Multimedia Service Management with Virtualized Cache Migration

Reza Shokri Kalan
International Computer Institute
Ege University
Izmir, Turkey
reza.shokrikalan@digiturk.com.tr

Muge Sayit
International Computer Institute
Ege University
Izmir, Turkey
muge.sayit@ege.edu.tr

Stuart Clayman
Dept of Electronic Engineering
University College London
London, UK
s.clayman@ucl.ac.uk

***Abstract*—The demand for playing the video with the highest quality possible leads video streaming companies to utilize network-based solutions and server assistance. As being one of the popular technology in video streaming area, HTTP adaptive video streaming systems can benefit from the emerging network technologies such as SDN and NFV. Such multimedia systems require the management of network resources efficiently to be able to provide acceptable service. In this paper, we propose a multimedia service management architecture that focuses on migration one of the most crucial components of such systems, virtualized caches. SDN controller gathers underlay network statistics and based on real-time network condition determines the optimal location for virtual cache and/or migrate it when network conditions change. We focus on hand-off affects the perceptual quality when clients migrate from one cache to another. The proposed approach was evaluated using the Mininet Emulator and its performance was given comparatively. The performance results show that the combination of SAND (Server and Network Assisted DASH) features with the advantage of the NFV-SDN technology can provide good service to its users.**

***Index Terms*—Adaptive streaming, NFV-SDN, DASH, SAND, vCache.**

I. Introduction

HTTP adaptive video streaming systems became a popular video streaming application due to their design characteristics that give them the flexibility to cope with dynamic network conditions by taking into account their internal parameters. The principle behind adaptive streaming is creating multiple copies of the same content encoded with a different bit rate which is called $representations$. Dividing each representation into small size chunks (or segments) enables clients to switch among representations and to adapt to the suitable video quality based on estimated network bandwidth or buffer fullness over time. MPEG group standardized DASH (*Dynamic Adaptive Streaming over HTTP*) to provide interoperability between these types of applications.

Rely on the fact that real-time information being extracted from the global view of the network [1], [2] could provide better adaption on the user level, network-assisted approaches have been proposed and are being standardized for such systems. Regarding this aspect, the MPEG group has proposed SAND (*Server and Network Assisted DASH*) standard, which is a new framework for DASH. Besides DANE (*DASH Aware Network Elements*), SAND also introduces a new type of messages which are exchanged between DANEs and RNEs (*Regular Network Elements)* for providing auxiliary information to the clients, or directing a client to act in a specific way which is defined by the service [3]. Caches used in video streaming systems can be employed as DANEs in case they use DASH related information when serving DASH clients.

The location of the cache plays an important role in the performance of video streaming applications. In our previous study [4], we implemented a virtual cache by combining the concepts of DANE and NFV *(Network Function Virtualization)*, which we call $vDANE$ or $vCache$. The SDN controller creates vDANE instance complimentary with NFV and executes functions for selection optimal location for the initial and migrated vDANEs. In this current study, we address latency problems during the "hand-off" operation, which is defined as the action of disconnecting a client from one vCache and connecting to another, by proposing a complementary approach to our previous works.

Although the clients adapt changing conditions by utilizing remarkable rate adaptation techniques, the QoE (*Quality of Experience*) might be dramatically affected due to the severe changes in network traffic and diminished end-to-end network capacity. Therefore, adapting the settings of underlying network elements and functions would be helpful to DASH clients to minimize the effects caused by changing network conditions on QoE. Changing network conditions or the distribution of the traffic sources/destinations on the network requires to migrate virtual functions serving network applications. When it comes to a video streaming service receiving assistance from virtual functions, the migration of these functions is a crucial issue due to a potentially high impact on QoE. In this work, we focus on the problem of migration of vCaches. On one hand, migrating a vCache function is a good response given to the changing network conditions. But on the other hand, this migration causes clients to do hand-off operations. This might be an important problem if the buffer of the clients is consumed because the buffer does not continue to be filled up during this process, which, in turn, might cause severe video stalls. In [5], the duration of pauses is defined as the QoE parameter one of the most negative effects on perceptual quality. So, this introduces us to a trade-off, while vCache migration is considered as a move for increasing QoE, it might cause an unexpected decrease of QoE if migration is not managed carefully, by considering the video streaming application characteristics.

