Utility Maximization in Peer-to-Peer Systems

Minghua Chen
Chinese University of Hong Kong
Information Engineering Dept.
Hong Kong, P.R.C.
minghua@ie.cuhk.edu.hk

Miroslav Ponec
Polytechnic University
Computer and Information Science Dept.
Brooklyn, NY 11201
mip@cis.poly.edu

Sudipta Sengupta
sudipta@microsoft.com

Jin Li
jinl@microsoft.com

Philip A. Chou
pachou@microsoft.com

Microsoft Research, Communication and Collaboration Systems
One Microsoft Way, Redmond, WA 98052

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. This may be understood as extending Kelly's seminal framework from single-path unicast over general topology to multi-path multicast over P2P topology, with network coding allowed. For certain classes of popular 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 (multi-source) network coding. This simplification result allows us to develop a new multi-tree routing formulation for the problem. Despite of the negative results in literature on applying Primal-dual algorithms to maximize utility under multi-path settings, we have been able to develop a Primal-dual distributed algorithm to maximize the aggregate utility under the multi-path routing environments. Utilizing our proposed sufficient condition, we show global exponential convergence of the Primal-dual algorithm to the optimal solution under different P2P communication scenarios we study. The algorithm can be implemented by utilizing only end-to-end delay measurements between P2P nodes; hence, it can be readily deployed on today's Internet. To support this claim, we have implemented the Primal-dual algorithm for use in a peer-assisted multi-party conferencing system and evaluated its performance through actual experiments on a LAN testbed and the Internet.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS Distributed Systems; C.4 [**Computer Systems Organization**]: PERFORMANCE OF SYSTEMS — *Performance attributes* ; H.5.1 [**Information Systems**]: INFORMATION INTERFACES AND PRESENTATION Multimedia Information Systems

General Terms

Algorithms, Design, Measurement, Performance

Keywords

content distribution, multi-party video conferencing, multicast, peer-to-peer, streaming, utility maximization

1. 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 [21].

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

The client-server 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 and thus incurs heavy deployment and egress ISP bandwidth costs. 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.

1.1 Related Work

In the past decade, network utility maximization have attracted significant attention ever since the seminal framework was introduced in [17] and [22]. In the framework, network protocols are understood as distributed algorithms that maximize aggregate user utility under wired or wireless network resource constraints. For the single-path unicast scenarios considered in [17] and [22], 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 [25]. This framework not only provides a powerful tool to reverse engineering existing protocols such as TCP [16], but also allows systematic design of new protocols, see [7] for a comprehensive review.

There have been work on extending the framework to multi-path unicast scenarios [11] [20] [30], as well as single-tree multicast scenarios [15] [9]. 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 [11] [20] [30].

For utility maximization in single-tree multicast scenarios where routers enable multicast functionality, the constraint set is non-linear, in particular, involving non-differentiable $\max(\cdot)$ terms. In [15] and [9], 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 [9], or introducing auxiliary variables and applying either Proximal or sub-gradient approaches [15].

There is also work focusing on multicast scenarios where routers can perform intra-session network coding [5] [31] [24]. 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 $\max(\cdot)$ 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 [5] [31] [24].

1.2 Our Contributions

In this paper, we consider the general utility maximization problem for multiple multicast in a P2P setting, with multi-path delivery and inter-session network coding allowed. This setting differentiates our work from other existing work, and highlights the challenges we encounter. Multi-party conferencing is one of the applications of our work.

The main contributions of this paper 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 classes of popular 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-session or inter-session) network coding on peer nodes. This result is a multi-source extension of the single source result studied in [19].
- **New Tree-based Formulation:** We introduce a new formulation for utility maximization in P2P topology in which the variables are rates of individual *trees*. In contrast, almost all prior formulations in the rate control literature are either path-based or link-based. Our tree based formulation uses linear constraints, thus avoiding the nonlinear $\max(\cdot)$ terms in path and link based formulations. 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 routing scenarios with supporting evidence in [30], we design a queuing delay based Primal-dual algorithm that solves the utility maximization problem for multi-tree routing under a general sufficient condition that holds in popular P2P settings. This distributed algorithm is used by each source to adjust its transmission rate and split that rate across multiple multicast trees by utilizing end-to-end delay measurements between peer nodes.
- **Evaluation on the Internet:** The proposed distributed algorithms can be easily implemented in practice. We have built a prototype multi-party conferencing system in Python programming language using the Primal-dual algorithm and evaluated its performance for several multi-party conferencing scenarios on a LAN testbed, in a virtual environment, and also on the Internet. Our system can satisfy the strict end-to-end packet delivery delay requirements for conferencing systems because every packet goes through at most one hop in the overlay and we tightly control the queuing delay between nodes.

