國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

點對點的多媒體近似直播串流儲存策略

P2P caching strategies for near live media streaming

研 究 生：陳佩詩

指導教授：蔡文錦　教授

中 華 民 國 九 十 七 年 十一 月

點對點的多媒體近似直播串流儲存策略

P2P caching strategies for near live media streaming

研 究 生：陳佩詩　　　　Student：Pei-Shih Chen

指導教授：蔡文錦　　　　Advisor：Wen-Jiin Tsai

國 立 交 通 大 學

資訊學院資訊科技（IT）產業研發碩士班

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Industrial Technology R & D Master Program on
Computer Science and Engineering

September 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年 11 月

（ 點 對 點 的 多 媒 體 近 似 直 播 串 流 儲 存 策 略 ）

學生：陳佩詩　　　　　　　　　　指導教授：蔡文錦教授

國立交通大學

資訊學院產業研發碩士班

摘　　要

點對點架構(peer to peer)是在application層建立協定，讓網路上的peer能遵從一樣協定來進行溝通與傳輸資料。每個peer都可像主從式架構的客戶端去尋找資源，也可像伺服器提供資料，回應他人需求。此篇論文模擬在網路的應用層上，建立p2p的協定，利用像[3]的mesh-pull架構，做直播的多媒體串流分享，並提出peer之間合作式的cache儲存協定，讓使用者可以選擇收過去播過的多媒體串流，不只侷限在直播的串流。我們稱此為「近似直播串流(Near-live streaming)」應用，其結合了直播(live streaming)與隨選播放(VoD)功能於系統中。 一般直播多媒體系統多只儲存與直播節目最近期的資料，其cache資料更新採用FIFO的方法(或稱continuous Overwriting)，此種方法雖然簡單卻在所提的近似直播的應用中，普遍有較差的效能。所以我們提出 No overwriting 的 caching，與前述方法做互補。為了適合更多數的 video popularity的情況，提出Adaptive overwriting的方法。實驗結果顯示所提的caching 方法，特別是 Adaptive overwriting，在大多數的情況下昆能有較好的效能。

關鍵字：相互合作快取儲存、點對點傳輸、直播串流、隨選播放

P2P caching strategies for media streaming

Student: Pei-Shih Chen　　　　Advisor: Dr. Wen-Jiin Tsai

Industrial Technology R & D Master Program of
Computer Science College
National Chiao Tung University

ABSTRACT

The peer to peer communication is a protocol for user sharing computer resource and service over the application layer in the Internet. Every peer acts client and server, there is no central server. This paper extends the P2P protocol at live media streaming in [3] and proposes several cooperative caching strategies to provide "Near-live streaming service" which combines live streaming and VoD service in P2P systems. In these caching strategies, the simple and general method is continuous overwriting replacement, but it shows low hit rate in most of the cases for providing VoD services in the near-live streaming application. Therefore, we purpose a method called "No overwriting" caching, it mends drawbacks of the overwriting continuous method. But the "no overwriting" method still can't satisfy the variable demand for users on large scale p2p system. Since this, we purpose an "adaptive overwriting" method that can suitable most of video popularity distribution. The simulation results demonstrate the superiority of the proposed methods.

Keywords: Cooperating caching, Peer to Peer (P2P), Live streaming, VoD

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

With the advances in high-performance network and digital video technology, the large-scale Video-on-demand (Vod) systems have come into practice in recent years. P2P technology is introduced for media streaming system for its scalability and low commercial cost. P2P is a distributed architecture consisting of a collection of resources (e.g. computing power, data, meta-data, network bandwidth) performing a distributed function. And every peer acted as either a client or a server. A general distinction of P2P architectures into three kinds is possible:

- Centralized:

  There is a central directory. Each peer contributes its local “table of contents” to the central directory. Peer issues requests to the central directory to discover the most appropriate peer with the desired information. (E.g. Napster)

- Pure:

  All peers perform equal functions. Requests of a peer are flooded from the original requestor to their neighbors and so on, recursively, for a number of steps or until the request is satisfied. The flooded-request is manipulated through network connectivity and the appropriate number of steps can result in good request coverage.

- Hybrid:

  In pure P2P systems, it caused redundant messages for flooded-requests since the forwarding is performed recursively. Super peers are introduced for efficiency purposes in Kazaa P2P system.

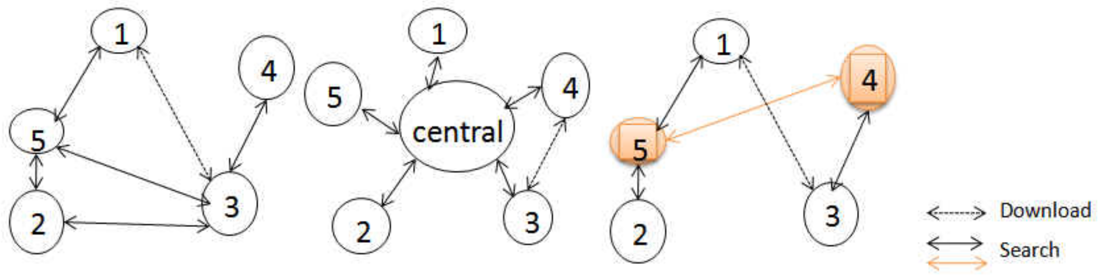


Figure 1 Centralized recovery, Flooded-request and super peer model

P2P and multicast are two common transmission manipulations to provide scalable and cost efficient solution for either Video-on-Demand (VoD) or live streaming. P2P technology has been widely adopted in live streaming system. The overlay network is an application level construction that was separated from the physical network. By overlay construction method, they can be roughly classified into three main categories:

- Tree based overlay:

  Peers are organized to form a tree-shape overlay network. By IP multicast support, data can transfer from a parent to multiple children as shown in figure 2 Enhance the efficiency of one-to-many and many to many communications over the internet. It concerns the way to build and maintain the tree. The parent peers in tree based overlay have a heavy burden in transferring data while leaf peers become isolation. On the other hand, the leave of parent peers caused tree adjustment which often affects the QoS (Quality of Service) of children peers.

- Mesh-based overlay:

  In mesh-based overlay, media data has been split into chunks and drive peers to get from multiple suppliers called partners. Meanwhile, each peer provides data to multiple “children”. The key issues in designing a mesh-based overlay include partnership management and data chunks scheduling algorithms. And a gossip based random algorithm is a solution to multicast message dissemination.

- Hybrid overlay:

  It divides the transmission of control messages and media data into different overlays in [1]. The mechanisms of peer join, peer leave and peer selection can be optimized in control tree. Through the control tree, peers can easily find their neighbors to construct a transfer mesh.

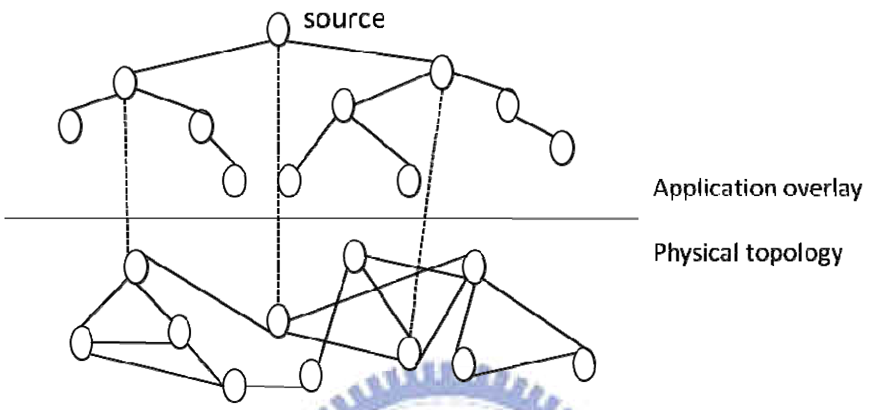


Figure 2 Network application overlay for live streaming

1.1. Live streaming

There have been significant studies on live video streaming over Internet recently; see the surveys in [2], [3]. System with mesh-based overlay like CoolStreaming [2] is popular for its adaptability to dynamic networks. We now describe some important protocols in mesh-based P2P streaming system:

Gossip-based protocol:

It employs a gossiping protocol in data-driven overlay network (DONet) for partnership management. In a typical gossip algorithm, a node sends a newly generated message to a set of randomly selected nodes; those nodes do similarly in the next round, and so do other nodes until the message is spread to all.

Partner refinement

In mesh-based overlay, a peer can receive media data that has been split into chunks from multiple peers called “partner”, and each peer can provides available chunks for multiple “partner”. Since any node can depart accidentally in DONet, it’s necessary to detect node failure for partnership refinement. It helps each node maintain a stable number of available partners by calculating a score for each partner. Let node j be the partner of node i. for node i, the score of partner node j is denoted by $Score_{i,j}$.

$$Score_{i,j}: \quad \mathrm{Max}\{\bar{S}_{i,j}, \bar{S}_{j,i}\}$$

Where $\bar{S}_{i,j}$ is the average number of segment that node i retrieved from node j. A partner that detects the failure will issue the departure message on behalf the failed node. And then the message is gossiped.

<u>Chunk distribution protocol</u>:

Every node periodically exchanges data availability information with a set of partners. The data availability information can be described as a buffer map.

<u>Peer's buffer map of video chunks</u>:

As figure 3 shows the buffer map message includes the offset that the ID of the first chunk, the length of the buffer map, and a string of zeros and ones indicating which chunks are available. The BM playable video indicates the number of continuous chunks in the buffer map, beginning from the offset

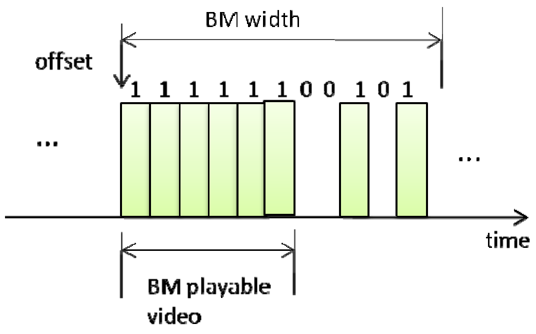


Figure 3 buffer map format

1.2. Video on Demand

Several approaches have been explored in the past to tackle scalability issues faced by VoD service. IP multicast has been proposed to enhance the efficiency of one-to-many and many-to-many communication over the Internet. A serial of IP multicast-based schemes, such as batching policy [4], has been developed that can drastically decrease the aggregate bandwidth requirement at the server. And both of [5] and [6] that based on P2P architecture proposed batching and patching scheme respectively. For instance, the P2P patching scheme in [6] can be partitioned into two functions for providing VoD service as following:

- Multicast for base stream forwarding:

  As figure 4 illustrates, the clients that arrive within the threshold of time interval constitute a session. And the clients are able to forward the received base stream to other clients so that the server and clients can form an application-level multicast tree.

