Utility Maximization in Peer-to-Peer Systems with Applications to Video Conferencing

Minghua Chen, Miroslav Ponec, Sudipta Sengupta, Jin Li, and Philip A. Chou

***Abstract*— In this paper, we study the problem of utility maximization in P2P systems, in which aggregate application-specific utilities are maximized by running distributed algorithms on P2P nodes, which are constrained by their uplink capacities. For certain P2P topologies, we show that routing along a linear number of trees per source can achieve the largest rate region that can be possibly obtained by intra-session and inter-session network coding. This observation allows us to develop a simple multi-tree formulation for the problem. For the resulting non-strictly concave optimization problem, we develop a Primal-dual distributed algorithm and prove its global convergence using our proposed sufficient conditions. These conditions are general and add understanding to the convergence of primal-dual algorithms under non-strictly concave settings. We implement the proposed distributed algorithm in a peer-assisted multi-party conferencing system by utilizing only end-to-end delay measurements between P2P nodes. We demonstrate its superior performance through actual experiments on a LAN testbed and the Internet.**

I. Introduction

The problem addressed in this paper is motivated by Peer-to-Peer (P2P) multi-party conferencing applications in which providing Quality-of-Service (QoS) is a crucial challenge. Because the Internet is not a dedicated network, voice or video conferencing applications must share the available network resource with other applications, and adjust the coding rate, protection scheme and network delivery path to maximize the quality of experience of all peers involved. We measure the quality of experience of the conferencing peer by a utility function. For video conferencing, it can be the Peak-Singal-to-Noise Ratio (PSNR) of the decoded video, or a more sophisticated subjective quality measure such as [1].

Traditional multi-party conferencing (VoIP and/or video conferencing) is conducted using either a client-server architecture or in an ad hoc simulcast way.

In the client-server approach, every peer sends its audio/video session to a centralized server and the server replicates the session and redistributes the copies to all other participating conference peers. This approach ensures that the entire upload bandwidth of each peer can be used for the delivery of just that peer's audio/video session; however, it places a heavy CPU and network bandwidth burden on the central server.

In the ad hoc simulcast approach, each user splits its uplink bandwidth equally among all receivers and sends its video to each receiver separately. Though simple to implement, this approach suffers from poor quality of service, especially when there is one peer with low upload bandwidth, as that peer is forced to use a low coding rate that degrades the overall experience of the other peers.

In contrast, the P2P approach for multiparty video conferencing that we consider in this paper *does not necessarily rely on centralized infrastructure and allows a peer to not only use its uplink to send its video stream but also to forward the video stream of other peers.* This approach facilitates optimal use of peer uplink bandwidth in the system and naturally accommodates peer uplink heterogeneity.

A. Related Work

In the past decade, network utility maximization have attracted significant attention. In the seminal framework introduced in [2] and [3], network protocols are understood as distributed algorithms that maximize aggregate user utility under wired or wireless network resource constraints. In the framework, user's utility function is typically assumed to be strictly concave function of user rate, and the resource constraints set is linear. Various types of fairness across users can be warranted by choosing different utility functions [4]. The framework not only provides a powerful tool to reverse engineer existing protocols such as TCP [5], but also allows systematic design of new protocols. See [6] for a comprehensive review.

There have been work on extending the framework from its original single-path unicast setting [2], [3] to multi-path unicast scenarios [7] [8] [9], as well as single-tree multicast scenarios [10] [11]. For utility maximization in multi-path unicast scenarios, the utility function is non-strictly concave with respect to the individual path rate due to multi-path routing. The challenge is to design distributed algorithms to solve non-strictly concave optimization problems with provable fast convergence and easy implementation. Primal and Dual algorithms, and proximal approach are proposed to address such challenges [7] [8] [9].

For utility maximization in single-tree multicast scenarios where routers enable multicast functionality, the multicast session rate over a link on the tree is the maximum of the rate of all downstream receivers of the session. Consequently, every link capacity constraint in general involves multiple non-differentiable $\max(\cdot)$ terms, i.e., the sum of multiple multicast session rates is less than the physical link capacity. In [10] and [11], distributed Primal and Dual algorithms are proposed to maximize utility, under the assumptions that multicast trees are given and every session has a unique source. The challenge of dealing with non-differentiable max function in the constraints is approached by either using continuous and concave approximation of the max function [11], or introducing auxiliary variables and applying either Proximal or sub-gradient approaches [10].

An earlier (and shorter) version of this paper appeared in ACM SIGMETRICS 2008. Minghua Chen is with Department of Information Engineering, The Chinese University of Hong Kong. His e-mail is minghua@ie.cuhk.edu.hk. Miroslav Ponec is with Akamai Technologies GmbH. His e-mail is mponec@akamai.com. Sudipta Sengupta, Jin Li and Philip A. Chou are with Microsoft Research, Redmond. Their e-mails are {sudipta, jli, pachou}@microsoft.com, respectively.

There is also work focusing on multicast scenarios where routers can perform intra-session network coding [12] [13] [14]. The challenge is to deal with non-strictly concave optimization under non-linear constraints. By exploring the Proximal approach, or a slow timescale traffic engineering control approach, or expressing the constraints involving max(·) terms with equivalent linear ones, distributed Primal, Dual subgradient and Primal-dual algorithms are proposed to maximize the sum of non-concave utility functions, or minimize the cost of using the network [12] [13] [14].

B. Our Contributions

In this paper, we consider the utility maximization problem for multiple multicast under P2P setting, with intra- and inter-session network coding allowed. This setting differentiates our work from prior arts, and highlights the challenges we encounter. Multi-party conferencing is one application of our work. Our main contributions are as follows:

- **The Optimality of Routing on P2P Topology:** We focus on typical P2P topology where peer uplinks are the *only* bottleneck in the network. For multi-source multicast on certain P2P topologies, we show that all feasible rates can be achieved by packing polynomial number of Steiner trees. As such, routing is optimal even if the system contains Steiner nodes (helpers), and surprisingly there is no gain to perform (intra- or inter-session) network coding on peer nodes. This result is a multi-source extension of the single source result studied in [15].
- **New Tree-based Formulation:** We introduce a new formulation for utility maximization in P2P topology in which the variables are rates of individual *trees*. Our tree based formulation uses linear constraints, thus avoiding the nonlinear $\max(\cdot)$ terms in the alternative path and link based formulations in the literature. Using unique properties of P2P topology, we show that our formulation achieves maximum utility by routing along a linear number of depth-1 or depth-2 trees for each source in the *overlay* network. As such, our solution is not only optimal but also readily implementable on today's Internet.
- **Primal-Dual Algorithm with Fast Convergence:** Contrary to popular belief that Primal-dual algorithms in general fail to converge in multi-path/multi-tree scenarios with supporting evidence in [9], we design a queuing delay based Primal-dual algorithm that solves the utility maximization problem under our multi-tree setting. The convergence of the algorithm is proved by using our proposed general sufficient conditions. Our proposed algorithm is distributed and is implementable by utilizing only end-to-end delay measurements between peer nodes.
- **Evaluation on the Internet:** We have built a prototype multi-party conferencing system based on the proposed Primal-dual algorithm, using Python programming language. We evaluated its performance for several multi-party conferencing scenarios on a LAN testbed, in a virtual environment, and also on the Internet. Our system achieves low end-to-end delay packet delivery for conferencing systems, since every packet goes through at most one hop in the overlay and we tightly control the queuing delay between nodes.

The rest of the paper is structured as follows. In Section II, we state the results on optimality of routing over P2P topology, and present the tree-rate based formulation for the utility maximization problem in P2P systems. In Section III, we propose a delay-based Primal-dual distributed algorithm to this problem. Its global exponential convergence is proved under the popular P2P settings we consider in this paper. We discuss how to implement our proposed algorithm in practice in Section IV. In Section V, we evaluated the performance of our proposed algorithm in several multi-party conferencing scenarios on a LAN testbed, in a virtual environment, and on the Internet. Conclusions and future works are provided in Section VI.

II. Problem Formulation

We consider a network presented by a directed graph $G = (V, J)$, where V is the set of vertices, i.e., nodes in the network, and J is the set of edges, i.e., links in the *physical* network. Assume each link $j \in J$ has a finite capacity C_j. Let $n = |V|$.

In the P2P systems we consider, a source node $s \in S$ which is a subset of V sends its content to a set of receivers, denoted by R_s. A set of helper nodes, denoted by H, are willing to help in distributing the content. In this paper, we assume a deterministic fluid model for sending rates of nodes and ignore packet dynamics. This assumption is reasonable when the timescale of rate control is sufficiently larger than that of packet dynamics.

Let z_s be the multicast rate of source s, and $z = \{z_s, s \in S\}$. Assume all members in R_s receive s's stream at this rate. Let $U_s(z_s)$ be the utility upon receiving the content from s at rate z_s. To prevent abusing the resources from helpers, sources and receivers should use helpers' resources only after they have used up their own. Putting this into consideration, we associate a cost, denoted by $G_h(z)$, with using a helper $h \in H$ to distribute a content.

In this multiple multicast scenario, a natural goal is to maximize the aggregate net utility of all receivers, subject to rate constraints, i.e.,

$$\max_z \sum_{s \in S} U_s(z_s) - \sum_{h \in H} G_h(z), \text{ subject to constraints on } \{z_s\}.$$

Before formulating the problem further, we need to understand the constraint region for $\{z_s\}$ to optimize over and how to achieve it.

A. Network Coding vs. Routing

The maximum achievable multicast rate of single source multicast scenario is characterized as the minimum of the min-cuts between the source node s and all nodes in its receiver set R [16], i.e., $\min_{t \in R}$ min-cut (s, t). For example, in the classical Butterfly network shown in Fig. 1.(a), a source s multicasts to two receivers t_1 and t_2. The min-cuts between s and t_1 and t_2 are all 2. Thus, the maximum achievable multicast rate is 2.

