Computer Communications xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Big Data-backed video distribution in the telecom cloud

Q1 M. Ruiz[a], M. Germán[a], L.M. Contreras[b], L. Velasco[a,*]

[a] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
[b] Telefónica Investigación y Desarrollo (TID), Madrid, Spain

ARTICLE INFO

Article history:
Received 23 November 2015
Revised 8 February 2016
Accepted 30 March 2016
Available online xxx

Keywords:
Telecom CDN
Q2 Cloud CDN
Big Data

ABSTRACT

Telecom operators are starting the deployment of Content Delivery Networks (CDN) to better control and manage video contents injected into the network. Cache nodes placed close to end users can manage contents and adapt them to users' devices, while reducing video traffic in the core. By adopting the standardized MPEG-DASH technique, video contents can be delivered over HTTP. Thus, HTTP servers can be used to serve contents, while packagers running as software can prepare live contents. This paves the way for virtualizing the CDN function. In this paper, a CDN manager is proposed to adapt the virtualized CDN function to current and future demand. A Big Data architecture, fulfilling the ETSI NFV guidelines, allows controlling virtualized components while collecting and pre-processing data. Optimization problems minimize CDN costs while ensuring the highest quality. Re-optimization is triggered based on threshold violations; data stream mining sketches transform collected into modeled data and statistical linear regression and machine learning techniques are proposed to produce estimation of future scenarios. Exhaustive simulation over a realistic scenario reveals remarkable costs reduction by dynamically reconfiguring the CDN.

1. Introduction

Live-TV and Video on Demand (VoD) distribution is in the portfolio of many telecom operators aiming at entering into competition with on-line, over-the-top broadcasters, such as Netflix. To this end, a Content Delivery Network (CDN) is being considered as a suitable option to be deployed by telecom operators internally within their network infrastructure by placing cache nodes in geographically distributed locations covering a territory [1,2]. Forecasts show that 79% of the global IP traffic will be related to video traffic by 2018 [3] thus managing its own CDN allows the network operator to better control and manage the video content injected into the network through predictable traffic sources strategically placed according to a careful network planning to maximize capacity savings. Cloud-based CDNs provide CDN functionalities using cloud resources. Nonetheless, the introduction of cloud imposes additional challenges that have to be addressed. Authors in [4] present a survey on available cloud-based CDNs and identify the open challenges.

In fact, the telecom infrastructure is undergoing a huge transformation since telecom operators are deploying their own cloud infrastructure [5] to prove cloud services and enabling *Software Defined Networking (SDN)* [6] and Network Functions Virtualization (NFV) [7]. The resulting infrastructure is referred to as the *telecom cloud* [8]. NFV decouples network functions (e.g., caching) from proprietary hardware appliances, so they can be implemented in software and deployed on virtual machines (VM) running on commercial off-the-shelf computing hardware. A Virtualized Network Function (VNF) can be functionally decomposed into one or more components and different VNF instances can be placed in geographically distributed locations and communicate among them.

Regarding video delivery, the standardized MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [9] technique enables media content to be delivered over the Internet. MPEG-DASH requires from a HTTP web server infrastructure to allow users' devices (e.g., Internet-connected televisions, desktop computers, smart phones and tablets, etc.) to consume multimedia content. MPEG-DASH divides contents into a sequence of small file *segments*, each containing a short interval of the content. At the start of a streaming session, the MPEG-DASH client downloads a Media Presentation Description (MPD) file with the resource identifiers (HTTP-URLs) for content's segments. A variety of different qualities (e.g., by changing bitrate and resolution) is made available for each content; while a content is being played back, the MPEG-DASH client automatically selects the segment with the highest quality possible that can be downloaded in time thus, dealing with variable Internet conditions. In addition, client buffer size can be adjusted to ensure a given probability of video re-buffering [10].

---

[*] Corresponding author. Tel.: +34934016999.
E-mail address: lvelasco@ac.upc.edu (L. Velasco).

http://dx.doi.org/10.1016/j.comcom.2016.03.026

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

MPEG-DASH enables CDN virtualization, where cache nodes are virtualized and be placed in datacenters (DC) (see use case in [7]). Virtualizing caching capabilities facilitates rapid distribution and/or scaling of cache nodes in a cost-efficient and scalable manner. For instance, as a result of using MPEG-DASH for delivery, multimedia contents can be served by HTTP servers. Another cache component must be in charge of generating DASH segments in several qualities and the related MPD files. However, the component that requires more computational effort is video transcoding/transrating, although it can be implemented in software and performed in real-time (see [11] for available software implementations).

To reduce traffic in the core network, cache nodes can be placed as close as possible to the end users. Authors in [12] presented a configurable, efficient and transparent in-network caching service to improve the VoD distribution efficiency by caching video contents as close to the end-user as possible. The solution leverages SDN technology improve network utilization and increasing the Quality of Experience for the end-user. Related to this, authors in [13] proposed a hierarchical *telecom CDN* and a caching algorithm to decide which objects to cache and a cache collaboration strategy to determine how cacheable items are propagated throughout the telecom CDN. In [14] authors studied the performance of distributing caches and the impact of its size and the cache update logic for VoD services, e.g. catch-up programs and movies. They concluded that placing caches in the aggregation network improves the percentage of requested content found in the cache (*Hit Ratio*, HR); in contrast, placing the cache in the access reduces the amount of traffic.

Apart from their right placement, cache nodes are typically dimensioned for peak demand and therefore, greatly underutilized at other times. Aiming at elastically adapt the allocated resources to the current service needs, authors in [15,16] proposed to leverage on the resources of cloud providers to increase capacity when required.

Analyzing video sessions, authors in [17] concluded that a centralized controller could improve user experience, while authors in [18] introduced presented a centralized algorithm for live video optimization providing real-time, fine-grained control. In addition, placing new cache nodes to accommodate spikes in demand and consolidate workload in few cache nodes when the load decreases can also bring benefits. Apart from classical optimization algorithms on conventional content distribution problems, the usage of cloud resources offers a new dimension for optimization that is the IT resource cost (i.e., storage, CPU, etc.) Commercial cloud infrastructure for CDN deployment was reported in [19]. While the concept is applicable to the idea of virtualized CDN, the proposed architecture does not fit to network operator scenarios.

Regarding the interconnection network, connection capacity adaptation is not trivial when it is based on optical technology. Authors in [20,21] proposed a cross-stratum orchestrator architecture to coordinate DC and network elastically.

To detect when resources have to be added or released, the performance and load of cache nodes need to be monitored. Monitoring a *variety* of network elements, servers and applications entails collecting huge *volumes* of data that needs to be transferred and stored assessing *validity*, as well as being analyzed and processed *fast* to achieve near real-time performance. Therefore, Big Data techniques for data collection, pre-processing, and analysis and visualization should be applied. In [22], the ITU-T Study Group 13 proposes a classification of the roles in a Big Data ecosystem. Among the identified roles, the *Big Data application provider* executes a specific set of data life-cycle to meet the requirements of data analysis and visualization as well as the security and privacy requirements. It utilizes the resources from a cloud provider for data analysis and provides analysis result to the Big Data service user. Another role is that of the *Big Data infrastructure provider*, which establishes a computing fabric (computation, storage, and networking resources as well as platforms and processing frameworks) in which certain transformation applications are executed, while protecting the privacy and integrity of data.

