A Simple Model for A Generic Class of P2P Streaming Algorithms

Yipeng Zhou, Dah Ming Chiu, *Fellow, IEEE*, and John CS Lui, *Senior Member, IEEE*

***Abstract*— P2P streaming tries to achieve scalability (like P2P file distribution) and at the same time meet real-time playback requirements. It is a challenging problem still not well understood. In this paper, we describe a simple stochastic model that can be used to compare different downloading strategies with random peer selection.Based on this model, we study the tradeoffs between supported peer population, buffer size and playback continuity. We first study two simple strategies: Rarest First and Greedy. The former is a well-known strategy for P2P file sharing that gives good scalability by trying to propagate the chunks of a file to as many peers as quickly as possible. The latter is an intuitively reasonable strategy to get urgent chunks first to maximize playback continuity from a peer's local perspective. Yet in reality, both scalability and urgency should be taken care of. With this insight, we propose a Mixed strategy that achieves the best of both worlds. Furthermore, the Mixed Strategy comes with an adaptive algorithm that can adapt it buffer setting to dynamic peer population. We validate our analytical model with simulation. Finally, we also discuss the modeling assumptions and the model's sensitivity to different parameters, and show that our model is robust.**

I. Introduction

Video streaming over the Internet is already a widely deployed service. The engineering of video streaming from a server to a single client is well studied and understood. This, however, is not scalable to serve a large number of clients simultaneously. In recent years, a clever solution has emerged - peer-to-peer (P2P) video streaming, which works surprisingly well. A number of commercial systems are in service today, such as [1], [2].

The idea is very simple: let the peers, other users interested in the same content, help the source of the content in its distribution. The more peers are interested in the content, the more helpers in distributing the content, so it becomes scalable. The original mechanism is P2P *file sharing*. Each peer obtains an entire file before this possession is known by others. Other peers may then request for the file. This mechanism is quite adequate for small files, such as a picture, or an audio file. For a large file, be it video, software, or other content, this mechanism can incur a large delay. It is like a store-and-forward system without pipelining.

A new kind of P2P algorithm soon got developed, known as P2P *file downloading*. The most well-known example is BitTorrent [3]. In this case, the file is divided into a number of *chunks*. In trying to download a file, a peer simultaneously engages in downloading (or more precisely sharing) all the chunks of that file. If there are N chunks in the file, one can visualize the situation as N file sharing sessions carrying on at the same time. The result is that all peers can become fully engaged in file sharing all the time, and the delay in propagating the whole file to all peers can be minimized. The key is that there needs to be a good *schedule* of which peer is to get which chunk from which other peer at each moment.

There are two main approaches to this scheduling problem: *structured* and *unstructured*. In the first case, the basic idea is to form K distribution trees, each is a spanning tree from the source to all the peers. The chunks of the file are distributed via different trees in a round-robin fashion. The amount of service each peer provides is related to the total out-degree it has in these spanning trees, and the timing of the service depends on the peer's position in different trees. The challenge of the structured approach is to come up with the distribution trees that fully utilize all the peers, which intuitively will also minimize delay. The difficulty with this approach is how to deal with peer churn, and how to get the peers provide their information reliably for such centralized planning. In the second case, there is no structure; peers just download from each other based on local information of what is available and what is needed. Besides selecting which chunks to download (share), each peer must select which neighbor peer to exchange with (known as *peer selection*) and how fast to request and serve others (we call that *load balancing*). All these mechanisms can be implemented as distributed algorithms, as exemplified by BitTorrent [3] and several other systems [1], [2]. Perhaps due to its simplicity (being distributed) and robustness (to peer churn), the unstructured approach is very popular in practice. It is quite surprising that the seemingly rather chaotic unstructured approach works at all. So the unstructured approach is also receiving a lot of attention from the academic community [4]–[18].

P2P *streaming* can be thought off as a special case of P2P file downloading. The focus of P2P streaming is no longer only delay and throughput, but also the more stringent playback performance. For this reason, some algorithms that are considered *optimal* for file downloading may not be optimal for streaming[1].

Yipeng Zhou and Dah Ming Chiu are from the Department of Information Engineering, whereas John Lui is from the Department of Computer Science and Engineering of The Chinese University of Hong Kong (email: {ypzhou6,dmchiu}@ie.cuhk.edu.hk) and cslui@cse.cuhk.edu.hk).

[1]This is the reason that some results in our paper are somewhat different from the conclusions in other recent papers [5], [7], [16]. We will discuss this point in more detail in the Related Works section.

In the study of P2P content distribution algorithms, whether it is for file downloading or video streaming, practice is leading theory. In practice, chunk selection, peer selection and load balancing algorithms must all be considered and designed to work together to achieve the best results. The methodology for evaluation is often based on controlled network experiments, such as PlanetLab, Emulab, or experimental deployment in campus networks. Practical systems are usually designed to be *upgradable*, so that new versions can be tested in real life environments. In spite of the success of practice, there is still great interest in theoretical models of these P2P distributed algorithms that are able to provide the insights of why these algorithms work; explain the design trade-offs; and provide a way to understand the robustness, i.e., the sensitivity of these algorithms to various system parameters.

In the theoretical models of P2P algorithms, it is usually not possible to model all the aspects (chunk selection, peer selection, and load balancing) at the same time. To focus on one (or two) aspect only, it is possible to assume an abstract setting in which only one problem is relevant. For example, in studying chunk selection algorithms, we can assume peer selection is random, and all peers have the same capacity so that there is no need for load balancing. This is the approach taken by [5], [6]. In [10], in order to focus on the load balancing problem [10], it is assumed that all peers already have all the content so that chunk selection is not needed.

The main results of the current paper are already published in [9]. The Zhou-Chiu-Lui model in [9] models the buffer state of peers; and by assuming homogeneous peers[2], and by making an approximation via an independence assumption [3], it is possible to write down the probability of buffer occupancy in terms of a set of differential equations. Hence the continuity, or the playback performance, can be explicitly computed and studied relative to various chunk selection algorithms and system parameters. This analysis allows us to understand the basic trade-offs in chunk selection, and propose a near-optimal yet practical algorithm. In this paper, (1) to improve the presentation, we re-organize and re-state the lemmas and propositions; (2) we discuss the optimality of the proposed algorithms, based on an upper bound; (3) we add a detailed discussion of the contribution of these results by comparing it to some recent and significant related works.

The organization of the paper is as follows. Section II is on the basic probabilistic model; Section III goes into the details of how to model different chunk selection strategies; Section IV provides various numerical examples, solved by both the discrete and the continuous version of our model, as well as validated by simulation. Section VI describes application of our protocol to real protocol design, while in section V discusses the reasonableness of the assumptions in our model and the conclusion is given in Section VII.

[2]All the peers are probabilistically the same.

[3]All the buffer positions can be considered independently.

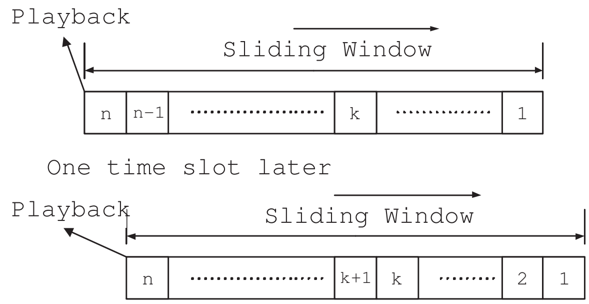


Fig. 1. Sliding Window Mechanism of the buffer B

II. Basic Model

We begin by defining notations and stating assumptions.

Let there be M peers in the network[4]. There is a single server that *pushes* chunks of (video) content, in playback order, to the M peers. New chunks are generated at the rate of one chunk per time slot. If the server selects the one peer randomly (to push a chunk) in each time slot, each peer would be receiving new chunks at the rate of $\frac{1}{M}$.

Each peer maintains a buffer B that can cache up to n chunks received from the network. We refer the buffer positions according to the *age* of the chunks stored: $B(n)$ is reserved for the chunk to be played back immediately; $B(1)$ is used to store the newest chunk that the server is distributing in the current time slot. In other words, when the server is distributing chunk t (at time t), if $t \geq n-1$ then chunk $t-n+1$ is the chunk being played back by that peer. After each time slot, the chunk played back in the previous time slot is removed from B and all other chunks are shifted up by 1. In other words, the buffer acts as a sliding window into the stream of chunks distributed by the server, as shown in Figure 1. Each buffer space is initially empty, and gets filled by the P2P streaming protocol, either from the server or from other peers. The goal is to ensure $B(n)$ is filled in as many time slots as possible, so as to support the continuous video playback.

