Cost-Effective Low-Delay Design for Multiparty Cloud Video Conferencing

Mohammad H. Hajiesmaili, Lok To Mak, Zhi Wang, *Member, IEEE*, Chuan Wu, *Senior Member, IEEE*, Minghua Chen, *Senior Member, IEEE*, and Ahmad Khonsari

***Abstract*—Multiparty cloud video conferencing architecture has been recently advocated to exploit rich computing and bandwidth resources in the cloud to effectively improve video conferencing performance. As a typical design in this architecture, multiple *agents*, i.e., virtual machines, are deployed in different cloud sites, and users are assigned to the agents. Then, the users communicate through the agents, and the agents might transcode the recorded videos given the heterogeneities among devices in terms of hardware specification and connectivity. In this architecture, two critical and nontrivial challenges are: 1) assigning users to agents to reduce the operational cost and the user-to-user conferencing delay and 2) identifying best agents to perform transcoding tasks, taking into account the heterogeneous bandwidth and processing availabilities. To address these challenges, we cast a joint problem of user-to-agent assignment and transcoding-agent selection. The ultimate objective is to simultaneously minimize the cost of the service provider and the conferencing delay. The problem is combinatorial in nature, which belongs to the NP-hard node assignment problems. We leverage the Markov approximation framework and devise an adaptive parallel algorithm that finds a close-to-optimal solution to our problem with a bounded performance guarantee. To evaluate the performance of our solution, we implement a prototype video conferencing system and carry out trace-driven experiments. In a set of large-scale experiments using PlanetLab traces, our solution decreases the operational cost by 77% and simultaneously yields lower conferencing delay compared with an existing alternative.**

***Index Terms*—Cloud computing, network combinatorial optimization, parallel algorithm, video conferencing.**

I. Introduction

NOWADAYS the usage of video conferencing as a multi-party applications has skyrocketed with 51.7% annual growth [2], which is mainly due to the popularity of front-facing cameras on laptops, tablets, and smart phones, and recent advances in cellular networks. Moreover, it is estimated that cloud applications generate over 90% of mobile data traffic by 2018 [3], especially in multimedia networking applications [4]. Hence, a promising trend is to leverage cloud computing services for multi-party video conferencing systems. The exploitation of rich on-demand resources spanning multiple geographic regions of a distributed cloud boosts the conferencing experience and overcomes the constraints of resource-limited user devices (e.g., [5]–[7], and see [8] for commercial examples).

A. Cloud Video Conferencing: Architecture and Benefits

As illustrated in Fig. 1(c), in cloud conferencing architecture, multiple *agents*, i.e., virtual machines, are deployed in different geographical locations, and *users* join a conferencing *session* by subscribing to the cloud agents. Then, the users exchange the streams indirectly via the cloud agents. In addition, the potential high processing tasks such as transcoding are performed by the proper agents. In this way, the lightweight operations performed on user devices consume less resources of the mainly battery-limited devices. On the other hand, high demand tasks are shifted to the resource-rich cloud agents, boosting the conferencing experience. This cloud architecture has more potentials compared to the traditional client/server (C/S) [Fig. 1(a)] and P2P [Fig. 1(b)] conferencing architectures [9], [10], for two main reasons.

1) *Better at meeting stringent delay requirements:* The tolerable conferencing delay is around 400 ms [11]. There are extensive studies on reducing the delay of different video applications [5], [12], [13]. In C/S architecture, depending on the distance between the servers and the clients, the end-to-end conferencing delay might be large. In P2P architecture there is no server in the middle, hence users

Manuscript received August 30, 2016; revised January 16, 2017 and March 27, 2017; accepted May 18, 2017. Date of publication June 1, 2017; date of current version November 15, 2017. This work was supported in part by National Basic Research Program of China under Project 2013CB336700 and in part by the University Grants Committee of the Hong Kong Special Administrative Region, China, under Area of Excellence Grant Project AoE/E-02/08 and Collaborative Research Fund C7036-15G, in part by the National Natural Science Foundation of China under Grant 61402247, and in part by Hong Kong RGC under Grant 718513, Grant 17204715, and Grant 17225516. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Honggang Wang. The paper was presented in part at the IEEE 36th International Conference on Distributed Computing Systems, Ohara Fujimino, Japan, June 2016 [1]. *(Corresponding author: Ahmad Khonsari.)*

M. H. Hajiesmaili was with The Chinese University of Hong Kong, Shatin, Hong Kong. He is now with the Whiting School of Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: hajiesmaili@jhu.edu).

L. T. Mak and M. Chen are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: mlt014@ie.cuhk.edu.hk; minghua@ie.cuhk.edu.hk).

Z. Wang is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China (e-mail: wangzhi@sz.tsinghua.edu.cn).

C. Wu is with the Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong (e-mail: cwu@cs.hku.edu.hk).

A. Khonsari is with the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1417466191, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran 193955746, Iran (e-mail: ak@ipm.ir).

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

Digital Object Identifier 10.1109/TMM.2017.2710799

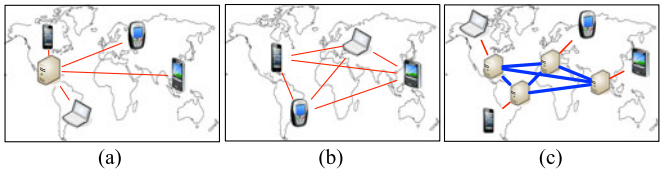


Fig. 1. Different video conferencing architectures. (a) Client/Server. (b) Peer-to-Peer. (c) Cloud.

communicate directly together that reduces the delay [9], [10]. Measurements in [6] have shown that the delay in a cloud architecture is comparable or even lower than the delay in P2P architecture.

2) *Providing more bandwidth and processing capacity at lower costs:* User devices in video conferencing are mainly smart phones that are highly heterogeneous in terms of screen resolution, hardware, and operating system [14]. This heterogeneity, along with different network connectivity demands for on-the-fly *transcoding* to convert the streams from one format/bitrate to another [15], [16]. The dedicated servers in C/S architecture can be used to perform such high complexity transcoding tasks. However, this architecture is not scalable and on-demand. On the other hand, the resource-limited devices in P2P architecture cannot be exploited to execute such computation-intensive tasks. In contrast, the cloud agents can effectively perform high demand tasks and this architecture provides scalability by using on-demand cloud agents, at a lower cost.

B. Critical Problem and Key Design Challenges

Nevertheless, there are two critical challenges to minimize the service provider's cost and the conferencing delay.

1) How to select a proper agent for each user is an important design issue that has substantial effects on both user-to-user delay and the costs to the service provider. The existing solutions follow nearest assignment policy, i.e., each user is assigned to the nearest nearby agent [6], [7]. This policy is not optimal in terms of user-to-user conferencing delay and inter-agent traffic cost.[1] The reason is that the nearest policy is oblivious to whereabouts of the other users in a conferencing session and diversity of transcoding latency in heterogeneous agents. For instance, in Fig. 2, the SG agent must be chosen for user 4 if we follow the nearest policy. However, assigning user 4 to the TO agent (the second nearest one) is better as (i) the inter-user delay is lower because the TO agent is closer to other agents, e.g., the delay of user 4 to user 1 via TO is at least $27 + 67 = 94$, whereas this value via the SG is at least $20 + 117 = 137$; (ii) as user 3 is already assigned to the TO agent, it is beneficial to assign user 4 to the TO agent rather than the SG agent to avoid exchange of the streams. This assignment decreases traffic cost. Note that reducing the cloud cost, i.e., traffic and processing costs, has been studied for other types of multimedia streaming applications [17], however, the multi-party nature of video conferencing applications makes the problem more challenging.

2) Proposing an approach that attempts to find a proper agent to transcode the streams has not been studied. The agents are heterogeneous in terms of resource availability; hence, they might have different transcoding latencies. The agent that must perform transcoding task should be chosen, to minimize the cost and latencies. For example in Fig. 2, we previously demonstrated that choosing the TO agent for user 4 is better since it reduces the delay and operational cost. However, since the SG agent has more computational power than the TO agent, it is better in terms of the latency of the potential transcoding tasks.

[1]According to our rough estimate, based on real data from Amazon EC2, the monthly cost of inter-region stream exchange may be in the order of multi-million dollars for a cloud conferencing service provider in the scale of Skype.

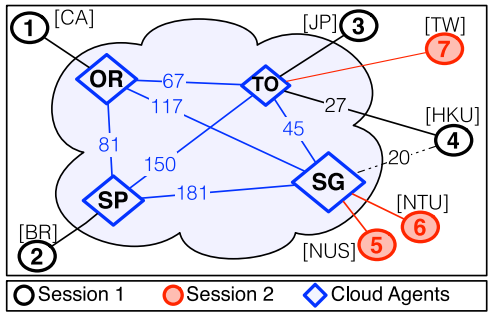


Fig. 2. Video conferencing scenario with seven users (PlanetLab nodes) in two sessions and four cloud agents (Amazon EC2 instances). The numbers on edges are real-world measured latency values. Agents in larger diamonds have higher processing capabilities. SG: Singapore, TO: Tokyo, OR: Oregon, SP: Sao Paulo, HK: Hong Kong, BR: Brazil, CA: Canada, JP: Japan.

C. Summary of Contributions

Almost all of the existing studies that we are aware, neglect to consider the cost to the service provider and simply adopt the nearest policy for user-to-agent assignment (e.g., *Airlift* [6] and *vSkyConf* [7]). To the best of our knowledge, this study is the first that aims to improve the cloud video conferencing design by tackling the user-to-agent and transcoding assignments problem in a unified combinatorial optimization framework. The main contributions of this study are summarized below.

1) We formulate the **User-to-agent Assignment Problem** (**UAP**), that tries to select the user-to-agent assignment and transcoding assignment in a multi-party application with the objective of simultaneously minimizing the operational cost to the service provider and conferencing delay. The problem is subject to the capacity constraints of the diverse agents and the user-to-user conferencing delay constraints. The problem is a nonlinear combinatorial optimization problem in the category of NP-hard node assignment problems [18] which are difficult to solve due to persistent dynamics in the system and large problem size.

2) We leverage the Markov approximation framework [19] which is a technique for solving the combinatorial network problems in a distributed fashion. We devise an efficient parallel and iterative algorithm to solve the **UAP**, which runs locally in a representative agent of each session and converges to a close-to optimal assignment. The algorithm adapts to the system dynamics, provides a bounded approximation gap, and is robust against the inaccurate measurements of the problem data. In addition, we improve the convergence of the algorithm by proposing another initialization algorithm called *AgRank*, which is a simple scheme with low complexity.
3) We implement a cloud video conferencing system prototype using Amazon EC2 [20] platform and also carry out trace-driven evaluation experiments using PlanetLab [21] nodes. The results demonstrate the significant improvement of our solution compared to the existing alternatives. In a representative experimental setting of PlanetLab traces, our algorithm outperforms the nearest assignment policy [6], [7] by reducing the operational cost and the delay by 77% and 2%, respectively.

The rest of this paper is organized as follows. We review the related work in Section II. In Section III, we introduce the cloud video conferencing system model. The problem is formulated in Section IV. Then, in Section V, our solution design is proposed. In Section VI, we propose a fast session initialization algorithm. We discuss the implementation and the complexity issues of our solution design in Section VII. The results of the prototype system implementation and trace-driven experiments are demonstrated in Section VIII. Finally, the paper is concluded in Section IX.

