Celerity: A Low-Delay Multi-Party Conferencing Solution

Xiangwen Chen, Minghua Chen, Baochun Li, Yao Zhao, Yunnan Wu and Jin Li

***Abstract*—In this paper, we revisit the problem of multi-party conferencing from a practical perspective, and to rethink the design space involved in this problem. We believe that an emphasis on low end-to-end delays between any two parties in the conference is a must, and the source sending rate in a session should adapt to bandwidth availability and congestion. We present *Celerity*, a multi-party conferencing solution specifically designed to achieve our objectives. It is entirely Peer-to-Peer (P2P), and as such eliminating the cost of maintaining centrally administered servers. It is designed to deliver video with low end-to-end delays, at quality levels commensurate with available network resources over arbitrary network topologies where *bottlenecks can be anywhere in the network*. This is in contrast to commonly assumed P2P scenarios where bandwidth bottlenecks reside only at the edge of the network. The highlight in our design is a distributed and adaptive rate control protocol, that can discover and adapt to arbitrary topologies and network conditions quickly, converging to efficient link rate allocations allowed by the underlying network. In accordance with adaptive link rate control, source video encoding rates are also dynamically controlled to optimize video quality in arbitrary and unpredictable network conditions. We have implemented *Celerity* in a prototype system, and demonstrate its superior performance over existing solutions in a local experimental testbed and over the Internet.**

***Index Terms*—Peer-to-Peer, Video Conferencing, Utility Maximization, Network Coding.**

I. Introduction

With the availability of front-facing cameras in high-end smartphone devices (such as the Samsung Galaxy S and the iPhone 4), notebook computers, and HDTVs, *multi-party* video conferencing, which involves more than two participants in a live conferencing session, has attracted a significant amount of interest from the industry. Skype, for example, has recently launched a monthly-paid service supporting multi-party video conferencing in its latest version (Skype 5) [1]. Skype video conferencing has also been recently supported in a range of new Skype-enabled televisions, such as the Panasonic VIERA series, so that full-screen high-definition video conferencing can be enjoyed in one's living room. Moreover, Google has supported multi-party video conferencing in its latest social network service *Google+*. Facebook cooperates with Skype to provide video conferencing service to its billions of users. We argue that these new conferencing solutions have the potential to provide an immersive human-to-human communication experience among remote participants. Such an argument has been corroborated by many industry leaders: Cisco predicts that video conferencing and tele-presence traffic will increase ten-fold between 2008-2013 [2].

While traffic flows in a live multi-party conferencing session are fundamentally represented by a multi-way communication process, today's design of multi-party video conferencing systems are engineered in practice by composing communication primitives (*e.g.,* transport protocols) over uni-directional feed-forward links, with primitive feedback mechanisms such as various forms of acknowledgments in TCP variants or custom UDP-based protocols. We believe that a high-quality protocol design must harness the full potential of the multi-way communication paradigm, and must guarantee the stringent requirements of low end-to-end delays, with the highest possible source coding rates that can be supported by dynamic network conditions over the Internet.

From an industry perspective, known designs of commercially available multi-party conferencing solutions are either largely server-based, e.g., Microsoft Office Communicator, or are separated into multiple point-to-point sessions (this approach is called Simulcast), e.g., Apple iChat. Server-based solutions are susceptible to central resource bottlenecks, and as such scalability becomes a main concern when multiple conferences are to be supported concurrently. In the Simulcast approach, each user splits its uplink bandwidth equally among all receivers and streams to each receiver separately. Though simple to implement, Simulcast suffers from poor quality of service. Specifically, peers with low upload capacity are forced to use a low video rate that degrades the overall experience of the other peers.

In the academic literature, there are recently several studies on peer-to-peer (P2P) video conferencing from a utility maximization perspective [3]–[8]. Among them, Li *et al.* [3] and Chen *et al.* [4] may be the most related ones to this work (we call their unified approach Mutualcast). They have tried to support content distribution and multi-party video conferencing in multicast sessions, by maximizing aggregate application-specific utility and the utilization of node uplink bandwidth in P2P networks. Specific depth-1 and depth-2 tree topologies have been constructed using tree packing, and rate control was performed in each of the tree-based one-to-many sessions.However, they only considered the limited

Manuscript received on February 14, 2011; revised on July 23, 2012.

Xiangwen Chen and Minghua Chen are with department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. Baochun Li is with Dept. of Electrical and Computer Engineering, The University of Toronto. Yao Zhao is with Google. Yunnan Wu is with Facebook Inc.. Jin Li is with Microsoft Research.

This work is partially supported by a China 973 Program (Project No. 2012CB315904), the General Research Fund (Project No. 411209, 411010, and 411011) and an Area of Excellence Grant (Project No. AoE/E-02/08), all established under the University Grant Committee of the Hong Kong SAR, China, as well as an Open Project of Shenzhen Key Lab of Cloud Computing Technology and Application and two gift grants from Microsoft and Cisco.

scenario where bandwidth bottlenecks reside at the edge of the network, while in practice bandwidth bottlenecks can easily reside in the core of the network [9], [10]. Further, all existing industrial and academic solutions, including Mutualcast, did not explicitly consider bounded delay in designs, and can lead to unsatisfied interactive conferencing experience.

A. Contribution

In this paper, we reconsider the design space in multi-party video conferencing, and present *Celerity*, a new multi-party conferencing solution specifically designed to maintain low end-to-end delays while maximizing source video rates in a session. *Celerity* has the following salient features:

- It operates in a pure P2P manner, and as such eliminating the cost of maintaining centrally administered servers.
- It can deliver video at quality levels commensurate with available network resources over *arbitrary network topologies*, while maintaining *bounded end-to-end delays*.
- It can automatically adapt to unpredictable network dynamics, such as cross traffic and abrupt link failures, allowing smooth conferencing experience.

Enabling the above features for multi-party conferencing is challenging. First, it requires a non-trivial formulation that allows systematic solution design over arbitrary network capacity constraints. In contrast, existing P2P system design works with performance guarantee commonly assume bandwidth bottlenecks reside at the edge of the network. Second, maximizing session rates subject to bounded delay is known to be NP-Complete and hard to solve approximately [11]. We take a practical approach that explores all 2-hop delay-bounded overlay trees with polynomial complexity. Third, detecting and reacting to network dynamics without *a priori* knowledge of the network conditions are non-trivial. We use both delay and loss as congestion measures and adapt the session rates with respect to both of them, allowing early detection and fast response to unpredictable network dynamics.

The highlight in our design is a distributed rate control protocol, that can discover and adapt to arbitrary topologies and network conditions quickly, converging to efficient link rate allocations allowed by the underlying network. In accordance with adaptive link rate control, source video encoding rates are also dynamically controlled to optimize video quality in arbitrary and unpredictable underlay network conditions.

II. Problem Formulation and Celerity Overview

One way to design a multi-party conferencing system is to formulate its fundamental design problem, explore powerful theoretical techniques to solve the problem, and use the obtained insights to guide practical system designs. In this way, we can also be clear about potential and limitation of the designs, allowing easy system tuning and further systematic improvements. Table I lists the key notations used in this paper.

TABLE I
Key notations.

