On Optimizing Resource Utilization in AVC-based Real-time Video Streaming

Alireza Erfanian[‡], Farzad Tashtarian[†], Reza Farahani[‡], Christian Timmerer[‡,*], Hermann Hellwagner[‡]
[‡]Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt, Austria
[†]Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran
[*]Bitmovin, Klagenfurt, Austria

***Abstract*—Real-time video streaming traffic and related applications have witnessed significant growth in recent years. However, this has been accompanied by some challenging issues, predominantly resource utilization. IP multicasting, as a solution to this problem, suffers from many problems. Using scalable video coding could not gain wide adoption in the industry, due to reduced compression efficiency and extra computational complexity. The emerging software-defined networking (SDN) and network function virtualization (NFV) paradigms enable researchers to cope with IP multicasting issues in novel ways. In this paper, by leveraging the SDN and NFV concepts, we introduce a cost-aware approach to provide advanced video coding (AVC)-based real-time video streaming services in the network. In this study, we use two types of virtualized network functions (VNFs): *virtual reverse proxy (VRP)* and *virtual transcoder (VTF)* functions. At the edge of the network, VRPs are responsible for collecting clients' requests and sending them to an SDN controller. Then, executing a mixed-integer linear program (MILP) determines an optimal multicast tree from an appropriate set of video source servers to the optimal group of transcoders. The desired video is sent over the multicast tree. The VTFs transcode the received video segments and stream to the requesting VRPs over unicast paths. To mitigate the time complexity of the proposed MILP model, we propose a heuristic algorithm that determines a near-optimal solution in a reasonable amount of time. Using the MiniNet emulator, we evaluate the proposed approach and show it achieves better performance in terms of cost and resource utilization in comparison with traditional multicast and unicast approaches.**

***Keywords*—Dynamic Adaptive Streaming over HTTP (DASH), Real-time Video Streaming, Software Defined Networking (SDN), Video Transcoding, Network Function Virtualization (NFV).**

I. Introduction

In recent years, video streaming has gained the largest portion of Internet traffic. According to [1], video streaming will grow up to 88% of the total Internet traffic by 2022. Thus, video streaming has become a dominant application for current networks and, especially due to real-time video streaming, also a major challenge. On the client side of video streaming, a wide variety of devices and applications have emerged; in conjunction with advances in mobile networks, this leads to a more and more heterogeneous environment. Hence, providing different customized services for video streaming applications is essential. Currently, real-time video streaming may have to use multiple unicast connections to stream a video to multiple clients. This approach, however, can generate redundant traffic and waste a significant amount of network resources. Thus, researchers try to address this issue by mostly relying on two main approaches: *i)* use multicast approaches to reduce redundant traffic, and *ii)* use more efficient video coding methods for real-time streaming.

Leveraging multicast communication could lead to a considerable reduction in overall bandwidth consumption in backbone networks [2]. However, current IP multicast approaches like IGMP [3] and PIM-SM [4] still face several challenges. First, each router has to maintain the state of a multicast group, which requires complicated operations in the routers. Second, IP multicast routers do not have a global view of the network status and can hardly determine the optimal multicast trees for ensuring end-to-end quality of service (QoS) requirements [5]. Finally, the multicast topology for video streaming is usually dynamic, *i.e.*, clients can join and leave on-the-fly. However, current IP networks are not able to reconfigure the routing paths dynamically and adaptively. To overcome the IP multicast limitations, some papers proposed application-layer multicasting [6]–[9]. Since this approach uses overlay networks, a stream may traverse several times through some nodes in IP networks. This can also result in wasting network resources and increasing transmission delays.

With the emergence of SDN, SDN multicast was proposed with advantages over IP multicast [10]–[12]. In SDN, the data plane and the control plane are decoupled [13], which allows applications to collaborate with network nodes to achieve intelligent and dynamic service provisioning. The data plane consists of hardware or software elements dedicated to forwarding packets. The control plane, *i.e.*, an SDN controller, manages the data plane elements. SDN provides a flexible platform that can perform adaptive routing algorithms for different network applications. Besides, SDN can modify traffic to guarantee QoS for certain traffic flows, *e.g.*, video streams.

In addition to deploying multicast for reducing the network traffic, specific video encoding methods can be adopted to achieve that goal. Scalable video coding (SVC) [14] provides scalable representations of AVC-encoded videos. It offers three types of scalability, *i.e.*, spatial, temporal, and quality scalability. SVC encodes a video into N layers including a base layer and multiple enhancement layers. The reception and decoding of more layers generally lead to improved video quality on the receiver's end. This allows to serve different requested qualities in heterogeneous networks by transmitting enhancement layers instead of videos encoded separately in different qualities; and it makes SVC suitable for a multicast environment. However, the scalability of SVC comes at a cost in coding efficiency. The SVC overhead can roughly be estimated at minimum 10% per enhancement layer in comparison with AVC. This means transferring a video with four enhancement layers utilizes at least 50% more network resources than transferring the best quality with AVC [15]. Moreover, when using DASH-SVC streaming, each layer of each video segment needs to be requested separately, imposing HTTP message overhead and RTT delays. Also, multicasting different SVC layers requires establishing different multicast trees; constructing and

978-1-7281-5684-2/20/$31.00 ©2020 IEEE

maintaining these trees are still challenging issues in this area.

In this paper, motivated by the above-mentioned issues, we propose an AVC-based real-time video multicast streaming framework by leveraging SDN and NFV. We note here that the proposed approach can be easily applied to high efficiency video coding (HEVC) and scalable high efficiency video coding (SHVC). In this study, we employ two types of VNFs: *i)* a set of virtual reverse proxy servers (VRPs) that are used on the edge of the network to aggregate the clients' requests and send them to the SDN controller [16], and *ii)* a set of virtual transcoder functions (VTFs) that are hosted by some point-of-presence (PoP) nodes to achieve a significant reduction in bandwidth consumption. After gathering requests from VRPs, the SDN controller executes a mixed-integer linear programming (MILP) optimization model to determine a multicast tree from an optimal source media server to an appropriate subset of PoPs running VTFs. Then, each AVC-encoded video segment in its highest requested quality must be delivered over the multicast tree from the source node to VTFs. Since VTFs are responsible to satisfy VRPs' requests, they produce the lower qualities of each segment received from the source and then transmit them to the VRPs in a unicast fashion. It is worth noting that using VTFs results in substantial saving in multicast bandwidth usage due to transcoding to other requested qualities; in contrast, in SVC multicasting, each desired quality should be streamed from the source to VRPs. However, transcoding imposes decoding and (predominantly) re-encoding costs on the network. Thus, placement of transcoders in the edge means more transcoders and therefore higher transcoding costs. On the other hand, placing transcoders closer to the video origin results in fewer transcoders but higher bandwidth utilization. In summary, the present study's main contributions can be described as follows:

- We leverage the SDN concept and NFV technology to efficiently serve DASH clients' requests in AVC real-time streaming.
- We propose an MILP model to jointly construct the optimal multicast tree and VTFs' placement with the objective of minimizing the resource utilization and VTFs' costs.
- We propose a heuristic approach to achieve a near optimal solution in polynomial time.
- We evaluate the performance of the proposed framework using MiniNet and compare it with other SVC- and AVC-based multicast and unicast approaches.

