Modeling and Analysis of P2P Streaming

ZHOU, Yipeng

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

August 2008

Abstract of thesis entitled:
Modeling and Analysis of P2P Streaming
Submitted by Zhou, Yipeng
for the degree of Master of Philosophy
at The Chinese University of Hong Kong in July 2008

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 thesis, two work are done. First, we analyze the p2p streaming in synchronized case and 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. In addition, some assumptions in the model are modified and discussed, which can prove these assumptions are reasonable and our model is robust.

However, in real-life P2P streaming, how the peers select playback offset and buffer size? In the second part, we generalize the previous work and study a number of interesting consequences of this reality. Given a set of neighbors, a reasonable strategy of a peer is to set its playback offset in order to maximize the overlap of its playback buffer with its neighbors' buffers. If a cluster of

peers all adopt this local strategy, we show that the Nash equilibrium is for all peers to have synchronized playback in the cluster. Secondly, if peers in a cluster have synchronized playback offset, we show that it is best for them to adopt the same buffer size, for a given total buffer size. These results imply that the simple model of relatively synchronized peers is still reasonable for studying peers in a cluster. For generalization, we show that for really large-scale P2P sessions, it makes sense to organize peers into different clusters with different (cluster-wide) playback offsets. This is shown analytically using two-cluster model; the results are supported by simulation experiments. The analysis of this part is more realistic and provides further insights in the design of large-scale P2P streaming systems.

中文摘要

点对点的流媒体网络不但可以满足用户数量的扩展性(例如点对点的文件传送软件 BT)， 并且能同时满足多媒体流文件播放所需要的实时性需求。目前这个系统的设计与可法有很多值得研究的课题。在这篇毕业论文中，主要讨论和研究了两个方面的问题。

首先，用简单的随机过程模型研究点对点网络在一个近似同步的系统中。在这个随机过程模型中，我们可以比较各种文件块的选择策略的优缺点。在这两，我们设计了两个性能衡量指标：播放成功率和启动延迟。然后选取两种直观简单的文件块选择策略：最稀缺块优先策略和贪心策略。最稀缺块优先策略被广泛应用于文件共享系统并且取得了很好的用户扩展性。而贪心算法对单独用户来讲是一种非常简单直观来最大化满足播放成功的策略。贪心算法能和好的减少启动延迟但却不能得到很好的成功播放概率，而最稀缺块优先策略正好相反。通过研究启动延迟和播放成功概率之间的矛盾和系统扩展性与播放成功概率之间的矛盾，一种新的混合策略被提出。这种混合策略能同时满足系统扩展性和播放成功概率高要求之间的矛盾。并且这种策略是一种可根据系统用户量而自发调节的自适应算法。另外，在这个模型下面，文章做了很多假设，一些特定的实验被用来检测这些假设的合理性和模型的健壮性。

其次，考虑的真实地点对点流媒体网络中的用户观看并不是完全同步的，我们进一步提出新的模型来分析在不同步的情况下，用户的选择策略以及由此造成的影响。这部分的工作是基于第一部分的模型的一种改进和提高。在实际系统中，任何一个用户只能拥有有限的邻居提供帮助。在这种情况下，用户的合理策略是最大化自己和邻居的缓冲区的公共区间以便获得最大的帮助。如果在一个网络里面所有用户都使用这种策略，我们可以证明，所有用户将达到那什均衡状态并且将是一个同步的网络。如果用户可以自行选择存储缓冲区的大小，我们可以证明这种选择的结果是所有用户选择相同大小的缓冲区并且是那什均衡。同时我们还分析了两个网络间流量很小的情况，并且证明在这种情况下，适当的延迟有助于提高整个系统的性能。这部分的研究更加接近真实网络，相信对点对点网络的设计更有帮助。

Acknowledgement

I would like to thank my supervisors professor Chiu DahMing and professor John C.S. Lui. They gave great help in my research work and thesis writing. Thank Tom fu very much for his help in my second part work. Thank Bridge for his idea of the upper bound of buffer length in synchronized case.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary

There are three sections in this chapter. First, the background of P2P streaming and P2P VoD is introduced. Secondly, the contribution and difference compared with other work of this thesis are discussed. At last the organization of the thesis is shown.

1.1 Background

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* [10] or *end-host multicast* [15]. Since then, there has been a significant body of work on P2P streaming. In an

invited paper [18], 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 [29, 28, 26, 11], 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) [8, 9]. 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.

Recently, people become more and more interested in P2P Video on Demand systems, which use peer to peer technology to improve VoD systems. Several P2P VoD systems have been designed by PPlive [1] and PPstream [2]. However, Little is know about the effectiveness of the P2P technology in VoD systems because of its complication. In this thesis, some newest research work about P2P Vod is presented. The features, the strength, the merit and weakness are discussed based on these work.

1.2 Contribution

The contribution of the thesis 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 par-

ticular, 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. Actually, the Mixed strategy is asymptotic optimized strategy. Our model is discussed based on some assumptions. Furthermore, these assumptions are discussed one by one, which indicates the correctness of our model. The first model is under synchronized network system. Therefore, we extended it to unsynchronized case. The peers are classified different clusters and there are lags among these clusters, which can not be ignored. The unsynchronized case in one cluster is also analyzed. Through these analysis, we can understand how the lag affects the performance.

To make the problem tractable, there is a major assumption that all peers are synchronized in their playback. In other words, all peers playback the same content (offset of the video content) at the same time, with exactly the same delay equal to the buffer size. The reality, there are many factors that will affect the peers' playback offset. For example, due to geographical reasons, some peers are closer to the source of the content and are matched to the source for direct downloading whereas other far away peers may not receive direct downloading from the source. The latter peers tend to have a lag in their playback offset compared to the former group. Secondly, the actually decision of when to start playback may not be centrally controlled, but rather determined locally based on each peer's buffer content reaching a threshold of reserve. For this reason, and the fact peers have different neighbors, the resulting playback offsets are likely to be randomly distributed to some extent. Furthermore, peers' buffer sizes may be different. Each peer is likely to have a (configured) maximum buffer size. The actual buffer size in use (to support P2P streaming) will depend on a peer's playback offset and its neighbors' offsets. So a natural consequence of unsynchronized playback implies likely unequal buffer size, assuming the configured maximum buffer size is generous.

A new model generalized from above simple model to allow unsynchronized playback. The key insight of the above simple model is about better use of the peer buffer spaces to provide scalability while taking playback continuity into consideration. In contrast, the generalized model explain the role of differential lag and what drives peers to select the same playback offset and buffer size in one cluster. More specifically, we address the following questions:

1. Given a cluster of peers (who have each other as neighbors), how would the peers choose their playback offset if they are allowed to choose? When they all choose to maximize their buffer overlap, we show that they will choose to have synchronized playback.
2. Given a cluster of peers choosing synchronized playback, would they choose different buffer sizes? We show the answer is negative.
3. Given multiple clusters with different (cluster-wide) playback offsets, how much improvement can we expect compared to the single-cluster case? We characterize the improvements analytically for the two-cluster case, and study the multiple cluster case using simulation.

1.3 Organization

The organization of the thesis is as follows. Chapter 2 introduce some current popular research work, including P2P streaming and P2P VoD. Chapter 3 is on the basic probabilistic model; Chapter 4 goes into the details of how to model different chunk selection strategies; Chapter 5 provides various numerical examples, solved by both the discrete and the continuous version of our model, as well as validated by simulation. This chapter also describes application of our protocol to real protocol design and discusses the reasonableness of the assumptions in our model. In Chapter 6, we present the model for unsynchronized playback and discuss on the buffer map overlap maximization problem. Then we go into the details of how to analyze the average playback continuity of peers in the unsynchronized cluster. Chapter 7 provides the simulation results. Chapter 8 make a conclusion of the whole research thesis.

□ **End of chapter.**

Chapter 2

Related Work

Summary

In this chapter, the current P2P streaming work completed by researchers are introduced, especially two works, which are most related to our model. Then, a more challenging and novel research work P2P VoD and current work are discussed.

2.1 Work of Streaming

Recently, a number of mesh-pull P2P streaming systems such as Coolstreaming[29], BASS[7], AnySee[17], BiToS[26], and [19, 22, 28] are proposed by researchers. Usually these systems are evaluated through simulation or run on the testbed like Planet-lab[3]. Meanwhile, there are also some measurement-based studies of P2P streaming systems[12, 4, 21, 24, 25, 27]. In these measurement studies two measuring techniques have been applied: passive sniffing and active crawling.

There are a number of analytical studies on P2P file sharing systems. An important contributor to P2P file sharing is the body of work based on the BitTorrent[9, 23]. Since P2P file sharing does not have real-time requirements, it is very different from P2P streaming application. For analytical models on P2P streaming, [30] was the first study on formulating tractable analytical models to help understand important system designing level issues such as buffer design and chunk selection policies so as to provide good playback continuity and at the same time, scalability to the system. There is another theoretical study applying the stochastic fluid model to model the

P2P streaming[14]. In [14], the authors only considered the whole system's performance, whether to perform *universal streaming* or not. There are two most closely related to our work are CoolStreaming [29] and BiTos [26]. The two papers are discussed in more details as following:

CoolStreaming [29] 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 [26]. 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. [26] 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.

2.2 Work of P2P VoD

There are some most coming research on P2P VoD system. Some of them study the framework of the P2P VoD system, such as [1, 2]. Some of them focus on measurement such as [6].

In paper [6], the authors measured performance from a real system GirdCast. They measured the gap between GirdCast and the best case, studied the reasons causing large latency and how the

peers' behavior affect the average performance. However, there are some weak points. There were not many users in the system, about 20000 totally, and several hundreds concurrent peers. Most of users came from CERNET, who had quite good network bandwidth.

In paper [16], the authors did a great work including P2P VoD framework design, user behavior and system performance. Actually, the work is based on PPLive [1], which is real P2P VoD software and support tens of thousands of users on the internet with several hundreds channels.

In paper [5], the authors presented one measurement studies of a large VOD system, using data covering 219 days and more than 150,000 users in a VOD system deployed by China Telecom. Their study focuses on user behavior, content access patterns, and their implications on the design of multimedia streaming systems. There are many ways to design the P2P VoD framework, however the user behavior will not change. Study of the user behavior is helpful for various system design in P2P VoD.

Paper [5] proposed a novel framework for P2P VoD system. It is a ring based overlay network, in which each peer maintains a gossip-ring to explore appropriate data suppliers and several skip-rings with power law radius to assist the quick relocation of VCR operations. However, the shortcoming of this framework is that, the authors only solve it in simulation instead of implementing it in real network.

Paper [5] analyzed large volume of user behavior logs during playing multimedia streaming and extracted a user viewing pattern. Through analysis the authors proposed a new efficient prefetching algorithm to facilitate the random seek functionality. The authors set up an analogy between the optimization problem of minimizing the seeking distance and the optimal scalar quantization problem and then propose an optimal prefetching scheduling algorithm based on the optimal scalar quantization theory.