Notation	Definition
$\mathcal{L}$	Set of all physical links
V	Set of conference participating nodes
E	Set of directed overlay links
C_l	Capacity of the physical link l
$a_{l,e}$	Whether overlay link e passes physical link l
$c_{m,e}$	Rate allocated to session m on overlay link e
$\boldsymbol{c}_m$	Overlay link rates of stream m, $\boldsymbol{c}_m = [c_{m,e}, e \in E]$
$\boldsymbol{y}$	Total overlay link traffics, $\boldsymbol{y} = \sum_{m=1}^{M} \boldsymbol{c}_m$
D	Delay bound
$R_m(\boldsymbol{c}_m, D)$	Session m's rate within the delay bound D
$q_l(z)$	Price function of violating link l's capacity constraint
p_l	Lagrange multiplier of link l's capacity constraint
$\mathcal{G}(\boldsymbol{c}, \boldsymbol{p})$	Lagrange function of variables $\boldsymbol{c}$ and $\boldsymbol{p}$

Note: we use bold symbols to denote vectors, e.g., $\boldsymbol{c} = [\boldsymbol{c}_1^T, \ldots, \boldsymbol{c}_M^T]^T$.

A. Settings

Consider a network modeled as a directed graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of all physical nodes, including conference participating nodes and other intermediate nodes such as routers, and $\mathcal{L}$ is the set of all physical links. Each link $l \in \mathcal{L}$ has a nonnegative capacity C_l and a nonnegative propagation delay d_l.

Consider a multi-party conferencing system over G. We use $V \subseteq \mathcal{N}$ to denote the set of all conference participating nodes. Every node in V is a source and at the same time a receiver for every other nodes. Thus there are totally $M \triangleq |V|$ sessions of (audio/video) streams. Each stream is generated at a source node, say v, and needs to be delivered to all the rest nodes in $V - \{v\}$, by using overlay links between any two nodes in V.

An overlay link (u, v) means u can send data to v by setting up a TCP/UDP connection, along an underlay path from u to v pre-assigned by routing protocols. Let E be the set of all directed overlay links. For all $e \in E$ and $l \in \mathcal{L}$, we define

$$a_{l,e} = \begin{cases} 1, & \text{if overlay link } e \text{ passes physical link } l; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The physical link capacity constraints are then expressed as

$$\boldsymbol{a}_l^T \boldsymbol{y} = \sum_{e \in E} a_{l,e} \sum_{m=1}^{M} c_{m,e} \leq C_l, \quad \forall l \in \mathcal{L},$$

where $c_{m,e}$ denotes the rate allocated to session m on overlay link e and $\boldsymbol{a}_l^T \boldsymbol{y}$ describes the total overlay traffic passing through physical link l.

Remark: In our model, the capacity bottleneck can be anywhere in the network, not necessarily at the edges. This is in contrast to a common assumption made in previous P2P works that the uplinks/downlinks of participating nodes are the only capacity bottleneck.

B. Problem Formulation

In a multi-party conferencing system, each session source broadcasts its stream to all receivers over a complete overlay graph on which every link e has a rate $c_{m,e}$ and a delay $\sum_{l \in \mathcal{L}} a_{l,e} d_l$. For smooth conferencing experience, the total delay of delivering a packet from the source to any receiver, traversing one or multiple overlay links, cannot exceed a delay bound D.

A fundamental design problem is to maximize the overall conferencing experience, by properly allocating the overlay link rates to the streams subject to physical link capacity constraints. We formulate the problem as a network utility maximization problem:

$$\textbf{MP}: \quad \max_{c\geq 0} \quad \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m, D)\right) \tag{2}$$

$$\text{s.t.} \qquad \boldsymbol{a}_l^T \boldsymbol{y} \leq C_l, \quad \forall l \in \mathcal{L}. \tag{3}$$

The optimization variables are $\boldsymbol{c}$ and the constraints in (3) are the physical link capacity constraints.

$R_m(\boldsymbol{c}_m, D)$ denotes session m's rate that we obtain by using resource $\boldsymbol{c}_m$ *within the delay bound* D, and is a concave function of $\boldsymbol{c}_m$ as we will show in Corollary 1 in the next section.

The objective is to maximize the aggregate system utility. $U_m(R_m)$ is an increasing and strictly concave function that maps the stream rate to an application-specific utility. For example, a commonly used video quality measure Peak Signal-to-Noise Ratio (PSNR) can be modeled by using a logarithmic function as the utility [4] [1]. With these settings and observations, $U_m(R_m)$ is concave in $\boldsymbol{c}$ and the problem **MP** is a concave optimization problem.

Remarks: (i) The formulation of **MP** is an overlay link based formulation in which the number of variables per session is $|E|$ and thus at most $|V|^2$. One can write an equivalent tree-based formulation for **MP** but the number of variables per session will be *exponential* in $|E|$ and $|V|$. (ii) Existing solutions, such as Simulcast and Mutualcast, can be thought as algorithms solving special cases of the problem **MP**. For example, Simulcast can be thought as solving the problem **MP** by using only the 1-hop tree to broadcast content within a session. Mutualcast can be thought as solving a special case of the problem **MP** (with the uplinks of participating nodes being the only capacity bottleneck) by packing certain depth-1 and depth-2 trees within a session.

C. Celerity Overview

Celerity builds upon two main modules to maximize the system utility: (1) a *delay-bounded video delivery* module to distribute video at high rate given overlay link rates (i.e., how to compute and achieve $R_m(\boldsymbol{c}_m, D)$); (2) a *link rate control* module to determine $\boldsymbol{c}_m$.

Video delivery under known link constraints: This problem is similar to the classic multicast problem, and packing spanning (or Steiner) trees at the multicast source is a popular solution. However, the unique "delay-bounded" requirement in multi-party conferencing makes the problem more challenging. We introduce a delay-bounded tree packing algorithm to tackle this problem (detailed in Section III).

Link rate control: In principle, one can first infer the network constraints and then solve the problem **MP** centrally.

[1]Using logarithmic functions also guarantees (weighted) proportional fairness among sessions and thus no session will starve at the optimal solution [12].

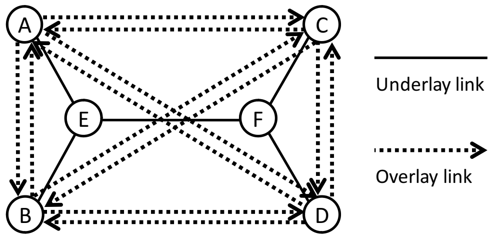


Fig. 1. An illustrating example of 4-party (A, B, C, and D) conferencing over a dumbbell underlay topology. E and F are two routers. Solid lines represent underlay physical links. To make the graph easy to read, we use one solid line to represent a pair of directed physical links. Dash lines represent overlay links.

However, directly inferring the constraints potentially requires knowing the entire network topology and is highly challenging. In *Celerity*, we resort to design adaptive and iterative algorithms for solving the problem **MP** in a distributed manner, without *a priori* knowledge of the network conditions.

We explain at a high level how *Celerity* works in a 4-party conferencing example in Fig. 1. We focus on session A, in which source A distributes its stream to receivers B, C, and D, by packing delay-bounded trees over a complete overlay graph shown in the figure. We focus on source A and one overlay link (B, C), which represents a UDP connection over an underlay path B to E to F to C. Other overlay links and other sessions are similar.

We first describe the control plane operations. For the overlay link (B, C), the head node B works with the tail node C to *periodically* adjust the session rate $c_{A,B\to C}$ according to *Celerity*'s link rate control algorithm. Such adjustment utilizes control-plane information that source A piggybacks with data packets, and loss and delay statistics experienced by packets traveling from B to C. We show such local adjustments at every overlay link result in globally optimal session rates.

The head node B also *periodically* reports to source A the session rate $c_{A,B\to C}$ and the end-to-end delay from B to C. Based on these reports from all overlay links, source A *periodically* packs delay-bounded trees using *Celerity*'s tree-packing algorithm, calculates necessary control-plane information, and delivers data and the control-plane information along the trees.

