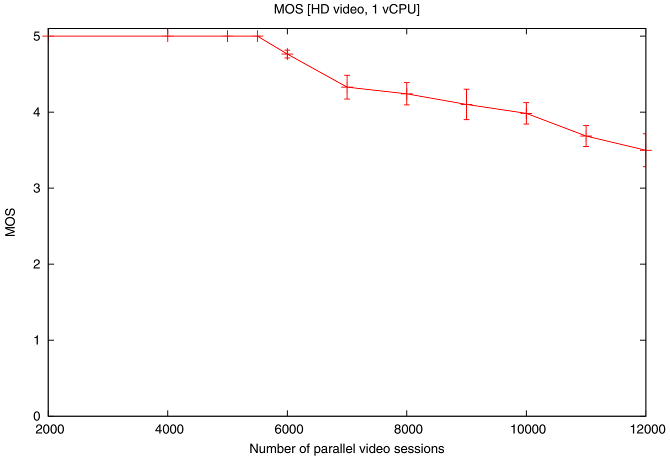

Fig. 2. Average QoE as a function of the number of parallel HD video streams on a web server hosted in a kvm VM using 1 CPU core. Each point is the mean of a few hundreds of QoE samples (MOS values), presented with 95% confidence intervals.

From a user experience perspective, apart from the average QoE observed across all 16-second video samples per case, another interesting metric is the ratio of viewing time that QoE is above a specific threshold (i.e., the ratio of samples in which QoE is above this threshold). This could be encoded in an SLA of a different format, where the customer wishes to ensure that end users will enjoy an acceptable QoE level for more than a specific percentage of the video viewing time. To evaluate this metric we resort to the empirical cumulative distribution function of the QoE samples collected in our experiments, examples presented in Fig. 3. Table I summarizes our results in terms of (i) MOS and (ii) the probability that QoE for a 16-second sample exceeds a MOS threshold of 3.6, which, in various contexts (e.g., VoIP), is considered acceptable quality. For loads of 6000 parallel users or more, video interruptions start to take place, which reduce QoE, especially as load grows. The last two columns of Table I correspond to the duration and the frequency of such playout interruptions. These follow an increasing trend with server load, reaching, on average, more than 7 s and 2.5 per minute respectively.

TABLE I
SUMMARY OF RESULTS. QOE AND PLAYOUT INTERRUPTIONS AS A FUNCTION OF THE NUMBER OF PARALLEL USERS (STREAMS) ACCESSING A VIDEO SERVICE HOSTED ON A SINGLE-CORE KVM-BASED WEB SERVER/CACHE VNF.

# parallel streams	MOS (mean)	Acceptable QoE ratio	Mean interruption duration (s)	Interruption freq. (/min)
2000	5.00	1.0	0.0	0.0
4000	5.00	1.0	0.0	0.0
5000	5.00	1.0	0.0	0.0
5500	5.00	1.0	0.0	0.0
6000	4.76	1.0	2.99	1.90
7000	4.34	0.89	5.36	2.43
8000	4.24	0.85	6.54	2.47
9000	4.10	0.81	6.61	2.62
10000	3.98	0.76	7.26	2.47
11000	3.69	0.69	7.34	2.79
12000	3.50	0.62	8.55	2.78

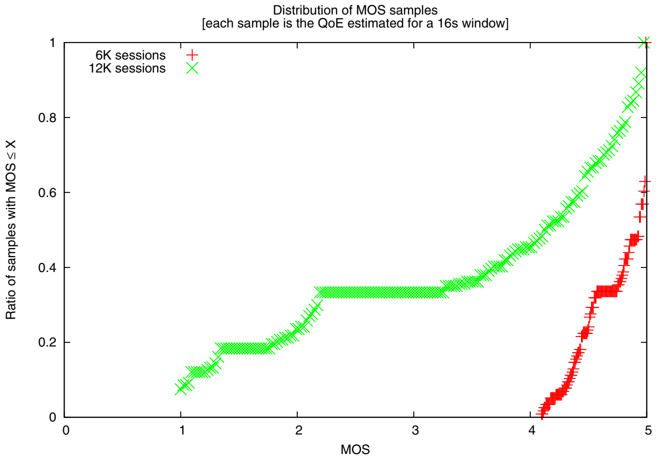

Fig. 3. Empirical CDF of the QoE of video samples for two different server loads.