□ **End of chapter.**

Chapter 3

Basic Model of Synchronized Case

Summary

Some basic knowledge based on synchronized case is introduced. Some basic definition and relationship of these definition are discussed. This is a foundation of the further study.

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 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] 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.

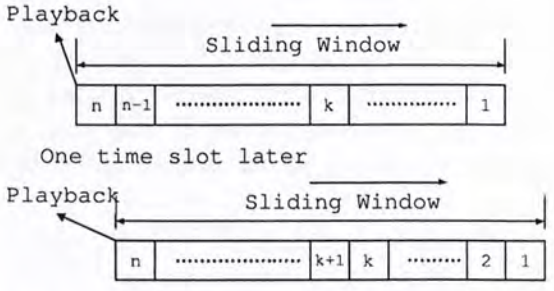


Figure 3.1: Sliding Window Mechanism of the buffer B

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 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[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:

$$p_k(1) = p(1) = \frac{1}{M} \qquad \forall k, \tag{3.1}$$

$$p_k(i+1) = p(i+1) = p(i) \quad i = 1, 2, \ldots, n-1 \ \ \forall k. \tag{3.2}$$

Eq. (3.1) reflects the odds for the local peer to be picked by the server, while Eq. (3.2) 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:

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

- 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.,[20].

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 [9]. 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. (3.2) becomes:

$$p(i+1) = p(i) + q(i) \quad i = 1, \ldots, n-1, \tag{3.3}$$

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:

$$\begin{aligned} q(i) &= \Pr[W(k,i) \cap H(h,i) \cap S(h,k,i)] \\ &= \Pr[W(k,i)] \Pr[H(h,i)|W(k,i)] \times \\ &\quad \Pr[S(h,k,i)|W(k,i) \cap H(h,i)]. \end{aligned} \tag{3.4}$$

The following assumptions help us to simplify Eq. (3.4):

- 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. (3.4) is:

$$q(i) \approx [1 - p_k(i)]\, p_h(i) s(i) = [1 - p(i)]\, p(i) s(i). \tag{3.5}$$

Since each of the terms in Eq. (3.5) is a probability (in particular $p(i) \leq 1$ and $p(i)s(i) \leq 1$), Eq. (3.3) becomes:

$$p(i+1) = p(i) + [1 - p(i)]p(i)s(i) \leq 1. \quad (3.6)$$

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

In the model, every peer has a strong constraint that one peer can only download one chunk per time slot. Based on this constraint, every peer has to adopt a chunk selection strategy, which is high related to the continuity and start up latency. Here, we relax the constraint so that the function $s(i)$ is equal to 1 all the time. That means if a peer find a neighbor from his neighbor list, the peer will download all these chunks in one time slot, which are absent from his own buffer. However, in real network, it is almost impossible to satisfy so large bandwidth. Therefore, we get an upper bound. Through letting $s(i) = 1$, we can derive the function for the upper bound based on equation Eq. (3.5):

$$p(i+1) = p(i) + p(i)(1 - p(i))$$

□ **End of chapter.**

Chapter 4

Model of Chunk Selection Strategies

Summary

First, two basic chunk selection strategies are discussed. Based on the discussion of the two basic strategies, a novel Mixed strategy is proposed, which can achieve best performance. Some propositions and conclusions are shown then. These propositions and conclusion are main contribution of the first part. At last, some metrics are defined.

4.1 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, 9], and P2P streaming protocol CoolStreaming [29]. 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. (3.3), 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.

4.1.1 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:

$$s(i) = \left(1 - \frac{1}{M}\right) \prod_{j=i+1}^{j=n-1} \left(p(j) + (1-p(j))^2\right). \tag{4.1}$$

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:

$$\Pr[\neg(W(k,j)H(h,j))] = p_k(j) + (1-p_k(j))(1-p_h(j)). \tag{4.2}$$

Eq. (4.1) 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. (4.2) 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. 4.1) has a surprisingly simple alternative form:

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

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

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. (3.6), we get the following "*difference equation*" for $p(i)$:

$$p(i+1) = p(i) + p(i)\Big(1-p(i)\Big)\Big(1-p(1)-p(n)+p(i+1)\Big) \quad \text{for } i = 1, \ldots, n-1. \tag{4.3}$$

4.1.2 Rarest First Strategy

The Rarest First Strategy is the opposite of the Greedy Strategy. Based on Eq. (3.3), we know $p(i)$ is an increasing function in i.[1] 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:

$$s(i) = \left(1-\frac{1}{M}\right) \prod_{j=1}^{j=i-1} \Big(p(j) + (1-p(j))\Big)\Big(1-p(j)\Big). \tag{4.4}$$

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. (4.4) has a simple form:

[1] 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.

Lemma 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.

Again, substituting $s(i)$ into Eq. (3.6), we have the following difference equation:

$$p(i+1) = p(i) + p(i)\Big(1-p(i)\Big)^2 \text{ for } i = 1, ..., n-1. \tag{4.5}$$

4.1.3 Buffer Size, Peer Population and Continuity

The difference equations for $p(i)$ in Eq. (4.3) and Eq. (4.5) 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. (4.3) and (4.5) 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:

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

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{\epsilon p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon p(1)}. \tag{4.6}$$

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

$$\frac{\partial n}{\partial p(1)} \approx -\frac{1}{p(1)} \quad ; \quad \frac{\partial n}{\partial \epsilon} \approx -\frac{1}{\epsilon^2} - \frac{1}{\epsilon}. \tag{4.7}$$

The proofs are included in the appendix.

Eq. (4.6) to (4.7) 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 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 the next section.

4.1.4 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, $1 \leq m \leq n$. 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:

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

The probability for $B(m+1)$ to $B(n)$ can be derived from Eq. (4.3) by substituting $p(1)$ with $p(m)$:

$$\begin{aligned} p(i+1) &= p(i) + p(i)(1-p(i)) \\ &\quad \times (1 - p(m) - p(n) + p(i+1)). \end{aligned} \tag{4.8}$$

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.

Lemma 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[2]. 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 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.

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

4.2 Some Conclusion and Extension

Lemma 6 For a p2p streaming network with large population, Greedy Strategy achieves better continuity than Rarest First Strategy, if the buffer length is large enough.

The proof is presented in the appendix. The results indicate that Greedy saves buffer length compared with RF if the continuity requirement is very high given fixed peer population.

Proposition 1 *In p2p streaming network with large population, if the number of peers is fixed, Greedy Strategy is good at continuity, while if continuity requirement is fixed, Rarest First Strategy is good at scalability.*

Proof: The result can be derived from Lemma 3,4. For given peer population, we can do differential n over discontinuity ϵ. From the Eq. (4.6) and (4.7), the absolute value for Rarest First Strategy is bigger than Greedy when discontinuity requirement ϵ is less than $p(1)$, that means the buffer length is more sensitive when continuity requirement increases for Rarest First Strategy. This result is consistent with lemma 6. Similarly, given discontinuity ϵ fixed, we do differential n over $p(1)$. The absolute value of Greedy is much bigger than Rarest First. That means the Greedy is more sensitive when the peer population increases.

Proposition 2 *Given peer population, Greedy Strategy certainly can achieve better continuity than Rarest First, if buffer length is large enough. Mixed Strategy beat both Greedy and Rarest First.*

Proof: The result is derived from the lemma 5,6. According to the proof of lemma 6, let $\epsilon = p(1)$, we can get the result that Greedy consumed less buffer length than Rarest First. While Mixed Strategy can be converted into a Greedy Strategy only with smaller peers as lemma 5 shows. Compared with Greedy, the Mixed Strategy has smaller peer population and achieves better performance. In other word, Mixed Strategy is the best one.

Proposition 3 *Assume the consumed buffer length for different strategies is a function of discontinuity ϵ and number of peers M. That is $n = f(\epsilon, M)$. The Mixed Strategy is an asymptotic optimized strategy.*

Proof: The proof is presented in appendix.

Actually, for Mixed strategy, the $p(n)$ in the Rarest First part is $p(1)$ for the Greedy part. The buffer length consumed is $n_{Mix} =$

$n_{RF} + n_{Greedy}$. Assume the $p(n)$ in the Rarest First part is λ, we can get a function $n_{Mixed} = g(\epsilon, M, \lambda)$. Given ϵ and M, there is only one variable λ. Through calculus, it is not hard to get the λ, which can minimize the buffer length. The closed form is quite complicate, so numerical results is shown in the next section.

4.3 Metrics

In this section, two metrics which are used to measure performance of p2p streaming system are shown as follow:

4.3.1 Continuity

So far we have focused on continuity $p(n)$ as the performance metric for evaluating various chunk selection strategies. From Eq. (3.3) and by defining $q(0)=p(1)$,[3] we have:

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

4.3.2 Start-up Latency

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:

$$\text{startup latency} = \sum_{i=1}^{n} p(i)/R. \tag{4.9}$$

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

[3]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.

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. (4.9), it will not improve its long-term steady state playback quality. Can we improve mixed strategy through a small change? Change the priority from discrete to continuous?

□ **End of chapter.**

Chapter 5

Experiment and Application

Summary

In this chapter, to convince our model, various simulation and computation results are shown and discussed. These experiment validate the correctness of our model. What is more, some issues of the application in real system are discussed at last.

5.1 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. 3.1,3.3,3.5,4.1,4.4,4.8). 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.

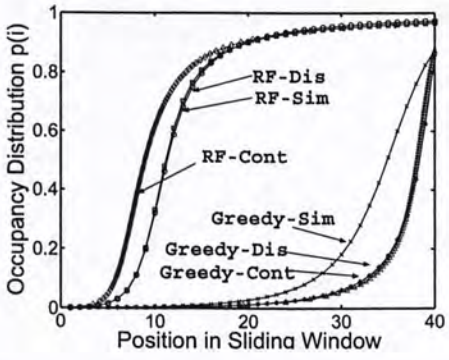


Figure 5.1: Buffer occupancy distribution for Rarest First and Greedy policies from discrete, continuous and simulation models

Continuous model: The continuous model is given by the differential equations in Eq. (4.3) and (4.5). 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 5.1. 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

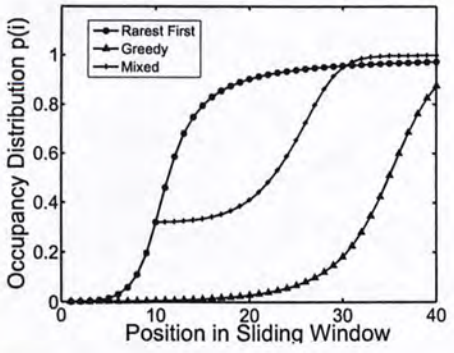


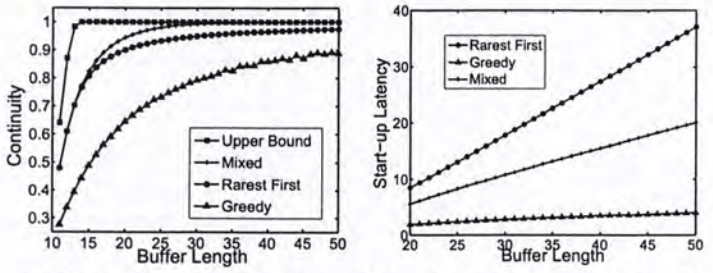
Figure 5.2: Comparison of Rarest First, Greedy and Mixed

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 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 5.2. 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. (4.9). 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



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

Figure 5.3: Performance Results for Exp. B.

