Capacity of P2P On-Demand Streaming with Simple, Robust and Decentralized Control

Can Zhao*, Jian Zhao†, Xiaojun Lin*, Chuan Wu†
*School of Electrical and Computer Engineering, Purdue University, West Lafayette
email: {zhao43, linx}@purdue.edu
†Department of Computer Science, The University of Hong Kong, Hong Kong
email: {jzhao, cwu}@cs.hku.hk

***Abstract*—The performance of large-scaled peer-to-peer (P2P) video-on-demand (VoD) streaming systems can be very challenging to analyze. In practical P2P VoD systems, each peer only interacts with a small number of other peers/neighbors. Further, its upload capacity may vary randomly, and both its downloading position and content availability change dynamically. In this paper, we rigorously study the achievable streaming capacity of large-scale P2P VoD systems with sparse connectivity among peers, and investigate simple and decentralized P2P control strategies that can provably achieve close-to-optimal streaming capacity. We first focus on a single streaming channel. We show that a close-to-optimal streaming rate can be asymptotically achieved for all peers with high probability as the number of peers N increases, by assigning each peer a random set of $\Theta(\log N)$ neighbors and using a uniform rate-allocation algorithm. Further, the tracker does not need to obtain detailed knowledge of which chunks each peer caches, and hence incurs low overhead. We then study multiple streaming channels where peers watching one channel may help in another channel with insufficient upload bandwidth. We propose a simple random cache-placement strategy, and show that a close-to-optimal streaming capacity region for all channels can be attained with high probability, again with only $\Theta(\log N)$ per-peer neighbors. These results provide important insights into the dynamics of large-scale P2P VoD systems, which will be useful for guiding the design of improved P2P control protocols.**

I. Introduction

Peer-to-Peer (P2P) Video-On-Demand (VoD) streaming systems have already become a major player on today's Internet. Their success (e.g., PPLive, TVAnts, UUSee, and Zattoo) has made high-quality on-demand streaming of rich contents available to millions of users at low server costs [1]. In contrast to their commercial success, however, in-depth theoretical understanding of these systems appears to be lacking. The performance of large-scaled P2P VoD systems can be extremely complex to study. As time progresses, the part of the video that a peer is interested in viewing, the cached content that it can use to serve others, and its upload capacity can all change substantially. Further, these systems are highly decentralized in nature, and each peer often only has a very limited view of the overall system through its sparsely-connected neighbors. Due to these reasons, it remains a challenging problem to understand the fundamental performance limits of highly dynamic and decentralized P2P VoD systems.

In this paper, we study a problem of fundamental interest to P2P VoD systems, i.e., what is the optimal streaming rate that all peers can reliably receive, and how to achieve this optimal rate with simple, robust and decentralized control. Note that a trivial upper-bound on the streaming rate can be obtained by dividing the total upload capacity of all peers by the total number of peers. In P2P live-streaming systems, it has been shown in our prior work that streaming rates close to this optimal value can be achieved through simple and decentralized control [2]. However, in P2P VoD system, it is unclear whether such an optimal rate can still be attained. In contrast to live-streaming [2]–[10], each peer in a VoD system is interested in playing a different portion of the video. Further, its viewing position may jump back and forth [11], [12]. As a result, the content availability at each peer can be highly discontinuous and dynamic. One way to alleviate this difficulty is to assume that some peers (referred to as "caches") have cached the entire video beforehand, and other downloading peers request the content only from the caches. In [13]–[15], the authors have studied the optimal cache-placement problem based on this assumption. An implicit assumption along this line of work is that there exists a central entity that can perfectly balance the downloading requests among caches. Otherwise, such a global balancing problem by itself can be very challenging in a decentralized setting when the upload capacity of the peers varies.

An alternate (and perhaps practically more relevant) approach is to directly model how peers downloading the same video can use their upload capacity to help each other, which is unfortunately more difficult. Such models were proposed in, e.g., [12], [16], [17]. However, it appears difficult to establish whether they can achieve close-to-optimal streaming rates. More recently, [18] proposes an algorithm that allocates the overall upload capacity in the system sequentially from the "oldest" peer to the "youngest" peer. For each peer, its requested capacity is first allocated from older peers. If there is no sufficient upload capacity, capacity is then requested from the server. Similarly, [19] proposes a global optimization problem for rate-allocation given the age of the peers. While these algorithms have been found to exhibit good performance, the resulting rate-allocation may need to be completely recalculated when the peers' upload capacity changes. Further, these analyses have not accounted for the possibility that the peers' playback positions may jump back and forth, in which case

even an older peer may not have the content to serve younger peers.

In summary, existing analytical studies of the streaming capacity of P2P VoD systems either require extensive centralized control, are sensitive to upload-capacity variations or do not account for the random-seek behavior of the peers. In contrast, in this paper we provide the first rigorous study of the streaming capacity of large-scale P2P VoD systems with simple decentralized control that are robust to upload-capacity variations, and random-seek behaviors. We focus on the setting of "hot" videos, i.e., there are a large number of peers interested in viewing each video. We first study a single-channel system, i.e., all users are interested in viewing the same video. Assuming that the contribution of bandwidth and cache capacities from the dedicated server(s) is minimal, we show that by using a (properly-designed) random neighbor-selection algorithm and a uniform rate-allocation algorithm, with probability approaching 1 as the total number of peers N increases, all peers can achieve a close-to-optimal streaming rate of $(1-\epsilon)\mu$, where μ is the average upload capacity per peer and ϵ is a small positive constant. In our algorithm, each peer is only assigned $\Theta(\log N)$ upstream neighbors, with which they exchange content-availability information. These neighbors are chosen uniformly randomly from a suitable *choice set* determined at the tracker (note that this is the only part of the algorithm that requires centralized knowledge). To determine the choice set, the tracker only needs to know the current downloading position of each peer, but does not need to know the detailed content/chunk availability at each peer. Further, regardless of the variation of its upload-capacity, each peer evenly distributes its upload capacity among downstream neighbors for whom it has the available chunk(s). As readers will see in Section II, our analytical studies provide key insights as to why these simple design principles can result in near-optimal performance, which was conjectured in some prior simulation-based studies [20]. Further, these insights reveal the critical and non-trivial roles that different design choices, e.g., the size of the choice set and the extent of content availability, play in the overall system.

We then turn to a multi-channel P2P VoD system where different groups of peers are interested in viewing different videos. Based on the single-channel control algorithm discussed earlier, we propose a cache-placement algorithm that can achieve (with high probability) a close-to-optimal streaming rate region for all channels (see Section III for the precise definition). Our cache placement policy shares some similarity to the "proportional-to-deficit-bandwidth" strategy in [18], which was conjectured to be close-to-optimal. However, our policy does not require a sequential rate-allocation algorithm as in [18].

Our results have a similar flavor to the results in our earlier work [2] for P2P live-streaming systems. However, as we discussed earlier and will elaborate further in Section II, P2P VoD systems are significantly different from live-streaming systems in many aspects. Thus, new control algorithms and analytic techniques are required. To the best of our knowledge, this work provides the first analytic result that demonstrates how to achieve close-to-optimal streaming capacity in large-scale P2P VoD systems using simple, robust, and decentralized control.

II. A Single-Channel P2P VoD System

In this section, we focus on a system with a single channel, i.e., all users are interested in viewing the same video. We first describe the system model. We will then propose simple, robust and decentralized peer selection and rate allocation algorithms that result in at most $\Theta(\log N)$ upstream neighbors per peer. We then prove that all peers can achieve the close-to-optimal streaming rate with high probability, when N is large.