II. Related Work

Multiparty video conferencing: Previously, P2P architecture [9], [10] was considered as an alternative to traditional client/server architecture. In [10], following network utility maximization framework, a problem of video conferencing in P2P architecture has been studied. In another recent work [9], the authors propose a scheme to maximize video quality under uplink-downlink capacity constraints for peer-to-peer multiparty video conferencing. However, in P2P there is no powerful server in the architecture which hinders executing the high demand tasks such as transcoding. The cloud architecture for video conferencing has been proposed in *Airlift* [6] for the first time, and it suggests to use cloud bandwidth resources to boost the conferencing experience. In *vSkyConf* [7], the computational resources of the cloud is exploited for executing processing tasks, in addition to the dedicated cloud communication infrastructure. These studies assume nearest assignment policy, which is not optimal in a multi-party application in terms of intra-cloud traffic and user-to-user conferencing delay. Two recent studies [5], [22] propose different server selection/placement and topology control approaches to *only* minimize the latency in *transcoding-free* video conferencing, without taking into account the operational cost. Finally, the delay-constrained video streaming in different network infrastructures and applications has been studied previously, e.g., in wireless and wireline networks [12], [13], [23]–[27]. For example, the authors in [24] propose a wireless-aware cloud server scheduling policy for mobile gaming applications. In contrast, this work focuses on a cloud video conferencing scenario, which has a different set of challenges mainly because of its multi-party and interactive nature. Finally in [28], a stochastic approach for cloud server scheduling with the goal of load-balancing has been proposed. This approach is for the general applications and cannot be leveraged to multi-party video conferencing settings mainly due to the delay consideration of such applications.

Virtual network embedding: In *network virtualization* [29] multiple virtual networks cohabit the same substrate network. In cloud video conferencing architecture, one can imagine the underlying cloud agents as the substrate networks and the conferencing sessions as the virtual networks on top of them. In [16], a competitive online algorithm has been proposed for general resources, i.e., bandwidth and processing resources, for delay-sensitive multimedia applications. The solution follows virtual network embedding approach and could be customized for video conferencing scenarios in cloud architecture. Unlike [16], further study of the **UAP** reveals that this is a challenging combinatorial problem that makes the approach in [16] inadequate. The idea of migrating the current configuration has been widely used in VN embedding problems to improve the acceptance rate of VNs [30], energy saving [31], and house cleaning [32]. These goals could be imagined as additional motivations for proper assignment in the **UAP**. Finally, we note that in another recent paper [33], the cost minimization problem of cloud clone migration using a Markov Decision Problem has been studied. While the problem studied in [33] is largely different from ours, the approach in [33] towards providing a trade-off between the migration cost and the content transmission cost could be considered as a potential method to extend our work to the more general setting.

III. Video Conferencing Model

A video conferencing system in cloud architecture is assumed with multiple sessions. Each session is established among multiple users. Within a session, each user transmits its video in a specific *representation* which represents a fixed format/bitrate. Each user is assigned to a cloud agent and inter-user transfer is done indirectly through the agents. In addition, each user demands specific representations from other parties of the session. Within each particular pair of the users in the session, the upstream representation of the sender might vary from the downstream one demanded by the receiver. In this situation, real-time converting of the representations, i.e., *transcoding*, is required which is carried out by the agents. The key notations of the system model are listed in Table I.

A. Session and User

Denote $\mathcal{S}$ and $\mathcal{U}$ as the set of conferencing sessions and users, respectively. Let $\mathcal{U}(s) \subseteq \mathcal{U}$ be the users of session s and $s(u) \in \mathcal{S}$ be the session that user u belongs to. Let

TABLE I
KEY NOTATIONS

Notation		Definition
Users	$\mathcal{S}$	Set of video conferencing sessions, $S \triangleq \lvert\mathcal{S}\rvert$
	$\mathcal{U}$	Set of users, $U \triangleq \lvert\mathcal{U}\rvert$
	$\mathcal{U}(s)$	Users of session s
	$s(u)$	Session of user u
	$\mathcal{P}(u)$	Set of other parties in user u's session
Representation	$\mathcal{R}$	Set of video representations, $R \triangleq \lvert\mathcal{R}\rvert$
	$\kappa(r)$	Bitrate of representation r
	$r_u^{\mathbf{u}}$	Upstream representation of user u
	$r_{uv}^{\mathbf{d}}$	Downstream repr. of user u from user v
	θ	$U \times U$ transcoding matrix; $\theta_{uv} = 1$, if $s(v) = s(u)$ and $r_u^{\mathbf{u}} \neq r_{vu}^{\mathbf{d}}$; 0, otherwise
Agents	$\mathcal{L}$	Set of cloud agents, $L \triangleq \lvert\mathcal{L}\rvert$
	u_l	Upload capacity of agent l
	d_l	Download capacity of agent l
	t_l	Transcoding capacity of agent l
	$\sigma_l(r_1, r_2)$	Transcoding latency of agent l from representation r_1 to representation r_2
	D	$L \times L$ inter-agent delay matrix
	H	$L \times U$ agent-to-user delay matrix
Opt. Vars.	λ_{lu}	User assignment variable; 1 if user u is assigned to agent l; 0, otherwise
	γ_{lruv}	Transcoding task assignment variable; 1 if $r_{vu}^{\mathbf{d}} = r$ and the transcoding is done at agent l; 0, otherwise

$\mathcal{P}(u) \subseteq \mathcal{U}$ be the set of other parties in user u's session, (i.e., $\mathcal{P}(u) = \{v | v \in \mathcal{U}, s(v) = s(u), v \neq u\}$).

B. Representation

By representation, we mean a particular format, bitrate, and resolution of a video, e.g., for YouTube videos, the standard representations are (360 p, 1 Mbps), (480 p, 2.5 Mbps), (720 p, 5 Mbps), (1080 p, 8 Mbps), etc. Denote $\mathcal{R}$ as the set of possible representations in cloud conferencing system. The *upstream* representation of user u, denoted by $r_u^{\mathbf{u}} \in \mathcal{R}$, is an input to the problem and takes into account its connection and device hardware specification. Moreover, the *downstream* representation, $r_{uv}^{\mathbf{d}} \in \mathcal{R}$ represents the required representation from another user v in the session. The parameter $r_{uv}^{\mathbf{d}} \in \mathcal{R}$ is the input to the problem, and it can be set so as to further limit the transcoding of each stream to a single target representation for all of the other users, e.g., to transcode each stream to the lowest common resolution to simplify the design.

By $\kappa(r)$, we represent the bitrate of representation r. Moreover, let $\theta = [\theta_{uv}]_{U \times U}$ be the *transcoding matrix*, where $\theta_{uv} = 1$ given that u and v belong to a same session and produce/demand different representations, i.e., $s(v) = s(u)$ and $r_u^{\mathbf{u}} \neq r_{vu}^{\mathbf{d}}$, and $\theta_{uv} = 0$ otherwise. Note that θ is considered in our model in a general form. It could be customized to be restricted to practical constraints, e.g., to support only high-to-low quality transcoding operations we can change the definition of $\theta_{uv} = 1$ if $s(v) = s(u)$ and $r_{vu}^{\mathbf{d}} < r_u^{\mathbf{u}}$. Finally, we assume these representations are computed by another design scheme, e.g., the approach in [10]. Hence, in this study, the representations are given as the input to the problem. However, because of the dynamic network conditions, representations are subject to change. Our solution design can effectively adapt to these dynamics, as explained in Section V.

C. Cloud Agent

Agents, in set $\mathcal{L}$, are typically virtual machines that a conferencing service provider, like Skype, leases from a cloud service provider, like Amazon. There can be multiple agents with diverse computation and communication resources each of which in a different data center. The agents are heterogeneous resources with the following properties for agent l: (i) u_l represents the upload capacity of l (in Mbps); (ii) d_l is the download capacity of l (in Mbps); (iii) t_l is the transcoding capacity of l (the number of concurrent transcoding tasks). We assume that each agent is a virtual machine with a fixed amount of processing resources for the transcoding tasks, i.e., one unit of its transcoding capacity, such that its number of concurrent transcoding tasks can be derived; and (iv) $\sigma_l(.)$ represents the transcoding latency of l (in ms). The transcoding latency $\sigma_l(r_1, r_2)$ is an increasing function of the bitrates of the input (r_1) and output (r_2) representations, given that transcoding is typically done by decoding the source stream to an intermediate format, and then re-encoding the stream from the intermediate format to the destination bitrate [7]. The more computing capacity an agent has, the more concurrent transcoding tasks it can perform, and the faster each of the tasks can be completed.

Let $D = [D_{lk}]_{L \times L}$ be the *inter-agent delay matrix* and $H = [H_{lu}]_{L \times U}$ be the *agent-to-user delay matrix*, where D_{lk} is the delay between agents l and k and H_{lu} is the delay between agent l and user u. We have $D_{ll} = 0$ and $D_{lk} = D_{kl}, \forall l, k \in \mathcal{L}$. Considering the dedicated cloud infrastructure, it is reasonable to assume that any pairs of the agents are connected together. Similar to the representations, in practice, the inter-agent and agent-to-user delay and transcoding latency values are subject to change, and thus a proper design should adapt to dynamics (Section V-C).

D. Illustrative Example

Fig. 3 illustrates a conferencing session with four users. There are three cloud agents and possible user-to-agent assignments are depicted in Fig. 4. In Fig. 3, when the upstream representation of a user ($r_1^{\mathbf{u}} = 2$, say) differs from the downstream representation requested by another user ($r_{21}^{\mathbf{d}} = 1$, say), a flow is marked using dotted lines with a transcoding task in the middle (e.g., the flow from user 1 to user 2). On the other hand, in flows where the upstream and downstream representations are the same solid lines are used (e.g., flows from user 2 to user 1). Fig. 4 plots a potential user-to-agent assignment by highlighting the assigned links with thick lines, i.e., users 1, 2, and 3 are assigned to agent 1 and user 4 is assigned to agent 2. Fixing this user-to-agent assignment, agents 1 and 2 are the best candidates for hosting the transcoding tasks. However, the result of comparing the transcoding capacity and latency values of these agents (eight concurrent tasks as the transcoding capacity and the transcoding latency 50 ms for agent 2, and transcoding

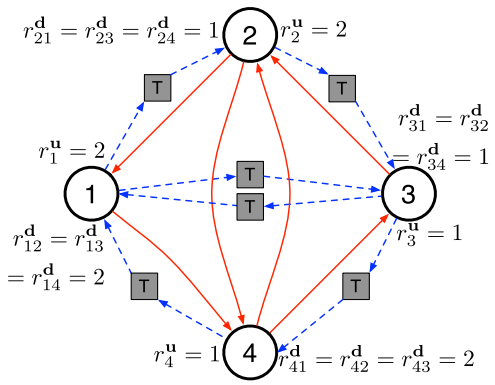


Fig. 3. Video conferencing scenario: $S = 1, U = 4, L = 3, \mathcal{R} = \{1, 2\}$. Users are labeled by their upstearm and downstream representations. Squares denote transcoding operations that are required due to different representations requested by the source and the destination users in some flows.

