Experimental Assessment of Big Data-Backed Video Distribution in the Telecom Cloud

Lluís Gifre[1], Marc Ruiz[2], and Luis Velasco[2*]
[1]*Universidad Autónoma de Madrid (UAM), Madrid, Spain*
[2]*Universitat Politècnica de Catalunya (UPC), Barcelona, Spain*
[*]*e-mail: lvelasco@ac.upc.edu*

ABSTRACT
The telecom infrastructure is undergoing a huge transformation since operators are deploying their own cloud infrastructure to provide cloud services and enabling Network Functions Virtualization (NFV). The resulting infrastructure is referred to as the telecom cloud. NFV decouples network functions 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.

One of the network functions currently being considered by many operators is Content Delivery Network (CDN) for live-TV and Video on Demand (VoD) distribution. In fact, a significant advance in video delivery is the standardized MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) technique that enables media content delivering over the Internet using standard HTTP web server infrastructure. Besides, the CDN reconfiguration, motivated by an increase in the number of users demanding content, entails a Virtual Network Topology (VNT) reconfiguration to cope with the demanded connection capacities.

A Big Data-based CDN manager, fulfilling the ETSI NFV guidelines, is proposed to adapt the virtualized CDN function to current and future demand. Its feasibility to control virtualized components while collecting monitoring data is experimentally assessed in a real environment.
Keywords: traffic monitoring, big data analytics, CDN, VNT reconfiguration.

1. INTRODUCTION

In the recent years, telecom operators are starting to integrate their datacenters (DC) and cloud infrastructure in their core networks, thus resulting in a huge transformation of the telecom infrastructure, which is referred to as *telecom cloud* [1]. Besides, Network Functions Virtualization (NFV) [2] decouples network functions 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. One of the network functions currently being considered by many operators is Content Delivery Networks (CDN) for live-TV and Video on Demand (VoD) distribution. By virtualizing the CDN function, CDN costs can be dynamically minimized by adapting computing and network resources to current and future users' needs while ensuring the highest quality [3].

To manage the telecom cloud, the operators require of an architectural framework providing an integrated view of their infrastructure. To cope with that need, ETSI proposed the Virtual Infrastructure Manager (VIM) in the framework of Management and Orchestration of NFV [4] to manage computing, storage and networking resources in an integrated manner.

Live-TV and VoD distribution experienced a significant advance with the standardization of the MPEG Dynamic Adaptive Streaming (MPEG-DASH) technique [5]. MPEG-DASH enables media content delivering over HTTP protocol using standard HTTP web servers' infrastructures to users' devices. In that regard, MPEG-DASH divides contents into a sequence of small file *segments*, each containing a short interval, e.g. 2 seconds, of the content and provides mechanisms to request the segmented contents. At the start of a streaming session, the MPEG-DASH client downloads a Media Presentation Description (MPD) file that contains, not only the multimedia content attributes, such as the audio and video qualities, the duration of the content, but also resource identifiers in the form of HTTP URLs for the content's segments. The user can use such MPD file to compose URLs specifying the appropriate interval of time and the quality for the desired multimedia content. It is important to note that MPEG-DASH can be used both for live-TV and VoD.

In this paper, we propose a Big Data-based CDN manager to adapt the virtualized CDN function for live-TV and VoD distribution that collects monitoring data from the caches and dynamically reconfigures them to current and future demand. The MPEG-DASH technique is used to mitigate disruptions in the video distribution service during the reconfigurations. The proposed CDN manager is integrated in a proprietary CDN architecture deployed on top of OpenStack and experimentally assessed in UPC's SYNERGY test-bed.

2. CONTENT DELIVERY NETWORK 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.

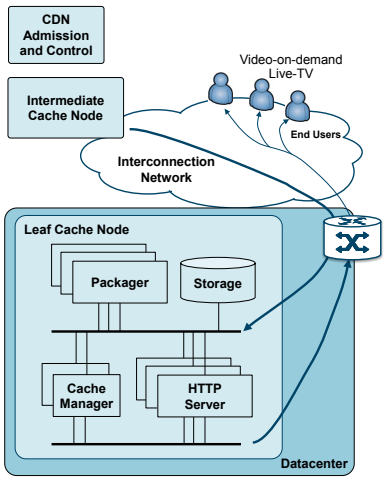

Figure 1. Virtualized leaf cache node.

