NFV and SDN-based Cost-efficient and Agile Value-added Video Services Provisioning in Content Delivery Networks

Narjes T. Jahromi†, Sami Yangui†, Adel Larabi‡, Daniel Smith‡, Mohammad A. Salahuddin†*, Roch H. Glitho†, Richard Brunner‡, Halima Elbiaze*
†Concordia University, Montreal, QC, Canada
‡Ericsson, Montreal, QC, Canada
*Université du Québec À Montréal, Montreal, QC, Canada

***Abstract*— Due to the recent surge in end-users demands, value-added video services (e.g. in-stream video advertisements) need to be provisioned in a cost-efficient and agile manner in Content Delivery Networks (CDNs). Network Function Virtualization (NFV) is an emerging technology that aims to reduce costs and bring agility by decoupling network functions from the underlying hardware. It is often used in combination with Software Defined Network (SDN), a technology to decouple control and data planes. This paper proposes an NFV and SDN-based architecture for a cost-efficient and agile provisioning of value-added video services in CDNs. In the proposed architecture, the application-level middleboxes that enable value-added video services (e.g. mixer, compressor) are provisioned as Virtual Network Functions (VNFs) and chained using application-level SDN switches. HTTP technology is used as the pillar of the implementation architecture. We have built a prototype and deployed it in an OPNFV test lab and in SAVI, a Canadian distributed test bed for future Internet applications. The performance is also evaluated.**

Keywords*— *Cloud Computing, Content Delivery Network (CDN), Network Function Virtualization (NFV), Software-defined Network (SDN)

I. INTRODUCTION

Content Delivery Networks (CDNs) enable the delivery of services like video to a large number of geographically distributed users [1]. Original contents located on origin servers are replicated on surrogate servers, (a.k.a replica servers), strategically located in the network for delivering content to end-users with the required quality of service (QoS). An entity known as CDN controller selects the specific surrogate server each individual end-user request should be routed to.

The recent surge in demand for rich and premium content has led to the provisioning of value-added video services such as in-stream video advertisements and optimized video delivery in addition to the basic and traditional video services. In-stream video advertisements consist of streaming commercials before, during or after the streaming of the videos requested by the end-users. Video optimization enhances the viewing experience of the end-users in a differentiated manner. Meanwhile, there are many other value-added video services, as the reader should note. Instead of commercials, sign language, for instance, could be inserted during the video streaming, leading to another value-added video service.

The building blocks of video value-added services are generally application-level middleboxes according to the IETF categorization [2]. Transcoder, mixer and compressor are among the examples. They are provisioned at fixed network locations and on a proprietary/dedicated hardware, like the way most middleboxes are implemented today. The shortcomings of this traditional mode of provisioning middleboxes are widely known and subject to high costs and lack of agility. It is also well known that they could be tackled by integrating emerging technologies such as Network Function Virtualization (NFV) and Software Defined Network (SDN) [3][4].

NFV decouples the software implementation of network functions from the underlying hardware and enables deployment anywhere anytime. Meanwhile, SDN enables the easy orchestration of these network functions, known as Virtual Network Functions (VNFs), by decoupling the control and data planes. Provisioning cost-efficient and agile value-added video services in CDN is rather challenging. This is essentially due to the application-level nature of the middleboxes and the fact that research so far has mainly focused on IP-level middleboxes provisioning.

This paper proposes and validates an NFV and SDN-based architecture for cost-efficient and agile value-added video service provisioning in CDN. Section II introduces a motivating scenario and discusses the related work. Section III is devoted to the proposed architecture. Section IV presents the validation prototype and the evaluation experimentations. Section V concludes the paper and discusses the lessons learned.