A. System Model

We consider a P2P VoD system where users/peers[1] would like to watch a common video. Let $T^{(0)}$ denote the length of the video. There is a server S and totally N peers. Let $\mathcal{N}$ denote the set of all peers in the system, i.e., $|\mathcal{N}| = N$. We assume that the number of peers N is fixed. In other words, if a peer leaves the system, a new peer is assumed to immediately join the system at a possibly random initial position. This assumption simplifies the analysis, while we believe that the insights under this assumption will also hold for a more dynamic model where peers randomly join and leave the system. In a VoD system, the viewing/downloading progress of different peers in the same channel is typically different. Peers who have already downloaded certain parts of the video can then serve the cached content to later peers. We define the downloading position of a peer as the immediately next position in the video that the peer will download. We assume that, the downloading position of each peer is *i.i.d.* according to a distribution with density function $\gamma(t)$. In other words, for a small δt, $\gamma(t)\delta t$ is the probability that the downloading position of a peer is between t and $t+\delta t$. Note that the downloading position of a peer is typically larger than its viewing position, with some buffering in between to absorb any fluctuations in the downloading speed. Some peers who have finished watching a channel may stay for some period of time and serve other peers in the channel. We thus allow $\gamma(t)$ to have a Dirac delta function at point $T^{(0)}$. Equivalently, let $\bar{Q}$ denote the probability that a peer's downloading position is $T^{(0)}$. For ease of exposition, we assume that, with probability 1, the downloading position of each peer before $T^{(0)}$ is different from that of other peers. From now on, we will index a peer watching a channel by its downloading position t. Let $\mathcal{N}^-$ denote the set of all peers with downloading position $t < T^{(0)}$.

To model how peers serve other peers, each peer t has a set of downstream neighbors $\mathcal{D}_t$ that this peer t may upload content to. Correspondingly, each peer $t \in \mathcal{N}^-$ also has a set of upstream neighbors $\mathcal{U}_t = \{s \in \mathcal{N} | t \in \mathcal{D}_s\}$ that this peer t can potentially download the content from. However, since peer may perform random seeks, it may not have all

[1] We use the terms "user" and "peer" interchangeably throughout the rest of the paper.

the content "before" its downloading position. Hence, not all neighbors in the upstream neighbor set $\mathcal{U}_t$ of peer t have the requested content of peer t. We denote $\underline{\mathcal{U}}_t \subset \mathcal{U}_t$ as the set of upstream neighbors of peer t who have the data that peer t is requesting and is willing to serve peer t. Correspondingly, let $\underline{\mathcal{D}}_t = \{s | t \in \underline{\mathcal{U}}_s\} \subset \mathcal{D}_t$ denote the set of downstream neighbors that peer t can actually serve. We call $\underline{\mathcal{D}}_t$ and $\underline{\mathcal{U}}_t$ the *effective* downstream neighbors and the *effective* upstream neighbors, respectively. Let $U_t = |\mathcal{U}_t|$, $D_t = |\mathcal{D}_t|$, $\underline{D}_t = |\underline{\mathcal{D}}_t|$ and $\underline{U}_t = |\underline{\mathcal{U}}_t|$.[2]

Let V_t denote the upload capacity of peer t. We assume that V_t is a bounded random variable between $[0, V_{\max}]$ with mean value μ, which is *i.i.d.* across all peers. Like other work [2]–[4], [8], [9], we assume that the download capacity and the core network capacity are sufficiently large, and hence the upload capacity is the only resource bottleneck. The system performance is determined by the relationship between the targeted streaming rate and the downloading rates. Let R denote the targeted streaming rate of the video. Let $C_{s\to t}$ denote the streaming rate from peer s to peer t. Clearly, $C_{s\to t} = 0$ for any $s \notin \underline{\mathcal{U}}_t$ (or equivalently for all $t \notin \underline{\mathcal{D}}_s$). We have the following upload capacity constraint on each peer s:

$$\sum_{t\in\mathcal{N}} C_{s\to t} = \sum_{t\in\underline{\mathcal{D}}_s} C_{s\to t} \le V_s.$$

Let C_t denote the achievable downloading rate for peer t, which is then given by:

$$C_t = \sum_{s\in\mathcal{N}} C_{s\to t} = \sum_{s\in\underline{\mathcal{U}}_t} C_{s\to t}.$$

To guarantee smooth playback, the downloading rate of each viewing peer must be no smaller than the targeted rate R of the video. Note that the peers whose downloading position is $T^{(0)}$ do not need to download new data, and hence we are only interested in the downloading rate of those peers in $\mathcal{N}^-$. We thus define the *streaming capacity* of the system as the largest value of R such that $C_t \ge R$ for all peers $t \in \mathcal{N}^-$.

We note that there is a simple upper bound on the streaming capacity. We assume that $\bar{Q}$ is away from 0 even with large N, and the contribution of the server capacity is negligible. In this case, it is easy to see that the largest possible streaming rate that all peers can attain is $\frac{N\mu}{N(1-\bar{Q})} = \frac{\mu}{1-\bar{Q}}$ on average. However, this upper bound completely ignores the details of the VoD system, especially whether a peer has the content and the upload capacity to help the other peer. Hence, it is unclear whether this upper bound is attainable in a large and decentralized VoD system. In practice, $\bar{Q}$ is usually not very large. Hence, in the rest of this section, we will omit the contribution of $\bar{Q}$ in the streaming capacity, and we will say that the channel achieves a close-to-optimal streaming capacity $(1-\epsilon)\mu$ with a small $\epsilon > 0$ if all peers attain a streaming rate no smaller than $(1-\epsilon)\mu$. Our goal in this section is to design simple, robust and decentralized algorithms that can achieve this close-optimal streaming capacity with high probability.

[2]As a convention, we will use script variable to denote a set (e.g., $\mathcal{U}_t$), and use a normal variable to denote its size (e.g., U_t).

B. A Simple and Distributed Peer Selection and Rate Allocation Algorithm

In our prior work for P2P live-streaming systems [2], we proposed a simple peer selection strategy where each peer uniformly randomly selects $\Theta(\log N)$ downstream neighbors, and divides its upload capacity evenly among its downstream neighbors. This simple algorithm has been shown to achieve a close-to-optimal streaming rate for live-streaming P2P systems. Although this result serves as a useful starting point, as reader will see below, the same design would have led to very poor performance in VoD systems. Thus, we need to design a new set of control algorithms tailored to VoD systems.

(i) Peer Selection: We first explain why a uniformly-random peer-selection algorithm will not work well for VoD systems. Note that unlike live-streaming systems, in a VoD system different peers are viewing different parts of the video, and their cached content is also different. If an older peer (whose downloading position is in the later part of the video) chooses a younger peer (whose downloading position is in the earlier part of the video) as an upstream neighbor, there is a high chance that the younger peer does not have the content to help the older peer. Hence, the connection between them is of no use. This problem will be the most severe for the oldest peers that are close to the end of the video. With uniformly-random peer selection, the peers who are interested in downloading this part of the video will find that most of their selected upstream neighbors are younger and do not have the desired content. Hence, the streaming rate to these oldest peers will be very poor. Hence, we need to design a new peer selection strategy for VoD P2P systems.

