2009

A Flexible Divide-And-Conquer Protocol for Multi-View Peer-to-Peer Live Streaming

Miao Wang
University of Nebraska-Lincoln, mwangcse@gmail.com

Lisong Xu
University of Nebraska-Lincoln, xu@cse.unl.edu

Byrav Ramamurthy
University of Nebraska-Lincoln, bramamurthy2@unl.edu

P2P '09. IEEE Ninth International Conference on Peer-to-Peer Computing, 2009.
Digital Object Identifier: 10.1109/P2P.2009.5284541 Publication Year: 2009 , Page(s): 291 - 300

A Flexible Divide-And-Conquer Protocol for Multi-View Peer-to-Peer Live Streaming

Miao Wang, Lisong Xu and Byrav Ramamurthy
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115 USA
{mwang, xu, byrav}@cse.unl.edu

***Abstract*—Multi-view peer-to-peer (P2P) live streaming systems have recently emerged, where a user can simultaneously watch multiple channels. Previous work on multi-view P2P streaming solves the fundamental inter-channel bandwidth competition problem at the individual peer level, and thus can be used with very limited types of streaming protocols. In this paper, we propose a new protocol for multi-view P2P streaming, called Divide-and-Conquer (DAC), which efficiently solves the inter-channel bandwidth competition problem using a divide-and-conquer strategy at the channel level, and thus is flexible to work with various streaming protocols. This makes DAC more suitable for upgrading current single-view P2P live streaming systems to multi-view P2P live streaming systems. Our extensive packet-level simulations show that DAC is efficient in allocating the overall system bandwidth among competing channels, is flexible in working with various streaming protocols, and is scalable in supporting a large number of users and channels.**

I. Introduction

Peer-to-peer (P2P) live streaming systems have been extensively studied [1], and most of the earlier works focus on single-view P2P live streaming, where a user can subscribe to and watch only one channel at a time. Multi-view[1] P2P live streaming has recently emerged, where a user can simultaneously subscribe to and watch multiple channels. For example, PPStream [2] supports a limited multi-view capability with the picture-in-picture feature.

In general, multi-view P2P streaming can be used in two types of applications: 1) Multi-View Internet TV Applications: Fig 1 illustrates a possible case where a user can enjoy a high-quality movie channel shown in a large window, while still being able to monitor the weather information on another channel displayed in a small window. 2) Multi-Camera Live Streaming Applications: Fig 2 illustrates a possible case where a user can watch a stock-car race from several selected cameras, such as a pit-box camera, a corner camera, and a driver's point-of-view camera.

A fundamental difference between single-view and multi-view P2P streaming is that the latter has a unique inter-channel bandwidth competition problem, which is how to optimally allocate the upload capacity of a user among its subscribed channels. This problem was first studied by Wu

[1]We use multi-view to refer to the case where a user can simultaneously watch multiple different channels. These channels are not necessarily correlated like in tele-immersive systems.

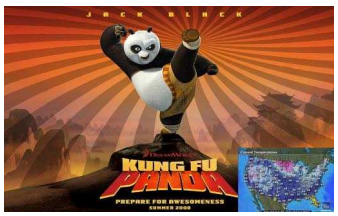

Fig. 1. Multi-View Internet TV application.

Fig. 2. Multi-camera live streaming of stock-car racing.

et al. [3] [4]. They proposed a protocol based on game theory, where all users in a multi-view P2P streaming system participate in a decentralized collection of bandwidth auctions with the goal to optimally allocate the system bandwidth among different channels. However, since the inter-channel bandwidth competition problem is solved at the individual peer level, it requires streaming protocols that can efficiently use the bandwidth allocated for each pair of users, such as network-coding-based streaming protocols [5].

While the inter-channel bandwidth competition problem is important for multi-view P2P streaming, another equally important problem is the choice of the streaming protocol used within a channel (referred to as the intra-channel streaming problem). A streaming protocol includes both an overlay construction method and a block scheduling algorithm, and it greatly influences the system streaming quality. Various streaming protocols [1] have been well studied and tested in the current single-view P2P streaming systems. Since most commercial P2P streaming systems construct a mesh-based topology which is resilient to peer churns, in this paper, we also assume a mesh overlay topology and focus on various block scheduling algorithms when studying streaming protocols.

Wu *et al.* [4] solve both the inter-channel bandwidth competition problem and the intra-channel streaming problem for multi-view P2P streaming systems at the peer level (or in other words, they solve both problems all at once, referred to as *AAO*). Therefore, their solution limits the choices of streaming protocols to those based on network coding. Even though network coding has been proven to be feasible for P2P live streaming [5], especially with relatively cheap GPU [6], few (if any) commercial P2P streaming systems actually use it due to various reasons (e.g. the implementation cost, new hardware requirements at end users etc). More importantly,

since there is no single streaming protocol that is better than all other streaming protocols in every aspect, commercial P2P streaming systems actually use different streaming protocols for different purposes. Therefore, in this paper, we are interested in the following problem: *Can we design a protocol for multi-view P2P live streaming that can efficiently solve the inter-channel bandwidth competition problem and is flexible to incorporate any streaming protocol?*

The answer to this question is *Yes* with our proposed DAC protocol. Inspired by the *divide and conquer* strategy, DAC first divides the overall system problem into several small channel problems, and then solves each channel problem separately. Specifically, DAC first solves the inter-channel competition problem to optimally allocate the bandwidth to different channels, and then solves the intra-channel streaming problem individually to achieve a good streaming quality for each channel.

There are three design goals for DAC: 1) Flexibility: The system should be able to incorporate various intra-channel streaming protocols. 2) Efficiency: The system should achieve a good overall streaming quality (e.g. packet delivery ratios) for all users across all channels. 3) Scalability: The system should be able to maintain a good overall streaming quality in a large-scale system with a large number of channels and users.

Flexibility and scalability are achieved by the divide and conquer strategy, in that it solves the inter-channel bandwidth competition problem and intra-channel streaming problem separately and divides the large problem into several smaller problems. Since DAC solves the inter-channel bandwidth competition at the channel level, one challenge of using divide and conquer strategy is how to effectively measure the information of each channel (e.g. the total upload bandwidth demand and supply), so as to achieve a reasonably good accuracy with affordable measurement overheads. DAC uses the statistical sampling method based on continuous-time random walk, which satisfies the accuracy and overhead requirements.

For the efficiency goal, DAC allocates the upload bandwidth to different channels according to their demands via our proposed utility-based optimal resource allocation model. This model is aware of the inter-channel bandwidth competition and has a larger feasible region than [4] (refer to Sec III-B). We evaluate DAC with extensive and carefully designed packet-level simulations and the results show that DAC meets the three design goals well.

The rest of this paper is organized as follows. Section II briefly summarizes the related work. Section III describes implementation details of our proposed DAC protocol. Section IV evaluates the performance of DAC with extensive packet-level simulations. Finally, we conclude the paper in Section V.