II. MOTIVATING SCENARIO & RELATED WORK

Let us first illustrate the problem space by the Graphical User Interface (GUI) of a few potential value-added services a CDN provider may wish to offer to its customers. As shown by the GUI in Fig.1, the video streaming remains the basic service. However a hearing-impaired end-user might select the streaming with the insertion of sign language as a value-added service. Some end-users may select higher quality streaming for a higher price while others may select lower

978-1-5090-6196-9/17/$31.00 ©2017 IEEE

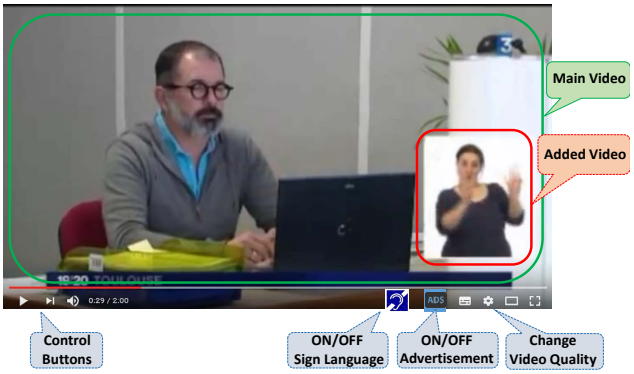

Fig. 1 GUI of a hearing-impaired/advertisement value-added video application

quality (or potentially the insertion of commercials) for a lower price. The application-level middleboxes that these value-added services could rely on are mixer, compressor, and transcoder. The on-the-fly deployment, dynamic placement and dynamic composition of these middleboxes are the key requirements for provisioning these value-added services in a cost-efficient and agile manner. As NFV and SDN are the ideal technology candidates, below, we review the work done so far on NFV and SDN in the CDN space and also the work done so far on the use of NFV and SDN for provisioning applications at large in other domains.

A. NFV and SDN in the CDN Domain

Some work already attempted to combine NFV and SDN in the CDN domain. However, their main contributions concerns IP-level routing rather the application-level we are proposing. For example, in [5], the authors propose an IP-level VNFs system for Traffic Monitoring and Network Embedder. It interact with an SDN Controller in order to shape the traffic and to provide privileged service for premium applications.

In what follows, we discuss the related work that discuss NFV for the CDN domain approaches, as well as, the works that focus on SDN for the CDN domain.

1) NFV for the CDN Domain:

In the literature, the work that discuss NFV for the CDN domain mainly focus on the CDN components rather than investigating the provisioning of value-added services. An example is the use case introduced by the European Telecommunications Standards Institute (ETSI) [6]. It motivates the need for CDN providers to deploy on-the-fly and manage surrogate servers as VNFs in a telecommunication network operator domain, but it overlooks the problem of on-the-fly deployment, dynamic placement and dynamic composition of application-level middleboxes for value-added service provisioning.

Yet, the case study on surrogate servers designed as VNFs is another example [7]. This work focuses on testing surrogate servers as VNFs before deployment in order to assess the effect of various factors such as the use of analytics and the level of granularity on the baseline performance. This work tackles the VNF deployment issue from the testing perspective. However, it does not meet any of our three key requirements.

In the CDN context, Ref [8] is another related work with focus on the algorithmic aspect of the VNF placement. This work formulates the problem of cost and resource optimization of surrogate server placement for various service function chains. It tackles placement to some extent but overlooks deployment and composition.

2) SDN for the CDN domain:

Reference [9] proposes the use of programmable storage routers instead of surrogate servers. They are controlled by an SDN controller augmented with functionality such as request re-routing and load balancing. Videos are periodically pulled from the origin servers by these storage routers. End-user requests are first sent to the controller redirecting it to the storage router. If this router is able to serve the request, it sends the video directly to the user. Otherwise, it forwards the video to the next storage that is able to serve it. In the worst case, the video is served by the origin server. This paper focuses on the basic video service; however, how the value-added video services could be developed, deployed and composed are out of the scope of this discussion.

Among other papers on the use of SDN in the CDN space, to the best of our knowledge, none tackles basic video service provisioning; nor have they discussed the value-added video service provisioning. An example is reference [10]. It proposes an architecture that enables CDN providers and Internet Service Providers (ISP) to manage high volume flows.

B. NFV and SDN in domains other than CDN

Beyond the use of NFV for the CDN domain, several papers investigate the NFV and SDN paradigms in areas other than CDN. However, the research focus here remains on the IP-level middleboxes and network services.

1) NFV in domains other than CDN:

In [11], the authors propose an NFV-based architecture for provisioning gateways for Virtualized Wireless Sensor and Actuator Network (VWSAN). The VNFs implement protocol converters and information model processors between the VWSAN and the applications. This work focuses on the VNF development and deployment. VNF chaining is done in a static manner while the dynamic chaining and placement of VNFs are not addressed.

