Liv(e)-ing on the Edge: User-Uploaded Live Streams Driven by "First-Mile" Edge Decisions

Jiasi Chen
University of California, Riverside
jiasi@cs.ucr.edu

Bharath Balasubramanian, Zhe Huang
AT&T Labs Research
{*bharathb,zhehuang*}*@research.att.com*

***Abstract*—As users equipped with high-resolution cameras spontaneously capture and live-stream videos to interested parties through distributors like Twitch or Facebook Live, a new style of video delivery is emerging: user-upload streaming. Unlike video-on-demand and traditional live-streaming in which video is stored by professional content providers and distributed using standard CDN techniques, in user-upload streaming, the video is generated and uploaded on-the-fly by heterogeneous clients with restricted bandwidth and long latencies to the distributor. Hence, it is crucial to factor in client-side "first-mile" factors when making edge decisions. In this paper, we present a systematic design of a video delivery architecture for user-upload streaming, which focuses on the upload server and upload bitrate as the important "first-mile" edge factors that influence downstream delivery and quality-of-experience for viewers. We present a polynomial-time algorithm to minimize the end-to-end latency and maximize the video rate for all users for the scenario where the upload and download server are the same. Further, we present efficient heuristics for the general (NP-complete) version where the upload streams are routed through the distributor's overlay network. Finally, we validate the efficacy of our algorithms through extensive trace-based simulations based on real-world data sets.**

***Keywords*-live streaming, video, edge networks**

I. Introduction

As increasing number of users armed with powerful smartphone cameras or GoPros spontaneously upload live streams such as graduation ceremonies and tourist sites, live-streaming is no longer the privilege of just a few select content providers. In response to this trend (that we refer to as user-upload streaming or crowd streaming) companies like Meerkat, Facebook (Live), twitch.tv and Google (YouTube Live) have all released mobile phone apps that enable live-streaming. While such upload streams are increasing at a prolific rate (1; 2), the constraints of video delivery such as high latency and constrained bandwidth remain challenging, particularly in the "first-mile" between the uploader and the distribution network. Furthermore, the emerging use-cases of AR/VR and 360° video will exacerbate the situation due to the high volume of video data being uploaded. Based on this, the main question we ask in this paper is: *What is the impact of "first-mile" edge decisions made by the uploader on the quality-of-experience of downstream viewers, and how should these edge decisions be made?*

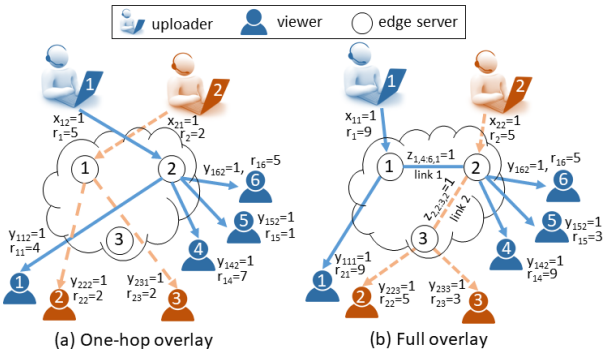


Figure 1: Two architectures for user-upload streaming. (a) One-hop-overlay architecture (e.g., Skype): the uploader and downloaders use the same edge server. (b) Full-overlay architecture (e.g., Twitch, Facebook and Akamai): uploaded video is routed through an Internet overlay. First-mile decisions (upload server and upload video bitrate) impact downstream decisions and viewer QoE.

There has been excellent work in both industry and academia on video-on-demand (VoD) delivery and large-scale live-streaming architectures (3; 4; 5; 6). Most VoD frameworks are based on traditional CDN techniques where content is co-located or located close to a CDN point of presence (PoP). These works have considered "last-mile" mapping between viewers and the CDN (3; 4), and centralized routing control within the CDN (6). Similarly, most frameworks for planned live-streaming of large-scale events (e.g., sports games, presidential speeches) consider the problem of CDNs deciding how to route their content to a reasonably predictable user-base (5). We claim that these architectures cannot be directly applied to user-upload streaming due to the following challenges:

1. When content is being generated spontaneously by end users (rather than pre-planned large-scale events such as sports games), we can no longer depend on content being co-located with the distribution network, or high-bandwidth and low-latency connections between the content source and the distribution network. The users uploading content will mostly have imperfect,

heterogeneous "first-mile" connections with potentially low bandwidth and high latency. Hence, these uploader edge decisions must be taken into account.

2. Both VoD and live-streaming plan and provision for content delivery based on *a priori* estimation of the usage patterns, the weaker links in the network, and other relevant parameters. In the case of user-upload streaming, this is tough to do because of the unplanned nature of uploads and the resulting user-base. Hence, a more nimble approach that periodically re-optimizes based on the current set of flows is required.

3. Since traditional video-delivery techniques have to contend with a huge viewer-base (even millions) spread across the world, they often have to use heavy-weight techniques such as reflectors in the network. This is overkill for user-upload streaming in which the viewer base is in the order of thousands for the most popular streamers. Hence, an architecture that is designed for current live-streaming environments is required (7).

To address these challenges, we focus on two crucial edge decisions: *upload server selection* and *video encoding bit rate*, for each user uploading a video. Clearly, it is important to upload the live stream to the appropriate server at the appropriate bit rate that will minimize the average latency and maximize the video rate experienced by viewers across all streams served by the distributor. While popular live-streaming platforms such as the Open Broadcaster Software (8) do provide options for the streamer to select the upload server and bit rate selection, no guidance is provided. Instead, streamers must rely on third-party applications (e.g., (9)) and heuristics to manually select these parameters, leading to globally suboptimal solutions.

In this paper, we consider the problem of the distributor (e.g., Twitch, Meerkat or Facebook) looking at the current set of uploader and downloaders (who are uploading and consuming videos during a certain window of time) and solving a holistic problem that minimizes the *end-to-end latency* and maximizes the *video rate* experienced by viewers, based on the choice of upload server and upload bitrate for each uploader. The solution to this problem can guide uploaders on how to select upload parameters that balance between the quality-of-experience of their own viewers as well as other streams utilizing the same distribution infrastructure.

Inspired by the measurements in (7), we solve this problem for two prevalent system architectures as shown in Fig. 1. In the *One-hop-overlay* architecture, the uploader and viewer use the same edge server. The streams traverse an overlay on the public Internet and the distributor is not attempting to route the streams through their overlay networks (a simple analysis of the IP addresses of the packets in a Skype conference, indicate that Skype uses this architecture). On the other hand, in the *Full-overlay* architecture (favored by Twitch, Facebook Live and Akamai (10; 6; 11; 12)) the distributor routes streams through its overlay network. Hence, apart from the upload server and bit rate, they can also control the flow rates through the overlay links.