- Unicast for patching serving:

  Every client need to have sufficient storage to cache the initial part of the video. So that the clients that earlier arrived can serve the patch to the later arriving clients.

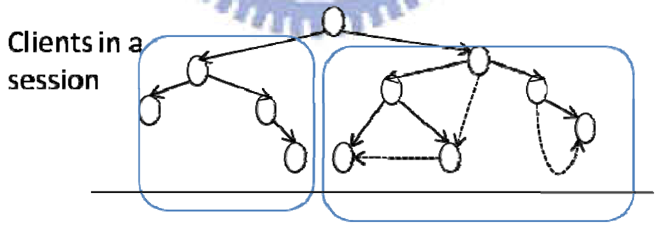


Figure 4 Client in a session

1.3. Motivation

We have an idea to approach a live streaming coupled with VoD system. Instead of providing services from pre-stored videos on video servers as conditional VoD systems, we proposed a P2P system which services VoD from cached live streaming. Peers receiving live streaming will retain the segment consist of chunks that played until they leave. The later peers can choose either the live streaming or previous programs (video). If the peers choose previous programs, they will retrieve data from other peers. We can exploit the data-driven overlay network to provide both of live streaming and VoD service without IP multicast support. Take the age of a segment retained in the storage into considerations; there are three proposed caching strategies.

- Continuous Overwriting
- No Overwriting
- Adaptive Overwriting

Chapter 2 Related Works

Most previous studies of cache management have been focused on the cache replacement based on video caching proxies.

2.1. Proxy caching scheme

In general, proxy cache on the internet places nearby client along the path from the server to the client, as shown in figure 5 proxy stores a portion of a stream data or an entire stream in a cache. The content is pushed from the server to proxies or CDN servers close the clients. Client can choose the server that incurs the least amount of congestion. Upon receiving the client request for one of the cached streams, the proxy transmits data directly from the cache. By caching the initial fraction of stream data, service startup latency can be reduced.

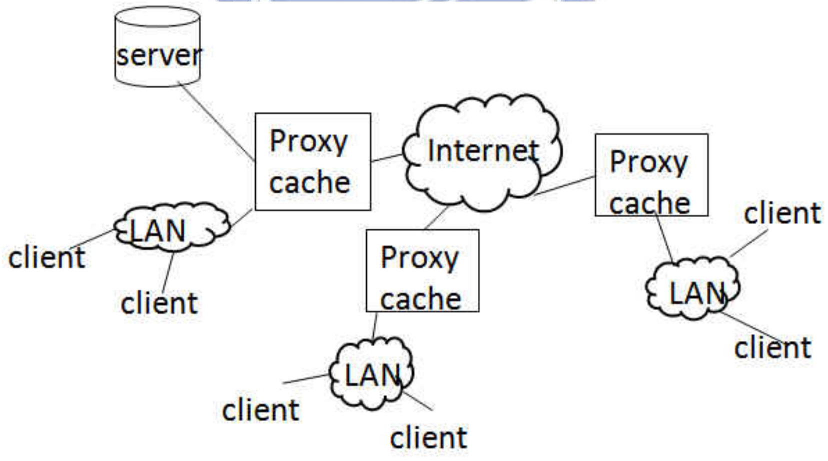


Figure 5 Proxy cache server

2.2. Replacement strategies

In table 1, we identify the elementary properties of cached objects that are taken into account in the removal decisions of caching algorithms. How the properties are used in replacement strategies are explained here in detail. Object will typically not fit into the allocated space completely, but only the initial of most recently used part is store.

Age [9]	Object have an order of being loaded into the cache.(FIFO)
Number of requests [7]	Provide a means of evaluating an object's popularity. It is not tied with any concept of time in its basic form.
Aging	It's used to give younger requests a higher relevance for removal decision than older requests.
Size [8] [9]	Cache of a large number of small objects, which increase the average response time of all user requests
Intervals [8] [9]	Measured the number of requests are collected for the time of interval and than simply discard.
Time since last request [7]	Identify the object that has been accessed most recently or least recently.
Data replicated	Due to the variations of bandwidth of links among caches, the amount of data replicated is taken into account.
Priorities	Priority can be used to move objects through a distribution system quickly if popularity or relevance can be predicted.

Table 1 Cache replacement strategies

The replacement policy of a proxy caching in [7] considered the number of requests and the last request time. A proxy has a caching value given by RF/ (T - T'), where T is the current time, T' is the timestamp of the last request to this object, and the RF is the number of requests for this object since the time it enters the cache.

[8] has proposed the replacement policy which considered the size and interval. Each proxy server calculates caching utility value that represents the correlation between popularity of a stream and the size of cache space of a stream allocated. And then the victim stream which has minimum value is selected to be replaced. The caching utility in [8] is given by:

$$CachingUtility = \frac{CachingBenefit}{Cost}$$

The cost is the size of storage space allocated for the stream. And the caching benefit is the total amount of data played back by clients. Figure 6 (a) shows the way to measure the total amount of data of a particular stream played back by clients during an interval Δ.

The replacement in [9] considered the size, interval and age. Assuming that m numbers of peers have played back video *i* in the *k*th period, $DATA_{i,j}^{k}$ is the amount of data of video *i* played back by client *j* in *k*th period, which is the caching utility in [9]:

$$DATA_{i}^{k} = \sum_{j=1}^{m} DATA_{i,j}^{k}$$

And then the proxy maintains the weighted mean played data, D_i^k for all videos, which is given by:

$$D_i^k = \alpha DATA_i^k + (1-\alpha)D_i^{k-1}, 0 < \alpha \le 1$$

Where α is a weighted value for recent information. This α should be set to a large value for objects that age quickly such as NoD (News on Demand) data. And α is set to a small value for objects that age relatively slowly.

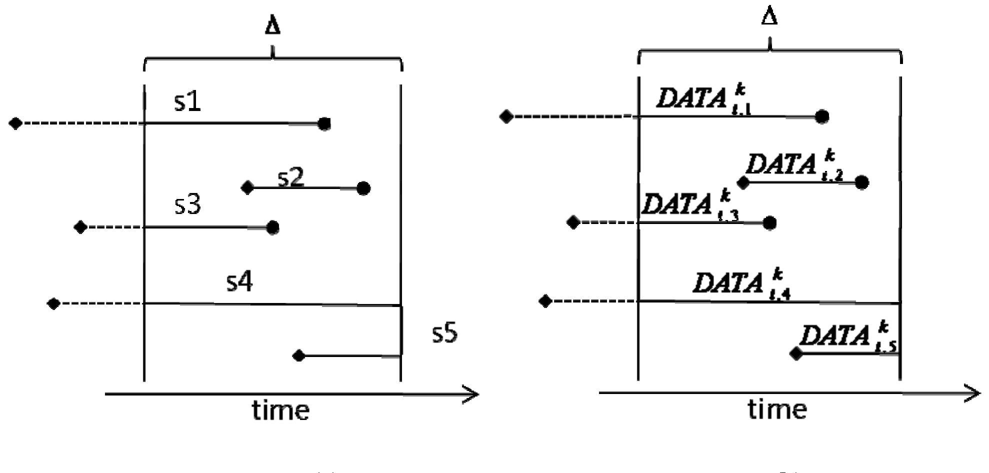


Figure 6 playback data for each video in a time slot

Chapter 3 Proposed near-live P2P streaming system

Due to the large size and timing constrains of multimedia streams, every peer usually cache part of streams as segments consisted of chunks. Take the age of a segment retained in the storage into consideration; there are three proposed caching strategies, "Overwriting continuous", "No overwriting", and "Adaptive overwriting". The architecture is a mesh-based network overlay. Our system approached a service which the peers can receive the live streaming or previous video. There have been researches in the VoD service; the general solutions for VoD assumed all peers receive the streaming from initial part of whole video, and the media streaming is pre-stored in each peer's buffer in P2P system. Due to the variable network environment, the peers share their available storage space among themselves, and a peer can exchange video data dynamically with a large number of peers.

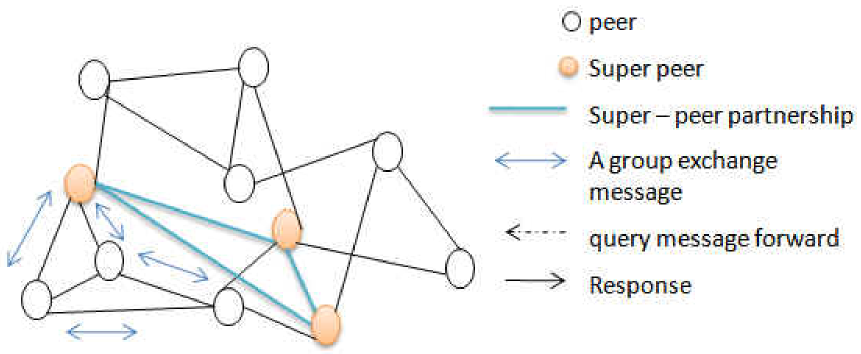
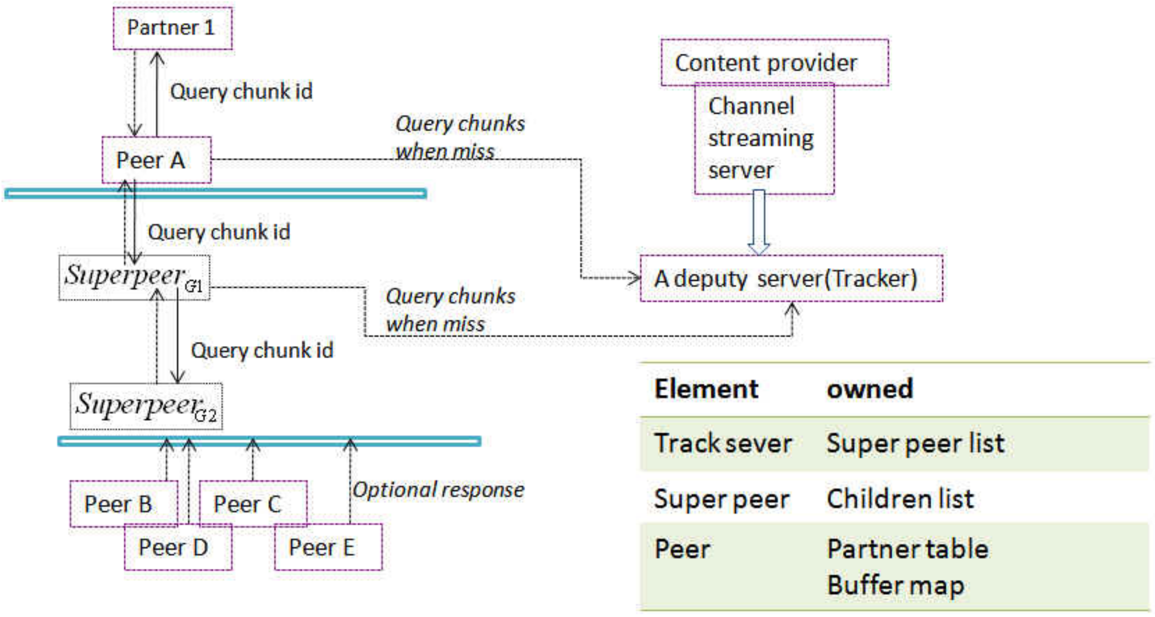


Figure 7 Co-operative caching

3.1. System architecture

The peer discovery is employ hybrid model. The distributed caching approach is co-operative caching as figure 7. Figure 7 illustrates the mesh-based network overlay, we employ p2p hybrid model which includes a content provider, a track server, and super peers.

