Autonomic Content Delivery Network Service

Luis Velasco[1], Lluís Gifre[2] and Marc Ruiz[1]*

(1) Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
(2) Universidad Autónoma de Madrid (UAM), Madrid, Spain
**e-mail: mruiz@ac.upc.edu*

ABSTRACT

We focus on a use case where a virtualized Content Delivery Network (CDN) service autonomously adapts to the load by requesting the instantiation of new VMs in selected leaf cache nodes, as well as by incrementing the capacity of the network connecting users with the caches. Autonomous decisions are supported by the use of data analytics techniques on monitoring data. Experimental validation supports the feasibility of the proposed autonomic CDN service.

Keywords: Autonomic Networking, Content Delivery Network, Monitoring and Data Analytics

1. INTRODUCTION

The need to provision 5G services within complex end-to-end scenarios, spanning multiple knowledge domains, technologies and administrative boundaries, while doing so dynamically and in a cost-effective way, has driven the design and refinement of functional and protocol architectures and frameworks for the operation of such telecommunication networks and infrastructures. A common trend is to design such service and resource orchestration systems by adopting, extending and building on top of frameworks that follow Software Defined Networking (SDN) and Network Function Virtualization (NFV) principles.

In the aforementioned line, the overall objective of the EU H2020 Metro-Haul project [1] is to architect and design cost-effective, energy-efficient, agile and programmable metro networks that are scalable for 5G access and future requirements, including the control, orchestration and management (COM) system that allows dynamic provisioning. Nevertheless, true autonomic operation is achieved if and only if SDN control and NFV management is augmented with instantaneous data-driven decision-making. Advanced monitoring concepts and machine learning (ML) tools enable local control loops allowing re-configuration to adapt resources to changing conditions.

Based on the Metro-Haul COM architecture, in this paper we target the adaptation of a typical operator-owned virtualized Content Delivery Network (CDN) CDN [2]; in the proposed architecture, a CDN manager is in charge of the virtualized CDN function for live-TV and video on demand (VoD) distribution. The virtualized CDN collects monitoring data from the caches and dynamically reconfigures them to fulfil current and future demand. By virtualizing the CDN function, CDN costs can be minimized by dynamically adapting computing and network resources to current and future users' needs while ensuring the highest quality [3]. Experimental validation supports the feasibility of the proposed autonomic CDN service.

2. CONTROL, ORCHESTRATION AND MANAGEMENT (COM) ARCHITECTURE

Since transport networks are increasingly segmented in domains, to enhance scalability, due to confidentiality reasons or by virtue of having non-interoperable vendor islands, Metro-haul Network Orchestration relies on an over-arching control adopting hierarchical control architectures with a parent SDN controller abstracting the underlying complexity. Regarding the control of the optical domain, disaggregated optical networks are considered in contrast to traditional proprietary optical transport networks, so networks are deployed by composing and assembling open, available components, devices and sub-systems. There are new challenges in its control and management associated to disaggregated networks, and open interfaces exporting programmability along with unified and systematic information and data modelling are adopted. The coordinated control of packet and the optical layers is needed to provision end-to-end services and steering traffic coming from aggregation networks.

Joint IT/Cloud and Network Orchestration is used to refer to the coordination of resources to deploy services and applications that require storage, computing and networking resources. The approach involves integrating the NFV management and network orchestration (MANO) functional elements with hierarchical transport network control planes, which are abstracted as WAN infrastructure managers (WIM), as show in Fig. 1.

The ETSI NFV framework can be used as a starting point for a concrete implementation of a generic slicing architecture, in which network slice instances are NFV Network Services (NS), encompassing NS endpoints and one or more VNFs interconnected by logical links, forming VNF forwarding graphs (VNFFGs). Logical links are thus mapped to supporting network connectivity services which may, in turn, span multiple network segments.

Finally, autonomic networking entails the capability to do measurements on the data plane and generating data records that are collected and analyzed to discover patterns; known as knowledge discovery from data (KDD). Such knowledge can be used to issue re-configuration/re-optimization recommendations toward COM modules, such as an SDN controller or orchestrator. In Metro-Haul, data analytics is distributed; a centralized *MDA controller* that contains a big data repository and data analytics capabilities is used to collate monitoring data records from COs. Data records and notifications are stored and processed in correlation with operational databases. In addition, every CO includes a *MDA agent* that collates monitoring data from the network, DCs and applications. The MDA agent includes a local module containing data analytics applications for handling and processing data records, and allows local control loops implementation, e.g., collecting monitoring data and applying re-configuration/re-tuning to devices in the CO.