In this work, we propose an architecture who manages vCache migration by considering the network topology, current network traffic and buffer level values of DASH clients. The clients connected to a network managed by an SDN controller, where we consider the case video streaming company and ISPs collaborate to serve DASH clients. The contributions of this work are twofold: (i) We present a lightweight algorithm that is run by the controller, determining when to migrate a vCache function and when the clients disconnect the old vCache. The algorithm aims to minimize the effects of $Hand-off$ operation which might cause the buffer to be emptied on the client's side and consequently increases video stalls. (ii) vCache migration is done by utilizing SAND characteristics as well as SDN and NFV concepts. The proposed architecture uses SAND messages and elements, hence fully compatible with the SAND standard.

The rest of the paper is organized as follows. Background and related works that give information about the main characteristics of SAND and SDN based SAND systems are in Section II. The proposed cache migration algorithm is presented in Section III. In Section IV, performance evaluations are presented comparatively and in Section V we give the conclusions as well as the future works.

II. Background and Related Works

A. Adaptive Video Streaming over HTTP with Network and Server Assistance

The performance of video streaming applications is highly bounded to the changing network conditions or traffic patterns. HTTP adaptive streaming comes to address this network dynamism, whereby the clients can request a video at an appropriate quality level according to the changing internal parameters such as obtained throughput or buffer fullness level. Adaptive streaming technologies provides a performance management technique that offers huge potential for OTT (*Over-the-top*) multimedia streaming. Thus, HTTP streaming has become a popular solution in commercial deployments [4]. The MPEG group introduced standardized DASH adaptive video applications which is later supported by SAND technology. In the concept of SAND standard, four classes of network elements are defined:

- DASH client: DASH client with SAND features, i.e. which can interpret SAND comments and act accordingly.
- DASH metric server: responsible for gathering DASH metrics from clients
- DANEs: DANEs are network elements with minimum intelligence about DASH, for instance DASH server
- RNEs: regular network elements are presented in DASH infrastructure, but they have no information about DASH format and characteristics.

Table I shows the messages passing between network elements defined by SAND architecture. PER *(Parameters for Enhancing Reception)* messages are used for providing information about the network to the clients. PED *(Parameters for Enhancing Delivery)* messages are exchanged between DANEs to enhance delivery. Status and metrics messages are sent by the clients to DANEs, which carries information about current QoE or internal parameters of clients. Buffer fullness or recently requested video quality can be given as examples to such client parameters [6]. On the other hand, DANEs help DASH clients to achieve better adaption by sending information about cached segments, alternative segment availability, possible delivery time, network throughput and QoS via PER messages [4], [6]. In conventional web and CDN caching mechanisms, content is proactively pushed to edge servers rely on prior knowledge of content demand and network structure which means that cache location is static. Cache content popularity has been considered in some studies such as PopCache [7] and BEACON [8], but none of these studies focused on the distributions of connection of clients. Selection of important nodes for reducing the number of hops [9] to reach the content is a crucial factor for improving system performance. In the edge cache methods [10], [11], clients request content from a connected edge, and edge as the nearest content resources respond if the requested content is available. Based on the prediction algorithm, the edge node decides whether and where to cache the requested content. Again, proposed algorithms mostly focus on the prediction of the user request and $pre-fetching$ the content rather than caching in optimal locations or migration of caches.

Routing decisions in central control can accelerate service delivery and provides more agility to forwarding devices. However, by pushing content to all edges, unnecessarily same content replicated in all edge routers/switches which affect forwarding device performance and consequently reduces overall network performance. However, in a large network, a centralized cache approach may not perform well. As a simple solution, a large network can be divided into multi-subnetworks and a cache is placed in each domain. This helps to reduce the cache server's load and end-to-end delay, but dividing the network into multiple domains as well as the cache placement algorithm need to be considered. To address those challenges, authors in [12] proposed multiple cache servers based on closeness centrality, $betweenness$ centrality, and path-stretch values. However, the proposed model has static nature which means that the cache location is fixed even the network's traffic pattern has changed. Also, the performance on the client's side is negatively effected while cache servers are placed behind bottleneck links.

