An Adaptable Storage Slicing Algorithm for Content Delivery Networks

André Moreira, Ernani Azevedo, Judith Kelner and Djamel Sadok
Federal University of Pernambuco
Recife, Brazil
{andre, ernani, jk, jamel}@gprt.ufpe.br

Arthur Callado
Federal University of Ceará in Quixadá
Quixadá, Brazil
arthur@ufc.br

Victor Souza
Ericsson Research
Stockholm, Sweden
victor.souza@ericsson.com

***Abstract*—Several works study the performance of Content Delivery Networks (CDNs) under various network infrastructure and demand conditions. Some go even further while considering demand elasticity by leveraging on Cloud elasticity where each Content Service Provider (CSP) is served only the amount of storage space and network throughput that it needs and pays accordingly. Many of today's CDN implementations are based on these proposals. However, the allocation of virtual servers in cloud computing is not yet very fine-grained and suffers from limitations and time constraints for reallocation. This work proposes an efficient cache slicing mechanism that frees the CDN provider from dedicating a different virtual surrogate for each CSP in each location. This is achieved by dealing internally with the allocation space for each CSP in a shared surrogate, while still providing services in terms of network storage allowing a CSP to select and pay the amount of content that can be stored. We show that the proposed approach deals better with peak surrogate usage, and reduces the total network traffic by around 15% and cross-traffic by as much as 35% under some scenarios.**

Keywords—cloud-oriented CDNs; storage slicing

I. INTRODUCTION

A Content Delivery Network (CDN) is an overlay distribution system that allows content providers to offer a faster way to deliver their services to a large number of users, overcoming the inherent limitations of the Internet in terms of user perceived Quality of Service [1]. According to CISCO's [2] forecast, due to the popularity of video streaming, CDN will prevail as the dominant technology for content delivery. In the same study, CISCO says that CDNs will carry 51% of all Internet traffic and 65% of all Internet video traffic in 2017.

The prices offered by CDNs are far from democratizing content delivery [3]. As a result, it is enticing to utilize cloud resources to build a so-called cloud-oriented CDN [4]. It merely uses the cloud to map the CDN infrastructure elements into virtual components across the cloud. The cloud is a conceptual layer on the Internet, which turns all available software and hardware resources and services transparent, making them accessible through a well-defined interface [5]. The heterogeneity and reusability of cloud components can be integrated to support new and powerful applications, leveraging business value at a lower cost [6].

Storage cloud providers operate data centers that can offer Internet-based content storage and delivery capabilities with the assurance of service uptime and end user perceived service quality [7]. These capabilities result in two new opportunities. The first is to allow a small business to become a customer of multiple providers in different locations or countries, building a dynamic overlay to take advantage of competitive prices at specific locations and in certain moments. The second business advantage is allowing small players to become CDN providers, offering content delivery without the cost of owning geographically distributed data centers (e.g., MetaCDN [3]).

The proposed approach is similar to the latter business case. Like cloud-based CDNs, we study the potential to build an adaptable overlay with a topology that may be different from the underlying network. We propose a model for a CDN that considers diversity, in which each type of content or content provider can be handled by "virtual CDNs", and can build on top of a traditional CDN or a cloud environment. Our main contributions are:

- To present a model that enhances content delivery capabilities, considering a plurality of content service providers, each one with its own content diversity, incorporated into an overall CDN infrastructure.
- To make a comparison between our proposed model and the traditional approach showing that ours could reach valuable gain in terms of network traffic, especially costly network cross traffic.

The rest of this paper is organized as follows. A research on cloud-oriented CDN architecture and similar detection strategies is first presented in Section II. Section III, describes the model for virtual CDN architecture. The cache slicing solution is detailed in section IV, and section V descants the experience obtained from the experiments. Finally, section VI draws some relevant conclusions.

II. RELATED WORK

MetaCDN [3] is a cloud-oriented CDN system that is commercially available. It offers a web portal allowing users to define different options related to cost and QoS by means of redirection policies. The research in [8] puts forward an architecture for virtual CDN based on cloud. It uses a Hadoop Distributed File System for cloud infrastructure services, and creates data clusters around the world. Jin et al. [9] introduced a conceptual architecture for Content Delivery as a Service (CoDaaS). It enables on-demand virtual content delivery service (vCDS) overlays for User Generated Content providers to deliver their content to a group of designated consumers. The proposed CoDaaS solution is built on a hybrid media cloud, and offers an elastic private virtual content delivery service with an agreed Quality of Service (QoS).