A key idea of our new strategy is to restrict the random neighbor selection of each peer t to be done within a *choice set* $\bar{\mathcal{U}}_t$, which contains peers with downloading positions larger than t. More specifically, we use the "random sequential choice-set selection strategy" as follows. Let Q be a constant such that $0 < Q < \bar{Q}$. In this strategy, the choice set $\bar{\mathcal{U}}_t$ of peer $t \in \mathcal{N}^-$ consists of the next NQ peers whose downloading positions are immediately larger than t's. If there are less than NQ peers after t and immediately before $T^{(0)}$, $\bar{\mathcal{U}}_t$ will be the set of all peers with downloading positions larger than t. In practice, the tracker can order all the peers according to their downloading positions and assign choice sets according to the above strategy. Recall our assumption that no two peers before $T^{(0)}$ are at the same downloading position. In practice, if this assumption does not hold, the tracker can always break ties arbitrarily. Then, the tracker server picks $M = \alpha \log N$ (where α is a positive constant to be determined later) peers uniformly randomly from peer t's choice set $\bar{\mathcal{U}}_t$, which constitute peer t's set of upstream neighbors $\mathcal{U}_t$. We have $\mathcal{U}_t \subset \bar{\mathcal{U}}_t$. Correspondingly, define the client set of peer t as $\bar{\mathcal{D}}_t = \{s \in \mathcal{N}^- | t \in \bar{\mathcal{U}}_s\}$. The set $\mathcal{D}_t$ of downstream neighbors of t must come from this client set and is given by $\mathcal{D}_t = \{s \in \mathcal{N}^- | t \in \mathcal{U}_s\}$. Let $\bar{U}_t = |\bar{\mathcal{U}}_t|$ and $\bar{D}_t = |\bar{\mathcal{D}}_t|$.

Remark: It appears that the tracker must maintain the current downloading position of all peers, which may incur high

overhead. However, as we will explain later, by enforcing that all peers advance their downloading position at the same speed, this overhead can be significantly reduced.

(ii) Content Availability: Even with the above peer-selection strategy, the streaming rate for some peer can still be very poor. This is because peers may fast-forward/backward in a VoD system. This discontinuous random-seek behavior means that a peer t may not always have all the content before t. Thus, even if a peer only picks an older peer as an upstream neighbor, the connection and the capacity may still be wasted. Unfortunately, the random-seek behavior of peers is quite complicated to model. To the best of our knowledge, no existing analytical works on P2P VoD systems are able to take into account the impact of this random-seek behavior.

Our strategy is to develop a condition for content availability that is sufficient for achieving close-to-optimal streaming rates, yet easy to satisfy even with random-seeks. This is perhaps the most difficult part of our design. To see why such a condition is non-trivial to formulate, consider the following scenario. Suppose that the NQ peers in the choice set $\bar{\mathcal{U}}_s$ of peer s are uniformly in the range $(s, s+\Delta)$. Let $t_0 = s + \Delta/2$. Each peer $t \in (s, t_0)$ has all the content before t. However, each peer $t \in (t_0, s+\Delta)$ only has the content in (t_0, t), possibly because it random-sought to t_0 before. Further, suppose that we use our peer-selection strategy described earlier, and each peer uniformly divides its constant upload capacity μ among its effective downstream neighbors. Then, peer s has M upstream neighbors uniformly in $(s, s+\Delta)$. However, only those peers in (s, t_0) can help peer s, each of which has on average M effective downstream neighbors. Hence, the average streaming rate of peer s is only $\frac{\mu}{M}\frac{M}{2} = \frac{\mu}{2}$, which is far from optimal. Clearly, the key difficulty here is that, due to its particular position, compared to other downstream neighbors, peer s has a much smaller probability to become an effective downstream neighbor of upstream peers in $(t_0, s+\Delta)$.

Our condition below addresses this difficulty. Fix a positive constant $q_{\min} \in (0,1)$. We require that, for any peer s and any one of its upstream neighbor t, the probability that peer t has the content for (and is willing to help) peer s is equal to $q_t > q_{\min}$, *independently of the position of peer* t. This content availability condition can be implemented as follows. Choose $q'_{\min}$ such that $(1 - e^{-q'_{\min}})/2 = q_{\min}$. Suppose that a peer (denoted by t) randomly seeks to position $t = t_0$ first. It will first download a fraction of the content from the range that may be requested by the peers in its client set. More specifically, let $\psi(t_0)$ be the downloading position of the youngest peers in this peer's client set $\bar{\mathcal{D}}_{t_0}$. This peer then selects K intervals within $[\psi(t_0), t_0]$, each of which has a length of $q'_t(t_0 - \psi(t_0))/K$, where $q'_t \geq q'_{\min} > 0$ satisfies $1 - \left(1 - \frac{q'_t}{K}\right)^K = q_t$. These K intervals are selected independently and uniformly randomly. At this point, it is easy to see that the above content-availability condition holds: for any peer s in $[\psi(t_0), t_0]$, the probability that peer $t = t_0$ has the required content for peer s is equal to the probability that peer s is in at least one of the K intervals, which is calculated as $1 - \left(1 - \frac{q'_t}{K}\right)^K = q_t$. For sufficiently large K, we will have $q_t \geq (1 - e^{-q'_{\min}})/2 = q_{\min}$. Next, as peer t continues to watch the video, it downloads the content from t_0 to its current downloading position $t > t_0$. In order to meet the content availability condition for all peers in the client set $\bar{\mathcal{D}}_t$, as long as $\bar{\mathcal{D}}_t$ contains at least one peer s whose downloading position is smaller than t_0, then for all other peers in $\bar{\mathcal{D}}_t \cap (t_0, t)$, peer t is only willing to serve it with probability q_t, independently of other peers. This restriction will continue until all peers $s \in \bar{\mathcal{D}}_t$ advance past t_0. Then, peer t can serve all of its downstream neighbors (equivalently, $q_t = 1$). As we will see later, this condition will be sufficient to achieve a close-to-optimal streaming rate.

TABLE I
RELATIONSHIP BETWEEN $\bar{\mathcal{D}}_t$, $\hat{\mathcal{D}}_t$, $\mathcal{D}_t$ AND $\underline{\mathcal{D}}_t$. THE RELATIONSHIP BETWEEN $\bar{\mathcal{U}}_t, \hat{\mathcal{U}}_t, \mathcal{U}_t$ AND $\underline{\mathcal{U}}_t$ ARE SIMILAR.

$\bar{\mathcal{D}}_t$	client set of peer t (containing roughly NQ peers)
$\hat{\mathcal{D}}_t$	a subset of $\bar{\mathcal{D}}_t$ that peer t has the requested content and is willing to serve
$\mathcal{D}_t$	a subset of $\bar{\mathcal{D}}_t$ with size M that are actual downstream neighbors of peer t
$\underline{\mathcal{D}}_t$	intersection of $\hat{\mathcal{D}}_t$ and $\mathcal{D}_t$, which are the peers that peer t serves

(iii) Rate Allocation: To serve downstream neighbors, each peer applies a uniform rate-allocation algorithm that takes into account content-availability. Specifically, let $\hat{\mathcal{D}}_t \subset \bar{\mathcal{D}}_t$ denote the set of peers in peer t's client set $\bar{\mathcal{D}}_t$, whom peer t has the requested data and is willing to serve. We call $\hat{\mathcal{D}}_t$ the *effective client set* of peer t. Let $\hat{D}_t = |\hat{\mathcal{D}}_t|$. Thus, the *effective downstream neighbor set* $\underline{\mathcal{D}}_t$ of peer t will be the intersection of the effective client set and the downstream neighbor set of peer t, i.e., $\underline{\mathcal{D}}_t = \hat{\mathcal{D}}_t \cap \mathcal{D}_t$. Then, each peer divides its upload capacity equally among all of its effective downstream neighbors. Thus, the streaming rate from peer s to peer t, $C_{s\to t}$, is equal to $V_s/\underline{D}_s$ if $t \in \underline{\mathcal{D}}_s$, and $C_{s\to t} = 0$, otherwise. Correspondingly, we can define the effective choice set $\hat{\mathcal{U}}_t$ of peer t as the set of peers in the choice set $\bar{\mathcal{U}}_t$ who has the required content of peer t. We have $\underline{\mathcal{U}}_t = \hat{\mathcal{U}}_t \cap \mathcal{U}_t$. See Table I for a summary of the relationship between these notations. Note that for rate-allocation, peers only need to know the content availability information at their neighbors. There is no need for the tracker to maintain content availability information, which leads to low control overhead.

