The Streaming Capacity of Sparsely-Connected P2P Systems with Distributed Control

Can Zhao and Xiaojun Lin
School of Electrical and Computer Engineering,
Purdue University, West Lafayette
Email: {zhao43,linx}@purdue.edu

Chuan Wu
Department of Computer Science,
The University of Hong Kong
Email: cwu@cs.hku.hk

***Abstract*—Peer-to-Peer (P2P) streaming technologies can take advantage of the upload capacity of clients, and hence can scale to large content distribution networks with lower cost. A fundamental question for P2P streaming systems is the maximum streaming rate that all users can sustain. Prior works have studied the optimal streaming rate for a complete network, where every peer is assumed to communicate with all other peers. This is however an impractical assumption in real systems. In this paper, we are interested in the achievable streaming rate when each peer can only connect to a small number of neighbors. We show that even with a random peer selection algorithm and uniform rate allocation, as long as each peer maintains $\Omega(\log N)$ downstream neighbors, where N is the total number of peers in the system, the system can asymptotically achieve a streaming rate that is close to the optimal streaming rate of a complete network. We then extend our analysis to multi-channel P2P networks, and we study the scenario where "helpers" from channels with excessive upload capacity can help peers in channels with insufficient upload capacity. We show that by letting each peer select $\Omega(\log N)$ neighbors randomly from either the peers in the same channel or from the helpers, we can achieve a close-to-optimal streaming capacity region. Simulation results are provided to verify our analysis.**

I. Introduction

With the proliferation of high-speed broadband services, the demand for rich multimedia content over the Internet, in particular high-quality video delivery over the Internet, has kept increasing. Streaming video directly from the server will require a large amount of upload bandwidth at the server, which can be costly. The service quality can also be poor when the clients are far away from the server. In addition, it may be difficult for the server bandwidth to keep up when the demand is exceedingly high. There have been different approaches to off-load traffic from the server, using either CDN (content distribution network) or P2P (peer-to-peer) technologies. Deploying a large CDN can introduce a high fixed cost. In contrast, P2P technologies are particularly attractive because they take advantage of the upload bandwidth of the clients, which does not incur additional cost to the video service provider. Several well-known commercial P2P live streaming systems have been successfully deployed, include CoolStreaming [1], PPLive [2], TVAnts [3], UUSee [4], PPStream [5]. A typical P2P live streaming system can now offer thousands of TV channels or movies for viewing, and may serve hundreds of thousands of users simultaneously [4].

In contrast to the practical success of these P2P live streaming systems, the theoretical understanding of the performance of P2P live streaming seems to be lagging behind, which may impede further improvement of P2P live streaming. For example, a basic question for a P2P live streaming system is that of its streaming capacity, i.e., what is the maximum streaming rate that all users can sustain? This question has been studied under the assumption of a complete network, where each peer can connect to all other peers simultaneously. Under this assumption, the maximum streaming capacity has been found in [6], and both centralized and distributed rate allocation algorithms to achieve this maximum streaming capacity have been developed [6]–[9]. However, the assumption of a complete network is impractical for any large-scale P2P streaming systems. In a real P2P streaming system, typically each peer is only given a small list of other peers (which we refer to as neighbors) chosen from the entire population, and each peer can only connect to this subset of neighboring peers (neighbors may not be close in terms of physical distance). The number of neighboring peers is often much smaller than the total population, in order to limit the control overhead.

When each peer only has a small number of neighbors, the P2P network can be modeled as an incomplete graph with node-degree constraints. In this case, the streaming capacity of P2P systems becomes more complicated to characterize. Liu et al. [10] investigate the case when the number of downstream peers in a single sub-stream tree is bounded. However, the number of neighbors that each peer could have over all sub-streams can still be very large (in the worse case it can be connected to all the other peers simultaneously). Some approximated and centralized solutions to solve the optimal streaming capacity problem on a given incomplete network have been proposed in [11]. However, for large-scale P2P streaming systems, such a centralized approach will be difficult to scale. Liu et al. [12] proposed a Cluster-Tree algorithm to construct a topology subject to a bounded node-degree constraint, which could achieve a streaming rate that is close to the optimal streaming capacity of a complete network. This result gives us hope that, even with node-degree constraints, a P2P network may achieve almost the same streaming rate as that of a complete network. However, the Cluster-Tree algorithm is not a completely de-centralized algorithm because it requires the tracker (a central entity) to apply the Bubble algorithm at the cluster level. The Bubble algorithm is a centralized algorithm. Some other works such as SplitStream [13] and Chainsaw [14] have also studied the problem of how

to improve the streaming capacity when there is a node-degree constraint. However, these works did not provide theoretical results on the achievable streaming rate. To the best of our knowledge, we have not been aware of a fully distributed algorithm in the literature that can achieve close-to-optimal P2P streaming capacity on incomplete networks.

All of the above works are for single-channel P2P systems. Today's P2P systems typically serve a large number of TV channels and movies at the same time. In most P2P streaming systems, peers exchange data only with other peers that are viewing the same channel. Hence, peers from different channels are isolated from each other. Recently, Wu et al. [15], [16] show that by allowing peers to exchange data with other peers that are not even viewing the same channel, the overall performance of a multi-channel system can be improved. Such cross-channel peer exchange is particularly helpful for channels that do not have enough upload capacity, and hence they need the upload capacity of peers from other channels to improve their streaming rate. [15], [16] have proposed a View-Upload Decoupling (VUD) algorithm that sets up a semi-permanent distribution group of peers, who are not necessarily the peers interested in viewing a channel, to help distribute the content of the channel. Although the VUD algorithm has been shown to improve the multi-channel streaming capacity, it is again a centralized algorithm and it assumes that all peers can connect to all other peers simultaneously, which is impractical for real systems.

In this paper, we are interested in the following question: without centralized control, how many neighbors does a peer in a large P2P network need to maintain in order to achieve a streaming capacity that is close to the optimal streaming capacity of an otherwise complete network? Further, can we develop fully-distributed algorithms for peer selection and rate allocation to achieve the close-to-optimal streaming capacity? This paper provides some interesting and positive answers to these questions. First, we show that, if each peer has $\Omega(\log N)$ neighbors, where N is the total number of peers in the system, close-to-optimal streaming rate can be achieved with probability approaching 1 as N goes to infinity. Further, in order to achieve this goal, each peer only needs to choose $\Omega(\log N)$ downstream neighbors uniformly and randomly from the entire population, and simply allocates its upload capacity evenly among all downstream peers. Only the server needs a slightly different peer selection policy (see Section II-B for details).