Existing works on video-conferencing (13; 14) and crowd-sourced live streaming (1; 2) focus on effective placement of transcoders that transmit the streams and transform bitrates across the users, often utilizing the cloud's elasticity and agility. While transcoder placement is important and complementary to our work, we focus on a crucial element for quality-of-experience: the "first-mile" link between the uploader (the user) and the distributor, and we provide optimal algorithms for selecting the upload server and bitrate. Since upload server selection impacts the downstream routing and downlink server selection we cannot simply choose the closest server, as is commonly done for download server selection (15).

In summary, we make the following contributions:

- We formulate a problem for user-upload streaming based on current distribution architectures, where we maximize the quality-of-experience of all users in the system based on crucial "first-mile" choices of upload server and upload bit rate.
- For a One-hop-overlay system architecture, we develop an optimal polynomial-time algorithm for this problem based on a novel reduction to the network flow problem that considers server load-balancing constraints (Section III-A).
- For the general Full-overlay system architecture, we prove that the problem is NP-Complete and provide an efficient heuristic that iteratively solves for upload bit rate, routing and server selection. We also provide an adaption algorithm to minimize the perturbations to the current solutions during the arrival and departure of the video uploaders and viewers (Sec. III-B).
- We demonstrate our algorithm efficacy through simulations based on real-world Twitch datasets (16), comparing them to strawman algorithms that choose the closest server to upload based on RTT (Sec. IV).

II. Problem Formulation

System model. In this section, we formulate the general problem that captures the important aspects of user-upload streaming. In our problem setting, we assume a set of independent uploaders, each streaming a video to a set of viewers. The video is distributed through the overlay network of the live streaming service (e.g., Meerkat, Facebook). Every uploader uploads his/her stream to an *upload edge server*, which transcodes the video into a set of bitrates for viewing. Each upload server can only accept a finite number of streams, due to the limited amount of compute resources available

for transcoding. The transcoded streams are then routed through the overlay network to one or more *edge download servers*, from which the viewers download the stream. The amount of bandwidth available in the overlay network depends on the service-level agreements between the live-streaming service and the ISP, as well as the amount of non-streaming traffic present.

The decisions to be made are as follows. Uploader u needs to decide which bitrate r_u to use, and whether to select upload server i, represented by the binary variable x_{ui}. The overlay network must decide whether stream u destined for viewer v should be sent on link l, using the binary variable z_{uvl}. Finally, viewer v watching stream u must decide which rate to download at, r_{uv}, and which download server i to choose, y_{uvi}. The main objective in these decisions is to minimize the total latency and maximize the viewer-perceived video rate across all the viewers in this system.

Core Problem. We capture our system model and the distributor decisions succinctly in Problem 1 and summarize its symbols in Table I.

Problem 1: General problem

$$\text{minimize} \quad \sum_{u\in U}\sum_{v\in V_u}\left(\sum_i h_i^{\text{up}}(r_u)x_{ui} + h_i^{\text{down}}(r_{uv})y_{uvi} + \sum_l h^{\text{int}}(f_l)r_{uv}z_{uvl} - \alpha r_{uv}\right) \tag{1}$$

$$\text{subject to} \quad f_l = \sum_{u,v} r_{uv}z_{uvl} \le c_l, \forall\, l \tag{2}$$

$$\sum_{i\in S} x_{ui} = 1, \forall\, u \tag{3}$$

$$\sum_{i\in S} y_{uvi} = 1, \forall\, u, v \in V_u \tag{4}$$

$$\sum_{u\in U} x_{ui} \le a_i, \forall i \in S \tag{5}$$

$$\sum_{l'\in O_i} z_{uvl'} + x_{ui} \ge z_{uvl}, \forall\, u, v \in V_u, i, l \in O_i \tag{6}$$

$$\sum_{l\in O_i} z_{uvl} \ge y_{uvi}, \forall\, u, v \in V_u, i \tag{7}$$

$$x_{ui}r_u \le b_{ui}^{\text{up}}, \forall\, u, i \tag{8}$$

$$y_{uvi}r_{uv} \le \min(r_u, b_{vi}^{\text{down}}), \forall\, u, v \in V_u, i \tag{9}$$

$$\text{variables} \quad x_{ui}, y_{uvi}, z_{uvl} \in \{0,1\},\ \forall u, v, i, l$$
$$r_u, r_{uv} \in \{R_1, \ldots, R_M\}$$

The objective function has four terms: the first three terms represent the end-to-end latency (sum of upload, internal, and download latency), and the fourth term represents the viewer-perceived video rate. The objective function balances between video bitrate and latency using the parameter α, which can be set by the user or application. Latency is a key consideration for live video streams, in contrast to video on demand which mainly cares about bitrate, because delayed streams are out of sync with external events (e.g., the group chat that appear next to Twitch streams, social media). More precisely, we define the upload (h_i^{up}), internal (h^{int}), and download (h_i^{down}) latencies as:

Symbol	Description
$d_{ui}^{\text{up}}, d_{vi}^{\text{down}}$	upload latency between uploader u and server i, download latency between viewer v and server i (sec).
$b_{ui}^{\text{up}}, b_{vi}^{\text{down}}$	upload bandwidth between uploader u and server i, download bandwidth between viewer v and server i (bits/sec).
$h_i^{\text{up}}, h^{\text{int}}, h_i^{\text{down}}$	upload, internal, and download latency of node i (sec).
$\mathcal{R}$	set of possible video rates for video $\{R_1, \ldots, R_M\}$.
M	number of of video rates available for uploader u.
S	set of servers.
$\mathcal{U}$	set of uploaders.
V_u	set of users viewing the video of uploader u.
O_i	set of incoming links to server i.
a_i	maximum number of inbound connections that can be served by server i.
c_l	capacity of link l (bits/sec).
f_l	flow on link l (bits).
α	parameter trading off viewer latency versus video bitrate.
x_{ui}	binary variable indicating whether user u uploads to server i.
y_{uvi}	binary variable indicating whether viewer v downloads from server i.
r_u	video rate of uploader u (bits required in the current time epoch).
r_{uv}	downloaded video rate for viewer v of uploaded video u (bits required in the current time epoch).
z_{uvl}	binary variable indicating whether video uv is sent on link l.