II. Related Work

Related work on multi-view P2P streaming: There is very little work on multi-view P2P streaming. Liang *et al.* [7] present a general framework for future IPTV based on multi-view P2P streaming, which supports content-based channel selection, multi-channel view customization, and semantics-aware bandwidth allocation. However, they only consider how to allocate the download capacity of a user to different channels. In contrast, our work focuses on how to allocate the upload capacity of a user to different channels, since the upload capacity is a more precious resource than the download capacity in the current Internet. Wang *et al.* [8] study the neighbor selection problem in multi-view P2P streaming systems and propose a simple neighbor selection algorithm based on peers' subscribed channels and upload bandwidth. However, they do not consider how to optimally allocate the upload bandwidth of peers watching multiple channels.

TABLE I
The Comparison of Three Protocols

Design Concerns	DAC	AAO	ISO
Rationale	Divide-and -Conquer	All-At -Once	Direct Exten--sion
Flexibility	Yes	No	Yes
Efficiency	Allocation with *Relaxed* Constraints for larger feasible region	Allocation with *Tight* Constraints for smaller feasible region	No
Scalability	Yes	Yes	Yes
Dynamics	Pause DAC in high dynamics	Use old values and loosely synchronized	-
Convergence	Yes, < 10 sec for 32 channels, 20,000 peers	Conditional, < 10 sec, 4 channels, 20,000 peers	-

The most related works are [3], [4] by Wu *et al.*, which tackle the inter-channel competition and intra-channel streaming simultaneously via organizing decentralized collections of bandwidth auctions at the peer level (*AAO*). Even though the proposed protocol has been proved to achieve Nash-Equilibrium and optimal allocation with *tight* constraints, it requires network coding for intra-channel streaming, which limits its flexibility of using the existing protocols. Compared with [3], [4], we solve the two problems separately with the goal of providing a *flexible* framework for existing protocols as well as achieving a good overall performance for all channels.

In Table I, we compare the three protocols DAC, AAO [2] and ISO. ISO is a direct extension from single-view systems and different channels are always isolated from one another (refer to Sec IV). Dynamics refer to the peer churn. AAO converges quickly when the total bandwidth supply is greater than the total bandwidth demand; while DAC always converges due to the relaxed constraints.

There is also extensive research work on related applications, such as multi-party and multi-stream systems [9], [10], which include multi-camera video conferencing and 3D tele-immersion. These applications usually consider a small number of relatively stable users due to the real-time interactivity constraint, whereas our multi-view P2P streaming applications consider a large number of dynamic users.

[2]Note that AAO refers to the protocol designed by Wu *et al.* [3] thereinafter, unless explicitly explained

Related work on intra-channel block scheduling: Since most of the commercial P2P streaming systems deployed over the Internet are mesh-based [2], which is resilient to peer churns, in this subsection, we only briefly summarize various block scheduling algorithms. Random scheduling is proposed, due to its simplicity and high performance with proper configuration [11]; Adaptive queue based chunk scheduling [12], mini-cost scheduling [13], randomized decentralized broadcasting [14] are examples of optimal scheduling algorithms to fully utilize the resources and achieve the maximum streaming rate. Other scheduling algorithms include rarest-first scheduling (DONet/Coolstreaming [15]), Chainsaw [16], PRIME [17] etc. Network-coding-based streaming protocols, originated from information theory, enhance the traditional block scheduling algorithms mentioned above, since they allow information mixture in peers, which simplifies the block scheduling and increases the data diversity. Wang *et al.* [5] perform a reality check for network coding and [18] proposed a market model for applying network coding. However, few (if any) commercial P2P streaming systems actually use network coding due to various reasons (e.g. it requires extra computation at end users for coding/decoding etc).

The coexistence of different block scheduling algorithms implies that there is no single streaming protocol that is better than all other streaming protocols in every aspect. Therefore, the flexibility to incorporate various streaming protocols is one of our primary design goals for DAC.

III. The Divide And Conquer Protocol (DAC)

In this section, we introduce the proposed DAC protocol from the perspective of how DAC meets the three design goals. We first focus on introducing the divide and conquer strategy with examples in Sec. III-A, which is the key design rationale of DAC to achieve flexibility and scalability. Sec. III-B describes the utility-based optimal bandwidth allocation model and algorithms, which mainly contributes to the goal of efficiency. In Sec. III-C and III-D, we discuss the statistical sampling scheme for channel information measurement and the distributed method for disseminating bandwidth allocation results to users, which makes DAC scale well. Finally, Sec. III-E describes how DAC deals with network *dynamics* (e.g. peer joining/leaving etc), which is a critical issue in all P2P systems.

A. Divide and Conquer Strategy

We explain the basic idea of DAC with the example shown in Fig 3. There are a total of three channels: A, B, and C. Some users watch only a single channel, and some users watch multiple channels. All users watching the same channel form a single P2P overlay for the channel, and there are some overlaps between different P2P overlays as shown on the left side of Fig 3. We do not show the overlay topology for each channel (i.e. how the users in a P2P overlay are connected to each other), in order to emphasize that DAC has no specific requirement on the topology of a P2P overlay.

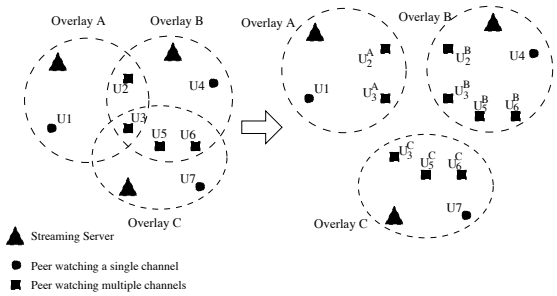

Fig. 3. DAC splits three *physically overlapping* P2P overlays into three *logically disjoint* P2P overlays.

To achieve flexibility and scalability, DAC follows the divide-and-conquer strategy to divide the overlapped overlays into different independent overlays (corresponding to different channels). Then, it solves the inter-channel competition at the channel level, which is different from [3], [4], which solve the problem at the peer level. Therefore, DAC does not have any specific requirement for intra-channel streaming protocols. For example, DAC splits three *physically overlapping* P2P overlays into three *logically disjoint* P2P overlays as shown in Fig 3. User U_2 is split into two logical users U_2^A and U_2^B, each of which has its own upload capacity and does not interfere with one another. Note that the upload capacity of physical user U_2 is the sum of the upload capacities of logical users U_2^A and U_2^B.

B. Optimal Bandwidth Allocation

To achieve the efficiency design goal, DAC properly allocates the peers' upload bandwidth to their subscribed channels by considering competitions for upload bandwidth among these channels due to their upload bandwidth imbalance [3]. As previously mentioned, DAC solves the bandwidth allocation at the channel level based on the divide and conquer strategy. Therefore, we first describe how DAC efficiently splits multiple physically overlapping P2P overlays into multiple logically disjoint P2P overlays by efficiently splitting each physical user into multiple logical users, one for each subscribed channel.

