

國立交通大學

資訊科學與工程研究所

碩士論文

使用再造編碼及快取技術之低成本高可用性點
對點儲存方法

A low cost and high availability P2P storage scheme with
Regenerating Code and caching methods

研究生：黎光明

指導教授：袁賢銘 教授

中華民國九十八年六月

使用再造編碼及快取技術之低成本高可用性點 對點儲存方法

研究生：黎光明

指導教授：袁賢銘

國立交通大學資訊科學與工程研究所

摘要

高的資料可靠度是分散式儲存系統的重要特性之一。錯誤更正碼(erasure code)是一個常用的方法來產生冗餘(redundancy)的資料，不過這個方法需要原始檔案來產生冗餘的資料。而再造編碼(Regenerating Code)分散式地蒐集多個節點(peer)的資料來產生冗餘，則不需要維護一份原始的檔案。不過這樣做也有一些缺點，存取時無法只透過一個節點(peer)來取得資料，加上編碼時需要更多的節點(peer)同時提供資訊才能達到比傳統方法更好的效率，而這在點對點(Peer-to-Peer)的環境下是一個嚴苛的條件。

我們在不額外維護一份原始檔案的條件下，利用最近下載完的節點(peer)中的使用最近最少使用演算法的快取(LRU cache)中的資料來提高存取的效率以及降低編碼的成本，我們用模擬的方式，在不同快取大小以及不同節點可靠度(peer availability)之下，記錄最近存取檔案的節點的資訊。在實驗中，我們將一個檔案被分成 7 個區塊(block)，並且發現這些被記錄的節點在快取大小為 64 個檔案區塊(block)的條件下，有 83%以上的比例可以改善存取效率和編碼成本。

A low cost and high availability P2P storage scheme with Regenerating Code and caching methods

Student : Guang-Ming Li

Advisor : Shyan-Ming Yuan

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

High data availability is one of the important properties in P2P storage system. To support the property, erasure coding is a common method to generate redundant data, but the technique requires an original file in the creating process. Regenerating Code solves the problem through collecting encoding information in a distributed way; so it does not need to save the replica. However, there are still some drawbacks in the design. When accessing a file, the peer cannot only communicate to one peer to get enough file blocks. Besides, in the encoding procedure, the scheme suggests that the number of connected peers should be large enough, larger than the number of decoding blocks, so that the maintenance cost can be lower than the traditional way. The suggestion is a harsh condition in P2P environment because there should be simultaneously more peers alive and keep the file blocks in the system.

On the condition that the system does not store a replica additionally, we utilize the data in the LRU cache of the peer last accessed a file to improve the access performance, and reduce the maintenance cost. Our scheme records the information of peers last accessed files to achieve the goal. Through the simulation, we experimented with different cache size under various peer availabilities. The experiments results showed that both the access performance and maintenance cost are improved at least 83% in the Regenerating Code scheme when the cache size is 64 file blocks, where a file is divided into 7 blocks.

Acknowledgement

首先要感謝我的老師袁賢銘教授的指導，在研究的過程中不斷的給我建議。讓我能夠突破許多的難題與瓶頸，並得到很多的啟發。也感謝實驗室中的陳育良學長花了很多時間陪我找尋研究的方向並教導我許多網路領域的知識，還有其他博士班的學長高永威、林家鋒和宋牧奇等，在我研究的過程中透過他們經驗的分享給我指引努力的方向。感謝實驗室的學長簡士強，介紹 Network Coding 領域的基礎概念，讓我能盡快進入狀況。感謝實驗室的同學羅國亨、洪偉翔、鄭婷文和周東興等，在這二年來一起研究討論，一起分享學習過程的辛勞與快樂，讓我得到許多寶貴的回憶。最後要感謝我的父母，給予我最好的學習環境和機會，謹以本篇論文來感謝您們的養育之恩。