II. Related Work

Current multicast standards such as PIM-SM [4] build the shortest path tree by employing unicast routing protocols. However, it is difficult to reduce bandwidth consumption efficiently when constructing the shortest path tree, because the path is calculated in a distributed manner by the participating routers and their local information may miss some edge aggregation chances. On the contrary, a Steiner tree [17] could determine the optimal tree with minimum cost. Nevertheless, finding the Steiner tree has been proven to be NP-hard; thus, it is not adopted in current multicasting approaches. Some heuristic algorithms, such as [18], have been proposed to address Steiner tree time complexity. The receiver-driven layered IP multicast protocol was proposed by McCanne *et al.* [19]. They proposed SVC-style transmission based on IP multicast over heterogeneous networks. IP multicast limitations have motivated researchers to deploy a multicast tree in the application layer. Ref. [8] proposed a hybrid approach to build an overlay tree on both tree and mesh designs; the problem of finding a set of stable nodes to construct a tree-based backbone was formulated in the paper. Constructing multiple multicast trees for SVC streaming by using a P2P overlay network was proposed in [20]. The paper tried to alleviate the IP multicast limitations by leveraging the overlay architecture to build a bandwidth-aware multicast tree. However, due to its communication overhead, this approach is not efficient compared to traditional IP multicast.

In the SDN area, some papers, such as [13], [21], showed the SDN capabilities for unicast traffic engineering. SDN was also considered for multicast traffic in the literature. Ref. [10] used SDN to propose a multicast approach in data center networks; the proposed algorithm achieved a significant reduction in bandwidth utilization. Huang *et al.* [22] built multiple multicast trees by considering both link and network capacities. Ref. [23] proposed an SDN multicast framework for SVC streaming called SDM2Cast. The framework is able to identify and process SVC streaming in the network nodes. However, the paper did not propose any routing algorithm for SVC streaming. A streaming framework was devised in [24] employing segment routing (SR), SDN, and multicasting. It provides an efficient live streaming service for 5G mobile users by constructing a multicast tree.

Using SDN and NFV technologies could lead to the deployment of infrastructure in a cost-effective way. The SDN controller can determine the optimal routing path that passes through a chain of VNFs to fulfill the requirement of different services like video transcoding. Zhang *et al.* proposed a mechanism to build a multicast tree where there is a single VNF processing step, in order to minimize the NFV and link costs [11]. A framework for routing and VNF placement in an optical network was proposed in [25]. These papers proposed their approaches based on the assumption that each server could host just one VNF type for each source-destination pair. On the contrary, [26] assumed that some servers could host all types of VNFs, so each flow needs to pass through only one server to meet all the required VNFs. Ref. [27] proposed an MILP model to jointly determine optimal routing and VNF placement with minimizing both NFV and link costs. Since the problem is NP-hard, the authors proposed a heuristic algorithm to alleviate time complexity. Xu *et al.* devised an approximation algorithm to obtain VNF placement in the SDN multicast tree in order to minimize the NFV implementation costs in terms of computing and bandwidth resource consumption [28].

The authors in [29] leveraged the SDN controller information to construct a multicast tree for a video conferencing system. A multi-source multi-destination manycast approach was proposed in [30]. The authors formulated an ILP model and a heuristic algorithm to determine the manycast tree for real-time SVC streaming. The objective of the proposed model was to improve the available link capacity utilization. The authors assumed that multiple geographically distributed servers provide video sources and clients can join a real-time streaming service dynamically. Yang *et al.* [31] presented a multicast approach for an OpenFlow network to save network bandwidth. The OpenFlow controller provides controllable video multicasting by routing the flow for each video layer separately. Instead of IGMP, they utilized a centralized management system to

maintain the relationships between hosts and groups. Ref. [32] introduced an approach for SVC video conferencing based on SVC video multicasting called ASCast. The paper formulated an ILP model to build a multicast tree with the goal of maximizing delivered video layers to clients while considering the ternary content-addressable memory (TCAM) size as a constraint. Ref. [33] uses an equivalent bandwidth theory to estimate the available bandwidth for the links in the network. A finite-state machine is employed to do in-network adaptation, based on the estimated bandwidth, and to determine the video layers that could be delivered through the network.

III. Motivating Example

Before introducing the proposed approach, let us present our main motivation through the following example. Assume the *Tears of Steel* video sequence is encoded by H.264/AVC in different qualities as shown in Table I, following [34]. Also, we use [35] to encode the sequence by SVC. We consider a simple topology consisting of one media server S, four point-of-presence nodes (PoPs) $P1$-$P4$, and two virtual reverse proxy servers (VRPs) $X1$ and $X2$ (Fig. 1). The two VRPs $X1$ and $X2$ are hosted by two local servers each of which is responsible to gather all requests from DASH clients in cells A and B, respectively. Suppose $X1$ and $X2$ collected the following quality requests from DASH clients: $X1$: [QId-0,QId-4], $X2$: [QId-1,QId-3]. In this example, we are going to evaluate the performance of using SVC and AVC encoding in terms of bandwidth usage for serving the clients in the following scenarios. In the first scenario, we evaluate the performance of SVC encoding. Since in SVC all layers up to the highest requested quality must be simultaneously delivered from the source server to each VRP [14], we consider four multicast trees for QId-0 to QId-3 and one unicast path for delivering QId-4 from S to $X1$ (Fig. 1(a)). All paths are shown in different colors according to the video layers' colors. As illustrated in Fig. 1(a), the total consumed bandwidth for serving all proxies is 136.8 Mbps. In the second scenario, by leveraging NFV technology, we employ VTFs to substantially enhance AVC-based video streaming performance. Using VTFs enables us to reduce bandwidth usage by sending only the maximum requested QId (here QId-4 in X1 is the maximum quality) from the source node to VTFs over a multicast tree. The VTFs are responsible for transcoding the received segments into the requested qualities and unicasting them to each proxy accordingly. As depicted in Fig. 1(b), by delivering QId-4 from S to a VTF on PoP $P2$, it can serve the requested qualities by separate unicast paths. The total consumed bandwidth is obtained at 112.2 Mbps. In a third scenario, by adding another VTF, we can achieve even less bandwidth usage. As Fig. 1(c) shows, the traffic flow from $P2$ to $X1$ can be reduced by launching another VTF on the server in cell A. Thus, the total bandwidth usage reaches 107.8 Mbps, at least a 20% reduction compared to the SVC scenario. Although using VTFs reduces total bandwidth usage, some issues must be addressed in this study: *i)* specifying the optimal number and locations of the VTFs, *ii)* determining an appropriate multicast tree from an optimal source media server to the VTFs, and *iii)* designing accurate data paths from the VTFs to the destination VRPs.

IV. System Model and Problem Formulation

The proposed approach can be introduced in three main layers: *i)* application/control, *ii)* network core, and *iii)* network edge (see Fig. 1(d)). Consider a limited set of media servers, denoted by $\mathcal{S}$, and an SDN controller placed in the first layer. For the next layer, we set $\mathcal{P}$ as a group of point-of-presence (PoP) nodes equipped with OpenFlow switches connected to the SDN controller and sufficient resources to be able to launch VTFs. In the network edge layer, using a finite number of VRPs, we can aggregate the DASH clients' requests. To cope with the addressed problem, we employ two sets of VNFs, as introduced above: *i)* virtual reverse proxies (VRPs) and *ii)* virtual transcoder functions (VTFs), denoted by $\mathcal{X}$ and $\mathcal{T}$, respectively. For simplicity of presentation, we consider that clients request to receive different qualities of a segment of real-time video stream v.