In [12], the researchers focus on dynamic chaining, implementing two topology alternatives using SDN controller to illustrate the feasibility of dynamic VNF chaining. The first is a Layer 2 topology where VNFs are deployed as Data Link devices and work within a single network. The second topology is Layer 3 where VNFs are implemented as IP layer devices and connected to two different networks. Their VNFs include IP-level VNFs such as Deep Packet Inspection (DPI). The goal is to enable a differentiated QoS for the privileged users who require acceleration in flow steering. This architecture mainly focuses on a dynamic chaining of IP-level VNFs but does not cover the VNF placement and deployment issues.

In [13], OpenSCaaS focuses on dynamic chaining as well. It proposes an architecture of a flexible and adaptable VNF deployment and chaining framework. The architecture

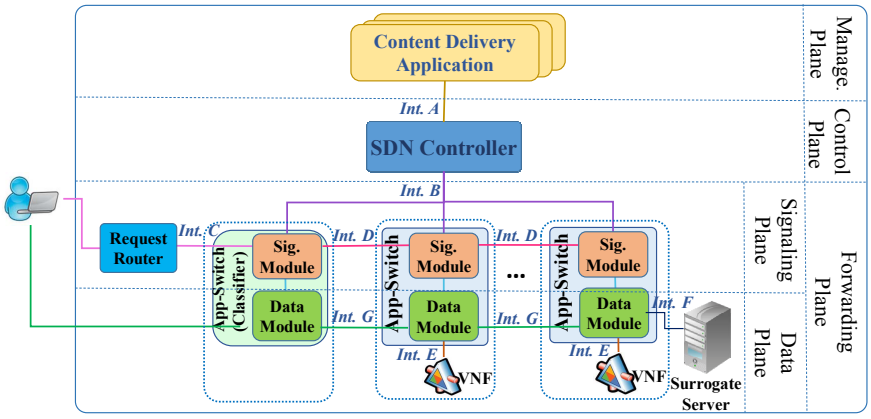

Fig. 2. High-Level Architecture

comprises of NFV, SDN and Policy Control modules and packages the VNFs into lightweight ClickOS [14] VM images. The NFV control module maps the logical chains to the physical resources and deploys the VM images on the commodity hardware. OpenSCaaS focuses on generic IP-level VNFs, such as NAT and Firewall, but provides no solutions for the dynamic VNF placements.

2) SDN in domains other than CDN:

Several papers have been published on SDN in domains other than CDN. In this paper, we focus on the articles related to the application layer. OpenADN [15] is one example. It provides a platform for flow management based on application-level context, such as end-user device type, link conditions, mobility, etc. OpenADN uses a cross layer design to insert application traffic flow information in a header between the transport and network headers. Such information is used by OpenADN-aware switches that operate at IP-layer. This work does not address a dynamic chaining of application-level VNF. It also overlooks the on-the-fly VNF placement and deployment issues.

III. Architecture for Value-added Video Services Provisioning

In this section, the architectural principles are first introduced. Then, the high-level architecture of the designed system, including its related architectural planes, components and interfaces, is discussed.

A. Architectural Principles

A value-added video could be a raw video to which value is added. This is done by appropriate middleboxes at each hop of a prescribed path between the surrogate server holding the raw video and the end-user. The first architectural principle is the implementation of the middleboxes as VNFs and the use of SDN to dynamically provision the paths once they are deployed. In this paper, the middleboxes are assumed to have already been implemented as VNFs and deployed in the network. The focus is therefore on how to dynamically provision the paths, i.e. dynamically compose the VNFs. The reader should note that the prior development of these middleboxes as VNFs and their deployment in the network could be done by the CDN provider itself or a third party as assumed in a previous work [11] for the development and deployment of IoT gateways as VNFs.

The second architectural principle is the establishment of a signaling path between the end-user and the surrogate server that holds the raw video. This path includes the VNFs that add more and more value to the video as it journeys from the surrogate server to the end-user.

B. High-Level Architecture

A high-level view of the designed architecture is depicted in Fig. 2. The proposed architecture is layered over a set of planes. These planes and the contained components are discussed here. We also discuss the interfaces between the components in this section.