A telecom company can take advantage of all the above when deploying its own CDN to provide VoD and live-TV services. In this paper, we assume the hierarchical CDN architecture presented in Section 2 that includes: (i) a Big Data CDN Manager that detects opportunities to minimize operational costs and dynamically serves users from the most proper cache node, while adapting the CDN to the current load by reconfiguring cache nodes (i.e., scaling them by adding new resources), adding and releasing cache nodes, and managing connectivity; (ii) the CDN Admission and Control module responsible for controlling content access and deciding from which cache every user will be served; and (iii) the virtualized leaf cache node with a number of HTTP servers, packagers, storage and a cache manager. Specifically, the contributions of this paper are the following:

1. A Big Data CDN Manager responsible for adapting the CDN to the current and future load as well as its internal components is presented in Section 2, including: (i) a prediction module to forecast likely scenarios; (ii) a decision maker module to select the most appropriate reconfiguration; and (iii) an optimizer in charge of computing the optimal configuration of the CDN.
2. To facilitate CDN optimization, three incremental optimization problems are proposed in Section 3; (i) single cache node optimization; (ii) users re-allocation among caches and connectivity re-configuration; and (iii) global CDN re-configuration. Integer Linear Program (ILP) formulations are proposed and heuristic algorithms to solve the problems in real time are devised.
3. Section 4 targets at making decisions from collected data: (i) data stream mining *sketches* conveniently summarize collected data into modeled data; (ii) a prediction module based on machine learning techniques predicts likely scenarios; and (iii) a simple decision maker module based on threshold violations triggers the most appropriate optimization problem.

The discussion is supported by the results from exhaustive simulation over a realistic scenario in Section 5.

2. Telecom CDN

2.1. CDN architecture

A virtualized hierarchical CDN infrastructure can be deployed in the telecom cloud with some (few) central *Intermediate Cache Nodes* receiving contents from several sources and a number of *Leaf Cache Nodes* placed close to end users (Fig. 1). A centralized *CDN Admission and Control* module implements CDN access policies and redirects users' requests, e.g., based on their geographical location, to the (intermediate or leaf) cache node that will serve them.

Intermediate cache nodes and leaf cache nodes distribute two kinds of contents: VoD and live-TV. VoD contents are prepared in intermediate cache nodes and stored in leaf caches based on its popularity (see e.g., [13]). Nonetheless, in line with [23], live-TV is distributed from intermediate cache nodes and locally prepared in those leaf cache nodes delivering every specific TV channel to the users.

A virtualized leaf cache node would consist of the following components running as software inside VMs deployed in the same DC. The *packager* is in charge of live-TV preparation, including stream transcoding/transrating, segmentation and MPD generation. The *HTTP server* component serves end users' segment requests.

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

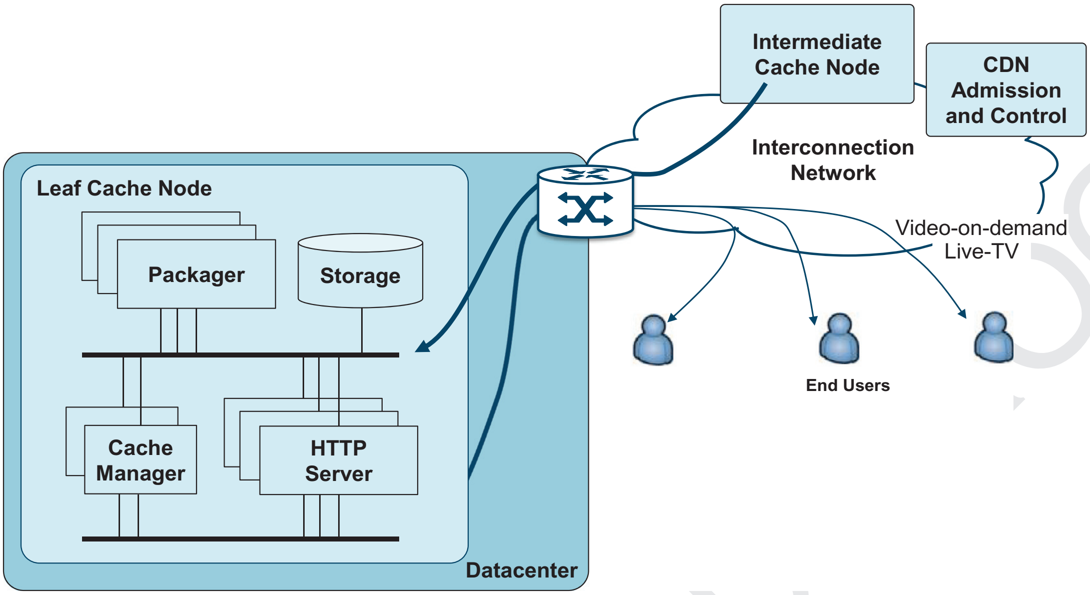

Fig. 1. Cache architecture and cache node main components.

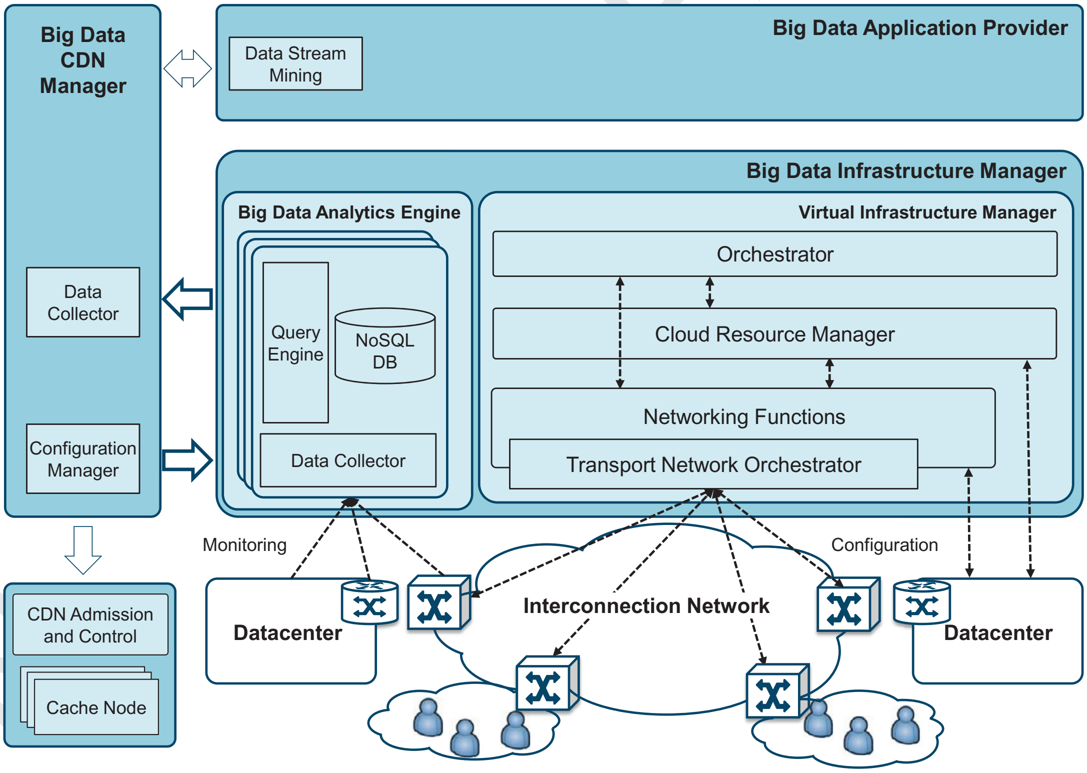

Fig. 2. Architecture supporting the Big Data CDN Manager.

The *Cache Manager* is the entry point of the cache node; it receives users' requests, identifies which contents will be locally stored, and redirects users' requests to the appropriate HTTP server. Each component usually consists of a pool of resources for load balancing and redundancy purposes.

We assume that the location for all intermediate cache nodes and those for leaf cache nodes distributing both, VoD and live-TV contents were selected during a pre-planning phase, based on the available connectivity, covered population, etc. Notwithstanding, the amount of resources in every resource pool can be dynamically adapted as a function of the load. In addition, new leaf cache nodes to deliver specific live-TV contents can be dynamically created and released in response to spikes in demand, e.g. a sports event.

