applied sciences

Article

Multipath Dynamic Adaptive Streaming over HTTP Using Scalable Video Coding in Software Defined Networking

Ali Gohar [1,2] and Sanghwan Lee [1,*]

[1] Faculty of Science and Technology, University of Stavanger, 4021 Stavanger, Norway; ali.gohar@uis.no
[2] School of Computer Science, Kookmin University, Seoul 02707, Korea;
* Correspondence: sanghwan@kookmin.ac.kr

Received: 21 September 2020 ; Accepted: 22 October 2020; Published: 30 October 2020

Abstract: Dynamic Adaptive Streaming over HTTP (DASH) offers adaptive and dynamic multimedia streaming solutions to heterogeneous end systems. However, it still faces many challenges in determining an appropriate rate adaptation technique to provide the best quality of experience (QoE) to the end systems. Most of the suggested approaches rely on servers or client-side heuristics to improve multimedia streaming QoE. Moreover, using evolving technologies such as Software Defined Networking (SDN) that provide a network overview, combined with Multipath Transmission Control Protocol (MPTCP), can enhance the QoE of streaming multimedia media based on scalable video coding (SVC). Therefore, we enhance our previous work and propose a Dynamic Multi Path Finder (DMPF) scheduler that determines optimal techniques to enhance QoE. DMPF scheduler is a part of the DMPF Scheduler Module (DSM) which runs as an application over the SDN controller. The DMPF scheduler accommodates maximum client requests while providing the basic representation of the media requested. We evaluate our implementation on real network topology and explore how SVC layers should be transferred over network topology. We also test the scheduler for network bandwidth usage. Through extensive simulations, we show clear trade-offs between the number of accommodated requests and the quality of the streaming. We conclude that it is better to schedule the layers of a request into the same path as much as possible than into multiple paths. Furthermore, these result would help service providers optimize their services.

Keywords: algorithm design and analysis; Dynamic Adaptive Streaming over HTTP; MPEG-DASH standard; scalable video coding (SVC) video streaming

1. Introduction

The continuous and rapid increase of digital multimedia streaming is estimated to comprise 82% of Internet traffic by 2021 [1]. The proliferation of multimedia streaming devices (e.g., smartphones, laptops, tablets, gaming consoles, etc.) with wide accessibility to wireless networks plays a pivotal role in characterizing the increased demand for multimedia streaming over the Internet. Moreover, the bandwidth capacity for universal Internet access has not grown as the host's computing power. The Hypertext Transfer Protocol (HTTP) is used in multimedia streaming without underlying network support via the Internet Protocol (IP) [2]. HTTP multimedia streaming provides device fluidity, cache-friendliness, data-agnostic web servers and the ability to avoid network translation and firewall problems [3]. However, HTTP is based on the Transmission Control Protocol (TCP). Therefore, having an overhead of about twice the media bit rate compared to the Real-Time Transport Protocol (RTP) and the User Datagram Protocol (UDP). This leads to delays in delivering multimedia content and the under-utilization of communication network resources [4].

Managing multimedia streaming for heterogeneous devices under unreliable network environments is challenging. Initially, a progressive download technique allowed HTTP applications to download a complete monolithic multimedia file to the end host using TCP. This results in all clients receiving the same video despite variations in network bandwidth and their capabilities. An improvement in progressive downloads resulted in adaptive streaming. In adaptive streaming, the multimedia file is divided into smaller duration chunks, known as segments. These segments are encoded with different qualities and transferred to the end host based on their available resources. Access to these media segments enables HTTP Adaptive Streaming (HAS). In HAS, adaptation is performed at the receiver end. A receiver estimates its network path conditions and capabilities and requests a different representation of the similar media encoded at different encoding rates appropriate at a certain point in time. A single layer video codec such as Advanced Video Coding (AVC) or multi-layer codec, such as Scalable Video Coding (SVC) can generate multiple media representations [5,6].

HAS represents most internet video traffic based on commercial solutions. This includes HTTP Live Streaming from Apple [7], smooth streaming from Microsoft [8], HD from Akamai [9], and HTTP Dynamic Streaming from Adobe [10]. Apart from proprietary alternatives, the Moving Picture Experts Group (MPEG) in collaboration with 3rd Generation Partnership Project (3GPP) contributed to the standardization of streaming solutions for the HTTP streaming of MPEG media. This resulted in Dynamic Adaptive Streaming over HTTP (DASH) or MPEG-DASH [11]. Unlike proprietary solutions DASH provides an open specification for adaptive streaming over HTTP and leaves the implementation of the adaptation logic to third parties. The Software Defined Networking (SDN) paradigm has revolutionized the way we looked at networks by decoupling control and data planes. A controller centralizes and manages network intelligence in a SDN. This allows external applications to program the network and simplify network management by developing, implementing, and testing new algorithms. We already know of the fact that routing plays an important role in enhancing the network resource utilization. Multiple paths between two systems result in reliable and increased data throughput. SDN can provide the routing flexibility needed by the network, while Multipath TCP (MPTCP) can provide multiple paths between two endpoints. MPTCP is an extension of TCP that, enables the use of different route paths between two nodes in a network. However, transferring packets across different route pathways can lead to a considerable reorganization of packages, leading to decreased throughput and the significant usage of CPU resources [12]. However, if MPTCP is utilized properly, it can enhance throughput, fairness, and robustness by introducing reliability and seamless fail-over as well as increased network capacity [13].

In our previous work [14], we propose a Dynamic Multi Path Finder (DMPF) algorithm that leverages the properties of multiple paths available in the network to transfer each client request SVC layers over different paths. Now we propose a DMPF scheduler to maximize QoE and network bandwidth of DASH streaming using SVC in an SDN environment. A SDN controller runs the DMPF Scheduler Application module to provide the basic QoE for the clients. The DMPF scheduler determines the optimal flow paths for customer requests through the use of a holistic network view provided by the SDN controller. In brief, we can describe the main contributions of the present study:

- Describe SDN-based scheduler application for DASH using SVC
- Consider QoE metrics and bandwidth utilization in the process of designing the architecture of the proposed scheduler
- Propose a mathematical model which determines the optimal data paths for delivering the requested video files and the quality adaptation.
- Implement the proposed architecture and evaluate its performance in real network topology.

The rest of this paper is organized as follows—a brief overview of DASH and SVC is provided in Section 2. We present related work that explores the foundation for multimedia steaming based on SVC in an SDN environment using multiple connections, in Section 3. We introduce the Dynamic Multi Path Finder (DMPF) Module which contains DMPF Scheduler and its processing and forwarding modes in Section 4. To enhance QoE, we present a mathematical formulation of the problem in Section 5. Detailed analysis of the performance of the proposed algorithms in given in Section 6. We finally conclude in Section 7.