1) Architectural planes: Similar to the traditional SDN-driven architectures, the proposed architecture consists of *management*, *control* and *forwarding planes*. The *management plane* handles the value-added video delivery policies. These policies are embedded in a set of value-added video delivery applications that specify how the middleboxes, implemented as VNFs, should be composed for specific value-added video services. The *control plane* enforces the policies of the *management plane* by programming the *application-level SDN switches* of the *forwarding plane*. The *control plane* includes an *SDN controller* that interacts with the architectural components of the *forwarding plane*. The reader should note that our *forwarding plane*, unlike the traditional SDN forwarding planes, has a *signaling sub-plane* in addition to the regular *data plane*. This is due to our second architectural principle that stipulates the establishment of a signaling link between the *end-user* and the *surrogate server* that holds the raw videos. The *forwarding plane* consists of the following architectural components: *CDN controller*, *application-level switches*, *surrogate server* and *VNFs*. The next subsection discusses the purpose of each one of these components.

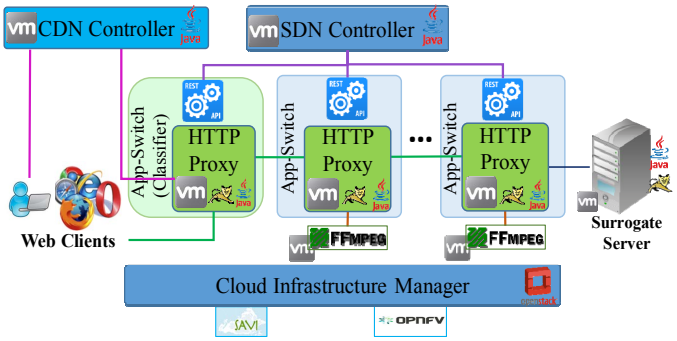

Fig. 3. The developed prototype architecture.

2) Architectural components: In what follows, we mainly focus on the components of the *forwarding plane* since they are the innovative components in this architecture. The main component is the *application-level switch*. Unlike the traditional SDN switches, such a switch is stateful. It has two modules: *Signaling module* that is related to the *signaling sub-plane* and *data module* related to the *data sub-plane*. On one hand, the *signaling module* is programmable by the *SDN controller*. The *signaling module* establishes the path between VNFs, through which the raw video should go for a value adding purpose. On the other hand, the *data module* is responsible for redirecting the raw video through associated VNFs. The first *application-level switch* is the *flow classifier*. It extracts the end-user preferences (e.g. content quality, formats) and, eventually, tags the request with a chain-ID. The chain-ID serves to identify the packets belonging to a given chain of VNFs. The other components are the *CDN controller*, the *surrogate server* and the *VNFs*. Unlike regular CDN controllers that redirect the request directly to the optimal *surrogate server* in order to optimize latency, our *CDN controller* is designed to redirect the end-user requests to the *flow classifier*. This is due to our second architectural principle. The *Surrogate Server* holds a copy of the raw content. *VNFs* add value to the content, for instance, by inserting overlay content (e.g. mixing) and optimizing it (e.g. transcoding, compressing). Each *VNF* is linked to a dedicated *application-level switch* following a linear topology model as it is depicted in Fig. 2.

3) Interfaces: In what follows, the main interfaces are described. *Int. A* is the northbound *SDN controller*. It is known as a very high-level interface and we do not make any changes to it. On the other hand, we extend *Int. B*, the southbound API that is a lower-level abstraction interface. Our enhancements consist of adding application-level fields to the matching fields of the flow rules. Examples are the standard fields (e.g. application-level addresses) and extended fields (e.g. content quality in HTTP). These extensions are indispensable for routing messages between *application-level switches* and for manipulating the

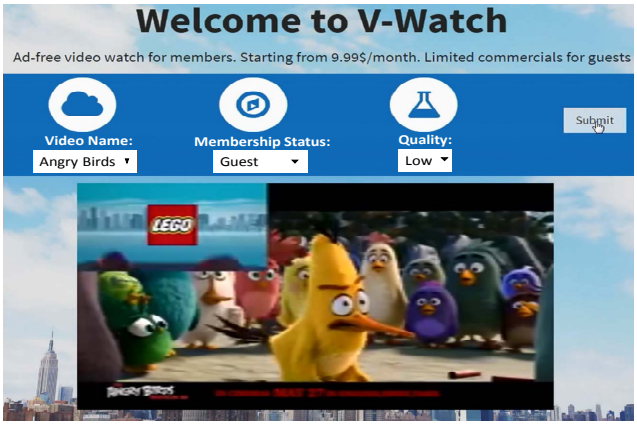

Fig. 4. Web GUI of the developed prototype.