Next, we also extend our analysis to multi-channel systems, and allow peers from those channels with abundant upload capacity to help other channels with insufficient upload capacity. Again, we show that by using a simple and distributed algorithm where each peer randomly selects a small number of neighbors from peers belonging to the same channel and from the helper peers from other channels, a close-to-optimal streaming capacity region for multi-channel systems can be achieved with high probability. Hence, our results indicate that the benefit of VUD can be retained in a distributed manner without the assumption of complete networks.

The results that we obtain have a similar flavor as scaling-law results in wireless ad hoc networks [17]. Although such results only hold when the size of the network N is large, they do provide important insights into the dynamics of the system. For example, our analysis indicates that, with a random peer selection strategy, for each user the most likely bottle-neck for its streaming capacity is at the "last hop", i.e. the sum of the upload capacity allocated to this user by its immediate upstream neighbors. This insight suggests that we could focus on balancing the capacity at the *last* hop when designing new distributed resource allocation algorithms for P2P streaming systems. As an initial application of this insight, we show with an example that, by slightly adjusting the uniform rate-allocation strategy, we can indeed improve the probability of attaining the near-optimal streaming rate. Hence, we believe that the insights from these results can be very helpful for designing more efficient control algorithms for P2P streaming.

II. Single-Channel P2P Networks

In this section, we will show that even without centralized control, $\Omega(\log N)$ neighbors are sufficient for large single-channel P2P streaming networks. Specifically, we will show that just by letting each peer select its $\Omega(\log N)$ neighbors randomly, the close-to-optimal streaming rate could be achieved will high probability when the network size N is large.

A. System Model

We consider a peer-to-peer live streaming network with N peers and one source s. In the rest of the paper, we will use the terms "source" and "server" interchangeably. Similarly, we will use the terms "peer", "node", and "user" interchangeably. Denote the set of all peers and the source as V (thus, $|V| = N+1$). We assume that the source has a video file with infinite size to be streamed to all peers and it has a fixed upload capacity u_s. Denote the upload capacity of peer i as U_i, which is a random variable defined as follows: each peer has an upload capacity of $U_i = u$ with probability p and an upload capacity of $U_i = 0$ with probability $1-p$, i.i.d. across peers. Although this is a simplified ON-OFF model, we believe that the insights obtained from this model can also be generalized to other models on the distribution of upload capacity. We assume that $u_s \geq u$. Like other works [6], [11], [12], [18], we assume that the download capacity and the core network capacity are sufficiently large, and hence the only capacity constraints are on the upload capacity. Each peer $i \in V \backslash \{s\}$ has a fixed set $\mathcal{E}_i$ of M downstream neighbors. Similarly, the source has a set $\mathcal{E}_s$ of M downstream peers. We can then model the P2P network as a directed and capacitated random graph [19]. If $j \in \mathcal{E}_i$, assign a directed edge (i, j) from i to j. Let the set of all edges be E. Note that there may be multiple peers that have a common downstream neighbor. Define C_{ij} and C_{sj} be the streaming rate from peer i and source s, respectively, to peer j.

The value of $\mathcal{E}_i$, $\mathcal{E}_s$, C_{ij} and C_{sj} depend on the peer selection and rate allocation algorithms. Given such an algorithm, we can define the "streaming capacity" of the system as the maximum rate that the source could distribute the streaming

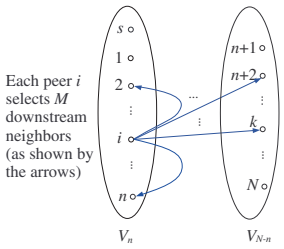

Fig. 1: Illustration of neighbor selection and a cut

content to all peers. For example, for a complete network, we have $\mathcal{E}_i = V\backslash\{i,s\}$ and $\mathcal{E}_s = V\backslash\{s\}$. [6] shows that the optimal streaming capacity is on average

$$C_f = \min\left\{u_s, \frac{u_s + \sum_{i\in V\backslash\{s\}} \mathbf{E}[U_i]}{N}\right\}, \qquad (1)$$

and can be achieved by setting $C_{ij} = U_i/(N-1)$ and $C_{sj} = U_s/N$ for all i, j. For our ON-OFF model of upload capacity, this optimal streaming capacity is equal to $C_f = \min\left\{u_s, \frac{u_s}{N} + up\right\}$. However, as we discussed in the introduction, the assumption of a complete network is impractical. In this paper, we are interested in the streaming capacity of an incomplete network, which can be calculated by the minimum cuts. Specifically note that for a given user t, a cut that separates s and t is defined by dividing the peers in V into a set V_n of size $(n+1)$ that contains the server, and the complementary set V_n^c of size $(N-n)$ that contains peer t, i.e.,

$$s \in V_n, |V_n| = n+1, t \in V_n^c \text{ and } |V_n^c| = N-n.$$

The capacity of cut C_n is defined as $C_n = \sum_{i\in V_n}\sum_{j\in V_n^c} C_{ij}$. See Fig. 1 for illustration.

Let $C_{\min}(s \to t)$ denote the minimum-cut capacity, which is the minimum capacity of all cuts that separate source s and destination t. It is well-known that this min-cut capacity is equal to the maximum rate from s to t. Let $C_{\min-\min}(s \to \mathcal{T})$ denote the min-min-cut which is the minimum cut of all individual min-cut capacities from the source to each destination t within a set $\mathcal{T}$, i.e.,

$$C_{\min-\min}(s \to \mathcal{T}) = \min_{t\in\mathcal{T}} C_{\min}(s \to t).$$

The streaming capacity of the network is then equal to $C_{\min-\min}(s \to V\backslash\{s\})$ [20]. Note that given the graph and the capacity of each edge, this streaming capacity can be achieved with simple transmission schemes, e.g., with network coding [21], [22] or with a latest-useful-chunk policy [7]. However, it may require global knowledge and centralized control in order to optimally construct the network graph and allocate the upload capacity. A natural question is then the following: without centralized control, can the streaming capacity over an incomplete network approach the optimal streaming capacity C_f of a complete network? In the next subsection we will provide simple and distributed peer selection and rate allocation algorithms that can achieve this with high probability when the network size is large.

B. Algorithms

We will now give explicit description of our simple control algorithms. First, we use a random peer selection algorithm. Specifically, each peer will randomly select M downstream neighbors uniformly from all other peers. On the other hand, the server will select M downstream neighbors uniformly and randomly among the ON peers. Second, we use a uniform rate allocation algorithm, i.e., each peer i simply divides its upload capacity equally among all its downstream neighbors in $\mathcal{E}_i$. Therefore, each peer in set $\mathcal{E}_i$ will receive a streaming rate U_i/M from peer i. Similarly, each downstream peer of the server receives U_s/M from the server. Under the above scheme, the link capacity C_{ij} is given by

$$C_{ij} = \begin{cases} U_i/M, & \text{if } j \in \mathcal{E}_i, i \neq s \\ U_s/M, & \text{if } j \in \mathcal{E}_s, i = s \\ 0, & \text{otherwise.} \end{cases}$$

Note that since $\mathcal{E}_i$ and $\mathcal{E}_s$ are chosen randomly, C_{ij}'s will also be random variables. We define another important parameter for the total capacity that each peer i receives from its direct upstream neighbors, which is given by $C_i^R = \sum_{j\in V} C_{ji}$. We will see that this value is the main factor that determines the streaming capacity from the source to each node.