- Content provider: The server provides media content.
- Track server: A server for super peer registration and super peer list maintenance.



Element	owned
Track sever	Super peer list
Super peer	Children list
Peer	Partner table Buffer map

Figure 8 Architecture

Here we define some terminologies that will be used throughout this thesis:

- Buffer size: The capability of a peer's storage can be shared with other peers.
- Urgent: We denote a portion of a peer's buffer as the urgent part, and the remaining portion as the cache as illustrated in figure 9. The urgent part must be filled with fix amount of chunks to enable the playback playable.
- Segment: The track server converts the media content into small video chunks for efficient transmission among peers. We have specified every minute as a chunk and separated every 30 minutes as a segment. A segment is the unit of caching.
- Group: The peers that requested the first chunk within the batch time interval form a group as depicted in figure 10. A peer having a larger bandwidth can be picked up as a super peer for the group. And every group has one super peer.

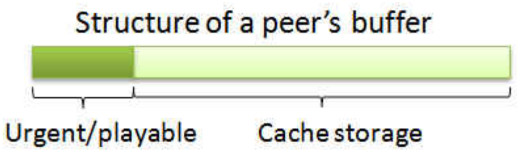


Figure 9 Peer's buffer structure

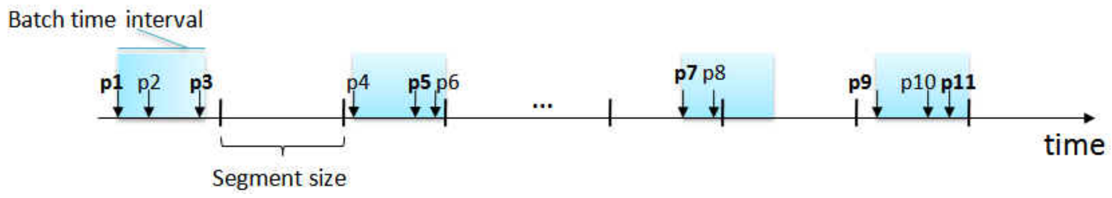


Figure 10 Peers form a group

Two media steaming procedures are constituted:

- Live streaming: The peers exchanged buffer map with partners and received media chunks from partners.
- Cache streaming: The peers that request previous videos can discover the peers that have available chunks through their super peers queried other super peers.

A <u>super peer</u> has several responsibilities that are described in the following:

A. Made contact with new peers: Since the track server has a super peer list that recorded the IP address of super peers in this system. A new peer can download super peer list from track server, and then issue join message to every super peer.

B. Maintain peers of the group: A super peer has a children list that recorded the addresses of the peers in the same group. Every peer in the same group periodically exchanges the message "alive" through the children list. They can pick up a new super peer quickly when their super peer departed.

C. Message forwarding: Requests are issued from the original requestor to its super peer, and then the super peer forwards the request to other near super peers. A super peer forwards the message when receiving the request message.

    - Get peer list: The peers can download peer list from their super peers to find appropriate partner for live streaming.

- Caching directory: The message forwarding service in super peer can be used to provide a role of content director. As figure 11 illustrated.

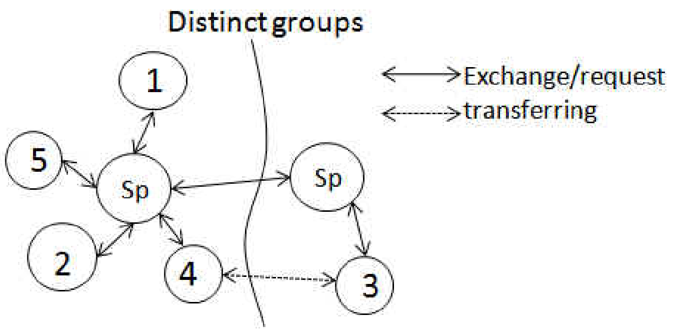


Figure 11 Caching directory

3.2. Proposed protocol

Under our system there are three phases for any peer: *Join, Work, and Leave*. And there are several operations during the work phase.

3.2.1 Join phase

In the join phase, a peer *v* arrived and set connection with track server, and then track server provided super peer list consisting of the IP address of every super peer registered. The new peer issued to every super peer the *join* message including the offset chunk id, zero one sequence, and address of itself. And then Super peer will reply message if the offset chunk id is within the batch time interval. According to the response of super peers, there are three situations:

a If there is no super peer response, this new peer will be upgraded to a super peer.

b If the bandwidth of the suitable super peer is smaller than itself, it will be upgraded to a super peer and take over the jobs of the original super peer.

c Join a suitable group.

After either a or b situation, the new peer registered itself in track server by sending a report. And tracker server broadcast the address of new super peer for others. And then each super peer can also maintain the super peer list.

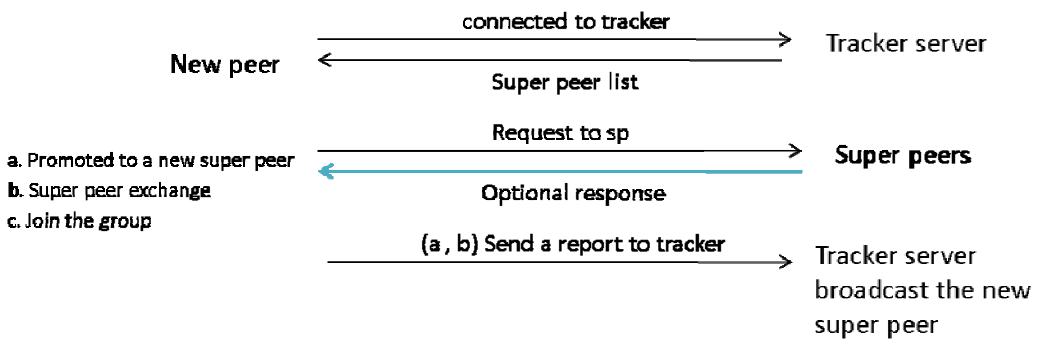


Figure 12 New peer processes

3.2.2 Work phase

Group membership:

The peers in the same group periodically exchanged "*Alive*" message including live time, address, and buffer map. Especially there are additional elements, children list, batch time interval, and "*Alive*" message of a super peer.

Super peer partnership:

Since the track server broadcasted the addresses of new super peers, every super peer can choose those super peers which have a short distance between themselves as their partners.

Partnership:

A peer can find partners through a super peer service. Due to the variable network situation, a peer calculates a score for each partner to maintain a stable number of partners.

Partner table	
Partner a	Score 1
Partner b	Score 2
Partner c	Score 3

$$\text{Score:}\ \ \log_2 \sum_{t=t'}^{T} C_t / (T - T')$$

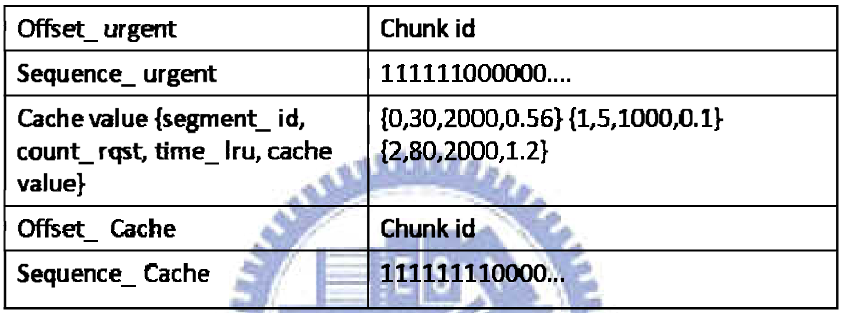
C_t	The amount of chunks received from a partner during a time unit t.

T	Now time.
t'	The latest time of calculating score.

Streaming:

Every peer maintains a buffer map as figure 13 (a) illustrated to specify clearly which available chunks it has.

A peer issued a query-chunk message that contains *Offset_urgent* , *Sequence_urgent*, *Cache value*, *Offset_Cache* and *Sequence_Cache* to either its super peer or partner. The *Cache_value*, there are four columns as explained in figure 13 (b).



Offset_ urgent	Chunk id
Sequence_ urgent	111111000000....
Cache value {segment_ id, count_ rqst, time_ lru, cache value}	{0,30,2000,0.56} {1,5,1000,0.1} {2,80,2000,1.2}
Offset_ Cache	Chunk id
Sequence_ Cache	111111110000...

(a) Query-chunk message

segment_id	The media streaming is separated as many continuously segments by the track server, and every segment has a unique id.
count_rqst	The segment at cache was requested in the last unit time that is defined a fix sliding time window.
time_lru	The segment at cache was requested at the last time.
cache value	The utilization of the segment.

(b) Cache value

Figure 13 Buffer map format of proposed protocol

The Offset_Cache is the first chunk id in the cache storage. And Sequence_Cache is a zero one string that specified existence of following chunks.

3.2.3 Leave phase

The peer can leave the system at any time without a notice. A peer that played a segment has high probability to leave the system.

3.3. Proposed Cache Replacement strategies

3.3.1 Continuous Overwriting

The "Continuous Overwriting" method exploits FIFO replacement while the storage of a peer is full. The most recently chunk that peer has already played is instead of the oldest chunk at the buffer.

In figure 14, for the initial state, assume the first user P1 arrived at t0, and requested data from the 1000th minute of the stream. During the stage there is no peer receiving the same stream with P1 therefore, P1 downloaded data from the content server. For the normal state, peer can receive media chunks from other peers. When P2 arrived at t2, 130 minutes later than t0, and requested data from the 1100th minute of the stream, P2 can receive from P1 whose live time of playback is at the 1130th minute and its storage capability is 50 minutes. Now P3 and P4 are arriving at t3, and request from the 1165th minute and the 1005th minute respectively. P3 can download the media chunks from P2, and P2 can simultaneously receive chunks from P1. But the media chunks that needed for P4 will be missing.