application-level messages to add value to the videos. The other interfaces are: *Int. C* (between the *CDN controller* and the *flow classifier*), *Int. D* (the signaling interface between the *application-level switches*), *Int. E* (between *VNF* and *application-level switch* for pushing/retreiving the raw/value-added video), *Int. G* (the data interface between the *application-level switches*), and *Int. F* (the last *application-level switch* and the *surrogate server* for pulling out the raw video).

IV. Implementation & Experimentations

This section details the implementation and the experimentations we performed in order to validate and evaluate our findings. We first present the developed prototype architecture. Then, we discuss an illustrative sequence diagram that implements dynamic chaining of mixer and compressor VNFs in order to deliver ad-inserted low quality video. Finally, we present and comment on the experiments we performed to evaluate the end-to-end delay when delivering a value-added video using our prototype.

A. Implementation architecture

The validation prototype was implemented according to the architecture depicted in Fig. 3. All the developed components are deployed on Virtual Machines (VMs) provisioned using OpenStack IaaS Manager. The latter is provided by either the Smart Applications on Virtual Infrastructure (SAVI) testbed[1] or Open Platform for NFV OPNFV[2] test lab since we have used both for the validation purpose. SAVI is a Canadian distributed test bed for future Internet applications while OPNFV is a new open source project with focus on accelerating NFV technology evolution.

We have used HTTP as the basis of our implementation since it is pervasive in content delivery settings. The *application-level switches* are extended HTTP proxies. Specifically, they were implemented as Java servlets and packaged as Java Web ARchives (WAR). They also expose a REST API to the SDN controller for the application-level rule handling. The API was implemented as a REST Java server

