vSkyConf: Cloud-assisted Multi-party Mobile Video Conferencing

Yu Wu
Dept. of Computer Science
The University of Hong Kong
ywu@cs.hku.hk

Chuan Wu
Dept. of Computer Science
The University of Hong Kong
cwu@cs.hku.hk

Bo Li
Dept. of Computer Science and Engineering, Hong Kong University of Science and Technology
bli@cse.ust.hk

Francis C.M. Lau
Dept. of Computer Science
The University of Hong Kong
fcmlau@cs.hku.hk

ABSTRACT

As an important application in today's busy world, mobile video conferencing facilitates people's virtual face-to-face communication with friends, families and colleagues, via their mobile devices on the move. However, how to provision high-quality, multi-party video conferencing experiences over mobile devices is still an open challenge. The fundamental reason behind is the lack of computation and communication capacities on the mobile devices, to scale to large conferencing sessions. In this paper, we present *vSkyConf*, a cloud-assisted mobile video conferencing system to fundamentally improve the quality and scale of multi-party mobile video conferencing. By novelly employing a surrogate virtual machine in the cloud for each mobile user, we allow fully scalable communication among the conference participants via their surrogates, rather than directly. The surrogates exchange conferencing streams among each other, transcode the streams to the most appropriate bit rates, and buffer the streams for the most efficient delivery to the mobile recipients. A fully decentralized algorithm is designed to decide the best paths of streams and the most suitable surrogates for video transcoding along the paths, such that the limited bandwidth is fully utilized to deliver streams of the highest possible quality to the mobile recipients. We also carefully tailor a buffering mechanism on each surrogate to cooperate with efficient stream distribution. We have implemented *vSkyConf* based on Amazon EC2 and verified the excellent performance of our design, as compared to the widely adopted unicast solutions.

1. INTRODUCTION

Video conferencing has been widely deployed for virtual, face-to-face communication among separate parties, as a greener solution to replace many of the energy-expensive conference travels. Benefiting from the advances in mobile and wireless communication technologies, a number of mobile video conferencing applications [1] have emerged. Many of them rely on expensive, dedicated architectures, *e.g.*, multiple control units (MCU), to process signaling messages, transcode ingress session streams and disseminate multiple streams to each end device. Alternatively, distributed peer-to-peer (P2P) based mobile video conferencing systems have also sprung up, *e.g.*, Skype mobile leverages intermediate super nodes for session relays.

In order to find out how well the the existing mobile video conferencing systems support multi-party video conferencing over mobile devices, we have conducted a survey of the seven representative applications, with results given in [10].We observe that applications with infrastructure support can support more concurrent users in each session (but still typically no more than 4), at the cost of expensive up-front investment which may prohibit their wide adoption by small or medium institutions; P2P-based solutions are reluctant to allow group video calls for a fear of compromising call qualities, (*e.g.*, Skype only supports two-way visual communication on mobile phones), and most of them stick to a single flat streaming rate, or a limited number of bit rates (*e.g.*, 2).A fully-adaptive, multi-party mobile video conferencing solution is still pending to achieve.

We summarize the key challenges as follows: (1) The workload on each node in a video conferencing session, in terms of both processing and transmission, scales *quadratically* to the number of participants in the session, leading to significant challenge when using mobile devices for multi-party video conferencing. (2) Mobile users are equipped with different devices and downlink speeds; a high-quality solution should enable differentiated call qualities to different users, instead of a homogeneous video broadcast quality enforced by the low-end users, as in a traditional solution.

In this paper, we present *vSkyConf*, a cloud-assisted mobile video conferencing solution to fundamentally enable high-quality, multi-party video conferencing over heterogeneous mobile devices. The cloud computing paradigm offers ubiquitously accessible computing resources, with on-demand resource provisioning at the modest cost. The paradigm particularly compensates well for the inherent resource deficiencies of mobile devices, and catalyzes the undergoing evolution in the burgeoning mobile computing industry. In *vSkyConf*, we dynamically provision a virtual machine in the cloud as the exclusive surrogate for a dialed-in mobile user. Each mobile device disseminates/receives the video streams to/from its surrogate; the surrogates exchange conferencing streams among each

MCC'13, August 12, 2013, Hong Kong, China.

other, transcode the streams to the most appropriate bit rates, and buffer the streams for the most efficient delivery to the mobile recipients. By leveraging the more powerful processing capabilities and stable wired network bandwidths, mobile users shift those otherwise on-device tasks to the cloud, yielding superior power reduction and quality enhancement, as well as achieving fully scalable communication among the conference participants.

