Celerity: Towards Low-Delay Multi-Party Conferencing over Arbitrary Network Topologies

Xiangwen Chen
Dept. of Information Engineering
The Chinese University of Hong Kong

Minghua Chen
Dept. of Information Engineering
The Chinese University of Hong Kong

Baochun Li
Dept. of Electrical and Computer Engineering
University of Toronto

Yao Zhao
Alcatel-Lucent

Yunnan Wu
Facebook Inc.

Jin Li
Microsoft Research at Redmond

ABSTRACT

In this paper, we attempt to 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 in a local experimental testbed.

Categories and Subject Descriptors

C.2.4 [**COMPUTER-COMMUNICATION NETWORKS**]: Distributed System—*Distributed applications*; H.4.3 [**INFORMATION SYSTEMS APPLICATIONS**]: Communications Applications—*video conferencing*

General Terms: Algorithm, Design, Experimentation, Performance

Keywords: Peer-to-peer, Multi-party video conferencing, Low delay, Utility maximization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NOSSDAV'11, June 1–3, 2011, Vancouver, British Columbia, Canada.
Copyright 2011 ACM 978-1-4503-0752-9/11/06 ...$10.00.

1. 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. 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 arbitrary network topologies over the Internet.

From the 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 sessions 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] [4] [5] [6] [7]. Among them, Mutualcast [3] and Chen *et al.* [4] may be the most related ones to this work. 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. 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 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.

In this paper, we reconsider the design space in multi-party video conferencing solutions, and present *Celerity*, a new multi-party conferencing solution specifically designed to maintain low end-to-end delays while maximizing source coding rates in a session. *Celerity* is designed to operate in a pure P2P manner, and as such eliminating the cost of maintaining centrally administered servers. It is designed to deliver video at quality levels commensurate with available network resources over arbitrary network topologies, while maintaining low end-to-end delays. 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 a prototype *Celerity* system and demonstrate its superior performance in a local experimental testbed.

2. PROBLEM FORMULATION AND CELERITY OVERVIEW

2.1 Problem Formulation

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 $V - \{v\}$. We use E to denote the set of directed overlay links between these nodes. Note an overlay link (u, v) means u can send data to v by setting up TCP/UDP connections. For all $e \in E$ and $l \in \mathcal{L}$, we define

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

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

A fundamental system design problem is to maximize the application-specific performance, by properly allocating the overlay link rates to the streams subject to physical link capacity constraints:

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

$$\text{s.t.} \quad \boldsymbol{a}_l^T \left(\boldsymbol{c}_1 + \ldots + \boldsymbol{c}_M\right) \leq C_l, \quad \forall l \in \mathcal{L}, \quad (3)$$

The variables in the above optimization are $\boldsymbol{c} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M]^T$, where $\boldsymbol{c}_m$ is the vector of overlay link rates allocated to stream m (with one entry for each overlay link). $R_m(\boldsymbol{c}_m)$ denotes the stream rate that we obtain by using resource $\boldsymbol{c}_m$ *within the given delay bound*, and is a concave function of $\boldsymbol{c}_m$ as we will show in Corollary 1 in the next section. The constraint in (3) is the physical link capacity constraint, where $\boldsymbol{a}_l^T \boldsymbol{c}$ describes the load on the physical link l incurred by overlay traffic $\boldsymbol{c}$.

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].

Remarks: 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 node uplinks being the only capacity bottleneck) by packing only certain depth-1 and depth-2 trees within a session.

2.2 Celerity Overview

To achieve the maximum system utility, the Celerity system has two main modules: (1) *delay-bounded packet delivery* at the highest possible source rate given known rates on each of the links (i.e., how to compute and achieve $R_m(\boldsymbol{c}_m)$); and (2) a *link rate control* module to determine $\boldsymbol{c}_m$.

Video content 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, and we introduce a delay-bounded tree packing algorithm in this paper to address this problem (detailed in Section 3).

Link rate control: Under our setting, the formulation in (2)–(3) is a concave optimization problem. In principle, one can first infer the network constraints and then solve the problem centrally. However, directly inferring the constraints potentially requires knowing the entire network topology and is highly challenging.