Note that the above algorithm is a very simple mesh-based algorithm with the following advantages:

- Simplicity - The random peer selection and uniform rate allocation are easy to implement.
- Robustness - If some peer leaves the system, only the upstream neighbors of that peer need to re-select another downstream neighbor. It is not necessary to reconstruct the whole topology. Further, when a peer switches between ON and OFF, its set of downstream neighbors does not need to change.
- Low signaling overhead - Only the server needs to know which peers are ON. The tracker does not need to update the upload capacity of peers to any other peer.

Somewhat surprisingly, we will show that, as long as $M = \Omega(\log N)$, the algorithm will achieve close-to-optimal streaming capacity, with probability approaching 1 as $N \to \infty$ (Theorem 1).

Remark: Note that the server will only choose ON peers as its downstream neighbors. This is essential for achieving the close-to-optimal streaming capacity. To see this, note that the optimal streaming capacity C_f of a complete network is also constrained by the server capacity (see Equation (1)). If the server had used a substantial fraction of its upload capacity to serve OFF peers, intuitively the rest of the peers would then suffer a lower streaming rate. With the same intuition, one would think that the peers directly connected to the server also need to be careful in choosing their downstream neighbors. However, this turns out to be unnecessary. For our main result (Theorem 1) to hold, no other peers (except the server) are required to differentiate their downstream neighbors. As

readers will see, this is because those cuts with V_n only containing the downstream neighbors of s will play a small role in the overall probability of attaining the close-to-optimal streaming capacity.

We also note that the above algorithm uses the "push" model, where upstream peers choose downstream neighbors. An alternate model is the "pull" model, where downstream peers choose upstream neighbors. Note that both models create a mesh topology, and there is considerable symmetry between the two models. We use the push model in this paper because it is easier to analyze, although we believe that the main results of the paper can also be generalized to the pull model, which we leave as future work.

C. Main Result

Theorem 1. *For any $\epsilon \in (0,1)$ and $d > 1$, there exists α and N_0 such that for any $M = \alpha \log(N)$ and $N > N_0$, the probability for the min-min-cut under the algorithm in Section II-B to be smaller than $(1-\epsilon)C_f$ is bounded by*

$$\mathbf{P}\left(C_{\min-\min}(s \to V) \le (1-\epsilon)C_f\right) \le O\left(\frac{1}{N^{2d-1}}\right).$$

Recall that the min-min-cut is equal to the streaming rate to all peers. Hence, Theorem 1 shows that as long as the number of downstream neighbors M is $\Omega(\log N)$, for any $\epsilon \in (0,1)$, the streaming rate of our algorithm will be larger than $(1-\epsilon)$ times the optimal streaming capacity, with probability approaching 1 as the network size N increases.

D. Proof of Theorem 1

We first find the min-cut for any fixed peer t. We will use a similar approach as the one in [19]. We will show that the probability for the capacity of a cut to be smaller than $(1-\epsilon)$ times its mean is very small as N becomes large. Then, we will take the union bound over all cuts and show that overall probability is also very small. However, the techniques in [19] do not directly apply to our model due to the following two reasons. First, due to ON-OFF model, there are fewer "ON" peers and hence the probability for each cut to fall below its expected value will be larger than the case when all peers' upload capacity is the same. However, there are still the same number of cuts we need to account for, which may cause the union bound in [19] to diverge. Second, the link capacity C_{ij} in [19] is assumed to be independent across j, which is not the case in our model. To address the first difficulty, we will first consider the subgraph that only contains the ON users, and hence the number of cuts is also reduced correspondingly. To address the second difficulty, we will show that the joint distribution of C_{ij} can be approximated by i.i.d. random variables, which significantly simplifies the analysis.

We first introduce the following general relationship between the min-cut from server s to peer t in a random graph G and the min-cut from server s to peer t in any subgraph H_t of G that contains s and t.

Proposition 2. *Let G be a random graph defined on some probability space Ω that has a fixed source s and a fixed destination t. Let H_t be another random graph defined on the same probability space such that $H_t(\omega) \subseteq G(\omega)$ for all $\omega \in \Omega$ and H_t contains s and t. Then for any given positive value C, the following holds,*

$$\mathbf{P}\left(C_{\min,G}(s \to t) \le C\right) \le \mathbf{P}\left(C_{\min,H_t}(s \to t) \le C\right). \quad (2)$$

where $C_{\min,G}(s \to t)$ is the min-cut in G from s to t, and $C_{\min,H_t}(s \to t)$ is the min-cut in H_t from s to t.

Proposition 2 is intuitive because every cut in $G(\omega)$ has a larger capacity than the corresponding cut in the subgraph $H_t(\omega)$. The complete proof is available in [23] and is omitted here due to space constraints. For a given destination t, let $H_t(W,F)$ be the subgraph of $G(V,E)$ such that W contains peer t, the server and all of the nodes whose channel condition is ON, and $F \subset E$ contains those edges between nodes in W. The capacity of the edges in F is the same as the capacity of the edges in E. Proposition 2 allows us to focus on the subnetwork H_t instead of the entire network G. Assume that there are Y ON peers in the network excluding peer t, and thus $|W| = Y + 2$. Clearly, Y is a random variable with binomial distribution with parameter $N-1$ and p. For ease of exposition, we assume that Y is fixed during the following discussion for one given cut, and we will consider the randomness of Y later when we take the union bound over all cuts. We define a cut on H_t by dividing the peers in W into a set W_m of size $m+1$ that contains the server, and the complementary set W_m^c of size $Y-m+1$ that contains peer t. The capacity of cut D_m is then given by

$$D_m = \sum_{k \in W_m^c} C_{si} + \sum_{i \in W_m} \sum_{k \in W_m^c} C_{ik}.$$

Note that for each peer $i \in W_m$ (and $i \neq s$), we have $\sum_{k \in W_m^c} C_{ik} = L_i u/M$, where L_i is the number of downstream neighbors of peer i that are in the set W_m^c. Note that the value of L_i must satisfy $\max\{0, M-(N-Y+m-2)\} \le L_i \le \min\{M, Y-m+1\}$. Since downstream neighbors of peer i are uniformly chosen from other peers, we have,

$$\mathbf{P}\left(\sum_{k \in W_m^c} C_{ik} = l \cdot \frac{u}{M}\right) = \frac{\binom{Y-m+1}{l}\binom{N-Y+m-2}{M-l}}{\binom{N-1}{M}}.$$