In terms of network resource utilization, optimal content placement mostly depends on the geographical distribution of

TABLE I: SAND message types

Message Types	Description
PER messages	sent from DANEs to the clients
PED messages	exchanged between DANEs
Status messages	sent from clients to DANEs

requests, request intensity, and content popularity. The content placement which discussed in [13] improves bandwidth usage by proactively caching content inside the ISPs network. However, proactive caching does not satisfy the geographical distribution and dynamic nature of the Video-on-Demand (VoD) requests. Furthermore, repeatedly content migration results in more bandwidth usage. Authors in [14] proposed Integer Linear Program (ILP) model to address frequently migration multi-tenant content and server selection. However, both proposed models relied on VoD and long term prediction (e.g., a week) which is not suitable in live streaming.

B. Utilizing NFV and SDN in Caching for Adaptive Video Streaming over HTTP

Conventional network architecture which is relied on the vertical design where data and control planes are bundled together makes network modification hard and complex task. SDN architecture with separating control and data plans overcomes this complexity and give more agility to network functions. SDN is a new form of networking technology that transforms hardware and device-centric network architecture into a flexible, and programmable network in order to improve agility and provide fast innovation. The SDN architecture conceptually divided into three layers; [15] Application, control, and infrastructure. The control layer monitors and manages the behavior of the network infrastructure and application through the API. The control layer continuously communicates with network infrastructures to monitor and collect network status. Leveraging SDN allows centrally observing the current network traffic and developing a flexible routing algorithm specific to the requirements of applications running on top of the network. Network operators develop and implement an application-specific routing strategy by putting the network intelligence to the control plane. In the context of separate data and control planes, well-defined API (*Application Programming Interface* can be leveraged between the SDN controller and underlay switches.

One of the main concepts of the SDN architecture relies essentially on modifying forwarding devices to support flow tables. Compared with legacy networks in which infrastructure devices have autonomous decision making, equipment in the SDN just act as simple forwarding devices. In other words, the absence of control in network equipment is the main difference between SDN and legacy networking. This new architecture is built on virtualization and standard interface such as OpenFlow. The control plane as a network controller fills tables of the OpenFlow enabled data plane devices. Data plane devices are responsible for packets forwarding based on matching rules and periodically sending link's statistic to the controller. Flow tables -Inside forwarding devices- define how the packet should be handled. If there is no defined rule for arriving packet, depend on configuration, the packet will be discarded or switch to send it to the controller.

Although the concept of SDN and NFV are independent, they are in practice highly correlated. The core similarity between them is that they both use network abstraction and depend heavily on virtualization. SDN and NFV are performing different roles, but leveraging NFV-SDN specifications enables networking architectures more flexible and results in network infrastructure agility. While NFV enhances the development of the network functions by separating physical and virtual devices, SDN offers central control on forwarding devices. Compared with traditional web caching and CDN caching, a virtual cache is ubiquitous, transparent to applications, and cache locations are dynamic. The main advantage of physical forwarding devices and the logically central controller has easier network monitoring and management. As a consequence, provides a mechanism to place caches in different suitable locations which leads to better performance.

In recent years, NFV and SDN techniques are utilized for caching video streaming. Authors in [16] provided a hierarchical network architecture consists of a three-layer structure (core, aggregation, and access layers) for multimedia delivery on NFV. The efficient placement of VNFs is discussed in [17], where authors introduced hybrid solutions inside providers network with a combination of the between the legacy network and NFV named NFV-P. Although these proposals present a good abstraction of the deployment of NFV in an operator's network, QoE parameters were not considered in both works. Proposed migration algorithm in OMAC [18] provides the optimal placement for virtual content delivery functions, but it only addressed minimizing the migration cost and not focused on content retrieval delay or the response time.

Authors in [19] considered different layers of vCDN request: $i)$ POP or core data center, $ii)$ home network, and $iii)$ the end-user devices. Although the focus of this work is on vCDN, they do not take into consideration video streaming and QoE parameters. Despite leveraging SDN, proposed algorithms mostly rely on VMs configuration and resource allocation rather than considered dynamism and distribution of the users' requests.

In [20] we introduced vDANE concept as a NFV function and proposed a solution for deployment additional virtual caches in term of vDANE. In [4], we leveraged SAND characteristic and proposed a solution to the problem of selecting the location of the initial vDANE. The proposed algorithm continuously monitoring user distribution as well as network conditions and migrates vDANE in other to provide seamless quality service to end users. Unfortunately, in the proposed solution, clients may suffer re-buffering during hand-off time. In this study we aim to minimize the negative effects of $hand-off$ latency when clients migrate from one vDANE to another.