In Celerity, instead of trying to learn the constraints directly, we resort to adaptive and iterative algorithms for solving an approximate version of the problem given in (2)-(3) in a distributed way (detailed in Section 4).

3. PACKING DELAY-BOUNDED TREES

Given the link rate vector $\boldsymbol{c}_m$, achieving the maximum broadcast/multicast stream rate under a delay bound 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 [8]. 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 trees suffices to achieve the

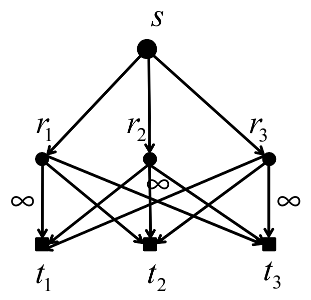


Figure 1: Illustration of the directed acyclic sub-graph over which we pack delay-bounded 2-hop trees.

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. 1. 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 (i, j) being $c_{m,(i,j)}$. Then in $\mathcal{D}_m$, the link from s to r_i has capacity $c_{m,(s,i)}$, the link from r_i to t_j (with $i \neq j$) has capacity $c_{m,(i,j)}$, and the link from r_i to t_i has infinite capacity. If the propagation delay of an edge (i, j) exceeds the delay bound, we do not include it in the graph. If the propagation delay of a two-hop path $s \to i \to j$ 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 [9], which achieves the optimal throughput in multicast. Below we simply describe the algorithm and more details can be found in [9].

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 augmenting a tree edge by edge, 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. Our tree-packing algorithm is easy to implement.

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 [9] achieves the minimum of the min-cuts separating the source and receivers in $\mathcal{D}_m$ and is expressed as*

$$R_m(\boldsymbol{c}_m) = \min_j \sum_i \min\{c_{m,(s,i)}, c_{m,(i,j)}\}. \tag{4}$$

Furthermore, the algorithm has a running time of $O(|V||E|^3)$.

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

Corollary 1. *The delay-bounded multicast rate $R_m(\boldsymbol{c}_m)$ obtained by our tree-packing algorithm is equal to the minimum min-cut over $\mathcal{D}_m$, and thus is a concave function of the overlay link rates $\boldsymbol{c}_m$.*

4. OVERLAY LINK RATE CONTROL

4.1 Packet Loss Rate Based Primal Subgradient Algorithm

The primal algorithm is derived by relaxing the constraints and adding a penalty to the objective function whenever constraints are violated. This leads to an unconstrained version of the original problem, making it easier to solve.

Consider a penalized version of the problem in (2)-(3):

$$\max_{\boldsymbol{c}\geq 0} \ \mathcal{U}(\boldsymbol{c}) \triangleq \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{c}_m)\right) - \sum_{l\in\mathcal{L}} \int_0^{\boldsymbol{a}_l^T \boldsymbol{c}} p_l(y)\, dy, \tag{5}$$

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

$$p_l(y) \triangleq \frac{(y - C_l)^+}{y} dy, \tag{6}$$

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.

We seek to maximize $\mathcal{U}(\boldsymbol{c})$ as an approximation of the original constrained optimization problem. With this choice of price function, $\mathcal{U}(\boldsymbol{c})$ is a linear combination of concave functions and is thus concave. However, because $R_m(\boldsymbol{c}_m)$ 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 [10].

For the unconstrained concave optimization problem with non-differentiable objective function, we can solve it by using subgradient algorithms. To proceed with subgradient algorithm design, we need to 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*

$$U_m'\left(R_m\right) \frac{\partial R_m}{\partial c_{m,e}}$$

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

Based on the above observation, we apply the following subgradient algorithm to solve the problem in 5: $\forall e \in E$, $m = 1, \ldots M$,

$$c_{m,e}^{(k+1)} = c_{m,e}^{(k)} + \alpha(k)\left[U_m'\left(R_m^{(k)}\right)\frac{\partial R_m^{(k)}}{\partial c_{m,e}} - \sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{c}^{(k)} - C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{c}^{(k)}}\right]^+_{c_{m,e}^{(k)}}, \tag{7}$$

where $\alpha(k)$ is a positive step size for the k-th iteration, and function

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

We have the following observations to the control law in (7):