Moreira et al. [10] proposed the virtualization of a VoD CDN, allowing, programmatically modification in a CDN infrastructure designed for video distribution, in order to adapt it to new operating conditions. Li et al. [11] proposed a hybrid VoD delivery service that combines the traditional CDN with a cloud-assisted deployment, where user requests are partly served by the self-owned servers and the cloud. Authors claim saving as much as a 30% bandwidth. Another hybrid system is presented in [12]. Its authors studied the trade-offs of peer assisted and cloud-based CDNs with service time guarantees.

Chen et al. [7] investigated the problem of building distribution paths and placing Web server replicas in order to minimize the cost while satisfying the QoS requirements of end-users. They enhanced previous works on the cache location problem [13], formulated the problem as an Integer Program and provided greedy heuristics as its solution. Another cost model is presented in [4]. The authors considered a multi-provider cloud environment and were concerned with intra and inter cloud communication costs. Special heuristics were adopted to provide cost efficient splitting of content distribution among cloud providers.

Finally, reference [14] proposes a dynamic allocation scheme for a hybrid system that considers content workload prediction. Its authors formulate the problem of allocating content as a finite horizon dynamic decision and provide a computationally efficient approximation. Using traces of commercial CDNs, their solution yielded a gain of up to 50%.

III. The Virtual CDN Architecture

Usually, the CDN infrastructure consists of a set of surrogates scattered over the edge of the network, routers and other network elements, an origin server, a request redirector, a monitoring system, and an accounting mechanism. At least three main actors can be distinguished in a CDN:

- Content provider (CSP): it is the owner of content. It can be either the producer of content or an entity that holds it to serve the end users. It is a customer to the CDN provider, a customer that wants to lower its infrastructure costs and hires the CDN service to have content placed on caches. The content provider controls the relationship with the end users and does not usually deal with ISPs.
- CDN provider: it is the infrastructure owner. It must provide a set of services and functionalities [15] and charges its customers (content providers) according to the content delivery to end users. Thus, it must monitor and collect usage information to support billing, but also to perform maintenance operations and service expansion. The relationship between the CDN provider and the CSP is regulated by specific SLAs.
- End users: They are the ones who ultimately access the services provided by the CDN provider. These services can be either free or paid, and are determined by the content provider. The CDN provider is transparent to end users.

The content of a CDN can be very diversified, which may lead to different SLA requirements. Lu et al. [16] showed that differentiated caching services are necessary to cope with the increasing heterogeneity in Internet clients and content. That raises the question of how many CDN architectures would be needed for use by a provider [17].

A first answer would be to have one architecture shared by all content providers or content types. The problem is that the presence of multiple content providers may result in several inefficiencies. For instance, differentiated caching services can be problematic: each set of policies and strategies aimed at a specific type of content or QoS requirements must have an allocated storage size. As a result, deciding the size that will be attributed to each type can be very tricky. Furthermore, it is common for a given CSP to experience content popularity periods (above average content requests) separated by and unpopular periods, even within a day [18]. This leads to cache usage inefficiency.

The uneven assignment of resources leads to setbacks that deplete the end user's QoE. In addition, resources may get idle (with associated maintenance costs). The combination of such "voids" in a certain domain and an unexpected increase in resource demand in another one, may harm the overall performance of the cache, possibly causing a breach of SLAs.

Thus, the second answer would be to have as many architectures as the number of applications. An application often targets a specific content provider or a specific content type. Each content type has its corresponding set of policies and algorithms that can handle it in order to satisfy its QoS requirements. The cloud-oriented architecture seems to be appropriate to solve the problem of deciding the amount of space for each application. The solution is to use the virtualization provided by the cloud. Each application can have its own virtual CDN. A surrogate can be seen as a Virtual Machine (VM) or virtual server, provided by the cloud infrastructure, with on demand resource reallocation.

This resource reallocation, however, may encounter some pitfalls. The cloud storage does not have infinite capacity; thus, in a multi-tenant environment, the cloud may not have sufficient resources to satisfy the demand of all of its virtual CDNs. Likewise, the need for storage space demanded by each application may exceed the capacity of the cloud storage. As a result, the CDN provider must decide how much space to allocate to each application, leading to the same problems faced by the first approach. Furthermore, the live re-assignment is usually a time consuming task, and may also take the cache to an unprepared state for future reallocations, possibly breaching the SLA.

To solve these problems, we present the slicing approach, which, differently of a traditional cloud-based approach, frees the CDN provider from dedicating a different virtual surrogate for each CSP. This is achieved by dealing internally with the allocation space for each CSP in a shared surrogate, while still forming the "virtual CDN" for each one, as in cloud-based CDN, but avoiding the cloud management or granularity constraints of a VM allocation.