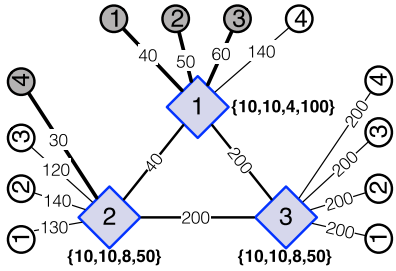


Fig. 4. Illustration of user-to-agent and inter-agent latency values of conferencing scenario of Fig. 3. Agents in diamonds are labeled with quadruple $\{u_l, d_l, t_l, \sigma_l\}$. The labels in edges are either user-to-agent or inter-agent latency values. A possible user-to-agent assignment is highlighted in gray.

capacity 4 and latency 100 ms for agent 1, respectively) suggests that agent 2 is the best host for the transcoding tasks.

IV. User-to-Agent Assignment Problem

A. Optimization Variables

Let λ_{lu} be the *user assignment* variable such that $\lambda_{lu} = 1$ if user u is assigned to agent l; and $\lambda_{lu} = 0$ otherwise. Each user must be assigned to one agent. Hence we have

$$\sum_{l \in \mathcal{L}} \lambda_{lu} = 1, \qquad \forall u \in \mathcal{U} \tag{1}$$

$$\lambda_{lu} \in \{0, 1\}, \qquad \forall l \in \mathcal{L}, \forall u \in \mathcal{U}. \tag{2}$$

The second decision variable indicates at which agents the transcoding tasks must be performed. Each transcoding task could be performed at the *source agent*, the *destination agent*, or a *tertiary agent*. Denote γ_{lruv} as the *transcoding task assignment* variable where $\gamma_{\text{lruv}} = 1$ if user v requires representation r from user u (i.e., $r^{\mathbf{d}}_{vu} = r$) and the transcoding is done at agent l; and $\gamma_{\text{lruv}} = 0$ otherwise. γ_{lruv} satisfies the following constraints:

$$\sum_{l \in \mathcal{L}} \sum_{r \in \mathcal{R}} \gamma_{\text{lruv}} = \theta_{uv}, \forall u \in \mathcal{U}, \forall v \in \mathcal{P}(u) \tag{3}$$

$$\gamma_{\text{lruv}} \in \{0, 1\}, \forall l \in \mathcal{L}, \forall r \in \mathcal{R}, \forall u \in \mathcal{U}, \forall v \in \mathcal{P}(u). \tag{4}$$

Constraint (3) states that the transcoding of the flow from u to v is needed only when $\theta_{uv} = 1$, and exactly one agent should carry out the transcoding to the required representation.

B. Capacity Constraints of Cloud Agents

To formulate the capacity constraint, first we introduce some notations for the ease of explanation. Let $\nu_{lru} \triangleq \max_{v \in \mathcal{P}(u)} \gamma_{\text{lruv}}$ denote whether agent l transcodes u's stream to representation r for at least one other participant in u's session (1 yes and 0 no), and $\nu'_{lu} \triangleq \max_{r \in \mathcal{R}} \nu_{lru}$ denote whether agent l transcodes u's stream at all (1 yes and 0 no). The download capacity constraint of agent l is formulated as

$$\sum_{u \in \mathcal{U}} \left(\lambda_{lu} \kappa(r^{\mathbf{u}}_u) + \sum_{k \in \mathcal{L}, k \neq l} \mu_{klu} \right) \leq d_l, \forall l \in \mathcal{L} \tag{5}$$

where the first term is due to the last-mile traffic of users that are assigned to agent l. The second term represents the outgoing traffic of user u from all of the other agents towards agent l. Denote μ_{klu} as the total download traffic at agent l that is the result of receiving the stream via another agent k and originated from user u, as follows:

$$\mu_{klu} = \lambda_{ku} \nu'_{lu} \kappa(r^{\mathbf{u}}_u) + \Big(\max_{\substack{v \in \mathcal{P}(u), \\ \theta_{uv} = 0}} \lambda_{lv} \Big) \lambda_{ku} (1 - \nu'_{lu}) \kappa(r^{\mathbf{u}}_u)$$

$$+ \sum_{\substack{r \in \mathcal{R}, \\ r \neq r^{\mathbf{u}}_u}} \Big(\max_{\substack{v \in \mathcal{P}(u), \\ r^{\mathbf{d}}_{vu} = r}} \lambda_{lv} \Big) (1 - \lambda_{lu}) \nu_{kru} \kappa(r)$$

where the first term represents the traffic from u's agent k to agent l so that u's stream can be transcoded at l; the second term depicts the traffic resulting from sending the upstream to other parties; and the last term is the traffic caused by bitrate changes after transcoding. Note that the definition of μ_{klu} captures cloud-level multicasting by considering the inter-agent stream of each user once, regardless of the number of parties on the target agent. In this way, this formulation takes the advantages of multicasting in the design. For the upload capacity we get

$$\sum_{u \in \mathcal{U}} \left(\lambda_{lu} \sum_{v \in \mathcal{P}(u)} \kappa(r^{\mathbf{d}}_{uv}) + \sum_{k \in \mathcal{L}, k \neq l} \mu_{lku} \right) \leq u_l, \forall l \in \mathcal{L}. \tag{6}$$

Now we formulate the transcoding capacity constraints of the agents. Note that regardless of the number of destinations, transcoding of user u's upstream to representation r occupies one unit of the transcoding capacity of agent l. This constraint is formulated as

$$\sum_{u \in \mathcal{U}} \sum_{r \in \mathcal{R}} \nu_{lru} \leq t_l, \quad \forall l \in \mathcal{L}. \tag{7}$$

C. End-to-End Delay Constraints of Users

The end-to-end delay of a flow from user u to user v is the summation of the following.

1) Propagation delay from u to its agent l; H_{lu}.
2) The propagation delay between u's agent and v's agent, including two cases:
   a) from u's agent l to v's agent k directly; D_{lk}; and

b) from u's agent l to a tertiary agent m (for transcoding) and then to v's agent k; $D_{lm} + D_{mk}$.

3) From v's agent k to v; H_{kv}.
4) (Possibly) the transcoding latency at an agent l; $\sigma_l(r_u^{\mathbf{u}}, r_{vu}^{\mathbf{d}})$.

We ignore any queuing delay at the agents, as our bandwidth and transcoding capacity constraints ensure the availability of resources for the respective tasks.

Using the transcoding matrix θ and defining $\bar{\theta}_{uv} = 1 - \theta_{uv}$, we get the user-to-user delay of flow $u \rightarrow v$ as

$$d_{uv} = \sum_{l\in\mathcal{L}} (\lambda_{lu} H_{lu} + \lambda_{lv} H_{lv}) + \bar{\theta}_{uv} \left(\sum_{l\in\mathcal{L}} \sum_{k\in\mathcal{L}} \lambda_{lu}\lambda_{kv} D_{lk} \right) + \theta_{uv} \left(\sum_{l\in\mathcal{L}} \sum_{k\in\mathcal{L}} \sum_{r\in\mathcal{R}} \gamma_{\text{lruv}} \left(D_{lk}(\lambda_{ku}+\lambda_{kv}) + \sigma_l(r_u^{\mathbf{u}}, r_{vu}^{\mathbf{d}}) \right) \right).$$

Let $D^{\max}$ be the maximum acceptable delay, so, the end-to-end conferencing delay constraint is

$$d_{uv} \leq D^{\max}, \quad \forall u \in \mathcal{U}, \forall v \in \mathcal{P}(u). \tag{8}$$

D. Optimization Problem

Objective function: The goal is to minimize the operational cost to the conferencing service provider, and the conferencing delay. The operational cost to the provider has two parts.

1) The bandwidth cost, which is expressed as $G(\boldsymbol{x}_s) = \sum_{l\in\mathcal{L}} g_l(x_{ls})$ for session s, where $x_{ls} = \sum_{u\in\mathcal{U}(s)} \sum_{k\in\mathcal{L},k\neq l} \mu_{klu}$ is the total incoming traffic to agent l from other agents in session s, and vector $\boldsymbol{x}_s = [x_{ls}]_{l\in\mathcal{L}}$. $g_l(.)$ is a convex and increasing function. By expressing the bandwidth cost in this way, we focus the inter-agent traffic cost. We do not consider the traffic cost between the users and the agent, since it is fixed regardless of the user-to-agent assignment.
2) Transcoding cost at the agents in session s is similarly formulated as follows:

$$H(\boldsymbol{y}_s) = \sum_{l\in\mathcal{L}} h_l(y_{ls}), \boldsymbol{y}_s = [y_{ls}]_{l\in\mathcal{L}}, y_{ls} = \sum_{u\in\mathcal{U}(s)} \sum_{r\in\mathcal{R}} \nu_{lru}$$

where y_{ls} indicates the number of transcoding tasks that agent l performs in session s and $h_l(.)$ is a convex function. In our experiments, we use linear functions for both bandwidth cost and transcoding cost functions.

The delay cost at users in session s is described by function $F(\boldsymbol{d}_s)$, where $\boldsymbol{d}_s = [d_u]_{u\in\mathcal{U}(s)}$, $d_u = \max_{v:u\in\mathcal{P}(v)} d_{vu}$ is the maximum end-to-end delay experienced by user u for receiving streams from other parties, and $F(.)$ is a convex and increasing function, e.g., $F(\boldsymbol{d}_s) = (\sum_{u\in\mathcal{U}(s)} d_u)/|\mathcal{U}(s)|$, which corresponds to the average user delay in the session.

Problem formulation: We cast the user-to-agent and transcoding task assignments problem as

$$\mathbf{UAP}: \quad \min_{\lambda_{lu},\gamma_{\text{lruv}}} \quad \sum_{s\in\mathcal{S}} (\alpha_1 F(\boldsymbol{d}_s) + \alpha_2 G(\boldsymbol{x}_s) + \alpha_3 H(\boldsymbol{y}_s))$$

$$\text{s.t.} \quad \text{Constraints (1)–(8).}$$

Remarks: The solution to the **UAP** finds the optimal user-to-agent and transcoding task assignments. The objective is to minimize the service provider's total bandwidth ($G(\boldsymbol{x}_s)$) and processing costs ($H(\boldsymbol{y}_s)$), and the conferencing delay ($F(\boldsymbol{d}_s)$). Considering delay in the objective function is intended to ensure that conferencing delay is as small as possible, although we have constrained their stringent requirement in (8). The objective function is the sum of the above costs, weighted by design parameters α_1, α_2, and α_3. The constraints of the problem are the bandwidth and processing capacity of the cloud agents (Section IV-B) and the end-to-end delay of the users (Section IV-C). Design parameters $\alpha_i \geq 0$ can be tuned towards any desired trade-off between reducing the operational cost and conferencing delay, e.g., a larger α_1 leans more towards optimizing conferencing delay, whereas larger α_2 and α_3 stress to reduce the cost to the service provider. In Section VIII, we experimentally evaluate the effect of these parameters. Finally, we remark that the user-to-agent assignment part of the **UAP** belongs to the node assignment problems which are NP-hard [18]; hence, tackling the **UAP** even in a centralized manner is difficult. We highlight that a proper solution for this problem have to be adapted to the dynamics in the system and be implemented in large-scale systems.

V. Markov Approximation-Based Parallel Algorithm

The goal is to devise a parallel and adaptive solution for the **UAP**. By parallel, we mean that each session solves its local problem separately. In this way, the solution can be implemented for the large-scale conferencing systems. The recently proposed Markov approximation approach [19] is a general framework to tackle network combinatorial problems (see e.g., [34], [35]) that allows us to design a parallel and adaptive solution. We proceed to briefly introduce the framework.

