Optimal Resource Allocation in Multi-Source Multi-Swarm P2P Video Conferencing Swarms

Chao Liang and Yong Liu
Polytechnic Institute of NYU, Brooklyn, NY, USA 11201
Email: cliang@photon.poly.edu, yongliu@poly.edu

***Abstract*—In a multi-party video conference, multiple users simultaneously distribute videos to their receivers. While pure server-based solutions are expensive, users in the conference alone may not have sufficient upload bandwidth to sustain the multiplied streaming workload in a pure P2P fashion. Recently proposed hybrid solutions employ helpers to address the bandwidth deficiency in P2P video conferencing swarms. In this paper, we focus on systems consisting of multiple video conferencing swarms. Instead of dedicating helpers to individual swarms, it is more economical to dynamically share a pool of helpers between them. Peers in a bandwidth-rich swarm can also share their bandwidth with peers in a bandwidth-poor swarm. We study the optimal bandwidth sharing in two scenarios: 1) Swarms are *independent* and only draw bandwidth from a shared helper pool; 2) Swarms are *cooperative* and share bandwidth 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 to peer dynamics, and can dynamically allocate peer and helper bandwidth across swarms to achieve the system-wide optimum.**

I. Introduction

Video conferencing applications, such as MSN Messenger [1] and Skype [2], are getting increasingly popular on the Internet. Two-party video conferencing can be implemented in a pure P2P fashion: one user sends its video directly to the other. In a multi-party video conference, multiple users simultaneously distribute their videos to multiple receivers. In a pure P2P solution, users in the same conference form a swarm and relay video to each other. However, users in the conference by themselves normally do not have enough upload bandwidth to broadcast the videos from multiple users. In a pure server-based solution, a user's video will first be uploaded to a server, and then be relayed to the receivers. The drawback is that the server bandwidth cost grows quickly with the number of users in the conference. 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 will be relayed to receivers by the helpers and other users in the same conference.

A video conferencing system normally hosts multiple parallel sessions, which are dynamic, and have different service requirements and bandwidth availability levels. Instead of dedicating helpers to individual sessions, it is more economical to share a pool of helpers between live sessions. Peers in a bandwidth-rich swarm can also share their bandwidth with peers in a bandwidth-poor swarm. Figure 1 illustrates an example of the coexistence of two swarms in a conferencing system. Cross-swarm bandwidth sharing effectively addresses the bandwidth heterogeneity and achieves the *multiplexing gain* between swarms in P2P file sharing [5] and video streaming [6]. How to optimally share bandwidth among peers and helpers in a multi-swarm multi-party video conferencing system is still an open research problem, which is the focus of this paper.

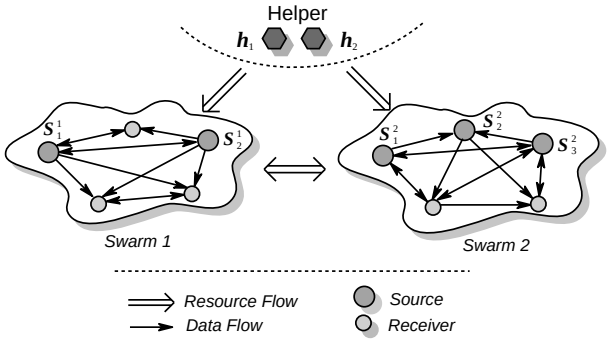


Fig. 1. Coexistence of Multiple Conferencing Swarms

We investigate optimal bandwidth sharing in two scenarios: 1) Swarms are *independent* and draw bandwidth only from a shared helper pool; 2) Swarms are *cooperative* and share bandwidth with each other. For both scenarios, we study the optimal bandwidth sharing under a utility-maximization framework. For scenario 1, bandwidth sharing operates at two levels: within each swarm, users adjust their video multicast rates to maximize the aggregate utility of the whole swarm; between swarms, helpers allocate their upload bandwidth to maximize the aggregate utility of the whole 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 quick convergence for practical implementations. For scenario 2, there is additional bandwidth sharing between bandwidth-rich swarms and bandwidth-poor swarms. We design cross-swarm bandwidth sharing rules and distributed algorithms that allow swarms to dynamically adjust the amount of resources shared with others in the face of peer churn and swarm churn. In the simulations, 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 simulations in Section VI. The paper is concluded in Section VII.

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. The authors in [7] proposed an End System Multicast architecture to support video conferencing applications. [8] proposed a full mesh conferencing protocol without a central point of control. Authors of [9] proposed to integrate ALM based P2P conferencing system with IP multicast. Utility maximization for a single conferencing swarm with helpers is studied in [4]. Most previous work focus on single P2P conferencing session. All users in the same conference form a swarm and help each other relay videos. In P2P systems, due to the heterogeneity in user bandwidth availability and the service differentiation requirements, it is necessary to enable resource sharing among swarms to maximize the global welfare. Cross-swarm bandwidth sharing has been proposed for P2P file sharing applications [5]. Server bandwidth allocation schemes among parallel file sharing swarms were studied in [10], [11]. In live video streaming, cross-channel sharing was proposed to improve the quality of small channels and provide service differentiation among channels [6], [12] . In video on-demand systems [13], a peer may serve content stored in the cache to peers in sessions different from his current viewing session. However, cross-swarm resource sharing for P2P video conferencing has not yet been explored. 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 peers not participating in any conference[1]. In conferencing swarm i, let V_i be the set of users participating in the conference. A subset $S_i \subset V_i$ of users are video sources.[2] Each source $s \in S_i$ distributes its own unique content to all other users in the conference. Videos from all sources are relayed by all users in the swarm and outside helpers. We use $N_i = V_i \cup H$ to denote the set of nodes for swarm i. The video multicast rate z_s on source s is upper-bounded by e_s, which reflects the maximum video encoding capability of s. Sources in the same swarm compete for bandwidth resources from peers and helpers to increase their multicast rates for better quality. For the multicast session of s, let $U_s(z_s)$ be the utility for a user to receive video from s at rate z_s. We assume $U_s(\cdot)$ is increasing and concave. For each conferencing swarm, the goal is to maximize the aggregate utility of all users in all multicast sessions. The global welfare of the whole system is the aggregate utility of all users in all swarms. We study the system-wide utility maximization under two scenarios: 1) Swarms are *independent*, and draw bandwidth only from the shared helper pool; 2) Swarms are *cooperative*, and can share bandwidth with each other.

B. Distribution Trees within Conferencing Swarm

Although network coding can achieve the maximum rate in single-source multicast with polynomial complexity [15], [16], how to achieve the maximum rates in multiple-source multicast with general network topology is challenging and still largely open. On the other hand, in P2P overlay network where each peer can reach all others, it is commonly assumed that the node upload links are the only network bottleneck. For uplink-throttled P2P network, it was shown in [17], [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, there are a set of receivers R_s and a set of helpers H_s. The maximum multicast rate can be achieved by packing $1 + |R_s| + |H_s|$ number of trees as in Figure 2.