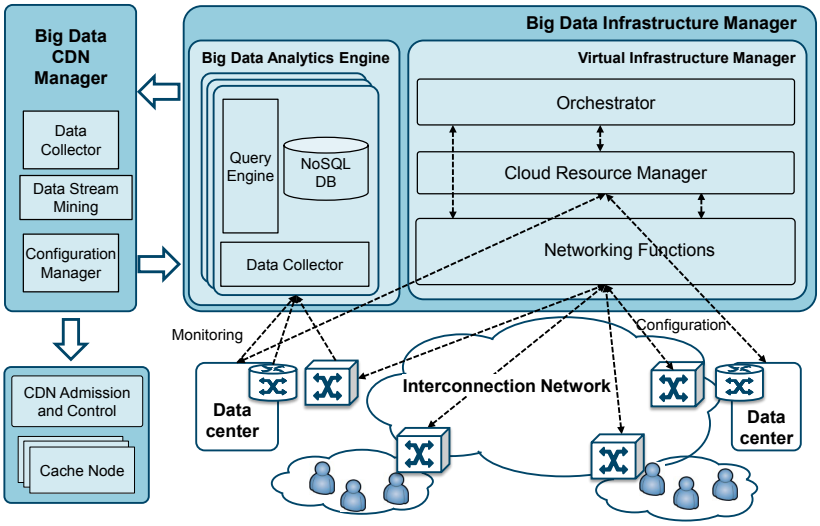

Figure 2. Architecture supporting the Big Data CDN manager.

Intermediate and leaf cache nodes distribute VoD and live-TV. VoD contents are prepared in intermediate caches and stored in leaf caches based on its popularity [6]. Live-TV is distributed from intermediate caches and locally prepared in those leaf cache nodes delivering every specific TV channel to the users, as in [7].

A virtualized leaf cache node would consist in the following components running as software inside VMs deployed in the same DC. The *packager* is in charge of live-TV preparation, including segmentation and MPD generation. The *HTTP server* component serves end users' segment requests. 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 in a pool of resources for load balancing and redundancy purposes. The amount of resources in every resource pool can be dynamically adapted in response to spikes in demand, e.g. a sports event. A Virtual Network Topology (VNT) to interconnect intermediate and leaf CDN cache nodes is created and its virtual links (*vlinks*) can be created on-demand and their capacity dynamically tuned.

Finally, a *CDN Manager* is responsible for adapting the CDN to the current and future load. To that end, the performance and load of the CDN needs to be monitored to elastically adapt its resources to current service needs. Data collected from the Big Data infrastructure manager is analysed using a Big Data application, e.g. Apache Spark [8], to provide data analysis and visualization. A configuration manager is in charge of interfacing a VIM to request and release IT and networking resources and of properly configuring every component.

In light of the above, an architecture supporting the CDN Manager is needed to control virtualized components and monitoring data collection and pre-processing functionalities. Figure 2 presents the proposed architecture, which is aligned with the ETSI NFV architectural guidelines [2].

The architecture of the Big Data infrastructure manager includes a 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 monitoring 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 engine, which collate and store the information in a horizontally scalable database (e.g., Apache Cassandra [9]).

3. EXPERIMENTAL ASSESSMENT

Dynamic re-configuration of a leaf cache node was experimentally demonstrated in our SYNERGY test-bed. A single DC with 3 physical servers interconnected by a Cisco SG220-26P switch was configured. Each physical server is equipped with an Intel i7 CPU, 16 GB of RAM, 1 TB of HDD, and a GbE NIC card, running Ubuntu 14.04.1 server. The architecture in Fig. 2 was configured, where the Big Data CDN manager runs Apache Spark 2.0.0, whilst the Big Data Infrastructure Manager consists in an Orchestrator module developed in C++, OpenContrail 3.0 [10] together with OpenStack Liberty as cloud resource manager, and running Apache Cassandra 2.1.9 as database engine. The cache node was configured as in Fig. 1, with one cache manager, two HTTP servers, and one packager. All components run Ubuntu 16.04.1 for cloud. The cache manager and HTTP servers run Apache HTTP Server 2.4.23 and the packager module runs *ffmpeg* 2.0.6 [11] for video transcoding/transrating as well as a proprietary segmenter implemented in C++.

Figure 3 lists the size of each VM extracted from OpenStack Dashboard. The packager generates two video qualities: HD (720p) at 4 Mbit/s and full HD (1080p) at 10 Mbit/s with 2 s segment duration for live contents.

Initially only one of the servers (HTTP Server 01) is active, whilst the other (Server 02) is shutdown (Fig. 3a). Monitoring data of the whole system is collected, in particular Server 01's network interface (see plot in Fig. 4 from OpenContrail). Every minute, the CDN manager submits queries to an OpenContrail analytics node to collect monitoring data (see message sequence in Fig. 5). JSON-enveloped datasets are retrieved and forwarded over TCP to a *Spark Streaming* program that receives and stores them as records in a *DStream*. At configurable intervals (e.g. 1 hour), data stream mining sketches process data available in the DStream to transform collected into modelled data that is asynchronously sent to the CDN manager. The CDN manager analyzes the received modelled data and makes decisions.