This is the probability that l out of M downstream neighbors of peer i are in W_m^c (of size $Y-m+1$) and $M-l$ of them are in the set W_m. The distribution of L_i is known as a hypergeometric distribution with expectation $\frac{(Y-m+1)M}{N-1}$ [24, p167]. We can get a similar expression for the source s, i.e.,

$$\mathbf{P}\left(\sum_{i \in W_m^c} C_{si} = l \cdot \frac{u_s}{M}\right) = \begin{cases} \frac{\binom{Y-m}{l}\binom{m}{M-l}}{\binom{Y}{M}} & \text{if } t \text{ is OFF,} \\ \frac{\binom{Y-m+1}{l}\binom{m}{M-l}}{\binom{Y+1}{M}} & \text{if } t \text{ is ON.} \end{cases}$$

$$\mathbf{E}\left[\sum_{i \in W_m^c} C_{si}\right] = \begin{cases} \frac{u_s(Y-m)}{Y} & \text{if } t \text{ is OFF,} \\ \frac{u_s(Y+1-m)}{Y+1} & \text{if } t \text{ is ON.} \end{cases}.$$

Hence, we obtain the expectation of D_m as

$$\mathbf{E}\left[D_m\right] = \mathbf{E}\left[\sum_{k\in W_m^c} C_{si}\right] + \sum_{i\in W_m} \mathbf{E}\left[\sum_{k\in W_m^c} C_{ik}\right]$$
$$= \begin{cases} \frac{u_s(Y-m)}{Y} + \frac{u}{N-1}m(Y-m+1) & \text{if } t \text{ is OFF,} \\ \frac{u_s(Y+1-m)}{Y+1} + \frac{u}{N-1}m(Y-m+1) & \text{if } t \text{ is ON.} \end{cases} \quad (3)$$

Next, we are interested in the probability that $D_m \geq (1-\epsilon)\mathbf{E}[D_m]$ for all m for a given constant $\epsilon \in (0,1)$. In other words, this is the probability that the min-cut value is no less than $(1-\epsilon)$ times its average. For all m, it is not hard to see

$$\mathbf{E}[D_m] \geq \min\{\mathbf{E}[D_0], \mathbf{E}[D_Y]\} = \min\left\{u_s, \frac{Y}{N-1}u\right\}.$$

If we have $Y \geq (1-\epsilon)p(N-1)$, we will get

$$\mathbf{E}[D_m] \geq (1-\epsilon)\min\left\{u_s, \frac{u_s}{N} + pu\right\}.$$

Recall that $C_f = \min\{u_s, \frac{u_s}{N} + pu\}$ is the optimal streaming capacity assuming a complete network [6]. Hence, $D_m \geq (1-\epsilon)\mathbf{E}[D_m]$ will then imply that $D_m \geq (1-\epsilon)^2 C_f$. In other words, the probability that $D_m \geq (1-\epsilon)\mathbf{E}[D_m]$ for all m will become a lower bound for the probability that the min-cut is no less than $(1-\epsilon)^2 C_f$. In the following, we will derive $\mathbf{P}(D_m \geq (1-\epsilon)\mathbf{E}[D_m])$. We first find a bound on the moment generating function for D_m and take advantage of the Chernoff bound to obtain a good estimate of the above probability. Towards this end, we have the following Proposition.

Proposition 3. *For any given cut V_k and V_k^c of a network $G(V,E)$, let $\tilde{W}_1$ and $\tilde{W}_2$ be subsets of V_k and V_k^c, respectively. Assume that $|\tilde{W}_1| = q \leq k+1$ and $|\tilde{W}_2| = r \leq N-k$. Let the upload capacity of each peer $i \in \tilde{W}_1$ be u. For each peer in $\tilde{W}_1$, it chooses M downstream neighbors uniformly randomly from a given subset $\tilde{V}$ of V that is a superset of $\tilde{W}_2$. Let $\tilde{N} = |\tilde{V}|$. Then the moment generating function of $\sum_{i\in\tilde{W}_1}\sum_{j\in\tilde{W}_2} C_{ij}$ satisfy*

$$\mathbf{E}\left[e^{-\theta\sum_{i\in\tilde{W}_1}\sum_{j\in\tilde{W}_2} C_{ij}}\right] \leq \exp\left[Mq\frac{r}{\tilde{N}}\left(e^{-\theta\frac{u}{M}} - 1\right)\right]. \quad (4)$$

Note that under the setting of Proposition 3, each C_{ij} takes the value of u/M with probability $M/\tilde{N}$. However, there exists correlation among C_{ij} across different j's. Suppose another random variable $\tilde{C}_{ij}$ has the same marginal distribution as C_{ij} for every j, but $\tilde{C}_{ij}$ is independent across j. It can be easily verified that the moment generating function of $\sum_{i\in\tilde{W}_1}\sum_{j\in\tilde{W}_2}\tilde{C}_{ij}$ is equal to the right hand side of (4). Hence, Proposition 3 indicates that the correlation across C_{ij} only reduces the value of the moment generating function of the sum. This result holds because hyper-geometric random variables are known to be "negatively related" [25], and the moment generating function of the sum of negatively related random variables is always no larger than the moment generating function of the sum of *independent* random variables with the same marginal distribution [25]. The details of the proof are provided in [23].

Proposition 3 combined with the Chernoff bound will be frequently used to estimate the probability for a cut to "fail", i.e., when the capacity of a cut is less than $(1-\epsilon)$ times its expected capacity. We have the following result for cuts W_m in H_t under the assumption of ON-OFF upload capacity.

Lemma 4. *Let $\epsilon \in (0,1)$. Given that the total number of ON peers in the entire network $Y = y$, the probability that the capacity of the cut D_m is less than $(1-\epsilon)\mathbf{E}[D_m]$ can be bounded by the following,*

$$\mathbf{P}(D_m \leq (1-\epsilon)\mathbf{E}[D_m] | Y = y)$$
$$\leq \exp\left[-\left(Mm\frac{y-m+1}{N-1} + M\frac{y-m}{y}\right)\frac{u}{u_s}\frac{\epsilon^2}{2}\right].$$

Due to page limits we omit the proof of this Lemma, which is available in [23]. Lemma 4 gives us an upper bound on the probability that the capacity D_m of a cut W_m is less than $1-\epsilon$ times its mean, conditioned on the event that the total number of ON peers $Y = y$. Note that $Mm\frac{y-m+1}{N}$ is the average number of edges from peers in W_m to peers in W_m^c, while $M\frac{y-m}{y}$ is a lower bound on the average number of edges from the server to peers in W_m^c. Hence, the upper bound in Lemma 4 decreases exponentially if the average number of edges increases. Furthermore, since the average number of edges is proportional to M, the upper bound also decreases exponentially if M increases. The following lemma then bounds the effect of all cuts separating s and t. Note that for each value of m, there are $\binom{Y}{m}$ possible cuts W_m. Due to symmetry, the capacity of all cuts has the same distribution.