In order to use the divide and conquer strategy and achieve better scalability, DAC makes the splitting decision for a group of users who watch the same set of channels, instead of considering the splitting decision for each individual user. For example, in Fig 3, DAC considers the splitting decision for both U_5 and U_6 together, since both of them watch channels B and C. Let Θ denote the set of all channels. For example, $\Theta = \{A, B, C\}$ for Fig 3. For a subset of channels $\theta \subseteq \Theta$, let S_θ denote the set of users, who are watching *just* the channels in channel set θ. As an example, if $\theta = \{B, C\}$, then user set S_θ (also written S_{BC}) denotes the set of users watching just channels B and C, and in Fig 3, $S_{BC} = \{U_5, U_6\}$. A streaming server is considered as a special user who only contributes its upload capacity and belongs to the corresponding user set.

For each user set S_θ, DAC considers how to optimally allocate the total upload bandwidth of all users in S_θ to all

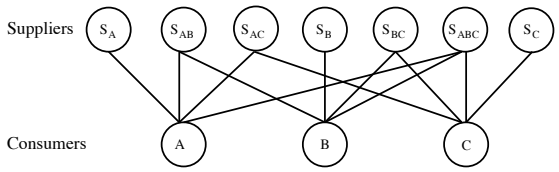

Fig. 4. A resource allocation graph for a multi-view P2P streaming system with three channels A, B and C.

channels $c \in \theta$. Intuitively, a user set S_θ provides its upload bandwidth to some channels and a channel c requests upload bandwidth from some user sets, therefore, we call a user set *a bandwidth supplier* and a channel *a bandwidth consumer*. The relationship between suppliers and consumers can be described by a bipartite resource allocation graph $G = (S, D, E)$, where vertex set S is the set of all suppliers (i.e., S contains S_θ for any $\theta \subseteq \Theta$), vertex set D is the set of all consumers (i.e., D contains c for any $c \in \Theta$), and edge set E represents the supplier-consumer relationship (i.e., $e = (\theta, c) \in E$ iff $c \in \theta$). Fig 4 illustrates the bipartite graph with 7 suppliers and 3 consumers for a multi-view P2P streaming system with M=3 channels: A, B, and C. For example, supplier S_{BC} allocates its upload bandwidth to consumers B and C.

1) Optimal Bandwidth Allocation Model: With the resource allocation graph $G = (S, D, E)$, we can model the upload bandwidth allocation problem as solving the global optimization problem below

$$\text{subject to} \quad \max_{a \geq 0} \sum_{(\theta,c)\in E} U_c^\theta(a_c^\theta) \tag{1}$$

$$\sum_{c\in\theta} a_c^\theta \leq B^\theta \qquad \forall\theta \tag{2}$$

where B^θ is the total upload bandwidth of all users in S_θ, and a_c^θ is the bandwidth to be allocated from supplier S_θ to consumer c. $U_c^\theta(\cdot)$ is the utility function of c associated with bandwidth obtained from S_θ. The constraint means that the total allocated bandwidth from supplier S_θ cannot exceed its total upload bandwidth.

Compared with [3], we relax the constraint that the total allocated bandwidth should be greater than or equal to the desired bandwidth by each consumer, in order to guarantee the convergence of the algorithm in the case of bandwidth fluctuations. The measurement of P2P networks [19] shows that the upload bandwidth fluctuates frequently due to congestion, jitter etc. of the underlying physical network and peer dynamics (e.g. joining/leaving the overlay). Therefore, the model for bandwidth allocation should consider the bandwidth fluctuation. Otherwise, the convergence of the allocation algorithm corresponding to the model will be affected when the fluctuation causes violations to the constraints.

We determine the utility function as follows: the utility obtained by each channel should be non-decreasing with respect to the allocated bandwidth. To achieve the efficiency goal, the solution to problem (1) should allocate bandwidth based on the demand of each consumer (i.e. to solve the competitions among different consumers). In order to make (1) a computationally solvable problem, we follow [20] to assume the utility function be an increasing and twice differentiable concave function, which is also practical. The utility function used in this paper is formulated as

$$U_c^\theta(a_c^\theta) = R_c \log(1 + a_c^\theta)$$

where R_c represents the c's bandwidth demand and the utility function is always non-negative. Moreover, due to the strict concavity of the logarithmic function used in the above utility function, the optimal bandwidth allocation strategy to convex program (1) is proportionally fair [21], which means that the solution to (1) allocates bandwidth based on each channel's demand. Our proposed protocol DAC uses the sampling method to determine R_c and B^θ, which is scalable and will be described in following sections. In the next subsection, we propose a distributed algorithm for the global optimization problem (1) for a large-scale system.

2) Algorithms for Solving Convex Problem (1)*:* The distributed solution to problem (1) is based on the standard dual decomposition [22], which is referred to *Dual-Algorithm* in the remaining of the paper. Before developing the Dual-Algorithm, we first establish the Lagrangian of (1)

$$\begin{aligned} L(\mathbf{a}, \boldsymbol{\lambda}) &= \sum_{e\in E} U_c^\theta(a_c^\theta) + \sum_\theta \lambda^\theta (B^\theta - \sum_{c\in\theta} a_c^\theta) \\ &= \sum_{e\in E} [U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta] + \sum_\theta B^\theta \lambda^\theta \\ &= \sum_{e\in E} L_{c,\theta}(a_c^\theta, \lambda^\theta) + \sum_\theta B^\theta \lambda^\theta \end{aligned} \tag{3}$$

where $e = (\theta, c)$ is an edge in the resource allocation graph indicating the bandwidth that c obtains from S_θ, $\lambda^\theta \geq 0$ is the Lagrange multiplier (bandwidth price of multi-view user set S_θ) associated with the linear capacity constraint (2) of S_θ, and $L_{c,\theta}(a_c^\theta, \lambda^\theta) = U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta$ is the Lagrangian associated with the edge (θ, c) to be maximized on that edge by the consumer c.

