Optimal Bandwidth Sharing in Multi-Swarm Multi-Party P2P Video Conferencing Systems

Chao Liang, *Student Member, IEEE,* Miao Zhao, *Member, IEEE,* and Yong Liu, *Member, IEEE*

***Abstract*—In a multi-party video conference, multiple users simultaneously distribute video streams to their receivers. As the traditional server-based solutions incur high infrastructure and bandwidth cost, conventional Peer-to-Peer (P2P) solutions only leveraging end users' upload bandwidth, are normally not self-sustainable: the video streaming workload increases quadratically with the number of users as each user could generate and distribute video streams, while the user upload bandwidth only increases linearly. Recently, hybrid solutions have been proposed that employ helpers to address the bandwidth deficiency in P2P video conferencing swarms. It is also noticed that a system hosting multiple parallel conferencing swarms can benefit from *cross-swarm bandwidth sharing*. However, how to optimally share bandwidth in such systems has not been explored so far. In this paper, we study the optimal bandwidth sharing in multi-swarm multi-party P2P video conferencing systems with helpers and investigate two cross-swarm bandwidth sharing scenarios: 1) Swarms are *independent* and peers from different swarms share a common pool of helpers; 2) Swarms are *cooperative* and peers in a bandwidth-rich swarm can further share their bandwidth with peers in a bandwidth-poor swarm. For each scenario, we develop distributed algorithms for intra-swarm and inter-swarm bandwidth allocation under a utility-maximization framework. Through analysis and simulation, we show that the proposed algorithms are robust to peer dynamics, and can adaptively allocate peer and helper bandwidth across swarms so as to achieve the system-wide optimum.**

***Index Terms*—Peer-to-Peer, conferencing, optimal bandwidth sharing, distributed algorithms, scheduling.**

I. Introduction

Video conferencing applications, such as MSN Messenger [1] and Skype [2], are getting increasingly popular on the Internet in recent years. Video conferencing between two users can be implemented simply by letting one user send its video streams directly to the other. However, in a multi-party video conference, multiple users simultaneously distribute their video streams to multiple receivers and the video streaming workload increases quadratically with the number of users in the conference, as each user could generate and distribute video streams. In a traditional server-based solution, a user's video streams will first be uploaded to a server, and then be relayed to the receivers. The drawback is that the server incurs high infrastructure and bandwidth cost. P2P technology can be utilized to offload servers. In a conventional P2P conferencing solution, users in the same conference form a *conferencing swarm* and relay video streams to each other. Unfortunately, conventional P2P conferencing solutions are normally not self-sustainable: users in a conference alone often do not have enough upload bandwidth to support video streaming from multiple sources to multiple receivers. In recently proposed hybrid solutions [3][4], helpers, which could be video conferencing servers, are employed to address the bandwidth deficiency in P2P video conferencing swarms. Video generated by a user is relayed to multiple receivers via not only other users in the same conferencing swarm but also the helpers.

C. Liang and Y. Liu are with the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, NY, 11201 USA e-mail: {cliang@photon.poly.edu, yongliu@poly.edu}.

M. Zhao is with the department of Electrical and Computer Engineering, the State University of New York at Stony Brook, Stony Brook, NY 11794 USA e-mail: mzhao@ic.sunysb.edu.

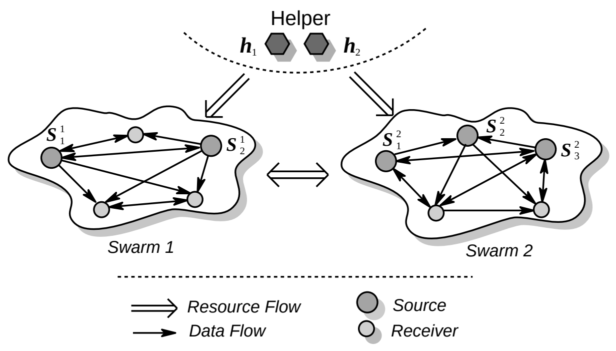


Fig. 1. Cross-swarm Sharing between Two Conferencing Swarms.

In a video conferencing system, there are commonly multiple parallel conferencing sessions. The corresponding swarms for different conferencing sessions are inherently heterogeneous: peers have different levels of bandwidth availability; swarms have diverse streaming service requirements, such as video streaming rate. In the context of P2P file sharing [5] and video streaming [6], cross-swarm bandwidth sharing has been demonstrated to effectively address the bandwidth heterogeneity and achieve the *multiplexing gain* among heterogeneous swarms. We propose cross-swarm sharing to address the bandwidth challenge in P2P video conferencing systems. Specifically, with hybrid P2P conferencing, it is more economical to share a pool of helpers among multiple swarms, instead of dedicating helpers to individual swarms. Furthermore, peers in a bandwidth-rich swarm can also share their bandwidth directly with peers in a bandwidth-poor swarm. Figure 1 illustrates an example of the cross-swarm bandwidth sharing between two swarms.

In this paper, we investigate optimal bandwidth sharing strategies for multi-swarm multi-party video conferencing systems. We study two sharing scenarios: 1) Swarms are *independent* and draw bandwidth only from a shared helper pool; 2) Swarms are *cooperative* and can further share bandwidth directly with each other. For both scenarios we study the optimal bandwidth sharing under a utility-maximization framework. For the first scenario, bandwidth sharing operates at two levels: within each swarm, users adjust their video multicast rates to maximize the aggregate utility of the swarm; among swarms, helpers allocate their upload bandwidth to maximize the aggregate utility of the entire system. Through proximal approximation and dual decomposition, we develop distributed algorithms for joint source rate control and helper bandwidth allocation to drive the system to the optimal operating point. We further design marginal-utility based algorithms with low overhead and fast convergence for practical implementations. For the second scenario, additional bandwidth sharing between bandwidth-rich swarms and bandwidth-poor swarms needs to be jointly considered. Accordingly, we design cross-swarm bandwidth sharing rules and distributed algorithms that allow swarms to dynamically adjust the resources shared with others in the face of peer churn and swarm churn. Through numerical and simulation results, we show that the proposed algorithms are robust to peer dynamics, and can adaptively allocate peer and helper bandwidth across swarms to achieve the system-wide optimum.

The remainder of this paper is organized as follows. We briefly discuss the related work in Section II. In Section III, we present the architecture of multi-swarm multi-party P2P conferencing systems under study. The utility maximization framework is formulated. In Section IV, we develop distributed algorithms for source rate control and helper bandwidth allocation for systems with independent swarms. The distributed bandwidth sharing algorithms for cooperative swarms are presented in Section V. The proposed algorithms are evaluated through numerical analysis and simulations in Section VI and Section VII, respectively. Finally, the paper is concluded in Section VIII.

II. Related Work

It is challenging to provide multi-party video conferencing service due to its high bandwidth demand and stringent streaming quality requirement. Compared with traditional server-based solutions, P2P conferencing solutions are more scalable and incur less infrastructure cost. Chu *et al.* [7] proposed an End System Multicast architecture to support video conferencing applications, where multicast functionality is pushed to the edge. Lennox and Schulzrinne [8] proposed a full mesh conferencing protocol without a central point of control. Luo *et al.* [9] proposed to integrate application layer multicast with native IP multicast in P2P conferencing systems. Recently, Chen *et al.* [4] proposed hybrid solutions to employ helpers to maximize the utility in P2P conferencing swarms, where helpers assist users in relaying video streams to receivers. Ponec *et al.* [10] then extended this solution to support multi-rate conferencing applications with scalable coding techniques. Most existing works focus on the design of isolated P2P conferencing swarms, where all users in the same conference form a swarm and exclusively help each other in the same swarm relay video streams. However, in P2P conferencing systems, there are generally multiple ongoing conferencing swarms with heterogeneous user bandwidth availability and different service requirements. Under such circumstances, it is necessary to enable resource sharing among swarms to maximize the global welfare.

Cross-swarm bandwidth sharing has been proposed for other kinds of P2P applications. Guo *et al.* [5] proposed cross-swarm bandwidth sharing strategy for P2P file sharing applications. Server bandwidth allocation schemes among parallel file sharing swarms were proposed in [11]. In live video streaming, cross-channel sharing was proposed to improve the streaming qualities of channels and provide service differentiation among channels [6][12]. Wu *et al.* [13] proposed server bandwidth provision algorithms to predict channel demands and dynamically allocate server bandwidth among multiple streaming channels. In video on-demand systems [14], peers leverage caches to store and serve content peers in different sessions to enable cross-channel sharing. However, cross-swarm bandwidth sharing for P2P video conferencing has not yet been explored so far. To the best of our knowledge, our paper is the first one to study the optimal bandwidth sharing in multi-swarm multi-party video conferencing systems.

III. Problem Formulation

A. Network Model

We consider a system consisting of a set I of parallel P2P conferencing swarms. All swarms share a set H of helpers, which could be the dedicated servers from service providers, or other altruistic nodes not participating in any conference.

In conferencing swarm i, let V_i be the set of users in the conference. We allow the existence of pure receivers who generate no video streams. A subset $S_i \subseteq V_i$ of users are video sources. We use N_i to denote the set of all involved nodes for swarm i, where $N_i = V_i \cup H$. Each source $s \in S_i$ establishes a multicast session in the application layer and distributes its own unique video streams to all other users in the conference. Videos streams from all sources are relayed by all users in the swarm and outside helpers. We assume that sources leverage scalable layered video coding technique(such as H.264/SVC [15]), so that the video streaming rates of sources are not fixed and higher streaming rates lead to better display quality at the receiver side. The video multicast rate z_s of source s is upper-bounded by e_s, which reflects the maximum video encoding capability of s or the willingness of s to delimit its maximum video streaming rate. Sources in the same swarm compete for bandwidth resources from peers and helpers to increase their multicast rates for better playback quality at receiver side.

