Challenges, Design and Analysis of a Large-scale P2P-VoD System

Yan Huang[*], Tom Z. J. Fu[†], Dah-Ming Chiu[†], John C. S. Lui[‡] and Cheng Huang[*]
[*]{galehuang, ivanhuang}@pplive.com, Shanghai Synacast Media Tech.
[†]{zjfu6, dmchiu}@ie.cuhk.edu.hk, The Chinese University of Hong Kong
[‡]cslui@cse.cuhk.edu.hk, The Chinese University of Hong Kong

ABSTRACT

P2P file downloading and streaming have already become very popular Internet applications. These systems dramatically reduce the server loading, and provide a platform for scalable content distribution, as long as there is interest for the content. P2P-based video-on-demand (P2P-VoD) is a new challenge for the P2P technology. Unlike streaming live content, P2P-VoD has less synchrony in the users sharing video content, therefore it is much more difficult to alleviate the server loading and at the same time maintaining the streaming performance. To compensate, a small storage is contributed by every peer, and new mechanisms for coordinating content replication, content discovery, and peer scheduling are carefully designed. In this paper, we describe and discuss the challenges and the architectural design issues of a large-scale P2P-VoD system based on the experiences of a real system deployed by PPLive. The system is also designed and instrumented with monitoring capability to measure both system and component specific performance metrics (for design improvements) as well as user satisfaction. After analyzing a large amount of collected data, we present a number of results on user behavior, various system performance metrics, including user satisfaction, and discuss what we observe based on the system design. The study of a real life system provides valuable insights for the future development of P2P-VoD technology.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Design, Measurement, Performance

Keywords

Peer-to-Peer/Overlay Networks, Video-on-Demand, Content Distribution

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.
Copyright 2008 ACM 978-1-60558-175-0/08/08 ...$5.00.

1. INTRODUCTION AND CONTRIBUTION

The effectiveness of using the P2P approach for content distribution has been proven by many deployed systems [1, 2, 3, 4, 9, 21]. These P2P systems offer many different services. One type of service is *P2P file downloading*, for example implemented by BitTorrent [9] and Emule [1]. When a file is downloaded by many users, these users help each other so that the server load is significantly reduced. The peers may experience different downloading rates, often depending on how much they are able to contribute to the process. Another type of service is *P2P live streaming* (for example implemented by a university project Coolstreaming [21] and many commercial systems such as PPLive [2]). When a live video is watched by many users, again these users can help each other to alleviate the load on the server. In this case, the new challenge to system design is to ensure all peers can receive the streamed video at the playback rate.

More recently, the interest has turned towards a new kind of service, P2P video-on-demand (P2P-VoD). Based on a detailed analysis of a current client-server VoD system at Microsoft, it was pointed out in [17] that P2P-VoD could bring significant savings in server loading. Apparently this conclusion was already shared among P2P-VoD developers, since a number of P2P-VoD systems were deployed at about the same time as the publication of [17]. These P2P-VoD systems are already enjoying a large viewer population [3, 4, 5, 6, 7]. Like P2P streaming systems, these P2P-VoD systems also deliver the content by streaming, but peers can watch different parts of a video at the same time, hence diluting their ability to help each other and offload the server. To compensate, this new genre of P2P systems requires each user to contribute a small amount of storage (usually 1GB) instead of only the playback buffer in memory as in the P2P streaming systems. This additional resource opens up vast new opportunities for arranging suitable patterns of content replication to meet diverse user demands. Essentially, the new system is a highly dynamic P2P replication system, plus a sophisticated distributed scheduling mechanism for directing peers to help each other in real time.

In this paper, we conduct an in-depth study of P2P-VoD based on a real-world P2P-VoD system built and deployed by PPLive in the fall of 2007. Since the deployment of this P2P-VoD service, the system has been keeping track of the number of users. As of late November 2007, a total of 2.2 million independent users had tried the system. A total of 3900 movies were published in November and December of 2007, with around 500 movies on-line simultaneously. In late January 2008, the number of simultaneous users reached over 150K and was still growing. The point is that this is a system with a reasonable scale, and there are valuable lessons to be learned by measuring and analyzing its behavior.

The organization of the paper is as follows. In Section 2, we first describe a general architecture and its important building blocks. This general model serves to provide a framework and taxonomy for studying different design issues in a P2P-VoD system; for example, strategies for replication, information search and lookup, peer service scheduling, and other building blocks. In Section 3, we discuss metrics for evaluating a P2P-VoD system, and how to instrument the measurement in a real-life system. Real-life measurement data were collected from the deployed PPLive VoD system. In Section 4, we show the collected data, and analyze user demand, the effectiveness of the system design (for example the replication and transmission scheduling strategies) to satisfy user demand, and user satisfaction. Finally, we discuss related work and conclusions.

2. DESIGN AND BUILDING BLOCKS

In this section, we describe the general architecture of a P2P-VoD system in terms of various building blocks. In the process, we explain many specific design decisions in the PPLive P2P-VoD system, as specific examples. We also compare the P2P-VoD building blocks to those of P2P file downloading and streaming: there are some similarities, but also some notable differences.

2.1 Major components of the system

Similar to many P2P file sharing or streaming systems, a P2P-VoD system has the following major components: (a) a set of servers as the source of content (e.g., movies); (b) a set of trackers to help peers connect to other peers to share the same content; (c) a bootstrap server to help peers to find a suitable tracker (e.g. based on which geographical region the peer is located), and to perform other bootstrapping functions; (d) other servers such as log servers for logging significant events for data measurement, and transit servers for helping peers behind NAT boxes. These servers are typically provided by the P2P-VoD operator.

The other major component, of course, is the set of peers. They typically run software downloaded from the P2P-VoD operator. The P2P-VoD peer software comes with protocols to talk to all the servers above, as well as protocols to talk to other peers to share content. The peers also implement DHT (distributed hash table) function to back up certain bootstrapping servers.

2.2 Segment sizes

In the design of a P2P-VoD system, a fundamental decision is about segmentation of content, or how to divide a video into multiple pieces. There are many considerations for making this decision. From a scheduling point of view, it is desirable to divide the content into as many pieces as possible (i.e., small segment size), so that it gives the most flexibility to schedule which piece should be uploaded from which neighboring peer. This is specially so when peers all have different upload capacity. From the overhead point of view, the larger the segment size the better, to minimize overheads. There are several types of overheads including: (a) Each piece of content comes with some *header* to describe the content, for example its sequence number and timestamp, and authentication information. The larger the segment, the smaller the header overhead. (b) Each peer needs to let other (neighboring) peers know which pieces it is holding. This information is usually represented by a bitmap, for ease of processing. The larger the segment size, the smaller the size of the bitmap, hence this is advertizing overhead. (c) In order for a peer to get a piece of content from another peer, there will be some protocol overhead, in terms of request packets or other protocol packets. The larger the segment size the smaller is such protocol overheads. A third perspective is due to the real-time nature of streaming. The video player expects a certain minimum size for a piece of content to be viewable (so a viewable piece always consists of multiple packets), and such viewable units must be delivered to the player with deadlines. By making these units (exchanged between the transport and the player) too large, it increases the chance that the transport fails to collect a complete viewable unit of content before the deadline.

Segment	Designed for	Size
movie	entire video	> 100MB
chunk	unit for storage and advertisement	2MB
piece	unit for playback	16KB
sub-piece	unit for transmission	1KB

Table 1: Different units of a movie

Due to these conflicting requirements, there are three levels of segmentation of a movie in PPLive's VoD system, as defined in Table 1.

The size of *piece* is dictated by the media player and a size of 16KB is chosen. PPLive uses the WMV format for video encoding. The source video rate is usually between 381 to 450 Kbps. For high-definition video, the rate can go up to 700 Kbps or higher. Using the 16KB size, a piece will contain a viewable segment as long as the source rate is less than 1.4 Mbps.

The size of *piece* is too large for efficient scheduling of transmission, so *sub-piece* is used. If piece is advertised to other peers, then a bitmap of thousands of bits would be needed (e.g., a 2GB movie would need a bitmap of size 64K bits. So *chunk* is defined and used for the purpose of advertizing to neighbors what parts of a movie a peer holds. In summary, a movie is composed of a number of chunks, a chunk is composed of a number of pieces, while a piece is composed of a number of sub-pieces.

Given these choice of segment sizes, PPLive experiences an overhead rate of 6.2%, considering all three types of overheads mentioned above, but assuming operating under perfect conditions with no losses, no unnecessary requests and no duplicate transmissions. Under real-life network conditions, the average overhead rate is about 10%.

2.3 Replication strategy

Each peer is assumed to contribute a fixed amount of hard disc storage (e.g., 1GB). The entire viewer population thus forms a distributed P2P storage (or file) system. A chunk is the basic unit for storing movies on a hard disc. Only when all the pieces in a chunk are available locally, the chunk is advertised to other peers.