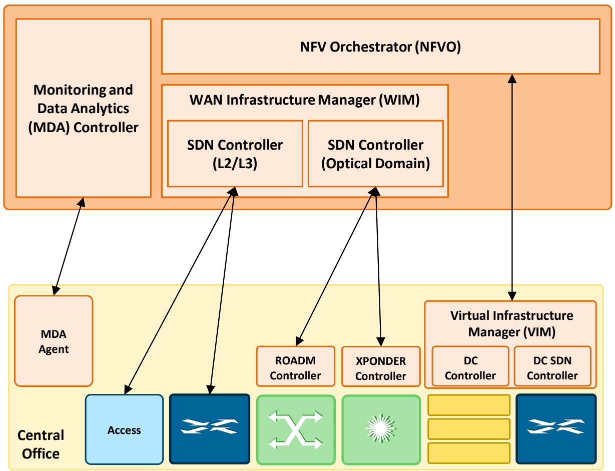

NFV Orchestrator (NFVO)
Monitoring and Data Analytics (MDA) Controller
WAN Infrastructure Manager (WIM)
SDN Controller (L2/L3)
SDN Controller (Optical Domain)
MDA Agent
Virtual Infrastructure Manager (VIM)
ROADM Controller
XPONDER Controller
DC Controller
DC SDN Controller
Access
Central Office

Fig. 1. Metro-Haul COM Architecture.

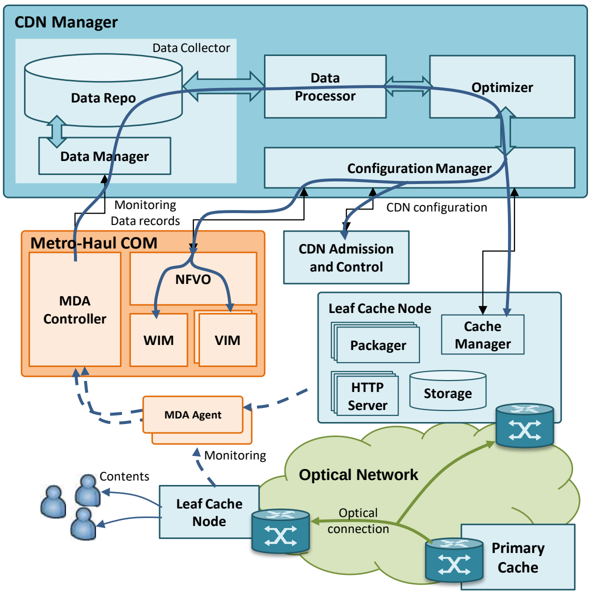


Fig. 2. Autonomous CDN service

3. VIRTUALIZED CONTENT DELIVERY NETWORK (CDN)

Many live-TV and VoD distribution systems are based on the MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) technique [4]. 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. Every time a new content is generated, a Media Presentation Descriptor (MPD) file is constructed containing all the parameters related to segment sizes, representations, codecs, file URLs, etc., required to reproduce that content. In fact, distribution of segments can be load balanced, among others, to mitigate video distribution service disruptions during the reconfigurations and spikes in demand.

Fig. 2 presents the proposed architecture and the control loops that allow to dynamically adapt the CDN. A virtualized hierarchical CDN can be deployed on the Metro-Haul infrastructure with some few primary caches that are the entrance point for new multimedia contents and a number of Leaf Cache Nodes running in COs, close to end users: a centralized CDN Admission and Control module implements CDN user access policies and redirects users' requests, e.g., based on their geographical location, to the leaf cache node that will serve them. Leaf cache nodes distribute VoD contents that are stored locally based on its popularity, whereas live-TV content is locally prepared from a raw video stream.

A virtualized leaf cache node would consist of the following components running as software inside VMs deployed in the same CO. The packager is in charge of live-TV preparation, including segmentation and MPD file 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. during a sports event. Additionally, the Cache Manager exposes an NBI to the CDN manager to allow its remote reconfiguration.