Table of Contents

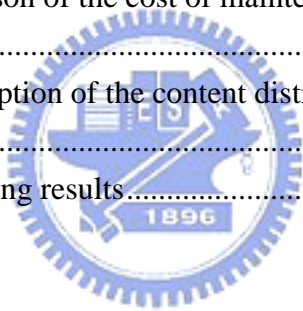
摘要.....	i
Abstract.....	ii
Acknowledgement.....	iii
Table of Contents.....	iv
LIST OF FIGURES.....	v
LIST OF TABLES.....	vi
1. Introduction.....	1
1.1. Preface.....	1
1.2. Motivation.....	1
1.3. Problem Description.....	3
1.4. Research Objective.....	4
1.5. Research Contribution.....	4
1.6. Thesis Outline.....	5
2. Background and Related Work.....	6
2.1. Background.....	6
2.1.1. Peer-to-Peer (P2P) Network.....	6
2.1.2. Network Coding.....	7
2.2. Related Work.....	8
2.2.1. Distributed Storage System.....	8
2.2.2. Redundancy Scheme.....	9
2.2.3. Regenerating Code.....	11
3. Regenerating Code with Cache.....	14
3.1. System assumption and setting.....	14
3.2. System design.....	16
3.3. System analysis.....	19
4. Experimental Result and Analysis.....	25
4.1. Experimental Setup.....	25
4.2. Evaluation and Analysis.....	26
4.2.1. Experimental results of access.....	26
4.2.2. Experimental results of generating redundant data.....	31
5. Discussion.....	37
6. Conclusion and Future Work.....	39
6.1. Conclusion.....	39
6.2. Future Work.....	39
7. References.....	41

LIST OF FIGURES

Figure 1-1 Layered mCDN architecture. From [33].....	2
Figure 2-1 A repair for a (4,2)-Minimum-Storage Regenerating Code, from [8].....	12
Figure 3-1 The relationship between the blocks and packets in a file	15
Figure 3-2 The composition of a coding block.....	15
Figure 3-3 A peer registers the block d to 3 indexers	17
Figure 3-4 The flow char of creating a coding block.....	19
Figure 3-5 Analysis of the hybrid scheme	20
Figure 3-6 Analysis of the Regenerating Code	21
Figure 3-7 Analysis of the Regenerating Code with LRU cache (Case 1) ..	23
Figure 3-8 Analysis of the Regenerating Code with LRU cache (Case 2) ..	24
Figure 4-1 The access results of the case the indexed peers have 7 blocks for each peer availability	28
Figure 4-2 Comparison of the connection times for peer availability = 0.9	30
Figure 4-3 Comparison of the connection times for peer availability = 0.65	30
Figure 4-4 Comparison of the connection times for peer availability = 0.4	31
Figure 4-5 The results of generating redundancy by the indexed peers for each peer availability	33
Figure 4-6 Comparison of the average cost of generating a coding block for peer availability = 0.9	35
Figure 4-7 Comparison of the average cost of generating a coding block for peer availability = 0.65	36
Figure 4-8 Comparison of the average cost of generating a coding block for peer availability = 0.4	36

LIST OF TABLES

Table 3-1 The successful probabilities of the encoding of the Regenerating Codes	22
Table 4-1 Individual parameters for each peer availability.....	26
Table 4-2 Common parameters for all peer availability	26
Table 4-3 The access results for peer availability = 0.9.....	27
Table 4-4 The access results for peer availability = 0.65.....	27
Table 4-5 The access results for peer availability = 0.4.....	28
Table 4-6 Comparison of the connection times for each availability	29
Table 4-7 The results of generating redundancy for peer availability = 0.9	32
Table 4-8 The results of generating redundancy for peer availability = 0.65	32
Table 4-9 The results of generating redundancy for peer availability = 0.4	33
Table 4-10 Comparison of the cost of maintenance for each peer availability	35
Table 5-1 The description of the content distributed components in mCDN	37
Table 5-2 The mapping results.....	38



1. Introduction

1.1. Preface

In the age of knowledge explosion, information growth in the Internet is very fast. Computers today need more powerful processing ability and larger storage capacity to face the information growth. Powerful workstations or servers having limited scalability for storage and limited computing power for service can't satisfy the requirement, whereas Peer-to-Peer storage system supporting high scalability, good performance and fault tolerance becomes a practical and better solution. However, it is not trivial to design a Peer-to-Peer storage system. In the paper [32], they indicate that there are several techniques to design a Peer-to-Peer storage system, namely data redundancy, data placement, failure detection and data maintenance. In this thesis, we focus on the data redundancy which considers how to use optimal space and bandwidth cost in Peer-to-Peer storage system to approach the goal of data availability.