Let $p_k(i)[t]$ denote the probability that the i^{th} buffer space, $B(i)$, of peer k is filled with the correct chunk at time t. We assume this probability reaches a steady state for sufficiently large t, namely $p_k(i)[t] = p_k(i)$. We call $p_k(i)$ the buffer occupancy probability of the k^{th} peer[5].

Let us first consider a simple case that the server is the only means for distributing chunks to peers, then the buffer occupancy distribution can be expressed as follows:

$$p_k(1) = p(1) = \frac{1}{M} \qquad \forall k, \tag{1}$$

$$p_k(i+1) = p(i+1) = p(i) \quad i = 1, 2, \ldots, n-1 \;\; \forall k. \tag{2}$$

Eq. (1) reflects the odds for the local peer to be picked by the server, while Eq. (2) reflects the fact that successful downloading only occurs at the first location of the buffer

[4]As we will see later, if M is reasonably large then our results are essentially independent of M, nor do they require M to be a constant.

[5]Note, the buffer occupancy probability is not a probability distribution of i since it is not necessarily true that $\sum p_k(i) = 1$.

(from the server). The playback performance, given by $p(n)$, is equal to $\frac{1}{M}$, would obviously be very poor for any $M > 1$. In general, we refer to $p(1)$ as the *input rate* from server, observed at each peer. This input rate must be greater or equal to $\frac{1}{M}$. The server's upload bandwidth to sustain an input rate of $p(1)$ is $p(1)M$. This shows the scalability problem when the server is the only means of distributing the content. In the rest of the paper, we assume $p(1) = \frac{1}{M}$ unless stated otherwise.

To improve playback performance, peers help each other when asked. We model the unstructured P2P mechanism as a *pull* process: each peer selects another peer in each time slot to try to download a chunk not already in its local buffer. This P2P downloading model has the following implications:

- A peer may be contacted by multiple other peers in a single time slot. In this case, it is assumed that the selected peer's uploading capacity is large enough to satisfy all the requests in the same time slot. If peers are selected randomly, the probability that it will be selected by $k \geq 0$ peers is $\beta(k)$, where
$$\beta(k) = \binom{M-1}{k} \left(\frac{1}{M-1}\right)^k \left(\frac{M-2}{M-1}\right)^{M-1-k}$$
for $k \geq 0$. The likelihood of being selected by many other peers is low, e.g., when there are $M = 100$ peers, the probability that it is selected by more than three peers is only around 1.8%.
- If the selected peer has no useful chunk, the selecting peer loses the chance to download anything in a time slot. This simplifying assumption can help us to derive closed-form expression[6].

Furthermore, we assume homogeneous peers[7], namely, all peers use the same strategy to select other peers and chunks to download at the same downloading rate. The implication is that in the steady state, all peers have the same distribution $p(i)$ for the buffer occupancy, as in the server-only downloading case above. In this paper, we only consider random peer selection strategies. Intuitively and from previous results in the literature, we know peer selection strategy is an important factor when peers have different uplink bandwidth, or when the paths to different peers have different bottleneck capacity. In these scenarios, peers are non-homogeneous and asymmetric. Once we assuming peers are homogeneous, however, it is reasonable to adopt the random peer selection strategy to keep the problem tractable.

Once a peer is selected, a chunk for downloading must also be specified. The chunk selection policy can be represented by a probability distribution q, where $q(i) \geq 0$, gives the probability that the chunk needed to fill $B(i)$ is selected. Hence, Eq. (2) becomes:

$$p(i+1) = p(i) + q(i) \quad i = 1, \ldots, n-1, \tag{3}$$

with the boundary condition of $p(1) = 1/M$. For $i > 0$, $q(i)$ is expected to be greater than 0 since there is a non-zero probability that a peer may be found to fill $B(i)$ if it is not already filled. This implies $p(i)$ is an *increasing* function of i, hence collaboration by peers improve the playback performance as expected.

Consider a particular peer k, and assume it selected peer h to download a chunk. The selection of a particular chunk to download is based on the following events:

- WANT(k,i): $B(i)$ of peer k is unfilled; we abbreviate this event as $W(k,i)$.
- HAVE(h,i): $B(i)$ of peer h is filled; we abbreviate this event as $H(h,i)$.
- SELECT(h,k,i): Using the chunk selection strategy, peer k cannot find a more preferred chunk than that of $B(i)$ that satisfies the WANT and HAVE conditions; we abbreviated this event as $S(h,k,i)$.

Therefore, we can express $q(i)$ as:

$$\begin{aligned} q(i) &= \Pr[W(k,i) \cap H(h,i) \cap S(h,k,i)] \\ &= \Pr[W(k,i)] \Pr[H(h,i)|W(k,i)] \times \\ &\qquad \Pr[S(h,k,i)|W(k,i) \cap H(h,i)]. \end{aligned} \tag{4}$$

The following assumptions help us to simplify Eq. (4):

- All peers are independent: the probability of the buffer state at the same position for different peers, $p(i)$, are the same. Therefore, $\Pr[W(k,i)] = 1 - p(i)$.
- There are a large enough number of peers so that knowing the state of one peer does not significantly affect the probability of the state at another peer. This implies that:
$$\Pr[H(h,i)|W(k,i)] \approx \Pr[H(h,i)] = p(i).$$
- The chunks are independently distributed in the network. The probability distribution for position i is not strongly affected by the knowledge of the state at other positions. This allows us to write the selection function as
$$s(i) = \Pr[S(h,k,i)|W(k,i) \cap H(h,i)] \approx \Pr[S(h,k,i)],$$
which is independent of the actual state at position i. As we will show, this assumption is more accurate for some chunk selection strategies than others.

Based on the above assumptions, Eq. (4) is:

$$q(i) \approx [1 - p_k(i)]\, p_h(i) s(i) = [1 - p(i)]\, p(i) s(i). \tag{5}$$

Since each of the terms in Eq. (5) is a probability (in particular $p(i) \leq 1$ and $p(i)s(i) \leq 1$), Eq. (3) becomes:

$$p(i+1) = p(i) + [1 - p(i)] p(i) s(i) \leq 1. \tag{6}$$

The chunk selection strategy $s(i)$, the focus of this study, is discussed in the next section.

Each peer tries to download one chunk from another peer in a time slot, which is reasonable for streaming[8]. Because of

[6]This type of assumption is also made in other P2P file sharing models [15]

[7]This assumption is made in many similar work on the modeling of p2p network, such as [5], [6], [17]. We make the same assumption so that the problem is tractable.

[8]For a progressive downloading system, a peer may try to download faster than that

this assumption, a peer's *chunk selection strategy*, $s(i)$, is a probability distribution although $s(i)$ may not sum up to 1 because there is always some probability that no useful chunk can be downloaded. The choice of $s(i)$ has a great effect on playback continuity. To help understand what the best $s(i)$ can possibly achieve, we can relax the assumption, by allowing each peer to fetch all useful chunks from the selected neighbor, in each time slot. This is equivalent to letting $s(i) = 1$ for all i. This *unconstrained* chunk selection strategy can be used to derive an upper bound playback continuity achievable by any $s(i)$. After setting $s(i) = 1$, Eq. (6) becomes:

$$p(i+1) \quad = \quad p(i) + p(i)(1 - p(i)) \qquad (7)$$

The upper bound continuity is derived from the solution of this equation, which will be used later to consider optimality of chunk selection strategies.

Another quantity of interest is the number of time slots it takes for a chunk to be distributed to all peers, which is a lower bound for the buffer size n. Intuitively, we know this lower bound must be greater than $\lceil \log_2(M) \rceil$ because in each time slot, the number of peers possessing a particular chunk can at most double from the previous time slot. Eq. 7 can give us a tighter lower bound on the buffer size, taking into consideration of the achieved playback continuity. This will be discussed in detail in the next section.

III. Chunk Selection Strategies

The simple stochastic model in the previous section set the stage for us to model and analyze different chunk selection strategies. We begin by considering some familiar strategies. The first one is the "*Rarest First Strategy*", which is widely adopted in P2P file distribution protocol BitTorrent [8], [17], and P2P streaming protocol CoolStreaming [4]. The second one is the "*Greedy Strategy*" (or the nearest deadline first strategy), and lastly the *mixed strategy*, which is a combination of the above two algorithms.