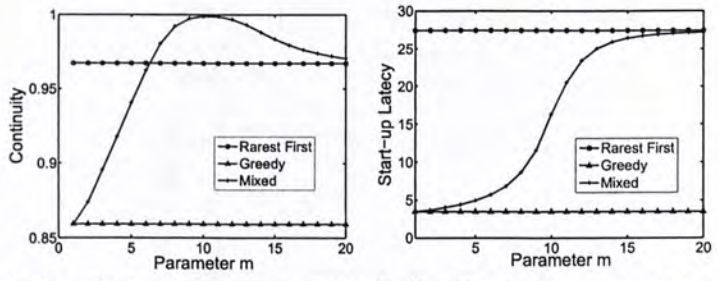
(than both Greedy and Rarest First) and lower startup latency (than Rarest First).

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 5.3(a) and Figure 5.3(b) respectively. It is observed that Rarest First consistently beats Greedy in continuity. The reason is evident from our analysis and Figure 5.1. 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 out-performs Rarest First in *continuity*. When the buffer length is larger than a relative small value 30, the gap between Mixed strategy and Upper bound becomes quite small. 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 5.4(a) and 5.4(b).

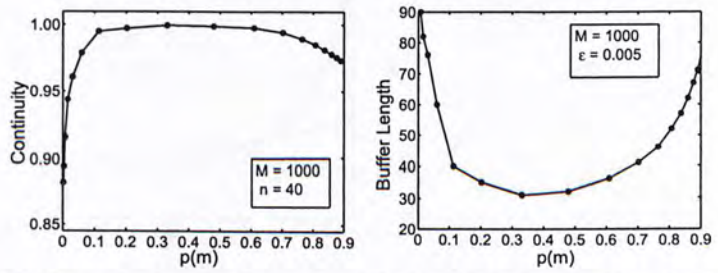
If m is large, the strategy is essentially Rarest First, hence there is a significant startup latency. When m increases, the startup latency decreases monotonically, and eventually the scheme becomes sufficiently like the Greedy Strategy with low startup latency. For



(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

Figure 5.4: Performance Results for Exp. C.



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

(b) The effect of varying λ on buffer length of the Mix strategy

Figure 5.5: λ v.s. best Mixed strategy

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.

Another way to view the Mixed strategy is the value of λ, which discussed in the proposition 3. In this numerical experiment, the number of peers is 1000. In the first experiment, the buffer length is given 40, while the value of λ varies. The continuity is not very sensitive for the varying λ. When λ is approximately equal to 0.3, the continuity is best. In the simulation, we assume $\lambda = 0.3$. In the second experiment, the discontinuity is fixed at 0.5%, while λ varies. The two figures show that continuity is not very sensitive when λ varies. In the dynamic network, the value λ is controlled to achieve good performance.

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 examples for small population size to illustrate when Greedy can perform better than

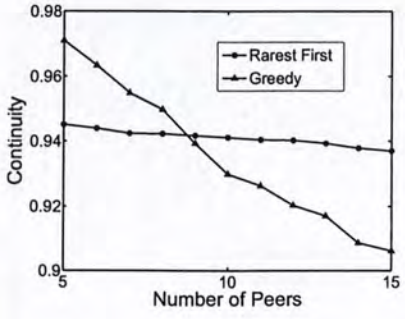
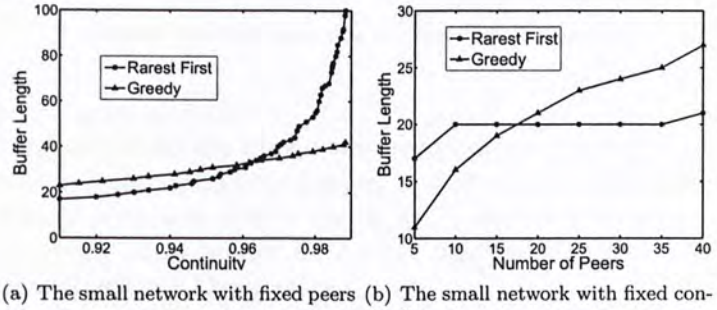


Figure 5.6: The small network



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

Figure 5.7: Second and Third Experiments in Exp. D.

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 5.6 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 Figire 5.7(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

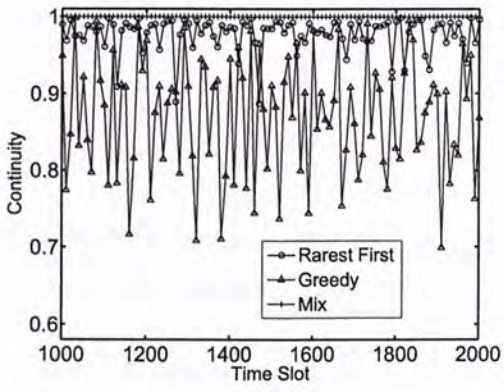


Figure 5.8: Continuity of the Network Simulation

shown in Figure 5.7(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.

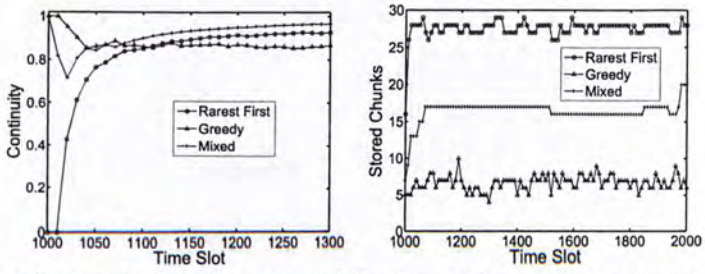
Exp. E: Study of Dynamics

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 5.8, 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 5.9(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 5.9(b) shows the number of chunks stored in the buffer of



(a) Start-up latency of the Network Simulation (b) Stored Chunks of the Network Simulation

Figure 5.9: Second Experiment in Exp. E.

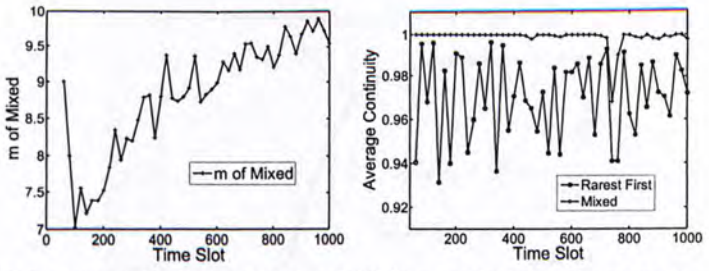
the newly arriving peer as a function of time. From our model, we know the average number of chunks of a peer is simply $\sum_{i=1}^{n} p(i)$. This computation yields the expected number of chunks for each of the three strategies to be $27.4, 3.5, 15.5$ respectively, which is consistent with the steady 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 $p(m)$, which actually is λ. 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 5.10(a) and 5.10(b), we observe that the average value of m (of the 100 tagged peers) adapts to the increasing peer population. Furthermore, the continuity of



(a) Average continuity as a function of time

(b) How m adapts to network dynamics

Figure 5.10: Performance Results from Exp. F.

the Mixed Strategy is quite steady (except a glitch between time slot 700-800) compared to that of Rarest First.

5.2 Sensitivity study

Up to now, our model is discussed based on some assumptions made in previous sections. However, are these assumptions reasonable? What should be corrected if some assumption is relaxed? Is the model robust under various network environments? These issues would be discussed in this section.

5.2.1 Discrete Model with Factor

One basic assumption in the model is that: there is enough bandwidth resource in the network to support the playback rate of all peers. However, in the real network, maybe the bandwidth is so limited 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 $(0, 1)$ expressing the limited bandwidth. We can prove that even in this case, it is not necessary to change our model much. The only difference compared with the original model is just the chunk selection function $s(i)$. Because of the limited bandwidth, each peer only can upload a chunk successfully with probability f. The server still push one chunk per time slot. For Greedy Strategy, $s(n-1)$ is changed to $f - \frac{1}{M}$ due to the limited bandwidth. Similar, for Rarest First Strategy, $s(1)$ is changed to $f - \frac{1}{M}$. Therefore, the corresponding $s(i)$ for Greedy Strategy becomes $s(i) = f - p(1) - p(n) + p(i+)$ and $s(i)$ for Rarest First Strategy becomes $s(i) = f - p(i)$. To sum up, the results for

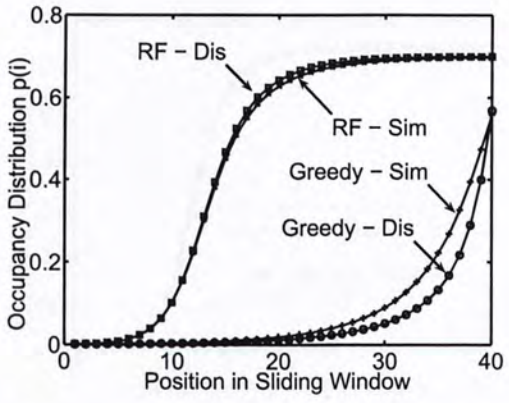


Figure 5.11: Buffer occupancy distribution of the network with limited bandwidth

discrete model become the following Eq. (5.1) and (5.2):

$$p(i+1) = p(i) + p(i)\Big(1-p(i)\Big)\Big(f - p(1) - p(n) + p(i+1)\Big) \quad \text{for } i = 1, \ldots, n-1. \tag{5.1}$$

$$p(i+1) = p(i) + p(i)\Big(1-p(i)\Big)\Big(f - p(i)\Big) \quad \text{for } i = 1, \ldots, n-1. \tag{5.2}$$

The simulation results in the next section will validate this new model.

5.2.2 Validate Discrete Model with Factor

This experiment is designed to validate our discrete model under limited bandwidth environment. In the simulation, there are 1000 peers. Each peer has a buffer with length 40. Set the factor $f = 0.7$. We run Greedy Strategy and Rarest First Strategy separately and compare them with the results, which are computed from discrete model Eq. (5.1) and (5.2). Figure 5.11 shows the comparing result.

