Bi-criteria Approximation for a Multi-origin Multi-channel Auto-scaling Live Streaming Cloud

Zhangyu Chang and S.-H. Gary Chan, *Senior Member, IEEE,*

***Abstract*—Live video traffic has been widely observed to vary significantly within short timescale. In order to manage such traffic dynamic of overlay live streaming, the Content Provider (CP) may deploy a set of geo-dispersed auto-scaling servers where the pay-as-you-go deployment cost is charged by the amount of resources used due to server uploading and data transmission between servers. To support geo-distributed user demands, we study a novel multi-origin multi-channel auto-scaling live streaming cloud that pushes each channel stream in the core network overlay as a tree covering the end servers who have local demand for the channel. The Origin-to-End (O2E) delay from an origin to an end server is due to the Server-to-Server (S2S) delays of the overlay links along the path.**

By optimizing the overlay of the core network, we seek to minimize the deployment cost and O2E delays of the channels (i.e., a bi-criteria problem), which can be equivalently phrased as minimizing the deployment cost while meeting certain given maximum O2E delay constraints. We formulate a realistic problem capturing the major cost and delay components, and show its NP-hardness. We propose Cost-optimized Multi-Origin Multi-Channel Overlay Streaming (COCOS), a novel, efficient and near-optimal bi-criteria approximation algorithm with proven approximation ratio. Trace-driven extensive experimental results based on real-world live streaming service data validate that COCOS outperforms other state-of-the-art schemes by a wide margin (cutting the cost in general by more than 50%).

***Index Terms*—Auto-scaling, live streaming cloud, multiple origins, multiple channels, bi-criteria approximation, overlay optimization**

I. Introduction

It has been estimated that video will account for 82% of the global Internet traffic, and live video will reach 17% of the Internet video traffic [1]. Meanwhile, it has been widely observed that live streaming traffic varies significantly over a day, by possibly more than an order of magnitude [2], [3]. We consider the overlay distribution of live video channels for a geo-distributed audience, whose network traffic has enormous total volume and significant daily variation.

To cost-effectively respond to the live video traffic of such volume and dynamic, the Content Provider (CP) can allocate geo-dispersed *auto-scaling* servers (e.g., virtual machines or instances) and overlay links between them *on the fly*. Compared with the traditional infrastructure approach that statically sets aside a certain number of geo-dispersed servers with fixed streaming capacities which suffers from overprovisioning and limited scalability, this auto-scaling system can hence elastically rescale the resources (e.g., Amazon EC2 [4] can rescale the server capacity within minutes) with *pay-as-you-go* cost model to reduce the deployment cost with high scalability.

For this novel live streaming cloud, the CP allocates geo-dispersed auto-scaling servers on demand. To support geo-distributed audience, each auto-scaling server streams the live content to its associated local users.[1] The *active* auto-scaling servers form an overlay *core network* of the live streaming cloud, relaying and streaming the live channels cooperatively. (An auto-scaling server is *inactive* when it has no local user demand.)

We study and optimize this core network of the multi-origin and multi-channel live streaming cloud where live video channels originate from some geo-distributed *origin servers* and each *end server* demands a set of channels requested by its local audience. (How to deliver the live channels from an end server to its local audience is orthogonal to this work, and we discuss the related work in Section II.) These origin and end servers form this overlay core network to deliver all the channels to meet the local demand of each end server. To reduce overall delay, in the core network of this cloud, the streams are *pushed* in streaming fashion from the origin server to the end servers with local demand.

Without loss of generality, we assume that a channel can only originate from one origin server (if a channel originates from multiple origin servers, we can create a single virtual *super origin* that directly connects to these origin servers with zero-delay, zero-cost and infinite bandwidth *virtual* link). The channels may have heterogeneous streaming rates. Each channel stream is delivered in a push manner, and its path forms a delivery tree, from the origin server as the root, covering all the end servers with the demand for this channel. Clearly, the aggregation of all delivery trees for all the channels forms a *mesh*.

In Fig. 1, we illustrate the core network of an auto-scaling live streaming cloud under our consideration. We consider the overlay cloud as a complete graph of connectivity. Origin X and Y originate channel 1 and 2, respectively. To serve their local demand, Server E requests channels 1, Server D requests channel 2, and Server A and C request both channel 1 and 2. Note that we can activate (or deactivate) any end server, from A to E, on demand. As Server B has no local demand, it is inactive, and hence not to be paid for.

An end server receives a stream either from the origin server directly, or from another end server who already has it. Server

This work was supported, in part, by Hong Kong General Research Fund under grant 16200120.

The authors are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China (e-mail: zchang@cse.ust.hk; gchan@cse.ust.hk).

[1]In this work, a server is a node that can relay streams and serve its local user demand. In practice, it can be a CDN node, server farm, data center, etc.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

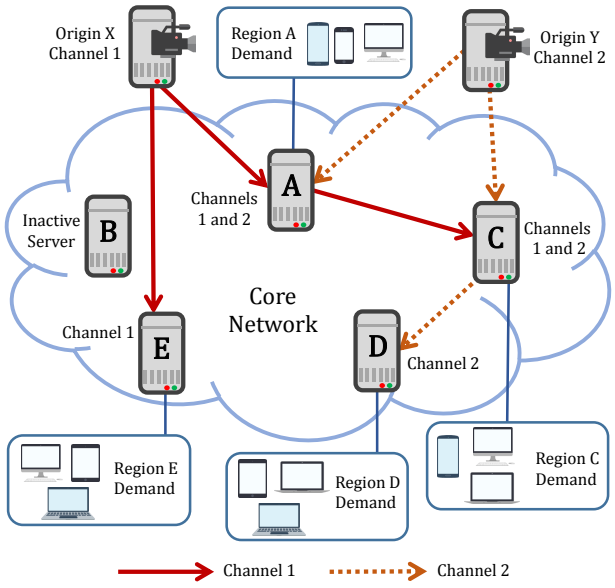


Fig. 1. A multi-origin multi-channel live streaming cloud.

A directly receives all the channels from the origin servers while Server C receives Channel 1 from A but Channel 2 from Origin Y. Server D receives Channel 2 from C, and Server E receives Channel 1 from Origin X.

The *Origin-to-End* (O2E) delay of a channel (i.e., the delay from the origin server to an end server that demands the channel) is an important Quality-of-Experience (QoE) measure. It is the sum of the *Server-to-Server* (S2S) delays of all the overlay links forming the stream path from the origin to the end server. S2S delay is the time for a signal to travel from one server to the next over the overlay, given by half of the round-trip time (RTT).

The *deployment cost* of the cloud is the sum of server and link costs. *Server cost* of a server depends on its uploading bandwidth to the other servers. *Link cost* depends on the data transmission through the overlay link between servers. Both auto-scaling servers and overlay links are shared with all the channels. Note that, as we mainly consider the optimization of the core network, given the local user demand, the cost for an end server to serve its local demand is an independent problem rather than an optimization parameter of our problem, and thus we do not include it in our model.

We seek to minimize deployment cost and O2E delays of the core network of this cloud (i.e., it is a bi-criteria problem). Without loss of generality, the problem is equivalently posed as minimizing deployment cost subject to certain given maximum O2E delay constraints[2] [5]. We refer to the decision variables of this optimization as *overlay construction* of the core network, which is the construction of the delivery tree of each channel, namely, *to what servers* and *what streams* an auto-scaling server should forward.

As the system parameters (e.g., user demand, S2S delays, server and link prices, etc.) may change over time, we have to regularly re-optimize the overlay such that, within each optimization period, the variations of these system parameters are negligible. To achieve this, we have a central optimizer that computes the overlay trees given the up-to-date parameters for the upcoming time period.

To the best of our knowledge, this is the first piece of work on bi-criteria optimization of streaming delay and deployment cost of the core network of a multi-origin multi-channel live streaming cloud with proven approximation ratio. Our contributions are as follows:

- *Bi-criteria problem formulation and complexity analysis:* We formulate a novel, comprehensive and realistic model for this bi-criteria problem, which captures cost and delay components. We formulate the Minimum Cost Streaming with Delay Constraints (MCSDC) problem that constructs multiple channel delivery trees to minimize the total deployment cost while meeting the given QoE (O2E delay) constraints. We prove that the problem is NP-hard.
- *COCOS: A novel bi-criteria approximation algorithm:* We propose an efficient and implementable bi-criteria approximation algorithm, termed **C**ost-optimized Multi-**O**rigin Multi-**C**hannel **O**verlay **S**treaming (COCOS), for an auto-scaling multi-origin multi-channel overlay live streaming cloud. We demonstrate that COCOS achieves low deployment cost with proven worst-case approximation ratio, strictly meets the QoE constraints, and has polynomial time complexity.
- *Extensive experimental results:* We conduct extensive trace-driven experimental studies based on real-world user trace (from a leading Chinese video service provider) and S2S delays from a real live streaming network. We show that COCOS significantly reduces the deployment cost and tightly meet the QoE constraints, cutting the cost significantly as compared with the state of the arts (in general by more than 50%).

This work is organized as follows. We first briefly review related work in Section II. We formulate the bi-criteria optimization MCSDC problem of minimizing the total deployment cost with the given delay constraints, and prove its NP-hardness in Section III. We present the bi-criteria approximation algorithm COCOS, and show its optimality in Section IV. We show illustrative experimental results and comparisons with other state-of-art schemes in Section V, and conclude in Section VI.

II. Related Work

We briefly discuss related work in this section. We show the major concerns of the related work and the comparison with our work in Table I.