2. 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, some source node $s \in S$ 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.,

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

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

2.1 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 [1], 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.

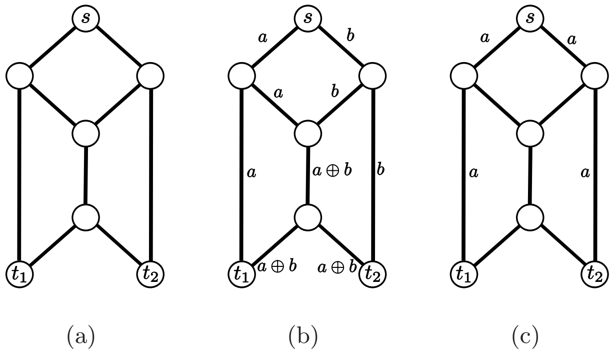


Figure 1: (a) Butterfly network with unit link capacities. (b) Network coding can achieve a multicast rate of 2. (c) Routing can achieve a multicast rate of 1.

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 [12]. 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 [32], 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 [12]. 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 [13]. 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.

2.2 Impact of P2P Topology

In P2P topology, we assume peer uplinks are the only bottlenecks in the whole network, and every peer can connect to every other peer through routing in the overlay. 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. 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 [19] 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.

This result extends and simplifies Edmonds' theorem [10] for P2P topology, in the sense that it allows helper (Steiner) nodes and uses only depth-1 and depth-2 Steiner trees. Fig. 4(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 [27], [8] and [18] also derived results similar to the above-stated Mutualcast theorem independently.

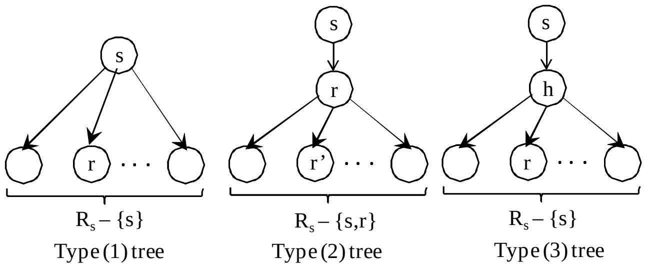


Figure 2: Different types of Mutualcast trees.

Given that the Mutualcast Theorem is for single source multicast scenario only, we first extend this result to the case of a multi-source multicast scenario when there is no coding across sessions belonging to different sources.

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. Then, the rate region $z = \{z_s, s \in S\}$ achievable by* **network coding within each session** *is also achievable by routing along $1+|R_s|+|H_s|$ Mutualcast trees for each source s independently.*

PROOF. Refer to [6]. □

This observation is interesting and practically important in the sense that it states that for practical P2P topology, routing is as good as intra-session network coding.

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 2. *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 [6]. □

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 1.

2.3 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:

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

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

Table 1: 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

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 [3]).

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 [2]. 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.

3. DISTRIBUTED ALGORITHM FOR MULTI-TREE BASED MULTICAST

The optimization problem we consider in (3) is a non-strictly concave optimization due to multi-path and 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.

The advantages of Primal algorithms are their wide applicability and fast convergence in multi-path/multi-tree routing scenarios [11]. The down side of the Primal algorithms is that they typically only generate approximate solutions.

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 [15] [31] and dual proximal algorithms [15] [20] are proposed to solve the problem. However, convergence of dual variables in these approaches are typically slow, and recovering optimal primal variables from optimal dual variables requires solving another optimization problem. Furthermore, it is not clear how to implement these algorithms on today's Internet.

In this paper, we focus on Primal-dual algorithms. As it will be clear later, 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 practice. Second, we show that our Primal-dual algorithm converges exponentially fast.

3.1 A Queuing Delay Based Primal-dual Algorithm

The Lagrangian of the optimization problem in (3) is as follows:

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

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 [2]: $\forall s\in S, \forall m\in s, \forall j\in J$,

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

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

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. 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 therefore 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, over the set $\{x\geq 0, p\geq 0\}$: $s\in S, \forall m\in s$, and $j\in J$,

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

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

where k_m is a positive constant controlling the adaptation rate of tree m and $(y_j-C_j)_{p_j}^{+}=y_j(t)-C_j$ if $p_j>0$, and is $\max(0,y_j-C_j)$ otherwise. It is known that p_j adapted according to (8) is simply queuing delay [23]. 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 common multi-path unicast scenarios [30, Section 2.5]. Consequently, to our best knowledge, almost no solution for multi-path routing utilizes such Primal-dual algorithm.

