A Simple Model for Analyzing P2P Streaming Protocols

Yipeng Zhou
Information Engineering Department
The Chinese University of Hong Kong

Dah Ming Chiu
Information Engineering Department
The Chinese University of Hong Kong

John C.S. Lui
Computer Science Department
The Chinese University of Hong Kong

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 data-driven downloading strategies based on two performance metrics: continuity (probability of continuous playback), and startup latency (expected time to start playback). 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, whereas the latter an intuitively reasonable strategy to optimize continuity and startup latency from a single peer's viewpoint. Greedy, while achieving low startup latency, fares poorly in continuity by failing to maximize P2P sharing; whereas Rarest First is the opposite. This highlights the trade-off between startup latency and continuity, and how system scalability improves continuity. Based on this insight, we propose a mixed strategy that can be used to achieve the best of both worlds. Our algorithm dynamically adapts to the peer population size to ensure scalability; at the same time, it reserves part of a peer's effort to the immediate playback requirements to ensure low startup latency.

I. Introduction

Video streaming over the Internet is already part of our daily life. 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. The earlier vision for solving this problem is based on IP multicast, which relies on the routers in the network to manage the distribution and duplication of content from one source to multiple receivers. Due to technical complexity and other deployment issues, IP multicast has not been widely deployed. Instead, what emerged is a form of multicast implemented by an overlay network. There are different types of overlay networks, but a peer-to-peer (P2P) overlay network proves to be especially scalable. In a P2P network, each client is also a server (when the P2P network is working well), thus when more clients join a multicast session more servers (peers themselves) are automatically added to share the additional load.

The earlier work on P2P content distribution was known as *application layer multicast* [4] or *end-host multicast* [6]. Since then, there has been a significant body of work on P2P streaming. In an invited paper [7], the existing approaches are classified into two categories: one is *tree-based*, the other is *data-driven*. Both tree-based and data-driven use multiple paths (i.e. multiple spanning trees) for distributing content from a source to each receiver, which is the key for achieving scalability. The data-driven approach [5], [12], [15], [16], by not focusing on trees explicitly, allows the distribution paths to be determined based on data availability, which can adapt to the dynamics of a P2P network.

Another important contributor to P2P streaming is the body of work on P2P file sharing protocols. The most representative and most influential work (in academic circles) is BitTorrent (BT) [2], [3]. P2P file sharing is subtly different from P2P streaming. On the one hand, it is less demanding since it does not have real-time requirements; but on the other hand, it is also more demanding because it requires the entire file (in P2P streaming, peers join the video session from the points determined by their arrival times). Nevertheless, both P2P file sharing and P2P streaming need to deal with scalability by connecting the peers together to serve each other, and the works on BT provided the necessary insight in this area.

The contribution of the paper is as follows. None of the studies on P2P streaming so far, to the best of our knowledge, has formulated a tractable analytical model to help understand the important system level design issues in P2P streaming - this is the contribution of this paper. By assuming independent and homogeneous peers (using the same size playback buffer and chunk selection strategy) in a symmetric network setting, we construct a simple analytical model that allows us to compute the distribution of what each peer has in its buffer. We can use this model to evaluate and compare a variety of *chunk selection strategies*, which is the *core* of the data-driven approach. Based on a simple model, one can understand the relationships of important system parameters and metrics. In particular, we first study two strategies: Rarest First and Greedy. We show that Rarest First is much better in dealing with scale, whereas Greedy is able to produce better playback performance (continuity) in small scale networks. Also, if all peers use Greedy, the playback delay can be smaller. We also prove an important property of our model, that is a certain number of buffer spaces used together with the Rarest First strategy can convert a large peer population problem into a much smaller peer population problem with equivalent playback performance. This insight allows us to propose a mixed strategy where a part of the buffer space is used to deal with the need for scalability, and the other part of the buffer

space is used to achieve the best playback performance and delay.

There are a large number of papers on various aspects of the data driven approach of P2P streaming, e.g. [1], [5], [9]–[16]. The two most closely related to our work are CoolStreaming [16] and BiTos [12].