By intention, a peer using the Rarest First Strategy will select a chunk which has the *fewest number of copies* in the system. To describe the Rarest First Strategy from the perspective of the buffer $B = \{B(n), B(n-1), \ldots, B(1)\}$, let us consider a particular peer, say peer k. From Eq. (3), we know that $p(i)$ is an increasing function of i, therefore $p(i+1) \geq p(i)$ for $i{=}1, \ldots, n{-}1$. Since peers are homogeneous, this inequality implies that the expected number of copies of chunk in $B(i+1)$ is greater than or equal to the expected number of copies of chunk in $B(i)$. Therefore, under the Rarest First Strategy, peer k will first select $B(1)$ to download if $B(1)$ is not available in k's buffer. If chunk $B(1)$ is already downloaded before or $B(1)$ is not available in its neighbor, peer k will select $B(2)$ to download if $B(2)$ is not in k's buffer and so on.

For the Greedy Strategy, peer k will select a chunk which is *closest to its playback deadline*. From buffer B's point of view, $B(n)$ is the closest to playback time, then $B(n-1)$ is the next, and so on. Therefore, peer k will first try to download $B(n)$ if it is not available in k's buffer. If the chunk $B(n)$ is already downloaded before or $B(n)$ is not available in k's neighbor, the peer k will select $B(n{-}1)$ to download if $B(n{-}1)$ is not in k's B and so on. Note that the Greedy Strategy seems intuitively the best strategy for streaming at the first sight. Through our analysis, we will show that while from a single peer's point of view Greedy may be the best for playback, it is often too short-sighted from a system's point of view, when the peer population is large. Instead, Rarest First is very effective in maximizing peer contribution as the population grows, hence produces good system-wide playback performance. On the other hand, the strength of Greedy is that it takes less buffer space, incrementally, to achieve higher continuity.

In trying to achieve the best of both worlds, we propose a new strategy, called the *mixed strategy*, which is a combination of Rarest First and Greedy. In the following subsections, we derive analytical results to analyze and compare the performance of these strategies. The key is to model the selection function $s(i)$ for each case, substitute it into the probabilistic model, and derive the buffer state probability distribution.

A. Greedy Strategy

We first present the analysis of the Greedy Strategy. This strategy aims to fill the empty buffer location closest to the playback time first. The chunk selection function, $s(i)$, which is the probability of selecting $B(i)$, can be expressed as follows:

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=i+1}^{j=n-1} \Big(p(j) + (1 - p(j))^2\Big). \qquad (8)$$

Since the event that downloading does not occur for a buffer at position $B(j)$ (for $j > i$) is $\neg(W(k,j)H(h,j))$, hence, the probability of this event is:

$$\Pr[\neg(W(k,j)H(h,j))] \quad = \quad p_k(j) + (1 - p_k(j))(1 - p_h(j)). \qquad (9)$$

Eq. (8) is based on the event that the server selects other peers to upload, and the chunk selection does not occur for all those positions closer to the deadline than $B(i)$, with the buffer position independence assumption stated earlier. Note, the first term of Eq. (9) is the probability the local peer already has the chunk for $B(j)$. The second term is the probability that the local peer does not have the chunk for $B(j)$ and the selected peer (h) does not have that chunk either. The rather complicated formula for $s(i)$ (Eq. 8) has a surprisingly simple alternative form:

Lemma *1: The selection function $s(i)$ for the Greedy Strategy can be expressed as*

$$s(i) = 1 - (p(n) - p(i+1)) - p(1) \quad \textit{for } i = 1, ..., n-1.$$

The proof is presented in the Appendix. Intuitively, it can be understood as follows. The term $(p(n) - p(i+1))$ is the probability that any particular chunk is downloaded into buffer positions between $B(n)$ to $B(i+1)$; and the term $p(1)$ is the probability that any particular chunk is downloaded directly from the server. The above expression for $s(i)$ is thus the probability that neither of these two scenarios are true.

Substituting the above formula for $s(i)$ into Eq. (6), we get the following *difference equation* for $p(i)$:

$$p(i+1) = p(i)+p(i)\Big(1-p(i)\Big)\Big(1-p(1)-p(n)+p(i+1)\Big) \quad \text{for } i = 1, \ldots, n-1. \tag{10}$$

B. Rarest First Strategy

The Rarest First Strategy is the opposite of the Greedy Strategy. Based on Eq. (3), we know $p(i)$ is an increasing function in i.[9] This means the expected rarest chunk is the *latest* chunk distributed by the server that is missing from the all local peers' buffer. So the chunk selection function $s(i)$ for the Rarest First Strategy can be expressed as:

$$s(i)=\Big(1-\frac{1}{M}\Big)\prod_{j=1}^{j=i-1}\Big(p(j) + (1 - p(j))^2\Big). \tag{11}$$

The meaning of each term is similar as before. The main point is that the search for missing chunks starts from the *latest chunk* $B(1)$, then to $B(2)$ and so on. Again, Eq. (11) has a simple form:

Lemma *2: The selection function $s(i)$ for the Rarest First Strategy can be expressed as*

$$s(i) \quad = \quad 1 - p(i).$$

The proof is presented in the Appendix. The rationale for this result is the same as that for the Greedy Strategy. The term $p(i)$ represents the probability that any particular chunk is downloaded into buffer positions $B(1)$ to $B(i-1)$. Therefore $s(i)$ as shown above represents the probability that this event does not occur.

Again, substituting $s(i)$ into Eq. (6), we have the following difference equation:

$$p(i+1) = p(i)+p(i)\Big(1-p(i)\Big)^2 \quad \text{for } i = 1, ..., n-1. \tag{12}$$

C. Buffer Size, Peer Population and Continuity

The difference equations for $p(i)$ in Eq. (10) and Eq. (12) help us express the relationships between the following key parameters:

- n, the buffer size;
- M, the population size (or equivalently $p(1)$, which is equal to $1/M$);
- $p(n)$, probability that $B(n)$ is available, which reflects the continuity and playback performance. For convenience, we also introduce $\epsilon = 1 - p(n)$ which simplify the expression of our results.

[9] In general, $p(i)$ is a non-decreasing function. But for both Greedy and Rarest First, $q(i) > 0$ for all buffer positions, so $p(i)$ is an increasing function.

To derive closed-form solutions, it is most convenient to consider the fluid form of Eq. (10) and (12) as continuous differential equations. We use the symbol y for $p(i)$ and the symbol x for i. This means:

$$y = p(i) \quad ; \quad dy = p(i+1) - p(i)$$
$$x = i \quad ; \quad dx = 1$$

The discrete equations now become:

$$\frac{dy}{dx} = \frac{y(1-y)(y-p(1)+\epsilon)}{1+y^2-y} \quad ; \quad \frac{dy}{dx} = (1-y)^2 y$$

respectively. Based on these equations, we obtain the following results:

Lemma *3: For the Greedy Strategy, the sensitivity of buffer size n to peer population M (or $p(1) = 1/M$) and discontinuity ϵ can be expressed as*

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{\epsilon p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon p(1)}. \tag{13}$$

Lemma *4: For the Rarest First Strategy, the sensitivity of buffer size n to peer population M and discontinuity ϵ can be expressed as*

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon^2} - \frac{1}{\epsilon}. \tag{14}$$

The proofs are included in the appendix.

Eq. (13) to (14) characterize the key difference between the Greedy and Rarest First Strategy. Due to the negative gradient of n relative to $p(1)$ and ϵ respectively, an immediate observation is that more buffer space is needed for larger peer population size M (or smaller $p(1)$), while other things (such as continuity) being held constant; similarly, more buffer space is needed for higher continuity (or smaller ϵ) while population is held constant. This is intuitive. Buffer size is directly proportional to the delay of playback relative to the source which we will refer to as *source delay*. Other papers have analyzed the relationship between population size, delay and throughput in p2p file downloading (e.g. [18]) which are consistent with our observation here.

The above equations also allow us to compare the Rarest First and Greedy strategies. For incremental increase in peer population, the need for additional buffer space when using the Rarest First Strategy is $1/\epsilon$ times less than that for the Greedy Strategy. This means that the Rarest First is more *scalable* than the Greedy strategy as the peer population increases.