The goal of the replication strategy is to make the chunks as available to the user population as possible to meet users' viewing demand while without incurring excessive additional overheads. This is probably the most critical part of the P2P-VoD system design. There are many possible replication strategies, many exploiting the various user demand characteristics. This is an important area for continued research.

The first design issue is whether to allow multiple movies be cached if there is room on the hard disc. If so, a peer may be watching one movie while providing uploading to another movie at the same time. This is referred to as *multiple movie cache* (MVC) rather than *single movie cache* (SVC). The design of SVC is simpler, but MVC is more flexible for satisfying user demands and is the choice by PPLive VoD.

The next important design consideration is whether to pre-fetch or not. Without pre-fetching, only those movies already viewed locally could possibly be found in a peer's disk cache. While Pre-fetching may improve performance, it may also unnecessarily waste precious peer uplink bandwidth. Also, for ADSL (commonly found in China), a peer's capacity to provide upload can be affected if there is simultaneous downloading. Furthermore, it is observed that the visit duration for the majority of peers is currently no more than one hour, which increases the risk of wastage. For these reasons, the design choice is no pre-fetching.

Another important choice by the replication algorithm is which chunk/movie to remove when the disk cache is full. In PPLive's case, this decision is primarily made on a movie basis. This means once a movie has been chosen as the next one to go, all the chunks of the movie immediately become candidates for removal one by one. Doing it at a chunk level would incur more overheads (for collecting necessary information about different chunks). How is the next movie picked? The favorite choices by many caching algorithms are *least recently used* (LRU) or *least frequently used* (LFU). Indeed, LRU is the original choice in PPLive VoD. After further studies, the simple LRU is replace by a weight-based evaluation process.

Each movie is assigned a weight based on primarily two factors: (a) how complete the movie is already cached locally; (b) how needed a copy of the movie is. The *need* level is determined by the *availability to demand* ratio (ATD). Suppose a movie is cached (including being viewed) by c peers and being viewed by n peers; then the ATD is c/n. The *need* of a movie is then defined as a decreasing function of its ATD, reaching a maximum value for all ATD beyond 6 (or 8). The value of this threshold (6-8) is determined by the prevailing uplink bandwidth contributed by peers, normalized by the source bitrate. For current situation in China, many peers have relatively low uplink bandwidth to contribute, therefore it takes 6-8 peers to offload the source (server).

The ATD information for weight computation is provided by the tracker. So the implementation of the weight-based replication strategy incurs additional overheads. This overhead depends on how often the caching decision is made. In current systems, the average interval between caching decisions is about 5 to 15 minutes, so this is not a significant overhead. The benefit of weight-based replication over LRU is significant. It improves the server loading from 19% down to a range of 11% to 7%. This is the biggest performance improvement achieved by a design change.

More detailed discussion of how to measure the effectiveness of the replication algorithms, will be discussed in section 3. Measurement results and analysis will be included in section 4.

2.4 Content Discovery and Peer Overlay Management

It is not enough to have good replication of content - peers must also be able to discover the content they need and which peers are holding that content. The challenge is to accomplish this with the minimum overhead. Without exception, P2P systems rely on the following methods for content advertising and look-up: (a) tracker (or super node); (b) DHT; (c) gossiping method. These methods provide different levels of availability, freshness and robustness, with commensurate levels of overhead. In PPLive VoD, all these mechanisms are used to some extent, depending on the different requirements for the information.

Trackers are used to keep track of which peers replicate a given movie (or part of that movie). As soon as a user (peer) starts watching a movie, the peer informs its tracker that it is replicating that movie; conversely, a peer also tells its tracker when it no longer holds a movie in its cache. When a peer wants to start watching a movie, it goes to the tracker to find out which other peers have that movie. Those other peers become this peer's *neighbors*.

The information about which chunks a peer has is kept in a *Chunk Bitmap*. A peer asks its neighbors for their Chunk Bitmaps. Based on this information, it selects which neighbor to download from. So discovering where chunks are is by the *gossip* method. This cuts down on the reliance on the tracker, and makes the system more robust. Even if the tracker is not available, a peer can switch to the gossip mode to find other peers watching the same movie.

In fact, each peer also regularly sends keep-alive messages to a tracker to report its chunk bitmap and other statistics. This information is collected for monitoring and management purposes, rather than for operational reasons. We will describe how this information is used to compute a (replication) *health index*.

Originally, DHT (implemented by tracker nodes) is used to automatically assign movies to trackers to achieve some level of load balancing. In later versions, peers also implement DHT so as to provide a non-deterministic path to the trackers. This prevents the trackers to be possibly blocked by some ISPs.

2.5 Piece selection

A peer downloads chunks from other peers using a *pull* method. For P2P-VoD, there are three considerations for selecting which piece to download first:

1. *sequential*: Select the piece that is closest to what is needed for the video playback.

2. *rarest first*: Select the piece that is the rarest (usually the newest piece in the system). Although it seems counter-intuitive for streaming, selecting the rarest piece helps speeding up the spread of pieces, hence indirectly helps streaming quality. This strategy tends to help the system scale, which is clearly explained in [22].

3. *anchor-based*: In VoD, users may skip parts of a movie and jump forward (backward). To support such VCR features, a number of video *anchor* points are defined

for a movie. When a user tries to jump to a particular location in the movie, if the piece for that location is missing then the closest anchor point is used instead.

In PPLive's system, a mixed strategy is used, giving the first priority to sequential, then rarest-first. The anchor-based method is not used in current design for two reasons. (a) From current experience, users do not jump around much. On average, only 1.8 times per movie is observed. (b) By optimizing the transmission scheduling algorithm, the initial buffering time after a jump can be reduced to an acceptable level[1] without implementing anchoring. For these reasons, the anchoring idea is still under study for future implementation.

2.6 Transmission strategy

After selecting a particular chunk to download, suppose this chunk is available at a number of neighbor peers, how to select which neighbor to download from? How many neighbors to use for simultaneous download? How to schedule requests and set timeouts to multiple neighbors for simultaneous download? All these are accomplished by the transmission scheduling algorithm.

There are two (sometimes conflicting) goals in designing the transmission algorithm: (a) maximize (to achieve the needed) downloading rate; (b) minimize the overheads, due to duplicate transmissions and requests.

In a data-driven overlay, the neighbors a peer connects to can be highly dynamic, since each neighbor may be answering to multiple requests at a time. So a peer must constantly juggle how to send download requests to different neighbors and how to deal with timeouts. There are different levels of aggressiveness: (i) a peer can send a request for the same content to multiple neighbors simultaneously, to ensure it gets the content in time; (ii) a peer can request for different content from multiple neighbors simultaneously; when a request times out, it is redirected to a different neighbor; (iii) work with one neighbor at a time; only when that neighbor times out, try to connect to a different neighbor.

Strategy (i) is very aggressive for achieving the deadline for downloads, but invariably generates duplicate transmissions. Strategy (iii) is very conservative in resource utilization. Both strategy (ii) and (iii) may still general duplicate transmissions because of timeouts, but the likelihood is much lower than (i). PPLive VoD's transmission algorithm is based on strategy (ii). In implementing strategy (ii), the algorithm tries to proportionally send more requests to the neighbor based on response time. A critical parameter for tuning is the number of simultaneous neighbors to send requests to. For playback rate of around 500Kbps, our experience is that 8-20 neighbors is the sweet spot. More than this number can still improve the achieved rate, but at the expense of heavy duplication rate. If the desired rate is 1Mbps, then 16-32 simultaneous neighbors tends to provide the best result. These numbers are highly empirical, depending on the prevailing uplink bandwidth of peers and many other factors. Overall, how to design the best transmission algorithm is an interesting topic for further research.

Finally, it should be pointed out that when the neighboring peers cannot supply sufficient downloading rate, the content server can always be used to supplement the need.

2.7 Other design issues

After describing the basic design for normal networking conditions, we describe a number of mechanisms designed to deal with abnormal operating conditions. These include: incentives for contribution; traversing NAT and firewalls; and content authentication.

It is well-known that the P2P file downloading protocol BitTorrent [9] uses *tit-for-tat* as incentive to induce peers to help each other. In P2P streaming, this does not work since many peers cannot contribute uploading bandwidth greater than or equal to the playback rate. So what incentives are used? In the PPLive system, the users do not have any built-in controls for adjusting their contribution levels. In order for the software to continue to deliver content for playback, the client software must regularly advertise its chunk bitmap to the tracker; else playback would be automatically turned off.