There has been much work on optimization of video-on-demand (VoD) service over content distribution network (CDN) or peer-to-peer (P2P) network with different objectives and constraints. The work on CDN-based VoD in [6], [7] mainly considers maximizing the hit ratio by optimizing the

[2]As a multi-criteria problem has multiple objectives, the optimum is given as a set of solutions that follow *Pareto optimality* where no single objective can be further better off without sacrificing the other objectives. Often we optimize one objective and phrase the other objectives as constraints so that they satisfy certain given bounds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

TABLE I
MAJOR CONCERNS OF THE RELATED WORK

Category	Work	Major concerns	Comparison with our work
VoD (CDN)	[6], [7]	Maximize the hit ratio	Different network architecture and objectives
VoD (P2P)	[8]–[10]	Minimize the channel switching delay	
Data Center	[11]–[14]	Optimize the internal operation	Orthogonal work: Focusing on different part of the network
Crowdsourced streaming	[15]–[18]	Deliver content from end server to users	
Overlay live streaming	[19]–[21]	Minimize inter-ISP traffic	Our work is bi-criteria optimization that considers both deployment cost and O2E delay together
	[22]–[25]	Minimize deployment cost	
	[26]–[29]	Minimize O2E delay	
	[30]–[34]	Minimize overall delay	
	[35], [36]	Reduce channel switching delay	
	[37]–[41]	Reduce the server load	
Measurement study	[3], [42], [43]	Draw the overall picture of a live cloud	Our work aims at optimizing the system
Streaming multicasting	[44]–[47]	Create minimum cost topology	Our work considers O2E delay besides cost

caching strategies. By contrast, we consider a *live streaming* network where the contents are not cached and the origin pushes the contents to all users in real time. The work on P2P-based VoD [8]–[10] mainly considers minimizing the channel switching delay, or minimizing the server load by effectively using the resource from peers. In such VoD network, the O2E delay is not a major QoE concern due to the VoD contents.

Work on auto-scaling servers [11]–[14] mainly focuses on the internal operation within a data center instead of the live streaming network as a whole. Recent work on crowdsourced live streaming [15]–[18] focuses on how to effectively deliver live contents from a CDN server (i.e., an end server) to its local audience while our work deals with how to form a core network overlay that timely and cost-effectively deliver the streams from the origin servers to the end servers. Though the advancement in such fields is beneficial to our live streaming cloud, these schemes are orthogonal to our problem of optimizing the core network overlay of a live streaming cloud.

While cost and delay optimization of overlay live streaming has been studied, most of the work only focuses on one of the objectives. Some works [22]–[25] seek to minimize total cost while others [26]–[29] focuses on minimizing O2E delay. We cannot directly extend these schemes to our bi-criteria optimization of the core network as applying them without considering the tradeoff between delay and cost would lead to either excessive overprovisioning of resource or unsatisfactory O2E delay.

Work on multi-origin multi-channel overlay design has considered various different objectives from ours, such as minimizing inter-ISP traffic [19]–[21], minimizing the overall delay [30]–[34], reducing channel switching delay [35], [36], or maximize the number of delivered channels while reducing the server load [37]–[41]. The multi-origin multi-channel auto-scaling live overlay network we formulate here is novel, with a general and realistic cost and delay model. To the best of our knowledge, our work is the first to consider both O2E delay and deployment cost of the core network of such live streaming cloud. Previous work even does not consider single-channel version of this optimization problem.

The measurement studies in [3], [42], [43] indicates that it is challenging to guarantee QoE for live contents for multi-origin multi-channel live streaming network, though they have not given a solution. Some work on streaming multicasting [44]–[47] aims at creating a minimum cost multicast topology on multiple streams, but they have not considered the O2E delay. Meanwhile, the scheme in [45] assumes a homogeneous bandwidth pricing schemes across all geographical locations, which only applies to the network with fixed unit price of data transmission from server to server. By contrast, the model we consider is general, realistic and comprehensive, capturing the cost of auto-scaling servers and link capacities with heterogeneous pricing. To address QoE, we also consider the major components of O2E delay as a bi-criteria problem, and propose a bi-criteria approximation algorithm with proven approximation ratio.

III. BI-CRITERIA PROBLEM FORMULATION AND ITS NP-HARDNESS

Our bi-criteria problem is to minimize deployment cost and O2E delays, which is equivalently posed as minimum cost streaming with given delay constraints. In this section, we first present the formulation of bi-criteria problem in Section III-A, followed by proving its NP-hardness in Section III-B. We show the major symbols used in this section in Table II.

A. Cost Minimization with Delay Constraints

We model the live streaming cloud overlay as a directed complete graph $G = (V, E)$, where V is the set of vertices corresponding to both origin and end servers. (If we cannot set up an overlay link between 2 servers, we can set the corresponding link price to a very large value so that any practical solution will not use this link.) Let S be the set of origin servers and R be the set of end servers, where $V = S \cup R$. Without loss of generality, we consider that the origin servers have no local users, i.e., $S \cap R = \emptyset$ (If not, we may split the server into two parts, one being the origin server with all the channels and the other being the end servers with all the local users. The two parts are then connected by a link of zero cost, zero delay and infinite bandwidth). We denote $\langle i, j \rangle$ as the directed overlay link from server i to j, and let $E \subseteq V \times V$ be the set of overlay links between servers (i.e., $\langle i, j \rangle \in E$).

Let M be the set of channels and $s(m) \in S$ be the origin server of channel $m \in M$ and $\tau(m)$ be the streaming rate of channel m. We denote $R(m)$ as the set of end servers that demand channel m. Each channel $m \in M$ is delivered to all

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

TABLE II
MAJOR SYMBOLS USED IN THE FORMULATION

Notation	Definition
S	The set of origin servers
R	The set of end servers
V	The set of both origin and end servers ($V = S \cup R$)
$\langle i,j \rangle$	The directed overlay link from server i to j
E	The set of all overlay links
M	The set of all channels
$R(m)$	The set of end servers that demand channel m
M_i	The set of channels that end server i demands
$\tau(m)$	Streaming rate of channel m (bit/s)
$s(m)$	Origin server of channel m
u_i	Uploading rate at server i (bit/s)
b_{ij}	Transmission rate through link $\langle i,j \rangle$ (bit/s)
$T(m)$	The delivery tree of channel m
$x_{ij}(m)$	Binary variable indicating whether link $\langle i,j \rangle$ is in $T(m)$
L_{ij}	Server-to-Server (S2S) delay of link $\langle i,j \rangle$ (in second)
$D_i(m)$	Origin-to-End (O2E) delay of channel m at server i (in second)
$\mathbb{D}_i(m)$	O2E delay upper bound of channel m at server i (in second)
Θ_i	Server cost of server i (per second)
θ_i	Unit price of uploading streaming at server i (per bit)
Φ_{ij}	Link cost due to traffic through link $\langle i,j \rangle$ (per second)
ϕ_{ij}	Unit price of data transmission through link $\langle i,j \rangle$ (per bit)
C	Total deployment cost (per second)

servers in $R(m)$ through a delivery tree $T(m)$, and hence we have a total of $|M|$ delivery trees. Note that in this delivery tree $T(m)$, the origin server $s(m)$ is the root and all the other end servers form the set $R(m)$.

Let $x_{ij}(m)$ be a binary variable indicating whether link $\langle i,j \rangle$ is used in the delivery tree $T(m)$, so we have

$$x_{ij}(m) = \begin{cases} 1, & \text{if } \langle i,j \rangle \in T(m); \\ 0, & \text{otherwise .} \end{cases} \tag{1}$$

As every end server $i \in R(m)$ needs to get channel m, we have

$$\sum_{\langle i,j \rangle \in E} x_{ij}(m) \geq 1, \quad \forall j \in R(m), m \in M. \tag{2}$$

To mathematically specify that $T(m)$ is a tree structure, we have to ensure that $T(m)$ has $|R(m)|$ edges, i.e.,

$$\sum_{\langle i,j \rangle \in E} x_{ij}(m) = |R(m)|, \quad \forall m \in M, \tag{3}$$

and $T(m)$ has no cycle, which can be equivalently expressed as that, for any subset of the vertices of $T(m)$ denoted as T', we must have at most $|T'|-1$ edges in this subset, namely,

$$\sum_{i,j \in T', \langle i,j \rangle \in E} x_{ij}(m) \leq |T'|-1, \quad \forall T' \subseteq T(m), m \in M. \tag{4}$$

For each link $\langle i,j \rangle \in E$, the transmission rate b_{ij} through the link is given as

$$b_{ij} = \sum_{m \in M} x_{ij}(m)\tau(m), \quad \forall \langle i,j \rangle \in E, \tag{5}$$

and the uploading rate u_i at server $i \in V$ to serve other end servers is therefore given as

$$u_i = \sum_{\langle i,j \rangle \in E} b_{ij}, \quad \forall i \in V. \tag{6}$$

We denote the server-to-server (S2S) delay of link $\langle i,j \rangle$ as L_{ij} and the origin-to-end (O2E) delay of server j in tree $T(m)$ as $D_j(m)$, which is equal to the delay of its direct parent i in tree $T(m)$ plus the S2S delay of link $\langle i,j \rangle$, i.e.,

$$D_j(m) = D_i(m) + L_{ij}, \quad \text{if } \langle i,j \rangle \in T(m). \tag{7}$$

By definition, the O2E delay is 0 for the origin server of each channel (i.e., $D_i(m) = 0, \forall i = s(m), m \in M$).