Table I: Table of notation

$$h_i^{\text{up}}(r_u) = d_{ui}^{\text{up}} + \frac{r_u}{b_{ui}^{\text{up}}} \tag{10}$$

$$h^{\text{int}}(f_l) = \frac{1}{[c_l - f_l]^+} \tag{11}$$

$$h_i^{\text{down}}(r_{uv}) = d_{vi}^{\text{down}} + \frac{r_{uv}}{b_{vi}^{\text{down}}} \tag{12}$$

The upload latency (10) is the sum of the queuing and processing delays (d_{ui}^{up}) and the transmission delay ($\frac{r_u}{b_{ui}^{\text{up}}}$), and similar for the download latency (12). The internal latency is given by the queuing delay and is modeled by an M/M/1 queue (17).

The control knobs are the discrete variables x_{ui}, the uploader's server selection; r_u, the upload video rate (these variable are the crucial "first-mile" factors); y_{uvi}, the viewer's download server selection; z_{uvl}, the routing decisions and r_{uv}, the download video rate. Constraint (2) says that the sum of flows on each link must be less than the link capacity. Constraints (3, 4) says that each uploader (viewer) must be connected to one upload (download) server. Constraint (5) says that each server i can accept at most a_i uploaders, due to limited transcoding ability on the server and desire for load-balancing. Constraint (6) says that flows can

only be forwarded on from a node if they are received or stored there. Constraint (7) say that the data must flow outward from a download server with a connected viewer. Constraint (8) says that the upload video bitrate must be less than the bandwidth between the uploader and the upload server, and constraint (9) says that the download video bitrate must be less than the bandwidth between the viewer and the download server. In the following section, we present solutions to Problem 1 for two different architectures that are prevalent in real-world content distribution systems.

III. Solutions for Different Architectures

First, we consider a One-hop-overlay architecture (Fig. 1) that is used for video-conferencing in Skype-like systems, in which both the uploader and viewer download from the same edge server. We present an optimal polynomial-time algorithm for this problem through a novel reduction to a network-flow problem. Second, we consider a Full-overlay architecture (Fig. 1) in which the overlay network can replicate its streams to different download servers. We prove that this problem is NP-complete and present convex relaxations that yield practically efficient solutions. All proofs can be found in the Appendix.

A. One-hop-overlay Architecture

In this architecture, since the download and the upload server are the same for the uploader and its set of viewers, the remaining decision variables are the upload server for each uploader x_{ui}, the upload video bitrate r_u, and the download video bitrate r_{uv}. Clearly, there is negligible internal routing latency h^{int}. Based on this, Prob. 1 can be re-written for the special case of the One-hop-overlay architecture:

Problem 2: One-hop-overlay

$$\begin{aligned}
&\min \sum_{u\in U}\sum_{v\in V_u}\Big(\sum_i \Big(h_i^{\text{up}}(r_u)+h_i^{\text{down}}(r_{uv})\Big)\,x_{ui}-\alpha r_{uv}\Big)\\
&\text{subject to}\quad (3),(5),(8),(9)\\
&\text{variables}\quad x_{ui}\in\{0,1\},\ \forall\, u,i\\
&\qquad\qquad r_u, r_{uv}\in\{R_1,\ldots,R_M\},\ \forall\, u,v\in V_u
\end{aligned}$$

Prob. 2 is a challenging integer non-linear problem where a naiive brute-force solution is very expensive with time-complexity $O(M^{|V_u|}(M|S|)^{|\mathcal{U}|})$. However, we find an optimal solution with polynomial time-complexity by breaking the problem down into two tractable parts. Firstly, in the inner loop, we solve for the download video rate r_{uv} when the upload video rate r_u and upload server x_{ui} are known. We show that given the upload rate and upload server, selecting the download rate reduces to a discrete choice between the minimum and maximum possible bit rates. We do this by reducing the problem to an instance of a min-cost network flow problem. Secondly, in the outer loop, we efficiently solve for the upload video rate r_u and upload server x_{ui}. These steps are combined and shown in Alg. 1, and are now described in further detail.

Inner loop (viewer bitrate): We can solve for the download rate directly when the upload server and upload bitrate are known, as given in Lemma 1.

Lemma 1: For a fixed server i, uploader u with upload rate r_u, viewer v, and weight parameter α, the objective function in Prob. 2 can be minimized by picking the appropriate download rate $r_{uv}(i)$:

$$r_{uv}(i)=\begin{cases} R_1, & \text{if}\ \frac{1}{b_{vi}^{\text{down}}}\geq\alpha\\ \max\{r\in\mathcal{R}: r\leq\min(b_{vi},r_u)\}, & \text{otherwise}\end{cases}$$

Intuitively, when the weight α is high, video rate is highly favored in the objective function, so we pick the highest bitrate. On the other hand, when the viewer's bandwidth is low, we tend to pick the lowest download bitrate. The "in-between" video bitrates are not selected.

Outer loop (upload server and upload bitrate): We can now iterate over different combinations of server selection x_{ui} and upload rate r_u, each time calculating the associated optimal download bitrate according to Lemma 1. However, a brute force approach to solve for the remaining variables x_{ui} and r_u would require iterating over all possible combinations of uploaders, servers and upload rate. This approach is extremely inefficient since it has time complexity $O((M|S|)^{|\mathcal{U}|})$, which is exponential in the number of uploaders.

Therefore, we reduce the problem to an instance of the network flow problem, which yields an optimal polynomial-time algorithm. The main idea is to transform Prob. 2 into a graph-based flow problem. An example transformed instance is shown in Fig. 2, where uploaders are represented by nodes and replicated M times, according to the number of possible upload bitrates. Latency (part of the objective in Prob. 2) is incorporated into the problem as link weights, and load balancing (constraint 5) as link capacity. Constraints (3) and (8) are satisfied by construction, and constraint (9) by Lemma 1. The resulting graph problem is solved using minimum-cost network flow algorithms to find the optimal matching.