IV. The Cache Slicing Solution

The proposed cache slicing approach aims at solving the allocation problem discussed, allowing the CDN provider to decide the amount of space to be allocated to each tenant and indicates where this should be allocated. This process considers

aspects such as the limitation of resource availability and associated costs. Our approach also simplifies the concept of virtual CDN. There is no need to have multiple VMs, each one serving a tenant (CSP). Instead, one VM (or a physical machine) can serve multiple tenants. The storage is divided into **slices** and each one is eventually assigned to a tenant. Thus, the resulting problem can be summarized by the following question: how do we distribute the storage space hired by the CSP throughout all available replica servers?

A. Overview

We start with storage partition within a CDN provider. In this initial scenario, we consider a contracting process between the content provider and the CDN in which the former purchases storage space for its objects, and the latter undertakes to distribute this space among its caches in order to give the best possible distribution that attends end users SLAs. The technical knowledge of the process of content delivery remains with the CDN. The content provider cannot determine in which cache content will be effectively stored.

The only way for the content provider to interfere in the delivery process is indirectly, through an SLA definition. An SLA can define the QoS requirements to be met by the CDN, the penalties for non-compliance, rules regarding the geographical distribution and property rights, content replication, and security etc..

The contracting process can be performed basically in two ways: the CSP defines the storage or the CSP defines the SLA. In the first model, since content providers decide how much storage they intend to hire and it directly interferes with the plausibility the of QoS attendance, the CDN will define or negotiate the SLA with which it is possible to comply. The second model is just the opposite. The CSP presents the QoS requirements and the CDN determines the amount of storage needed.

B. Problem formulation

We model the problem as a modified capacity allocation problem as described in [19], relaxing the topology tree constraint. We extend the model to an arbitrary number of CSPs in the multi-tenant environment as described earlier, and introduce the cache slicing approach by including the proper constraints.

The input consists of the following: a set of content objects, first assumed to be unit-sized, $\mathcal{O}$; an available storage capacity $\mathcal{S}$ of storage units, the same measure of objects; a set of m clients, $\mathcal{J}$, each client j has a distinct request rate λ_j and a distinct object demand distribution $p_j: \mathcal{O} \rightarrow [0,1]$; where 1 represents the demand of the object. The input also contains a tree graph $\mathcal{T}$ with a node set of n nodes, $\mathcal{V}$, and a distance function $d_{j,v}: \mathcal{J} \times \mathcal{V} \rightarrow R^+$ associated with the jth leaf node and node v; this distance captures the cost paid when the client j retrieves an object from the node v. Each client issues a request for an object that must be fulfilled by a node holding that object or by the origin server (not represented in this model). A client always receive an object from a unique node.

The storage capacity allocation problem is defined as identifying a set $\mathcal{A} \subseteq \mathbb{A}$ with no more than $\mathcal{S}$ elements (node-object pairs) $(v,k), v \in \mathcal{V}, k \in \mathcal{O}$; $\mathbb{A}$ is the set that contains all node-object pairs. Differently from [19], in cache slicing approach, there is restriction of the nodes an object can be placed, since only the nodes that have a surrogate can be candidate to receive an object. Another restriction is to consider the capacity of that surrogate (see equation 3). Let Φ, $\Phi \subseteq \mathcal{V}$, be the set of nodes v that has an installed surrogate, then $\mathcal{A}$ must be chosen so as to minimize the following expression cost:

$$\min_{\mathcal{A} \subseteq \mathbb{A}: |\mathcal{A}| \leq \mathcal{S}} \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{O}} p_j(k) \cdot d_j^{min}(k) \text{ where } d_j^{min}(k) = \min\{d_{j,os}, d_{j,v}\} : v \in \Phi, (v,k) \in \mathcal{A} \quad (1)$$

Where $d_{j,os}$ is the distance between the jth client and the origin server, while $d_{j,v}$ is the distance between the jth leaf node and a node v. To define the capacity allocation problem as a integer linear program, let $X_{j,v}(k)$ denote a binary integer variable which is equal to one if client j gets object k from node v, and zero otherwise. Also let $\delta_v(k)$ denote an auxiliary binary integer variable which is equal to one if object k is placed at the node v, and zero otherwise. The two types of variables are related as follows:

$$\delta_v(k) = \begin{cases} 1, & if \quad \sum_v X_{j,v}(k) > 0 \\ 0, & otherwise \end{cases} \text{ ,where } v \in \Phi \quad (2)$$

A surrogate installed at the node v has a capacity $\mathcal{C}_v$, $\sum \mathcal{C}_v \leq \mathcal{S}$, thus, the allocation must also be subject to the following constraint:

$$\sum_{k \in \mathcal{O}} \delta_v(k) \leq \mathcal{C}_v \quad (3)$$

Under this cache slicing problem, each tenant, a content provider, has its own sets of objects, clients and performs an individual storage allocation. In other words, the problem of capacity allocation must be solved for each tenant. But the overall solution of cache slicing problem has its own set of constraints.