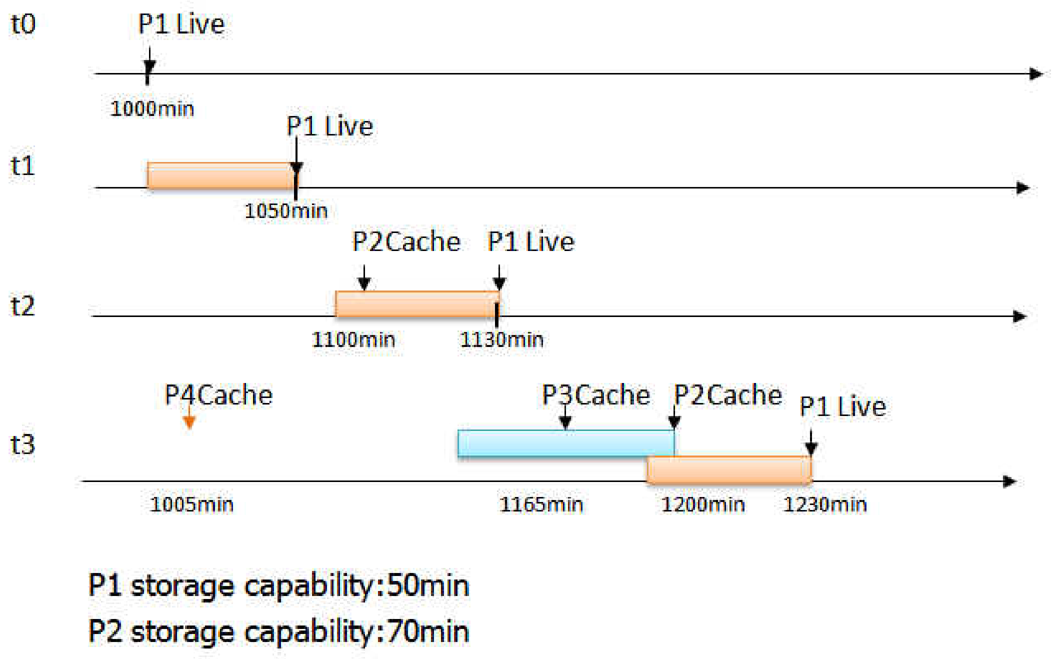


Figure 14 Overwriting continuous

3.3.2 No overwriting

In this section, the concept of caching strategy is opposite to Continuous overwriting. Consider all the peers on the system. The peers retain every segment consisting of chunks that has been played until their buffers are full. Then keep the segments without overwriting all the time. For an instance in the figure 15, the P1 and P2, arrived at t0 and t2, respectively, retained the initial segment they played. So, before P1 and P2 leave the system, P1 always keeps its cache with stream from the 1000th minute to the 1050th minute and P2 from the 1100th minute to the 1170th minute. Both of P3 and P4 arrived at t3 can retrieve segments of the 1005th minute and the 1165th minute from P2 and P1, respectively.

As figure 16 shows the peers that requested the first chunk within the time interval constitute a batch. A super peer can confirm the first chunk id of peers in the group quickly to enhance efficiency for the query message of a peer.

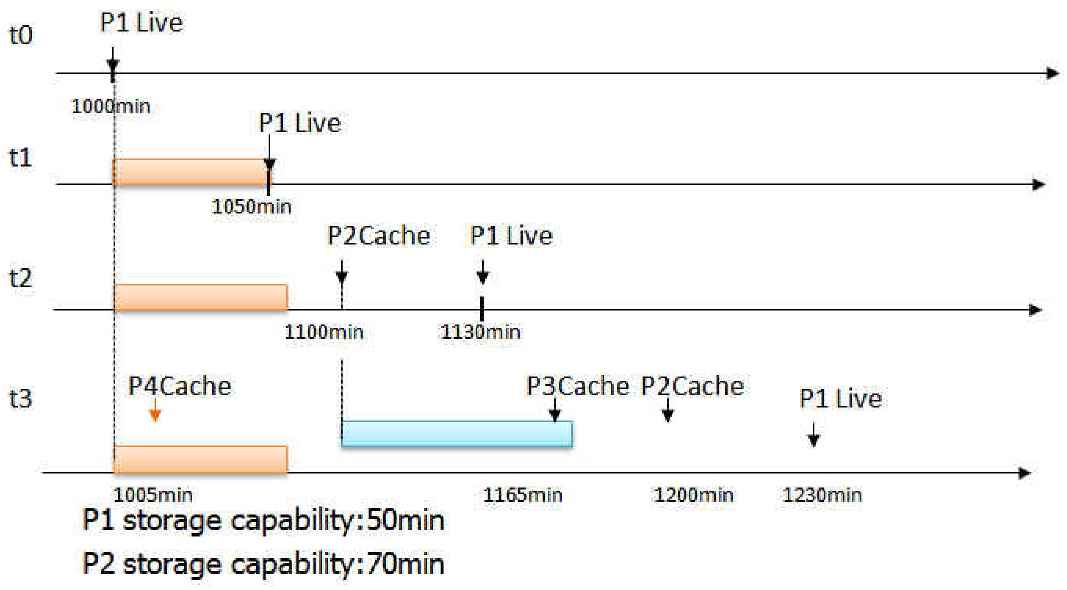


Figure 15 No overwriting

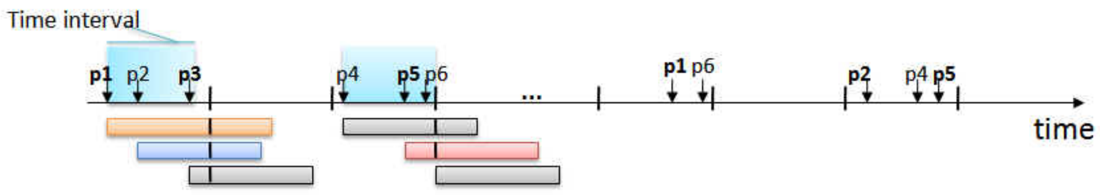


Figure 16 Group as a batch within the time interval

3.3.3 Adaptive Overwriting

The continuous overwriting method may improve hit rate which retrieved chunks from the caches of all peers, while all of the peers request the same video concurrently. The no overwriting method may improve hit rate, when the previous segments on the cache still are popular. Nevertheless, the no overwriting method will waste the space of a cache that is allocated for the unpopular segments.

Taking the popularity of a video-on-demand service with a great variety of content type into considerations, the adaptive overwriting method is proposed to mend the drawback of the no overwriting method. Two replacement policies are used to update the unpopular segment in cache.

As figure 17 shows, assume the segment in the cache of P1 has not been requested for long time. The storage space for the segment is expected to record a new segment at t_n . For another instance of figure 17, all the segments in the cache of both p2 and p5 have no access for long time that we called T_{long}, the caches were cleaning to record at t_{n+1}. Here we won't specify the T_{long} for a replacement decision. We used several replacement strategies that mentioned at 2.2 to apply a replacement decision. There were two aspects of the replacement policy that were described as following:

- Cache value: Each peer has recorded the access frequency of every segment in its cache, and calculated the cache value for each segment every minute. It's capable of capturing the changing popularities of the segment by attaching a caching value according to how recently and how frequently the segment was requested.

$$C\_Value_i = \frac{rqst_i^{ts}}{T - T_i^{lru}}$$

Where the $rqst_i^{ts}$ is the number of request in the last unit time that is defined as a fixed sliding time window. T_i^{lru} is the most recently request time, T is now time. And then we have taken the accessibility of a popular segment into consideration:

- PR_ratio: The ratio of the popularity to the replica of a segment. That is,

$$PR\_ratio = \frac{numberOfrequest(segment_i)}{replica(segment_i)}$$

We made sure the popularity of the replaced segment is lower than the target of a replacement segment. Set PR_ratio_P as the ratio of the popularity to the replica of the segment that client is playing. For every segment *i* of cache, *i* from 1 to m, the following conditions (1) and (2) are checked. If both conditions are satisfied, the storage space of the segment *i* is cleaned.

$$PR\_ratio_P \;>=\; PR\_ratio_i \qquad (1)$$

$$C\_Value_i < 0.01 \qquad (2)$$

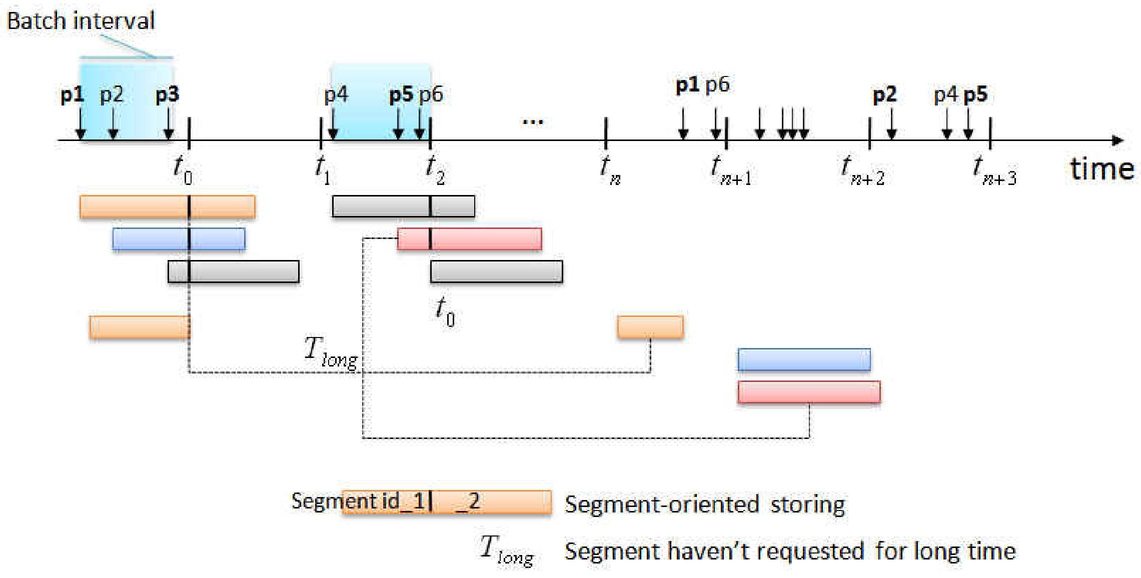


Figure 17 Adaptive overwriting

Chapter 4 Experimental Results

In this chapter, we evaluate the hit rate of each method that mentioned in chapter 3 through simulation experiments. We compare three caching strategies: Continuous overwriting, No overwriting, and Adaptive overwriting. In our simulation results, method A can outperform both method 1 and method 2 in most case.