Instance Name	IP Address	Size	a) Status	Power State	b) Status	Power State
Packager 01	10.0.1.6 192.168.200.13	transcoder | 2GB RAM | 8 VCPU | 20.0GB Disk	Active	Running	Active	Running
HTTP Server 02	10.0.1.5 192.168.200.12	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	Shutoff	Shutdown	Active	Running
HTTP Server 01	10.0.1.4 192.168.200.11	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	Active	Running	Active	Running
Cache Manager 01	10.0.1.3 192.168.200.10	m1.small | 2GB RAM | 1 VCPU | 20.0GB Disk	Active	Running	Active	Running

Figure 3. VMs before (a) and after (b) re-configuration.

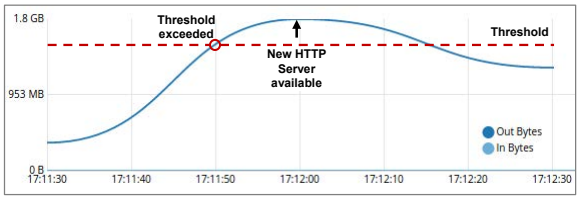

Figure 4. Capacity utilization of the NIC in HTTP Server 01.

Time	Src Addr	Dst Addr	Proto	Info
REF	CDNManager	OpenContrail	HTTP	POST /analytics/query HTTP/1.1 (application/json)
1.023655	OpenContrail	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)
1.024665	CDNManager	ApacheSpark	JSON-over-TCP	JSON-over-TCP: JSON Object: collected_data
61.088940	CDNManager	OpenContrail	HTTP	POST /analytics/query HTTP/1.1 (application/json)
62.101560	OpenContrail	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)
62.102837	CDNManager	ApacheSpark	JSON-over-TCP	JSON-over-TCP: JSON Object: collected_data
122.153510	CDNManager	OpenContrail	HTTP	POST /analytics/query HTTP/1.1 (application/json)
123.165446	OpenContrail	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)
123.166771	CDNManager	ApacheSpark	JSON-over-TCP	JSON-over-TCP: JSON Object: collected_data
174.206638	ApacheSpark	CDNManager	JSON-over-TCP	JSON-over-TCP: JSON Object: modelled_data

Figure 5. Message sequence for data collection and analysis.

Let us assume that the CDN manager decides to increase the capacity of the leaf cache by adding one more HTTP server to cache's pool and increasing the capacity of the vlinks interconnecting the leaf cache with the users' community. Thus, it sends a command to the Orchestrator through its REST API to reconfigure the computing resources (see sequence in Fig. 6). The reconfiguration starts when the Orchestrator requests an authentication token to OpenStack, gets the server list to find the Id of the instance to be started up and checks if it is shutdown. Next, it requests the Server 02 instance to be started up, keeps pooling until the instance becomes active and it returns the result to the CDN manager. Whenever the instance is active, the CDN manager sends a command to the cache manager to add the new HTTP server into the leaf cache's pool of HTTP servers. When multiple HTTP servers are available in a leaf cache, the cache manager applies a round robin policy. Figure 3b shows the final status of the computing resources in the leaf cache.

Next, the CDN manager increases the capacity of the vlinks connecting the leaf cache to the users' community (see sequence in Fig. 7), so more users can benefit of the leaf cache reconfiguration. To do so, the CDN manager issues a request to the Orchestrator through its REAT API to reconfigure the VNT. The Orchestrator first requests an authentication token to the Networking Function module and issues reconfiguration requests for each of the vlinks, "Bcn-Sab" and "Bcn-Tar" in this scenario, interconnecting the users' communities with the cache.

Time	Src	Dst	Info	
REF	CDNMgr	Orch	POST /nbi/startInstance	Orchestration
0.011	Orch	OStack	POST /v2.0/tokens	
0.078	OStack	Orch	HTTP/1.1 200 OK	Get Token
0.087	Orch	OStack	GET /v1.1/89…12/servers	
0.161	OStack	Orch	HTTP/1.1 200 OK	Get ServerList
0.162	Orch	OStack	GET /v1.1/89…12/servers/3c…6b	
0.267	OStack	Orch	HTTP/1.1 200 OK	Get ServerStatus
0.269	Orch	OStack	POST /v1.1/89…12/servers/3c…6b/action	
0.388	OStack	Orch	HTTP/1.1 202 Accepted	Server Startup
1.389	Orch	OStack	GET /v1.1/89…12/servers/3c…6b	
1.447	OStack	Orch	HTTP/1.1 200 OK	
2.450	Orch	OStack	GET /v1.1/89…12/servers/3c…6b	
2.537	OStack	Orch	HTTP/1.1 200 OK	Get ServerStatus
2.544	Orch	CDNMgr	HTTP/1.1 200 OK	
6.515	CDNMgr	CacheMgr	POST /nbi/videoServers	
6.559	CacheMgr	CDNMgr	HTTP/1.1 200 OK	Add HTTP Svr to Pool

Figure 6. Message sequence for making available a new HTTP Server in the leaf cache.