For ease of exposition, we rewrite the inner terms of the objective function of Prob. 2, for a specific uploader u with upload rate r_u selecting server i, as:

$$w_{ui}(r_u)=|V_u|h_i^{\text{up}}(r_u)+\sum_{v\in V_u}h_i^{\text{down}}(r_{uv}(i))\alpha r_{uv}(i),\ r_u\leq b_{ui}.$$

This cost $w_{ui}(r_u)$ is used as the link cost between the set of replicated uploaders $\mathcal{U}'$ and the set of servers S. We construct the network flow instance and solve for the upload server and upload rate as described in Alg. 1. Fig. 2 illustrates an example of the network flow

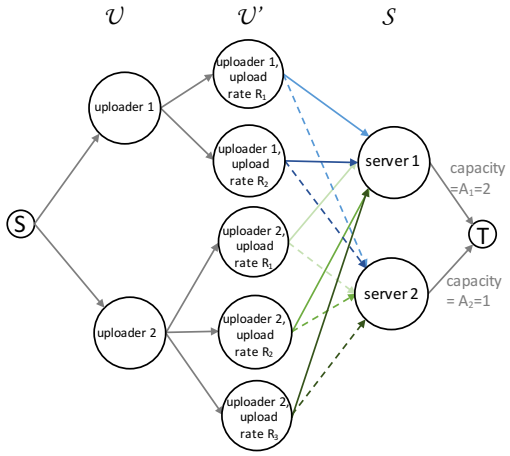


Figure 2: Example of Alg. 1. The capacity of all links is 1, except for the links between $\mathcal{S}$ and T. The costs of all links are 0, except for the links between $\mathcal{U}'$ and S.

instance constructed by Algorithm 1. The following proposition proves the correctness of Alg. 1:

Proposition 1: Alg. 1 solves Problem 2.

B. Full-overlay Architecture

The Full-overlay architecture is where an uploader sends its video stream to an edge server, the overlay network internally routes the stream from the upload server to the set of download servers, and viewers stream the video from the download servers. The Full-overlay architecture is similar to content distribution networks (CDN) but also includes the "first-mile" decisions, which are relatively less important in CDNs because content can be pre-stored on servers. In the context of CDNs, the "last-mile" mapping between viewers and download servers has been well-studied (e.g., (4)); for example, a CDN can select the download server with the minimum RTT predicted using historical data, or according to other factors such as load balancing. Following current practice, we assume that the live streaming service uses typical CDN download server selection techniques (i.e., the variable y_{uvi} is known). The remaining job of the live streaming service is to determine (a) upload server selection and upload video bitrate, and (b) the internal routing. Therefore, in the Full-overlay architecture, Prob. 1 becomes:

Problem 3: Full-overlay

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\in U}\sum_{v\in V_u}\left(\sum_i h_i^{\text{up}}(r_u)x_{ui}\right. \\
& \left. + \sum_l h(f_l)r_{uv}z_{uvl} - \alpha r_{uv}\right) \\
\text{subject to} \quad & (2),(3),(5),(6),(7),(8),(9) \\
\text{variables} \quad & x_{ui}, z_{uvl} \in \{0,1\},\ \forall u,v,i,l \\
& r_u, r_{uv} \in \{R_1,\ldots,R_M\}
\end{aligned}$$

Algorithm 1 One-hop-overlay Network Flow Algorithm

Input: Constants $d_{ui}^{\text{up}}, d_{iv}^{\text{down}}, b_{ui}^{\text{up}}, b_{iv}^{\text{down}}, R_1, ..., R_M$

Network flow graph construction:

1) Create uploader nodes for each $u \in \mathcal{U}$, connect them to one source node. Create server nodes for each $s \in S$, connect them to one sink node.
2) For each uploader node, create new data rate nodes for each $R_i \in \mathcal{R}$ and connect. For each server node, connect it to all data rate nodes.
3) Set the capacity of edge between server node i and sink to be a_i. Set the capacity of the rest of the edges to be 1.
4) For each uploader u, set the cost of edge between server node i and the data rate node R_u to be $w_{ui}(R_u)$. Assign 0 cost to the rest of the edges.
5) Set the demand of the source node to be $|\mathcal{U}|$.

Minimum cost network flow solution:

1) Apply the cost-scaling push-relabel algorithm to the constructed graph.
2) For each uploader u, set $r_u^* \leftarrow R_u$ if node u is connect to data rate node R_u in the solution. Set $x_{ui}^* \leftarrow 1$ if the data rate node is further connected to server node i. Set $r_{uv}^* \leftarrow r_{uv}(i)$ for each $v \in V_u$ according to Lemma 1.

Output: Decision variable values $x_{ui}^*, r_u^*, r_{uv}^*$

Compared to Prob. 1, constraint (4) is assumed to be satisfied and is removed, and thus the objective function is re-written to be slightly simplified. This problem is a non-linear integer problem that is expensive to solve in a brute-force manner. In Prop. 2, we show that the decision version of Prob. 3 is NP-complete, thus the optimization problem is hard to solve.

Proposition 2: The decision version of Prob. 3 is NP-complete.

The proof involves showing that set cover, which is NP-complete, reduces to the decision version of Prob. 3.

Heuristic: Therefore, eschewing optimality, we present a heuristic solution in which we relax the problem and iteratively solve for two sets of variables. Specifically, for fixed x_{ui} and z_{uvl}, Prob. 3 is a convex problem if allow the relaxation $0 \le r_u, r_{uv} \le 1$. Similarly, for fixed r_u and r_{uv}, it can be shown that Prob. 3 is a convex problem if we allow the relaxation $0 \le x_{ui}, z_{uvl} \le 1$. In particular, $h^{\text{int}}(f_l)r_{uv}z_{uvl}$ is convex. Based on these relaxations, we design a heuristic where we first solve for x_{ui} and z_{uvl}, which correspond to the upload servers and the links on which video will be sent, and then we solve for r_u and r_uv, which corresponds to the upload and download rate. We continue until the difference between values of the objective function over time becomes small.

Iterative updates: In practice, optimization decisions have to be made for every arrival and departure event of either video uploaders or viewers. It is impractical to repeatedly solve Prob. 3 when these events occur. Therefore, it is important to solve Prob. 3 in an incremental manner that allows us to select new upload server and video bitrate selection solutions based on the current solutions with minimal perturbations. We propose a decomposition approach that decomposes Prob. 3 into multiple sub-problems that are associated with each individual video uploader sessions. Then, when arrival and departure events occur and an incremental solution is required, only the affected sub-problems for the uploader and relevant links need to be recalculated. Video streams that are unaffected by the arrival or departure of a viewer will not be altered.

