Demonstration of OpenFlow-Controlled Network Orchestration for Adaptive SVC Video Manycast

Nana Xue, Xiaoliang Chen, Long Gong, Suoheng Li, Daoyun Hu, and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—Software defined networking (SDN) makes networks programmable and application-aware by decoupling network control and management (NC&M) from data forwarding and leveraging centralized NC&M to facilitate user-customized routing and switching. Inspired by these, this paper investigates how to realize the OpenFlow-controlled (OF-controlled) network orchestration that can facilitate efficient scalable video coding (SVC) streaming to heterogeneous clients. Specifically, we consider real-time SVC streaming and address the situation in which video sources reside in geographically-distributed servers and clients can join and leave the streaming services dynamically. We formulate this as a multi-source multi-destination manycast problem and realize the networking system with an OF-controlled SDN architecture. We first design the OF controller to enable efficient network operations. Then, we focus on solving the multi-source multi-destination SVC video manycast problem and design several algorithms. Initially, an integer linear programming (ILP) model is formulated to obtain the optimal solutions for small-scale problems. Next, we try to make the manycast algorithm suitable for practical implementation, and design two time-efficient heuristics. Simulation results indicate that the heuristics can provide close-to-optimal solutions. Finally, we build an OF network testbed that consists of OF switches, SVC video servers and clients, and perform SVC streaming experiments to demonstrate our design. Experimental results verify that the proposed scheme can allocate bandwidth intelligently and ensure high-quality video streaming. To the best of our knowledge, this is the first work that accomplishes experimental demonstration of OF-controlled network orchestration for adaptive SVC video manycast.**

***Index Terms*—Manycast, openflow, scalable video coding (SVC), software-defined networking (SDN), video streaming.**

I. Introduction

OVER the past decades, multimedia streaming applications, such as video conferencing, online video games and high-definition IP television, have become increasingly popular on the Internet and pushed the network traffic to grow exponentially. It is known that by 2012, video traffic has already occupied more than 50% of the Internet bandwidth [1]. Therefore, multimedia streaming, especially the live video streaming, has definitely become a killer-application for the current Internet. Moreover, due to the rise of mobile networks, the clients of video streaming are more and more heterogeneous, i.e., different clients can have various system capacity, access bandwidth, as well as quality-of-service (QoS) requirements. Hence, customized and differentiated service provisioning for video streaming is imperative. Currently, in order to realize customized service provisioning, a video server may have to stream the same video to multiple clients through multiple unicast connections. This scheme, however, can generate redundant traffic and waste a fair amount of network bandwidth. Hence, to achieve a better tradeoff between the QoS provisioning and bandwidth utilization, an intelligent network orchestration mechanism that can realize efficient video streaming to heterogeneous clients with low latency, small delay jitter and low packet loss rate is necessary.

The aforementioned network orchestration needs technical innovations in both the video and network areas. Firstly, from the video side, the scalability of video coding plays an important role. Thanks to the advances on scalable video coding (SVC) [2], one can encode a video into several layers (i.e., sub-streams) such that different levels of playback qualities can be achieved by receiving different subsets of them [3]. Specifically, SVC encodes a video into a base layer (BL) and several enhancement layers (ELs). A streaming client can decode the video with only the BL, but the more consecutive ELs it receives, the higher playback quality it would enjoy. This property, together with an intelligent bandwidth provisioning scheme, provide a promising solution to the problem of how to realize efficient video streaming to heterogeneous clients.

Secondly and more importantly, from the network side, we need the architecture and protocols that can accomplish highly-efficient video streaming to heterogeneous clients. Previously, people have demonstrated that video streaming with IP multicast could improve bandwidth utilization [4]. However, it is also known that IP multicast still faces several challenges. First of all, since the state of each multicast group has to be maintained on the routers, IP multicast makes the operations in routers more complicated. Secondly, because IP networks use distributed network control and management (NC&M), the routers have difficulties getting the global network status and can hardly figure out the optimal multicast trees for ensuring the end-to-end QoS requirements [5]. Finally, the multicast

Manuscript received December 05, 2014; revised March 19, 2015 and June 18, 2015; accepted June 23, 2015. Date of publication June 25, 2015; date of current version August 10, 2015. This work was supported in part by the NCET program under Project NCET-11-0884, by the NSFC Project 61371117, by the Fundamental Research Funds for the Central Universities under Grant WK2100060010, by the Natural Science Research Project for Universities in Anhui under Grant KJ2014ZD38, and by the Strategic Priority Research Program of the CAS under Grant XDA06011202. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tommaso Melodia.

The authors are with the School of Information Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: xuenana@mail.ustc.edu.cn; arabus@mail.ustc.edu.cn; lzzitc@mail.ustc.edu.cn; gonglong@mail.ustc.edu.cn; 517266593@qq.com; zqzhu@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2015.2450014

scheme for video streaming is usually dynamic in the Internet, i.e., clients can join and leave on-the-fly. Nevertheless, the current IP networks do not have the capability of reconfiguring the routing paths dynamically and adaptively.

Recently, software defined networking (SDN) has been proposed to make the network programmable and application-aware [6]. It decouples network control from data forwarding and leverages centralized NC&M to facilitate software-/user-customized routing, switching and network management. Due to its flexibility and programmability, SDN provides us a new paradigm to realize the network orchestration that can facilitate efficient video streaming to heterogeneous clients. The launch of OpenFlow (OF)[1] [7] makes this target more practical. Developed as an open standard protocol, OF implements flow-based switching in the OF switches to make them forward packets based on the flow-tables, which are calculated and provided by a centralized OF controller. Since the OF controller has the global view of the network, it can adjust bandwidth provisioning schemes adaptively and utilize the network resources more efficiently.

Previous studies have investigated SDN-based multicast from different perspectives. In [8], the authors proposed an OF-controlled multicast system in which the OF controller could adjust the multicast provisioning scheme with fast tree switching. However, as all the candidate multicast trees are pre-calculated, the proposed system may have difficulty handling the situation in which the clients can join and leave the multicast groups dynamically. Yang *et al.* leveraged OF to facilitate the join-and-leave operations for multicast in [9]. They only described the operation principle of the OF system but did not address the algorithm design for realizing efficient multicast group adjustments. Aiming at improving the security and controllability of multicast, a secure multicast system was designed based on SDN in [10]. In [11], Iyer *et al.* discussed the SDN-based multicast in datacenter networks and proposed to make multicast available in the commodity switches.

On the other hand, the authors of [12]–[15] are pioneering in the field of QoS routing for the SVC video streaming over OF/SDN networks. In [12], [13], they presented an analytical framework to optimize the QoS-aware routing for SVC video streaming over OF networks. In the framework, the BL of an SVC video is treated as the QoS flow that has the highest priority, while its ELs are considered as the best-effort flows. Then, they transformed the QoS-aware routing into a constrained shortest-path routing problem and solved it. More recently, they extended the work to consider the end-to-end QoS provisioning for SVC video streaming over multi-domain SDN networks in [14], [15]. Even though the ideas and results in [12]–[15] were very interesting, the authors only addressed the QoS routing for unicast-based video streaming and did not consider the full network orchestration.

Note that with the idea of network orchestration, we can expect the OF controller to take care of more NC&M tasks than routing calculation. For instance, it can also be in charge of the video server selection for clients. More specifically, if we consider real-time SVC video streaming and try to address the generic situation in which the same video source can reside in multiple servers and several clients can join and leave the streaming dynamically, the OF controller has a multi-source multi-destination manycast problem [16] to solve. Basically, the controller needs to properly select the source video server for each client and adjust the manycast forest (i.e., a group of multicast trees) adaptively when the network status changes.

In this paper, we propose an OF-controlled network system to realize the aforementioned network orchestration for adaptive SVC video manycast. We first design the functional modules for the OF controller. Then, we focus on how to solve the multi-source multi-destination SVC video manycast problem efficiently and perform theoretical analysis for algorithm design. An integer linear programming (ILP) model is formulated to obtain the optimal solutions for small-scale problems. Next, we try to reduce the computational complexity and make the manycast algorithm suitable for practical implementation in the OF controller, and design two time-efficient heuristics. The ILP and heuristics are evaluated with numerical simulations and the results indicate that the heuristics can provide close-to-optimal solutions for the manycast problem. Finally, we build an OF network testbed that consists of OF switches, SVC video servers and clients, design necessary OF protocol extensions, implement the heuristics in the OF controller, and perform SVC video streaming experiments to demonstrate our design. In the experiments, we collect the results on streaming bandwidth, packet-loss-rate (PLR), luminance components peak signal-to-noise-ratio (Y-PSNR) of video playback, and delay jitter, and they verify that the proposed scheme can allocate the bandwidth intelligently and ensure high-quality video streaming. To the best of our knowledge, this is the first work that accomplishes the network-wide experimental demonstrations of OF-controlled network orchestration for adaptive SVC video manycast.

The rest of the paper is organized as follows. In Section II, we describe the network architecture and our system design. Section III discusses the problem of multi-source multi-destination SVC video manycast and formulates an ILP model to solve it. The time-efficient heuristics are proposed in Section IV, and Section V shows the simulation results. Section VI presents the system implementation and experimental demonstrations. Finally, Section VII summarizes the paper.

II. OpenFlow Network System for Adaptive SVC Video Manycast

This section discusses the architecture of the OF network and our design of the functional modules in the OF controller.