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 $K = \text{diag}\{k_m, m \in s, s \in S\}$, $C = \text{diag}\{C_j, j \in J\}$ 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 $V_0 = \{(\bar{x}, \bar{p}) : [\bar{z}, \bar{y}^H]^T = B\bar{x} = const\}$, 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:*
$$\begin{cases} \dot{\bar{x}} = KU' - KA_H G' - KA^T\bar{p} \\ \dot{\bar{p}} = C^{-1}A\bar{x} - \mathbf{1} \end{cases} \tag{9}$$
where U' and G' are constant matrices;
- *the above linear system is marginally stable, each of its trajectories do not converge, and all trajectories form limit cycles.*

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 [30].

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 3. *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 [6]. □

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

THEOREM 4. *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. Refer to [6]. □

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.

The condition in (10) may not be satisfied under some network settings, and the Primal-dual algorithm may not converge. One example is shown in [30, Section 2.5].

Interestingly, the unique structure of the P2P topology allows us to prove that the sufficient condition can be easily satisfied for two typical P2P systems as explored in the following subsections.

3.1.1 P2P Content Dissemination Scenarios

Consider a P2P data dissemination system with n peers, among which there are sources, receivers, and helpers. Every source distributes its content to its receivers, with or without the assistance of helpers. A receiver can receive contents from multiple sources simultaneously, while sources are servers that only distribute data but do not receive contents. These scenarios correspond to the popular P2P file distribution and P2P streaming scenarios in practice.

Assume the first n_s number of the nodes are sources. Define $R_i, 1 \le i \le n_s$, to be the set of peers that want to receive contents from source i. Define $H_i, 1 \le i \le n_s$ be the set of helpers that help distributing the content of source i. For the ease of explanation, we assume there is no helper in the following analysis, i.e., $H_i = \emptyset$. The analysis can be straightforwardly extended to the case where $H_i \neq \emptyset$.

Let $n_i = |R_i| + 1$, $1 \le i \le n_s$. Each source uses total $n_i + 1$ Mutualcast trees to deliver its content. For the sake of simplicity, we use the following notation when stating the result. We denote x_{ij} as the rate of tree j of source i passing through node j in the level one, with $1 \le j \le n$ and $1 \le i \le n_s$. Let k_{ij} represent how fast the tree rate x_{ij} adapts, and $k_{ij} = 0$ if $j \notin R_i$ and $j \neq i$. This is equivalent to having exactly $n_i + 1$ trees for source i. Since a source is not receiver for other sources in our P2P data dissemination scenarios, we also have $k_{ij} = 0$ for all $1 \le j \le n_s$ and $j \neq i$.

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 data dissemination systems.

THEOREM 5. *For P2P data dissemination systems in P2P topology, all (x, p) trajectories of the system in (7)-(8) converge to one of its equilibria globally asymptotically, if the following conditions are satisfied:*

- *For all $1 \le i \neq j \le n$, $\xi_i \neq \xi_j$, where*
$$\xi_l = \begin{cases} \frac{(n_l - 1)n_l}{C_l} k_{ll}, & 1 \le l \le n_s; \\ \frac{1}{C_l} \sum_{j: l \in R_j} (n_j - 1)^2 k_{jl}, & \textit{otherwise} \end{cases}$$
- *$k_{ii} < \frac{C_i}{2C_j} k_{ij}$, for all $1 \le i \le n_s$ and $n_s < j \le n$.*

PROOF. Refer to [6]. □

In practice, these conditions are in fact easy to satisfy with high probability. For example, source i can generate $k_{ij}, j \neq i$ in a suitable range *randomly* such that the first condition is satisfied with a good chance.

Source i can then select k_{ii} such that the second condition is satisfied under practical relationship between C_i, normally the server bandwidth, with C_j, normally peers' (home users') uplink bandwidth. For instance, we can assume practically $\min(\frac{C_i}{C_j}) = 1$ and set k_{ii} to be less than $\frac{1}{2}\min_j(k_{ij})$ in a random fashion.

In practice, satisfying this condition also forces the source to adapt the depth-2 Mutualcast trees with high priority, indicating source nodes, normally the server, will adapt quickly to the network condition changes in receivers, as compared to the response to its own uplink condition change.