To guarantee the QoE, we have to ensure that, for an end server i that demands channel m, the O2E delay $D_i(m)$ is bounded by a given parameter termed as O2E delay upper bound $\mathbb{D}_i(m)$, i.e., we must have

$$D_i(m) \leq \mathbb{D}_i(m), \quad \forall i \in R(m), m \in M. \tag{8}$$

The deployment cost consists of server and link cost. We adopt a general linear cost model, which is widely used in practice (e.g., a very popular 95th-percentile pricing scheme [48] is a special case of linear cost). Let Θ_i be the *server cost* of server i, and θ_i be its unit price of uploading streaming. Θ_i is charged by the server uploading rate u_i of server i, i.e.,

$$\Theta_i = \theta_i u_i, \quad \forall i \in V. \tag{9}$$

Let Φ_{ij} be the *link cost* of link $\langle i,j \rangle$, and ϕ_{ij} be its unit price of data transmission. Φ_{ij} depends on the transmission rate b_{ij} of link $\langle i,j \rangle$, i.e.,

$$\Phi_{ij} = \phi_{ij} b_{ij}, \quad \forall \langle i,j \rangle \in E. \tag{10}$$

Note that $\Theta_i = 0$ indicates that server i is not in use, and similarly $\Phi_{ij} = 0$ indicates that link $\langle i,j \rangle$ is not in use.

The total deployment cost C is the sum of server cost and link cost, given by

$$C = \sum_{i \in V} \Theta_i + \sum_{\langle i,j \rangle \in E} \Phi_{ij}. \tag{11}$$

We state our cost optimization problem of Minimum Cost Streaming with Delay Constraints (MCSDC) as follows: given overlay topology G, origin server $\{s(m)\}$, end server set $\{R(m)\}$, S2S delay $\{L_{ij}\}$, unit price of server uploading streaming $\{\theta_i\}$ and link data transmission $\{\phi_{ij}\}$, and O2E delay upper bounds $\{\mathbb{D}_i(m)\}$, we seek to find the overlay construction $\{x_{ij}(m)\}$ for streams of all channels $m \in M$ to minimize the total deployment cost C subject to constraints from (2) to (10).

B. The NP-Hardness of MCSDC Problem

MCSDC is NP-hard since the NP-hard Shallow-Light Spanning Tree (SLST) Problem [49] is reducible to our MCSDC problem.

We state the SLST problem as follows. In a directed graph $G(V,E)$, each link $\langle i,j \rangle \in E$ has an associated positive cost Φ_{ij} and a positive delay L_{ij}. Let $s \in V$ be the origin vertex, and for the other vertices $v \in V$ we associate each vertex with a given delay upper bound $\mathbb{D}_v$. SLST is to find the spanning tree from s as the root in G with minimum total cost such that the delay measured from origin vertex s to any vertex v along this spanning tree is no greater than $\mathbb{D}_v$.

TABLE III
MAJOR SYMBOLS USED IN COCOS ALGORITHM

Notation	Definition
$\alpha+\delta$	COCOS approximation ratio of the deployment cost
β	COCOS approximation ratio of the O2E delay
k	Number of substreams for a channel
C_{SO}	Deployment cost of LP super optimum solution (per second)
C_{EO}	Deployment cost of exact optimum solution (per second)
C_{ID}	Deployment cost of COCOS given $k\to\infty$ (per second)
C_{CC}	Deployment cost of COCOS (per second)
$z_{ij}(m)$	Fractional stream of channel m through link $\langle i,j\rangle$
$f_{ij}^{l}(m)$	Flow fraction of channel m to server l through link $\langle i,j\rangle$
$u_i(m)$	Uploading rate of channel m at server i in LP (bit/s)
$b_{ij}(m)$	Transmission rate of channel m through link $\langle i,j\rangle$ in LP (bit/s)
$C(m)$	Deployment cost due to channel m in LP (per second)
k	Number of substreams for a channel
$\Psi(m)$	The set of substreams of channel m
$\Gamma(\psi)$	The delivery tree of substream ψ
$n_{ij}(m)$	Number substreams allowed on link $\langle i,j\rangle$ for channel m

It is straightforward to see that SLST problem is a special case of MCSDC problem with only one origin server corresponding to one channel and all the other end servers demanding this channel. Meanwhile, the server cost in the graph is omitted, and we only consider the S2S delay and link cost. Hence, it is clear that SLST is polynomial-time reducible to MCSDC.

IV. COCOS: A BI-CRITERIA APPROXIMATION ALGORITHM FOR AN AUTO-SCALING LIVE CLOUD

In this section, we present a bi-criteria approximation algorithm termed **C**ost-optimized Multi-**O**rigin Multi-**C**hannel **O**verlay **S**treaming (COCOS). We first present the overall idea and basic concepts of bi-criteria approximation in Section IV-A, and reformulate the original MCSDC problem as a Linear Programming (LP) problem in Section IV-B. Then, in Section IV-C, we present the COCOS algorithm to construct low-cost and QoE-assured delivery trees for each channel based on the LP solution. We analyze the algorithmic complexity and prove the approximation ratio of COCOS in Section IV-D. We show the major symbols used in this section in Table III.

A. Preliminaries for Bi-criteria Approximation

1) Optimal Solution: We first discuss the optimal solution of the MCSDC problem. As we have shown in Section III-B that the original MCSDC problem is NP-hard, it is very unlikely to find the *exact* optimal solution efficiently in both theory and practice. In Section IV-B, we reformulate the original MCSDC problem as an efficiently solvable LP problem by relaxing some constraints so we can treat each channel as an *infinitesimally divisible* stream (i.e., we can split a channel stream into an arbitrary number of fractional streams). We denote the *optimal* solutions of the MCSDC problem and the LP problem as the *exact optimum* and the *super optimum*, and their corresponding deployment cost as C_{EO} and C_{SO}, respectively. A *feasible* (but not necessarily optimal) solution of the original MCSDC problem must be a feasible solution of the LP problem, and thus the deployment cost of the exact optimum cannot be better than the super optimum (i.e., $C_{\mathrm{SO}}\le C_{\mathrm{EO}}$). Note that, as we cannot get the exact optimum, we only compare COCOS with the super optimum. If we can give an approximation ratio with respect to the super optimum, this ratio must also hold for the exact optimum.

2) Approximation Ratio: We then briefly introduce the approximation ratio of this bi-criteria problem. As the bi-criteria problem of MCSDC considers both the deployment cost and the O2E delay, to address the optimality of COCOS as an approximation algorithm, we have to show that the difference between the solution given by COCOS and the super/exact optimum is bounded by a specific ratio in terms of both cost and delay. With the deployment cost given by COCOS as C_{CC}, we denote the approximation ratio of the deployment cost as $\alpha+\delta$ and that of the O2E delay as β, which means that $C_{\mathrm{CC}}\le\alpha C_{\mathrm{EO}}$ and the delay at each end server for channel m is at most β times the optimum solution.

Specifically, COCOS achieves $\alpha=1+\varepsilon$ and $\beta=1+1/\varepsilon$ where ε, as a positive real number, is given as a tunable parameter that allows tradeoff between the optimality of cost and delay, and the positive real number δ is also a tunable parameter that allows tradeoff between cost and algorithmic time complexity, which can be arbitrarily close to 0 at the expense of computation time. Note that by cancelling ε in the equations, we have $1/\alpha+1/\beta=1$. We evaluate the impact of ε through experiments in Section V-B.

3) Approximation Solution: We outline the overall structure of the algorithm of COCOS that constructs the core network overlay from the LP solution. When computing the LP solution (i.e., the super optimum), we first let the delay constraints in LP formulation be $1/\beta$ of the given delay upper bound $\mathbb{D}_i(m)$ such that, even with the O2E delay approximation ratio β, COCOS can still meet the delay constraints given in (8) of the original MCSDC problem formulation in Section III-A. To efficiently approximate the LP solution, we then construct a k-substream solution as an intermediate step such that we evenly divide a channel stream into k substreams, and the substream solution approaches the super optimum with the cost approximation ratio α as k goes to infinity ($k\to\infty$). We call this the *COCOS-ID* solution as it would be the solution of COCOS if we allow all the streams to be infinitesimally divisible, and we denote the corresponding deployment cost as C_{ID}. For a finite k, we can get k candidate substream delivery trees, and we choose the topology of the substream tree with the lowest cost as the final solution of COCOS.

We give an example of substream solution in Fig. 2. Unlike the original system in Fig. 1, here both Channel 1 and 2 have 2 substreams. (We have no upper limit for number of substreams, and here we choose $k=2$ substreams only for simplicity.) Each channel is evenly divided into 2 substreams, and every end server that demands this channel shall get both substreams.