5.2.3 Server Use Pull Strategy

In our model, the server is assumed to use push strategy to distribute the newest chunks. However, how it will affect the performance, if the server also adopts pull strategy? Here, a simulation is designed to observe the performance when the server uses pull strategy. Still

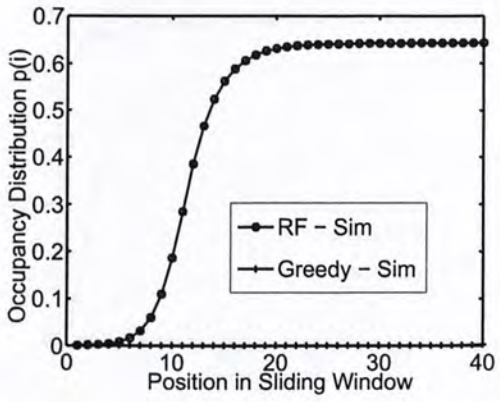


Figure 5.12: Buffer occupancy distribution of the network when server uses pull strategy

1000 peers stay in the network and buffer length is 40 for each peer. Figure 5.12 shows the result which indicates quite poor performance especially for Greedy Strategy. The result indicates the assumption that server uses push strategy is reasonable.

5.2.4 Vary Subset Size Touched by Server

In the original model, we assume the server could randomly push out newest chunk to the whole network. However, in the real network, the server can only talk with a subset peers of the network. How it will affect our model? In the simulation, we pick out a particular subset of peers from network and only peers in the subset could get the newest chunks pushed by server. Through varying the size of the subset, how it affect the performance can be seen from the Figure 5.13. When the subset size is relative small, for example 40, the curve has become quite flat. That means the difference between real network and our model is quite small.

5.3 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

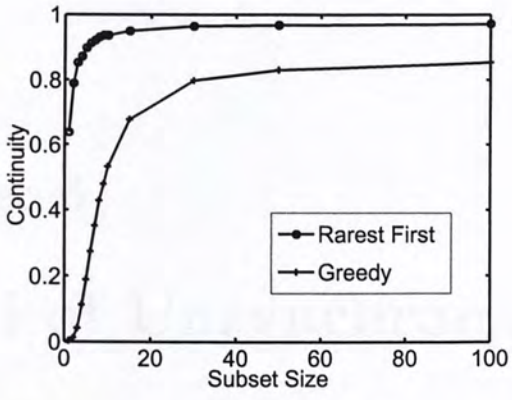


Figure 5.13: Buffer occupancy distribution of the network when server talks with a subset

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.

□ **End of chapter.**

Chapter 6

Model of Unsynchronized Case

Summary

This chapter is a generalization of last model. An unsynchronized model is constructed and the same issues are discussed under the unsynchronized case. Some interesting results are derived, which can give us more insights of the P2P streaming system.

6.1 The model for unsynchronized playback

In above model, we assume a P2P streaming session with M peers, fed by one server. We are interested in deriving $p_k(i)$, the probability that peer k's buffer space i is filled with the correct content in steady state. The departure from last part is that the playback offset is no longer assumed to be synchronized for all peers.

How do peers end up having different playback offsets? There are at least two major factors:

a) Peers have different neighbors. Some peers may have the server as one of the neighbors; while other peers may never get any chunk from the server directly. Generally speaking, those peers *closer* to the server tend to have less playback delay (or smaller playback offset).

b) The peers' playback start-up algorithm plays a role.

In order to analyze these factors, let us define some notations and terminologies.

- *Neighbors*: Each peer has L neighbors. Peers will only exchange chunks with its neighbors. Peers can acquire their neighbors in a variety of ways. For example, there exists some *tracker* that keeps track all M peers in the P2P session, and try to assign peers as neighbors to each newly arriving peer. Alternatively, a newly arriving peer may discover neighbors using some kind of search algorithm which tends to discover neighbors with some locality property.

- *Playback Offset*: In each time slot, the server will send out a new chunk of content; the time slot then marks the *birth time* of that chunk. To mask network jitter and to support other peers, each peer chooses to playback chunks with a delay. The playback offset of a peer is the delay of playback relative to the *birth time* of a chunk, normally denoted as O_k for the k^{th} peer. The unit of offset is in number of time slots. Different peers may have different playback offsets for a variety of reasons.

- *Buffer map*: Each peer has a buffer map, which labels the chunks in that peer's buffer. For streaming, the playback is assumed to be sequential, hence the buffer map is contiguous. Therefore, the buffer map can be represented by two offsets: *Buffer Head Offset* and *Buffer End Offset*, normally denoted by H_k and E_k for the k^{th} peer. The difference $n_k = E_k - H_k + 1$ is the buffer length of peer k. Unless a peer is particularly altruistic in helping others, it would discard the chunks after playback. That means $H_k = O_k$.

- *Overlap*: Overlap is simply the intersection of two peers' buffer maps. It is normally denoted $V(j,k)$ for peer j and k. The overlap $V(j,k)$ represents the maximum number of chunks each of the two peer can expect the other peer to provide help on, taking the optimistic view that the other peer has all its buffer filled with content.

- *Lag*: Lag is the difference between two peers' playback offsets.

A simple two-peer example illustrating the definition of *Buffer map*, *Overlap* and *Playback offset* is shown in Fig. 6.1. Assume the server sends out chunk 1 at time $t = 1$ and this chunk arrives to peer 1's buffer at position 1 (denote as $P1$ in the figure). At time slot $t = 2$, server sends out chunk 2 while peer 1 performs a *sliding window operation*: chunk 1 in position 1 will move to position 2 (or $P2$) while chunk 2 will be placed in $P1$. At $t = 7$, chunk 1 is in P7, which is the buffer head offset of peer 1, and peer 1 can start to

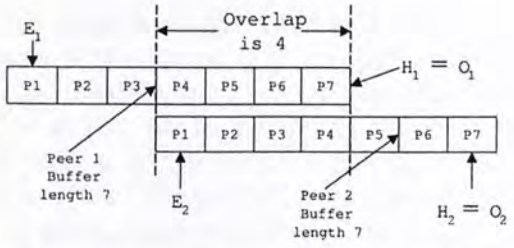


Figure 6.1: Illustration of *Buffer map*, *Overlap* and *Playback offset*

consume the data. In this case, peer 1 has a playback offset $O_1 = 7$. Similarly, due to network delay, chunk 1 only arrives to peer 2's buffer at $t = 4$ and it is placed in position $P1$. At $t = 10$, peer 2 can start to consume the data so $O_2 = 10$. Therefore, the lag between these two peers is 3. At time $t = 7$, the content of peer 1 in positions $P4$ to $P7$ are exactly the same as the content of peer 2 in positions $P1$ to $P4$, and this is the *overlap* between peer 1 and peer 2.

We now make two assumptions, and explain the rationale for each. First, we assume the neighbor lists of peers form clusters. Formally, clustering is a partition of the set of M peers into non-overlapping subsets, satisfying the condition:

$$L_{int}(i) \gg L_{ext}(i) \ \forall i$$

where $L_{int}(i)$ and $L_{ext}(i)$ are the number of peer $i's$ neighbors in the same cluster as i, and not in the same cluster as i respectively. If each peer's neighbors are chosen randomly, then they would not form clusters as above. However, measurement results show that peers normally form clusters [12] intentionally due to policy or unintentionally due to geographic locality. For the same reasons, we further assume that the network bandwidth between peers in the same cluster is essentially unlimited whereas with external peers is constrained. Modeling the process in which the peer overlay is formed is beyond the scope of this paper. The clustering model gives the framework for subsequent analysis.

The second assumption is that peers decide on their playback offsets based on *buffer overlap maximization*. In a unicast streaming scenario, the major consideration for a receiver to buffer is to mask out network jitter and enhance reliability. A plausible start-up algorithm is simply to let the buffer accumulate enough reserves before starting playback. In a P2P streaming scenario, a good start-up algorithm should also take into consideration of the buffer maps neighboring peers choose to use. To formalize this consideration, we postulate that peers pick their own buffer maps (and playback off-

set) by solving a *neighbor buffer overlap maximization problem* for a fixed maximum buffer size n. It is intuitively appealing for a pear to carry this optimization since a peer can put itself in a position to maximize the ability for its neighbors to help it download content. Although such playback start-up algorithms may be built into the peer software (not changeable), it is interesting to explore the consequences of letting each peer choose the buffer map selfishly.

6.1.1 Overlap maximization problem

Assume each peer is aware of all its neighbors' buffer maps. To simplify our explanation without loosing generality, let us assume each peer picks the playback offset to be the head of the buffer; namely $H_k = O_k$ and $E_k = O_k + n - 1$. This means peer k's buffer map is explicitly represented by its offset O_k. For peer k, let its neighbors be denoted as $a_k(i), i = 1, 2, \ldots, L$. The neighbors' offsets are $O_{a_k(i)}, i = 1, 2, \ldots, L$.

For a simple pair of neighbors j and k, the buffer overlap can be expressed in terms of their respective offsets:

$$V(j,k) = \min\{0, n - |O_j - O_k|\}. \tag{6.1}$$

This is clearly illustrated in Fig. 6.1. Then the overlap maximization problem for peer k can be written as:

$$\arg\max_O \; F_k(O) = \sum_{i=1}^{L} V(O, O_{a_k(i)}). \tag{6.2}$$

Due to the clustering assumption, $L_{int}(k) \gg L_{ext}(k)$, we refine the overlap maximization problem to focus only on the neighbors within the same cluster. That means:

$$\arg\max_O \; F_k(O) \approx \sum_{i=1}^{L_{int}(k)} V(O, O_{a_k(i)}). \tag{6.3}$$

According to Eq. (6.1), V has an upper bound of n. So $F_k(O)$ is bounded by $L_{int} * n$. This upper bound is reached when this peer and all its L_{int} neighbor peers have the same playback offset as in Eq. (6.4),

$$O = O_{a_k(1)} = O_{a_k(2)} = \ldots = O_{a_k(L_{int})}. \tag{6.4}$$

Meanwhile because these L_{int} neighbor peers of peer k are arbitrarily selected from k's cluster, one can conclude that all the peers in the cluster have the same playback offset. Such a cluster is referred to as a *synchronized cluster*.

6.1.2 Properties of the synchronized cluster

We make several observations about the synchronized cluster, under the assumption that each peer can see all other peers in the cluster. Therefore, each cluster becomes the neighbor set for all peers in the cluster.

First, define the *total overlap* of a cluster, $V_{cluster}$ as the sum of overlaps between all pairs of neighbors in a cluster. Suppose a cluster has M peers and each peer has a fixed buffer size n, then we observe:

Proposition 4 The upper bound of the total overlap of a cluster is $M * (M-1) * n$ and the total overlap of a synchronized cluster achieves this bound.

Proof 1 *Given any two peers j and k, from Eq. (6.1), the overlap is given by*

$$V(j,k) \leq n$$

Hence,

$$\begin{aligned} V_{cluster} &= \sum_{(i,j)} V(j,k) \\ \Rightarrow V_{cluster} &\leq M(M-1)n. \end{aligned} \tag{6.5}$$

For the synchronized cluster, the overlap between each pair of peers is exactly n, so the upper bound is achieved.

Considered as a game, the solution for the synchronous cluster is a unique Nash Equilibrium achieved by selfish peers:

Proposition 5 The synchronized cluster is a Nash Equilibrium.