We define θ_l as the inverse residual capacity of link l. It represents the value of the internal latency at link l according to equation (11). We also define h^{int}_{uvl} as a slack variable for uploader u, user v and link l, to linearize the internal latency equation in (11). Let L be an arbitrarily large value. Then we can rewrite Prob. 3 as an equivalent Prob. 4 (proof omitted due to lack of space):

Problem 4: Transformed version of Prob. 3

$$\begin{aligned}
\text{minimize} \quad & \sum_{u\in U}\sum_{v\in V_u}\left(\sum_i h^{\text{up}}_i(r_u)x_{ui} \right. \\
& \left. + \sum_l h^{\text{int}}_{uvl} r_{uv} - \alpha r_{uv}\right) \\
\text{subject to} \quad & (3),(5),(6),(7),(8),(9) \\
& h^{\text{int}}_{uvl} + (1-z_{uvl})L \geq \theta_l, \forall\, u,v,l \qquad (13) \\
& \sum_{u,v} r_{uv} z_{uvl} + \frac{1}{\theta_l} = c_l, \forall\, l \qquad (14) \\
\text{variables} \quad & x_{ui}, z_{uvl} \in \{0,1\},\ \forall u,v,i,l \\
& r_u, r_{uv} \in \{R_1,\ldots,R_M\}\ \forall u,v \\
& h^{\text{int}}_{uvl}, \theta_l \in \mathbf{R}^+\ \forall u,v,l
\end{aligned}$$

To decompose Prob. 4, the dual decomposition technique is applied to relax the new constraints (18). We introduce λ_{uvl} for constraint (13) and β_l for constraint (14) to create the following Lagrangian equation:

$$\begin{aligned}
& L(X,Z,R,\lambda,\beta) = \\
& \sum_{u\in U}\sum_{v\in V_u}\left(-\alpha r_{uv} + \sum_l h^{\text{int}}_{uvl} r_{uv} + \sum_i (h^{\text{up}}_i(r_u)x_{ui}) \right. \\
& \left. + \sum_l \lambda_{uvl}(\theta_l - h^{\text{int}}_{uvl} + z_{uvl}L - L)\right) \\
& + \sum_l \beta_l \Big(\sum_{u\in U}\sum_{v\in V_u} r_{uv} z_{uvl} + \frac{1}{\theta_l} - c_l\Big) \qquad (15)
\end{aligned}$$

The dual decomposition technique is designed to minimize the Lagrangian equation by solving a primal problem and a dual problem. The primal problem is designed to minimize the Lagrangian equation according to the primal variables (i.e., X,Z,R) while the dual problem is maximizing the equation according to the dual variables (i.e., λ, β).

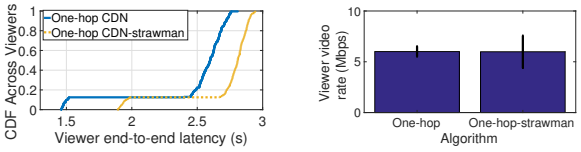
In the primal problem, since the θ variables only appear in the Lagrangian equation, we can separate the decision of θ from the rest of the primal problem. We refer to these sub-problems as the residual capacity adjustment problems. For link l, the corresponding residual capacity adjustment problem is to minimize $\beta_l \frac{1}{\theta_l} + \sum_u \sum_v \lambda_{uvl}\theta_l$, subject to $\theta_l \geq c_l$. Solving this problem produces a closed-form solution of $\theta^*_l = \sqrt{\frac{\beta_l}{\sum_u \sum_v \lambda_{uvl}}}$.

Next, by relaxing the capacity constraint in (14), the video sessions that share network links are no longer coupled by the capacity constraint. Therefore, the primal problem can be further decoupled into primal sub-problems that are associated with each individual video uploader sessions. We refer to these sub-problems as the min-flow problem. For uploader u and all of its viewers in V_u, the corresponding min-flow problem can be written as follows:

Problem 5: Uploader u: Min-flow

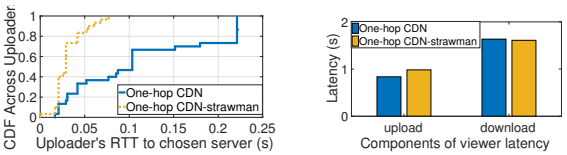
$$\begin{aligned}
\text{minimize} \quad & \sum_{v\in V_u}\left(-\alpha r_{uv} + \sum_l h^{\text{int}}_{uvl} r_{uv} + \sum_i h^{\text{up}}_i(r_u)x_{ui} \right. \\
& \left. + \sum_l \lambda_{uvl}(-h^{\text{int}}_{uvl} + z_{uvl}L)\right) \\
& + \sum_l \sum_{v\in V_u} \beta_l r_{uv} z_{uvl} \\
\text{subject to} \quad & (3),(4),(5),(6),(7),(8),(9) \\
\text{variables} \quad & x_{ui}, z_{uvl} \in \{0,1\},\ \forall v,i,l \\
& r_u, r_{uv} \in \{R_1,\ldots,R_M\}\ \forall v \\
& h^{\text{int}}_{uvl}, \theta_l \in \mathbf{R}^+\ \forall v,l
\end{aligned}$$

Prob. 5 can be solved using the heuristic algorithm proposed previously. After solving Prob. 5 and the residual link capacity problem described above, the dual problem of updating the dual variables λ and β is solved using subgradient descent to maximize the value of (15) with respect to the dual variables. Specifically, in the k-th iteration, the dual variable λ_{uvl} is updated using the subgradient value of $\theta_l(k) - h^{\text{int}}_{uvl}(k) + z_{uvl}(k)L - L$, and θ_l is updated using the subgradient value of $\sum_{u\in U}\sum_v r_{uv}(k) z_{uvl}(k) + \frac{1}{\theta_l(k)} - c_l$. $\theta_l(k), h^{\text{int}}_{uvl}(k), z_{uvl}(k), r_{uv}(k)$ are the primal variables produced by solving the primal sub-problems in the k-th iteration. In summary, when arrival and departure events occur and an incremental solution is required, only the affected sub-problems for the uploader and relevant links need to be recalculated. Video streams that are unaffected by the arrival or departure of a viewer will not be altered.



(a) End-to-end latency (b) Viewer's video rate

Figure 3: Viewers have low end-to-end latency and good video rates in the One-hop-overlay architecture.



(a) Upload server selection (b) Latency breakdown

Figure 4: Latency components in the One-hop-overlay architecture. Choosing a further uploader server can reduce viewer latency.