A. Network Architecture

Fig. 1 shows the overall architecture of the OF-based SVC video manycast system. In the control plane, we have a centralized OF controller that communicates with the OF switches using the OF protocol. The OF controller is in charge of the NC&M tasks, e.g., managing video streaming paths and selecting video servers for clients. The data plane consists of OF switches, video servers and clients. Each OF switch

[1][Online]. Available: https://www.opennetworking.org/

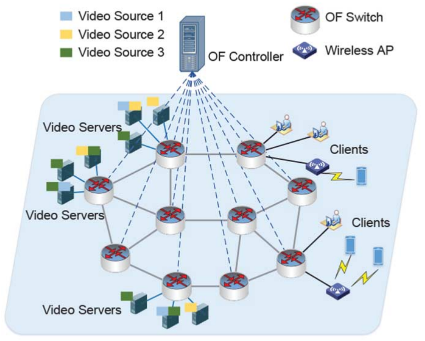


Fig. 1. Network architecture of OF-based video manycast system.

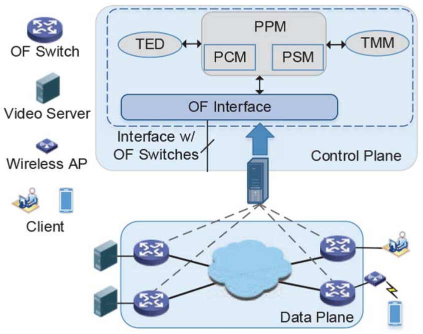


Fig. 2. Design of functional modules in OF controller.

receives the flow-tables from the OF controller, configures its forwarding actions accordingly, and uses them to forward the video traffics correctly. The geographically-distributed video servers attach to the OF switches, and they carry overlapped video sources, i.e., the same video content can be duplicated on multiple servers. The clients access to the real-time video streaming service through the OF switches. They can either directly connect to an OF switch with a wired connection (e.g., Ethernet) or join the network through a wireless access-point (AP) that attaches to an OF switch.

B. Functional Modules in OF Controller

Our design of the OF controller is illustrated in Fig. 2, where we plot the functional modules and the relation among them. When a client tries to require a video streaming service, it sends a connection request to the nearest OF switch. The OF switch checks its flow-tables and finds out that this is a new service request as no flow-table has been configured for it. Then, the OF switch encodes the request's information in a *PacketIn* message and sends it to the OF controller. The OF controller parses the message, checks the current network status, and calculates the video streaming scheme, which includes the video server and streaming path, for the client. Next, the OF controller encodes the information of the streaming path in *FlowMod* messages, distributes them to the related OF switches to install the path, and informs the video server about the new client. When the path is set up, the client starts to enjoy the video streaming service.

Note that with the manycast scenario, the OF controller does not have to set up a new server-to-client path for the client. If the client is close to one of the intermediate OF switches in the manycast forest and the bandwidth resource is sufficient, the OF controller can inform the OF switch to work as a splitting node and relay the video to it. By doing so, we can utilize the bandwidth resources more efficiently. Also, during the network operation, the OF controller can adjust the manycast forest adaptively, when the network status changes.

The details of the functional modules in the OF controller are explained as follows.

- *Path Provision Module* (PPM): It interacts with the OF switches to handle the OF messages and manages the service provisioning schemes for SVC video streaming. There are two sub-components in PPM, i.e., the path computation module (PCM) and the path switching module (PSM). They work collaboratively to determine the service provisioning scheme for each client. When it receives a *PacketIn* message, PPM parses the message, extracts the request's information, and forwards the information to PCM for server selection and path computation.
- *Path Computation Module* (PCM): It is responsible for calculating the service provisioning scheme for each client. When it receives a task from PPM, PCM gets the current network status by combining the information stored in the topology management module (TMM) and the traffic engineering database (TED), and performs server selection and path computation accordingly. Then, it returns the result to PMM, which will build the *FlowMod* messages for all the related OF switches.
- *Path Switching Module* (PSM): It handles the dynamic path reconfiguration in the manycast forest. For instance, when PCM cannot find a valid service provisioning scheme for a client to satisfy its QoS requirements, PSM can be invoked to reconfigure certain paths in the manycast forest so as to serve the client with best effort.
- *Topology Management Module* (TMM): It gathers the information on network topology and updates the topology in real time when there are OF switches joining or leaving the network and/or the connectivity among the OF switches changes. Specifically, it uses the link layer discovery protocol (LLDP) to realize the topology discovery with OF messages. In our implementation, the OF controller requests for the network status by sending *FlowStatus_Request* messages to the OF switches repeatedly, and the OF switches reply with *FlowStatus_Reply* messages. According the information included in the *FlowStatus_Reply* messages, TMM can obtain the information on network topology.
- *Traffic Engineering Database* (TED): It stores the status of in-service manycast traffic, e.g., the bandwidth utilization on each link, the load on each video server, the manycast forests for each contents, etc.

III. Problem Formulation

In this section, we describe the theoretical model for the multi-source multi-destination SVC video manycast problem and formulate an ILP model to solve it.

A. Network Model and Problem Description

The topology of the OF network is denoted as a directed graph $G(V, E, B, \Delta)$, where V and E are the sets of nodes and links, respectively. Here, V includes three types of nodes, i.e., the video servers that each attaches to an OF switch locally, the clients and the OF switches. We denote the sets of video servers and clients as S and D, and hence the set of OF switches is $V\backslash(S \cup D)$. If link e is from u to v $(u, v \in V)$, we have $e = (u, v)$ and denote its bandwidth capacity and delay as $b_{(u,v)} \in B$ and $\delta_{(u,v)} \in \Delta$, respectively. To get each $\delta_{(u,v)}$, we consider both the transmission delay and propagation delay of link (u, v). Similar to the work in [17], we assume that for each link (u, v), $b_{(u,v)}$ and $\delta_{(u,v)}$ are not directly related.

On the servers in S, we have the video sources/contents, the set of which can be denoted as C. For all the contents in C, we assume that each of them can be encoded into at most N_L SVC layers, which are BL and ELs.

We consider the situation that in the network, there can be multiple pending requests trying to get the SVC streaming service. Hence, we denote the set of all the pending requests as $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_k, \cdots\}$. Each request is represented as $\gamma_k(d_k, c_k, l_k, t_k)$, where k is its index, $d_k \in D$ is the client, $c_k \in C$ is the requested content, l_k is the number of requested SVC layers, and t_k is the holding time (i.e., the time duration that the request wants to enjoy the video streaming service). Note that in our network model, it is possible that the same client asks for different contents in C, or different clients ask for the same content. In both cases, the requests are treated as different ones. Basically, we distinguish the requests in Γ with their indices regardless of the clients and requested contents.

We introduce the following definitions to describe the service provisioning in the network.

Definition 1: We define the actual number of SVC layers that request γ_k is provisioned with as l_k^a.

Definition 2: We define $t_{k,i}^a$ as the actual service time for the i-th SVC layer of c_k, when serving request γ_k.

Here, we have $l_k^a \leq l_k$ and $t_{k,i}^a \leq t_k, \forall i \in [1, l_k]$. Basically, in the service provisioning, we may have $l_k^a < l_k$ due to bandwidth insufficiency. Moreover, even when the i-th SVC layer is provisioned to γ_k, we may have $t_{k,i}^a < t_k$. This is because our service provisioning algorithm may switch off certain ELs to some in-service clients dynamically. The details on this will be presented in SubSection IV-B.

Definition 3: To evaluate a service provisioning scheme, we define R_k as the revenue gain from serving request γ_k.

Basically, the satisfaction of client d_k decides that the revenue gain R_k should be related to the number of continuous SVC layers delivered to d_k and the service duration of each layer. Hence, we assume that the revenue from each layer has a linear relation with its service duration [18]. Specifically, for request γ_k, if d_k has received the i-th SVC layer for a duration of $t_{k,i}^a$, the revenue gain is $w_i \cdot t_{k,i}^a$, where w_i is the revenue weight (i.e., price per time-unit for the i-th SVC layer). For an SVC streaming service, the BL ($i = 1$) is the most important because we cannot decode the video without it, and then, the received video's quality would become better if more subsequent ELs ($i > 1$) are delivered. Hence, we choose the weights as $w_1 > w_2 \geq w_3 \cdots \geq w_{N_L}$, which is similar to the setting in [19]. Finally, after normalization, we can obtain the revenue gain from serving request γ_k (i.e., R_k) as

$$R_k = \frac{1}{\sum_{j=1}^{l_k} w_j} \sum_{i=1}^{l_k^a} w_i \cdot t_{k,i}^a, \quad \forall k \in [1, |\Gamma|]. \tag{1}$$

Then, the total revenue gain will be

$$R = \sum_{k=1}^{|\Gamma|} R_k. \tag{2}$$

The objective of the service provisioning is to maximize the total revenue gain in (2), and we refer to this problem as maximizing revenue with layered manycast (MRLM).

B. ILP Model

We first present an ILP model to solve the MRLM problem exactly. With the network model described above, we can formulate the manycast problem as an optimization to maximize the total revenue defined in (2). The problem can be solved with the ILP model discussed below. Here, for simplicity, the ILP model only tries to serve the new clients that are still pending, but it does not consider the reconfiguration of the in-service requests. Hence, when using (1) to calculate the revenue gain, we need to replace $t_{k,i}^a$ with t_k.

