# Virtualized Cache Placement in an SDN/NFV Assisted SAND Architecture

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

Reza Shokri Kalan
International Computer Institute
Ege University
Izmir, Turkey
Email: reza.shokri@hotmail.com

Müge Sayıt
International Computer Institute
Ege University
Izmir, Turkey
Email: muge.fesci@ege.edu.tr

***Abstract*—Web-caches play an important role in the architecture of a Dynamic Adaptive Streaming over HTTP (DASH) system, which can bring video files near to the clients. In such systems, the placement of the caches affects the performance of the video streaming applications. In this paper we select the optimal network nodes for cache placement by taking into account characteristics such as the bandwidth of the paths, the places, and the number of the online clients, for a given network graph. For this purpose, we use a placement algorithm which is enhanced version of the Pressure algorithm proposed by Clegg which we call PressureCache, in an implementation of virtualized DASH Aware Network Elements (vDANEs). The results show that the proposed approach gives better performance in terms of received quality than random cache placement.**

## I. Introduction

Adaptive video streaming over HTTP has become quite popular among video streaming systems recently, since HTTP provides mechanisms to utilize web caches and HTTP traffic can pass easily through firewalls. In HTTP Adaptive Streaming (HAS) systems, more than one *representation* of the media, each with different qualities, are encoded and kept on the server, thereby providing clients with an ability to adapt video quality by requesting the different representations over time. Dynamic Adaptive Streaming over HTTP (DASH) has been standardized by MPEG group, and it defines common formats for storing files with information about representations plus parser functions on the client side.

In DASH systems, quality adaptation is done by a client rate adaptation algorithm. The inputs to the algorithm are the network parameters observed by the clients, such as average throughput, delay, and jitter. If the clients can be assisted by network elements, the Quality of Experience (QoE) achieved by the clients can be further enhanced. For this purpose, the MPEG group started working on the standardization of Server and Network Assisted DASH (SAND) [1]. In the SAND architecture, some network elements have the information about DASH characteristics and are called DASH Aware Network Elements (DANEs). These DANEs help to increase the QoE by providing network related information to the clients.

Software Defined Networking (SDN) was introduced as a network architecture proposing to decouple the data plane from the control plane of computer networks [2]. This approach provides mechanisms to design and implement new routing algorithms, as well as providing information about the network topology and network conditions to applications. As a complementary technology to SDN, Network Function Virtualization (NFV) virtualizes the network functions of the conventional physical hardware [3], providing a wide range of implementations that can be instantiated on-demand at runtime and can be used to improve both network management capabilities and the performance of Internet applications.

SDN and NFV technologies can be utilized together by using them as components of a video streaming system architecture. Today, the biggest video streaming companies deploy Content Delivery Networks (CDN) servers in order to bring video content closer to their customers. The virtualization of CDN servers can be performed by defining CDN functions as VNFs that run on top of the physical elements within ISPs [4]. Based on the co-operation between ISPs and Over the Top (OTT) video service providers, vCDN instances can be created in several points in the network in order to create a streaming service infrastructure that can deliver video packets to the clients efficiently [5].

Caches in the form of HTTP/web-caches play an important role in the architecture of a DASH system, and they can bring video files nearer to the clients than the original source. In such systems, the placement of the caches affects the performance of the video streaming applications.

In this paper, we propose a SAND video streaming architecture utilizing both SDN and NFV concepts, whereby virtual cache instances are created on-demand by considering the live DASH attributes and characteristics. These virtual caches are defined as DANEs, and using such an approach gives the right level of flexibility and control. vDANEs are the virtualized caches with the knowledge of DASH characteristics, which uses NFV hosted by the servers connected to the switches. We select close to optimal network nodes for virtual cache placement by taking into account characteristics such as the bandwidth of the paths, the places, and the number of the online clients, for a given network graph.

The rest of the paper is organized as follows. In section 2, we give the related works with the information of SAND. The details of the proposed architecture and vDANE placement algorithm are given in section 3. Performance evaluations are presented in section 4 and we conclude in section 5.