IV. Evaluation

In this section, we evaluate the performance of our algorithms in the One-hop-overlay and Full-overlay architectures through trace-driven (16) simulations. We compare several algorithms:

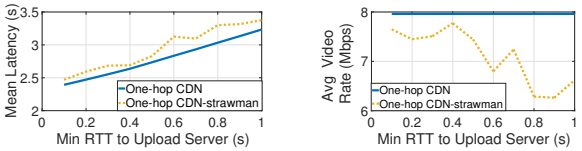
- *One-hop-overlay*: Solve Prob. 2 using Alg. 1 for upload server (same as download server), upload bitrate and viewer rate.
- *One-hop-overlay-strawman*: The uploader picks the server with the lowest RTT with rate equal to the upload bandwidth. The viewers connect to the same server and download at a rate equal to the minimum of the uploader rate and download bandwidth.
- *Full-overlay*: Solve Prob. 3 for upload bitrate, viewer rate, and internal routing including the upload server and the download server.
- *Full-overlay-strawman*: The uploader picks the server with the lowest RTT with rate equal to the upload bandwidth. The viewers pick the server with the lowest RTT with a rate equal to the minimum of the uploader rate and download bandwidth. The internal routing is optimized using a simplified version of Prob. 3 that solves for the link usage only.

Setup: We perform simulations using traces from real Twitch sessions (16). These traces give the time zone of each uploader, and the number of users viewing each uploaded stream. We take the top 30 most popular uploaders in a 24-hour period, who have a total of 232,000 viewers with group size ranging from 2000-58,000 (average 7700 users/group). We map the time zone of each uploader to a central geographical location based on their time zone, and randomly distribute clusters of 1000 viewers geographically (since the trace does not provide geographical information of the viewers). For the topology, we map 17 Twitch servers (19) to their geographical location, and generate 65 directional links between them. It is difficult to know realistic link capacities in real overlays/CDNs, so we model link capacity using two distributions (power law and uniform) between 100 Mbps and 10 Gbps (20). The upload and download edge latencies are generated proportional to geographical distance between 0-1 s, and the upload and download edge bandwidths are are generated inversely proportional to latency with a range between 0-10 Mbps.

A. *One-hop-overlay*

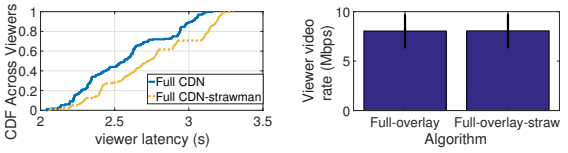
We first wish to compare *One-hop-overlay* with the *One-hop-overlay-strawman*. We run Alg. 1 on the Twitch data and plot the viewer end-to-end latency and video rate in Fig. 3. In terms of viewer latency, we can see that with our approach in Fig. 3a, approximately 10% of the viewers reduce their latency by about 0.5 s, and approximately 90% of users reduce their latency by about 0.2 s. Relative to the mean latency of *One-hop-overlay-strawman*, 2.7 s, this represents a relative decrease (17% and 8%, respectively). We can also see in Fig. 3b that the video rates maintained are at a level similar to *One-hop-overlay-strawman*, but overall the viewer experiences decreased latency, which is important in live-streaming. The tradeoff between video rate and latency can be controlled by changing the parameter α (in this particular experiment, $\alpha = 0.5$). By changing α (not shown), we observe that *One-hop-overlay* can perform better than *One-hop-overlay-strawman* in latency or video rate, but not both, which is an engineering trade-off.

We next wish to delve deeper to understand the choices of Alg. 1. In Fig. 4a, we plot the RTT between the uploader and its selected server. We can see that *One-hop-overlay* is selecting upload servers with higher RTT than the servers selected by *One-hop-overlay-strawman*. How can this be possible if the end-to-end viewer latency is reduced? Indeed, looking at the breakdown in latency between uploading and downloading in Fig. 4b, we can see the *One-hop-overlay* achieves lower upload latency despite choosing servers with higher RTT. Upon examining the data, we find that *One-hop-overlay* chooses a much lower upload video rate of 6.9 Mbps on average, while *One-hop-overlay-strawman* uploads at 9.8 Mbps on average. This means that *One-hop-overlay-strawman*, despite choosing an upload server with lower RTT, was wastefully uploading at a higher rate in order to benefit a few of its viewers, increasing the latency for its other viewers who could not receive the high video rate due to constrained download bandwidth. *One-hop-overlay*, on the other hand, picked an upload server with slightly higher RTT because it considered its viewers' download latencies and bandwidths, and chose an upload server based jointly on RTT and bandwidth across its viewers.



(a) Latency (b) Video rate

Figure 5: Impact of uploader RTT for One-hop-overlay architecture.



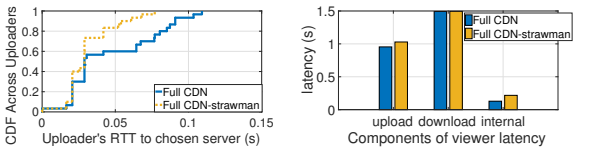
(a) End-to-end latency (b) Viewer's video rate

Figure 6: Viewer end-to-end latency is reduced with good video rates in the Full-overlay architecture.

Further, we wish to answer the question: what happens if we remove the "obvious" low-RTT servers from the topology? Will *One-hop-overlay-strawman* still make good decisions if there are many "mediocre" server choices with moderate RTT? To answer these questions, we increase the minimum RTT between the uploader and server (keeping the total number of servers constant), run Alg. 1 in each scenario, and plot the average viewer latency and video rate in Fig. 5. In Fig. 5a, one can see that the viewer latencies increase for both *One-hop-overlay* and *One-hop-overlay-strawman* as fewer low-RTT servers are available, as expected. However, in Fig. 5b, we can see that *One-hop-overlay* maintains high video rate by still selecting the appropriate server and video rates, while *One-hop-overlay-strawman* suffers as it chooses servers with low RTT that also have low bandwidth, thus decreasing the upload and download video rates. Thus, *One-hop-overlay* performs well even as edge latency increases.

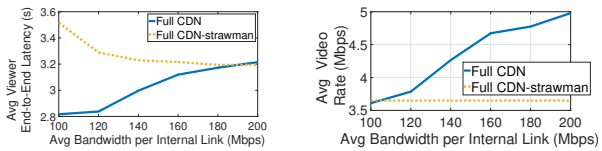
B. *Full-overlay*

We now turn our attention to the Full-overlay case, and run our iterative algorithm (Section III-B) to solve Prob. 3. We plot the end-to-end viewer latency and viewer video bitrates in Fig. 6. We see that the end-to-end latency is reduced by approximately 0.16 s for all users, and the video rates are similar for *Full-overlay* and *Full-overlay-strawman*. Again, this shows that latency can be reduced while keeping video rates constant, but we cannot achieve both lower latency and higher bitrates. We examine the choices made by *Full-overlay* to see how they differ from *Full-overlay-strawman* and the reason for *Full-overlay*'s reduced latency. In Fig. 7a, we plot the RTT from the uploader to the selected upload server for *Full-overlay* and *Full-overlay-strawman*. The strawman chooses the server with the lowest RTT, by definition, while *Full-overlay*



(a) Upload server selection (b) Viewer latency breakdown

Figure 7: Latency components in the Full-overlay architecture. Choosing a further upload server can reduce overall viewer latency.