Lemma 5. *Define $\tilde{\mathcal{B}}_m$ to be the event $\{D_m \leq (1-\epsilon)C_f$ for any cut W_m among the $\binom{Y}{m}$ cuts $\}$. The probability of the union of all $\tilde{\mathcal{B}}_m$'s can be bounded by*

$$\mathbf{P}\left(\bigcup_{m=0}^{Y}\tilde{\mathcal{B}}_m\right) \leq O(\exp(-\epsilon'^2p^2N)) + \beta^\gamma\left[\left(1+p\beta^{\frac{\gamma}{2}}\right)^{N-1}\right]. \quad (5)$$

More specifically, we can separate the union bound into two parts:

$$\mathbf{P}\left(\bigcup_{m=0}^{Y-1}\tilde{\mathcal{B}}_m\right) \leq O(\exp(-\epsilon'^2p^2N)) \quad (6)$$
$$+ \beta^\gamma\left[\left(1+p\beta^{\frac{\gamma}{2}}\right)^{N-1} - 1\right], \quad (7)$$
$$\mathbf{P}\left(\tilde{\mathcal{B}}_Y\right) \leq O(\exp(-\epsilon'^2p^2N)) + \beta^\gamma. \quad (8)$$

where $\epsilon' = 1-\sqrt{1-\epsilon}$, $\gamma = (1-\epsilon')p$ and $\beta \triangleq \exp(-M\frac{u}{u_s}\frac{\epsilon'^2}{2})$.

The proof of Lemma 5 is available in [23]. One can then show that for any $\epsilon \in (0,1)$ and $\epsilon' = 1-\sqrt{1-\epsilon}$, we can choose a sufficiently large α such that $\epsilon' = \sqrt{\frac{4d}{\alpha\gamma}\frac{u_s}{u}}$, and $\beta^\gamma = O(1/N^{2d})$. Therefore, the probability for the min-cut of one source-destination pair to be smaller than $(1-\epsilon)C_f$ can be shown to be $O(1/N^{2d})$. We then have the result in Theorem 1. The detailed proof is available in [23].

We remark on several implications of Theorem 1. First, Theorem 1 not only shows that pure random selection is sufficient to achieve close-to-optimal streaming capacity as long as each peer has $\Omega(\log N)$ downstream neighbors, it also reveals important insights on the significance of different types of cuts. To see this, note that if we choose α as in the proof such that $\beta^\gamma = O(1/N^{2d})$, we have

$$\mathbf{P}\left(\bigcup_{m=0}^{Y-1}\tilde{\mathcal{B}}_m\right) \le 2\beta^\gamma\left[\left(1+p\beta^{\frac{\gamma}{2}}\right)^{N-1}-1\right]$$
$$=O(1/N^{2d})O(e^{1/N^{d-1}}-1) = o(1/N^{2d}).$$

On the other hand, we have $\mathbf{P}\left(\tilde{\mathcal{B}}_Y\right) = O(1/N^{2d})$. Hence, the probability that the last cut (the W_Y and W_Y^c cut) fails is much larger than the probability that any other cuts fails. Thus, for each peer t, the min-cut from the source to t is mainly determined by C_t^R (recall that C_t^R is the total capacity received by peer t from its direct upstream neighbors, which is also the capacity of the last cut).

The above insight suggests that, if we want to design improved distributed control algorithms for P2P streaming systems, we may want to focus on improving the capacity C_t^R at the last hop. Note that one of the main reasons for C_t^R to fall below its mean value is the imbalance of C_t^R across t. More specifically, some peers t may have a larger number of upstream peers, and hence have a larger-than-average value of C_t^R, while other peers may have a smaller-than-average value of C_t^R. Such imbalance will lead to an increase in the probability that some peers have low streaming rates. Based on this intuition, we can use the following slightly-modified algorithm. Suppose that a peer already receives enough capacity from its direct upstream neighbors (i.e., $C_t^R > C_f$), it is very likely that this peer will also have a min-cut from the source that is larger than C_f. We can then take away some upstream neighbors from this peer and allocate them to other peers. Intuitively, this modification will help to balance the values of C_t^R. Simulation results show that this "adaptive" algorithm indeed reduces the "failure" probability compared to the pure random algorithm with the same network size.

Theorem 1 also reveals important relationship between the number of neighbors required and key system parameters. For example, if we require a better performance (smaller ϵ or larger d) or have fewer ON peers (smaller p), the number of downstream neighbors needed by each peer will increase. Specifically, according to the proof, we need $\alpha \ge \frac{4du_s}{\gamma u\epsilon'^2}$. If we require higher streaming rate or faster convergence rate, i.e., ϵ is smaller (consequently ϵ' is smaller) or d is larger, we will need a larger α. If the probability that a peer is ON is reduced, i.e., p is reduced, we will also need a larger α.

III. Multi-Channel P2P Networks

In this section, we will extended our analysis to a multi-channel network containing J different channels. We are interested in the scenarios where the upload capacity from one channel can be used to "help" another channel. For single-channel networks, the streaming capacity of the network is a real number. However, for a multi-channel network, the streaming rate requirements of different channels can be different. Let R_j be the streaming rate requirement of channel j. There is clearly a tradeoff between the values of R_j in different channels, i.e., with finite upload capacity, increasing R_j for one channel j must be at the cost of reducing R_k of another channel k. To capture this tradeoff, we define the capacity region Λ as the set of streaming rate vectors $\mathbf{R} = [R_1, R_2, ..., R_J]^T$ such that whenever $\mathbf{R} \in \Lambda$, each user in the network will receive enough capacity to view its own channel of interest with high probability. Intuitively, if the upload capacity of the users and the server is the only bottleneck in the network, the best we can do is to support those rate vectors $\mathbf{R}$ such that the summation of all the demands is equal to all the supply. Hence, the largest possible capacity region will be no larger than

$$\Lambda_m = \left\{\mathbf{R}\left|\sum_{j=1}^{J}N_jR_j \le \sum_{i\in V}\mathbf{E}[U_i], \sum_{j=1}^{J}R_j \le u_s\right.\right\}, \quad (9)$$