2.2. Big Data CDN Manager

A *CDN manager* is responsible for adapting the CDN to the current and future load. However, an architecture supporting the CDN manager is needed to control virtualized components and data collection and pre-processing functionalities. Fig. 2 presents the

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

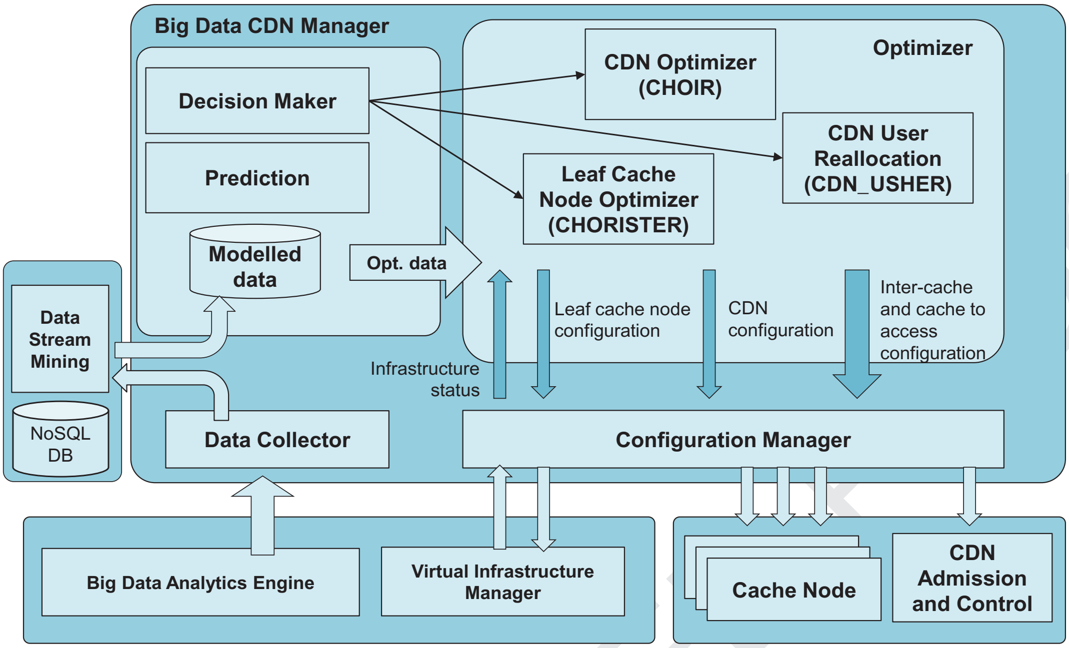

Fig. 3. Components of the Big Data CDN Manager.

proposed architecture, which is aligned with the ETSI NFV architectural guidelines [24].

A Big Data application provider offering Big Data processing to provide data analysis and visualization is shown on the top. The architecture of the Big Data infrastructure manager includes a Virtual Infrastructure Manager (VIM) and a Big Data Analytics Engine. The VIM architecture includes an orchestrator module, which is the common entry point for services and performs an overall coordination of cloud and networking resources. The Big Data analytics engine includes data collection, pre-processing, and storage and allows applications to monitor and manage allocated resources, while protecting the privacy and integrity of data. Each computing, network, and application node generates logging records that are collected and sent to one of multiple instances of the analytics node, which collate and store the information in a horizontally scalable database. Hence, the performance and load of a CDN can be monitored to elastically adapt its resources to current service needs. Data collected from the Big Data infrastructure manager is analyzed using services from the Big Data application provider.

A configuration manager is in charge of interfacing the VIM to request and release resources and of properly configuring each cache node. A more detailed view of the proposed CDN manager is presented in Fig. 3. After processing collected data from the analytics engine, it can be used to predict likely scenarios, thus anticipating future demand load distribution. A *prediction module* (PROMPTER) based on machine learning and time series modeling is proposed to that end.

Based on current and future load, the CDN can be optimized to minimize total costs while serving contents to the end users ensuring the highest Quality of Service (QoS) level. To that aim, three optimization problems have been devised: (i) the *Global CDN Optimization* (CHOIR) problem targets at serving users with the highest QoS level from leaf cache nodes with the minimum cost; (ii) the CDN User Reallocation (CDN_USHER) problem focuses on reallocating users among leaf cache nodes, just updating connections between cache nodes and metro areas; and (iii) the *Leaf Cache Node Optimizer* (CHORISTER) problem that scales a cache node down.

Analyzing current and predicted load distribution, a *decision maker module* (TUNER) is responsible of triggering the most appropriate optimization problem as well as selecting meaningful input data for its solving.

3. CDN optimization

As anticipated in the previous section, we face the CDN optimization problem by dividing it into three sub-problems. The CHOIR problem performs a global CDN optimization by redimensioning existing leaf cache nodes and creating and releasing leaf cache nodes to deliver live-TV according to the load, while ensuring that end users are served with the highest video quality. In addition, the CHOIR problem decides the connectivity needed between intermediate and leaf cache nodes and between leaf caches and metro areas.

For the sake of simplicity, we configure each cache component as follows: (i) a different VM flavor is defined for cache managers, packagers, and HTTP servers; (ii) two cache managers are configured in each cache for load balancing and redundancy purposes; (iii) every packager works on a single live-TV channel; (iv) every HTTP server in the pool can serve any content. Cache managers use a round-robin policy to select the server for every incoming request; (v) the size of the storage is preconfigured according to the target *HR*.

Globally optimizing the CDN might entail a large number of reconfigurations. However, in some situations just reallocating users between cache nodes will balance load of cache nodes, thus reducing the load of those running close to its currently allocated capacity. For this very reason, we propose the CDN_USHER problem that performs such reallocations, managing the connectivity between leaf cache nodes and metro areas. In addition, the CHORISTER problem focuses on releasing unused resources of a given cache node. HTTP servers are the only component which load really varies as a function of the number of users being served. However, the limiting factor is not the CPU but the use of bandwidth, so we use that parameter to decide whether the number of HTTP servers could be reduced.

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

3.1. Global CDN optimization (CHOIR) problem

The problem can be formally stated as follows:

Given:

- A set *IC* of intermediate cache nodes.
- A set of cache node types: {*void, TV, VoD + TV*}.
- A set *LC* of locations where leaf cache nodes are deployed and the allowable cache node types.
- A set *A* of metro areas with users consuming contents.
- The set *O* of contents being consumed. Contents include VoD and live-TV channels.
- A set *U* of user groups. Each group *u* contains all users inside a metro area that are currently playing a specific content with a specific device.

Output:

- Configuration of every $l \in LC$, including creating or releasing leaf cache nodes,
- Assignment (u, l) for every $u \in U$,
- Connections to be created/released/reconfigured.

Objective: Minimize the CDN cost from setting up resources in leaf caches and the needed connections.

The following sets and parameters have been defined:

IC	set of intermediate caches, index i
LC	set of leaf cache locations, index l
U	set of user groups, index u
A	set of metro areas, index a
O	set of contents, index o
O_{TV}	subset of O with live-TV contents
E	set of links, index e. Each link can be supported by a number of individual connections
$E_{IC\text{-}LC}$	subset of links connecting intermediate to leaf caches
$E_{LC\text{-}A}$	subset of links connecting leaf caches to metro areas
δ_{oi}	1 if content o is available in intermediate cache i
δ_{uo}	1 if user group u requests content o
δ_{ua}	1 if user group u is in metro area a
δ_l	1 if a leaf cache node is currently installed in location l
γ_l	1 if a leaf cache node in location l can be released
δ_{ul}	1 if user group u can be served from location l. This is computed based on transmission delay and policy rules
δ_{ol}	1 if content o can be served from location l
δ_{il}	1 if location l can be served from intermediate cache i
h_u	fraction of HTTP server in terms of bandwidth required to serve user group u with the best video quality supported by their device
b_u	bitrate needed to serve user group u with the best video quality supported by their device
b_{oil}	bitrate needed to convey content o from intermediate cache i to leaf cache l
hr_l	hit ratio of leaf cache node l, e.g. 70%
b_e	bitrate of each connection supporting link e, e.g. 1 Gb/s, 10 Gb/s, etc.
max_l	maximum amount of VMs available in leaf cache l
max_e	maximum capacity of link e
c_{HTTP}	cost per HTTP server to be allocated
c_{TV}	cost per packager component to be allocated
c_l	fixed cost for creating a new leaf cache, coming from VMs for cache managers, storage, and connections
c_e	cost per connection supporting link e