On the other hand, for given peer population size, in order to increase continuity, the need for additional buffer space by the Greedy Strategy is $p(1)/\epsilon$ times less than that for the Rarest First. This means for sufficiently large $p(1)$ (hence sufficiently small M), the Greedy Strategy can achieve better continuity than Rarest First. This will be illustrated in Section IV.

The above observations are more formally summarized as follows.

Proposition *1:* Based on the p2p streaming model with large peer populations, asymptotically

1) As peer population increases, both the RF and Greedy strategies need larger buffers to maintain same continuity;
2) For incremental population increase, RF needs less buffer size to maintain continuity;
3) For given population size, Greedy can eventually achieve better continuity than RF for sufficiently large buffer size; and conversely, RF is better than Greedy given limited buffer size.

The proof, parts of it already evident from the discussion above, is included in the appendix.

D. Mixed Strategy

The intuition about the different strengths of the Greedy and Rarest First strategies lead us to propose a mixed strategy that can take advantage of both of these chunk selection algorithms.

Let the buffer B be partitioned by a point of demarcation m, $1 \le m \le n$. The Rarest First Strategy is used first with buffer spaces $B(1), ..., B(m)$. If no chunk can be downloaded using the Rarest First Strategy, then the Greedy Strategy is used with the other partition of the buffer, $B(m+1), B(m+2), ..., B(n)$. When $m = n-1$, the Mixed Strategy is the same as the Rarest First Strategy; when $m = 1$, the Mixed becomes the same as the Greedy Strategy. Through variation of m, a peer can adjust the download probability assigned for each partition.

The buffer state probability for $B(1)$ to $B(m)$ satisfies the following equations:

$$\begin{aligned} p(1) &= 1/M, \\ p(i+1) &= p(i) + p(i)(1-p(i))^2 \quad \text{for } i = 1, \ldots, m-1. \end{aligned}$$

The probability for $B(m+1)$ to $B(n)$ can be derived from Eq. (10) by substituting $p(1)$ with $p(m)$:

$$p(i+1) = \frac{p(i) + p(i)(1-p(i))(1-p(m)-p(n))}{1-p(i)(1-p(i))} \quad \text{for } i \ge m. \tag{15}$$

These equations can be solved numerically.

Recall at the end of Section II, we derived a way to compute an *upper bound* on continuity that can be achieved by any chunk selection strategy. This upper bound can help us prove an asymptotic notion of optimality for the Mixed Strategy. Assume the needed buffer length for different strategies is a function of discontinuity ϵ and number of peers M, that is $n = f(\epsilon, M)$:

Proposition *2:* For large peer population M, and small discontinuity ϵ, asymptotically, the Mixed Strategy is optimal in the sense that the most significant terms for its needed buffer size is the same as that needed by the strategy achieving the upper bound.

Proof: The proof is presented in appendix.

This result is rather surprising. The proof shows that Mixed can achieve the same order of required buffer length as that needed for the upper bound strategy[10], yet RF and Greedy cannot. In other words, Mixed always needs a smaller buffer than RF or Greedy to achieve a given continuity (or discontinuity ϵ).

Proposition *3:* For a given common buffer length, the continuity of the Mixed Strategy is asymptotically (large M and small ϵ) always better than that of Rarest First or Greedy. 5 better or same continuity than RF or Greedy;

Proof: The continuity $p(n)$ is an increasing function of buffer length n for all strategies. In Proposion 2, we proved that the Mixed Strategy can always achieve the same continuity as Rarest First or Greedy with fewer buffers. It therefore follows that Mixed can always use additional buffer space to achieve better continuity than Rarest First or Greedy. ■

The basic idea of the Mixed Strategy is to use the front part of the buffer, from position 1 to m, to implement the Rarest First Strategy to help distribute the content to as many peers as quickly as possible; and to use the tail part of the buffer, from position $m+1$ to n, to implement the Greedy Strategy to maximize continuity.

For given buffer length and population size, a good question is how to find the optimal m? This can be done by a brute force search, since there are only n possible values for m. In practice, there is an adaptive method to search for a suitable m in very few steps. This makes it easy to implement the Mixed Strategy even for dynamic peer populations. This point will be discussed in detail in the numerical examples section.

IV. Numerical Examples and Analysis

In this section, we consider a number of numerical examples to illustrate our results and their application to protocol design. For each numerical example, the results can be computed in the following ways:

Discrete model: The discrete model is given by the difference equations corresponding to the various chunk selection strategies (Eq. 1,3,5,8,11,15). The solution for the buffer state distribution $p(i)$ can be derived numerically. For the Greedy Strategy, we first give $p(n)$ a fixed value, substitute n steps inversely from $p(n)$ to $p(1)$ and then compare $p(1)$ with $1/M$. If $p(1)$ is approximately equal to $1/M$ then we get the solution; else $p(n)$ is adjusted accordingly and the inverse substitution process is repeated. For the Rarest First Strategy, substitute $p(i)$ from $p(1)$ until $p(n)$. For the Mixed Strategy, we compute the first part, from 1 to m, using the same substitution process as that for Rarest First and then compute what is left using the same trick as that for Greedy.

Continuous model: The continuous model is given by the differential equations in Eq. (10) and (12). In general, they can be solved numerically using MatLab. For some relationships, we also derived closed-form solutions.

[10]Of course, this is not exactly saying Mixed is optimal. What strategy is optimal is still an open problem.

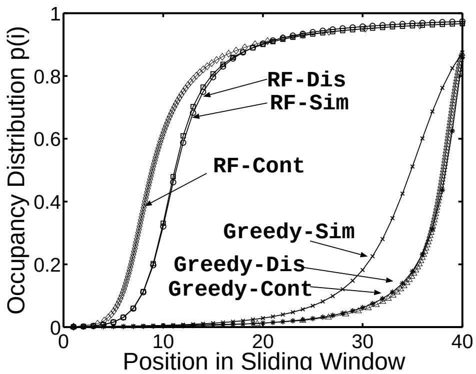


Fig. 2. Buffer occupancy distribution for Rarest First and Greedy policies from discrete, continuous and simulation models

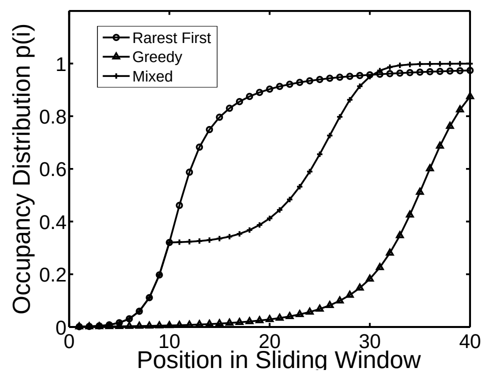


Fig. 3. Comparison of Rarest First, Greedy and Mixed

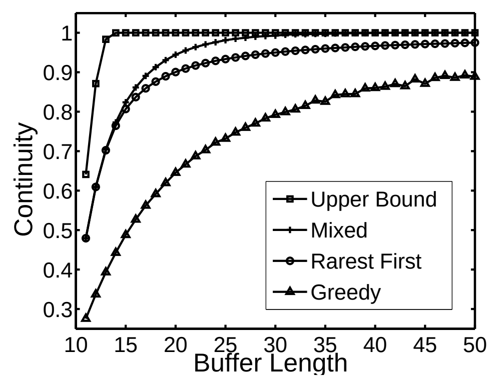


Fig. 4. Exp B. Continuity versus buffer size

Simulation model: We built a simulation program based on our discrete model. There is one server and M peers. In each time slot, the server distributes one chunk to a random peer; each peer randomly selects only one other peer to contact and download one chunk, but may upload at most two chunks to its neighbors. The peers form an overlay network where each peer is neighbor with a subset of the peers, randomly selected from the peer population. The values of various parameters, such as M, n, and average degree are specified as part of the description of the experiment. The simulation model is used to check to what extent the independence assumption may affect the analytical models, specially in the case with small peer population. Furthermore, simulation can produce a lot more details about specific peer behavior and the dynamics of the system including transient behavior.

Important Parameters: In most experiments, we set the peer population to 1000, which we think is large enough to validate our model. Actually, the larger the population is, the more accurate is our model. The choice of buffer length is based on the likely expectation that the achievable continuity is high ($> 99\%$) to enjoy a video. The minimum required buffer length derived from the formula of the upper bound is about 13 (for $M = 1000$). Therefore, it is reasonable to set the buffer length to three times this minimum size (which is 40) for most experiments.