To realize such a solution, we design a fully decentralized, efficient algorithm to decide the best delivery paths of streams among the surrogates (possibly distributed in different cloud data centers), and the most suitable surrogates for video transcoding along the paths. We also carefully tailor a buffering mechanism on each surrogate to cooperate with the efficient stream distribution. Together they guarantee bounded, small end-to-end latencies and smooth stream playback at the mobile devices at the highest possible qualities, in each video conferencing session. We have implemented a preliminary version of *vSkyConf* based on Amazon EC2, and conducted experiments in the real-world settings. The results reveal the high scalability, full adaptability, and excellent video conferencing qualities achieved by our design, as compared to the widely adopted unicast solutions.

The remainder of this paper is organized as follows. We introduce related literature in Sec. 2, present unique challenges and the system architecture in Sec. 3, unfold design details in Sec. 4, introduce our prototype implementation and evalution in Sec. 5, and finally conclude the paper in Sec. 6.

2. RELATED WORK

Despite extensive studies during the past decades, video conferencing (VC) has recaptured people's interest in this new "smartphone" era, with a series of works and systems springing up recently [9][2] [4][7], which can be categorized into Server-to-Client (S/C) based and Peer-to-Peer (P2P) based solutions. Compared to the S/C-based structure, P2P has been deemed as a more promising and scalable solution. Both Ponec *et al.* [9] and Chen *et al.* [2] formulate utility maximization problems and enable multi-party VC by building multi-rate multicast trees. They focus more on the streaming rate allocation over physical links, but do not investigate much the transcoding flexibilities. Liang *et al.* [7] leverage the upload capacities of "helpers" from different swarms, in similar ways as adopted by Skype (not Skype mobile). Though promising, these solutions are difficult to achieve in practice among mobile users, who are reluctant to contribute resources to strangers due to constrained batteries and expensive cellular data fees. The dominant P2P-based solution in the real world is still pair-wise unicast (*e.g.*, Fring, Tango, etc.[1]), where a user directly exchanges streams with each of the other users in the same conferencing session. In this way, the limited uplink bandwidths of mobile devices significantly restrict the session size.

Cloud computing, as an agile solution, compensates well for the deficiencies of mobile devices for media streaming in terms of both processing and bandwidth supports. Traditional players in the VC market have recently claimed their support for mobile users via their private clouds. But their solutions are mostly centralized in a private cloud with abundant infrastructure resources to support enterprise users. For example, Vidyo [1] provides its cloud-based solution by provisioning virtual MCUs on top of their VidyoRouters, bearing similar flavors to their traditional dedicated infrastructures. In contrast, our design novelly provisions a VM surrogate for each mobile user in a public cloud in a more affordable manner, catering for the needs of ordinary mobile users in their daily life. A recent work by Feng *et al.* [3] optimally leverages inter-datacenter network bandwidths to maximize the overall throughput of all conferencing sessions, based on intra-session network coding. *vSkyConf* considers both dynamic session routing and adaptive session transcoding, and exploits surrogates in a cloud infrastructure for scalable video conferencing among mobile devices. Huang *et al.* [4] leverage clouds to encode videos into layered rates using scalable video coding (SVC), to enable differentiated streaming qualities to users with different available bandwidths. We do not consider SVC encoding in this work, since the encoding complexities inevitably incur intolerable delays for a time-sensitive application like video conferencing.

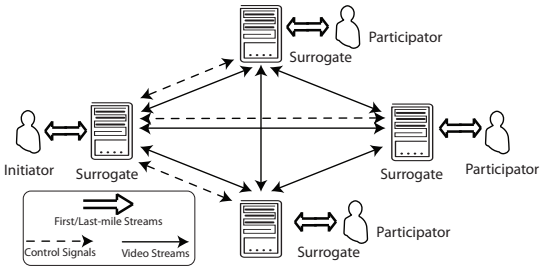

Figure 1: The architecture of vSkyConf.

3. ARCHITECTURE

vSkyConf enables efficient, peer-to-peer fashioned, multi-party mobile video conferencing via an IaaS cloud, with the architecture presented in Fig. 1. We refer to a video conference call among multiple mobile users as a *session*. The user which starts the conference call is the *initiator* of the session. A surrogate, *i.e.*, a virtual machine (VM) instance, is created in the IaaS cloud for each mobile user. The IaaS cloud consists of disparate data centers in different geographic locations, and the surrogate for each mobile user is assigned in a data center proximate to the user. As a proxy for the mobile device, a mobile user's surrogate is responsible for the following: (i) session maintenance, by exchanging control messages with other surrogates in a timely and efficient manner; (ii) video stream dissemination and transcoding, by receiving the video stream its corresponding mobile user produces, transcoding it into appropriate format(s), distributing it to other users' surrogates, and the other way round as well; (ii) efficient video stream buffering for its mobile user, for timely, smooth and robust streaming to the corresponding mobile device. A mobile user just needs to send the stream it generates and receive streams other users produce to and from its surrogate, and is effectively freed from power-consuming stream processing and intra-session communication. A gateway server in *vSkyConf* loosely keeps track of participating users and their surrogates, which can be implemented by a standalone server or VMs in the IaaS cloud.