The data plane operations are simple. *Celerity* uses delay-bounded trees to distribute data in a session. Nodes on every tree forward packets from its upstream parent to its downstream children, following the "next-children" tree-routing information embedded in the packet header. *Celerity*'s tree-packing algorithm guarantees that (i) packets arrive at all receivers within the delay bound, and (ii) the total rate of a session m passing through an overlay link e does not exceed the allocated rate $c_{m,e}$.

In the following two sections, we present the designs of the two main modules in *Celerity*. Due to the space limitation, we leave the practical implementation of *Celerity* in practical peers in our technical report [13].

III. Packing Delay-bounded Trees

Given the link rate vector $\boldsymbol{c}_m$ and delay for every overlay link e (i.e.,$\sum_{l\in\mathcal{L}} a_{l,e} d_l$), achieving the maximum broad-

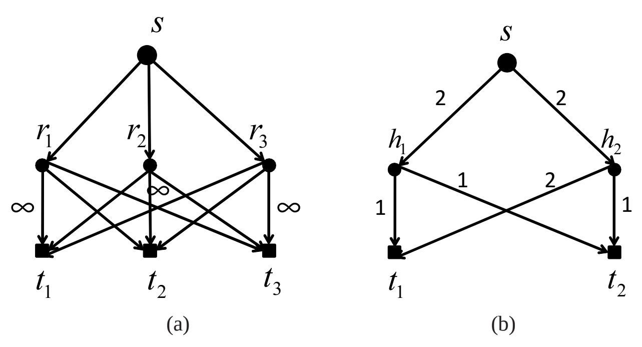


Fig. 2. (a): Illustration of the directed acyclic sub-graph over which we pack delay-bounded 2-hop trees. (b): Critical cut example. Source s and its two receivers t_1, t_2 are connected over a directed graph. The number associated with a link represents its link capacity.

cast/multicast stream rate under a delay bound D is a challenging problem. A general way to explore the broadcast/multicast rate under delay bounds is to pack delay-bounded Steiner trees. However, such problem is NP-hard [14]. Moreover, the number of delay-bounded Steiner trees to consider is in general exponential in the network size.

In this paper, we pack 2-hop delay-bounded trees in an overlay graph of session m, denoted by $\mathcal{D}_m$, to achieve a good stream rate under a delay bound. Note by graph theory notations, a 2-hop tree has a depth at most 2. Packing 2-hop trees is easy to implement. It also explores all overlay links between source and receiver and between receivers, thus trying to utilize resource efficiently. In fact, it is shown in [3], [4] that packing 2-hop multicast trees suffices to achieve the maximum multicast rate for certain P2P topologies. We elaborate our tree-packing scheme in the following.

We first define the overlay graph $\mathcal{D}_m$. Graph $\mathcal{D}_m$ is a directed acyclic graph with two layers; one example of such graph is illustrated in Fig. 2a. In this example, consider a session with a source s, three receivers $1, 2, 3$. For each receiver i, we draw two nodes, r_i and t_i, in the graph $\mathcal{D}_m$; t_i models the receiving functionality of node i and r_i models the relaying functionality of node i.

Suppose that the prescribed link bit rates are given by the vector $\boldsymbol{c}_m$, with the capacity for link e being $c_{m,e}$. Then in $\mathcal{D}_m$, the link from s to r_i has capacity $c_{m,s\to r_i}$, the link from r_i to t_j (with $i \neq j$) has capacity $c_{m,r_i\to t_j}$, and the link from r_i to t_i has infinite capacity. If the propagation delay of an edge e exceeds the delay bound, we do not include it in the graph. If the propagation delay of a two-hop path $s \to r_i \to t_j$ exceeds the delay bound, we omit the edge from r_i to t_j from the graph. As a result, every path from s to any receiver t_i in the graph has a path propagation delay within the delay bound.

Over such 2-layer sub-graph $\mathcal{D}_m$, we pack 2-hop trees connecting the source and every receiver using the greedy algorithm proposed in [15]. Below we simply describe the algorithm and more details can be found in [15].

Assuming all edges have unit-capacity and allowing multiple edges for each ordered node pair. The algorithm packs unit-capacity trees one by one. Each unit-capacity tree is constructed by greedily constructing a tree edge by edge starting from the source and augmenting towards all receivers. It is similar to the greedy tree-packing algorithm based on Prim's algorithm. The distinction lies in the rule of selecting the edge among all potential edges. The edge whose removal leads to least reduction in the multicast capacity of the residual graph is chosen in the greedy algorithm.

The above greedy algorithms is very simple to implement and its practical implementation details are further discussed in the technical report [13].

Utilizing the special structure of the graph $\mathcal{D}_m$, we obtain performance guarantee of the algorithm as follows.

Theorem 1: The tree-packing algorithm in [15] achieves the minimum of the min-cuts separating the source and receivers in $\mathcal{D}_m$ and is expressed as

$$R_m(\boldsymbol{c}_m, D) = \min_j \sum_i \min\left\{c_{m,s\to r_i}, c_{m,r_i\to t_j}\right\}. \tag{4}$$

Furthermore, the algorithm has a running time of $O(|V||E|^2)$.

Hence, our tree-packing algorithm achieves the maximum delay-bounded multicast rate over the 2-layer sub-graph $\mathcal{D}_m$. The achieved rate $R_m(\boldsymbol{c}_m, D)$ is a concave function of $\boldsymbol{c}_m$ as summarized below.

Corollary 1: The delay-bounded multicast rate $R_m(\boldsymbol{c}_m, D)$ obtained by our tree-packing algorithm is a concave function of the overlay link rates $\boldsymbol{c}_m$.

IV. Overlay Link Rate Control

A. Considering Both Delay and Loss

We revise original formulation to design our link rate control algorithm with both queuing delay and loss rate taken into account. Adapting link rates to both delay and loss allows early detection and fast response to network dynamics.

Consider the following formulation with a penalty term added into the objective function of the problem **MP**:

$$\begin{aligned}\textbf{MP}-\textbf{EQ}: \max_{\boldsymbol{c}\geq 0} \quad \mathcal{U}(\boldsymbol{c}) \overset{\Delta}{=} \quad & \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m, D)\right) - \\ & \sum_{l\in\mathcal{L}} \int_0^{\boldsymbol{a}_l^T \boldsymbol{y}} q_l(z)\,dz, \end{aligned} \tag{5}$$

$$\text{s.t.} \qquad \boldsymbol{a}_l^T \boldsymbol{y} \leq C_l, \quad \forall l \in \mathcal{L}, \tag{6}$$

where $\int_0^{\boldsymbol{a}_l^T \boldsymbol{y}} q_l(z)\,dz$ is the penalty associated with violating the capacity constraint of physical link $l \in \mathcal{L}$, and we choose the price function to be

$$q_l(z) \overset{\Delta}{=} \frac{(z - C_l)^+}{z}, \tag{7}$$

where $(a)^+ = \max\{a, 0\}$. If all the constraints are satisfied, then the second term in (5) vanishes; if instead some constraints are violated, then we charge some penalty for doing so.

Remark: (i) The problem **MP-EQ** is equivalent to the original problem **MP**. Because any feasible solution $\boldsymbol{c}$ of these two problems must satisfy $\boldsymbol{a}_l^T \boldsymbol{y} \leq C_l$, and consequently the penalty term in the problem **MP-EQ** vanishes. (ii) It can be verified that $-\sum_{l\in\mathcal{L}} \int_0^{\boldsymbol{a}_l^T \boldsymbol{y}} q_l(z)\,dz$ is a concave function in $\boldsymbol{c}$; hence, $\mathcal{U}(\boldsymbol{c})$ is a linear combination of concave functions and is concave. However, because $R_m(\boldsymbol{c}_m, D)$ is the minimum min-cut of the overlay graph $\mathcal{D}_m$ with link rates being $\boldsymbol{c}_m$, $\mathcal{U}(\boldsymbol{c})$ is not a differentiable function [16].