Proof 2 *Let us consider peer k's optimal offset when the other $M-1$ peers in the cluster are synchronized, with the same buffer length n and same playback offset O^*. The payoff function for peer k is given in Eq. (6.3), which can be expressed as:*

$$F_k(O) \approx \sum_{i=1}^{L_{int}(k)} V(O, O_{a_k(i)}) \tag{6.6}$$

$$= (M-1)(n-(O-O^*)). \tag{6.7}$$

This quantity is obviously maximized if $O = O^*$. In other words, when the cluster is synchronized, peers have no incentives to change their offsets and therefore the synchronized cluster is a Nash Equilibrium.

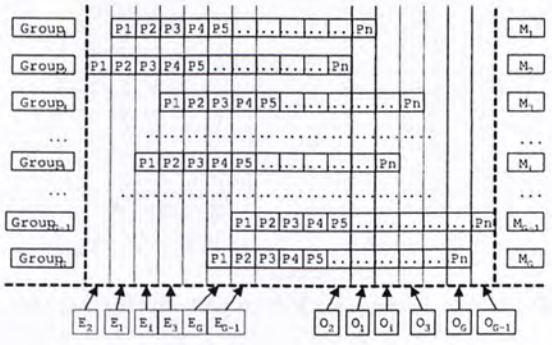


Figure 6.2: Illustration of an unsynchronized cluster in the proof of unique Nash Equilibrium

Proposition 6 The synchronized cluster is the only Nash Equilibrium.

Proof 3 *Let us assume the contrary and prove by contradiction. Assume there is another equilibrium where the cluster is not synchronized. A general representation of unsynchronized cluster is to let there be G groups of peers, where peers in each group share the same playback offset. The numbers of peers in these G groups are* $M_i, i = 1, \ldots, G$ *with* $\sum M_i = M$*, and the playback offset of group i is denoted* O_i*, see Fig. 6.2. Let us denote the total overlap for any member of group i with the rest of the cluster as* $V_{group}(i)$*. It can be expressed as:*

$$V_{group}(i) = \sum_{j \in [1,G], j \neq i} M_j * V(O_i, O_j) + (M_i - 1) * n. \tag{6.8}$$

We have two cases to consider. (1) peers in different groups should have the same total overlap with other peers in the cluster, which means Eq. (6.9) should be satisfied.

$$V_{group}(i) \;=\; V_{group}(j) \;\; \forall i, j \;\in\; [1, G]. \tag{6.9}$$

Otherwise, there will exist $i, j \in [1, G]$ *and* $V_{group}(i) < V_{group}(j)$*. If a peer of group i moves to group j by changing its playback offset, its total overlap after moving will become:*

$$V'_{group}(j) = V_{group}(j) + n - V(O_i, O_j) > V_{group}(i), \tag{6.10}$$

which is greater than its total overlap before moving. That means some peers have the incentive to change their playback offsets, which contradicts the Nash Equilibrium condition. So Eq. (6.9) should be satisfied.

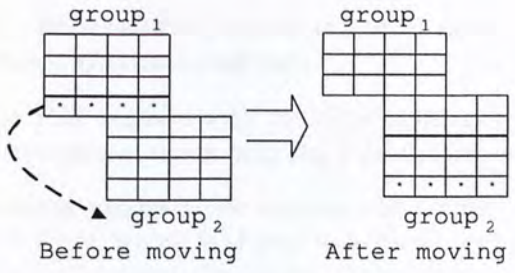


Figure 6.3: An example of the proof of uniqueness in Nash Equilibrium.

(2) Even when $V_{group}(i) = V_{group}(j)$ (case 1) holds, it is not a stable situation. Peers still have incentive to change their playback offset to get larger payoffs. In this situation, any peer moving from Group i to j where $O_i \neq O_j$ (or vice versa), its total overlap after moving becomes $V'_{group}(j) = V_{group}(j)+n-V(O_i, O_j)$, which is larger than that before moving, $V_{group}(i)$. The logic is as follows:

$$\begin{aligned} & \because \ O_i \neq O_j \\ \Rightarrow \ & V(O_i, O_j) < n \\ \Rightarrow \ & V'_{group}(j) = V_{group}(j) + n - V(O_i, O_j) \\ > \ & V_{group}(j) = V_{group}(i). \end{aligned} \tag{6.11}$$

This proves that any unsynchronized cluster is not stable, and the synchronized cluster is the only Nash Equilibrium.

An intuitive illustration of the uniqueness of this Nash Equilibrium is shown in Fig. 6.3. Let us consider there are two groups in the cluster, $group_1$ and $group_2$ and both groups have 4 peers with same buffer length of 4. From Eq. (6.8), the total overlap of member in either group before any peer's moving is $V_{group}(1) = V_{group}(2) = 20$, (left part in Fig. 6.3). After a peer moves from $group_1$ to $group_2$, $V'_{group}(1) = 18$ and $V'_{group}(2) = 22$ (right part in Fig. 6.3), which means this peer gains two more chunks of its total overlap after moving.

6.2 Analysis of playback continuity

While buffer overlap is a useful consideration for the start-up algorithm, the ultimate measure of performance for each peer is *playback continuity*. We now further evaluate unsynchronized playback based on continuity. In particular, we are interested in the following questions:

- Does it make sense for peers that are in the same cluster to have different playback buffers?
- In a large P2P session with multiple (synchronized) clusters, does lag between clusters help the P2P system scale?

In the following analysis, we assume the chunk selection algorithm is *Rarest First*, which is shown to achieve very good continuity asymptotically as population grows. Our methodology is similar to previous chapters, which assumes large peer population size.

6.2.1 Peers with different buffer sizes

Although based on buffer overlap consideration, peers in the same cluster would choose to maximize their buffer overlap, and hence have the same playback offset, that is a highly idealized scenario. In reality, the peers' playback offsets tend to be less synchronized, even in t he same cluster. For peers in the same cluster, it is reasonable to assume they have the same buffer end offsets E_k, since they are assumed to know each other's buffer maps. If peers have the same E_k but different playback offset O_k, it means they have different buffer sizes n_k. An example is shown in the second cluster in Fig. 6.4. An interesting question is whether different buffer sizes help improve continuity.

To make an apple-to-apple comparison, let us consider the "*different-buffer-size*" case with the "*same-buffer-size* case under the following conditions: (a) The number of peers in both clusters are the same; (b) The average buffer size of both clusters are the same. The conclusion, to be proven later in this subsection, is that the average continuity in the synchronous cluster (same buffer size) is better than that in the unsynchronized cluster (different buffer sizes but same buffer end offset).

To begin, let us consider a simple example illustrating the two kinds of clusters in Fig. 6.4. Both clusters have four peers. In the synchronized cluster (labeled C_1), all peers have the same buffer size (seven). In the unsynchronized cluster (C_2), peers have different buffer sizes but with the same average as that of C_1, for fair comparison.

For the synchronized cluster case, the methodology of previous chapters can be applied to derive the average continuity. We briefly explain it here again for convenience. Since all peers are symmetric in all respects, their playback buffer states are expected to have the same probability distribution for occupancy, where $p(i)$ denotes the probability that buffer position i has the appropriate chunk.

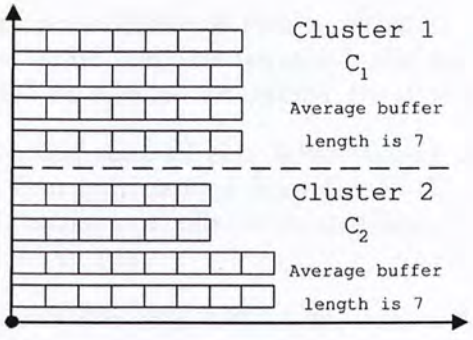


Figure 6.4: Comparing synchronized and unsynchronized cluster with different buffer length

For streaming, one buffer's worth of content is consumed in each time slot, and the rest of the buffer positions slide forward by one position. This leads to a difference equation for $p(i)$:

$$p(1) = \frac{1}{M}, \tag{6.12}$$

$$\begin{aligned} p(i+1) &= p(i) + q(i) \\ &= p(i) + (1 - p(i))p(i)(1 - p(i)), \end{aligned} \tag{6.13}$$

where M is the number of peers in the cluster. Eq. (6.12) is true because it is the probability that the server selects this peer and delivers the chunk to position 1. The first term of Eq. (6.13) represents the probability the local buffer already has the chunk; the second term $q(i)$ is the product of three components, representing respectively (1) the probability the local buffer does not have the chunk; (2) the probability a randomly selected neighbor has the chunk; (3) the probability the chunk for buffer i is the highest priority chunk to be fetched. These equations can be solved to derive $p(n)$, the probability of playback continuity.

In fact, Eq. (6.13) can be used to recursively derive the continuity $p(n)$ for any buffer size n. Furthermore, since the second term of Eq. (6.13), $q(i)$ is a probability density distribution (in i), $p(i)$ can be viewed as a cumulative probability distribution, that is increasing in i and $p(n)$ approaches 1 as the buffer size n increases.

For a cluster of unsynchronized buffer sizes, it turns out that Eq. (6.13) can still be used to derive the buffer occupancy distribution for peers of different buffer sizes, with the assumption of *rational peer selection* and large peer population. When peers have different buffer sizes, given a particular chunk (at buffer position i in the current time slot, say), this chunk may not be part of the

buffer map of all peers. Rational peer selection assumes only those neighbors whose buffer maps contain the buffer position (i) for the desired chunk will be selected for sending the downloading request.

Lemma 7 Assuming rational peer selection and large peer population, the function $p(i)$ derived from Eq. (6.13) can be used to compute the continuity $p(n)$ of peers with different buffer sizes n in an unsynchronized cluster.

The proof follows obviously from the recursive nature of Eq. (6.13) once the rational peer selection assumption is satisfied. Note, the rational peer selection assumption is only needed for the second term in the three term product in Eq. (6.13). In a cluster with unsynchronized peers, if a random peer is selected in each time slot to download, then the probability that peer is helpful for buffer position i will be smaller than $p(i)$. So the assumption of rational peer selection yields an upper bound for the unsynchronized case. Since even this upper bound will be proven to be worse than the synchronized cluster, the unsynchronized cluster without rational peer selection will be even worse in the playback continuity performance metric.

Another important fact needed for our comparison concerns the nature of $p(i)$ as a function of i, as stated in the following lemma.

Lemma 8 Using the Rarest First chunk selection strategy, the steady state buffer occupancy distribution $p(i)$ is a concave function in i for i above some threshold $n^* = O(\log M)$ where M is the population size.

The proof is included in the Appendix.

From Lemma 7, we know that the continuity of peers with buffer size n in an unsynchronized cluster can be computed as $p(n)$. The implication of Lemma 8 is that the contribution of peers with buffer size $n > n^*$ increases with a decreasing rate with n. We refer to the threshold n^* as the *concave buffer threshold.*

Suppose there is an unsynchronized cluster C with G groups of peers. The number of peers in group i is M_i, and the total number of peers in C is M, so $\sum_{i=1}^{G} \frac{M_i}{M} = 1$. Peers of group i use n_i as the buffer size.