Parameters:

- $b_{(u,v)} \in B$: the bandwidth capacity on link (u, v);
- $\delta_{(u,v)} \in \Delta$: the delay on link (u, v);
- Γ: the set of pending requests;
- C: the set of video contents;
- $b_{c,i}$: the bandwidth requirement for delivering the i-th SVC layer of content $c \in C$;
- $S_{c,i}$: the set of video servers that store the i-th SVC layer of content c, $S_{c,i} \subseteq S$;
- δ_{max}: the maximum delay that the system can tolerate for each SVC layer;
- σ_{max}: the maximum delay jitter between any pair of SVC layers to the same client. For example, if $\sigma_{max} = 20$ time-units and the delay of BL is 50 time-units, then the delay of the first EL should be within [30,70] time-units;
- w_i: the weight of the i-th SVC layer of any content;
- ϵ: the bandwidth scaling factor.

Variables:

- $x_{k,i,s}$: Boolean variable that equals 1 if client d_k in request γ_k receives the i-th layer of the requested video content c_k from the server $s \in S$, otherwise $x_{k,i,s} = 0$;
- $f_{k,i}^{(u,v)}$: Boolean variable that equals 1 if link (u, v) is used to deliver the i-th layer of the requested video content c_k for request γ_k, otherwise $f_{k,i}^{(u,v)} = 0$;
- $g_{c,i}^{(u,v)}$: Boolean variable that equals 1 if link (u, v) is used to deliver the i-th layer of content c, otherwise $g_{c,i}^{(u,v)} = 0$;

Objective:

The total revenue gain per time-unit from serving all the request in Γ can be calculated as

$$R = \sum_{k=1}^{|\Gamma|} \left[\frac{1}{\sum_{j=1}^{l_k} w_j} \sum_{i=1}^{l_k} \left(w_i \cdot \sum_{s \in S} x_{k,i,s} \right) \right]. \tag{3}$$

With (3), we define the optimization objective as

$$Maximize \quad R. \tag{4}$$

Constraints

1) *Flow Conservation Constraints*:

$$\sum_{(u,v)\in E} f_{k,i}^{(u,v)} - \sum_{(v,u)\in E} f_{k,i}^{(v,u)} = \begin{cases} x_{k,i,s}, u = s, \\ -x_{k,i,s}, u = d_k, \\ 0, \qquad \text{otherwise}, \end{cases} \quad \forall k \in [1, |\Gamma|], s \in S_{c_k,i}, i \in [1, l_k]. \quad (5)$$

Equation (5) ensures that if the manycast forest covers client d_k and carries the traffic of the i-th layer of c_k to it, the video streaming for that layer uses a single path from server s to d_k.

$$\sum_{s\in S_{c_k,i}} x_{k,i,s} \le 1, \quad \forall k \in [1, |\Gamma|], i \in [1, l_k]. \quad (6)$$

Equation (6) ensures that we can use at most one video server to send the i-th layer of c_k to client d_k.

2) *Traffic Aggregation Constraints*:

$$g_{c,i}^{(u,v)} \ge f_{k,i}^{(u,v)}, \{k : c_k = c\}, \quad \forall (u,v) \in E, i \in [1, N_L] \quad (7)$$

$$g_{c,i}^{(u,v)} \le \sum_{\{k:c_k=c\}} f_{k,i}^{(u,v)}, \quad \forall (u,v) \in E, i \in [1, N_L] \quad (8)$$

Equations (7)–(8) ensure that on each link (u, v), the traffics to different clients are aggregated if they are for the same SVC layer of the same video content, since we use a multi-source multi-destination manycast forest to serve all the clients that require for the same video content.

3) *Delay Constraints*:

$$\sum_{(u,v)\in E} f_{k,i}^{(u,v)} \cdot \delta_{(u,v)} \le \delta_{max}, \forall i \in [1, l_k], k \in [1, |\Gamma|] \quad (9)$$

Equation (9) ensures that all the service provisioning schemes in the manycast forest can satisfy the delay requirement from the real-time video streaming service.

4) *Jitter Constraints*:

$$\sum_{(u,v)\in E} f_{k,i}^{(u,v)} \cdot \delta_{(u,v)} - \sum_{(u,v)\in E} f_{k,j}^{(u,v)} \cdot \delta_{(u,v)} + \sigma_{max} \ge 0,$$
$$\sum_{(u,v)\in E} f_{k,j}^{(u,v)} \cdot \delta_{(u,v)} - \sum_{(u,v)\in E} f_{k,i}^{(u,v)} \cdot \delta_{(u,v)} + \sigma_{max} \ge 0,$$
$$\{i, j : i \ne j, i, j \in [1, l_k]\}, \forall k \in [1, |\Gamma|] \quad (10)$$

Equation (10) ensures that all the service provisioning schemes in the manycast forest satisfy the delay jitter requirement from the real-time video streaming service [20], [21].

5) *Video Decodable Constraints*:

$$\sum_{s\in S_{c_k,1}} x_{k,1,s} \ge \cdots \ge \sum_{s\in S_{c_k,l_k}} x_{k,l_k,s}, k \in [1, |\Gamma|] \quad (11)$$

Equation (11) ensures that the SVC layers received by each client are decodable. Note that according to the principle of SVC, the client has to receive consecutive layers to decode a video, which means that the i-th ($i > 1$) layer is decodable only if all the layers from the first to $(i-1)$-th have been received successfully. For instance, if we want to decode the video data in the first EL successfully, the BL has also to be received.

6) *Bandwidth Constraints*:

$$(1+\epsilon) \cdot \left(\sum_{c\in C} \sum_{i=1}^{N_L} g_{c,i}^{(u,v)} \cdot b_{c,i} \right) \le b_{(u,v)}, \quad \forall (u,v) \in E \quad (12)$$

Equation (12) ensures that the service provisioning schemes cannot use bandwidth that is more than the link capacity. Note that for practical SVC video streaming, the bandwidth requirement $b_{c,i}$ can change with time. Therefore, we introduce a bandwidth scaling factor ϵ to assign certain redundant bandwidth to the streaming flows and make them more reliable [22].

C. Complexity Analysis

One way to look into the complexity of solving an ILP is to analyze the numbers of variables and constraints that it uses [23]. For the ILP discussed above, the number of variables is $N_L \cdot |\Gamma| \cdot |S| + (|\Gamma| + |C|) \cdot |V|^2 \cdot N_L$. And the upper-bound of the number of equality constraints is $|S| \cdot |\Gamma| \cdot |N_L|$, and the number of inequality constraints is bounded below $3 \cdot N_L \cdot |\Gamma| + (N_L - 1) \cdot |\Gamma| + N_L \cdot (|\Gamma| + |C|) \cdot |V|^2$.

The analysis above indicates that MRLM is a relatively complex problem. Actually, we can verify this observation from another perspective. Let us consider an over-simplified version of the MRLM, i.e., we only have one request $\gamma_1(d_1, c_1, l_1, t_1)$ to serve, c_1 is encoded into $N_L = 2$ SVC layers, and there is only one server $S = \{s\}$ in the network. The network is still the directed graph $G(V, E, B, \Delta)$ in which the bandwidth capacity of each link $(u, v) \in E$ is bounded. Then, in order to serve γ_1 and minimize the possibility of network congestion in the future (i.e., maximizing the revenue gain from subsequent requests), we need to solve the maximum 2-splittable flow problem, which is known to be $\mathcal{NP}$-hard [24]. Due to the complexity, we will propose several time-efficient heuristics for MRLM in the next section.

IV. Heuristic Algorithms

In this section, we design two time-efficient heuristics to solve the MRLM problem in real-time such that they can be practically implemented in the OF controller.

A. Layer-Based Serving Algorithm (LBS)

At each provision time, the OF controller needs to serve a set of requests $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_k, \cdots\}$. Therefore, in order to avoid the service unfairness that some requests do not even get the BLs of their requested contents while others receive not only the BLs but also the ELs, we propose a layer-based serving algorithm (LBS). LBS determines the provisioning schemes for the requests layer-by-layer, i.e., trying to serve the BLs first and then moving to the ELs in sequence. Specifically, we decompose a request $\gamma_k(d_k, c_k, l_k, t_k)$ into l_k sub-requests as $\gamma_{k,1}(d_k, c_k, t_k), \cdots, \gamma_{k,l_k}(d_k, c_k, t_k)$, each of which requires one SVC layer of c_k. For instance, $\gamma_{k,i}(d_k, c_k, t_k)$ is the sub-request for the i-th SVC layer of c_k. Then, we merge the sub-requests into groups based on their SVC layers, as

$$\Gamma_i = \{\gamma_{1,i}(d_1, c_1, t_1), \cdots, \gamma_{|\Gamma|,i}(d_{|\Gamma|}, c_{|\Gamma|}, t_{|\Gamma|})\} \tag{13}$$

and we have

$$\Gamma = \Gamma_1 \cup \Gamma_2 \cdots \cup \Gamma_{N_L}. \tag{14}$$

Basically, Γ_i represents the set of sub-requests that are for the i-th SVC layer of their contents. Note that in the same group, the requested contents of the sub-requests can be different.