- It is known that $\sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{c} - C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{c}}$ can be interpreted as the packet loss rate observed at overlay link e [11]. The intuitive explanation is as follows. The term $(\boldsymbol{a}_l^T\boldsymbol{c} - C_l)^+$ is the excess traffic rate offered to physical link l; thus $\frac{(\boldsymbol{a}_l^T\boldsymbol{c} - C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{c}}$ 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 rate seen by the overlay link e is given by $\sum_{l\in\mathcal{L}} a_{l,e}\frac{(\boldsymbol{a}_l^T\boldsymbol{c}-C_l)^+}{\boldsymbol{a}_l^T\boldsymbol{c}}$.

- It turns out that the utility function, the subgradients, and packet loss rate are sufficient statistics to update $c_{m,e}$ independently of the updates of other link rates. This way, we can solve the problem in (5) without knowing the physical network topology and physical link capacities.

The subgradient method [12] maximizes a non-differentiable concave function in a way similar to gradient methods for differentiable functions — in each step, the variables are updated in the direction of a subgradient. However, such a direction may not be an ascent direction; instead, the subgradient method relies on a different property. If the variable takes a sufficiently small step along the direction of a subgradient, then the new point is closer to the set of optimal solutions.

We have the following convergence results for the algorithm in (7), which is adapted from those for standard subgradient algorithms [12].

Theorem 2. *Assume $\mathcal{U}^*$ is the optimal value of the problem in (5) and $|U_m'|$ is upper bounded by $\bar{u}$ for all $m = 1,\ldots,M$. The l_2-norm of any subgradients of $R_m(\boldsymbol{c}_m^{(k)})$ is upper bounded by $\Delta = \bar{u}^2|V|^2|E|^2$ for all k. Thus,*

- *if $\alpha(k) = \alpha$ is constant, then $\lim_{k\to\infty}\left[\mathcal{U}^* - \max_{i=1,\ldots k}\mathcal{U}(\boldsymbol{c}^{(k)})\right] \le \Delta^2\alpha$.*
- *if $\alpha(k) \to 0$ or $\sum_{k=1}^{\infty}\alpha^2(k) < \infty$, and $\sum_{k=1}^{\infty}\alpha(k) = \infty$, then $\lim_{k\to\infty}\left[\mathcal{U}^* - \max_{i=1,\ldots k}\mathcal{U}(\boldsymbol{c}^{(k)})\right] = 0$.*

In this paper, we choose a constant step size for easy implementation.

4.2 Computing Subgradient of $R_m(\boldsymbol{c}_m)$

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

Definition 1. *Given a convex function f, a vector ξ is said to be a subgradient of f at $x \in$ **dom**f if*

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

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, $V = Z \cup \bar{Z}$ with $s_m \in Z$ and $t \in \bar{Z}$ for some $t \in V_m$, determines an s_m-t-cut. Define

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

be the set of overlay links originating from nodes in set Z and going into nodes in set $\bar{Z}$. Define the capacity of cut $(Z,\bar{Z})$ 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 $(Z,\bar{Z})$ is an s_m-V_m critical cut if it separates s_m and any of its receivers and $\rho(Z) = R_m(\boldsymbol{c}_m)$.*

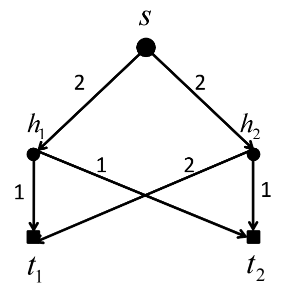


Figure 2: Critical cut example. Source s and its two receivers t_1, t_2 are connected over a directed graph. The number associated with a link represents its link capacity.

We show an example to illustrate the concept of critical cut. In Fig. 2, 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 $R_m(\boldsymbol{c}_m)$. Since $R_m(\boldsymbol{c}_m)$ 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 [10].