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

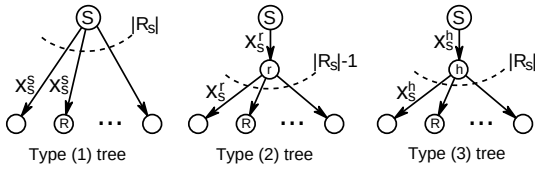


Fig. 2. Different Types of Distribution Trees

We adopt this two-hop relay scheme for multi-source multicast in video conferencing. In each swarm, peers form the above distribution trees for each source. Since the scale of a conference swarm is commonly small, it is affordable to form this fully-connected mesh. The distribution trees have at most two hops, which implies short propagation delays when video

[1]Those peers are either altruistic, or motivated to accumulate credits to obtain conferencing service in the future under asynchronous incentive frameworks, such as [14]

[2]We allow the existence of pure receivers generating no video.

TABLE I
NOTATIONS

Definition	Description
I	set of multiple-party swarms in the conferencing system
H	set of helper nodes in the conferencing system
V_i	set of participating users in the ith conferencing swarm
N_i	set of nodes in the ith swarm, $N_i = V_i \bigcup H$
S_i	set of video sources in the ith swarm, $S_i \subset V_i$
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
c_v	the upload bandwidth capacity of node v
U_s	the utility function of receiving video from source s
e_s	the maximum multicast rate from s

is forwarded along the trees. This makes the two-hop relay scheme appropriate for video conferencing applications with tight delay requirement. [3] Given the conferencing architecture, each source only needs to determine the multicast rates on its distribution trees.

C. Utility Maximization

1) Optimizing Rates of Distribution Trees: As illustrated in Figure 2, let x_s^m denote the rate of video generated by $s \in S_i$ and relayed by $m \in N_i$ (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. The notations are summarized in Table I. To encourage peers first use their own bandwidth before resorting to the helpers, a peer incurs a cost of $G_h(f)$ when drawing bandwidth f from helper h. To maximize the aggregate utility of all users, we obtain

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

The aggregate multicast rate of each source is bounded, which yields

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

And the bandwidth contributed by each helper should be limited by the its upload capacity, we have

$$\sum_{i \in I} \sum_{s \in S_i} |R_i| x_s^h \leq c_h, \quad \forall h \in H.$$

For a peer $v \in V_i$ in conference i, its aggregate upload rate cross all distribution trees for all sources in the swarm can be calculated as

$$y_v = \begin{cases} b_v + \sum_{s \in S_i \setminus \{v\}} (|R_i| - 1) x_s^v & \text{if } v \in S_i \\ \sum_{s \in S_i} (|R_i| - 1) x_s^v & \text{otherwise,} \end{cases} \quad (1)$$

where b_s denotes the required bandwidth of source s to drive the associated distribution trees. Suppose $s \in S_i$, we obtain

$$b_s = |R_i| x_s^s + \sum_{r \in V_i \setminus \{s\}} x_s^r + \sum_{h \in H_s} x_s^h. \quad (2)$$

The upload rate of a peer is limited by his upload capacity, i.e., $y_v \leq c_v$.

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

Even though U_s is strictly concave, the objective function in (1) is not strictly concave in $\mathbf{x} = \{x_s^m\}$. The optimal solution is not unique. It has $\sum_i |S_i||N_i|$ number of variables and may involve complex computations. On the other hand, the aggregate multicast rate of source s is $z_s = \sum_{m \in N_i} x_s^m$. We can observe the objective function is strictly concave in $\mathbf{z} = \{z_s\}$. If we could attain the optimal solution of z_s and recover each x_s^m at low cost, the complexity of the original problem can then be greatly reduced. To ease the problem solving, we tackle the problem in an alternative way.

2) Optimizing Source Multicast Rates: In this section, we study utility maximization based on source multicast rate vector $\mathbf{z}$. We first investigate the domain of $\mathbf{z}$ and then show how to recover $\mathbf{x}$ for any feasible $\mathbf{z}$.

Let f_i^h denote the bandwidth contributed to swarm i by helper h. We have the following results on the maximal multicast rates in swarm i.

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

$$\begin{cases} |R_i| \sum_s z_s \leq \sum_v c_v + \frac{|R_i|-1}{|R_i|} \sum_h f_i^h \\ z_s \leq \min(c_s, e_s), \forall s \in S_i. \end{cases} \quad (3)$$

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 (2) can be rewritten as

$$b_s = (|R_i| - 1) x_s^s + z_s. \quad (4)$$

Let $z_v = 0$ for pure receivers $v \in V_i \setminus S_i$. Equation (1) can be reformulated as

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

Because in type (3) trees, helpers need to broadcast content to all $|R_i|$ participating parties, we have $f_i^h = |R_i| \sum_{s \in S_i} x_s^h$. The summation of bandwidth that all participating parties spend yields

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

Hence, the maximal rate region is bounded by Equation (3).

To prove the maximal region is achievable, it is sufficient to show that for any multicast rate vector falling in the region defined by (3), we can find a set of delivery rates for all distribution trees of all sources under the bandwidth constraints on peers and helpers. We present an algorithm to

recover $\mathbf{x}$ from $\mathbf{z}$ favoring the depth-1 tree, which has short propagation delays.

- Given z_s, the source s would try to maximize the delivery rate of depth-1 tree. Based on (4) and $b_s \leq z_s$, we construct

$$x_s^s = \begin{cases} z_s & \text{if } c_s > |R_i| z_s \\ \frac{c_s - z_s}{|R_i|-1} & \text{otherwise.} \end{cases} \quad (5)$$

- A source first allocates bandwidth of $|R_i|x_s^s$ to broadcast on its depth-1 tree. It then needs to upload one stream to its depth-2 trees. The total rate is $z_s - x_s^s$. Let $\bar{c}_v$ denote the remaining bandwidth on node v to relay video for depth-2 trees of other sources. Then we have $\bar{c}_s = c_s - y_s = c_s - (|R_i|-1)x_s^s - z_s$ for sources and $\bar{c}_v = c_v$ for pure receivers.
- Let $\zeta_v = \frac{\bar{c}_v}{|R_i|-1}$ for $v \in V_i$ and $\zeta_h = \frac{c_h}{|R_i|}$ for $h \in H$. Source s calculates the delivery rate of the depth-2 tree through node m as

$$x_s^m = (z_s - x_s^s)\frac{\zeta_m}{\sum_{v\in V_i}\zeta_v + \sum_{h\in H}\zeta_h}.$$

It can be easily verified that the constructed delivery rate vector $\mathbf{x}$ recovers the multicast rate vector $\mathbf{z}$, and the bandwidth constraints on all peers and helpers are preserved. ■

In the type (3) trees, the helpers would distribute content after they receive one copy of content from sources first. Since helpers do not require these content, this inevitable overhead may occupy up to $1/|R_s|$ helper bandwidth. Here we formally define it as follows.