A. Markov Approximation Framework

Generally, the Markov approximation framework [19] aims to devise distributed solutions for network combinatorial optimization problems by 1) constructing a certain Markov chains with a target steady-state distribution following the structure of the problem and 2) investigating a particular structure of the Markov chain that could be implemented in distributed manner.

We first begin with a brief primer on the theoretical approximation framework. Denote $f = \{\boldsymbol{\lambda}, \boldsymbol{\gamma}\} \in \mathcal{F}$ as an instance of feasible solutions to the **UAP**, where $\mathcal{F}$ is the set of all of the feasible solutions, i.e., all of the assignments that satisfy constraints (1)–(8). Let Φ_f be the objective value of the **UAP** when the assignment is f. In addition, let p_f be the fraction of time that f is used as the solution to the **UAP**. Using these notations, we can rewrite the **UAP** as follows:

$$\mathbf{UAP-EQ}: \min_{p_f,\forall f\in\mathcal{F}} \quad \sum_{f\in\mathcal{F}} p_f \Phi_f, \quad \text{s.t.} \quad \sum_{f\in\mathcal{F}} p_f = 1.$$

We cast an approximate version, **UAP**-β, of the **UAP-EQ** using *log-sum-exp* approximation [36] as

$$\textbf{UAP}\text{-}\beta: \quad \min_{p_f, \forall f \in \mathcal{F}} \quad \sum_{f\in\mathcal{F}} p_f \Phi_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f \log p_f$$
$$\text{s.t.} \quad \sum_{f\in\mathcal{F}} p_f = 1$$

where β is a sufficiently large fixed parameter that can be exploited to control how accurate is the approximation [19]. The **UAP**-β is a convex optimization problem. The optimal solution could be achieved by solving the KKT conditions [36] as follows:

$$p_f^\star = \frac{\exp(-\beta\Phi_f)}{\sum_{f'\in\mathcal{F}}\exp(-\beta\Phi_{f'})}, \quad f \in \mathcal{F} \tag{9}$$

and the optimal objective function value is

$$\hat{\Phi} = -\frac{1}{\beta}\log\Big(\sum_{f\in\mathcal{F}}\exp(-\beta\Phi_f)\Big). \tag{10}$$

Moreover, the optimality gap between the optimal objective values of the **UAP-**β and the **UAP** is characterized by

$$\min_{f\in\mathcal{F}}\Phi_f - \frac{1}{\beta}\log|\mathcal{F}| \le \hat{\Phi} \le \min_{f\in\mathcal{F}}\Phi_f. \tag{11}$$

Note that the approximation gap vanishes as β approaches infinity, i.e., the larger β is, the more accurate the approximation model is. We further investigate the effect of β on the performance of our algorithm in Section VIII.

We use the above approximation framework to find the optimal solution to the **UAP-**β by time-sharing among its feasible solutions $f \in \mathcal{F}$ according to $p_f^\star$ in (9). The key is to construct a Markov chain that models feasible solutions as states, achieves stationary distribution $p_f^\star, \forall f$, and allows efficient parallel construction among the conferencing sessions.

B. Algorithm Design

We propose a parallel algorithm that finds a close-to-optimal assignment by simulating such a Markov chain over time. Specifically, the algorithm starts with a feasible assignment solution f of the **UAP**, and transits to another feasible solution f' according to transition rate $q_{f,f'}$. After several iterations, the algorithm converges to the Markov chain's steady-state $p_f^\star$ as defined in (9), which is the optimal solution to the **UAP-**β and a close-to-optimal solution to the **UAP**. The transition rates, however, must be carefully computed to achieve the steady-state distribution. In addition, although we have given the concrete form of $p_f^\star$ in (9), we should note that it is computed using KKT conditions in a centralized fashion, which requires complete, static information of the entire system. This incurs a further challenge of computing the transition rates in a parallel fashion (in each session respectively), to achieve the desired overall stationary distribution.

Based on the theoretical insights from [19], the sufficient conditions for constructing such a Markov chain are to ensure that in the Markov chain the following are true:

1) any two states are reachable from each other (i.e., the Markov chain is irreducible);
2) the detailed balance equations are satisfied, i.e., $p_f^\star q_{f,f'} = p_{f'}^\star q_{f',f}, \forall f, f' \in \mathcal{F}$.

The sufficiency of the above conditions is the key in providing two degrees of freedom in Markov chain design that leads to a parallel implementation of the desired Markov chain.

The *first* degree of freedom is that we are allowed to set the transition rate between any two arbitrary states to zero, i.e., remove their edge in the underlying graph, if they are still reachable from the remaining Markov chain. By doing so, the stationary distribution of the modified Markov chain distribution is still $p_f^\star$. In the implementation, direct transition between two states is equivalent to the migration of the current assignment to another feasible one. This imposes migration overhead to the system, that could be minimized by allowing direct links between two states in the Markov chain only if either one user or transcoding assignment differs between the two states. An example Markov chain is depicted in Fig. 5(b) corresponding to the scenario in Fig. 5(a). Consider feasible solution 1 in Fig. 5(a) where both users and the transcoding task are assigned to L1, and feasible solution 2 where both users are assigned to L1 but the transcoding task is assigned to L2. They are different in only one particular assignment; hence, a direct link is depicted between these two states in Fig. 5(b).

The *second* degree of freedom is that for two assignments (Markov chain states) f and f' with direct transitions, there are many options in the design of transition rates $q_{f,f'}$ and $q_{f',f}$. To facilitate parallel Markov implementation, we design the transition rate between two states as

$$\begin{aligned} q_{f,f'} &= \tau\exp\left(\frac{1}{2}\beta(\Phi_f - \Phi_{f'})\right) \\ &= \tau\exp\left(\frac{1}{2}\beta(\Phi_{s,f} - \Phi_{s,f'})\right) \end{aligned} \tag{12}$$

where $\Phi_{s,f}$ and $\Phi_{s,f'}$ are the *local objective* values of session s (i.e., $\alpha_1 F(\boldsymbol{d}_s) + \alpha_2 G(\boldsymbol{x}_s) + \alpha_3 H(\boldsymbol{y}_s)$) at solutions f and f', respectively, and τ is a positive constant. In our algorithm based on the value of τ, each session initiates a timer and when timer expires, the session (possibly) migrates to another assignment. Larger τ reduces the convergence of the algorithm, but it may impose the overhead of frequent assignment migration. In Section VIII, we explain the migration overhead of our algorithm in more details. Note the transition rate could be computed using the local values of each conferencing session, thereby the algorithm can be implemented per session locally, in a parallel manner. The rationale is that we allow only one decision variable's value to be different between f and f'. Note that by this construction of the transition rates, the detailed balance equations are respected.

Our proposed algorithm is listed as Algorithm V-B. First, we mention that the algorithm runs in each session separately, hence it is parallel. In addition, the algorithm procedure is performed in a representative agent of each session, e.g., the session initiator's agent, so, it has no overhead on the user devices. In the HOP procedure, session s migrates to another feasible assignment with a probability proportional to the local objective value of the target solution. Since our design allows migrat-

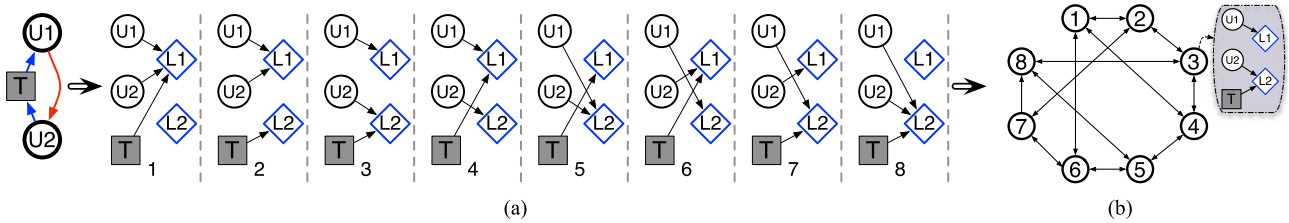


Fig. 5. Simple video conferencing scenario with one session, two users, one transcoding operation, and two agents. (a) All eight feasible assignment solutions, assuming both cloud agents are powerful enough and the end-to-end delays of both flows are always less than $D^{\max}$. (b) The Markov chain.

Algorithm 1: Markov Approximation-Based Assignment (*for Each Session* s)

```
1  procedure WAIT
2    Generate an exponentially distributed random
     number with mean 1/τ and begin countdown
     according to it
3    while the timer has not expired
4      If Receive a FREEZE message then Pause
5      If Receive a UNFREEZE message then Resume
6    end
7    Invoke HOP
8  end procedure
9  procedure HOP
10   Broadcast a FREEZE message to other sessions
11   Fetch the updated list of residual capacities of agents
12   F_s ← set of all feasible solutions with only one
     different decision
13   Migrate to solution f' ∈ F_s with probability
     proportional to exp((1/2)β(Φ_{s,f} − Φ_{s,f'}))
14   Broadcast a UNFREEZE message to other sessions
15   Invoke WAIT
16 end procedure
```

ing to the assignments with at most one change in assignments and given the residual capacity of the agent as in Line 11, session s finds $\mathcal{F}_s$ (Line 12) which is the set of all target feasible assignments with one change. We refer to Section VII-C for the complexity analysis of computing the target feasible assignments.

In the WAIT procedure, if session s receives a FREEZE message, it pauses its countdown, as another session is migrating. After receiving UNFREEZE message which means that the migration is done, it resumes the countdown, which is still exponentially distributed. This is true since exponential distribution is memoryless and because of this property an exponential count down timer is used in our algorithm design. Last but not the least, we remark that the FREEZE message is an intra-message within the cloud agents that are synchronized together in a cloud environment owned by a single cloud provider. The result in Proposition 1 shows that regardless of the initial assignment, Algorithm V-B converges to the stationary state with provable convergence time, with the proof given in [19].

Proposition 1: Algorithm V-B realizes a continuous-time Markov chain, which converges to the stationary distribution in (9).

C. Robustness to System Dynamics and Noisy Measurements

In practice, almost all inputs to the **UAP** are subject to change because of dynamics in network connectivity, duration of the conferencing sessions, and users' device conditions. As an important feature, our parallel algorithm is robust to variations due to system dynamics, e.g., arrival and departures of the sessions. In particular, upon arrival of a session, a feasible assignment must be found, and the session's local algorithm is initiated by generating its counting process.

In addition, our algorithm adapts well to inaccurate measurements of the problem data. For example, in practice, the latency values between the users and the agents are *perturbed*. Furthermore, transcoding latency is highly dependent on both content characteristics and agents' load. Hence, the transition rate of each session is subject to the perturbation with inaccurate values of either $\Phi_{s,f}$ and $\Phi_{s,f'}$. In this situation, the Markov-based algorithm converges to another steady-state point, since the objective value is perturbed. The Markov approximation framework comes with an optimality gap bound when the problem data is perturbed.

We assume the perturbed Φ_f belongs to one of the following discrete values:

$$[\Phi_f - \Delta_f, \ldots, \Phi_f - \frac{1}{n_f}\Delta_f, \Phi_f, \Phi_f + \frac{1}{n_f}\Delta_f, \ldots, \Phi_f + \Delta_f]$$

and the perturbed Φ_f takes the value $\Phi_f + j/n_f\Delta_f$ with probability $\eta_{j,f}$ and $\sum_{j=-n_f}^{n_f} \eta_{j,f} = 1$, where Δ_f is the error bound on configuration f and n_f is a positive constant.