Let $q_{c,x}$ be the requested quality of DASH client c received by VRP x. After collecting requests, each VRP x should update its serving quality list for video v, denoted by Q_x, and send it to the SDN controller. Furthermore, the SDN controller sets Q^* as the maximum quality level requested by VRPs for video v. It is also assumed that all requests must be served during τ seconds as a deadline. A data path from a set of source media servers to the DASH clients can be split into three parts: *i)* a multicast tree from an optimal media server denoted by $\mathcal{S}_{opt}$ to $\mathcal{T}_{opt}$ running on $\mathcal{P}_{opt}$, *ii)* unicast data paths from $\mathcal{T}_{opt}$ to each VRP, and *iii)* data transmission from each VRP to DASH clients, where $\mathcal{T}_{opt}$ and $\mathcal{P}_{opt}$ are the optimal subsets of $\mathcal{T}$ and $\mathcal{P}$ determined by the SDN controller, respectively. In this paper, we are going to address the first two parts. Now, to find an optimal solution in terms of determining the multicast tree, the unicast data paths and optimal data transmission rates, we introduce a mixed-integer linear program (MILP). Let us first represent the deadline constraint. Considering the given deadline τ, assume that each segment of video v is sent over the multicast and unicast paths in θ_{multi} and θ_{uni} units of time, respectively; furthermore, it should be processed by $\mathcal{T}_{opt}$ in θ_{trans} time. Thus, we have:

$$\theta_{multi} + \theta_{uni} + \theta_{trans} \leq \tau \tag{1}$$

For ease of explanation, the remaining constraints can be categorized into two groups, according to the subproblems: *i)* create multicast tree and VTF placement; and *ii)* design unicast paths and assign $\mathcal{T}_{opt}$ to VRPs (see Table II for notations).

i) Multicast Tree and VTF Placement

Creating the multicast tree and using VTFs enables us to optimize bandwidth usage by sending quality level Q^* to $\mathcal{T}_{opt}$ instead of delivering all quality levels requested by VRPs. The VTFs in $\mathcal{T}_{opt}$ are responsible for transcoding and delivering the requested level(s) to each VRP x according to Q_x. Thus, this group of constraints focuses on how to create a multicast tree and its data transmission rate from $\mathcal{S}_{opt}$ to $\mathcal{T}_{opt}$. To this end, we need to determine an optimal source media server ($\mathcal{S}_{opt}$) and an optimal number of VTFs ($\mathcal{T}_{opt}$); also, an optimal subset of $\mathcal{P}$ ($\mathcal{P}_{opt}$) must be selected to host $\mathcal{T}_{opt}$. The following equation determines appropriate VTF locations, with binary

Table I: **Resolutions and bitrates of *Tears of Steel* encodings [34]**.

QualityId	Resolution	AVC bitrate	SVC bitrate
QId-0	1280x760	2.2 Mbps	2.2 Mbps (BL)
QId-1	1920x1080	4.1 Mbps	4.5 Mbps (BL+EL1)
QId-2	2560x1440	6.7 Mbps	8.1 Mbps (BL+EL1-2)
QId-3	3840x2160	11.6 Mbps	15 Mbps (BL+EL1-3)
QId-4	3840x2160	19.1 Mbps	26.7 Mbps(BL+EL1-4)

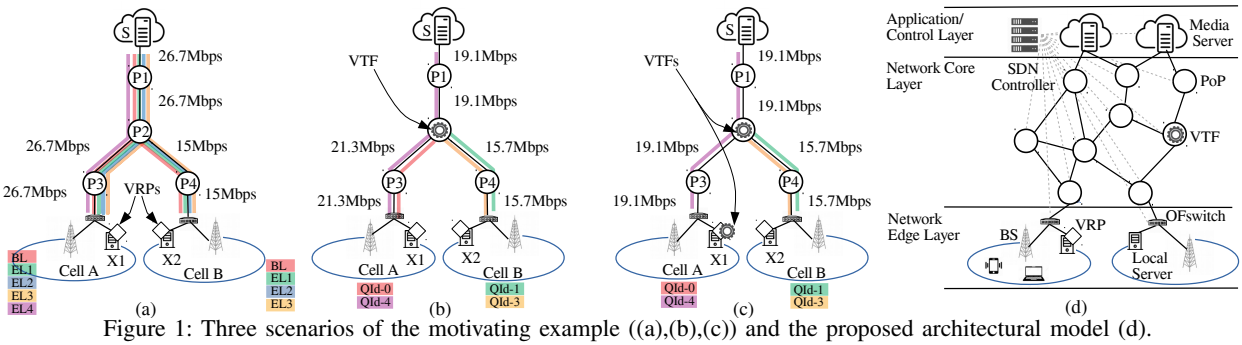

Figure 1: Three scenarios of the motivating example ((a),(b),(c)) and the proposed architectural model (d).

variable $B_{t,p} = 1$ denoting that VTF t is placed on PoP p:

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} B_{t,p} \geq 1 \tag{2}$$

It is assumed at least one and at most $|\mathcal{X}|$ VTFs can be selected to serve all proxies (VRPs) in $\mathcal{X}$. It is worth noting that for designing a multicast tree, the accuracy of the tree in terms of connectivity and supporting sufficient bandwidth must be guaranteed. Therefore, to satisfy the connectivity feature, we simply consider that $\mathcal{S}_{opt}$ has to send a very small segment with size m to the $k = |\mathcal{X}|$ VTFs:

$$\sum_{s\in\mathcal{S}} Y_s = 1 \tag{3}$$

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} r^t_{s,p} = mY_s \qquad \forall s \in \mathcal{S} \tag{4}$$

$$\sum_{j\in\{\mathcal{P}\cup\mathcal{X}\}} r^t_{p,j} - \sum_{j\in\{\mathcal{P}\cup\mathcal{S}\}} r^t_{j,p} \leq -B_{t,p}\frac{m}{k} \qquad \forall t \in \mathcal{T}, p \in \mathcal{P} \tag{5}$$

where the binary variable Y_s determines whether media server s is responsible for sending data ($Y_s = 1$) or not ($Y_s = 0$). Moreover, for two connected nodes i and $j \in \{\mathcal{S}\cup\mathcal{P}\}$, $r^t_{i,j}$ is the amount of data toward destination $t \in \mathcal{T}$ that should be transmitted from node i to j. Equality constraint (3) selects only one media server as $\mathcal{S}_{opt}$. Equation (4) forces $\mathcal{S}_{opt}$ as the only source node to send the assumed small segment with size m. Constraint (5) states that those PoPs that participate in the multicast tree but do not host a VTF (or $B_{t,p} = 0$) must deliver all received data to their neighbors; the other PoPs that host the VTF(s) must receive at least m/k amount of data. However, we use $r^t_{i,j}$ only for satisfying the connectivity among selected nodes in the multicast tree, while to support the segment with quality Q^* of video v, denoted by ξ, the following constraints must be met:

$$r^t_{i,j} \leq L_{i,j} \qquad \forall t \in \mathcal{T}, i, j \in \{\mathcal{S}\cup\mathcal{P}\cup\mathcal{X}\} \tag{6}$$

$$L_{i,j}\xi \leq w_{i,j}\theta_{multi} \qquad \forall i, j \in \{\mathcal{S}\cup\mathcal{P}\cup\mathcal{X}\} \tag{7}$$

where $L_{i,j}$ is a binary variable that determines whether link (i, j) is used in the multicast tree ($L_{i,j} = 1$) or not ($L_{i,j} = 0$). Constraint (6) states that if $r^t_{i,j} \geq 0$ then $L_{i,j}$ must be set to 1; however, if link (i, j) does not provide the required bandwidth for transmitting the requested segment with length ξ during θ_{multi}, it must be set to 0 (Equation (7)). In fact, the value of ξ is adjusted according to the quality Q^* by the SDN controller. We define $\mathcal{F}$ as the set of available VTF instance types, each of which provides different VTF processing capacity. Note that a VTF with greater processing capacity needs more resources to be launched and is more expensive. Now, to select at least one instance type $f \in \mathcal{F}$ for each selected VTF t, the following constraint should be satisfied:

$$\sum_{p\in\mathcal{P}} B_{t,p} \leq \sum_{f\in\mathcal{F}} F_{t,f} \leq |\mathcal{F}| \sum_{p\in\mathcal{P}} B_{t,p} \qquad \forall t \in \mathcal{T} \tag{8}$$

where $|\mathcal{F}|$ shows the size of set $\mathcal{F}$. For each $t \in \mathcal{T}_{opt}$ with VTF instance type f, its processing time $\theta_{t,f}$ and required resources $c_{t,f}$ should be taken into account. Thus, through the next constraint, the maximum segment processing time by the instance type f of VTF t is measured:

$$\sum_{f\in\mathcal{F}} F_{t,f}\theta_{t,f} \leq \theta_{trans} \qquad \forall t \in \mathcal{T} \tag{9}$$