Finally, a CDN Manager is responsible for adapting the CDN to the current and future load. To that end, monitoring data is collected from the MDA controller and analyzed; specifically, the logs from the cache manager that contain useful information regarding user activity and contents access together with the activity of the packagers and HTTP servers are analyzed. Specifically, contents' popularity can be computed not only at the leaf cache level, but also at the CDN manager level. This enables reassigning users among leaf caches without affecting the structure of the cached contents. The analyzed performance and load of the CDN is used to elastically adapt its resources to current and near future service needs.

To illustrate how the targeted elastic adaptation can be realized, we state a possible optimization problem that a CDN operator might want to solve, so as to use resources as a function of the demand. Hence, the Dynamic CDN Adaptation problem can run periodically, e.g., every hour, focusing on minimizing the CDN cost by using more

efficiently the resources and releasing unused resources, as well as on minimizing the impact of the reconfiguration on the users being served, while ensuring the committed Quality of Service (QoS).
The problem is next formally stated. *Given: i*) a set *LC* of leaf cache locations; *ii*) a set *A* of metro areas with users consuming contents; *iii*) a set *C* of multimedia contents being consumed, e.g., an episode of a TV series; *iv*) a set *F* of content formats, e.g., high definition (HD), full HD, ultra HD, each of then containing the maximum size of its segments (in MB) and the maximum delay to be ensured; *v*) the capacity (in terms of number of VMs that can be requested) and the number of VMs currently used of every *LC* location; *vi*) the current assignments of users to areas, contents, and formats, as well as the leaf cache from where they are being served; *vii*) the current connectivity between areas and LC locations in terms of capacity of the connection, round-trip time, and effective throughput; and *viii*) The number of users in an area, content and format to be considered for the next period t+Δt.
A solution of the problem consists of: *i*) the configuration of every location in terms of number of VMs; *ii*) the assignment of every set of users in an area consuming a content in a given format to a given location; and *iii*) the capacity of every connection between areas and locations, limited to a maximum. The objective of the problem is to minimize the CDN cost from serving the demand for the next period, while guaranteeing the committed QoS in terms of maximum delay and minimize the migration costs from the current configuration to the new one.
Although most of the input data (sets and parameters) of the problem can be precisely determined, it is worth highlighting that the number of users for which the CDN is optimized requires from some sort of prediction. Such demand has to be estimated for the next optimization period; ML algorithms can be used to produce a predictive model for every tuple area-content-format, based on monitoring data available in the Data Repository. In addition, ML-based models can be estimated to predict the evolution of the relative popularity of a group of contents, e.g., a TV series, with time; these models would require a longer period of time to be produced, e.g., months. In addition to these two groups of predictive models based on monitoring data, some other information, that might be represented as well as models, is the so-called contextual; it includes the contents plans, the competence counter-programming, special events, etc. With all such sources of predictions, the estimated number of users can be computed. Fig. 3 presents a possible approach for the CDN manager.