Another impediment to P2P overlay networks is the NAT boxes and firewalls. The PPLive VoD system uses standard methods[2] for peers to discover different types of NAT boxes on their path to the Internet, and advertize their addresses accordingly. This is quite necessary in the current Internet since about 60%-80% of peers are found to be behind NAT. We have included some data to show the distribution of the NAT types in section 4.

Also, a significant number of peers are protected by firewalls. For this reason, the PPLive software carefully pace the upload rate and request rate to make sure the firewalls will not consider PPLive peers as malicious attackers.

It is important for the P2P-VoD system to include mechanisms to authenticate content, so that the system is resistant to *pollution* attacks [10]. Such authentication can be implemented based on message digest or digital signature. In the case of a proprietary system, it is not difficult for the operator to implement some means to distribute a key for each movie.

Authentication can be done at two levels: chunk level or piece level. If chunk level, authentication is done only when a chunk is created and is stored to the hard disc. In this case, some pieces may be polluted and cause poor viewing experience locally at a peer. However, further pollution is stopped because the peer would detect a chunk is bad and discard it. The advantage of the chunk level authentication is its minimal overhead.

Chunk-level authentication has at least two significant drawbacks. Sometime, polluted pieces may cause more damage than poor viewing experience, for example it may crash or freeze the player. Secondly, chunk is a rather large segment of content. There is some non-zero probability that a piece is bad not due to pollution; but this would cause the entire chunk to be discarded. For these reasons, it is wise to do authentication at the piece level to ensure good video quality. In the current version of PPLive VoD, a weaker form of piece level authentication is also implemented, leveraging on the same key used for chunk level authentication.

3. PERFORMANCE METRICS AND MEASUREMENT METHODOLOGY

An important goal of this paper is to study a large-scale real-life P2P-VoD system through measurement. A large

[1] In recent tests, the average buffering time is around 18 seconds.

[2] Similar to the STUN protocol.

User ID	Movie ID	Start time	End time	Start pos.

Table 2: MVR format

amount of information was measured, and we will focus on those aspects most interesting. In this section, we first describe what we try to measure and the metrics used for measurement. Then we explain how we collected the data.

3.1 What to measure

What we measure can be roughly grouped into three areas:

1. *User behavior*: This includes the user arrival patterns, and how long they stayed watching a movie. One major difference between VoD and live streaming is that users may not be watching the same thing at the same time, but there is often some overlap (for example, they are watching different parts of the same movie). Another important difference is in VoD users can jump from one position to another in a movie, while skipping the content in between. Understanding these kinds of user behavioral information can be used to improve the design of the replication strategy.

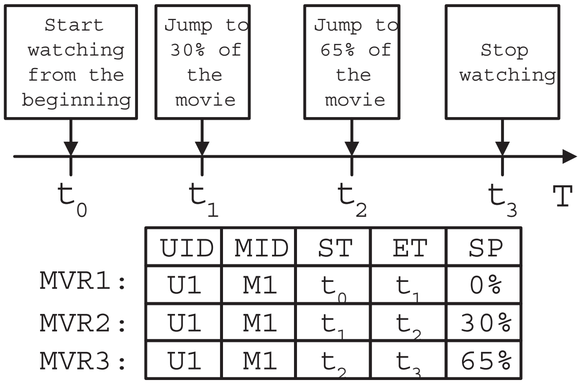
2. *External performance metrics*: This includes *user satisfaction* and *server load*. Server load can be objectively defined in terms of CPU, memory, and bandwidth resources needed from the server at different loading (number of peers). The definition of user satisfaction is more subjective. These metrics are used to measure the system performance perceived externally.

3. *Health of replication*: In comparison to P2P streaming or file downloading, replication is a new aspect of P2P technology designed for VoD. It is therefore particularly important and interesting to design yardsticks to measure how well a P2P-VoD system is replicating a content. This is an *internal metric* used to infer how well an important component of the system is doing.

3.2 Measuring User Behavior

In a P2P-VoD system, a typical user does not just watch a movie sequentially, but rather, he is allowed to jump from one movie to another; and when viewing one movie, is allowed to skip parts of the movie and jump to a new position in the movie. Therefore, the most basic user activity is the continuous viewing of a stretch of a movie. This basic activity is recorded in a *movie viewing record* (MVR). The important parts of an MVR record are shown in Table 2, where ST is the starting time, ET is the ending time, and SP is the starting position.

Each user has a unique User ID, which is included as part of the MVR. To ensure uniqueness, part of this ID is derived from the hardware component serial numbers (HCSN) of the computer (or the memory module on that computer) that is running the P2P-VoD client software. Each movie also has a unique ID, which is usually a hash of the movie content. The MVR records the starting time (ST) and ending time (ET), as well as the starting position (SP) of a particular continuous viewing activity. Based on these three numbers, the ending position can also be computed. In most cases, as soon as a user finishes recording one MVR, a new MVR is initialized to record the next viewing activity.



	UID	MID	ST	ET	SP
MVR1:	U1	M1	t_0	t_1	0%
MVR2:	U1	M1	t_1	t_2	30%
MVR3:	U1	M1	t_2	t_3	65%

Figure 1: Example to show how MVRs are generated

Figure 1 illustrates how MVRs are generated based on a sequence of user viewing activities. In this example, the user's ID is U1 and the movie's ID is M1. The user starts to watch the movie at t_0 from the beginning of the movie. After watching for a duration of $(t_1 - t_0)$, he/she jumps to position 30% of the movie (i.e., if the length of the movie is 5000 seconds, this user jumps to the 1500th second of the movie). At t_2, the user jumps again to start watching at position 65% (or the 3250th second of the 5000-second movie) and he stops watching at t_3. As a result of this sequence of activities, three MVRs are generated as shown in Figure 1.

Clearly, a complete user behavior information can be represented by these MVR records. We explain later how this information is collected.

3.3 User satisfaction

From the MVRs, we can determine the total viewing time of each user for each movie. A very crude statistic for the P2P-VoD service is the *total viewing time* (TVT) for all users and all movies.

TVT gives us the total amount of service provided by the P2P-VoD system, but it does not tell us the level of user satisfaction. How to measure user satisfaction is in itself an interesting research problem.

Let us first consider a simple version of user satisfaction. Given an MVR, actually part of the duration between start time and end time is not spent on viewing, but on buffering. Assume this information is captured by the P2P-VoD client software together with the MVR, denoted as BT (for buffering time). Let $R(m,i)$ denote the set of all MVRs for a given movie m and user i, and $n(m,i)$ the number of MVRs in $R(m,i)$. Let r denote one of the MVRs in $R(m,i)$. Then we can define the *fluency* $F(m,i)$ for a movie m and user i to be:

$$F(m,i) = \frac{\sum_{r \in R(m,i)} (r(ET) - r(ST) - r(BT))}{\sum_{r \in R(m,i)} (r(ET) - r(ST))}. \qquad (1)$$

In simple words, $F(m,i)$ measures the fraction of time a user spends watching a movie out of the total time he/she spends waiting for and watching that movie.

Ideally, we want something more refined than *fluency* to guage the user satisfaction. For the time spent viewing a movie, a user may not be satisfied with the quality of the delivery of the content[3]. Let us go one step further and assume that the user gave a *grade* for the average viewing quality for an MVR r, denoted as $r(Q)$. Let the value of the grade be in the range of $[0, 1]$. The fluency expression in Eq. (1) can be rewritten in terms of the contribution of each MVR:

$$F(m,i) = \sum_{k=1}^{n(m,i)} W_k, \qquad (2)$$

where each k indexes a particular MVR for movie m and user i, and

$$W_k = \frac{(r_k(ET) - r_k(ST) - r_k(BT))}{\sum_{r \in R(m,i)} (r(ET) - r(ST))}.$$

Now, we can define a more sophisticated version of user satisfaction index, $S(m,i)$, as:

$$S(m,i) = \sum_{k=1}^{n(m,i)} W_k r_k(Q). \qquad (3)$$

To illustrate, consider the example in Fig. 1, and assume there is a buffering time of 10 (time units) for each MVR. The fluency can be computed as:

$$F = \frac{(t_1 - t_0 - 10) + (t_2 - t_1 - 10) + (t_3 - t_2 - 10)}{(t_3 - t_0)}.$$

Suppose the user grade for the three MVR were 0.9, 0.5, 0.9 respectively. Then the user satisfaction index can be calculated as:

$$S = \frac{0.9(t_1 - t_0 - 10) + 0.5(t_2 - t_1 - 10) + 0.9(t_3 - t_2 - 10)}{(t_3 - t_0)}.$$

In reality, it is not likely (or even possible) for the P2P-VoD software to get explicit user feedback for his viewing experience. Instead, what can be done is for the P2P-VoD client software to *infer/estimate* user satisfaction for each MVR based on user actions. For example, if the user's viewing duration for the MVR exceeds a certain minimum time T_{min}, that may indicate that the user is basically satisfied with the viewing and the subsequent jump is not due to poor viewing quality but due to content reasons. Another example is if the termination of the viewing is due to a manual control to "STOP" the viewing altogether, it may also be inferred that the user is likely to be terminating the viewing session for some other activities rather than poor viewing quality. Based on these kinds of additional events and inferences, it is possible to estimate a suitable value for $r(Q)$. A detailed study of this topic, however, is beyond the scope of this paper. It is a good topic for further research. For the data analysis in the next section, we simply take fluency as the indicator for user satisfaction.