For each sub-request $\gamma_{k,i}$, the OF controller provisions the SVC streaming by preprocessing the topology $G(V, E, B, \Delta)$ first to address the bandwidth constraints, i.e., ensuring that $\gamma_{k,i}$ will use a path that has sufficient bandwidth capacity. Specifically, since $\gamma_{k,i}$ is for the i-th layer of content c_k, the OF controller finds the set of links whose bandwidths are less than $b_{c_k,i}$ and denote it as E_0, i.e.

$$E_0 = \{(u, v) : b_{(u,v)} < (1 + \epsilon) \cdot b_{c_k,i}\}. \tag{15}$$

We introduce a delay set $\Delta_0 = \{\delta_{0,(u,v)}\}$, make $\Delta_0 = \Delta$, and then assign the delays of the links in E_0 as $+\infty$ in Δ_0, i.e.

$$\delta_{0,(u,v)} = +\infty, \quad \forall (u, v) \in E_0. \tag{16}$$

The streaming path for the i-th layer of c_k is obtained with the topology $G(V, E, B, \Delta_0)$. Here, we denote the path for the i-th layer of c_k as $p_{k,i}$, and the available bandwidth on $p_{k,i}$ is

$$b^p_{k,i} = \min_{(u,v) \in p_{k,i}} (b_{(u,v)}). \tag{17}$$

which should satisfy the following condition automatically

$$b^p_{k,i} \geq (1 + \epsilon) \cdot b_{c_k,i}. \tag{18}$$

Algorithm 1 shows the detailed procedure of LBS. Note that in the dynamic network operation where requests come and leave on-the-fly, *Algorithm 1* will be invoked repeatedly. The outer for-loop that covers *Lines* 1-22 handles an SVC layer of the requested contents, while the inner for-loop from *Line* 2 to *Line* 21 tries to find the provisioning scheme for each sub-request $\gamma_{k,i}(d_k, c_k, t_k)$ in group Γ_i. *Lines* 3-5 are for initialization, and then we run *Lines* 6-12 to find the feasible paths from each node in $S_{c_k,i}$ to client d_k. To adapt to the real-time video streaming service, *Line* 7 calculates the shortest path. To make the video decodable, we use *Lines* 13-17 to remove all the subsequent sub-requests for γ_k (i.e., $\{\gamma_{k,j}(d_k, c_k, t_k), j \geq i\}$) immediately, if no feasible path can be found for $\gamma_{k,i}$. Otherwise, in *Line* 18, we stream the i-th SVC layer of c_k to d_k with a feasible path $p_{k,i} \in P$. Specifically, we first try to select the streaming path in ascending order of their delays, and if there are multiple paths that have the same minimum delay, we select the one that can provides the maximum available bandwidth.

Algorithm 1: Layer-Based Serving Algorithm (LBS)

```
1.  for $i = 1$ to $l_k$ do
2.      for each sub-request $\gamma_{k,i} \in \Gamma_i$ do
3.          obtain the information of $\gamma_{k,i}(d_k, c_k, t_k)$;
4.          $P = \emptyset$;
5.          obtain $E_0$ and $\Delta_0$ with Eqs. (15)–(16);
6.          for each node $v \in S_{c_k,i}$ do
7.              calculate the shortest path $p$ with a finite length
                for $v \to d_k$ in $G(V, E, B, \Delta_0)$;
8.              store the delay of $p$ in $\delta^p$;
9.              if $\delta^p$ satisfies the delay and jitter constraints
                in (9) and (10) then
10.                 $P = P \cup \{p\}$;
11.             end
12.         end
13.         if $P = \emptyset$ then
14.             $l^a_k = i - 1$;
15.             remove $\{\gamma_{k,j}(d_k, c_k, t_k), j \geq i\}$ from $\Gamma$;
16.             break;
17.         end
18.         stream the $i$-th layer of $c_k$ to $d_k$ with the path
            $p_{k,i} \in P$ such that it has the minimum delay and
            maximum available bandwidth
19.         update $G(V, E, B, \Delta)$;
20.         $S_{c_k,i} = S_{c_k,i} \cup \{v : v \in p_{k,i}\}$;
21.     end
22. end
```

At this moment, the i-th SVC layer of c_k is successfully provisioned for request γ_k and we update the network status accordingly with *Lines* 19-20. Note that initially, when the manycast forest of c_k is empty, $S_{c_k,i}$ only includes the set of video servers that store the i-th SVC layer of c_k. After more and more clients joining in, $S_{c_k,i}$ grows with the manycast forest to include all of its nodes (as shown in *Line* 20). Therefore, we can leverage the split-and-relay feature enabled by OF to improve bandwidth utilization efficiency.

B. Layer-Based Serving With Layer Switching (LBS-LS)

Even though LBS considers the service fairness of video manycast, it still has a drawback for dynamic network operation. Basically, when *Algorithm* 1 is invoked at each provision time, it only treats the unused bandwidth as available resources. Nevertheless, this may still cause the situation in which some clients may not even get the BLs of their requested contents while others that were served earlier receive not only the BLs but also the ELs. One way to overcome this drawback is to introduce layer switching, i.e., we purposely switch off certain in-service streaming for ELs to make room for BLs. Meanwhile, in order to avoid the "ping-pong" effect, i.e., SVC layers are switched on and off frequently, we assume that if an EL is switched off, its service is terminated permanently and it would not be switched on again even when there is enough bandwidth. This is because the quality of the video will fluctuate significantly due to switching the EL(s) constantly [25], which can damage the quality of experience (QoE).

Specifically, for each request $\gamma_k(d_k, c_k, l_k, t_k)$, if we cannot find a feasible path for the BL of c_k, we will try to apply the layer switching. Then, if we select K requests that have been served before γ_k and switch off certain ELs of their video streaming for accommodating the BL of γ_k, the switching revenue is defined as follows.

Definition 4: Assuming that the selected requests are $\{\gamma'_1(d'_1, c'_1, l'_1, t'_1), \cdots, \gamma'_K(d'_K, c'_K, l'_K, t'_K)\}$, the switching revenue is the revenue gain per time-unit from the layer switching, as

$$R^s_k = \left(\frac{w_1}{\sum\limits_{j=1}^{l_k} w_j} \right) - \left(\sum_{m=1}^{K} \frac{\sum\limits_{j=i}^{l'_m a} w_i}{\sum\limits_{j=1}^{l'_m} w_j} \right), \quad i > 1 \tag{19}$$

where $j = i, \cdots, l'_m a, i > 1$ represent the ELs of request γ'_m, which we decide to switch off.

Apparently, the layer switching is meaningful only when we have $R^s_k > 0$, i.e., obtaining a positive revenue gain. *Algorithm* 2 shows the detailed procedure for the layer switching to accommodate the BL of c_k. *Lines* 2-3 are for initialization. We then use the for-loop that covers *Lines* 4-10 to calculate a path set P. Here, different from the corresponding operations in LBS, we find the paths with the maximum available bandwidth in the original topology $G(V, E, B, \Delta)$. The for-loop from *Line* 11 to *Line* 19 tries to use the layer switching to accommodate the BL of c_k with best-effort. Specifically, for each $p \in P$, we select the in-service requests such that the layer switching on them can make p available for the BL of c_k and then check whether the switching revenue is positive. We perform the layer switching only if the switching revenue is positive. By combining the layer switching strategy with LBS, we obtain the LBS-LS algorithm.

Algorithm 2: Layer Switching Strategy

```
1. if there is no feasible path for the BL of $c_k$ then
2.     obtain the information of $\gamma_{k,1}(d_k, c_k, t_k)$;
3.     $P = \emptyset$;
4.     for each node $v \in S_{c_k,1}$ do
5.         find the path $p$ in $G(V, E, B, \Delta)$ from $v$ to $d_k$ with
           the maximum available bandwidth;
6.         mark the delay of $p$ as $\delta^p$;
7.         if $\delta^p$ satisfies the delay and jitter constraints in
           (9) and (10) then
8.             $P = P \cup \{p\}$;
9.         end
10.    end
11.    for each $p \in P$ do
12.        find in-service requests such that the layer switching
           on them makes $p$ available for $\gamma_{k,1}$;
13.        calculate switching revenue $R^s_k$ with (19);
14.        if $R^s_k > 0$ then
15.            perform the layer switchingp;
16.            serve $\gamma_{k,1}(d_k, c_k, t_k)$ with $p$;
17.            break;
18.        end
19.    end
20. end
```

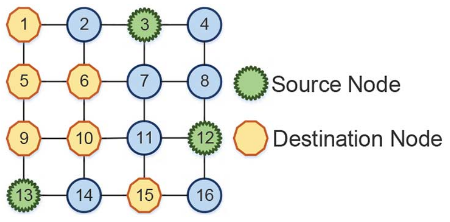


Fig. 3. 4 × 4 grid topology for one-time operation simulations.

C. Complexity Analysis

The computational complexity of LBS is $O(N_L \cdot |\Gamma| \cdot |V| \cdot (|V|^2 + |E|))$. Since LBS-LS is based on LBS and considers the layer switching as a supplement, its complexity is $O(N_L \cdot |\Gamma| \cdot (|V|^3 \cdot |E| + N_L \cdot |\Gamma| \cdot |V| \cdot (N_L + |\Gamma|)))$, which is higher than LBS. Even though the complexity of LBS-LS is higher, it also contributes more revenue gains as shown in Section V.

V. Numerical Simulations

In this section, we use numerical simulations to evaluate the performance of the ILP model and proposed heuristics. Here, we perform simulations in two categories: 1) one-time operation, and 2) dynamic operation. Considering the complexity of the ILP, we first conduct simulations in the former category to compare the results from the heuristics to the optimal ones provided by the ILP. Basically, the simulations only try to provision a set of requests and emulate the one-time operation at a provision time. While the simulations in the latter category are used to evaluate the performance of the heuristics further by considering the dynamic network operation where the requests come and leave on-the-fly and service provisioning need to be invoked repeatedly.