CoolStreaming [16] is a very important prior study on data-driven P2P streaming protocols because it is based on a real prototype implementation and a relatively large scale experiment (involving thousands of simultaneous peers) in the real Internet. It serves as a proof of concept, and a benchmark for a real working system. Our model captures the main ingredients of the CoolStreaming system while stays simple enough for analysis. The chunk selection strategy, Rarest First (originally from BitTorrent), is one of the basic algorithms we model. The playback performance derived from our model matches closely to that observed in CoolStreaming's experimental results. Our abstract model allows us to consider different chunk selection strategies and gain insight into the trade-off of different metrics. In the end, we propose a better chunk selection strategy and explain why it is better.

Another interesting data-driven P2P streaming study is BiTos [12]. BiTos is also based on BitTorrent. In BiTos, the chunk buffer is divided into two parts, one part for high priority chunks and the other for lower priority chunks. As playback deadline nears, a low priority chunk (still missing) becomes high priority. A peer downloads high priority chunks with probability p, and downloads lower priority chunks with probability $1-p$. For each part of the buffer, BiTos still adopts the Rarest First Strategy. This is somewhat similar to the mixed strategy we study, although there are important differences. [12] provides no modeling and analysis of the chunk selection strategy, and little experimentation to show the advantages and disadvantages. All these issues are dealt with in this paper. In fact, BiTos can also be analyzed by our model; but based on our theory, our mixed strategy should be superior to BiTos.

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 V describes application of our protocol to real protocol design, and the conclusion is given in Section VI.

II. **Basic Model**

In this section, we present the mathematical model for P2P streaming applications. Let us first define the notations and assumptions.

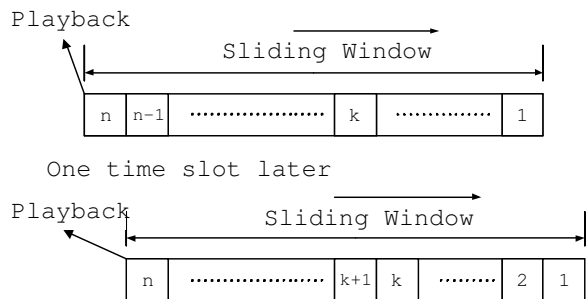Let there be M peers in the network[1]. There is a single

Fig. 1.   Sliding Window Mechanism of the buffer B

server which streams chunks of (video) content, in playback order, to the M peers. Each chunk has a sequence number, starting from 1. Time is slotted and the server selects a peer randomly in time slot t and sends chunk t to that peer.

Each peer maintains a buffer B that can cache up to n chunks received from the network. We reference 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 slot as possible, so as to support the continuous video playback and reduce the frame loss probability.

Let $p_k(i)[t]$ denote the probability that the ith 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 kth peer[2].

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 (from the server). The playback performance, given by $p(n)$, is equal to $\frac{1}{M}$, would obviously be very poor for any $M > 1$. This simple mathematical argument shows the scalability

problem when the server is the only means of distributing the media.

To improve playback performance, peers help each other when asked. We model the 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, i.e., 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, and this type of assumption is also made in other P2P file sharing models, i.e., [8].

Furthermore, we assume homogeneous peers, namely, all peers use the same strategy to select other peers and chunks to download. 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 do not consider 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. Peer selection has implications on system performance and peers' incentive to contribute [3]. Since the focus of this paper is on the performance of P2P streaming systems, we focus on the case that peers are homogeneous and adopt the same (random) peer selection strategy.

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 the base 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:

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

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.

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 [2], [3], and P2P streaming protocol CoolStreaming [16]. 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 B, else peer k will select $B(2)$ to download if $B(2)$ is not in the system 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 select $B(n)$ to download if it is not available in B, else peer k will select $B(n-1)$ to download if $B(n-1)$ is not in 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, Greedy is good in minimizing the start-up latency.