The decision variables are as follows:

x_{ul}	binary, 1 if user group u is served from location l; 0 otherwise
x_l	non-negative integer with the number of HTTP servers to be configured in location l
y_{ol}	binary, 1 if content o is required at leaf cache location l; 0 otherwise
y_{oil}	binary, 1 if content o in leaf cache location l is provided from intermediate cache node i; 0 otherwise
w_e	non-negative integer with the number of connections supporting link e
z_l^+	binary, 1 if leaf cache node l is created; 0 otherwise
z_l^-	binary, 1 if leaf cache node l is released; 0 otherwise

The formulation of the CHOIR problem is as follows:

$$\min \sum_{l \in LC} \left(c_{HTTP} \cdot x_l + \sum_{o \in O_{TV}} c_{TV} \cdot y_{ol} + c_l \cdot \left(z_l^+ - z_l^- \right) \right) + \sum_{e \in E} c_e \cdot w_e \tag{1}$$

subject to:

$$\sum_{l \in LC} \delta_{ul} \cdot x_{ul} = 1 \quad \forall u \in U \tag{2}$$

$$x_l \geq \sum_{u \in U} h_u \cdot x_{ul} \quad \forall l \in LC \tag{3}$$

$$y_{ol} \leq \delta_{ol} \quad \forall l \in LC, o \in O \tag{4}$$

$$\sum_{u \in U} \delta_{uo} \cdot x_{ul} \leq |U| \cdot y_{ol} \quad \forall l \in LC, o \in O \tag{5}$$

$$\sum_{i \in IC} \delta_{il} \cdot \delta_{oi} \cdot y_{oil} \geq y_{ol} \quad \forall l \in LC, o \in O \tag{6}$$

$$\sum_{i \in IC} y_{oil} \leq 1 \quad \forall l \in LC, o \in O \tag{7}$$

$$max_l \cdot z_l^+ \geq (1 - \delta_l) \cdot x_l \quad \forall l \in LC \tag{8}$$

$$x_l + \sum_{o \in O_{TV}} y_{ol} + 2 \cdot z_l^+ \leq (1 - z_l^-) \cdot max_l \quad \forall l \in LC \tag{9}$$

$$z_l^- \leq \delta_l \cdot \gamma_l \quad \forall l \in LC \tag{10}$$

$$b_e \cdot w_e \geq \sum_{o \in O_{TV}} b_{oil} \cdot y_{oil} + \sum_{o \in O \setminus O_{TV}} (1 - hr_l) \cdot b_{oil} \cdot y_{oil} \qquad \forall e \in E_{IC-LC}, i = IC(e), l = LC(e) \tag{11}$$

$$b_e \cdot w_e \geq \sum_{u \in U} \delta_{ua} \cdot b_u \cdot x_{ul} \quad \forall e \in E_{LC-A}, a = A(e), l = LC(e) \tag{12}$$

$$b_e \cdot w_e \leq max_e \quad \forall e \in E \tag{13}$$

The objective function (1) minimizes the CDN cost from setting up resources in leaf cache nodes and from the needed connections.

Constraint (2) guarantees that every user group will be served from one leaf cache location. Constraint (3) accounts for the number of HTTP servers that need to be set up in each location. Constraint (4) ensures that each cache location contains only allowed contents, e.g. it prevents from creating new leaf cache nodes serving VoD contents. Constraint (5) computes the contents required at each leaf cache location and, for each of them, constraint (6) assigns as source an intermediate cache node containing that content. Constraint (7) ensures that only one content source is selected.

Constraints (8)–(10) decide whether a leaf cache node is created or released. Constraint (8) computes whether a cache is created. Constraint (9) guarantees that the number of VMs to be configured in *l* does not exceeds a given maximum and releases the

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

Table 1
CHOIR heuristic algorithm.

```
1:  $bestS \leftarrow \emptyset$
2:  for $i = 1..maxIter$ do
3:      ResetResources(); $S \leftarrow \emptyset$
4:      sort $U_{VoD}$ and $U_{TV}$ randomly
5:      $U \leftarrow$ concatenate($U_{VoD}$, $U_{TV}$)
6:      for each $u$ in $U$ do
7:          $l* \leftarrow 0$; $cost \leftarrow \infty$
8:          for each $l$ in $LC$ do
9:              if $\delta_{ul} = 0$ then continue
10:             if ComputeCost($u$, $l$) < $cost$ then
11:                 $l* \leftarrow l$; $cost \leftarrow$ ComputeCost($u$, $l$)
12:         if $l* = 0$ then return INFEASIBLE
13:         $S \leftarrow S \cup \{(u, l*)\}$
14:         UpdateResources ($u$, $l*$)
15:     RemoveUnusedResources()
16:     if evaluate ($S$) < evaluate ($bestS$) then $bestS \leftarrow S$
17: return $bestS$
```

cache if no VMs are configured, while constraint (10) ensures that only designated cache nodes can be released.

Constraints (11)–(13) deal with the capacity of the interconnection network. Constraint (11) computes the amount of connections to support links between intermediate and leaf cache nodes and constraint (12) computes those for the links between leaf cache nodes and metro areas. Finally, constraint (13) guarantees that the requested bitrate for every link does not exceed a given maximum.

Note that different solutions can be obtained depending on the values for parameters max_l and max_e. Those parameters provide differentiated limits for every l and every e and need to be fixed every time the problem is to be solved; values can be obtained from the VIM to reflect the current resource availability, technology constraints, as well as operator policies. For instance, an operator might want to guarantee that maximum capacity of links connecting intermediate to leaf caches is 10 Gb/s, whereas that of the links connecting leaf caches to metro areas is 1 Gb/s, as a result of technology constraints. Moreover, the operator can also partition resources and reserve some amount of servers for the CDN service, while the rest of the servers are reserved for other services in its portfolio.

The CHOIR problem can be considered NP-hard since it is based on the *unsplittable capacitated assignment* problem that has been proved to be NP-hard [25]. Regarding its size, it entails $O(|LC| \cdot (|U| + |O| \cdot |IC|) + |E|)$ variables and $O(|U| + |O| \cdot |LC| + |E|)$ constraints. As an example, taking into account the instances presented in Section 5, the problem size raises to $7 \cdot 10^6$ variables and $7 \cdot 10^4$ constraints.

Since the CHOIR problem needs to be solved on-line, its exact solution is impractical. As a result, we propose the heuristic algorithm in Table 1 to obtain near optimal solutions in short computation times (e.g., hundreds of ms). The algorithm is an iterative randomized procedure that builds a solution by assigning user groups to cache locations and computing the cost of using and increasing (or releasing) cache and network resources. At each iteration, user groups are randomly sorted and assigned to a cache location with the minimum cost. After processing all users, unused resources are removed (if allowed) and the cost of the solution is computed. The best solution is returned upon exiting the algorithm.

3.2. CDN user reallocation (CDN_USHER)

The problem can be formally stated as:

Given:

- A set LC of locations where leaf caches are currently deployed.
- A set O of contents currently being served.
- A set A of metro areas with users consuming contents.
- A set U of user groups. Each group u contains all users inside a metro area that are currently playing a specific content with a specific device.

Output:

- Assignment (u, l) for every $u \in U$,
- Connections to be created/released/reconfigured.

Objective: Minimize the cost from the new connections to be established and the total number of users reallocated.

The following sets and parameters have been (re)defined:

δ_{ul}	1 if user group u can be served from location l. The definition has been extended to cover also whether the contents requested by user group u are available at l
γ_{ul}	1 if user group u is currently being served from l
h_l	current number of HTTP servers running in location l
g_e	current number of connections supporting link e
c_u	cost of reallocating user group u. Based on its size and other policies

A new decision variable is defined:

x_u	binary, 1 if user group u is reallocated; 0 otherwise

The ILP formulation is as follows:

$$\min \sum_{e \in E_{LC-A}} c_e \cdot (w_e - g_e) + \sum_{u \in U} c_u \cdot x_u \tag{14}$$

subject to:

$$\sum_{l \in LC} \delta_{ul} \cdot x_{ul} = 1 \quad \forall u \in U \tag{15}$$

$$\sum_{u \in U} h_u \cdot x_{ul} \le h_l \quad \forall l \in LC \tag{16}$$

$$x_u \ge x_{ul} - \gamma_{ul} \quad \forall u \in U, l \in LC \tag{17}$$

$$b_e \cdot w_e \ge \sum_{u \in U} \delta_{ua} \cdot b_u \cdot x_{ul} \quad \forall e \in E_{LC-A}, a = A(e), l = LC(e) \tag{18}$$

$$b_e \cdot w_e \le max_e \quad \forall e \in E_{LC-A} \tag{19}$$

The objective function (14) minimizes the cost of establishing new connections and reallocating users.