For the multicast session of source s, let $U_s(z_s)$ be the utility for a user to receive video streams from s at rate z_s. We assume $U_s(\cdot)$ is increasing, concave and twice-differentiable. For each conferencing swarm, the goal is to maximize the aggregate utility of all users in all multicast sessions. The global welfare of the entire system is the aggregate utility of all users in all swarms.

B. Distribution Trees within Conferencing Swarm

Here we introduce the distribution structures within a conferencing swarm to accommodate the multi-source multicast sessions. How to achieve the maximum rates in multiple-source multicast with general network topology is challenging and still largely open, although the maximum rate in single-source multicast can now be achieved by network coding with polynomial complexity [16], [17]. In P2P overlay networks, where each peer can reach all others, it is commonly assumed that peer upload links are the only bandwidth bottleneck. For uplink-throttled P2P networks, it was shown in [18], [3] that the maximum multicast rate for a single source can be achieved by packing a linear number of Steiner trees. For a source s, with a set R_s of receivers and a set H_s of helpers, the maximum multicast rate can be achieved by packing $1 + |R_s| + |H_s|$ number of trees as in Figure 2.

- One *depth-1* tree, source directly reaches all receivers in R_s, indicated as type (1) tree.
- $|R_s|$ *depth-2* trees, one receiver r relays traffic to all other receivers and $r \in R_s$, indicated as type (2) tree.
- $|H_s|$ *depth-2* trees, one helper h relays traffic to all receivers in R_s and $h \in H_s$, indicated as type (3) tree.

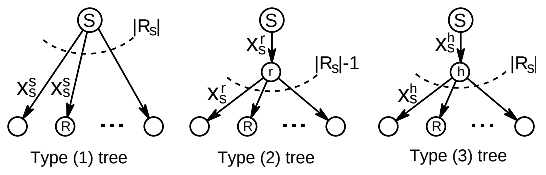


Fig. 2. Different Types of Distribution Trees.

We adopt this two-hop relay scheme for multi-source multicast in multi-party video conferencing system. In each swarm, peers form the above distribution trees for each source. Since the scale of a conference swarm is typically small, it is affordable to form such fully-connected mesh. The distribution trees have at most two hops, which implies short propagation delays when video streams are forwarded along the trees. This makes the two-hop relay scheme appropriate for video conferencing applications with tight delay requirement [1]. Given the conferencing architecture, each source only needs to determine the multicast rates on each of its distribution trees.

C. Utility Maximization

1) Rates of Distribution Trees based Optimization: As illustrated in Figure 2, there are different kinds of distribution trees for sources. We let x_s^m denote the rate of video streams generated by $s \in S_i$ and relayed by $m \in N_i$. Then we have $z_s = \sum_{m \in N_i} x_s^m$. Note that x_s^s denotes the video broadcast rate in the depth-1 tree rooted at s. Let y_v and c_v denote the aggregate upload rate and upload capacity of node v, respectively. For source s in swarm i, we use R_i^s to represent the set of corresponding receivers. Because each source distributes its content to all other users in the conference, the sets of receivers have the same degree $\mathcal{R}_i$, that is, $|R_i^s| = \mathcal{R}_i, \forall s \in S_i$. The notations are summarized in Table I. Now we formulate the problem when swarms are independent and only draw bandwidth from the helpers.

[1]Server-based conferencing solution also incurs two-hop propagation delay.

TABLE I
NOTATIONS

Notation	Definition
I	set of swarms in the conferencing system
H	set of helper nodes in the conferencing system
V_i	set of participating users in conferencing swarm i
N_i	set of nodes in swarm i, $N_i = V_i \bigcup H$
S_i	set of video sources in swarm i, $S_i \subseteq V_i$
R_i^s	set of receivers of source s in swarm i
$\mathcal{R}_i$	number of receivers for source in swarm i
x_s^m	the multicast rate of tree relayed by m from source s
y_v	the aggregate upload rate of node v
z_s	the aggregate multicast rate from source s
b_s	source s spends on distributing its own content
c_v	the upload bandwidth capacity of node v
e_s	the maximum multicast rate from source s
ρ	helper bandwidth efficiency factor
μ_i	marginal utility of swarm i to helpers
f_i^h	helper h contributes to swarm i
$\mathbf{U}^i$	aggregate utility of users in swarm i
U_s	the utility function of receiving video streams from source s
G_h	the cost function of drawing bandwidth from helper h

We now derive the aggregate upload rate of each node. Let b_s denotes the required bandwidth of source s to drive its own distribution trees. For source $s \in S_i$, we obtain

$$b_s = \mathcal{R}_i x_s^s + \sum_{r \in V_i \setminus \{s\}} x_s^r + \sum_{h \in H_s} x_s^h. \tag{1}$$

As we have learned from Figure 2, while a regular peer uploads to receivers only in type (2) trees of sources in the swarm, a source node not only drives its own distribution trees for broadcasting its content, but also helps relay traffic in type (2) trees of other sources. Hence, for a peer $v \in V_i$ in conference i, we can calculate its aggregate upload rate on all distribution trees for all sources in the swarm as

$$y_v = \begin{cases} b_v + \sum_{k \in S_i \setminus \{v\}} (\mathcal{R}_i - 1) x_k^v & \text{if } v \in S_i \\ \sum_{s \in S_i} (\mathcal{R}_i - 1) x_s^v & \text{otherwise,} \end{cases} \tag{2}$$

Similarly, we can derive the aggregate upload rate of each helper as $y_h = \sum_{i \in I} \sum_{s \in S_i} \mathcal{R}_i x_s^h, \quad \forall h \in H.$

To encourage peers to use their own bandwidth first before resorting to the helpers, we assume peers incur a cost of $G_h(f)$, which is increasing and convex, when drawing bandwidth f from helper h. To maximize the aggregate utility of all users in the system, we aim to maximize

$$\max_{x_s^m \geq 0} \sum_{i \in I} \sum_{s \in S_i} \mathcal{R}_i U_s\Big(\sum_{m \in N_i} x_s^m\Big) - \sum_{h \in H} G_h\Big(\sum_{i \in I} \sum_{s \in S_i} \mathcal{R}_i x_s^h\Big).$$

subject to

$$\sum_{m \in N_i} x_s^m \leq e_s, \quad \forall s \in S_i, i \in I. \tag{3}$$

$$y_v \leq c_v, \quad \forall v \in N_i, i \in I. \tag{4}$$

Equation (3) indicates that the source multicast rate is bounded by e_s. Equation (4) enforces that the upload rate of a peer or helper is constrained by its upload capacity.

The difficulty of directly solving the problem falls in that the objective is not strictly concave with respect to $\mathbf{x}$, such that the optimal solution is not unique. Moreover, it has $\sum_i |S_i||N_i|$ number of variables and involves complex computations. Fortunately, we find that the objective function is strictly concave with respect to the aggregate multicast rate of sources. To ease the problem solving, we tackle the problem in an alternative way to resolve the problem with respect to $\mathbf{z} = \{z_s\}$. With the optimal solution of z_s, we can recover each x_s^m at low cost. In this way, the complexity of the original problem can then be greatly reduced.

2) Source Multicast Rates based Optimization: To maximize the utility with respect to source multicast rate vector $\mathbf{z}$, we first investigate the domain of $\mathbf{z}$ and then show how to recover video streaming rates $\mathbf{x}$ along distribution trees from any feasible $\mathbf{z}$. Let f_i^h be the bandwidth contributed to swarm i by helper h. We have the following results on the maximal multicast rates in swarm i.

Theorem 1: In conferencing swarm i with helper contributions $\mathbf{f_i} = \{f_i^h\}$, the maximal multicast rate region $\mathcal{Z}_i^*$ is characterized by

$$\begin{cases} \mathcal{R}_i \sum_s z_s \le \sum_v c_v + \frac{\mathcal{R}_i - 1}{\mathcal{R}_i} \sum_h f_i^h \\ z_s \le \min(c_s, e_s), \forall s \in S_i. \end{cases} \quad (5)$$

For any feasible multicast rate vector $\mathbf{z} \in \mathcal{Z}_i^*$, a delivery rate vector $\mathbf{x}$ for all distribution trees of all sources can always be recovered correspondingly.

Proof: Since $z_s = \sum_{m \in N_i} x_s^m$, Equation (1) can be rewritten as follows

$$b_s = (\mathcal{R}_i - 1)x_s^s + z_s. \quad (6)$$

Let $z_v = 0$ for pure receivers $v \in V_i \setminus S_i$. Equation (2) can thus be reformulated as

$$y_v = (\mathcal{R}_i - 1) \sum_{s \in S_i} x_s^v + z_v, \quad \forall v \in V_i.$$

In type (3) trees, helpers broadcast content to $\mathcal{R}_i$ participating users for each source, we have $f_i^h = \mathcal{R}_i \sum_{s \in S_i} x_s^h$. The summation of bandwidth that all participating users spend yields

$$\begin{aligned} \sum_{v \in V_i} y_v &= \sum_{v \in V_i} z_v + (\mathcal{R}_i - 1) \sum_{s \in S_i} \sum_{v \in V_i} x_s^v \\ &= \sum_{v \in V_i} z_v + (\mathcal{R}_i - 1) \sum_{s \in S_i} (z_s - \sum_{h \in H} x_s^h) \\ &= \mathcal{R}_i \sum_{v \in V_i} z_v - (\mathcal{R}_i - 1) \sum_{h \in H} \sum_{s \in S_i} x_s^h \\ &= \mathcal{R}_i \sum_{v \in V_i} z_v - \frac{\mathcal{R}_i - 1}{\mathcal{R}_i} \sum_{h \in H} f_i^h \le \sum_{v \in V_i} c_v. \end{aligned}$$

Hence, the maximal rate region is bounded by Equation (5).