Fig. 2 depicts the key modules implemented on a single surrogate, which can be divided into two parts:

Control Plane is the brain of the surrogate, responsible for control signaling between this surrogate and neighboring surrogates. It measures the latencies and bandwidths on the connections from/to neighboring surrogates, and all the collected information is stored in the "peer table", which constructs a partial view of the video conferencing topology from this surrogate's point of view. Utilizing the collected information, the surrogate computes routing paths for streams from its corresponding mobile user to other mobile users, and participates in the construction of optimal video dissemination trees. It also monitors the call qualities and determines the best video encoding parameters (codecs, bitrates, etc.) for streams from/to its mobile device.

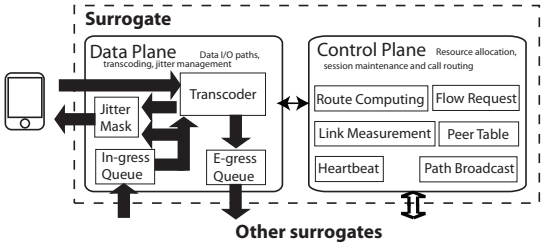

Figure 2: The key modules of a surrogate.

Data Plane is responsible for processing in/out video streams, in terms of both transcoding and forwarding, as directed by the control plane. The video stream from its mobile user is captured continuously and disseminated to other surrogates after necessary transcoding. In the reverse direction, all video streams from other mobile users, via their respective surrogates, are transcoded into appropriate rates (if necessary) and delivered to the mobile user by a key module "jitter mask", which deals with random jitters caused by fluctuations of processing and network latencies, as well as any anomalies along the dissemination paths.

Our design of *vSkyConf* observes the following principles.

Decentralized Control. Except necessary bootstrapping from the gateway server, each session is to be maintained by the surrogate of the initiator of a conference session, in order to provide good scalability and flexibility. The video routing and transcoding decisions are to be made in a fully distributed fashion by collaborations among surrogates.

Self-Evolving Routing Topology with Full Adaptivity. We seek to build a best routing topology among the surrogates for disseminating the stream from each participant, which achieves a small end-to-end latency and fully exploits the available bandwidths among the surrogates. Transcoding decisions to convert the original stream to acceptable formats/bit rates of the recipients are dynamically made at the best points along the dissemination paths, according to different computation capacities of the surrogates and needs of downstream mobile devices.

Robust, Smooth Video Streaming. To guarantee smooth stream playback at each mobile user even in cases of inaccurate route computation (*e.g.*, due to inaccurate estimates of link bandwidths and latencies), we seek to design an advanced error correction mechanism to search for better routing paths before the call quality drops, by monitoring a carefully designed jitter buffer at each surrogate.

4. DETAILED DESIGN

4.1 Session Maintenance

Establishment: When a mobile user logs in to the *vSkyConf* system via the gateway server, it is assigned a surrogate VM. The gateway can maintain information on a pool of available, pre-initiated VMs in the IaaS cloud, and assign one from the pool to a mobile user based on geographic proximity of the two, to expedite the service. The surrogate of the session initiator finds out IP addresses of surrogates of the other online users from the gateway server, which it wishes to invite to join the video conferencing session. The initiator then contacts and invites the interested participants through their surrogates directly, and maintains a list of IP addresses of all active surrogates in the session.

Tear-down: When a mobile user leaves the system, its surrogate VM is released and returned to the pool of available VMs in the IaaS cloud. If the initiator of a session departs, its hosting role is handed over to another substitute surrogate in the participant list.

4.2 Routing Computation

In a video conferencing session with S users, there are S streams, each produced by one of the mobile users, to be delivered to all the other users. We model a mathematical optimization problem for constructing efficient dissemination topologies of all streams in a session and deciding the optimal transcoding locations. We then design efficient, fully distributed heuristic to approach the optimal solution in a dynamic system. For transcoding, we practically only consider down-sampling of a stream, *i.e.*, the reduction of streaming bit rate, but not the reverse, since up-sampling provides no quality improvement but consumes unnecessary bandwidth. We also focus on transcoding due to mismatched bit rates of streams of the same format, while the case of transcoding from one format to another can be readily addressed with similar efforts.