V. A QOE-AWARE VIRTUAL CPU RESOURCE ALLOCATION ALGORITHM

In this section, we show how experimental results on video QoE vs. service load can be used to derive a computing resource allocation for a vCDN instance.

A. System model

We assume that the service provider (SP) has presence in N regions and a content provider (CP) wishes to deploy a virtual CDN over a subset of them. The CP (customer) request includes the demand $D = \{d_1, d_2, ...d_N\}$, where d_i is the maximum number of users simultaneously accessing the service[4] in region i, as well as the respective target quality specification $Q = \{q_1, q_2, ...q_N\}$, where q_i is the desired QoE[5] for end users in region i. The SP needs to decide on the appropriate number of CPUs to allocate per region to satisfy

[4] To express demand we use the terms "users" and "video streams" interchangeably, implying one video stream per user terminal.

[5] Depending on the scenario, this could be expressed in terms of the expected MOS or the percentage of viewing time when QoE is higher than a specific quality threshold. Our algorithm is equally applicable to both cases.

the demand with the requested QoE. Our resource unit is a virtual CPU (vCPU) which in practice corresponds to a single CPU core.

We formulate the problem as an optimization one. The objective is to find an assignment $X = \{x_1, x_2, ...x_N\}$ which minimizes the sum of the number of vCPUs x_i allocated to each region i following a customer request, taking into account that (i) the quality delivered to the end user must be better or equal to the quality that the customer requested for, and (ii) the number of vCPUs deployed must not exceed the per-region capacity $C = \{c_1, c_2, ...c_N\}$, where c_i is the number of vCPUs available at the data center of region i. Therefore, our problem is formulated as follows:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \sum_{i \in N} x_i && (1) \\ \text{subject to} \quad & x_i \leq c_i \qquad \forall i \in N && (2) \\ & U(x_i, d_i) \geq q_i \qquad \forall i \in N, && (3) \end{aligned}$$

where $U(x_i, d_i)$ is the expected QoE for the end users in region i when the demand is d_i and x_i vCPUs are allocated to cover it. We further make the assumption that if the requested resources for a single region exceed its capacity, then the problem has no solution. Table II summarizes our notation.

TABLE II
SUMMARY OF NOTATION

N	Number of regions
$X = \{x_1, x_2, ...x_N\}$	Number of vCPUs allocated per region i
$Q = \{q_1, q_2, ...q_N\}$	Requested quality specification per region i
$D = \{d_1, d_2, ...d_N\}$	Demand specification (maximum number of parallel streams) per region i
$C = \{c_1, c_2, ...c_N\}$	Resource capacity vector (available vCPUs per region i)
T	Load-Quality matrix

B. An algorithm for the minimum cost solution

In this section, we present an algorithm to solve (1), i.e, to find the minimum sum of resources to serve a customer request under capacity (2) and quality (3) constraints. Apart from the demand and quality specifications and constraints, the input to our algorithm is T, a 2-dimensional matrix with the format of Table I. The first column (load) contains the number of parallel video streams handled by a single vCPU, and the second (quality) contains the QoE that can be achieved under this load. We derive this table from measurements (Section IV).

Our algorithm operates as follows: For each region, the minimum number of resources necessary so that the quality constraint is not violated is calculated. For this, the entry j in T which corresponds to the maximum number of parallel streams that can be handled without violating this constraint is located (line 4). Then, the algorithm calculates the number of vCPUs for region i by dividing the demand value (customer demand) by the load value of entry j (line 5) and rounding the result up (we do not allow fractional resources or load sharing across regions). The intuition is that since we aim at minimizing cost, we wish to pack as many streams as possible on a single vCPU; Line 5 guarantees that $U(x_i, d_i) \geq q_i$. Line 6 will ensure that the capacity constraint is not violated.