1.2. Motivation

Although there are many designs for Peer-to-Peer (P2P) storage systems, such as OceanStore[28], CFS[19], PAST[31] and Total Recall[23], we are interested in the application of P2P storage system. Multimedia Content Discovery and Delivery (mCDN) architecture [33] is an interesting application where P2P storage system plays an important role. mCDN architecture is a new architecture for Content delivery networks (CDNs), and can combine many CDN services to support a variety of business models. It defines three layers (Shown in Figure 1-1 Layered mCDN

architecture, from [33]). First, Content Service Layer contains all entities providing functionality to the end users of mCDN and content service. Second, Content Distribution Layer is responsible for distributing and maintaining content and metadata. Third, Network Infrastructure Layer contains all components observing network state and trigger mCDN related work events.

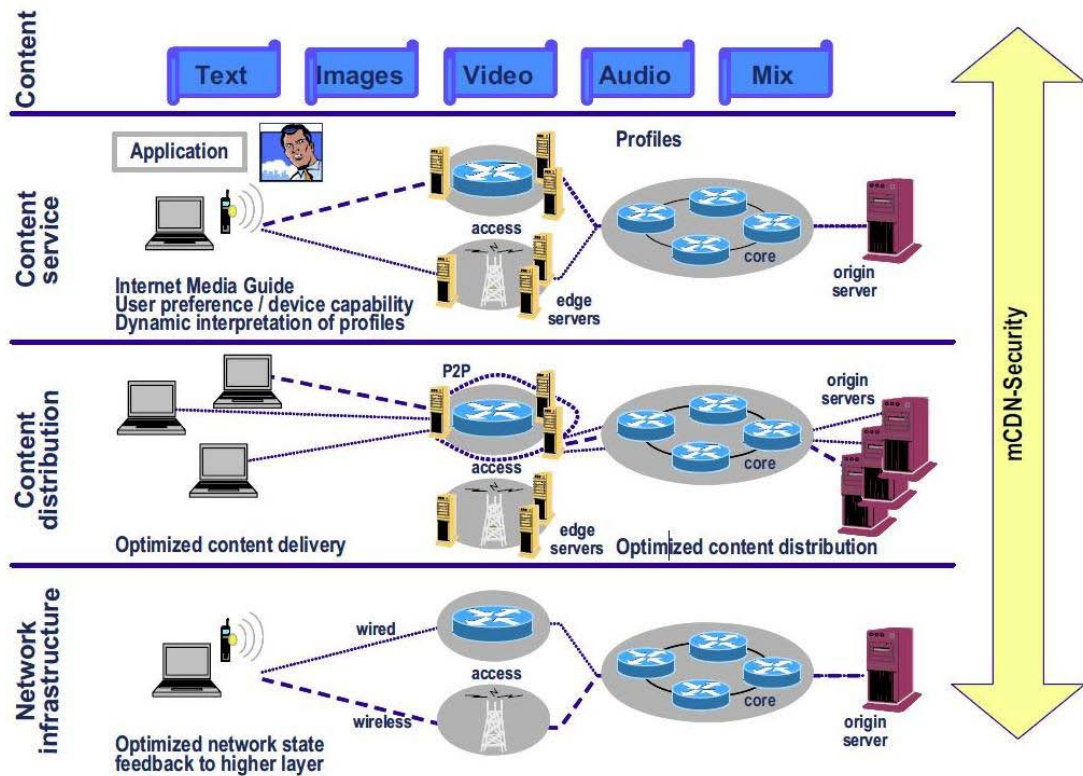


Figure 1-1 Layered mCDN architecture. From [33]

A P2P storage system can be used in Content Distribution Layer to provide content service. It can place optimally the content to each peer and allow users access quickly the requested content. For P2P storage system, the data availability will impact directly the service quality, so we are concerned about it in system design. In the web application, the data availability that end users expect is at least three nines (99.9%) [34] . To achieve high availability, it can't avoid spending more storage space. Replication scheme and erasure coding scheme were common methods to create redundant data before, but recently Network Coding is gradually popular and is

applied to generate redundant data. Regenerating Code [8] utilizes the property of Network Coding to solve some problems occurring in erasure coding, but meanwhile their scheme is not efficient for requesting. It is very critical for web application to supply fast response to users, so our work is to overcome the drawback to make their scheme more practical in web application.