4.1 Study of content popularity evolution

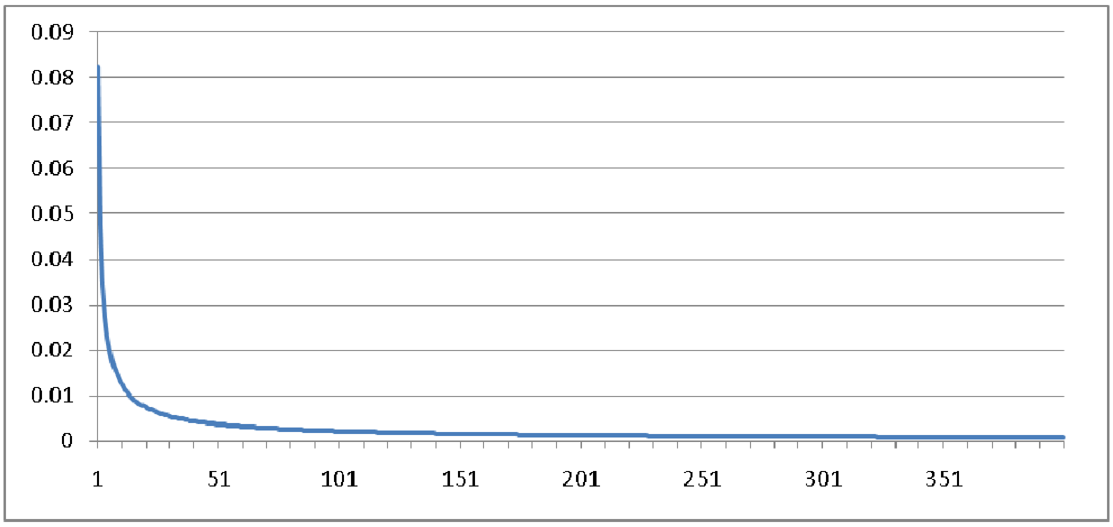
In order to evaluate the effectiveness of our caching strategies, we need to take the popularity of the videos into account. In our simulation model, the hit rate that each method performed is related to specific videos. Especially when VoD is available, which video is the choice of a client depended on video's popularity. The Zipf-distribution is the model used for request probability and [10] is used for modeling video life cycle.

4.1.1 Zipf distribution

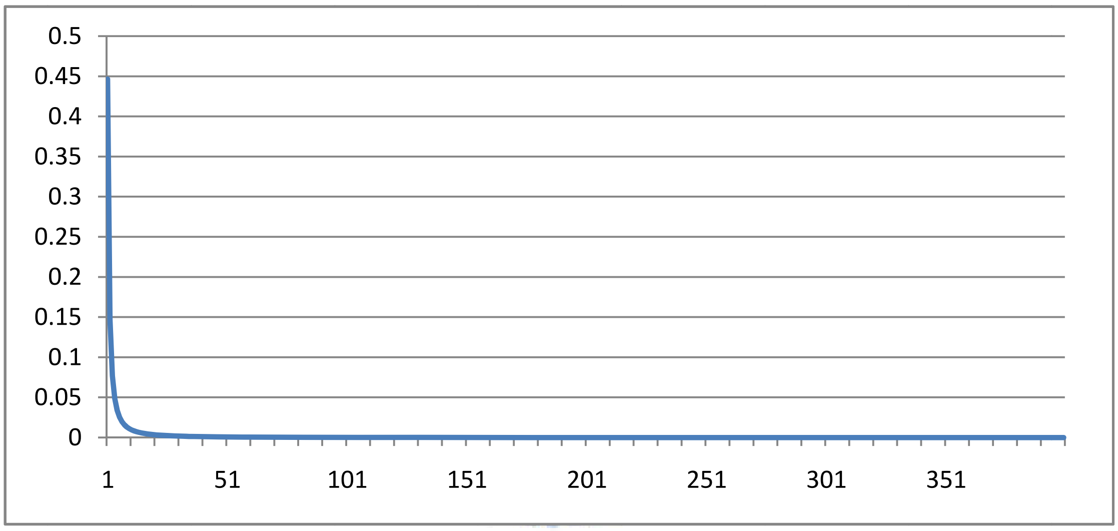
We assume N is the amount of videos at this time, k is the rank of the distribution, and α is the value of the exponent characterizing the distribution.

$$f(k;\alpha,N)=\frac{1/k^{\alpha}}{\sum_{n=1}^{N}(1/n^{\alpha})}$$

The figure 18 has shown a comparison for two different α . Figure 18 (a) and (b) have shown the Zipf PMF for α = 0.8 and α =1.6 respectively. The horizontal axis is the index *k* and N=400. The vertical axis is the access probability. The function is only defined at integer values of *k*.



(a) 0.8



(b) 0.6

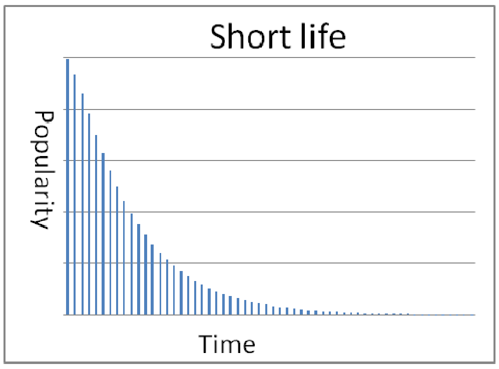
Figure 18 (a, b) Comparison of different Zipf parameters

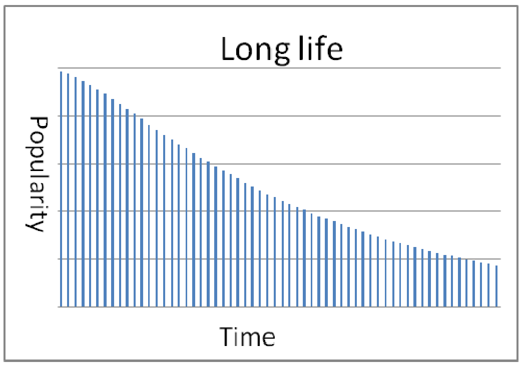
4.1.2 Video life cycle

In our experiments, three distinct video life cycles have been taken into account: short-term, long-term, and up-and-down. Assuming the model has a temporal resolution of every 30 minutes, as figure 19 (a, b, c) illustrated, these three kinds of life cycle are described as following:

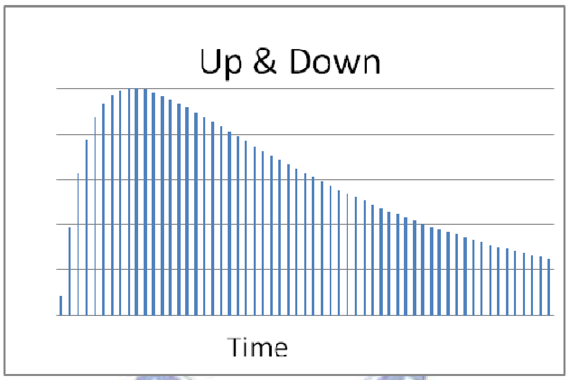
- Short life: This content reaches the highest popularity during the first 30 minutes, and the number of accesses per time unit decreases suddenly.

- Long life: This content reaches the highest popularity during the first 30 minutes, and the number of accesses for every time unit decreases slowly.
- Up and down: The popularity of contents goes up and down.





(a) (b)



(c)

Figure 19 (a, b, c) Content life-cycles

As mentioned in 4.1.1, each video has distinct life-cycles; we can map the index of Zipf-distribution to the distinct videos according to the content popularity evolution model. We used both of the population models in [7] and [10].

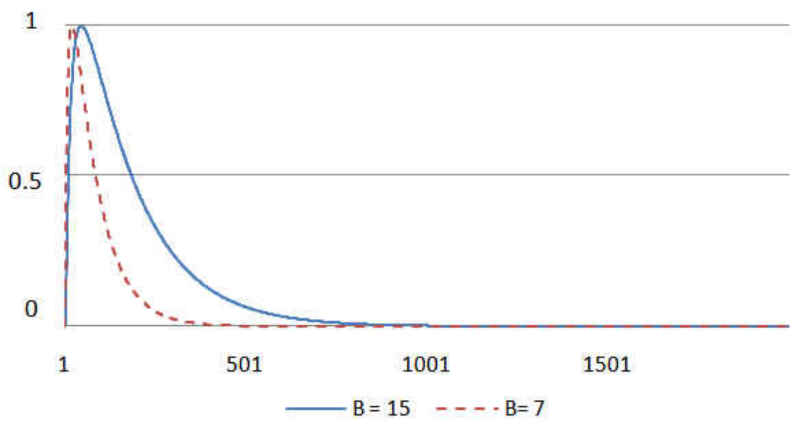
The population model in [7] is called Range n in our simulation. The parameter n can be adjusted to a smaller value for a low variability of content popularity environment, and a larger value for high variability.

The population model in [10] is expressed by RP (t):

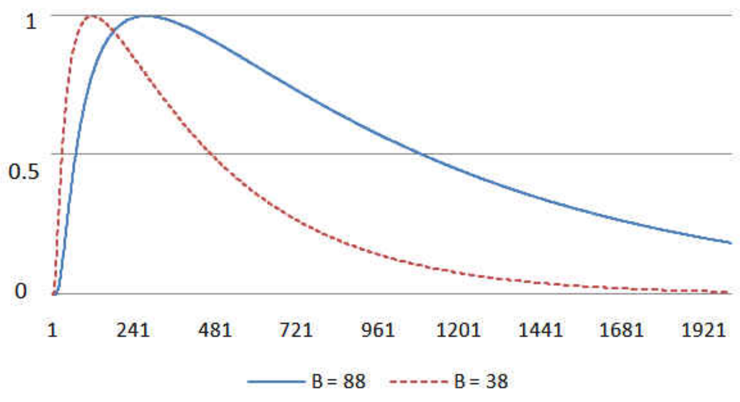
$$RP(t) = a \times e^{\left(\frac{2}{\sqrt{10}} - \frac{t}{10 \times B} - \frac{B}{t}\right)} + c$$

The coefficient of correlation, both of a and c are set equal to 0.01, and B is a variable value. B can be adjusted to a smaller value for modeling short-term life cycle, and a larger value for long-term life.

Since our model has a temporal resolution of every 30 minutes, the t of RP (t) is 30 minutes as a unit. As figure 20 illustrated, the horizontal axis is how long a video has been produced, and the vertical axis represents the popularity of this video, which is measured in terms of access probability.



(a)



(b)

Figure 20 (a, b) Long-term life-cycles

4.2 Simulation setting

In our simulations, we compared the three methods which are used in distinct popularity environments.

4.2.1 Topology

The network topology in our simulation environment is modeled by a formula called Power Law:

$$P(K) = CK^{-r}, \quad C = \sum_{n-1}^{N} 1/n^{r}$$

Where the K is how many outgoing links a peer has. N is the amount of peers or the topology size, r is the value of the exponent characterizing the distribution.

We set the r to 2; the topology size is set to 3000. K is set from 1 to 6 and K is an integer. Table 2 illustrated the relation between K and the distance. Since the peers who have the most outgoing degree are given precedence to decide the links between other peers and itself, we found out a peer who has larger outgoing degree can has shorter distance with other peers, but a peer who has smaller outgoing has longer distance with other peers.