A. One-Time Operation

Fig. 3 shows the 4 × 4 grid topology used for small-scale simulations. Here, for simplicity, we only show the OF switches but omit the video servers and clients. If the OF switch is directly connected to a video server, we mark it as a source node in Fig. 3. On the other hand, if the client(s) can access the video streaming service through an OF switch, we mark it as a destination node. We randomly select *Nodes* 3, 12 and 13 as the source nodes, while the destination nodes are also selected randomly as *Nodes* 1, 5, 6, 9, 10 and 15. The simulation parameters are listed in Table I. We use Lingo v11.0[2] to solve the ILP and simulate the heuristics with MATLAB R2013a, and all the simulations run on a Windows server with 2.2-GHz CPU and 32-GB RAM.

Fig. 4 shows the results on the total revenue from serving different numbers of requests. For each request number, we randomly generate 50 sets of requests, perform simulation on them, and obtain the average results to plot in Fig. 4. Meanwhile, we also plot the maximum and minimum values to illustrate the statistical accuracy of the simulations. Note that here, since we only consider the one-time operation, there is no need to simulate LBS-LS, which is designed for dynamic network operation. As expected, the ILP provides the largest total revenues, but we

[2][Online]. Available: http://www.lindo.com/

TABLE I
SIMULATION PARAMETERS

Parameters	One-Time Operation	Dynamic Operation
$\delta_{(u,v)}$, Delay on links (time-units)	1	$[2, 6]$
$b_{(u,v)}$, Bandwidth on links (units)	4	$[60, 120]$
$\|S\|$, Number of source nodes	3	8
$\|D\|$, Number of destination nodes	6	45
$\|C\|$, Number of video sources/contents	4	100
N_L, Maximum number of SVC layers per content	2	3
δ_{max}, Maximum delay (time-units)	10	130
σ_{max}, Maximum jitter (time-units)	3	30
Bandwidth of each BL (units)	1	$[1, 3]$
Bandwidth of each EL1 (units)	3	$[4, 6]$
Bandwidth of each EL2 (units)	–	$[6, 8]$
w_1, w_2, w_3, Revenue weights [20]	8, 1, –	8, 1, 1

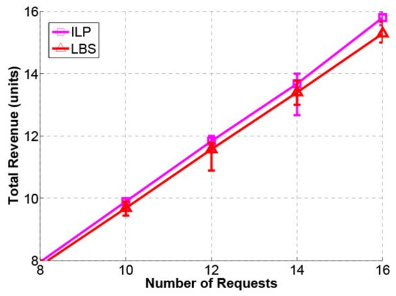


Fig. 4. Simulation results on total revenue for one-time operation.

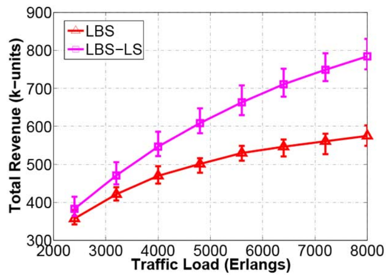


Fig. 5. Simulation results on total revenue for dynamic operation.

TABLE II
SIMULATION RESULTS ON AVERAGE RUNNING TIME (SECONDS)

# of Requests	ILP	LBS
8	239.85	0.22
10	375.55	0.24
12	537.82	0.31
14	668.37	0.32
16	1247.6	0.35

can also see that the total revenues from LBS are close to those from the ILP.

In order to check whether the algorithms are suitable for being practically implemented in the OF controller, we record their average running time and Table II shows the results.

It can be seen that due to its complexity, the ILP needs very long time to finish the computation, which makes it not suitable for managing real-time network operations. LBS is three magnitudes faster than the ILP. Hence, the results suggest that LBS provides a reasonably good tradeoff between the time complexity and the performance of service provisioning.

B. Dynamic Operation

We then perform simulations for dynamic operation in a 14×14 grid topology (i.e., including 196 nodes) to evaluate the performance of the heuristics further. The locations of the source and destination nodes are still randomly selected and the details on the simulation parameters are also listed in Table I. Here, the requests are randomly generated according to the Poisson traffic model, i.e., the average arrival rate is λ and the holding time of each request follows the negative exponential distribution with an average of $\frac{1}{\mu}$. Hence, we can quantify the traffic load as $\frac{\lambda}{\mu}$ in Erlangs. In each simulation, we fix the traffic load, generate the requests that come and leave on-the-fly, and use the heuristics to serve them dynamically. For each traffic load, we simulate around 16,000 requests and compare LBS and LBS-LS. To emulate the practical scenario, we also add random background traffic in the network.

Fig. 5 shows the results on total revenue, each of which is obtained by averaging the results from 50 simulations. We still plot the maximum and minimum values to show the statistical accuracy. We observe that LBS-LS performs better than LBS. This is because LBS-LS manages the bandwidth allocation intelligently with the layer switching and serves the most clients with BL streaming. According to (1), the revenue gain from streaming the BL is higher than that from streaming the subsequent ELs. Hence, by making the service provisioning fairer, LBS-LS provides the larger total revenues. It is interesting to notice that when the traffic load is relatively low (e.g., 2400 Erlangs), there is no significant difference among the results from the two heuristics. This is because when the traffic load is low, the network capacity is sufficient enough to accommodate almost all the requests.

VI. EXPERIMENTAL DEMONSTRATIONS

In this section, we discuss the system implementation of the OF-controlled network orchestration for adaptive SVC video manycast and show the experimental setup and results.

A. System Implementation

We implement the proposed heuristics in the OF controller, design necessary OF protocol extensions, and realize a practical OF network system for adaptive SVC video manycast. Specifically, we implement the OF controller with the POX platform[3] and run it on a Linux server (Lenovo ThinkServer RD540). It is known that POX platform is relatively fast and efficient and can handle over 30,000 flows per second . Meanwhile, with the message bundling scheme developed for the latest OF standard,

[3][Online]. Available: http://www.noxrepo.org/pox/about-pox/

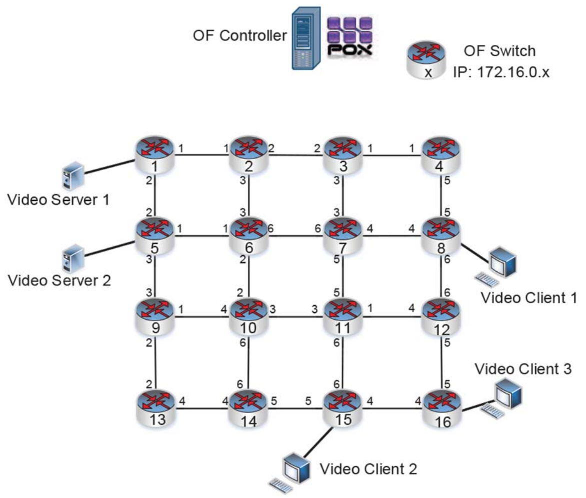


Fig. 6. Network topology of experimental testbed.

the OF controller can bundle a set of OF messages to a switch in a single message to reduce the signaling time. Therefore, we can see that the proposed system has relatively good scalability. The OF switches are programmed based on OpenvSwitch,[4] and each of them is also realized with a high-performance Linux server (Lenovo ThinkServer RD530 or RD540) equipped with multiple Ethernet linecards.

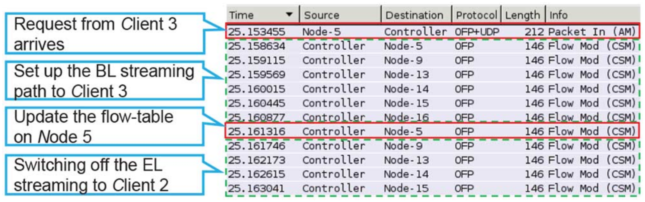
We realize the video servers and clients with common personal computers (PCs) running open-source softwares that are modified to support adaptive SVC video streaming. For instance, the client should decode the received video correctly when we reconfiguring the EL streaming dynamically. The video server uses the joint scalable video mode (JSVM) module for SVC transcoding. When a streaming session starts, the server encapsulates the video data in packets with corresponding RTP/UDP/IP headers and sends them out. The OF switches forward the packets by checking the IP addresses and UDP ports and applying the forwarding actions provided by the OF controller. Hence, for the streaming of the same content, the packets for difference SVC layers are distinguished with the combination of IP addresses and UDP port. Hence, they can be forwarded over the OF network independently, even though the source and destination IP addresses are the same. In each client, the packets for difference SVC layers are redirected to the same UDP port and aggregated. The clients buffer received packets for reordering and decoding, and play back the video. In the experiments, we record all the reconstructed video frames and relevant event-logs for performance analysis.