3.1.2 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 6. *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 [6]. □

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.

4. 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.

4.1 Peer Functionality

Besides encoding and decoding video streams, each peer (except helpers) is also involved in building and maintaining multicast trees, and forwarding packets along the trees.

As a source, every peer is the root of its n multicast trees. It sets up these trees at the beginning of the conference, adjusts them upon peer joining and leaving, and splits the encoded data among the trees. Peers also collect queuing delay information from other peers on their trees and adjust the tree rates according to (7).

Each peer on multicast trees also forwards the packets from upstream tree branches to downstream branches. It is achieved 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. 4(a) makes two copies of every packet it receives from one peer (A, B, or C, respectively), and unicasts the copies to the other two peers.

[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.

Meanwhile, peers on multicast trees also measure queuing delay between two peer nodes and return these measurement to the root. This is done in a simple way that will be discussed later.

4.2 Queuing Delay Measurement and Updating Tree Rates

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 known that queuing delay between two peers can be estimated by the difference between current ROWD and the smallest ROWD seen in the history [4].

The advantage of measuring delay based on ROWD is that it does not require time synchronization across peers. 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.

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).

4.3 Utility (PSNR) Modeling

In video processing, PSNR 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 frame 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.

Interestingly, we *empirically* found that the PSNR of a source s's video coded at rate z_s can be approximated by logarithmic function $\beta_s \log(z_s)$, with higher β_s for videos with large amount of motion and lower β_s for rather still scenes, and they can be obtained from the video encoder of source s during encoding. Based on this empirical observation, we use utility function $U_s(z_s) = \beta_s \log(z_s)$ in our

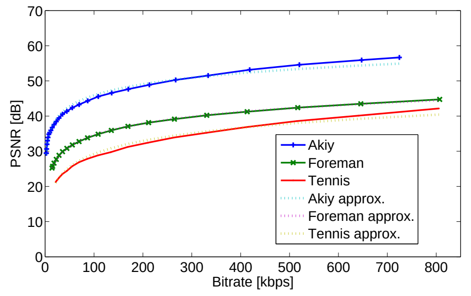


Figure 3: The logarithmic approximation of PSNR curves of Akiyo, Foreman and Tennis sequence.

experiments.

Fig. 3 shows PSNR curves of three videos as functions of encoding rates. These represent the receiving video quality if a source peer encodes and sends its video at these rates. We have chosen three video sequences in CIF format: Akiyo, 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) [14] with various bitrates to obtain the PSNR curves.

4.4 Implementing Helper Cost Function Using Delay

In order to implement the Primal-dual algorithm in (7)-(8) in practice, we need to generate cost of using helpers according to their cost functions $G_h(y_h)$, and to update source peers about their values.

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 $G'_h(y_h)$ 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 $G'_h(y_h)$ cost into account when adjusting the tree rates according to (7).

4.5 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.

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.

Let $(\bar{x}, \bar{p})$ be the equilibrium 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 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

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

As such, given a lower bound on z_s, we can bound $\bar{d}_m$ with a designed value by tuning the constant α. For example, for P2P multi-party conferencing system, the system designer may want to set a limit on how low the equilibrium source rate can be, since the video quality will be unacceptable at such limited rate. This will give a lower bound on z_s, and hence a lower bound 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 by solving α according to (12) with the designed value and the lower bound for lower bound of $U'_s(\bar{z}_s)$. 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.

5. 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 [28]. Scenarios 1 and 2 use the topology and network conditions described by Fig. 4(a).

We have also conducted experiments in Scenario 3 and 4, under real Internet settings with peers being spread around the US and virtual machines in a Virtual Lab [29], respectively.

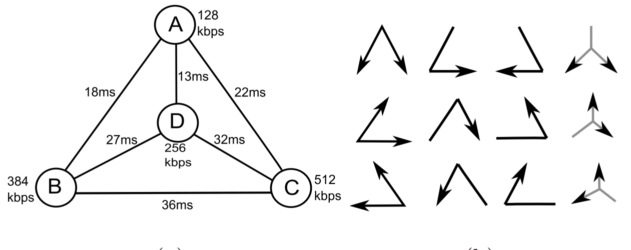


Figure 4: Scenarios 1 and 2: (a) Propagation delays and uplink bandwidth constraints for each peer and link in our experimental testbed. (b) Multicast trees for three nodes (A, B, C) and one helper (D) from Section 2.2.

5.1 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.

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 [26] 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. 4(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.

5.2 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.