## II. Related Work

The co-operation of CDN companies and ISP network providers can be beneficial for DASH applications [6]. The SAND architecture introduces a message exchange between DASH clients and network aware elements. Parameters Enhancing Delivery (PED) and Parameters Enhancing Reception (PER) exchanges between DANEs and DANE to DASH client respectively [1]. Also, SAND enables clients to send status message to the DANEs. Content providers can leverage SAND in order to conduct the DASH client to specific CDN servers.

The approach presented in [7], combines the SAND standard with CDN management features for DASH video streaming by implementing content aware-caching and by deploying content and service-aware networks elements. In the proposed model, a Monitoring and Management Server(MMS) receives metric signals from clients and communicates with CDN servers via PER messages in order to inquire status of the servers and network conditions. Nevertheless, CDN server placement and SDN/NFV related solutions was not studied in this work. Although there are several SAND architecture utilizing SDN advantages proposed in the literature [8], [9], our study differs from these studies since virtualization concepts and cache placement problems are not previously considered.

Virtualized networking such as NFV/SDN is good for addressing high demand of resources, unpredictable traffic patterns, and agility in network configuration. In [10], the authors introduce OpenCache (or cache as a service) architecture which leverage SDN and OpenFlow to provide a control plane that orchestrates caching and steers the content to the clients as close as possible. The objective of OpenCache is more related to what content should be cached, but the proposed method does not consider cache placement, network bandwidth, or traffic patterns. Optimal cache placement based on several constraints such as server load, link capacity, and migration cost is proposed in [5]. The authors do not focus on adaptive video streaming, selection of paths between clients, and the caches and cache selection considering the number of clients.

Leveraging NFV increases network flexibility and reduces cost by having the displacement of network functions over virtual instances deployed on generic servers. However, NFV is not only available in the data center, and NFV can be deployed in core networks where bandwidth is more valuable. Efficient placement of NFV functions is discussed in [11]. In the proposed solution of [12], when the network load rises up the base load is handled by physical hardware while virtual instances deal with the overflow. However, this study does not elaborate on the specific data and traffic in practice. NFV based multimedia delivery is reported in [4]. This study more focus on optimal allocation of the data center and intention to find the trade-off between distributed and centralized topology and less attention to real multimedia traffic.

The impact of CDN and Information-Centric Networks (ICN) for DASH streaming has been investigated by Zhe et al. in [13]. The authors present a network friendly DASH architecture under joint management of both CDNs and ISPs, and propose a communication protocol between virtual and original CDN. However, proposed architecture does not aim at determining the location of Cache server or proxy, but determining a copy of requested content.

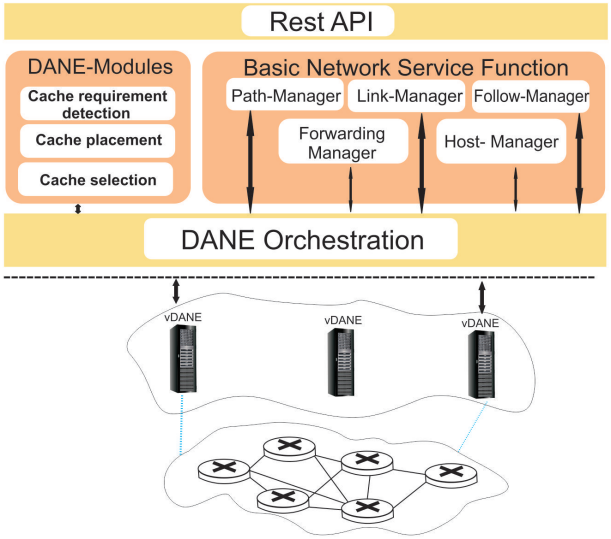


Fig. 1: vDANE Architecture

## III. System Design

The system design for vDANEs is presented here, showing the overall architecture and the cache selection and cache placement algorithm. The controller and the caches are setup as DASH Aware Network Elements (DANEs) in this architecture. We use NFV to create vDANE instances which are added based on the output of the modules running on the controller.