(a) Latency (b) Video rate

Figure 8: Impact of internal bandwidth on Full-overlay architecture.

chooses servers which are slightly further away. In Fig. 7b, we see that *Full-overlay* has a lower upload latency despite choosing an upload server with higher RTT. We analyze the data and find that *Full-overlay* tends to upload at a lower rate (9.2 Mbps versus 10.4 Mbps for *Full-overlay-strawman*), thus decreasing the upload latency. The intelligent choice of upload server also enables *Full-overlay* to decrease the internal latency by choosing less congested paths, especially since the 9 of the 30 uploaders are located in the same city and can benefit from upload server load-balancing.

Next, we wish to understand conditions when *Full-overlay* can offer higher video rates, in addition to low viewer latencies shown previously. We wish to examine the effect of the internal overlay network, and see if higher link capacities can increase the video rates. To test this, we randomize the upload and download bandwidths and latencies to isolate the effect of the internal network. We increase the average link capacity of the internal network (e.g., Twitch pays the ISP for more bandwidth) and measure the average end-to-end latency and average video rate, and plot the results in Fig. 8. In Fig. 8b, *Full-overlay* can take advantage of increased link capacity to increase the video rates (which also increases latency, although not to the level of *Full-overlay-strawman*). *Full-overlay-strawman*, on the other hand, enjoys decreased latency (Fig. 8a) as internal link capacity increases, but its video rates stay constant at low values since its video rate decision depends only on the uploader-upload server or the downloader-download server edge links. Thus, we can see that increasing the internal bandwidth of the overlay network is one way for live-streaming providers to improve video rates under *Full-overlay*, while still lowering end-to-end latencies.

V. Related Work

Existing works (1; 2) consider matching users to cloud transcoders to minimize the latency while accounting for the cloud provider's latency, bandwidth, and transcoders. (21) provides a transcoding framework that allows viewers different bit-streams. While transcoder placement and downstream decisions are important, they are complementary to our work as we focus on the edge decisions made by the uploader and the route followed by the streams in a multi-hop overlay network, which are not considered in these works.

Prior work on mapping uploaders to edge servers (22) focuses on minimizing economic cost. Since we consider the problem from the perspective of a single live streaming CDN, we consider multi-user interactions when multiple viewers select the same upload server as well as video-specific QoE metrics. (15) show that choosing the best replica and route leads to better results than just choosing the replica for online services. While they focus on the benefit to the viewer, we consider the multi-user scenario with joint benefit across viewers.

Traditional CDN solutions (4; 3; 23) that use historical data to improve video caching or placement are not directly applicable to our problem since user-uploads are generated and consumed in relatively short, unpredictable time spans, and are not pre-stored in the CDN. While works such as (23) jointly optimize the cost to the CDN and the viewer we optimize for upload bitrate, with a focus on the uploader's link to the overlay network. (5) solves the problem of constructing a network of reflectors that split and forward live streams to viewers. This approach is too heavy-weight and costly for user-uploads, where the relative number of viewers is far less. Moreover, we consider more general network topologies than their tree topologies. While (6) focuses on the optimization of CDN routing for live streams, we posit that along with routing, edge server and upload bitrate selection are also important factors.

Several works (17; 24) consider joint content placement and traffic engineering. (24) shows how content placement is more important than traffic engineering for CDNs. We believe that for user-upload streaming, not only should the mapping between viewers and the overlay network be considered, but also mapping uploaders to the upload servers of the overlay network, since that impacts downstream decisions and thus end-to-end throughput and delay.

VI. Conclusions

In this paper, we contend that streaming high-quality, voluminous, user-uploaded videos is different from the existing systems for video-on-demand or live streaming due to the "first-mile" between the uploader and the network. Specifically, we show that the edge decisions of where to upload the video and the video's upload bitrate are crucial to achieve quality-of-experience (both latency and video rate) across all users. We developed efficient algorithms and performed trace-based evaluation to solve this problem. We plan to extend this framework to consider edge containers that can perform transcoding and IoT camera devices.

Acknowledgements

Initial work on this project by Hui Yang is gratefully acknowledged. This work has been supported in part by NSF CNS-1817216.

Appendix

Lemma 1

Proof: We can rewrite the objective function as: $\sum_i \sum_u \left(|V_u| h_i^{\text{up}}(r_u) + \sum_{v \in V_u} h_i^{\text{down}}(r_{uv})\right) x_{ui} - \alpha \sum_u \sum_v r_{uv}$. For a fixed server i, uploader u with upload rate r_u, and viewer v, then the objective can be minimized by solving: $\min_{r'_{uv} \leq \min(r_u, b_{ui})} h_i^{\text{down}}(r'_{uv}) - \alpha r'_{uv} = \min_{r'_{uv} \leq \min(r_u, b_{ui})} d_{vi} + \frac{r'_{uv}}{b_{vi}} - \alpha r'_{uv}$ Clearly, when $\frac{1}{b_{vi}} \geq \alpha$, then the expression is positive and we should pick the minimum rate. When $\frac{1}{b_{vi}} < \alpha$, then the expression is negative and we should pick the maximum possible rate. ■

Proposition 1

Proof: The objective function of Prob. 2 can be re-written as: $\sum_u \sum_i \sum_t w_{ui}(r_u) x_{ui}$. The weights $w_{ui}(r_u)$ are integer (or can be scaled to be integer). The capacity of each link in the graph is 1 or a_i, both integers. The source and sinks have demand $|\mathcal{U}|$ are integer. Therefore, we can use the integrality theorem that states that as long as the input data to the network flow problem are integer, there is an optimal solution consisting only of integers. The graph $\mathcal{G}$ in Alg. 1 will thus have integer flows and, by construction, be an integer solution to Prob. 2.