4) Theoretical Proof: We finally outline the proof of the approximation ratio. To prove the cost approximation ratio of COCOS, we show that $C_{\mathrm{ID}}\le\alpha C_{\mathrm{SO}}$ (thus $C_{\mathrm{ID}}\le\alpha C_{\mathrm{EO}}$). In Section IV-D, we further prove that the gap between C_{ID} and C_{CC} can be arbitrarily close to 0 (i.e., $\delta\to 0$) as k increases, which finally shows the deployment cost approximation ratio $\alpha+\delta$. Note that our theoretical approximation ratio gives

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

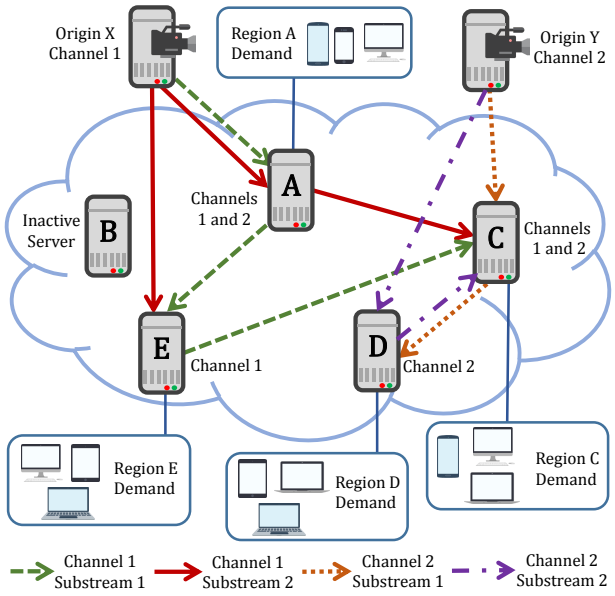


Fig. 2. An example of substream solution for $k = 2$.

the performance guarantee in the worst-case scenario, which rarely happens in practice. In Section V-B, we show that, under practical setting, COCOS gives near-optimal solution through trace-driven experiments based on real-world data.

B. Relaxing MCSDC Problem to an LP Formulation

The solution of the LP formulation serves as the super optimum and the basis of constructing the approximation solution in COCOS. Instead of considering the deployment cost C as a whole, we can decompose the original multi-origin multi-channel problem, and optimize each channel separately and independently. Specifically, we denote $u_i(m)$ as the uploading rate of channel m at server i, $b_{ij}(m)$ as the transmission rate of channel m through link $\langle i, j\rangle$, and $C(m)$ as the deployment cost due to channel m.

Furthermore, instead of considering the channel stream as a whole in the LP formulation, we treat it as an infinitesimally divisible stream. By relaxing (1), we use a continuous variable $z_{ij}(m)$ to replace the binary variables $x_{ij}(m)$ for $m \in M$. Specifically, we let $z_{ij}(m)$ be the fraction of the stream of channel m that transmitting through link $\langle i, j\rangle$, and we have $0 \le z_{ij}(m) \le 1$ by definition. Hence,

$$b_{ij}(m) = z_{ij}(m)\tau(m), \quad \forall \langle i, j\rangle \in E, m \in M. \tag{12}$$

Accordingly, we also have

$$u_i(m) = \sum_{\langle i,j\rangle \in E} b_{ij}(m), \quad \forall i \in V, m \in M, \tag{13}$$

and the deployment cost due to channel m is given as,

$$C(m) = \sum_{i \in V} \theta_i u_i(m) + \sum_{\langle i,j\rangle \in E} \phi_{ij} b_{ij}(m), \quad \forall m \in M. \tag{14}$$

We further replace the constraints in (2) by using the property of flow conservation. We denote $f_{ij}^l(m)$ as the fraction of conceptual stream flow of channel m from the origin server $s(m)$ to a destination end server $l \in R(m)$ via link $\langle i, j\rangle$. By considering the flows that go out of and come into the server j, we have

$$\sum_{\langle j,k\rangle \in E} f_{jk}^l(m) - \sum_{\langle i,j\rangle \in E} f_{ij}^l(m) = \begin{cases} 1, & \text{if } j = s(m); \\ -1, & \text{if } j = l; \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

In other words, we only have outgoing flow for the origin server $s(m)$, and only have incoming flow for the destination end server l. For the other servers, the incoming flow shall be equal to the outgoing flow. As $f_{ij}^l(m)$ only consider the flow to end server l, for the transmission rate of channel m on link $\langle i, j\rangle$, we have

$$0 \le f_{ij}^l(m) \le z_{ij}(m) \le 1, \quad \forall l \in R(m), \langle i, j\rangle \in E. \tag{16}$$

Instead of letting every fractional flow satisfy the delay upper bound, we relax (8) such that we only let the average delay of all flows satisfy the delay upper bound where we can write the average delay for fractional stream as $\sum f_{ij}^l(m) L_{ij}$. Meanwhile, we let the delay upper bound be $\mathbb{D}_l(m)/\beta$ so that COCOS can still satisfy the delay upper bound $\mathbb{D}_l(m)$ given in (8) even with the delay approximation ratio β. So we have

$$\sum_{\langle i,j\rangle \in E} f_{ij}^l(m) L_{ij} \le \frac{1}{\beta} \mathbb{D}_l(m), \quad \forall l \in R(m), m \in M. \tag{17}$$

Note that a very large β or a too small $\mathbb{D}_l(m)$ may let the whole problem have no feasible solution. Thus, we have to set an appropriate $\mathbb{D}_l(m)$ to avoid such situation.

It is straightforward that $C_{\text{SO}} = \sum_{m \in M} C(m)$. Hence, to minimize C_{SO}, we just need to optimize the LP formulation for each channel m independently whose objective is minimizing $C(m)$ given the constraints (12) to (17).

C. Overlay Construction from LP Solution

To efficiently approximate the LP solution, we first construct a substream solution where we split each channel evenly into k substreams, and thus each substream has a streaming rate of $\tau(m)/k$. With more substreams, we can make the cost approximation ratio of the substream solution closer to α. Especially, when k goes to infinity, our substream solution is infinitesimally divisible, and thus the approximation ratio is α (i.e., $\delta \to 0$).

To construct the overlay topology for finite numbers of substreams k, we first compute how many substreams can be allocated in each link. As our proposed cost approximation ratio for C_{ID} is α, for channel m on link $\langle i, j\rangle$, the transmission rate is therefore given as $\alpha b_{ij}(m)$, and the uploading rate at server i is given as $\alpha u_i(m)$. Denoting the number of substreams of channel m on link $\langle i, j\rangle$ as $n_{ij}(m)$, we further round up b_{ij} a little bit so that we have integral number of substreams, and thus we have

$$n_{ij}(m) = \lceil \alpha k b_{ij}(m)/\tau(m) \rceil. \tag{18}$$

Given $n_{ij}(m)$, we start to construct k delivery trees for channel m. We denote the set of substreams of channel m

as $\Psi(m)$, and delivery trees of the substream $\psi \in \Psi(m)$ as $\Gamma(\psi)$. To construct one tree, we set the origin server $s(m)$ as the root, and use Dijkstra's algorithm to include all the end servers in $R(m)$ such that, in this substream delivery tree, each end server achieves the minimum delay with the links who have $n_{ij}(m) > 0$.

After constructing a substream delivery tree, if we have used link $\langle i, j\rangle$, we deduct $n_{ij}(m)$ by 1. We repeat the Dijkstra's algorithm until we have all the k substream delivery trees. Note that this algorithm is extended from the greedy arborescence packing algorithm introduced in [50], which also gives the proof that our algorithm can correctly generate k trees. We give the pseudocode of the whole process of constructing all k delivery trees in Algorithm 1.

Algorithm 1: Substream Delivery Trees

```
foreach ψ ∈ Ψ(m) do
    source ← s(m)
    nodes ← R(m)
    links ← {⟨i,j⟩ | n_ij(m) > 0, ⟨i,j⟩ ∈ E}
    Γ(ψ) = Dijkstra(source, nodes, links)
    foreach ⟨i,j⟩ ∈ E do
        if ⟨i,j⟩ ∈ Γ(ψ) then
            n_ij(m) −= 1
        end
    end
end
```

Given these k substream delivery trees for channel m, we choose the substream $\psi \in \Psi(m)$ whose delivery tree $\Gamma(\psi)$ has the minimum cost, and deliver the whole channel through this tree (i.e., we let $T(m) = \Gamma(\psi)$). As this tree has minimum cost among the substreams, the final single stream solution of COCOS is no worse than the substream solution.

Note that it is very likely that this algorithm would first generate substream delivery trees with very high cost despite of low delay. Therefore, to find a good approximation solution for this bi-criteria optimization problem with balanced cost and delay, we have to generate enough trees as candidates.

D. Algorithmic Complexity and Approximation Ratio

For the algorithmic complexity of solving the LP problem, it has been proven that this method has $O(N^3)$ overall time complexity, where N is the number of variables in the linear program [51]. As there are $|V|^3$ variables in the linear program for each channel, the time complexity of LP is $O(|V|^9)$ for one channel, and $O(|V|^9|M|)$ for the whole problem.

In the overlay topology construction step, the major part of algorithmic time complexity is to compute all the substream delivery trees, which runs in $O(|E| + |V|\log(|V|)$ time for the Dijkstra's algorithm to generate a substream delivery tree. As we are considering a complete graph where $O(|E|) = O(|V|^2)$, and we have $k|M|$ substreams in total for all the channels, COCOS runs in $O(|V|^9|M| + k|V|^2|M|)$ time in total. With a moderate number of k, the predominant term is $O(|V|^9|M|)$.

For a typical large-scale real-world system (e.g., the system under study in our trace-driven experiments in Section V), it takes less than a minute to run the algorithm on a normal desktop PC. The frequency to execute the algorithm depends on the time scale of the system parameters. Normally, as the parameters vary very little within an hour, the time scale is in the order of an hour.

To prove the the delay approximation ratio β, we first show that COCOS can generate k substream trees that satisfy the O2E delay constraint in (8). In the LP formulation, we require that the average delay of channel m at end server i is bounded $\mathbb{D}_i(m)/\beta$. By considering the Markov's inequality, at least $1/\alpha = 1 - 1/\beta$ of the fractional stream must satisfy the delay bound $\mathbb{D}_i(m)$. Therefore, as COCOS allocates the resource that can accommodate $\lceil \alpha k \rceil$ substreams, at least we have k substreams whose delay are bounded by $\mathbb{D}_i(m)$. Note that, because the Markov's inequality describes the worst case, usually more than $1/\alpha$ fraction of flow is bounded by $\mathbb{D}_l(m)$.