We note that the processing time of different segment sizes by each VTF instance type is assumed to be equal. Furthermore, the following constraint guarantees that the total resource consumption by the selected VTF instances does not exceed the available resource of PoP p:

$$\sum_{t\in\mathcal{T}}\sum_{f\in\mathcal{F}} F_{t,f}c_{t,f} \leq C_p \qquad \forall p \in \mathcal{P} \tag{10}$$

where C_p is the total available resources of PoP p.

ii) Unicast Paths and $\mathcal{T}_{opt}$ to VRP Assignment

To deliver transcoded data from the set $\mathcal{T}_{opt}$ to each VRP x regarding its requested quality list Q_x, we first need to find an optimal VTF $t \in \mathcal{T}$ and then determine an optimal unicast path from t to VRP x. Let us assume that each VRP x must be served by one VTF $t \in \mathcal{T}$; thus, we have:

$$\sum_{t\in\mathcal{T}} A_{t,x} = 1 \qquad \forall x \in \mathcal{X} \tag{11}$$

where the case that binary variable $A_{t,x} = 1$ means that VTF t is in charge of serving VRP x. Let $d^{t,x}_{i,j}$ be the amount of data that must be forwarded from $i \in \mathcal{P}$ to $j \in \{\mathcal{P}\cup\mathcal{X}\}$, with origin VTF t and destination VRP x. As we mentioned earlier, each VRP x must receive a specific amount of data denoted by ζ_x according to its Q_x which can be formulated as follows:

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}} d^{t,x}_{p,x} = \zeta_x \qquad \forall x \in \mathcal{X} \tag{12}$$

Since we allocated θ_{uni} units of time for unicast data transmission (see Equation (1)), the next constraint selects links with sufficient bandwidth:

$$\sum_{x\in\mathcal{X}}\sum_{t\in\mathcal{T}} d^{t,x}_{i,j} \leq w_{i,j}\theta_{uni} \qquad \forall i, j \in \{\mathcal{S}\cup\mathcal{P}\cup\mathcal{X}\} \tag{13}$$

Table II: **Notations**.

Notation	Description
Input Parameters	
$\mathcal{S}, \mathcal{S}_{opt}$	Set of origin servers and an optimal one for serving clients
$\mathcal{P}, \mathcal{P}_{opt}, C_p$	Set of PoPs and optimal subset for serving users where C_p shows available resources of $p \in \mathcal{P}$
$\mathcal{T}, \mathcal{T}_{opt}, k$	Set of k available virtual transcoder functions (VTFs) and optimal subset for serving clients
$\mathcal{F}_t, c_{t,f}, \theta_{t,f}$	Set of various instance types for $t \in \mathcal{T}$ where each instance f needs $c_{t,f}$ amount of resources and takes $\theta_{t,f}$ seconds to process a segment
$\mathcal{X}$	Set of virtual reverse proxy servers (VRPs) that received at least one request from users
τ	Given deadline for delivering the requested segment from $\mathcal{S}_{opt}$ to $\mathcal{X}$
$w_{i,j}, e_{i,j}$	$w_{i,j}$ is the available bandwidth of link $e_{i,j}$ where $i,j \in \{\mathcal{S} \cup \mathcal{P} \cup \mathcal{X}\}$
Q_x, Q^*	Each VRP x holds Q_x as the list of qualities requested by its clients; Q^*=$max\{Q_x \vert \forall x \in \mathcal{X}\}$
ξ	Size of requested segment with the quality Q^* that must be delivered from $\mathcal{S}_{opt}$ to $\mathcal{T}_{opt}$
ζ_x	Total size of segments with qualities Q_x that must be delivered from $\mathcal{T}_{opt}$ to VRP x
m	Small positive value
Variables	
$B_{t,p}$	$B_{t,p}$=1 if PoP p hosts transcoder t else $B_{t,p}$=0
$A_{t,x}$	$A_{t,x}$=1 if $t \in \mathcal{T}$ serves proxy x else $A_{t,x}$=0
$L_{i,j}$	$L_{i,j}$=1 if link $e_{i,j}$ is selected else $L_{i,j}$=0
Y_s	$Y_s = 1$ if server s does serve requests else Y_s=0
$F_{t,f}$	Shows whether instance type f is selected for VTF t ($F_{t,f} = 1$) or not ($F_{t,f} = 0$)
$r^t_{i,j}$	Amount of transmitted data on link $e_{i,j}$ toward $t \in \mathcal{T}_{opt}$ during θ_{multi} where $i,j \in \{\mathcal{S} \cup \mathcal{P}\}$
$d^{t,x}_{i,j}$	Amount of transmitted data on link $e_{i,j}$ from $t \in \mathcal{T}_{opt}$ to $x \in \mathcal{X}$ during θ_{uni}, $i,j \in \{\mathcal{P} \cup \mathcal{X}\}$
$\theta_{multi}, \theta_{uni}$	Time duration for transmitting data from set $\mathcal{S}$ to $\mathcal{T}$ and from $\mathcal{T}$ to $\mathcal{X}$, respectively
θ_{trans}	Amount of time for processing a segment by $\mathcal{T}_{opt}$

Moreover, we need to guarantee that if a VTF is launched on a PoP, it has to send data to at least one VRP; in addition, the other participating PoPs that do not host any VTFs must deliver all incoming data to their next hop. We formulate these constraints as follows:

$$\sum_{j \in \mathcal{P}} d^{t,x}_{p,j} - \sum_{j \in \mathcal{P}} d^{t,x}_{j,p} \leq A_{t,x}\zeta_x \qquad \forall x \in \mathcal{X}, t \in \mathcal{T}, p \in \mathcal{P} \quad (14)$$

$$\sum_{j \in \mathcal{P}} d^{t,x}_{p,j} - \sum_{j \in \mathcal{P}} d^{t,x}_{j,p} \leq B_{t,p}\zeta_x \qquad \forall x \in \mathcal{X}, t \in \mathcal{T}, p \in \mathcal{P} \quad (15)$$