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:

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=i+1}^{j=n-1} \left(p(j) + (1-p(j))^2\right). \tag{7}$$

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:

$$\Pr[\neg(W(k,j)H(h,j))] = p_k(j) + (1-p_k(j))(1-p_h(j)). \tag{8}$$

Eq. (7) models 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. (8) 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. 7) has a surprisingly simple alternative form:

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

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

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 $p(i)$:

$$p(i+1) = p(i) + p(i)\Big(1 - p(i)\Big)\Big(1 - p(1) - p(n) + p(i+1)\Big)$$
$$\text{for } i = 1, \ldots, n-1. \tag{9}$$

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.[3] 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:

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=1}^{j=i-1} \Big(p(j) + (1-p(j))\Big)\Big(1 - p(j)\Big). \tag{10}$$

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. (10) has a simple form:

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

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

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.

---

[3]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.

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

$$p(i+1) = p(i) + p(i)\Big(1-p(i)\Big)^2 \quad \text{for } i = 1, ..., n-1. \quad (11)$$

C. Buffer Size, Peer Population and Continuity

The difference equations for $p(i)$ in Eq. (9) and Eq. (11) help us to derive closed-form solutions of the distribution $p(i)$. Also, the model allows us to derive some relationships between the key performance metrics and design parameters of the streaming system, these parameters are:

- 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 (or $\epsilon = 1 - p(n)$ is the probability of discontinuity).

To facilitate the derivation of these relationships, we convert the difference equations of Eq. (9) and (11) into continuous differential equations. They 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. The symbol y stands for $p(i)$ and the symbol x corresponds to i in the discrete case. These continuous differential equations can be derived by substituting dy/dx for $\frac{p(i+1)-p(i)}{1}$ and y for $p(i)$. Based on these equations, we obtain the following *sensitivity* relationships among these parameters:

Proposition *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*

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{\epsilon p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon p(1)}. \quad (12)$$

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

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon^2} - \frac{1}{\epsilon}. \quad (13)$$

The proofs are included in the appendix.

Eq. (12) to (13) characterize the key difference between the Greedy and Rarest First Strategy. These results indicate that more buffer space is needed for larger peer population size M (or smaller $p(1)$), and higher continuity (or smaller ϵ). This is due to the negative gradient of n relative to $p(1)$ and ϵ respectively. But as peer population grows, the need for additional buffer space when using the Rarest First Strategy is $1/\epsilon$ times less than that for the Greedy Strategy, which means that the Rarest First is more *scalable* than the Greedy strategy as the peer population increases. On the other hand, in order to increase continuity, the need for additional buffer space by the Greedy Strategy is about $p(1)/\epsilon$ 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.

D. Mixed Strategy

The intuition about the different strengths of the Greedy and Rarest First strategies derived from our model 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 Strategy, then the Greedy Strategy is used using 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:

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

The probability for $B(m + 1)$ to $B(n)$ can be derived from Eq. (9) by substituting $p(1)$ with $p(m)$:

$$\begin{aligned} p(i + 1) &= p(i) + p(i)(1 - p(i)) \\ &\times (1 - p(m) - p(n) + p(i + 1)). \quad (14) \end{aligned}$$

Another perspective that helps us to understand the advantage of the mixed strategy is the following observation about the equivalence between peer population size M and buffer size n. Consider two P2P networks. The first is a *reference network* with population M, buffer size n and some chunk selection strategy that yields buffer state distribution $p(i)$. The second is a *baby network* with a fraction of the population size equal to $1/p(m)$ and buffer size $n - m$, that uses the same chunk selection strategy as that used for buffer positions $B(m + 1)$ to $B(n)$ in the reference network. Let the buffer state distribution of the baby network be denoted $p'(i)$ for $i = m + 1, \ldots, n$. We have the following result.

Proposition *5: The continuity for the reference network, $p(n)$, is equal to the continuity for the baby network, $p'(n-m)$.*

Proof: Due the same chunk selection strategy used, $q(i)$ in the reference network is the same as $q'(i - m)$ of the baby network[4]. This means $p(i) = p'(i - m)$, for $i = m + 1, \ldots, n$, hence $p(n) = p'(n - m)$. ∎

The implication of this proposition is that we should use a mixed strategy, whenever the peer population size M relative

---

[4] As with the rest of the results in our model, this relies on the independence assumption to be true.

to the desired playback performance (continuity) is larger than a threshold (given by $p(1)/\epsilon > 1$). For the *baby network* part of the buffer positions, we used the Greedy Strategy to maximize continuity. For the rest of the buffer positions, Rarest First is used as it is the more economical strategy (in terms of buffer space needed) to support a large peer population.