Specifically, constraint (5) is satisfied because the capacity between server i and T has capacity a_i. (3) is satisfied because the capacity of the links between $S, \mathcal{U}, \mathcal{U}'$ is 1, and the demand of $|U|$ at S forces 1 unit of flow between for each node $u \in \mathcal{U}$ to be forwarded on to one server. Constraints (8),(9) are satisfied by definition of $w_{ui}(r_u)$ and $r_{uv}(i)$. in Lemma 1. ■

Proposition 2

Proof: We will show that set cover, which is NP-complete, reduces to the decision version of Prob. 1. The proof technique is inspired by (24). The set cover problem defines elements $1, \ldots, n$ sets $S_1, \ldots, S_m$. Consider the following special case of the network problem, as shown in Fig. 9. There are are m origin servers which are fully connected with each other, and each origin server has upload capacity 1. There are $n+1$ edge servers. There are $m-k+1$ uploaders which form

a fully connected bipartite graph with the origin servers. Origin server i is connected to edge servers in the set S_i. The first uploader is very popular and is requested one viewer at each of the first n edge servers. The remaining $m-k$ uploaders are requested by $m-k$ viewers at the $(n+1)^{th}$ edge server. Each viewer has bandwidth $b_{vi}^{\text{down}} = 1$ to her edge server, so y_{uvj} is fixed.

The only possible video rate is $R_1 = 1$, so $r_u^* = 1, r_{uv}^* = 1$. Each link between uploaders and origin servers has upload bandwidth $b_i^{\text{up}} = 1$ and latency $d_{ui}^{\text{up}} = 0$, and therefore cost 1 if the link is used (10). Each link between origin and edge servers has capacity 2 and therefore cost 1 if the link is used (11). Each link between edge servers and viewers (not shown in the example) has download bandwidth $b_{vi}^{\text{down}} = 1$ and latency $d_{vi}^{\text{down}} = 0$, and therefore costs 1 if the link is used (12). Based on the problem setup, only the placement x_{ui} and the routing z_{uvl} must be determined.

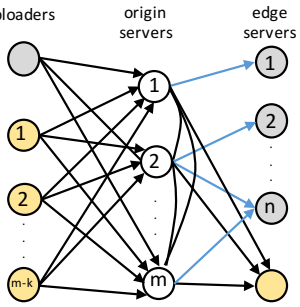


Figure 9: Mapping our problem to set cover to prove Prop. 2.

The decision problem is: Can the above network achieve a value $\leq b$? The set cover problem is: Is there a set cover of size k? We claim: There is a set cover of size k iff the above network can achieve an value $\leq 2n + 3m - 2k$.

$\Longrightarrow$: *If there is a set cover of size k, the network achieves a value $n + m - 1$.* For the popular uploader, select k origin servers according to the k set cover locations. Send the popular video to one of the selected origin servers, and forward from there to the other $k-1$ selected origin servers. This costs k. Send from the selected origin servers to the n edge servers. This costs n. Send from the edge servers to the corresponding viewers. This costs n. For the $m-k$ regular uploaders, transmit to the remaining unused $m-k$ origin servers. This costs $m-k$. From there, send to the regular edge server. This costs $m-k$. From there, send to the corresponding viewers. This costs $m-k$. The total cost is $2n + 3m - 2k$.

$\Longleftarrow$: *If there is no set cover of size k, the network achieves a value $> 2n + 3m - 2k$.* Since each origin server can only process one uploader, and each of $m-k$ regular videos must be uploaded to one origin server, there are k remaining origin servers for the popular video. If there is no set cover of size k, then one of the n edge servers must be satisfied by using a link of capacity 0. Therefore, the cost is ∞.

Therefore, since the decision version of Prob. 1 is equivalent to set cover which is NP-complete, our decision problem is NP-complete. ■

REFERENCES

[1] F. Chen, C. Zhang, F. Wang, and J. Liu, "Crowdsourced live streaming over the cloud," *CoRR*, vol. abs/1502.06314, 2015.

[2] Q. He, J. Liu, C. Wang, and B. Li, "Coping with heterogeneous video contributors and viewers in crowdsourced live streaming: A cloud-based approach," *IEEE Trans. Multimedia*, vol. 18, pp. 916–928, May 2016.

[3] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang, "C3: Internet-scale control plane for video quality optimization," *USENIX NSDI*, 2015.

[4] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," *ACM SIGCOMM*, 2012.

[5] K. Andreev, B. M. Maggs, A. Meyerson, J. Saks, and R. K. Sitaraman, "Algorithms for constructing overlay networks for live streaming," *CoRR*, vol. abs/1109.4114, 2011.

[6] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for cdn-based live video delivery," in *ACM SIGCOMM*, 2015.

[7] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: Measurement study of google+, ichat, and skype," *IEEE/ACM Trans. Net.*, vol. 22, pp. 826–839, June 2014.

[8] "Open broadcaster software." https://obsproject.com/.

[9] "Twitch test." https://r1ch.net/projects/twitchtest.

[10] K. Andreev, B. M. Maggs, A. Meyerson, and R. K. Sitaraman, "Designing overlay multicast networks for streaming," *ACM SPAA*, 2003.

[11] "Justin.tv's live video broadcasting architecture." http://highscalability.com/blog/2010/3/16/justintvs-live-video-broadcasting-architecture.html.

[12] "Under the hood: Broadcasting live video to millions." https://code.facebook.com/posts/1653074404941839/under-the-hood-broadcasting-live-video-to-millions/, 2015.

[13] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay cloud video conferencing," in *IEEE ICDCS*, 2015.

[14] Y. Wu, C. Wu, B. Li, and F. C. Lau, "vskyconf: Cloud-assisted multi-party mobile video conferencing," in *ACM SIGCOMM Workshop on Mobile Cloud Computing*, 2013.

[15] V. Valancius, B. Ravi, N. Feamster, and A. C. Snoeren, "Quantifying the benefits of joint content and network routing," *ACM SIGMETRICS*, 2013.

[16] "Live streaming sessions dataset." http://dash.ipv6.enstb.fr/dataset/live-sessions/.

[17] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an isp network," *ACM SIGMETRICS*, 2009.

[18] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[19] "Twitch status." https://twitchstatus.com/.

[20] M. Yu, W. Jiang, H. Li, and I. Stoica, "Tradeoffs in cdn designs for throughput oriented traffic," *ACM CoNEXT*, 2012.

[21] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen, and G. Zhang, "Online cloud transcoding and distribution for crowdsourced live game video streaming," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 27, pp. 1777–1789, Aug 2017.

[22] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, "Optimizing cost and performance for content multihoming," *ACM SIGCOMM*, 2012.

[23] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale vod system," *ACM CoNEXT*, 2010.

[24] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies isp traffic engineering," *ACM SIGMETRICS*, 2013.