For any multicast rate vector falling in the region defined by (5), we can find a set of delivery rates for all distribution trees of all sources under the bandwidth constraints on peers and helpers. The following algorithm presents a method to realize $\mathbf{x}$ from $\mathbf{z}$ favoring the depth-1 trees. Algorithm 1 first assigns the largest possible multicast rates on depth-1 trees of all sources, then assigns multicast rates on depth-2 trees proportional to relay node's capability. It can be easily verified that the multicast rate vector $\mathbf{x}$ is recovered from $\mathbf{z}$ consistently, and the bandwidth constraints on all peers and helpers are preserved. ■

Algorithm 1: Recovery of tree multicast rates

```
   Input  : multicast rates $\mathbf{z}$ of sources in swarm $i$
   Output: multicast rates $\mathbf{x}$ of distribution trees
 1 for $s \in S_i$ do                    // depth-1 trees
 2     if $c_s > \mathcal{R}_i z_s$ then  $x_s^s = z_s$
 3     else  $x_s^s = \frac{c_s - z_s}{\mathcal{R}_i - 1}$
 4 end
 5 for $s \in S_i$ do  $\zeta_s = \frac{c_s - z_s - (\mathcal{R}_i - 1)x_s^s}{\mathcal{R}_i - 1}$
 6 for $v \in V_i \setminus S_i$ do  $\zeta_v = \frac{c_v}{\mathcal{R}_i - 1}$
 7 for $h \in H$ do  $\zeta_h = \frac{c_h}{\mathcal{R}_i}$
 8 for $s \in S_i, m \in N_i \setminus \{s\}$ do   // depth-2 trees
 9     $x_s^m = (z_s - x_s^s)\frac{\zeta_m}{\sum_{v \in V_i} \zeta_v + \sum_{h \in H} \zeta_h}$
10 end
```

In type (3) trees, the helpers would retrieve one copy of content from sources first before they could relay content to $\mathcal{R}_i$ receivers in swarm i. However, unlike the conference participating users, helpers themselves do not require the received content for viewing, which occupies up to $1/\mathcal{R}_i$ bandwidth helpers contribute. In terms of system utility, the content retrieval by helpers causes inevitable bandwidth overhead. To facilitate the following formulation, we formally define it as follows.

Definition The *helper bandwidth efficiency factor* of swarm i is defined as $\rho_i = \frac{\mathcal{R}_i - 1}{\mathcal{R}_i}$.

Now we can reformulate the utility maximization problem using $\{z_s\}$ and $\{f_i^h\}$ as follows.

$$\textbf{ID-OPT} \quad \max_{z_s, f_i^h} \sum_{i \in I} \sum_{s \in S_i} \mathcal{R}_i U_s(z_s) - \sum_{h \in H} G_h(\sum_{i \in I} f_i^h) \quad (7)$$

subject to

$$\begin{aligned} z_s &\le \min(c_s, e_s), \quad \forall s \in S_i, i \in I && (8) \\ \sum_{i \in I} f_i^h &\le c_h, \quad \forall h \in H && (9) \\ \sum_{s \in S_i} \mathcal{R}_i z_s &\le \sum_{v \in V_i} c_v + \rho_i \sum_{h \in H} f_i^h, \quad \forall i \in I && (10) \\ z_s, f_i^h &\ge 0, \quad \forall s \in S_i, i \in I, h \in H. && (11) \end{aligned}$$

The constraints (8)(10) actually explain the multicast rate region presented in Theorem 1. Equation (9) states the upload capacity limitations of helpers.

IV. Distributed Algorithms for Independent Conferencing System

In this section, we present our distributed solutions, proximal approximation based algorithm and marginal utility driven algorithm, for optimal bandwidth sharing in conferencing systems where swarms are independent and share the bandwidth of helpers.

A. Proximal Approximation Based Algorithm

1) Main Steps of Proximal Algorithm: Unfortunately, the reformulated problem ID-OPT is nonlinear and still not strictly concave in f_i^h. Directly solving it with dual approach in a distributed manner, such as the subgradient method [19], may incur oscillation when the solution is not unique lacking of strictly concavity, which is not amenable for practical implementation. Therefore, we resort to the proximal optimization algorithm [20], through which we can make the primal target function strictly concave by adding a quadratic term and still get an unique optimal solution of the original problem. In our algorithm, a quadratic term $-\frac{c}{2}||\mathbf{f}-\mathbf{d}||_2^2 = -\frac{c}{2}\sum_h\sum_i(f_i^h - d_i^h)^2$ is then added to the objective function to make it strictly concave in f_i^h, where $\mathbf{f}=\{f_i^h\}$, $\mathbf{d}$ is an additional vector and c is a positive constant. The proximal algorithm operates in iterations. At the t-th iteration, the problem is solved in two steps.

Step (1) Fix $d_i^h = d_i^h(t)$ for all $h \in H, i \in I$ and solve the following problem to get the optimal solution $f_i^h(t)$.

$$\max \sum_{i\in I}\sum_{s\in S_i}\mathcal{R}_i U_s(z_s) - \sum_{h\in H} G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}||\mathbf{f}-\mathbf{d}||_2^2 \quad (12)$$

subject to the constraints (8)(9)(10)(11).

Step (2) Set $d_i^h(t+1) = f_i^h(t)$ for all $h \in H, i \in I$.

After we get the optimal $\mathbf{f}^*(t)$ in the first step, we then assign $\mathbf{d}(t+1) = \mathbf{f}^*(t)$ and begin the next iteration. It requires convergence at two levels. An outer level adjustment in step (2) needs to be conducted after the inner level iteration converges in step (1). Our later simulation results show that this algorithm converges fast. (To further facilitate the implementation, some extended approach [21] even allows the fixed number of inner iteration in step (1) and still guarantees the convergence of the entire problem.)

2) Dual Decomposition: To solve (12) in the first step, we apply the dual decomposition techniques. We relax the constraint (10) with Lagrangian multipliers λ_i. Then we can get the Lagrangian function

$$\begin{aligned} L(z_s, f_i^h, \lambda) = & \sum_{i\in I}\sum_{s\in S_i}\mathcal{R}_i U_s(z_s) - \sum_{i\in I}\lambda_i\sum_{s\in S_i}\mathcal{R}_i z_s \\ & -\sum_{h\in H} G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}\sum_{h\in H}\sum_{i\in I}(f_i^h - d_i^h)^2 \\ & +\sum_{i\in I}\lambda_i\rho_i\sum_{h\in H} f_i^h + \sum_{i\in I}\lambda_i\sum_{v\in V_i} c_v. \quad (13)\end{aligned}$$

By duality, we obtain the following equivalent dual problem

$$\min g_{\lambda\geq 0}(\lambda) = \min\max L_{\lambda\geq 0}(z_s, f_i^h, \lambda). \quad (14)$$

We can observe that the problem has nice separable property for decomposition. Thus, we decompose the problem into two sub-problems, which can be resolved in a distributed manner.

Source multicast rate adjustment: in swarm i, given λ_i, each source s solves a local optimization problem:

$$\max_{0\leq z_s\leq min(c_s,e_s)} \mathcal{R}_i U_s(z_s) - \lambda_i\mathcal{R}_i z_s. \quad (15)$$

Accordingly, s should adjust its multicast rate as follows:

$$z_s = \begin{cases} 0 & \text{if } U_s'(0) < \lambda_i \\ (U_s')^{-1}(\lambda_i) & \text{else if } U_s'(\min(c_s,e_s)) \leq \lambda_i \\ \min(c_s,e_s) & \text{otherwise.}\end{cases} \quad (16)$$

Helper bandwidth allocation: given λ_i, each helper node h also solves a local optimization problem:

$$\max_{0\leq\sum_{i\in I} f_i^h\leq c_h} -G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}\sum_{i\in I}(f_i^h - d_i^h)^2 + \sum_{i\in I}\lambda_i\rho_i f_i^h. \quad (17)$$

The above sub-problem is strictly concave in f_i^h and can be directly solved. Under the Karush-Kuhn-Tucker conditions [22], it can be solved by the approach in [21] with the complexity of $O(|I|log(|I|))$.

After source and helper nodes adjust their rates, the Lagrangian multipliers will be updated to solve the dual problem with gradient projection algorithm [20] at $\hat{t}$th iteration

$$\lambda_i(\hat{t}+1) = [\lambda_i(\hat{t}) + \theta(\hat{t})(\sum_{s\in S_i}\mathcal{R}_i z_s(\hat{t}) - \sum_{v\in V_i} c_v - \rho_i\sum_{h\in H} f_i^h(\hat{t}))]^+, \quad (18)$$

where θ is a positive stepsize to guarantee the convergence and $[\cdot]^+$ means the projection onto the domain of non-negative real number. Due to the strict concavity in z_s and f_i^h, the optimal solution of the dual problem is primal feasible.

To facilitate the information exchange among network entities, communication protocols for the distributed algorithms can be developed as follows. One node in each swarm can be promoted as the *swarm coordinator* to communicate with helpers. The coordinator maintains the Lagrangian multiplier λ associated with its own swarm. It broadcasts the latest multiplier to the sources in the same swarm and all helpers. Upon receiving the multiplier information, the helpers decide the bandwidth allocation among the swarms, and the sources in each swarm adjust their multicast rates accordingly. Given the allocated resources of helpers, the sources try to grab resources to reach their targeted multicast rates. They will notify the coordinators about the resource gap information. Then the coordinators can update the multiplier information according to (18) after they collect information from all sources. The updated multipliers are broadcast to all sources and helpers to trigger the next iteration. The system finally enters the equilibrium after multiple iterations.

B. Marginal Utility Driven Algorithm

Next, we present an alternative approach which relies on the optimality condition and primal decomposition [23]. It incurs less computation overhead and iterations. Furthermore, the optimal sharing strategy for cooperative swarms in the following section is also developed based on this approach.