1.3. Problem Description

For the data redundancy research, the previous comparisons concentrated on replication and erasure coding and the two targets, low storage cost and low bandwidth cost, are the key analysis points in the design of P2P storage system. Unlike to replication scheme just distributing full replica to other peers, erasure coding scheme requires an encoding process before sending coding blocks. The replication scheme is simpler, but erasure coding scheme offering better storage efficiency is the winner in their comparisons; nevertheless, erasure coding scheme has to exist a full file before the system start to generate new coding blocks, which is a limitation of redundancy mechanism causing erasure coding scheme to keep additionally one full replica in system, which is called hybrid scheme. Hybrid scheme including the advantage of two schemes reduces the cost of bandwidth and storage space and becomes the best one in later analysis.

Regenerating code breaks the limitation by using the linear relationship of Network Coding, which is that coding blocks can be generated through collecting enough existing coding blocks. They surprisingly show the result and find that hybrid scheme complicates the redundancy management and has the bottleneck in disk I/O because storing the full replica, so advocate that coding scheme should only manage one type of redundancy. In their analysis, they conclude that their scheme spend lower cost of storage and bandwidth than hybrid scheme but is useful for backup application

due to lower access performance. Besides, we consider that their generating method has a condition, which is that there must be enough, more than the same setting of erasure coding, peers alive in P2P storage system. In the dynamic environment, the peers will join and leave at any time, so the success probability of their method is lower. For example, we suppose there are 14 coding blocks allocated to 14 different peers in a P2P storage system, and then the probability of 7 of the 14 peers concurrently occurring is higher than the probability of 13 of the 14 peers.

In short, our work is to extend their scheme to support efficient data service in web application.

1.4. Research Objective

Through the observation above, the problem is how to extend their scheme in request aspect by following their central idea, only storing one type of redundancy. We start at this point and put in other idea next. In the research [27], they take user download behavior into account. As a result, the system only pays little overhead to make hybrid scheme get better. Similarly, we consider that even the system just owns coding blocks, but in runtime a whole file will be reconstructed in the system when some peer wants to read it. In real system, each peer usually allocates some storage space for its LRU cache to save used data. The full copy will take place there, so our research objective is how to exploit it before it is replaced in cache and the peer leaves the P2P system.

1.5. Research Contribution

We implement our scheme which lets Regenerating Code system more useful for content service in a discrete-event packet level simulator, p2psim [21]. Our scheme basically follows the Regenerating Code scheme and sets a LRU cache with some

fixed size to each peer in the P2P environment. At each access, the system records the information of peers last accessed files and then that will be the first choice when the system wants to generate new coding block or some peer wants to access a file. We do some experiments with two parameters, cache size and peer availability. The experiment results showed that in different peer availability the LRU cache with size of 64 blocks can help to accelerate the most access in Regenerating Code system. In addition, we find that in P2P environment even though we set the LRU cache size very large the benefit is limited because each peer may leave and carry the data away. Finally, even if the file blocks in cache are not enough to reconstruct the original file, the peer still can decrease the connection times. Then the bandwidth cost is down and the success probability of encoding is higher since the peer only needs to connect to fewer peers.

On the other hand, based on our research, we consider that it is possible that P2P storage systems don't need to additionally maintain a full file. Although the authors of Regenerating Code want to use the Regenerating Code to achieve the goal, the access performance of the scheme is not acceptable for content service. In our research, we overcome the drawback and then the goal is possible.

1.6. Thesis Outline

We arrange the remainder of this thesis as follows. In chapter 2, we introduce some associated knowledge and show the details of each redundancy scheme, which will assist to understand and analyze the redundancy policy. In chapter 3, we describe how we use the LRU cache to decrease the cost of access and maintenance. In chapter 4, we explain the design of experiments and analyze the improvement. In chapter 5, we discuss how to use P2P storage system in mCDN architecture. Finally, in chapter 6, we conclude our design and analysis.

2. Background and Related Work

2.1. Background

2.1.1. Peer-to-Peer (P2P) Network

With regard to the P2P network, most computer users might associate with some well-known P2P application in thinking, such as BitTorrent and Skype. The P2P architecture can connect all participants and cumulate all the bandwidth and computing in the P2P network to provide all kinds of services for the members in P2P network and it is unlike traditional client/server architecture, or called server-based architecture, which relies on relatively low number of servers to serve overall clients. It allows users to access to its resources by other systems and supports resource sharing, which requires fault-tolerance, self-organization, and massive scalability properties. [17] With the powerful properties above, it becomes an important research target and a useful application today.