Exp. A: Comparing Discrete and Continuous Results with Simulation

Our first task is to compare our discrete model, the continuous model based on the differential equation approximation, with simulation.

In this experiment, $M = 1000$ and $n = 40$. In the simulation, the number of neighbors for each peer is $L \leq 60$. The results are shown in Figure 2. There are two groups of curves, one for Greedy and one for Rarest First. In each group, there are three curves: one calculated using the discrete iterative equations, one calculated using the approximate continuous differential equations, and one from simulation.

We will compare Greedy and Rarest First (as chunk selection strategies) later on. At this point, let us focus on the accuracy of the different methods. First, we note that the analytical results are reasonably close to the simulation results. Secondly, we expect the discrepancy between the discrete model and simulation is mainly due to the independence assumption. For Greedy, there are fewer chunks in the buffers, hence the independence assumption is less accurate. Thirdly, we expect the discrepancy between the discrete and the continuous models is mainly due to the approximation of $p(i+1)-p(i)$ by a continuous gradient, which happens to have a bigger effect on the equation for Rarest First this time.

Exp. B: Comparing Rarest First, Greedy and Mixed

To compare the three chunk selection strategies, we keep the buffer size at $n = 40$; and set $m = 10$ for Mixed (this means the number of buffer positions running Rarest First is 10). The results (from the discrete model) are shown in Figure 3.

To compare the different strategies for different buffer sizes, we plot the continuity for buffer sizes between 20 and 50 in Fig 4. It is observed that Rarest First consistently beats Greedy in continuity. The reason is evident from our analysis and Fig 2. Rarest First works hard at distributing new chunks from the server, achieving a performance not far from the theoretical limit of $\log_2(i)$. The Greedy, however, is like a procrastinator, making a great effort to fill the buffers only near the playback time for each chunk.

From analysis earlier, we also know that Mixed can always outperform RF and Greedy. From Fig 4 we can see that when

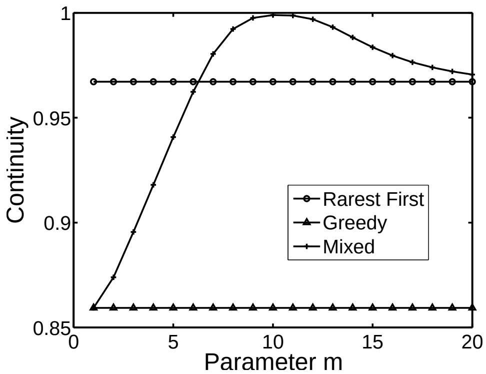


Fig. 5. Exp C. The effect of varying m on continuity of the Mix Strategy

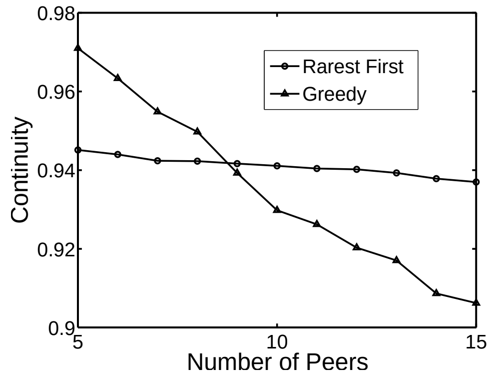
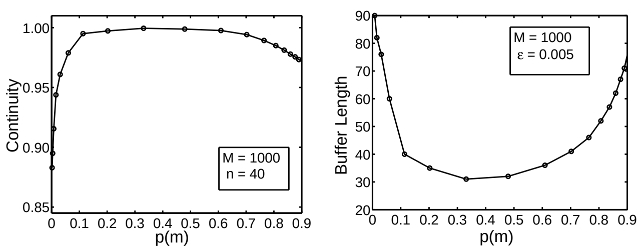
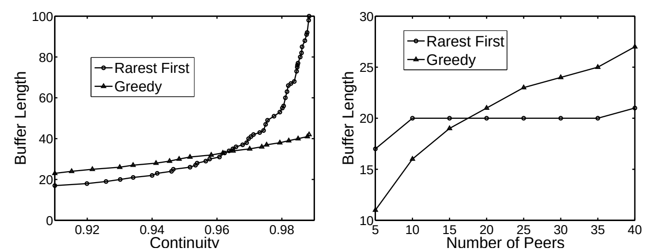


Fig. 7. A small network



(a) The effect of varying $p(m)$ on continuity of the Mix Strategy (b) The effect of varying $p(m)$ on buffer length of the Mix strategy

Fig. 6. $p(m)$ v.s. best Mixed strategy



(a) The small network with fixed peers (b) The small network with fixed continuity

Fig. 8. Second and Third Experiments in Exp. D.

the buffer length is larger than a threshold (around 25), the gap between Mixed and Upper bound becomes quite small.

Exp. C: Picking the optimal m in Mixed Strategy

We now take a closer look at the Mixed Strategy. In the last experiment, the parameter used to partition the buffer, m, is a constant. Here, we fix the buffer size to be 40 and vary m. The performance of continuity is plotted against m in Figure 5.

For continuity, it is quite interesting. There is an optimal m when continuity is maximized. These two plots show that there is a *knee*, occurring at $m \approx 10$ when a balance of high continuity is achieved.

Another way to view the Mixed strategy is the value of $p(m)$, which was discussed in the Mixed Strategy in last section. The value of m is used to partition of Rarest First and Greedy in the Mixed Strategy. In this numerical experiment, the number of peers is 1000 and the result is shown in Figure 6(a) and 6(b). In the first experiment, the buffer length is given 40, while the value of $p(m)$ varies. The continuity is not very sensitive for the varying $p(m)$. When $p(m)$ is approximately equal to 0.3, the continuity is best. In the simulation, we assume $p(m) = 0.3$. In the second experiment, the discontinuity is fixed at 0.5%, while $p(m)$ varies. The two figures show that continuity is not very sensitive when $p(m)$ or m varies. In the dynamic network, the value $p(m)$ is controlled to achieve good performance.

Exp. D: Performance for Small Scale Networks

Here, we test the relationship between buffer size, population and continuity, as studied in Proposition 1.

There are three examples in this experiment and the result in each case is derived from simulation (the analytical models are less accurate for small networks). Each result is calculated based on the average values of 3000 time slots.

In the first experiment, the number of peers in the network varies from 5 to 15 and each peer sets $n = 15$. We compare the continuity achieved by Greedy and Rarest First. Figure 7 shows that Greedy achieves better continuity when the number of peers is sufficiently few relative to the value of continuity (in this case 9), as we expect.

In the second and third experiment, we study network with small peer population. Though peer population in real system is much larger, the small network case is more appropriate for comparison of different chunk selection strategies. In the second experiment, the number of peers be fixed, $M = 40$. However, the peers have different quality requirements (denoted $1 - \epsilon$), and have to change their buffer length to meet the requirements. The result is shown in Figire 8(a).

In the third experiment, we let the peers' continuity requirement be fixed at 0.93, but the number of peers (M) vary from 5 to 40. In order to make sure the continuity is larger than 0.93, each peer has to enlarge its buffer if the number of peers increases. The result is shown in Figure 8(b).

The results from the above two experiments are consistent with Lemma 3 and 4, and Proposition 1, namely Greedy is able to provide a high quality requirement with less buffer length while Rarest First can provide good playback performance for

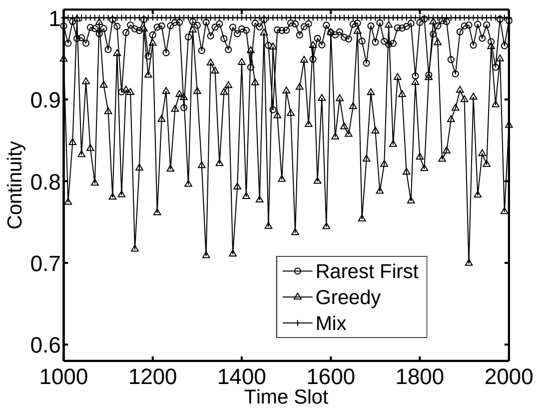
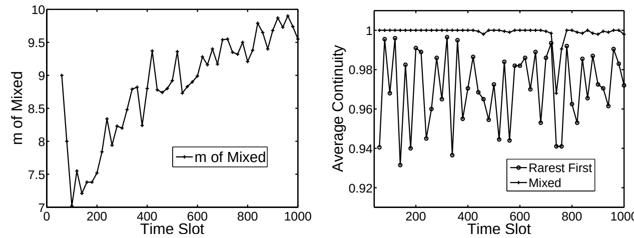


Fig. 9. Continuity of the Network Simulation



(a) Average continuity as a function of time (b) How m adapts to network dynamics

Fig. 10. Performance Results from Exp. F.

a large number of peers.