Definition 1: The *helper bandwidth efficiency factor* of a swarm is defined as $\rho_i = \frac{|R_i|-1}{|R_i|}$, which represents the ratio of helper bandwidth efficient to increase the system utility.

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

$$\textbf{ID-OPT} \quad \max_{\{z_s, f_i^h\}} \sum_{i\in I}\sum_{s\in S_i} |R_i|U_s(z_s) - \sum_{h\in H} G_h(\sum_{i\in I} f_i^h) \quad (6)$$

subject to

$$z_s \leq \min(c_s, e_s), \quad \forall s \in S_i, i \in I \quad (7)$$

$$\sum_{i\in I} f_i^h \leq c_h, \quad \forall h \in H \quad (8)$$

$$\sum_{s\in S_i} |R_i| z_s \leq \sum_{v\in V_i} c_v + \rho_i \sum_{h\in H} f_i^h, \quad \forall i \in I \quad (9)$$

$$z_s, f_i^h \geq 0, \quad \forall s \in S_i, i \in I, h \in H. \quad (10)$$

IV. Distributed Algorithms for Independent Conferencing System

In this section, we develop distributed algorithms for optimal bandwidth sharing in independent conferencing systems.

A. Proximal Approximation Based Algorithm

1) Main Steps of Proximal Algorithm: The problem ID-OPT is still not strictly concave in f_i^h. Directly solving it in a distributed manner may incur oscillation, which is not amenable for implementation before the system enters the equilibrium. We resort to the standard proximal optimization algorithm [18]. 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 added to the objective function to make it strictly concave, 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 $f_i^h(t)$.

$$\max \sum_{i\in I}\sum_{s\in S_i} |R_i|U_s(z_s) - \sum_{h\in H} G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}||\mathbf{f}-\mathbf{d}||_2^2 \quad (11)$$

subject to the constraints (7)(8)(9)(10).

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

In the proximal algorithm, after we get the optimal $\mathbf{f}^*$ in step 1, then we turn to next step to assign $\mathbf{d}(t+1) = \mathbf{f}^*(t+1)$ and begin the next iteration. Actually 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 make it more amenable for online implementation, some extended approach [19] can allow the fixed number of inner iteration in step 1 and still guarantee the convergence of entire problem.

2) Dual Decomposition: To solve the problem (11) of step 1 in a distributed fashion, we can apply the dual decomposition techniques. We relax the constraint (9) with Lagrangian multipliers λ_i. Then we can get the Lagrangian function

$$\begin{aligned} L(z_s, f_i^h, \lambda) = & \sum_{i\in I}\sum_{s\in S_i} |R_i|U_s(z_s) - \sum_{i\in I}\lambda_i \sum_{s\in S_i} |R_i| z_s \\ & - \sum_{h\in H} G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}\sum_{h\in H}\sum_{i\in I}(f_i^h - d_i^h)^2 \\ & + \sum_{i\in I}\lambda_i\rho_i \sum_{h\in H} f_i^h + \sum_{i\in I}\lambda_i \sum_{v\in V_i} c_v. \end{aligned} \quad (12)$$

By duality, we obtain the following equivalent dual problem

$$\min g_{\lambda\geq 0}(\lambda) = \min \max L_{\lambda\geq 0}(z_s, f_i^h, \lambda). \quad (13)$$

We can observe that the problem has nice separable property for decomposition. Hence, we can solve the following subproblem in a distributed manner.

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

$$\max_{0\leq z_s \leq min(c_s, e_s)} |R_i|U_s(z_s) - \lambda_i |R_i| z_s. \quad (14)$$

Then s should adjust its rate as follows:

$$z_s(t) = \begin{cases} 0 & \text{if } U_s'(0) < \lambda_i \\ (U_s')^{-1}(\lambda_i) & \text{else if } U_s'(\min(c_s, e_s)) \leq \lambda_i \\ \min(c_s, e_s) & \text{otherwise.} \end{cases} \quad (15)$$

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

$$\max_{0\leq \sum_{i\in I} f_i^h \leq c_h} -G_h(\sum_{i\in I} f_i^h) - \frac{c}{2}\sum_{i\in I}(f_i^h - d_i^h)^2 + \sum_{i\in I}\lambda_i\rho_i f_i^h. \quad (16)$$

The above sub-problem is strictly concave in f_i^h and can be locally solved by the helper h. Under the Karush-Kuhn-Tucker conditions [20], it can be solved by the approach in [19] with the complexity of $O(|I|log(|I|))$. The first term can be removed, i.e., let $G_h = 0$, if the aggregate bandwidth of helpers is not large enough to let all sources in the system reach their maximum multicast rates.

After source and helper nodes adjust their rates, the Lagrangian multipliers will be updated with gradient projection algorithm [18]

$$\lambda_i(t+1) = [\lambda_i(t) + \theta_t(\sum_{s\in S_i}|R_i|z_s - \sum_{v\in V_i} c_v - \rho_i \sum_{h\in H} f_i^h)]^+, \tag{17}$$

where θ_t is a positive stepsize to guarantee the convergence and $[]^+$ 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.

In practical implementation, one node in each swarm can be promoted as the *coordinator* to communicate with the helpers. The coordinator maintains the Lagrangian multiplier associated with its own swarm. It broadcasts the latest multiplier to the sources within the same swarm and all the 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 helper allocated resources, the sources try to grab resources to reach their targeted multicast rates. They will notify the coordinators with the resource gap information. Then the coordinators can update the multiplier information according to (17) after they collect information from all sources. The updated multipliers are broadcasted to all sources and helpers to trigger the next iteration. The system finally enters the equilibrium after multiple iterations.

B. Marginal Utility Driven Algorithm

In this section, we present an alternative approach which relies on the optimality condition and primal decomposition. It incurs less computation overhead and iterations, and consists of two levels of optimization. Within each swarm, sources adapt multicast rates asynchronously to reach the optimum. At the outer level, helpers adjust their resource allocation to maximize the system-wide performance.

1) Inner-swarm Adaptation: Given certain bandwidth from helpers f_i^h, each swarm needs to adjust the resource allocation to achieve the optimum. We intend to maximize the aggregate utilities $\mathbf{U}^i$ of swarm i with resource allocation,

$$\max_{\{0\le z_s\le \min(c_s,e_s)\}} \sum_{s\in S_i} |R_i| U_s(z_s) \tag{18}$$

subject to $\sum_{s\in S_i}|R_i|z_s \le \sum_{v\in V_i} c_v + \rho_i \sum_{h\in H} f_i^h$. By adding a virtual source node connecting to all sources of swarm i, the problem becomes a pure routing problem (similar to that in (pp.452 [21])). We have the following sufficient and necessary optimality conditions.