3.4 Health of Replication

The health index (for replication) can be defined at three levels:

a. *Movie level*: For a given movie m, the movie level health index is defined simply as the number of active peers who have advertised storing chunks of that movie. This is basically the information that the tracker collects about movies.

b. *Weighted movie level*: The movie level index is very coarse. Some peers may store a very small part of a movie but are still counted towards the index. So the weighted movie level index takes the fraction of chunks a peer has into account in computing the index. If a peers stores 50 percent of a movie, it is counted as 0.5.

c. *Chunk bitmap level*: The movie level indexes do not show how well individual chunks are replicated. The chunk level health index is in fact a vector representing the number of copies each chunk of a movie is stored by peers in a P2P-VoD system. Given the chunk level health index, various other statistics can be computed. For example, the average number of copies of a chunk in a movie; the minimum number of chunks; the variance of the number of chunks, and so on.

In this study, we instrumented the P2P-VoD system to collect the chunk level health index information, to be shown in the next section.

3.5 Measurement Methodology

We now explain how measurement data are collected in the P2P-VoD system we studied. The general mechanism is supported by a log server that collects various sorts of measurement data from peers. Sometimes, the measurement data are part of information that the tracker collects, for example the chunk replication information. In that case, peers would send the collected information to the tracker, and the tracker can then aggregated the information and pass it on to the log server.

Generally speaking, to avoid a large amount of traffic and a large number of interruptions on the log server, peers collect data and do some amount of aggregation, filtering and pre-computation before passing them to the log server. For example, peers do not report individual MVRs as they are generated. Instead, a peer sends a single report to the log server when the user generates a "STOP" event (pressing the STOP button, changing to another movie or turning off the client software). Furthermore, the report also includes the user satisfaction index (in this case the fluency) computed by the local peer. Based on these MVRs, various other user behavior information can be deduced by post processing, either as online or offline.

For replication health index, a peer needs to report its chunk bitmap to the log server whenever one of the following events occurs:

1. Some of the chunks or a whole movie is removed from the storage due to the replication strategy.

2. The local user starts to watch a new movie, and chunks of the new movie are added to local storage.

3. A refresh timer (pre-defined, e.g. 10 minutes) goes off.

4. MEASUREMENT RESULTS AND ANALYSIS

In this section, we present the measurement and data analysis of the P2P-VoD system in PPLive. We summarize the measurement results into five categories, namely, statistics for the video objects, user behavior, system health index, user satisfaction index, and server loads.

[3] Note, this is different than when the user is not happy with the content itself.

4.1 Statistics on video objects

We have collected the data trace on ten movies from the P2P-VoD log server. As mentioned before, whenever a peer selects a movie for viewing, the client software creates the MVRs and computes the viewing satisfaction index, and these information are sent to the log server. The collection of MVRs of a particular movie constitutes the data trace of that movie. All these data traces were collected from December 23, 2007 to December 29, 3007 (about one week worth of trace). For the ease of presentation, we select three "*typical*" movies to illustrate the measurement results. Table 3 lists the overall statistics of these three typical movies.

Movie Index:	Movie 1	Movie 2	Movie 3
Total Length (in sec)	5100s	2820s	6600s
No. of Chunks	121	67	151
Total No. of MVRs	56157	322311	15094
Total No. of MVRs with Start Position = 0 (or # of unique viewers)	35160	95005	8423
Ave. # of Jump	1.6	3.4	1.8
Ave. viewing Duration for a MVR	829.8s	147.6s	620.2s
Normalized viewing Duration (normalized by the movie duration)	16.3%	5.2%	9.4%

Table 3: Overall statistics of the three typical movies.

Based on these statistics, we have the following observations:

1. Given that the size of a chunk is about 2 MBytes (assuming the playback rate is about 380kbps), this implies that the viewing duration of a chunk is approximately 40 seconds. Movie 2 is the smallest video object with a viewing duration of about 45 minutes, while Movie 3 is the longest video object with a viewing duration of about 110 minutes.

2. To determine the most popular movie, we count only those MVRs whose starting position (SP) is equal to zero (e.g., MVRs which view the movie at the beginning). From the measurement, one can determine that Movie 2 is the most popular movie with 95005 users while Movie 3 is the least popular movie with 8423 users.

3. One interesting statistics we like to extract is the average number of jumps for a given movie. Note that a peer generates at least one MVR (with starting position being zero) and possibly a number of MVRs due to viewing jumps. Therefore, the average number of jumps for a given movie is approximately equal to the total number of MVRs divided by the total number of MVRs with starting position being zero. Based on this computation, we can conclude that Movie 2 has the highest average number of jumps (3.4) while Movie 1 has the lowest average number of jumps (1.6).

4. Each MVR indicates a viewing activity and one interesting characteristics is to determine the viewing duration per viewing activity. One can extract this information by computing the difference between the end time (ET) and start time (ST) of each MVR; by averaging over all MVRs of a movie, we obtain the average viewing duration per viewing action. Table 3 shows that Movie 1 has the largest viewing duration (829.8s), and this is *consistent* since Movie 1 has the least average number of jumps.

5. From the derived statistics mentioned above, we can also derive the normalized viewing duration (or average viewing duration divided by the movie length) and this is listed in the last row of Table 3, which shows that for Movie 1, each viewing length lasts on average 16.3% of the movie duration, while for Movie 2, each viewing length is around 5.2% of the movie duration.

4.2 Statistics on user behavior

4.2.1 Interarrival time distribution of viewers

One characteristic we are interested to study is the viewers' interarrival time to a particular movie. Again, this can be derived from the trace. Given a particular movie, one can extract all those MVRs with a start time (ST) equal to zero. These MVRs represent viewers who start to watch the video object from the beginning. Given this set of MVRs, we can sort them in an increasing order based on the start time (ST) field. The differences of the ST fields between to consecutive MVRs represent the interarrival times of viewers.

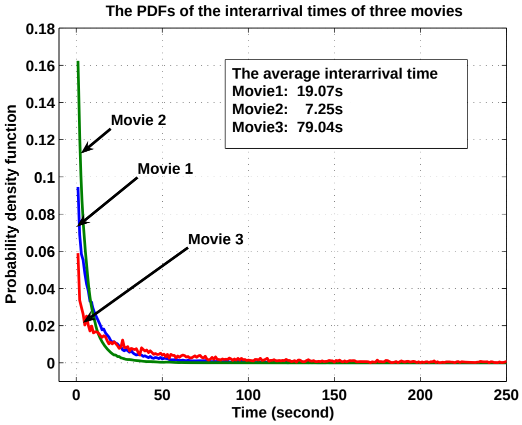


Figure 2: PDFs of the interarrival times of Movie 1, 2 and 3.

Figure 2 represents the probability density functions (PDFs) of the interarrival time distributions of Movie 1, 2 and 3. From the figure, we can observe that Movie 2 is the most popular movie (also can be verified by data in Table 3) and the average interarrival time between viewers is about 7.25s, while the interarrival times for Movie 1 and 3 are 19.07s and 7.25s respectively.

We can easily represent the PDF of the interarrival time by a simple mathematical model. In here, we use the following function to approximate the PDF:

$$f(t) \quad = \quad at^b \quad \text{where } a > 0, b < 0. \qquad (4)$$

For Movie 1, we have $a = 0.1152$, $b = -0.7894$ and the root mean square error (RMSE) is 0.0032. For Movie 2, we have $a = 0.1893$, $b = -0.9283$ and the RMSE is 0.0051. For Movie 3, we have $a = 0.0647$, $b = -0.7052$ and the RMSE is 0.0012. Figure 3 illustrates the PDF of all three movies and the corresponding $f(t)$.

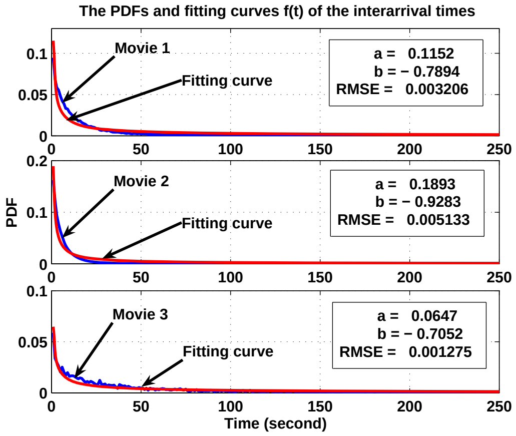


Figure 3: Mathematical models for the PDF of interarrival times of Movie 1, 2 and 3.