### A. System and Controller Architecture

The architecture is illustrated in Figure **??**, and shows the layering devised. The top layer consists of the *Controller*, where *DANE Orchestration* and control plane functions related to the proposed architecture are managed. This layer provides all of the modules and functions for SDN / NFV – the *Basic Network Service Functions*, and for managing DANEs – the *DANE Modules*. This top layer also has a REST API that can used by higher level Apps and Services. The modules executing the Basic Network Service Functions include the Link Manager that obtains the traffic statistics from switches and a Host Manager which keeps the information about online clients. There are three DANE Modules within the controller, and these modules determine (i) if a new vDANE instance should be added to the network, (ii) where it should be placed, and (iii) to which one of clients it should connect.
The middle layer is represented by an abstraction of vDANEs, made up from all the of the NFVs executing DANEs across the network. The bottom layer is the physical infrastructure, which consists of OpenFlow enabled switches and links.

### B. Cache Selection and Cache Placement

vDANEs are designed to be aware of the DASH system and its application specific parameters, such as the number and

the bitrates of the media representations. vDANEs periodically send the average quality of the received video representation requested by the clients back to the *Controller*. This information is retrieved by the *Cache requirement detection* module running in the controller. The Cache requirement detection module checks if the average quality, received by the clients, is under a specific threshold, called *the quality threshold*. This threshold can be specified by either the network operator or the video streaming system company. For example, it can be selected as the bitrate of the representation having moderate quality. If the average requested quality is under the quality threshold, the controller then decides to create of a new vDANE instance. The placement of the new vDANE is determined by the *Cache placement* module of the controller. When the Cache requirement detection module triggers, it signals the Cache placement module, and the placement process starts.

For the selection of the location of vDANE, we use a placement algorithm which is a modified version of the Pressure algorithm proposed by Clegg et al. [14] and extended in [15], which we call the $PressureCache$ algorithm. This algorithm has never been used for determining cache placement in the literature. In the $PressureCache$ algorithm a score, called the $PressureCache$ score, is calculated for all potential switches that can be selected as hosting a vDANE instance. The value of the $PressureCache$ score is calculated for switch i by using formula (1) and (2), which are modified versions of the score calculation formula given in [14]. $P(i)$ refers to $PressureCache$ score of switch $i$.

$$avgb_j = \frac{\sum_{c=1}^{k} b_j^c}{k} \tag{1}$$

$$P(i) = \frac{\sum_{j=1, i \neq j}^{n} [max(1, avgb_j - b_{ij})]}{n} \tag{2}$$

The score calculation was modified in this study since the formula in [14] only considers the number of hops between the switches and does not consider the bandwidth values of the paths. However, available bandwidth of the paths between the clients and the server is the one of main criteria that directly effects the performance of the video streaming application. In addition, the number of clients connected to a switch, which is also a criteria that effects available bandwidth of the paths between clients and the caches, was not considered in the calculation given in [14]. Also, the shortest path may not lead to better performance because it rapidly changes to bottleneck point while requests increase. In this paper, we consider the available bandwidth of the paths as well as the number of clients connected to the a switch when calculating the score for the switches. In formula (1), $b_j^c$ refers to bandwidth of the path between switch $j$ and the vDANE which the clients transferred packets from, where $c$ and $k$ represent the clients and number of the clients connected to switch $j$, respectively. The formula gives the average bandwidth value for the paths between the clients and their associated vDANEs. Note that, if there is no client connected to a switch, the bandwidth values related to that switch's links are not used in the calculation of the formula. In formula (2), $b_{ij}$ is the bandwidth of the path between the switch $i$ and switch $j$, and $n$ is the number of clients connected to the switch $j$.

**Algorithm 1:** PressureCache Algorithm

**Input:** Set of switches
let *selected* nodes hosting vDANE function;
**foreach** *switch j not in selected* **do**
    ComputePressureCacheScore();
**end**
**Output:** Select the switch with the lowest score

The $PressureCache$ algorithm is given in Algorithm 1. The algorithm selects an almost optimal point for vDANE placement, by considering the bandwidth of the paths between the vDANEs and the clients and also between potential places for a new vDANE instance and the clients. While the Pressure algorithm proposed in Tuncer [15] selects more than one switch to connect local manager nodes to by considering the number of hops between switches, the $PressureCache$ algorithm only selects the most optimal switch for vDANE placement and makes this selection by using the $PressureCache$ score. After the placement of the new vDANE is determined, the controller creates a new vDANE instance connected to the specified switch. Creating a new instance of a vDANE brings about the further problem of selecting the vDANE for each client to connect to. vDANE selection for each client is done by the Cache selection module of the Controller. This module compares the maximum bandwidth of the paths between each client and the vDANE instances and selects the vDANE with maximum available bandwidth.