- Find an s_m-V_m critical cut for session m, denote it as $(Z,\bar{Z})$. 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 $R_m(\boldsymbol{c}_m)$ with respect to $c_{m,e}$ is given by

$$\frac{\partial R_m(\boldsymbol{c}_m)}{\partial c_{m,e}} = \begin{cases} 1, & \text{if } e \in \delta(Z); \\ 0, & \text{otherwise.}\end{cases} \tag{8}$$

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 Section 5.

5. PRACTICAL IMPLEMENTATION

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++.

In our Celerity prototype implementation, all peers are to perform the following functions:

- Peers in broadcast trees forward packets received from its upstream parent to its downstream children. Sufficient information about downstream children in the tree is embedded in the packet header, for a packet to become "self-routing" from the source to all leaf nodes in a tree.
- Every 200 ms, each peer adjusts the rates of its incoming links based on the link rate control algorithm, and then sends them to their corresponding upstream senders for the new rates to take effect.
- Every 300 ms, each peer sends the allocated rates of all its outgoing links for each session to the source of the session.

Upon receiving allocated rates for all the links, the *source* of each session uses the received link rates to pack a new set of delay-bounded trees, and starts transmitting session packets along these trees. When a source packs delay-bounded trees, it also calculates *one* critical cut and the source sending rate for its session based on

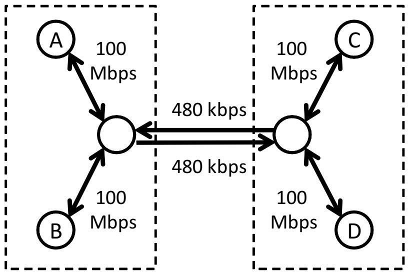


Figure 3: Dumbbell topology of the experimental testbed. Two conference participating nodes *A* and *B* are in one “office” and another twos nodes *C* and *D* are in a different “office”. The two “offices” are connected by directed links, each having a capacity of 480 kbps. Link propagation delays are negligible.

the allocated link rates. In addition, the source embeds the information about the critical cut and the source sending rate in the header of outgoing packets. When these packets are received, a peer learns the source rate and whether a link belongs to the critical cut or not; it then adjusts the link rate accordingly.

The calculation of critical cuts, *i.e.*, the subgradient of $R_m(C_m)$, is the key to our implementation of the primal subgradient algorithm. There can be multiple critical cuts in one session, but it is sufficient to find any one of them. Since the source collects allocated rates of all overlay links in its own session, it can calculate the min-cut from the source to every receiver, and record the cut that achieves the min-cut. Then, the source compares capacities of these min-cuts, and the cut with the smallest capacity is a critical cut.

With respect to the *utility function* in our prototype implementation, the PSNR (peak-to-peak signal-to-noise ratio) metric is the *de facto* standard criterion to provide objective quality evaluation in video processing. We observed that the PSNR of a video stream coded at a rate z can be approximated by a logarithmic function $\beta \log(z + \delta)$, in which a higher β represents videos with a larger amount of motion [4]. δ is a small positive constant to ensure the function has a bounded derivative for $z \geq 0$. Due to this observation, we use a logarithmic utility function in our implementation.

In order to quickly bootstrap our system to close-to-optimal operating points, we implement a method called “quick start” to aggressively ramp up the rates of all sessions during a conference initialization stage, during which peers are joining the conference and nothing significant is going on. We achieve this by using larger values for β in the utility functions and a large step size in link rate adaptation during the first 30 seconds. After the initialization stage, we reset β and step sizes to proper values and allow our system converge gradually and avoid unnecessary performance fluctuation.

6. EXPERIMENTS

We evaluate our prototype Celerity system over a LAN testbed. The testbed is 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 the scheme in [4], fail to efficiently utilize the bottleneck bandwidth and optimize system performance.