(iv) Uniform Progress: There remains one serious high-overhead problem. In a P2P VoD system, it is possible that some peer downloads content at a higher speed than others. If that is the case, the tracker needs to constantly update and re-order their downloading positions. Further, some upstream neighbors of peer t may either fall behind or advance too far ahead. As a result, the neighbors of each peer may need to be re-selected constantly. There will then be significant overhead at the tracker.

We introduce the following condition to significantly reduce the overhead. Suppose that the targeted streaming rate is $(1 -$

$\epsilon)\mu$ at the video's normal playback speed. We enforce that the downloading position of each peer will also advance ahead of its playback position at the normal playback speed of the video. In other words, even if the available download rate that a peer receives from its upstream neighbors is larger than $(1-\epsilon)\mu$, it will still download content at the speed of $(1-\epsilon)\mu$. This condition ensures that the downloading positions of all peers advance at the same speed. In practice, the above design choice can be easily satisfied by the following protocol design: a peer will prefetch content for the video only up to a maximum lead-time ahead of its current playback position.

There are three benefits of this design. First, since the streaming rate of a video is known before-hand, the tracker can easily predict the advancement of each peer's downloading position. Unless a peer fast-forwards/backwards, there is no need for the tracker to update and re-order peers' downloading position. Hence, the signaling overhead is reduced significantly. Second, the upstream neighbors and downstream neighbors of each peer do not need to change constantly either, unless a neighbor leaves the system or fast-fowards/backwards. Third, the above design significantly simplifies our analysis because it is sufficient to focus on the streaming rates at a snapshot of time. On the other hand, some readers may be concerned that this design may unnecessarily constrain the downloading speed of those peers who could have downloaded faster. However, since our goal is to achieve the highest possible streaming rate for all peers, it is in fact more beneficial to maintain fairness. As we will show in our main result, our design is sufficient for attaining the close-to-optimal streaming capacity.

C. Performance Analysis

We have proposed a simple and decentralized algorithm that is easy to implement, is robust to changes in the peers' upload capacity, and incurs low control overhead at the tracker. Next, we show that the above algorithm will attain close-to-optimal streaming rate. Recall from the content availability condition that $q_t \geq q_{\min}$ for all peers t, and C_t is the downloading rate of peer t.

Theorem 1. *For any $\epsilon \in (0,1)$ and $d > 1$, choose $\alpha \geq \frac{8d}{pq_{\min}\epsilon^2}$ with $p = \frac{\mu}{V_{\max}}$. Suppose that each peer chooses $M = \alpha \log N$ upstream neighbors. Then for sufficiently large N and K, the following holds*

$$\mathbf{P}\left(C_t \leq (1-\epsilon)\mu, \text{ for some } t \in \mathcal{N}^-\right) \leq O\left(\frac{1}{N^d}\right). \quad (1)$$

Theorem 1 shows that $\Theta(\log N)$ upstream neighbors are sufficient for achieving close-to-optimal streaming rate of $(1-\epsilon)\mu$ for all peers with high probability. Further, it provides additional insights on the required number of neighbors as a function of the system parameters. First, if we wish to achieve a closer-to-optimal streaming rate (i.e., smaller ϵ) or a faster convergence of the probability (i.e., larger d), we need more neighbors per peer. Second, α is inversely proportional to $p = \frac{\mu}{V_{\max}}$. Hence, if there are higher levels of variation in the distribution of upload capacities (i.e., the peak rate $V_{\max}$ is large and/or a significant fraction of peers have small upload capacities), the required number of neighbors per peer must also be larger to tackle the extra level of randomness.

Another important consequence of Theorem 1 is that α is inversely proportional to $q_{\min}$. First, it is no longer necessary to ensure that an upstream neighbor of peer t always has the content that peer t requests (i.e., $q_t = 1$ for all t). According to Theorem 1, in order to ensure near-optimal streaming rates, it would be sufficient if each peer has at least $q_{\min}$ fraction of the content that its downstream peers will likely request. This relaxation significantly simplifies the system design when there are random-seeks. For example, the content availability strategy described earlier would be sufficient. On the other hand, in order to improve system performance, we should design P2P protocols with large values of $q_{\min}$, since it reduces the required number of neighbors.

We next provide a sketch of the proof of Theorem 1. We first fix any peer t and show that the probability for its downloading rate C_t to be smaller than $(1-\epsilon)\mu$ is $\frac{1}{N^{2d}}$. Theorem 1 then follows by taking the union bound. Note that peer t has exactly M upstream neighbors that may help it. Index these M upstream neighbors as $i = 1, ..., M$. Let I_i be the indicator function of the event that the i-th upstream neighbor of peer t is an effective upstream neighbor, and let $\mathbf{I} = [I_1, I_2, ...I_M]^T$. Then C_t can be represented by $C_t = \sum_{i=1}^{M} \frac{V_i I_i}{\underline{D}_i}$. We note that compared to our prior work [2] for live streaming, a main difficulty here stems from the number of effective downstream peers $\underline{D}_i$ for each upstream neighbor i. In [2], each upstream neighbor serves exactly M downstream peers. In contrast, here $\underline{D}_i$ is random and varies with an unknown parameter q_i. Further, there exists non-trivial correlation across i because the client sets of different upstream neighbors of peer t overlap. To address this difficulty, we use the following main supporting lemma.

Lemma 2. *Fix $q_{\min} > 0$. (a) Let $\tilde{\mathbf{I}} = [\tilde{I}_1, \tilde{I}_2, ..., \tilde{I}_M]^T$ be a set of M independent Bernoulli random variables such that $\mathbf{P}(\tilde{I}_i = 1) = q_i \geq q_{\min}$. (b) Let $\tilde{D}_i^+$, $i = 1, 2, ..., M$, be M positive (and possibly correlated) random variables such that $\mathbf{E}[\tilde{D}_i^+ | \tilde{\mathbf{I}}, \tilde{I}_i = 1] \leq \rho q_i M$ for some constant $\rho > 0$. (c) Let $\tilde{D}_i$, $i = 1, 2, ..., M$ be M positive (and possibly correlated) random variables such that for any $r_1, r_2, ..., r_M \geq 0$,*

$$\mathbf{E}\left[\exp\left(-\sum_{i=1}^{M} \frac{r_i}{\tilde{D}_i}\right) \middle| \tilde{\mathbf{I}}\right] \leq \prod_{i=1}^{M} \mathbf{E}\left[\exp\left(-\frac{r_i}{\tilde{D}_i^+}\right) \middle| \tilde{\mathbf{I}}\right]. \quad (2)$$

(d) Let $\tilde{V}_i$, $i = 1, 2, ..., M$, be M i.i.d. random variables independent from $\tilde{D}_i^+$'s and $\tilde{I}_i$'s such that $\mathbf{E}[\tilde{V}_i] = \mu$ and $0 < \tilde{V}_i < V_{\max}$ for all i. For and $d > 0$, let $\alpha \geq \frac{2dV_{\max}}{\epsilon^2 \mu q_{\min}}$. Then, for any $\epsilon > 0$ there exists N_0 such that when $N > N_0$ and $M = \alpha \log N$, the following holds

$$\mathbf{P}\left(\sum_{i=1}^{M} \frac{\tilde{V}_i \tilde{I}_i}{\tilde{D}_i} \leq \frac{(1-\epsilon)\mu}{\rho}\right) \leq O\left(\frac{1}{N^{2d}}\right).$$