Based on the dual decomposition, each edge $e \in E$ whose starting vertex is c, for the given λ^θ, solves

$$a_c^{\star\theta}(\lambda^\theta) = \arg\max_{a\geq 0} [U_c^\theta(a_c^\theta) - \lambda^\theta a_c^\theta] \qquad \forall c \tag{4}$$

which is unique due to the strict concavity of $U_c^\theta(\cdot)$. The master dual problem which determines the bandwidth price of S_θ, is

$$\text{subject to} \quad \min_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}) = \sum_c g_c(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \mathbf{B} \tag{5}$$

$$\boldsymbol{\lambda} \geq 0 \tag{6}$$

where $g_c(\boldsymbol{\lambda}) = L_{c,\theta}(a_c^{\star\theta}(\lambda^\theta), \lambda^\theta)$. The unique solution to (4) indicates that the dual function $g(\boldsymbol{\lambda})$ is differentiable and therefore there exists a gradient method that updates the λ^θ at each iteration

$$\lambda^\theta(t+1) = [\lambda^\theta(t) - \alpha(B^\theta - \sum_{(\theta,c)\in E} a_c^{\star\theta}(\lambda^\theta(t)))]^+ \ \forall\theta \tag{7}$$

where t is the iteration index, $\alpha > 0$ is the step size, $[\cdot]^+$ represents the nonnegative orthant projection.

Theorem 1: The Dual-Algorithm solves the problem (1) in a distributed manner.

Proof Sketch: Due to the concavity of the utility function, the duality gap for problem (1) is zero. Therefore, the dual variable $\boldsymbol{\lambda}(t)$ converges to $\boldsymbol{\lambda}^\star$ as $t \to \infty$. The solution to (4) has a unique solution and the primal variable $a_c^{\star\theta}(\boldsymbol{\lambda}(t))$ will converge to the primal optimal variable $\boldsymbol{a}^\star$. Detailed proofs for the convergence of concave maximizations are available in [23]. Problem (4) can be independently solved on the edges at each consumer and the gradient based update (7) can be independently carried out at each supplier.

We summarize the algorithms carried out at the consumers and suppliers at round t in Algorithm 1 and 2.

```
1 Wait for update messages of λ^θ
2 foreach θ, such that edge (θ, c) ∈ E do
3     Independently solve the problem (4) and submit
      the solution to corresponding S_θ
4 end
```

Algorithm 1: Consumer c at round t

```
1 Wait for update messages of a_c^{⋆θ},   ∀(θ, c) ∈ E
2 Independently update λ^θ(t) with the λ^θ(t − 1) and
  the updated a_c^{⋆θ}
3 foreach c, such that edge (θ, c) ∈ E do
4     Send the new λ^θ(t) to consumer c
5 end
```

Algorithm 2: Supplier S_θ at round t

3) Discussions: What if there are a large number (i.e. M) of channels and then a large number (i.e. 2^M) of suppliers? Notice that each supplier S_θ with $|\theta| = 1$ has only one consumer, so it does not need to run the allocation algorithm 2 and it can directly allocate all of its bandwidth to the consumer. Furthermore, in order to achieve better scalability, only if a supplier has a large enough number of users, does it run the bandwidth allocation algorithm (i.e. the set of users has sufficiently large impact on the system performance). Specifically, supplier S_θ runs allocation algorithm 2, only if $N_\theta/N > \alpha$, where N_θ is the number of users in S_θ (i.e. $N_\theta = |S_\theta|$), N is the total number of users across all channels (i.e. $N = \sum_{\theta \subseteq \Theta} N_\theta$), and α is a system parameter. Note that this implies that there are at most $1/\alpha$ suppliers running the allocation algorithm. For example, if $\alpha = 0.001$, then there are at most $1/\alpha = 1000$ concurrent allocations in the system. If supplier S_θ does not run an allocation, it directly allocates its bandwidth to its consumers in proportion to their corresponding streaming rates (i.e. r^c for consumer c). For a system with a large number of channels, this method can significantly reduce the total number of concurrent allocations while not greatly affecting the system efficiency (based on our simulation, for 32 channels with 20,000 peers, it converges within several seconds). The value of α is determined by the required accuracy of the bandwidth allocation. Smaller α provides better accuracy due to better approximation of the bandwidth allocation in the system.

What if there is insufficient bandwidth for the system? In case of insufficient bandwidth, the system either suffers a degraded quality of service, if all the channels are considered equally important, or provides differentiated quality of service depending on the priorities of different channels. The proposed bandwidth allocation program (1) has the potential to provide differentiated QoS, in that we can change the order of utility functions based on the priority of each channel. Therefore, channels with higher priorities have the privileges to obtain more bandwidth to sustain their service quality than those with lower priorities.

How to implement the concurrent allocations? We require that there will be a small group of dedicated allocation servers in the system, each handling multiple suppliers. The total number of allocation servers is proportional to the value of $1/\alpha$. We expect that very few allocation servers will be necessary for a small value of $1/\alpha$, such as 1000. Nevertheless, additional allocation servers can provide better fault tolerance. The tracker server (bootstrap server) of each channel can act as the consumer for the channel. Since it only needs to communicate with a small group of allocation servers for at most $1/\alpha$ allocations during each allocation round, we do not expect that this would overload the tracker server.

C. Measuring System Information Required by the Allocations

In this section, we describe our design choices and implementation details on how to measure the system information required by the bandwidth allocation.

1) Information to measure: To allocate its upload bandwidth, supplier S_θ must know B^θ which is the total upload bandwidth of all users watching just the channels in θ. To determine whether to allocate it bandwidth, supplier S_θ must know N_θ and N (i.e. whether $N_\theta/N > \alpha$), where N_θ is the total number of users watching just the channels in θ and N is the total number of users in the system. The consumer c use formula $R_c = N_c \times r_c \times \gamma$ to determine the bandwidth demand of channel c, which is used in the utility function, where r_c is the streaming rate of channel c and γ is a scalar for considering the control overhead required by intra-channel streaming protocols (e.g. random block scheduling achieves near optimal performance with $\gamma \geq 1.1$ [11]). Since r_c and γ are known for channel c, consumer c only needs to measure N_c which is the total number of users watching channel c.

2) Design Choices: One straightforward method to measure the above required information is to use a distributed information management system, such as DHT-based RandPeer [24], to keep track of the information of all users. This method can accurately measure the required information. However, in order to keep track of the information of dynamic users (joining/leaving/failure), this method generates a significant amount of traffic overhead between users and the information management system. That is, it is not scalable to systems with a large number of users and channels.

Instead of directly keeping track of the information of all users, DAC adopts a sampling method that statistically measures the information of all users with a reasonably good accuracy and an affordable traffic overhead. Sampling

methods [25], [26], [27] have been studied recently for selecting peers uniformly at random from a P2P overlay. The difficulty lies in how to select peers uniformly at random in a dynamic and heterogeneous P2P overlay, where peers may join and leave the overlay and have different numbers of neighbors. There are two types of unbiased sampling methods: Metropolized Random Walk with Backtracking (MRWB) [26] method based on Metropolis-Hastings method for Markov Chains, and Sample and Collide (S&C) [27] method based on Continuous Time Random Walk. While both the MRWB method and the S&C method can be used to uniformly sample the information of peers, the S&C method can also be used to estimate the total number of peers in a group. Therefore, DAC chooses the S&C method to measure the required information.