In our experiments, all four peers run our prototype system. We run a four-party conference for 300 seconds, evaluate the system performance, and show the results in Fig. 5. Since the experimental settings are symmetric for each participating peer, it is straightforward to verify the optimal source rate for each peer to be 240 kbps, and the optimal aggregate utility is around 175.

Figs. 5a-5d show the source sending rate and receivers’ receiving rates of each session (one session originates from one peer to all other 3 peers). As seen, our system demonstrates fast convergence: the sending rate of each session quickly ramps up to 95% to the optimal within 50 seconds. Fig. 5e shows that our system quickly achieves the optimal utility. The tiny gap between the converged sending rate and the optimal one is because our proposed system only aims to solve an approximate version of original problem and hence is only expected to obtain close-to-optimal solutions.

As a comparison, we also plot the theoretical maximum rates achievable by Simulcast and the scheme in [4] in Figs. 5a-5d. As seen, within 20 seconds, our system already outperforms the maximum achievable rates of Simulcast and the scheme in [4].

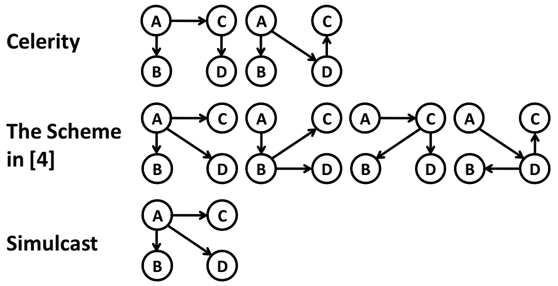


Figure 4: Session A’s trees used by Celerity (upon convergence), the scheme in [4] and Simulcast in dumbbell topology.

Upon convergence, our system achieves sending rates that are nearly twice of the theoretical maximum rate achievable by Simulcast and the scheme in [4]. This significant gain is due to that our system can utilize the bottleneck resource more efficiently.

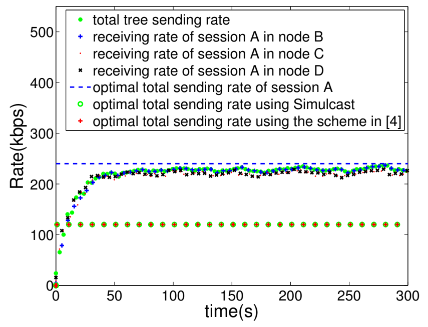
In Fig. 4, we show the trees that are used by our algorithm, the scheme in [4] and Simulcast in the dumbbell topology. As seen, for session *A*, every tree used by Simulcast and the scheme in [4] consumes the bottleneck link resource twice, thus to deliver one-bit of information they consume two-bit 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 link between the two “offices” , consuming twice of the bottleneck link resource. Consequently, the maximum achievable rates of Simulcast and the scheme in [4] are all 120 kbps. In contrast, our system upon convergence utilizes the trees that only consume bottleneck bandwidth once, thus it achieves close-to-optimal rates of about 240 kbps.

There are small gaps between the sending rate and the receiving rates in the same session. For instance, upon convergence, the lowest receiving rate is about 97% of the sending rate in session *A*. This is because our system relies on packet loss to adapt and converge, and the gaps are due to the packet loss that occurs at the bottleneck links. To verify, we show the end-to-end packet loss rate of session *A* in the bottom sub-figure in Fig. 5f. The packet loss rate is about 3% upon convergence, and explains the 3% gap between the sending rate and the lowest receiving rate of session *A*.

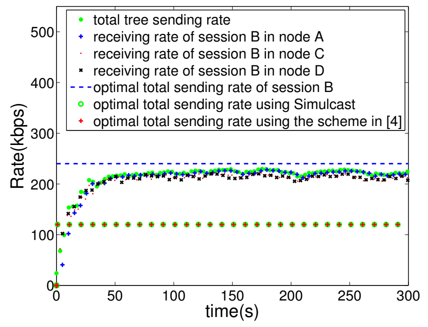
We show the end-to-end delay and packet loss rate in Fig. 5f, all for session *A*. The results for the other sessions are essentially the same due to their symmetric settings.