## IV. Performance Evaluation

### A. Testbed and Topology Setup

In order to evaluate the performance of the $PressureCache$ algorithm with respect to improving QoE, we applied our experiments over three network topologies and used Elephants Dream (ED-II) media dataset [16] for streaming video and benefit from the network-friendly nature of the Scalable Video Coding (SVC) which is an extension of the H.264 standard. The SVC codec provides one base layer and one or more enhancement layers. The base layer has the lowest quality and can be decoded independently, but enhancement layers depend on the base layer and previous layers to improve video quality. Thus, clients need to receive the base layer and all the enhancement layers to achieve topmost quality. As shown in the representations Table I, there is one base layer (L0) and two enhancement layers (L1 and L2). Each of the layers includes 327 video segments with equal length of two second video,

TABLE I: Elephants Dream(ED-II) representations

| Name | Base layer | Enhancement 1 | Enhancement 2 |
|---|---|---|---|
| Synonym | L0 | L1 | L2 |
| Bitrate | 2400 | 3417 | 5167 |

TABLE II: Testbed Data

(a) Network topologies

| Topology | # Nodes | # Links |
|---|---|---|
| Custom | 8 | 11 |
| Compuserve | 11 | 14 |
| BellCanada | 42 | 58 |

(b) Average received bitrate ($kbps$)

| Topology | PressureCache | Random |
|---|---|---|
| Custom | 4051 | 3546 |
| Compuserve | 4289 | 3784 |
| BellCanada | 4718 | 3313 |

(c) Outage duration ($milliseconds$)

| Topology | PressureCache | Random |
|---|---|---|
| Custom | 223 | 498 |
| Compuserve | 420 | 432 |
| BellCanada | 81 | 750 |

giving a total video length of 654 seconds. The base layer, L0, has an encoding bitrate of 2400 $kbps$, L1 has bitrate of 3417 $kbps$, and L2 has bitrate of 5167 $kbps$.

For performance evaluation, we use the following QoE metrics: (i) average received bitrate, (ii) outage duration, and (iii) number of the video segments which received from each quality layer. The three topologies used are two real world topologies from the Internet Topology Zoo [17] and a custom one. The number of nodes and links are presented in Table IIa.

This study aims to find the optimal location for the next incoming vDANE while number of the clients increases and lead to decrease received bitrate in client side. As discussed in Section 3, controller aligns with SAND estimate better location for vDANE placement. We also choose random feasible places-with maximal and minimum distance from the current vDANE- which have adequate bandwidth for vDANE placement in order to compare with optimal placement and we refer this approach as random approach. Mininet emulator, consists of OpenFlow enabled switches is adapted for simulation. DASH clients joins to the network based on Poisson distributions. The average interval time and number of the clients is set to 15 and 10, respectively.

### B. Experimental Results

In order to investigate the performance of the previously described setup, we conduct a simulation which focus on the QoE parameters during the streaming. Our experiment includes close to optimal virtual server that is selected based on the $PressureCache$ algorithm. We compare those results with an experiment using randomly placed servers which are randomly distributed in overlay networks and have the capability to serve DASH clients. In order to derive tolerable and acceptable values, all of the topologies are simulated 5 times and the average value of the results are shown in the tables and figures. Also, instead of reporting individual value for each random server, we just gives average value due to the space limitations.

We see in Figure 2 that the $PressureCache$ algorithm outperforms the random vDANE selection. In this figure, L0 refers to lower video quality, L1 the enhancement 1, and L2 has higher bitrate or representation quality. A client who receives more video from the L0 layer will be displaying video with lower quality. In contrast, receiving more video segments from layer L2 is proof that the client is experiencing better quality video. It is clear that clients in the $PressureCache$ setup have received far more video segments of a higher bitrate, while in the random approach, clients have received