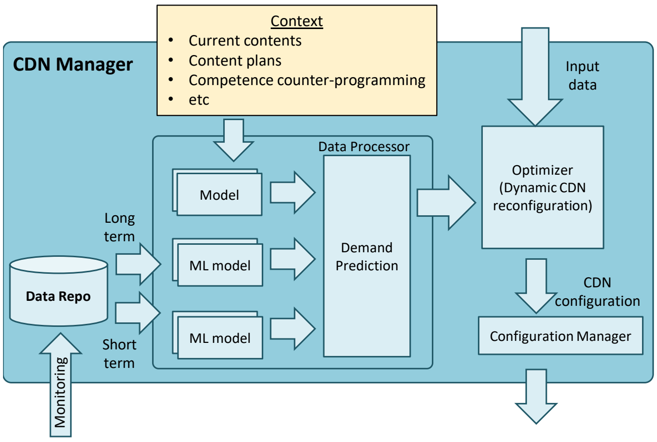


Fig. 3. Data generation and optimization in the CDN Manager

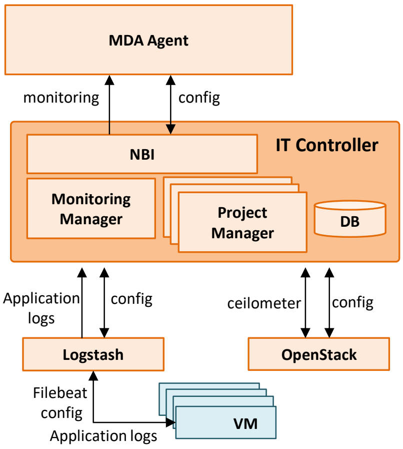


Fig. 4. Measurement collection

4. EXPERIMENTAL ASSESSMENT

Experimental setup includes the CDN service, running in a OpenStack Pike DC, the NFVO, and the MDA system based on CASTOR Monitoring and Data Analytics platform [5]. The CDN manager and cache manager were developed in Python. Communication between the CDN manager and the MDA controller and NFVO is performed through ad-hoc, bidirectional REST APIs. The Ceilometer module in OpenStack has been used to periodically send measurements such as CPU, disk, and network statistics toward the local MDA agent. In addition, *Filebeat* is used to export applications' logs toward a centralized *Logstash* instance, which is in charge of processing and forwarding them. Both, Ceilometer and Logstash use HTTP to export monitoring data. An IT controller has been developed in Python as an intermediate module between MDA agents and OpenStack and Logstash aiming at reducing the complexity of the MDA agent (see Fig. 4). The IT controller includes a Monitoring Manager in charge of the monitoring configuration and a pool of Project Managers where each manager is in charge of resource tracking of individual OpenStack project. Extended details of the experimental setup can be found in [6].
For demonstration purposes, a CDN service with two leaf caches serving contents was set up. Up to 1400 users were distributed in three different metro areas, where video segment requests (we defined video segments of 2 seconds) were emulated according to the number of active end users at every time instant. Finally, 10 different contents were released at different time instants according to a pre-defined schedule. In consequence, the load of the CDN was synthetically generated, based on the generic users' evolution and the popularity of contents (see Fig. 5a-b), among others.

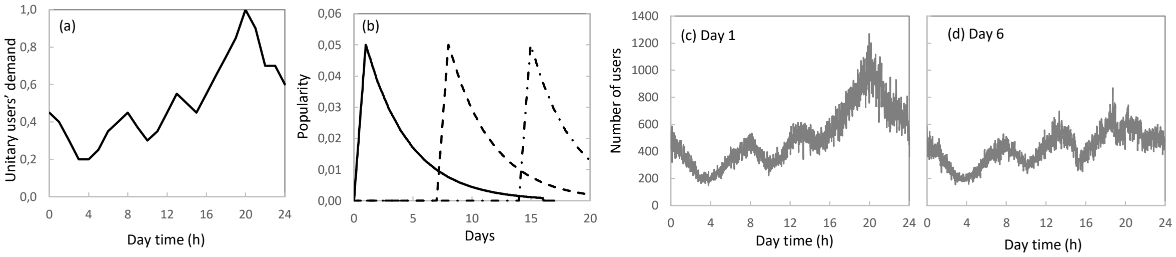
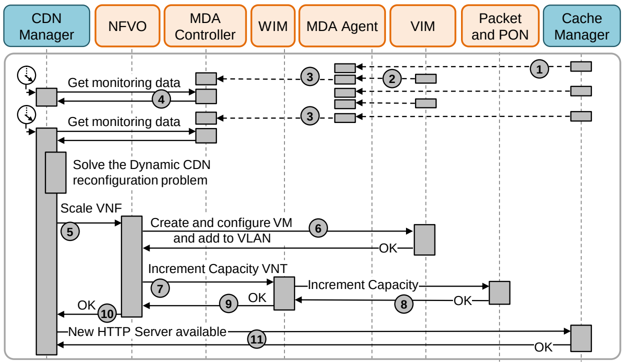


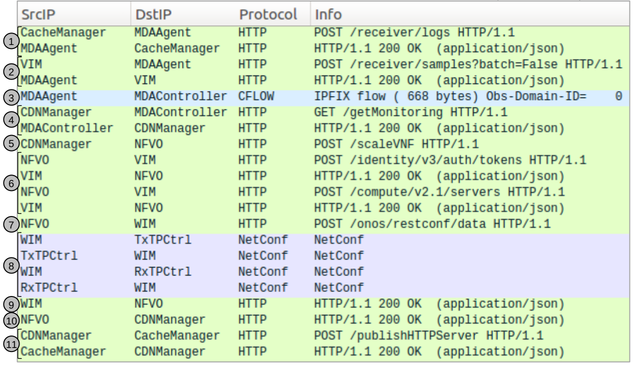
Fig. 5. Users daily profile (a); contents release and popularity evolution (b); active user during day 1 (c) and day 6 (d)

To illustrate how the users' demand can largely change from one day to another as a function of contextual information, Fig. 5c-d reproduce the variation of the number of active users for two different days. Differences can be as a result of a sports event, or some sort of counter-programming that has attracted a large number of users.