As seen in the top sub-figure in Fig. 5f, When our system converges, the average end-to-end delay from node A to the receiver in the same “office” (in this case node *B*) is negligible. The delays to the receivers in the other “office” (nodes *C* and *D*) are about 90 ms on average, which is contributed by the queuing delay at the bottleneck links. All delays are within the acceptable range for smooth conferencing experience.

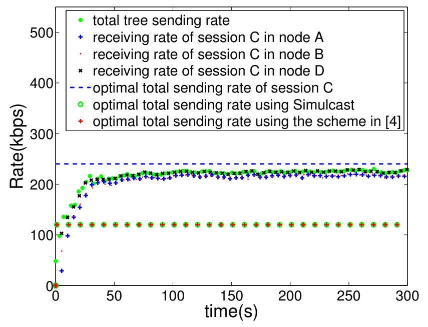
The bottom sub-figure in Fig. 5f shows the average end-to-end packet loss rate of session *A*. As seen, the packet loss rate is high initially, and decreases and stabilizes to small values afterwards.



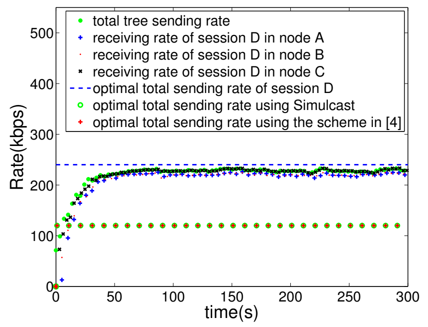
(a) Rate performance of session A



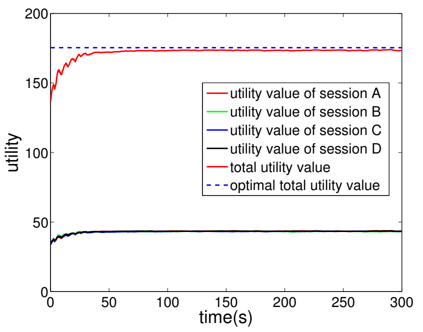
(b) Rate performance of session B



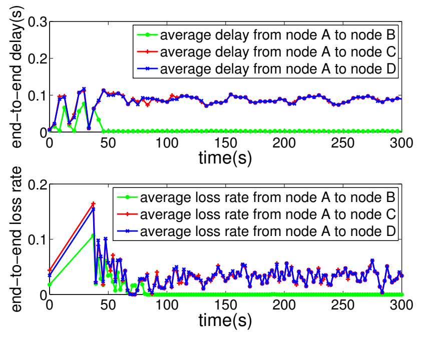
(c) Rate performance of session C



(d) Rate performance of session D



(e) Aggregate system utility performance



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

Figure 5: Performance of Celerity over a dumbbell LAN testbed. (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.

The initial high loss rate is because at the beginning our system increases the sending rates aggressively to bootstrap the conference and explore the network resource limit. For instance, for session A, the source node A explores all possible paths to deliver packets to node B at the beginning, including path $A \rightarrow C \rightarrow B$ and path $A \rightarrow D \rightarrow B$. These two paths suffer from high loss rate. This aggressive behavior introduces high loss rate, but only during the conference initialization stage when usually nothing significant is transmitted. Our system quickly learns the resource bottleneck and adapts to the network topology, ending up with using the cost-effective trees to deliver data. After the initialization stage, our system adapts and converges gradually, avoiding unnecessary performance fluctuation that deteriorates user experience.

7. 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 a first step towards making this vision a reality: by combining a polynomial-time tree packing algorithm on the source and 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 preliminary experimental results over a "tough" dumbbell topology are very encouraging. As future work, we will continue to fine-tune our rate control algorithm and evaluate Celerity's performance by Internet experiments. We will also consider alternatives and additions that involve random network coding.

8. 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] 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.

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

[9] Y. Wu, P. A. Chou, and K. Jain, "A comparison of network coding and tree packing," in *International Symposium on Information Theory*, Chicago, USA, 2004.

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

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

[12] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific Belmont, MA, 1999.