5.3 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 4.5, 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.

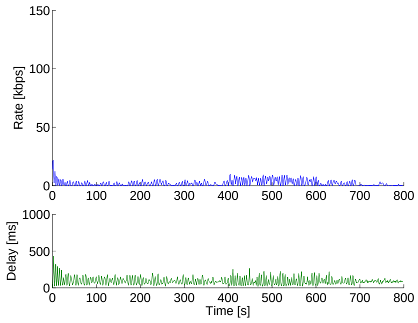
5.4 Scenario 4: Scalability Study

The last scenario (Fig. 7) shows a large conference with 10 peers successively joining. The conference starts with 3 peers and every 60 seconds a new peer joins. All 10 nodes have an uplink bandwidth of 384 kbps and they were run on virtual machines in a Virtual Lab [29].

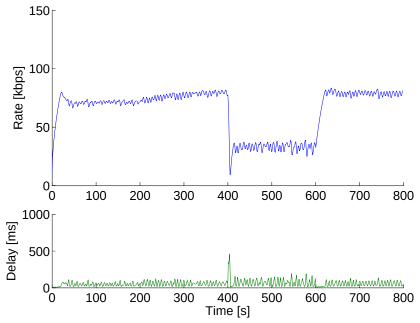
Forming a large video-conference does not incur excessive overhead and costs to set-up and maintain the multicast trees. When a new peer joins a conference, all peers update their trees which is a very simple operation and the rate updating algorithm continuously changes the rates to achieve optimal use of all the available bandwidth (as described in Section 4). Note that all the necessary information can be easily spread throughout the system by appending it to any video packet as it is delivered to all peers.

During peer joins we can observe a peak in the queuing delay (see a detail in Fig. 7(b)). This is due to the necessity to immediately deliver all video streams to the new peer and the trees react by reducing their rates. After few seconds the rates increase as the new peer helps to distribute the videos through newly created trees. The peaks can be avoided by reducing the tree rates when new peer joins, proportionally to the number of tree branches before and after, which will also remove the drops in utility graph (Fig. 7(a)) and speed up the convergence. We disabled this feature to show the natural reactions of the system. The computational costs for updating tree rates and memory requirements are negligible. Peers only keep track of the last few queuing delay measurements for each other peer's uplink and compute the running averages over them.

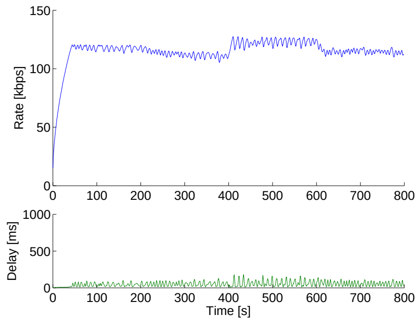
However, the rates decrease in general as peers join because there is a need to deliver each peer's video stream to more peers and new streams of the newly added peers further increase this demand, but the new peers provide only a small amount of newly available bandwidth for the system to use. This is a scalability issue and you can observe



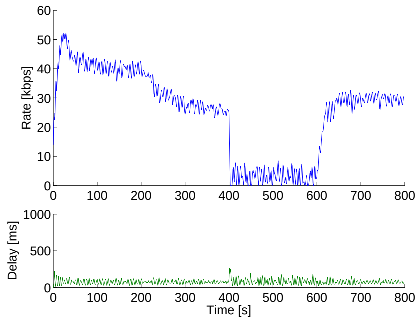
(a) Tree (A→B,A→C)



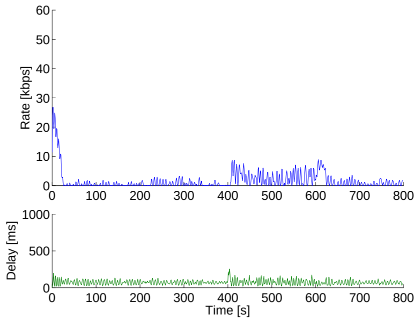
(b) Tree (B→A,B→C)



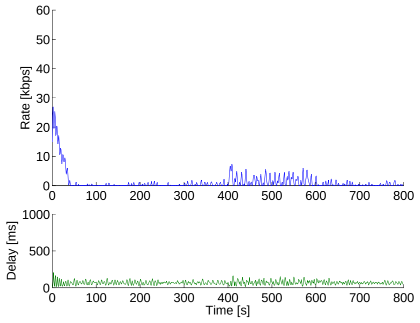
(c) Tree (C→A, C→B)