Based on Lemma 7, the average continuity for Cluster C, can be computed as

$$\bar{C} = \sum_{i=1}^{G} p(n_i) \frac{M_i}{M}.$$

Let us denote the average continuity of a synchronized cluster by $\bar{C}_{syn}$. To ensure fair comparison, we also denote the average buffer size by $\bar{n}$. For C, $\bar{n} = \sum_{i=1}^{G} n_i \frac{M_i}{M}$.

Proposition 7 Given a synchronized cluster with M peers, buffer size n, and any comparable unsynchronized cluster with the same population size M, average buffer size $\bar{n}$ with $\bar{n} = n$ and average continuity $\bar{C}$, we have $\bar{C} \leq \bar{C}_{syn}$, provided all the buffer sizes in the unsynchronous cluster are above the concave buffer threshold n^*.

The proof of this proposition is straightforward given Lemma 7 and 8, and is stated informally here. It follows from Jensen's Inequality that given a concave function $p()$,

$$p\left(\frac{\sum_{i=1}^{M} n_i}{M}\right) \geq \frac{\sum_{i=1}^{M} p(n_i)}{M},$$

where n_i denote the buffer sizes of individual peers. This proves the above proposition since we are focusing on buffer sizes when $p(i)$ is concave. Note, since n^* is of order $\log M$, it is justifiably smaller than normally adopted buffer sizes.

Finally, we observe that peers in an unsynchronized cluster have no reason to prefer different buffer sizes even from their selfish view point.

Proposition 8 Picking equal buffer sizes is a Nash Equilibrium.

If only one peer enlarges its buffer without others' cooperation, it cannot gain anything if all other peers continue to use the same smaller buffers. If any peer uses buffer length larger than others, others can improve continuity by enlarging buffer length until they use buffers of same length.

6.2.2 Analysis of two clusters with a lag

So far, we have shown that peers in the same cluster tend to synchronize their playback; yet when peer population increases, it helps scalability to desynchronize the playback. One practical way for peers to re-organize themselves into groups with different lags is to split themselves into separate clusters. From measurement studies, it is apparent that peers in the same session are organized into different clusters [12]. A study of such cluster-forming algorithms is beyond the scope of this paper. Instead, assume peers are partitioned into separate clusters, and each cluster is more or less synchronized. An

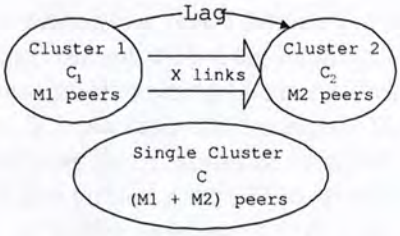


Figure 6.5: Comparing one single cluster with two smaller clusters

interesting question is how the multi-cluster scenario compares with the scenario of a single cluster with the combined population size.

Earlier, by considering peers as part of the same cluster, peers are assumed to have high connectivity with each other. Now, when we split peers into separate clusters, peers from different clusters are assumed to have only sparse connectivity with each other. This affects the extent different clusters help each other; but it also helps us to be more explicit about how different clusters help each other. Also, by modeling each cluster as a synchronized cluster, the problem for each cluster becomes tractable. Under these assumptions, the cluster with a smaller playback offset behaves like a server to the lagging cluster. This indeed is the basic approach we use to analyze the two-cluster model as shown in Fig. 6.5. Our goal is to compare the two cases in the figure: (a) the two-cluster with a lag case (with x peer connections per time slot and lag equal to D), and (b) the single cluster with the combined population $(M_1 + M_2)$.

For case (b), if we treat it as a synchronized cluster, the methodology to derive the average continuity is already known, which is explained in previous chapters. For case (a), we can treat each of the clusters as a synchronized cluster, derive the average continuity of each cluster and take the weighted average. The methodology is approximately correct, as verified by simulation to be explained later. But from a theoretical point of view, there are two effects that need to be understood.

The first cluster (the one with a smaller playback offset) is directly connected to the server, with population M_1. Suppose the average continuity we derive using the standard methodology for a synchronous cluster is $\bar{c_1}(M_1)$. Since the peers of the first cluster collectively help the second cluster, however, the first cluster gives up a small percentage of its capacity for serving itself. After incorporating this factor, the actual average continuity of C_1 should be $\bar{c_1'}(M_1)$ which is smaller. Denote $d_1 = \bar{c_1}(M_1) - \bar{c_1'}(M_1)$. We will describe a method to estimate d_1 later in this section.

For the second cluster, we can assume it is served by a virtual server that is feeding content with a lag equal to the lag between the two clusters. We assume the average number of connections between the two clusters, x, is 1 per time slot. Suppose the lag is D. In each time slot, a random peer in C_1 would be selected to provide content to a peer in C_2. If the virtual server (the selected peer in C_1) does not have content in buffer position D, it may alternatively feed any content in position $\{(D+1),(D+2),\ldots,n\}$. But in reality, the virtual server cannot be as good as a real server, since there is a small probability that the selected peer in C_1 does not have any content in buffer position $\{D,(D+1),\ldots,n\}$. Furthermore, a virtual server for position $D+j$ is not as good to C_2 as a virtual server for position D. Let us denote this degradation factor by d_2, so the actual continuity of cluster C_2 is $\bar{c_2} = d_2\bar{c_2}(M_2)$. The value of d_2 can be approximately computed from the solution for C_1 as

$$d_2 \approx 1-(1-p(D))^{n-D}$$

which is very close to 1. Therefore, we are going to skip the detailed accounting of this factor.

Let us return to the problem of computing d_1, which is actually non-trivial. We came up with an alternative method to estimate d_1 as follows. Consider adding a server to serve C_1, feeding it content at buffer position D. It is possible to compute the additional average continuity this second server will bring, which is denoted a_1. It can be argued that the effect of C_1 serving C_2 is approximately the same as the effect gained due to the additional server, that is, $d_1 = a_1$. This gives us a way to estimate d_1. The following Lemma gives what we can say about a_1:

Lemma 9 a_1 is a decreasing function of the lag D. a_1 is bounded by $(\epsilon^2 p(1), \epsilon^2(1-\epsilon)\ln 2)$.

The proof is in the Appendix. It is intuitive that a_1 is a decreasing function of D. When D is large, the extra server providing content to buffer position D does not help the scalability of C_1 as much. Similarly, it is more important for C_1 to disseminate chunks in smaller buffer positions to its own cluster first. In larger buffer positions, more peers in C_1 have those chunks, hence it costs less to help C_2 with those chunks. The important conclusion from this lemma is that the magnitude of a_1, hence d_1 is also very small, based on the bounds in the Lemma. This will be further validated by simulation in the next section.

Once we have masked the possible effects of d_1 and d_2, we can easily compare the two cases in Fig. 6.5, by applying the standard

methodology of synchronized clusters to each of the clusters in the figure.

Proposition 9 For sufficiently large population, if there is a lag between two clusters and the lag is smaller than the buffer length n, the lag will help both clusters improve average continuity compared to the case of a single synchronized cluster of combined population. Furthermore, the larger the lag the more is the improvement.

The proof is given in the Appendix. The implication of Proposition 9 is as follows. When the number of peers in a synchronized cluster increases, the average continuity will decrease. There are two ways to compensate for this degradation of average continuity. One is to increase the buffer sizes of all peers; and the other is to split the original cluster into two smaller synchronized clusters with a lag between them.

Proposition 10 When the number of peers in the two smaller clusters between which there is a lag is the same $\frac{M_1}{M_2} = 1$, the improvement of the average continuity achieve the maximum value.

The proof is given in the Appendix.

□ **End of chapter.**

Chapter 7

Performance Evaluation of Unsynchronized System

Summary

To validate our discussion of last chapter, the experiments are designed. Through these experiment results, we can better understand how the lag affect the performance of P2P streaming system.

7.1 Performance Evaluation

We carry out simulations to validate the theoretical claims that we made in previous sections. Results are presented in this section.
Experiment A: Comparison between synchronized cluster and unsynchronized cluster with different buffer length: In this experiment, we want to validate that the average playback continuity of peers in a *synchronized cluster* (in which peers have the same playback offset and buffer length) is better than peers in an *unsynchronized cluster with different buffer length* (in which peers have different buffer lengths but the same *Buffer End Offsets*). For the unsynchronized cluster, peers are divided into different groups according to their buffer length.

Before we start this experiment one needs to determine the distribution of peers population of each group in the unsynchronized cluster. In previous chapters when we discuss the reason why peers in the unsynchronized buffer length cluster have the same Buffer End Offsets but different buffer length, we argue that it is equivalent to the case that peers have the same Buffer End Offsets. This

implies that peers request the same newest chunk from the server at the same time slot, but they playback that chunk at different time slot (different Playback Offsets) according to the start-up algorithm mentioned in [12, 13]. The description of the start-up algorithm is that each peer will not start the playback until they have successfully downloaded certain number (a pre-defined parameter) of consecutive chunks. To determine this distribution, we run a simulation to see what is the distribution of time slots that peers take to download the first N chunks. In the simulation, we set the total number of peers in a cluster $M = 1000$, buffer length $n = 40$, and $N = 10$ (which means that peers will start as long as they get the first ten chunks). The result is shown in Fig. 7.1. According to the points conjuncted by the solid line, the time slots when peers start to playback is distributed between $[19, 42]$ and the mean is 25.26. From Fig. 7.1, one can see that the curve of simulation result is very

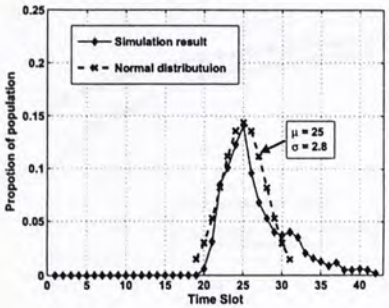


Figure 7.1: Simulation result of the start-up latency and the approximate normal distribution

similar to a normal distribution (the dotted curve in Fig. 7.1) with $\mu = 25$ and $\sigma = 2.8$. In later experiments, this normal distribution (dotted line) is used to approximate the original curve (solid line). This normal distribution is listed in Table 7.1.

x	19	20	21	22	23	24	25
y	0.015	0.03	0.053	0.082	0.112	0.136	0.144
x	26	27	28	29	30	31	
y	0.136	0.112	0.082	0.053	0.03	0.015	

Table 7.1: Coordinates of points for the normal distribution

To evaluate the performance of playback continuity between these two types of clusters, we first use the above mentioned normal distribution to determine the distribution of buffer lengths for peers

in the unsynchronized cluster. In order to have a fair comparison between these two types of clusters, we make both clusters to have the same average buffer length n, and vary n from 20 to 40. For the unsynchronized cluster with different buffer length, there are $G = 13$ groups (based on the 13 points in Table 7.1). In addition, when the average buffer length n is changed, we use a different normal distribution with the same variance but adjust $\mu = n$. Fig. 7.2 shows the simulation results of the average playback continuity when n varies from 20 to 40 for these two types of clusters. For each value of n, we run the simulation for 2000 time slots. In each time slot, the average playback continuity is the ratio of number of peers which playback the chunks to the number of all peers in the cluster. As shown in Fig. 7.2, the synchronized cluster has *better* playback continuity than the unsynchronized cluster with different buffer length.