We introduce ϕ_i^h to denote the fraction of the bandwidth provided by helper h that is used by swarm i, i.e., $f_i^h = \phi_i^h c_h$. For swarm i, the aggregate bandwidth from helpers is $\sum_h f_i^h = \sum_h \phi_i^h c_h$. To solve the problem ID-OPT, we first consider intra-swarm source rate adaptation problem with fixed helper bandwidth fraction variable ϕ. For swarm i, we have

$$\textbf{IA-MUD} \quad \max_{0\leq z_s\leq\min(c_s,e_s)}\sum_{s\in S_i}\mathcal{R}_i U_s(z_s) \quad (19)$$

subject to

$$\sum_{s\in S_i}\mathcal{R}_i z_s \leq \sum_{v\in V_i} c_v + \rho_i\sum_{h\in H}\phi_i^h c_h. \quad (20)$$

The above problem is strictly concave with respect to source multicast rate z. Let $\mathbf{U}^i$ be the aggregate utility of all users in swarm i, i.e., $\mathbf{U}^i = \sum_{s\in S_i} \mathcal{R}_i U_s(z_s)$. We use $\mathbf{U}^i_{opt}(\phi)$ denote its maximum. Then the original problem can be transformed as follows.

$$\textbf{IT-MUD} \quad \max_{0\leq\phi} \sum_{i\in I} \mathbf{U}^i_{opt}(\phi) \tag{21}$$

subject to

$$\sum_{i\in I} \phi^h_i \leq 1, \forall h \in H. \tag{22}$$

In this primal decomposition, peers in a swarm can be easily scheduled to first use up their own resources before resorting to the helpers, thus we let $G_h = 0$ to ease the presentation. (Otherwise, instead of contributing the maximum of their upload capacity, helpers have to further compare the marginal cost of their bandwidth expenditure and the maximum marginal utility of swarms to determine optimally how much they would upload in total.)

Solving the problem consists of two levels of optimizations. At the inner level, sources in each swarm adapt their multicast rates to reach the optimum of IA-MUD. At the outer level, after swarms finish the inner level optimization and enter the equilibrium, helpers adjust their resource allocation according to IT-MUD to maximize the system-wide performance. The system converges to global optimum in iterations.

1) Intra-swarm Source Rate Adaptation: We first focus on the problem of intra-swarm optimization. Given the allocated bandwidth from helpers, sources in each swarm adjust their multicast rates to maximize the utility. The IA-MUD problem is similar to a distributed routing problem, and we have the following sufficient and necessary optimality conditions on the multicast rates of sources.

Theorem 2: Let $\mathbf{z}^*$ be the optimal multicast rates of the IA-OPT problem. For any two sources p and q, if it is feasible to switch some upload resources distributing the content of source p to increase the multicast rate of source q (i.e., $z_p > 0, z_q < \min(c_q, e_q)$), we will obtain $\frac{\partial \mathbf{U}^i(z^*)}{\partial z_q} \leq \frac{\partial \mathbf{U}^i(z^*)}{\partial z_p}$. That is, the multicast rates of sources p and q should satisfy

- $\frac{dU_p(z^*_p)}{dz_p} \leq \frac{dU_q(z^*_q)}{dz_q}$, if $z^*_p = 0, z^*_q > 0$
- $\frac{dU_p(z^*_p)}{dz_p} = \frac{dU_q(z^*_q)}{dz_q}$, if $z^*_p \in (0, \min(e_p, c_p))$ and $z^*_q \in (0, \min(e_q, c_q))$
- $\frac{dU_p(z^*_p)}{dz_p} \geq \frac{dU_q(z^*_q)}{dz_q}$, if $z^*_p = \min(e_p, c_p)$ and $z^*_q \in (0, \min(e_q, c_q))$.

Proof: Please refer to the technical report [24]. ■

As $\mathbf{U}^i$ is linear with the summation of source utility functions and concave, the first derivative $\frac{dU_p(z^*_p)}{dz_p}$ of U_p is proportional to the marginal utility of the aggregate $\mathbf{U}^i$ with respect to z_p. From the theorem, we can conclude the following properties at system optimum: *1) Saturated sources with maximal multicast rates have larger marginal utility than unsaturated sources 2) Ruling out the saturated ones, only the sources with largest marginal utility have positive multicast rates*. It also enables us to develop the heuristic to allocate upload resources: *the multicast rate of the source with the largest marginal utility should be increased*.

Based on the drawn heuristic, we can develop a *centralized* solution of the IA-MUD problem. Equation (10) defines the multicast rate region the sources can achieve. *Beginning with $z_s = 0$ ($\forall s \in S_i$), we identify one source p with the largest marginal utility and increases its multicast rate. Since $\mathbf{U}^i$ is concave, $\frac{\partial \mathbf{U}^i}{\partial z_p}$ decreases as z_p increases. We increase the multicast rate z_p until its marginal utility is no longer the largest or it reaches its upper limit,* $\min(c_p, e_p)$*. The source reaching its maximal multicast rate would be ruled out and not involved in the following procedures. Then we identify a new source q with current largest marginal utility and increase its multicast rate. The operation repeats until the summation of all source multicast rate reaches the bound.* According to the optimality condition stated in the theorem, the above multicast rate assignment is adjusted towards the system optimum.

To facilitate the practical implementation, we further propose a *distributed* marginal utility driven algorithm. In a swarm, sources are assigned with initial multicast rates. Sources notify each other their current marginal utility and multicast rates periodically. One source tries to identify another source whose multicast rate is less than its maximal rate and has larger marginal utility. Then two sources can establish a process to let the one with smaller marginal utility switch its upload resources to increase the multicast rate of the other. Each source operates locally and the system utility converges to the optimum eventually. We discuss the details of the distributed implementation as follows.

First, sources identify proper ones and establish the rate adaptation processes in pairs. Each source maintains a set of other sources, with which the rate adaptation can be potentially conducted. For instance, source p maintains a candidate set $\{q | U'_q(z_q) < U'_p(z_p) - \epsilon, z_q > 0, q \in S_i\}$, where ϵ is the configurable threshold to tolerate the marginal utility gaps among sources. One source would not consider to adjust rate with another, if the gap between their marginal utility is less than ϵ. If a source has not reached its maximal rate, it periodically picks one from the set and sends out request for rate adaptation.

Second, sources negotiate to adapt the rates in the pair-wise adaptation process. We assume peers would not unveil their utility functions to others. To facilitate the negotiation, the request sender would prepare a negotiation option list with maximum M entries, each of which contains the amount of adjustment and its expected marginal utility. Algorithm 2 gives an example how source p prepare the list according to its marginal utility difference between source q. After source q receives the list, it chooses the appropriate and feasible option j from the list to make their marginal utility closest, such that $\arg\max_j U'_p(z_p + \delta_j) > U'_q(z_q - \delta_j)$. Then source q shifts resources to increase source p's multicast rate by the amount equivalent to the first item of entry j. Finally both of them broadcast their new marginal utility to all other sources in the swarm.

In the above pair-wise rate adaptation process, we only need $O(\log_M(\frac{\Delta_{pq}}{\epsilon}))$ steps to narrow down the gap of the marginal utility of two sources p and q to make it less than threshold ϵ. Furthermore, the sources can adapt asynchronously to accelerate the adaptation processes. Our later simulation results show

Algorithm 2: Option list for negotiation

```
Input  : source $p$ ($U_p'(z_p)$,$z_p$),$U_q'(z_q)$
Output: Option list $L$ of item $< d, m >$
1  $w$ = InitializeList()
2  $\Delta_{pq} = U_p'(z_p) - U_q'(z_q)$
3  for $i = 1 : M$ do
4      $w(i).d = U_p'^{-1}\left(U_p'(z_p) - \frac{\Delta_{pq}}{M} i\right) - z_p$
5      $w(i).m = U_p'(z_p) - \frac{\Delta_{pq}}{M} i$
6      if $w(i).d + z_p > \min(c_p, e_p)$ then
7          $w(i).d = \min(c_p, e_p) - z_p$
8          $w(i).m = U_p'(\min(c_p, e_p)$
9          break
10     end
11 end
12 for $j = 1 : i$ do  InsertList($L, w(j)$)
```

that the intra-swarm adaptation enables the system utility to converge to the optimum fast.

2) Inter-swarm Helper Bandwidth Allocation: After sources in a swarm have conducted the previous pair-wise adaptation process, the utility of the swarm converges and the swarm enters an *equilibrium* state. Then the coordinator of the swarm broadcasts the marginal utility information to all helpers. Helpers collect the information from all swarms and update the bandwidth allocations among all swarms accordingly, which then trigger swarms to adapt source rates towards new equilibrium states. Before we discuss the helper bandwidth allocation, we first investigate the marginal utility of swarm to helpers.

Definition A swarm is called *choked* if all sources in the swarm reach the maximal multicast rates. For instance, swarm i is choked, then $z_s = \min(e_s, c_s)$, $\forall s \in S_i$.

Definition The *marginal utility of swarm i to helper's bandwidth* in its equilibrium state is defined as $\mu_i = \rho_i \min_{\{z_s>0, s\in S_i\}} U_s'(z_s)$.