Exp. E: Study of Dynamics

While the analytical model is able to give us average steady state system behavior, simulation has the advantage of giving us the dynamic behavior of specific settings. In this experiment, we simulate the case of $M = 1000$ and $n = 40$, and look at how continuity evolves over time.

We compare the continuity achieved by different strategies. We simulate 2000 time slots. The data is taken from time slots 1000 to 2000 to capture the steady state conditions. In each time slot, the continuity is the average continuity of all peers, that is the number of peers being played chunks divided by total peers. As shown in Figure 9, Mixed not only achieves the best continuity, but its continuity is also much more steady than that of other two strategies.

Exp. F: Adapting the Mixed Strategy to Peer Population

Based on our analysis and the numerical examples, we show that the Mixed Strategy can achieve the best continuity given a fixed peer population size in the network. In reality, the peer population size is unknown and is likely to change over time. Here we describe an algorithm to adaptively adjust the Mixed Strategy's m to the network dynamics.

In the previous experiments, m is fixed (at 10). One way to adapt m is by observing of the value of $p(m)$. We can set a target value for $p(m)$, say $p_m = 0.3$. When a peer finds the average value of $p(m)$ is less than p_m, the peer increases m, else the peer decreases m. In our simulation, every peer calculates the average value of $p(m)$ for 20 time slots and then decides the value of m based the average value.

We conduct the following experiment. Let there be 100 peers in the network initially. After every 100 time slots, another 100 new peers with empty buffer are added to the network, which means there are $i \times 100$ peers in the network after $i \times 100$ time slots. For all the peers, the initial value of m is 10. We calculate the average continuity and average value of m for the initial 100 peers in the network as a function of time. From Figure 10(a) and 10(b), we observe that the average value of m (of the 100 tagged peers) adapts to the increasing peer population. Furthermore, the continuity of the Mixed Strategy is quite steady (except a glitch[11] between time slot 700-800) compared to that of Rarest First.

V. Robustness of the Model

For simplicity and tractability, we have made a number of assumptions in the P2P streaming model. It is important to understand the implication of these assumptions. In this section, we rely on simulation to study the robustness of the model, to look at what happens when some of the assumptions are violated.

A. Discrete Model with Fractional Bandwidth

One basic assumption in the model is about physical bandwidth constraints. It is assumed that there is enough bandwidth in the network to support the playback rate of all peers. In reality however, the bandwidth may be limited so that it is not sufficient to satisfy all peers' requirement. Assume the total playback rate is P and the total download rate of all peers is $f \times P$ and f is a real number in $(0, 1)$ modeling limited bandwidth. We show that in this case, only a small adjustment to the chunk selection function $s(i)$ is necessary, to keep our model still fairly accurate. Because of limited bandwidth, suppose each peer can only upload a chunk successfully with probability f. The server still pushes one chunk per time slot. For Greedy, $s(n-1)$ is changed to $f - \frac{1}{M}$ due to the limited bandwidth. Similar, for Rarest First, $s(1)$ is changed to $f - \frac{1}{M}$. Therefore, the corresponding chunk selection function for Greedy becomes $s(i) = f - p(1) - p(n) + p(i+)$ and that for for Rarest First becomes $s(i) = f - p(i)$. The resultant difference equations for the discrete model become: (16) and (17):

$$p(i+1) = p(i)+p(i)\Big(1-p(i)\Big)\Big(f-p(1)-p(n)+p(i+1)\Big) \quad \text{for } i = 1, \dots, n-1. \tag{16}$$

$$p(i+1) = p(i)+p(i)\Big(1-p(i)\Big)\Big(f-p(i)\Big) \quad \text{for } i = 1, \dots, n-1. \tag{17}$$

[11]Probabilistically speaking, there is always some chance that a peer with a new chunk does not get requested by other peers due to random peer selection, and this initial delay can unfortunately significantly affect the continuity of that chunk.

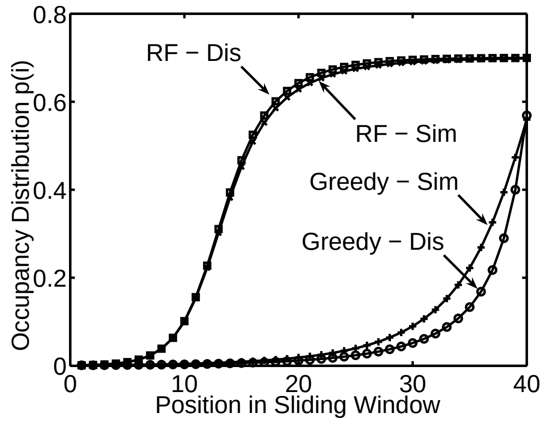


Fig. 11. Buffer occupancy distribution of the network with limited bandwidth

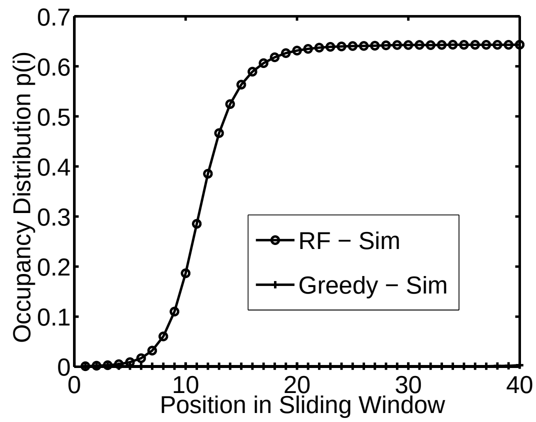


Fig. 12. Buffer occupancy distribution of the network when server uses pull strategy

The following experiment is designed to validate our discrete model with a fractional of the bandwidth requirement. In the simulation experiment, there are 1000 peers. Each peer has a buffer with length 40. Set the fraction of bandwidth support to $f = 0.7$. We run separate experiments using the Greedy Strategy and Rarest First Strategy, and compare them with the results computed from the discrete model (Eq. (16) and (17)). Figure 11 shows the modified model is quite accurate.

B. Server Using Pull Strategy

In our model, the server is assumed to *push* the newest chunks to peers. One question is whether it is possible to do away with this asymmetry between the server and peers complete, and let the peers *pull* the chunks from the server. A simulation experiment is carried out to observe the performance when the server stops *pushing*. Again let there be 1000 peers in the network and buffer length be 40 for each peer. Figure 12 shows the result. The Rarest First strategy is still able to perform reasonably well, although continuity reduced by about 20%. But for Greedy, the P2P mechanism becomes completely ineffective. Each peer's continuity reduces to $1/M$, as if there is no P2P support. The result indicates the assumption that server uses push is necessary.

C. Vary Size of Server Fan-out

In the original model, we assume the server randomly pushes out the newest chunk to the whole network of peers. In reality however, the server may only be able to push to a subset of the peers. To study this situation, we changed the simulation to allow the server to only work with a subset of peers in its push. The effect of different sizes of the subset is shown in Figure 13. When the subset size is greater than a relatively small threshold, in this case 40 for a total population size of 10000, the curve has become quite flat. The implication of this experiment result is that our assumption that the server connects to all peers is justifiable. In real P2P streaming networks, having the server connect to all peers is not implementable, but the same performance can be achieved by connecting to $40 - 60$ peers.

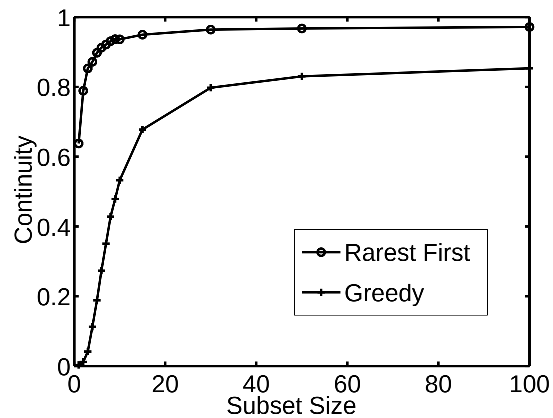


Fig. 13. Buffer occupancy distribution of the network when server talks with a subset

VI. Brief Discussion of Related Works

In P2P content distribution, practice is currently leading theory. A number of operational or experimental P2P systems have been developed and successfully deployed for file sharing [3], live streaming [4], [16] and Video-on-Demand streaming [11]. Following the success of these systems, there is significant interest in modeling and analyzing how such systems work, and in understanding the underlying factors that these systems depend on.