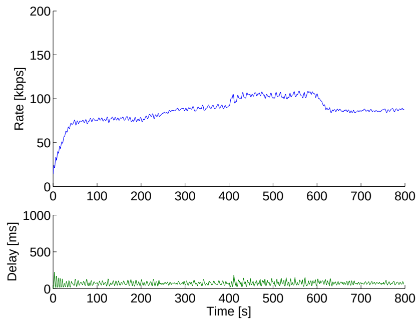
(d) Tree (A→B, B→C)



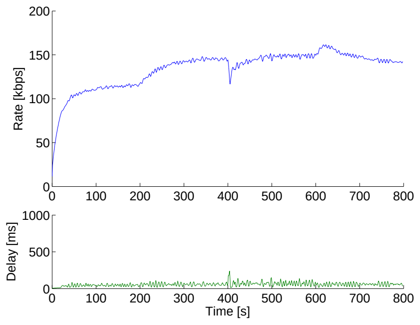
(e) Tree (B→A, A→C)



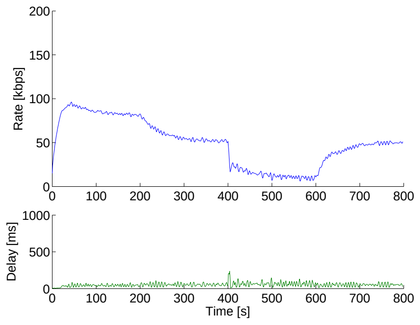
(f) Tree (C→A, A→B)



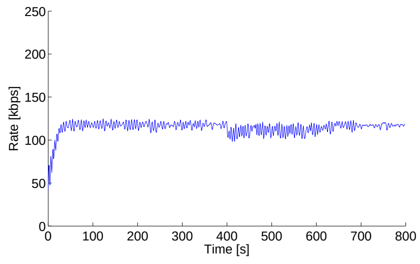
(g) Tree (A→C, C→B)



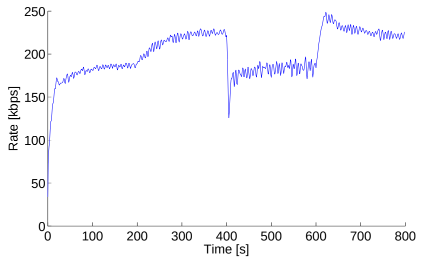
(h) Tree (B→C, C→A)



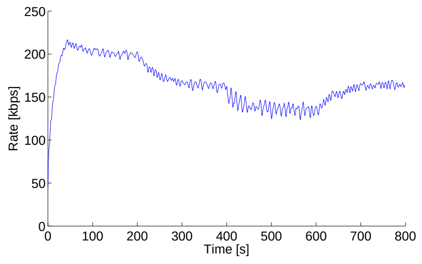
(i) Tree (C→B, B→A)



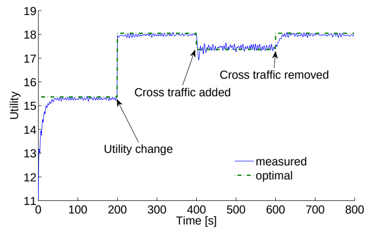
(j) Aggregate rate for A



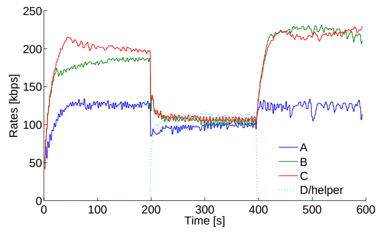
(k) Aggregate rate for B



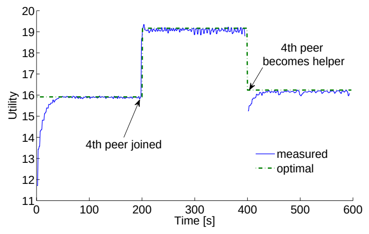
(l) Aggregate rate for C



(m) Utility (Scenario 1)



(n) Rates (Scenario 2)



(o) Utility (Scenario 2)

Figure 5: Scenario 1 – (a) through (i): Sending rates and total delays for trees with edges and topology shown in Fig. 4(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.