3) Implementation Details: Considering that $B^\theta = N_\theta \times \overline{b^\theta}$ where $\overline{b^\theta}$ is the average upload capacity of every user in S_θ, DAC first measures N_θ and $\overline{b^\theta}$, and then calculates B^θ as $N_\theta \times \overline{b^\theta}$. Overall, DAC needs to measure four types of information: N, N_c for any $c \in \Theta$, N_θ for any $\theta \subseteq \Theta$, and $\overline{b^\theta}$ for any $\theta \subseteq \Theta$.

There are a few sampling servers in the system (for fault-tolerant reasons), which are responsible for statistically measuring all the required information, and periodically reporting them to each consumer and each supplier. Below we first explain how the sampling server measures N, and then explain how it measures N_c, N_θ and $\overline{b^\theta}$.

Every Δt time interval, the sampling server uses the S&C method [27] to measure N as described below.

- The sampling server randomly identifies a series of users in the system as initiators.
- Each initiator initiates a continuous-time random walk, which may cross different channels if a visited user watches multiple channels. The random walk will finally stop at a uniformly selected user.
- Each selected user reports its information such as its unique user ID, its upload capacity, and its subscribed channels to the sampling server.
- The sampling server keeps identifying new initiators until it obtains the information of n selected users such that there are exactly β pairs of equal user IDs among these n selected users (called β collisions in [27]).
- Finally, N can be estimated by solving the following equation with the standard bisection search.

$$\sum_{i=0}^{n-\beta-1} \frac{i}{N-i} - \beta = 0 \tag{8}$$

According to the theory of the S&C method, any user in the system can be identified as an initiator, and it does not need to be uniformly selected. Therefore, any standard P2P neighbor selection method can be used to identify an initiator. However, in practice for better accuracy, we try to identify users in different channels and in different locations of the system. Note that, the sampling server can identify a series of initiators back to back, so that multiple continuous-time random walks can be performed by different initiators simultaneously.

Parameter β is a system parameter determining the accuracy of the estimation and the overhead of sampling traffic. The larger the size of a P2P system, the bigger the value of β to maintain a certain degree of accuracy. For example, based on our simulation results, for a P2P system with 20,000 users, $\beta = 50$ can achieve a good estimate with less than $\pm 10\%$ error and a light-weight sampling traffic.

The information N_c, N_θ, and $\overline{b^\theta}$ can be measured simultaneously while the sampling server is measuring N. Recall that the sampling server selects n users uniformly at random in the system, and it knows all the information of these n users. Therefore, N_c can be estimated by the product of N times the percentage of n peers watching channel c, N_θ can be estimated by the product of N times the percentage of n users watching just the channels in θ, and $\overline{b^\theta}$ can be estimated by the average upload capacity of all users (among these n users) watching just the channels in θ. We can see that it does not require any extra sampling traffic to measure N_c, N_θ, and $\overline{b^\theta}$.

D. Distributing Allocation Results to Users

Because there are at most $1/\alpha$ concurrent allocations at the suppliers, the proposed allocations converge very quickly. Therefore, every Δt time interval, DAC distributes only the final allocation results (i.e. the results at the optimal point) but not the intermediate results to users, in order to reduce the overhead of control traffic. DAC encapsulates the final results into a single packet, which is then distributed to all users in the system by using any standard gossip-style protocol. Note that the allocation servers do not have to directly distribute the allocation results to all users; instead, it sends the results to some randomly selected peers first and the selected peers will spread the results with the epidemic style update, which guarantees that all peers will receive the results in $O(\log(N))$ (N is the total number of peers in the system) rounds with high probability [28]. Even though this packet contains the result of each allocation, its size is not very large since there are at most $1/\alpha$ concurrent allocations. When a user in S_θ receives the packet, it checks whether the packet contains the result of allocation S_θ. If so, it allocates its upload capacity among its subscribed channels according to the received allocation result; otherwise, it allocates its upload capacity among its subscribed channels proportional to their corresponding streaming rates (i.e. r^c for channel c). When a new user joins the system or when an existing user changes its subscribed channels, it also allocates its upload bandwidth to its subscribed channels proportional to their streaming rates until it receives a packet containing the allocation results.

E. DAC Dynamics

An important feature of a real P2P system is user dynamics; that is, a user may randomly join or leave the system, and change its subscribed channels. In response to user dynamics, DAC *periodically* performs the divide-and-conquer strategy to divide the system into different sets of logically disjoint P2P overlays at every Δt time interval as illustrated in Fig 5. The

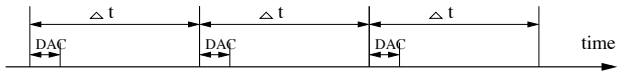

Fig. 5. DAC periodically performs the divide-and-conquer strategy every Δt time interval in response to user dynamics.

response time period Δt is a system parameter, which depends on how long DAC takes to perform the divide-and-conquer strategy, how much control overhead DAC generates, and how dynamic the system is.

DAC sets Δt on the order of minutes, for example 2 minutes in our simulations, for the following reasons: 1) It takes only a short time on order of seconds for DAC to perform the divide-and-conquer strategy, and then it introduces only a light-weight control overhead for performing DAC once every Δt time interval which is on the order of minutes. 2) Recent P2P measurement studies [29], [30], [15] show that a P2P system is relatively stable over an interval of minutes, because most users have a lifetime longer than a minute, and at any time instant a significant percentage of users (e.g. $> 70\%$ on average reported in [30]) even have a lifetime on the order of hours. On average, the percentage change of a P2P system population in a minute [29], [15] is usually less than 1%. Therefore, we believe that a Δt on the order of minutes is fast enough for DAC to respond to user dynamics.

When a new user joins the system or when an existing user changes its subscribed channels, it allocates its upload bandwidth to its subscribed channels in proportion to their streaming rates until it receives a divide-and-conquer result from DAC. In the special cases when a large number of users simultaneously join or leave a P2P overlay (e.g. at the beginning and the end of a program), DAC can detect the sharp population change and then temporarily pause the divide-and-conquer strategy until the system population becomes relatively stable. Therefore, even in these special cases, a system with DAC should perform at least as good as a system without DAC.

IV. Simulation Results

In this section, we use packet-level simulations to evaluate the performance of our proposed DAC protocol.

A. Simulation Setup

We have developed a multi-view P2P streaming simulator based on the single-view P2P streaming simulator originally developed by Zhang [31]. In addition to supporting multiple channels and multiple views, the original simulator is also enhanced so that it can simulate a much larger system with up to 32 channels and 100,000 users at the packet level.

We simulate three protocols for the inter-channel competition problem in multi-view P2P streaming: 1) Our DAC protocol based on the divide-and-conquer strategy; 2) The protocol [3] by Wu *et al.* (referred to as AAO) based on the all-at-once strategy; 3) A reference protocol in which each user always allocates its upload capacity to its subscribed channels in proportion to their streaming rates (referred to as ISO) so that different channels are always isolated from one another.