The number of vCPUs obtained is the minimum that can be deployed under the specific problem settings and assumptions. If in a single region we reduce by 1 the number of vCPUs, the demand cannot be handled and the quality constraint is violated, while if we add 1 vCPU, the derived solution will be suboptimal, given that our algorithm aims only at satisfying quality constraints, and not at optimizing for quality.

Algorithm 1 Minimum total number of vCPUs

```
1:  INPUT : $N$, $Q$, $D$, $T$
2:  OUTPUT : $X = \{x_1, x_2, ...x_N\}$    ▷ Assignment which minimizes $\sum_{i=1}^{N} x_i$
3:  for each region $i$ do
4:      $j = \arg\max_k \{T_{k,1} \mid T_{k,2} \geq q_i\}$ ▷ Row in T with the maximum load not violating the quality constraint
5:      $x_i = \lceil d_i / T_{j,1} \rceil$;
6:      if $x_i > c_i$ then    ▷ Capacity constraint
7:          return NULL    ▷ No solution exists;
8:      end if
9:  end for
10: return $X$
```

VI. Performance evaluation

To demonstrate the importance of QoE awareness in resource dimensioning decisions, we compare our proposed mechanism for vCPU allocation with a QoE-unaware baseline strategy. This strategy allocates a fixed number of resources for a given service demand, i.e., it allocates one vCPU per F streams. For example, if the customer demand for region i is $D_i = 4000$ simultaneous streams and $F = 1000$, the baseline will allocate 4 vCPUs. Since this mechanism does not take into account the amount of resources necessary to satisfy the customer demand with a specific quality level, it can lead to under- or over-provisioning. In other words, depending of the value chosen for F by the operator, it may either lead to the allocation of fewer resources than necessary to attain the customer's desired quality level, or to an excessive amount of resources, which will lead to unnecessary operational costs.

Our algorithm, on the other hand, optimally addresses this quality-cost tradeoff, being aware of the relationship between load and user experience. To demonstrate this advantage, we simulate both strategies on a real PoP topology. We use the publicly available POP map of the Greek Research and Technology Network (GRNET) [18], which has presence in 49 cities, and assume that the operator has deployed data centers with a specific vCPU capacity at each one of them, which can host CDNaaS VNF instances. In our simulation, the capacity of each region ranges from 100 to 900 vCPUs. We vary the value of F for the baseline mechanism and calculate the total number of allocated vCPUs for a specific customer demand (number of parallel video streams/users), which is selected, for each region, uniformly at random from the $(500, 20000)$ range. Note that the baseline scheme does not take into account the customer quality constraint, which is fixed to 4.5 for all regions. Then, for the same customer request (same demand/quality specification), we run our algorithm.

Fig. 4 shows the results of our simulations. The x-axis represents the total number of vCPUs allocated (cost), and the y-axis represents the respective MOS (quality). The square point corresponds to the outcome of our algorithm (1210 vCPUs in total, resulting in an average MOS value of 4.76 across all regions), while the cross points represent the performance of the baseline scheme, starting from $F = 12000$ and down to $F = 1000$ users/vCPU. (Note that the value of F, i.e., the number of video streams served per vCPU is not shown in the figure.) The flat line is the quality constraint. The figure shows that our algorithm outperforms the baseline scheme: The latter achieves a comparable level of quality only for $F < 6000$, when it decides to allocate at least 1612 vCPUs. Conversely, the baseline scheme satisfies the quality constraint only for $F < 7000$, in which case it allocates 1397 vCPUs.

The baseline scheme performance for decreasing numbers of users per vCPU reflects the tradeoff between quality and cost: The more the resources to allocate to handle a specific load, the better the quality. Our QoE-aware algorithm addresses this tradeoff optimally, offering both user-centric (improved QoE) and operator-centric (less resources) benefits.