Fig 6 presents the proposed workflow and the capture of messages exchanged using the same message numbering for the sake of clarity. A REST API has been used for the Cache Manager and the VIM to send monitoring data periodically to the local MDA agent (messages 1 and 2), whereas IPFIX is used between MDA agents and the MDA controller (3). The CDN manager uses a REST API to retrieve monitoring data for a set of OPs from the MDA controller (4) and uses such measurements to decide scaling leaf caches and connectivity (5). In the demonstration, one HTTP server is added to one of the leave caches and the connectivity capacity between users and that cache is increased (6-9).

Finally, once the VNF has been scaled, the new HTTP servers are announced to the Cache Manager (11) by means of an ad-hoc REST API.





	SrcIP	DstIP	Protocol	Info
1	CacheManager	MDAAgent	HTTP	POST /receiver/logs HTTP/1.1
	MDAAgent	CacheManager	HTTP	HTTP/1.1 200 OK (application/json)
2	VIM	MDAAgent	HTTP	POST /receiver/samples?batch=False HTTP/1.1
	MDAAgent	VIM	HTTP	HTTP/1.1 200 OK (application/json)
3	MDAAgent	MDAController	CFLOW	IPFIX flow (668 bytes) Obs-Domain-ID= 0
4	CDNManager	MDAController	HTTP	GET /getMonitoring HTTP/1.1
	MDAController	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)
5	CDNManager	NFVO	HTTP	POST /scaleVNF HTTP/1.1
6	NFVO	VIM	HTTP	POST /identity/v3/auth/tokens HTTP/1.1
	VIM	NFVO	HTTP	HTTP/1.1 200 OK (application/json)
	NFVO	VIM	HTTP	POST /compute/v2.1/servers HTTP/1.1
	VIM	NFVO	HTTP	HTTP/1.1 200 OK (application/json)
7	NFVO	WIM	HTTP	POST /onos/restconf/data HTTP/1.1
8	WIM	TxTPCtrl	NetConf	NetConf
	TxTPCtrl	WIM	NetConf	NetConf
	WIM	RxTPCtrl	NetConf	NetConf
	RxTPCtrl	WIM	NetConf	NetConf
9	WIM	NFVO	HTTP	HTTP/1.1 200 OK (application/json)
10	NFVO	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)
11	CDNManager	CacheManager	HTTP	POST /publishHTTPServer HTTP/1.1
	CacheManager	CDNManager	HTTP	HTTP/1.1 200 OK (application/json)

Fig. 6. Workflow and message list

5. CONCLUSIONS

Self-adaptive virtualized services based on the Metro-Haul COM architecture has been presented. Specifically a CDN has been proposed to show how the service autonomously adapts to the load by requesting the instantiation of new resources and increasing the capacity of existing ones. An MDA system supports autonomous decisions using data analytics techniques on monitoring data.

ACKNOWLEDGEMENTS

This work was partially supported by the EC through the METRO-HAUL project (G.A. nº 761727), from the AEI/FEDER TWINS project (TEC2017-90097-R), and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] METRO High bandwidth, 5G Application-aware optical network, with edge storage, compUte and low Latency (Metro-Haul). [On-line] https://metro-haul.eu.

[2] S. Spagna, et al. "Design Principles of an Operator-Owned Highly Distributed Content Delivery Network," IEEE Communications Magazine, vol. 51, pp. 132-140, 2013.

[3] M. Ruiz *et al*., "Big Data-backed Video Distribution in the Telecom Cloud," Elsevier Computer Communications, vol. 84, pp. 1-11, 2016.

[4] ISO Standard, "Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats," ISO/IEC 23009-1, 2014.

[5] L. Velasco *et al*., "An Architecture to Support Autonomic Slice Networking [Invited]," IEEE/OSA Journal of Lightwave Technology (JLT), 2018.

[6] L. Velasco *et al*., "A Control and Management Architecture Supporting Autonomic NFV Services," Springer Photonic Network Communications, vol. 37, pp. 24-37, 2019.