Problem Formulation

Since the SDN controller is in charge of launching data paths and serving all clients with the minimum cost, the objective function of the proposed MILP model considers two main normalized costs for launching VTFs and creating data paths (multicast tree and unicast paths) as follows:

$$Min. \quad \alpha_1 \sum_{t,f} \frac{F_{t,f}\pi_{t,f}}{F^*} + \alpha_2 \Big(\sum_{i,j,t,x} \frac{d^{t,x}_{i,j}\pi^{'}_{i,j}}{D^*} + \sum_{i,j} \frac{L_{i,j}\pi^{'}_{i,j}}{L^*} \Big) \quad (16)$$

s.t. Constraints (1) – (15)

var. $B_{t,p}, A_{t,x}, L_{i,j}, Y_s, F_{t,f} \in \{0,1\}$,

$$r^t_{i,j}, d^{t,x}_{i,j}, \theta_{multi}, \theta_{uni}, \theta_{trans} \geq 0$$

where F^*, D^*, and L^* are the maximum costs of running VTFs, unicast data transmission, and multicast tree, respectively. Moreover, we define two weight coefficients α_1 and α_2 to set desirable priorities for these terms. The weights can be adjusted by the application manager where $\alpha_1 + \alpha_2 = 1$. On the one hand, the SDN controller has to appropriately react to any modifications of serving video stream v, in terms of clients joining and leaving stream v, by running the proposed MILP model; on the other hand, it is not reasonable to re-create data paths and change VTFs' instance types or even their locations at each execution of the MILP model due to its overhead, *e.g.*, incurred by reconfiguring OpenFlow switches. Therefore, we set priority for the current configuration (*e.g.*, selected multicast data path, and the number and positions of VTFs) by defining two groups of cost values $\pi_{t,f} \forall t \in \mathcal{T}, f \in \mathcal{F}$ and $\pi^{'}_{i,j} \forall i,j \in \{\mathcal{S} \cup \mathcal{P} \cup \mathcal{X}\}$ for existing $\mathcal{T}_{opt}$ and data paths, respectively. In fact, the costs of links that are currently used in the multicast tree and unicast paths and also the costs of launched VTFs are set lower than those of other links and VTFs that do not collaborate in any streaming service. Thus, we give more preference to the existing configuration of underlying data paths to be considered by the MILP model.

V. Heuristic Algorithm

Due to the high time complexity of the proposed MILP model (16), we introduce an efficient heuristic method to find a near-optimal solution. By considering the objective function and the proposed constraints, the heuristic algorithm is designed in two main procedures: *(I)* selecting an optimal source node and creating a low-cost multicast tree from that node to the given proxy servers (VRPs), and *(II)* cost-aware VTF placement on the obtained multicast tree. In fact, the first procedure determines $\mathcal{S}_{opt}$ and $\mathcal{P}_{opt}$ while the second one selects $\mathcal{T}_{opt}$. However, to obtain the multicast tree and $\mathcal{T}_{opt}$, the maximum time duration for delivering data over the multicast tree and also an accurate instance type for $\mathcal{T}_{opt}$ must be given to the procedures (I) and (II) as input parameters, respectively.

To do this, we devise a caller procedure to execute the other two procedures with different input parameters iteratively. As shown in Alg. 1, this procedure begins with the given set of $\mathcal{X}$ and τ as input parameters and returns an appropriate multicast tree, the positions of VTFs, and selected instance type as the

Algorithm 1: Caller Procedure

Input: $\mathcal{X}, \tau$
Output: $MulticastTree, TransPos, VTFType$
Global $\mathcal{S}, \mathcal{P}, \mathcal{T}, \mathcal{F}$
Initialize: $Flag$ =True, $Results = [\,]$, $BestCost = \infty$
$\mathcal{F}^*$=Sort $\mathcal{F}$ based on their cost in ascending order
for $f^* \in \mathcal{F}^*$ **do**
  θ^* =ProcessingDelay(f^*)
  $\theta_{path} = \tau - \theta^*$
  if $\theta_{path} > 0$ **then**
    Update the required bandwidth for creating multicast tree
    $MulticastTree$ =CreateMulticastTree($\theta_{path}, \mathcal{X}$)
    if $MulticastTree \neq \emptyset$ **then**
      $[Pos, Cost]$=TranscoderPlacement($f^*, MulticastTree$)
      if $Cost < BestCost$ **then**
        $Results = [MulticastTree, Pos, f^*]$
        $BestCost = Cost$
return $Results[MulticastTree, Pos, f^*]$

Algorithm 2: Procedure (I): CreateMulticastTree

Input: $\theta_{path}, \mathcal{X}$
Output: $MulticastTree$
1 **Initialize:** $MulticastTree = [\,]$
2 $[x^*, q^*] = max\{Q_x | \forall x \in \mathcal{X}\}$
3 $PathSet$=AllShortestPaths($\mathcal{S}, x^*, q^*, \theta_{path}$)
4 $MulticastTree$=LowCostPath($PathSet$)
5 $\mathcal{X}$=$\mathcal{X}$-x^*
6 **while** $\mathcal{X} \neq \emptyset$ **do**
7 $[x^*, q^*] = max\{Q_x | \forall x \in \mathcal{X}\}$
8 $PathSet$=AllShortestPaths($MulticastTree, x^*, q^*, \theta_{path}$)
9 $BestPath$=LowCostPath($PathSet$)
10 $MulticastTree$=AddNewPath($BestPath$,$MulticastTree$)
11 $\mathcal{X}$=$\mathcal{X}$-x^*
12 **return** $MulticastTree$

Algorithm 3: Procedure (II): TranscoderPlacement

Input: $f^*, MulticastTree$
Output: $BestPosition, BestCost$
1 $BestPosition$ =Root($MulticastTree$)
2 $BestCost$=CostCalculation($MulticastTree, BestPosition, f^*$)
3 $Flag$ =True
4 **while** $Flag$ **do**
5 **if** $NewPositions == \mathcal{X}$ **then**
6 Break()
7 $NextCandidate$ =FindBestChild($BestPosition$)
8 $NewPositions = BestPosition + NextCandidate$
9 **for** *pos in* $NewPositions$ **do**
10 **if** *AllChild(pos) in* $NewPositions$ **then**
11 $NewPositions = NewPositions - pos$
12 $NewCost$=CostCalculation($MulticastTree, NewPositions, f^*$)
13 **if** $NewCost < BestCost$ **then**
14 $BestCost = NewCost$
15 $BestPosition = NewPositions$
16 **else**
17 $Flag$ =False
18 **return** $BestPosition$,$BestCost$

results. After sorting the VTF instance type set $\mathcal{F}$ (line 3), the main *for* loop starts by selecting the lowest cost instance that provides the highest processing delay. In each iteration, θ_{path}, the maximum deadline for delivering data from the source node to the VRPs, is computed (lines 5 and 6); moreover, the required bandwidth is updated accordingly (line 8). For each VTF instance type and $\theta_{path} > 0$, procedure (I) is executed and, for its valid output, procedure (II) then will be launched. The total cost of each solution is calculated by procedure (II) and the best solution is kept in the $Results$ variable (line 13).

For creating the multicast tree, we use a greedy strategy in Alg. 2. First, we select proxy x^* that requested the maximum quality q^* in VRPs and then among all shortest paths from source nodes to x^*, the shortest one that supports the required bandwidth is selected (lines 2-4) and stored in the $MulticastTree$ variable. Each remaining proxy x^* in $\mathcal{X}$ joins the $MulticastTree$ through finding the shortest path from it to all nodes on the multicast tree using the Dijkstra algorithm (see *while* loop, lines 6-11). In each iteration, we select the proxy with the maximum requested quality among remaining proxies (line 7) to ensure that there is no bandwidth limitation on the multicast tree for the next proxy. In Alg. 3, we place an optimal number of VTFs with instance type f^* on the given multicast tree. We start from the root of the tree as the $BestPosition$ and calculate the total cost of placing a VTF on the $BestPosition$ as $BestCost$ (lines 1 and 2). In each iteration of the *while* loop, we select the best child node in $BestPosition$ (line 7) and then by having $BestPosition$, $NewPositions$ will be updated (line 8). Our criterion for selecting a child as the best child is the amount of unicast traffic that can originate from that node to the proxies if this node hosts a VTF. We select a child node that should send the most unicast traffic because this node can significantly change the total cost. If we have a node with all its children in $NewPositions$, that node must be omitted since its child(ren) should host VTFs (lines 9-11). If the $NewPositions$ results in lower cost, the algorithm proceeds for the next best child while the best position and cost are kept in $BestPosition$ and $BestCost$, respectively. Finally, if the $NewCost$ is greater than or equal to the $BestCost$ (line 17) or we have all proxies in $NewPositions$ (line 5), then the procedure returns the best results (line 18).