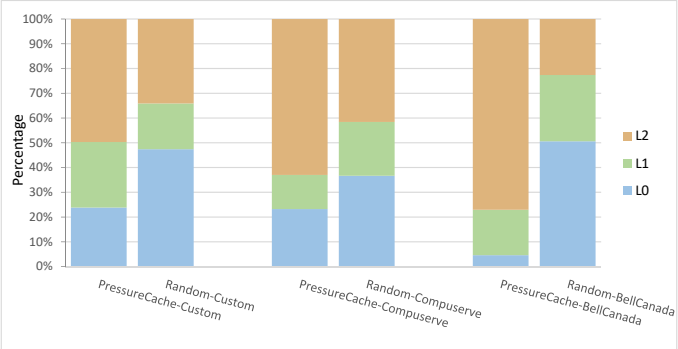


Fig. 2: Received video layer and playback quality

more segments from layer L0, the lower bitrate. We observe that the BellCanada network shows particularly good video quality using $PressureCache$. Table IIb show the average received bitrate in both $PressureCache$ and random vDANE selection. The numbers show that in all the three topologies, the $PressureCache$ approach derives a better performance since it has maximum received bitrate.

Increasing network traffic can result in reducing download speed and can lead to draining a client's buffer and consequently pausing video playback. In Table IIc the total outage duration (while the buffer is empty and client has to wait) per client is presented for both $PressureCache$ and random selection. It is obvious from the data that the $PressureCache$ has less buffer outages in all topologies and hence clients achieve an uninterrupted display. This is because the $PressureCache$ algorithms utilize the network links and use bandwidth in a desirable manner. Forwarding a client's request to the qualified vDANE improves network bandwidth utilization and reduce delay.

Figure 3 shows the first 60 segments of a single sample of the requested layers during streaming, shown as a sample time line. We see that the requested layer starts at Base layer (L0), goes to Enhancement 1 (L1) and then to Enhancement 2 (L2). The requested layer is mostly at L2 but drops to L1, and occasionally L0, depending on network conditions.

During the tests, the simulation is repeated 5 times for each topology. In order to show the average values of the requested layers across each of the runs, we have mapped the values observed, which are from a discrete domain of L0, L1, L2, into values in the continuous domain. We set L0 $\rightarrow$ 0.0, L1 $\rightarrow$ 1.0, and L2 $\rightarrow$ 2.0. Using this approach we can calculate the average values of the mapped requested layers. The averaged time-line trend of mapped values is depicted in Figure 4. As shown, at the beginning of the streaming, the client gets a

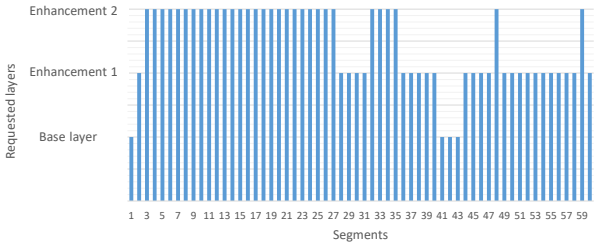


Fig. 3: Requested video layer per segment

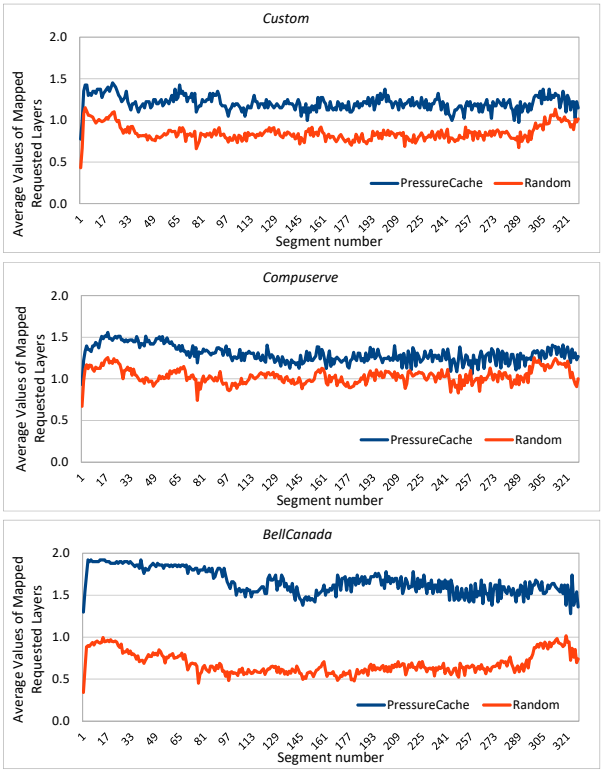


Fig. 4: Requested video layer per segment

lower bitrate layer in order to fill the buffer and get fast startup. Then it adaptively downloads video segments from the best layer as possible. In the $PressureCache$ models, clients request and received most of their video segments from enhancement layers which have a higher bitrate, while in the random case, the clients get a lower bitrate.