Theorem 1: The stationary distribution of the perturbed assignment-hopping Markov chain is

$$\bar{p}_f = \frac{\delta_f \exp(-\beta\Phi_f)}{\sum_{f'\in\mathcal{F}} \delta_{f'} \exp(-\beta\Phi_{f'})}, \quad \forall f \in \mathcal{F} \tag{13}$$

where $\delta_f = \sum_{j=-n_f}^{n_f} \eta_{j,f} \exp(\beta\frac{j\Delta_f}{n_f})$, and optimality gaps are

$$0 \le \Phi^{\text{avg}} - \Phi^{\min} \le \frac{(U + \theta^{\text{sum}}) \log L}{\beta}, \tag{14}$$

$$0 \le \bar{\Phi}^{\text{avg}} - \Phi^{\min} \le \frac{(U + \theta^{\text{sum}}) \log L}{\beta} + \Delta^{\max} \tag{15}$$

where $\theta^{\text{sum}} = \sum_{u\in\mathcal{U}} \sum_{v\in\mathcal{U}} \theta_{uv}$ is the total number of transcoding tasks, $\Delta^{\max} = \max_{f\in\mathcal{F}} \Delta_f$ is the maximum perturbation error, $\Phi^{\min} = \min_{f\in\mathcal{F}} \Phi_f$ is the optimal value of the **UAP**, $\Phi^{\text{avg}} = \sum_{f\in\mathcal{F}} p_f^\star \Phi_f$ is the expected objective with the original Markov chain, and $\bar{\Phi}^{\text{avg}} = \sum_{f\in\mathcal{F}} \bar{p}_f \Phi_f$ is the expected objective with the perturbed Markov chain.

The proof is given in the Appendix. (14) and (15) demonstrate that when β increases the optimality gap of the perturbed Markov chain decreases; however, larger β values increase the convergence time of Algorithm V-B [35].

VI. AgRank: A Fast Session Bootstrapping Algorithm

We proceed to design an agent ranking algorithm for identifying a good starting feasible assignment solution, for bootstrapping the Markov approximation based algorithm. The intuition is that if Algorithm V-B can start from a close-to-optimal assignment, not only high-quality conferencing experience can be provided to the users starting from the beginning, but also fast convergence of the algorithm can be achieved.

A first idea might be using the *nearest* assignment policy [6], [7] as the initial assignment. However, the motivating example in Section I clearly evinces that nearest assignment is a *resource-oblivious* and *inter-agent proximity oblivious* policy, which only considers the one-hop distance between the user and the nearest agent, while this agent might be far away from other agents and also might impose severe inter-agent traffic to the service provider. Instead, we prefer to seek a *resource-aware and proximity-aware* policy.

In a nutshell of the algorithm which we refer to as *AgRank*, upon the start of a session, a potential agent of the session (e.g., the nearest agent to the session initiator) identifies a set of potential agents, ranks the agents, and assigns the users and transcoding tasks based on the ranking. Based on the example in Fig. 2, inter-agent delay is important in agent ranking, in addition to the agents' residual capacities and user-to-agent delay. The design of *AgRank* is motivated by the idea of Google's PageRank [37] and topology-aware node ranking in virtual network embedding [38] and is summarized in Algorithm 2.

Algorithm 2: *AgRank* (for Each Session s).

```
   // The 1st step - constructing
   potential agents
1  N(u) ← ∅ // set of potential agents of user u
2  N(s) ← ∅ // set of potential agents of session s
3  foreach user u ∈ U(s) do
4  |  N(u) ← top n^ngbr nearest agents to u in L.
5  |  N(s) ← N(s) + N(u)
6  end
   // The 2nd step - agent ranking
7  ε > 0, t ← 0
8  Initialize π_l[0] = (û_l+d̂_l+t̂_l+ô_l) / Σ_{k∈L}(û_k+d̂_k+t̂_k+ô_k),  l ∈ N(s)
      // û_l, d̂_l, t̂_l, and ô_l are the normalized
      residual quadruple of agent l
9  repeat
10 |  π^T[t+1] ← π^T[t] D̂
11 |  δ ← ||π[t+1] − π[t]||
12 |  t ← t+1
13 until δ < ε
14 π* ← π[t]
   // The 3rd step - user assignment
15 foreach user u ∈ U(s) do
16 |  Assign u to l_u^sel ← arg max_{l∈N(u)} π*_l
17 end
```

A. Constructing the Potential Agent List

In the first step, the set of all potential agents is constructed. Toward this, a set of top n^{ngbr} closest agents, $\mathcal{N}(u)$, for user u are picked as the possible agents and then the set of potential agents of the session, $\mathcal{N}(s)$, is constructed by putting together $\mathcal{N}(u)$ of all users (Lines 1–6). The parameter $n^{\mathrm{ngbr}} \in [1, L]$ is the maximum number of potential agents for each user that could be set on a per-session or per-user basis. Setting $n^{\mathrm{ngbr}} = 1$ yields the nearest assignment. In addition, $n^{\mathrm{ngbr}} = L$ results in subscribing all users to the highest ranked agent, which is similar to the traditional C/S architecture, where the whole session is maintained in a single server.

B. Agent Ranking

The second step is to rank the potential agents based on a random walk model [37]. We define the initial ranking of agent $l \in \mathcal{N}(s)$ as in Line 8, based on the normalized residual quadruple of agent l. In this way, the initial ranking of the agents is aware of the resource availability of the potential agents which turns *AgRank* into a *resource-aware* algorithm. Let $\hat{D} = [\hat{D}_{lk}]_{|\mathcal{N}(s)|\times|\mathcal{N}(s)|}$ as normalized inter-agent delay matrix where $\hat{D}_{lk} = (\min_{l',k' \in \mathcal{N}(s)} D_{l'k'})/D_{lk}$, and $\boldsymbol{\pi} = [\pi_l]_{l\in\mathcal{L}}$ is the vector of ranking of the agent. The rank vector is updated iteratively with $\boldsymbol{\pi}^T[t+1] = \boldsymbol{\pi}^T[t]\hat{D}$, whose rationale is to capture inter-agent delay in ranking and find the optimal ranking of the agents (Lines 7–14). It has been shown that this iterative procedure converges very fast to a unique vector $\boldsymbol{\pi}^\star = [\pi_l^\star]_{l\in\mathcal{L}}$, as the optimal ranking of the agents [37].

C. User and Transcoding Assignment

Next, user u is assigned to the highest ranked agent within the set $\mathcal{N}(u)$ (Line 16). For transcoding task assignment, we apply the rule of thumb that when there are at least two destinations with the same downstream representations for the outgoing flow of a particular user, assigning the corresponding transcoding task at the source agent is a good solution, whose transcoded stream can be served to more than one destination. One may imagine other schemes for assigning the transcoding tasks, but here we are only seeking a good feasible one.

VII. Discussion

A. Differences With Similar Approaches

Note that in simulated annealing [39], Gibbs sampling, and other Monte Carlo Markov chain approaches [40], the general approach of iterative execution following a particular Markov chain is similar to the Markov approximation framework [19]. However, different from the Markov approximation framework, above approaches do not explicitly design the Markov chain in such a way that it can be implemented in a parallel manner. Thus, the alternative approaches cannot be used to devise

parallel algorithm for the **UAP**. In addition, the alternative approaches are inadequate in handling the dynamics in the system and perturbation of the input to the problem. As discussed in Section V-C, Markov approximation, however, is robust against both system dynamics and inaccurate data. Last but not the least, our framework can provide a performance guarantee that is characterized in (11). Finally, in another recent work [41], a solution approach based on spectral clustering methodology has been proposed for resource selection and task assignment in distributed computing environments. In this paper, we apply Markov approximation framework to solve our problem because of the need for parallel implementation and robustness against system dynamics and perturbation in input measurements.

B. Real-Time Assignment Migration Without User Experience Degradation

Our proposed algorithm executes in an iterative manner and eventually converges to a bounded neighborhood of the optimal solution. However, these iterations come at the expense of imposing overhead needed to establish the new assignments. In each migration, the users might suffer from a momentary interruption, such as a freeze in playback, since switching into a new cloud agent is in progress. To prevent degradation of the user experience, one may suggest to keep both the new and the old assignments alive during the switching process. Finally, we note that transcoding migration could be seamlessly implemented by finishing the current task in the previous agent and beginning the next transcoding in the next agent, following the idea of segmentation-based transcoding [15]. We give the implementation details in Section VIII.

C. Complexity Analysis

Recall that a representative agent at each session (e.g., the session initiator's agent) runs the proposed algorithms. By doing so, user devices are not involved in running the procedures of the algorithms, thereby there is no overhead in the user devices. At each iteration of Algorithm V-B, the time complexity of calculating all of the feasible solutions with only one different decision is $O(|\mathcal{U}(s)|^2 L)$. In addition, for each potential target assignment, a delay feasibility check is required which could be done by measuring the round-trip time of each target assignment and pruning those that are beyond the end-to-end delay requirement of conferencing session as in (8). Note that the transition probability in Line 13 of Algorithm V-B could be computed given the local input of the session. Thus, Algorithm 1 is a parallel algorithm that does not require the global knowledge of the other conferencing sessions.

The iterative scheme in *AgRank* yields precision ϵ with the number of iterations proportional to $\max\{1, -\log\epsilon\}$ [37]. Constructing candidate agents, user assignment, and transcoding assignment takes a computation time of $O(|\mathcal{U}(s)|L\log L)$, $O(|\mathcal{U}(s)|)$ and $O(|\mathcal{U}(s)|^2)$, respectively.

VIII. Performance Evaluation

Our experimental results are categorized into two different sets of experiments with different goals. First, we implement a prototype conferencing system (Section VIII-A), to investigate the practicality of the algorithms in real environment with several actual users in different locations. Second, using PlanetLab traces [21], we implement large-scale experiments (Section VIII-B), to investigate the effectiveness of the proposed algorithms in Internet-scale scenarios.

The baseline for evaluating the effectiveness of the proposed solution is the *nearest* assignment policy (*Nrst*), i.e., the assignment policy used in *Airlift* [6] and *vSkyConf* [7]. The transcoding tasks are assigned to the agent of the sender of the stream at the initial assignment in either *Nrst* or our proposed *AgRank*. For the performance metrics, we report the inter-agent traffic (corresponding to the operational cost) and the conferencing delay, i.e., the average delay of all of the users, where the delay of each user is $d_u = \max_{v:u\in\mathcal{P}(v)} d_{vu}$. We set $D^{\max} = 400$ ms according to ITU-U G.114 [11]. Finally, as the proposed algorithm runs in an iterative manner, we report the evolution of the results over the time, wherein the initial values are either *Nrst* [6], [7] or our proposed *AgRank* assignments.

A. Experiments on Prototype System

1) Prototype Overview and Setup: A prototype cloud video conferencing system is implemented based on our solution design. The OpenCV library [42] is used to record video streams of device cameras in two different representations and to transcode the streams. In addition, we deploy six Linux-based EC2 instances in different regions. These virtual machines are used as the cloud agents in our prototype. A software is implemented and installed on the virtual machines to run our solutions and to stream the videos and also transcode different representations. Conferencing users are located in ten regions (five in the US and Canada, three in Asia, one in Middle-east, and one in Europe). We install a lightweight video conferencing application on the user devices. This software only transfers the conferencing videos to/from the virtual machines.