K	Average distance with all peers
1	616640
2	181788
3	80169
4	40809
5	19846
6	10310

Table 2 Relation between outgoing link and distance

4.2.2 Parameters and complex popularity environment

We evaluated the performance of each method in distinct popularity environments that illustrated in table 4. And table 3 illustrated the parameters in our simulation. We assume that peer arrived according to Poisson process:

$$f(k,\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where the λ is the expected number of occurrences during one minute. The k is the number of occurrences of an arrival event. Both of the peer arrival rate and leave rate are λ .

Notation	value	Definition(default values)
$Time_{run}$	1440 hours or 60 days	The total running time of the simulation
N	3000	The topology size
r	2	The exponent of Power Law
K	From 1 to 6	The distinct outgoings
S_{sgmt}	30 minutes	A stream is composited of segments. S_{sgmt} is the segment size.
S_B	30 minutes	the peers arrived within the interval time form a group(30minutes)
S_{store}	50 minutes	The storage size a peer has
ts	30 minutes 60 minutes	A sliding time window is used in calculating the number of request.
λ	0.5/minute	The expected number of peers arrived during one minute
$SP_{collect}$	50% of all super peers	The amount of partners that a super peer has. A super peer chooses newly and nearly super peers.
α	0.8	The exponent of Zipf used for the population model.
B	7, 15, 38, 88	The B value of RP(t)

Table 3 System parameters and default values

Table 4 illustrated the complex popularity environment. Since the t of RP(t) was defined by 30 minutes, we also assume the popularity distribution changes every 30 minutes. There are four distinct B values used for RP (t), which are 7, 15, 38, and 88, and the distinct b value is generated by Zipf distribution that the α parameter is equal to 1. The short-term popularity is modeled with B=7 and the long-term one with B=88. “7-88” indicated that the amount of short-term popularity which is modeled with B=7 is the most, the amount of another one which is modeled with B=15 is secondary most and so on. “88-7” is contrary to “7-88”, the amount of popularity model which is modeled with B=38 is the secondary most.

Name	**Describe**
Range n[7]	Small n: Low variability of content popularity. Lager n: High variability of content popularity.
Half-RP7-88	The amount of short-term popularity model is much more than long-term. “Half” means that only half of the RP model is used. The popularity goes from high to low only
Half-RP 88-7	Similar to Half-RP7-88, except that the amount of long-term popularity model is much more than short-term.
Full-RP7-88	Full RP model is used so the popularity goes up and down. The amount of short-term popularity model is much more than long-term.
Full-RP88-7	The popularity models are up and down. The amount of long-term popularity model is much more than short-term.

Table 4 Population models

4.2.3 User behavior

A peer that arrived in the system can select one of all segments which have been produced by the content provider, and also can select the live streaming. The peer will leave the system or select another segment while it has played a segment.

4.3 Analysis of result

4.3.1 Sliding window size

As we have mentioned in section 3.5, each segment in buffer has a cache value to reveal its popularity. The problem is how the value of *ts* is decided. If the *ts* is adjusted too small, the segment that will be popular at future may be replaced. If the *ts* is adjusted too large, the replacement strategy will be insensitive. There are two goals that the cache value approached can be classified:

A. To keep the segment that will be popular at future for longer time.

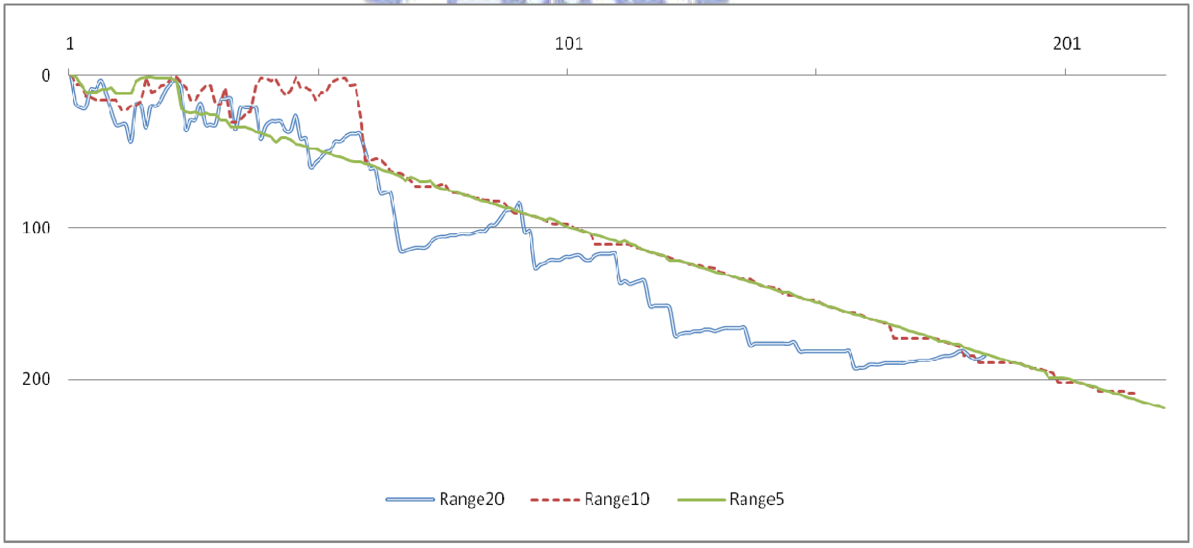
B. To detect the wasted storage sensitively to replace the unpopular segments.

For the goal A, we concern that the video life cycle is up and down. In order to predict the *ts*, we have ranked all segments according to their popularity over time, and then we evaluated the rank of a segment over time in figure 21. Assuming the rank is from 0 to 250, the rank 0 is most popular, and rank 250 is most unpopular. According to the figure 21 (a), assume that the top 10 ranks are popular enough, and it’s unpopular when the number of a rank is over 10. Now we discuss how to decide *ts* in order to keep top 10 ranks of videos in cache even though these videos may fall out of top 10 ranks for short time.

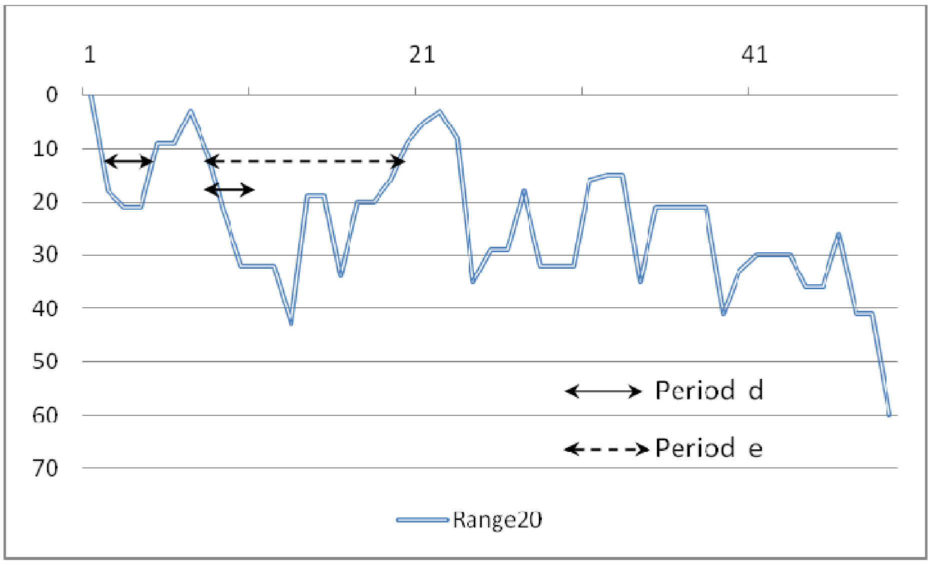
First of all, we discuss the Range n environment. The figure 21 (a) has shown that the variability of the population is higher when the value of n is 20. Taking the Range 20 situation into consideration, we scale up the figure 21 (a) curve that the Range 20 video stayed at top 10 ranks and show it in figure 21 (b). In figure 21 (b), it is observed that the popularity of a video falls below top 10 ranks in two periods. To keep this video in cache when it goes up to top 10 again, since the maximum period e is long enough to cover the shorter period d, the value of the *ts* can be predicted as the interval of period e.

Secondary, we discuss the complex video life-cycle environment where the popularity of video may go up and down. The population models are generated by RP(t) that parameter B are 7 and 38 as figure 21 (c) and (d), respectively. For instance, a peer has requested the segment for the duration a, we hope that the peer will retain the segment in its buffer until the duration c, so that it can be accessed when it is at top 10 ranks. In this case, the value of *ts* is supposed to be as the duration b at least.

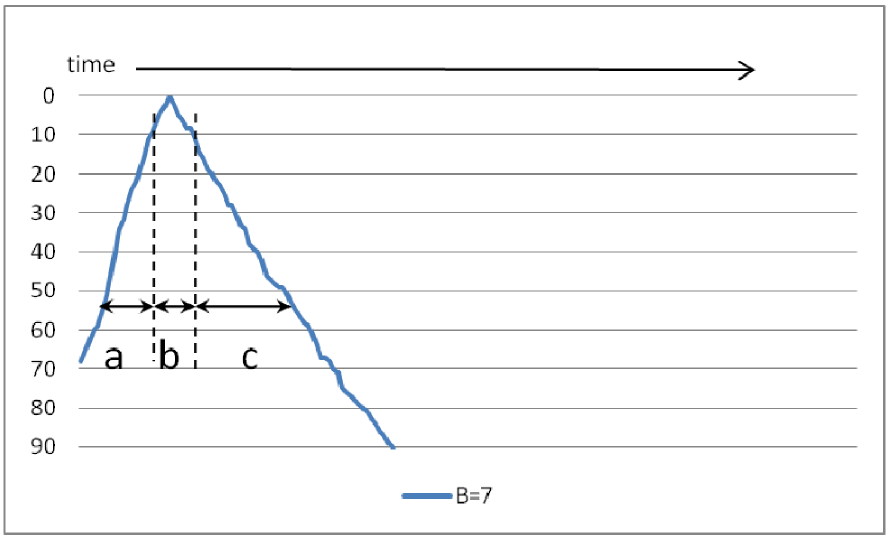
For the goal B, in order to improve the sensibility of our replacement strategy, we set the value of *ts* to be 30 minutes through our experiments except the value of ts is 60 minutes instead of 30 minutes in “Half 88-7” and “Pop 88-7” popularity environments which are having more long life segments.