where N_j is the number of peers that are viewing channel j. With this largest possible capacity region in mind, we are going to present our multi-channel algorithm and we will show that our algorithm could achieve the following close-to-optimal capacity region with high probability when $N \to \infty$:

$$(1-\epsilon)\Lambda_m = \{(1-\epsilon)\mathbf{R}|\mathbf{R} \in \Lambda_m\}.$$

A. System Model

We consider a multi-channel P2P networks with N peers, one source s and J different channels. We will reuse the same notations as in the single-channel section. Let $\mathcal{J} = \{1, 2, ..., J\}$ denote the set of all the channels. We have $|\mathcal{J}| = J$. Denote the set of all peers that are viewing channel j as $\mathcal{N}_j$ and we have $|\mathcal{N}_j| = N_j$. We assume that $N_j = p_jN$, where p_j is a fixed constant. We refer to the peers in $\mathcal{N}_j$ as the *normal peers* of channel j. Let the set of all peers (including the source) be V. Obviously, $V = \left(\bigcup_{j\in\mathcal{J}}\mathcal{N}_j\right) \cup \{s\}$ and $|V| = N+1 = \sum_{j\in\mathcal{J}}N_j+1$. Assume that the server allocates $u_{s,j}$ amount of capacity to channel j and $\sum_{j\in\mathcal{J}}u_{s,j} = u_s$. We also assume that each peer has an ON-OFF upload capacity, i.e., U_i is i.i.d. with the distribution $\mathbf{P}(U_i = u) = p$ and $\mathbf{P}(U_i = 0) = 1-p$. Each node (including the server) will still have M downstream neighbors. We will describe how these neighbors are chosen later.

In multi-channel P2P networks, the popularity N_j and streaming rate requirement R_j can vary from channel to channel. If we let the peers in the same channel to form a sub-network, for some channel j the streaming rate requirement R_j may exceed the maximum streaming capacity, while for other channels the upload capacity of the peers is more than needed. More specifically, for any $\epsilon \in (0, 1)$, let

$$\mathcal{I}_\epsilon = \{j \in \mathcal{J}|R_j > (1-\epsilon)up + u_{s,j}/N_j,\}$$
$$\mathcal{S}_\epsilon = \{j \in \mathcal{J}|R_j \le (1-\epsilon)up + u_{s,j}/N_j.\}$$

We call the channels in $\mathcal{I}_\epsilon$ insufficient channels, and the channels in $\mathcal{S}_\epsilon$ sufficient channels. With a multi-channel network, we can let some peers in the sufficient channels help another insufficient channel, and both channels may be able to achieve their own streaming rate requirements [16]. We call these peers that are helping the other channel j the "helpers" for channel j. A helper will not allocate its upload capacity to its own channel but will contribute to the channel it is helping. Let the number of helpers for channel j be H_j. For convenience, we allow H_j to be positive, negative or 0: For an insufficient channel j, $H_j > 0$, which denotes the number of helpers that are helping to distribute the content in channel j; for a sufficient channel j, $H_j < 0$, and $|H_j|$ is the number of helpers that channel j is providing to assist other channels. We emphasize that a helper from a sufficient channel j to help an insufficient channel k needs to receive its interested content of channel j at streaming rate R_j, as well as the content of channel k (at a smaller streaming rate than R_k) from some native peer in that channel.

B. Algorithm

We will now present the scheme for allocating server capacity to each channel and for choosing the number of helpers that each channel needs. Fix $\epsilon \in (0,1)$. In the following discussions, we assume that $\mathbf{R} \in (1-\epsilon)\Lambda_m$. For an insufficient channel j, denote the set of all helpers that are helping channel j as $\mathcal{H}_j$. Recall that each ON peer can have M downstream neighbors, which correspond to M links, each with capacity u/M. Each peer i in $\mathcal{N}_j$ will reserve K downstream links (out of a total of M links) to allow helpers to connect to peer i, where K is a constant such that the total reserved links KN_j is larger than the number of helpers H_j. Each helper i will choose one connection among KN_j reserved links uniformly randomly, and the owner of that reserved link becomes an upstream neighbor of that helper. In other words, each helper will try to find one and only one upstream neighbor in $\mathcal{N}_j$ and no peers support more than K downstream helpers. Each peer in $\mathcal{N}_j$ will choose $M-K$ downstream neighbors from $\mathcal{N}_j$ and each helper will also choose M downstream neighbors from $\mathcal{N}_j$. Note that there will be no connection between helpers, which avoids loops among helpers. Our multi-channel algorithms preserve the advantages of our single-channel algorithms. The random peer selection is simple, robust, and mesh-based.

The parameters are chosen as follows. The server will allocate the capacity for channel j as $u_{s,j} = R_j/(1-\epsilon)$. Each channel will determine the number of helpers it needs by the following equation

$$H_j = \left\lfloor \frac{N_j R_j}{(1-\epsilon)u} - \frac{u_{s,j}}{u} - pN_j \right\rfloor. \tag{10}$$

We require $\sum_{j\in\mathcal{J}} H_j \le 0$, i.e., the total number of helpers provided by the sufficient channels must be no smaller than the total number of helpers demanded by insufficient channels. We can check that (10) satisfies this condition (for details, please refer to [23]). Note that according to (10), a channel with higher streaming rate requirement will require more helpers. The constant K is chosen as $K = \max_{j\in J} \frac{R_j}{(1-\epsilon)u} - 1$. One can verify that for any insufficient channel j, we have $H_j \le KN_j$.

C. Performance Analysis

Next we will provide the analysis of the capacity region of our algorithm. We will show that as long as $M = \Omega(\log N)$, with high probability our algorithm can achieve the capacity region of $(1-\epsilon)\Lambda_m$, where Λ_m is the optimal capacity region given by (9). We start with a result on the performance bound of each channel, and then use that result to analyze the capacity region.

For each channel j, denote the set containing the server, the peers in $\mathcal{N}_j$ and the helpers in $\mathcal{H}_j$ as $\mathcal{V}_j$, i.e., $\mathcal{V}_j = \{s\} \cup \mathcal{N}_j \cup \mathcal{H}_j$. Let the subnetwork that contains all the nodes in $\mathcal{V}_j$ and the links between them as $\mathcal{G}_j$. We have on average pN_j native ON peers and H_j helpers that contribute their bandwidth for uploading. There are N_j peers that require the full streaming rate. Therefore, even under a complete-network assumption, the maximum streaming rate for each channel j will be

$$C_{f,j} \triangleq \min\left\{ u_{s,j}, \frac{pN_j + H_j}{N_j} u + \frac{u_{s,j}}{N_j} \right\}. \tag{11}$$

Similar to the single-channel P2P network, our algorithm could achieve a close-to-optimal streaming rate for each channel. The following theorem holds for each channel under our multi-channel algorithm.