The proof is omitted due to page limits and is available in [21]. We will soon relate $\tilde{I}_i$, $\tilde{D}_i$ and $\tilde{V}_i$ to I_i, $\underline{D}_i$ and V_i. To

interpret the result of Lemma 2, note that if $\tilde{D}_i = \tilde{D}_i^+$ and $\tilde{D}_i^+$'s are independent from each other conditioned on $\tilde{\mathbf{I}}$, then the condition in (2) trivially holds. Using Jensen's inequality, it is then easy to see that $\mathbf{E}[\tilde{C}_t] \geq \mu/\rho$, where $\tilde{C}_t = \sum_{i=1}^{M} \frac{\tilde{V}_i \tilde{I}_i}{\tilde{D}_i}$. Lemma 2 implies that, as long as $M = \alpha \log N$, the probability that $\tilde{C}_t \leq (1-\epsilon)\mu/\rho$ will diminish to zero. The conditions in the lemma, however, allows the result to hold even if $\tilde{D}_i$'s are correlated, and hence is very useful.

We will use Lemma 2 to show Theorem 1. For ease of exposition, we consider instead an alternative choice-set selection strategy called "random sequential-range", which is slightly different from the "random sequential" choice set selection strategy that we originally used. In such a "random sequential-range" choice set selection strategy, each user t choose a choice set $\bar{\mathcal{U}}_t$ that contains all the other peers whose downloading position are in the range $(t, \phi'(t)]$, where $\phi'(t)$ satisfies that $\int_t^{\phi'(t)} \gamma(\tau)\mathrm{d}\tau = Q$, if $\int_t^{T^{(0)}-} \gamma(\tau)\mathrm{d}\tau \geq Q$, and $\phi'(t) = T^{(0)}$, otherwise. Correspondingly, the client set $\bar{\mathcal{D}}_t$ of each peer t contains all the peers in the range $[\psi'(t), t)$, where $\psi'(t)$ satisfies that $\int_{\psi'(t)}^{t} \gamma(\tau)(d)\tau = Q$, if $\int_0^t \gamma(\tau)\mathrm{d}\tau \geq Q$, and $\psi'(t) = 0$, otherwise. Clearly, for any $t < T^{(0)} - \psi'(T^{(0)})$, $\mathbf{E}[\bar{U}_t] = NQ$. When N is large, $\bar{U}_t$ should concentrate on NQ. Hence, we would expect that the performance of the two choice-set selection strategy are close to each other. A more general statement can be made as in the following lemma.

Lemma 3. *Let $\mathcal{X}$ be the collection of all continuous intervals $\Gamma \subset [0, T^{(0)})$. Fix $L \geq 1$. Given any $\epsilon \in (0,1)$, let*

$$\mathcal{A} = \left\{ \left| \frac{\sum_{l=1}^{L} n_l}{N} - \int_{\cup_{l=1}^{L}\Gamma_l} \gamma(\tau)d\tau \right| \leq \epsilon \int_{\cup_{l=1}^{L}\Gamma_l} \gamma(\tau)d\tau + \epsilon, \right.$$
$$\left. \text{for all disjoint } \Gamma_1, ..., \Gamma_L \in \mathcal{X} \right\},$$

where n_l is the number of peers in Γ_l. Then, for any $d > 1$, there exists N_0 such that for any $N > N_0$, $\mathbf{P}(\mathcal{A}) \geq 1 - O\left(\frac{1}{N^{2d}}\right)$.

The proof of Lemma 3 is provided in [21]. Note that if $\mathcal{A}$ happens, then the number of peers in every $\cup_{l=1}^{L}\Gamma_l$ will be close to its mean value. Lemma 3 states that such an event $\mathcal{A}$ happens with high probability. In the following, we will focus on the situation when event $\mathcal{A}$ holds. Let $\mathbf{P}_{\mathcal{A}}(\cdot)$ and $\mathbf{E}_{\mathcal{A}}(\cdot)$ denote the probability and the expectation conditioned on $\mathcal{A}$.

We are now ready to prove Theorem 1. Fix a peer t and its set of M upstream neighbors $i = 1, ..., M$. First, we note that I_i's are independent because the content availability of each upstream neighbor i is independent. Further, let q_i be the parameter introduced in the content availability condition in Section II-B. Then $\mathbf{P}(I_i = 1) = \mathbf{P}_{\mathcal{A}}(I_i = 1) = q_i \geq q_{\min}$. Thus, condition (a) of Lemma 2 is met with $\tilde{I}_i = I_i$. Next, we will analyze the correlation between $\underline{D}_i$'s. Consider an upstream neighbor i. Let t_i be its current downloading position. If peer i recently random-sought to a position before t_i, let $t_{i_0} < t_i$ be the position that it first jumped to. Further, let Γ_i be the range of content from $[\psi'(t_{i_0}), t_{i_0}]$ that peer i randomly downloaded when it first jumped to t_{i_0}, according to the content availability strategy in Section II-B. Recall that the effective client set $\hat{\mathcal{D}}_i$ is a subset of $\bar{\mathcal{D}}_i$ that peer i has the requested content. $\hat{\mathcal{D}}_i$ consists of two parts: (a) all the n_1 peers in $\Gamma_i \cap [\psi'(t_i), t_{i_0})$, and (b) for the n_2 peers in $[t_{i_0}, t_i)$, each of them is in $\hat{\mathcal{D}}_i$ with probability q_i independent of others. Given $\mathcal{A}$ in Lemma 3, we must have, for any $\epsilon \in (0,1)$,

$$n_1 \leq N\left(\int_{\Gamma_i \cap [\psi'(t_i), t_{i_0}]} \gamma(\tau)\mathrm{d}\tau + \epsilon\right)(1+\epsilon) \triangleq n_1^+,$$
$$n_2 \leq N\left(\int_{[t_{i_0}, t_i]} \gamma(\tau)\mathrm{d}\tau + \epsilon\right)(1+\epsilon) \triangleq n_2^+.$$

Now, consider an alternative system by adding $(n_1^+ - n_1) + (n_2^+ - n_2)$ dummy peers. Construct a new set $\hat{\mathcal{D}}_i^+$ that contains all peers in $\hat{\mathcal{D}}_i$. In addition, the first group of $(n_1^+ - n_1)$ dummy peers are always added to $\hat{\mathcal{D}}_i^+$. For the second group of $(n_2^+ - n_2)$ dummy peers, each of them is in $\hat{\mathcal{D}}_i^+$ with probability q_i, independently of others. The advantage of making use of $\hat{\mathcal{D}}_i^+$ is that $\hat{D}_i^+$ only depends on I_i, t_{i_0} and Γ_i. Further, Γ_i and t_{i_0} are independent across i. Hence, $\hat{D}_i^+$'s are independent across i conditioned on $\mathcal{A}$. Further, $\hat{D}_i \leq \hat{D}_i^+$ by our construction. Next, consider $\underline{\mathcal{D}}_i \subset \hat{\mathcal{D}}_i$, i.e., the set of effective downstream neighbors of i. For each peer in $\hat{\mathcal{D}}_i$, it randomly choose M upstream neighbors, one of which may be i. Further, for each dummy peers in $\hat{\mathcal{D}}_i^+$, we also let it choose peer i as an upstream neighbor with prob $\frac{M}{NQ}$. Let $\underline{D}_i^+$ be the number of effective downstream neighbors of i in this alternative system. Note that $\underline{D}_i^+$ may still be correlated across i (even though $\hat{D}_i^+$'s are independent). This is because the sets $\hat{\mathcal{D}}_i^+$ may overlap, and if an overlapped peer s has picked i as an upstream neighbor, it will be less likely to pick another upstream neighbor $i' \in \{1, 2, .., M\}$. Fortunately, we can show a negative dependency between $\underline{D}_i^+$'s. Specifically, if $\underline{D}_i^+$ is large, then it is likely that less peers will pick i', and hence $\underline{D}_{i'}^+$ will likely be small. This negative dependency is made precise in the following lemma (see [21] for proof).