For all three protocols DAC, AAO, and ISO, we construct an overlay with a mesh topology for each individual channel. Since we are interested in the capability of DAC and AAO in supporting various block scheduling algorithms other than network coding, we simulate two blocking scheduling algorithms: random scheduling representing simple scheduling algorithms, and min-cost scheduling [13] representing optimization-based scheduling algorithms.

In order to evaluate our proposed DAC protocol, we have to configure three groups of parameters, which are different from simulations in single-view systems [11]). The three groups of parameters are: 1) the DAC protocol parameters; 2) the system bandwidth information parameters; 3) the channel and peer information parameters.

The DAC protocol parameters include the time interval Δt for running DAC (default value is 2 min), the system parameter α for the maximal number of concurrent allocations (default value is 0.001), the scalar γ for intra-channel streaming control overhead (default value is 1.1 [11]), and the collision number β for S&C sampling (default value is 50, selected based on our Group I simulations).

The system bandwidth information parameters include the streaming rate r_c for each channel c (default value $300Kbps$) and the resource index for each channel (the total upload bandwidth over the total required bandwidth [11] in that channel. The resource index varies in each group of simulations and will be provided separately). Note that the resource index for each channel is calculated based on the ISO protocol, which allocates multi-view peers' upload bandwidth based on the streaming rate of each subscribed channel. To achieve the desired resource index, we change the fraction of peers with upload bandwidth of 3 Mbps, 1Mbps, 784Kbps, 300Kbps, and 200Kbps. As in [4] [11], we assume that the peer's download bandwidth is enough for sustaining the channel's streaming rate.

The channel and peer information parameters include the number of channels M, the total number of peers N, the channel structure, and beta distribution parameters y, z. Beta distribution is a general type of statistical distribution, with the probability function $P(x) = \frac{(1-x)^{z-1}x^{y-1}}{B(y,z)}$, where $B(y,z)$ is the beta function defined as $B(y,z) = \frac{(y-1)!(z-1)!}{(y+z-1)!}$ [32]. We simulate a multi-view P2P system with one of the following three types of channel structures illustrated in Fig 9: a) a chain structure where a user can view only the feeds from either a single camera or two consecutive cameras in a row of cameras. b) a mesh structure where every user watches a random number of channels, and c) a star structure where there is one popular channel that every user watches. The population of each channel set is determined as follows: we first arrange the channel sets in lexicographical order and then assign a channel set a fraction f of the total number of peers N, which means the number of peers watching that channel set is $f*N$. We use the beta distribution to determine the fraction. Fig 6, 7, 8 illustrate the shapes of population distributions for chain, mesh, and star channel structures simulated in this

paper with a small number of channels as an example. Since the shape of the population distribution is determined by beta distribution, we will give the channel structure with parameters for the beta distribution in each group of simulations below.

Our simulation results fall into ***three*** categories based on different evaluation motivations for DAC. Group I: we evaluate the accuracy of sampling and then the impact of different sampling parameters on DAC. This group of simulation is also for selecting proper sampling parameter for Group II and III. Group II: we evaluate the flexibility of DAC compared with AAO using the two block scheduling algorithms. Group III: the comprehensive performance evaluation of DAC compared with ISO.

To compare the performance of DAC with ISO and AAO, we measure the packet delivery ratio of the system. The packet delivery ratio of a user for channel c is defined as the ratio of the total number of packets of channel c received by the user before the playback deadline to the total number of packets sent by the streaming server of channel c. The packet delivery ratio of channel c is defined as the average delivery ratio of all users watching channel c. Finally, the packet delivery ratio of the system is defined as the lowest delivery ratio among all channels. Intuitively, this is because the satisfaction of a user watching multiple channels is usually determined by the channel with the worst quality.

B. Group I: Impact of the Sampling Method on DAC

1) Sampling accuracy: Fig 10 shows the impact of collision number β on the sampling accuracy. We simulate a static system with a total of 16,800 users, and with a chain channel structure of 4 channels. We use beta function with parameters (1,1), whose shape is shown in Fig 6. We can see that when β is larger than 20, the estimated number of users is very close to the actual result. DAC sets β to 50 by default, which can achieve good sampling accuracy for systems with up to 100,000 users based on our simulation results. Fig 11 shows the estimated number of users in a very dynamic system where the total number of users first increases quickly from 10,000 to 60,000, and then drops to 20,000. Even in this case, the sampling method still achieves good accuracy.

2) Impact on DAC: To study the impact of the sampling method on the performance of DAC, we simulate two protocols: DAC and Oracle (as a reference protocol). The channel structure and the beta distribution is same as the above simulations. Oracle is very similar to DAC, except that it has the accurate information of the system and thus does not use the sampling method as DAC. We simulate the same system as the one simulated for Fig 10, where the total bandwidth of the system is enough to support all channels, but different channels have different resource indices with ISO. This is likely to happen in a P2P system with multiple channels as shown by a recent measurement study of PPLive [29]. The resource index with ISO for four channels are: 0.9, 1.3, 1.0 and 1.3. Fig 12 shows the bandwidth satisfaction ratio (the total allocated bandwidth over the total required bandwidth) of each channel with DAC and Oracle. We can see that when β is large enough (in this case 20), DAC with the estimated information achieves very similar results as Oracle.

C. Group II: Flexibility Evaluation DAC vs. AAO

We use two representative block scheduling algorithms as intra-channel streaming protocols and compare DAC with AAO. We vary three parameters to show that the performances of the two block scheduling algorithms depend on the flexibility of DAC and AAO. The three parameters are: the average resource index, the maximum number of neighbors and the streaming rate. By default, we generate 2000 peers and 4 channels with a mesh structure (the parameters for beta distribution is (2,2), whose shape is shown in Fig 7). The resource indices with ISO for 4 channels are: 0.9, 1.3, 1.0 and 1.3. The default streaming rate is 300 Kbps. The simulation time for all simulations is 1000 seconds.

From Figs 13, 14, 15, DAC always provides near optimal performance with both streaming protocols. Compared with DAC, AAO suffers bad performance, due to its inflexible design. Fig 13 shows that AAO needs more bandwidth to provide good performance for both streaming protocols. Fig 14 seems somehow counterintuitive. The reason is that AAO requires network coding to control the utilization of allocated bandwidth. Without network coding, the allocated bandwidth to each neighbor is poorly utilized, when the number of neighbors is large. Fig 15 shows that even with sufficient bandwidth (average index is 1.15), AAO's performance fluctuates with different streaming rates. From the comparison of AAO and DAC with the two intra-streaming protocols, we can conclude AAO is not as flexible as DAC.