Table III: **Different VTF instance types**.

Index	Resources		Price($)	Processing Time
	CPU	RAM (GB)		
Micro	1 core	1	0.02	5.1 sec.
Small	2 cores	2	0.04	3.8 sec.
Medium	4 cores	4	0.06	2.3 sec.
Large	8 cores	4	0.12	1.7 sec.

VI. Performance Evaluation

A. Evaluation Setup and Description of Scenarios

In order to evaluate the proposed model in a realistic environment, we consider real network topologies provided by the *Internet Topology Zoo*[1]: a small-, a medium-, and a large-scale network. These topologies consist of 11, 47 and 113 PoP nodes and 5, 15 and 30 VRPs, respectively. The number of DASH clients for each VRP and the quality levels requested by each DASH client are randomly set between 1 and 10, and 1 and 5, respectively. Also, we use *MiniNet*[2] as the emulation system and *Floodlight*[3] as the SDN controller. Further, we use the *Tears of Steel* video sequence encoded by H.264/AVC with 4 sec. segment length [34] (Table I). Moreover, we employ the *JSVM11* [35] to create a scalable version of the video. Although the proposed MILP model supports a deadline for delivering segments, the users' requests are held back in VRPs for two segments in order to have time for transcoding in VTFs. The costs and specifications of the different VTF instance types are extracted from [36]. In addition, the required time for transcoding the received segments in VTFs to serve the requested quality levels by VRPs, is calculated empirically for each VTF instance type and shown in Table III. The variable τ is set to 6 sec. in the experiments to have time for delivering segments from VRPs to DASH clients. The performance of the proposed models is evaluated in three scenarios. In *scenario I*, we conduct experiments with different values of α_1 and α_2 for both the proposed MILP model and the heuristic algorithm. *Scenario II* evaluates the behavior of streaming the AVC- and SVC-encoded sequence using multicast and unicast approaches. The impact of the homogeneity of the clients' requests on the performance of the proposed approaches is investigated in *scenario III*.

[1] http://www.topology-zoo.org, last access: April 4, 2020.
[2] http://mininet.org, last access: April 4, 2020.
[3] http://www.projectfloodlight.org, last access: April 4, 2020.

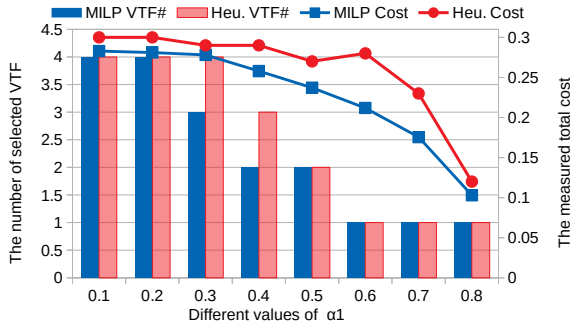

Figure 2: Number of selected VTFs and measured total costs.

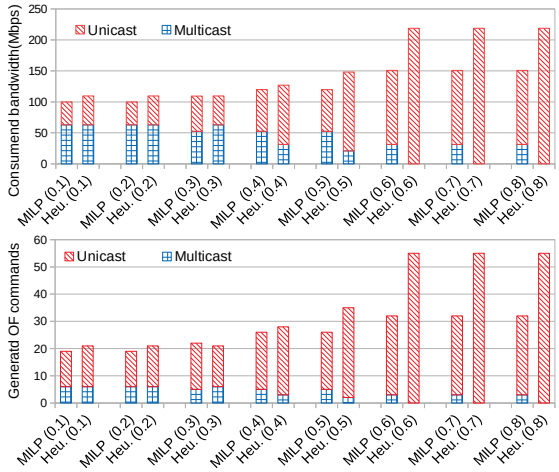

Figure 3: Consumed bandwidth and generated OF commands.

The performance is evaluated in terms of bandwidth utilization and path selection overhead. Due to leveraging SDN, we use the generated OpenFlow (OF) commands as the main criterion to express path selection overhead. Moreover, this parameter implicitly reveals the amount of TCAM [13] usage by the proposed approaches. Simulations were performed on a 64-bit operating system with Windows 10, Intel Core i7-3520M (2.90 GHz) processor and 16 GB of memory.

B. Scenario I

In the first scenario, we conduct evaluations of the proposed MILP model and the heuristic algorithm using the small-scale topology. Note that due to the high time complexity of the MILP model, it is just applicable to the small topology. In this experiment, we investigate the performance of the proposed approaches for different values of α_1, defined as the weight parameter for VTF instance selection (see MILP model (16)), from 0.1 to 0.8, and α_2 is set to $1-\alpha_1$. It is worth mentioning that in each topology, the maximum number of transcoders could be equal to the number of VRPs (refer to constraint 8); thus, in the small-scale topology, we could have up to five VTF instances. As depicted in Fig. 2, both approaches show the same behavior. By setting $\alpha_1 = 0.1$ and $\alpha_2 = 0.9$, we explicitly give more priority to the bandwidth usage that leads to the same number of VTFs selection (four VTFs) by both approaches. For greater values of α_1, the approaches try to minimize the total cost (MILP model (16)) by using fewer VTFs as it is expected (both approaches select only one VTF for $\alpha_1 = 0.8$). We extend our investigation by measuring the amount of bandwidth consumption and the number of generated OF commands for the considered values of α_1. As shown in Fig. 3, by increasing α_1, both the proposed MILP model and the heuristic algorithm consume more bandwidth and generate more OF commands. In more detail, the amount of bandwidth used by the multicast tree declines gradually while the unicast paths show the inverse behavior. This is because by growing α_1 both algorithms attempt to optimize toward fewer VTFs which results in a shorter multicast tree by placing VTFs closer to the source node. For example, by setting $\alpha_1 = 0.8$, the MILP model and the heuristic algorithm select only one VTF. However, due to placing the selected VTF on the source node in the heuristic algorithm, the total bandwidth is consumed by the unicast paths from the source node to the VRPs. For a more detailed comparison, the solutions of both approaches for $\alpha_1 = 0.2$ and $\alpha_1 = 0.4$ are shown in Fig. 4. As illustrated in Fig. 3, the generated OF commands to set up the data paths and the bandwidth utilization are very close for $\alpha_1 \leq 0.4$ in both models. For $\alpha_1 = 0.3$, the MILP model generates more OF commands than the heuristic algorithm due to the use of fewer transcoders in comparison with the heuristic algorithm. However, the bandwidth utilization and the normalized cost for the MILP model are a little less. Although the MILP model produces better solutions in all cases, it suffers from high time complexity and is not applicable in the larger-scale topologies. Note that the heuristic algorithm first finds the multicast tree in a greedy fashion, then by considering the weight parameters determines the optimal number and locations for transcoders. For $\alpha_1 > 0.4$, the heuristic algorithm exhibits more growth in OF command generation and bandwidth utilization in contrast to the MILP model. However, as shown in the next subsection, the heuristic approach has better performance than other approaches studied.