Lemma 4. *For any $r_1, r_2, ..., r_M \geq 0$, $\underline{D}_i^+$'s satisfy*

$$\mathbf{E}_{\mathcal{A}}\left[\exp\left(-\sum_{i=1}^{M}\frac{r_i}{\underline{D}_i^+}\right)\middle|\mathbf{I}\right] \leq \prod_{i=1}^{M}\mathbf{E}_{\mathcal{A}}\left[\exp\left(-\frac{r_i}{\underline{D}_i^+}\right)\middle|\mathbf{I}\right].$$

Note that $\underline{D}_i \geq \underline{D}_i^+$ by our construction. Hence, condition (c) of Lemma 2 holds with $\tilde{D}_i = \underline{D}_i$ and $\tilde{D}_i^+ = \underline{D}_i^+$. To verify condition (b), We can show the following lemma based on the content availability condition. The proof is in [21].

Lemma 5. *Suppose $\gamma_{\min} \leq \gamma(t) \leq \gamma_{\max}$ for all $t \in [0, T^{(0)})$ for some $0 < \gamma_{\min} \leq \gamma_{\max}$. For any $\epsilon \in (0,1)$, there exists K_0, such that for $K > K_0$, we have*

$$\mathbf{E}_{\mathcal{A}}\left[\underline{D}_i^+ \middle| \mathbf{I}, I_i = 1\right] \leq (1+\epsilon)\, q_i M.$$

Thus, condition (b) of Lemma 2 holds. Finally, note that V_i's are i.i.d. and independent of all other random variables. Hence, Theorem 1 follows from Lemma 2 for the "random sequential-range" choice set selection strategy. One can then show that Theorem 1 also holds for our original policy [21].

III. A Multi-Channel P2P VoD System

In the last section, we have focused on a single-channel P2P system. In this section, we study a multi-channel P2P system. Peers in each channel are interested in viewing a common video, which is however different across channels. Based on our single-channel algorithm, we will propose a simple and robust cache placement policy that could achieve a close-to-optimal streaming capacity for all channels.

A. System Model

We consider a P2P VoD system containing J channels. Let $\mathcal{J} = \{1, 2, ..., J\}$ denote the set of all channels, and $T_j^{(0)}$ denote the video length of channel j. Let $\mathcal{N}_j$ denote the set of peers that are watching channel j, and $N_j = |\mathcal{N}_j|$. Let $\mathcal{N}$ denote the set of all peers in the system, i.e., $\mathcal{N} = \bigcup_{j\in\mathcal{J}} \mathcal{N}_j$ and $N = |\mathcal{N}|$. We assume that $N_j = p_j \cdot N$, where p_j is the fraction of peers viewing channel j, which represents the popularity of channel j. Later on we will consider a system with large N, in which case we assume that p_j's are fixed and do not change with N. Note that N_j is fixed for a given N, which is consistent with our single-channel model. Within each channel, we use the same model as Section II-A, except that a subscript or superscript j is added to each notation to denote the channel. For example, $\bar{Q}_j$, V_t^j and $\mathcal{D}_t^j$ represents the probability that a channel j peer's downloading position is at $T_j^{(0)}$, the upload capacity of a peer in channel j and the set of downstream neighbors of peer t in channel j, respectively. We assume that $\mathbf{E}[V_t^j] = \mu$ for all j, i.e., the upload capacity in each channel has the same distribution.

Using the results from Section II, we know that each channel j can sustain a maximum streaming rate around $(1-\epsilon)\mu$. However, in a multi-channel system, it is typical that different channels have different streaming rate requirements. Let R_j denote the targeted streaming rate for the video of channel j. Let $\mathbf{R} = [R_1, R_2, ..., R_J]^T$. Naturally, the streaming rate in some channel j may satisfy $R_j \leq (1-\epsilon)\mu$, which implies that the upload capacity of peers viewing the channel is sufficient to support the targeted streaming rate. Such channels are referred to as *sufficient* channels. On the other hand, some other channel may have $R_j \geq (1-\epsilon)\mu$. We call such channels *insufficient* channels. We denote the set of insufficient channels as $\mathcal{I} = \{j \in \mathcal{J} | R_j > (1-\epsilon)\mu\}$, and the set of sufficient channels as $\mathcal{S} = \{j \in \mathcal{J} | R_j \leq (1-\epsilon)\mu\}$. Seemingly, peers in an insufficient channel will not have enough upload capacity to stream the desired video.

A natural idea to improve the overall system performance is to use the extra capacity from sufficient channels to help the peers in insufficient channels. This kind of helping will obviously support a larger set of vectors $\mathbf{R}$ of streaming rate requirements. We define the *streaming capacity region* Λ of the multi-channel system as the set of streaming rate vectors, such that for each $\mathbf{R} \in \Lambda$, under some centralized peer-selection and rate-allocation strategy, every peer in the system can receive a sufficient downloading rate R_j to view its desired channel. Assuming that the contribution of server capacity is minimal, the largest possible streaming capacity region is given by $\Lambda'_m = \left\{ \mathbf{R} \left| \sum_{j=1}^{J} (1-\bar{Q}_j) N_j R_j \leq \sum_{i\in\mathcal{N}} \mathbf{E}[V_i] \right. \right\}$. In other words, since the upload capacity of peers is the only in the system, the best we can do is to support those rate vectors $\mathbf{R}$ such that the summation of all demand is no greater than the summation of the overall upload capacity. Again, $\bar{Q}_j$'s are usually not very large in practice, and hence we will omit the contribution of $\bar{Q}_j$ in the rest of this section. Let

$$\Lambda_m = \left\{ \mathbf{R} \left| \sum_{j=1}^{J} N_j R_j \leq \sum_{i\in\mathcal{N}} \mathbf{E}[V_i] \right. \right\}.$$

We say that a multi-channel control algorithm achieves a close-to-optimal capacity region, if for any $\mathbf{R} \in (1-\epsilon)\Lambda_m$ with some $\epsilon > 0$, all peers in each channel j can sustain the streaming rate R_j.

In order for peers from a sufficient channel k to help peers in an insufficient channel j, the peers in channel k must already have the content for channel j, in addition to the content for channel k that they are interested in viewing. For this purpose, we assume that, in addition to the video from its own channel, each peer also caches an additional video from one other channel, and hence can serve this cached video to peers in that channel. (Note that although we assume that the entire video from another channel is cached in this case, a similar line of analysis can be carried out if the video from another channel is divided into a small number of parts, and each peer only cached one part of the video.) Further, we assume that the cached content has already been pre-loaded, and we ignore the bandwidth resources to place these cached contents. We will then study the optimal placement probabilities for each video and how to best use the cached content. We note that a similar assumption of pre-loading cached content has been made in other prior works [13], [14], [18] that study the optimal cache placement probability. In practice, this kind of proactive deployment can be implemented in several ways. One possibility is to let the peers download the cached videos from the server during non-busy hours. Such a method is especially useful when the peers are always online, e.g., when using set-top boxes. Another possibility is to perform active push or passive replacement using a randomized algorithm [18]. The key assumption here and in [13], [14], [18] is that the cache content will change at a much slower time-scale than the content that each peer is interested in viewing. Hence, the cache replenishment process can be performed much more slowly, and thus the amount of bandwidth consumed for cache placement will be significantly smaller than the amount of bandwidth consumed for streaming.