Let T be a content provider that belongs to a set $\mathbb{T}$, of all content providers of cloud-based CDN. Each content provider forms a virtual CDN installed on top of the cloud infrastructure of the cloud-based CDN and must solve the capacity allocation independently of each other to find an optimal allocation of objects on the available caches. The set Φ is related to all points of presence of the cloud-based CDN, L, but not necessarily $\Phi = L$, in other words, the virtual CDN of a content provider may not have presence on all available locations of the cloud-based CDN. Each content provider T acquires a certain amount of storage space from the cloud-based CDN, $\mathcal{S}_T$, and must construct the virtual CDN infrastructure subject to this hired capacity. Each node $v \in L$, has also a capacity $\mathbb{C}_v$, which represents the size of underlying physical infrastructure available to cloud-based CDN. Consider $\mathcal{C}_{v,T}$ the capacity of surrogate installed on node v of the content provider T. $\mathbb{S}$ is the total capacity of entire cloud underlying physical infrastructure. The cache slicing problem can hence be formulated as follow:

Maximize:

$$\mathbb{z} = \sum_{T \in \mathbb{T}} \sum_{j \in \mathcal{J}_T} \sum_{k \in \mathcal{O}_T} p_j(k) \cdot \sum_{v \in \Phi_T} (d_{j,os} - d_{j,v}) \cdot X_{j,v,T}(k) \quad (4)$$

Subject to:

$$\sum_{v \in \Phi_T} X_{j,v,T}(k) \leq |\Phi_T| \qquad j \in \mathcal{J}_T, k \in \mathcal{O}_T, T \in \mathbb{T} \quad (5)$$

$$\sum_{k \in \mathcal{O}_T} \delta_{v,T}(k) \leq \mathcal{C}_{v,T} \quad (6)$$

$$\sum \mathcal{C}_{v,T} \leq \mathcal{S}_T \quad (7)$$

$$\sum_{T \in \mathbb{T}} \mathcal{C}_{v,T} \leq \mathbb{C}_v \ \ for\ each\ v \in L \quad (8)$$

$$\sum_{v \in \Phi_T} \mathcal{C}_{v,T} \leq \mathcal{S}_T \ \ for\ each\ \ T \in \mathbb{T} \quad (9)$$

$$\sum_{T \in \mathbb{T}} \mathcal{S}_T \leq \mathbb{S} \quad (10)$$

where $X_{j,v,T}(k), \delta_{v,T}(k)$ binary decision variables of virtual CDN T, v $\in \mathcal{V}, k \in \mathcal{O}_T, j \in \mathcal{J}_T, T \in \mathbb{T}$.

The cache slicing is a mixed linear programing problem with complicated constraints [20]. The master problem is in equation (4), defined in the first sum, and the sub-problems are related to the equations 4-10. The complicating constraint is the equation (6). The ILP formulation of the cache slicing problem is NP-hard, thus, it cannot be used for practical purposes. The solution of each sub-problem has a complexity of at least $O(|V| \cdot |\mathcal{O}| \cdot |\mathcal{J}| \cdot |\Phi|)$. In this paper, we provide an heuristic solution.

C. The algorithm

The CDN has a number of caches / surrogates, which can be changed when it decides to expand or reduce its infrastructure. This process of expansion or reduction can be done, for example, with the **acquisition of new equipment, space allocation in the cloud, or CDN Interconnection (CDNI)** [21] **contracts**. The important thing is that the change in structure will trigger a new run of the algorithm, as explained below.

First, the cache on each surrogate is divided into fixed-size small portions, called slots. The slots have a fixed size and are unique for all CDNs and all content providers. Each content provider receives a portion of each cache, but not necessarily all the surrogate's caches. When the CSP receives a portion of the cache, we say that it has a slice of it. The aim of this approach is to create a virtual CDN. This virtual CDN will have, as virtual surrogates, the set of slices of the underlying surrogate the corresponding content provider received. A slice has the same size as a slot. Slots are empty slices of the cache or unallocated storage.

The number of slices a content provider can receive from all caches depends on the contracted storage capacity, i.e., the sum of the number of slices a CSP has on all caches is equal to the total hired storage capacity. The problem that we intend to solve with our approach is how to distribute the slices in each cache among the content providers. For this, our algorithm is divided into three phases, as detailed next.

1) Phase 1

At this stage the virtual CDN is formed by defining where the slices allocated to each content provider will be located, i.e., exactly which slices on which caches each content provider will receive. We call this phase "placement of slices" because it is similar to the placement of surrogates. In fact, our approach is inspired by it and uses the same approach.