2. Background

In this section, we give a brief overview of DASH and SVC.

2.1. Dynamic Adaptive Streaming over HTTP

Dynamic adaptive streaming over HTTP (DASH) is an MPEG standard defining a multimedia format and description. DASH contains media files divided into small chunks of 2 to 10 s called segments. Each media file segment is encoded into multiple versions of bit-rate streams or quality levels. The quality level comprises frame rate (temporal), bit-rate (signal-to-noise) and resolution (spatial). Details related to different quality levels of media are stored in XML manifest, known as Media Presentation Description (MPD). One or more *Periods* are defined in each MPD. These *Periods* contain various parts of media, such as video with different viewing angles, various codes used, audio for different languages, captions, and subtitles. These media parts have some features that do not change for one *Period*, (such as sound channels, bit rate, and frame rate) and are arranged in *Adaptation Sets*. One or more *Adaptation Sets* are included in each *Period*. This allows the clustering of logically similar multimedia parts. The *Adaptation Set* comprises a set of representations that contains interchangeable versions of each media content, such as different resolutions, bit-rates. For example, the *Adaptation Set* may contain media parts encoded with the same codec, resolution, language, and audio channel formats (e.g., stereo or 5.1 channel). The *Adaptation Set* enables the client to eliminate a range of multimedia parts that do not meet their requirements. Although a single representation is sufficient for a playable stream, multiple *Representation Sets* enable the client to adapt the media stream to their current requirements for network bandwidth. Over time, the client can adapt to the resolutions, bit-rates, codecs, and so forth, available in a *Period*. Figure 1 shows the detailed logical structure of MPD. The DASH client first obtains the MPD through HTTP standard complaint GET requests. By parsing the information provided by the MPD, the DASH client chooses the appropriate encoded alternative based on network conditions and starts content streaming by selecting segments. MPEG-DASH specification for server defines only MPD and segment formats, while on the client-side it defines the parsers. The transmission of MPD, media encoding formats comprising segments, customer behavior for obtaining the segments, adapting heuristics, and playing content are beyond the MPEG-DASH scope. The size of the segment and the selected encoding technique define the granularity of the streaming adaptation.

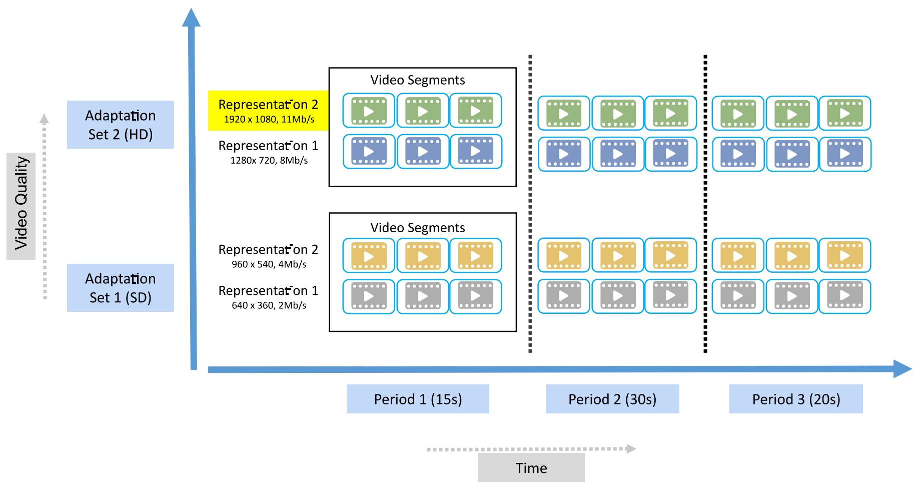

Figure 1. Structure of Media Presentation Description (MPD).

2.2. Scalable Video Coding (SVC)

Sending a raw quality video over the Internet consumes enormous network resources. Various techniques of video compression were considered compressing raw video to a smaller size without compromising quality. In 2003, AVC was introduced, which compressed/encoded the video with a specific quality (i.e., resolution, frame rate and signal-to-noise ratio). In contrast, SVC is an AVC extension that can encode the video in separate layers. If combined, these layers can improve the overall quality of the video. SVC comprises two layer types: base layer (BL) and enhancement layer(s) (EL). The SVC media has one base layer and multiple enhancement layers. Figure 2 shows such segmented layers of the "Representation 2" in Figure 1. The base layer provides the basic quality of the video. Adding enhancement layers to the base layer can increase the quality of the video. All lower quality level enhancement layers are needed to decode a higher level EL. SVC enables graceful video quality degradation compared to AVC by coping with congestion in real time by adding or removing layers to meet the network capacities of all devices. It also provides the opportunity to serve many users with heterogeneous devices with an improved QoE [15]. AVC generates multiple versions with different qualities of the same media that are transferred simultaneously over a network. This is also called simulcasting, as illustrated in Figure 3. Simulcast forwards several versions of media with higher qualities to higher bandwidth subscribers and lower qualities to lower bandwidth ones. Using AVC with DASH (AVC-DASH) to stream multimedia to heterogeneous devices, each segment will be encoded with different qualities. The multiple storage of these media segments increase the requirements for storage and network resources use. In AVC-DASH takes decisions are taken at the end of the segment. Since SVC media comprises layers, it has only one version of each segment, reliving storage strain on servers. DASH coupled with SVC (SVC-DASH) enables flexible segment selections as compared to DASH only [16] The granularity of SVC-DASH quality control extends horizontally over time and vertically over layers.

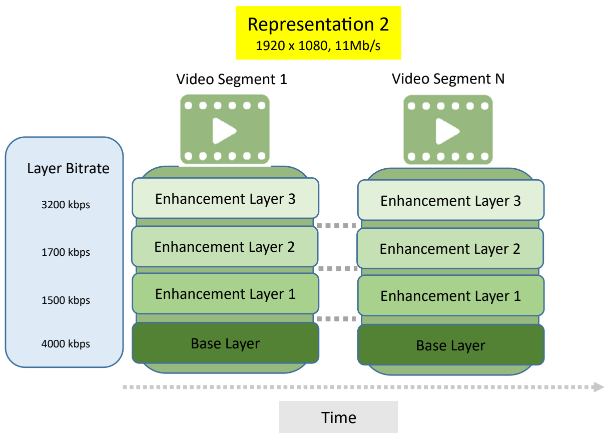

Figure 2. Segmented Layers of Scalable Video Coding (SVC).