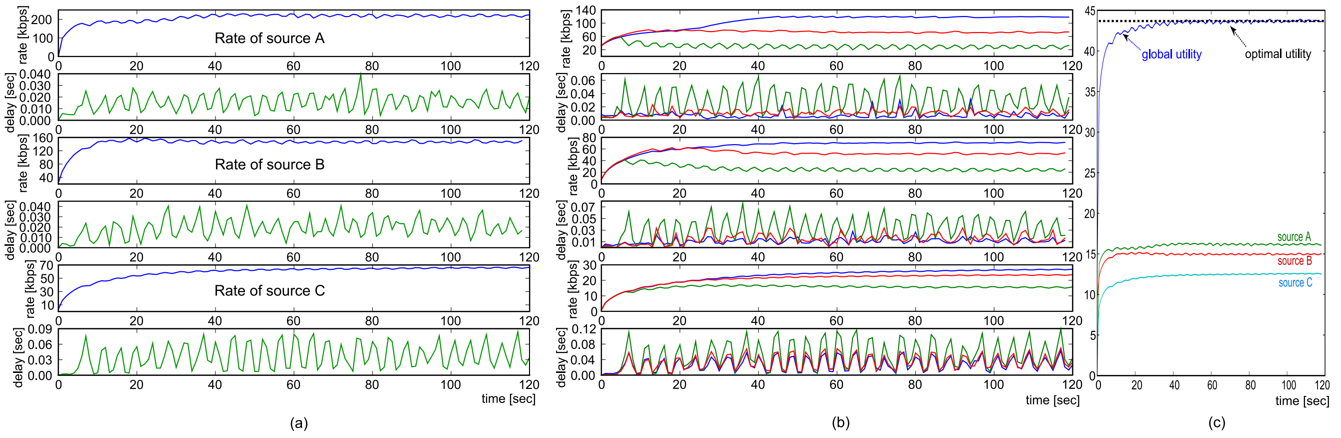


Figure 6: Experimental results for Scenario 3: (a) Source rates of A, B, and C, respectively, with the average tree branch delays. (b) Tree rates for trees of A, B and C, respectively, with the tree branch delays. (c) The aggregate utility achieved by the system, and the utilities per source.

from the graph that to run a large video conference peers need to have sufficient bandwidth or they would suffer from low video bitrates otherwise. A nice property of our scheme is that helper servers with an extensive bandwidth capacity can be easily accommodated to help in a large conference and we can control their use and the associated costs (Section 4.4). The multi-party video conferencing use case is inherently intended for a small number of peers, unlike other P2P streaming applications, and is limited by the fact that adding a new peer brings a burden of both delivering all previous video streams to a new destination and also delivering a new stream to all previous peers. The significant advantage of using our scheme is that it combines common video delivery schemes (direct, helper server assisted and peer assisted delivery) into one framework and uses them in an optimal way.

With more peers both the delays and rates exhibit more oscillations and the delays increase as there are more nodes and trees involved but still stay reasonably small. The oscillations in Fig. 7(c) grow with the number of trees because the graphs show the aggregated rates of many trees and all the tree rates behave according to the same queuing delay measurements for the uplinks of peers. The measurements are correlated (see Fig. 7(b)) and thus the noise amplitude increases. We can also see that the utility achieved is optimal even though for 10 peers the oscillations are too big to let the rates stay at the optimum as our system tries to keep queuing delays low. Note also that we can temporarily exceed the optimum utility in the plot 7(a) because our methodology of computing utility is based on sending rates which can temporarily exceed the bandwidth limits.

6. 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. Another area of future work would be to consider multirate multicast where different receivers can receive the same video at different rates through the use of scalable coding or transcoding.

7. REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Inform. Theory*, (4):1204–1216, July 2000.

[2] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[3] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[4] L. S. Brakmo and L. L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global internet. *IEEE J. Select. Areas Commun.*, (8):1465–1480, Oct. 1995.

[5] 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.

[6] 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.