Constraint (15) guarantees that every user group will be served from one leaf cache location. Constraint (16) limits the demand served in each location to the capacity currently installed in each location. Constraint (17) stores whether a user group has been reallocated. Constraints (18) and (19) deal with the capacity of the links between leaf cache nodes and metro areas. Constraint (18) computes the amount of connections supporting each link and constraint (19) assures that the requested bitrate does not exceed a given maximum.

Note that after solving the CDN_USHER problem, a post-process for scaling down (see the CHORISTER problem), and even releasing, leaf caches is needed.

Similarly as for the CHOIR problem, the CDN_USHER problem can be considered NP-hard since it is based on the *unsplittable capacitated assignment* problem. The size of the problem is $O(|U| \cdot |LC| + |E_{LC\text{-}A}|)$ variables and $O(|U| \cdot |LC| + |E_{LC\text{-}A}|)$ constraints. The problem size rises to $7 \cdot 10^6$ for both, variables and constraints, thereby making impractical its exact solution. Thus, aiming at obtaining near optimal solutions in short computation times, a heuristic algorithm similar to that presented in Table 1, with the specific constraints of this problem, was developed; specifically, resources updating in lines 14 and 15 in Table 1 are constrained to network connections in the CDN_USHER problem.

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

3.3. Leaf cache node optimizer (CHORISTER)

The CHORISTER problem can be formally stated as:

Given:

- The current size of the HTTP servers' pool (s^{cur}) and its current bandwidth utilization k^{cur} (in %).
- A target bandwidth utilization (th_l), e.g. 60%.

Output:

- Target size (s^{tgt}) of the HTTP servers' pool.

Objective: Minimize the size of the HTTP servers' pool.

The CHORISTER problem can be solved using Eq. (20).

$$s^{tgt} = \left\lceil s^{cur} \cdot \frac{k^{cur}}{th_l} \right\rceil \tag{20}$$

4. Collecting data and making decisions

As previously introduced, each computing, network, and application node generates monitoring data that are collected and sent to the analytics engine. To preserve privacy, however, the CDN manager can only access data related to the CDN service. Monitored variables include: (i) bandwidth utilization of HTTP servers and links in $E_{LC\text{-}A}$, generated from network nodes; and (ii) A video quality metric generated by cache managers that measures whether the video quality provided to users from each cache node is the one requested.

Following a predefined time period, e.g. every minute, the CDN manager collects monitored data from the analytics engine. A time series is retrieved for each monitored point and average values are stored using the Big Data Application Provider facilities. Therefore, up to 60 consecutive observations are available every hour for each collected variable. Data stream mining *sketches* conveniently summarize collected data from every monitored point into modeled data representing the current state of a cache node. The following modeled variables have been defined:

q_l^{min}	minimum average video quality metric provided to users by cache node $l \in LC$
q_l^{cur}	current average video quality metric provided to users by cache node $l \in LC$
k_l^{max}	maximum average bandwidth utilization (in %) of interfaces in VMs running HTTP servers in cache l
k_l^{cur}	current average bandwidth utilization (in %) of interfaces in VMs running HTTP servers in cache l
k_e^{max}	maximum average bandwidth utilization (in %) of link $e \in E_{LC\text{-}A}$
k_e^{cur}	current average bandwidth utilization (in %) of link $e \in E_{LC\text{-}A}$

To select which of the optimization problems defined in Section 3 needs to be solved, we propose a simple *decision maker module* (TUNER) based on threshold violations. TUNER compares the evolution of modeled variables against a low threshold ($th_{(\cdot)}$). In brief, the CDN-USHER problem is solved to reassign groups of users in the case that the quality of the video being served from some cache nodes starts degrading, whereas the capacity of some other cache nodes is underutilized. In the case that a more in depth CDN reconfiguration is needed, the CHOIR problem needs to be solved. Finally, the CHORISTER problem is solved to scale underutilized cache nodes down. Table 2 summarizes the TUNER algorithm.

Table 2
Tuner decision algorithm.

```
$Underu_L$, $Underu_E$ ← false
for each $l$ in $LC$ do
  if $s^{tgt} < s^{cur}$ then $Underu_L$ ← true
  for each $e$ in $E_{LC\text{-}A}$ do
if $k_e^{max} < th_e$ then $Underu_E$ ← true
for each $l$ in $LC$ do
  if $q_l^{min} < th_q$ AND ($Underu_L$ OR $Underu_E$) then
    if CDN_USHER (input data) = FEASIBLE then
      return
for each $l$ in $LC$ do
  if $q_l^{min} < th_q$ then
    CHOIR (input data); return
for each $l$ in $LC$ do
  if $s^{tgt} < s^{cur}$ then CHORISTER (input data)
```

An important decision to be made is regarding the values of input data to solve the optimization problems. This is specifically relevant for user load, i.e. parameter h_u. When input data estimation is based on the observations in the collected data, we are making decisions following a *reactive strategy*, trying to update the CDN to changes in the demand after those changes have actually been detected (Fig. 4, top). Nonetheless, some prediction in the input data estimation could be introduced trying to anticipate those changes. To that end, the *predictive strategy* (Fig. 4, bottom) includes a prediction module, named as PROMPTER, able to produce estimation of future scenarios. The PROMPTER module estimates the value of modeled variables for the next period as well as other input data needed for solving optimization problems (e.g., h_u values). We refer to any of those variables to be estimated as a *response variable*, and will be in general denoted by Y. For such prediction, we use a methodology that combines statistical linear regression and time series prediction based on machine learning techniques [26].

Let us denote $Y(t)$ as the value of Y at time t. m *explanatory variables*, denoted as $X_i(t)$, are defined for each t, where each $X_i(t)$ can be deterministically and independently computed from the response variable. Three explanatory variables are considered in this work: (i) the time of the day (X_1), (ii) the capacity of the current CDN resources in terms of the potential amount of users that could be served (X_2), and (iii) a popularity measure of the available live-TV and VoD contents at t, computed from historic audience ratings. It is worth noting that these explanatory variables are strongly correlated to response variables. For instance, if the popularity of contents is expected to be high at prime time but the potential capacity of CDN is limited, then it is likely that a large expected number of users will be accessing to those contents and, consequently, they will experience poor video quality (i.e., low q_l^{min} and q_l^{cur}) due to over utilized CDN resources (i.e., high k_l^{max} and k_l^{cur}).

Starting from an initial data set with collected data for response and explanatory variables, the methodology consists of two phases. Response variables are first transformed to eliminate their correlation to explanatory variables; $Y(t)$ is transformed into $Z(t)$ by solving the multivariate linear model:

$$Z(t) = Y(t) - \sum_{i=1..m} \beta_i \cdot X_i(t) \tag{21}$$

where β_i represents the coefficient of variable X_i. Note that $Z(t)$ is the error of estimating $Y(t)$ with the linear model depending on $X_i(t)$ variables. The optimal values of β_i coefficients can be estimated by ordinary least squares fitting applied to the initial data set.