As stated in the previous section, for a swarm in equilibrium, the sources with positive multicast rates have the same marginal utility after we rule out the sources whose rates are already saturated. These sources are also the ones with current largest marginal utility, and we could increase their rates if there are additional resources. Based on Equation (10),we know for unchoked swarm i, the space for source rates to increase is $\sum_{s\in S_i} \mathcal{R}_i \Delta z_s = \rho_i \Delta f$ given additional helper bandwidth Δf. To maximize the increase in the swarm's aggregate utility, one should increase the rates of sources with the largest marginal utility. Hence, the optimal utility gain of swarm i by the additional helper bandwidth is obtained

$$\begin{aligned}\Delta \mathbf{U}_{opt}^i &= \sum_{s\in S_i} \mathcal{R}_i (U_s(z_s + \Delta z_s) - U_s(z_s)) \\ &\approx \min_{z_s>0, s\in S_i} U_s'(z_s) \sum_{s\in S_i} \mathcal{R}_i \Delta z_s \\ &= \rho_i \min_{z_s>0, s\in S_i} U_s'(z_s) \Delta f. \end{aligned} \tag{23}$$

As for a choked swarm, the source with the smallest marginal utility is also the last one to reach its maximal multicast rate if we keep increasing the incoming helper bandwidth before the swarm is choked. Hence, the marginal utility of swarm to helper bandwidth implies the optimal aggregate utility gain/loss of users in the swarm if we increase/decrease the incoming helper bandwidth.

Now we present the inter-swarm helper bandwidth allocation algorithm. Helpers adapt the distribution among swarms individually and dynamically. Since $\mathbf{U}_{opt}^i$ is also concave function with respect to f_i^h, intuitively we should put more resources to swarms with larger marginal utility. Similar to the distributed algorithm in distributed optimal routing [25], our approach could operate in the following way towards the equilibrium:

At stage t, given $\phi(t)$, suppose $w = \arg\max_{j\in J} \mu_j$, where $J = \{i | \text{swarm } i \in I, \text{not choked}\}$. The split ratio ϕ_i^h of helper h bandwidth for swarm i would be updated according to

$$\phi_i^h(t+1) = \phi_i^h(t) + \gamma_i^h(t) \tag{24}$$

with

$$\gamma_i^h(t) = \begin{cases} -\min\{\phi_i^h(t), \kappa_n(\mu_w - \mu_i)\} & \text{if } i \neq w \\ -\sum_{k\neq w} \gamma_k^h & \text{if } i = w \\ 0 & \text{if } \mu_i \geq \mu_w \text{ and choked,} \end{cases} \tag{25}$$

where κ_n is a positive scalar stepsize.

In the above algorithm, for swarms with $\mu_i \geq \mu_w$ but choked, the helper bandwidth fractions on these swarms remain unchanged because they are unable to accept further resources. If the amount of the shifted resources in one iteration is more than that makes the swarm choked, the helpers would spend the residual bandwidth to the swarm with the largest marginal utility in the next iteration. The iterations enable helpers to shift resources from swarms with smaller marginal utility to the one with the largest μ. Finally, the system enters equilibrium and the relationship of the marginal utility among swarms have the similar relationship to that of sources in a single swarm stated in Theorem 2: choked swarms have larger marginal utility to helper bandwidth than unchoked ones; only the swarms with largest marginal utility receive helper bandwidth after the choked swarms are ruled out.

V. Distributed Algorithm for Cooperative Conferencing System

In previous sections, we have discussed the distributed implementation for sharing among independent swarms. In this section, we investigate the resource sharing strategies among cooperative swarms.

The problem of sharing among cooperative swarms can also be formulated under the utility maximization framework. In cooperative conferencing system, swarms are able to share bandwidth resources from not only helpers but also each other. We introduce π_i^j to denote the amount of bandwidth resources shared by swarm j with swarm i. From the perspective of swarm i, the nodes in swarm j those share bandwidth are regarded as extra helpers, and additional depth-2 trees leveraging them as relay nodes would be built to accelerate the content distribution. Therefore, they are subject to the same bandwidth overhead as dedicated helpers for relaying traffic in swarm i. So, the rate region of sources in swarm i can be formulated as follows,

$$\sum_{s\in S_i}\mathcal{R}_i z_s \le \sum_{v\in V_i} c_v - \sum_{j\in I, j\neq i}\pi^i_j + \rho_i\Big(\sum_{h\in H} f^h_i + \sum_{j\in I, j\neq i}\pi^j_i\Big). \tag{26}$$

The resources one swarm can spare for sharing should be limited by the upload capacities of the participating users, which yields

$$\sum_{j\in I, j\neq i}\pi^i_j \le \sum_{v\in V_i} c_v, \forall i\in I. \tag{27}$$

We can leverage the optimization based approach similar to that in Section IV-A to solve the problem. However, due to the increased number of variables and computation complexity, the convergence speed could be slowed and more communication overhead would be incurred. Furthermore, the solution by solving the optimization problem may cause a swarm to widely share resources with other swarms, which results in excessively large number of connections not amenable for practical implementation. To ease the problem solving, we propose a distributed approach based on the marginal utility driven algorithm.

A. Cooperation Criteria

In cooperative conferencing system, although users in a swarm can share bandwidth to help other swarms as well, they are different from dedicated helpers in increasing the system utility. We discuss several rules of cooperative sharing among swarms before we present the distributed algorithm.

1) Sharing among swarms: From Equation (26), we can learn that if swarm i shares its own resources with swarm j, it can be regarded as another helper to swarm j and subject to the bandwidth overhead determined by the helper bandwidth efficiency factor ρ_j of swarm j. The marginal utility gain of swarm j is μ_j. However, at the same time, swarm i suffers utility loss by shifting its own resources to support swarm j. The optimal way of sharing is to reduce the multicast rate of the source with the smallest marginal utility, i.e., $\min_{\{z_s>0, s\in S_i\}} U'_s(z_s)$, which is equal to μ_i/ρ_i. Hence, *in order to improve the system utility by shifting the resources from swarm i to swarm j, the following criteria should be satisfied*

$$\mu_i/\rho_i < \mu_j. \tag{28}$$

The P2P system is dynamic with frequent swarm and peer churn, and the relationship among the marginal utility of swarms also varies correspondingly. A relatively resource-rich swarm perhaps becomes relatively resource-poor at next moment, and vice versa. Due to the cooperative sharing, swarms can be providers and receivers of resources. If a swarm provides resources to others and receives resources from others at the same time, it acts as a *resource relay*, in which the resources cannot be fully utilized to maximize the system utility. Figure 3 illustrates an example of the resource relay. Swarm i receives δ amount of resources from swarm

$$\boxed{j} \xrightarrow{\delta} \Big(+\rho_i\delta \;\; \boxed{i} \;\; -\delta\Big) \xrightarrow{\delta} \boxed{k}$$

Fig. 3. Bandwidth overhead in resource relay.

TABLE II
SWARM STATUS

Type	μ	Choked	Resource Receiver
NC-R	$\mu=\hat{\mu}$	N	Y
C-R	$\mu\ge\hat{\mu}$	Y	Y
NC-NR	$\mu\le\hat{\mu}$	N	N
C-NR	Any	Y	N

j and provides the same amount of resources to swarm k simultaneously. Because only $\rho_i\delta$ amount of resources from swarm j effectively increase the rates of sources in swarm i, the swarms waste $(1-\rho_i)\delta$ amount of resources compared with that swarm j directly provides δ amount of resources to swarm k. Thus, *resource relay should be avoided to fully utilize the resources*.

2) Marginal utility of swarm in equilibrium: In the marginal utility driven algorithm for independent swarms, helpers will re-allocate their bandwidth among swarms after the intra-swarm optimization in swarms. The re-allocation of helpers turns out to reduce the differences of marginal utility of swarms. Now the cooperative sharing can further narrow the gaps. Figure 4 shows an example of the relationship among marginal utility of swarms in system equilibrium. Suppose

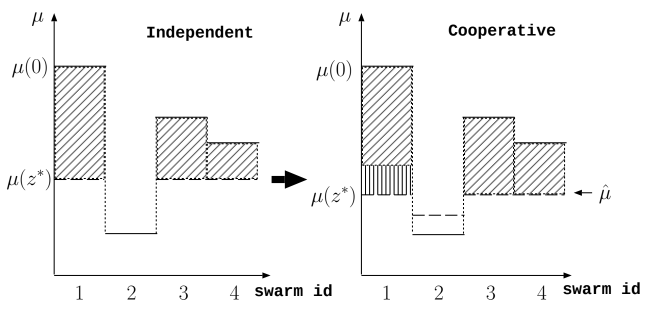


Fig. 4. Marginal utility of swarm in equilibrium.

no swarm becomes choked during the resource sharing. The marginal utility μ_2 of swarm 2 is far less than that of others. In independent conferencing system, helpers are capable of leveling the marginal utility of swarms except that of swarm 2. While in cooperative conferencing system, swarm 2 could further share its resources with other swarms to improve the system utility. In the Figure, swarm 2 spares resources to help swarm 1 and μ_2 increases. The marginal utility of other three swarms are finally stabilized at a lower level $\hat{\mu}$. Based on the sharing criteria, we can deduce that this cooperative sharing is optimal if $\mu_2/\rho_2 = \hat{\mu}$.

The relationship of marginal utility of swarms could be more complex, taking into account a swarm could be choked. Suppose no resource relay exists in the system. Let $\hat{\mu}$ be the largest marginal utility of swarms which are not choked. After helpers finish their bandwidth re-allocation among the swarms, the status of swarms can be generally categorized into four types listed in Table II.

NC-R	This type of swarms receive outside resources and can potentially receive more, like swarm 1, 3 and 4 in Figure 4. The marginal utility of them have been decreased to the same level $\hat{\mu}$.
C-R	They receive resources and become choked. The marginal utility of them is no smaller than $\hat{\mu}$.
NC-NR	They are not offered with resources because of small marginal utility, like swarm 2 in Figure 4. The marginal utility of them is no larger than $\hat{\mu}$.
C-NR	The sources in these resource-rich swarms can achieve the maximal multicast rates without help from outside. Their marginal utility can be of any relationship with $\hat{\mu}$.