4.2.2 View duration distribution, residence distribution and start position distribution

When viewing a movie, users can jump to different positions of the movie. The knowledge of which positions users tend to jump to is important because it can help us design where to put the anchor points for a movie. For this purpose, it also helps to know the distribution of the viewing durations of MVRs.

Figure 4 illustrates the cumulative distribution function (CDF) of the viewing duration of the three movies. Note that the viewing duration is measured in terms of MVRs collected from the log server. As discussed before, when a user drags the track-bar to a different position of the movie (a jump event), or when the user presses the STOP button to stop watching the movie, or when the user changes to another movie or programme, or when the user turns off the software, any one of these operations will generate an MVR and the duration of this MVR is just the difference between the End Time (ET) and the Start Time (ST). From Figure 4, we can observe that a very high percentage of MVRs are of short duration (e.g., less than 10 minutes). This implies that for these three movies, the viewing stretch is of short duration with high probability.

Given the above statistics, we want to find out whether peers can assist each other in the movie viewing. To answer this question, we examine the residence distribution of peers. Residence measures how long a peer stays in a P2P-VoD system, and it is the time interval when a user activates the P2P-VoD client software and that the client successfully contacts the tracker server, to the time when a user turns off the client software. Figure 5 shows the residence distribution of peers staying in the P2P-VoD system during the one week measurement period. From this figure, we can observe that there is a high fraction of peers (i.e., over 70%) which stays in the P2P-VoD system for over 15 minutes, and these peers provide upload services to the community.

Based on the observation on Figure 4 and Figure 5, we can conclude that:

- Although the length of these three movies are different, users tend to watch Movie 1 with a higher viewing duration.
- Most of the viewing duration of the MVRs are short, e.g., less than 10 minutes.
- There is a high percentage (e.g., over 70%) of users who stay in the P2P-VoD system for longer than 15 minutes. This is significant because even though the duration of an viewing action is short, but since peers stay in the system for a long duration, they still assist each other in viewing movie, and this translates to good system scalability.
- From the above two points, we infer that when users start the P2P-VoD client software, they probably first quickly scan a few movies or programmes until they find an interesting one and continue to watch that movie; or just leave the system if nothing is of their interest. This process would cause a large number of short duration MVRs. After they find a movie that they are interested in, they might watch it for a long duration, and this could explain why there is a certain fraction of users stay in the system for more than one hour.

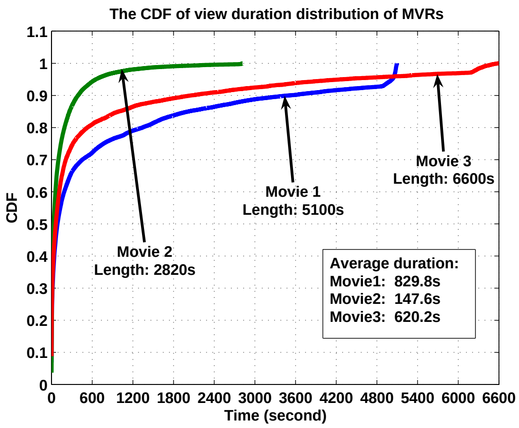


Figure 4: The CDF of view duration distribution of MVRs.

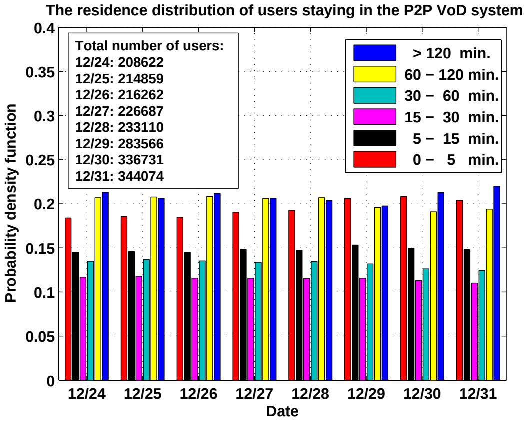


Figure 5: The residence distribution of users staying in the P2P-VoD system.

Figure 6 illustrates the CDF of the viewing position of MVRs in these three movies. Since users who watch Movie 2 are more likely to jump to some other positions than users who watch Movie 1 and 3, the probability that a viewing action starts from the beginning of Movie 2 is relatively low. Another important observation is that beside the starting position 0, the landing points of various jump operations is uniformly distributed. This implies that one can *uniformly space* the anchor points and these measured data provide developers a strong hint on designing the proper chunk selection algorithm.

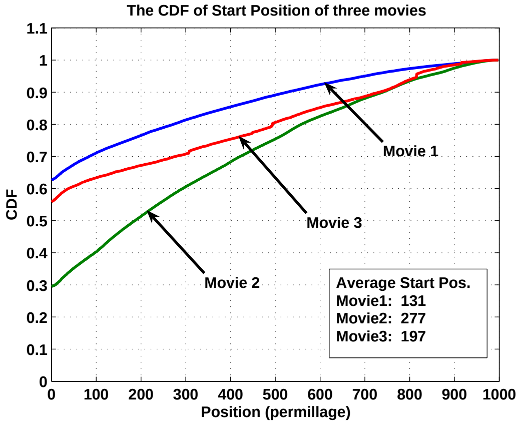


Figure 6: The CDF of the viewing position of movies.

4.2.3 *Analysis of peer population watching a movie*

One important data that the system needs to keep track is the viewer population size and how it varies in time. We carried out the measurement from 00:00, December 24, 2007 to 23:00, December 29, 2007 (total of six days). To reduce the size of the collected data, sampling is carried out in an hourly basis.

Figure 7 shows the total number of viewing activities (or MVRs) at *each* sampling time point. While Figure 8 shows the total number of viewing activities (or MVRs) that occurs *between* two sampling points. Both of these figures provide some indications on the size of viewing population.

It is interesting to observe from both figures that there is a *"daily periodicity"* of user behavior. There are two daily peaks, which occur at around 2:00 P.M. and 11:00 P.M. This may imply that users tend to subscribe to the P2P-VoD service at the end of the lunch break or late in the evening. We also observe that the viewing population drops at around 8:00 A.M. to 9:00 A.M. This may imply that users are usually on their way to work around this time of the day.

4.3 Health index of Movies

Health index is used to reflect the effectiveness of the content replication strategy of a P2P-VoD system. In this subsection, we focus on the health index of the three typical movies. Measurements were carried out in 24 hours from 00:00, January 6, 2008 to 24:00, January 6, 2008.

Figure 9 illustrates the number of peers that own the movie. In here, owning a movie implies that the peer is still in the P2P-VoD system, and that the peer has *at least one chunk* of that movie (similar to the movie-level heath index

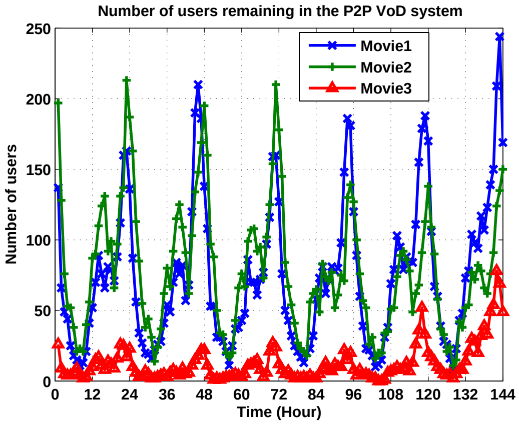


Figure 7: Number of viewing actions at each hourly sampling point (six days measurement).

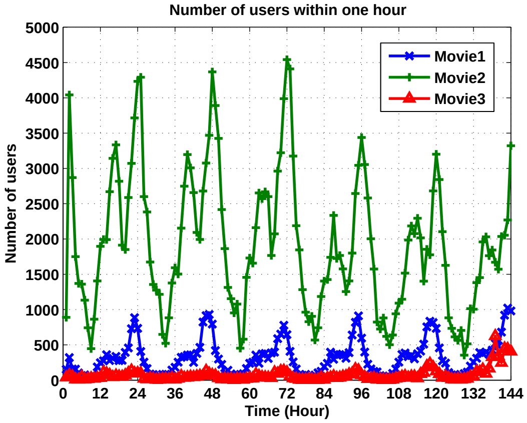


Figure 8: Total number of viewing actions within each sampling hour (six days measurement).

we discussed in Section 3). We can make two observations here: (a) Movie 2 being the most popular movie, the number of users owning the movie is higher than that of movie 1 and 3. (b) The number of users owning the movie is lowest during the time frame of 05:00 A.M. to 09:00 A.M. As we mentioned in the previous subsection, this corresponds to the period that the number of active peers is very low. To provide QoS-guarantee to users, one may need to allocate more upload capacity to the server.