D. Group III: Performance Evaluation of DAC vs. ISO

1) Systems with a large number of users: We simulate a system with a chain channel structure of 4 channels (parameters of beta distribution are (1,1)). Specifically, the resource index with ISO is 1.2, 1.0, 1.0, and 0.9 for channels A, B, C, and D, respectively. Fig 16 shows the packet delivery ratio of the system for DAC and ISO as the total number of users increases from 5000 to 20,000. Since the resource index of channel D with ISO is only 0.9, ISO achieves a poor packet delivery ratio. We can see that DAC outperforms ISO across a wide range of system sizes, due to efficiently allocating bandwidth among different channels. Fig 17 shows the results with similar simulation setting, except that the system has a mesh channel structure (parameters for beta distribution are (2,2)). We can see that DAC also outperforms ISO for a mesh channel structure.

2) Systems with a large number of channels: We simulate a system with a star channel structure (parameters of beta distribution are (0.8,0.8)). The number of channels varies from 2 to 32. The total bandwidth of the system is enough to support all channels, but different channels have different resource indices with ISO. Specifically, the resource index with ISO is 1.2 for half of the channels and 0.9 for the other half of channels. The total number of users is 10,000. Again, we can see that DAC outperforms ISO in a wide range

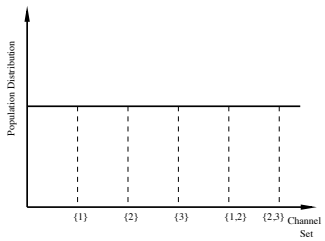

Fig. 6. Population distribution of chain structure with 3 channels, beta distribution with parameters (1,1).

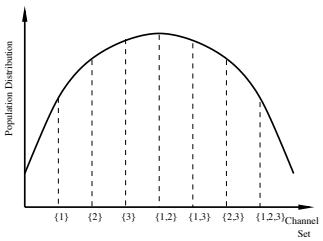

Fig. 7. Population distribution of mesh structure with 3 channels, beta distribution with parameters (2,2).

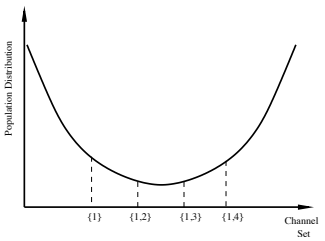

Fig. 8. Population distribution of star structure with 4 channels, beta distribution with parameters (0.8, 0.8).

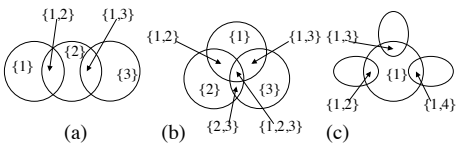

Fig. 9. Three types of channel structures: a) chain, b) mesh, and c) star.

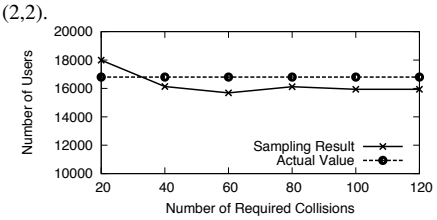

Fig. 10. Impact of collision number β on sampling accuracy in a static system.

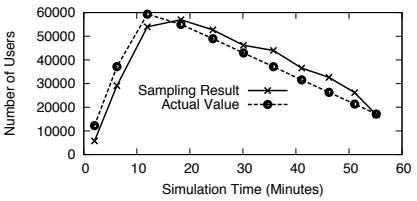

Fig. 11. Sampling a dynamic system.

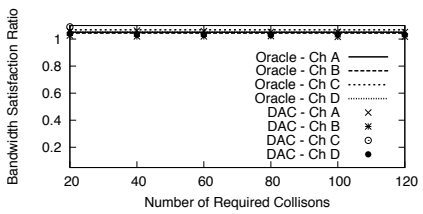

Fig. 12. For large enough β, the performance of DAC is insensitive to the value of β.

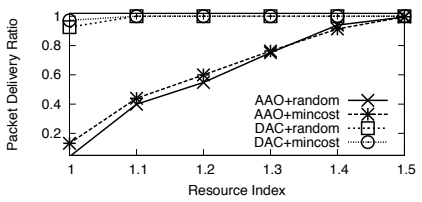

Fig. 13. DAC provides good streaming quality for various resource indices; AAO requires more bandwidth to achieve similar performance.

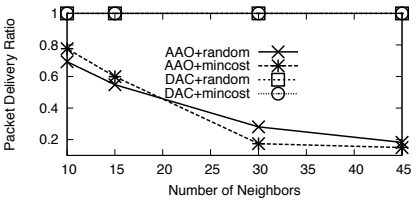

Fig. 14. AAO streaming quality decreases as the number of neighbors increases due to inefficient bandwidth utilization; DAC always achieves good performance.

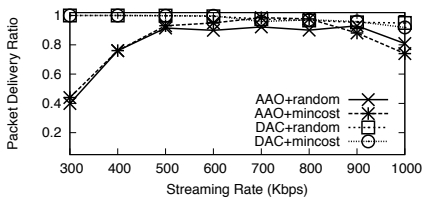

Fig. 15. AAO's streaming quality fluctuates with different video streaming rates; DAC always achieves good performance.

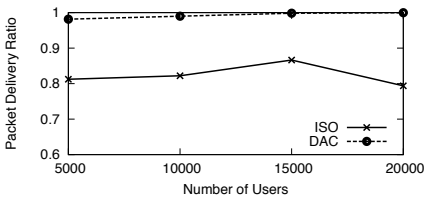

Fig. 16. DAC outperforms ISO in systems with a chain channel structure when the number of peers increases from an intermediate scale to a large scale.

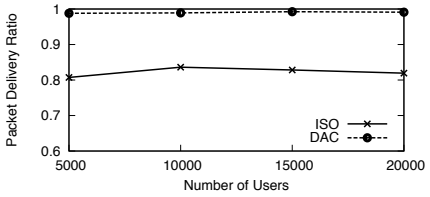

Fig. 17. DAC outperforms ISO in systems with a mesh channel structure when the number of peers increases from an intermediate scale to a large scale.

of channel numbers, due to efficiently allocating bandwidth among different channels.

3) Systems with a dynamic number of users: We simulate a dynamic system with a mesh channel structure of 4 channels (parameters of beta distribution are (2,2)). The average user arrival rate is 3 users per second, and the average life time of a user is 15 minutes. The average number of users is 20,000. The total bandwidth of the system on average is enough to support all channels, and the resource index with ISO on average is 1.2, 1.0, 1.0, and 0.9 for channels A, B, C, and D, respectively. Fig 19 shows the packet delivery ratio of each channel for DAC and ISO. ISO achieves a good packet delivery ratio for channels A, B, and C, but not for channel D because channel D has insufficient bandwidth with ISO. We can see that DAC achieves a good packet delivery ratio for every channel.