Eq. (21) predicts $Y(t)$ from explanatory variables observed at time t; however, the effect of past periods is not collected in this model. For this very reason, a second phase consisting in modeling variable $Z(t)$ based on previous observations is needed. We apply an Artificial Neural Network (ANN) model [27] due to its inherent capability of adapting to changes in a non-supervised manner, in contrast to auto-regressive models to fit continuous time series. Assuming an ANN with one hidden layer, the notation *p:s:1* indicates an ANN with p inputs, s neurons in the hidden layer, and 1 output. In our model, the inputs represent the last p observations

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

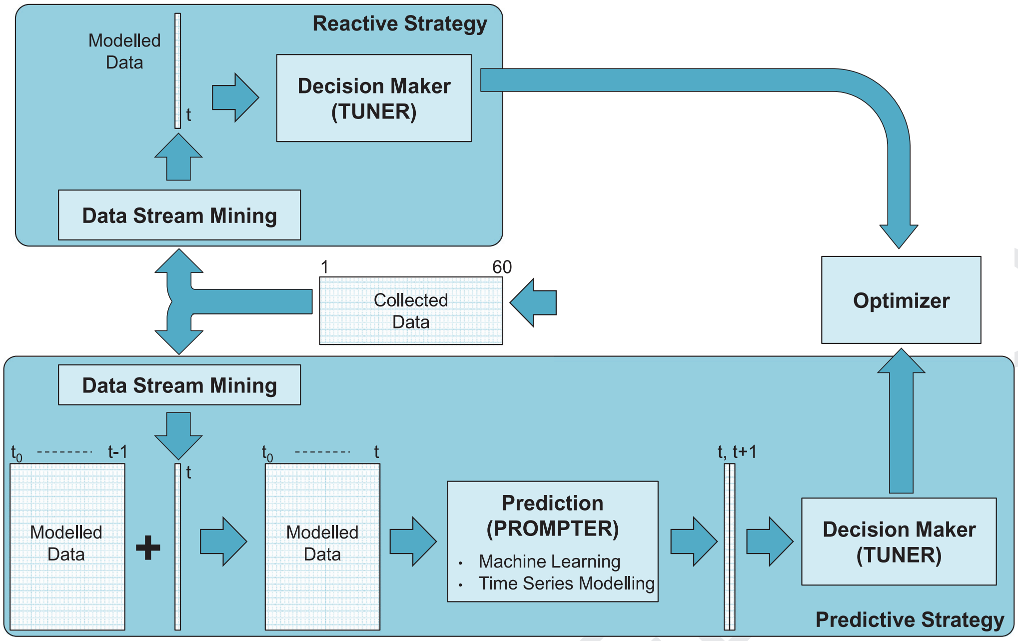

Fig. 4. From collected data to making decisions.

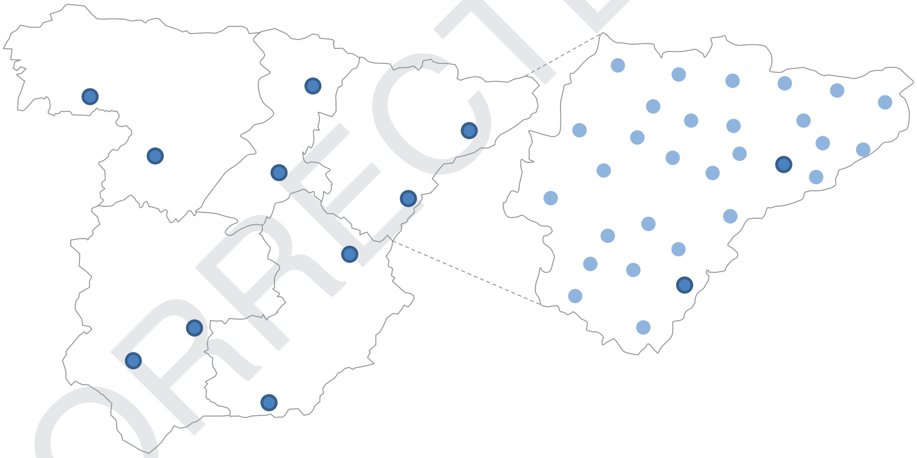

Fig. 5. Telefonica's scenario with five regions. Details of regional nodes in Catalonia–Aragon region.

before observation in t, i.e., $Z(t-p),\ldots, Z(t-1)$, whist the output returns the observation $Z(t)$. The Levenberg–Marquardt backpropagation algorithm [27] can be applied for training the ANN from the initial data set. Hence, predicting $Y(t+1)$ from current and stored observations is made by first estimating $Z(t+1)$ from previous observations using the ANN model and second transforming $Z(t+1)$ into $Y(t+1)$ using Eq. (21).

Once models have been obtained, refitting is applied to adapt models to changes in explanatory variables. To that end, the relative error between predictions and real observations is monitored and models are refitted when a threshold (e.g., 10%) is reached. A fixed sliding window allows limiting fitting to new observations.

Finally, note that the TUNER module for the predictive strategy receives two values for each modeled variable (e.g., $q_l^{min}(t)$ and $q_l^{min}(t+1)$) (Fig. 4). In that regard, we use the value that would result in the best service.

5. Simulation results

This section presents exhaustive simulation results evaluating the proposed CDN architecture and management algorithms over the realistic scenario 180-node Telefonica's optical national network, with 5 regional 30-node domains connected through an express 30-node core network. We assume a telecom cloud, where small datacenters are deployed in each node location with resources available to deploy leaf cache nodes. Two leaf cache nodes per region have been deployed providing both live-TV and VoD services (Fig. 5). In addition, new leaf cache nodes can be deployed on demand on any regional location to serve exclusively live-TV contents.

We assume a video distribution service with 500,000 subscribers geographically distributed among 150 metro areas, i.e., each area is connected to a regional node. 35 national and 15 regional TV channels are available per region with live and VoD contents. In line with recent studies [28], we assume a contents share of 65% for live and 35% for VoD contents.

Realistic time-varying demand is generated following a uniform random distribution centered on a typical hourly pattern with three demand peaks at morning, afternoon, and evening. Hot events targeting potential audience ranging from 40% to 80% subscribers are additionally generated with an inter-arrival time following a Poisson random distribution. VoD contents are requested

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

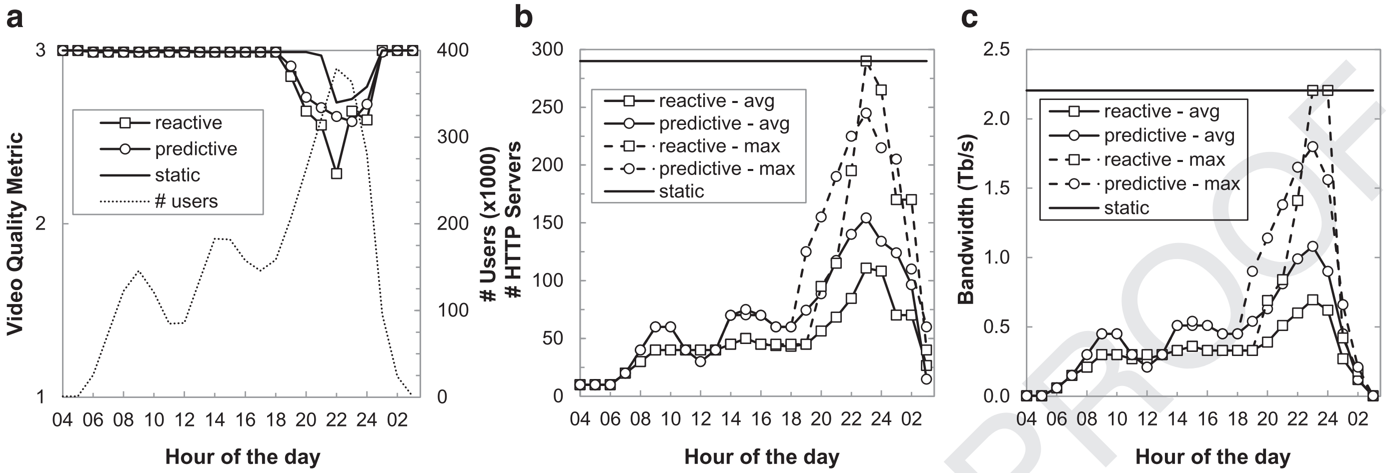

Fig. 6. Results vs. of hour of the day: served video quality metric (a), number of HTTP servers (b), and total bandwidth (c).

Table 3
Adoption scenarios for the years to come (%).