One important question studied by the theoretical papers is the capacity of the P2P system for disseminating content to a population of peers, irrespective of whether the overlay P2P network is structured or unstructure. The general answer is related to that of the max-flow problem, which can be complicated depending on the network topology. Under the *uplink sharing model* assumption and for large peer population, [6] derived a closed-form upper bound, for static populations. Separately, [14], [15], [17] studied the problem for dynamic peer population. Following [17], [12] studied design trade-off between system throughput and contribution fairness, indicating the price for achieving optimal capacity will be uneven contribution. There are a number of extensions to the capacity bound, taking into consideration of degree limits, availability of helpers and other factors, and these references are not listed here separately.

Once we know the limit, the important remaining question is how to achieve the limit. In this regard, there are a number of studies of distributed algorithms based on the unstructured approach, notably [5], [9], [13], [18]. Out of these studies, [9] was the paper from which the current paper is derived. The other three papers all studied various chunk selection algorithms for P2P content distribution. All these papers make the same abstraction: that the peer to peer content exchanges occur in slotted time. [9] assumes a *pull* method: a peer finds another peer to download a chunk of content. The other three papers assume a *push* method: a peer pushes a chunk of content to another selected peer. In most cases, the selection of a neighbor (to pull or push a chunk) is random. In the case of pull, there is some chance that two or more peers try to pull from the same target; in the case of push, there is the chance that two or more peers try to push to the same peer. In both cases, the problem can be avoided either by assuming peers are omniscient and try to avoid such collisions, or by assuming the number of peers is large so that such collisions occur rarely and it can be assumed they don't occur.

The work from [9] and the other three papers reach some similar, but also some different conclusions. This is because they define different metrics. In [5], [13], [18], the authors define *diffusion rate* and (source) *delay* as general metrics for content distribution. These metrics are not specifically targeted at file downloading or streaming. Asymptotically, they are important goals for any content distribution mechanism. These papers proceed to prove that certain P2P algorithms can achieve optimal diffusion and optimal delay. Out of these optimal algorithms, some require global knowledge, which implies potentially high message exchange overheads. Most amazingly, it is shown that a simple chunk selection algorithm (essentially corresponding to the *rarest first*) with random peer selection is proven to be optimal for both diffusion rate as well as delay.

The model and metric in [9], however, specifically targets P2P streaming. The model incorporates buffers from each peer, and each P2P algorithm yields a different steady state buffer state distribution. The metric to optimize is defined as the *continuity*, or the percentage of peers able to play back the content from its buffer (of fixed size). Based on this model, [9] is able to conclude that *rarest first* alone is usually not optimal; you can do better by devoting part of the buffer to fetching chunks that are more urgent due to the deadline for playback. This is the first successful effort, to the best of our knowledge, to model and study P2P streaming algorithms analytically[12].

[12]To be fair, from a practical perspective, two other works on P2P streaming that preceeded [9] are very influential. One is Coolstreaming [4] that first demonstrated convincingly by experiments that P2P streaming based on unstructure algorithms can work; the other is BiTos [16] that showed a mixed strategy works well, although their mixed strategy is somewhat different that that in [9] and there was no analysis in [16] to back the idea up.

VII. Conclusion

The art of modeling is on the one hand to capture the essential aspects of the original system, and on the other hand to be simple enough to yield some insights about the original system. We feel that is what our model accomplished for the P2P streaming problem. In addition, the insights from our model also lead to some practical algorithm that can be incorporated into well established systems as improvements.

There are a number of interesting directions for further studies. We believe the simple probability model can be extended to analyze other chunk selection and peer selection algorithms. The buffer requirements for p2p streaming is not the focus of this study, and can certainly be more thoroughly analyzed. Finally, whether there exist an *optimal* strategy is still an open problem.

VIII. Acknowledgement

We would like to thank Zhao Qiao who provided the idea about the Upper Bound in Section II. We would also like to thank Prof Bruce Hajek, who asked interesting questions and made valuable suggestions for many of the extensions of this paper from its original conference version.

References

[1] PPStream, "http://www.ppstream.com/."
[2] PPLive, "http://www.pplive.com/."
[3] BitTorrent, "http://www.bittorrent.com/."
[4] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *Proc. INFOCOM*, 2005.
[5] T. Bonald, L. Massoulie, F. Mathieu, D. P. Andrew, and Twigg, "Epidemic live streaming: Optimal performance trade-offs," in *ACM Sigmetrics*, 2008.
[6] J. Mundinger, R. Weber, and G. Weiss, "Optimal scheduling of peer-to-peer file dissemination," in *Preprint version in arXiv*, 2006.
[7] Z. Qiao, J. Lui, and D. Chiu, "Exploring the optimal chunk selection policy for data-driven p2p streaming systems," in *P2P*, 2009.
[8] B. Fan, D. M. Chiu, and J. C. Lui, "The delicate tradeoffs in bit torrent-like file sharing protocol design," in *ICNP*, 2006.
[9] Z. Yipeng, D. Chiu, and J. C. Lui, "A simple model for analyzing p2p streaming protocols," in *ICNP*, 2007.
[10] Y. Wang, T. Z. Fu, and D.-M. Chiu, "Analysis of load balancing algorithms in p2p streaming," in *Allerton Conference*, 2008.
[11] Y. Huang, T. Z. Fu, D. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p vod system," in *ACM Sigcomm*, 2008.
[12] B. Fan, D.-M. Chiu, and J. C. Lui, "The delicate tradeoffs in bit torrent-like file sharing protocol design," in *ICNP*, 2006.
[13] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *Infocom*, 2007.
[14] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for p2p streaming systems," in *Infocom*, 2007.
[15] L. Massoulie and M. Vojnovic, "Coupon replication systems," in *ACM Sigmetrics*, 2005.
[16] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: Enhancing bit-torrent for supporting streaming applications," in *INFOCOM 25th IEEE International Conference on Computer Communications. Proceedings*, 2006.
[17] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *ACM SIGCOMM*, 2004.
[18] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping with multiple messages," in *IEEE Transactions on Information Theory*, 2007.

Appendix

Proof of Lemma 1: From Eq. (6), we have

$$p(i+1) - p(i) = s(i)p(i)\Big(1 - p(i)\Big).$$

From Eq. (8), we have

$$s(i+1) - s(i) = s(i+1)p(i+1)\Big(1 - p(i+1)\Big).$$

Note the right-hand-side of the above two equations are the same, except the index i versus $i+1$. This means

$$\begin{aligned} s(i+1) - s(i) &= p(i+2) - p(i+1), \\ \sum_{j=i}^{n-2}(s(j+1) - s(j)) &= \sum_{j=i}^{n-2}(p(j+2) - p(j+1)), \\ s(i) &= s(n-1) - p(n) + p(i+1). \end{aligned}$$

From the equation of $s(i)$ (Eq. 8), we get $s(n-1) = 1 - 1/M$. Therefore, we have $s(i) = 1 - p(1) - p(n) + p(i+1)$. ∎

Proof for Lemma 2: Again, from Eq. (6), we have

$$p(i+1) - p(i) = s(i)p(i)\Big(1 - p(i)\Big).$$

From Eq. (11), we have

$$s(i+1) - s(i) = s(i)p(i)\Big(p(i) - 1\Big).$$

This time, the right-hand-side of these equations are again the same except the sign (and index off by 1). This gives us

$$\begin{aligned} s(i+1) - s(i) &= -\Big(p(i+1) - p(i)\Big), \\ \sum_{j=0}^{i-1}(s(j+1) - s(j)) &= -\sum_{j=0}^{i-1}\Big(p(i+1) - p(i)\Big), \\ s(i) &= s(1) + p(1) - p(i). \end{aligned}$$

When there are M peers in the network, $p(1) = 1/M$, which is the probability the sever selects it for sending the newest chunk. From Eq. (11), we have $s(1) = 1 - 1/M$. Therefore, we have $s(i) = 1 - p(i)$. ∎