B. Algorithm and Performance

We start with our cache-placement algorithm, which has some similarity to the "proportional-to-deficient-bandwidth" policy in [18]. (However, note that its optimality is not rigorously shown in [18].)

(i) Cache Placement: As we discussed earlier, each peer will cache one other video in addition to its currently-watching

video. The tracker maintains which peers cache which videos. Given $\mathbf{R} \in (1-\epsilon)\Lambda_m$, the tracker determines the required number of additional helpers for each channel j, h_j^r, according to $h_j^r = \frac{N_j R_j}{\mu\sqrt{1-\epsilon}} - \sqrt{1-\epsilon}N_j$. Here, h_j^r can be interpreted as the deficit of upload bandwidth in channel j. Note that using h_j^r, the tracker can classify sufficient and insufficient channels: for a sufficient channel j, h_j^r is negative or zero; for an insufficient channel j, h_j^r gives a positive value. Every peer in each sufficient channel k caches a video randomly chosen from those of insufficient channels with the following distribution: the probability η_{kj} that a peer in channel k caches the video of channel j satisfies

$$\eta_{kj} = \eta_j \triangleq \frac{h_j^r}{\sum_{l\in\mathcal{I}} h_l^r}, \text{ for all } k \in \mathcal{S}, j \in \mathcal{I}. \tag{3}$$

Note that this probability only depends on R_j (video rate), μ (average upload capacity), p_j (video popularity), but is independent of N. Due to such a randomized cache placement policy, a random number of peers in each sufficient channel k cache a copy of channel j's video. Let us denote this number by $\tilde{H}_{kj}$. The total number of peers in sufficient channels that cache the video for channel j is then $\tilde{H}_j = \sum_{k\in\mathcal{S}} \tilde{H}_{kj}$. In our algorithm, the tracker randomly chooses H_{kj} peers among the $\tilde{H}_{kj}$ peers in channel k (which cache video j) to help channel $j \in \mathcal{I}$, where H_{kj} is given by $H_{kj} = \left\lceil \frac{|h_k^r h_j^r|}{\sum_{l\in\mathcal{S}} |h_l^r|} \right\rceil$. We call these H_{kj} peers "helpers" for channel j, and we use $\mathcal{H}_j$ to denote the set of all helpers assigned to help channel j. Note that if $H_{kj} > \tilde{H}_{kj}$, our algorithm would fail because there is not a sufficient number of peers who cache the video. However, we show in [21] that this failure probability goes to 0 as $N \to \infty$. Hence, the actual number of helpers for each channel j is $H_j \triangleq |\mathcal{H}_j| = \sum_{k\in\mathcal{J}} H_{kj}$.

(ii) Peer Selection and Rate Allocation: Each peer t in an insufficient channel j uniformly randomly selects M_N upstream neighbors from its choice set $\bar{\mathcal{U}}_t^j$ and uniformly randomly picks M_H upstream neighbors from its helper set $\mathcal{H}_j$, where $M_N + M_H = M$. Each peer in a sufficient channel only needs to select $M_N = M$ upstream neighbors from its choice set (i.e., $M_H = 0$ for peers in sufficient channels). Note that if a peer in a sufficient channel k is selected into the helper set $\mathcal{H}_j$ of an insufficient channel j, its upload capacity will be completely reserved for serving peers in channel j, and will not be used to serve peers in its own viewing channel. Each upstream peer still applies the uniform rate-allocation strategy. All other parts of the peer selection and rate allocation algorithms remain the same as in the single-channel case. We can show that with our simple multi-channel control algorithms, the targeted streaming rate of each channel can be attained with high probability. Specifically, let C_t^k be the achieve streaming of peer t in channel $k \in \mathcal{J}$. We have the following main result for multi-channel systems. Detailed analysis and proofs are omitted due to space constraints and are available in our online technical report [21].

Theorem 6. *Given any $\epsilon \in (0,1)$, $d > 1$ and $\mathbf{R} \in (1-\epsilon)\Lambda_m$. Let $\epsilon' = 1 - \sqrt{1-\epsilon}$. There exists N_0 such that if $N \geq N_0$, $M = \alpha \log N$ and $\alpha \geq \frac{16d}{\min\{\rho_{\min}, p, 2\sigma_{\min} p\} q_{\min} \epsilon'^2}$, then we can find M_H and M_N such that*

$$\mathbf{P}\left(C_t^k \leq R_k, \text{ for some } k \in \mathcal{J} \text{ and } t \in \mathcal{N}_k^-\right) \leq O\left(\frac{1}{N^d}\right).$$

IV. Simulation Results

In this section, we provide simulation results of both single-channel and multi-channel systems to verify our analytical results. We first simulate a single-channel system and study the probability that peers achieve close-to-optimal streaming capacity as the number of each peer's upstream neighbor number increases. We will also compare performance as we vary different system parameters, such as the distribution of peers' upload capacity (represented by $p = \frac{\mu}{V_{\max}}$) and the content availability at peers (represented by $q = q_{\min}$). Throughout, the single channel system has $N = 20000$ peers. The upload capacity of each peer is assumed to be ON-OFF, i.e., $\mathbf{P}(V_i = V_{\max}) = p$ and $\mathbf{P}(V_i = 0) = 1 - p$ for each peer i. We assume that $V_{\max} = 10$. The average upload capacity of peers is $\mu = pV_{\max}$. The video length of the channel is $T^{(0)} = 3$ (hours), and we assume that the downloading positions of all the peers satisfy an exponential distribution, with the density function $\gamma(t) = e^{-t}$ for $t \in [0, 3)$. The parameter $\bar{Q}$, which is the probability that a peer's downloading position is $T^{(0)}$, is given by $\bar{Q} = e^{-3} \approx 0.05$. We vary the number of upstream neighbors per peer from $M = 10 \log N = 99$ to $M = 90 \log N = 891$, which correspond to 0.5% to 4.45% of the total number of peers N. Then, for each choice of the system parameters (p, q, ϵ) and the number of upstream neighbors per peer, we generate a single-channel P2P VoD streaming system according to our single-channel P2P control algorithms for 1000 times. In each run of the simulation, we record the smallest downloading rate among all peers and compare it with $(1-\epsilon)\mu$. We count the number of times that this smallest downloading rate is larger than $(1-\epsilon)\mu$ and plot the probability for that to happen. The result is shown in Fig. 1. We can observe from the simulation results that, when $p = 0.9$, $q = 0.9$, $\epsilon = 0.3$, and when each peer selects no fewer than $10 \log N = 100$ (which corresponds to 0.5% of N) upstream neighbors, a downloading rate higher than $1 - \epsilon = 70\%$ of the average peer upload-capacity can be achieved in the entire network with probability close to 1. (We note that while $q_{\min} = 0.9$ appears to be large, it only means that each peer has 90% of the content for the range of its client set, which is of a small size $NQ = 0.05N$.) When p is reduced to 0.5 or q is reduced to 0.5, more upstream neighbors are needed to achieve the same performance. Further, under the same values of p and q, when we reduce ϵ to 0.2, more upstream neighbors are needed to achieve the same performance. These observations verify our insights following Theorem 1.