We apply Lagrange dual approach to design distributed algorithms for the problem **MP-EQ**. The advantage of adopting distributed rate control algorithms in our system is that it allows robust adaption upon unpredictable network dynamics.

The Lagrange function of the problem is given by:

$$\begin{aligned}\mathcal{G}(\boldsymbol{c},\boldsymbol{p}) \quad \triangleq \quad & \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m, D)\right) - \sum_{l\in\mathcal{L}} \int_0^{\boldsymbol{a}_l^T\boldsymbol{y}} q_l(z)\,dz - \\ & \sum_{l\in\mathcal{L}} p_l\left(\boldsymbol{a}_l^T\boldsymbol{y} - C_l\right), \end{aligned} \tag{8}$$

where $p_l \geq 0$ is the Lagrange multiplier associated with the capacity constraint in (6) of physical link l. p_l can be interpreted as the price of using link l. Since the problem **MP-EQ** is a concave optimization problem with linear constraints, strong duality holds and there is no duality gap. Any optimal solution of the problem and one of its corresponding Lagrangian multiplier is a saddle point of $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$ and vice versa. Thus to solve the problem **MP-EQ**, it suffices to design algorithms to pursue saddle points of $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$.

B. A Loss-Delay Based Primal-Subgradient-Dual Algorithm

There are two issues to address in designing algorithms for pursuing saddle points of $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$. First, the utility function $\mathcal{U}(\boldsymbol{c})$ (and consequently $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$) is not everywhere differentiable. Second, $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$ is not strictly concave in $\boldsymbol{c}$, thus distributed algorithms may not converge to the desired saddle points under multi-party conferencing settings [4].

To address the first concern, we use subgradient in algorithm design. To address the second concern, we provide a convergence result for our designed algorithm.

To proceed, we first compute subgradients of $\mathcal{U}(\boldsymbol{c})$. The proposition below presents a useful observation.

Proposition 1: A subgradient of $\mathcal{U}(\boldsymbol{c})$ with respect to $c_{m,e}$ for any $e \in E$ and $m = 1, \ldots M$ is given by

$$U'_m(R_m)\frac{\partial R_m}{\partial c_{m,e}} - \sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{y} - C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{y}}$$

where $\frac{\partial R_m}{\partial c_{m,e}}$ is a subgradient of $R_m(\boldsymbol{c}_m, D)$ with respect to $c_{m,e}$.

Motivated by the pioneering work of Arrow, Hurwicz, and Uzawa [17] and the followup works [18] [19], we propose to use the following *primal-subgradient-dual* algorithm to pursue the saddle point of $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$:$\forall e\epsilon E$, $m = 1, ...M$, $\forall l\epsilon\mathcal{L}$,

Primal-Subgradient-Dual Link Rate Control Algorithm:

$$\begin{aligned} c_{m,e}^{(k+1)} \quad = \quad & c_{m,e}^{(k)} + \alpha\Bigg[U'_m\left(R_m^{(k)}\right)\frac{\partial R_m^{(k)}}{\partial c_{m,e}} \\ & \sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{y}^{(k)} - C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{y}^{(k)}} - \sum_{l\in\mathcal{L}} a_{l,e}p_l^{(k)}\Bigg]^+_{c_{m,e}^{(k)}} \end{aligned} \tag{9}$$

$$p_l^{(k+1)} \quad = \quad p_l^{(k)} + \frac{1}{C_l}\left[\boldsymbol{a}_l^T\boldsymbol{y}^{(k)} - C_l\right]^+_{p_l^{(k)}} \tag{10}$$

where $\alpha > 0$ represents a constant the step size for all the iterations, and function

$$[b]_a^+ = \begin{cases}\max(0,b), & a \leq 0;\\ b, & a > 0.\end{cases}$$

We have the following observations for the control algorithm in (9)-(10):

- It is known that $\sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{y}-C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{y}}$ can be interpreted as the packet loss rate observed at overlay link e [20]. The intuitive explanation is as follows. The term $(\boldsymbol{a}_l^T\boldsymbol{y} - C_l)^+$ is the excess traffic rate offered to physical link l; thus $\frac{(\boldsymbol{a}_l^T\boldsymbol{y}-C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{y}}$ models the fraction of traffic that is dropped at l. Assuming the packet loss rates are additive (which is a reasonable assumption for low packet loss rates), the total packet loss rates seen by the overlay link e is given by $\sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{y}-C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{y}}$.
- It is also known that p_l updating according to (10) can be interpreted as queuing delay at physical link l [21]. Intuitively, if the incoming rate $\boldsymbol{a}_l^T\boldsymbol{y} > C_l$ at l, then it introduces an additional queuing delay of $\frac{\boldsymbol{a}_l^T\boldsymbol{y}-C_l}{C_l}$ for l. If otherwise the term $\boldsymbol{a}_l^T\boldsymbol{y} \leq C_l$, then the present queueing delay is reduced by an amount of $\frac{C_l-\boldsymbol{a}_l^T\boldsymbol{y}}{C_l}$ unless hitting zero. The total queuing delay observed by the overlay link e is then given by the sum $\sum_{l\in\mathcal{L}} a_{l,e}p_l$.
- It turns out that the utility function, the subgradients, packet loss rate and queuing delay are sufficient statistics to update $c_{m,e}$ independently of the updates of other link rates. This way, we can solve the problem **MP-EQ** without knowing the physical network topology and physical link capacities.

The algorithm in (9)-(10) is similar to the standard primal-dual algorithm, but since $\mathcal{U}(\boldsymbol{c})$ is not differentiable everywhere, we use subgradient instead of gradient in updating the overlay link rates $\boldsymbol{c}$.

Establishing convergence of subgradient algorithms for saddle-point optimization is in general challenging [18]. We explore convergence properties for our primal-subgradient-dual algorithm in the following theorem.

Theorem 2: Let $(\boldsymbol{c}^*, \boldsymbol{p}^*)$ be a saddle point of $\mathcal{G}(\boldsymbol{c},\boldsymbol{p})$, and $\overline{\mathcal{G}}^{(k)}$ be the average function value obtained by the algorithm in (9)-(10) after k iterations:

$$\overline{\mathcal{G}}^{(k)} \triangleq \frac{1}{k}\sum_{i=0}^{k-1}\mathcal{G}\left(\boldsymbol{c}^{(k)}, \boldsymbol{p}^{(k)}\right).$$

Suppose $\left|U'_m(R_m(\boldsymbol{c}_m))\right| \leq \overline{U}$, $\forall m = 1, \ldots, M$, where $\overline{U}$ is a constant, then we have

$$-\frac{B_1}{2\alpha k} - \frac{\Delta^2}{2}\alpha \leq \quad \overline{\mathcal{G}}^{(k)} - \mathcal{G}(\boldsymbol{c}^*, \boldsymbol{p}^*) \quad \leq \frac{B_2}{2k} + \frac{\Delta^2}{2}\max_{l\in\mathcal{L}} C_l^{-1},$$

where $B_1 = \left\|\boldsymbol{c}^{(0)} - \boldsymbol{c}^*\right\|^2$ and $B_2 = \left[\boldsymbol{p}^{(0)} - \boldsymbol{p}^*\right]^T \operatorname{diag}(C_l, l \in \mathcal{L})\left[\boldsymbol{p}^{(0)} - \boldsymbol{p}^*\right]$ are two positive distances depending on $(\boldsymbol{c}^{(0)}, \boldsymbol{p}^{(0)})$, and Δ is a positive constant depending on $\overline{U}$ and $(\boldsymbol{c}^{(0)}, \boldsymbol{p}^{(0)})$.