Quality (Mb/s)	2016	2017	2018	2019
SD (2.1)	28.6	22.0	13.6	7.0
HD (4)	38.6	34.3	28.2	23.7
Full HD (10)	27.1	37.7	42.3	47.3
4 K UHD (25)	5.7	6.0	15.9	22.0

according to a popularity metric following the model in [29]; after a period of time, e.g. 10 days, contents are taken off the service.

Table 3 summarizes the considered video qualities adoption scenarios for the years to come, based on [3]; total bandwidth in the network can be easily computed by multiplying the number of active users and the average bitrate for every adoption scenario. The quality actually served to users can be reduced up to 2 levels (e.g., from 4K to Full HD or even to HD) in case that not enough resources (HTTP servers) are available in a cache. We define the video quality metric using a three-level scale: a value of 3 is obtained when the served video quality equals the one requested, a value of 2 when the video quality is degraded to the immediately lower quality, etc.

We implemented a simulator in Matlab with the following main elements: (i) a user demand generator following the model above described; (ii) a CDN monitor that evaluates the state of the system and provides collected data to the CDN manager; (iii) the CDN manager including the data collector, TUNER and PROMPTER modules, as well as the algorithms for solving the proposed optimization problems; and (iv) the Big Data application provider in charge of processing collected into modeled data.

The ILP formulations presented in Section 3 were implemented using the CPLEX's Matlab API and integrated in the simulator. The performance of the proposed heuristics was compare against that of solving the ILPs in terms of quality of the solutions (optimality gap was set to 1%) and computation time. After solving more than one hundred problem instances, we concluded that heuristic provides quasi-optimal solutions with a gap between heuristic and ILP solutions as low as 0.5% in the worst case. Regarding computation time, solving the ILPs took more than 2 h for some instances compared to 1 s in the case of the heuristic.

Besides the reactive and predictive strategies introduced in Section 4, a *static* strategy is also defined for comparison purposes, where cache and network resources are statically configured; CDN is off-line planned and two leaf cache nodes per regional domain are deployed with a configuration, in terms of number of packagers and HTTP servers, to ensure enough video quality.

Let us first assume that leaf caches can be scaled adding/removing HTTP servers, but no new cache nodes can be created. We also assume that physical machines in the DCs are equipped with a 10 Gb/s network interface and the VM flavor for HTTP servers define a 10 Gb/s network interface; thus, one single VM instance can be placed per server.

Initial datasets to train prediction models were obtained by simulation and concluded that ANNs with 5 inputs and 10 neurons provide a goodness-of-fit higher than 95% for all response variables. Graphs in Fig. 6 plot CDN performance metrics as a function of the hour of the day for the 2016 scenario. Specifically, Fig. 6a plots the video quality metric under the different strategies under study; the number of users is also included for reference. When the load is under some value, all three strategies provide the best video quality metric value, i.e., the served video quality equals the one requested. However, when the load goes beyond some point, video quality metric values decrease; the best video quality metric is provided by the static strategy, while the reactive strategy provides the worst one. Interestingly, the predictive strategy performs better that the reactive one. In fact, the effectiveness of the PROMPTER module is validated since it allows improving the served video quality metric up to 45% with respect to the predictive strategy. This is as a result of anticipating video quality metric degradation and scaling the CDN accordingly.

Fig. 6b focuses on the number of active HTTP servers; average and maximum values are plotted for reactive and predictive strategies. In general, the number of HTTP servers allocated by the predictive strategy is slightly higher on average. Notwithstanding, the maximum number of servers for the highest load is requested by the reactive strategy. Aiming at fairly comparing strategies, we computed the area under the plots expressed in server-hour. The static strategy is dimensioned according to the maximum number of servers needed by the reactive strategy. 6960 servers-hour are required under the static strategy in contrast to 1147 required by the reactive one (a reduction of 83.5%). The predictive strategy needs 1593 servers-hour, 39% more than the reactive.

Finally, Fig. 6c presents average and maximum total bandwidth allocated by each strategy. Similarly as for HTTP servers, the reactive strategy requires less bandwidth than the predictive one on average, although peak values are higher for the former. It is now

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

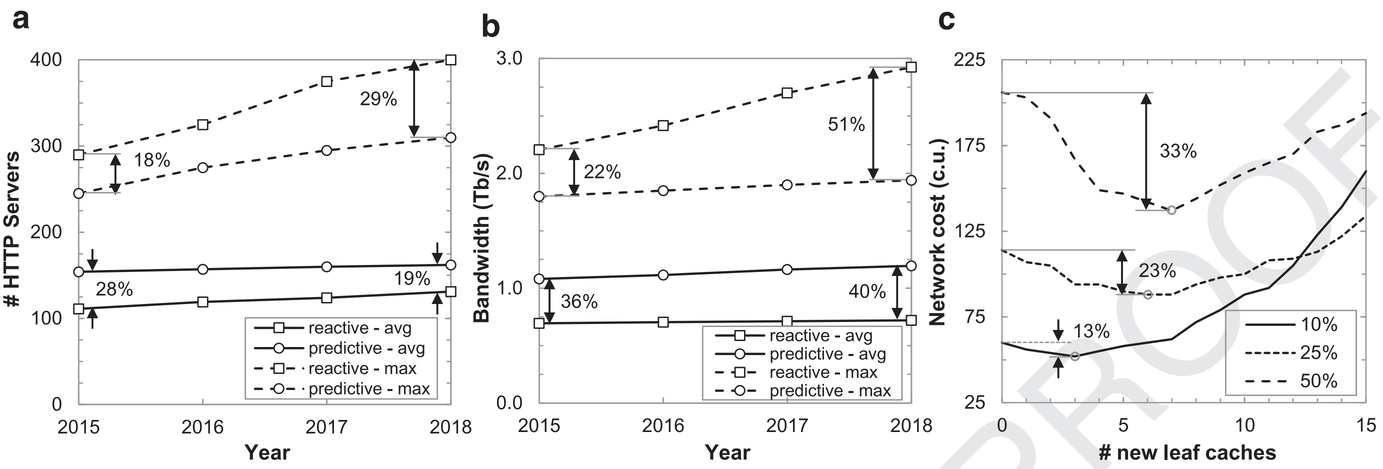

Fig. 7. Evolution of number of HTTP servers (a) and total bandwidth (b) for the years to come. Network cost vs. new leaf caches for live-TV (c).

Table 4
Number of servers at peak hour.

		1 Gb/s	10 Gb/s
Average	Reactive	70	111
	Predictive	113	154
Maximum	Reactive	286	290
	Predictive	195	245

clear that adapting CDN resources leads to enormous savings in terms of computational and network resources while providing virtually the same video quality metric.

Aiming at analyzing the impact of the VM flavor network interface capacity for HTTP servers, we run additional simulations assuming 1 Gb/s interfaces. We assume that each physical server can be shared by up to 10 of such VM instances. Table 4 summarizes the results comparing flavors with 1 Gb/s and 10 Gb/s interfaces. The VM flavor with 1 Gb/s interface clearly adds flexibility in the use of resources, resulting in savings in the number of required physical servers.

Fig. 7a and b shows the evolution of average and maximum amount of HTTP servers and total bandwidth, respectively for the adoption scenarios presented in Table 3. Although on average values show a quite flat evolution, maximum values increase significantly and show different slopes; the relative difference for peak values almost doubles since it increases from 18% to 29% for HTTP servers and from 22% to 51% for total bandwidth. Thus, the predictive strategy scales the best.

Let us now study whether being able to add and release new leaf cache nodes might bring some benefit. These new leaf cache nodes can be added to deliver specific live contents, such as a sports event or a concert. Each new leaf node cache is configured with the minimum resources required for serving the event, i.e., the cache manager, one packager per live-TV channel, and a number of HTTP servers. Since HTTP servers are required even if the event is served from fixed caches, the only additional costs rely on the extra amount of managers and packagers. In contrast, by placing transcoding closer to end users, the amount of traffic through the interconnection network is reduced, thus reducing network costs. Note, however, that adding new leaf caches entails creating new connections from intermediate caches to those leaf caches.