Theorem 6. *Assume that p_j, R_j are given and $H_j = \Theta(N)$ for large N. In addition, assume that for any j such that $H_j < 0$, there exists $\eta < 1$ such that $|H_j| \le \eta p N_j$. Then for any channel j, any $\epsilon \in (0,1)$ and $d > 1$, there exist α_j and $N_{0,j}$ such that for any $M = \alpha_j log(N)$ and $N > N_{0,j}$ the probability for the min-min cut of channel j to be smaller than or equal to $(1-\epsilon)C_{f,j}$ is bounded by*

$$\mathbf{P}\left(C_{\min-\min}(s \to \mathcal{N}_j) \le (1-\epsilon)C_{f,j}\right) \le O\left(\frac{1}{N^{2d-1}}\right).$$

Due to space constraints, we omit the proof of this theorem, which is available in [23]. Theorem 6 provides a performance bound for each channel. The result is similar in flavor to the single-channel case. The choice of α_j is also very similar, i.e., the higher the streaming rate is (smaller ϵ) and the smaller the ON probability p is, the larger α_j is required to achieve faster convergence rate (larger d). Note that if $N_j = p_j N$, under the assumption of ON-OFF upload capacity, the set of Λ_m in (9) can be written as

$$\Lambda_m = \left\{ \mathbf{R} \,\middle|\, \sum_{j=1}^{J} p_j R_j \le up, \sum_{j=1}^{J} R_j \le u_s \right\},$$

which is independent of N. We will adopt this definition of Λ_m in the rest of the paper. Assume that $\mathbf{R} \in (1-\epsilon)\Lambda_m$. With the choice of H_j in (10), we can show that for any $j \in \mathcal{J}$,

$(1-\epsilon)C_{f,j} \geq R_j$. We can then conclude that

$$\begin{aligned}&\mathbf{P}\left(C_{\min-\min}(s\to\mathcal{N}_j)\leq R_j\right)\\ \leq&\mathbf{P}\left(C_{\min-\min}(s\to\mathcal{N}_j)\leq (1-\epsilon)C_{f,j}\right)\leq O\left(\frac{1}{N^{2d-1}}\right).\end{aligned}$$

Theorem 7 summarizes the final result on the capacity region of our algorithm.

Theorem 7. *For any $\epsilon\in(0,1)$, $d>1$ and $\mathbf{R}\in(1-\epsilon)\Lambda_m$, choose H_j as (10) for $j=1,2,...,J$. There exist α and N_0 such that for any $M=\alpha log(N)$ and $N>N_0$, the following holds*

$$\mathbf{P}\left(C_{\min-\min}(s\to\mathcal{N}_j)\leq R_j,\ \textit{for some } j\right)\leq O\left(\frac{1}{N^{2d-1}}\right).$$

The proof of this theorem is available in [23]. We see that $\Omega(\log N)$ neighbors are again sufficient for achieving a close-to-optimal streaming capacity with high probability when $N\to\infty$.

IV. Simulation

In this section we provide simulation results to verify our analytical results in previous sections. We first simulate a single-channel P2P network with $N=10000$ peers and one server. Each user has a ON-OFF upload capacity with ON probability p. When a user is ON, it will contribute an upload capacity $u=10$. The server has a capacity of $u_s=20$. The optimal streaming capacity would be $C_f=5.002$. We vary the number of downstream neighbors of each user from $10\log N=90$ to $80\log N=720$, which correspond to 0.9% and 7.2% of the total number of peers N. For each choice of the number of downstream neighbors, we generate the network for 200 times. During each iteration all users select their downstream neighbors randomly as described in section II-B, and we use the algorithm in [26] to find the min-min cut from the source to all the users and compare it with $(1-\epsilon)C_f$. We count the number of times that the min-min cut of the network is larger than $(1-\epsilon)C_f$ and plot the probability for that to happen as the number of downstream neighbors of each peer varies. We also simulate the adaptive algorithm described at the end of section II-D where each peer only selects those peers who have not received enough capacity as its downstream neighbors. The result is shown in Fig. 2, where we simulate 4 different combinations of p and ϵ. We can observe that, using pure random selection, when $p=0.5$ and when the number of downstream neighbors of each peer is more than $40\log N=360$ (3.6% of N), the success probability that the system could sustain a streaming rate higher than 70% of the optimal streaming capacity is greater than 0.9. If $p=0.9$, the number of downstream neighbors needed by each peer to achieve the same success probability of 0.9 reduces to $30\log N=270$ (2.7% of N). Further, we can observe that at the same ON probability, when we increase ϵ, the required number of downstream neighbors to achieve the same success probability of 0.9 decreases. These observations verify our remarks following Theorem 1 that M needs to be larger if ϵ is smaller or p is smaller. For the adaptive algorithm, the same performance is achieved when each peer only has $30\log N=270$ (2.7% of N) downstream neighbors. We see that by improving the capacity of the last cut, the performance of the system is also improved. We caution, however, that when the peer selection strategy is significantly different from the baseline random algorithm in Section II-B, Theorem 1 will no longer apply. Hence, it remains an open question as to how to design hybrid schemes that adaptively improve the capacity in the last hop, while retaining the robustness of a random peer selection scheme at the same time. We leave this question for future work.

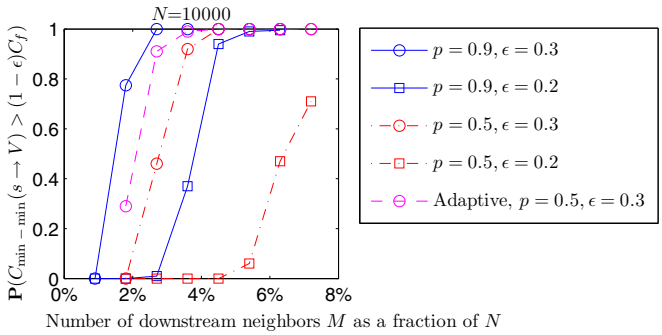

Fig. 2: Single-Channel: The Success Probability versus The Number of Downstream Neighbors

Next we simulate a multi-channel P2P network with $N=10000$ peers and 2 channels. We use the same setting as the single-channel simulation for the upload capacity for all ON peers, the capacity of the server, and the probability for a peer to be ON. We set $N_1=4000$ and $N_2=6000$. We choose a streaming rate vector $\tilde{\mathbf{R}}=[35/6,25/6]^T$ in Λ_m and let our target streaming rate vector be $\mathbf{R}=0.7\tilde{\mathbf{R}}$ (i.e., $\epsilon=0.3$). Channel 1 will become an insufficient channel and channel 2 will be a sufficient channel. For channel 1, we need 500 helpers and channel 2 could provide 500 helpers. In this case, each normal peer in channel 1 needs to reserve 1 link for helpers. We plot the probability that the streaming rate of each channel j is greater than its target streaming rates R_j and the probability that both channels 1 and 2 sustain a streaming rate greater than their corresponding target streaming rate R_1 and R_2, respectively, as the number of downstream neighbors of each peer varies. The result is shown in Fig. 3. We see that the performance of sufficient channel is worse than the insufficient channel. The reason is that there is only 6000 peers in the sufficient channel, and on average there are 2500 ON peers. However, 500 of the ON peers are helping the insufficient channel. Hence, there are only 2500 ON peers left in the sufficient channel, which is equivalent to having an ON probability of 5/12. Hence, the network size and the ON probability of the sufficient channel are both smaller. We will then need a larger number of downstream neighbors to achieve the same success probability.