Remarks: (i) The results bound the time-average Lagrange function value obtained by the algorithm to the optimal in terms of distances of the initial iterates $(\boldsymbol{c}^{(0)}, \boldsymbol{p}^{(0)})$ to a saddle point. In particular, the averaged function values $\overline{\mathcal{G}}^{(k)}$ converge to the saddle point value $\mathcal{G}(\boldsymbol{c}^*, \boldsymbol{p}^*)$ within a gap of $\max\left(\alpha, \max_{l\in\mathcal{L}} C_l^{-1}\right)\frac{\Delta^2}{2}$, at a rate of $1/k$. (ii) The requirement of the utility function is easy to satisfied; one example is $U_m(z) = \log(z+\epsilon)$ with $\epsilon > 0$. (iii) Our results generalize the one in [18] in the sense that the one in [18] only applies to the case of uniform step size, while we allow different p_l to update with different step size $\frac{1}{C_l}$, which is critical for p_l to be interpreted as queuing delay and thus practically measurable. Our results also have less stringent requirement on the utility function than the one in [18]. (iv) Although the results may not warranty convergence in the strict sense, our experiments over LAN testbed and on the Internet in Section V show the algorithm quickly stabilizes around optimal operating points.

C. Computing Subgradients of $R_m(\boldsymbol{c}_m, D)$

A key to implementing the Primal-Subgradient-Dual algorithm is to obtain subgradients of $R_m(\boldsymbol{c}_m, D)$. We first present some preliminaries on subgradients, as well as concepts for computing subgradients for $R_m(\boldsymbol{c}_m, D)$.

Definition 1: Given a convex function f, a vector ξ is said to be a subgradient of f at $x \in \mathbf{dom} f$ if

$$f(x') \geq f(x) + \xi^T(x'-x), \forall x' \in \mathbf{dom} f,$$

where $\mathbf{dom} f = \{x \in \mathbf{R}^n \| f(x)| < \infty\}$ represents the domain of the function f.

For a concave function f, $-f$ is a convex function. A vector ξ is said to be a subgradient of f at x if $-\xi$ is a subgradient of $-f$.

Next, we define the notion of a *critical cut*. For session m, let its source be s_m and receiver set be $V_m \subset V - \{s_m\}$. A partition of the vertex set, $V = Z \cup \overline{Z}$ with $s_m \in Z$ and $t \in \overline{Z}$ for some $t \in V_m$, determines an s_m-t-cut. Define

$$\delta(Z) \triangleq \left\{(i,j) \in E | i \in Z, j \in \overline{Z}\right\}$$

be the set of overlay links originating from nodes in set Z and going into nodes in set $\overline{Z}$. Define the capacity of cut $(Z, \overline{Z})$ as the sum capacity of the links in $\delta(Z)$:

$$\rho(Z) \triangleq \sum_{e\in\delta(Z)} c_{m,e}.$$

Definition 2: For session m, a cut $(Z, \overline{Z})$ is an s_m-V_m critical cut if it separates s_m and any of its receivers and $\rho(Z) = R_m(\boldsymbol{c}_m, D)$.

We show an example to illustrate the concept of critical cut. In Fig. 2b, s is a source, and t_1, t_2 are its two receivers. The minimum of the min-cuts among the receivers is 2. For the cut $(\{s, h_1, h_2, t_1\}, \{t_2\})$, its $\delta(\{s, h_1, h_2, t_1\})$ contains links (h_1, t_2) and (h_2, t_2), each having capacity one. Thus the cut $(\{s, h_1, h_2, t_1\}, \{t_2\})$ has a capacity of 2 and it is an $s-(t_1, t_2)$ critical cut.

With necessary preliminaries, we turn to compute subgradients of $R_m(\boldsymbol{c}_m, D)$. Since $R_m(\boldsymbol{c}_m, D)$ is the minimum min-cut of s_m and its receivers over the overlay graph $\mathcal{D}_m$, it is known that one of its subgradients can be computed in the following way [16].

- Find an s_m-V_m critical cut for session m, denote it as $(Z, \overline{Z})$. Note there can be multiple s_m-V_m critical cuts in graph $\mathcal{D}_m$, and it is sufficient to find any one of them.
- A subgradient of $R_m(\boldsymbol{c}_m, D)$ with respect to $c_{m,e}$ is given by

$$\frac{\partial R_m(\boldsymbol{c}_m, D)}{\partial c_{m,e}} = \begin{cases} 1, & \text{if } e \in \delta(Z); \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

In our system, these subgradients are computed by the source of each session, after collecting the overlay-link rates from each receiver in the session. More implementation details are in the technical report [13].

V. Experiments

Using the asynchronous networking paradigm supported by the asynchronous I/O library (called `asio`) in the `Boost` C++ library, we have implemented a prototype of *Celerity*, our proposed multi-party conferencing system, with about 17,000 lines of code in C++.

Due to the space limitation, details about practical implementation are discussed in the technical report [13].

For the performance evaluation of *Celerity*, we evaluate our prototype *Celerity* system over a LAN testbed as well as over the Internet. The LAN experiments allow us to (i) stress-test *Celerity* under various network conditions; (ii) see whether *Celerity* meets the design goal – delivering high delay-bounded throughput and adapting to dynamics in the network; (iii) demonstrate the fundamental performance gains over existing solutions, thus justifying our theory-inspired design.

The Internet experiments allow us to further access *Celerity*'s superior performance over existing solutions in practice.

A. LAN Testbed Experiments

We evaluate *Celerity* over a LAN testbed illustrated in Fig. 3, where four PC nodes (A, B, C, D) are connected over a LAN dumbbell topology. The dumbbell topology represents a popular scenario of multi-party conferencing between branch offices. It is also a "tough" topology – existing approaches, such as Simulcast and Mutualcast, fail to efficiently utilize the bottleneck bandwidth and optimize system performance.

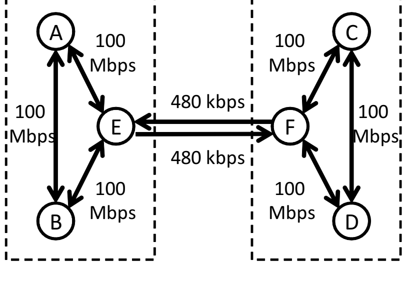


Fig. 3. The "tough" dumbbell topology of the experimental testbed. Two conference participating nodes A and B are in one "office" and another twos nodes C and D are in a different "office". The two "offices" are connected by directed links between gateway nodes E and F, each link having a capacity of 480 kbps. Link propagation delays are negligible.

In our experiments, all four nodes run *Celerity*. We run a four-party conference for 1000 seconds and evaluate the system performance. In order to evaluate *Celerity*'s performance in the presence of network dynamics, we reduce cross traffic and introduce link failures during the experiment. In particular, we introduce an 80kpbs cross-traffic from node E to node F between the 300th second and the 500th second, reducing the available bandwidth between E and F from 480 kbps to 400 kbps. Further, starting from the 700th second, we disconnect the physical link between A and E; this corresponds to a practical situation where node A suddenly cannot directly communicate with nodes outside the "office" due to middleware or configuration errors at the gateway E.

Figs. 4a-4d show the sending rate of each session (one session originates from one node to all other three nodes). For comparison, we also show the maximum achievable rates by Simulcast and Mutualcast, as well as the optimal sending rate of each session calculated by solving the problem in (2)-(3) using a central solver. Fig. 4e shows the utility obtained by *Celerity* and its comparison to the optimal. Fig. 4f shows the average end-to-end delay and packet loss rate of session A. Delay and loss performance of other sessions are similar to those of session A.