E. Start-up Latency

So far we have focused on continuity $p(n)$ as the performance metric for evaluating various chunk selection strategies. From Eq. (3) and by defining $q(0)=p(1)$,[5] we have:

$$p(n) = \sum_{i=0}^{n-1} q(i).$$

Another metric worth paying attention to is the *start-up latency*, which is the time a peer should wait before starting playback. As long as all peers cooperate by following the same chunk selection strategy and offering downloading when requested, a peer may choose to start its own playback independently without affecting other peers except itself. But what is the *best* start-up latency for a newly arriving peer (with empty buffer) to choose, assuming all the other peers have already reached steady state? We argue each peer should wait until its buffer has reached steady state, which means:

$$\text{startup latency} = \sum_{i=1}^{n} p(i)/R. \qquad (15)$$

where R is the average downloading rate of chunks experienced by the newly arriving peer. Since all other peers are in their steady state, R should be the same as the average steady state downloading rate, which must also equal to the effective playback rate. For all the chunk selection strategies we are interested in, the effective downloading rate must be close to 1 (chunk per time slot), the video's playback rate. Therefore we have

$$\text{startup latency} \approx \sum p(i)$$

Why is the quantity in the above equation a good representation for startup latency? When a peer starts with an empty buffer, every peer it contacts is likely to result in a successful download. After $\sum_{i=1}^{n} p(i)$ time slots, the newly arriving peer is expected to have acquired the same number of chunks as the rest of the peers in steady state, which also equals to $\sum_{i=1}^{n} p(i)$. If the newly arriving peer starts with earlier, it is likely to suffer from below steady state playback quality initially. If the newly arriving peer waits longer (than that in Eq. (15)), it will not improve its long-term steady state playback quality.

[5]By defining $q(0) = p(1)$, we are treating the buffer update from server the same as updates from peers. This is just for convenience.

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. 1,3,5,7,10,14). 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. (9) and (11). In general, they can be solved numerically using MatLab. For some relationships, we also derived closed-form solutions.

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.

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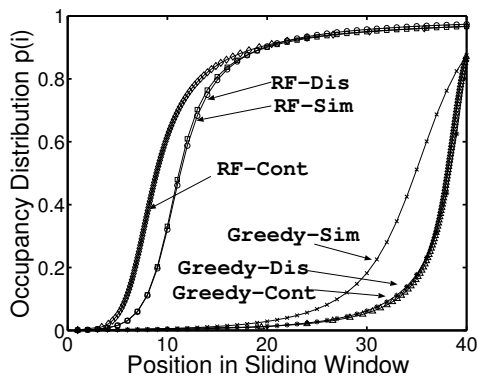
Fig. 2. Buffer occupancy distribution for Rarest First and Greedy policies from discrete, continuous and simulation models
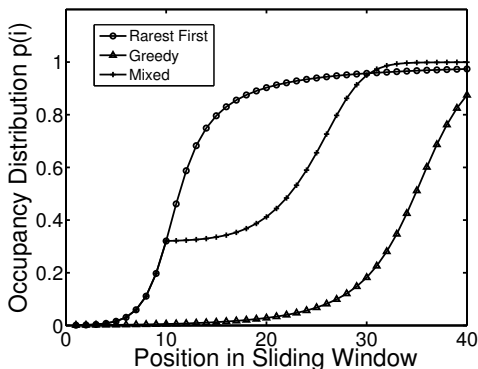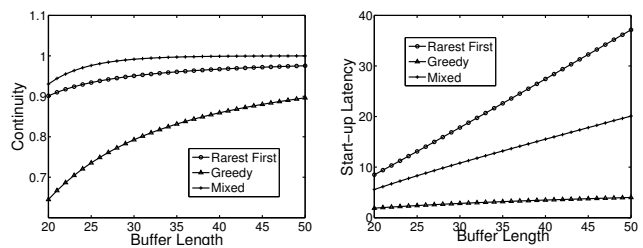

Fig. 3. Comparison of Rarest First, Greedy and Mixed

(a) Continuity versus buffer size (b) Start-up latency versus buffer size

Fig. 4. Performance Results for Exp. B.