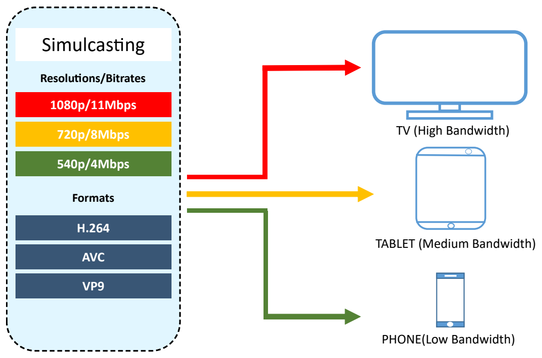

Figure 3. Simulcasting.

3. Related Work

DASH clients locally implement adaptation techniques without being aware of the complete network topology. There are three main solutions for optimum adaptation, client-based only, client-based assisted by a network element, and network-based. Client performs adaptation by looking at its local parameters which include its buffer size and network capacity in client based adaption. In the second approach client may be supported by network elements, such as proxy servers. Central network element is utilized in network based adaptation. In this section we describe the related work briefly and motivate the advantages of SVC based multimedia streaming using multiple connections in an SDN environment.

3.1. Advantages of SVC over AVC

A few studies have focused on the comparison of AVC and SVC based streaming. Schwarz et al. [6] show the bit rate for spatial and SNR scalability increases by 10% for SVC based streaming. However, the findings of Grafl et al. [17] show that replacing AVC encoded videos with SVC encoded videos saves up to 50% of the storage space. Another study by Sanchez et al. [18] concluded that SVC results in improved caching efficiency for HTTP streaming while storing AVC representations leads to spoilage of the cache capacity. In addition, SVC-DASH needs less buffering compared to AVC-DASH and improves the quality of Experience (QoE) for client [15]. Famaey et al. [19] use heuristics to evaluate performance of SVC and AVC based streaming using NS3. They concluded that SVC shows decreased performance in restricted bandwidths but performs well when using a single connection in a variable bandwidth environment. Grafl et al. [17] give a detailed objective assessment of SVC with DASH in mobile environments with various layer setups. They suggest creating a separate SVC stream for each resolution. This provides a nice balance between the advantages of SVC and its overall coding overhead. Their important contribution was to validate the bit rate recommendations derived from industry solutions. Andelin et al. [20] analytically investigated the SVC layer segment selection policy. It is concluded that policy with diagonal selection is optimal that balances quality variation and quality score for variable rate while the vertical policy is optimal under fixed bandwidth.

3.2. Using Multiple Connections for Streaming

Some works have investigated multiple connections in HTTP streaming. Bouten et al. [21] quantitatively investigate the SVC based HAS live streaming for Live TV in high delay networks. For a client with limited buffer capacity, they propose a cursor-based SVC client heuristic. A fixed number of parallel connections and pipeline download of segment layers are suggested to overcome end-to-end delay problems. Tullimas et al. [22] implement a MultiTCP receiver driven system with multiple TCP connections in an application layer. The system provides resilience against short-term bandwidth fluctuations. Their system in congested periods can control and achieve the desired sending rate. Han et al. [23] propose a multi-path DASH (MP-DASH) streaming framework taking into account the preferences of the client network interface by strategically scheduling a chunk delivery to satisfy client preferences, such as preferring WiFi over the cellular connection. Instead of improving the quality of video streaming with MPTCP, it reduces overall streaming costs. Bradai et al. [24] proposes a model to improve bandwidth allocation and incentive mechanisms for layered video streaming in peer-to-peer (P2P) networks using an auction game model. The purpose is to make sure that neighbouring peers first have a minimum quality level stream, and then the remaining bandwidth is allocated to the higher layers. This optimization results in efficient use of the bandwidth and improved video quality.

3.3. SDN Assisted Streaming Using Multiple Connections

In some works, MPTCP in SDN is considered. References [25,26] focus on determining the appropriate number of SVC layers for transferring across selected streaming paths. The approach of [25] is based on priorities that first determines the shortest route N and then sends the video layer using one of the selected routes. The approach in References [26], on the other hand, uses the optimal paths rather than the shortest. In addition, since they focus on the UDP protocol, References [25,26] do not target DASH video streaming. Cetinkaya et al. [27] propose an architecture based on SDN to improve scalable video streaming efficiency using DASH. They measure their proposed system against QoE parameters and conclude that for each QoE parameter, traditional shortest-path routing provides better performance. Determining routing and providing quality adaptation is considered by few works. Cetinkaya et al. [28] develop an optimization model for maximizing video quality. In order to select optimal paths for different SVC layers over SDN, they consider the current segment bit rate, available network bandwidth, path length and competing flows. Herguner et al. [29], and Barakabitze et al. [30], propose to use routing strategy in an SDN based networks for the routing of the MPTCP sub-flows to provide QoS/QoE-guaranteed services to DASH clients. Gohar et al. [14], propose a Dynamic Multi Path Finder (DMPF) algorithm that leverages the properties of multiple paths available in the network to transfer each client request SVC layers over different paths. They accommodates maximum requests by DASH clients while utilizing the network bandwidth efficiently by dynamically adapting to the varying connection characteristics.

3.4. DMPF Algorithm

Several paths between the client and the server are available for communication in a network topology. Figure 4 shows the schematic communication topology using multi-path TCP in an SDN environment. By default, all layers are transmitted when a client requests SVC layers using a single path between the client and the server. The reliability of the connection between the client and the server reduces if multiple SVC layers are transmitted on a single path. This results in overloading a single path and under-utilization of multiple paths that are available to transmit multiple SVC layers. SVC is composed of layers, and each SVC layer can be sent over different routes in a network. SVC based streaming requires a base layer to view the lowest video representation. Sending each SVC layer on different paths helps to distribute the network bandwidth load, improved connection reliability and graceful degradation in view quality. However, it also increases the number of paths found, path length, bandwidth and computing resources. It is difficult to find an optimal mechanism that can accommodate the maximum number of client requests with the least use of bandwidth. We propose a Dynamic Multi Path Finder (DMPF) algorithm in our previous work [14] to forward all requested SVC layers of each client on a different path in a network.

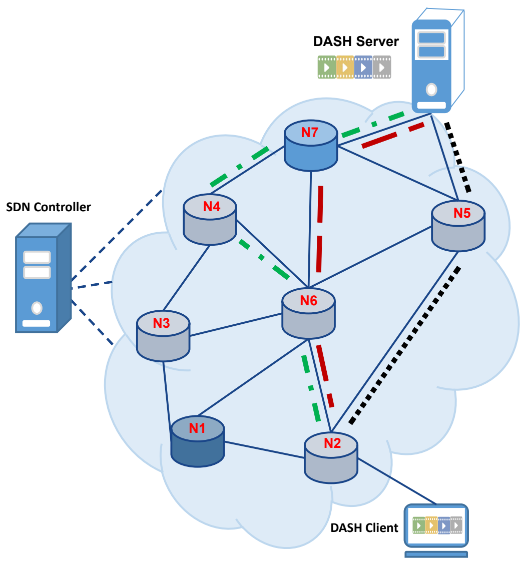

Figure 4. Using Multi-path Transmission Control Protocol (TCP) in a Software Defined Networking (SDN) environment.