4.2.1 Optimization Formulation

Let graph $G = (\mathcal{S}, \mathcal{E})$ represent the network of surrogates in a session, where $\mathcal{S}$ is the set of surrogates and $\mathcal{E}$ is the set of directed connections among the surrogates. For each surrogate $m \in |\mathcal{S}|$, let $\hat{m}$ represent the corresponding mobile user. Let $S = |\mathcal{S}|$. Suppose C_{ij} is the maximum available bandwidth on link $(i, j) \in \mathcal{E}$, and d_{ij} denotes the link latency. We refer to the stream from a surrogate $m \in \mathcal{S}$ as *flow* m, with source rate $R_{\hat{m}}^{(m)}$, which is the rate of incoming stream from mobile user $\hat{m}$ to surrogate m, determined by the source capturing rate by the user's mobile camera and the uplink rate from the mobile user. Let $R_{\hat{n}}^{(m)}$ be the maximum acceptable bit rate of flow m at mobile user $\hat{n}$, as decided by the last-mile down-link bandwidth from surrogate n to $\hat{n}$, and the allocation of this down-link bandwidth among streams from different users, *e.g.*, if user $\hat{n}$ sizes playback windows of streams from $S-1$ other conference participants equally on its device screen, $\frac{1}{S-1}$ of the down-link bandwidth should be allocated to each stream.

The multicast flow m from surrogate m to all other surrogates can be viewed as consisting of $S-1$ conceptual unicast flows [6], from m to each of the other surrogates, respectively. These conceptual unicast flows co-exist in the network without contending for link bandwidths, and the multicast flow rate on a link is the maximum of the rates of all the unicast flows going along this link. For ease of transcoding implementation, we restrict each unicast flow from m to n to be an integral flow along one path with the end-to-end rate $r_n^{(m)}$, and the multicast topology is the overlap of all the $S-1$ unicast flow paths. Let binary variable I_{ij}^{mn} indicate whether the conceptual unicast flow from m to n traverses link $(i, j) \in \mathcal{E}$, and $c_{ij}^{(m)}$ denote the actual rate of the multicast flow m on link (i, j).

Let function $\varphi_n(r_1, r_2)$ give the transcoding latency at surrogate n, if the rate r_1 of an ingress flow received by n is higher than the rate r_2 of the egress flow from n. $\varphi_n(r_1, r_2) = 0$ if $r_1 \leq r_2$. Typical transcoding steps are to decode the source stream of rate r_1 to an intermediate format, and then re-encode the stream from the intermedia format to the destination rate r_2 [8]. Hence, transcoding delay $\varphi_n(r_1, r_2)$ is monotonously increasing on both r_1 and r_2, and depends on computation capacity of the surrogate VM n: the more powerful the VM is, the faster the transcoding can be accomplished.

The quality of service in the conferencing session relies on two aspects: (i) the end-to-end latency and (ii) the flow rate received by each participant for each flow. We bound the end-to-end latency, from the time a source surrogate m emits flow m to the time a receiver surrogate n is ready to push the stream to its corresponding mobile user, by $L_n^{(m)}$, whose value is dynamically set as discussed

in Sec. 4.3. Let $U(\frac{r_n^{(m)}}{R_{\hat{n}}^{(m)}})$ be an increasing, concave utility function on the rate of flow m received by surrogate n, $r_n^{(m)}$. We maximize the aggregate utility of all receivers in all flows as our objective. The optimization problem is formulated in (1).

$$\max \sum_{m\in\mathcal{S}} \sum_{n\in\mathcal{S}, n\neq m} U(\frac{r_n^{(m)}}{R_{\hat{n}}^{(m)}}) \quad (1)$$

subject to:

$$\sum_{i:(i,j)\in\mathcal{E}} I_{ij}^{mn} - \sum_{k:(j,k)\in\mathcal{E}} I_{jk}^{mn} = b_j^{mn}, \forall j, m, n \in \mathcal{S}, m \neq n, \quad (2)$$

$$I_{ij}^{mn} r_n^{(m)} \leq c_{ij}^{(m)}, \forall (i,j) \in \mathcal{E}, m, n \in \mathcal{S}, m \neq n, \quad (3)$$

$$\sum_{m\in\mathcal{S}} c_{ij}^{(m)} \leq C_{ij}, \forall (i,j) \in \mathcal{E}, \quad (4)$$

$$\sum_{(i,j)\in\mathcal{E}} I_{ij}^{mn} d_{ij} + \sum_{(i,j)\in\mathcal{E}} \sum_{k:(j,k)\in\mathcal{E}} I_{ij}^{mn} I_{jk}^{mn} \varphi_j(c_{ij}^{(m)}, c_{jk}^{(m)}) + \varphi_n(\sum_{j:(j,n)\in\mathcal{E}} I_{jn}^{mn} c_{jn}^{(m)}, R_{\hat{n}}^{(m)}) \leq L_n^{(m)}, \forall m, n \in \mathcal{S}, m \neq n, \quad (5)$$

$$I_{ij}^{mn} \in \{0,1\}, \forall m, n \in \mathcal{S}, m \neq n, (i,j) \in \mathcal{E}, \quad (6)$$

$$0 \leq r_n^{(m)} \leq R_{\hat{m}}^{(m)}, \forall m, n \in \mathcal{S}, \quad (7)$$

$$0 \leq r_n^{(m)} \leq R_{\hat{n}}^{(m)}, \forall m, n \in \mathcal{S}, \quad (8)$$

where

$$b_j^{mn} = \begin{cases} -1, & j = m \\ 1, & j = n \\ 0, & otherwise \end{cases}.$$