B. Experimental Testbed

We build an experimental testbed for the OF network, which consists of 16 OF switches connected according to the 4×4 grid topology as shown in Fig. 6, where the numbers around the OF switches are the port numbers. The OF controller connects to all the OF switches directly. We insert two video servers in the testbed and make them connect to *Nodes* 1 and 5, and the video clients are connected to *Nodes* 8, 15 and 16. Two video contents are placed on the video servers to emulate the real-time

TABLE III
MINIMUM BIT-RATES OF VIDEO STREAMING (MBPS)

SVC layer	Video Content 1	Video Content 2
BL	0.41	0.60
EL	1.19	0.76


Request from Client 3 arrives
Set up the BL streaming path to Client 3
Update the flow-table on Node 5
Switching off the EL streaming to Client 2

```
Time       Source      Destination  Protocol  Length  Info
25.153455  Node-5      Controller   OFP+UDP   212 Packet In  (AM)
25.158634  Controller  Node-5       OFP       146 Flow Mod  (CSM)
25.159115  Controller  Node-9       OFP       146 Flow Mod  (CSM)
25.159569  Controller  Node-13      OFP       146 Flow Mod  (CSM)
25.160015  Controller  Node-14      OFP       146 Flow Mod  (CSM)
25.160445  Controller  Node-15      OFP       146 Flow Mod  (CSM)
25.160877  Controller  Node-16      OFP       146 Flow Mod  (CSM)
25.161316  Controller  Node-5       OFP       146 Flow Mod  (CSM)
25.161746  Controller  Node-9       OFP       146 Flow Mod  (CSM)
25.162173  Controller  Node-13      OFP       146 Flow Mod  (CSM)
25.162615  Controller  Node-14      OFP       146 Flow Mod  (CSM)
25.163041  Controller  Node-15      OFP       146 Flow Mod  (CSM)
```

Fig. 7. OF messages for adaptive SVC video manycast.

video sources, and we mark them as *Content* 1 and 2, which are generated by using the MPEG standard test-sequence "Bridge" and "Paris",[5] respectively. Both of the videos are encoded in the common-intermediate-format (CIF) resolution (352×288), and last for 35 seconds (i.e., 1,064 frames). We play the videos multiple times to get longer video sessions. For each of the video contents, JSVM on the servers performs SVC transcoding and generates two SVC layers (i.e., BL and EL) for different playback signal-to-noise-ratios (SNRs). The bit-rates of the SVC layers (minimum values) are shown in Table III. In the experiments, we have *Content* 1 on the server connected to *Node* 1 and *Content* 2 on the server connected to *Node* 5.

We set up a bandwidth-limited network by limiting the capacity of each link that connects two OF switches as 1.7 Mbps. We implement LBS and LBS-LS in the OF controller and perform packet-level experiments to compare them. Specifically, we collect experimental results on streaming bandwidth, packet-loss-rate (PLR), luminance component's peak signal-to-noise-ratio (Y-PSNR) of video playback, and delay jitter.

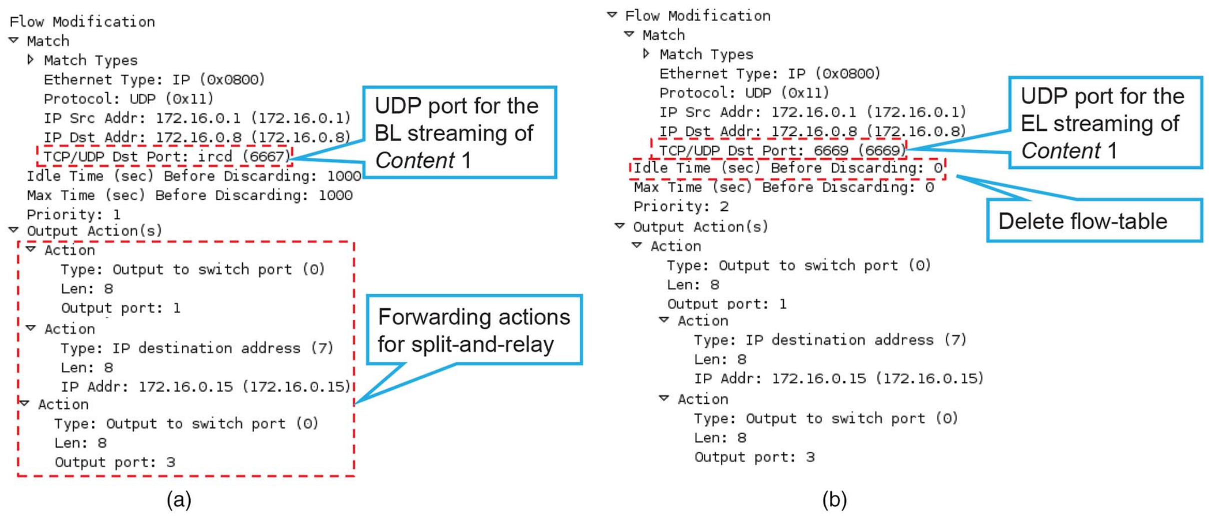
C. Experimental Procedure

In order to verify that the OF network can realize the network orchestration for adaptive SVC video manycast, we compare the performance of LBS and LBS-LS. Meanwhile, we also realize the conventional IP multicast scenario and use it as the benchmark. The experimental procedure are as follows.

- *Step 1*: At $t = 1$ second, a background traffic with the capacity of 0.2 Mbps is generated to use link $1 \rightarrow 2$, which will last for 34 seconds. Then, the available bandwidth on link $1 \rightarrow 2$ becomes 1.5 Mbps. *Client* 1 on *Node* 8 sends a request for *Content* 1 and asks for both the BL and EL. *Node* 8 forwards the request to the OF controller, which selects the server connected to *Node* 1 as the streaming source and obtains the streaming paths for the BL and EL as $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 8$ and $1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$, respectively. However, in the Conventional scenario, the paths for the BL and EL are both $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 8$, which is the shortest path. Then, the streaming service for *Client* 1 is set up for both of the requested SVC layers. Note that since the total capacity of the BL and EL of *Content* 1 is about 1.6 Mbps, the provisioning scheme from the Conventional scenario will cause congestion on link $1 \rightarrow 2$.

[4][Online]. Available: http://www.openvswitch.org/

[5][Online]. Available: http://trace.eas.asu.edu/yuv/index.html



```
Flow Modification
 ▽ Match
   ▷ Match Types
     Ethernet Type: IP (0x0800)
     Protocol: UDP (0x11)
     IP Src Addr: 172.16.0.1 (172.16.0.1)
     IP Dst Addr: 172.16.0.8 (172.16.0.8)
     TCP/UDP Dst Port: ircd (6667)
   Idle Time (sec) Before Discarding: 1000
   Max Time (sec) Before Discarding: 1000
   Priority: 1
 ▽ Output Action(s)
   ▽ Action
       Type: Output to switch port (0)
       Len: 8
       Output port: 1
   ▽ Action
       Type: IP destination address (7)
       Len: 8
       IP Addr: 172.16.0.15 (172.16.0.15)
   ▽ Action
      Type: Output to switch port (0)
      Len: 8
      Output port: 3
```

UDP port for the BL streaming of *Content* 1
Forwarding actions for split-and-relay
(a)

```
▽ Flow Modification
  ▽ Match
    ▷ Match Types
      Ethernet Type: IP (0x0800)
      Protocol: UDP (0x11)
      IP Src Addr: 172.16.0.1 (172.16.0.1)
      IP Dst Addr: 172.16.0.8 (172.16.0.8)
      TCP/UDP Dst Port: 6669 (6669)
    Idle Time (sec) Before Discarding: 0
    Max Time (sec) Before Discarding: 0
    Priority: 2
  ▽ Output Action(s)
    ▽ Action
        Type: Output to switch port (0)
        Len: 8
        Output port: 1
      ▽ Action
          Type: IP destination address (7)
          Len: 8
          IP Addr: 172.16.0.15 (172.16.0.15)
      ▽ Action
          Type: Output to switch port (0)
          Len: 8
          Output port: 3
```

Fig. 8. Details on *FlowMod* messages.

- *Step 2*: At $t = 10$ seconds, another background traffic is generated to use link $3 \to 7$, which will last for 30 seconds, and the available bandwidth on link $3 \to 7$ becomes 1.5 Mbps. *Client* 2 on *Node* 15 tries to join the video streaming for *Content* 1 and also sends a request for the BL and EL. This time, the OF controller finds that *Client* 2 can directly get the BL streaming from *Node* 3 based on manycast scenario, and instructs the OF switch on *Node* 3 to perform split-and-relay. Meanwhile, the OF controller set up the path $1 \to 5 \to 9 \to 13 \to 14 \to 15$ for the EL streaming. Then, the manycast forest of *Content* 1 becomes $\{\{\text{BL}: 1 \to 2 \to 3 \to 4 \to 8, 3 \to 7 \to 11 \to 15\}$, $\{\text{EL}: 1 \to 5 \to 6 \to 7 \to 8, 5 \to 9 \to 13 \to 14 \to 15\}\}$. The Conventional scenario selects $1 \to 2 \to 3 \to 7 \to 11 \to 15$ as the path for both the BL and EL and creates congestion on link $3 \to 7$.
- *Step 3*: At $t = 24$ seconds, 1.2 Mbps background traffic is injected to links $6 \to 10$ and $9 \to 10$, lasting for 10 seconds. Then, the available bandwidth on links $6 \to 10$ and $9 \to 10$ becomes 0.5 Mbps. *Client* 3 on *Node* 16 tries to join the video streaming for *Content* 2 and sends a request for the BL. With LBS, the OF controller selects the server connected to *Node* 5 as the streaming source and obtains the streaming path as $5 \to 9 \to 13 \to 14 \to 15 \to 16$.[6] However, as the available bandwidth on path $5 \to 9 \to 13 \to 14 \to 15$ is 1.7 Mbps at that time, the service provisioning scheme will make it become congested. On the other hand, with LBS-LS, the OF controller switches off the EL streaming to *Client* 2 on path $5 \to 9 \to 13 \to 14 \to 15$ to make room for the BL streaming of *Client* 3. Then, there will be no congestion on path $5 \to 9 \to 13 \to 14 \to 15$. The Conventional scenario selects the path for the BL streaming as $5 \to 9 \to 13 \to 14 \to 15$.