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. The Rarest First Strategy is able to maximize the contribution of peers, hence its buffer occupancy probability is higher than other strategies in most buffer positions. When using the Greedy Strategy, all peers are focusing on the short-term playback needs; hence the buffer occupancy probability stays low except for those positions close to the playback position ($p(n)$). This has the advantage of minimizing the startup latency as we defined in Eq. (15). For Mixed, the buffer probability distribution is the same as Rarest First for positions $m \leq 10$, and follows the same shape as Greedy for

$m > 10$. By devoting a fraction of the buffer positions to Rarest First and the rest to Greedy, the Mixed Strategy can achieve higher continuity (than both Greedy and Rarest First) and lower startup latency (than Rarest First).

To further compare the different strategies for different buffer sizes, we plot the continuity and startup latency for buffer sizes between 20 and 50 in Figure 4(a) and Figure 4(b) respectively. It is observed that Rarest First consistently beats Greedy in continuity. The reason is evident from our analysis and Figure 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 somewhat like a procrastinator, making a great effort to fill the buffers only near the playback time for each chunk. It is interesting to note that the Mixed Strategy usually outperforms Rarest First in *continuity*.

In terms of startup latency, Greedy and Rarest First take opposite positions. To guarantee good playback continuity, Rarest First occupies a significant amount of buffer space. On the other hand, Greedy uses relatively less buffer space, hence it takes a newly arriving peer much less time to reach the steady state buffer occupancy. It is important to note that, Mixed is able to keep startup latency lower than Rarest First.
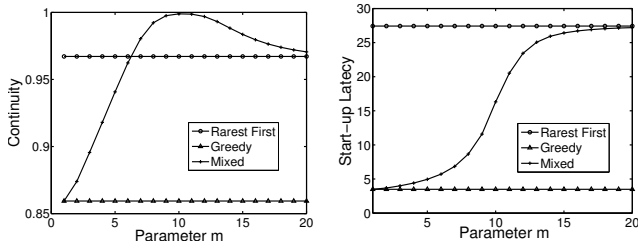
Exp. C: Optimizing the Mixed Strategy

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 and startup latency are plotted against m in Figure 5(a) and 5(b).

If m is large, the strategy is essentially Rarest First, hence there is a significant startup latency. When m decreases, the startup latency decreases monotonically, and eventually the scheme becomes sufficiently like the Greedy Strategy with low startup latency. 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 and low startup latency is achieved.

Exp. D: Performance for Small Scale Networks

In here, we consider the sensitivity of buffer size to continuity requirements and buffer size. We focus on some

(a) The effect of varying m on continuity of the Mix Strategy

(b) The effect of varying m on startup latency of the Mix strategy
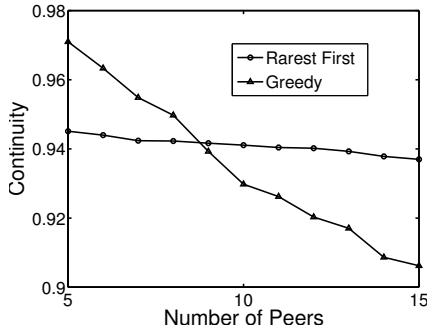
Fig. 5. Performance Results for Exp. C.

(a) The small network with fixed peers

(b) The small network with fixed continuity

Fig. 7. Second and Third Experiments in Exp. D.