Based on the possible status of swarms, we can determine how to conduct cooperative sharing among swarms to improve the system utility. First, we notice that swarm of type C-NR may have surplus bandwidth not fully utilized, which could serve other swarms like the dedicated helpers. The amount of surplus bandwidth of swarm j in this type equals to $\sum_{v\in V_j} c_v - \mathcal{R}_j \sum_s \min(e_s, c_s)$. Second, to maximize the utility gain, the swarms of type NC-R should be the ones to receive further resources at next cooperative sharing procedure. The resource provider should be the swarms of type NC-NR and C-NR with marginal utility satisfying the sharing criteria based on Equation (28).

As there possibly exist multiple swarms of type NC-R that require to receive further resources, instead of choosing one of them randomly at each cooperative sharing procedure, a helping swarm could stick to one of them and share its resources if the resource receiver is still of type NC-R. In this way, we can effectively minimize the number of swarms sharing the resources of a swarm such that the number of parallel connections is reduced for practical implementation.

B. Cooperation Strategies

With the above insights, we could develop a distributed algorithm for swarms to share resources with others. An additional resource sharing adjustment would be conducted at a larger timescale based on the marginal utility driven algorithm for independent conferencing system. Each time the helper bandwidth allocation is stabilized, swarms would conduct this resource sharing adjustment. The system approaches optimum in iterations.

The resources those a swarm can share come from the bandwidth resources of its participating users. For swarms of type C-NR, since the surplus bandwidth resources would be spared to help other swarms as dedicated helpers, we only consider their remaining resources in the resource sharing adjustment procedure.

It is known that resource relay should be avoided. If swarm i has accepted resources from helpers, i.e., $\sum_{h\in H} \phi_i^h > 0$, or from other swarms, i.e., $\sum_{j\in I, j\neq i} \psi_i^j > 0$, where ψ_i^j denotes the fraction of resources of swarm j sharing with swarm i, it does not initiate any resource sharing request and would act passively.

The sharing adjustment of a swarm is similar to the helper bandwidth allocation strategy. The basic idea is that each swarm iteratively identifies one proper resource receiver to maximize the utility gain of cooperative sharing. In the following, we discuss how to determine the ideal resource receiver w for swarm i. Note that we can identify $\hat{\mu}$, the largest marginal utility of swarms which are not choked in the system. For swarms of type NC-R, their marginal utility is equal to $\hat{\mu}$. If μ_i/ρ_i is larger than $\hat{\mu}$, we get $w = i$ according to cooperation criteria. It achieves larger utility gain by moving back its resources helping other swarms. Otherwise, swarm i should choose a swarm of type NC-R as the receiver. To minimize the number of cross-swarm connections, it would preferentially choose one from the candidate set $\{j|\psi_j^i > 0, \mu_j = \hat{\mu} \text{ and not choked}, j \in I\}$, which contains the unchoked type NC-R swarms that already receive the resources from swarm i. In case this set is empty, swarm i chooses one from the remaining swarms of type NC-R.

The resource sharing adjustment would be conducted as follows:

At stage t, given $\psi(t)$ and w identified for swarm i receiving no resources from outside, the fraction of resources of swarm i would be updated according to

$$\psi_j^i(t+1) = \psi_j^i(t) + \vartheta_j^i(t) \tag{29}$$

with

$$\vartheta_j^i(t) = \begin{cases} -\min\{\psi_j^i(t), \kappa_n(\bar{\mu}_w - \bar{\mu}_j)\} & \text{if } j \neq w \\ -\sum_{k\neq w} \vartheta_k^i & \text{if } j = w \\ 0 & \text{if } \bar{\mu}_j \geq \bar{\mu}_w \text{ and choked,} \end{cases} \tag{30}$$

where κ_n is a positive scalar stepsize and

$$\bar{\mu}_j = \begin{cases} \mu_i/\rho_i & \text{if } j = i \\ \mu_j & \text{otherwise.} \end{cases}$$

The actual amount of resource sharing adjustment also depends on the status of the resource receiver. Suppose swarm i intends to provide B_j^i amount of resources to swarm j. If swarm j has not shared its resources with others, i.e., $\sum_{k\in I, k\neq j} \psi_k^j = 0$, it would accept the offer from swarm i. Otherwise, to avoid resource relay, instead of accepting the offer, it takes back its own resources sharing with other swarms. The amount of resources to be taken back would be the minimum of its sharing resources and the offer from swarm i, i.e., $\min(\sum_{v\in V_j} c_v \sum_j \psi_j^i, B_j^i)$. For the swarms originally sharing the resources of swarm j, swarm i helps them with resources equivalent to those taken back by swarm j. Similarly, during the helper bandwidth allocation procedure, if a swarm with resources shared with others gets resource offer from helpers, it takes back its own resources sharing with others instead of accepting the helper resources as well.

For the sake of clarity, in the above discussion, we present the cooperative strategies from the perspective of swarm-level resource sharing. Actually, further considerations are required on how each user inside the providing swarm shares in the offering resources. To minimize the utility loss due to resource sharing with other swarms, when a swarm determines to share its resources, it preferentially chooses those non-source participating users to spare their resources. Once only the resources of sources remain and those of non-source participating users are used up for sharing, it is necessary for the sources to optimally determine the resource sharing taking into account their marginal utility. We present the details in the technical report [24].

VI. Numerical Results

In this section, we provide the numerical results of the proposed algorithms. We use a small network with at most four conferencing swarms to clearly present how the system evolves. In each swarm, there are two to three sources and other four participating users. Table III lists the normalized bandwidth of peers in all swarms. There are two helpers with normalized bandwidth 40 and 60. The maximum multicast rate for all sources is set to be 5, i.e., $e_s = 5$ for any source s. In this setting, swarm 3 has abundant resources that the sources of swarm 3 can all reach the maximal multicast rate without any helper resources. On the contrary, swarm 1 has the least amount of resources. The utility function for sources is set to be $U_s = \mathcal{C}_s \log(z_s + 1)$, where $\mathcal{C}_s$ is the utility weight pertaining to source s. The cost function for helper h is set to be $G_h = f_h/(c_h - f_h)$, where $f_h = \sum_{i \in I} f_i^h$.

TABLE III
Node Bandwidth Setting

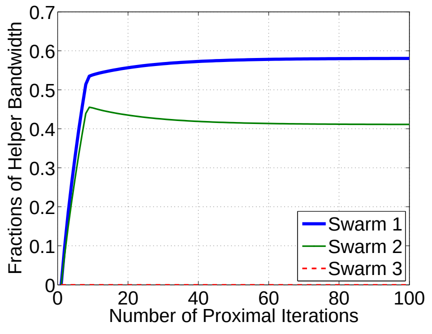
	Sources	Participating Users
Swarm 1	5,5,4	5,4,5,3
Swarm 2	10,7,4	5,4,8,5
Swarm 3	10,20	5,2,3
Swarm 4	6,6,3	6,8,10

A. Independent Conferencing System

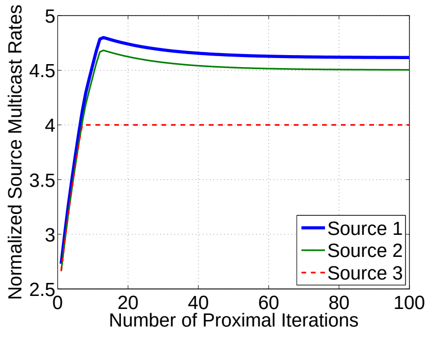
1) Proximal Approximation Based Algorithm: In this simulation, only the first three swarms join the system. We let the inner loop iterate at most 500 times. Swarm 1 has the least amount of resources, while swarm 3 has the maximum amount of resources. Helpers are expected to contribute more to swarm 1. Figure 5(a) shows how the bandwidth fractions of helper 1 evolve versus the number of proximal iterations. The resources of the helper would be allocated to the first two swarms accordingly and swarm 3 does not need any helper resources. Figure 5(b) presents the evolution of the multicast rates of all sources in swarm 2 during the iterations. We let the utility weight of source 1 be slightly larger than other sources. The multicast rate of source 1 is slightly larger in return. The rate of source 3 increases first with other sources until it reaches the upload capacity limit 4 and then keeps stable. We can observe that the resource allocation of helpers can be coordinated and the optimal multicast rates of sources can be adjusted with utility weight setting.

2) Marginal Utility Driven Algorithm: First, we study the performance of Intra-swarm rate adaptation algorithm specifically. We let the threshold of marginal utility gap be $\epsilon = 0.1$ and the length of option list be $M = 10$. Figure 6 illustrates how the multicast rates of sources in a single swarm converge to the optimum by the pair-wise adaptation process. The utility weights of the three sources are in decreasing order. After few times of pair-wise adaptions, the aggregate system utility gets close to the optimum and the multicast rates of sources become stable. We let a new regular user join the ongoing conference suddenly in the middle of simulation. We can observe that the sources adapt the rates accordingly and quickly converge to the optimum again. This shows the robustness of the pair-wise adaptation procedure against peer churn.

Next, we investigate the performance of marginal utility driven algorithm and also examine its robustness under swarm churn. At the beginning, there are only the first three conferences. The 4th conferencing swarm would join the system in the middle. Figure 7(a) presents the evolution of helper bandwidth allocation. We can observe that the system is able to enter the equilibrium after helpers adjust less than 20 steps. Swarm 3 still has no helper bandwidth input. After swarm 4 joins the system, part of the bandwidth fraction allocated to



(a) Fractions of Helper Bandwidth



(b) Source Rate Evolution

Fig. 5. System evolution with proximal approximation based algorithm.