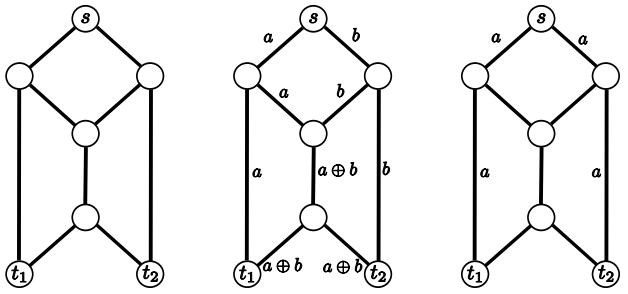


Fig. 1. (a) Butterfly network with unit link capacities. (b) Network coding can achieve a multicast rate of 2. (c) Routing can achieve a multicast rate of 1.

If network coding is allowed, then the single source multicast rate region can be achieved for arbitrary topology by solving the routing and coding problems separately, each being of polynomial complexity [17]. For example, as seen in Fig. 1.(b), by performing XOR operation in the Steiner nodes in the Butterfly network, we can achieve the maximum achievable multicast rate 2.

The achievable rate region for multi-source multicast scenarios was recently implicitly characterized in [18], but currently no scheme is known to achieve it. It is believed that information from different multicast groups should be coded in a nonlinear fashion in order to achieve the rate region (inter-session coding). However, doing such mixing and coding is complex and largely open.

Regardless of its power, network coding is not quite practical in today's P2P applications. It cannot be used in the Internet routing layer because it requires changes in all routers (for encoding) and end-hosts (for decoding). If deployed in the overlay (P2P layer), it will introduce new complexity in end-host software (for encoding and decoding) and additional delays in video delivery. A practical way to explore the achievable rate region is by routing. Each source s packs directed Steiner trees rooted at s and reaching all receivers in R_s. For the *general case of arbitrary topologies*, this approach of routing brings up the following difficulties:

1) For a given source, the maximum rate achieved by routing can be a factor of up to $\log|V|$ lower than that achieved by network coding [17]. For example, as seen in Fig. 1.(c), by packing Steiner trees in the Butterfly network, we can only achieve multicast rate of 1, as compared to 2 achieved by network coding approach.
2) To achieve the maximum rate for routing, the problem of packing directed Steiner trees is $\mathcal{NP}$-hard [19]. Moreover, the number of Steiner trees used in an optimal solution may be exponential.

As such, routing cannot achieve the optimal rate region in general topology and its cost could be prohibitively large. However, the fact that our problem involves a P2P topology where peer uplinks are the only bottlenecks (in practice) in the network allows us to tackle all of the above difficulties in a surprisingly elegant manner.

B. Impact of P2P Topology

In P2P topology, we assume every peer can connect to every other peer through routing in the overlay. Similar to other P2P work [15], [20], [21], [22], we make the uplink bottleneck assumption that node uplinks are the only rate limiting bottlenecks in the network. In the overwhelming majority of residential broadband connections, bottlenecks typically are at the edge of the access networks rather than in the middle of the Internet. Furthermore, it is common to have the uplink capacity of a peer to be several times smaller than the downlink capacity, thus justifying the practicality of our assumption on P2P topology to some extent. Formally, if a peer i has uplink capacity r_i^U, downlink capacity r_i^D, and is a source of data at rate R_i, and a sink of data at rate R'_i (i.e., it is not uploading this data to any other peer), then its downlink is not a bottleneck if $r_i^D \geq R'_i + (r_i^U - R_i)$.

In the context of P2P topology with the above uplink constraint assumptions, a powerful theorem established in the Mutualcast paper [15] states the following. Consider a network with P2P topology consisting of a source s, a set of receivers R_s, and a set of helpers H. Then, the min-cut capacity for source s and receivers R_s can be achieved by packing at most $1 + |R_s| + |H|$ Mutualcast trees as follows:

- One depth-1 tree rooted at s and reaching all receivers in R_s, i.e. the type (1) tree in Fig 2.
- $|R_s|$ depth-2 trees, each rooted at s and reaching all other receivers in R_s via different $r \in R_s$, i.e. the type (2) tree in Fig 2.
- $|H|$ depth-2 trees, each rooted at s and reaching all receivers in R_s via different $h \in H$, i.e. the type (3) tree in Fig 2.

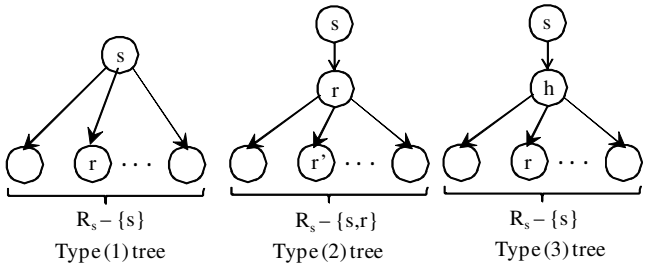


Fig. 2. Different types of Mutualcast trees.

This result extends and simplifies Edmonds' theorem [23] for P2P topology, in the sense that it allows helper (Steiner) nodes and uses only depth-1 and depth-2 Steiner trees. Fig. 3(b) shows all 12 Mutualcast trees for a three peers and one helper scenario where each peer wants to multicast its content to the other two. For a special case where there is no helper nodes in the network, authors in [21] and [20] also derived results similar to the above-stated Mutualcast theorem independently.

The original Mutualcast Theorem is for single source multicast scenario only, we have extended the result to the case of a multi-source multicast scenario when there is no coding across sessions belonging to different sources in [24].

Further, surprisingly, we show in the following theorem that routing is optimal and *inter-session network coding* is not needed, if we require that each receiver is part of every session, i.e., $R_s \cup \{s\} = R$ for all $s \in S$. (Note that each receiver need not be a source though.) Such a scenario is common in multi-party conferencing systems.

Theorem 1: Consider a P2P topology in which peer uplinks are the only bottleneck. Consider multiple multicast sessions given by source nodes $s \in S$, receiver set R_s, and helper nodes $H_s = V - \{s\} - R_s$ for session with source s. Further, assume that each receiver is part of every session, i.e., $R_s \cup \{s\} = R$ for all $s \in S$, and hence $H_s = V - R = H$ for all s. Then, the largest achievable rate region $z = \{z_s, s \in S\}$, achievable by **network coding across sessions**, can be achieved by routing along $1 + |R_s| + |H_s|$ Mutualcast trees for each source s independently. Furthermore, let $C_o(v)$ be the uplink capacity constraint for node v in V, then the largest achievable region is given by

$$\left\{ z : z_s \le C_o(s), \forall s \in S,\ |R| \sum_{s \in S} z_s \le \sum_{v \in V} C_o(v) - \frac{1}{|R|} \sum_{h \in H} C_o(h) \right\}$$

Proof: Refer to Appendix A. ∎

In contrast to the known results that inter-session coding is needed to achieve the maximum rate region in general topology, the unique structure of the P2P topology we consider in this paper allows us to achieve the maximum rate region by packing only linear number of Steiner trees per source, if each receiver is part of every multicast session. This result is not only surprising but also elegant.

We summarize the advantages and disadvantages of using network coding and packing (directed) Steiner trees to achieve multicast rate region in Table I.

C. Optimization Framework

With the packing Mutualcast trees approach, each source $s \in S$ builds a set of depth-1 and depth-2 Mutualcast trees to send data to all receivers in R_s along the trees. We denote this set of trees also as s, and a source is identified by the set of trees of which it is the root. Another advantage of this packing trees approach is that the resulting solution also includes session scheduling; therefore, the latter need not be solved as a separate problem.

A tree $m \in s$ is a set of links and nodes that the tree passes through; all receiver nodes on a tree receive the same content at the same rate. We denote the rate of tree m as x_m. Rates of the trees rooted at source s sum up to the source rate z_s, i.e., $\sum_{m \in s} x_m = z_s, \ \ \forall s \in S$. The injecting rate of link j is the aggregate rate of the trees that pass through link j, denoted by y_j, and is given by,

$$y_j \triangleq \sum_{s \in S} \sum_{m \in s: j \in m} b_j^m x_m, \ \ \forall j \in J, \tag{1}$$

where b_j^m is the number of tree m's branches that pass through physical link j. Since different branches of a tree in the overlay can pass through the same physical link in the underlay, the tree rates might be counted multiple times when computing the injecting rate of a link, hence the multiplication by b_j^m.

Similarly, define the forwarding rate of a helper node h as

$$y_h \triangleq \sum_{s \in S} \sum_{m \in s: h \in m} b_h^m x_m, \ \ \forall h \in H, \tag{2}$$

where b_h^m is the out-degree of helper node h in multicast tree m. Denote $y^H = [y_h, h \in H]$.

The aggregate utility maximization problem in P2P systems can be formulated as follows:

$$\begin{aligned} \max_{\{x_m\}} \quad & \sum_{s \in S} |R_s| U_s \left(\sum_{m \in s} x_m \right) - \sum_{h \in H} G_h(y_h) \quad (3) \\ s.t. \quad & y_j \le C_j, \ \ \forall j \in J, \end{aligned}$$

where $|R_s| U_s \left(\sum_{m \in s} x_m \right)$ is the aggregate utility of a group R_s upon receiving content at rate $\sum_{m \in s} x_m = z_s$, and $G_h(y_h)$ is the cost of using helper node h to deliver content at rate y_h. As discussed earlier, this cost is to prevent peers from abusing resources from helpers – sources and receivers should use helpers' uplink capacities only after they use up their own. Formally, if the optimum objective function value can be achieved without using (or using lower) helper uplink capacities, then this should be preferred.

We assume that the utility functions $U_s(\cdot), s \in S$, are strictly concave, and the cost functions $G_h(\cdot), h \in H$ are strictly convex.