Theorem 2: Let z^* be the optimal streaming vector of the above problem. If it is feasible to shift some resources from a source s to another source s', we will obtain $\frac{\partial \mathbf{U}^i(z^*)}{\partial z_{s'}} \le \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s}$. In the optimal solution, the rates of any pair of sources p and q satisfy

- $\frac{dU_p(z_p^*)}{dz_p} \le \frac{dU_q(z_q^*)}{dz_q}$, 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 $z_q^* \in (0, \min(e_q, c_q))$
- $\frac{dU_p(z_p^*)}{dz_p} \ge \frac{dU_q(z_q^*)}{dz_q}$, if $z_p^* = \min(e_p, c_p)$ and $z_q^* \in (0, \min(e_q, c_q))$

Proof: Please refer to the Appendix A. ■

Using the above properties, we propose the following pair-wise balance algorithm to optimize the resource allocation within a swarm. The sources notify each other their current marginal utility periodically. Then each source finds another source to balance their multicast rates. During the pair-wise balance between source p and q, suppose $U_p'(z_p) > U_q'(z_q)$, the source q with smaller marginal utility shifts its owned resource to p until the difference between their marginal utilities is less than a small scalar ϵ or p reaches the maximum multicast rate.

PWBalance(z_p) algorithm describes the distributed implementation of pair-wise balance process on source p:

(I) **Forming a Pair** Source p establishes the balance process with another source. It could be triggered either by accepting requests from other sources or initiating requests proactively.

1) *Accepting requests*. Source p receives multiple requests from other sources.
   a) It accepts the request from the one with the maximum marginal utility if it is idle.
   b) It rejects all requests once it is already in a balance process with another source.
2) *Initiating requests*. Source p sends requests to other sources. Source p maintains a candidate set $S = \{q | U_p'(z_p) > U_q'(z_q) + \epsilon, z_q > 0\}$ where ϵ is a scalar parameter.
   a) It sends request to the one $q \in S$ with the minimum marginal utility. If q rejects the request, source p removes q from the candidate set $S = S \setminus \{q\}$. The operation repeats until one source accepts the request or S becomes empty.
   b) If one source q updates the marginal utility information, it can be inserted into the candidate set of source p if it satisfies the constraint of the set.

(II) **Negotiating Marginal Utility** After forming a pair, the source with larger marginal utility prepares a list according to the marginal utility information. Suppose $U_p'(z_p) > U_q'(z_q)$ and M is a constant.

1) Source p issues a list of M elements $(\delta_j, U_p'(z_p) - \frac{\Delta}{M} j)$ to source q, where $\Delta = U_p'(z_p) - U_q'(z_q)$ and $\delta_j = U_p'^{-1}\left(U_p'(z_p) - \frac{\Delta}{M} j\right) - z_p$.
   a) The first item is the possible increase amount of multicast rate.
   b) The second item corresponds to the marginal utility for the new session multicast rate of source p.

2) Let $m = \arg\max_j \delta_j + z_p \geq \min(c_p, e_p)$. If the number of feasible entries is not enough, i.e., $m < M$, we add an additional entry $(\min(c_p, e_p) - z_p, U'_p(\min(c_p, e_p)))$.

(III) **Adjusting Rates** After source q receives the list from p, it determines the amount of resource to shift between them.

1) Source q picks the appropriate and feasible one to make them first derivative nearest, $j = \arg\max_j U'_p(z_p+\delta_j) > U'_q(z_q - \delta_j)$.
2) Source q shifts the amount of resource equivalent to δ_j session multicast rate to source p.

(IV) **Updating** Once the balance process completes, the source p quits the pair-wise balance process if it reaches the maximum multicast rate $z_p = min(e_p, c_p)$ and $U'_p(z_p) \geq U'_s(z_s), \forall s \in \{s|z_s < \min(e_s, c_s)\}$. Otherwise, it broadcasts its new marginal utility to all other sources in the swarm.

In the above pair-wise balance process, to reduce the difference of the marginal utilities of two sources less than certain threshold ϵ, i.e., $(\frac{1}{M})^n \Delta \leq \epsilon$, we only need $O(\log_M(\frac{\Delta}{\epsilon}))$ steps. Given a reasonably large M, q can certainly find a feasible entry. In case the two functions have very large difference that $U'_p(z_p + \delta_1) < U'_q(z_q - \delta_1)$, source p can reissue a new list with new $\Delta' = \Delta/M$ to narrow the gap. Furthermore, the sources can adapt asynchronously to achieve inner-swarm optimization in this pair-wise balance process.

2) Helper Scheduling: After the swarms enter equilibrium, helpers adapt the amount of bandwidth allocated to swarms at an outer level. First we need to find out the marginal utility of each swarm to helpers.

A swarm i is called *choked* if all sources in the swarm reach the maximum possible multicast rates, i.e., $z_s = min(e_s, c_s)$ for all sources $s \in S_i$. For an unchoked swarm, it has $\sum_{s\in S_i} |R_i| z_s = \sum_{v\in V_i} c_v + \rho_i \sum_{h\in H} f_i^h$. Suppose we increase the helper bandwidth contribution by Δf, then the increase space for source rates is $\sum_{s\in S_i} |R_i| \Delta z_s = \rho_i \Delta f$. To maximize the increase in the swarm's aggregate utility, one should increase the rates of sources with the maximum marginal utility. As indicated in the equilibrium property, the sources with positive multicast rates have the same marginal utility after we rule out the sources whose rates are already saturated. Hence, the optimal system utility gain is obtained

$$\begin{aligned} \Delta \mathbf{U}^i_{opt} &= \sum_{s\in S_i} |R_i|(U_s(z_s + \Delta z_s) - U_s(z_s)) \\ &\approx U'_{max} \sum_{s\in S_i} |R_i| \Delta z_s \\ &= \rho_i U'_{max} \Delta f. \end{aligned} \tag{19}$$

We have the following definition accordingly.

Definition 2: The *marginal utility of swarm i to helper's bandwidth* is defined as $\mu_i = \rho_i U'_p(z_p)$, where $p = \arg\max_{s\in S_i} U'_s(z_s)$ and $z_s < \min(e_s, c_s)$.

Suppose swarm i is choked with resource f_i^c from helpers. The choked swarm could not gain any utility because they cannot receive any bandwidth further. To facilitate the description, we define the marginal utility of an choked swarm j as $\mu_j = \rho_i \min U'_s(z_s), s \in S_j$. Actually we have $\mu_j(f_i^c) = \lim_{\delta\to 0} \mu_j(f_i^c - \delta)$, since the source with the minimum marginal utility is also the last one to reach its maximum multicast rate if we keep increasing f_i^c until the swarm is choked.

When one swarm enters inner-swarm equilibrium, the coordinator of the swarm broadcasts the marginal utility of the swarm to all helpers. When helpers receive the marginal utility of all swarms, they can adapt the bandwidth allocations among all swarms accordingly. Given the new helper bandwidth allocation, the sources in each swarm then continue to conduct the pair-wise balance process to maximize the utilities until the swarm enters a new equilibrium. The system finally enters the global equilibrium after multiple above iterations.