[1] savinetwork.ca

[2] opnfv.org

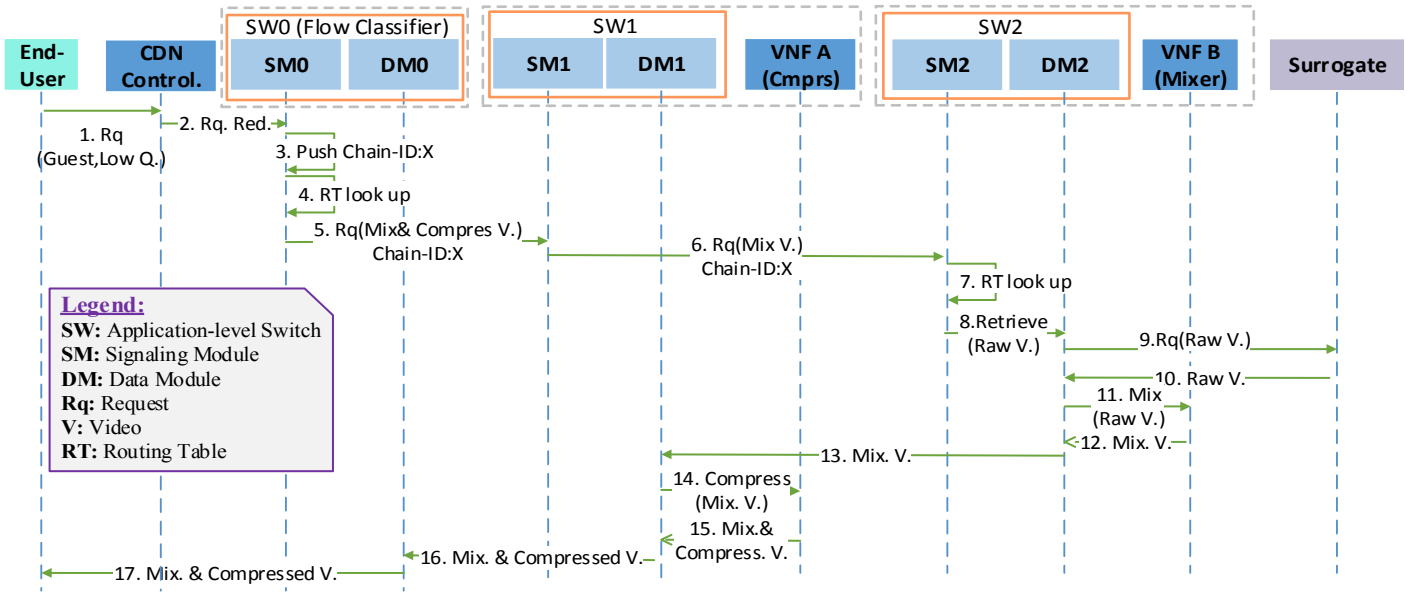

Fig. 5. Sequence diagram of value-added video delivery.

while the *SDN Controller* was implemented as a REST client of that API. Both are based on the same Restlet3 API.

For the VNF implementation, we used the FFmpeg server – an open source software4 that provides libraries and tools to handle multimedia data. We installed and configured it in order to provide mixing and compressing features, respectively.

The *CDN controller* and the *surrogate server* were implemented as Java servlets. The reader should note that we used Apache Tomcat container to host and execute all the developed servlets. It should also be noted that the interfaces in *forwarding plane* were implemented according to HTTP rendering (i.e. *Int. C* and *Int. D* as HTTP requests and *Int. G* as HTTP responses). Finally, we designed a Web GUI that allows end-users to connect to the developed prototype and request for the added value content videos. A screenshot of the GUI is depicted in Fig. 4. The GUI shows a guest end-user requesting a low quality version of the Angry Birds video. The final video is customized by a LEGO advertisement inserted on top of it and the quality is degraded to match the end-user preferences.

TABLE I. PROSPECTIVE CHAINS ACCORDING TO THE END-USER'S PREFERENCES.

	End-user preferences		Final Video	Chain of VNFs
	Membership	Quality		
1	Registered	High	Raw Video	No VNF
2	Registered	Low	Compressed Video	Compressor
3	Guest	High	Mixed Video	Mixer
4	Guest	Low	Mixed & Compressed Video	Mixer, Compressor

TABLE II. APPLICATION-LEVEL FLOW RULES.

Rule	App-Switch ID	Application-level Flow Rule	
		Matching field	Action
1	SW0	Membership=Guest & Quality=Low	Push tag Chain-ID:X
2	SW0	Chain-ID=X	Output to SW1
3	SW1	Chain-ID=X	Output to SW2
4	SW2	Chain-ID=X	Output to Surrogate

B. *Illustrative sequence diagram:*

The envisioned validating scenario implements dynamic chaining of mixer and compressor VNFs in order to deliver ad-inserted low quality video. We assume a content provider that offers ad-free video clips for the registered end-users who pay monthly fees for watching videos. For guest log-in, the content provider inserts an advertisement on top of the raw video. Besides, the end-user might select the lower quality video for a lower price or less bandwidth usage. To that end, two VNFs are needed: (i) Mixer to insert an advertisement and (ii) a compressor to degrade the video quality. Consequently, we need two *application-level switches* (one for each VNF) and one *flow classifier*.

TABLE I shows four prospective chains that could be defined according to the end-user's preferences. In this illustration, we consider the fourth case, where the end-user logs in as a guest and requests the low quality video. The raw video, which is initially stored in the *surrogate server*, should then pass through the mixer and the compressor. To make this happen, a set of policies/chains is defined by the value-added video delivery applications. Afterward, a set of application-level flow rules is injected into the *application-level switches* by the *SDN controller*. An abstracted form of these rules are listed in TABLE II given that *SW0* is the *flow classifier*.

Fig. 5 depicts a sequence diagram showing the interaction of the architectural components to enable this scenario.