Fig. 6. The small network

Fig. 8. Continuity of the Network Simulation

examples for small population size to illustrate when Greedy can perform better than Rarest First in terms of continuity.
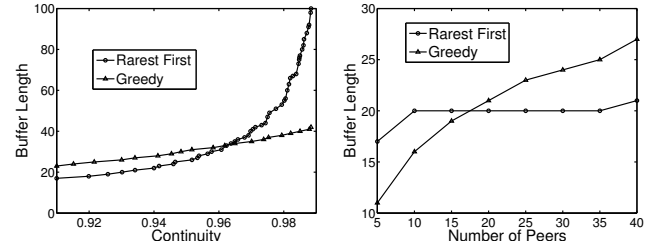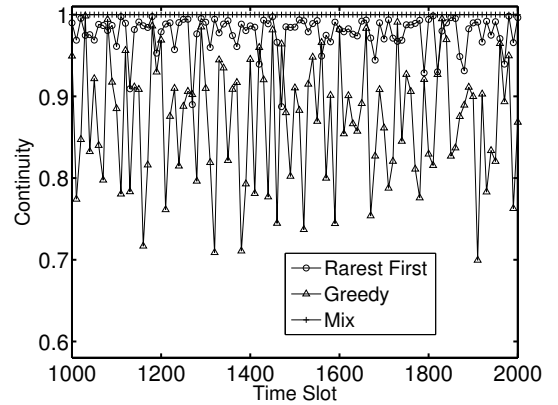
There are three examples in this experiment and the result in each case is derived from simulation (the analytical models are less accurate for small networks). Each result is calculated based on the average values of 3000 time slots.

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 6 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 experiment, we let 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 Figure 7(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 7(b).

The results from the above two experiments are consistent with Proposition 3 and 4, namely Greedy is able to provide a high quality requirement with less buffer length while Rarest First can provide good playback performance for a large number of peers.
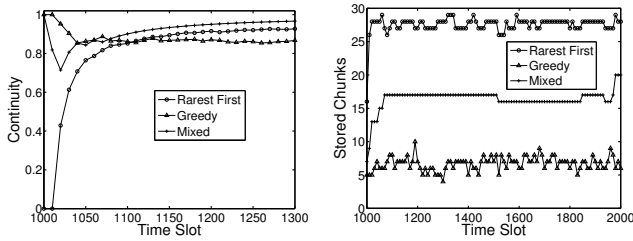
Exp. E: Study of Dynamics

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 and startup latency evolve over time.

First, we compare the continuity achieved by different strategies. We simulate 2000 time slots. 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 8, Mixed not only achieves the best continuity, but its continuity is also much more steady than that of other two strategies.

Secondly, consider the case that a new peer with empty buffer joins the network. Before the new peer arrives, we give 1000 time slots to let the existing (1000) peers reach steady state first. The newly arriving peer waits for $D = 16$ time slots before it starts playback. The arrival time is $1000 - D$ so that playback starts at the 1000th time slot. In Figure 9(a), we compare the playback performance of the newly arriving peer using each of the three chunk selection strategies. The continuity value in each case is computed as $\frac{s}{t-1000}$ where s is the number of time slot with successful playback.

Figure 9(b) shows the number of chunks stored in the

(a) Start-up latency of the Network Simulation

(b) Stored Chunks of the Network Simulation

Fig. 9.   Second Experiment in Exp. E.

(a) Average continuity as a function of time

(b) How m adapts to network dynamics

Fig. 10.   Performance Results from Exp. F.

buffer of the newly arriving peer as a function of time. From our model, we know the average number of chunks of a peer is simply $\sum_{i=1}^{n} p(i)$. This computation yields the expected number of chunks for each of the three strategies to be $27.4, 3.5, 15.5$ respectively, which is consistent with the steady state number achieved in the Figure. These numbers correspond to the appropriate startup latency suitable for each strategy.

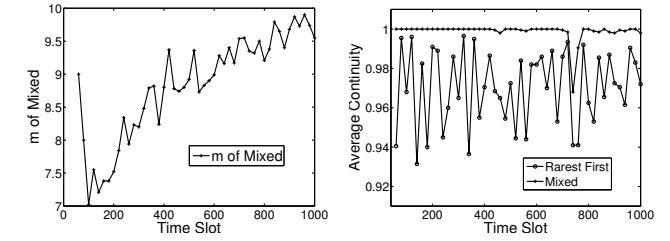**Exp. F: Adapting the Mixed Strategy to Peer Population**

Based on our analysis and the numerical examples, we show that the Mixed Strategy can achieve the best continuity and low startup latency 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 between time slot 700-800) compared to that of Rarest First.