For the cost approximation ratio $\alpha+\delta$, we prove it in 2 steps. We first show that $C_{\rm ID} \leq \alpha C_{\rm EO}$. It is because that, in COCOS-ID, we directly let all the flow fraction parameter $f^l_{ij}(m)$ be α times of the original value given by the super optimum so that each end server can get enough fraction of the stream that satisfies the O2E delay constraint in (8). Consequently, u_i and b_{ij} are at most α times of the super optimum (i.e., $u_i = \alpha \sum_{m\in M} u_i(m)$ and $b_{ij} = \alpha \sum_{m\in M} b_{ij}(m)$), and the deployment cost $C_{\rm ID}$ is at most $\alpha C_{\rm SO}$. As $C_{\rm SO} \leq C_{\rm EO}$, it is clear that $C_{\rm ID} \leq \alpha C_{\rm EO}$.

We finally show that $\delta \to 0$ as we increase k to infinity. In fact, the gap between COCOS and COCOS-ID is due to that we have to round up the link and server resources to ensure that an integral number of substreams in a link or a server. As the resource to round up is at most for one substream whose bitrate is $\tau(m)/k$, with a greater k, each substream requests less resource, and when k goes to infinity, the substream bitrate $\tau(m)/k$ goes to 0. In Section V, we show that, with a moderate number of k (e.g., $k = 10$), the gap between $C_{\rm CC}$ and $C_{\rm ID}$ is negligible, and COCOS achieves near-optimal performance.

V. Data-driven Experimental Results

In this section, we first present our trace-driven experimental environment and performance metrics in Section V-A. Then we discuss the illustrative results based on real-world video service data in Section V-B.

A. Experimental Setup and Performance Metrics

We have implemented our experiments based on real-world network topology and user traces to study our algorithm. The experiments are carried out on a real Internet topology provided by CAIDA. The round trip times (RTTs) between inter-connected routers are also given in the topology. In underlay routing, we use distance-vector to compute the S2S delay between any two servers in the network. To generate the experimental environment, origins and end servers are randomly attached to the router nodes in this live streaming network. From Fig. 5 to 10, the regional demand of channels is based on real-word data trace from a leading video service

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

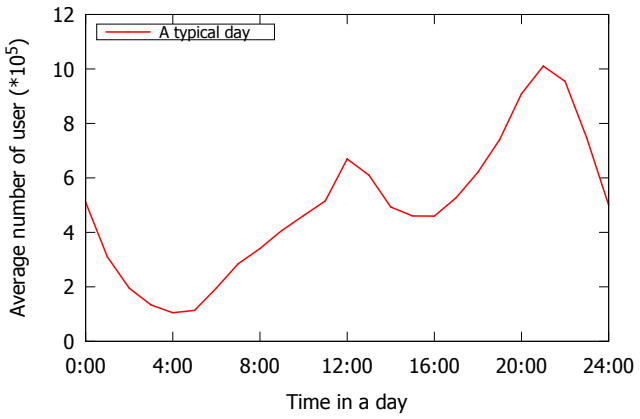


Fig. 3. Average user number over a typical day.

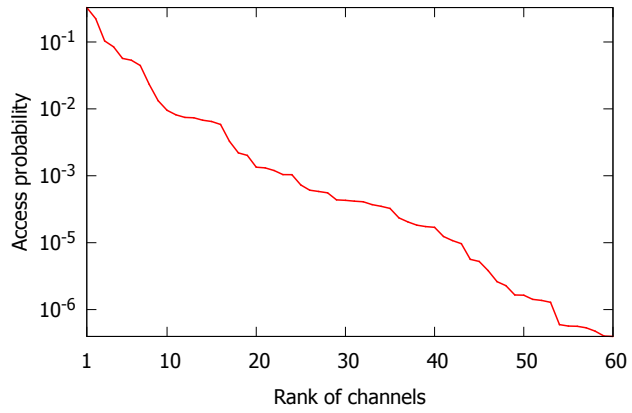


Fig. 4. Access probability of the channels.

website in China over 2 weeks. We give the average user number over a typical day in Fig. 3 and the access probability of the channels in Fig. 4. We re-optimize the system every hour, and take the average of the deployment cost in each hour as the result.

As COCOS is applicable to any channel popularity distribution and network environment, to further validate COCOS's performance, we also use synthetic data for the regional demand in our experiments from Fig. 11 to 15 where the channel popularity follows the Zipf's distribution. With Zipf's parameter z, the mth popular channel has the demanded server number $R(m) \propto 1/m^z$. Channels are randomly assigned to the servers. A greater Zipf's parameter indicates that a popular channel has more servers demanding it and an unpopular channel has fewer servers demanding it.

We show the baseline parameters in Table IV. Unless otherwise stated, we use the following parameters in our experiments: number of origin and end servers $|V| = 100$, number of channels $|M| = 60$, and delay upper bound $\mathbb{D} = 800$ms. The streaming rates of the channels have a mean of 1.2 Mbps and a standard deviation of 0.2 Mbps. The prices of link data transmission have a mean of 0.1 per Mbit and a standard deviation of 0.05 per Mbit. The prices of server uploading streaming have a mean of 0.1 per Mbit and a standard deviation of 0.05 per Mbit.

TABLE IV
BASELINE PARAMETERS USED IN OUR STUDY.

Parameter	Value
Server number (origin and end) $\|V\|$	100
Number of channels $\|M\|$	60
Delay upper bound $\mathbb{D}$	800 ms
Streaming rate mean μ_τ	1.2 Mbps
Streaming rate standard deviation σ_τ	0.2 Mbps
Server price mean μ_θ	0.1 per Mbit
Server price standard deviation σ_θ	0.05 per Mbit
Link price mean μ_ϕ	0.1 per Mbit
Link price standard deviation σ_ϕ	0.05 per Mbit
Zipf's parameter z	0.5
Tradeoff parameter ε	5
Number of substreams k	10

As mentioned in Section II, previous work seldom considers the bi-criteria problem of minimizing deployment cost and O2E delays. To capture all the important components, we extend some of the traditional and state-of-the-art work as comparison schemes.

- *Nearest Peer* [28], [32]: which is an overlay construction algorithm used in many state-of-the-art work, whose objective is to minimize the local streaming latency. With minor modification, we can easily adapt this algorithm into our network setting. A server gets its demanding channels from the origin or another end server so that its peer-to-peer delay is minimized.
- *Prim*: which is a well-known optimization algorithm for minimum cost delivery tree construction. However, it does not consider the delay constraints. To address this, after the construction of the deliver tree through Prim, if an end server violates the delay constraint, it gets the stream from another server so that it can meet the delay constraint with minimum cost.
- *Super optimum*: which serves as the theoretical performance bound (i.e., no scheme performs better than super optimum). The super optimum in this work is the optimal solution of the LP formulation from Section IV-B.

We evaluate the performance of our proposed algorithm and the comparison schemes mainly by several delay and cost metrics:

- *Deployment cost*, which is the sum of server cost and link cost according to (11). This is the deployment cost of the whole live streaming cloud.
- *Cost component*, which consists of server cost and link cost. We are also interested in each cost component as they reflect how the optimization works.
- *Delay Constraint*, which is the maximum O2E delay allowed in this system.

B. Illustrative Experimental Results

We compare in Fig. 5 the total deployment cost versus the approximation ratio tradeoff parameter ε. A smaller ε indicates a smaller cost approximation ratio α but a greater delay approximation ratio parameter β. With small ε, though we have a small α, the delay upper bound for super optimum

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

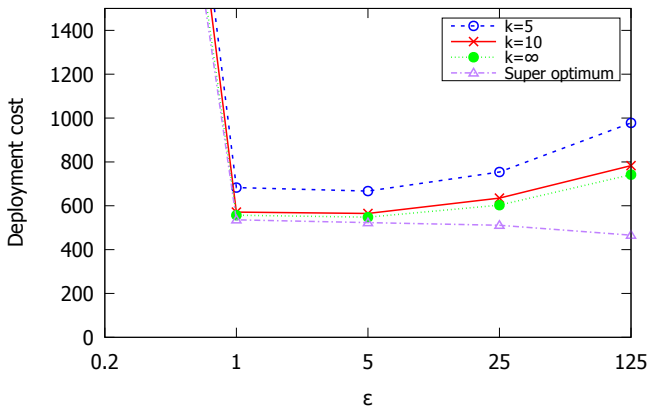


Fig. 5. Deployment cost versus approximation ratio tradeoff parameter.

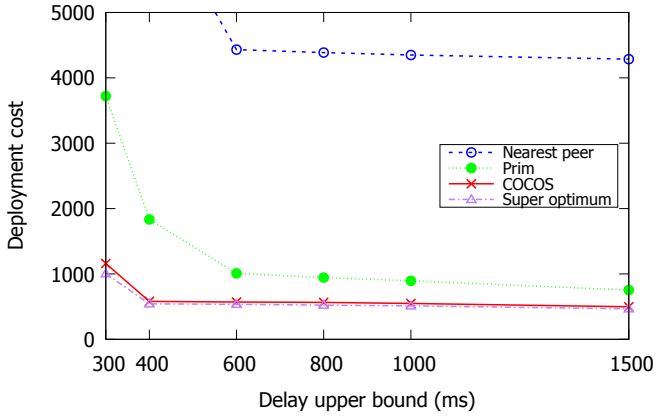


Fig. 7. Deployment cost versus delay upper bound given different schemes.