Constraints (2) and (6) enforce a single path for the unicast flow from surrogate m to n, and ensures flow conservation along the path. Constraint (3) implies that the unicast flow from m to n with rate $r_n^{(m)}$ is conceptual, "hidden" in the actual multicast flow m with rate $c_{ij}^{(m)}$, on each link (i,j). Constraint (4) requires that the overall rate of actual flows from different sources should not exceed the capacity of each link. Constraint (5) bounds the end-to-end delay along the path from source surrogate m to receiver surrogate n, which consists of three parts: (i) the overall link delay along the path, $\sum_{(i,j)\in\mathcal{E}} I_{ij}^{mn} d_{ij}$; (ii) the sum of potential transcoding delay at intermediate surrogates j's along the path, *i.e.*, $\sum_{(i,j)\in\mathcal{E}} \sum_{k:(j,k)\in\mathcal{E}} I_{ij}^{mn} I_{jk}^{mn} \varphi_j(c_{ij}^{(m)}, c_{jk}^{(m)})$, where a surrogate j is on the path if there exist neighboring links (i,j) and (j,k), such that $I_{ij}^{mn} = 1$ and $I_{jk}^{mn} = 1$, and a transcoding delay occurs if the flow rate on (i,j), $c_{ij}^{(m)}$, is larger than the flow rate on (j,k), $c_{jk}^{(m)}$; (iii) the potential transcoding delay at surrogate n, $\varphi_n(\sum_{j:(j,n)\in\mathcal{E}} I_{jn}^{mn} c_{jn}^{(m)}, R_{\hat{n}}^{(m)})$, to transcode the received stream to the maximum receiving rate allowed at mobile user $\hat{n}$, if needed. Constraints (7) and (8) restrict the end-to-end rate of virtual unicast flow from surrogate m to n to be no larger than the maximum sending rate from mobile user $\hat{m}$ and the maximum receiving rate at mobile user $\hat{n}$.

The solutions to the optimization problem, $r_n^{(m)*}$, $c_{ij}^{(m)*}$, I_{ij}^{mn*}, $\forall m, n \in \mathcal{S}, n \neq m, (i,j) \in \mathcal{E}$, give us (i) the rate at which each mobile user $\hat{m}$ should send its stream to its surrogate m, which is the maximum of all conceptual unicast flow rates from m to the other surrogates, $\max_{n\in\mathcal{S}, n\neq m} r_n^{(m)}$; (ii) the delivery rate of flow m along each link (i,j) and hence the flow routing topology among the surrogates ($c_{ij}^{(m)*} = 0$ indicates flow m is not to be routed over link (i,j)); and (iii) where the transcoding of each flow m should happen, *i.e.*, a surrogate j where an egress flow rate $c_{jk}^{(m)}$ is smaller than the ingress rate $c_{ij}^{(m)}$ along the same conceptual unicast path, should transcode flow m to the lower rate.

4.2.2 Distributed Heuristic

The optimization problem (1) is non-convex with integer variables.We design an efficient heuristic algorithm, as given in Alg. 1 and Alg. 2, to decide flow routing, rate assignment and transcoding locations in a fully distributed fashion.

Algorithm 1 Flow Routing and Rate Allocation

1: Construct shortest-path trees from each surrogate m, $T^{(m)}$;
2: **if** $\exists m, n \in \mathcal{S}, \omega_n^{(m)} > L_n^{(m)}$ **then**
3: No feasible solution exists; **return** ;
4: **end if**
5: N_{ij} := Number of dissemination trees on (i,j);
6: $\forall (a,b) \in T^{(m)}, c_{a,b}^{(m)} := min_{k\in\mathcal{S}, (i,j)\in T^{(m)}}\{R_{\hat{k}}^{(m)}, \frac{C_{ij}}{N_{ij}}\}$;
7: Search for better routing paths, following Alg. 2;

We first decide a basic, feasible dissemination topology for each flow m, on which the end-to-end delay constraint for each receiver, constraint (5), is satisfied. Though the optimization problem (1) does not restrict the topologies into trees, we seek to construct a dissemination tree for each flow for ease of practical implementation. For conciseness, $\omega_n^{(m)}$ represents the overall latency (including both link and necessary transcoding latencies) for flow m from surrogate m to surrogate n. A shortest-path tree is constructed from surrogate m to all the other surrogates, using a distributed Bellman-ford algorithm (Line 1 in Alg. 1). If the overall link latency on the path from surrogates m to n is larger than $L_n^{(m)}$, we know that this pre-set end-to-end latency bound is by no means satisfiable, and should be adjusted to a more reasonable value (Lines 2-4). We then decide a basic, end-to-end rate of flow m on this shortest path tree, from surrogate m to all the other surrogates: the capacity C_{ij} of each link (i,j) is evenly divided by the (actual) flows generated by different surrogates, that pass through this link; the end-to-end rate of each flow m is set to the rate allocated to this flow on the bottleneck link its shortest-path tree spans (Lines 5-6).