4. System Model

Using multi-path TCP in an SDN environment, this section provides for the simplified operation of the SVC integrated into DASH. First, we provide a brief description of the DMPF Scheduler Module and its components in the context of SDN. This is followed by a detailed description of the working of the DMPF Scheduler Module, its processing and forwarding modes with an example.

4.1. DMPF Scheduler Module

DMPF Scheduler Module (DSM) runs as an application over the SDN controller. This allows the DASH clients to stream adaptively over HTTP. In this research study, we assume that DASH client requests are provided by local HTTP media servers. In Figure 5, we provide a brief description of the operation of the proposed approach and also illustrate the three-layer SDN architecture model containing DSM. DSM considers the received requests and available network resources, for example, link bandwidth, to determine an optimal solution. The DSM consists of three components: the request analyzer, resource monitor, and flow setup component.

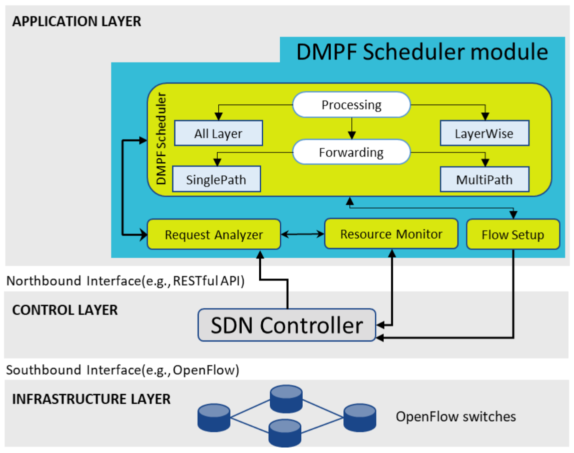

Figure 5. Dynamic Multi Path Finder (DMPF) Scheduler Module in SDN three layered architecture.

4.1.1. Request Analyzer

SDN controller forwards the receiving DASH client request to the Request Analyzer Component (RAC). RAC collects the received requests from the DASH clients and performs some pre-processing on the collected requests to prepare them as an input parameter for the DMPF scheduler. RAC gives important meta-data on the video files in DASH servers. The meta-data includes the server list, file segments stored, layers, and their properties, such as each layer size and bit rate. In our work, we consider only the bit rate of each SVC layer.

4.1.2. Resource Monitor

Let's define graph $G = (V, E)$ before describing the Resource Monitor Component (RMC). Where V is the set of DASH clients, and HTTP-media servers; and set E represents the edges of G. RMC uses RESTful APIs to obtain the available bandwidth of the links from the SDN controller. The measured values are stored in a two-dimensional array B, in which B_{ij} indicates the available bandwidth between i and $j \in V$. The connectivity among DASH clients and HTTP-media servers can easily be inferred from B. In fact, this component constructs graph $G = (V, E)$ and the available bandwidth among the nodes in V.

4.2. DMPF Scheduler

The DMPF scheduler (DS) is at the core of the DMPF scheduler Module (DSM) because it is responsible to run algorithms. DS collects the buffered requests from the Request Analyzer component

(RAC) and current network status from the Resource Monitor Component (RMC). The DS selects a mode for processing and forwarding of requests. When processing requests, DS can process requests in *allLayer* or *layerWise* mode. The DS can forward requests on a *singlePath* or *multiPath* in the request forwarding mode. First, we explain the DS request processing modes. During *allLayer* request processing mode, the scheduler accommodates all SVC layers of the current client request and then accommodates all SVC layers of the next client request. In *layerWise* request processing mode, the scheduler incrementally accommodates the SVC layers of each client request. We explain the DS request processing modes, *allLayer*, and *layerWise* with the help of Figure 6.

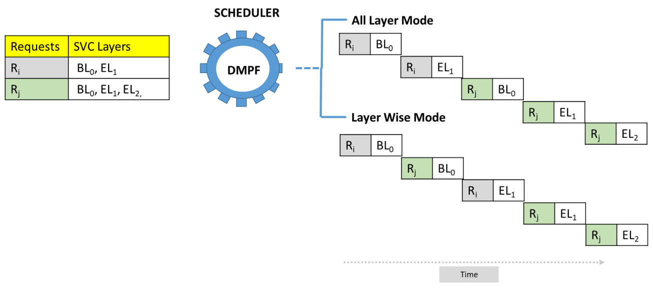

Figure 6. DMPF scheduler request processing modes.

4.2.1. DMPF Scheduler Processing Modes

In Figure 6, we have two client requests R_i and R_j. Each client request demands multiple SVC layers. The clients decide the requirement of multiple layers after looking at their playback and network bandwidth capacity. The client R_i requests two SVC layers BL_0 and EL_1, while client R_j requests three SVC layers, BL_0, EL_1, and EL_2. If the DS is configured to be in *allLayer* processing mode, then it takes the request R_i and accommodates all of its layers and then processes the next request R_j. If the DS is configured to be in *layerWise* request processing mode, then the DS processes the base layer (BL_0) of R_i and R_j first. Next, the DS processes the EL_1 of R_i and R_j. Since R_i does not contain EL_2 the DS in *layerWise* processing mode processes request R_j EL_2.

4.2.2. DMPF Scheduler Forwarding Modes

Now we are going to describe the DS request forwarding modes. In the request forwarding mode, the DS can forward all the SVC layers of the request on a *singlePath* or each SVC layer on a *multiPath*. If the DS is configured to be in the *singlePath* mode, all the SVC layers of the request will be transferred to the same path. If the DS is configured to be in *multiPath* mode, the DS will send all the requested SVC layers of the client to different paths. When sending a request on a different path in *multiPath* mode, we assume that the path is at least one edge different. The forwarding modes of the DS are described in detail in Figure 4. Figure 4 shows a network topology where one node acts as a DASH client and another acts as a DASH server. The SDN controller has a view of the entire topology of the network. There is a possibility of multiple paths that exist between the DASH client and the DASH server. We use Multipath TCP (MPTCP) to use these multiple paths and send SVC layers on different paths. MPTCP manages multiple sub-flows which are transferred via multiple paths between the DASH client and the DASH server. Besides, if DASH client requests three SVC layers, that is, one BL and two EL(s) of a video, the controller executes the proposed Dynamic Multi Path Finder (DMPF) algorithm. DMPF algorithm finds three paths and allows the DASH client to create three multi-path TCP sub-flows with the DASH server. Figure 4 depicts the different paths between the DASH client and the DASH server as black dotted, red dashed and green dotted-dashed lines. However, the baseline algorithm finds only one path and transmits all SVC layers on it. Baseline and DMPF algorithm use the Dijkstra's algorithm to find paths between the DASH client and the DASH server.