III. Deploying vDANEs in SAND based Architecture

A. General Characteristics of the System Architecture and Locating the Initial Cache

In this work, we consider a network which can be denoted as graph $G(N, L)$, where N represents the nodes and L represents the bidirectional links. DASH clients are distributed over the topology, and they can connect to the network via any node

$n \in N$. The networks are managed by an SDN controller having modules that execute SAND functionalities. The controller tracks online DASH clients, directs them to achieve higher QoE by sending specific SAND messages (PER messages) and initiates and migrates vDANE instances. Suppose there is no vDANE instance in the network or the existing vDANE instance is decided to be migrated by the controller in a particular moment. In both cases, the SDN controller selects an optimal place for vDANE instance and this selection is done by considering a value which we call $PressureInitialCache$ (PIC) and introduce in our previous work [4]. This value basically shows the pressure of vDANE instances by taking into account the network topology, distribution of the clients and current traffic.

The formula for calculating of PIC value for the node n is given in 1. The formula sets a pressure value for each possible candidate location for initializing a vDANE instance when there is H number of DASH clients in the system. In the formula, b_l represents the bandwidth of the link $l \in L$. h_n and L_n refers to the total number of links and number of clients directly connected to the candidate node n, respectively. $D_n = \sum_{k=1}^{|H|} d_{kn}$ shows the total distance between all clients and node n, where d_{kn} is the number of hop counts between node k and node n. Node k represents the node which a client is connected to.

$$PressureInitialCache(n) = \frac{\sum_{l=1}^{|L|} b_l * (D_n - h_n)}{\sum_{l=1}^{L_n} b_l * h_n * L_n} \quad (1)$$

The PIC value is small when the ratio of $\sum_{l=1}^{|L|} b_l / \sum_{l=1}^{L_j} b_l$ is small, the number of directly connected clients and the number of outgoing links is high. The location for the vDANE instance is selected as the candidate node whose has the minimum PIC value. Therefore, a candidate node having more bandwidth, more outgoing links as well as in a central position considering the distribution of DASH clients in the topology is selected as the location for initializing the vDANE.

If the number of the connected client to the network increases and the average quality in terms of bitrate received by the clients is under the *quality threshold*, the controller then decides to deploy additional vDANE. Installing additional vDANE has been discussed in our previous work [20], we mainly focus on vCache migration affects on QoE.

B. TCP Connection Re-establishment and Hand-off Latency

The clients connected to the system might leave and new clients can connect from another location which is different from the previous ones. When the client distribution has changed in a way that the current position of vDANE is not optimal anymore, it is necessary to change its location to prevent deterioration in perceptual quality. TCP hand-off is a set of actions that happens when a connection is moved from one connection point to another. Transport redirection, load balancing, traffic path optimization, and roaming are the reasons for the hand-off. Especially, hand-off appears in the area of CDN (*Content Delivery Networks*, where CDN architectures leverage a redirect mechanism to initiate a connection between a client and the most appropriate server [21].

In the case of cache migration in our architecture, the current TCP connection between a client and vDANE should be terminated and a new connection is established with new vDANE which recommended by the controller via PER enforcement messages. Furthermore, controller updates forwarding information in underlying forwarding devices according to the address of clients and new vDANE. Unfortunately, TCP handshake causes latency and hence clients may experience re-buffering due to the time required to the establishment of a new TCP connection.

Fig. 1 illustrates the connection setup when a new client joins the network. While at the beginning there is no entry for this client in the switch, it forwards the client's request toward controller in the form of PACKET_IN message. In the next step, switch asks about related routing information from the SDN controller. The controller updates the switch's routing table with the related routing information and finally client connects to the server (or vCache) and starts requesting and downloading content. In the case of vCache migration, the SDN controller updates the routing table inside forwarding devices and also informs clients to connect new vCache via PED and PER messages, respectively. As shown, connection and establishing process takes more time, therefore, during hand-off time clients may experience re-buffering.

C. Cache Migration

After starting the initial vDANE instance, the controller continues collecting statistics of the network and monitoring DASH and cross-traffic patterns. For this purpose, we deploy a SAND metrics server as a northbound application running in the SDN controller. The clients periodically send the current buffer fullness level to this server within the controller. Note that, the traffic introduced by status and metrics messages can be negligible when compared to traffic caused by the transmission of the video files. In case this is seen as problematic, the metrics can be sent by adding this information into the request messages and they can be extracted by the controller as proposed in [22].