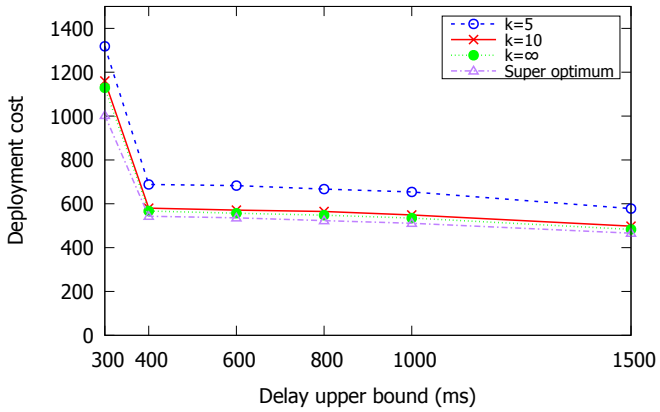


Fig. 6. Deployment cost versus delay upper bound given different number of substreams.

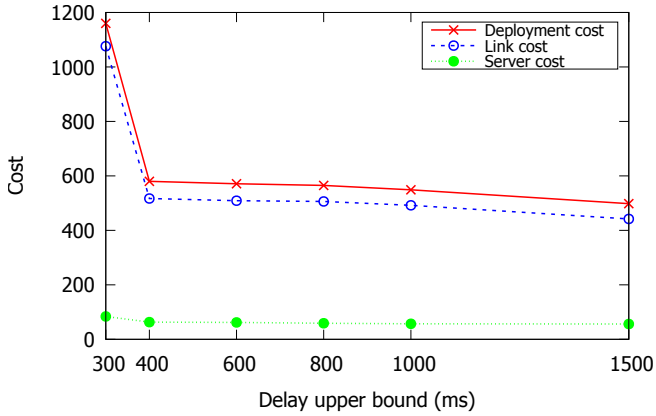


Fig. 8. Components of deployment cost versus delay upper bound for COCOS.

is tighter and leads to a much greater C_{SO}. With great ε, a great α causes the cost of COCOS to increase though we have a small C_{SO}. Therefore, both a too small and a too great ε can impede the optimality of COCOS. The deployment cost of COCOS is closer to the super optimum rather than the upper bound given by the approximation ratio (i.e., C_{CC} is closer to C_{SO} rather than αC_{SO}), which shows that it is more likely that COCOS achieves near-optimal performance in practice.

We compare in Fig. 6 the total deployment cost versus the O2E delay upper bound for different number of substreams k. As k increases, the deployment cost approaches C_{ID} given by COCOS-ID (i.e., $k \to \infty$). With humble value of k (say $k = 10$), the performance is already very close to C_{ID}. This shows that with reasonable computation time, COCOS can achieve near-optimal performance.

We compare in Fig. 7 the total deployment cost versus the O2E delay upper bound for different schemes. Total cost increases with a tighter delay constraint. COCOS clearly achieves the lowest deployment cost as the gap between COCOS and other schemes is usually beyond 100 percent. When the delay constraint is relaxed to some extent (e.g. $>$ 1000ms), the cost of all the schemes remains steady as the delay constraint can be easily satisfied with an arbitrary overlay topology. However, in this case, the deployment cost of COCOS is still significantly lower than the comparison schemes. The deployment cost of Prim decreases with a loose delay constraint as more cheap links can be used at the cost of S2S delay. For Nearest Peer, as S2S delay weighs more in its delay components, its deployment cost is not sensitive to the change of delay constraint.

We show in Fig. 8 the components of deployment cost versus delay upper bound for COCOS. Server cost remains steady as the delay constrain changes, but the link cost first decreases with a larger delay constraint, and then keeps steady after the delay constraint exceeds a certain extent. This trend of cost component shows that a higher QoE constraint demands mainly on links with small S2S delay despite the cost.

We compare in Fig. 9 the deployment cost versus average link price given different schemes. The deployment cost for all the schemes increase with the increasing of the link price, but the increasing trends of the deployment cost of COCOS and Prim are not that steep when the price is too high or too low. Such trend shows the effect of the tradeoff between cost and delay. When the link price is higher, the overlay topology tends to use cheap link. On the other hand, when the link price is low, the price difference between links is also small. Note that with a higher link price, the gap of deployment cost between COCOS and other comparison schemes become

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

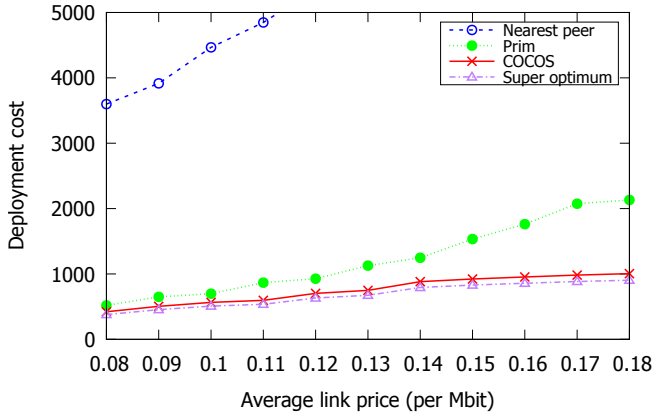


Fig. 9. Deployment cost versus average link price given different schemes.

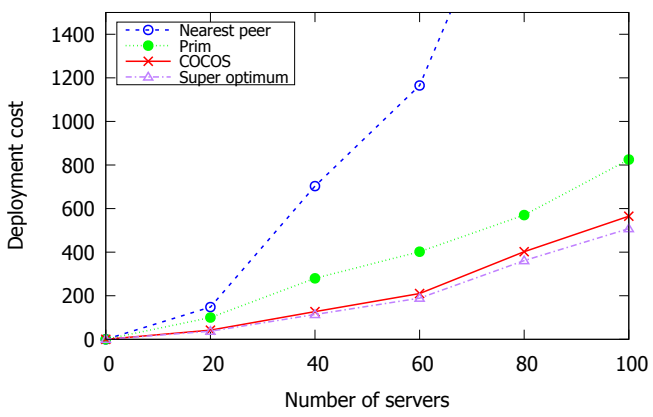


Fig. 11. Deployment cost versus number of servers given different schemes.

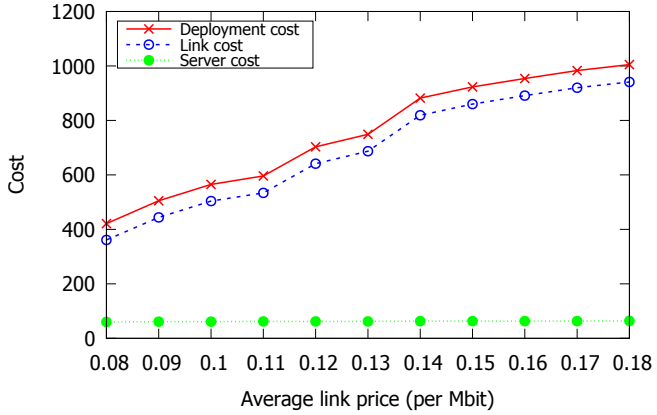


Fig. 10. Components of deployment cost versus average link price for COCOS.

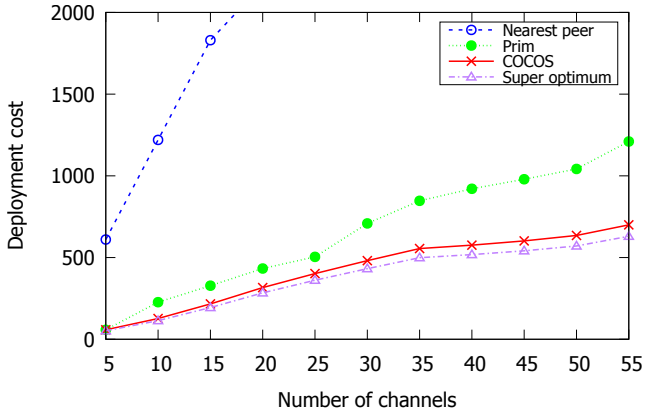


Fig. 12. Deployment cost versus number of channels given different schemes.

larger, which shows that COCOS has a stronger capability of finding and using cheap resource. The cost of Nearest Peer increases sharply as it has a rigid topology construction step and cannot effectively use cheap resource. On the other hand, Prim has a more flexible topology construction method and is able to find some cheap resource.

We show in Fig. 10 the components of deployment cost versus average link price for COCOS. The server cost remains nearly unchanged as the link price increases, but the link cost increases, which contributes most of the increase of the total deployment cost. As the server number and the number of demanded channels at each server do not change, the workload to deliver live content keeps same. Therefore, the demand of server uploading remains almost steady.

We show in Fig. 11 the deployment cost versus the number of servers given different schemes. The deployment costs for COCOS and Prim increase moderately with the increasing of the number of servers, but the cost increment of Nearest Peer is sharper with a larger server number. With more servers and more total demands for channels, we need more links to cover all the demands and deliver the live contents. Cost of Nearest Peer increases more sharply compared with other 2 schemes because it always tries to deliver content to its nearest peers despite the cost. With more servers, its overlay topology will be unnecessarily expensive.