In this experiment, *Celerity* goes through three different experiment stages: absence of network dynamics, cross traffic, and link failure. Due to the space limitation, we only explain the results according to the first stage. For the other two stages, please refer to the technical report [13].

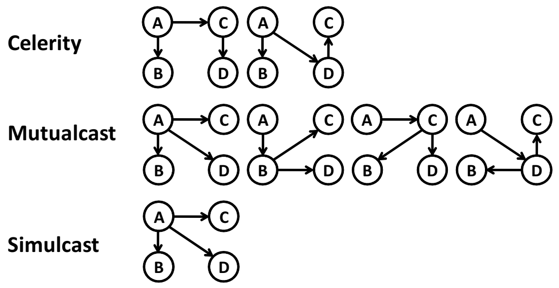


Fig. 7. Session A's trees used by *Celerity* (upon convergence), Mutualcast and Simulcast in the dumbbell topology, in the absence of network dynamics.

1) Absence of Network Dynamics: We first look at the first 300 seconds when there is no cross traffic or link failure. In this time period, the experimental settings are symmetric for all participating peers; thus the optimal sending rate for each session is 240 kbps.

As seen in Figs. 4a-4d, *Celerity* demonstrates fast convergence: the sending rate of each session quickly ramps up to 95% to the optimal within 50 seconds. Fig. 4e shows that *Celerity* quickly achieves a close-to-optimal utility. These observations indicate any other solution can at most outperform *Celerity* by a small margin.

As a comparison, we also plot the theoretical maximum rates achievable by Simulcast and Mutualcast in Figs. 4a-4d. We observe that within 20 seconds, our system already outperforms the maximum rates of Simulcast and Mutualcast.

Upon convergence, *Celerity* achieves sending rates that nearly double the maximum rate achievable by Simulcast and Mutualcast. This significant gain is due to that *Celerity* can utilize the bottleneck resource more efficiently, as explained below.

In Fig. 7, we show the trees for session A that are used by *Celerity*, Mutualcast and Simulcast in the dumbbell topology. As seen, Simulcast and Mutualcast only explore 2-hop trees satisfying certain structure, limiting their capability of utilizing network capacity efficiently. In particular, their trees consumes the bottleneck link resource twice, thus to deliver one-bit of information it consumes two-bit of bottleneck link capacity. For instance, the tree used by Simulcast has two branches $A \rightarrow C$ and $A \rightarrow D$ passing through the bottleneck links between E and F, consuming twice the critical resource. Consequently, the maximum achievable rates of Simulcast and Mutualcast are all 120 kbps. In contrast, *Celerity* explores all 2-hop delay-bounded trees, and upon convergence utilizes the trees that only consume bottleneck link bandwidth once, achieving rates that are close to the optimal of 240 kbps.

Fig. 4f shows the average end-to-end delay and packet loss rate of session A. As seen, the packet loss rate and delay are high initially, but decreases and stabilizes to small values afterwards. The initial high loss rate is because *Celerity* increases the sending rates aggressively during the conference initialization stage, in order to bootstrap the conference and explore network resource limits. *Celerity* quickly learns and adapts to the network topology, ending up with using cost-effective trees to deliver data. After the initialization stage, *Celerity* adapts and converges gradually, avoiding unnecessary performance fluctuation that deteriorates user experience. By adapting to both delay and loss, we achieve low loss rate upon convergence as compared to the case when only loss is taken into account [22].

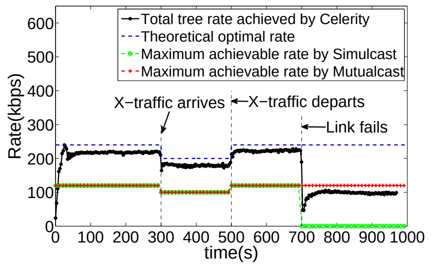
B. Peer Dynamics Experiments

In order to evaluate the *Celerity* performance in peer dynamics scenario, we conduct another experiment over the same LAN testbed in Fig. 3. We first run a three-party conference among node A, B, and C, at the 120th second, a node D joins the conferencing session and leaves at the 300th second, the entire conferencing session lasts for 480 seconds.

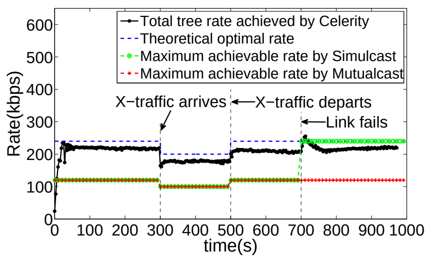
Fig. 5a shows the sending rate of each session as well as the optimal sending rate of each session, Fig. 5b-5c show the average end-to-end delay and packet loss rate of session A and C. Delay and loss performance of session B are similar to those of session A.

As seen in Fig. 5a, when node D joins the conferencing session at the 120th second, the sending rates of session A, B and C first drop immediately, then quickly adapt to close to the optimal value again. This is because when node D joins, the initial allocated rates for each session in the overlay links from other nodes to node D are very low, when node A, B and C pack trees respectively according to the allocated rates to deliver their data to the receivers including node D, the achieved sending rates are low. Then, *Celerity* detects the change of underlay topology, updates the allocated rates and quickly converges to the new close to optimal operating point. When node D leaves, we also observe that *Celerity* quickly adapts to the peer dynamic.

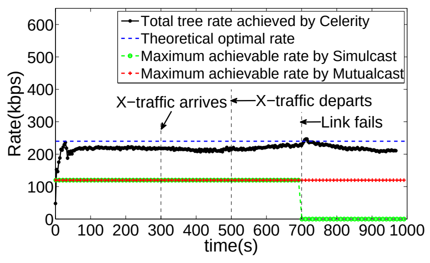
Celerity's excellent performance adapting to peer dynamics is expected from its design. We involve both loss and queuing