Unless otherwise mentioned, the bandwidth and transcoding capacities of the cloud agents are considered sufficiently large. In addition, the transcoding latency of agents are in $[30, 60]$ ms, which varies based on the processing capability of each EC2 virtual machine. We remark that the agent associated with the initiator of each session is responsible for executing Algorithm V-B and *AgRank*, hence by migrating the execution of the algorithms to the cloud agents, no additional overhead is imposed to the client devices. Finally, each session is established among 3–5 participants and there are in total ten sessions.

2) Implementation Details of Algorithm V-B: As for the main parameter in Algorithm V-B, we set $\beta = 400$. This value is in order of the logarithm of the state space of the problem [19]. Moreover, we set timer expiration to $[0, 10]$ seconds based on exponential distribution. The migration transparency (See Section VII for details) is done as follows. First, we remark that at each iteration one assignment is migrated. In this way, if the previous assignment is teared down immediately, there would be an interruption on the receiving streams of the other participants. In the prototype system, we can see a frozen screen for 2–3 frames, which is less than a second. Our solution to mitigate

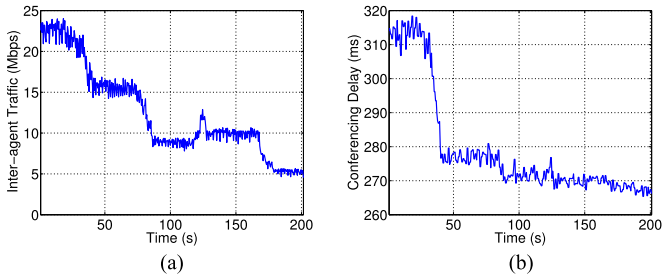


Fig. 6. Evolution of traffic and delay over time (200 s) by executing Algorithm 1 with $\beta = 400$ and *Nrst* for initial assignment. (a) Inter-agent traffic. (b) Conferencing delay.

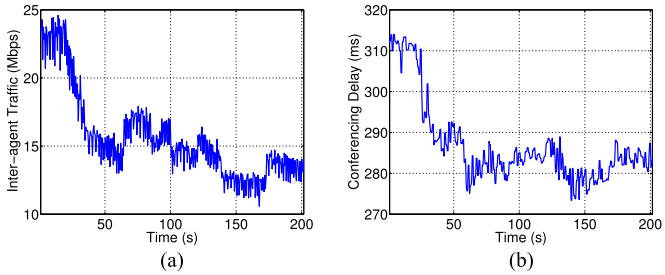


Fig. 7. Evolution of traffic and delay over time (200 s) by executing Algorithm 1 with $\beta = 200$ and *Nrst* for initial assignment. (a) Inter-agent traffic. (b) Conferencing delay.

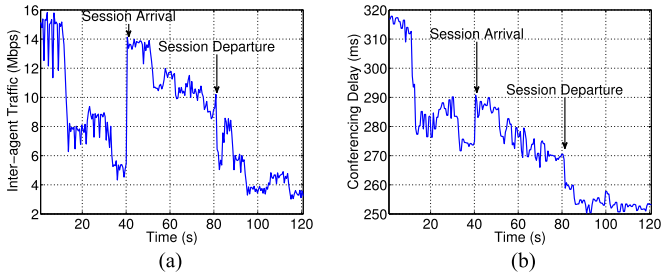


Fig. 8. Evolution of traffic and delay over time by executing Algorithm 1 with $\beta = 400$ in the presence of session arrival/departure. (a) Inter-agent traffic. (b) Conferencing delay.

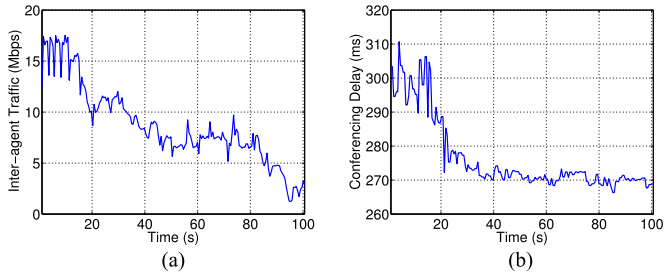


Fig. 9. Evolution of traffic and delay over time (100 s) by executing Algorithm 1 with $\beta = 400$ and *AgRank* with $n^{\mathrm{ngbr}} = 2$ for initial assignment. (a) Inter-agent traffic. (b) Conferencing delay.

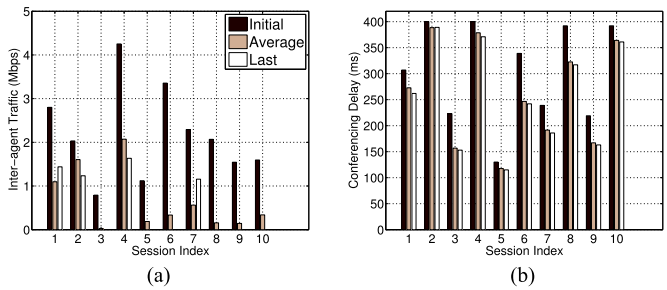


Fig. 10. Per-session improvements of Algorithm 1 with *Nrst* as the initial assignment. (a) Traffic per session. (b) Delay per session.

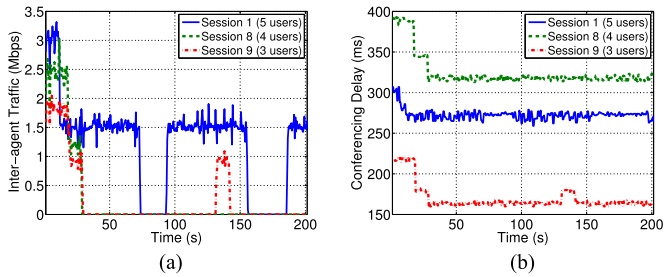


Fig. 11. Evolution of traffic and delay with Algorithm 1 for the case of 3 sample sessions with different number of users. (a) Inter-agent traffic. (b) Conferencing delay.

this negative effect is as follows. The migrated device streams its video to both the previous and the new agents for a period of less than 30 ms, on average, according to the distance between user and agent. We note that this overhead, whose volume is around 13.2 Kb is negligible compared to the amount of traffic reduction after migration due to the execution of our algorithm.

In Figs. 6–11, the initial traffic/delay values at time 0 are results of either the *Nrst* or *AgRank* assignment policies, and running Algorithm 1following the initial assignment reduces them over time.

3) Traffic and Delay Reduction of Algorithm 1: The results in Fig. 6 clearly show that Algorithm 1reduces traffic and delay of the initial assignment by *Nrst* significantly. In addition, after around 180 seconds it converges to the final assignment. To explore the impact of parameter β on the evolution of Algorithm 1, we also execute our algorithm with $\beta = 200$. The result is shown in Fig. 7. Algorithm 1with a lower value of β is highly fluctuating and converges to the final solution more slowly. Consequently, a larger value of β is preferable. In a scenario with session arrival/departure (Fig. 8), six sessions are initialized at time 0, four more sessions arrive at $t = 40$, and three sessions depart at $t = 80$. The results show that the algorithm adapts well to the dynamics and converges to the new stable assignment quickly.

4) Effectiveness of AgRank: Now, we report the results of Algorithm 1 when our proposed *AgRank* algorithm is used for the initial solution of user-to-agent assignment. The results in Fig. 9 demonstrate that *AgRank* outperforms *Nrst* by reducing initial inter-agent traffic from 22 Mbps to 15 Mbps. In addition, the initial conferencing delays are almost similar for both *AgRank* and *Nrst*. Also, the inter-agent traffic and conferencing delay values of *AgRank* at 100th second are the same as those with *Nrst* at around 200th second. This shows that starting from a good initial point by *AgRank*, Algorithm V-B converges faster. Note that *AgRank* is a fast algorithm, and takes less than 200 ms to find the initial assignment of the users after session

initialization. In addition to *AgRank*, which is a proper algorithm that reduces the convergence time of Algorithm V-B, the other candidate parameter that can further reduce the convergence is the countdown parameter which may, however, increase the migration overhead of the algorithm.

5) Case Study: The previous experiments demonstrated the average/aggregate results of ten sessions. In this experiment shown in Figs. 10–11, we study the results of a particular session in details. A comparison between the initial values in Fig. 10 with *Nrst* as initialization, and the average and last values of executing Algorithm 1 clearly shows that Algorithm 1reduces both inter-agent traffic and the conferencing delay of all of the users. More specifically, in Fig. 11, the traffic and delay of three random sessions are shown. In session 8, following *Nrst* policy, four users are assigned to three different EC2 instances in Tokyo, Singapore, and Ireland. However, by executing our algorithm, all of the users are assigned to Tokyo agent, which reduces the inter-agent traffic to zero. This migration eliminates the inter-agent traffic and also reduces the average delay of the users by 18% (388 to 318 ms). Finally, we note that because of the probabilistic nature of our solution design, a session may migrate to a worse assignment for some time, e.g., migration of session 9 at $t = 131$, but can recover soon, e.g., session 9 migrates back to the optimal assignment at $t = 141$. As a "rule-of-thumbs", our solution recommends assigning the users as much as possible to the same agents, but not if this increases the user delay.

B. Large-Scale Trace-Driven Experiments

1) Experimental Setup: In this section, we scrutinize the performance evaluation of solution using Internet-scale scenarios where we set up trace-driven experiments using 256 PlanetLab nodes as the users and the traces of seven EC2 instances as the agents. The user-to-agent and inter-agent delay values (approximately RTTs divided by 2) are from [43], [44]. We use four different representations 360 p, 480 p, 720 p, and 1080 p. A sparse transcoding matrix is considered such that 80% of users use 720p as their representation. We generate 100 random scenarios for each set of experiments and the statistical results are reported. In each experiment 200 out of 256 PlanetLab nodes are randomly chosen, who join different sessions, each of which with at most five users.

2) Trace Data Analysis: In this section, we briefly analyze the trace data [43], [44] of the latency values between 256 PlanetLab nodes and seven EC2 instances. We have two goals.

First, to investigate the contribution of inter-agent latency values in total end-to-end delays. Toward this, we report the inter-agent latency values in Table II. The values clearly show that the latency between some agents is quite large. On average, inter-agent latency values are 101 ms, which is about 25% of total end-to-end delay requirements (400 ms). This means that the contribution of inter-agent latency to end-to-end delay is not negligible in most cases. In addition, the results in Table II further corroborate our decision to consider inter-agent proximity values when ranking the potential agents in the *AgRank* algorithm.

TABLE II
INTER-AGENT DELAY MATRIX (IN MILLISECONDS) FOR AMAZON EC2 VMS [43], (VA: VIRGINIA, OR: OREGON, CA: CALIFORNIA, IR: IRELAND, SI: SINGAPORE, TO: TOKYO, SY: SYDNEY, SP: SAO PAULO)

	OR	CA	IR	SI	TO	SY	SP
VA	41.3	42.3	54	127	101	133	81.5
OR		11.5	85.5	117	67.5	100.5	104
CA			85	94	72	89.5	93
IR				117	138	168	120
SI					45.3	121.5	181.5
TO						72	150.5
SY							166.5

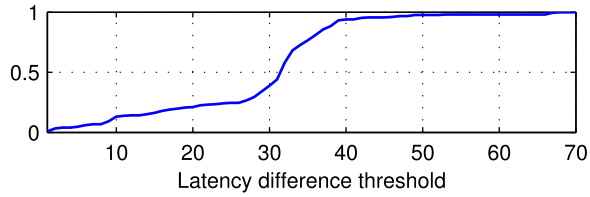


Fig. 12. Fraction of the number of users (among total 256 PlanetLab nodes) whose ρ_u is lower than a threshold.