This problem formulation is applicable to many P2P applications in practice. For example, in P2P video conferencing systems with utility being the video quality, the problem in (3) corresponds to maximizing the aggregate video quality of all receivers. This formulation is flexible in the sense the tree loss and delay characteristics can be easily taken into account by adding a term $e_m x_m$ with negative e_m into the utility function representing the delay or loss cost of using tree m.

The optimization problem in (3) is a non-strictly concave optimization problem with linear constraints. It might have more than one optimal $\{x_m\}$. However, the optimal aggregate rate associated with each source $\{z\}$ is unique. This is because the objective function is strictly concave with respect to $\{z_s\}$, and the rate constraint region of $\{z_s\}$ can be shown to be a polyhedron by eliminating the tree-rate variables x_m (for example, by Fourier-Motzkin elimination [25]).

For the concave optimization problem shown in (3), interior-point and simplex based algorithms can be applied to solve the problem in a centralized manner [26]. However, centralized solutions may put a huge burden on the central solver and it requires the central solver to know the up-to-date topology, peer uplink rates, cross traffic, and the utility function of each peer. Tracking these information may not be feasible in practice and it is therefore desirable to have a distributed algorithm that can be deployed in practice.

III. Distributed Algorithms for Multi-tree Based Multicast

The optimization problem we consider in (3) is a non-strictly concave one due to multi-tree routing between sources and their receivers. There are three ways to approach such a problem in a distributed manner, namely Primal algorithms, Dual algorithms, and Primal-dual algorithms.

Due to the non-strictly concave objective function, standard Dual gradient algorithms fail to work since the gradient is not everywhere defined. Alternatively, dual subgradient algorithms [10] [13] and dual proximal algorithms [10] [8] are proposed to solve the problem. However, convergence of dual variables in these approaches are typically slow, and to recover

TABLE I
COMPARISONS OF APPROACHES TO ACHIEVE MULTICAST RATE REGION

	single source multicast (P2P topology)	single source multicast (general topology)	multi-source multicast (P2P topology)	multi-source multicast (general topology)	complexity
network coding	optimal	optimal	? (open)	? (open)	polynomial
packing Steiner trees	optimal	suboptimal	optimal in certain cases	suboptimal	$\mathcal{NP}$-hard in general, polynomial in P2P

optimal primal variables requires solving another optimization problem. Furthermore, it is not clear how to implement these algorithms on today's Internet.

The advantages of Primal algorithms lie in their ease of applicability and fast convergence in multi-path/multi-tree routing scenarios. However, in general it gives only an approximated solution to the optimization problem. Moreover, their practical implementation typically relies on observing packet loss due to buffer overflow in routers, which means the buffer is full and the queuing delay is large. This may not be desirable for real-time P2P conferencing applications that prefer minimum end-to-end delay.

In this paper, we focus on Primal-dual algorithms. The advantage of our Primal-dual algorithm are two folds. First, it can be implemented by utilizing the delay measurements between peers, which makes it particularly attractive in the targeting conferencing applications. Second, we show that our Primal-dual algorithm converges exponentially fast.

A. A Queuing Delay Based Primal-dual Algorithm

Another way to solve the optimization problem in (3) in a distributed manner is to look at its Lagrangian:

$$L(x,p)=\sum_{s\in S}|R_s|U_s\left(z_s\right)-\sum_{h\in H}G_h(y_h)-\sum_{j\in J}p_j\left(y_j-C_j\right), \tag{4}$$

where p_j is the Lagrangian multiplier, and can be interpreted as the price of using link j. Since the original problem in (3) is a concave optimization problem with linear constraints, strong duality holds and there is no duality gap. Any optimal solution of the problem in (3) and one of its corresponding Lagrangian multiplier is a saddle point of L over the set $\{x\geq 0, p\geq 0\}$, and vice versa. Further, (x,p) is one such saddle point of L if and only if it satisfies the Karush-Kuhn-Tucker conditions [26]: $\forall s\in S,\forall m\in s,\forall j\in J$,

$$p_j\geq 0,\ \ y_j\leq C_j,\ \ p_j\left(y_j-C_j\right) = 0, \tag{5}$$

$$|R_s|U_s'(z_s)-\sum_{h\in m}b_h^m G_h'(y_h)-\sum_{j\in m}b_j^m p_j = 0. \tag{6}$$

The first equation is the complementary slackness condition. The optimal Lagrangian multiplier can be nonzero only if the capacity constraint of link j is activated, i.e., $y_j=C_j$. We denote the set containing all (x,p) that satisfy the above conditions by E. As the original problem has at least one solution, E contains at least one point and is not empty.

There could be multiple saddle points of L since the original optimization problem is not strictly concave. To pursue one of the saddle points, we consider the following Primal-dual algorithm: $s\in S,\forall m\in s$, and $j\in J$,

$$\dot{x}_m = k_m\left(|R_s|U_s'(z_s)-\sum_{h\in m}b_h^m G_h'(y_h)-\sum_{j\in m}b_j^m p_j\right)_{x_m}^{+} \tag{7}$$

$$\dot{p}_j = \frac{1}{C_j}\left(y_j-C_j\right)_{p_j}^{+}, \tag{8}$$

where k_m is a positive constant controlling the adaptation rate of tree m and $\left(y_j-C_j\right)_{p_j}^{+}=y_j(t)-C_j$ if $p_j>0$, and is $\max(0,y_j-C_j)$ otherwise. It is known that p_j adapted according to (8) can be interpreted as queuing delay [27]. Every saddle point of L is an equilibrium of the above system in (7)-(8).

Whether the Primal-dual algorithm can be applied to multi-path/multi-tree routing scenarios is an open problem. Served as a negative result, it is shown that (x,p) following (7)-(8) oscillates indefinitely in a simple multi-path unicast scenario [9, Section 2.5].

In this paper, we give a general sufficient condition for the Primal-dual algorithm in (7)-(8) to converge to the optimal solution, regardless of unicast or multicast, single path or multipath routing. To our best knowledge, this is the first attempt to characterize the applicability of the Primal-dual algorithm. We believe its applicability is beyond the P2P systems we studied in this paper.

We give the definitions and notations to be used in later analysis. Let A be the connectivity matrix, where the (i,j) entry is the number of branches of tree j passing through link i. This is different from traditional connectivity matrix (for unicast) as its entries can take values other than 1 or 0. Similarly, let A_H be the helper connectivity matrix whose entries being the number of branches of a tree passing through a helper. Let $K=\mathrm{diag}\{k_m, m\in s, s\in S\}$, $C=\mathrm{diag}\{C_j, j\in J\}$ where J is assumed to contain only the bottlenecks without loss of generality. Let B be the matrix representing the relation of source rate, rate passing through helpers and the tree rate, with the (i,j) entry being 1 if tree j belongs to source i, being b_i^j if tree j passes through helper i, and 0 in any other cases.

The following Lemma shows that the nonlinear system in (7)-(8) converges to an invariant set, over which the nonlinear system turns into a *linear* one.

Lemma 1: All (x,p) trajectories of the system in (7)-(8) converge to an invariant set, denoted by $V_0=\{(\bar{x},\bar{p}) : \left[\bar{z},\bar{y}^H\right]^T=B\bar{x}=\text{const}\}$, over which the following is true:

- $\bar{z}$ and $\bar{y}^H$ are the unique solution to the problem in (3);
- the nonlinear system reduces to a linear one:

$$\begin{cases}\dot{\bar{x}}=KU'-KA_HG'-KA^T\bar{p}\\ \dot{\bar{p}}=C^{-1}A\bar{x}-\mathbf{1}\end{cases} \tag{9}$$

where U' and G' are constant matrices;

- the above linear system is marginally stable, and all its trajectories do not converge and form limit cycles.

Proof: Refer to Appendix B. ■

Shown by the above theorem, (x, p), trajectories of the system in (7)-(8) converge to a set V_0 where the source rates $\bar{z}$ are optimal. Clearly, all saddle points of L belong to V_0, and $E \subseteq V_0$. If we also have $V_0 \subseteq E$, then the Primal-dual algorithm solves the problem in (3).

However, it is possible that V_0 contains some $(\bar{x}, \bar{p})$ that are not in E; $\dot{\bar{x}}$ and $\dot{\bar{p}}$ are not zero. If (x, p) moves onto these points, then they will keep oscillating and never converge. This is exactly the challenge of using the Primal-dual algorithm in multi-path/multi-tree routing scenarios, and explains the oscillations in rates and delay discussed in [9].

One way to guarantee $V_0 = E$ is to utilize the fact that $B\bar{x}$ is constant to explore the conditions for V_0 to not include those singular points, as explored in the following theorem.

Theorem 2: All trajectories (x, p) of the system in (7)-(8) converge globally asymptotically to one of its equilibria and $V_0 = E$, if $\bar{p}$ is completely observable from $(\bar{z}, \bar{y}^H)$ through the linear system in (9). Equivalently, $V_0 = E$ if for any eigenvalue of $C^{-1}AKA^T$, denoted by λ,

$$\text{rank} \begin{pmatrix} C^{-1}AKA^T - \lambda I \\ BKA^T \end{pmatrix} = |J|. \tag{10}$$

Proof: Refer to Appendix C. ■

Furthermore, we can access a stronger convergence result for the Primal-dual algorithm in (7)-(8), if the above condition is satisfied.

Theorem 3: If the Primal-dual algorithm in (7)-(8) converges globally asymptotically, then the following is also true: there exists a compact set Ω such that any compact set containing Ω is a positive invariant set of the system in (7)-(8). Further, if (x, p) are bounded within one such compact set, then the system trajectories (x, p) converge to the equilibria globally exponentially.

Proof: Due to space limitation, we omit the details and refer interested readers to [28]. ■

The Primal-dual algorithm described in (7)-(8) can be implemented by each link generating its queuing delay and each source adjusting the rates of its trees by observing sum of the queuing delays introduced by using the trees. As such, the algorithm can be implemented in a distributed manner.