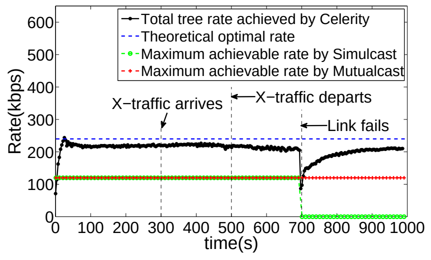
(a) Rate Performance of Node A



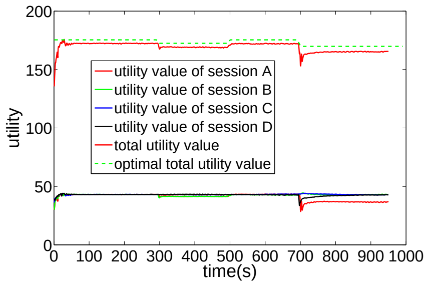
(b) Rate Performance of Node B



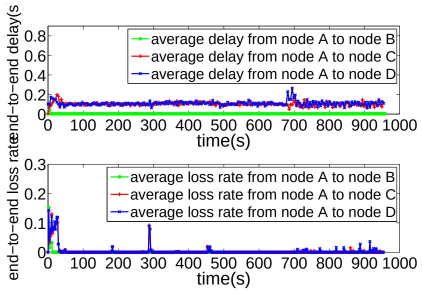
(c) Rate Performance of Node C



(d) Rate Performance of Node D

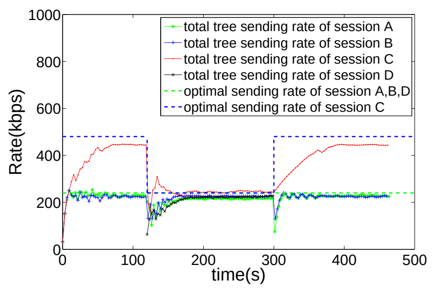


(e) Total utility of all sessions

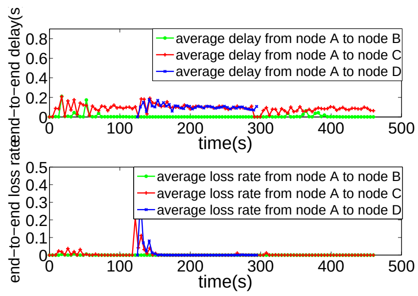


(f) Average end-to-end delay and loss rate from node A to other nodes

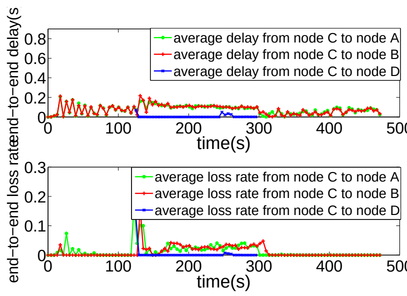
Fig. 4. Performance of *Celerity* in the LAN Testbed Experiments. (a)-(d): Sending rates and receiving rates of individual sessions. (e): Utility value achieved compared to the optimum. (f): End-to-end delay and loss rate of session *A*.



(a) Rate Performance of all Nodes



(b) Average end-to-end delay and loss rate from node A to other nodes



(c) Average end-to-end delay and loss rate from node C to other nodes

Fig. 5. Performance of *Celerity* in the Peer Dynamics Experiments. (a)-(f): Sending rates of all sessions. (b)-(c): End-to-end delay and loss rate of session *A* and *C*.

delay in our design, when peers join and leave, loss and queuing delay reflect such events well, thus allowing *Celerity* to adapt rapidly to the peer dynamics. For instance, in this experiment when node *D* joins the conferencing session, we observe a spike in session *A*'s end-to-end delay and packet loss rate in Fig. 5b.

In Fig. 5a another important observation is that as compared to the conference initialization stage, the convergence speed of node *C* after node *D* leaves the conferencing session is slow. This is because during the conference initialization stage, *Celerity* uses a method called ”quick start” described in the technical report [13] to quickly ramp up the rates of all sessions, while after the initialization stage, such method is not used in order to avoid unnecessary performance fluctuation. It is of great interest to design source rate control mechanisms to achieve quick convergence in peer dynamics scenario without incurring system fluctuation.

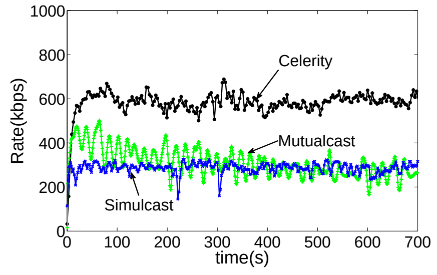
C. Internet Experiments

Beside the prototype *Celerity* system, we also implement two prototype systems of Simulcast and Mutualcast, respectively. Both *Celerity* and Mutualcast use the same log utility functions in their rate control modules. We evaluate the performance of these systems in a four-party conferencing scenario over the Internet.

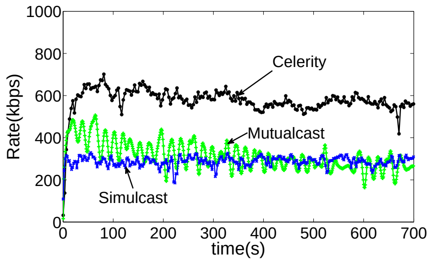
We use four PC nodes that spread two continents and tree countries to form the conferencing scenario. Two of the PC nodes are in Hong Kong, one is in Redmond, Washington, US, and the last one is in Toronto, Canada. This setting represents a common global multi-party conferencing scenario.

We run multiple 15-minute four-party conferences using the prototype systems, in a one-by-one and interleaving manner. We select one representative run for each system, and summarize their performance in Fig. 6.

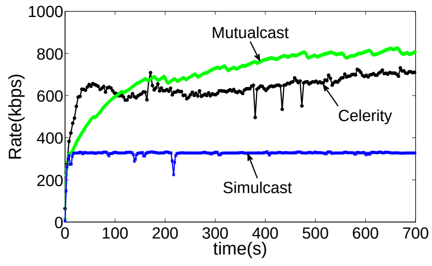
Figs. 6a-6d show the rate performance of each session.



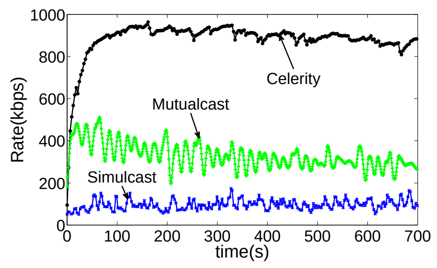
(a) Sending Rate of Node A (Hong Kong)



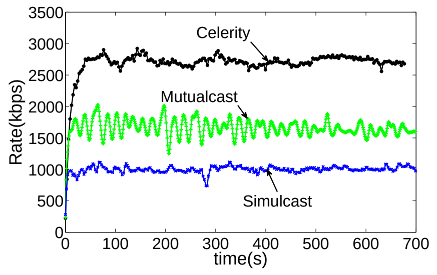
(b) Sending Rate of Node B (Hong Kong)



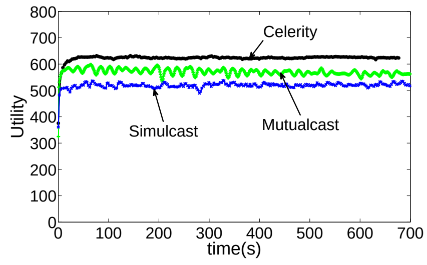
(c) Sending Rate of Node C (Redmond)



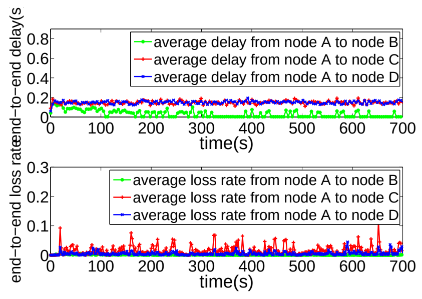
(d) Sending Rate of Node D (Toronto)



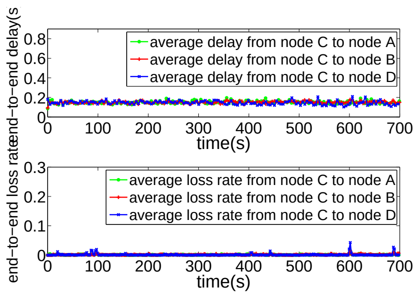
(e) Total sending rate of all sessions



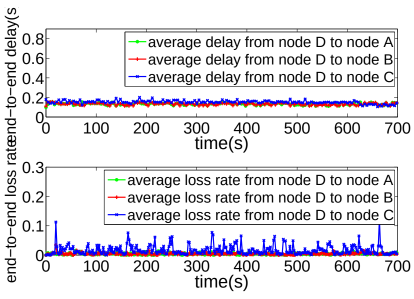
(f) Total utility of all sessions



(g) Average end-to-end delay and loss rate from node A to other nodes



(h) Average end-to-end delay and loss rate from node C to other nodes



(i) Average end-to-end delay and loss rate from node D to other nodes

Fig. 6. Performance of four-party conferences over the Internet, running prototype systems of *Celerity*, Simulcast, and the scheme in [4]. (a)-(d): Throughput of individual sessions. (e): Total throughput of all sessions. (f): Utility achieved by different systems. (g)-(h): End-to-end delay and loss rate of session *A*, *C*, and *D* for the *Celerity* system.