5. Problem Formulation: Mathematical Model and Algorithm

Now, we provide a simple analysis of the lowerbound of the required bandwidth on the links adjacent to a server. For simplicity, we first define a few notations.

- Let $S = \{0, 1, 2, \ldots, s-1\}$ be the set of s DASH Servers.
- Let E be the set of edges in the network.
- Let $R = \{0, 1, 2, \ldots, r-1\}$ be the set of r requests.
- Let $L = \{0, 1, 2, \ldots, l-1\}$ be the set of l layers. 0 means the base layer and 1 means the enhancemenet layer 1, and so forth.
- Let Q be the set of requests to the servers S.
- Let Q_i be the set of requests to a server i
- Let Q_e be the set of requests going through an edge e
- Let C_e be the capacity of a link e
- Let $f_{q,k}$ be the required bandwidth of a request q for the layer k. For example, if a request 0 needs layers only up to 2, then the bandwidth of the layers higher than 2 is 0 such as $f_{0,3} = 0$.
- Let $b_{q,k}$ be the flag having value of 0 or 1. When the layer k of a request q is accepted, $b_{q,k} = 1$. Otherwise, $b_{q,k} = 0$. It should be noted that for any k, if $b_{q,k} = 0$, then $b_{q,k+1} = 0$.
- Let d_i be the degree of a server i.

With this notation, we can formulate the objective of this problem as follows

$$\max \textstyle\sum_{q \in Q} \sum_{k \in L} b_{q,k} \tag{1}$$
$$\text{s.t} \quad \textstyle\sum_{q \in Q_e} \sum_{k \in L} f_{q,k} b_{q,k} \leq C_e, \qquad e \in E \tag{2}$$
$$b_{q,k+1} = 0, k \in \{k | b_{q,k} = 0\} \quad q \in Q, k \in L. \tag{3}$$

Basically, we want to maximize the number of layers accepted to the network with the link capacity constraint (2) and the SVC layer constraint (3). Thus, for the given set of queries Q, we need to decide which queries to accept upto which layers.

Intuitively, the bottleneck links are the links adjacent to the servers. We can easily compute the amount of bandwith needed for a server i.

$$B_i = \sum_{q \in Q_i} \sum_{k \in L} f_{q,k} b_{q,k}. \tag{4}$$

The worst case is that all the requests in Q_i need to use a specific link of a server i.Then, the required bandwith is B_i, which is likely to be higher than the bandwidth of the link. The best case is when the traffic is distributed evenly to all the links of the server. Then, each link has the following amount of traffic.

$$Avg(B_i) = \frac{\sum_{q \in Q_i} \sum_{k \in L} f_{q,k} b_{q,k}}{d_i}. \tag{5}$$

Equation (5) can be written in a different way as follows.

$$Avg(B_i) = \frac{\sum_{k \in L} \sum_{q \in Q_i} f_{q,k} b_{q,k}}{d_i}. \tag{6}$$

Basically, Equation (5) roughly represents the processing mode of allLayer. For a request q, we accept all the layers of the request q. On the other hand Equation (6) represents the processing mode of layerWise. For a layer k, we accept the layer k from all the requests.

Actually, the total required bandwith can be approximated as $f \times r$, where f is the average required bandwidth of a request and r is the number of requests. This is simply the multiplication of the average bandwith and the number of requests. A lower bound of the required bandwidth to the links adjacent to the servers can be computed as follows. We assume that the requests are distributed evenly to each of s servers. Thus each server needs $\frac{f \times r}{s}$. Finally, each server has d links. In a best case, the traffic is evenly distributed to each link. Thus, we have

$$\text{Average Bandwidth} = \frac{f \times r}{s \times d} \tag{7}$$

The average bandwidth given in (7) is can be considered as the lower bound of the required bandwidth. In practice, some links may need higher bandwidth and other links may need lower bandwidth. However, this can be a good indicator to determine how many requests can be accepted. The detailed algorithm is described in Algorithm 1 by using the notations in Table. 1.

Table 1. Notations.

Notation	Description
V	The set of OF switches, DASH clients, and HTTP-media servers
D	The set of DASH clients
S	The set of HTTP-media servers
m_c	The desirable maximum video quality determined by client $c \in D$
B, b_{ij}	B is a two-dimensional array where b_{ij} shows the available bandwidth between i and $j \in V$
N	Total number of nodes in a topology
src	A dash client D node in Graph G
$dest$	A server node S in Graph G
$P_{src \to dest}$	Path from src to $dest$

Algorithm 1 *Dynamic Multipath Finder (DMPF) Core.*

```
1:  Input: $G, src, dest, N$
2:  Output: $\hat{S}$
3:  $N$: Total number of requested layers
4:  $\hat{S} \to \phi$ // To store $N$ paths for $N$ layers from $src$ to $dest$
5:  $P_{src \to dest} = 0$ // To save a path from $src$ to $dest$
6:  while $P_{src \to dest}$ == NULL do
7:      $P_{src \to dest} = Dijkstra(src, dest, G)$
8:      if $P_{src \to dest}$ FOUND then
9:          Check Shortest Path Edges Capacity
10:         if $P_{src \to dest}$ edge(s) are $overUtilized$ then
11:             FIND $overUtilized$ edge of $P_{src \to dest}$
12:             REMOVE $overUtilized$ edge(s) of $P_{src \to dest}$ from $G$
13:             $P_{src \to dest}$ == NULL
14:         else
15:             $\hat{S_{(l)}} \leftarrow P_{src \to dest}$
16:             REMOVE edges of $P_{src \to dest}$ from $G$
17:             UPDATE path counter value $l++$
18:             UPDATE $G$
19:             LOG Request Accepted
20:     else
21:         LOG Request Rejected
22: RETURN $P_{src \to dest}$
```

6. Evaluation

We analyze the performance of the proposed algorithms in this section by examining different parameters that is, number of different layers accepted/rejected, bandwidth utilization, and so forth.

We create a network model of client and servers for evaluation. The model includes a topology of a hundred (100) nodes with two hundred (200) edges. The average topology degree is four (4). The topology is generated using BRITE topology generator [31]. In the evaluation model, ten (10) nodes act as a server, and remaining nodes act as a client. The bandwidth for the client is ten megabits per second (Mbps) and for the servers, its hundred Mbps. All client media requests are randomly generated which includes the selection of video resolution and its layers. We use four SVC layers with different resolutions in SVC-DASH based streaming. The SVC layers include one base layer (BL_0) and three enhancement layers (EL_1, EL_2, EL_3). The bit-rate recommendations for SVC layers with different resolutions from 1920 × 1080 to 640 × 360, are shown in Table 2, which are based on Reference [17]. For extensive analysis, five typologies are generated for the simulations.