[6]Here, strictly speaking, LBS should just block the request from *Client* 3 as no feasible path can be found due to the bandwidth limitation. The reason why we still make LBS to provision the video streaming is that we want to see how the quality of video streaming would be affected by network congestions.

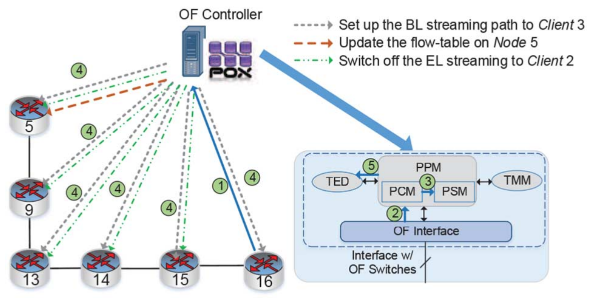

OF Controller
POX
Set up the BL streaming path to *Client* 3
Update the flow-table on *Node* 5
Switch off the EL streaming to *Client* 2
1
2
3
4
5
5
9
13
14
15
16
TED
PPM
TMM
PCM
PSM
OF Interface
Interface w/ OF Switches

Fig. 9. Procedure of service provisioning for *Client* 3.

- *Step 4*: At $t = 35$ seconds, the real-time video streaming of *Content* 1 ends.
- *Step 5*: At $t = 55$ seconds, the real-time video streaming of *Content* 2 ends.

D. OF Messages

Fig. 7 shows the Wireshark capture of the OF messages used for provisioning the request from *Client* 3 with LBS-LS. When the request from *Client* 3 arrives, the OF controller switches off the EL streaming of *Client* 2 to make room for the BL streaming of *Client* 3. Therefore, we can see that the flow-table on *Node* 5 is updated and those on *Nodes* 9, 13, 14 and 15 are modified. We also capture and parse the *FlowMod* messages to verify the operations in the OF network. Fig. 8(a) shows the *FlowMod* message received on *Node* 3 to provision the BL streaming to *Clients* 1 and 2. We can see that the OF controller instructs the OF switch to identify the packets of the BL streaming with the combination of source and destination IP addresses and UDP port number. The *FlowMod* message also shows that at *Node* 3, the split-and-relay is realized by using three forwarding actions. Basically, *Node* 3 switches the packets to *Outputs* 1 and 3, and by checking the topology in Fig. 6, we can find that *Outputs* 1 and 3 of *Node* 3 connect to *Nodes* 4 and 7, respectively. Also, when the streaming packets are duplicated, the OF switch on *Node* 3 will modify its destination IP address to that of *Node*

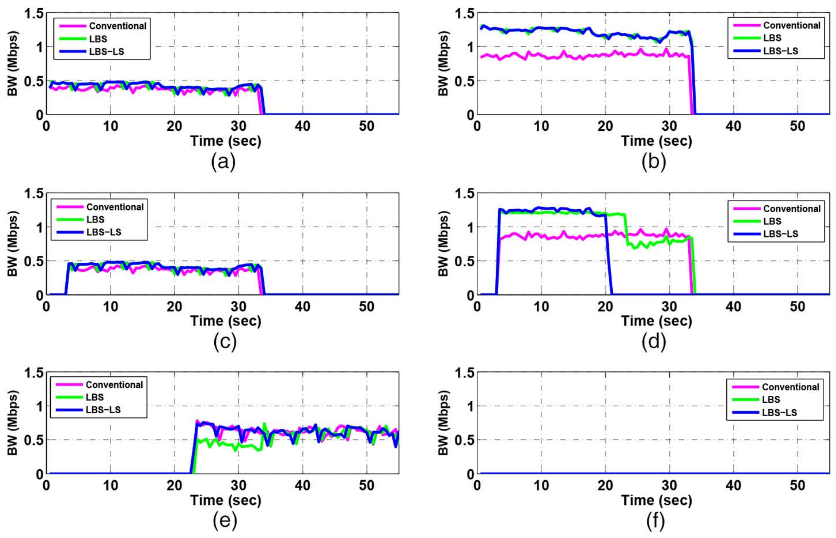


Fig. 10. Results on streaming bandwidth.

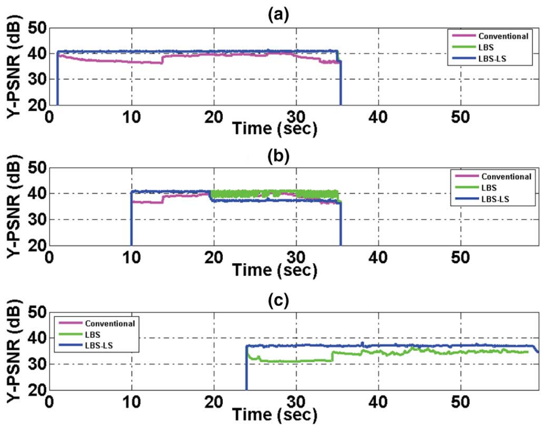


Fig. 11. Results on Y-PSNR of the received videos. (a) Client 1. (b) Client 2. (c) Client 3.

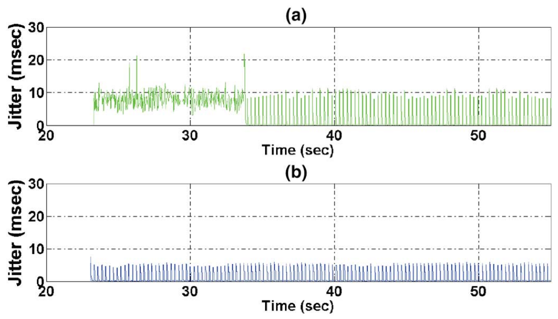


Fig. 12. Results on packet-level delay jitter on *Client* 3. (a) LBS. (b) LBS-LS.

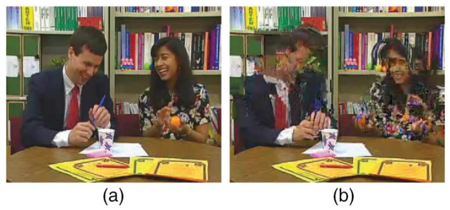


Fig. 13. Screen-shots of the received video on *Client* 3. (a) LBS-LS. (b) LBS.

15 (i.e., 172.16.0.15), which is the OF switch that connects to *Client* 2. The *FlowMod* message that is used to switch off the EL streaming to *Client* 2 on *Node* 5 is presented in Fig. 8(b), in which the OF controller sets the *IdleTime* field as 0 and lets the flow-table for the EL streaming packets become invalid.

Finally, we use Fig. 9 to explain the operation details in the OF controller. Here, we take the service provisioning for *Client* 3 as an example, and make the explanation as follows.

- *Step 1*: *Node* 16 sends a *PacketIn* message to the OF controller for *Client* 3.
- *Step 2*: The OF controller receives the *PacketIn* message. PPM parses the request information and instructs PCM to calculate the service provisioning scheme. With the network status in TED and TMM, PCM tries to obtain the service provisioning scheme for the BL.
- *Step 3*: Due to bandwidth insufficiency, PCM cannot find a feasible path for the BL streaming, and hence it contacts PSM to invoke the layer switching. PSM figures out that by switching off the EL streaming to *Client* 2 and reconfiguring the existing manycast forest, the BL streaming to *Client* 3 can be accommodated.
- *Step 4*: PPM encodes the new forwarding actions in *FlowMod* messages and sends them to the related OF switches. The OF switches update their flow-tables to provision the streaming service to *Client* 3.
- *Step 5*: The OF controller updates TED and TMM to include the most-updated network status.

E. Experimental Results

Fig. 10 shows the experimental results on streaming bandwidth. The streaming bandwidths for the BL and EL of *Client* 1 are plotted in Figs. 10(a) and 10(b). In Fig. 10(a), we observe that for both LBS and LBS-LS, the streaming bandwidth for the BL stays almost unchanged after $t = 10$ seconds, when *Client* 2 joins and starts to receive the BL from *Node* 3 with the manycast scenario. This verifies that the OF-controller can adjust the manycast forest without affecting the in-service streaming. While in the Conventional scenario, the streaming bandwidths for both the BL and EL are less than those in LBS and LBS-LS

TABLE IV
STATISTICS OF EXPERIMENTAL RESULTS

Client	Scenario	PLR (%)	Bandwidth (Mbps)		Y-PSNR (dB)		Delay Jitter (msec)	
		Average	Average	Variance	Average	Variance	Average	Variance
Client 1	Conventional	0.06	0.38	0.03	38.28	1.24	7.91	4.61
	LBS	0	0.42	0.05	40.73	0.41	0.06	0.04
	LBS-LS	0	0.42	0.05	40.74	0.46	0.05	0.05
Client 2	Conventional	0.06	0.37	0.04	38.52	1.23	7.91	4.60
	LBS	0	0.42	0.05	40.02	0.98	0.07	0.06
	LBS-LS	0	0.42	0.05	38.58	1.73	0.07	0.06
Client 3	Conventional	0	0.62	0.08	36.99	0.30	1.17	2.98
	LBS	0	0.55	0.10	33.60	1.58	3.76	5.13
	LBS-LS	0	0.62	0.08	36.99	0.30	1.17	2.98

during the whole experimental, due to the insufficient bandwidths on links $1 \rightarrow 2$ and $3 \rightarrow 7$. That is because the Conventional scenario does not have the global information regarding the network status and hence cannot facilitate dynamic path setup for video streaming.