(Recall that a session originates from one node to all other three nodes.) As seen, all the session rates in *Celerity* quickly ramp up to near-stable values within 50 seconds, and outperforms Simulcast within 10 seconds. Upon stabilization, *Celerity* achieves the best throughput performance among the three systems and Simulcast is the worst. For instance, all the session rates in *Celerity* is 2x of those in Simulcast and Mutualcast, except in session C where Mutualcast is able to achieve a higher rate than *Celerity*.

We further observe *Celerity*'s superior performance in Fig. 6e, which shows the aggregate session rates, and in Fig. 6f, which shows the total achieved utilities. In both statistics, *Celerity* outperforms the other two systems by a significant margin. Specifically, the aggregate session rate achieved by *Celerity* is 2.5x of that achieved by Simulcast, and is 1.8x of that achieved by Mutualcast.

These results show that our theory-inspired *Celerity* solution can allocate the available network resource to best optimize the system performance. Mutualcast aims at similar objective but only works the best in scenarios where bandwidth bottlenecks reside only at the edge of the network [4].

Figs. 6g-6i show the average end-to-end loss rate and delay from source to receivers for session *A*, session *C* and session *D*. The results for session *B* is very similar to session *A* and is not included here. As seen, the average end-to-end delays of all sessions are within 200 ms, which is our preset delay bound for effective interactive conferencing experience. The average end-to-end loss rate for all sessions are at most 1%-2% upon system stabilization.

The overall operation overhead of *Celerity* in the 4-party Internet experiment is around 3.9%. In particular, the packet overhead accounts for 3.4%, and the link-rate control and link-state report overhead is around 0.5%.

VI. Concluding Remarks

With the proliferation of front-facing cameras on mobile devices, multi-party video conferencing will soon become an utility that both businesses and consumers would find useful. With *Celerity*, we attempt to bridge the long-standing gap between the bit rate of a video source and the highest possible delay-bounded broadcasting rate that can be accommodated by the Internet where *the bandwidth bottlenecks can be anywhere in the network*. This paper reports *Celerity* solution as a first step in making this vision a reality: by combining a polynomial-time tree packing algorithm on the source and an adaptive rate control along each overlay link, we are able to maximize the source rates without any *a priori* knowledge of the underlying physical topology in the Internet. *Celerity* has been implemented in a prototype system, and extensive experimental results in a "tough" dumbbell LAN testbed and on the Internet demonstrate *Celerity*'s superior performance over the state-of-the-art solution Simulcast and Mutualcast.

As future work, we plan to explore source rate control mechanisms beyond the 2-hop tree-packing limitation in *Celerity* to further improve its performance without incurring excessive overhead.

References

[1] Skype, "http://www.skype.com/intl/en-us/home."

[2] Cisco, "http://newsroom.cisco.com/dlls/2010/prod_111510c.html."

[3] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: an efficient mechanism for content distribution in a P2P network," in *Proc. ACM SIGCOMM Asia Workshop*, Beijing, 2005.

[4] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems," in *Proc. ACM SIGMETRICS*, Annapolis, MD, 2008.

[5] İ. E. Akkuş, Ö. Özkasap, and M. Civanlar, "Peer-to-peer multipoint video conferencing with layered video," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 137–150, 2011.

[6] M. Ponec, S. Sengupta, M. Chen, J. Li, and P. Chou, "Multi-rate peer-to-peer video conferencing: A distributed approach using scalable coding," in *IEEE International Conference on Multimedia and Expo*, New York, 2009.

[7] ——, "Optimizing Multi-rate Peer-to-Peer Video Conferencing Applications," *IEEE Trans. on Multimedia*, 2011.

[8] C. Liang, M. Zhao, and Y. Liu, "Optimal Resource Allocation in Multi-Source Multi-Swarm P2P Video Conferencing Swarms," *accepted for publication in IEEE/ACM Trans. on Networking*, 2011.

[9] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area internet bottlenecks," in *Proc. of the 3rd Internet Measurement Conference*, 2003.

[10] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks: Algorithms, measurements, and implications," in *Proc. of ACM SIGCOMM*, 2004.

[11] V. Vazirani, *Approximation algorithms*. Springer Verlag, 2001.

[12] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, no. 5, pp. 556 – 567, Oct. 2001.

[13] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: A low-delay multi-party conferencing solution," The Chinese University of Hong Kong, Hong Kong, Tech. Rep., 2011. [Online]. Available: http://arxiv.org/abs/1107.1138

[14] L. Guo and I. Matta, "QDMR: An efficient QoS dependent multicast routing algorithm," in *Proc. IEEE Real-Time Technology and Applications Symposium*, Canada, 1999.

[15] L. Lovasz, "On two minimax theorems in graph theory," *Journal of Combinatorial Theory, Series B*, vol. 21, no. 2, pp. 96–103, 1976.

[16] Y. Wu, M. Chiang, and S. Kung, "Distributed utility maximization for network coding based multicasting: A critical cut approach," in *Proc. IEEE NetCod 2006*, 2006.

[17] K. Arrow, L. Hurwicz, H. Uzawa, and H. Chenery, *Studies in linear and non-linear programming*. Stanford university press, 1958.

[18] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *Journal of optimization theory and applications*, vol. 142, no. 1, pp. 205–228, 2009.

[19] R. Bruck, "On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space," *Journal of Mathematical Analysis and Applications*, vol. 61, no. 1, pp. 159–164, 1977.

[20] F. Kelly, "Fairness and stability of end-to-end congestion control," *European Journal of Control*, vol. 9, no. 2-3, pp. 159–176, 2003.

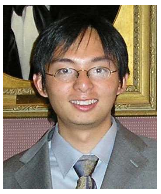
[21] S. H. Low, L. Peterson, and L. Wang, "Understanding vegas: A duality model," *Journal of ACM*, vol. 49, no. 2, pp. 207–235, Mar. 2002.

[22] X. Chen, M. Chen, B. Li, Y. Zhao, Y. Wu, and J. Li, "Celerity: Towards low-delay multi-party conferencing over arbitrary network topologies," in *ACM NOSSDAV*, 2011.

Xiangwen Chen is currently a mobile developer at a start-up company. He received his B.Eng. degree from the Department of Automation at University of Science and Technology of China in 2009, and his Master of Philosophy degree from the Department of Information Engineering at The Chinese University of Hong Kong in 2012.

Minghua Chen is currently an Assistant Professor at CUHK, Hong Kong. He received his B.Eng. and M.S. degrees from the Department of Electronics Engineering at Tsinghua University in 1999 and 2001, respectively, and his Ph.D. degree from the Department of Electrical Engineering and Computer Sciences at University of California at Berkeley in 2006.

Baochun Li is currently a Professor at University of Toronto. He received the B.Eng. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 1995 and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, in 1997 and 2000.



Yao Zhao is currently a software engineer in Site Reliability Engineering of Google. He got his master degree in Computer Science at Tsinghua University in 2001, and his Ph.D. degree in of Electrical Engineering and Computer Science at Northwestern University in 2009.

Yunnan Wu is currently a software engineer at Facebook, Inc. He received the Ph.D. degree from Princeton University in January 2006.

Jin Li is currently a Research Manager and Principal Researcher at Microsoft Research (Redmond, WA). He received his B.S., M.S and Ph.D. degrees from the Electronic Engineering Department, Tsinghua University, Beijing, China, all with honors.