C. Scenario II

We consider four different approaches to compare the performance of the proposed heuristic algorithm: *Multicast-SVC*, *Multicast-AVC*, *Unicast-SVC*, and *Unicast-AVC*. All approaches are implemented in the proposed architecture and the experiments are conducted using the small-, medium-, and large-scale topologies. For the SVC-based methods, we use the SVC-encoded *Tears of Steel* sequence. We employ the PRIM algorithm [37] to build the multicast tree for the multicast approaches. In the SVC-based approaches, when a VRP requests a quality level of a video, it must receive all quality levels up to the requested one. In contrast, in the AVC-based models, each VRP needs to receive just the requested quality level. Thus, although the Multicast-SVC method consumes less bandwidth in all topologies, the Multicast-AVC scheme has the best performance in terms of generated OF commands among these four methods (see Fig. 5(a)). For the unicast approaches, the shortest paths between source and VRPs are determined and the requested qualities are delivered through it. Using AVC encoding generates fewer OF commands for both multicast and unicast cases, while SVC encoding approaches have better performance in terms of bandwidth usage. Finally, we run the proposed heuristic algorithm over the considered topologies with $\alpha_1 = 0.2$. As Fig. 5(a) depicts, our proposed method saves a remarkable amount of bandwidth by transferring only the highest requested quality to an appropriate set of VTFs. Thus, due to creating one multicast tree from the source node to VTFs, it imposes fewer OF commands to the network. Moreover, unlike the traditional IP multicast approaches, it

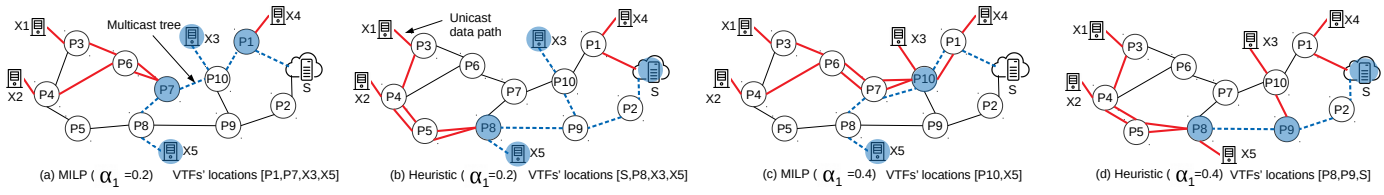

Figure 4: Results of running the proposed MILP model and the heuristic algorithm over the small-scale topology for $\alpha_1 = 0.2$ and 0.4.

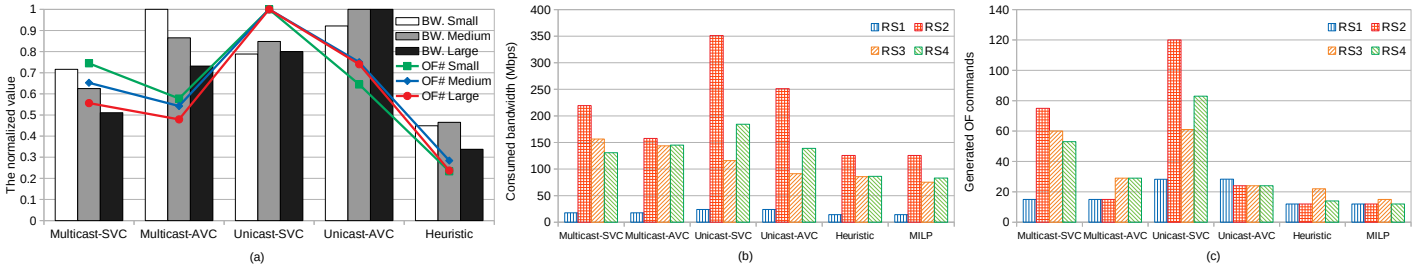

Figure 5: Comparing the performance of the proposed approach with other methods: (a) bandwidth consumption and generated OF commands (Scenario II), (b-c) homogeneity levels of VRPs' requests (Scenario III).

does not need to convert the multicast traffic to unicast. However, the transcoding cost and its processing delay should be taken into account.

D. Scenario III

In this section, we investigate the impact of the homogeneity level among requests of DASH clients on the performance of the approaches under study in terms of bandwidth usage and generated OF commands. To do this, we consider the small-scale topology that consists of 5 VRPs (Fig. 4) with the following request sets (RSs); each 5-tupel denotes the requests issued by the VRPs (X1,X2,X3,X4,X5): RS1: (1,1,1,1,1); RS2: (5,5,5,5,5); RS3: (1,2,3,4,5); RS4: (5,4,3,2,1). RS1 and RS2 are the homogeneous cases where all VRPs request identical quality levels. However, since producing the multicast tree and unicast paths are affected by the VRPs' locations and requests, we define the two heterogeneous request sets R3 and R4 with reverse orders. We note here that for the homogeneous request case, the proposed MILP model and the heuristic algorithm are forced to ignore using VTFs by setting $\alpha_1 = 0$ because the requested quality can be delivered from the source node to the VRPs over an optimal multicast tree. As Fig. 5(b) shows, for homogeneous VRPs' requests (RS1 and RS2) SVC-based approaches, especially the unicast strategies, consume more bandwidth due to sending identical segments over different multicast or unicast paths. It is worth noting that among the SVC-based approaches, as the requested quality level increases, the SVC-based unicast method reveals worse performance in bandwidth usage (compare RS1 and RS2 for the SVC-based methods in Fig. 5(b)). Moreover, the SVC-based methods suffer from high overhead of path establishment (Fig. 5(c)) because of sending all quality layers up to the requested one. For the heterogeneous VRPs' requests (R3 and R4), it is not easy to distinguish which method of these four approaches has the best behavior. This is because, in the heterogeneous environment, the results strictly depend on the location of VRPs and their requested qualities. However, as Fig's. 5(b) and (c) depict, for all cases of VRPs' request sets (RS1–RS4), our proposed MILP and heuristic approaches (with $\alpha_1 = 0.2$ for RS3 and RS4) outperform the other algorithms in both criteria due to combining unicast and multicast techniques as well as leveraging VTFs.

E. Execution Time of Proposed Heuristic Algorithm

As described earlier, the proposed heuristic algorithm includes the two procedures (I) (Alg. 2) and (II) (Alg. 3). Assume $\mathcal{G} = (\mathbb{V}, E)$ is the graph of the considered network topology where $\mathbb{V} = \{\mathcal{S} \cup \mathcal{P} \cup \mathcal{X}\}$ and E is the set of edges connecting nodes in $\mathbb{V}$. In Alg. 2, the main loop iterates $|\mathcal{X}| \times |\mathcal{P}|$ times where in each iteration we need to run the Dijkstra algorithm. Considering the complexity of that algorithm, the overall time complexity of Alg. 2 can be given as $O(|\mathcal{X}||\mathcal{P}|(|E| + |\mathbb{V}|log|\mathbb{V}|))$. The time complexity of Alg. 3 is $O(|\mathcal{P}|)$. We measured the execution time of the proposed heuristic algorithm on the three topologies under study. Table IV shows the execution times of Alg. 2 and Alg. 3 separately. The runtime of Alg. 2 does grow considerably with the network size, but polynomially as expected. In comparison, the execution time of Alg. 3 is very low.