There are many classifications of P2P networks. For example, it can be classified by what they can be used for, such as file sharing, telephony, media streaming and discussion forums. Other classification may be according to the degree of centralization. We here introduce one of unstructured and structured P2P networks. [17] The P2P network consists of all the participating peers as network nodes. If a peer knows the location of the other peer, then there is a directed edge from the former node to the latter. Any P2P network can use the links to form a graph, and this classification uses the characteristic to classify the P2P networks as unstructured or structured. The unstructured P2P network organize peers in a random graph either in a

flat or hierarchical manner, so it can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time. The main disadvantage with unstructured P2P networks is that the queries may not always be resolved. Because the queries have to be flooded through the network to find as many peers as possible, they may not be successful to find all the peers in the network. Structured P2P network usually has a distributed hash table (DHT) to resolve the problem for the reason that DHTs propose a determined object locating service. There are already some studies about its abstraction, such as Chord [15] , Pastry.[14] and the study [13] , and some applications using it, such as these papers [16] [20] , CFS[19] , OceanStore[28] and PAST[31] .

On the other hand, there is an important requirement to the P2P application, which is due to that the P2P network is an overlay network on the Internet. Sometimes two nodes are logically near, but their distance is very long physically, so some research uses some methods, such as response time, to judge if two nodes are near really. The problem may reduce the service performance of P2P storage system as the peer does not connect to the nearest peer owned files to get the data. In the mCDN, this is a requirement in content service.

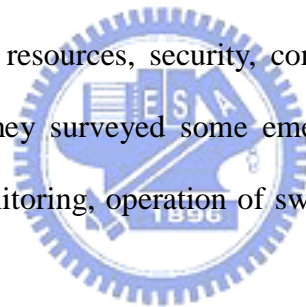
2.1.2. Network Coding

In recent years, Network Coding is gradually popular. The research domain began at the paper [5] , where showed Network Coding can increase the network throughput, and in the multicast case can achieve the maximum data rate theoretically possible. In the later research [2] [3] , they demonstrated that linear Network Coding can achieve maximum multicast capacity, and then linear Network Coding is enough to use. Following them, in [1] and [9] they proved random linear Network Coding can achieve maximum multicast capacity when the field size is large enough. Finally, to

make Network Coding more practical, in [7] they gave deterministic polynomial time algorithms for designing linear codes for directed acyclic graphs with edges of unit capacity.

Besides these theory research above, in [4] , an instant primer for network coding which explains what network coding does and how it does it, points out some practical consideration. There is an example used in P2P environment from [35] where they use linear Network Coding to design a single source P2P file sharing.

The core notion of network coding is to allow mixing of data at intermediate network nodes; therefore, many collisions of data flows can be eliminated. It overthrows the concept of conventional routing which considers that encoding at intermediate network nodes is not efficient. Network Coding can use in many places, such as throughput, wireless resources, security, complexity, and resilience to link failures [11] , and in [12] they surveyed some emerging applications of Network Coding, namely network monitoring, operation of switches, on-chip communication, and distributed storage.



2.2. Related Work

2.2.1. Distributed Storage System

A number of studies about distributed storage [6] [36] [37] [38] [39] made use of Network Coding to acquire data in the sensor network. In [30] they proposed a scheme for P2P content distribution of large files that is based on network coding. Their work improved the efficiency of file downloading and the robustness of the system. The study [18] , using Network Coding by means of presenting a detailed performance analysis of the P2P system, also contributed to P2P content distributed system. For the data maintenance in distributed storage system, the paper [8] they

presented fundamental bounds and constructions for network codes that need to be maintained over time.

The data redundancy strategies for the system were compared and evaluated in [22] [24] [25] [26] . Erasure coding scheme with higher storage and bandwidth savings is better than replication scheme when the peer availability is low. The main consideration includes storage, bandwidth and overhead of encoding complexity. Another research in [10] proved that erasure coding brings the advantage if the condition below is satisfied. The two key parameters in system are the storage overhead S and peer availability μ and then S and μ should obey $S * \mu > 1$. Other research [27] [29] tried to adopt the hybrid scheme, namely erasure coding and replication, to get the efficient access from replication scheme and reach more optimal tradeoff between benefits of erasure coding and its overhead. The hybrid scheme actually is successful to achieve the goal, but in [8] [25] they argued that hybrid's reduced bandwidth is limited. Therefore, in the Regenerating Code system, there is only one type data maintaining there.

2.2.2. Redundancy Scheme

Here we introduce the details of the three redundancy strategies, replication, erasure coding and hybrid that were analyzed before.