As it can be easily seen, the surrogate placement problem is computationally complex, which leads to development of some heuristics, such as the Greedy replica placement [13] and the scalable replica placement [22]. These heuristics use existing information from the CDN, such as workload pattern and network topology. The Greedy algorithm brings very good results and we use it as the basis of our approach. We tailor our approach to this problem by replacing potential sites by slots on caches, and surrogate servers by slices. Note that since we have many slots on each cache, we can have many slices for the same CSP on that physical cache. In fact, this is somewhat similar to the concentration of surrogates around a spot of high demand.

In summary, in this phase, each CSP has a number of "desired" slices on each cache. However, there may be a situation where the sum of "desired" slices of all CSPs on a specified cache exceeds the capacity of that cache. When this is the case, phase 2 of our algorithm is executed.

2) Phase 2

This phase is executed after the first one and only if the number of slices allocated for each content provider for a specific cache exceeds the number of slots on that cache. Then, those content providers will have to compete for them. Before we present how this competition occurs, some considerations must be made regarding the goals of the system.

We propose and study two different strategies of competition for slices: Proportional and Greedy. The former uses a simple criterion to decide which CSP should win the competition for a slot. The decision is based on the CSP's popularity. It is important to notice that a CSP's popularity is considered independently, and this means that each CSP will define its desired slices' configuration based only on its content popularity. Based on the CSP's desired slices we calculate the Slice Proportion (SP) and then multiply the SP by the number of desired slices; the real number of slices allocated to each CSP is defined.

The other approach is a heuristic based on the number of requests, the size, and the location of objects. The goal of the heuristic is to maximize the CDN's profit by minimizing the cost of content delivery. This is done by ranking objects according to their number of requests and calculating the score for each CSP. This score represents the benefit to a CSP receiving a slice. To estimate this benefit, we test what the allocation of a new slice, for each competing CSP, would represent to content delivery. Thus, according to the size of the slice we can estimate that a CSP would place a new set of objects in caches and then estimate the gain of the CDN by allocating a slice for each CSP. This is done by summing the estimated gain obtained by all objects that could be placed in

the new slice, and then subtracting all objects left out of the cache. The gain for each object is calculated by multiplying the object's size by the number of times it was requested during the last time window. Fig. 1 and Fig. 2 illustrate a slice competition between two CSPs. The size of the new slice is 150KB. In the former, the score is better for CSP 1. But with a similar scenario and greater popularity, as shown in Fig. 2, for CSP 2's first object, the score changes making it more beneficial for the CDN to give this new slice to CSP 2.

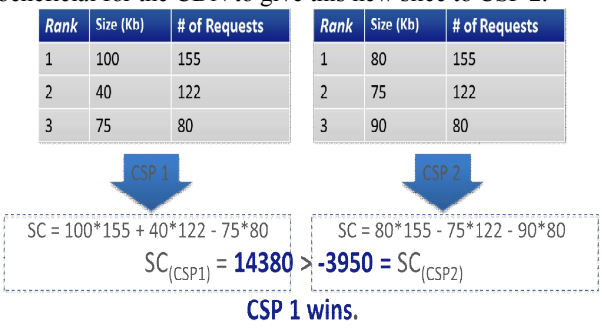

Fig. 1. Cache slicing, greedy strategy

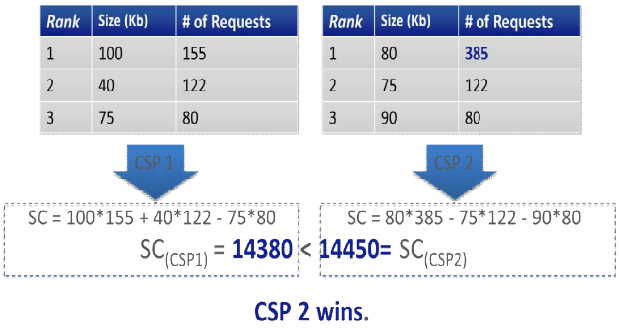

Fig. 2. Cache slicing, greedy strategy

Because of the competing process, after the execution of phase 2, there could be some content providers that asked for a slice on a given cache but did not get it; see Fig. 3.

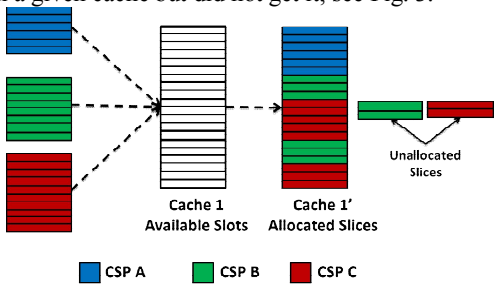

Fig. 3. Example of cache slicing configuration after phase 2

3) Phase 3

The third is the simplest phase; it only happens when, after phase 2, there are yet slices to be placed. In this case, CSPs are probed to place one of the remaining slices in a Round Robin fashion, but considering only available slots remaining after the execution of Phase 2. Each CSP places one, and only one, slice per round and it does so according to its content demand.