When the distribution or number of the clients in the system changes or it determines a change in traffic amount, the controller calculates pressure values for each node again. If the location of vDANE is changed according to the new values of pressure values, the controller sends PER enforcement messages to the clients to change their server. In addition, the shortest paths between the clients and the new vDANE location are selected for each client for streaming packets.

In our previous work, it is stated that cache migration is necessary when the new candidate node for deploying new vDANE has better (smaller) pressure value. However, the experiments show that this approach has a drawback. When vDANE is migrated to its new location, a small change in network condition -for example connecting a few new clients close to the first vDANE location - may result to the minimum

Algorithm 1: Cache Migration Algorithm.

```
Input:
  δ: threshold value for migration
  $P$: PIC value of the current vDANE
  $P'$: PIC value of the new vDANE
1 if (1-(P'/P) >= δ) then
2     run timer
3     while (∃ client connected to the current vDANE) do
4         c ← next client
5         if (c.buffer_fullness >= thr) or timer>= thr then
6             -update routing devices forwarding tables
7             -enforce client c to request seg. from new vDANE
```

pressure value at the previous location and consequently, SDN controller triggers the migration process again to migrate vDANE to its previous location. This process continues unless remarkable changes reshape the traffic pattern completely. We call this phenomenon as $Ping - Pong$ effect.

The cache migration algorithm run by the SDN controller is given in *Algorithm 1*. In order to avoid $Ping-Pong$ effect and minimize the number of migrations by eliminating the unnecessary ones, we use a difference metric as to be utilized in migration decision process executed by SDN controller. Difference metric is calculated as $1 - (P'/P)$, where P and P' refer to PIC values of old and new locations of vDANE, respectively. And it should be above a certain threshold value, δ, for migration process to start as given in the first line of the algorithm.

Each client has buffer whose size of b seconds, which can store maximum b/t segments, where t is the size of a segment in terms of seconds. In order to fast start up and also avoid re-buffering, it is assumed that the client requests the first 4 segments ($4 * t$ seconds length) with a minimum bit rate representation. In line 4, algorithm checks buffer fullness and timer. If the client has enough segment in its buffer, then SDN controller enforces the client to start the next request form new vCache. A client with a more buffered segment has less probability to enter buffering stage during disconnecting from current vCache and reconnecting to new vCache.

The proposed algorithm forces the clients whose buffer contains more data to connect to the new vDANE first. This approach brings two advantages. First, while the clients have more buffered data, it takes time to buffer to get empty and experience re-buffering except clients have less buffered data. Second, by moving the clients to new vCache in two phases, i.e. migrating one client group after another, the clients which are still connected to old vCache may utilize more prosperous bandwidth and download some segments faster before starting hand-off operation.