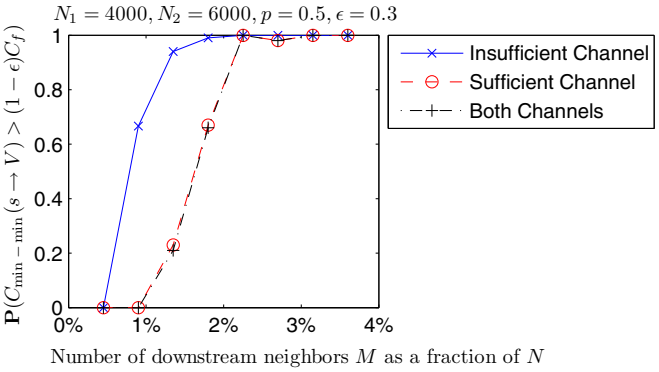

Fig. 3: Multi-Channel: The Success Probability versus The Number of Downstream Neighbors

V. Conclusion

In this paper, we study the streaming capacity of sparsely-connected P2P networks. We show that even with a random peer selection algorithm and uniform rate allocation, as long as each peer maintains $\Omega(\log N)$ downstream neighbors, the system can achieve close-to-optimal streaming capacity with high probability when the network size is large. We then extend our analysis to multi-channel P2P networks, and we let "helpers" from channels with excessive upload capacity to help peers in channels with insufficient upload capacity. We show again that we can achieve a close-to-optimal streaming capacity region by letting each peer uniformly randomly select $\Omega(\log N)$ neighbors from either the peers in the same channel or from the helpers.

These results provide important new insights on the streaming capacity of large P2P networks with sparse topology. In future work, we plan to study how to improve the peer selection and rate allocation algorithms to further optimize the streaming capacity. We note that although our analytical results show that having $\Omega(\log N)$ neighbors is sufficient to achieve close-to-optimal streaming capacity with high probability, our simulation results indicate that the actual number of peers required can still be fairly large. A natural next step is to improve the constant in the $\Omega(\log N)$ result while still retaining the simplicity and robustness of a random selection scheme. Our analysis provides an important insight that the capacity of the last cut (i.e., the capacity from direct upstream neighbors) is often the bottleneck. Our simulation results demonstrate that, by slightly modifying the control at the last hop, the performance of the system can indeed be improved. We envision that hybrid schemes, that both balance the capacity at the last hop and exploit some level of random peer selection, may be able to achieve the best tradeoff between performance and complexity.

Acknowledgments: This work has been partially supported by the National Science Foundation through awards CNS-0643145, CNS-0721484, CNS-0721477 and CNS-0813000, and the Research Grants Council of Hong Kong (RGC GRF Ref: HKU 718710E).

References

[1] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceedings IEEE INFOCOM*, vol. 3, 2005, pp. 2102 – 2111.

[2] 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, Dec. 2007.

[3] T. Silerston and O. Fourmaux, "Measuring P2P IPTV systems," in *Proceedings of NOSSDAV'07*, June 2007.

[4] C. Wu, B. Li, and S. Zhao, "Exploring large-scale peer-to-peer live streaming topologies," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 4, no. 3, pp. 1–23, 2008.

[5] W. Liang, J. Bi, R. Wu, Z. Li, and C. Li, "On characterizing PPStream: Measurement and analysis of P2P IPTV under large-scale broadcasting," in *IEEE GLOBECOM 2009*, Nov. 2009, pp. 1 –6.

[6] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *IEEE INFOCOM*, May 2007, pp. 919 –927.

[7] L. Massoulié and A. Twigg, "Rate-optimal schemes for peer-to-peer live streaming," *Perform. Eval.*, vol. 65, no. 11-12, pp. 804–822, 2008.

[8] C. Feng and B. Li, "On large-scale peer-to-peer streaming systems with network coding," in *Proceeding of the 16th ACM International Conference on Multimedia*, Vancouver, British Columbia, Canada, 2008.

[9] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs," in *ACM SIGMETRICS '08*, 2008, pp. 325–336.

[10] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *ACM SIGMETRICS '08*, 2008, pp. 313–324.

[11] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Streaming capacity in peer-to-peer networks with topology constraints," *submitted to IEEE Transactions on Information Theory*, 2009.

[12] S. Liu, M. Chen, S. Sengputa, M. Chiang, J. Li, and P. A. Chou, "P2P streaming capacity under node degree bound," in *Proc. IEEE ICDCS*, Genoa, Italy, June 2010.

[13] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments," in *SOSP '03*, Bolton Landing, NY, 2003, pp. 298–313.

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

[15] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-upload decoupling: A redesign of multi-channel P2P video systems," in *IEEE INFOCOM*, 2009, pp. 2726 –2730.

[16] D. Wu, Y. Liu, and K. Ross, "Queuing network models for multi-channel P2P live streaming systems," in *IEEE INFOCOM*, 2009, pp. 73 –81.

[17] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388 –404, Mar 2000.

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

[19] A. Ramamoorthy, J. Shi, and R. Wesel, "On the capacity of network coding for random networks," *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2878 –2885, Aug. 2005.

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

[21] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204 –1216, Jul 2000.

[22] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371 –381, Feb. 2003.

[23] C. Zhao and X. Lin, "The streaming capacity of sparsely-connected P2P system with low complexity algorithm," Purdue University, Tech. Rep., 2010. [Online]. Available: http://cobweb.ecn.purdue.edu/%7elinx/papers.html

[24] S. Ross, *A first course in probability*, 5th ed. Upper Saddle River, NJ: Prentice-Hall, 1998.

[25] A. D. Barbour, L. Holst, and S. Janson, *Poisson Approximation*. Oxford University Press, 1992.

[26] J. Hao and J. B. Orlin, "A faster algorithm for finding the minimum cut in a graph," in *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, Orlando, Florida, 1992.