Regardless of the nice convergence properties and the easy implementation of the Primal-dual algorithm in (7)-(8), it is possible that with some network settings, the condition in (10) is not satisfied and the Primal-dual algorithm may not converge. One example is shown in [9, Section 2.5].

Interestingly, the unique structure of the P2P topology allows us to prove that the sufficient condition can be easily satisfied for the targeting multi-party conferencing systems, as explored in the following subsection.

1) Multi-party Conferencing Scenarios: Consider a P2P multi-party conferencing system with the first n_s of them being participants and the rest n_h of them being helpers. Every participant wants to receive contents from all other participants. The following theorem gives a sufficient condition for the Primal-dual algorithm in (7)-(8) to converge to the saddle points of L in P2P multi-party conferencing systems.

Theorem 4: For multi-party conferencing systems in P2P topology, all (x, p) trajectories of the system in (7)-(8) converge to one of its equilibria globally asymptotically, if for source s, all its $k_m, m \in s$ are the same.

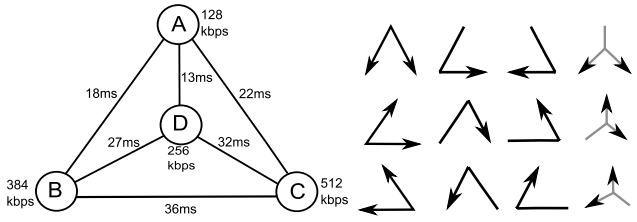
Proof: Refer to Appendix D. ■

The requirement of having all k_m to be the same for all $m \in s$ is easy to satisfy in practice. It implies every source should adjust its tree rates at the same adaptation rate, which is also convenient.

IV. Practical Implementation

We implemented a prototype of a P2P multi-party conferencing system. In such a system, each participant peer is a source of audio and video streams and at the same time wants to receive videos from all other participants[1]. Some peer nodes not interested in sending and receiving videos, such as the MCU, may decide to become a helper and assist in the audio and video delivery.

Implementing the Primal-dual algorithm in (7)-(8) appears to be straightforward. We first describe the functionality implemented by each peer, then highlight four important issues we addressed in the implementation. First, we *empirically* find one common utility function to use in video conferencing, which can be modeled by logarithmic functions. Second, we we explain how to measure and collect queuing delays along the trees without global time synchronization and how we define the cost of using a helper for content delivery by using delays. Finally, we discuss how to bound the experienced delay to meet the tight delay requirements of real-time conferencing.



(a) (b)

Fig. 3. Scenarios 1 and 2: (a) Propagation delays and uplink bandwidth constraints for each peer and link in our experimental testbed. (b) Multicast trees for three nodes (A, B, C) and one helper (D) from Section II-B.

A. Peer Functionality

In our implemented P2P multi-party conferencing system, all peers, including both participants and helpers, perform the following functions:

- Peers on a multicast tree forwards the packets from the upstream tree branches to the downstream branches. It achieves this by building a forwarding table, which maps a tree number contained in every packet to a list of its downstream peers. For instance, the helper D in Fig. 3(a) makes two copies of every packet it receives from one

[1]The audio stream rate is constant and typically small compared to the video stream rate. In practice, audio streams are transmitted to a Media Control Unit (MCU) and delivered to peers by this central server. As such, only transmission of video streams needs to be considered.

peer (A, B, or C), and unicasts the copies to the other two peers. If a peer is the leaf node in a tree, it doesn't forward any packets on that tree.
- Peers on multicast trees measure queuing delay between two peer nodes for each packet they receive and distribute these measurements to the roots of the trees. This is done in a simple and effective way that will be discussed later.

In addition, each participant peer, i.e. non-helper node, performs several functions:
- Peers, as sources of the video streams, are roots of several multicast trees which are used to deliver the videos to other peers. They need to encode their videos at specified rates, split the encoded data among the trees and send data packets with routing information (i.e., a tree number) along the trees. Each peer sets up its trees at the beginning of the conference and adjusts them when peers join or leave.
- Peers adjust the tree rates according to (7) based on the queuing delay measurements they receive from other peers.
- All peers upon receiving the data decode the video streams of other peers and show to the user.

B. Queuing Delay Measurement and Updating of Tree Rates

We use UDP protocol to transmit video packets along tree m according to tree rate $x_m(t), m \in s$. Aside from the video content, each packet contains a tree number and a timestamp, representing the time when this packet was sent from the current peer node, such that next-hop peer node can measure the delay between the two peer nodes.

Seen from (7)-(8), the key in implementing the Primal-dual algorithm is to measure the queuing delay p_j of peer j's uplinks, for all $j \in J$. Under the setting that peer uplinks are the only bottleneck in P2P systems, the end-to-end queuing delay between peer j and its offspring peers on multicast trees is equal to the queuing delay p_j of the peer j's uplinks. Therefore, we can measure p_j by measuring the end-to-end delay between peer j and its offspring peers. To ensure a fully distributed solution, it is desirable to carry out such end-to-end delay measurement without global synchronization across peer nodes.

In our implementation, we use the difference in relative One-Way-Delay (ROWD) to measure the queuing delay between two peer nodes. ROWD is the relative difference between the packet sending time at the sender peer, and the packet receiving time on the receiver peer. It is the sum of propagation delay, queuing delay, and clock offset between the two peers. The smallest ROWD seen in the history between two peers simply corresponds to the sum of propagation delay and clock offset, and is constant. By subtracting the smallest ROWD seen in the history from the current ROWD, we can get an measurement of current queuing delay between two peers.

The following is a simple procedure that we follow to measure the queuing delays of peers' uplinks and to distribute them among peers.
- Whenever a peer sends or forwards a packet, a timestamp is attached to it.
- Each of its offspring peers on the tree computes the current ROWD, subtracts the minimum ROWD observed so far, and generates a measurement of the queuing delay of the sender peer's uplink.
  - If the offspring peer is a source, it will piggyback this queuing delay measurement to its next video packet which guarantees its distribution among all other peers.
  - If the offspring peer is a helper, it will piggyback this measurement with the packets being forwarded to the downstream peers. Those peers (which are always sources) distribute it to all other peers as part of the next packet they send out through one of their trees.

The overhead of distributing the delay information is negligible as it only requires few bytes per packet and it is distributed together with each peer's video.

The advantage of measuring delay based on ROWD is that it does not require time synchronization across peers. Such method is well adopted in the context of measuring congestion [29]. We use it in this paper for a different purpose of measuring queuing delay. The disadvantage of using ROWD to measure the queuing delay is that its measurement can be inaccurate if the underlying route between peers changes; there have been some efforts to overcome this drawback [29].

Upon collecting the necessary delay measurement p_j ($j \in R_s \cup H \cup \{s\}$), source peer s computes an average queuing delay for each peer on its trees, by doing a running average over the last three queuing delay measurements of the peer. The purpose of doing so is to achieve a balance between robustness to measurement noise and quick response to network condition changes. Source s then updates its tree rates x_m ($m \in s$) according to (7)[2].

It is also possible to share the helper nodes among multiple video conferences. Our proposed algorithm will automatically utilize as many resources from the helpers as possible to improve the aggregate video quality across conferences. How much server resource a conference receives depends on the number of participating peers in the conference as well as the utility functions involved in the conference.

C. Utility (PSNR) Modeling

In video processing, PSNR (peak-to-peak signal-to-noise ratio) metric is the de facto standard criterion to provide objective quality evaluation between the original frame and the compressed one. For the original video frame f_1 and the compressed one f_2, each containing $N \times N$ pixels with values in $[0, 255]$, the PSNR is computed as follows:

$$PSNR(f_1, f_2) = 10 \log_{10} \left[\frac{255^2 \times N^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} (f_1^{ij} - f_2^{ij})^2} \right],$$

where f_1^{ij} and f_2^{ij} are the pixel values in i-th row and j-th column of frames f_1 and f_2, respectively.

Fig. 4 shows PSNR curves of three videos as functions of encoding rates. These represent the receiving video quality if a

[2]In the theoretical analysis we assume the rates are followed exactly, i.e., the variable bit rate (VBR) encoder generates the video stream at the current source rate as the sum of aggregate tree rates.

source peer encodes and sends its video at these rates. We have chosen three video sequences in CIF format: Akiy, Foreman, and Tennis. They represent low, medium, and high motion scenes, respectively. We encoded the videos by H.264/AVC Reference Software Encoder (ver. 12.2) [30] with various bitrates to get PSNR curves.

Interestingly, from Fig. 4, we observed that PSNR of a video coded at rate z can be approximated by a logarithmic function $\beta \log(z)$, with higher β for videos with large amount of motion and lower β for rather still scenes. The β value can be obtained from the video encoder of source s during the encoding. Based on this empirical observation, we use utility function $U_s(z_s) = \beta_s \log(z_s)$ in our experiments.

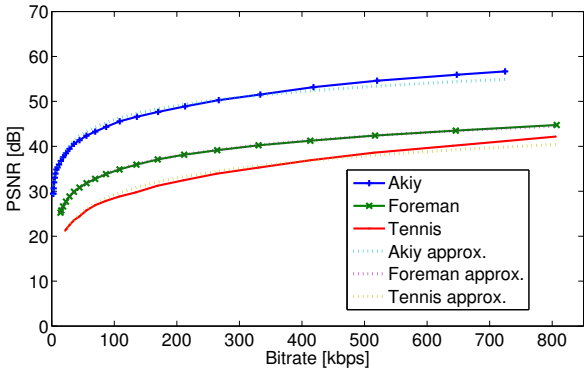


Fig. 4. PSNR curves of three video sequences with low, medium, and high motion and their approximation by logarithmic functions.

D. Implementing Helper Cost Function Using Delay

In order to implement the Primal-dual algorithm in (7)-(8) in practice, we need to define helper cost functions, i.e. $G_h(y_h)$, that are meaningful, easy to generate, and cause little or no overhead to update source peers of its value when y_h changes. Having these goals in consideration, we use *delay* based functions in our implementation.