Algorithm 2 Self-Evolving Route/Rate Adjustment at Surrogate n in Flow m

1: **while** $\exists (j,n) \in T^{(m)}, c_{jn}^{(m)} < R_n^{(m)}$ **do**
2: **if** $\exists (i,k) \in T^{(m)}, min\{c_{ik}^m, \bar{C}_{kn}\} > c_{jn}^{(m)}$ **then**
3: $\Lambda := \{n\} \bigcup \{q : (n,q) \in T^m\}$;
4: **if** $\forall p \in \Lambda, \omega_p^{(m)} \leq L_p^{(m)}$ **then**
5: $T^{(m)} := T^{(m)} - (j,n) + (k,n)$;
6: **end if**
7: **end if**
8: **end while**

Based on the basic dissemination topology, each surrogate then carries out dynamic edge and rate adjustments by following Alg. 2, to maximally utilize the available capacity to stream high-quality streams, without violating the latency constraints. For each flow m, suppose surrogate j is the parent to surrogate n on the current dissemination tree of flow m. n contacts other neighboring surrogates in the flow, to discover if there is a better path from source surrogate m with higher capacity via another parent k. It compares the current receiving rate $c_{jn}^{(m)}$ from j with the potential receiving

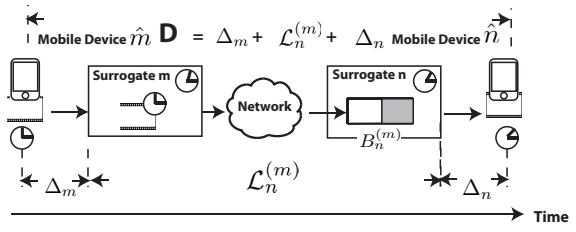

Figure 3: An illustration of the end-to-end delay for flow m.

rate from k, $\min(c_{ik}^{(m)}, \bar{C}_{kn})$, where we suppose surrogate i is the parent of k in the current tree, and $\bar{C}_{kn}$ is the remaining available bandwidth on link (k, n) (Line 2 in Alg. 2). If the potential receiving rate via k is larger, n needs to further evaluate the increased latency along the new path, due to changes of link latencies and potential transcoding latencies at k and n. Only if the latency of the new path from m to n, *i.e.*, $\omega_n^{(m)}$, is still within $L_n^{(m)}$, and the updated latency to each of the descent surrogates from n on the tree is still within the respective delay bound, can n safely change its parent from j to k (Lines 3-6).

4.3 Jitter Masking

In multi-party video conferencing, synchronization among streams received at all users is crucial to users' perceived quality of experience. It is much desired that the video frames captured at all users at the same time, are played at all the recipient user devices at the same time. We design an effective buffering mechanism, which collaborates with the routing algorithms, for this purpose.

Surrogate n maintains a buffer $B_n^{(m)}$ for each stream $m \in \mathcal{S}/\{n\}$ from each of the other surrogates. The buffer holds video packets of flow m, ready to be delivered to mobile device n. *vSkypeConf* enforces an end-to-end delay of D, from when a video frame is captured at one mobile device, to the time it is synchronously played at all the other mobile devices. The value of D can be set based on reasonable estimation of the maximum delay between two mobile users in the system, and should fall in the acceptable delay range for real-time communication. Let Δ_m indicate delay between mobile device $\hat{m}$ and its surrogate m, $\forall m \in \mathcal{S}$. For a frame in buffer $B_n^{(m)}$, which is produced at t at the source $\hat{m}$, it will be pushed out from the buffer no earlier than $t + \mathcal{L}_n^{(m)}$, where $\mathcal{L}_n^{(m)} = D - \Delta_m - \Delta_n$, in order to guarantee playback of the frame at the mobile device $\hat{n}$ at $t + D$ (Fig. 3).

If there were no jitter in the cloud, we could set the delay bound $L_n^{(m)}$ in optimization (1), used to find the routing path from surrogate m to surrogate n, to $L_n^{(m)} = \mathcal{L}_n^{(m)}$, and rest assured that the buffer will never starve. However, in a practical system, jitter may occur due to various reasons, *e.g.*, variation of transcoding delay at surrogates, inaccurate estimate of link delay and bandwidth when running our routing algorithm, etc. Hence, $L_n^{(m)}$ in the optimization for route selection should be set smaller than $\mathcal{L}_n^{(m)}$, in order to absorb the inaccuracy and jitter.