In the following algorithm, helpers adapt the distribution among swarms dynamically. Since $\mathbf{U}^i_{opt}$ is also concave function on f_h^i, intuitively we need to put more resource to swarm with larger marginal utility. Suppose ϕ_i^h denotes the fraction of helper h bandwidth which is used by the swarm i. **SCHHelper(ϕ_i^h)** would update the allocations towards the equilibrium, as in [23].

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

$$\phi_i^h(t+1) = \phi_i^h(t) + \gamma_i^h(t) \tag{20}$$

with

$$\gamma_i^h(t) = \begin{cases} -\min\{\phi_i^h(t), \kappa_n(\mu_w - \mu_i)\} & \text{if } i \neq w \\ -\sum_{i\neq w} \gamma_i^h & \text{if } i = w \\ 0 & \text{if } \mu_i \geq \mu_w \text{ and choked,} \end{cases} \tag{21}$$

where κ_n is a positive scalar stepsize.

For swarms, with $\mu_i \geq \mu_w$ but choked, which are unable to accept further resources, the bandwidth fractions of these swarms remain unchanged. The helpers shift resource from swarms with smaller marginal utility to the one with the largest μ. If the amount of the shifted resources is more than that makes the swarm choked, the helpers would spend the surplus bandwidth to the swarm with the largest marginal utility in the next iteration.

V. Distributed Algorithm for Sharing between Cooperative Swarms

In this section we investigate the resource sharing strategies between cooperative swarms.

Let π_j^i denote the bandwidth shifted from swarm i to swarm j. The nodes in swarm i who share bandwidth with nodes in swarm j can be regarded as extra helper nodes. Hence, the sources in swarm j can build additional depth-2 tree relayed by these nodes. Given the new resource distribution, the possible rate region of sources can be formulated as follows,

$$\sum_{s\in S_i} |R_i| z_s \leq \sum_{v\in V_i} c_v + \tau_i + \rho_i \sum_{h\in H} f_i^h, \quad \forall i \in I, \tag{22}$$

where $\tau_i = -\sum_{j\in I, j\neq i} \pi_j^i + \rho_i \sum_{j\in I, j\neq i} \pi_i^j$ denotes the swarm i's net resource gain in cross-swarm sharing. If the problem is directly solved in optimization based approach as in

Section IV-A, additional variables for the resource allocation among all swarms of each node in the system need to be introduced. It will inevitably incur heavy computation and slow convergence. Furthermore, the resources of a swarm may be shared by too many other swarms, resulting in excessively large number of connections, which might be difficult for practical implementation. To ease the problem solving, we can leverage the approaches of independent conferencing swarms.

A. Criteria of Cooperation among Swarms

After the system enters the equilibrium under the adaption of helpers, swarms can also conduct the similar resource allocation process as those of helpers. For swarm i, the amount of resources available for sharing are from all participating parties, i.e., $B_i = \sum_{v \in V_i} c_v$.

1) Comparison of marginal utilities: First we identify the marginal utilities of swarms receiving the resources from swarm i. If swarm i shares its own resources with swarm j, swarm i can be regarded as another helper to swarm j. To build type (3) tree is subject to the same overhead. Hence, the marginal utility of swarm j to swarm i would still be μ_j. When swarm i takes its resource back, the marginal utility would be μ_i/ρ_i since there is no overhead to increase the rates of its own type (1)(2) trees. Therefore, if it could bring utility gain by shifting the resources of swarm i to swarm j, then it needs to satisfy

$$\mu_i/\rho_i \leq \mu_j. \tag{23}$$

2) Avoidance of resource relay: As churn occurs, a swarm getting help from other swarms may have extra bandwidth to share with other swarms. On the other hand, a swarm helping others may run into resource deficit and need to get help from others. Both cases lead to resource relays between swarms. However, the resource relay is not efficient. As illustrated in

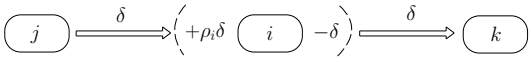


Fig. 3. Illustration of overhead due to resource relay

Figure 3, swarm j shifts δ amount of resource to swarm i which is not choked, and swarm i moves the same amount of its own resource to swarm k at the same time. Swarm i would suffer $(1-\rho_i)\delta$ resource loss, compared with the case that swarm j directly shifts resource to swarm k. The resource relay should be prevented to avoid resource loss.

3) Minimizing the degree of cross-swarm sharing: Each time swarms conduct an adaptation, they will wait the system enter new equilibrium before the next adaptation. Helpers will re-allocate the bandwidth to reach this new equilibrium. Without resource relay, there are four possible swarm statuses after the system enters new equilibrium with helper scheduling. There exists a constant μ_m, the relationship between the marginal utility of swarms can be listed as follows. We suppose the aggregate amount of helper resources is not large enough to make all swarms choked.

TABLE II
SWARM STATUS

Type	μ	Choked	Resource Inflow
(a)	$\mu = \mu_m$	N	Y
(b)	$\mu \geq \mu_m$	Y	Y
(c)	$\mu \leq \mu_m$	N	N
(d)	Any	Y	N

If swarm i of type (c) determines to share resource, it will choose a swarm in status of type (a), which has the maximum marginal utility while not choked. With the balance of helpers, multiple swarms may belong to this type. Instead of picking one of them randomly each time during adaptation, the swarm i can always stick to a specific one until it no longer belongs to this type. In this way, the number of swarms those share the resources of swarm i is to be minimized.

Furthermore, we can observe that the swarm of type (d) without incoming resources may have surplus bandwidth not fully utilized, which should be spent at the beginning. The amount of surplus bandwidth of swarm j in this type equals to $\sum_{v \in V_j} c_v - |R_j| \sum_s min(e_s, c_s)$.

B. Cooperation Strategies of Swarms

With the above insights, we would conduct an additional adjustment at larger timescale. Each time it needs to be applied after the system enters the new equilibrium. Let ψ_j^i denote the fraction of resources from swarm i to swarm j. **SCSWarm(ψ_j^i)** adapts the fraction of resources of swarm i.

a) If swarm i identifies itself of type (d), it spends the surplus bandwidth to other swarms of type (a) at the system beginning. In later adaptation process, φ_j^i would denote the fraction of remaining resources.

b) At stage t, if swarm i has accepted resource from helpers or other swarms, it would not conduct the below adaptation process.

c) For swarm i to adapt the resource allocation, first we identify the swarm w to receive resource. If μ_i/ρ_i is larger than the marginal utility of type (a) swarm μ_m, then $w = i$. Otherwise, swarm i chooses one from the set $\{j | \psi_j^i > 0, \mu_j = \mu_m$ and not choked$\}$. In case this set is empty, swarm i picks one from the remaining type (a) swarms.