Table 2. Scalable Video Coding (SVC) layers bit-rate recommendations derived from industry solutions.

SVC Layers Bitrate [kbps]				
Resolution	**Base Layer**	**EL_1**	**EL_2**	**EL_3**
1920 × 1080	4000	1500	1700	3200
1280 × 720	1500	1250	2050	3000
960 × 540	1200	775	725	800
640 × 360	600	390	510	575

We evaluate our algorithms with the combination of DS modes that is, processing and forwarding. The two DS modes outcome can be analyzed under four categories as follows:

- allLayers-multiPath: Send all the client requested SVC layers on different paths.
- allLayers-singlePath: Send all the client requested SVC layers on a single path.
- layerWise-multiPath: Send each client requested SVC layers on a different path.
- layerWise-singlePath: Send each client requested SVC layers on a single path.

Now we compare the number of layers accepted by DS with the combination of processing and forwarding modes. It is evident from Figure 7, that scheduler in *allLayer* processing mode accommodates maximum number of layers as compared to scheduler in *layerWise* processing mode. Moreover, it also reveals that the number of requests accommodated in *singlePath* forwarding mode is higher as compared to *multiPath* forwarding mode.

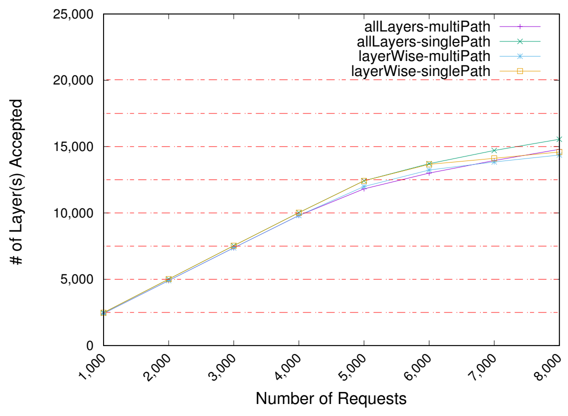

Figure 7. SVC Layers accepted.

We further analyze our DS by looking into the total number of layers rejected in Figure 8. It can be clearly seen that the scheduler in *singlePath* forwarding mode rejects less number of layers as compared to the scheduler in *multiPath* forwarding mode. However, Figures 7 and 8 do not provide the complete picture of individual layer types rejected, therefore we look briefly at Figures 9 and 10, that provide statistics on types of layers rejected.

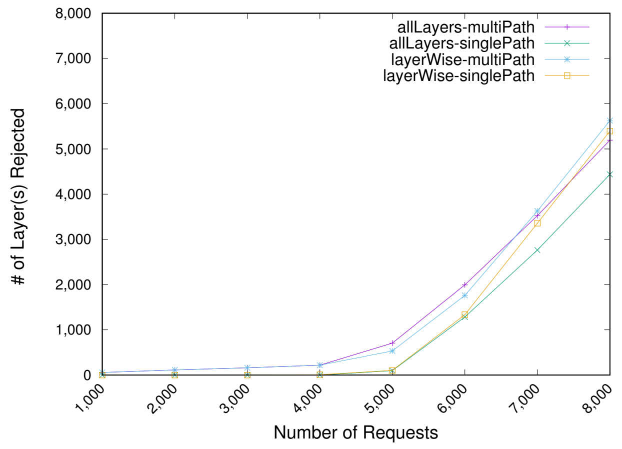

Figure 8. SVC Layers rejected.

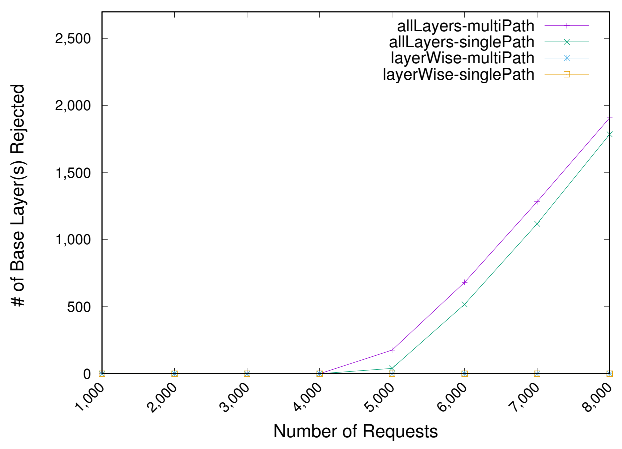

Figure 9. Base Layer rejected.

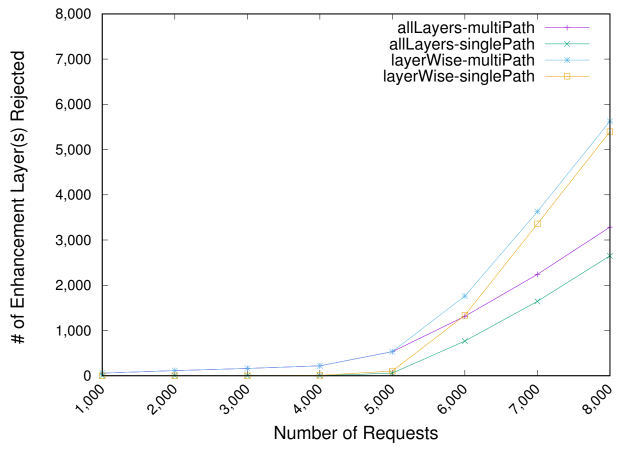

Figure 10. Enhancement Layer rejected.

Figure 9 shows the scheduler in *allLayer* processing mode rejects base layers. Moreover, none of the base layers are rejected when the scheduler is in *layerWise* processing mode. If a base layer is rejected the remaining layers are not accommodated therefore, a base layer rejection is equivalent to a client request rejection. The base layers are not rejected in *layerWise* processing mode because the scheduler incrementally accommodates all client requests that is, the base layer of all clients are accepted first, then enhancement layer one, and so on. However, the number of rejected requests for enhancement layers bumps up, for the scheduler in *layerWise* processing mode, as shown in the Figure 10, which is intuitive. The reason for this is because the maximum number of base layers have already saturated the server links for the scheduler in *layerWise* processing mode.