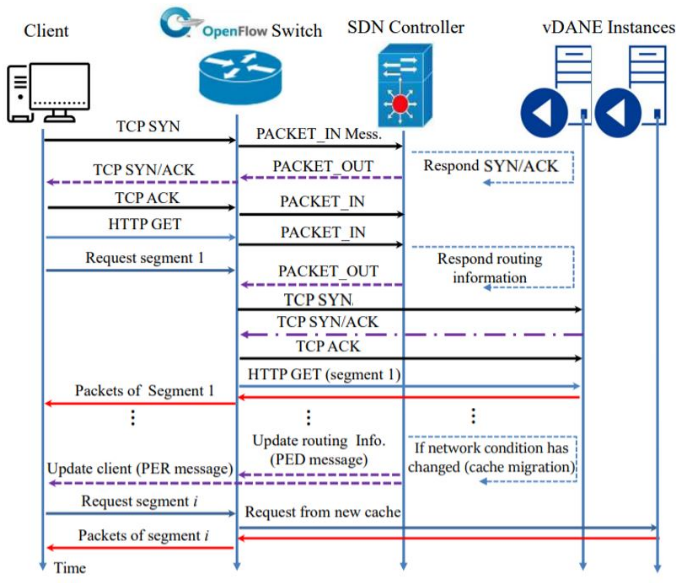

Fig. 1: TCP establishment and hand-off latency timeline

IV. Simulation Setup and Experiment Results

A. Simulation Setup

As shown in Table II, *BellCanada* topology with three different bandwidths settings represented as λ=7, λ=10, λ=15 Mbps, where λ is the mean value of the Poisson distribution, were used to represent the impact of the proposed algorithm. For streaming, we used *Big Buck Bunny* [23] video with six different representations whose bitrate distribution is given in Table. III. We deployed $FloodLight$ controller for network monitoring and controlling and used $Mininet$ for simulation with the topology consisting of 20 DASH clients and a server. Clients run throughput base adaption algorithm, where the requested quality is determined by considering the throughput, and achieve better QoE while requesting video from the highest bitrate ($R6$). In this study, we consider cache migration when new vDANE in the P' location has at least 10%(δ =10% improvement in PIC value compared with the previous one (P). We achieved this value based on test results, but it should be noted that it is not constant and depends on network size and the number of connected clients. Finally, we repeated the simulations for 10 times for each algorithm and provided averaged outcomes in the following tables and the figure.

In the context of performance measurement, the proposed architecture is compared with the *Best effort*, and the $HotSpot$ approach described in [24]. Typically, in the *Best effort* algorithm, a vCache is placed in a location that approximately has the same distance -in terms of hop counts- from all clients. The other comparison approach is $HotSpot$ algorithm, which relies on the locality concept by introducing $F(i) = a^3+b^2+c$ as a pressure function. In this function, a, b and c, indicates the number of clients with one, two, and three hops away from node i, respectively. This function triggers $HotSpot$ algorithm deploys vCache in a place that has the highest client density.

It is worth emphasizing that, in order to observe the effect of the proposed architecture we consider tight bandwidth

TABLE II: Configuration of simulated network topology

Topology	#Nodes	#Links	Average bandwidth	#Clients
BellCanada	43	58	λ=7, λ=10, λ=15 Mb	20

TABLE III: Big Buck Bunny representations

Representations	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
Bit rates	2133	2484	3078	3526	3840	4219

values which result in moderate video quality and more delay. Furthermore, when a larger network is used and the number of clients on the network increases, the outage duration value also increases in all approaches. The main reason for that is due to the increase in distance between the cache and the clients, which causes additional delays in the communication between these entities.

B. Experimental Results

The quality metric values obtained from the simulations are given in Table IV. In the table, *vDANE migration* represents the developed migration approach for this work and the results obtained with the proposed architecture. It is evident from Table IVa that the minimum video quality in terms of bit rate obtained by *Best effort* approach. It is because this approach only considers the locality without considering the number and distribution of the clients. Clients start requesting video from the lowest bitrate when they are placed behind a bottleneck link. static nature of the *Best effort* approach may result in a bottleneck in specific parts of the network that have high client density. The values given in Table. IVb indicates that clients in the *Best effort* approach experience re-buffering duration less than $HotSpot$ approach. It is because requesting video with lower bitrate which has a small size reduces download time and avoids more re-buffering.

We observe that the proposed architecture with the introduced vDANE migration algorithm outperforms the other two approaches for all bandwidth settings used in the simulations. In the average received video quality, when network's bandwidth is more tight (λ=7 Mb) *vDANE migration* algorithm slightly behind the $HotSpot$ algorithm. But, at the same time when considering re-buffering time, we see that $HotSpot$ algorithm has double size re-buffering time, which is unacceptable in terms of QoE. The reason for that is, clients need to TCP re-establishment process after a vDANE is migrated, and at the hand-off time the clients stop receiving data. However, the proposed *vDANE migration* algorithm eliminates the negative effects of TCP re-establishment process as well as avoiding the $Ping-Pong$ effect.

Fig. 2 shows averaged the received quality as a function of segment numbers for the whole run, where λ equals to 15 Mbps. The duration of each segment equals to 2 seconds. When we follow the time line, clients with the *vDANE migration* approach always experience higher video quality. This shows that, managing $hand-off$ operation by considering the buffer level and distribution of the clients has a positive impact on the received quality. It is also important to see that,

TABLE IV: Quality metrics values obtained from simulation

(a) *Average received video quality (Kbps)*

Algorithm	λ=7	λ=10	λ=15
Best effort	2304	2451	2883
HotSpot	2650	2542	2917
vDANE migration	2635	2647	3133