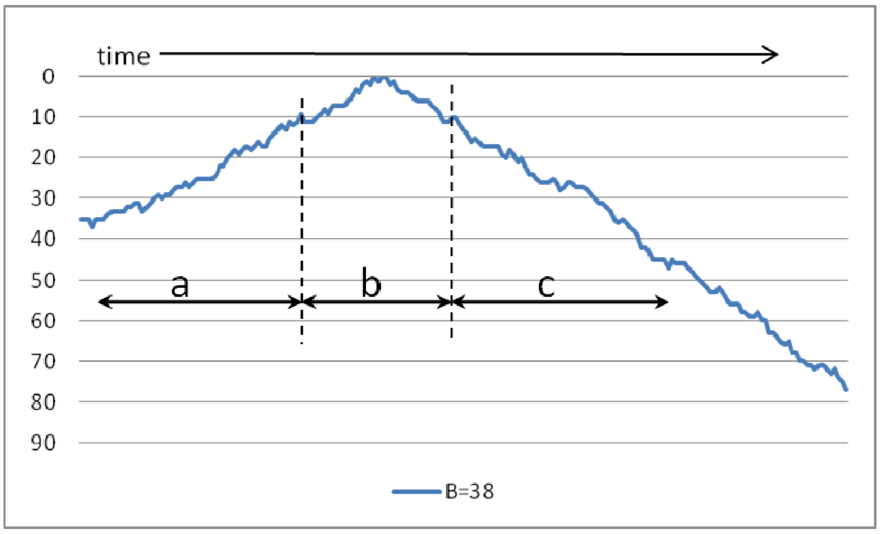
(a)


1
21
41
0
10
20
30
40
50
60
70
Period d
Period e
Range20

(b)


time
0
10
20
30
40
50
60
70
80
90
a
b
c
B=7

(c)


time
0
10
20
30
40
50
60
70
80
90
a
b
c
B=38

(d)

Figure 21(a, b, c, d) Popularity ranked

4.3.2 Experimented results

As the figure 23 shows the results of performance evaluation of the method 1, method 2, and method A. And the method 1, method 2, and method A indicate the "Overwriting continuous", "No overwriting", and "Adaptive overwriting" respectively. The figure 23 (a, b) are the results in all population models that have been defined in 4.1 and illustrated in table 4. The figure 23 (c, d, e) are the results in Range 50 population environment.

In figure 23 (a) and (b), we set the Zipf parameter α equal to 0.8 and 1.6, respectively, and evaluated all of three methods in all population environments that were mentioned in 4.2.2. By comparing the Zipf PMF for α = 0.8 and α =1.6 as figure 18 (a, b), we can describe the Zipf PMF for α = 0.8 has the lower difference in popularity between the segments. And the Zipf PMF for α = 1.6 has the higher difference in popularity between the segments. Due to the characteristic of Zipf PMF, we found out the method 2 can obtain higher hit-rate obviously than the hit-rate of method 1 while α is equal to 1.6.

Figure 23 (a) shows the method 2 outperforms the method 1 in most case. And the method A outperforms the method 1 and method 2 in all cases. However, in the Full-RP and Half-RP7-88 environments, the method 2 has lower hit-rate than method 1. The possible reasons are discussed below:

- Range n/Half-RP88-7: Method 2 can obtain a higher hit-rate in Range n and Half-RP88-7 because each segment was popular when it is newly produced from the content provider. For an instance in figure 22 (a), we assume the segment 1 is a long life video, and it's popular when it is newly produced from the content provider at t0. According to method 2, segment 1 will be stored on the peers arrived until the storage space is full, so other peers (including early arrived peers and lately arrived peers) can receive segment 1 from the caches of these peers. The access to segment 1 may last for a long time because it is a long life video. The peers that have the segment 1 can serve other peers for a long time as a popular duration of segment 1. Taking advantages of more long-life videos in Half-RP88-7 model, method 2 performs much better than method 1.

- Half-RP7-88: There are more short-life segments in Half-RP7-88. More peers still retained the segments that have become unpopular, because the method 2 is a No overwriting strategy. For Zipf with α =1.6, method 2 still can takes advantages of skewed access to achieve a better result than method 1. However, for Zipf with α =0.8, method 2 lost its benefits and resulted in a worse performance.
- Full-RP: We have mentioned the videos in Full-RP environment are up-and-down life cycle. For an instance of Full-RP in figure 22 (b), there is a serial requests for segment 1. The segment 1 is becoming popular at t2, but there is no peer has already stored the segment 1, because it had low request frequency before t2.

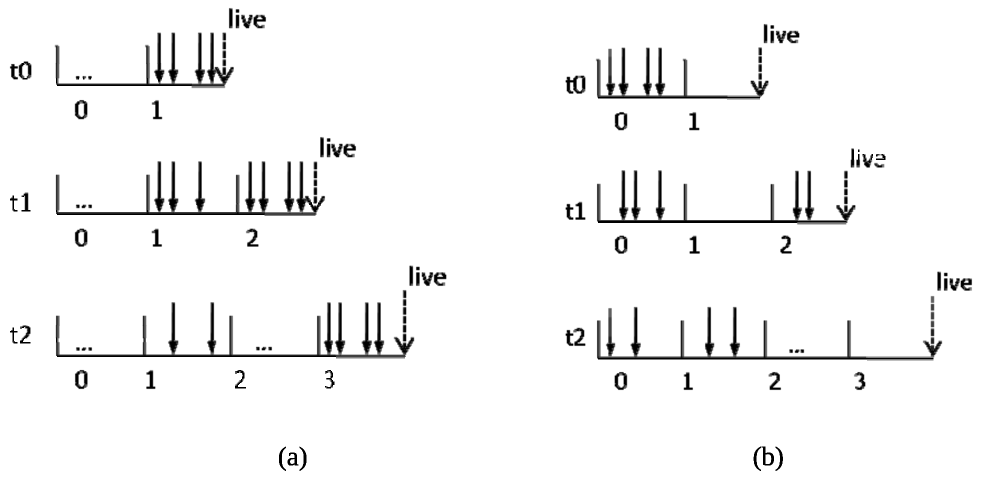


Figure 22 (a, b) Instance for a serial requests

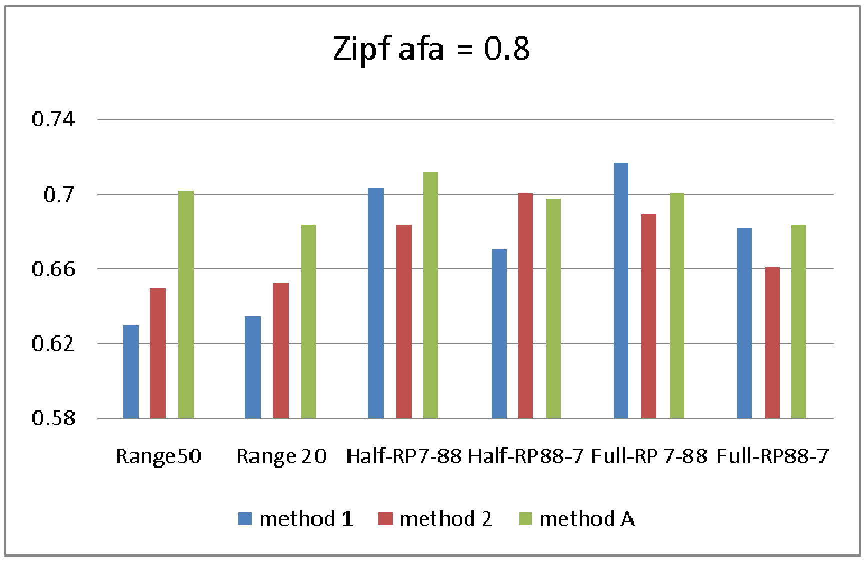
Figure 23 (c) shows the method 2 can enhance the hit-rate of method 1, and the enhancement that method 2 improved can be more when the buffer size is increased. The method 3 outperforms the method 1 and method 2 while the buffer size of a peer has distinct limits except the constraint of buffer size is 120 minutes.

In figure 23 (d), the horizontal axis represents the amount of peers arrived per minute. The method A performs well in terms of the chunk hit-rate in different peer arrival rates. Especially when the arrival rate in the system is low; method A can improve more enhancement than method 2 because of the peers retained the segments that online peers are actually interested at this time. This demonstrated the advantage of method A which accomplished the results by adaptive replacement.

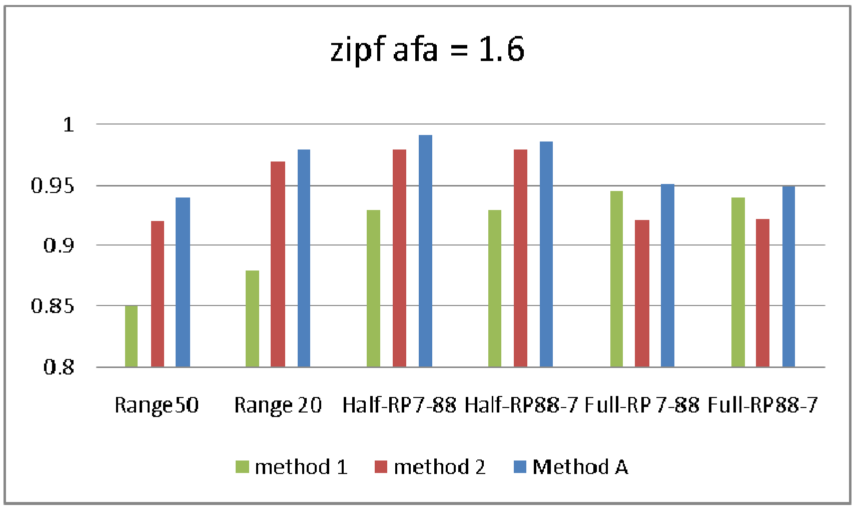
Figure 23 (e) shows the performance evolvement of three methods in terms of the chunk hit-rate over fifty days. The arrival rate in system becomes double from the 12000th minute to the 12120th minute, and the leave rate becomes double from the 42000th minute to the 42120th minute.

The super peers in method A have a responsibility to collect information such as cache value and the replica of a segment from other super peers. As figure 23 (f) showed, we observe method A can improve more enhancement while the super peers collect information from more groups. There are large gaps among the curves of 3.2%, 14.7%, and 69.8%, but small gaps among 69.08%, 86.9%, and 92.9%. Since collecting information from more super peers suffer from the cost of more workload on the super peers and the traffic on the networks, the results in figure 23 (f) implies that collecting from 69.08% super peers is a good choice for the tradeoff between cost and performance.

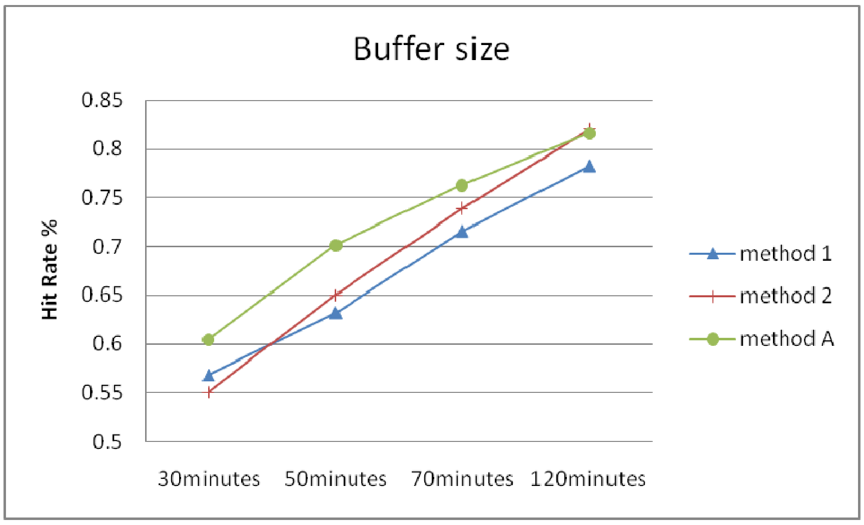
We calculate the average time how long a segment hasn’t been accessed before it’s removed. As figure 23 (g) shows method A can has more sensibility to detect the unpopular segments when the supper peers collected information from more super peers.