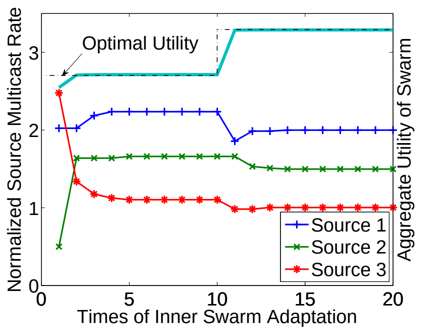
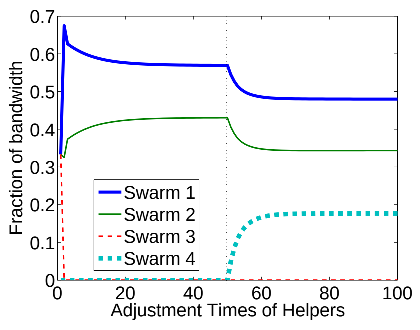
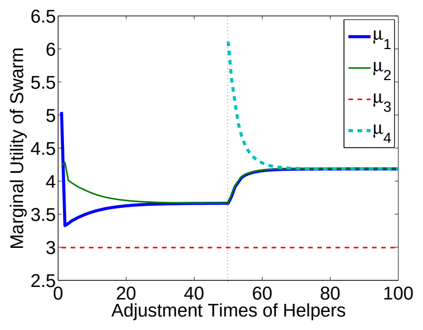


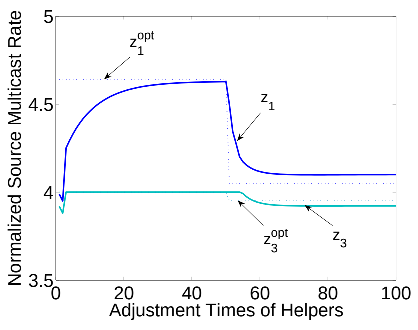
Fig. 6. Pair-wise Adaption in a single swarm.



(a) Bandwidth fraction of helpers

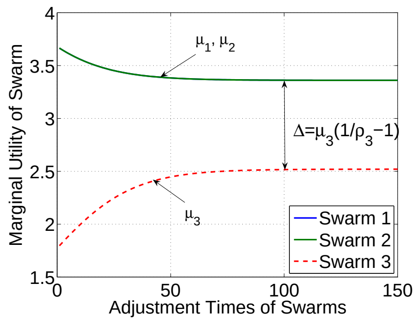


(b) Marginal utility of swarm

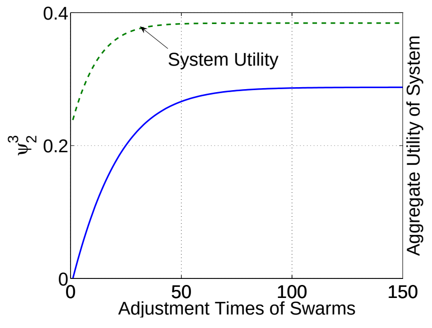


(c) Source multicast rate

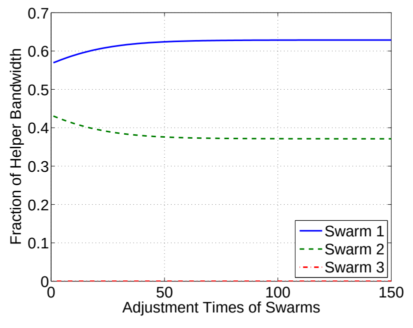
Fig. 7. Simulation results of marginal utility based algorithm. The are three conferencing swarms at the beginning and swarm 4 joins the system in the middle.



(a) Marginal utility of swarm



(b) Bandwidth fraction of swarm



(c) Bandwidth fraction of helpers

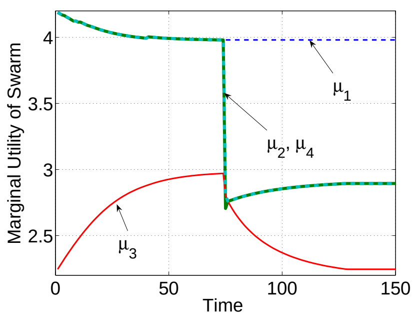
Fig. 8. System evolution in cooperative conferencing systems. Swarm 3 shares its bandwidth resources with other swarms.

the first two swarms is gradually shifted to the newly joined one. Figure 7(b) shows the marginal utility of swarms to helper bandwidth. The marginal utility of swarm 1 and 2, i.e., μ_1 and μ_2, converge to the same value fist. They change accordingly after the joining of swarm 4 and converge to a new level finally. Swarm 3 is choked and its marginal utility is always larger than that of others. This is consistent with the optimality conditions stated in Theorem 2. Figure 7(c) plots the multicast rate evolution of two sources of swarm 2. As the amount of incoming resources from helpers varies, the multicast rates of these two sources adapts to the changes accordingly. The small gap between the achieved rates and the optimal ones follows by the allowed marginal utility difference ϵ set in pair-wise adaptation algorithm. It is clear that the smaller ϵ is, the smaller the gap would be. However, it also results in increased intra-swarm adjustment times in pair-wise adaptation algorithm.

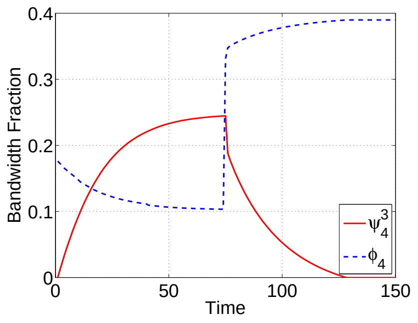
B. Cooperative Conferencing System

In cooperative conferencing system, swarms are allowed to share the bandwidth resources of participating users with others. Hence, Swarm 3, which is choked in independent conferencing system, now could spare its surplus resources to help augment the utility of the entire system. We first study the performance of the cooperative conferencing system in static scenario. There are only the first three swarms in the system. After the system enters equilibrium, the marginal utility of swarm 3 is far lower than that of other two swarms. Swarm 3 does not receive any resources from helpers and spares its own bandwidth to others gradually. Its marginal utility would increase as it gradually shifts resources to swarm 2 until $\mu_3/\rho_3 = \mu_2$, as shown in Figure 8(a). Swarm 3 only shares its resources with swarm 2 and keeps $\psi_1^3 = 0$. The algorithm effectively minimizes the number of swarms sharing resources with swarms 3 to reduce number of connections. Figure 8(b) plots the evolution of both the system utility and the bandwidth fraction from swarm 3 spent on swarm 2, i.e., ψ_2^3. The system utility increases as ψ_2^3 increases. Since the helpers has shared bandwidth in swarm 2, each time when swarm 3 shifts some resources to swarm 2, the helpers would move certain amount of resources previously spent in swarm 2 back to swarm 1. Figure 8(c) presents the adaptation of bandwidth fraction of helpers to the bandwidth sharing cooperation of swarms. Compared with the independent design, cooperative bandwidth sharing among swarms can further enhance the system utility.

Next, we investigate how the system adapts to the swarm churn. At the beginning, all four swarms join the system and swarm 1 leaves the system in the middle of the simulation. Figure 9 shows the evolution of marginal utility of swarms and bandwidth fraction of helpers. The system first enters the equilibrium with swarm 3 sharing resources with swarm 4. After the swarm departure happens, a part of helper resources originally assigned to swarm 1 is shifted to swarm 2 and swarm 4. The marginal utility of swarm 3, μ_3, is larger than that of others at that time. However, instead of receiving resources from helpers, swarm 3 takes its resources back



(a) Marginal utility of swarm



(b) Bandwidth fractions

Fig. 9. Swarm churn in cooperative conferencing systems. There are fours swarms at the beginning and swarm 4 leaves the system in the middle.

from swarm 4 gradually. Accordingly, helpers increase their bandwidth expenditure on swarm 4. That is, ψ_4^3 decreases and ϕ_4 increases. Finally, the system enters a new equilibrium.

VII. Simulation Results

To demonstrate the effectiveness of proposed algorithms, we further examine their performance with packet-level simulations.

A. Simulation Setting

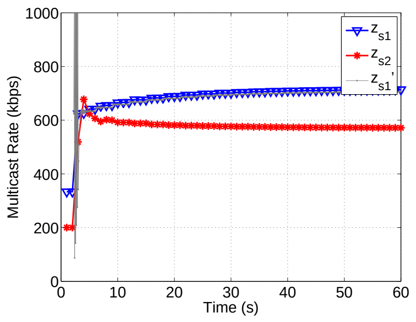
We developed a packet-level event-driven simulator in C++, in which all control and streaming packets are carefully simulated. In the simulator, we employ real-world node-to-node latency matrix used in [26] to simulate the end-to-end propagation delay. The average end-to-end propagation delay of nodes is 41ms. We also assume all control messages can be delivered reliably. We follow the common assumption that the user upload links are the only bottlenecks in the network. To make the simulation more realistic, three DSL types of nodes are assigned with upload bandwidth of 1Mbps, 384kbps and 128kbps based on the measurement study [27].

In this simulation, we employ three swarms and one helper node. The helper has upload bandwidth of 2Mbps to help these three swarms. Each swarm has around 2 sources and 4 other participating users. Specifically, swarm 3 has abundant resources that the sources of swarm 3 can all reach the maximal multicast rate without any helper resources. We let the source utility function to be $\mathcal{C}_s \log(z_s + 1)$ and the helper cost function to be zero.

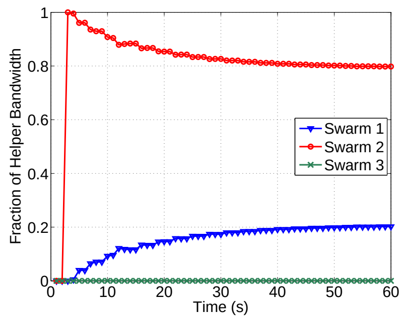
The implementation of intra-swarm data delivery along multicast trees is developed based on the framework of our proposed one-hop optimal data schedule algorithm [28] in fully-connected network. To facilitate the message exchanges among nodes for optimization, we further develop and implement the communication protocols in our packet-level simulator. The implementation of proximal approximation based algorithm is developed in the way discussed in Section IV-A. Taking into account the possible oscillations in optimization, we let sources adjust their rates based on their allocated resources every one second instead of every time when they get new optimization solutions on multicast rates. The helper would assign $\mathbf{d}$ in the outer-level adjustment every three seconds, which is the minimum time required for the convergence of inner iteration based on our simulation results. As for the marginal utility based algorithm, we let ϵ and κ to be 0.1 and 0.01, respectively. Sources notify each other their current marginal utility every 300 milliseconds, and the helper adjust its bandwidth allocation every one second.