(b) *Average re-buffering duration (sec)*

Algorithm	λ=7	λ=10	λ=15
Best effort	386	250	147
HotSpot	662	672	417
vDANE migration	323	219	73

the migration of the vDANE instance didn't cause any negative affect on the received video quality during the whole simulation thanks to the utilization of SAND messages and metric server within the design of the architecture. The decrease in the bitrate values can also be seen in other approaches too, which means these are mainly due to the congestion on the network, not because of negative affects of the migration. Considering re-buffering time in both other approaches indicate that the proposed architecture with *vDANE migration* algorithm has provide a good balance between bitrate and re-buffering time. We also run similar tests with the architecture proposed in our previous work [4]. However, due to $Ping-Pong$ effect, the clients experience severe video stalls as well as lower received video quality compared to the current architecture. The difference between the re-buffering times of current and previous proposals is up to 20 secs on average for each time hand-off operation occurs.

V. Conclusion

In this study, we proposed a SAND architecture using an algorithm for reducing the cost of cache migration in terms of re-buffering time on the client's side by taking the advantages of the NFV-SDN into account. To achieve this, we leveraged SAND architecture and deployed a vDANE element as a cache by using network function virtualization as well as we utilized the metric server deployed as a module of the SDN controller. The proposed cache migration is developed by considering both the client's buffer fullness and updating forwarding devices routing information.

In order to show the performance improvement provided by the proposed algorithm, we implemented simulations in Mininet and compared obtained results with two cache placement algorithms proposed in the literature. The simulation results show that our proposed algorithm outperforms both comparison algorithms on average received bitrate and re-buffering duration and there has not been any observation of deterioration in received video quality due to vCache migration.

The evaluation of the proposed architecture is based on two QoE parameters; throughput or received bitrate, and buffer fullness or re-buffering duration. We observed that our architecture with *vDANE migration* provided improvement in throughput when compared to *Best effort* and $HotSpot$

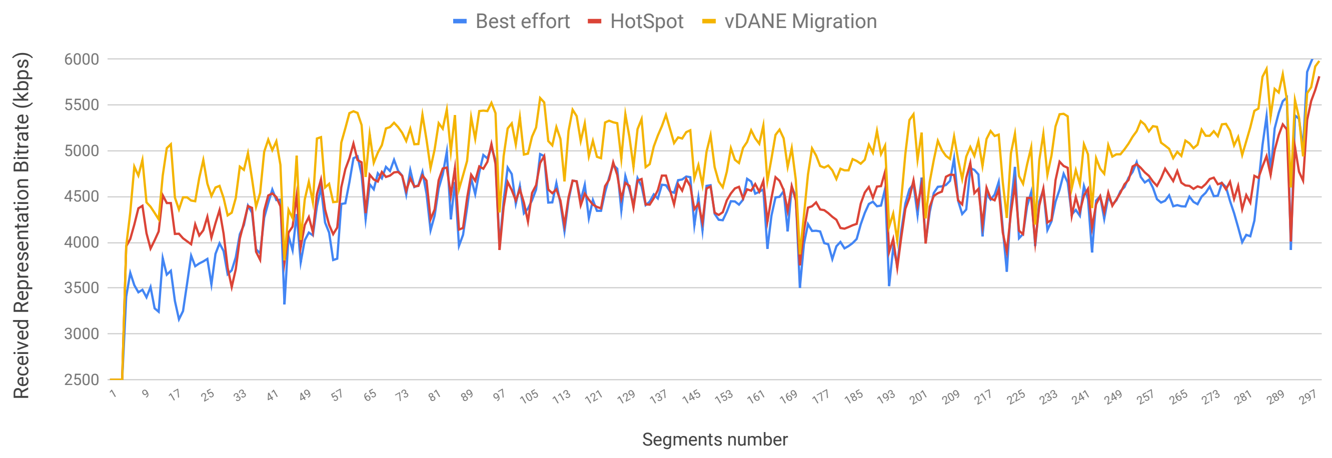

Fig. 2: Long term received video representation per segment (λ=15 Mb)

approaches, up to 14% and 7%, respectively. Furthermore, the proposed algorithm has better performance in terms of quality metrics when compared with our previous study [4]. Considering buffer fullness indicates that our algorithm also provided a decrease in re-buffering duration 50% and 82% when compared to *Best effort*, $HotSpot$, respectively, which shows the good impact on QoE of our approach.