V. **Application to Real-world Protocols**

In this section, we briefly discuss the applicability of the Mixed Strategy in real P2P streaming protocols. There are two points we would like to make.

First, the Mixed Strategy can be viewed as an optimization of the CoolStreaming protocol. Although our analytical model does not try to capture all aspects of the implementation of CoolStreaming, our chunk selection strategy can be easily incorporated into that protocol as an improvement of the existing algorithm. This makes us quite confident about the practical utility of our results, in addition to the insights we get from the model.

Second, the Mixed Strategy is also compatible with BiTos, and can be viewed as an alternative (very likely enhancement) of BiTos. Since $p(m) = \sum_{i=0}^{m-1} q(i)$, we can make our algorithm quite similar to BiTos which uses a probability p for high priority buffer positions and $(1 - p)$ for the rest. In fact, as we explained in the last section, we can implement the Mixed Strategy by using a fixed probability for the Rarest First part of the buffer, allowing m to adapt to a suitable value for the prevailing peer population. There is a subtle difference between the Mixed Strategy and BiTos: the latter uses Rarest First for both high priority and low priority chunks whereas we use Greedy for our high priority chunks.

VI. **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. Additional experimentation and prototyping would also help further validate our ideas.

REFERENCES

[1] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proc. 13th ACM NOSSDAV. New York: ACM Press 162-171*, 2003.
[2] R. S. Dongyu Qiu. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM*, 2004.
[3] B. Fan, D. M. Chiu, and J. C. Lui. The delicate tradeoffs in bit torrent-like file sharing protocol design. In *ICNP*, 2006.

[4] P. Francis. Yoid: Extending the internet multicast architecture. In *http://www.icir.org/yoid/docs/index.html*, 2000.

[5] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang. Delving into internet streaming media delivery: A quality and resource utilization perspective. In *Internet Measurement Conference Proceedings of the 6th ACM SIGCOMM on Internet measurement*, 2006.

[6] Y. hua Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *IEEE J on Selected Areas in Communications*, 2002.

[7] J. Liu, S. G. Rao, B. Li, and H. Zhang. Opportunities and challenges of peer-to-peer internet video broadcast. In *(invited) Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[8] L. Massoulie and M. Vojnovic. Coupon replication systems. In *ACM Sigmetrics*, 2005.

[9] T. Small, B. Liang, and B. Li. Scaling laws and tradeoffs of peer-to-peer live multimedia streaming. In *Proceedings of ACM Multimedia*, 2006.

[10] S. Tewari and L. Kleinrock. Analytical model for bittorrent-based live video streaming. In *Proceedings of IEEE NIME Workshop, Las Vegas*, 2007.

[11] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. A hierarchical characterization of a live streaming media workload. In *IEEE/ACM Transactions on Networking*, 2006.

[12] A. Vlavianos, M. Iliofotou, and M. Faloutsos. Bitos: Enhancing bittorrent for supporting streaming applications. In *INFOCOM 25th IEEE International Conference on Computer Communications. Proceedings*, 2006.

[13] C. Wu and B. Li. Optimal peer selection for minimum-delay peer-to-peer streaming with rateless codes. In *Proc. of ACM Workshop on Advances in P2P Multimedia Streaming*, 2005.

[14] G. Wu and T. cker Chiueh. How efficient is bittorrent? In *Proceedings of SPIE – Volume 6071 Multimedia Computing and Networking*, 2006.

[15] M. Zhang, L. zhao, Y. Tang, J.-G. Luo, and S.-Q. Yangt. Large-scale live media streaming over peer-to-peer networks through global internet. In *International Multimedia Conference Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming*, 2005.

[16] 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.

Appendix

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

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

From Eq. (7), 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

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

From the equation of $s(i)$ (Eq. 7), we get $s(n-1) = 1 - 1/M$. Therefore, we have $s(i) = 1 - p(1) - p(n) + p(i+1)$. ∎

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

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

From Eq. (10), we have

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

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

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

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. (10), we have $s(1) = 1 - 1/M$. Therefore, we have $s(i) = 1 - p(i)$. ∎

Proof of Proposition 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:

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

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 Proposition. ∎

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

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

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 Proposition. ∎