Then we update the bandwidth allocations. Let

$$\bar{\mu}_s = \begin{cases} \mu_i/\rho_i & \text{if } s = i \\ \mu_s & \text{otherwise.} \end{cases}$$

And we have

$$\psi_j^i(t+1) = \psi_j^i(t) + \vartheta_j^i(t) \tag{24}$$

with

$$\vartheta_j^i(t) = \begin{cases} -\min\{\psi_j^i(t), \kappa_n(\bar{\mu}_w - \bar{\mu}_j)\} & \text{if } j \neq w \\ -\sum_{j \neq w} \vartheta_j^i & \text{if } j = w \\ 0 & \text{if } \bar{\mu}_j \geq \bar{\mu}_w \text{ and choked,} \end{cases} \tag{25}$$

where κ_n is a positive scalar stepsize.

We can deduce that if swarm i determines to share resource to another swarm k at certain iteration with $\mu_k > \mu_i$, it never

takes place that another swarm j sends resource offer to swarm i at the same time. After each swarm determines the new resource allocation, swarm i may receive resource offers from others.

- If $\sum_j \psi_j^i = 0$, swarm i only receives resource from others. Swarm i will preferentially receive the resources from the ones with smaller marginal utility.
- If $\sum_j \psi_j^i > 0$, swarm i shares its own resource with others. To avoid resource relay, instead of accepting the offers, it reclaims its own resource back. The amount of resource claimed back would be the minimum of its resource shared with others $B_i \sum_j \psi_j^i$ and that of the resource offer.

Similarly, if a swarm with resource shared by others gets offer from helpers during the inner-level helper resource adaptation, the swarm reclaims its own resource back instead of accepting the helper resource. The amount of resource to be claimed is determined as the minimum of the resource it shares with others and that of the resource offer from helpers.

Finally, we briefly discuss how a swarm share its own resources of its member nodes with other swarms. The resources to be shared can firstly come from those non-source participating users, i.e., $V_i \setminus S_i$. When only the resources of sources remain, some optimization based allocation needs to be conducted. Please refer to the Appendix B for details.

VI. Simulation Results

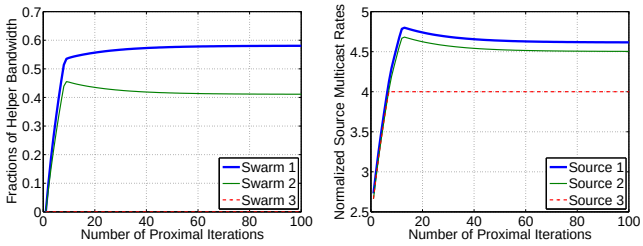
In this section, we provide numerical results of the proposed algorithms. To present clearly how the system evolves, we conduct the simulation with a small scale. Table III lists the normalized bandwidth of all swarms. There are two helpers with normalized bandwidth 40 and 60. To ease the analysis, the maximum multicast rate for all sources is set to be 5. In this case, the sources of swarm 3 can all reach the maximum multicast rate without any helper resource. 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.

TABLE III
Node Bandwidth Setting

	Sources	Other 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 the swarm 3 has the maximum amount of resources. Figure 4(a) shows the fraction of helper 1 bandwidth at different proximal steps. The helper resources have been allocated to the first two swarms accordingly. Swarm 3 does not need any helper resource. Figure 4(b) shows



(a) Fractions of Helper Bandwidth (b) Source Rate Evolution

Fig. 4. System Evolution with Proximal Approximation Based Algorithm

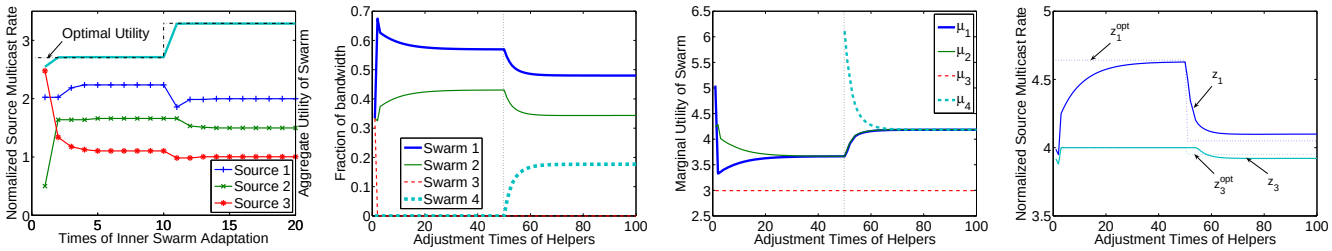
the evolution of the multicast rates of all sources in swarm 2. We let the utility weight of source 1 slightly larger than other sources. The multicast rate of source 1 is slightly larger in return. The rate of source 3 increases first with other sources until it reaches the upload capacity limit. We can observe that the resource allocation of helpers can be coordinated and the multicast rates of sources can achieve the optimum.

2) Marginal Utility Driven Algorithm: First we study the performance of pair-wise balance algorithm. We let the threshold of marginal utility gap be $\epsilon = 0.1$ and $M = 10$. Figure 5(a) illustrates how the multicast rates of sources in a single swarm can converge to the optimum by the pair-wise balance procedure. The three sources have utility weight in decreasing order. After few times of pair-wise balance procedure, the aggregate system utility gets close to the optimum. When a new party joins the ongoing conference suddenly, the sources adapt the rates and quickly converge to the optimum. This shows the robustness of the procedure against peer churn.

Next we show the system evolution under swarm churn. The helpers adjust the bandwidth distribution dynamically. At the beginning, there are only the first 3 conference sessions. The 4th swarm joins the system later. Figure 5(b) shows the evolution of helper bandwidth allocation. We can observe that the system is able to enter the equilibrium in less than 20 steps. Still swarm 3 has no helper bandwidth input. Figure 5(c) shows the marginal utility of swarms. The marginal utility of swarm μ_1 and μ_2 converge to the same value. After swarm 4 joins the system, part of the bandwidth fraction allocated to the first two swarms is gradually shifted to the newly joined one. The marginal utilities of swarms change accordingly, and they converge to the new level. Figure 5(d) presents the multicast rate evolution of two sources belonging to swarm 2. As the amount of incoming resources from helpers changes, the multicast rates of these two sources adapts accordingly. The small gap between the achieved rates and the optimal ones comes from the allowed marginal utility difference ϵ in pair-wise balance algorithm. Setting smaller ϵ threshold could reduce the gap further, while possibly increases the inner-swarm adjustment times in pair-wise balance algorithm.