Relying on simulation results, we can conduct that dynamically deploying vCache instance in a suitable location and forwarding clients' requests toward appropriate instance improves both throughput and re-buffering duration. In general, one vCache is not enough for supporting more clients especially when the number of clients is high. As the future work plan, we consider installing additional vCaches to the network and forwarding each client requests to the suitable vCaches by jointly taking into account the localization of the vCaches and their contents to achieve better streaming experience.

Acknowledgment

This study is supported by Digiturk beIN MEDIA GROUP, TUBITAK EEEAG under grant 115E449, and has been elaborated as part of the Lucerne Summer University: *Ethics in a Global Context*.

References

[1] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for cdn-based live video delivery," in *SIGCOMM '15*, 2015, pp. 311–324.
[2] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," in *SIGCOMM '12*, ser. SIGCOMM '12, 2012, pp. 359–370.
[3] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing mpeg dash performance via server and network assistance," *SMPTE Motion Imaging Journal*, Jan 2017.
[4] R. S. Kalan, M. Sayit, and S. Clayman, "Optimal Cache Placement and Migration for Improving the Performance of Virtualized SAND," in *NETSOFT2019*, 2019.
[5] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys Tutorials*, Firstquarter 2015.
[6] MPEG's, "DASH-IF Position Paper: Server and Network Assisted DASH (SAND), ISO/IEC 23009-5, published December 2016."
[7] K. S. et al, "Popcache: Cache more or less based on content popularity for information-centric networking," in *38th Annual IEEE Conference on Local Computer Networks*, 2013, pp. 236–243.
[8] Z. Xiaoqiang, Z. Min, and W. Muqing, "An in-network caching scheme based on betweenness and content popularity prediction in content-centric networking," in *(PIMRC)*, Sep. 2016, pp. 1–6.
[9] S. He, H. Tian, and X. Lyu, "Edge popularity prediction based on social-driven propagation dynamics," *IEEE Communications Letters*, May 2017.
[10] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, "Edge caching with mobility prediction in virtualized lte mobile networks," *Future Generation Computer Systems*, vol. 70, pp. 148–162, 2017.
[11] Y. Tang, K. Guo, J. Ma, Y. Shen, and T. Chi, "A smart caching mechanism for mobile multimedia in information centric networking with edge computing," *Future Generation Comp. Syst.*, 2019.
[12] K. Alhazmi, A. Moubayed, and A. Shami, "Distributed sdn controller placement using betweenness centrality & hierarchical clustering," in *DIVANet'18*. ACM, 2018, pp. 15–20.
[13] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. D. Turck, "Proactive multi-tenant cache management for virtualized isp networks," *10th CNSM and Workshop*, pp. 82–90, 2014.
[14] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latré, G. Pavlou, and F. De Turck, "Hybrid multi-tenant cache management for virtualized isp networks," *Journal of Network and Computer Applications*, 2016.
[15] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, Jan 2015.
[16] R. M. B. N. N. Bouten, J. Famaey, S. L. J. Serrat, and F. Turck, "Towards NFV-based multimedia delivery," in *IFIP/IEEE 2015*, pp. 738–741.
[17] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *CNSM 2014*, Nov, pp. 418–423.
[18] H. Ibn-Khedher, E. Abd-Elrahman, and H. Afifi, "Omac: Optimal migration algorithm for virtual cdn," in *23rd ICT*, May 2016.
[19] H. Ibn-Khedher and E. Abd-Elrahman, "Cdnaas framework: Topsis as multi-criteria decision making for vcdn migration," *Procedia Computer Science*, vol. 110, 2017.
[20] S. Clayman, R. S. Kalan, and M. Sayit, "Virtualized Cache Placement in an SDN/NFV Assisted SAND Architecture," in *BlackSea2018*, 2018.
[21] A. Binder, T. Boros, and I. Kotuliak, "A SDN Based Method of TCP Connection Handover," in *3rd International Conference on Information and Communication Technology-EurAsia (ICT-EURASIA), 2015*.
[22] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "Sdnhas: An sdn-enabled architecture to optimize qoe in http adaptive streaming," *IEEE Transactions on Multimedia*, pp. 2136–2151, Oct 2017.
[23] "Big Buck Bunny." [Online]. Available: http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny/2sec/
[24] L. Mamatas, S. Clayman, M. Charalambides, A. Galis, and G. Pavlou, "Towards an information management overlay for emerging networks," in *NOMS 2010*, April 2010, pp. 527–534.