One example is as follows: for all helper nodes $h \in H$, $G_h(y_h) = \gamma_h y_h^2$, where γ_h is assumed to be a small positive constant. Its derivative is: for all $h \in H$, $G'_h(y_h) = \gamma_h y_h$.

The idea is to implement it as an additional *artificial* delay that helper h injects when forwarding every packet. When peers compute the queuing delay for packets received from a helper node h, they add a small amount of $\gamma_h y_h$ to it. This artificial portion of delay will be then distributed back to the source peers. In this way, the source peers will naturally take this $\gamma_h y_h$ cost into account when adjusting the tree rates according to (7).

The above process can be carried out for arbitrary helper cost functions. In the P2P multi-party conferencing systems we consider, delay is crucial for the quality of participants' experience. As such, it is reasonable to use delay as a cost function. The cost generation and distribution is fully distributed and because it is included in the existing queuing delay measurements it adds no extra overhead.

E. Bounding the Average Queuing Delay At the Equilibrium

On one hand, our solution uses only depth-1 and depth-2 Steiner trees to deliver contents from a source peer to its receiver peers. Consequently, every packet goes through at most one hop (i.e., two tree branches) in the overlay before reaching all receivers, resulting in low end-to-end *propagation* delay in packet delivery. This makes our solution especially suitable for real-time P2P multi-party conferencing systems, as conferencing systems have a strict requirement, in particular 150 ms, on the end-to-end packet delivery delay between participant peers.

On the other hand, one solution relies on the queuing delay experienced by packets to control the tree rates properly. As queuing delay also contributes to the end-to-end packet delivery delay, it is then desirable to bound the queuing delay experienced by packets at the steady state of the system after the tree rates converge. Note that according to Theorem 4 and 3, tree rates converge exponentially fast in P2P multi-party conferencing systems.

Let $(\bar{x}, \bar{p})$ be the converged tree rates and queuing delays, and let $\bar{z} = B\bar{x}$, $\bar{y}^H = A_H \bar{x}$ and $\bar{q} = A^T \bar{p}$. Let $\bar{d}_m$ be the average queuing delay in packet delivery from source s to its receiver peers along tree m at the equilibria. The following proposition states the relationship of $\bar{d}_m$ and utility functions:

Proposition 1: The following optimization problem, with α being a positive constant, has the same solution as the one in (3):

$$\max_{\{x_m\}} \quad \alpha \left[\sum_{s \in S} |R_s| U_s \left(\sum_{m \in s} x_m \right) - \sum_{h \in H} G_h(y_h) \right] \quad (11)$$
$$s.t. \qquad y_j \le C_j, \ \ \forall j \in J,$$

Meanwhile, at the equilibria of the above system, for all $m \in s$, we have

$$\bar{d}_m \le 2\alpha U'_s(\bar{z}_s). \quad (12)$$

Proof: Refer to Appendix E. ■

As such, we can bound $\bar{d}_m$ with a designed value by tuning the constant α based on a lower bound on z_s for all $s \in S$. For example, for P2P multi-party conferencing system, the system designer may want to set a limit on how low the converged source rate can be, since the video quality will be unacceptable at such limited rate[3]. This will give a lower bound on z_s, and hence a lower bound value of $U'_s(\bar{z}_s)$ for all $s \in S$. Then the designer can bound the *worst-case* $\bar{d}_m$ with a designed value, e.g., 150 ms, by solving α according to (12) with these two values. It should be noted that in practice the converged source rate is larger than the video rate limit, the experienced $\bar{d}_m$ will be therefore smaller than the worse-case bound set above.

V. Experimental Results

We use PCs running Windows XP and Network Emulator for Windows (NEW) connected to a LAN for Scenarios 1 and 2. NEW is a software based network emulator that allows realistic emulation of different network characteristics such as bandwidth emulation, packet loss and latency [31]. Scenarios

[3]In such case, the system designer might want to involve stronger helpers with large uplink bandwidths to allow higher video rates.

1 and 2 use the topology and network conditions described by Fig. 3(a).

We have also conducted experiments in Scenario 3, under real Internet settings with peers being spread around the US and virtual machines in a Virtual Lab [32], respectively.

A. Scenario 1: X-Traffic and Utility Changes

In the first experimental scenario we show how our system adapts to network dynamics (i.e., cross traffic) and utility changes. We have three peers A, B, C initialized with the same utility function, i.e., parameter $\beta_A = \beta_B = \beta_C$. At time 200 seconds β_B is increased by 50%. At time 400 seconds we start sending additional traffic of 200 kbps from peer B and stop at 600 seconds.

Fig. 5(a) through 5(i) show the rates and total queuing delays for each tree in the system. As seen in Fig. 5(a), peer A does not utilize its depth-1 direct tree, because it requires twice as much bandwidth of peer A compared to sending content through other peers and peer A has the lowest bandwidth capacity. Moreover, other peers are not utilizing trees in Fig. 5(e) and 5(f) in order to avoid excessive congestion at peer A and to allow it to fully use its upload bandwidth for trees going through other peers (Fig. 5(d) and 5)(g) to distribute A's video.

In the introduction, we mentioned how ad hoc simulcast approach would not perform well in a scenario where a peer has very low upload bandwidth. Scenario 1 has one such peer, A, which would encode its stream at only 64 kbps if simulcast approach was used. Remember that with simulcast every peer splits its uplink bandwidth equally among all receivers, for A these are B and C. During the first 200 seconds of this experiment simulcast would achieve utility of only up to 14.96, while our scheme achieves the optimal utility value and the encoding rate for peer A is twice as high. This shows that our framework improves the aggregate utility further by optimally utilizing bandwidth of all peers.

The sending rate of peer B starts to increase at time 200 seconds as its utility function becomes steeper, indicating the conference participant starts to introduce a large amount of motion in its video. Specifically, the rate of tree in Fig. 5(h) increases at the expense of peer C (Fig. 5(h)) which has lower utility. All peers are using peer C, the peer with the maximum bandwidth, as can be observed from Fig. 5(g) and 5(h). The cross traffic at peer B initiated at 400th second causes a decrease in rates for trees in Fig. 5(b) and 5(i) as peer A stops using congested peer B and peer B decreases utilization of the direct depth-1 tree. The system always quickly converges to one of the optimal solutions after network conditions or utility function changes (in less than 20 seconds), see Fig. 5(m)).

In order to confirm the results of our distributed algorithm we run a Mosek [33] program to solve the optimization problem in (3) using the same topology and utility functions. The optimal tree rates allocation generated by Mosek confirms our above observations and the optimal utility value is shown in Fig. 5(m) and 5(o).

It takes $76ms$ on average to deliver a packet containing video from a sender to a receiver in Scenario 1 (with latencies between peers $A - B$, $B - C$ and $C - A$, 18, 36, $22ms$, respectively, described in Fig. 3(a), and queuing delays from Fig. 5). If we distributed the videos in a simulcast way the average delay would be $27ms$ but the maximum utility would not be possible to achieve. We see that the proposed algorithm incurs very little queueing delay in the system.

B. Scenario 2: Peer Joining and Helper

In the second experimental scenario (Fig. 5(n)), the three peers are sending videos with various motion characteristics ($\beta_B = 0.9\beta_A$, $\beta_C = 1.2\beta_A$, $\beta_D = \beta_A$). A fourth peer (D) joins the group, first as a source&receiver peer at time 200 seconds, and as a helper at time 400 seconds. When it becomes a helper, it is no longer generating its own video stream and is not interested in receiving the videos from other peers but it just helps forwarding the video content to them.

In this scenario we see that the system adapts the sending rates quickly to accommodate the new joining peer at time 200 seconds. Maximum utility is achieved within 30 seconds and note that the convergence rate can be controlled by the k_m parameter in (7). As each video has to be delivered to more peers, we see a drop in the total rates. The system adapts again as the peer becomes a helper at time 400 seconds, where the rates react to fully utilize the available bandwidth and maximize the utility function (Fig. 5(o)). Note that with the helper, rate of each source monotonically increases and the converged utility is higher than the one without helper, i.e., before second 200 seconds.

C. Scenario 3: Internet Experiment - 3 Peers

In this scenario we run a short 2-minute 3-party conferencing over the Internet using 3 computers spread around the US. In this case, every peer will use 3 spanning trees to deliver its contents. Uplink bandwidth limits are 384 kbps for peer A, 256 kbps for peer B, and 128 kbps for peer C. The utility functions for all peers are set to be the same. The average round trip time between peers are: 79 ms between A and B, 40 ms between A and C, and 65 ms between B and C. Fig. 6 shows the source rates, tree rates and average tree branch delays for each peer. Fig. 6 also shows the utility achieved in the experiments as well as theoretical optimum. We use the same k_m for all the trees of a peer. As such, we use k_A, k_B and k_C to denote the tree rates adaptation speeds for A, B, and C, respectively.

Seen from Fig. 6(a), the source rate of A ramps up fastest among the three peers, this is because we set k_A to be the largest among the three. Similarly, the source rate of C ramps up slowest since k_C is the smallest among the three.

We observe that the queuing delay varies as the programs adjust the tree rates. We also observe from Fig. 6(a) that the average tree branch delays for A, B and C are about 19 ms, 20 ms, and 45 ms, respectively. Shown in Section IV-E, the average packet delivery delay is approximately twice the sum of the average one way propagation delay and the average tree branch delay. Therefore, the average packet delivery times for A, B and C are about 91 ms, 105 ms, and 128 ms, respectively. These values are within the acceptable range for smooth conferencing experience.