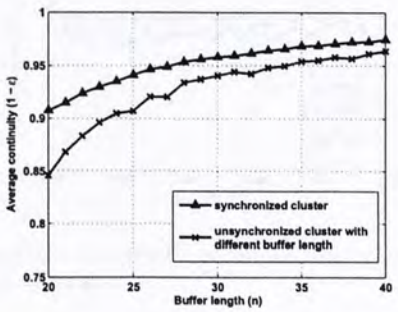


Figure 7.2: Simulation results of the playback continuity with different average buffer length

Fig. 7.3 shows the average playback continuity for these two clusters between 1000 to 2000 time slots when n is fixed at 25. The simulation results indicate that the average playback continuity of synchronized cluster is better than that of unsynchronized cluster with different buffer length under fair comparison. These validate the theoretical claim of Proposition 7.

Experiment B: Comparing the playback continuity of single cluster vs. two smaller clusters with Lag: In this experiment, we examine the performance of a single synchronized cluster containing totally M peers with buffer length n, and two smaller synchronized clusters containing M_1 and M_2 peers respectively with the same buffer length n, where $M = M_1 + M_2$. We carry out three different types of simulations by varying a parameter while keeping all other parameters fixed.

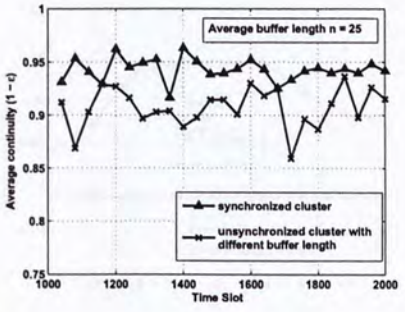


Figure 7.3: The playback continuity of two types of clusters when average buffer length $n = 25$.

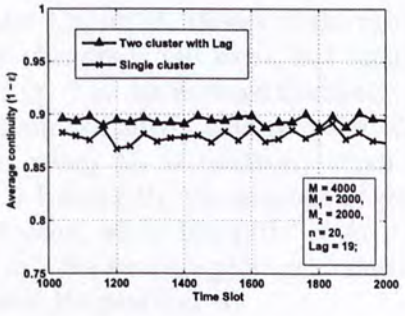


Figure 7.4: The playback continuity of single cluster and two smaller cluster during time slot $1000 - 2000$.

Exp. B.1: Performance under different buffer length: In this simulation, the total number of peers M is fixed to 4000 and the ratio of the population of two small clusters is fixed to 1, or $M_1 = M_2 = 2000$. In order to make sure that the second cluster could get the newest chunk with a high probability, we set $Lag = n - 1$. Fig. 7.4 shows the average playback continuity of both the single cluster and the two smaller clusters from time slot 1000 to 2000, with the buffer length $n = 20$ and the Lag is set to 19. In each time slot, the average playback continuity is the ratio of the number of peers doing the playback and the total number of peers in the cluster. Fig. 7.4 shows that the two smaller clusters with Lag perform *better* in terms of playback continuity than the single cluster. The result validate Proposition 9.

Experiment B.2: Performance under different population ratio in the two smaller clusters: For this experiment, the total number of peers in the single cluster is $M = 4000$, buffer

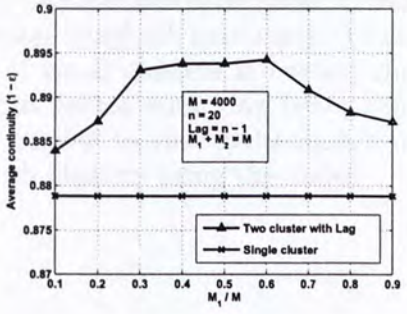


Figure 7.5: The playback continuity of single cluster and two smaller clusters with different population ratio.

length is $n = 20$ and $Lag = 19$. However, the ratio of the population of the two smaller clusters is not fixed, but rather vary from 0.1 to 0.9. As shown in Fig. 7.5, the average continuity of the two smaller cluster is better than the single cluster under all population ratio. One can find that when M_1 is relatively small or large compared to M (e.g. $\frac{M_1}{M} = 0.1$ and 0.9), the average playback continuity becomes poorer, whereas, when the ratio of M_1 to M_2 is close to 1 (e.g. $\frac{M_1}{M} \in [0.4, 0.6]$), the average playback continuity is quite good. This result validates Proposition 10.

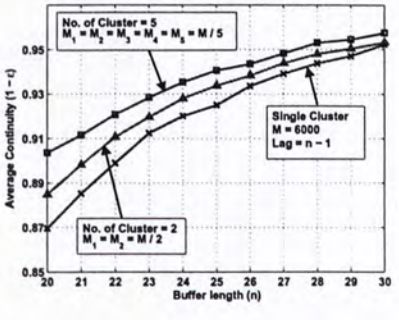


Figure 7.6: The playback continuity of single cluster, two small clusters and five smaller clusters.

Experiment B.3: Average playback continuity under different number of smaller clusters: In the simulation, the total number of peers is fixed as $M = 6000$, we vary the number of small clusters but all these small clusters have the same number of peers. Fig. 7.6 depicts the average playback continuity for the single cluster, two small cluster, and five smaller cluster. All three curves increase when the buffer length n increases. As indicated in Fig.

7.6, when the P2P system has more small clusters, the system will have a better average playback continuity. To analyze and seek the optimal number of small clusters is beyond the scope of this paper and part of our future work. We believe that, in real life, the topology is very similar to this multi-cluster topology, forming a multi-cast tree with clusters being the nodes.

□ **End of chapter.**

Chapter 8

conclusion

Summary

This is a summary of our work and contribution.

8.1 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 first model also lead to some practical algorithm that can be incorporated into well established systems as improvements.

In the last model, the P2P system is modeled as a synchronized system. In the second model, relaxed several assumptions in previous chapter to study the effect of unsynchronized peers. We define the terminology and formulated several different perspectives of looking at this problem. The conclusion is that under decentralized algorithms, there are reasons for forming synchronized clusters. However, in order to support a large number of peers with a fixed amount of buffer space, having different playback offsets for different clusters can help improve overall continuity.

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.

□ **End of chapter.**

Appendix A

Equation Derivation

Summary

Give equation proof in Appendix.

Appendix

Proof of Lemma 1: From Eq. (3.6), we have

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

From Eq. (4.1), 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. 4.1), 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. (3.6), we have

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

From Eq. (4.4), 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

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

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. (4.4), 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:

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

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 Proposition. ∎

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 Proposition. ∎

Proof of lemma 6: The proof is derived from the continuous model. In the continuous model, let $\epsilon = p(1)$, the necessary buffer length for each strategy can be derived through the model. For Greedy strategy, the continuous function can be simplified as:

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

Through solving the function, we can get the following equation:

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

let $\epsilon = p_1$, the derived buffer length is:

$$n_G = \frac{1}{\epsilon} + \ln(\frac{1-\epsilon}{\epsilon}) - \frac{\epsilon}{1-\epsilon}$$

The continuous function for RF strategy is unchange. The required buffer length n_{RF} is:

$$n_{RF} = \frac{1}{\epsilon} + 2\ln(\frac{1-\epsilon}{\epsilon}) - \frac{\epsilon}{1-\epsilon}$$

The results indicate that Greedy saves buffer length compared with RF if the continuity requirement is bigger than $1 - p_1$.

Proof of Proposition 3:[changed by yipeng] First, we derive the continuous form of Upper Bound from the discrete model.

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

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

The complexity of the buffer length n_{UB} is $O(\ln(\frac{1}{\epsilon}) + \ln(M))$. From the proof of lemma 4, we can get the complexity of buffer length for Rarest First Strategy is $O(\frac{1}{\epsilon} + \ln(M))$, while the complexity for Greedy Strategy is $O(\frac{1}{p(1)-\epsilon}(\ln(\frac{1}{\epsilon}) + \ln(M)))$. Complexities of buffer length for both Rarest First Strategy and Greedy Strategy are larger than Upper Bound's. However, for 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 discontinuity for Rarest First Strategy is λ. Therefore, the complexity of combined buffer length is $O(\frac{1}{1-\lambda} + \ln(M) + \frac{1}{\lambda-\epsilon}(\ln(\frac{1}{\epsilon}) + \ln(M)))$. Here, we can control the value λ in a range, such as from 0.2 to 0.4. Then, the complexity of Mixed strategy can be simplified $O(\ln(\frac{1}{\epsilon}) + \ln(M))$, which is the same with the Upper Bound's.

A. Proof of Lemma 8

Proof 4 *In [30], from the continuous model of Rarest First Strategy, we can obtain the closed form of the relationship between n (buffer size), ϵ ($= 1 - p(n)$, discontinuity) and M ($= \frac{1}{p(1)}$, number of peers in the system) Eq. (A.1).*

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

When we fixed M, the total number of peers in Eq. (A.1), the discontinuity ϵ only relates to buffer length n. In order to take a close look at the relationship between, we rewrite the formula, in which we define $\bar{\epsilon} = p(n) = 1 - \epsilon$ to denote the continuity and take the derivative on n, since now the part $-\ln\left(\frac{p(1)}{1-p(1)}\right) - \frac{p(1)}{1-p(1)}$ is just a constant and its derivative on n is zero. We derive the first and second derivative in Eq. (A.2) and (A.3).

$$\frac{d\bar{\epsilon}}{dn} = \bar{\epsilon} * (1-\bar{\epsilon})^2, \tag{A.2}$$

$$\frac{d^2\bar{\epsilon}}{dn^2} = 3\bar{\epsilon}(1-\bar{\epsilon})^2(\bar{\epsilon}-\frac{1}{3})(\bar{\epsilon}-1). \tag{A.3}$$

Because $\bar{\epsilon} = p(n)$ is the probability that the nth chunk position of the buffer is filled or not, $\bar{\epsilon}$ is a real number between 0 and 1. The following conclusions hold based on Eq. (A.4) and (A.5).

(i) The first derivative of $\bar{\epsilon}$ and n is always non-negative. In other words, the continuity function $p(n)$ related to buffer size is an non-decrease function.

(ii) When buffer size is small ($n < n^*$ and $\bar{\epsilon} < \frac{1}{3}$), the second derivative is positive, which means the continuity function $p(n)$ is convex.