4) Systems with insufficient bandwidth: We simulate a system with a chain channel structure of 4 channels (parameters of beta distribution are (1,1)). The total number of users is 20,000. But the total bandwidth of the system is insufficient to support all channels. In this case, different channels are assigned different priorities by changing R_c in the utility function (e.g. R_A is larger than R_C, which means that channel A has higher priority than channel C). Specifically, channels A and B are assigned the highest priority. Channel C has lower priority than A and B. Channel D is assigned the lowest priority. Fig 20 shows the packet delivery ratio of each channel measured every 10 seconds. We can see that channels A and B achieve the highest delivery ratio and channel D achieves the lowest delivery ratio.

V. Conclusions

In this paper, we propose a flexible, efficient and scalable protocol called DAC for multi-view P2P streaming systems using a divide-and-conquer strategy. To achieve flexibility and

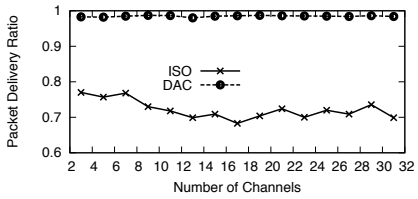

Fig. 18. DAC outperforms ISO in systems with a star channel structure when the number of channels increases from small to large.

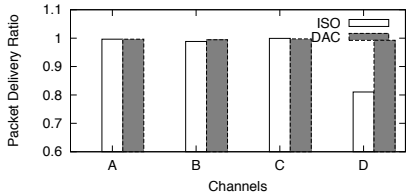

Fig. 19. DAC outperforms ISO in dynamic systems for a large scale network.

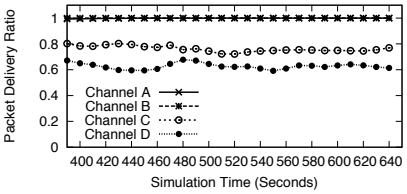

Fig. 20. DAC provides a better packet delivery ratio to a channel with a high priority, when the total upload bandwidth is insufficient.

scalability, DAC solves the inter-channel competition problem at the channel level, compared with existing work AAO, which solves the problem at the peer level. Moreover, DAC integrates with the statistical sampling module to measure the system information used by DAC, and achieves reasonably good accuracy with affordable overheads. To meet the efficiency goal, DAC allocates the upload bandwidth to different channels according to their demands via our proposed utility based optimal resource allocation model. Our extensive packet level simulations show that DAC achieves the three design goals.

ACKNOWLEDGEMENTS

We thank Chuan Wu for her generous help in answering our questions about [3], [4]. We thank Meng Zhang for his generous help in answering our questions about his P2P streaming simulator [31]. The work reported in this paper is supported in part by UNL Layman Award and NSF CAREER 0644080.

REFERENCES

[1] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Journal of Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, March 2008.

[2] PPStream, http://www.ppstream.com.

[3] C. Wu and B. Li, "Strategies of conflict in coexisting streaming overlays," in *Proceedings of IEEE INFOCOM*, Anchorage, AK, May 2007.

[4] C. Wu, B. Li, and Z. Li, "Dynamic bandwidth auctions in multi-overlay P2P streaming with network coding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 806–820, June 2008.

[5] M. Wang and B. Li, "Lava: A reality check of network coding in peer-to-peer live streaming," in *Proceedings of IEEE INFOCOM*, May 2007.

[6] H. Shojania and B. Li, "Nuclei: GPU-accelerated many-core network coding," in *Proceedings of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[7] J. Liang, B. Yu, Z. Yang, and K. Nahrstedt, "A framework for future Internet-based TV broadcasting," in *Proceedings of IPTV Workshop*, Edinburgh, Scotland, May 2006.

[8] M. Wang, L. Xu, and B. Ramamurthy, "Channel-aware peer selection in multi-view peer-to-peer multimedia streaming," in *Proceedings of IEEE International Workshop on IP Multimedia Communications (IPMC) at ICCCN*, 2008, pp. 1–6.

[9] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nahrstedt, S. Jung, and R. Bajscy, "TEEVE: the next generation architecture for tele-immersive environments," in *Proceedings of International Symposium on Multimedia*, Irvine, CA, December 2005.

[10] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy, "Viewcast: View dissemination and management for multi-party 3D tele-immersive environments," in *Proceedings of ACM Multimedia*, Germany, September 2007.

[11] M. Zhang, Q. Zhang, and S. Yang, "Understanding the power of pull-based streaming protocol: Can we do better?" *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 8, pp. 1678–1694, 2007.

[12] Y. Guo, C. Liang, and Y. Liu, "Adaptive queue-based chunk scheduling for P2P live streaming," in *Proceedings of IFIP Networking*, 2008.

[13] M. Zhang, Y. Xiong, Q. Zhang, and S. Yang, "Optimizing the throughput of data-driven peer-to-peer streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, 2009.

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

[15] B. Li, Y. Qu, Y. Keung, S. Xie, C. Lin, J. Liu, and X. Zhang, "Inside the new Coolstreaming: principles, measurements and performance implications," in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.

[16] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, A. E. Mohr, and E. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proceedings of IPTPS*, 2005, pp. 127–140.

[17] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver-driven mesh-based streaming," in *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[18] X. Zhang and B. Li, "On the market power of network coding in p2p content distribution systems," in *Proceedings of IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[19] S. Saroiu, K. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking (MMCN) 2002*, San Jose, CA, USA, January 2002.

[20] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, September 1995.

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

[22] D. P. Palomar and M. Chiang, "Alternative distributed algorithms for network utility maximization: Framework and applications," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2254–2269, December 2007.

[23] S. H. Low and D. E. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," *IEEE Transactions on Networking*, vol. 7, no. 6, pp. 861–875, December 1999.

[24] J. Liang and K. Nahrstedt, "RandPeer: membership management for QoS sensitive peer-to-peer applications," in *Proceedings of IEEE INFOCOM*, Spain, April 2006.

[25] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proceedings of IEEE INFOCOM*, Hong Kong, March 2004.

[26] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," in *Proceedings of ACM IMC*, Rio de Janeiro, Brazi, October 2006.

[27] L. Massoulie, E. Merrer, A. Kermarrec, and A. Ganesh, "Peer counting and sampling in overlay networks : Random walk methods," in *Proceedings of ACM PODC*, Denver, Colorado, July 2006.

[28] D. Alan, G. Dan, H. Carl, I. Wes, L. John, S. Scott, S. Howard, S. Dan, and T. Doug, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, 1987, pp. 1–12.

[29] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, December 2007.

[30] F. Wang, J. Liu, and Y. Xiong, "Stable peers: Existence, importance, and application in peer-to-peer live video streaming," in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.

[31] M. Zhang, http://media.cs.tsinghua.edu.cn/~zhangm/.

[32] Beta-Distribution, http://mathworld.wolfram.com/BetaDistribution.html.