A series of measurement work [5] have shown that jitter on a network path approximately follows a normal distribution [11]. Let $J_n^{(m)}$ be a random variable, representing the path delay from surrogate m to surrogate n, such that $J_n^{(m)} \sim N(\mu, \sigma^2)$, where μ is the mean and σ is the standard deviation. For a normal distribution, we can derive that 99.97% of the samples fall within the range of $(-\infty, \mu + 3.4\sigma)$. If we set $L_n^{(m)}$ to the mean μ in the path delay distribution while allowing $\mathcal{L}_n^{(m)} = \mu + 3.4\sigma$, we derive $L_n^{(m)} = \mathcal{L}_n^{(m)} - 3.4\sigma$. Using this $L_n^{(m)}$ in solving optimization (1), we can make sure that 99.97% of the video packets, following the path selected, can be sent out from surrogate n by $\mathcal{L}_n^{(m)}$, and catch their playback deadlines at the mobile device $\hat{m}$.

In *vSkyConf*, each surrogate n dynamically estimates the delay variance σ along the path from m to n, based on inter-packet latencies of flow m it receives. It also observes the current queueing delay in buffer $B_n^{(m)}$, and adjusts $L_n^{(m)}$ used in path selection according to $L_n^{(m)} = \mathcal{L}_n^{(m)} - 3.4\sigma$. That is, if there are less packets in the buffer caused by larger delay variance, it tunes $L_n^{(m)}$ down to be more stringent on the latency requirement in the path selection; otherwise, it tunes $L_n^{(m)}$ up to explore paths with better bandwidths. In this way, this buffering mechanism at the surrogates collaborates with the routing algorithm, to deal with randomness in the system and inaccuracy in the computation, while maximally guaranteeing synchronized playback of all streams at all the mobile devices.

5. PERFORMANCE EVALUATION

We implement a prototype of *vSkyConf* and deploy it in Amazon Elastic Compute Cloud (EC2). Surrogates are provisioned from "ap-southeast-1a" (Singapore), "eu-west-1a" (Ireland), "us-west-1b" (California) and "us-east-1a" (Virginia), for users located near the respective region. Each mobile user is emulated by a machine near its assigned EC2 instance (within 50 ms), where video frames are generated at a constant rate around 768 kbps. We implement an application-layer packet controller to limit the uplink and downlink bandwidths of each user within the range of [1.5, 2] Mbps — the same as those on regular 3G cellular connections. We apply the concave function $\log(x)$ as the utility function in our routing computation. Each surrogate dynamically measures the link delays to its neighboring surrogates. The transcoding latencies are pre-evaluated on the VM instances and used in our routing computation, for transcoding from 768kbps to 256kbps, from 768kbps to 128kbps, from 256kbps to 128kbps, respectively. At each user, the stream from one of the other conference participants is displayed in a large screen (corresponding to a maximal acceptable streaming rate of 768 kbps), and streams from other participants are displayed using smaller screens (corresponding to maximal acceptable streaming rates of 128 kbps or 256 kbps). A fixed 400 ms end-to-end delay (D in Sec. 4.3) is configured, and the buffer for each flow at each surrogate is set to a size corresponding to 400ms stream playback.

5.1 Adaptive Flow Rates

We test a video conferencing session among 10 participants: 5 from Hong Kong, 1 from Europe, 2 from US West and 2 from US East. As a potential bottleneck for scalability, the surrogate of the session initiator is responsible for session maintenance by exchanging messages with the other surrogates in the session. We therefore investigate the conferencing performance at the initiator's surrogate: if its performance is satisfactory, then the performance at the other surrogates should be even better.

Fig. 4 plots the flow rates of streams from 3 out of the other 9 participants. "Flow-b" is the flow with a maximum streaming rate of 768kbps; "flow-a" and "flow-c" correspond to a maximum streaming rate of 128kbps and 256kbps respectively, with the user of the latter joining the session at a later time. We can see that each flow goes through a "fast" start stage, when the basic stream dissemination topology is being constructed (as introduced in Sec. 4.2), and then evolve towards their maximal acceptable rates. Fig. 5 presents the load in the jitter buffer for "flow-b" at the initiator's surrogate, where we see that the buffering level varies significantly when "flow-b" takes a path with large delay jitters. When our rout-

ing algorithm redirects "flow-b" through a better path, a more stable buffering level is achieved later on. Fig. 6 shows the latency experienced by each flow, from the corresponding source surrogate to the initiator's surrogate. We observe that latencies only vary slightly whenever the routing paths are adjusted, and can well meet the end-to-end latency required (400ms).

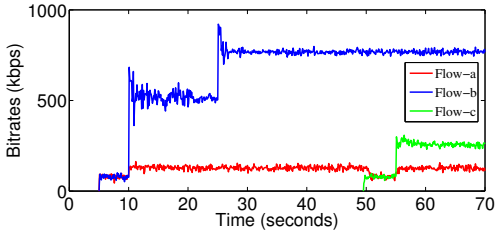

Figure 4: Flow rates at the initiator's surrogate.