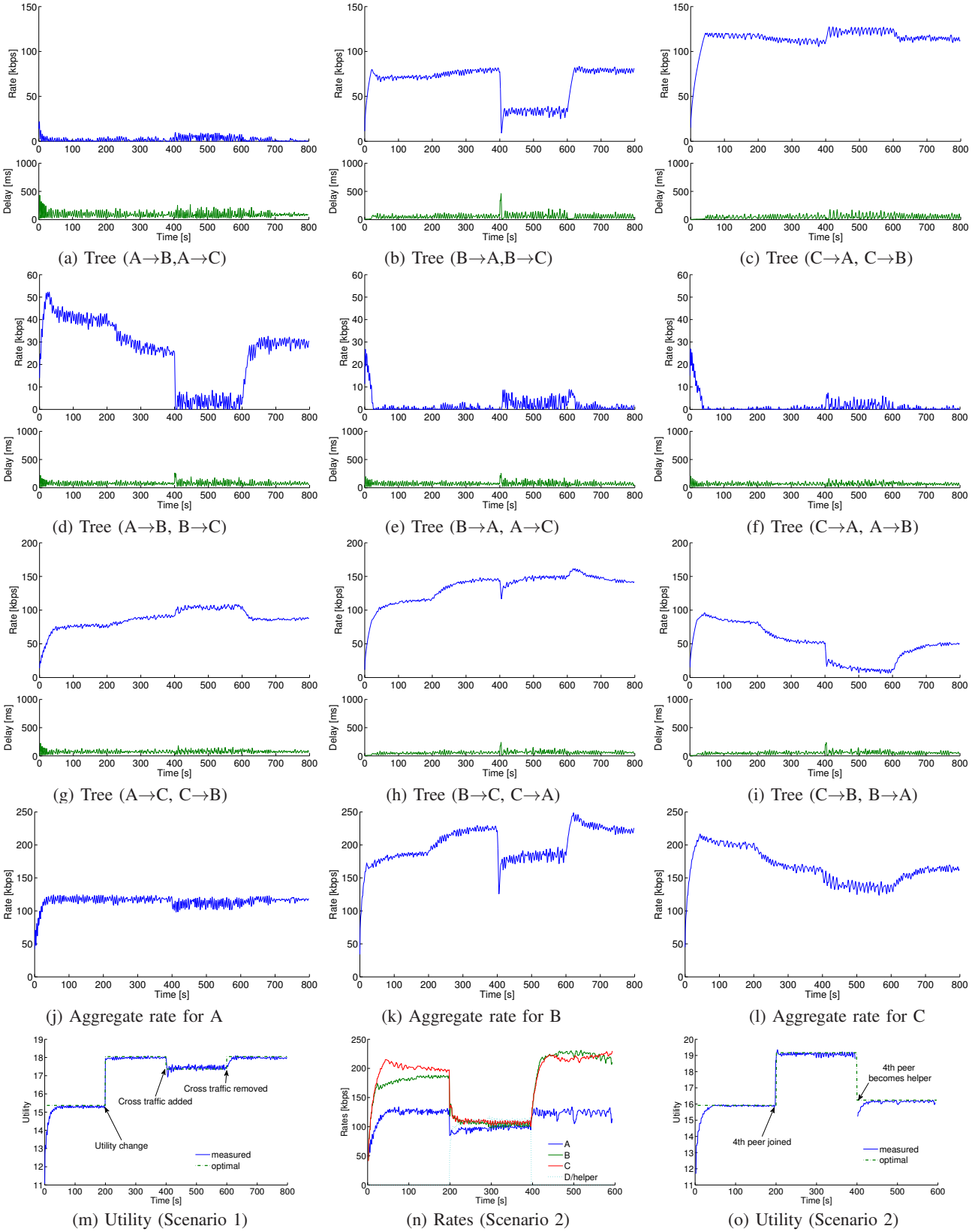


Fig. 5. Scenario 1 – (a) through (i): Sending rates and total delays for trees with edges and topology shown in Fig. 3(a). (j) through (l): Coding rates of each video nodes A, B, and C are sending. (m) Utility value achieved compared to the optimum. Scenario 2 – (n) and (o): Coding rates and utility values.

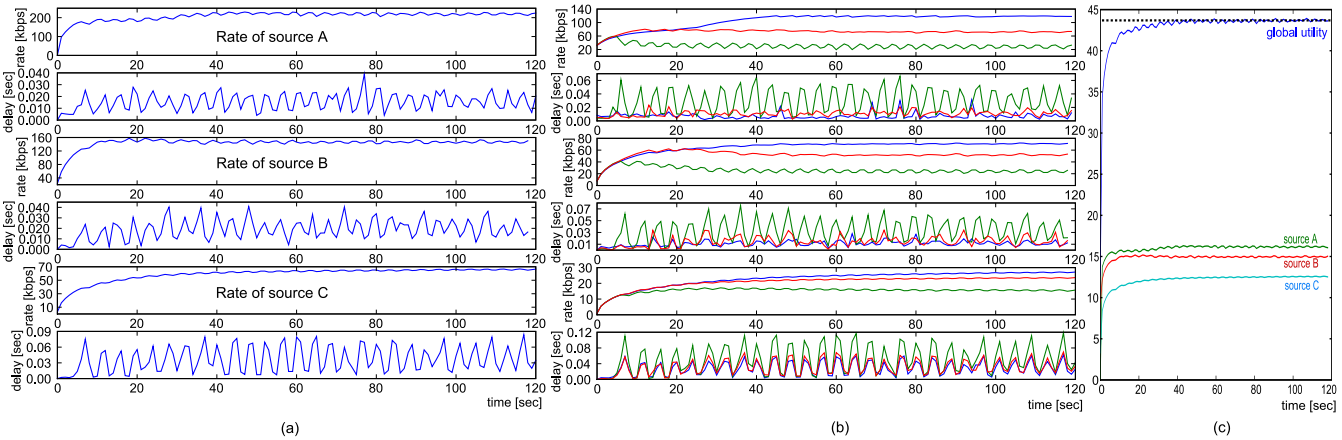


Fig. 6. Experimental results for Scenario 3: (a) Source rates of A, B, and C, respectively, with the average tree branch delays. (b) Tree rates for trees of A, B and C, respectively, with the tree branch delays. (c) The aggregate utility achieved by the system, and the utilities per source.

VI. Conclusion and Future Work

We investigate the multi-source multicast utility maximization problem in P2P systems. The nature of P2P topologies allows us to tackle difficulties arising in the general network case in a surprisingly elegant manner. We show that routing along a linear number of trees per source can achieve the same rate region as that obtained through (inter-session) network coding. We develop a new multi-tree routing formulation for the multicast utility maximization problem. It not only eliminates some mathematical difficulties associated with previous formulations, but also leads to practical solutions. We further develop a Primal-dual distributed algorithm to maximize the aggregate utility. We propose a sufficient condition to evaluate convergence of the Primal-dual algorithm in multi-path routing scenarios, and prove its global exponential convergence under different P2P scenarios we studied. Our approach naturally accommodates helper nodes within the optimization framework. The developed algorithms are practical and easy to implement in a P2P overlay over the current Internet. Experimental results over both testbed and the Internet show that our solution converges quickly to the optimal utility, and re-optimizes itself after network conditions or utility function change. It is also resilient to peer nodes joining and leaving over time. Its scalability is also studied.

We are investigating the scalability of our solution in large P2P systems. The scalability of our current solution is limited by the fact that branching out-degree of multicast trees used is linear in the number of receivers. We extended this framework to apply to multi-rate multicast where different receivers can receive the same video at different bitrates using scalable coding [34].

References

[1] T. Liu, Y. Wang, J. Boyce, Z. Wu, and H. Yang, "Subjective quality evaluation of decoded video in the presence of packet losses," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2007, pp. 1125–1128.

[2] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness, and stability," *Journal of the Operationl Research Society*, pp. 237–252, 1998.

[3] S. H. Low and D. E. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, no. 6, pp. 861–875, Dec. 1999.

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

[5] F. P. Kelly, "Fairness and stability of end-to-end congestion control," *European Journal of Control*, pp. 159–176, 2003.

[6] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition:a mathematical theory of network architectures," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[7] H.Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the internet," *IEEE/ACM Trans. Networking*, 2006.

[8] X. Lin and N. B. Shroff, "Utility maximization for communication networks with multi-path routing," *IEEE Trans. Automat. Contr.*, 2006.

[9] T. Voice, "Stability of congestion control algorithms with multi-path routing and linear stochastic modelling of congestion control," Ph.D. dissertation, University of Cambridge, Cambridge, UK, May 2006.

[10] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization based rate control for multirate multicast sessions," in *IEEE INFOCOM*, Anchorage, 2001.

[11] S. Deb and R. Srikant, "Congestion control for fair resource allocation in networks with multicast flows," *IEEE Trans. Automat. Contr.*, no. 2, pp. 274–285, Apr. 2004.

[12] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multi-cast with network coding," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[13] Y. Wu and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: a shortest path approach," *IEEE J. Select. Areas Commun.*, no. 8, pp. 1475–1488, Aug. 2006.

[14] D. S. Lun, N. Ratnakar, M. M. edard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Trans. Inform. Theory*, no. 6, June 2006.

[15] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: an efficient mechanism for content distribution in a p2p network," in *Proceedings of Acm Sigcomm Asia Workshop*, Beijing, China, Apr. 2005.

[16] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, no. 4, July 2000.

[17] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhiuzen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. on Info. Thy.*, vol. 51(6), 2005.

[18] X. Yan, R. W. Yeung, and Z. Zhang, "The capacity region for multi-source multi-sink network coding," in *2007 IEEE International Symposium on Information Theory (ISIT2007)*, Nice, France, June 2007.

[19] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, Jan. 2003.

[20] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for p2p streaming systems," in *Proc. IEEE INFOCOM*, Anchorage, 2007.

[21] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, "Can network coding help in p2p networks?" in *Proc. of IEEE NetCod*, Boston, 2006.

[22] L. Massoulié, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *INFOCOM*, 2007, pp. 1073–1081.