TABLE III
EFFECT OF DESIGN PARAMETER α ON ALGORITHM 1

Algorithm	Cost	Init.	Algorithm 1		
			$\alpha_2 = 0$ (delay only)	$\alpha_1 = \alpha_2$	$\alpha_1 = 0$ (traffic only)
Nrst	Traffic	1443	979	829	521
	Delay	166	149	150	209
AgRank	Traffic	384	499	335	296
	Delay	176	162	163	214

Second, to further investigate the scenario in which some conferencing users are quite close to more than one cloud agent (e.g., the HK user in Fig. 2 is in the vicinity of both the TO and the SG cloud agents). In such cases, by choosing agents intelligently, both conferencing delay and inter-agent traffic can be reduced simultaneously. We introduce $\rho_u = min_{l,k\in\mathcal{L}}|H_{lu} - H_{ku}|, \forall u \in \mathcal{U}$, i.e., the difference between the user-to-agent latency of user u and its top two closest cloud agents (e.g., $\rho_4 = 27 - 20 = 7$, in Fig. 2). In Fig. 12, we show the fraction of the number of PlanetLab nodes (among all 256 nodes) whose ρ_u is lower than a threshold. The results show that 76% of the PlanetLab nodes are close to two different agents with a delay difference lower than 35 ms. Thus, in most cases each user is close enough to at least two cloud agents, so the design space for assignment is large.

3) Effect of Design Parameters: Parameters α_1 and α_2 are the weighting parameters of the hybrid objective function. In this experiment, we investigate the effect of these parameters on the performance of our algorithm. For simplicity $\alpha_3 = 0$. Table III and Figs. 13–14 show the highlights of the comparisons. The results in Table III demonstrate that when the importance of cost and delay are the same, i.e., $\alpha_1 = \alpha_2$, Algorithm 1with *Nrst* (*AgRank*) as initialization reduces the traffic

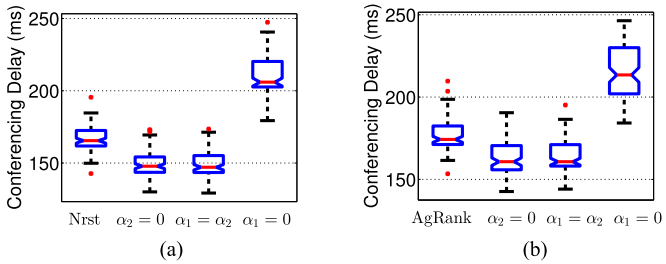


Fig. 13. Comparison of conferencing delay of Algorithm 1 with two initialization *AgRank* and *Nrst* and with different values of design parameter α.

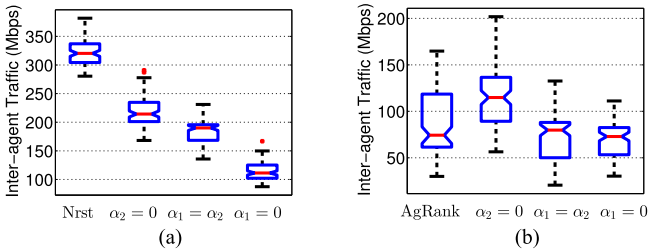


Fig. 14. Comparison of inter-agent traffic of Algorithm 1 with two initialization *AgRank* and *Nrst* and with different values of design parameter α.

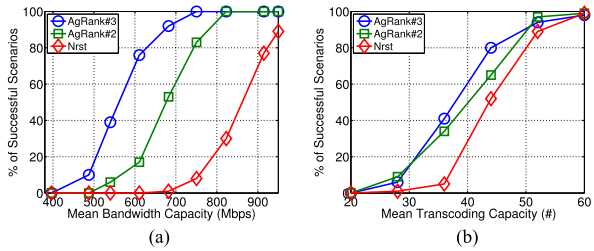


Fig. 15. Comparison of *AgRank* and *Nrst*. (a) Different bandwidth capacities. (b) Different transcoding capacities.

and the delay compared to *Nrst* by 42% (77%) and 10% (2%), respectively. Moreover, initialization by *AgRank* reduces the traffic by 73% at the expense of a 6% higher conferencing delay compared to *Nrst*. Figs. 13–14 show the statistical results for the conferencing delay and inter-agent traffic with different values of α. The results in Fig. 13 demonstrate that the conferencing delay of *AgRank* is slightly higher than *Nrst*. However, the inter-agent traffic of *AgRank* is significantly lower than in *Nrst*, as shown in Fig. 14.

Above result clearly shows that the nearest policy yields neither minimal delay nor minimal traffic cost. Furthermore, it signifies that our solution design improves the user experience and reduce the operational cost.

4) Performance of AgRank When Bandwidth and Transcoding Capacities of Agents Are Limited: The previous results showed that *AgRank* significantly outperforms *Nrst* by reducing the initial traffic cost. This reduction could be translated into an increased success rate of the initial assignment, i.e., all users in the system can successfully subscribe to agents, by serving more sessions with limited capacities of the agents. In Fig. 15, we show the percentage of successfully initialized scenarios (out of 100 random scenarios), with two

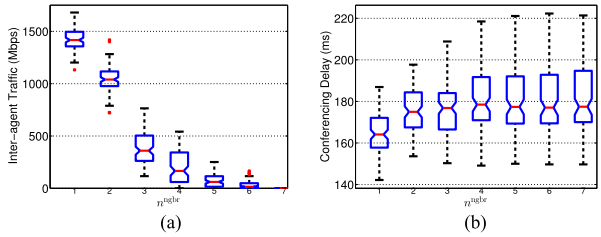


Fig. 16. Impact of n^{ngbr} on *AgRank*. (a) Inter-agent traffic. (b) Conferencing delay.

versions of *AgRank*, *AgRank#2* with $n^{\mathrm{ngbr}} = 2$ and *AgRank#3* with $n^{\mathrm{ngbr}} = 3$, and *Nrst* under different average bandwidth capacities (Fig. 15(a), unlimited transcoding capacity) and transcoding capacities of the agents (Fig. 15(b), unlimited bandwidth capacity). We observe that with *AgRank#3*, all 100 random scenarios can be successfully initialized under average bandwidth capacity 750 Mbps, while with the resource-oblivious *Nrst*, only 8% of the randomly generated scenarios can be successfully initialized. The higher success rates of *AgRank*#3 than *AgRank*#2 show that picking among a larger number of potential agents provides a larger feasible set. To explore this further, we compare the performance of *AgRank* under different values of n^{ngbr} in Fig. 16. Clearly, $n^{\mathrm{ngbr}} = 1$, by which *AgRank* is equivalent to *Nrst*, yields the highest traffic cost. With $n^{\mathrm{ngbr}} = L$, all users of each session are subscribing to one agent and hence suffer from long conferencing delays.

IX. Conclusion and Future Directions

This study addresses the problem of user-to-agent assignment and transcoding task assignment in cloud video conferencing architecture. Considering the challenges of the problem due to the underlying large-scale combinatorial problem, we devise a parallel and adaptive solution to optimize the assignment tasks. The algorithm achieves a suboptimal solution with a bounded performance guarantee. Observations on prototype system implementation corroborate our claim that user assignment is a critical design issue in cloud architecture that can lead to a big difference in entire system performance. In addition, trace-driven simulations demonstrate that our solution design outperforms the existing solutions in terms of reduced delay and cost, and thus demonstrates its viability as a win-win solution for both users and conferencing service provider. Finally, in future research, a promising direction to tackle is the more general problem in which other tasks, rather than transcoding are performed at the cloud agents. The problem could be further generalized to consider other types of communication and computation cloud resources for general interactive real-time multimedia applications.

Appendix

A. Proof of Theorem 1

To prove, first we should prove that the stationary distribution of the perturbed Markov chain is (13), this part is very similar to the proof in [35] and hence we proceed to prove (14)–(15).

Let us define the dirac distribution as follows:

$$\hat{p}_f = \begin{cases} 1, & \text{if } f = f^{\min} \\ 0, & \text{otherwise} \end{cases}$$

where $f^{\min}$ is the optimal solution of the **UAP** (i.e., $f^{\min} = \arg\min_{f\in\mathcal{F}} \Phi_f$). In addition, $p_f^\star$ as defined in (9) is the optimal solution for the **UAP**-β. Hence, using the result is (11) we have

$$\sum_{f\in\mathcal{F}} p_f^\star \Phi_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f^\star \log p_f^\star \le \sum_{f\in\mathcal{F}} \hat{p}_f \Phi_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} \hat{p}_f \log \hat{p}_f. \tag{16}$$

By Jensen's inequality [36] we get

$$-\sum_{f\in\mathcal{F}} p_f \log p_f = \sum_{f\in\mathcal{F}} p_f \log \frac{1}{p_f} \tag{17}$$

$$\le \log\left(\sum_{f\in\mathcal{F}} p_f \frac{1}{p_f}\right)$$

$$= \log|\mathcal{F}|. \tag{18}$$

Moreover, we have $\Phi^{\text{avg}} = \sum_{f\in\mathcal{F}} p_f^\star \Phi_f$ and $\Phi^{\min} = \sum_{f\in\mathcal{F}} \hat{p}_f \Phi_f$, by combining these equations, we get

$$\Phi^{\text{avg}} \le \Phi^{\min} + \frac{1}{\beta}\log|\mathcal{F}|. \tag{19}$$

We know that $|\mathcal{F}| \le L^{U+\theta^{\text{sum}}}$, where U, θ^{sum}, and L are the total numbers of the users, the transcoding tasks, and the agents, respectively, hence

$$0 \le \Phi^{\text{avg}} - \Phi^{\min} \le \frac{(U+\theta^{\text{sum}})\log L}{\beta}. \tag{20}$$

The above equation proves the inequality in (14).

In the next step, we prove the optimality gap of the perturbed Markov chain as characterized in (15). By reformulating (14) for the perturbed Markov chain we have

$$\sum_{f\in\mathcal{F}} \bar{p}_f \Phi'_f - \min_{f'\in\mathcal{F}} \Phi'_f \le \frac{(U+\theta^{\text{sum}})\log L}{\beta} \tag{21}$$

where Φ'_f is the modified objective function of the **UAP**-β in perturbed setting and is defined as $\Phi'_f = \Phi_f - \frac{\log\delta_f}{\beta}$, then by substituting the value of Φ'_f in (21) we get

$$\sum_{f\in\mathcal{F}} \bar{p}_f \left(\Phi_f - \frac{\log\delta_f}{\beta}\right) - \min_{f'\in\mathcal{F}} \left(\Phi_f - \frac{\log\delta_f}{\beta}\right) \le \frac{(U+\theta^{\text{sum}})\log L}{\beta}. \tag{22}$$

In addition, since $\Delta^{\max}$ is the maximum perturbation error, we have $\delta_f \le \exp(\beta\Delta^{\max})$, $f\in\mathcal{F}$ and hence

$$\frac{\log\delta_f}{\beta} \le \Delta^{\max}, \quad f\in\mathcal{F}. \tag{23}$$

Finally, we have $\bar{\Phi}^{\text{avg}} = \sum_{f\in\mathcal{F}} \bar{p}_f \Phi_f$ and then by combining (22) and (23), we get

$$0 \le \bar{\Phi}^{\text{avg}} - \Phi^{\min} \le \frac{(U+\theta^{\text{sum}})\log L}{\beta} + \Delta^{\max}. \tag{24}$$

This proves the inequality in (15).