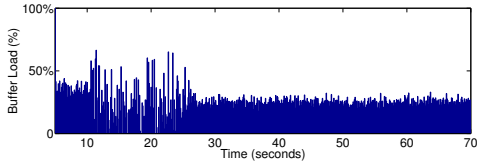

Figure 5: Load of flow-b's buffer at the initiator's surrogate.

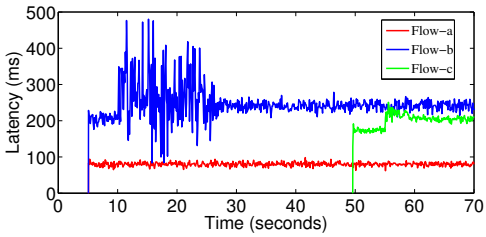

Figure 6: Flow latencies at the initiator's surrogate.

5.2 Comparison with a Unicast Solution

We next evaluate the performance of *vSkyConf* against a unicast scheme typically applied in P2P video conferencing, where each flow is directly transmitted from the source to the destination via the network. We establish a 3-user video conferencing session, and emulate a 50-minute long conferencing session with one user coming from each of the regions, Hong Kong, Europe and west US. Fig. 7 shows the perceived end-to-end latencies of the two flows received at the Hong Kong user, where "eu" stands for Europe and "usw" stands for west US. We can see that the end-to-end latency achieved with *vSkyConf* is generally smaller, and much more stable than that achieved by the unicast solution, verifying the smooth stream playback experienced by *vSkyConf* users. This validates that our cloud-assisted design is suitable to achieve high-quality video conferencing among multiple mobile participants.

6. CONCLUSION AND FUTURE WORK

This paper presents *vSkyConf*, a cloud-assisted mobile video conferencing solution, designed to fundamentally improve the quality

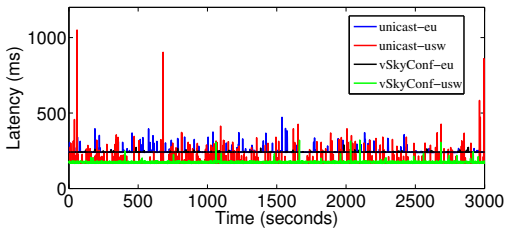

Figure 7: End-to-end delay at the Hong Kong user.

and scale of multi-party mobile video conferencing. We employ a virtual machine in a cloud infrastructure as the proxy for each mobile user, to send and to receive conferencing streams, and to transcode the streams into proper formats/rates. We design a fully decentralized, efficient algorithm to decide the best paths of stream dissemination and the most suitable surrogates for video transcoding along the paths, and tailor a buffering mechanism on each surrogate to cooperate with efficient stream distribution. We have implemented the *vSkyConf* prototype on Amazon EC2 and verified its excellent performance. In our ongoing work, we are implementing *vSkyConf* on real mobile devices and evaluating its performance under more dynamic settings.

Acknowledgments

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 717812E.

7. REFERENCES

[1] *Survey on mobile video conferencing apps*, http://culiuliu.net/videoconf/survey.html.

[2] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li. Celerity: a low-delay multi-party conferencing solution. In *Proceedings of the 19th ACM international conference on Multimedia*, New York, NY, USA, 2011.

[3] Y. Feng, B. Li, and B. Li. Airlift: Video conferencing as a cloud service using inter-datacenter networks bibtex. In *Proceedings of IEEE ICNP*, 2012.

[4] Z. Huang, C. Mei, L. E. Li, and T. Woo. Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy. In *Proceedings of INFOCOM*, 2011.

[5] M. J. Karam and F. A. Tobagi. Analysis of delay and delay jitter of voice traffic in the internet. *Computer Networks*, 40(6):711–726, Dec. 2002.

[6] Z. Li, B. Li, D. Jiang, and L. C. Lau. On achieving optimal throughput with network coding. In *Proceedings of IEEE INFOCOM*, 2005.

[7] C. Liang, M. Zhao, and Y. Liu. Optimal bandwidth sharing in multiswarm multiparty p2p video-conferencing systems. *IEEE/ACM Trans. Netw.*, 19(6):1704–1716, 2011.

[8] J. L. Ozer. *Video Compression for Flash, Apple Devices and HTML5*. Doceo Publishing, USA, 2011.

[9] M. Ponec, S. Sengupta, M. Chen, J. Li, and P. A. Chou. Optimizing multi-rate peer-to-peer video conferencing applications. *IEEE Transactions on Multimedia*, 2011.

[10] Y. Wu, C. Wu, B. Li, and F. C. Lau. vSkyConf: Cloud-assisted Multi-party Mobile Video Conferencing. Technical report, http://arxiv.org/abs/1303.6076.

[11] E. R. Ziegel. *Probability and Statistics for Engineering and the Sciences (8th Ed.), by Jay L. Devore*. American Statistical Association, eight edition, 2012.