[7] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition:a

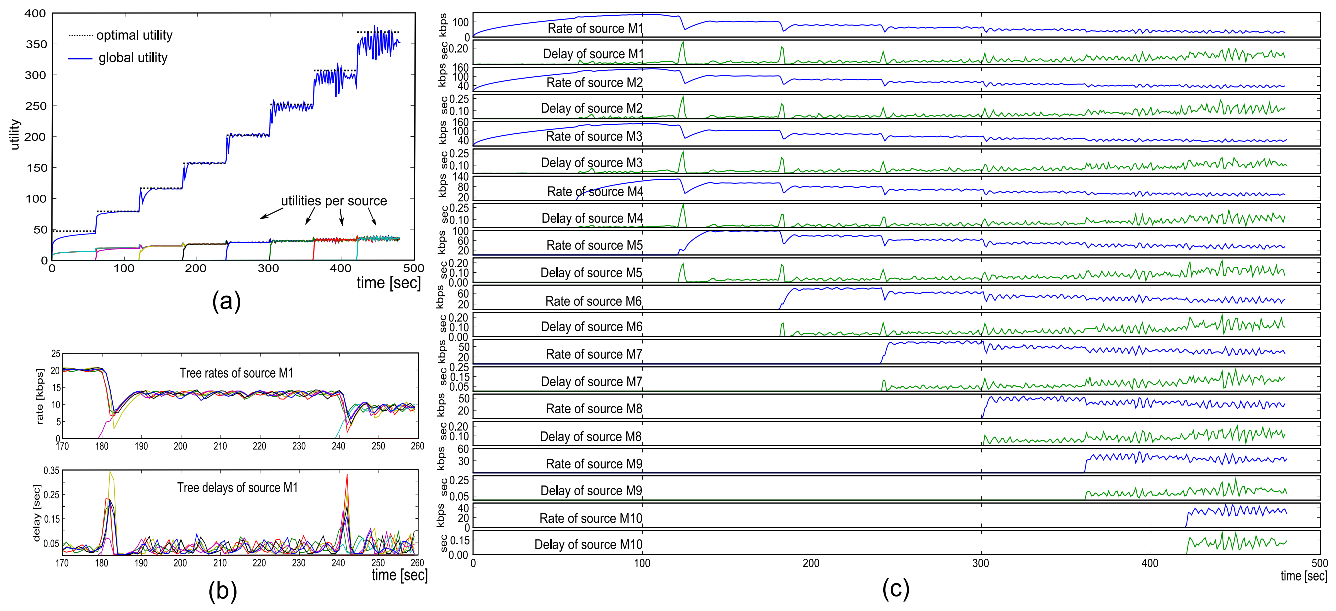


Figure 7: (a) The aggregate utility achieved by the system. (b) A detailed view on tree rates and delays of peer $M1$. (c) Source rates of individual peers $M1$ through $M10$ and the average tree branch delays.

mathematical theory of network architectures. *Proc. IEEE*, 95(1):255–312, Jan. 2007.

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

[9] S. Deb and R. Srikant. Congestion control for fair resource allocation in networks with multicast flows. *IEEE Trans. Automat. Contr.*, (2):274–285, Apr. 2004.

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

[11] 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.

[12] 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.*, 51(6):1973–1982, 2005.

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

[14] H. Kalva. The h.264 video coding standard. *IEEE Trans. Multimedia*, 13(4):86–90, Oct. 2006.

[15] K. Kar, S. Sarkar, and L. Tassiulas. Optimization based rate control for multirate multicast sessions. In *IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001.

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

[17] 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*, pages 237–252, 1998.

[18] R. Kumar, Y. Liu, and K. Ross. Stochastic fluid theory for p2p streaming systems. In *Proc. IEEE INFOCOM*, Anchorage, AL, USA, 2007.

[19] 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.

[20] X. Lin and N. B. Shroff. Utility maximization for communication networks with multi-path routing. *IEEE Trans. Automat. Contr.*, (5):766–781, May 2006.

[21] 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)*, pages 1125–1128, April 2007.

[22] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Trans. Networking*, (6):861–875, Dec. 1999.

[23] S. H. Low, L. Peterson, and L. Wang. Understanding vegas: A duality model. *Journal of ACM*, 49(2):207–235, Mar. 2002.

[24] D. S. Lun, N. Ratnakar, M. Mąäedard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Trans. Inform. Theory*, (6):2608–2623, June 2006.

[25] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Networking*, (5):556 – 567, Oct. 2001.

[26] MOSEK ApS. MOSEK Optimization Software.

[27] J. Mundinger, R. Weber, and G. Weiss. Analysis of peer-to-peer file dissemination amongst users of different upload capacities. In *Proceeding of IFIP Performance*, 2005.

[28] Z. Ni. Network Emulator for Windows/CE. Microsoft Research Asia, Internal documentation.

[29] 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*, West Point, NY, June 2002.

[30] T. Voice. *Stability of Congestion Control Algorithms with Multi-Path Routing and Linear Stochastic Modelling of Congestion Control*. PhD thesis, University of Cambridge, Cambridge, UK, May 2006.

[31] Y. Wu and S.-Y. Kung. Distributed utility maximization for network coding based multicasting: a shortest path approach. *IEEE J. Select. Areas Commun.*, (8):1475–1488, Aug. 2006.

[32] 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.