(iii) When buffer size is large enough ($n > n^*$ and $\bar{\epsilon} > \frac{1}{3}$), the second derivative is negative, which means the continuity function $p(n)$ is concave.

$$\frac{d\bar{\epsilon}}{dn} \geq 0, \tag{A.4}$$

$$\frac{d^2\bar{\epsilon}}{dn^2} \begin{cases} > 0 & (0 < \bar{\epsilon} < \frac{1}{3}) \Longleftrightarrow (0 < n < n^*) \\ < 0 & (\frac{1}{3} < \bar{\epsilon} \leq 1) \Longleftrightarrow (n > n^*) \\ = 0 & (\bar{\epsilon} = \frac{1}{3}) \Longleftrightarrow (n = n^*), \end{cases} \tag{A.5}$$

where,

$$\begin{aligned} n^* &= \frac{3}{2} - \ln(2) - \ln\left(\frac{p(1)}{1-p(1)}\right) - \frac{p(1)}{1-p(1)} \\ &= O(\log(M)). \end{aligned}$$

C. Proof of Lemma 9

Proof 5 *Refer to Eq. (A.1), if the number of peers M varies, the discontinuity would vary accordingly when the buffer length n is fixed. Do differentiation to both sides of Eq. (A.1), one could derive Eq. (A.6).*

$$\frac{d\epsilon}{dp(1)} = -\frac{\epsilon^2(1-\epsilon)}{p(1)(1-p(1))^2} \approx -\frac{\epsilon^2(1-\epsilon)}{p(1)}. \tag{A.6}$$

We make the approximation based on the basic assumption that the total number of peers M is very large and in result, $1 - p(1) = 1 - \frac{1}{M} \approx 1$.

Then, assume that there are already M peers and one server in the cluster. Another server joins in and in each time slot it provides a chunk, which locate position D in peers' buffer. At position D, the system is equivalent to another baby network in [30], where the number of peers is $\frac{1}{p(D)}$, buffer length is $n - D$. From Eq. (A.6), one more server can improve the continuity:

$$\begin{aligned} a_1 &\approx \int_{p(D)}^{p(D)+\frac{1}{M}} -\frac{\epsilon^2(1-\epsilon)}{p(1)} dp(1) \\ &= -\epsilon^2(1-\epsilon)\ln(1+\frac{p(1)}{p(D)}), \end{aligned} \tag{A.7}$$

where $p(1) = \frac{1}{M}$. Because $p(D)$ is an increasing function with D, $|a_1|$ is decreasing when D increases. When $D = 1$, $|a_1|$ reaches the maximum value $\epsilon^2(1-\epsilon)\ln 2$. On the other hand, when $D = n$, the last position of the buffer, $p(n) = 1 - \epsilon$ is very close to 1 and $\ln(1 + \frac{p(1)}{1-\epsilon}) \approx \frac{p(1)}{1-\epsilon}$ ($p(1)$ is very small) at which point, the lower bound is achieved.

D. Proof of Proposition 9

Proof 6 *In order to see the improvement on the average continuity when there is a lag between two clusters, we need to consider a single larger cluster C, with $M = M_1 + M_2$ peers that have the same buffer length n and playback offset, to be the benchmark, as depicted in Fig. 6.5.*

In each time slot, C_1 sends a chunk, of which the position in the buffer is D, to C_2. For C_1, it is just as a server at position D leaves the cluster and for C_2, it is equivalent that there is a server sending one newest chunk in each time slot. And the condition for the larger cluster C is the same as described in [30].

From Eq. (A.6), the change of the ϵ comparing C and C_1 is in Eq. (A.8):

$$\begin{aligned} \Delta\epsilon_1 &\approx \int_{\frac{1}{M}}^{\frac{1}{M_1}} -\frac{\epsilon^2(1-\epsilon)}{p(1)} dp(1) \\ &= -\epsilon^2(1-\epsilon)\ln(\frac{M}{M_1}). \end{aligned} \tag{A.8}$$

And the change of the ϵ comparing C and C_2 is in Eq. (A.9):

$$\begin{aligned}\Delta\epsilon_2 &\approx \int_{\frac{1}{M}}^{\frac{1}{M_2}} -\frac{\epsilon^2(1-\epsilon)}{p(1)}dp(1) \\ &= -\epsilon^2(1-\epsilon)\ln(\frac{M}{M_2}). \end{aligned} \tag{A.9}$$

And the influence on the discontinuity when a server provide chunks at position D of the buffer leave is in Eq. (A.10):

$$a_1 \approx -\epsilon^2(1-\epsilon)\ln(1+\frac{p(1)}{p(D)})\Big|_{p(1)=\frac{1}{M_1}}. \tag{A.10}$$

The total change of discontinuity is in Eq. (A.11):

$$\begin{aligned}\Delta\epsilon &= \frac{M_1}{M}\Delta\epsilon_1 + \frac{M_2}{M}\Delta\epsilon_2 - \frac{M_1}{M}a_1 \\ &= -\epsilon^2(1-\epsilon)[\frac{M_1}{M}\ln\frac{M}{M_1} + \frac{M_2}{M}\ln\frac{M}{M_2} \\ &\quad -\frac{M_1}{M}\ln(1+\frac{1}{p(D)M_1})]. \end{aligned} \tag{A.11}$$

Then focusing on other terms of $\Delta\epsilon$, because $\frac{M_2}{M}\ln\frac{M}{M_2} > 0$ and

$$\begin{aligned}&\frac{M_1}{M}\Big(\ln\frac{M}{M_1} - \ln(1+\frac{1}{p(D)M_1})\Big) \\ =\ &\frac{M_1}{M}\ln(\frac{M_1+M_2}{M_1+\frac{1}{p(D)}}) > 0, \text{ when } p(D) > \frac{1}{M_2}. \end{aligned} \tag{A.12}$$

Under the assumptions that both M_1 and M_2 are large enough and $p(D)$ is close to 1, inequality (A.12) could be easily held and consequently $\Delta\epsilon$ is negative which means the continuity is increased. In addition, $|\Delta\epsilon|$ increases when D increases which indicates that the larger the D is, the more the improvement of the average playback continuity is.

E. Proof of Proposition 10

Proof 7 *In fact, the term $\ln(1+\frac{1}{p(D)M_1})$ in Eq. (A.11) could be simplified as:*

$$\begin{aligned}\ln(1+\frac{1}{p(D)M_1}) &\approx \frac{1}{p(D)M_1} \\ \Rightarrow\ -\frac{M_1}{M}\ln(1+\frac{1}{p(D)M_1}) &\approx -\frac{1}{p(D)M}. \end{aligned} \tag{A.13}$$

When we fix M and $p(D)$, $\Delta\epsilon$ in Eq. (A.11) becomes:

$$\Delta\epsilon(M_1) = A_1[\frac{M_1}{M}\ln\frac{M}{M_1} + \frac{M-M_1}{M}\ln\frac{M}{M-M_1} + A_2], \quad \text{(A.14)}$$

where, $A_1 = -\epsilon^2(1-\epsilon), A_2 = -\frac{1}{p(D)M}$ are not related to M_1. It could be derived that $|\Delta\epsilon|$ has a maximum value $|A_1(\ln 2 - A_2)|$, when $M_1 = M_2 = \frac{M}{2}$ which means when $\frac{M_1}{M_2} = 1$, the increment of the average continuity will be the largest.

□ **End of chapter.**

Bibliography

[1] "PPLive," http://www.pplive.com/.

[2] "ppstream," http://www.pps.tv/.

[3] "Planet-Lab," http://www.planet-lab.org/.

[4] S. Ali, A. Mathur, and H. Zhang. Measurement of commercial peer-to-peer live video streaming. In 1^{st} *Workshop on Recent Advances in P2P Streaming*, Aug. 2006.

[5] B. Cheng, H. Jin, and X. Liao. Rindy: A ring based overlay network for peer-to-peer on-demand streaming. In *UIC*, 2006.

[6] B. Cheng, X. Liu, Z. Zhang, and H. Jin. A measurement study of a peer-to-peer video-on-demand system. In *ICPCS*, Feb. 2007.

[7] C. Dana, D. Li, D. Harrison, and C. N. Chuah. Bass: Bittorrent assissted streaming system for video-on-demang. In *IEEE MMSP*, Oct. 2005.

[8] R. S. Dongyu Qiu. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM*, 2004.

[9] B. Fan, D. M. Chiu, and J. C. S. Lui. The delicate trafeoffs in bit torrent like file sharing protocol design. In *Proceedings of IEEE ICNP*, 2006.

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

[11] 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.

[12] X. Hei, C. Liang, Y. Liu, and K. W. Ross. Insights into pplive: A measurement study of a large-scale P2P iptv system. In *IPTV workshop in WWW2006*, May 2007.

[13] X. Hei, Y. Liu, and K. W. Ross. Inferring network-wide quality in p2p live streaming system. *to appear: IEEE Transactions on Multimedia*, November 2007.

[14] X. Hei, Y. Liu, and K. W. Ross. Stochastic fluid theroy for p2p streaming systems. In *Proceedings of INFOCOM*, 2007.

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

[16] Y. Huang, T. Fu, D. Chiu, and J. C. L. and. Challenges, design and analysis of a large-scale p2p-vod system. In *Sigcomm*, 2008.

[17] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng. Anysee: Peer-to-peer live streaming. In *IEEE INFOCOM*, Apr. 2006.

[18] 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.

[19] N. Magharei and R. Rejaie. Understanding mesh-based peer-to-peer streaming. In *NOSSDAV '06*, May 2006.

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

[21] M. Pinson and S. Wolf. A new standardized method for objectively measureing video quality. *IEEE Transaction on Broadcasting*, 50(3):312–322, Sep 2004.

[22] T. Piotrowski, S. Banerjee, S. Bhatnagar, S. Ganguly, and R. Izmailov. Peer-to-peer streaming of stored media: the indirect approach. *SIGMETRICS/Performance*, pages 371–372, 2006.

[23] D. Qiu and R. Srikant. Modeling and perfromance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of ACM Sigcomm*, Portland, OR, Aug 2004.

[24] T. Silverson and O. Fourmaux. P2p iptv measurement: A case study of tvant. In *ACM CONEXT '06*, 2006.

[25] T. Silverson and O. Fourmaux. P2p iptv measurement: A comparison study. University Paris 6 C LIP6/NPA Laboratory, Tech. Rep., Oct 2006.

[26] 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.

[27] L. Vu, I. Gupta, J. Liang, and K. nahrstedt. Mapping the pplive network: studying the impacts of media streaming on p2p overlays. Dept. of Computer Science, UIUC, Tech. Rep. UIUCDCS-R-2006-275, Aug 2006.

[28] M. Zhang, L. Zhao, Y. Tang, J. G. Luo, and S. Q. Yang. Large-scale live media streaming over peer-to-peer networks through global internet. *P2PMMS'05*, pages 21–28, 2005.

[29] X. Zhang, J. Liu, B. Li, and T. S. P. Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In *INFOCOM*, 2005.

[30] Y. Zhou, D. M. Chiu, and J. C. S. Lui. A simple model for analyzing p2p streaming protocols. In *Proceedings of IEEE ICNP*, 2007.