Replication scheme is the simplest of the three. In the P2P system, each file will be duplicated and kept in n different peers, where n is the redundancy setting in the system. When any peer wants to access the file, it just chooses one to request the file. If all the n replicas are disappeared, then the file is unavailable. For each file, the system will maintain its copies to the required number. The number n is depended on system availability target, $1 - \epsilon$, which has some number of nines. The equation is determined as below, where p is the peer availability which is independent and

identically distributed in the system.

$$\varepsilon = (1 - p)^n \quad (1)$$

which upon solving for n yields

$$n = \frac{\log \varepsilon}{\log(1 - p)} \quad (2)$$

In the erasure coding scheme, each file is divided into k blocks and the k blocks are encoded into n coding blocks. When collecting any k of n coding blocks, a peer can decode them to recovery the original file and retrieve it. On the other hand, when the system wants to create new a coding block, the encoding peer must get either a copy of original file or k coding blocks. A well-known erasure code is Reed-Solomon code [40] which is a class of Maximum Distance Separable codes (MDS). If there are fewer than k coding blocks existing in the system, then the file is unavailable. We exhibit the redundancy analysis as follows. The redundancy factor is $r = \frac{n}{k}$ in the scheme and the assumption about availability is the same to replication scheme.

$$1 - \varepsilon = \sum_{i=k}^{r*k} \binom{r*k}{i} p^i (1 - p)^{r*k-i} \quad (3)$$

In [41] , using algebraic simplifications and the normal approximation to the binomial distribution, they get the following formula for the erasure coding redundancy factor:

$$r = \frac{n}{k} = \left(\frac{\sigma_\varepsilon \sqrt{\frac{p(1-p)}{k}} + \sqrt{\frac{\sigma_\varepsilon^2 p(1-p)}{k} + 4p}}{2p} \right)^2 \quad (4)$$

where σ_ε is the number of standard deviations in a normal distribution for the required level of availability.

The hybrid scheme has both the coding blocks and the full file in the system and the coding blocks and the full file are allocated to different peers in the system. The file is unavailable when both the full file is not presenting and the number of coding

blocks is smaller than k . In the scheme, the benefit is that the full copy can support not only encoding but also access. The redundancy factor corresponds to increasing the redundancy factor of erasure coding by 1. The related formula is similar to erasure coding and the assumption is also the same as before.

$$1 - \varepsilon = 1 - (1 - p) \left(1 - \sum_{i=k}^{r*k} \binom{r*k}{i} p^i (1 - p)^{r*k-i} \right) \quad (5)$$

The equation is derived in [27]

$$r = 1 + \frac{n}{k} = 1 + \left(\frac{\sigma_\varepsilon \sqrt{\frac{p(1-p)}{k}} + \sqrt{\frac{\sigma_\varepsilon^2 p(1-p)}{k} + 4p}}{2p} \right)^2 \quad (6)$$

Here the ε is derived as below where ε' is equal to the ε in erasure coding scheme and the equation (5) is used.

$$1 - \varepsilon' = \sum_{i=k}^{r*k} \binom{r*k}{i} p^i (1 - p)^{r*k-i} \quad (7)$$

$$\varepsilon' = \frac{\varepsilon}{1 - p} \quad (8)$$

2.2.3. Regenerating Code

Regenerating Code, a new technique using linear network codes and proposed in [8], allows a new node to download functions of the stored data from the surviving nodes, which is the different point to erasure coding. The pair (n, k) is that there are n coding blocks for each file in the system and any k of the n coding block can reconstruct the file. The decoding requirement, namely the k number, is the same to erasure coding. They exploited information flow graph, a particular graphical representation of a distributed storage system, to analyze how Regenerating Code works and its principle, and showed that there is a fundamental tradeoff between storage and repair bandwidth. There are two special cases of the storage-bandwidth

tradeoff—Minimum-Storage Regenerating (MSR) Codes with the best storage efficiency and Minimum-Bandwidth Regenerating (MBR) Codes with the minimum repair bandwidth.

An illustrated example is in Figure 2-1 from [8]. It is a (4,2)-Minimum-Storage Regenerating Code. A file is divided into two blocks and then each block contains two packets with the same size, so each peer stores two packets. For the decoding, a peer should retrieve any two blocks to reconstruct the file. When creating a block, the new peer mixes the packets from three other peers which do not send their information but generate smaller parity packets of their data and forward them. The coefficients of linear operation are selected at random. In the case, the repair bandwidth is smaller than the size of original file. In contrast, erasure coding scheme has to access the original file to generate a new encoded block.