VII. Conclusion

In this paper, we have leveraged the SDN and NFV paradigms to propose an AVC-based real-time video multicast streaming framework. We employ two types of VNFs named VRP and VTF. The VRPs are implemented at the edge of the network, collect the DASH clients' requests, and send them to the SDN controller. The SDN controller determines a multicast tree from an optimal source to an appropriate subset of VTFs hosted in PoPs. Then AVC-encoded segments in the highest requested quality are transferred through the multicast tree. The VTFs transcode the segments to the quality levels requested by the VRPs. Finally, the obtained quality

Table IV: **Execution time of proposed heuristic algorithm**.

Topology	Alg. 2	Alg. 3
Small-scale topology	3ms	0.8ms
Medium-scale topology	82ms	3ms
Large-scale topology	1210ms	17ms

levels are transferred to the VRPs in a unicast fashion. To address time complexity of the proposed MILP model, we introduce a heuristic algorithm. The proposed approaches are evaluated using MiniNet and Floodlight and compared with other unicast and multicast approaches. The performance of the approaches is measured in terms of bandwidth usage and network path selection effort. The proposed heuristic method shows promising results: it is close to the MILP model and results in a significant reduction in bandwidth usage and path selection overhead in various scenarios investigated.

The transfer of segments between VTFs and modifying the initial multicast tree instead of re-creating it are interesting items for future work. Considering and reducing the end-to-end streaming delay are open issues as well.

ACKNOWLEDGMENT

This research has been supported in part by the *Christian Doppler Laboratory ATHENA* (https://athena.itec.aau.at/) and by the *5G Playground Carinthia* (https://5gplayground.at/).

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022," *White Paper*, February 2019.

[2] R. Malli, X. Zhang, and C. Qiao, "Benefits of multicasting in all-optical networks," in *All-Optical Networking: Architecture, Control, and Management Issues*, vol. 3531. SPIE, 1998, pp. 209–220.

[3] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3 (RFC 3376)," 2002.

[4] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C.-G. Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (PIM-SM): Protocol specification (RFC 2362)," 1998.

[5] A. Striegel and G. Manimaran, "A survey of QoS multicasting issues," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 82–87, 2002.

[6] J. Liebeherr and M. Nahas, "Application-layer multicast with Delaunay triangulations," in *IEEE Global Telecommunications Conference (GLOBECOM'01)*, vol. 3. IEEE, 2001, pp. 1651–1655.

[7] Y. Cui, B. Li, and K. Nahrstedt, "oStream: Asynchronous streaming multicast in application-layer overlay networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 91–106, 2004.

[8] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A hybrid tree/mesh overlay for application-layer live video multicast," in *27th Int'l. Conf. on Distributed Computing Systems (ICDCS'07)*. IEEE, 2007.

[9] M. Firdhous, "Multicasting over overlay networks – a critical review," *arXiv preprint arXiv:1211.2026*, 2012.

[10] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *6th Int'l. Conf. on Communication Systems and Networks (COMSNETS'14)*. IEEE, 2014, pp. 1–8.

[11] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing algorithms for network function virtualization enabled multicast topology on SDN," *IEEE Transactions on Network and Service Management*, vol. 12, no. 4, pp. 580–594, 2015.

[12] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *IEEE Conf. on Computer Communications (INFOCOM'15)*. IEEE, 2015, pp. 181–189.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[14] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[15] M. Grafl, C. Timmerer, H. Hellwagner, G. Xilouris, G. Gardikis, D. Renzi, S. Battista, E. Borcoci, and D. Negru, "Scalable media coding enabling content-aware networking," *IEEE MultiMedia*, vol. 20, no. 2, pp. 30–41, 2012.

[16] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "QoE-assured 4K HTTP live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, 2018.

[17] F. K. Hwang and D. S. Richards, "Steiner tree problems," *Networks*, vol. 22, no. 1, pp. 55–89, 1992.

[18] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[19] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 4, pp. 117–130, 1996.

[20] M. Fesci-Sayit, E. T. Tunali, and A. M. Tekalp, "Bandwidth-aware multiple multicast tree formation for P2P scalable video streaming using hierarchical clusters," in *16th IEEE Int'l. Conf. on Image Processing (ICIP'09)*. IEEE, 2009, pp. 945–948.

[21] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE Conf. on Computer Communications (INFOCOM'13)*. IEEE, 2013, pp. 2211–2219.

[22] L.-H. Huang, H.-C. Hsu, S.-H. Shen, D.-N. Yang, and W.-T. Chen, "Multicast traffic engineering for software-defined networks," in *IEEE Conf. on Computer Communications (INFOCOM'16)*. IEEE, 2016, pp. 1–9.

[23] J. Yang, E. Yang, Y. Ran, and S. Chen, "SDM^2 Cast An OpenFlow-Based, Software-Defined Scalable Multimedia Multicast Streaming Framework," *IEEE Internet Computing*, vol. 19, no. 4, pp. 36–44, 2015.

[24] T.-H. Chi, C.-H. Lin, J.-J. Kuo, and W.-T. Chen, "Live Video Multicast for Dynamic Users via Segment Routing in 5G Networks," in *IEEE Global Telecommunications Conference (GLOBECOM'18)*. IEEE, 2018, pp. 1–7.

[25] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *Journal of Lightwave Technology*, vol. 34, no. 14, pp. 3330–3341, 2016.

[26] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Approximation and online algorithms for NFV-enabled multicasting in SDNs," in *37th Int'l. Conf. on Distributed Computing Systems (ICDCS'17)*. IEEE, 2017, pp. 625–634.

[27] O. Alhussein, P. T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, "Joint VNF placement and multicast traffic routing in 5G core networks," in *IEEE Global Telecommunications Conference (GLOBECOM'18)*. IEEE, 2018, pp. 1–6.

[28] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Efficient NFV-Enabled Multicasting in SDNs," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2052–2070, 2019.

[29] M. Zhao, B. Jia, M. Wu, H. Yu, and Y. Xu, "Software defined network-enabled multicast for multi-party video conferencing systems," in *IEEE Int'l. Conf. on Communications (ICC'14)*. IEEE, 2014, pp. 1729–1735.

[30] N. Xue, X. Chen, L. Gong, S. Li, D. Hu, and Z. Zhu, "Demonstration of OpenFlow-controlled network orchestration for adaptive SVC video manycast," *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1617–1629, 2015.

[31] E. Yang, Y. Ran, S. Chen, and J. Yang, "A multicast architecture of SVC streaming over OpenFlow networks," in *IEEE Global Telecommunications Conference (GLOBECOM'14)*. IEEE, 2014, pp. 1323–1328.

[32] S.-H. Shen, "Efficient SVC Multicast Streaming for Video Conferencing with SDN Control," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 403–416, 2019.

[33] J. Yang, E. Yang, Y. Ran, Y. Bi, and J. Wang, "Controllable multicast for adaptive scalable video streaming in software-defined networks," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1260–1274, 2017.

[34] A. Zabrovskiy, C. Feldmann, and C. Timmerer, "Multi-codec DASH dataset," in *9th ACM Multimedia Systems Conference (MMSys'18)*. ACM, 2018, pp. 438–443.

[35] J. Reichel, H. Schwarz, and M. Wien, "Joint scalable video model 11 (JSVM 11)," *Joint Video Team, Doc. JVT-X202*, p. 23, 2007.

[36] F. Tashtarian, M. F. Zhani, B. Fatemipour, and D. Yazdani, "CoDeC: a cost-effective and delay-aware SFC deployment," *IEEE Transactions on Network and Service Management*, 2019.

[37] K. Rosen, *Discrete Mathematics and Its Applications (7th ed.)*. McGraw-Hill Science, 2011.