We plot in Fig. 12 the deployment cost versus the number of channels. The deployment cost increases as the channel number increases for all schemes. The number of demanded channels for each server increases with the rise of the number of channels. COCOS enjoys lower deployment cost because it comprehensively considers the cost components and delay constraints by balancing between them, and constructing connectivity with low cost through server collaboration. On the other hand, with more channel number, Nearest Peer will blindly deliver channels to its neighbor peers in some scenario. Some unpopular channels cause extra cost and can be delivered more efficiently with some longer and direct links with less hops.

We show in Fig. 13 the deployment cost versus average streaming bitrate given different schemes. The deployment cost for all the schemes increase with the increasing of the average streaming rate. Obviously, with higher streaming rate, more resource is used in links and servers. With the link price fixed, this increment of streaming rate has little impact on the topology. Therefore, the deployment cost increases almost linearly with the streaming bitrate.

We plot in Fig. 14 the deployment cost versus average number of demanded channels per each server given different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3152093, IEEE Transactions on Multimedia

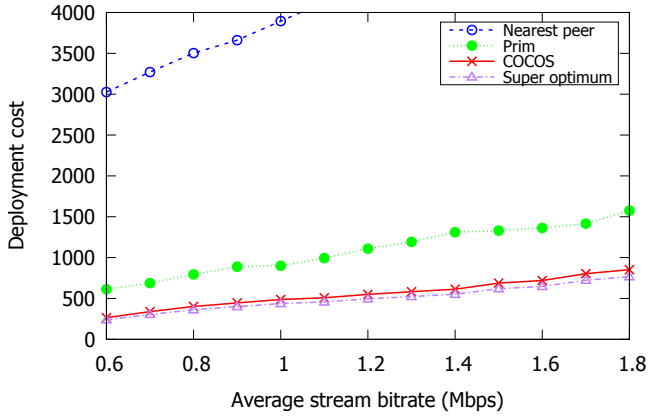


Fig. 13. Deployment cost versus average streaming bitrate given different schemes.

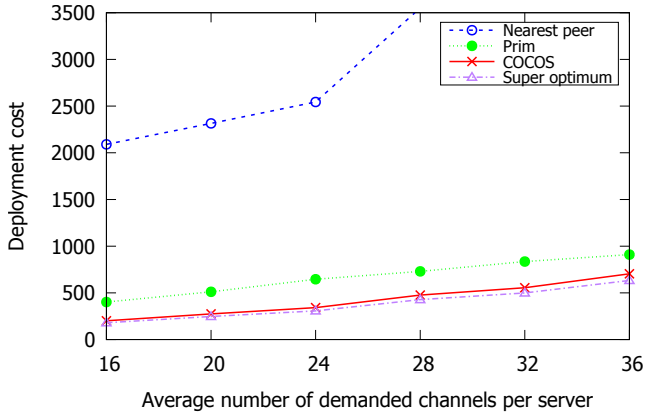


Fig. 14. Deployment cost versus average number of demanded channels per server given different schemes.

schemes to validate COCOS under different levels of traffic. The deployment cost for all the schemes increase with the increasing of demanded channel number for each server given different schemes. More demanded channel on a server will not change too much on the topology but will ask for more resource to ensure the QoE. Therefore, more deployment cost is required.

We show in Fig. 15 the deployment cost versus the Zipf's parameter given different schemes. The deployment cost for all the schemes decreases with the increasing of the Zipf's parameter. With a higher Zipf's parameter, the number of servers that demands cold channels is decreased. As the increment of deployment cost for popular channels is limited (at most all servers demand it), the decreasing of cold channel demand will decrease the total deployment cost.

VI. Conclusion

Auto-scaling cloud computing can elastically rescale system resources to support dynamic video traffic. We have studied a novel multi-origin multi-channel auto-scaling live streaming cloud where each channel stream is pushed in a delivery tree covering the end servers that demand the channels. We consider a pay-as-you-go cost model where the deployment

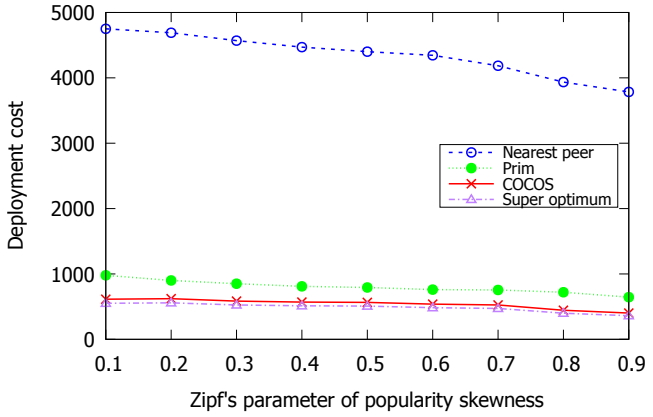


Fig. 15. Deployment cost versus Zipf's parameter of popularity skewness given different schemes.

cost is charged by the actual amount of resources used due to server uploading and data transmission between servers. Our problem is bi-criteria in nature as we aim at minimizing both the O2E delay of the channels and the deployment cost. Equivalently, we formulate the MCSDC problem as optimizing the overlay topology to minimize the deployment cost given certain maximum O2E delay constraints of the channels.

We present a realistic model capturing major costs and delay components, and show the NP-hardness of this problem. We reformulate the original MCSDC problem as an LP problem by relaxing some constraints, propose an efficient and near-optimal bi-criteria approximation algorithm termed COCOS based on LP solution, and prove its worst-case approximation ratio. We have conducted extensive trace-driven experiments under real-world settings to evaluate COCOS based on real-world video service data. Our results demonstrate that COCOS achieves much lower deployment cost while tightly meeting the delay constraints, outperforming other traditional and state-of-art schemes by a wide margin (cutting the cost in general by more than 50%).

References

[1] Cisco, "Cisco visual networking index: forecast and trends, 2017–2022," accessed date February 10, 2022,. [Online]. Available: https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf

[2] E. Veloso, V. Almeida, W. M. Jr., A. Bestavros, and S. Jin, "A hierarchical characterization of a live streaming media workload," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 133–146, Feb. 2006.

[3] C. Zhang and J. Liu, "On crowdsourced interactive live streaming: A Twitch.tv-based measurement study," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2015, pp. 55–60.

[4] A. W. Services, "Amazon web services," accessed date February 10, 2022,. [Online]. Available: http://aws.amazon.com

[5] M. V. Marathe, R. Ravi, R. Sundaram, S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, "Bicriteria network design problems," *Journal of algorithms*, vol. 28, no. 1, pp. 142–171, 1998.

[6] Y. Zhao, H. Jiang, K. Zhou, Z. Huang, and P. Huang, "Meeting service level agreement cost-effectively for video-on-demand applications in the cloud," in *INFOCOM, 2014 Proceedings IEEE*, Apr. 2014, pp. 298–306.

[7] X. Jin and S.-H. G. Chan, "Unstructured peer-to-peer network architectures," in *Handbook of Peer-to-Peer Networking*. Springer, 2010, pp. 117–142.

[8] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 566–579, Apr. 2013.

[9] S. Yun, H. Lim, and K. Chung, "The biometric signature delegation scheme to balance the load of digital signing in hybrid P2P networks," *Peer-to-Peer Networking and Applications*, pp. 1–10, 2014.

[10] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: A redesign of multi-channel P2P video systems," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2726–2730.

[11] H. Zhao, J. Wang, Q. Wang, and F. Liu, "Queue-based and learning-based dynamic resources allocation for virtual streaming media server cluster of multi-version VoD system," *Multimedia Tools and Applications*, vol. 78, pp. 21 827–21 852, Apr. 2019.

[12] J. Niño-Mora, "Resource allocation and routing in parallel multi-server queues with abandonments for cloud profit maximization," *Computers and Operations Research*, vol. 103, pp. 221–236, 2019.

[13] C. Valliyammai and R. Mythreyi, "A dynamic resource allocation strategy to minimize the operational cost in cloud," in *Emerging Technologies in Data Mining and Information Security*, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds. Springer Singapore, 2019, pp. 309–317.

[14] Z. Chang and S.-H. G. Chan, "An approximation algorithm to maximize user capacity for an auto-scaling VoD system," *IEEE Transactions on Multimedia*, vol. 23, pp. 3714–3725, Oct. 2021.

[15] R.-X. Zhang, M. Ma, T. Huang, H. Pang, X. Yao, C. Wu, J. Liu, and L. Sun, "Livesmart: A QoS-guaranteed cost-minimum framework of viewer scheduling for crowdsourced live streaming," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 420428.

[16] R.-X. Zhang, M. Ma, T. Huang, H. Li, J. Liu, and L. Sun, "Leveraging QoE heterogenity for large-scale livecaset scheduling," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 3678–3686.

[17] R.-X. Zhang, T. Huang, M. Ma, H. Pang, X. Yao, C. Wu, and L. Sun, "Enhancing the crowdsourced live streaming: A deep reinforcement learning approach," in *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 55–60.

[18] F. Haouari, E. Baccour, A. Erbad, A. Mohamed, and M. Guizani, "QoE-aware resource allocation for crowdsourced live streaming: A machine learning approach," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[19] T. Fernando and C. Keppetiyagama, "ISP friendly peer selection in bittorrent," in *Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on*. IEEE, 2013, pp. 160–167.

[20] N. Magharei, R. Rejaie, I. Rimac, V. Hilt, and M. Hofmann, "ISP-friendly live P2P streaming," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 1, pp. 244–256, 2014.