The previous measure is a coarse indication of the replication health of a movie. One may choose a finer indicator to reflect the effectiveness of the replication. In here, we define $OR_i(t)$ to be the owning ratio of chunk i at time t, or

$$OR_i(t) \quad = \quad \frac{\text{Number of replicas of chunk } i \text{ at time } t}{\text{Number of movie owners at time } t}.$$

Again, a movie owner is a peer which possesses at least one chunk of that movie. If $OR_i(t)$ is low, it means low availability of chunk i in the system. We average $OR_i(t)$ over a 24-hour period and Figure 10 illustrates the average owning ratios for different chunks in a 24-hour period. From this figure, we observe that the health index for "early" chunks is very good. This is due to the fact that many peers may

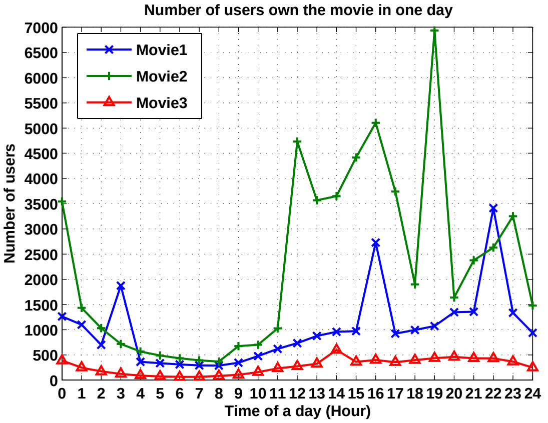


Figure 9: Number of users owning at least one chunk of the movie at different time points

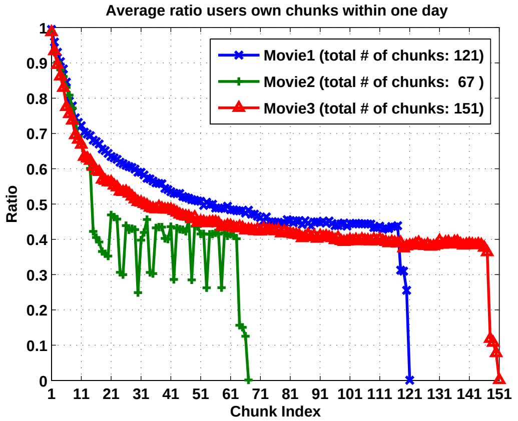


Figure 10: Average owning ratio for all chunks in the three movies

browse through the beginning of a movie. The health index of other chunks, is still acceptable since at least 30% of the peers have those chunks.

To obtain a finer granularity of the health index, we compare the number of replication of chunk i versus the total number of demand for chunk i. The total number of demand of chunk i in one day is derived from the user-behavior data, MVRs, and they were collected at the same measurement time as the health index data (from 00:00, January 6, 2008 to 24:00, January 6, 2008). Based on the starting viewing time (ST), ending viewing time (ET) and the starting position (SP), one can determine which chunks a peer demanded during the viewing action. Figure 11 shows the chunk availability and chunk demand within a 24-hours measurement period. From the figure, we can conclude that: (a) The health index for these three movies are very good since the number of replicated chunk is much higher than the workload demand. (b) The large fluctuation of the chunk availability for Movie 2 is due to the high interactivity of users. (c) Users tend to skip the last chunk of the movie (which corresponds to the movie epilog). To improve the quality of viewing, one may want to provide an anchor point at that position.

Another way to evaluate the replication health of a movie is to determine the ratio between the number of available (or replicated) chunks to the number of chunks demanded as time evolves. Let $ATD_i(t)$ be the available to demand ratio for chunk i at time t, or

$$ATD_i(t) = \frac{\text{Number of replicated chunk } i \text{ at } t}{\text{Number of demand for chunk } i \text{ at } t},$$

then the available to demand ratio for movie m at time t is:

$$ATD_m(t) = \frac{\sum_{i=1}^{N} ATD_i(t)}{N},$$

where N is the total number of chunks for movie m. Figure 12 shows the available to demand ratios, $ATD_1(t)$, $ATD_2(t)$ and $ATD_3(t)$ in a 24-hours period. Figure 13 shows the temporal average (means) and standard deviations on the number of available (replicated) chunks of these three movies in a 24-hours period.

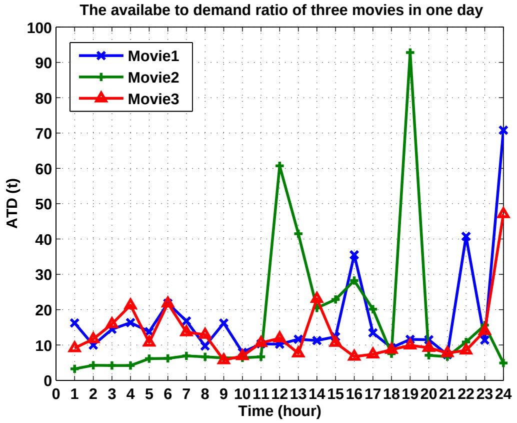


Figure 12: The ratio of the number of available chunks to the demanded chunks within one day

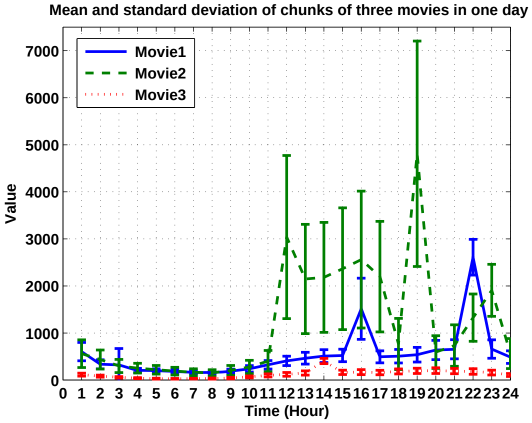


Figure 13: Temporal means and standard deviations of the number of replicas of all chunks of these three movies within one day

To provide good scalability and quality viewing, $ATD_i(t)$ has to be greater than 1. From Figure 12, we observe that $ATD_i(t) \geq 3$ for all time t. One can also observe from

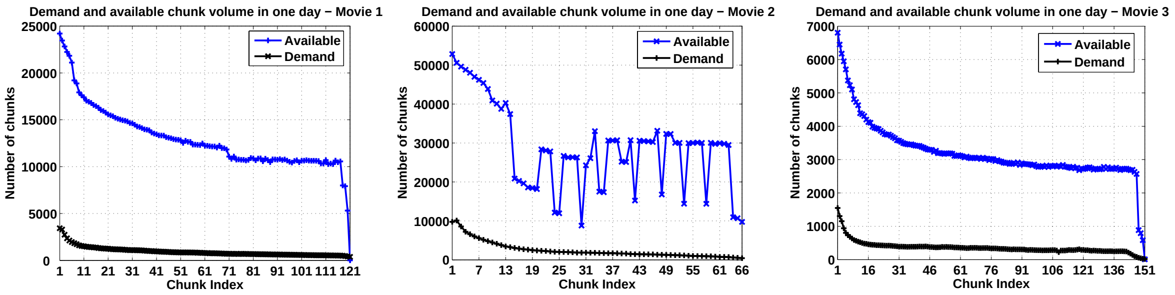


Figure 11: Comparison of number replicated chunks and chunk demand of three movies in one day

Figure 13 that these three movies have high temporal means and low standard deviations. This indicates that the overall replication health of these video object is very good. One interesting observation is that we have two peaks for Movie 2 at 12:00 or 19:00. At these two peaks, the $ATD_2(t)$ and the temporal mean of Movie 2 are very high. This is contributed by the fact that there is a large number of arrivals at these times (refer to Figure 9) and all these users want to watch Movie 2.

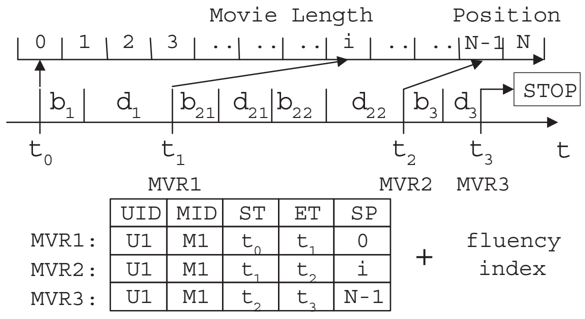
4.4 User satisfaction index

User satisfaction index is used to measure the quality of viewing as experienced by users. A low user satisfaction index implies that peers are unhappy and these peers may choose to leave the system. As a result, this will reduce the service capacity and scalability of a P2P-VoD system. From the view point of a streaming content provider, this is probably one of the most important measures that he/she needs to keep track.