B. Cooperative Conferencing System

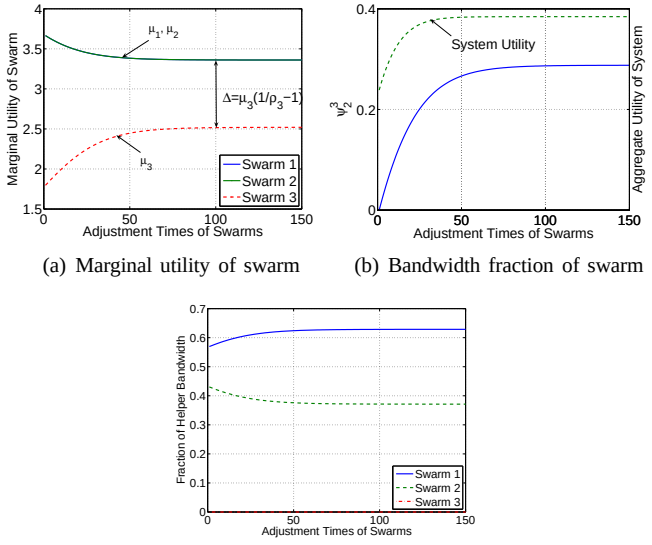
First we study the performance of the algorithm in static scenario. Only the first three swarms join the system. After the system enters equilibrium, the marginal utility of swarm



(a) Pair-wise balance (b) Bandwidth fraction of helpers (c) Marginal utility of swarm (d) Source multicast rate

Fig. 5. Simulation results of marginal utility based algorithm. (a) illustrates the evolution of source multicast rate with the pair-wise balance algorithm in a single swarm. (b)(c)(d) present the system evolution under swarm churn.

3 is far lower than those of other two swarms. Swarm 3 does not receive any resource from helpers and spares its own bandwidth to others gradually. It gradually shifts resources to swarm 2 until $\mu_3/\rho_3 = \mu_2$, as shown in Figure 6(a). We can observe from Figure 6(b) that as the fraction of bandwidth from swarm 3 spent to swarm 2(ψ_2^3) increases, the system utility increases. Actually swarm 3 only shares its resource with swarm 2 and keeps $\psi_1^3 = 0$. Each time when swarm 3 shifts some resource to swarm 2, the helpers would move certain amount of resource previously spent in swarm 2 to swarm 1 accordingly to reach the new equilibrium, as shown in Figure 6(c).

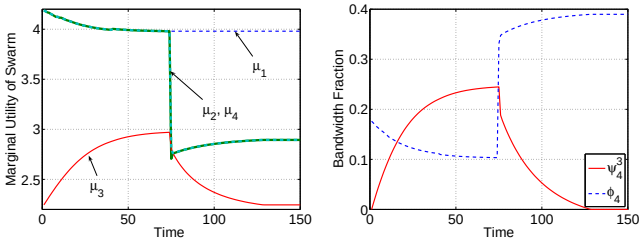


(a) Marginal utility of swarm (b) Bandwidth fraction of swarm

(c) Bandwidth fraction of helpers

Fig. 6. Simulation results in cooperative conferencing systems. (a)(b)(c) present the system performance gain due to swarm cooperation.

Next we investigate how the system adapts under swarm churn. First the four swarms join the system at the beginning, and then swarm 1 leaves the system in the middle. Figure 7 shows the evolution of marginal utility and bandwidth fraction. The system first enters the equilibrium and swarm 3 shares resource 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 is larger than others at that time. However, instead



(a) Marginal utility of swarm (b) Bandwidth fractions

Fig. 7. Simulation results in cooperative conferencing systems. (a)(b) show how the system evolves with swarm cooperation strategies when swarm churn happens.

of receiving resources from helpers, swarm 3 takes its resource back from swarm 4 gradually until the system enters a new equilibrium.

VII. CONCLUSION

In summary, our paper investigates the bandwidth sharing strategies in multi-swarm multi-party P2P conferencing systems. Specifically, we study two possible scenarios: swarms are independent or cooperative. 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.

How to guarantee low latency is crucial in multi-party conferencing applications. Although our paper focuses on optimization system utility from the aspect of bandwidth sharing, it is amenable for further work on reducing latency under this framework. In practice, swarms can only be associated with a subset of the helpers given large system scale. How to group swarms and assign helpers has impact on the system latency performance. It can also be optimized combing with ISP-friendly considerations. In the next step, we are also interested in prototyping the system with the distributed algorithms, and examining the performance in real Internet environment.

APPENDIX

A. Proof of Theorem 2

The inner-swarm utility maximization problem can be converted to a routing problem by adding a virtual source connecting all sources. As illustrated in Fig. 8, the virtual source S' has fixed traffic input rate and intends to optimally allocate the transmission rates along the paths. Compared with the optimal routing problem in pp.452 [21], the inner-swarm utility maximization problem has the following differences:

1) It is a utility maximization problem and $\mathbf{U}^i$ is concave function.
2) The multicast rate z_s of each source s in the swarm is additionally subject to the maximum multicast rate e_s, besides the capacity limit c_s.
3) The upper bounds of z_s can be attained, i.e., it is possible to let $z_s = \min(e_s, c_s)$. In the contrast, the flow rate of a path p must be less than the capacities of links contained by the path in the optimal routing problem, due to the $M/M/1$ model based delay cost approximation.

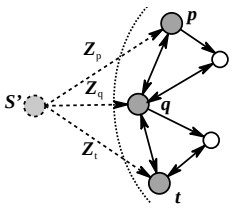


Fig. 8. Conversion to a routing problem

Despite these differences, we can still extend the optimality conditions for this inner-swarm utility maximization problem. Let $z^* = \{z_s^*\}$ be an optimal multicast rate vector for swarm i, where $s \in S_i$. Let $F_i = \{s | z_s = \min(e_s, c_s), s \in S_i\}$ denote the set of sources whose multicast rates are saturated and $F_i \subset S_i$. It is infeasible for sources $s \in F_i$ to increase their multicast rates.

First we provide the necessary condition for optimality of z^*. If $z_s^* > 0$ for some source s, we must be unable to shift a small amount $\delta > 0$ from source s to another source s', if the shift is feasible, with improving the system utility; otherwise, the optimality would be violated. The utility change from this shift is

$$\delta \frac{\partial \mathbf{U}^i(z^*)}{\partial z_{s'}} - \delta \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s}$$

and the change must be negative. Furthermore, the shift is not feasible when $s' \in F_i$. Therefore, we obtain

$$z_s^* > 0 \quad \Rightarrow \frac{\partial \mathbf{U}^i(z^*)}{\partial z_{s'}} \leq \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s}, \quad \forall s' \in S_i \setminus F_i. \quad (26)$$

For two sources $p \in F_i$ and $q \notin F_i$, the above condition indicates $\frac{\partial \mathbf{U}^i(z^*)}{\partial z_q} \leq \frac{\partial \mathbf{U}^i(z^*)}{\partial z_p}$ since $z_p^* > 0$.

In terms of Equation (18), we have $\frac{\partial \mathbf{U}^i(z^*)}{\partial z_p} = |R_i| \frac{dU_p(z_p^*)}{dz_p}$. Hence, when the swarm enters equilibrium with optimal multicast rate vector z^*, the marginal utility relationship between any two sources can be elaborated equivalently as follows.