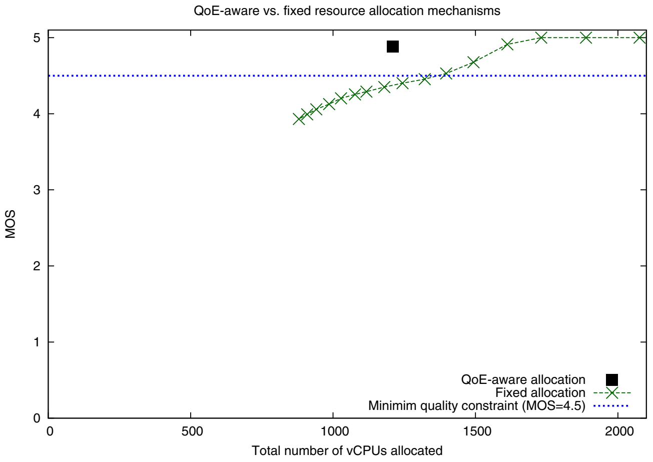

Fig. 4. Comparison of a QoE-aware with a fixed resource allocation mechanism. Depending on its configuration, the baseline scheme may lead to under- or over-provisioning. Our QoE-driven approach, on the other hand, derives an assignment which minimizes cost respecting the quality constraint.

Our algorithm also runs efficiently. For the GRNET topology (318 POPs/regions), it takes less than 10 ms to calculate the minimum vCPU assignment when executed on a single CPU core of an Intel Xeon E5-1603 processor.

VII. Conclusion and future work

We focused on the on-demand virtual CDN service deployment over a telco cloud infrastructure for the delivery of video content. Building on our prior work where we developed an NFV-based CDNaaS architecture, we addressed issues of VNF placement and resource allocation. We proposed a QoE-aware method to optimize the amount of vCDN resources allocated across the regions where the telco cloud operator has presence, in order to serve a content provider's vCDN deployment request under quality and capacity constraints. Our mechanism is measurement-driven, utilizing the results of experiments that we have carried out to quantify the relationship between user experience and server processor load. The importance of QoE-awareness in effectively addressing the tradeoff between service quality and cost is demonstrated by simulation. Our ongoing and future work focuses on enhancing our mechanisms so that they can more efficiently respond to demand dynamics, by means of real-time load and QoE monitoring. Importantly, we are revisiting various of our design assumptions and study different settings of the resource allocation and VNF placement problems.

References

[1] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, "An architecture for on-demand service deployment over telco cdn," in *Proc. IEEE ICC*, 2016.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 90–97, 2015.

[3] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *Proc. IEEE NOMS*, 2014.

[4] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. IFIP CNSM*, 2014, pp. 418–423.

[5] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing, 2010 International Conference on*. IEEE, 2010, pp. 87–92.

[6] M. G. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, 2013, pp. 177–184.

[7] A. Aggarwal and S. Malhotra, "Goals and constraints for devising efficient heuristics in virtual machine placement plan," in *Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on*. IEEE, 2015, pp. 98–103.

[8] C. Canali and R. Lancellotti, "Exploiting classes of virtual machines for scalable iaas cloud management," in *Network Cloud Computing and Applications, 2015 IEEE Fourth Symposium on*, 2015, pp. 15–22.

[9] OpenStack - open source cloud computing software. [Online]. Available: https://www.openstack.org/

[10] nginx. [Online]. Available: http://nginx.org

[11] G. Rubino, "Quantifying the Quality of Audio and Video Transmissions over the Internet: The PSQA Approach," in *Communication Networks & Computer Systems*, J. A. Barria, Ed. Imperial College Press, 2005.

[12] K. D. Singh, Y. H. Aoul, and G. Rubino, "Quality of experience estimation for adaptive HTTP/TCP video streaming using H.264/AVC," in *Proc. IEEE CCNC*, 2012.

[13] Vlc media player. [Online]. Available: http://www.videolan.org/vlc/

[14] C. Müller and C. Timmerer, "A VLC media player plugin enabling dynamic adaptive streaming over HTTP," in *Proc. ACM Multimedia*, 2011, pp. 723–726.

[15] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proc. 3rd ACM MMSys*, 2012, pp. 89–94.

[16] A constant throughput, correct latency recording variant of wrk. [Online]. Available: https://github.com/giltene/wrk2

[17] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "kvm: the Linux virtual machine monitor," in *Proc. Linux Symposium*, 2007.

[18] Greek Research and Technology Network - Network Topology. [Online]. Available: https://mon.grnet.gr/pops/