While looking at the results from Figures 9 and 10 we can conclude that the scheduler with *allLayers* processing mode rejects the request of new clients at the cost of accommodating maximum request layers of clients it has accepted. Therefore, assigning priority to the client requests that are already accepted. Moreover, the scheduler in *layerWise* processing mode gives priority to accepting a maximum number of client requests. In simpler words the scheduler in *allLayers* processing mode accommodates fewer client requests as compared to the scheduler in *layerWise* mode, but provides the accepted clients with a maximum number of SVC layers. Therefore, *allLayers* results in higher video quality for each client. However, the scheduler in *layerWise* processing mode accommodates the maximum number of client requests as compared to the scheduler in *allLayers* mode, but provides the accepted clients with a minimum number of SVC layers. Therefore, the scheduler in *layerWise* results in basic video quality for each client. But what makes the analysis interesting is the bandwidth consumption of scheduler configured in different forwarding and processing modes, as seen in Figure 11.

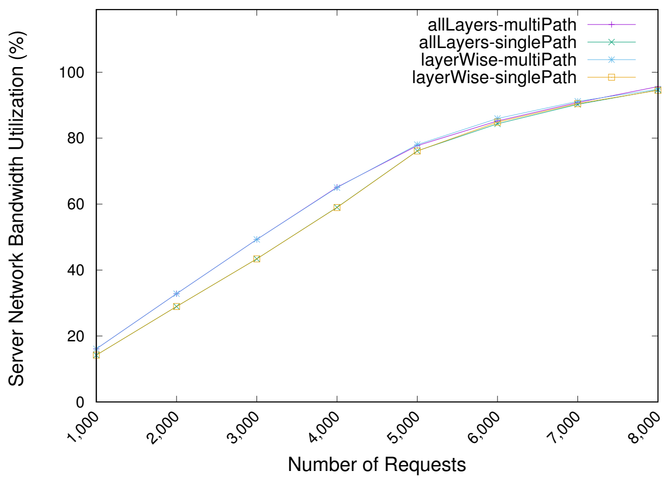

Figure 11. Server network bandwidth utilization.

Figure 11 shows the network bandwidth utilization for the scheduler configured with different processing and forwarding modes. Scheduler with *allLayer* processing mode accommodates the maximum number of layers as seen from previous results. However, the bandwidth utilization for scheduler in *allLayer* and *layerWise* processing mode is similar. The reason for similar bandwidth is due to the forwarding modes of the scheduler. When the scheduler is in *singlePath* forwarding mode all requests are placed on one path depending on the processing mode of the scheduler. However, when the scheduler is in *multiPath* forwarding mode all requests are placed multiple paths depending on the processing mode of the scheduler. when the scheduler is in *multiPath* forwarding mode, the first path found is usually the shortest path. Finding another path usually results in a longer path, therefore more bandwidth of links is utilized. We can also see in Figure 11 that around eighty percent of the paths that are connected to the servers are utilized while accommodating around five thousand requests. In order to accommodate the remaining client requests, maximum number of requests are rejected due to the server paths being saturated. Therefore, the server paths cannot accommodate the remaining layers of the requests. This results in sharp curve for the rejected number of layers as seen in Figures 8–10. Therefore, we can conclude from this that the network operators can utilize eighty percent of the server bandwidth for accepting client requests and the remaining twenty percent for network diagnostics, forwarding or switching.

We finally look at our last graph which shows the total number of paths found by the scheduler, as shown in Figure 12. It is obvious that sending each SVC layer on different paths will result in finding more number of paths.

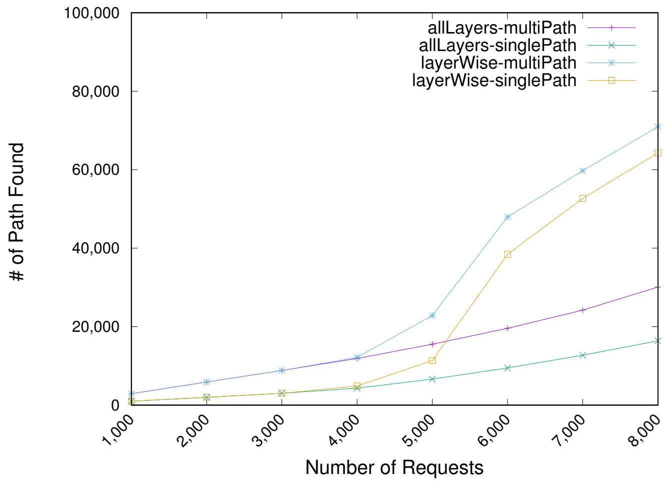

Figure 12. Number of paths found to accept requests.

7. Conclusions

In this work, we present a DMPF scheduler that uses multi-path TCP (MPTCP) to transmit different Scalable Video Coding (SVC) layers, over Moving Picture Experts Group (MPEG) Dynamic Adaptive Streaming over HTTP (DASH) standard in software defined network (SDN). The DMPF scheduler has advanced forwarding and processing modes. The processing modes are categorized into *allLayer* and *layerWise*. The forwarding mode consists of *singlePath* or *multiPath*.

We compare the combination of different DMPF scheduler processing and forwarding modes of the DMPF schedular and dive deep into identifying which mode combination yields the acceptance of the maximum number of client streaming requests and identify the trade-offs associated with the combination. We find that *allLayers-multiPath* mode results in accepting the maximum number of SVC layers after *allLayers-singlePath* mode. However, both of these modes do not accommodate the maximum number of clients. Therefore, both modes focus more on providing the maximum number of enhancement layers to each client request. Therefore, the quality of experience of the client is improved. Moreover, *layerWise-multiPath* and *layerWise-singlePath* modes accept the maximum number of client requests as they do not reject base layers. However, they reject enhancement layers therefore providing each client with a basic representation of the video. We see that *layerWise* processing mode is slower and more time consuming as compared to *allLayer* processing mode. Moreover, we also conclude that it is wise to use eighty percent of the capacity of the paths connected to the DASH server to accept client requests. The remaining twenty percent of the capacity can be used for network diagnostics, forwarding and switching and so forth.