More importantly, though, is the maintenance of the system. The allocation of slices for each CSP is not static. We argue that this is the advantage of using virtual CDNs for each CSP in our approach. Thus, the CDN monitor will provide information to a database to trigger the adaptation of the system (see [10] for architecture). The trigger will run the algorithm (all phases) when a change in the traffic behavior happens, which is periodically monitored, according to [10]. Also, it can be triggered when one of the following conditions are met: 1. a new content provider enters the system; 2. a content provider acquires more capacity; 3. the costs of the infrastructure change; 4. the CDN provider changes the infrastructure; 5. there is a change to the SLA of a CSP.

The main advantage of our approach is not having a static configuration. Further, the adaptation may only be needed by a specific CSP and it may not impact the allocation of the others, generating less configuration overhead. The virtual CDN does not necessarily need a cloud infrastructure.

V. Evaluation

To evaluate the slicing strategy, we created a simulation scenario with three CSPs. Each CSP has its own objects with specified sizes and requests. The main metric observed is the hit ratio and each scenario is executed twice, with and without the slicing strategy. When cache slicing is not active, the objects of both CSPs are inserted into surrogate caches without identifying their origins, i.e., they are treated as if belonging to the same CSP, like in a traditional CDN. When slicing is active, each CSP has a portion of the surrogate cache. Insertion and removal are still performed by the same cache replacement technique, but independently. The algorithm for cache replacement is the same for both executions, and thus the only difference when slicing is active is the division of caches, as to considering these as one big cache for all CSPs. In addition, for both scenarios, the total storage capacity of the CDN is the same. The specific scenario configuration is described as follows. The simulator was validated in [24].

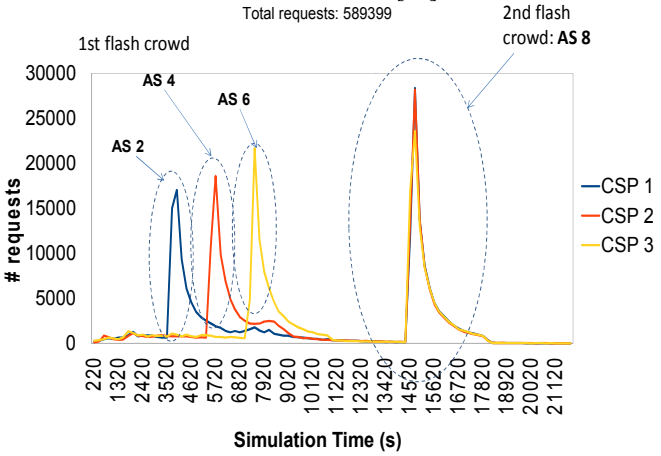

Fig. 4. Request evolution over time

1) Scenario Description

The scenario has three CSPs, and a number of requests, as shown in Fig. 4, with a total of 589,399 requests generated with MediSyn [23]. Two different algorithms were used for distributing the slices, namely Proportional and Greedy. They were based on the HotSpot and Greedy algorithms for Replica server location. CSP 1 has a baseline traffic of 43,008 requests,

a first flash crowd of 62,792 requests concentrated on AS2 that starts at 3,600s, and a second flash crowd of 90,802 requests concentrated on AS8 that starts at 14,400s. CSP 2 has a baseline traffic of 43,076 requests, a first flash crowd of 62,856 requests concentrated on AS4 that starts at 5,400s and a second flash crowd of 90,543 requests concentrated on AS8 that starts at 14,400s. CSP 3 has a baseline traffic of 43,706 requests, a first flash crowd of 63,165 requests concentrated on AS6 that start at 7,200s, and a second flash crowd of 89,455 requests concentrated on AS8 that starts at 14,400s. The first flash crowd events occur on different ASes (see Fig. 4). Thus, the CSPs should not compete for slices. This happens only with the last flash crowd.