[23] J. Edmonds, "Edge-disjoint branchings," *Combinatorial Algorithms, R. Rustin, ed.*, pp. 91–96, 1973.

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

[25] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997.
[26] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.
[27] 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.
[28] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility Maximization in Peer-to-Peer Systems," *Microsoft Research Technical Report*, August 2007.
[29] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM Trans. Networking*, vol. 11, pp. 537–549, Aug. 2003.
[30] H. Kalva, "The h.264 video coding standard," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 86–90, Oct. 2006.
[31] Z. Ni, "Network Emulator for Windows/CE," Microsoft Research Asia, Internal documentation.
[32] V. Padman and N. Memon, "Design of a virtual laboratory for information assurance education and research," in *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security*, NY, 2002.
[33] MOSEK ApS, "MOSEK Optimization Software." [Online]. Available: http://www.mosek.com/products_mosek.html
[34] M. Ponec, S. Sengupta, M. Chen, J. Li, and P. A. Chou, "Multi-rate peer-to-peer video conferencing: A distributed approach using scalable coding," in *IEEE International Conference on Multimedia & Expo (ICME)*, June 2009.
[35] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhäuser, 2004.
[36] F. Callier and C. Desoer, *Linear System Theory*. Springer-Verlag, 1991.

APPENDIX

A. Proof of Theorem 1

Proof: Let $In(v)$ and $Out(v)$ be the set of incoming and outgoing edges of node v, respectively. Assume every edge e in E have infinite capacity. Let $f(e) : e \to [0, \infty)$ be the rate assigned to edge e in E. We denote $f(E) = [f(e), e \in E]$ as the edge rate vector defined on E. The node upload capacity constraints are expressed as follows: $\sum_{e \in In(v)} f_z(e) \leq C_o(v)$.

Let $z = (z_s, s \in S)$ be a rate vector achieved by assigning a set of edge rates to E and perform scheduling, routing and coding. For one z, there could be many $f(E)$ that can achieve z. Define a set $F_z(E) = \{f(E) : z \text{ is achieved by } f(E)\}$.

Definition: A edge rate vector $f_z^*(E) \in F_z(E)$ is irreducible with respect to z, if there does not exist $f_z'(E) \in F_z(E)$ s.t.

$$\sum_{v \in V} \sum_{e \in Out(v)} f_z'(e) < \sum_{v \in V} \sum_{e \in Out(v)} f_z^*(e).$$

Irreducible edge rate vector represents the most efficient edge rate assignment that achieves z. There can be multiple irreducible edge rate vectors in $F_z(E)$.

Under the problem setting of Theorem 1, we have $R_s \cup \{s\} = R$ and $H_s = H$ for all s in S. Let $N = |R| - 1$.

The following Lemma describes an important property of any irreducible edge rate vectors, under our settings in this section.

Lemma 2: For any feasible rate vector z, let $f_z^*(E)$ be irreducible with respect to z. We have

$$\sum_{h \in H} \sum_{e \in Out(h)} f_z^*(e) \leq N \sum_{h \in H_s} \sum_{e \in In(h)} f_z^*(e).$$

Proof: Suppose it is not true, then

$$\sum_{h \in H} c_{out}(h) \geq \sum_{h \in H} \sum_{e \in Out(h)} f_z^*(e) = M \sum_{h \in H} \sum_{e \in In(h)} f_z^*(e) \tag{13}$$

for some $M > N$.

We now show $f_z^*(E)$ can not be irreducible, by constructing a $f_z'(E)$ as follows:

- for all e between any two nodes in S, let $f_z'(e) = f_z^*(e)$;
- for all e between any two nodes in H, let $f_z'(e) = 0$;
- for all r in R that originally send information to H in the assignment $f_z^*(E)$, let them send the same amount of information $\sum_{h \in H} \sum_{e \in In(h)} f_z^*(e)$ to H. This is done by filling every helper h in H one by one, each at an amount of at most $\frac{1}{N} c_{out}(h)$. Such operation is doable according to (13). Each helper then replicates the bits it receives from node r in R, and broadcast to all other nodes in R.

By carrying out all the above steps, all r in R in the assignment $f_z'(E)$ have same information from other nodes in R as in the assignment $f_z^*(E)$. Further, all r in R also have all the information that entering H from R in the assignment $f^*(E)$. If r originally gets coded packets from helpers in H in the assignment $f_z^*(E)$, then it can generate the same coded packets by itself in the assignment $f_z'(E)$. As such, all r in R in the assignment $f_z'(E)$ also have same information from helper nodes in H as in the assignment $f_z^*(E)$.

Therefore, z are still achieved by $f_z'(E)$, and for

$$\sum_{h \in H} \sum_{e \in Out(h)} f_z'(e) = N \sum_{h \in H} \sum_{e \in In(h)} f_z^*(e) < \sum_{h \in H} \sum_{e \in Out(h)} f_z^*(e).$$

Summing the rates of all outgoing edges of all nodes for $f_z'(E)$ and $f_z^*(E)$, we have

$$\sum_{v \in V} \sum_{e \in Out(v)} f_z'(e) < \sum_{v \in V} \sum_{e \in Out(v)} f_z^*(e).$$

As such, $f_z^*(E)$ is not irreducible, leading to contradiction. ■

Now we show packing Mutualcast trees is optimal under the problem settings. Given an achievable rate vector $z = (z_1, \ldots, z_{|S|})$, it must be achievable by at least one irreducible rate vector. Denote one of such vector as $f_z(E)$.

Considered a receiver node r in R, its aggregate consumption rate must be no more than the sum of $f_Z(e)$ over all edges e in $In(r)$. That is, $\sum_{s: r \in R_s} z_s \leq \sum_{e \in In(r)} f_Z(e)$. Note this is true even if network coding is allowed in information transmission. Summing over all receiver nodes r in R, we have a necessary condition for z to be feasible as follows:

$$\begin{aligned}
N \sum_{s \in S} z_s &\leq \sum_{r \in R} \sum_{e \in In(r)} f_Z(e) \\
&= \sum_{v \in V} \sum_{e \in In(v)} f_Z(e) - \sum_{h \in H} \sum_{e \in In(h)} f_Z(e) \\
&= \sum_{v \in R} \sum_{e \in Out(v)} f_Z(e) + \sum_{h \in H} \sum_{e \in Out(h)} f_Z(e) \\
&\quad - \sum_{h \in H} \sum_{e \in In(h)} f_Z(e) \\
&\quad \text{(by Lemma 2)} \\
&\leq \sum_{v \in R} c_{out}(v) + \left(1 - \frac{1}{N}\right) \sum_{h \in H} \sum_{e \in Out(h)} f_Z(e) \\
&\leq \sum_{v \in V} c_{out}(v) - \frac{1}{N} \sum_{h \in H} c_{out}(h).
\end{aligned}$$

The sum of first two terms is the total system upload bandwidth, the third term is the bandwidth consumed by the helper nodes. Thus the aggregate term is the upload bandwidth available to receivers.

An outer bound for z is then given by

$$\left\{ z:\ z_s \le C_o(s), \forall s \in S;\ N\sum_{s\in S} z_s \le \sum_{v\in V} C_o(v) - \frac{1}{N}\sum_{h\in H} C_o(h) \right\}.$$

The above set is exactly the achievable rate regions of $|S|$ *independent* multicast sessions superposition on top of each other, where session s uses a set of $N+|H|+1$ Mutualcast trees to broadcast to all receivers in R at rate z_s. ■

B. Proof of Lemma 1

Proof: Let (x^*, p^*) be one point in the non-empty set E, which is a saddle point for L as defined in Section III and is an equilibrium point of the system. Similar to [9, Section 2.6] and [35, Section 3.4], we define a function V as follows:

$$V(x,p) = \sum_{m\in s, s\in S} \int_0^{x_m} \frac{(w-x_m^*)}{k_m}\, dw + \sum_{j\in J}\int_0^{p_j} C_j(w-p_j^*)\, dw.$$

Define the tree price $q = A^T p$. V's Lee derivative satisfies

$$\begin{aligned}\dot{V} &\le (p-p^*)^T(y-y^*) - (q-q^*)^T(x-x^*)\\ &+ \sum_{s\in S}\sum_{m\in s}(x_m - x_m^*)\left(|R_s|U_s'(z_s) - \sum_{h\in m} b_h^m G_h'(y_h^*) - q_m^*\right)\\ &+ \sum_{j\in J}(p_j - p_j^*)\left(y_j^* - C_j\right) + \sum_{h\in H}(y_h - y_h^*)(G_h'(y_h^*) - G_h'(y_h)).\end{aligned}$$

The first two terms simply cancel. Based on the unique structure of the optimization problem we consider, we have q_m^*, z^* and y_h^* are unique, U_s' is a decreasing function, and G' is an increasing function. We also have $(p_j - p_j^*)\left(y_j^* - C_j\right) \le 0$ since $y_j^* \le C_j$ and $p_j^* = 0$ if $y_j^* < C_j$.

Therefore, we have $\dot{V} \le 0$. By La Salle principle, the system will asymptotically converge to an invariant set in $V_0 = \{\bar{x}, \bar{p} : \dot{V} = 0\}$, which indicates $\sum_{m\in s}\bar{x}_m = z_s^*$, $\bar{y}_h = y_h^*$, and $\bar{p}_j$ can only be nonzero at link j satisfying $y_j^* = C_j$.