Author Contributions: Conceptualization, A.G. and S.L.; methodology, A.G. and S.L.; software, A.G.; formal analysis, A.G. and S.L.; investigation, A.G. and S.L.; resources, S.L.; data curation, A.G.; writing–original draft preparation, A.G.; writing–review and editing, A.G. and S.L.; visualization, A.G.; supervision, S.L.; project administration, S.L.; funding acquisition, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Global Scholarship Program for Foreign Graduate Students at Kookmin University in Korea and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2019R1F1A1061136).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*; White Paper; Cisco Public: San Francisco, CA, USA, 2019.
2. Sripanidkulchai, K.; Maggs, B.; Zhang, H. An Analysis of Live Streaming Workloads on the Internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*; ACM: New York, NY, USA, 2004; pp. 41–54. [CrossRef]
3. Sanchez, Y.; Schierl, T.; Hellge, C.; Wiegand, T.; Hong, D.; De Vleeschauwer, D.; Van Leekwijck, W.; Le Louédec, Y. Efficient HTTP-based Streaming Using Scalable Video Coding. *Image Commun.* **2012**, *27*, 329–342. [CrossRef]
4. Wang, B.; Kurose, J.; Shenoy, P.; Towsley, D. Multimedia Streaming via TCP: An Analytic Performance Study. *ACM Trans. Multimedia Comput. Commun. Appl.* **2008**, *4*, 1–22. [CrossRef]
5. Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A. Overview of the H.264/AVC Video Coding Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 560–576. [CrossRef]
6. Schwarz, H.; Marpe, D.; Wiegand, T. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Trans. Circuits Syst. Video Technol.* **2007**, *17*, 1103–1120, [CrossRef]
7. Apple. Apple HTTP Live Streaming. Available online: https://developer.apple.com/streaming/ (accessed on 12 October 2020).
8. Zambelli, A. *IIS Smooth Streaming Technical Overview*; Technical Report; Microsoft Corporation: Redmond, WA, USA, 2009.
9. Akamai. Akamai HD. Available online: https://www.akamai.com/uk/en/about/news/press/2009-press/akamai-unveils-the-akamai-hd-network.jsp (accessed on 12 October 2020).
10. Adobe. Adobe HTTP Dynamic Streaming (HDS). Available online: https://www.adobe.com/devnet/hds.html (accessed on 12 October 2020).
11. Sodagar, I. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia* **2011**, *18*, 62–67. [CrossRef]
12. Raiciu, C.; Paasch, C.; Barre, S.; Ford, A.; Honda, M.; Duchene, F.; Bonaventure, O.; Handley, M. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *NSDI'12: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*; USENIX Association: Berkeley, CA, USA, 2012; pp. 29–29.
13. Barré, S.; Paasch, C.; Bonaventure, O. MultiPath TCP: From Theory to Practice. In *NETWORKING'11: Proceedings of the 10th International IFIP TC 6 Conference on Networking—Volume Part I*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 444–457.
14. Gohar, A.; Raza, A.; Lee, S. Dynamic Adaptive Streaming Over HTTP Using Scalable Video Coding with Multipath TCP in SDN. In Proceedings of the 2019 International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 9–11 January 2019; pp. 480–484. [CrossRef]
15. Huysegems, R.; De Vleeschauwer, B.; Wu, T.; Van Leekwijck, W. SVC-Based HTTP Adaptive Streaming. *Bell Lab. Tech. J.* **2012**, *16*, 25–41. [CrossRef]
16. Sieber, C.; Hoßfeld, T.; Zinner, T.; Tran-Gia, P.; Timmerer, C. Implementation and user-centric comparison of a novel adaptation logic for DASH with SVC. In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 1318–1323.
17. Grafl, M.; Timmerer, C.; Hellwagner, H.; Cherif, W.; Ksentini, A. Evaluation of hybrid Scalable Video Coding for HTTP-based adaptive media streaming with high-definition content. In Proceedings of the 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Madrid, Spain, 4–7 June 2013; pp. 1–7. doi:10.1109/WoWMoM.2013.6583506.
18. Sánchez de la Fuente, Y.; Schierl, T.; Hellge, C.; Wiegand, T.; Hong, D.; De Vleeschauwer, D.; Van Leekwijck, W.; Le Louédec, Y. iDASH: Improved Dynamic Adaptive Streaming over HTTP Using Scalable Video Coding. In *MMSys '11: Proceedings of the Second Annual ACM Conference on Multimedia Systems*; ACM: New York, NY, USA, 2011; pp. 257–264. [CrossRef]
19. Famaey, J.; Latré, S.; Bouten, N.; de Meerssche, W.V.; Vleeschauwer, B.D.; Leekwijck, W.V.; Turck, F.D. On the merits of SVC-based HTTP Adaptive Streaming. In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 419–426.

20. Andelin, T.; Chetty, V.; Harbaugh, D.; Warnick, S.; Zappala, D. Quality Selection for Dynamic Adaptive Streaming over HTTP with Scalable Video Coding. In *MMSys '12: Proceedings of the 3rd Multimedia Systems Conference*; ACM: New York, NY, USA, 2012; pp. 149–154. [CrossRef]
21. Bouten, N.; Latré, S.; Famaey, J.; Turck, F.D.; Leekwijck, W.V. Minimizing the impact of delay on live SVC-based HTTP adaptive streaming services. In Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium, 27–31 May 2013; pp. 1399–1404.
22. Nguyen, T.; Cheung, S.C.S. Multimedia streaming using multiple TCP connections. In Proceedings of the PCCC 2005 24th IEEE International Performance, Computing, and Communications Conference, Phoenix, AZ, USA, 7–9 April 2005; pp. 215–223. [CrossRef]
23. Han, B.; Qian, F.; Ji, L.; Gopalakrishnan, V. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *CoNEXT '16: Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*; ACM: New York, NY, USA, 2016; pp. 129–143. [CrossRef]
24. Bradai, A.; Ahmed, T.; Boutaba, R.; Ahmed, R. Efficient content delivery scheme for layered video streaming in large-scale networks. *J. Netw. Comput. Appl.* **2014**, *45*, 1–14, [CrossRef]
25. Yue, Y.; Ran, Y.; Chen, S.; Yang, B.; Sun, L.; Yang, J. Joint Routing and Layer Selecting for Scalable Video Transmission in SDN. In Proceedings of the 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, USA, 6–10 December 2015; pp. 1–6, [CrossRef]
26. Uzakgider, T.; Cetinkaya, C.; Sayit, M. Learning-based Approach for Layered Adaptive Video Streaming over SDN. *Comput. Netw.* **2015**, *92*, 357–368, [CrossRef]
27. Cetinkaya, C.; Ozveren, Y.; Sayit, M. An SDN-assisted system design for improving performance of SVC-DASH. In Proceedings of the 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), Lodz, Poland, 13–16 September 2015; pp. 819–826, [CrossRef]
28. Cetinkaya, C.; Karayer, E.; Sayit, M.; Hellge, C. SDN for segment based flow routing of DASH. In Proceedings of the 2014 IEEE Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin), Berlin, Germany, 7–10 September 2014; pp. 74–77, [CrossRef]
29. Herguner, K.; Kalan, R.S.; Cetinkaya, C.; Sayit, M. Towards QoS-aware routing for DASH utilizing MPTCP over SDN. In Proceedings of the 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 6–8 November 2017; pp. 1–6.
30. Barakabitze, A.A.; Mkwawa, I.; Sun, L.; Ifeachor, E. QualitySDN: Improving Video Quality using MPTCP and Segment Routing in SDN/NFV. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 182–186.
31. Medina, A.; Lakhina, A.; Matta, I.; Byers, J. BRITE: An approach to universal topology generation. In Proceedings of the MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Cincinnati, OH, USA, 15–18 August 2001; pp. 346–353. [CrossRef]