[21] S. Hu, M. Xu, H. Zhang, C. Xiao, and C. Gui, "Affective content-aware adaptation scheme on QoE optimization of adaptive streaming over HTTP," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 15, no. 3s, pp. 1–18, Dec. 2019.

[22] J. Liu and G. Simon, "Fast near-optimal algorithm for delivering multiple live video channels in CDNs," in *22nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2013, pp. 1–7.

[23] X. Tan and S. Datta, "Building multicast trees for multimedia streaming in heterogeneous P2P networks," in *Systems communications, 2005. Proceedings*. IEEE, 2005, pp. 141–146.

[24] C. Ding, Y. Chen, T. Xu, and X. Fu, "CloudGPS: a scalable and ISP-friendly server selection scheme in cloud computing environments," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*. IEEE Press, 2012, p. 5.

[25] I. Irondi, Q. Wang, C. Grecos, J. M. A. Calero, and P. Casaseca-De-La-Higuera, "Efficient QoE-Aware scheme for video quality switching operations in dynamic adaptive streaming," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 15, no. 1, pp. 1–23, Feb. 2019.

[26] N. Magharei and R. Rejaie, "PRIME: peer-to-peer receiver-driven mesh-based streaming," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1052–1065, Aug. 2009.

[27] D. Ren, Y.-T. H. Li, and S.-H. G. Chan, "Fast-mesh: A low-delay high-bandwidth mesh for peer-to-peer live streaming," *IEEE Transactions on Multimedia*, vol. 11, no. 8, pp. 1446–1456, Dec. 2009.

[28] Z. Lu, X. Gao, S. Huang, and Y. Huang, "Scalable and reliable live streaming service through coordinating CDN and P2P," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*. IEEE, 2011, pp. 581–588.

[29] H. K. Yarnagula, P. Juluri, S. K. Mehr, V. Tamarapalli, and D. Medhi, "QoE for mobile clients with segment-aware rate adaptation algorithm (SARA) for DASH video streaming," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 15, no. 2, pp. 1–23, Jun. 2019.

[30] J. Dai, Z. Chang, and S.-H. G. Chan, "Delay optimization for multi-source multi-channel overlay live streaming," in *Proceedings of IEEE ICC 2015 - Communications Software, Services and Multimedia Applications Symposium (ICC'15)*, London, United Kingdom, 2015, pp. 6959–6964.

[31] X. Liao, H. Jin, Y. Liu, and L. M. Ni, "Scalable live streaming service based on interoverlay optimization," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 12, pp. 1663–1674, 2007.

[32] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "AnySee: Peer-to-peer live streaming," in *Proc. IEEE Infocom*, 2006, pp. 1–10.

[33] K. Pires and G. Simon, "DASH in Twitch: Adaptive bitrate streaming in live game streaming platforms," in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. ACM, 2014, pp. 13–18.

[34] A. Bentaleb, P. K. Yadav, W. T. Ooi, and R. Zimmermann, "DQ-DASH: A queuing theory approach to distributed adaptive video streaming," *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 16, no. 1, pp. 1–14, Mar. 2020.

[35] D. Kondo, Y. Hirota, A. Fujimoto, H. Tode, and K. Murakami, "P2P live streaming system for multi-view video with fast switching," in *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, Sep. 2014, pp. 1–7.

[36] R. Jannapureddy, Q.-T. Vien, P. Shah, and R. Trestian, "An auto-scaling framework for analyzing big data in the cloud environment," *Applied Sciences*, vol. 9, no. 7, p. 1417, 2019.

[37] F. Zhou, L. Jiayi, G. Simon, and R. Boutaba, "Joint optimization for the delivery of multiple video channels in telco-CDN," in *CNSM 2013: International Conference on Network and Service Management*, 2013, pp. 161–165.

[38] Z. Zhuang and C. Guo, "Optimizing CDN infrastructure for live streaming with constrained server chaining," in *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on*. IEEE, 2011, pp. 183–188.

[39] C. Wu, B. Li, and S. Zhao, "Multi-channel live P2P streaming: Refocusing on servers," in *IEEE INFOCOM*. Phoenix, Arizona: IEEE, Apr. 2008, pp. 1355–1363.

[40] F. Zhou, S. Ahmad, E. Buyukkaya, R. Hamzaoui, and G. Simon, "Minimizing server throughput for low-delay live streaming in content delivery networks," in *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video*. ACM, 2012, pp. 65–70.

[41] C. Hu, M. Chen, C. Xing, and B. Xu, "EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture," *Peer-to-Peer Networking and Applications*, vol. 5, no. 4, pp. 312–322, 2012.

[42] F. Lombardi, A. Muti, L. Aniello, R. Baldoni, S. Bonomi, and L. Querzoni, "PASCAL: An architecture for proactive auto-scaling of distributed services," *Future Generation Computer Systems*, vol. 98, pp. 342–361, 2019.

[43] S. Budhkar and V. Tamarapalli, "An overlay management strategy to improve QoS in CDN-P2P live streaming systems," *Peer-to-Peer Networking and Applications*, pp. 1–17, 2019.

[44] H. Azarpira and S. Yousefi, "On optimal topology in hierarchical P2P live video streaming networks," in *6th International Symposium on Telecommunications (IST)*. IEEE, 2012, pp. 644–649.

[45] M. Kucharzak, K. Walkowiak, and M. Klinkowski, "On modeling of minimum cost multicast topology with multiple static streams in overlay communication networks," in *Transparent Optical Networks (ICTON), 2013 15th International Conference on*. IEEE, 2013, pp. 1–4.

[46] F. Zhang, X. Tang, X. Li, S. U. Khan, and Z. Li, "Quantifying cloud elasticity with container-based autoscaling," *Future Generation Computer Systems*, vol. 98, pp. 672–681, 2019.

[47] X. Jin, K.-L. Cheng, and S.-H. G. Chan, "Island multicast: Combining IP multicast with overlay data distribution," *IEEE Transactions on Multimedia*, vol. 11, no. 5, pp. 1024–1036, Aug. 2009.

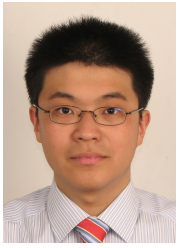
[48] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *18th USENIX Symposium on*

Networked Systems Design and Implementation (NSDI 21), 2021, pp. 201–216.

[49] J. Naor and B. Schieber, "Improved approximations for shallow-light spanning trees," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997, pp. 536–541.

[50] H. N. Gabow and K. Manu, "Packing algorithms for arborescences (and spanning trees) in capacitated graphs," *Mathematical Programming*, vol. 82, no. 1, pp. 83–109, 1998.

[51] D. Kraft *et al.*, *A software package for sequential quadratic programming*. DFVLR Obersfaffeuhofen, Germany, 1988.



Zhangyu Chang received the B.Sc. degree (Hons.) in physics and computer science (double major) and the M.Phil. degree in computer science and engineering, in 2011 and 2015, respectively, from The Hong Kong University of Science and Technology, Hong Kong, where he is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include multimedia networking, fog/edge computing, and video/location data analytics.

S.-H. Gary Chan (S'89-M'98-SM'03) received the B.S.E. degree (highest honor) in Electrical Engineering from Princeton University (Princeton, NJ) in 1993, with certificates in Applied and Computational Mathematics, Engineering Physics, and Engineering and Management Systems. He obtained the MSE and PhD degrees in Electrical Engineering from Stanford University (Stanford, CA) in 1994 and 1999, respectively, with a Minor in Business Administration. He is currently Professor in the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology (HKUST), Hong Kong. He is also Affiliate Professor in Innovation, Policy and Entrepreneurship Thrust of HKUST(GZ), Chair of the Committee on Entrepreneurship Education Program at HKUST, and Board Director of Hong Kong Logistics and Supply Chain MultiTech R&D Center (LSCM). His research interest includes smart sensing and IoT, cloud and fog/edge computing, indoor positioning and mobile computing, video/location/user/data analytics, and IT entrepreneurship.

Professor Chan has been an Associate Editor of IEEE Transactions on Multimedia (2006-11), and a Vice-Chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee (2006-13). He has been Guest Editor of Elsevier Computer Networks (2017), ACM Transactions on Multimedia Computing, Communications and Applications (2016), IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (IEEE CCNC) 2010, Multimedia symposium of IEEE Globecom (2007 and 2006), IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005).

Professor Chan has co-founded and transferred his research results to several startups. Due to their innovations and commercial impacts, his startups and research projects have received local and international awards (2012-2020). Notably, he received Hong Kong Chief Executive's Commendation for Community Service for "outstanding contribution to the fight against COVID-19" in 2020. He is the recipient of Google Mobile 2014 Award (2010 and 2011) and Silver Award of Boeing Research and Technology (2009). He was a visiting professor and researcher in Microsoft Research (2000-11), Princeton University (2009), Stanford University (2008-09), and University of California at Davis (1998-1999). He was Director of Entrepreneurship Center (2016-20), Undergraduate Programs Coordinator in Department of Computer Science and Engineering (2013-15), Director of Sino Software Research Institute (2012-15), Co-director of Risk Management and Business Intelligence program (2011-2013), and Director of Computer Engineering Program (2006-2008) at HKUST. He was a William and Leila Fellow at Stanford University (1993-94), and the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence at Princeton (1993). He is a member of honor societies Tau Beta Pi, Sigma Xi and Phi Beta Kappa, and a Chartered Fellow of The Chartered Institute of Logistics and Transport (FCILT).