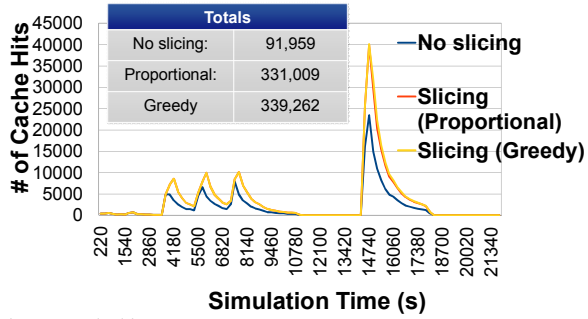

Fig. 5. Cache hit progress

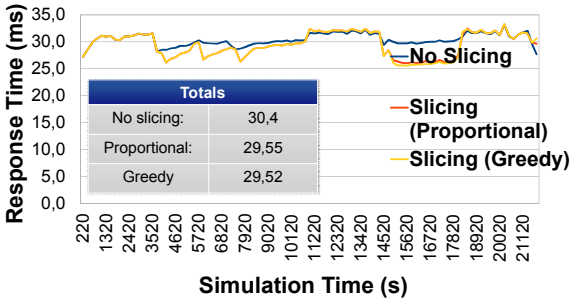

Fig. 6. Response time progress

2) Results

In the case of cache hits, shown in Fig. 5, both cache slicing algorithms performed similar to each other, with the majority of requests being fulfilled by content already on the cache, and much better than the case without cache slicing. The Greedy algorithm represents a small improvement on cache hits, 339,262 to 331,009. In the case of response time, there is a small difference between Proportional and Greedy algorithms during the second flash crowd, even though the total average for both algorithms is similar. During the second flash crowd, however, the Greedy performed systematically better, though with a very small overall advantage. When considering network traffic (Fig. 7), especially the cross traffic (Fig. 8), it is observable that using cache slicing represents a significant benefit for a CDN, especially a network provider-operated CDN, since it saves network bandwidth and may significantly decrease the CDNs connectivity costs. Once again, the Greedy algorithm yielded slightly better results when compared to the Proportional algorithm: respectively 17,629 GB and 18,304 GB for cross traffic; 104,798 and 106,076 GB for total traffic. The reduction, when compared to the traditional CDN, can reach 15% for total traffic and 35% for cross-traffic.

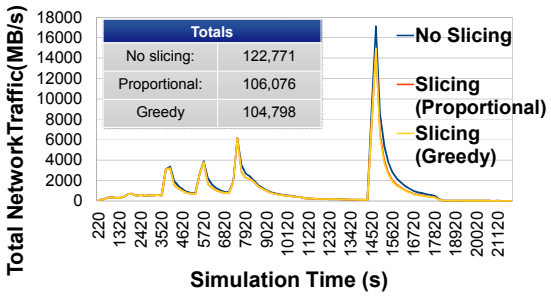

Fig. 7. Total traffic progress

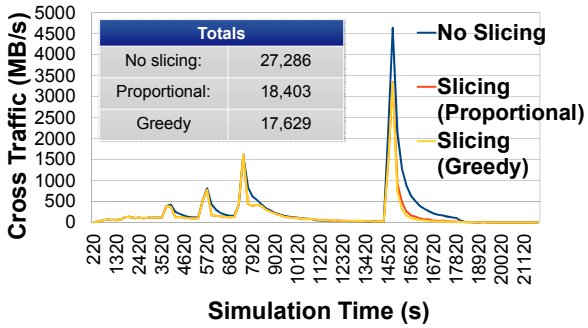

Fig. 8. Cross traffic progress

VI. Conclusions

Cache slicing represents an elegant and effective solution to the problem of virtualizing network resources for CDN providers with multiple tenants. The economy represented by resource allocation stability, fewer processing activities due to frequent network rearrangement, fewer network resources used, and better performance delivered to users can certainly leverage a CDN provider efficiency and profitability.

We evaluated two different cache slicing algorithms and compared them to the use of a CDN without cache slicing. Cache preference is given to the CSP that has more content requests. This optimization is enforced to save resources. From the perspective of our results, using cache slicing not only represents considerable savings of network resources, but also provides better performance for a CDN serving more than one content provider, especially during flash crowds. Throughout our tests, using cache slicing allows a CDN to achieve much higher cache hit rates and results in lower response times, while at the same time reducing cross-traffic (costly traffic between different autonomous systems).

Thus, the use of cache slicing greatly improves the QoE results of the CDN. This happens because the use of slicing allows the CDN to behave as a set of CDNs, each one acting in the interest of its content provider, instead of a big CDN where content providers compete for resources and QoE. The management of the caches in a per-CSP manner allows each set of content to be managed independently, as opposed to having content among different providers competing per popularity. The slicing, however, prevents this competition among content and chances it to a controllable competition for slices.

Acknowledgment

This work was supported by the RLAM Innovation Center, Ericsson Telecomunicações S.A., Brazil.

REFERENCES

[1] Pathan, A. K., and Buyya, R. "A taxonomy and survey of content delivery networks." Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report, 2007.