REFERENCES

[1] M. H. Hajiesmaili, L. T. Mak, Z. Wang, C. Wu, M. Chen, and A. Khonsari, "Cost-effective low-delay cloud video conferencing," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 103–112.

[2] "Cisco VNI service adoption forecast, 2012–2017," White Paper, Feb., 2013.

[3] "Cisco VNI global mobile data traffic forecast update, 2013–2018," White Paper, Feb., 2014.

[4] Y. Wen, X. Zhu, J. J. Rodrigues, and C. W. Chen, "Cloud mobile media: Reflections and outlook," *IEEE Trans. Multimedia*, vol. 16, no. 4, pp. 885–902, Jun. 2014.

[5] Y. Hu, D. Niu, and Z. Li, "A geometric approach to server selection for interactive video streaming," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 840–851, May 2016.

[6] Y. Feng, B. Li, and B. Li, "Airlift: Video conferencing as a cloud service using inter-datacenter networks," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2012, pp. 1–11.

[7] Y. Wu, C. Wu, B. Li, and F. C. Lau, "vSkyConf: Cloud-assisted multi-party mobile video conferencing," in *Proc. ACM SIGCOMM Workshop Mobile Cloud Comput.*, 2013, pp. 33–38. [Online]. Available: http://doi.acm.org/10.1145/2491266.2491273

[8] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: measurement study of Google+, iChat, and Skype," in *Proc. ACM Internet Meas. Conf.*, 2012, pp. 371–384.

[9] E. Kurdoglu, Y. Liu, and Y. Wang, "Dealing with user heterogeneity in P2P multi-party video conferencing: Layered distribution versus partitioned simulcast," *IEEE Trans. Multimedia*, vol. 18, no. 1, pp. 90–101, Jan. 2016.

[10] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: A low-delay multi-party conferencing solution," in *Proc. ACM Multimedia*, 2011, pp. 493–502. [Online]. Available: http://doi.acm.org/10.1145/2072298.2072362

[11] "One-way transmission time," ITU-T Recommendations, G.114, 2003. [Online]. Available: https://www.itu.int/rec/T-REC-G.114-200305-I/en

[12] S.-P. Chuah, Y.-P. Tan, and Z. Chen, "Rate and power allocation for joint coding and transmission in wireless video chat applications," *IEEE Trans. Multimedia*, vol. 17, no. 5, pp. 687–699, May 2015.

[13] A. Khalek, C. Caramanis, and R. Heath, "Delay-constrained video transmission: Quality-driven resource allocation and scheduling," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 1, pp. 60–75, Feb. 2015.

[14] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A server's perspective of internet streaming delivery to mobile devices," in *Proc. IEEE INFOCOM*, 2012, pp. 1332–1340.

[15] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu, "Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1286–1296, Aug. 2015.

[16] J. Liao, P. Chou, C. Yuan, Y. Hu, and W. Zhu, "Online allocation of communication and computation resources for real-time multimedia services," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 670–683, Apr. 2013.

[17] Y. Jin, Y. Wen, and K. Guan, "Toward cost-efficient content placement in media cloud: Modeling and analysis," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 807–819, May 2016.

[18] D. G. Andersen, "Theoretical approaches to node assignment," Comput. Sci. Dept., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2002.

[19] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.

[20] "Amazon elastic compute cloud," 2017. [Online]. Available: http://aws.amazon.com/ec2/

[21] B. Chun *et al.*, "Planetlab: An overlay testbed for broad-coverage services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[22] S. Zhang, D. Niu, Y. Hu, and F. Liu, "Server selection and topology control for multi-party video conferences," in *Proc. ACM Netw. Oper. Syst. Support Digital Audio Video*, 2014, pp. 43–48.

[23] H. Bobarshad, M. van der Schaar, A. Aghvami, R. Dilmaghani, and M. Shikh-Bahaei, "Analytical modeling for delay-sensitive video over WLAN," *IEEE Trans. Multimedia*, vol. 14, no. 2, pp. 401–414, Apr. 2012.

[24] S. Wang, Y. Liu, and S. Dey, "Wireless network aware cloud scheduler for scalable cloud mobile gaming," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 2081–2086.

[25] M. H. Hajiesmaili, M. S. Talebi, and A. Khonsari, "Multi-period network rate allocation with end-to-end delay constraints," *IEEE Trans. Control Netw. Syst.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/document/7869285/

[26] M. H. Hajiesmaili, A. Khonsari, A. Sehati, and M. S. Talebi, "Content-aware rate allocation for efficient video streaming via dynamic network utility maximization," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 2016–2027, 2012.
[27] M. S. Talebi, A. Khonsari, and M. H. Hajiesmaili, "Utility-proportional bandwidth sharing for multimedia transmission supporting scalable video coding," *Comput. Commun.*, vol. 33, no. 13, pp. 1543–1556, 2010.
[28] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. IEEE INFOCOM*, 2012, pp. 702–710.
[29] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surv. Tut.*, vol. 15, no. 4, pp. 1888–1906, Jul.–Sep. 2013.
[30] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355737
[31] E. Rodriguez, G. Alkmim, D. Batista, and N. da Fonseca, "Live migration in green virtualized networks," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 2262–2266.
[32] X. Chen, Y. Luo, and J. Wang, "Virtual network embedding with border matching," in *Proc. IEEE COMSNETS*, 2012, pp. 1–8.
[33] Y. Jin, Y. Wen, H. Hu, and M.-J. Montpetit, "Reducing operational costs in cloud social TV: An opportunity for cloud cloning," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1739–1751, Oct. 2014.
[34] B. Alinia, M. H. Hajiesmaili, and A. Khonsari, "On the construction of maximum-quality aggregation trees in deadline-constrained WSNs," in *Proc. IEEE INFOCOM*, 2015, pp. 226–234.
[35] S. Zhang, Z. Shao, M. Chen, and L. Jiang, "Optimal distributed P2P streaming under node degree bounds," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 717–730, Jun. 2014.
[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ., 2004.
[37] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Int. Tech.*, vol. 5, no. 1, pp. 92–128, 2005.
[38] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, 2011. [Online]. Available: http://doi.acm.org/10.1145/1971162.1971168
[39] P. J. Van Laarhoven and E. H. Aarts, *Simulated Annealing*. New York, NY, USA: Springer, 1987.
[40] P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. New York, NY, USA: Springer, 1999, vol. 31.
[41] N. D. Doulamis, P. Kokkinos, and E. Varvarigos, "Resource selection for tasks with time requirements using spectral clustering," *IEEE Trans. Comput.*, vol. 63, no. 2, pp. 461–474, Feb. 2014.
[42] "OpenCV," 2017. [Online]. Available: http://opencv.org/
[43] P. Bailis, A. Davidson, A. Fekete, A. Ghodsi, J. M. Hellerstein, and I. Stoica, "Highly available transactions: Virtues and limitations," in *Proc. Very Large Databases*, 2014, pp. 181–192.
[44] Z. Wu and H. V. Madhyastha, "Understanding the latency benefits of multi-cloud webservice deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 1, pp. 13–20, 2013.

Mohammad H. Hajiesmaili received the B.Sc. degree in computer engineering from the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, in 2007, and the M.Sc. and Ph.D. degrees in computer engineering from the Electrical and Computer Engineering Department, University of Tehran, Tehran, Iran, in 2009 and 2014, respectively.

He was a Postdoctoral Fellow with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, from 2014 to 2016. He is currently a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, USA. His research interests include optimization, algorithm, and mechanism design in communication, energy, and transportation networks.

Lok To Mak received the B.Eng. degree from the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, in 2014, where he is currently working toward the M.Phil. degree.

His research interests include multimedia networking and electric vehicle scheduling.

Zhi Wang (S'10–M'14) received the B.E. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2008 and 2014, respectively.

He is currently an Assistant Professor with Tsinghua University. His research areas include multimedia big data, mobile cloud computing, and large-scale multimedia systems.

Prof. Wang was the recipient of the China Computer Federation Outstanding Doctoral Dissertation Award in 2014, the ACM Multimedia Best Paper Award in 2012, and the MMM Best Student Paper Award in 2015.

Chuan Wu (SM'16) received the B.Eng. and M.Eng. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2000 and 2002, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, in 2008.

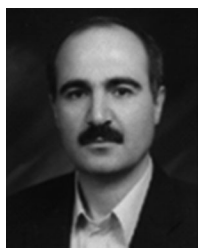
Since September 2008, she has been with the Department of Computer Science, University of Hong Kong, Pokfulam, Hong Kong, where she is currently an Associate Professor. Her research interests include cloud computing, data center networking, distributed machine learning, and big data analytics.

Dr. Wu is a member of ACM, and was the Chair of the Interest Group on Multimedia services and applications over Emerging Networks of the IEEE Multimedia Communication Technical Committee from 2012 to 2014. She has also served as TPC member and reviewer for international conferences and journals, including IEEE INFOCOM, IEEE ICDCS, ACM MM, IEEE ICC, IEEE GLOBECOM, TPDS, TON and TMM. She was the co-recipient of the best paper awards of HotPOST 2012 and ACM e-Energy 2016.

Minghua Chen (S'04–M'06–SM'13) received the B.Eng. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 1999 and 2001, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley (UC Berkeley), Berkeley, CA, USA, in 2006.

He spent one year visiting Microsoft Research Redmond as a Postdoctoral Researcher. In 2007, he joined the Department of Information Engineering, Chinese University of Hong Kong, where he is currently an Associate Professor. He is also an Adjunct Associate Professor with the Institute of Interdisciplinary Information Sciences, Tsinghua University. His current research interests include energy systems (e.g., smart power grids and energy-efficient data centers), intelligent transportation system, networked systems, online competitive optimization, distributed optimization, and delay-constrained network coding.

Prof. Chen was the recipient of the Eli Jury award from UC Berkeley in 2007 (presented to a graduate student or recent alumnus for outstanding achievement in the area of Systems, Communications, Control, or Signal Processing) and The Chinese University of Hong Kong Young Researcher Award in 2013. He also received several best paper awards, including the IEEE ICME Best Paper Award in 2009, the IEEE Transactions on Multimedia Prize Paper Award in 2009, and the ACM Multimedia Best Paper Award in 2012. He is currently an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING. He serves as TPC Co-Chair of ACM e-Energy 2016 and General Chair of ACM e-Energy 2017.



Ahmad Khonsari received the B.Sc. degree in electrical and computer engineering from Shahid-Beheshti University, Tehran, Iran, in 1991, the M.Sc. degree in computer engineering from Iran University of Science and Technology, Tehran, in 1996, and the Ph.D. degree in computer science from the University of Glasgow, Glasgow, U.K., in 2003.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Tehran, Tehran, and a Researcher with the School of Computer Science, Institute for Research in Fundamental Sciences, Tehran. His research interests include performance modeling/evaluation, wired/wireless networks, distributed systems, and high-performance computer architectures.