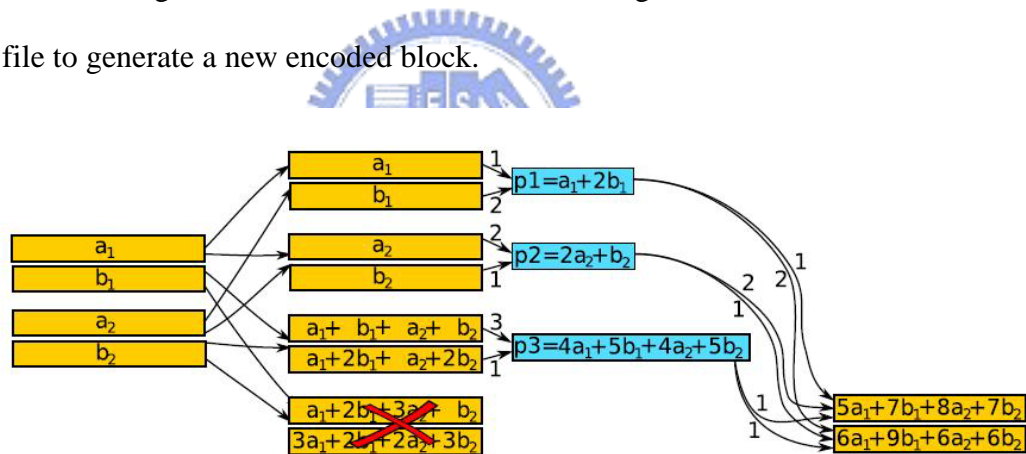


Figure 2-1 A repair for a (4,2)-Minimum-Storage Regenerating Code, from [8]

There is an important number d , which is the communication number of repairing a coding block. For example, d is equal to three in the (4,2)-Minimum-Storage Regenerating Code but d is equal to two in the (4,2)-erasure code. It is clear that d in the Regenerating Code is larger than erasure code, and in their analysis the benefits of the Regenerating Code, due to the saving of the storage and bandwidth, are better as d is larger. Therefore, they let d be very close to n in their

analysis to get more optimal results, which is that they set $d=n-1$. The relationship of the storage and bandwidth of MSR and MBR is showed as follows and the complete proof of this theorem is given in the Appendix of [8] .

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{\mathcal{M}}{k}, \frac{\mathcal{M}d}{k(d-k+1)} \right) \quad (9)$$

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2\mathcal{M}d}{2kd - k^2 + k}, \frac{2\mathcal{M}d}{2kd - k^2 + k} \right) \quad (10)$$

Where α is the block size, β is the packet size, $\gamma = d\beta$ and \mathcal{M} is the original file size.



3. Regenerating Code with Cache

In this chapter, we describe our scheme in detail. In the section 3.1, we interpret the setting of Regenerating Code and the assumption where our scheme is suitable to use. In section 3.2, we demonstrate how we design the P2P storage system. Finally, in the section 3.3, we analyze and illustrate the lookup and redundancy maintenance through some cases expected to make our system better in runtime.

3.1. System assumption and setting

Suppose all peers in the P2P storage system have two types of storage space, permanent space and temporary space. The files in permanent space will exist in the system until the peer crashes or leaves, but the files in temporary space may be replaced if the space is full. The replacement rule follows the algorithm of the LRU cache. Here we name the permanent space “database” and name the temporary space “LRU cache”. Our work adopts Chord[15] to maintain routing tables for lookup and query management. Each peer and each file have their unique id.

The target of file availability is set to 99.9%. We utilize the $(n, 7)$ -Regenerating Code to encode the files where n is decided by the peer availability and 7 is used and analyzed in DHash++[20] , and we set $d = 13$. Then each file is divided into 7 blocks and each block contains 7 packets (see Figure 3-1). Clearly, the total packet number in a file is 49, and these packets are used to generate the coding blocks. Each packet in the coding blocks also stores its combination coefficients related to the 49 packets in the initial file (see Figure 3-2).