To measure the user satisfaction index, we use the *fluency* $F(m,i)$ as defined in Equation (1). The computation of $F(m,i)$ was described in Section 3. Note that this computation is carried out by the client software in examining all the MVRs generated during the viewing period. Therefore, the fluency computation is carried out in a distributed manner. The client software reports all MVRs and the fluency $F(m,i)$ to the log server whenever a “stop-watching” event occurs, that is, either the STOP button is pressed, or another movie/programme is selected, or the user turns off the P2P-VoD software.

We give a simple example in Figure 14 to illustrate the generation of MVRs and fluency index. As shown in Figure 14, a user starts to watch the movie at time t_0 and he drags the track-bar to position i at time t_1. The first MVR is created (and stored in the peer's local disk) at this time. After then, the user drags the track-bar again and jumps to the $(N-1)^{th}$ position of the movie and the second MVR is created. Finally, the user presses the STOP button to stop viewing the movie at time t_3. At this point, the last MVR is created and because it is a stop-watching event, the client software computes the fluency index F. All three MVRs as well as the fluency index will then be transmitted to the log server.

To quantify the fluency index, we collected one day worth of trace for the three movies and the measurement duration is the same as the measurement period of the movie health index data, that is, from 00:00, January 6, 2008 to 24:00, January 6, 2008. To reduce the amount of data traffic, sampling is carried out once every hour.


Movie Length
Position
0 | 1 | 2 | 3 | .. | .. | .. | i | .. | .. | N-1 | N
b1 | d1 | b21 | d21 | b22 | d22 | b3 | d3
STOP
t0
t1
t2
t3
t
MVR1
MVR2
MVR3
+
fluency
index

	UID	MID	ST	ET	SP
MVR1:	U1	M1	t_0	t_1	0
MVR2:	U1	M1	t_1	t_2	i
MVR3:	U1	M1	t_2	t_3	N-1

Figure 14: Example of generating fluency index

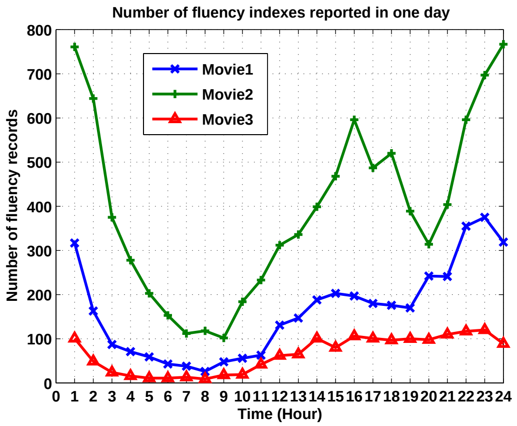


Figure 15: Number of fluency indexes reported by users to the log server

Figure 15 shows the *number* of fluency records that are reported to the log server at different times during the one day period. The number of fluency records is a good indicator of the number of viewers of that movie. We observe that the number of viewers drops after 01:00. Then it starts to increase after 09:00 and reaches the peak around 16:00. The number of viewers drop afterward and again picks up after 19:00 and reaches the second peak at time 24:00. In summary, this figure illustrates the number of viewers in the system at different time points.

Figure 16 illustrates the distribution of fluency index of these three movies within a 24-hour period. First, we divide the fluency interval $[0,1]$ into ten sub-intervals: $[0,0.1)$, $[0.1,0.2)$, ..., $[0.9,1.0)$. A fluency value greater than 0.8 is

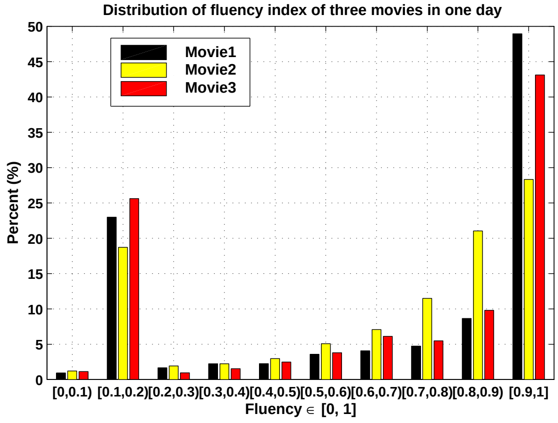


Figure 16: Distribution of fluency index of users within a 24-hour period

considered as having a good viewing quality while a fluency value less than 0.2 is considered as having a poor viewing quality. One can observe that there is a high percentage of fluency indexes whose values are greater than 0.7. However, around 20% of the fluency indexes are less than 0.2. One explanation of this phenomenon is that there is a high buffering time (which causes long start-up latency) for each viewing operation. This indicates an area for improvement in the current system.

One interesting question is how does the rate of change of viewer's population affects the fluency index. Figure 17 shows the change in percentage of good fluency index (i.e., fluency value between 0.8 and 1.0) and bad fluency index (i.e., fluency value between 0.0 and 0.2) when the number of viewers changes. When the rate of change of viewer's population takes on a negative (positive) value, it implies that the number of viewers is decreasing (increasing). One can observe that when the rate of change in viewer's population is of negative value, there is no effect on the percentage of good or bad fluency indexes. But when the rate of change in viewer's population goes from negative value to positive value (which corresponds to a sudden increase in viewer's population), then the percentage of good (bad) fluency index will decrease (increase). This is contributed by the fact that more users need to spend time to buffer the data before viewing the video.

4.5 Server Load and NAT related statistics

Figure 18 shows the load conditions of one content publishing server within a 48-hour measurement period. The server provides 100 movies, and it is a Dell PowerEdge 1430 server equipped with Intel DueCore 1.6GHz CPU, 4GB RAM and a Gigabit Ethernet Card. Figure 18 shows the upload rate, CPU utilization and memory usage in the measurement period. As shown in Figure 18, the patterns of upload rate and CPU usage vary with time and this pattern is similar with the pattern of number of users as shown in Figure 7. The server upload rate and CPU utilization are correlated with the number of users viewing the movies. Although we observe that there are some periods with high upload demand and high CPU utilization, comparing to the results reported in [17] (a client/server architecture), one can conclude that the P2P technology helps to reduce the server's load. The server has implemented the memory-pool technique which makes the usage of the memory more efficient and this could be observed at the bottom sub-figure of Figure 18 that the memory usage is very stable.

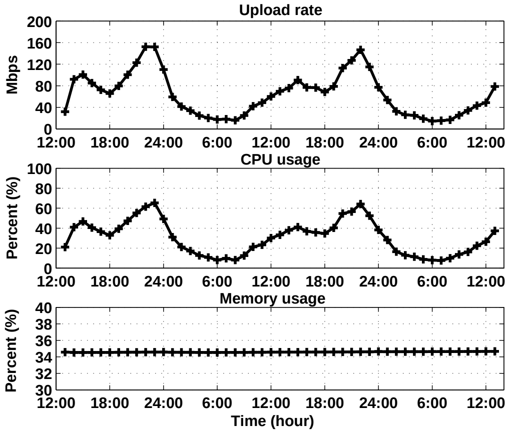


Figure 18: Server load within a 48-hour period.

Figure 19 shows the ratio of peers behind NAT boxes and Figure 20 shows the distribution of different NAT types within a 10-day measurement period from May 3 to May 12, 2008. From Figure 19, we observe that the ratio of peers behind NAT boxes remains stable, around 80%. Figure 20 implies that the *Full Cone* NAT has the largest proportion (47%) and the *Symmetric* NAT is the second (30%) and the *Port-restricted* NAT is the third (23%) while there is no *Restricted Cone* NAT.

Table 4 illustrates the distribution of peers' average upload and download rate measured on May 12, 2008. A total of 182544 peers are recorded. The average rate of a peer downloading from the server is 32Kbps and 352Kbps from the neighbor peers. On the other hand, the average upload rate of a peer is about 368Kbps. The average server loading during this one-day measurement period is about 8.3%.