Proof of Lemma 3: Assume $\epsilon = 1 - p(n)$ and $\epsilon - p(1) \neq 0$, which covers all the chunk selection strategies we are interested in. We get the following solution for the differential equation:

$$\begin{aligned} x &= \frac{\ln\left(\frac{y}{y+\epsilon-p(1)}\right)}{\epsilon - p(1)} + \frac{\ln\left(\frac{y+\epsilon-p(1)}{1-y}\right)}{1+\epsilon-p(1)} \\ &\quad - \ln(y+\epsilon-p(1)) - C. \end{aligned}$$

Here C is a constant that can be derived from the boundary condition $y = p(1) = 1/M$:

$$C = \frac{\ln(\frac{p(1)}{\epsilon})}{\epsilon - p(1)} + \frac{\ln(\frac{\epsilon}{1-p(1)})}{1+\epsilon-p(1)} - \ln(\epsilon) - 1.$$

Solving the above equation, we can express n, the buffer size, in terms of the other parameters $p(1)$ and ϵ:

$$n = \frac{\ln\left(\frac{(1-p(1))p(1)}{(1-\epsilon)\epsilon}\right)}{p(1)-\epsilon} + \frac{2\ln\left(\frac{1-p(1)}{\epsilon}\right)}{1+\epsilon-p(1)} + 1 + \ln\left(\frac{\epsilon}{1-p(1)}\right).$$

Although n is an integer, we can still study its sensitivity with respect to $p(1)$ and ϵ by differentiation, which yields the results in the Lemma. ∎

Proof of Lemma 4: With a similar method as in the proof for Lemma 3, we derive the solution for the differential equation for the Rarest First algorithm:

$$\begin{aligned} x &= \frac{1}{1-y} + \ln\left(\frac{y}{1-y}\right) - C, \\ C &= \ln\left(\frac{p(1)}{1-p(1)}\right) + \frac{p(1)}{1-p(1)}. \end{aligned}$$

Again, $p(1)$ and ϵ represent the number of peers and the streaming quality respectively, and $y(n) = 1 - \epsilon$. Similarly, we express n as a function of $p(1)$ and ϵ:

$$n = \frac{1}{\epsilon} + \ln\left(\frac{1-\epsilon}{\epsilon}\right) - \ln\left(\frac{p(1)}{1-p(1)}\right) - \frac{p(1)}{1-p(1)}.$$

Differentiating, we get the results in the Lemma. ∎

Proof of Proposition 1 The proofs for part (1) and (2) follow directly from Lemma 3 and 4.

The proof for (3) can be derived by going back to the differential equations of the continuous model. We prove it in three steps. First, a special buffer length n_s is found, where the discontinuity $\epsilon_G(n_s)$ is less than $\epsilon_{RF}(n_s)$. Secondly, we show the buffer required to satisfy incremental continuity requirement beyond n_s is less for Greedy, which means the Greedy Strategy beats Rarest First beyond the special buffer length n_s. Thirdly, we compare $\frac{\partial n}{\partial \epsilon}$ from the beginning point $n = 1$, to support the statement: Rarest First is better when buffer length is limited.

First step: M is given. Assume a target discontinuity ϵ_s such that $\epsilon_s = p(1) = 1/M$. This simplifies the differential equation for Greedy to the following:

$$\frac{dy}{dx} = \frac{y^2(1-y)}{1+y^2-y}$$

This equation can be solved to yield the solution:

$$\begin{aligned} x &= -\frac{1}{y} - \ln(1-y) - C, \\ C &= -\frac{1}{p(1)} - \ln(1-p(1)) - 1 \end{aligned}$$

Substituting $\epsilon_s = p_1$ back, the needed buffer length for this value of ϵ_s is:

$$\begin{aligned} n_G &= \frac{1}{\epsilon_s} + \ln(\frac{1-\epsilon_s}{\epsilon_s}) - \frac{\epsilon_s}{1-\epsilon_s} \\ &\doteq n_s \end{aligned}$$

The continuous differential equation for RF is not simplified, but can be solved to yield:

$$\begin{aligned} n_{RF} &= \frac{1}{\epsilon_s} + 2\ln(\frac{1-\epsilon_s}{\epsilon_s}) - \frac{\epsilon_s}{1-\epsilon_s} \\ &> n_s \end{aligned}$$

Because the function $p(n)$ is an increasing function in n, therefore the discontinuity $\epsilon_{RF}(n_s)$ is greater than $\epsilon_G(n_s) = p(1)$. This ensures Greedy out-performs Rarest First for all buffer lengths greater than n_s.

Second Step: For buffer lengths beyond n_s, the approximate absolute value of $\frac{\partial n}{\partial \epsilon}$ in equation Eq. (13, 14) becomes:

$$|\frac{\partial n}{\partial \epsilon}| \approx \frac{1}{\epsilon \times p(1)}, \quad for\ Greedy$$

$$|\frac{\partial n}{\partial \epsilon}| \approx \frac{1}{\epsilon^2} + \frac{1}{\epsilon}, \quad for\ RarestFirst$$

The value of ϵ for buffer length beyond n_s is less than $p(1)$; therefore $|\frac{\partial n}{\partial \epsilon}|$ for Greedy is less than that for Rarest First, which means Greedy consumes less buffer length for the same incremental continuity requirement beyond n_s. Based on step 1 and 2, the conclusion is that Greedy achieves better continuity if buffer length is large enough.

Third Step: If the buffer length is very limited, which means ϵ is much bigger than $p(1)$. By the same argument as that in the second step, $|\frac{\partial n}{\partial \epsilon}|$ for Greedy is larger than that for Rarest First, which means Greedy consumes more buffer length for the same incremental continuity requirement. Both Greedy and Rarest First start from $n = 1$ with the same continuity $p(1) = \frac{1}{M}$. Therefore, Rarest First is better when buffer is very limited. ∎

Proof of Proposition 2: Based on the definition of Upper Bound chunk selection function at the end of Section II (that is $s(i) = i$ for all i), we can write down the corresponding differential equation for it, and derive the following solution:

$$\begin{aligned} x &= \ln(y) - \ln(1-y) - C, \\ C &= \ln(\frac{p(1)}{1-p(1)}) - 1 \end{aligned}$$

$$n_{UB} = \ln(\frac{1-\epsilon}{\epsilon}) - \ln(\frac{p(1)}{1-p(1)}) + 1$$

We now compare n_{UB}, the needed buffer length for Upper Bound, with n_{Mixed}, n_{RF} and n_G, the corresponding buffer length requirements for Mixed, RF and Greedy, based on their most significant terms. For n_{UB}, it is $O(\ln(\frac{1}{\epsilon})) + O(\ln(M))$. From the proof of lemma 4, n_{RF} is $O(\frac{1}{\epsilon}) + O(\ln(M))$, while n_G is $O(\frac{1}{p(1)-\epsilon}(\ln(\frac{1}{\epsilon}) + \ln(M)))$. So the *order* of n_{RF} and that of n_G are both larger than that of n_{UB}. However, for the Mixed Strategy, the Rarest First part is given a relative large discontinuity and the Greedy part is given a relative large $p(1)$. Assume the continuity for the Rarest First part is λ, or $p(m) = \lambda$. This means the order of n_{Mixed} is $O(\frac{1}{1-\lambda} + \ln(M) + \frac{1}{\lambda-\epsilon}(\ln(\frac{1}{\epsilon}) + \ln(M)))$.

In the Mixed Strategy, λ is controlled by varying the buffer length of the Rarest First Strategy. The maximum λ we can get is $p(m)$, which is the continuity of Rarest First Strategy with buffer length m. If the desired value for λ is not close to 1, we show that it can be achieved by picking m from a narrow range of values for any M in a large range of values. From the proof of lemma 4, we have a closed-form solution of the buffer length n for RF, as a function of M and λ. Consider the regime when $p(1) \approx 0$, this function is simplified to:

$$M = e^{n - \ln\frac{1-\lambda}{\lambda} - \frac{1}{\lambda}}$$

If λ is picked to be not close to 0 and 1, $\ln\frac{1-\lambda}{\lambda} - \frac{1}{\lambda}$ is relative small compared with n. This means M from a large range of values can be satisfied using n from a narrow range of values.

Let us go back to the expression for n_{Mixed} above. Since for almost any M we can easily pick m to make λ a constant, the order of n_{Mixed} becomes $O(\ln(\frac{1}{\epsilon})) + O(\ln(M))$, which is the same as that of n_{UB}. ∎