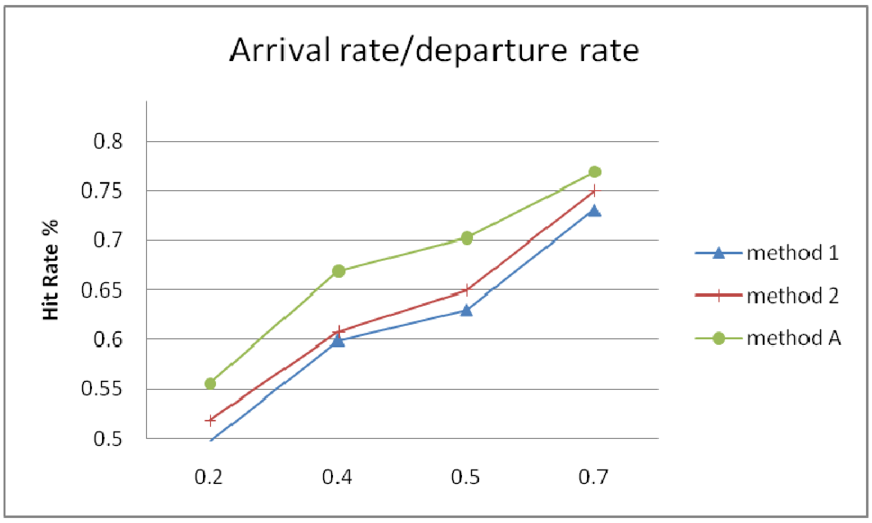
(a) various popularity model for Zipf with α =0.8



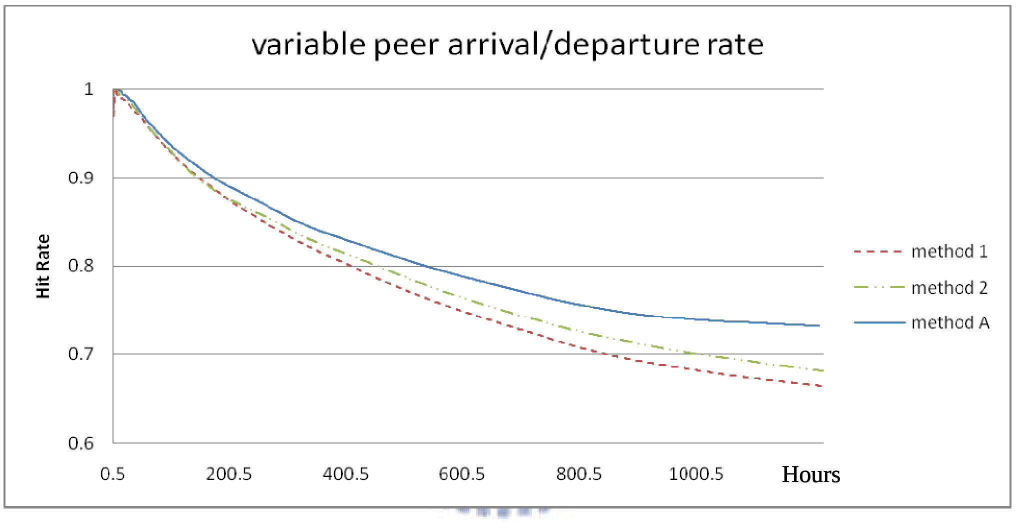
(b) various popularity model for Zipf with α =1.6



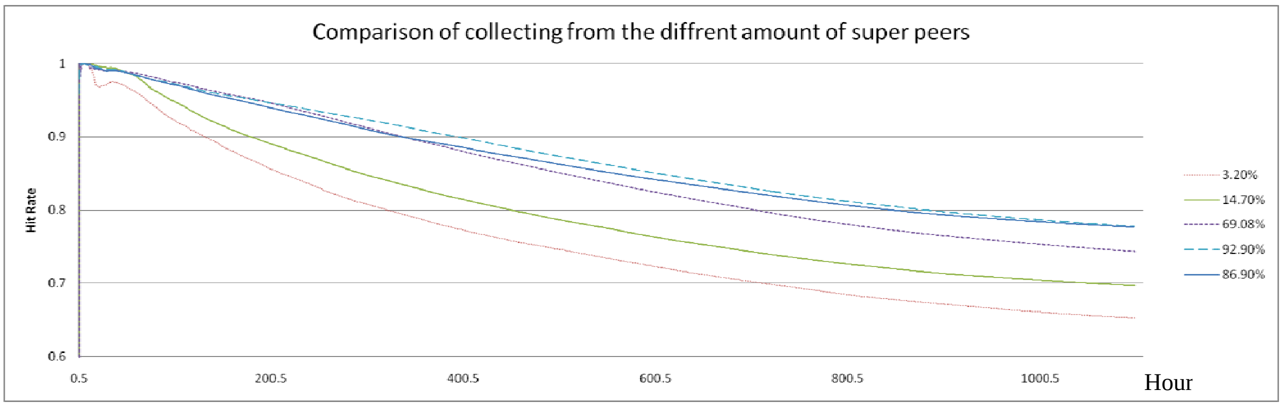
(c) hit-rate as a function of buffer size



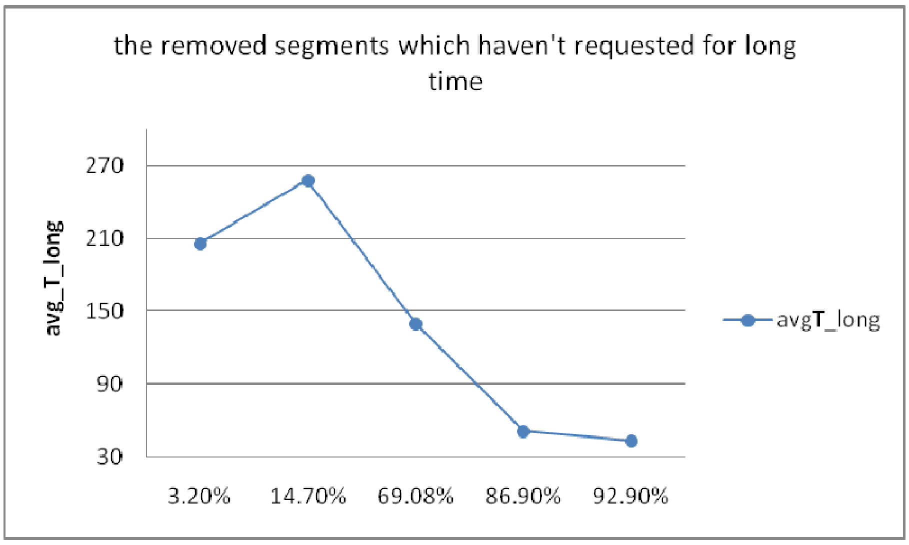
(d) hit-rate as a function of arrival and departure rates



(e) hit-rate of variable arrival and departure rates over time



(f) hit-rate of collecting information from different amounts of super peers over time



(g) The average time how long a segment hasn't been accessed before it's removed as a function of the amount of super peers collected from

Figure 23 Performance comparisons for three caching strategies

Chapter 5 Conclusion

Instead of providing services from pre-stored videos on video servers as conditional VoD systems, we proposed a P2P system which services VoD from cached live streaming. The general solutions for VoD assumed all peers receive the streaming from initial part of whole video, and the media streaming is pre-stored in each peer's buffer in P2P system. In this thesis, we proposed three caching strategies "Continuous overwriting", "No overwriting", and "Adaptive overwriting" for P2P near-live streaming applications. The ideal satiation of "Continuous overwriting" is most of peers are receiving the same segment, so that they can share streaming to each other. And the "No overwriting" caching strategy can approach higher hit-rate while each time interval has arrived peers that store corresponding segments in their cache. "Adaptive overwriting" strategy take both the variable bandwidth of each peer, the amount of a segment's replica, and life-cycle of a segment into considerations. In our experiments, the "Adaptive overwriting" strategy can result in better performance in terms of chunk hit-rate. In the future works, we can adjust the sliding time window size dynamically according to different population models instead of the fixed size. And another issue we are interested in is the relevance among contents, if the popularity of a video can be predicted by characterizing the identity of contents, the trivial video can be removed to make the utilization of cache efficiently.

Reference

[1] Qi Huang, Hai Jin, Xiaofei Liao, "P2P Live Streaming with Tree-Mesh based Hybrid Overlay.", 2007 International Conference on Parallel Processing Workshops

[2] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming." In *Proceedings of IEEE INFOCOM*, March 2005.

[3] Xiaojun Hei, Chao Lian, Jian Liang, Yong Liu, Keith W. Ross, "A Measurement Study of a large-scale P2P IPTV System", 2007 IEEE

[4] W. F. Poon and K. T. Lo, "New batching policy for providing true video-on-demand (T-VoD) in multicast system," in *Proc. IEEE ICC*, vol. 2, 1999, pp. 983–987.

[5] K.M. Ho, W.F. Poon and K.T Lo, "Peer-to-peer Batching Policy for Video-on-Demand System", 2006 IEEE

[6] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley, "P2Cast: Peer-to-peer patching scheme for VoD service." in *Proc. WWW'03*, Budapest, Hungary, May 2003.

[7] Anna Statsiou, Michael Paterakis, "Frequency-based cache management policies for collaborative and non-collaborative topologies of segment based video caching proxies", Springer-Verlag 2006

[8] E.J. Lim, S.H. Park, H.O. Hong, K.D. Chung, "A Proxy Caching Scheme for Continuous Media Streams on the Internet", To appear in ICOIN-15, 2001.

[9] PARK, S. H., LIM, E. J., AND CHUNG, K. D. 2001." Popularity-based partial caching for VOD systems using a proxy server." In *Proceedings of the Workshop on Parallel and Distributed Computing in Image Processing, Video*

Processing and Multimedia. IEEE Computer Society, Piscataway, NJ.

[10] C. Griwodz, M. Bar, and L. C. Wolf. Long-term movie popularity models in video-on-demand systems. In *Proceedings of ACM Multimedia 1997*, Seattle, WA, November 1997.