[3] restlet.com

[4] github.com/ffmpeg/ffmpeg

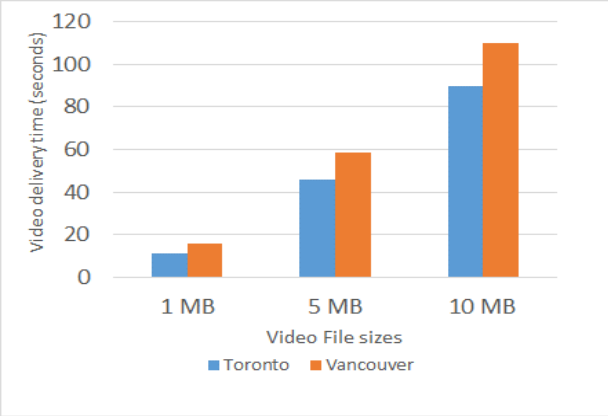

Fig. 6. Video delivery delay for VNFs deployed in Toronto vs. Vancouver for the end-users in Toronto.

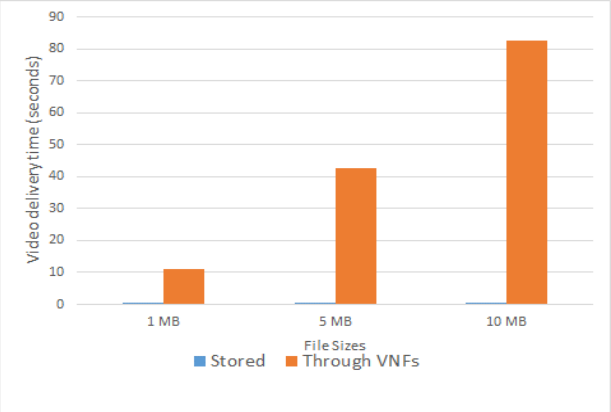

Fig. 7. Video delivery latency for various file sizes.

Firstly, a guest end-user sends a request for a low quality video (Fig.5, action 1). The *CDN controller* redirects the received request to the *flow classifier* (action 2). The *SM0* performs the flow classification. It receives the request packet and extracts application-level header values such as end-user membership status (e.g. guest) and required video quality (e.g. low). Then, it pushes the chain-ID on the request (action 3) according to a matching rule (TABLE II, Rule 1) in its Routing Table (RT). The chain-ID indicates that the mixed and compressed video is being requested. Using this chain-ID (TABLE II, Rule 2), *SM0* selects *SW1* as next hop (action 4). After that, it initiates a request for the mixed and compressed video (action 5). Once the request, with chain-ID:X, is received by *SM1*, it initiates a request to the next hop (action 6) i.e. *SW2* (TABLE II, Rule 3). Similarly, *SM2* receives the request with Chain-ID:X. *SM2* finds the *surrogate server* as the next hop (TABLE II, Rule 4). It delegates the video retrieval to the *DM2* (action 8). *DM2* initiates a request for the raw video to the *surrogate server* (action 9). The *surrogate server* delivers the raw video to the *DM2* (action 10). Then, *DM2* pushes it to the mixer for an ad-insertion and receives the mixed video back (actions 11, 12). Eventually, following the signaling path saved by the stateful signaling module, it pushes the mixed video to *DM1* (action 13). *DM1* pushes the mixed video to the compressor and receives then the customized video (actions 14, 15). *DM1* sends the customized video to the *DM0* (action 16) to serve the *end-user* with the requested customized video (action 17).

C. Measurements

To obtain some insights about the gains associated with the dynamic compositions of application-level VNFs for value-added video delivery, two sets of experimentations have been done. The first experiment consists of varying the locations of the VNFs between two cloud sites, one in Vancouver and the other in Toronto, and measuring the corresponding value-added video delivery delay, for an end-user located in Toronto. The second set of experiments measures value-added video delivery when the raw video needs to pass through VNFs to gain value, vis-à-vis when the value-added video is stored on the surrogate server and is delivered to the end-users directly.

In order to conduct the experiments, an environment was set up as follows: To each *application-level switch*, *VNF*, *CDN controller* and the *surrogate server*, a medium-size OpenStack flavor were allocated with 4 GB of RAM, 2 vCPUs and 40 GB of Disk. For both scenarios, the *application-level switches*, *CDN controller*, the *surrogate server* and the *end-user* are deployed in the Toronto site. Each scenario is followed for a raw video with various sizes of 1, 5, and 10MB and the measurements are taken for 10 times.