- $\frac{dU_p(z_p^*)}{dz_p} \leq \frac{dU_q(z_q^*)}{dz_q}$, 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 $z_q^* \in (0, \min(e_q, c_q))$

Next we prove the above properties can also be shown to be sufficient for optimality. Since the utility functions of all sources are concave, the aggregate utility function $\mathbf{U}^i$ is also concave in that it is the summation of concave functions. Due to its concavity, to prove multicast rate vector z^* is optimal, we need to show that the above properties can lead to

$$\sum_{s \in S_i} (z_s - z_s^*) \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s} \leq 0,$$

for any feasible multicast rate vector z. Let $D_i^* = \max_{\{s \in S_i \setminus F_i\}} \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s}$ (Actually it is the marginal utility of swarm in Definition 2). Provided another feasible multicast rate vector z, we can divide the sources into three types.

1) For source $s \in F_i$ in optimal rate vector z^*, we have $z_s < z_s^*$ since z_s^* is already saturated. And the properties indicate that the marginal utility $\frac{\partial \mathbf{U}^i(z^*)}{\partial z_s} \geq D_i^*$.
2) For sources $s \in \{S_i \setminus F_i | z_s < z_s^*\}$, $z_s < z_s^*$ implies $z_s^* > 0$. In terms of the Equation (26), the marginal utilities of them are also no smaller than D_i^*.
3) For the last set of sources $s \in \{S_i \setminus F_i | z_s > z_s^*\}$, the marginal utilities of them are no larger than D_i^* since $z_s > z_s^*$ implies $z_s^* \leq 0$.

For each above type, we then have

$$(z_s - z_s^*) \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s} \leq (z_s - z_s^*) D_i^*.$$

Therefore, We obtain

$$\begin{aligned} 0 &= \sum_{s \in S_i} (z_s - z_s^*) D_i^* \\ &= \sum_{\{s \in F_i\}} (z_s - z_s^*) D_i^* + \sum_{\{s \in S_i \setminus F_i | z_s < z_s^*\}} (z_s - z_s^*) D_i^* \\ &\quad + \sum_{\{s \in S_i \setminus F_i | z_s > z_s^*\}} (z_s - z_s^*) D_i^* \\ &\geq \sum_{s \in S_i} (z_s - z_s^*) \frac{\partial \mathbf{U}^i(z^*)}{\partial z_s}. \end{aligned}$$

B. Optimal Sharing of The Resources of Member Nodes

Suppose a swarm decides to share its resources with other swarms through outer-level iterations, how it shares its own resources of its member nodes with other swarms needs to be taken into account carefully. The resources to be shared can firstly come from those non-source participating users, i.e., $V_i \setminus S_i$. The resources of non-source participating users which facilitate the distribution of type (2) tree, can be regarded as the public resource to the sources. The pair-wise balance can enable the system reach the optimum allowed by the achievable rate region.

However, we need to resort to optimization based technique when only the resources of sources remain in the system. Suppose source s shares m_s amount of bandwidth with other swarms. This directly changes the achievable rate region of z_s to $z_s \leq \min(e_s, c_s - m_s)$. How to share the resources of the sources has impact on the maximum achievable utility of the swarm.

We formulate this problem as follows. Give swarm i within which only the resources of sources remain, we want to decide the m_s of each source and $\sum_s m_s = \Delta$. The target aggregate amount of resources to be shared is Δ. We then obtain

$$\textbf{CO-OPT} \quad \max_{\{m_s \geq 0, z_s \geq 0\}} \sum_{s \in S_i} |R_i| U_s(z_s) \tag{27}$$

subject to

$$\sum_{s \in S_i} |R_i| z_s \leq \sum_{s \in S_i} c_s - \Delta \tag{28}$$

$$z_s \leq c_s - m_s, \quad \forall s \in S_i \tag{29}$$

$$z_s \leq e_s, \quad \forall s \in S_i. \tag{30}$$

Since the problem does not involve any network entity (helpers, nodes in other swarm) outside the swarm, once the coordinator knows all the inner-swarm information, if possible, the problem can be solved with KKT conditions. Actually the problem solving of CO-OPT can also leverage the techniques in Section IV-A.

Next we present a greedy approximation algorithm, which can be readily implemented in a distributed manner. We define a step size δ for each iteration. Suppose at the beginning, the system has already entered the equilibrium under the pair-wise balance process. In the Algorithm 1, each time the source with the minimum marginal utility intends to share its owned resource equivalent to δ amount of multicast rate. The rates of depth-2 trees are reduced preferentially. The process iterates until the target of sharing Δ amount of resources is fulfilled.

Algorithm 1: Resource Sharing of Source Nodes

input : s($z, x, \frac{dU_s}{dz_s}$), Δ
output: m_s
while $\Delta > 0$ **do**
  $p \leftarrow$ `ChooseMinMarginalUtilSource(`z_s,U_s`)`
  $t \leftarrow \delta$
  while $t > 0$ && $z_p > 0$ **do**
    if $\exists x_p^q > 0$ **then**
      $r_a = \min(t, x_p^q)$
      $t = t - r_a$
      //reduce the rate of depth-2 tree by r_a
      $x_p^q = x_p^q - r_a$ and $z_p = z_p - r_a$
      //share the resources for this rate decrease
      $m_p = m_p + r_a$ and $m_q = m_q + (|R_i| - 1) r_a$
    else
      if $x_p^p > 0$ **then**
        $r_a = \min(t, x_p^p)$
        $t = t - r_a$
        //reduce the rate of depth-1 tree by r_a
        $x_p^p = x_p^p - r_a$ and $z_p = z_p - r_a$
        //share the resources for this rate decrease
        $m_p = m_p + |R_i| r_a$
      end
    end
  end
  $\Delta = \Delta - |R_i| t$ *//update with the achieved amount*
end

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] R. S. Peterson and E. G. Sirer, "Antfarm: Efficient Content Distribution with Managed Swarms," in *Symposium on Networked System Design and Implementation*, 2009.

[11] M. Li, J. Yu, and J. Wu, "Free-Riding on BitTorrent-like Peer-to-Peer File Sharing Systems: Modeling Analysis and Improvement," in *IEEE Transactions on Parallel and Distributed Systems*, 2008.

[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] 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.

[14] Z. Liu, H. Hu, Y. Liu, M. Mobius, K. Ross, and Y. Wang, "P2P Trading in Online Social Networks: The Value of Staying Connected," *Technical Report, Polytechnic Institute of NYU*, January 2009, http://eeweb.poly.edu/faculty/yongliu/docs/sig09_tech.pdf.

[15] 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.

[16] 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.

[17] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proceedings of IEEE INFOCOM*, 2007.

[18] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[19] X. Lin and N. B. Shroff, "Utility Maximization for Communication Networks with Multi-path Routing," in *IEEE Transactions on Automatic Control*, 2006.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[21] D. P. Bertsekas and R. G. Gallager, *Data Networks*, 2nd ed. Prentice Hall, 1992.

[22] C. Liang and Y. Liu, "Optimal Bandwidth Sharing in Multi-Swarm Multi-Party P2P Video Conferencing Systems," in *Technical Report, Polytechnic Institute of NYU*, 2009.

[23] R. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," in *IEEE Transactions on Communication*, 1977.