B. Simulation Result

We investigate the performance of our proposed algorithms in independent swarm sharing scenario. Figure 10(a) shows



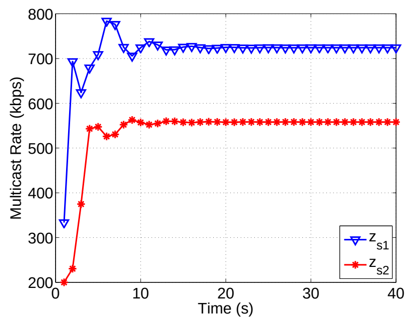
(a) Multicast rates



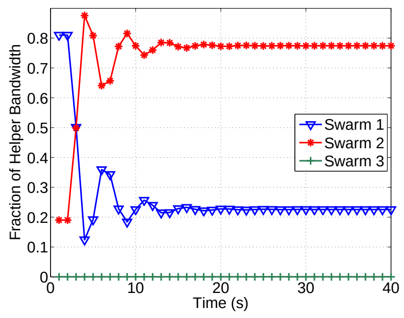
(b) Bandwidth fractions

Fig. 10. Proximal approximation based alogrithm.

the multicast rate evolution of two sources in two swarms. z_{s1} and z_{s2} denote the multicast rates of sources $s1$ and $s2$ in swarm 1 and 2, respectively. We can observe that the rates of these two sources enter equilibrium around 40 seconds after the simulation starts. z'_{s1} plots every single solution of the optimization, which has much oscillation at the beginning. Our implementation proves to be able to adjust source multicast rates smoothly. Figure 10(b) plots the evolution of the helper bandwidth fraction, which shows that the helper gradually adapts its bandwidth allocation to the optimum. The swarm 3 has abundant resources and needs no helper bandwidth.



(a) Multicast rates



(b) Bandwidth fractions

Fig. 11. Marginal Utility Driven alogrithm.

Figure 11 shows the system evolution with the marginal utility driven algorithm under the same setting. The system enters the equilibrium within 20 seconds, which is much faster than the proximal approximation based algorithm.

Furthermore, we record and compare the control traffic in these two algorithms. In proximal approximation based algorithm, the average number of control packets per second is 144 and the control traffic overhead is 1.16%. However, in marginal utility driven algorithm, the number of control packet per second is 25 and the overhead is 0.19%. we conclude that marginal utility driven algorithm has much less signaling overhead.

VIII. Conclusion & Discussion

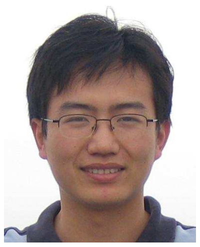
In this paper, we propose optimal cross-swarm bandwidth sharing strategies to address the bandwidth challenge in multi-swarm multi-party P2P conferencing systems. Specifically, we consider two sharing scenarios: swarms are independent and share a common pool of helper; swarms are cooperative and further share bandwidth directly with each other. For each scenario, we develop distributed algorithms for intra-swarm and inter-swarm bandwidth allocation under a utility-maximization framework. Through analysis and simulation, we show that the proposed algorithms are robust in the face of peer churn and swarm churn, and can dynamically allocate peer and helper bandwidth across swarms to achieve the system-wide optimum.

This paper focuses on optimal bandwidth sharing and future work will be conducted in the following directions. We will augment our algorithms to build the two-hop relay trees with short propagation delays, as video conferencing applications are also highly sensitive to delay. Taking into account signalling and coordination overhead incurred by the proposed optimal bandwidth sharing algorithms, to accommodate a large number of parallel conferences in real system, swarms and helpers can be grouped into clusters and apply the proposed algorithms within each cluster. How to group swarms and assign helpers can be further optimized by taking into account the ISP-friendly considerations [29]. In the next step, we are also interested in prototyping the system with the proposed distributed algorithms, and examining its performance in real Internet environment.

References

[1] MSN, “MSN Homepage,” http://www.msn.com.
[2] Skype, “Skpe Homepage,” http://www.skype.com.
[3] J. Li, P. A. Chou, and C. Zhang, “Mutualcast: An Efficient Mechanism for Content Distribution in a P2P Network,” in *Sigcomm Asia Workshop*, 2005.
[4] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. Chou, “Utility Maximization in Peer-to-peer Systems,” in *Proceedings of ACM SIGMETRICS*, 2008.
[5] L. Guo, S. Chen, Z. Xiao, E. Tan, and X. D. X. Zhang, “Measurements, analysis, and modeling of bittorrent-like systems,” in *Internet Measurement Conference (IMC 05)*, Berkeley, California, USA, Oct. 2005.
[6] D. Wu, C. Liang, Y. Liu, and K. Ross, “View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems,” in *Proceedings of INFOCOM Mini-Conference*, 2009.
[7] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, “Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture,” in *Proceedings of ACM SIGCOMM*, 2001.
[8] J. Lennox and H. Schulzrinne, “A Protocol for Reliable Decentralized Conferencing,” in *Proceedings of ACM NOSSDAV*, 2003.
[9] C. Luo, W. Wang, J. Tang, J. Sun, and J. Li, “A Multiparty Videoconferencing System Over an Application-Level Multicast Protocol,” in *IEEE Transactions on Multimedia*, 2007.
[10] M. Ponec, S. Sengupta, M. Chen, J. Li, and P. Chou, “Multi-rate Peer-to-Peer Video Conferencing: A Distributed Approach Using Scalable Coding,” in *Proceedings of ICME*, 2009.
[11] R. S. Peterson and E. G. Sirer, “Antfarm: Efficient Content Distribution with Managed Swarms,” in *Symposium on Networked System Design and Implementation*, 2009.
[12] C. Wu and B. Li, “Diverse: Application-Layer Service Differentiation in Peer-to-Peer Communications,” in *IEEE Journal on Selected Areas in Communications*, 2007.
[13] C. Wu, B. Li, and S. Zhao, “Multi-channel live p2p streaming: Refocusing on servers,” in *Proceedings of IEEE INFOCOM*, 2008.
[14] Y. Huang, T. Z. J. Fu, D.-M. Chiu, J. C. S. Lui, and C. Huang, “Challenges, Design and Analysis of a Large-scale P2P-VoD System,” in *Proceedings of ACM SIGCOMM*, 2008.
[15] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the h.264/avc standard,” in *IEEE TCSVT*, vol. 17, no. 9, 2007.
[16] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle, “Optimization Based Rate Control for Multicast with Network Coding,” in *Proceedings of IEEE INFOCOM*, 2007.
[17] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, “Polynomial Time Algorithms for Multicast Network Code Construction,” in *IEEE Transactions on Information Theory*, 2005.
[18] R. Kumar, Y. Liu, and K. Ross, “Stochastic fluid theory for P2P streaming systems,” in *Proceedings of IEEE INFOCOM*, 2007.
[19] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.
[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
[21] X. Lin and N. B. Shroff, “Utility Maximization for Communication Networks with Multi-path Routing,” in *IEEE Transactions on Automatic Control*, 2006.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[23] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," in *Proceedings of the IEEE*, vol. 95, no. 1, 2007.
[24] C. Liang, M. Zhao, and Y. Liu, "Optimal Bandwidth Sharing in Multi-Swarm Multi-Party P2P Video Conferencing Systems," *Technical Report, Polytechnic Institute of NYU*, October 2010, http://wan.poly.edu/cliang/pconf_tech_rep.pdf.
[25] R. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," in *IEEE Transactions on Communication*, 1977.
[26] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the power of pull-based streaming protocol: can we do better?" *IEEE Journal on Selected Areas in Communications*, 2007.
[27] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," in *Proceedings of IEEE INFOCOM*, 2006.
[28] Y. Guo, C. Liang, and Y. Liu, "AQCS: Adaptive Queue-based Chunk Scheduling for P2P Live Streaming," 2008.
[29] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *Proceedings of ACM SIGCOMM*, 2008.



Chao Liang (S'07) received his B.Engr. and M.Engr. degrees from Department of Electronic and Information Engineering, Huazhong University of Science & Technology (HUST), China, in 2000 and 2002, respectively. He is currently a Ph.D. candidate at the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, New York. His research interests include peer-to-peer networks, network optimization and content distribution, design and evaluation of algorithms and protocols.

Miao Zhao (M'10) received the B.Eng. and M.S. degrees in electrical engineering from Huazhong University of Science and Technology, Wuhan, China. She is currently a PhD candidate in the Department of Electrical and Computer Engineering, Stony Brook University, New York. Her research interests include cross-layer design and optimization in wireless networks, and performance evaluation of protocols and algorithms.

Yong Liu (M'01) has been an assistant professor at the Electrical and Computer Engineering department of the Polytechnic Institute of NYU since March, 2005. He received his Ph.D. degree from Electrical and Computer Engineering department at the University of Massachusetts, Amherst, in May 2002. He received his master and bachelor degrees in the field of automatic control from the University of Science and Technology of China, in July 1997 and 1994 respectively. His general research interests lie in modeling, design and analysis of communication networks. His current research directions include robust network routing, Peer-to-Peer IPTV systems, overlay networks and network measurement. He is the winner of the IEEE INFOCOM Best Paper Award in 2009, and the IEEE Communications Society Best Paper Award in Multimedia Communications in 2008. He is a member of IEEE and ACM.