[2] CISCO. “Cisco Visual Networking Index: Forecast and Methodology, 2012–2017”, Available at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf. Last access in February, 2014.

[3] James Broberg, Rajkumar Buyya, Zahir Tari. MetaCDN: Harnessing ‘Storage Clouds’ for high performance content delivery.Journal of Network and Computer Applications, Volume 32, Issue 5, September 2009, Pages 1012-1022.

[4] C. Papagianni, A. Leivadeas and S. Papavassiliou, "A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment," IEEE Transactions on Dependable and Secure Computing, Sept.-Oct., 2013.

[5] Buyya, R., et al, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems. Volume 25, Number 6, Pages: 599-616, June 2009.

[6] Mikkilineni, Rao; Sarathy, Vijay, Cloud Computing and the Lessons from the Past. In Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE '09. 18th IEEE International Workshops, July, 2009.

[7] Fangfei Chen; Guo, K.; Lin, J.; La Porta, T., "Intra-cloud lightning: Building CDNs in the cloud," INFOCOM, 2012 Proceedings IEEE , vol., no., pp.433,441, 25-30 March 2012.

[8] Haitao Li; Lili Zhong; Jiangchuan Liu; Bo Li; Ke Xu, "Cost-Effective Partial Migration of VoD Services to Content Clouds," Cloud Computing (CLOUD), 2011 IEEE International Conference on , vol., no., pp.203,210, 4-9 July 2011.

[9] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. Vasilakos, “CoDaaS: An Experiment Cloud-Centric Content Delivery Platform for User-Generated Contents,”Proc. Int’l Conf. Computing, Networking and Comm. (ICNC),pp. 934-938, Feb. 2012.

[10] Moreira, A., Moreira, J., Sadok, D., Callado, A., Rodrigues., Neves, M., Souza, V., and Karlsson, P. 2011. “A Case for Virtualization of Content Delivery Networks”, In: IEEE. Winter Simulation Conference.

[11] Chia-Feng Lin; Muh-Chyi Leu; Chih-Wei Chang; Shyan-Ming Yuan, "The Study and Methods for Cloud Based CDN," Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011.

[12] Carlsson, N.; Dan, G.; Eager, D.; Mahanti, A, "Tradeoffs in cloud and peer-assisted content delivery systems," Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on Sept. 2012.

[13] P. Krishnan, D. Raz, and Y. Shavitt, “The cache location problem,” IEEE/ACM Transactions on Networking (TON), vol. 8, no. 5, 2000.

[14] Dán, G. & Carlsson, N. (2014). Dynamic Content Allocation for Cloud-assisted Service of Periodic Workloads. In: 33rd Annual IEEE (INFOCOM'14); Toronto, Canada, 27 April - May 2, 2014.

[15] Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A model for content internetworking (CDI). Internet Engineering Task Force RFC 3466, 2003.

[16] Ying Lu, Tarek F. Abdelzaher and Avneesh Saxena. Design, Implementation, and Evaluation of Differentiated Caching Services. IEEE Transactions on Parallel and Distributed Systems archive. Volume 15 Issue 5, May 2004 Page 440-452.

[17] Plagemann, T., Goebel, V., Mauthe, A., Mathy, L., Turletti, T., and Urvoy-Keller, G. From content distribution to content networks – issues and challenges.Computer Communications, 29(5), pp. 551–562, 2006.

[18] N. Yoshida, “Dynamic CDN against flash crowds,” http://www.springerlink.com/content/q2j5130831r84365/. [Online].

[19] Nikolaos Laoutaris, Vassilios Zissimopoulos and Ioannis Stavrakakis. On the optimization of storage capacity allocation for content distribution. Computer Networks: The International Journal of Computer and Telecommunications Networking. Volume 47 Issue 3, February, 2005. Pages 409-428.

[20] Conejo, A.J., Castillo, E., Minguez, R., Garcia-Bertrand, R. Decomposition Techniques in Mathematical Programming. Engineering and Science Applications. 2006, XVI, 542 p.

[21] Content Delivery Network Interconnection (CDNI), https://datatracker.ietf.org/wg/cdni/charter/

[22] X. Jia, D. Li, H. Du, and J. Cao, “On Optimal Replication of Data Object at Hierarchical and Transparent Web Proxies,” IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 8, 2005.

[23] Tang, W., Fu, Y., Cherkasova, L., and Vahdat, A. 2003. "Medisyn: A synthetic streaming media service workload generator." In Proceedings of the 13th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 12-21.

[24] Rodrigues, M. B. E. ; Moreira, A. L. C. ; Azevedo, E. ; Neves, M.; Sadok, D.; Callado, A. ; Moreira, J. . On Learning How to Plan Content Delivery Networks. In: 46th ANSS, 2013, San Diego