Time	Src	Dst	Info	
REF	CDNMgr	Orch	POST /reconfigVNT	Orchestration
0.003	Orch	NetFunc	GET /getToken	
0.004	NetFunc	Orch	HTTP/1.1 200 OK	Get Token
0.006	Orch	NetFunc	POST /38ea18/reconfigVLink?id=Bcn-Sab	
0.008	NetFunc	DC-BCN	OpenFlow Type: OFPT_FLOW_MOD	
0.008	NetFunc	DC-SAB	OpenFlow Type: OFPT_FLOW_MOD	Reconfig VLink
0.008	NetFunc	Orch	HTTP/1.1 200 OK	
0.010	Orch	NetFunc	POST /38ea18/reconfigVLink?id=Bcn-Tar	
0.011	NetFunc	DC-BCN	OpenFlow Type: OFPT_FLOW_MOD	
0.011	NetFunc	DC-TAR	OpenFlow Type: OFPT_FLOW_MOD	Reconfig VLink
0.012	NetFunc	Orch	HTTP/1.1 200 OK	
0.012	Orch	CDNMgr	HTTP/1.1 200 OK	

Figure 7. Message sequence for reconfiguring the leaf cache's access links.

Figure 8 reproduces the requests from a user's player for audio and video segments during the reconfiguration process. The MPD file with the definition of a VoD content is first downloaded from the cache manager. Among

others, the `media` and `initiation` attributes of the `SegmentTemplate` node in the MPD file contain URLs pointing to the cache manager and specifying the segment to be served. Initially only HTTP Server 01 is active, so the cache manager redirects every request to that server. When the second HTTP server is added, the cache manager applies round-robin load balancing among the servers in the pool. All operations are performed without any notice from user's perspective.

Figure 8. User's player requests for video and audio segments.

4. CONCLUSIONS

A Big Data -backed virtualized CDN architecture to be deployed in the telecom cloud has been proposed. The architecture is hierarchical since it consists of intermediate cache nodes that receive live-TV channels and prepare VoD contents, and leaf cache nodes, located close to the end users, to manage the access to VoD contents and adapt live-TV channels to users' devices.

A CDN manager is responsible for adapting the CDN in terms of computing, storage and networking resources to current and future load as a function of data collected from the CDN cache components. In addition, the CDN manager needs from an architecture to control the virtualized components, the VNT interconnecting the caches, and the data collection functionalities. In that regard, the CDN manager was designed fulfilling the ETSI NFV guidelines.

Aiming at minimizing the traffic disruption to the end-users, the standardized MPEG-DASH technique was used. The technique consists in dividing the multimedia content in small file *segments*, and distributing them to the users' players. Moreover, that technique enables to distribute the multimedia contents for both live-TV and VoD using standard HTTP servers.

Finally, the proposed CDN manager was implemented and its feasibility was experimentally assessed in a real environment deployed in UPC's SYNERGY test-bed. The experiments demonstrated the feasibility of dynamically reconfiguring the CDN leaf caches without perceived disruption in the user's player.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the EC through the METRO-HAUL project (G.A. no 761727), from the Spanish MINECO SYNERGY project (TEC2014-59995-R), from the MINECO/FEDER TRÁFICA project (TEC2015-69417-C2-1-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernández-Palacios, "A service-oriented hybrid access network and cloud architecture*," IEEE Comm. Mag*., vol. 53, pp. 159-165, 2015.
[2] ETSI, "Network Functions Virtualisation (NFV). Architectural Framework," ETSI GS NFV 002 v1.1.1, 2013.
[3] M. Ruiz, M. Germán, L. M. Contreras, and L. Velasco, "Big data-backed video distribution in the telecom cloud," *Elsevier Computer Communications*, vol. 84, pp. 1-11, 2016.
[4] ETSI, "Network Functions Virtualisation (NFV). Management and Orchestration," ETSI GS NFV-MAN 001 v1.1.1, 2014.
[5] ISO Standard, "Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats," ISO/IEC 23009-1, 2014.
[6] D. De Vleeschauwer and D. C. Robinson., "Optimum caching strategies for a telco CDN," *Bell Labs Technical Journal*, vol. 16, 2011.
[7] A. Asensio, L. M. Contreras, M. Ruiz, V. López, and L. Velasco, "Scalability of telecom cloud architectures for live-TV distribution," in *Proc. OFC 2015*.
[8] Apache Spark: http://spark.apache.org/
[9] Apache Cassandra: http://cassandra.apache.org/
[10] OpenContrail, http://www.opencontrail.org/
[11] ffmpeg, https://www.ffmpeg.org/