## V. Conclusions

In this paper, we presented an architecture utilizing SDN and NFV concepts as a proposal for SAND, which is a standardization process managed by the MPEG Group. For this purpose, we introduced a concept called vDANE, which are virtual web-caches that are aware of DASH characteristics, and we proposed a placement algorithm for those vDANEs. In order to show the performance of the proposed architecture, we comparatively tested the proposed algorithm over different custom and real world topologies. The results show that the proposed algorithm provides up to 42% increase in received video bitrate and up to 90% decrease in outage durations. This means that clients in the $PressureCache$ approach receive higher bit rate and experience better quality, while the average of random vDANE placement drives to lower bit rates.

As future work, we plan to enhance our study by improving the algorithm by considering dynamic network topologies. We also plan to develop an approach for vDANE replacement.

## Acknowledgements

This work was partially supported by the EU projects: 5GEX – "5G Multi-Domain Exchange" (671636), NECOS – "Novel Enablers for Cloud Slicing" (777067) and the Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 115E449.

## References

[1] MPEG's, "DASH-IF Position Paper: Server and Network Assisted DASH (SAND), ISO/IEC 23009-5, published December 2016."
[2] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *CoRR*, vol. abs/1406.0440, 2014.
[3] M. Chiosi, D. Clarke, P. Willis, A. Reid *et al.*, "Network Functions Virtualisation," White paper at the SDN and OpenFlow World Congress, ETSI, Tech. Rep., 2012.
[4] N. Bouten, J. Famaey, R. Mijumbi, B. Naudts, J. Serrat, S. Latré, and F. D. Turck, "Towards NFV-based multimedia delivery," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 738–741.
[5] "Opac: An optimal placement algorithm for virtual cdn," *Computer Networks*, vol. 120, pp. 12 – 27, 2017.
[6] C. Cetinkaya and M. Sayit, "Video-on-demand system architecture with alto-sdn integration," in *2016 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2016.
[7] A. Heikkinen, T. Ojanperä, and J. Vehkaperä, "Dynamic cache optimization for dash clients in content delivery networks," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*.
[8] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering stable high-quality video: An sdn architecture with dash assisting network elements," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16, 2016, pp. 4:1–4:10.
[9] R. S. Kalan, C. Cetinkaya, and M. Sayit, "Design of a layer-based video streaming system over software-defined networks," in *2017 8th International Conference on the Network of the Future (NOF)*.
[10] P. Georgopoulos, M. Broadbent, B. Plattner, and N. Race, "Cache as a service: Leveraging SDN to efficiently and transparently support video-on-demand on the last mile," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–9.
[11] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The Dynamic Placement of Virtual Network Functions," in *1st IEEE / IFIP International Workshop on SDN Management and Orchestration*, 2014.
[12] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, 2014.
[13] Z. Li, M. K. Sbaï, Y. Hadjadj-Aoul, A. Gravey, D. Alliez, J. Garnier, G. Madec, G. Simon, and K. Singh, "Network friendly video distribution," in *2012 Third International Conference on The Network of the Future (NOF)*, Nov 2012, pp. 1–8.
[14] R. G. Clegg, S. Clayman, G. Pavlou, L. Mamatas, and A. Galis, "On the selection of management/monitoring nodes in highly dynamic networks," *Computers, IEEE Trans. on*, vol. 62(6), pp. 1207–1220, 2013.
[15] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *IEEE Trans. on Network and Service Management*, vol. 12(1), pp. 18–33, 2015.
[16] "Elephants Dream (ED-II), http://concert.itec.aau.at/SVCDataset."
[17] "http://www.topology-zoo.org."