Fig. 6. shows the obtained video delivery values. For example, for 5MB raw video file, the latency is 58.2358 seconds for VNFs located in Vancouver and it is reduced to 45.6654 seconds when it is located in Toronto. The results show the significant impact of the physical location of VNFs on the end-user perceived latency.

Fig. 7 shows the value-added video delivery latency in two cases. The first case consists on passing the raw video through the required *VNFs* in order to get customized. While the second case consists on storing the customized video in the *surrogate server* to avoid going through the *VNFs*. For example, for a 5-MB raw video file, the latency in the first case is 42.7198 seconds and reduced to 0.0566 seconds in the second case. This result highlights the significant overhead of the dynamic chaining.

V. CONCLUSION & LESSONS LEARNED

This paper proposes an architecture for a cost-efficient and agile provisioning of value-added video services in CDN using NFV and SDN technologies. The application-level middleboxes that provide value-added video services are provisioned as VNFs and chained on-the-fly using application-level SDN switches. A prototype is implemented. Measurements have also been made. A first lesson learned in this work concerns the location of the VNFs. It has a critical impact on the end-to-end delay. We will research algorithms in the future for optimally placing these VNFs. Yet another lesson learned, is the high overhead induced by the implementation as VNFs and the chaining with application-level switches. We plan to introduce caching in the architecture for reducing the very important end-to-end delay caused by this high overhead.

ACKNOWLEDGEMENT

This work was supported in part by Ericsson Canada, and the Natural Science and Engineering Council of Canada (NSERC) through the SAVI Research Network. We also would like to thank Aida Rangy and Sandhya Shanmugasundaram who actively participate in the implementation process.

REFERENCES

[1] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep. 2002.

[2] S. W. Brim and B. E. Carpenter, "Middleboxes: Taxonomy and Issues." Available at: https://tools.ietf.org/html/rfc3234 (2016).

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

[4] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," *IEEE Netw.*, vol. 29, no. 3, pp. 36–41, May 2015.

[5] K. Giotis, Y. Kryftis, and V. Maglaris, "Policy-based orchestration of NFV services in Software-Defined Networks," in *1st IEEE Conference on Network Softwarization (NetSoft)*, Bologna, Italy, pp. 1–5, April 2015.

[6] "NFV, GS. '001: Network Functions Virtualisation (NFV); Use Cases, V 1.1. 1.' ETSI." Dec. 2013.

[7] P. Veitch, M. J. McGrath, and V. Bayon, "An instrumentation and analytics framework for optimal and robust NFV deployment," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 126–133, Feb. 2015.

[8] N. Bouten, J. Famaey, R. Mijumbi, B. Naudts, J. Serrat, S. Latré, and F. D. Turck, "Towards NFV-based multimedia delivery," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, Canada, pp. 738–741, May 2015.

[9] H. Liu, Y. Hu, G. Shou, and Z. Guo, "Software Defined Networking for HTTP video quality optimization," in *15th IEEE International Conference on Communication Technology (ICCT)*, Guilin, China, pp. 413–417, Nov. 2013.

[10] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An SDN-Based CDN/ISP Collaboration Architecture for Managing High-Volume Flows," *IEEE Trans. Netw. Serv. Manag.*, vol. 12, no. 1, pp. 48–60, Mar. 2015.

[11] C. Mouradian, T. Saha, J. Sahoo, R. Glitho, M. Morrow, and P. Polakos, "NFV based gateways for virtualized wireless sensor networks: A case study," in *IEEE International Conference on Communication Workshop (ICCW)*, London, UK, pp. 1883–1888, Jun. 2015.

[12] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Dynamic chaining of Virtual Network Functions in cloud-based edge networks," in *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, Bologna, Italy, pp. 1–5, Apr. 2015.

[13] W. Ding, W. Qi, J. Wang, and B. Chen, "OpenSCaaS: an open service chain as a service platform toward the integration of SDN and NFV," *IEEE Netw.*, vol. 29, no. 3, pp. 30–35, May 2015.

[14] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "ClickOS and the Art of Network Function Virtualization," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, pp. 459–473, Aug. 2014.

[15] S. Paul and R. Jain, "OpenADN: Mobile apps on global clouds using OpenFlow and Software Defined Networking," in *IEEE Globecom Workshops*, Anaheim, CA, USA, pp. 719–723, Dec. 2012.