Fig. 10(b) also indicates the difference between LBS and LBS-LS. Basically, when *Client* 3 tries to join at $t = 24$ seconds, LBS-LS switches off the EL streaming to *Client* 2 to make room on path $5 \rightarrow 9 \rightarrow 13 \rightarrow 14 \rightarrow 15$, while LBS does nothing to *Client* 2. We can also see that with LBS, the streaming bandwidth for EL decreases for about 0.2 Mbps due to the network congestion. Figs. 10(c) and 10(d) show the streaming bandwidths for the BL and EL to *Client* 2, which stay almost unchanged during the whole provisioning period, i.e., $t \in [10, 35]$ seconds. Even though *Clients* 1 and 2 receive the same video, the streaming bandwidth for the EL to *Client* 1 is not affected when LBS-LS switches off the EL to *Client* 2 at $t = 24$. In Fig. 10(e), we observe that with LBS, the streaming bandwidth for the BL to *Client* 3 decreases for about 0.2 Mbps during $t \in [24, 35]$ seconds. This is still due to the network congestion on path $5 \rightarrow 9 \rightarrow 13 \rightarrow 14 \rightarrow 15$. Basically, the network congestion causes packet loss and makes the effective streaming bandwidth decline.

Fig. 11 illustrates the results on Y-PSNRs of the received videos. In Fig. 11(a), we can see that with LBS-LS, the Y-PSNR of *Content* 1 during $t \in [24, 35]$ seconds is around 4 dB less than that before $t = 24$ seconds. This is because LBS-LS switches off the EL streaming to accommodate the request from *Client* 3. While for the case with LBS, even though the results on Y-PSNR is slightly better during the same period, the network congestion makes the Y-PSNR unstable. Fig. 11(c) shows that for *Client* 3, the results on Y-PSNR from LBS is significantly worse than those from LBS-LS. We also notice that even after $t = 35$ seconds, when the video streaming of *Content* 1 has already ended, the results on Y-PSNR from LBS are still worse than those from LBS-LS. This is caused by the chain effect from the congestion-induced packet loss.

Fig. 12 shows the results on the packet-level delay jitter of *Client* 3. With LBS-LS, the delay jitter in Fig. 12(b) stays at around 3 msec stably, while the results from LBS are much larger [as shown in Fig. 12(a)]. Due to the congestion on path $5 \rightarrow 9 \rightarrow 13 \rightarrow 14 \rightarrow 15$ during $t \in [24, 35]$ seconds, the jitter results from LBS are the largest. In Fig. 12(a), after $t = 35$ seconds, when the video streaming of *Content* 1 ends, the bandwidth on these links is sufficient to serve *Client* 3, the jitter results decrease. We capture two screen-shots of the video playback on *Client* 3 in Fig. 13 to compare the actual streaming quality illustratively. The statistics of the experimental results are listed in Table IV, which indicate that LBS-LS also outperforms LBS and the Conventional scenario in long-run.

VII. CONCLUSION

In this paper, we investigated how to realize network orchestration for adaptive SVC video manycast and proposed to realize it with an OF-controlled SDN architecture. The functional modules for the OF controller were first designed to facilitate efficient network operations. Then, we focused on solving the multi-source multi-destination SVC video manycast problem efficiently and designed several algorithms. Initially, an ILP model was formulated to obtain the optimal solutions for small-scale problems. Next, we tried to reduce the computational complexity and make the manycast algorithm suitable for practical implementation in the OF controller, and designed two time-efficient heuristics, namely, LBS and LBS-LS. The ILP and heuristics were evaluated with numerical simulations and the results indicated that the heuristics could provide close-to-optimal solutions for the manycast problem and the one that incorporates the layer switching (i.e., LBS-LS) provided the best performance on total revenue gain in the simulations for dynamic operations.

Finally, we built an OF network testbed that consisted of OF switches, SVC video servers and clients. With the testbed, we designed necessary OF protocol extensions, implemented the heuristics in the OF controller, and performed SVC video streaming experiments to demonstrate our design. In the experiments, we collected the results on streaming bandwidth, packet-loss-rate (PLR), luminance component's peak signal-to-noise-ratio (Y-PSNR) of video playback, and delay jitter, and they verified that the proposed scheme could allocate bandwidth intelligently and ensure high-quality video streaming. Basically, the centralized OF controller could manage the SVC video manycast efficiently by adjusting the manycast forest dynamically according to the network status. Hence, the proposed networking system could effectively improve the quality of real-time SVC video streaming.

ACKNOWLEDGMENT

The authors would like to thank Mr. H. Huang from the University of Science and Technology of China for his help on the experiments.

REFERENCES

[1] "Cisco visual networking index: Forecast and methodology, 2012–2017," Cisco. San Jose, CA, USA, May 2013.

[2] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[3] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," in *Proc. INFOCOM*, Apr. 2002, pp. 1736–1745.

[4] K. Phan, J. Moulierac, C. Tran, and N. Thoai, "Xcast6 treemap islands: Revisiting multicast model," in *Proc. ACM CoNEXT Workshop*, Dec. 2012, pp. 33–34.

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

[6] G. Goth, "Software-defined networking could shake up more than packets," *IEEE Internet Comput.*, vol. 15, no. 4, pp. 6–9, Jul./Aug. 2011.

[7] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[8] D. Kotani, K. Suzuki, and H. Shimonishi, "A design and implementation of openflow controller handling IP multicast with fast tree switching," in *Proc. IEEE IPSJ*, Jul. 2012, pp. 60–67.

[9] Y. Yang, Z. Qin, X. Li, and S. Chen, "OFM: A novel multicast mechanism based on Openflow," *Adv. Inform. Sci. Service*, vol. 4, pp. 278–286, Sep. 2012.

[10] J. Zou, G. Shou, Z. Guo, and Y. Hu, "Design and implementation of secure multicast based on SDN," in *Proc. IC-BNMT*, Nov. 2013, pp. 124–128.

[11] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *Proc. COMSNETS*, Jan. 2014, pp. 1–8.

[12] H. Egilmez, B. Gorkemli, A. Tekalp, and S. Civanlar, "Scalable video streaming over OpenFlow networks: An optimization framework for QoS routing," in *Proc. ICIP*, Sep. 2011, pp. 2241–2244.

[13] H. Egilmez, S. Civanlar, and A. Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over OpenFlow networks," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 710–715, Apr. 2013.

[14] H. Egilmez, S. Civanlar, and A. Tekalp, "A distributed QoS routing architecture for scalable video streaming over multi-domain OpenFlow networks," in *Proc. ICIP*, Sep. 2012, pp. 2237–2240.

[15] H. Egilmez and A. Tekalp, "Distributed QoS architectures for multimedia streaming over software defined networks," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1597–1609, Oct. 2014.

[16] Q. She and J. Jue, "Min-cost tree for multi-resource manycast in mesh networks," in *Proc. ANTS*, Dec. 2007, pp. 1–2.

[17] J. Chen, S. Chan, and V. Li, "Multipath routing for video delivery over bandwidth-limited networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 10, pp. 1920–1932, Dec. 2004.

[18] A. Krishnamurthy, T. Little, and D. Castanon, "A pricing mechanism for scalable video delivery," *Multimedia Syst.*, vol. 4, pp. 328–337, 1996.

[19] C. Wang, Y. Chen, H. Wei, and K. Liu, "Optimal pricing in stochastic scalable video coding multicasting system," in *Proc. INFOCOM*, Apr. 2013, pp. 540–544.

[20] W. Zhang *et al.*, "Reliable adaptive multipath provisioning with bandwidth and differential delay constraints," in *Proc. INFOCOM*, Apr. 2010, pp. 1–9.

[21] Z. Zhu, S. Li, and X. Chen, "Design QoS-aware multi-path provisioning strategies for efficient cloud-assisted SVC video streaming to heterogeneous clients," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 758–768, Jun. 2013.

[22] S. Li, Z. Zhu, W. Li, and H. Li, "Efficient and scalable cloud-assisted SVC video streaming through mesh networks," in *Proc. ICNC*, Jan. 2012, pp. 944–948.

[23] P. Ho, J. Tapolcai, and T. Cinkler, "Segment shared protection in mesh communications networks with bandwidth guaranteed tunnels," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 1105–1118, Dec. 2004.

[24] G. Baier, E. Kohler, and M. Skutella, "The k-splittable flow problem," *Algorithmica*, vol. 42, pp. 231–248, Jul. 2005.

[25] D. Song and C. W. Chen, "Scalable H.264/AVC video transmission over MIMO wireless systems with adaptive channel selection based on partial channel information," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1218–1226, Sep. 2007.

Nana Xue, photograph and biography not available at the time of publication.

Xiaoliang Chen, photograph and biography not available at the time of publication.

Long Gong, photograph and biography not available at the time of publication.

Suoheng Li, photograph and biography not available at the time of publication.

Daoyun Hu, photograph and biography not available at the time of publication.

Zuqing Zhu (S'02–M'06–SM'12), photograph and biography not available at the time of publication.