Fig. 7c plots network costs, computed as used bandwidth per km, as a result of adding new leaf cache nodes serving specific live-TV events, for three relative audience sizes with respect to the total amount of subscribers. As observed, costs savings range from 13% to 33% and are reached when 3, 6, and 7 new caches are added for 10%, 20%, and 50% audience sizes, respectively.

6. Concluding remarks

A Big Data-backed virtualized CDN architecture to be deployed in the telecom cloud has been proposed in this paper. The telecom CDN consists of a hierarchy of cache nodes: the centralized intermediate cache nodes receive live-TV channels and prepare VoD contents, whereas the leaf cache nodes, located close to the end users, manage VoD contents access and adapt live-TV channels to users' devices.

Media content can be delivered over HTTP by using the standardized MPEG-DASH technique and therefore, a virtualized leaf cache node would consist of a number of HTTP servers serving live and stored contents to users. In addition, packagers are needed for live-TV preparation as well as a cache manager in charge of applying caching policies to locally stored VoD contents. All these components can run as software inside VMs deployed in the same DC.

A CDN manager is responsible for adapting the CDN function to current and future load. The CDN manager needs from an architecture to control virtualized components and data collection and pre-processing functionalities; the proposed architecture follows the ETSI NFV guidelines.

The CDN manager optimizes the CDN by minimizing total costs, while ensuring that contents are served with the highest video quality metric level. The optimization problem is divided into: (i) the CHOIR problem that manages resources, i.e. VMs and connectivity, focused on global CDN optimization and assigns users to leaf cache nodes; (ii) the CDN_USHER problem that rebalances the CDN by reassigning users to leaf caches, and (iii) the CHORISTER problem that optimizes the number of VMs of a given leaf cache. ILP formulations were devised and heuristic algorithms proposed for its real time solving.

Re-optimization is run based on threshold violations. Data stream mining sketches transform collected data into modeled data representing the state of the CDN. The TUNER module compares current values against predefined thresholds and decides whether re-optimization need to be performed, which problem should be solved and the input data for the selected problem. Because updating the CDN to changes in the demand after they have occur might result in quality degradation, the PROMPTER module to produce estimation of future scenarios was proposed; it uses statistical linear regression and machine learning techniques to estimate the value of modeled variables for the next period.

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026

Exhaustive simulation results over a realistic scenario showed that a reduction of 83.5% in the number of allocated HTTP servers and a similar amount in total bandwidth can be reached when CDN reconfiguration is performed, while providing equivalent video quality metric to end users. Comparison between the reactive and the predictive strategies revealed that the reactive strategy uses fewer resources on average but more resources in the peak than the predictive one. The effect of allowing adding and releasing new leaf cache nodes was also analyzed; remarkably network costs reduction as high as 33% can be achieved by placing transcoding close to the end users.

Acknowledgment

The research leading to these results has received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R).

References

[1] G. Hasslinger, F. Hartleb, Content delivery and caching from a network provider's perspective, Comput. Netw. 55 (2011) 3991–4006.
[2] N. Kamiyama, T. Mori, R. Kawahara, H. Hasegawa, Optimally designing ISP-Operated CDN, IEICE Trans. Commun. E96-B (2013) 790–801.
[3] Cisco, Cisco Visual Networking Index, Cisco, 2014.
Q3 [4] M. Wang, et al., An overview of cloud based content delivery networks: research dimensions and state-of-the-art, Trans. Large-Scale Data Knowl. Cent. Syst. (2015).
[5] L.M. Contreras, V. Lopez, O. González, A. Tovar, F. Muñoz, A. Azanon, J.P. Fernandez-Palacios, J. Folgueira, Toward cloud-ready transport networks, IEEE Commun. Mag. 50 (2012) 48–55.
[6] Open Networking Foundation (ONF); www.opennetworking.org. (accessed 02/2016).
[7] ETSI, ETSI GS NFV 001: Network Functions Virtualisation (NFV): Use Cases, V1.1.1, ETSI, October 2013.
[8] L. Velasco, L.M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, J.P. Fernández-Palacios, A service-oriented hybrid access network and cloud architecture, IEEE Commun. Mag. 53 (2015) 159–165.
[9] ISO Standard, Information Technology – Dynamic Adaptive Streaming Over HTTP (DASH) – Part 1: Media Presentation Description and Segment Formats: ISO/IEC 23009-1, ISO, 2014.
[10] P. Wiśniewski, et al., On delimiting video rebuffering for stream-switching adaptive applications, in: Proceedings of IEEE ICC, 2015.
[11] DASH Industry Forum. Software. http://dashif.org/software/ (accessed 02/2016).
[12] P. Georgopoulos, et al., Using software defined networking to enhance the delivery of video-on-demand, Comput. Commun. 69 (2015) 79–87.
[13] D. Vleeschauwer, D. Robinson, Optimum caching strategies for a telco CDN, Bell Labs Tech. J. 16 (2011) 115–132.
[14] Z. Avramova, D. Vleeschauwer, S. Wittevrongel, H. Bruneel, Performance analysis of a caching algorithm for a catch-up television service, Multimed. Syst. 17 (2011) 5–18.
[15] A. Blair, G. Parr, P. Morrow, B. Scotney, A. McConnell, S. Appleby, M. Nilsson, Cloud based dynamically provisioned multimedia delivery: an elastic video endpoint, in: Proceedings of International Conference on Cloud Computing, GRIDs and Virtualisation, IARIA, 2012.
[16] X. Guan, B. Choi, Push or pull? Toward optimal content delivery using cloud storage, J. Netw. Comput. Appl. 40 (2014) 234–243.
[17] X. Liu, et al., A case for a coordinated internet video control plane, in: Proceedings of ACM SIGCOMM, 2012.
[18] M. Mukerjee, et al., Practical, real-time centralized control for CDN-based live video delivery, in: Proceedings of ACM SIGCOMM, 2015.
[19] C. Lin, M. Leu, C. Chang, S. Yuan, The study and methods for cloud based CDN, in: Proceedings of IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2011, pp. 469–475.
[20] L. Velasco, A. Asensio, J. Ll. Berral, A. Castro, V. López, Towards a carrier SDN: an example for elastic inter-datacenter connectivity, OSA Opt. Express 22 (2014) 55–61.
[21] L. Velasco, A. Asensio, J.Ll. Berral, V. López, D. Carrera, A. Castro, J.P. Fernández-Palacios, Cross-stratum orchestration and flexgrid optical networks for datacenter federations, IEEE Netw. Mag. 27 (2013) 23–30.
[22] ITU-T, Requirements and capabilities for cloud computing based big data, Draft Rec. Y.BigData-reqts, July 2014.
[23] A. Asensio, L.M. Contreras, M. Ruiz, V. Lopez, L. Velasco, Scalability of telecom cloud architectures for live-TV distribution, in: Proceedings of IEEE/OSA Optical Fiber Communication Conference (OFC), 2015.
[24] ETSI, Network Functions Virtualisation (NFV). Architectural Framework, ETSI GS NFV 002 v1.1.1, ETSI, 2013.
[25] M. Bateni, M. Hajiaghayi, Assignment problem in content distribution networks: unsplittable hard-capacitated facility location, ACM Trans. Algorithms 8 (2012) 1–19.
[26] L. Mora-López, I. Martínez-Marchena, M. Piliougine, M. Sidrach-de-Cardona, Binding statistical and machine learning models for short-term forecasting of global solar radiation, in: Proceedings of the X International Symposium on Advances in Intelligent Data Analysis, 2011, pp. 294–305.
[27] R. Adhikari, A neural network based linear ensemble framework for time series forecasting, Neurocomputing 157 (2015) 231–242.
[28] Ericsson, ConsumerLab, Annual TV & Media Report, Ericsson, 2014.
[29] D. De Vleeschauwer, K. Laevens, Performance of caching algorithms for IPTV on-demand services, IEEE Trans. Broadcast. 55 (2009) 491–501.

Please cite this article as: M. Ruiz et al., Big Data-backed video distribution in the telecom cloud, Computer Communications (2016), http://dx.doi.org/10.1016/j.comcom.2016.03.026