Upload	# of Peers	Download	# of Peers
(Kbps)	(%)	(Kbps)	(%)
$[0, 200)$	65616(35.94%)	$[0, 360)$	46504(25.47%)
$[200, 360)$	51040(27.96%)	$[360, 600)$	118256(64.78%)
$[360, 600)$	45368(24.86%)	$[600, 1000)$	14632(8.01%)
$[600, 1000)$	9392(5.14%)	$[1000, 2000)$	3040(1.67%)
> 1000	11128(6.10%)	> 2000	112(0.07%)
Total	182544	Total	182544

Table 4: Distribution of average upload and download rate in one-day measurement period

5. RELATED WORK

Nowadays, P2P steaming technology attracts great research interests. A number of P2P live streaming systems are deployed with high viewing quality but low server burden, including CoolStreaming [21], PPLive [2], PPStream [6], UUSee [7], AnySee [19] and Joost [3] etc. Meanwhile, the theoretical analysis and measurement on P2P living streaming applications could help to improve the viewing quality and make the P2P live streaming system more robust and scalable. Hei et al. [18] have applied queueing theory and

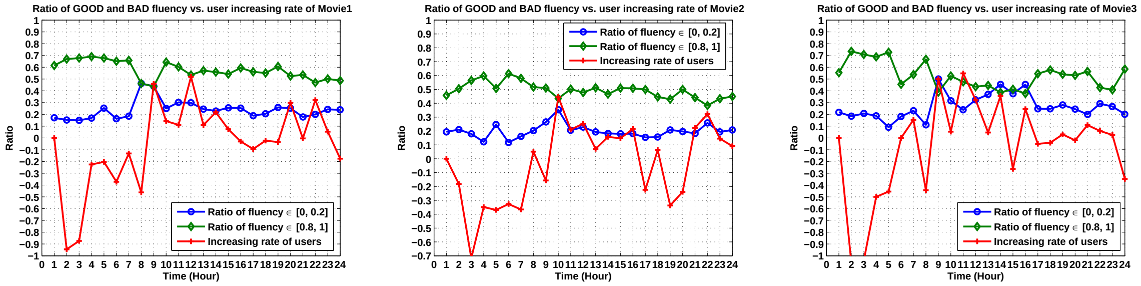


Figure 17: Comparison of the ratio of "good"and "bad" fluency vs. the rate of change in viewer's population within a 24-hour period

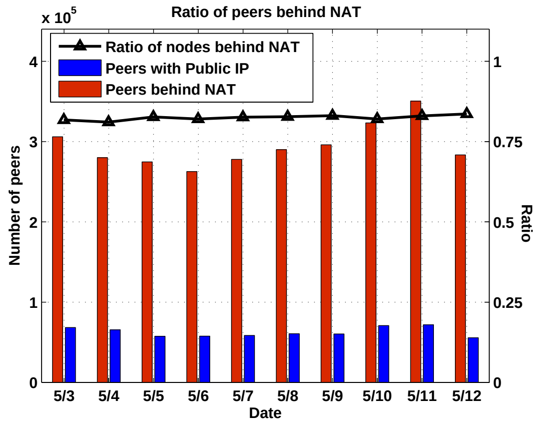


Figure 19: Ratio of peers behind NAT boxes within a 10-day period

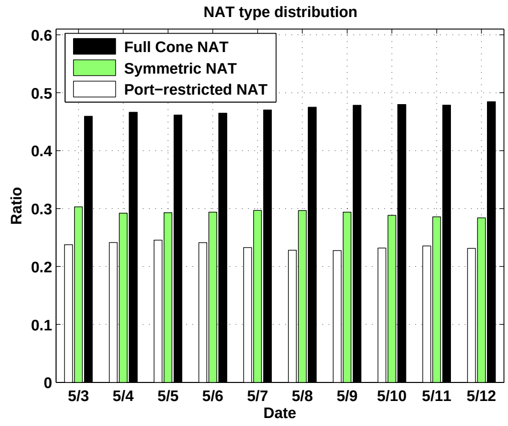


Figure 20: Distribution of peers with different NAT types within a 10-day period

stochastic fluid process to model the P2P streaming system. Zhou et al. [22] proposed a model to calculate the filling probability of streaming buffers based on a sliding window mechanism. They have used the peers' playing back continuity and start-up latency as the performance metrics. Two kinds of measuring methods, passive sniffing [8], and active crawling [15] have been applied to measure the existing P2P live streaming software from the client side. Wu et al. [20] have reported results of a measurement from a popular P2P live streaming system via the server side.

However, there are vital differences between live streaming and VoD streaming. For example, users' interactive behaviors like pausing and random jumping are allowed when they are subscribe to VoD services but live streaming systems do not provide these features. As a result, the design and deployment of a real world P2P-VoD system is more difficult than a P2P live streaming system.

In fact, on demand video streaming is not a new topic and the research begins since early 90's. IP multicast based proposals like patching [12], periodic broadcasting [16] and merging [11] faced the deployment problems of IP multicast. Later on, there are a number of proposals for peer-assisted Video-on-Demand streaming such as tree-based approach [14] and overlay tree based patching [13]. Recently Huang et al. [17] have carried out analysis through measurement and simulation on the data trace from an existing client-server based VoD system. To the best of our knowledge, our work is the first to conduct an in-depth study on practical design and measurement issues deployed by a real-world P2P-VoD system. In addition, we have measured and collected data from this real-world P2P-VoD system with totally 2.2 million independent users.

6. CONCLUSION AND REMARKS

P2P-VoD streaming service is an up and coming application for the Internet. As we prepare this paper, the P2P-VoD service in PPLive is already supporting up to over 150K simultaneous users, and we expect the number of users to grow further. In this paper, we present a general architecture and important building blocks of realizing a P2P-VoD system. One can use this general framework and taxonomy to further study various design choices. The building blocks we presented include the file segmentation strategy, replication strategy, content discovery and management, piece/chunk selection policy, transmission strategy and authentication. We also define various performance measures and illustrate how to instrument the measurement so as to evaluate the health of the systems and the user satisfaction (e.g., fluency index). We carried out a large scale measurement analysis to quantify the users' behavior, the effectiveness of the replication scheduling strategies, and the level of user satisfaction. In closing, this paper provides the general framework for further research in P2P-VoD systems, in particular, to address the following important issues: (1) how to design a highly scalable P2P-VoD system to support millions of simultaneous users; (2) how to perform dynamic movie replication, replacement, and scheduling so as reduce the workload at the content servers; (3) how to quantify various replication strategies so as to guarantee a high health index; (4) how to select proper chunk and piece transmission strategies so as to improve the viewing quality; (5) how to accurately measure and quantify the user satisfaction level.

7. REFERENCES

[1] "Emule", http://www.emule.com/.

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

[3] "Joost", http://www.joost.com/.

[4] "GridCast", http://www.gridcast.cn/.

[5] "PFSVOD", http://www.pplive.com/subject/20070808pfsvod/.

[6] "PPStream", http://www.ppstream.com/.

[7] "UUSee", http://www.uusee.com/.

[8] 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*, August 2006.

[9] B. Cohen. Incentives build robustness in bittorrent. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf, May 2003.

[10] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena. The pollution attack in p2p live video streaming: measurement results and defenses. In *Proceedings of Peer-to-Peer Streaming and IP-TV workshop (P2P-TV'07)*, Kyoto, Japan, August 2007.

[11] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth skimming: a technique for cost-effective video-on-demand. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, San Jose, CA, USA, January 2000.

[12] L. Gao, D. Towsley, and J. Kurose. Efficient schemes for broadcasting popular videos. In *Proceedings of the 8th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, July 1998.

[13] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2cast: peer-to-peer patching scheme for vod service. In *Proceedings of the 12th ACM International World Wide Web Conference (WWW)*, Budapest, Hungary, May 2003.

[14] A. A. Hamra, E. W. Biersack, and G. Urvoy-Keller. A pull-based approach for a vod service in p2p networks. In *IEEE HSNMC*, Toulouse, France, July 2004.

[15] X. Hei, C. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale P2P iptv system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.

[16] A. Hu. Video-on-demand broadcasting protocols: a comprehensive study. In *Proceedings of IEEE INFOCOM'01*, Anchorage, AK, USA, April 2001.

[17] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *Proceedings of ACM SIGCOMM'07*, Kyoto, Japan, August 2007.

[18] R. Kumar, Y. Liu, and K. W. Ross. Stochastic fluid theory for p2p streaming systems. In *Proceedings of IEEE INFOCOM'07*, May 2007.

[19] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng. Anysee: Peer-to-peer live streaming. In *Proceedings of IEEE INFOCOM'06*, April 2006.

[20] C. Wu, B. Li, and S. Zhao. Multi-channel live p2p streaming: refocusing on servers. In *Proceedings of IEEE INFOCOM'08*, April 2008.

[21] X. Zhang, J. Liu, B. Li, and T. S. P. Yum. Coolstreaming/donet: A data-driven overlay network for efficient live media streaming. In *Proceedings of IEEE INFOCOM'05*, March 2005.

[22] Y. Zhou, D. M. Chiu, and J. C. S. Lui. A simple model for analyzing p2p streaming protocols. In *Proceedings of IEEE ICNP'07*, October 2007.