Observed that $\bar{z}$ and $\bar{y}^H$ are unique overV_0, the nonlinear system turns into a linear one. We consider the worst case where prices on all links are positive in the converged set, and express the linear system as follows:

$$\begin{bmatrix} \bar{z} \\ \bar{y}^H \end{bmatrix} = B\bar{x},\ \ \dot{\bar{x}} = KU' - KA_H G' - KA^T\bar{p},\ \ \dot{\bar{p}} = C^{-1}A\bar{x} - \mathbf{1},$$

where U' and G' are constant matrices. The characteristic function of the above linear system, denoted by F, is a product of a positive diagonal matrix and a skew-symmetric matrix, as follows:

$$F = \begin{bmatrix} K & 0 \\ 0 & C^{-1} \end{bmatrix}\begin{bmatrix} 0 & -A^T \\ A & 0 \end{bmatrix}.$$

Since all eigenvalues of a skew-symmetric matrix are either zero or purely imaginary, so do those of F. Consequently, trajectories of the linear system will not converge. ■

C. Proof of Theorem 2

Proof: Observed $[\bar{z}, \bar{y}^H]$ is constant, if the system state $\bar{p}$ is *complete observable* from $[\bar{z}, \bar{y}^H]$, then $\bar{p}$ can be fully determined by all orders of Lee derivatives of $[\bar{z}, \bar{y}^H]$ and must be constant. According to [36], this is true if and only if for any λ, the eigenvalue of $C^{-1}AKA^T$,

$$\begin{bmatrix} C^{-1}AKA^T - \lambda I \\ BKA^T \end{bmatrix} \text{ has rank } |J|. \tag{14}$$

If above condition is satisfied, then $\dot{\bar{p}} = 0$ because:

$$\frac{d^n\,\bar{z}}{dt^n} = 0, \frac{d^n\,\bar{y}^H}{dt^n} = 0,\ n = 1, 2, \ldots \ \ \Rightarrow\ \ M\bar{p} = \text{const.}$$

If $\dot{\bar{p}} = 0$, then $\dot{\bar{x}} = \text{const}$. Observing that $\bar{x} \ge 0$ in V_0 and z is constant, we must have $\dot{\bar{x}} = 0$. Therefore, V_0 contains only the equilibria of the system in (7)-(8). ■

D. Proof of Theorem 4

Proof: Following the setting, we have $k_m = k_s$ for all m in s. Correctness of the theorem is easy to verify for the case where $n_s = 1$. In the rest of this proof, we focus on the case where $n_s \ge 2$.

Let $K_S = diag\{k_s, s \in S\}$. In the P2P multi-party conferencing scenario we consider, we have $H \cap S = \emptyset$ and $H \cup S = V$. Let $n_s = |S|$ and $n_h = |H|$. Under the setting, every helper forwards messages from a participant to other $n_s - 1$ participants. Matrix B is an then $n \times (n \cdot n_s)$ matrix, and can be expressed as follows:

$$B = \begin{bmatrix} \mathbf{1}_{1\times n} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{1}_{1\times n} \\ (0|(n_s-1)I_{n_h}) & \cdots & (0|(n_s-1)I_{n_h}) \end{bmatrix}.$$

The connectivity matrix A is given by $A = [A_1, A_2, \cdots, A_{n_s}]$ where

$$A_i = (n_s - 2)I_n + D_i + \begin{bmatrix} 0 & 0 \\ 0 & I_{n_h} \end{bmatrix}, 1 \le i \le n_s,$$

and D_i is an $n \times n$ $1-0$ matrix with all entries of the i-th row being one and zero elsewhere.

To show that

$$\begin{bmatrix} C^{-1}AKA^T - \lambda I \\ BKA^T \end{bmatrix}$$

has rank n for any eigenvalue λ of matrix $C^{-1}AKA^T$, it is sufficient to show BKA^T has rank n. Let $k_\Sigma = \sum_{s\in S} k_s$, we give BKA^T as the following block matrix:

$$\begin{bmatrix} K_S\left[nI_{n_s} + (n_s-2)\mathbf{1}_{n_s\times n_s}\right] & (n_s-1)K_S\mathbf{1}_{n_s\times n_h} \\ (n_s-1)\mathbf{1}_{n_h\times n_s}K_S & (n_s-1)^2 k_\Sigma I_{n_h} \end{bmatrix}.$$

Its determinant is given by [4]

$$\begin{aligned}&\det((n_s-1)^2\bar{k}I_{n_h})\det\left(K_S\left[nI_{n_s} + (n_s-2)\mathbf{1}_{n_s\times n_s}\right]\right.\\ &\left.-\frac{n_h}{k_\Sigma}K_S\mathbf{1}_{n_s\times n_s}K_S\right).\end{aligned}$$

[4]Determinant of a block matrix $\begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}$ is $\det(M_4)\det(M_1 - M_2M_4^{-1}M_3)$.

We are done if the determinant is non-zero. Define a matrix Q as follows:

$$Q \triangleq nI_{n_s} + (n_s-2)\mathbf{1}_{n_s\times n_s} - \frac{1}{\bar{k}}\mathbf{1}_{n_s\times n_h}\mathbf{1}_{n_h\times n_s}K_S.$$

To show Q has full rank, it is sufficient to show that $(Q+Q^T)$ is positive definite. We notice that

$$1_{n_s\times n_h}1_{n_h\times n_s} = n_h 1_{n_s\times n_s} = U^T\Lambda U,$$

where

$$U = \begin{bmatrix} \frac{1}{\sqrt{n_s}} & \frac{-1}{\sqrt{2}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{n_s}} & 0 & \frac{-1}{\sqrt{2}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{n_s}} & 0 & 0 & \cdots & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{n_s}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \end{bmatrix},$$

and

$$\Lambda = \begin{bmatrix} \frac{1}{n_s} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

For a nonzero vector $\xi = [\xi_1, \xi_2, \cdots, \xi_{n_s}]^T$, we study $U(Q+Q^T)U^T$ as follows:

$$\begin{aligned} & \xi^T U(Q+Q^T)U^T\xi \\ = \; & 2n\xi^T\xi + 2\frac{n_s-2}{n_s}\xi_1^2 - 2\frac{n_h}{n_s^2}\frac{k_1}{k_\Sigma}\sum_{i=1}^{n_s}\xi_1\xi_i + \\ & \frac{n_h}{n_s}\frac{k_{n_s}}{k_\Sigma}\left(\xi_1\xi_{n_s} - \xi_1^2\right) \\ > \; & \frac{n_h}{n_s^2}\frac{k_1}{k_\Sigma}\left[\left(n_s^2+1\right)\xi_1^2 + \sum_{i=2}^{n_s}\xi_i^2\right] - 2\frac{n_h}{n_s^2}\frac{k_1}{k_\Sigma}\sum_{i=1}^{n_s}\xi_1\xi_i + \\ & \frac{n_h}{n_s}\frac{k_{n_s}}{k_\Sigma}\left(2\xi_1^2 + \xi_{n_s}^2\right) + \frac{n_h}{n_s}\frac{k_{n_s}}{k_\Sigma}\left(\xi_1\xi_{n_s} - \xi_1^2\right) \\ = \; & \frac{n_h}{n_s^2}\frac{k_1}{k_\Sigma}\sum_{i=2}^{n_s}\left(\xi_1-\xi_i\right)^2 + \frac{n_h}{n_s}\frac{k_{n_s}}{k_\Sigma}\left(\xi_1+\xi_{n_s}\right)^2 \geq 0. \end{aligned}$$

Thus, $U(Q+Q^T)U^T$ is positive definite, and so does $Q+Q^T$. Consequently, Q has full rank. ■

E. Proof of Proposition 1

Proof: Clearly the problem in (11) and the problem in (3) have the same optimal solution.

According to (7)-(8), at the system equilibria, for all $s \in S$ and $m \in s$, we have

$$\dot{\bar{x}}_m = k_m|R_s|\left(\alpha U_s'(\bar{z}_s) - \frac{\alpha}{|R_s|}\sum_{h\in m} b_h^m G_h'(\bar{y}_h) - \frac{1}{|R_s|}\bar{q}_m\right) = 0,$$

where $\bar{q}_m = \sum_{j\in m} b_j^m \bar{p}_j$ is the aggregate queuing delay experienced by packets passing all the branches on tree m.

In our solution, the total number of branches on a tree m is $|R_s|$ if it does not contain a helper, and is $|R_s|+1$ if it contains one (note there is at most one helper per tree in our solution). Combining with this observation and the definition of $\bar{q}_m$, the average queuing delay experienced by packets passing through a branch on tree m at the equilibria is bounded by $\frac{1}{|R_s|}\bar{q}_m$.

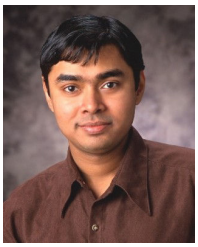
Let $\bar{d}_m$ be the average queuing delay in packet delivery from source s to its receiver peers along tree m at the equilibria. Noticing that a packet at most passes through two tree branches, we can bound $\bar{d}_m$ as follows:

$$\bar{d}_m \leq \frac{2}{|R_s|}\bar{q}_m \leq 2\alpha U_s'(\bar{z}_s) - \frac{2\alpha}{|R_s|}\sum_{h\in m} b_h^m G_h'(\bar{y}_h) \leq 2\alpha U_s'(\bar{z}_s).$$

■

Minghua Chen is currently 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.

Miroslav Ponec is currently at Akamai Technologies GmbH, Germany. He received his M.S. degree from Czech Technical University in Prague in 2005, and his Ph.D. degree from the Computer and Information Science Department at Polytechnic University in New York in 2008.



Sudipta Sengupta is currently at Microsoft Research. He received a Ph.D. and an M.S. in EECS from Massachusetts Institute of Technology (MIT), USA, and a B.Tech. in Computer Science from Indian Institute of Technology (IIT), Kanpur, India.

Jin Li is currently at Microsoft Research. He received his B.S., M.S and Ph.D. degree from the Electronic Engineering Department, Tsinghua University, Beijing, China, all with honors.

Philip A. Chou is currently at Microsoft Research. He received the BSE degree from Princeton University, Princeton, NJ, in 1980, and the MS degree from the University of California, Berkeley, in 1983, both in electrical engineering and computer science, and the Ph.D. degree in electrical engineering from Stanford University in 1988.