Next, we simulate a multi-channel P2P VoD system with 4 channels. We use the same settings as in the single-channel simulations on the distribution of peer upload capacities and the distribution of peers' downloading positions. We set $V_{\max} = 10$ and $p = 0.5$. The content availability is given by $q = q_{\min} = 0.9$. We set $N_1 = 4000$, $N_2 = 6000$, $N_3 = 3000$, $N_4 = 7000$.

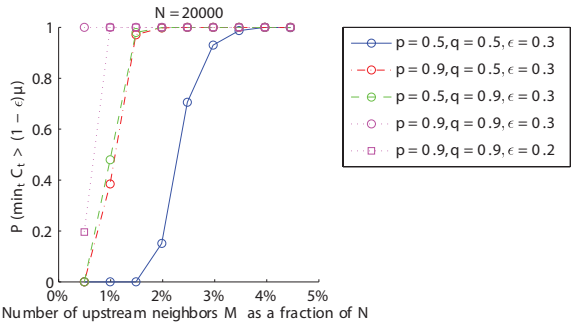

Fig. 1. Single-channel system: the probability of success as the number of upstream neighbors increases.

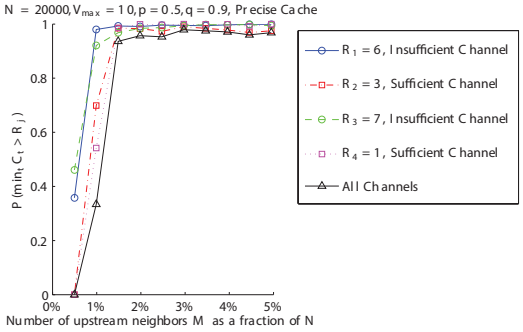

Fig. 2. Multi-channel system: the probability of success as the number of upstream neighbors increases.

We choose a target streaming rate vector $\mathbf{R} = [6, 3, 7, 1]^T$, which is in $0.7\Lambda_m$ (*i.e.*, $\epsilon = 0.3$). Channels 1 and 3 are insufficient channels, and channels 2 and 4 are sufficient channels. In Fig. 2, we plot the probability that the downloading rate of a peer in channel j is greater than its target streaming rates R_j, for each of the four channels as the number of upstream neighbors per peer varies. Further, the curve with "Δ" plots the probability that all peers in all channels simultaneously sustain downloading rates greater than their corresponding target streaming rates. As we can see from Fig. 2, all channels attain with high probability their required streaming rates even with a small number of upstream neighbors.

V. Conclusion

In this paper we provide a rigorous analytical study on the performance of large-scale P2P VoD systems with sparse connectivity and simple, robust, and decentralized control. For both single-channel and multi-channel systems, we provide easy-to-implement P2P control algorithms and show that the system can achieve close-to-optimal streaming capacity with probability approaching 1, as the total number of peers N increases. Under our control algorithms, each peer is only assigned $\Theta(\log N)$ upstream neighbors, with which it exchanges content availability information. Most parts of the control algorithms are decentralized. These algorithms incur low control overhead and are easy-to-implement in practice. Our analytical studies provide easy-to-verify conditions for such close-to-optimal streaming to hold, which shed important insights to guide the design of improved P2P streaming protocols. For future work, it would be interesting to study whether the required number of per-peer neighbors can be further reduced, possibly by using more sophisticated peer-selection and rate-allocation algorithms than those studied in this paper. The challenge would be how to improve the system performance while retaining the simplicity and decentralized properties.

Acknowledgments: This work is partially supported by the National Science Foundation through grant CNS-0643145, CNS-0721484 and CNS-0831999, and a grant from Hong Kong RGC under the contract HKU 718710E.

References

[1] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale Operational On-Demand Streaming with Random Network Coding," in *Proc. of IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[2] C. Zhao, X. Lin, and C. Wu, "The Streaming Capacity of Sparsely-Connected P2P Systems with Distributed Control," in *Proc. of IEEE INFOCOM*, Apr. 2011, pp. 1449–1457.

[3] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, May 2007, pp. 919–927.

[4] L. Massoulié, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized Decentralized Broadcasting Algorithms," in *Proc. of IEEE INFOCOM*, 2007, pp. 1073–1081.

[5] C. Feng and B. Li, "On Large-scale Peer-to-Peer Streaming Systems with Network Coding," in *Proc. of ACM Multimedia*, 2008, pp. 269–278.

[6] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic Live Streaming: Optimal Performance Trade-offs," in *Proc. of ACM SIGMETRICS*, 2008, pp. 325–336.

[7] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance Bounds for Peer-Assisted Live Streaming," in *Proc. of ACM SIGMETRICS*, 2008, pp. 313–324.

[8] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Peer-to-Peer Streaming Capacity," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5072–5087, Aug. 2011.

[9] S. Liu, M. Chen, S. Sengputa, M. Chiang, J. Li, and P. A. Chou, "P2P Streaming Capacity under Node Degree Bound," in *Proc. of IEEE ICDCS*, Jun. 2010, pp. 587–598.

[10] D. Wu, C. Liang, Y. Liu, and K. Ross, "View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems," in *Proc. of IEEE INFOCOM*, Apr. 2009, pp. 2726–2730.

[11] Y. Huang, T. Z. J. Fu, D. M. Chiu, J. C. S. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-scale P2P-VoD System," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 375–388, Aug. 2008.

[12] K. Wang and C. Lin, "Insight into the P2P-VoD System: Performance Modeling and Analysis," in *Proc. of IEEE ICCCN*, Aug. 2009, pp. 1–6.

[13] B. Tan and L. Massoulié, "Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems," in *Proc. of IEEE INFOCOM*, Apr. 2011, pp. 694–702.

[14] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "Statistical Modeling and Analysis of P2P Replication to Support VoD Service," in *Proc. of IEEE INFOCOM*, Apr. 2011, pp. 945–953.

[15] J. Wu and B. Li, "Keep Cache Replacement Simple in Peer-Assisted VoD Systems," in *Proc. of IEEE INFOCOM*, Apr. 2009, pp. 2591–2595.

[16] N. Parvez, C. Williamson, A. Mahanti, and N. Carlsson, "Analysis of Bittorrent-like Protocols for On-Demand Stored Media Streaming," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, pp. 301–312, Jun. 2008.

[17] N. Carlsson and D. L. Eager, "Peer-assisted On-Demand Streaming of Stored Media Using BitTorrent-like Protocols," in *Proc. of the 6th international IFIP-TC6 conference on Ad Hoc and sensor networks, wireless networks, next generation internet*, 2007, pp. 570–581.

[18] W. Wu and J. Lui, "Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation," in *Proc. of IEEE INFOCOM*, Apr. 2011, pp. 1206–1214.

[19] J. Wang, C. Huang, and J. Li, "On ISP-Friendly Rate Allocation for Peer-Assisted VoD," in *Proc. of ACM Multimedia*, 2008, pp. 279–288.

[20] C. Liang, Y. Guo, and Y. Liu, "Is Random Scheduling Sufficient in P2P Video Streaming?" in *Proc. of IEEE ICDCS*, Jun. 2008, pp. 53–60.

[21] C. Zhao, J. Zhao, X. Lin, and C. Wu, "Capacity of p2p on-demand streaming with sparse connectivity and simple decentralized control," Purdue University, Tech. Rep., 2012, also available at https://engineering.purdue.edu/~linx/papers.html.