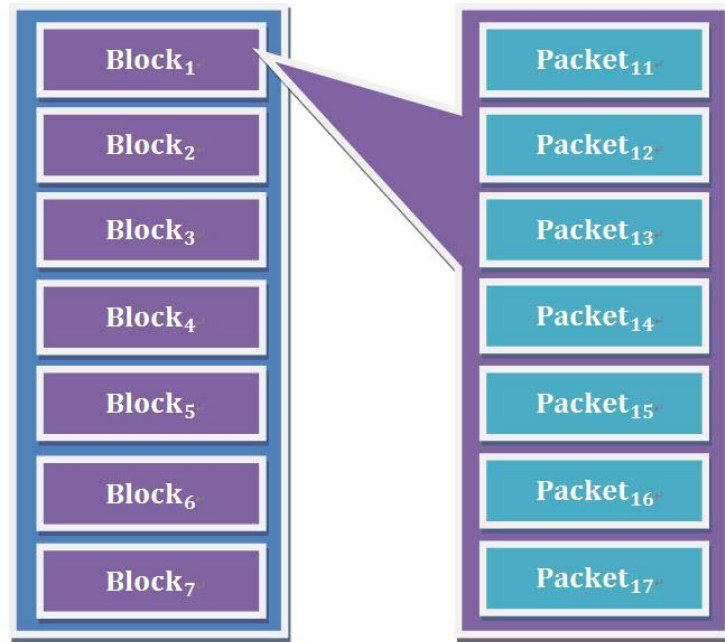


Figure 3-1 The relationship between the blocks and packets in a file

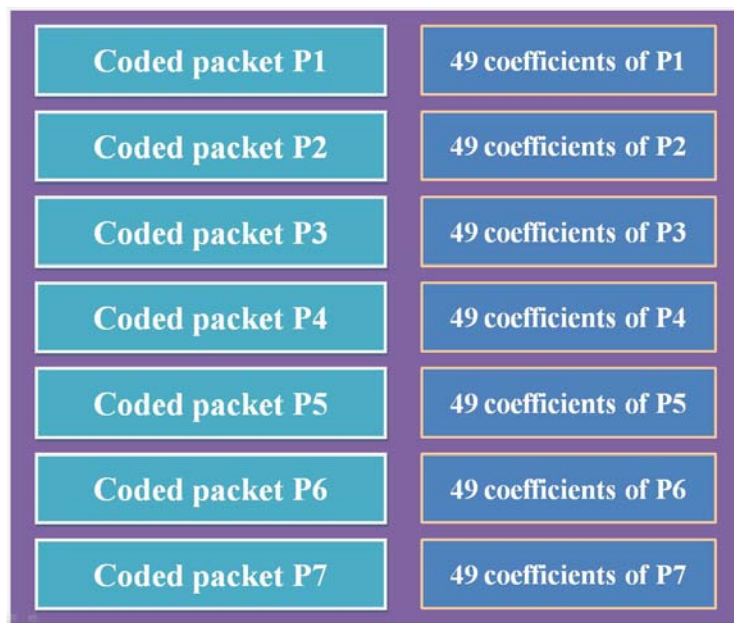


Figure 3-2 The composition of a coding block

For the linear combination, we let the field be large enough to be able to decode successfully and suppose the packet size be also large enough to neglect the storage overhead of the coefficients.

We choose the MSR codes, a case of Regenerating Code, to implement in the P2P storage system. By our setting above, we get the pair of the block size and repair

bandwidth $\left(\frac{M}{7}, \frac{13M}{49}\right)$ from formula (9). The MSR codes here require communicating with 13 different peers to encode a new coding block. If there are not sufficient peers in the system, we change the way to collect 7 different blocks, which is the same as erasure coding scheme.

3.2. System design

Our scheme is fundamentally based on the scheme in [27]. The difference is that we encode with Regenerating code, take the LRU cache into account and add some information in DHT as the peers are requesting to make the access and the encoding more efficient. Overall, each peer is responsible for three works, indexer, register and maintenance, which controls the data placement, the data lookup and the data availability in the P2P storage system. These mechanisms are supported by the Chord protocol in the bottom layer.

For the indexer and the register, each peer periodically registers the unique file IDs of all the holding blocks in its database with some indexers of the file. Here the coding blocks in each peer belonged to the same file have the same ID but own different coefficients. When the indexers receive the report, they record a new index including the IP of the reporting peer and the file IDs in the report into their index table and they set a timer to decide if they should remove these indexes. The indexer can use the timer to detect the situations of the indexed peers, namely the peers are alive or not, in the system. The first indexer is decided by the hash function used in DHT $H(ID)$ and others are the adjacent and continuous successors of the first indexer. We illustrate this in Figure 3-3.

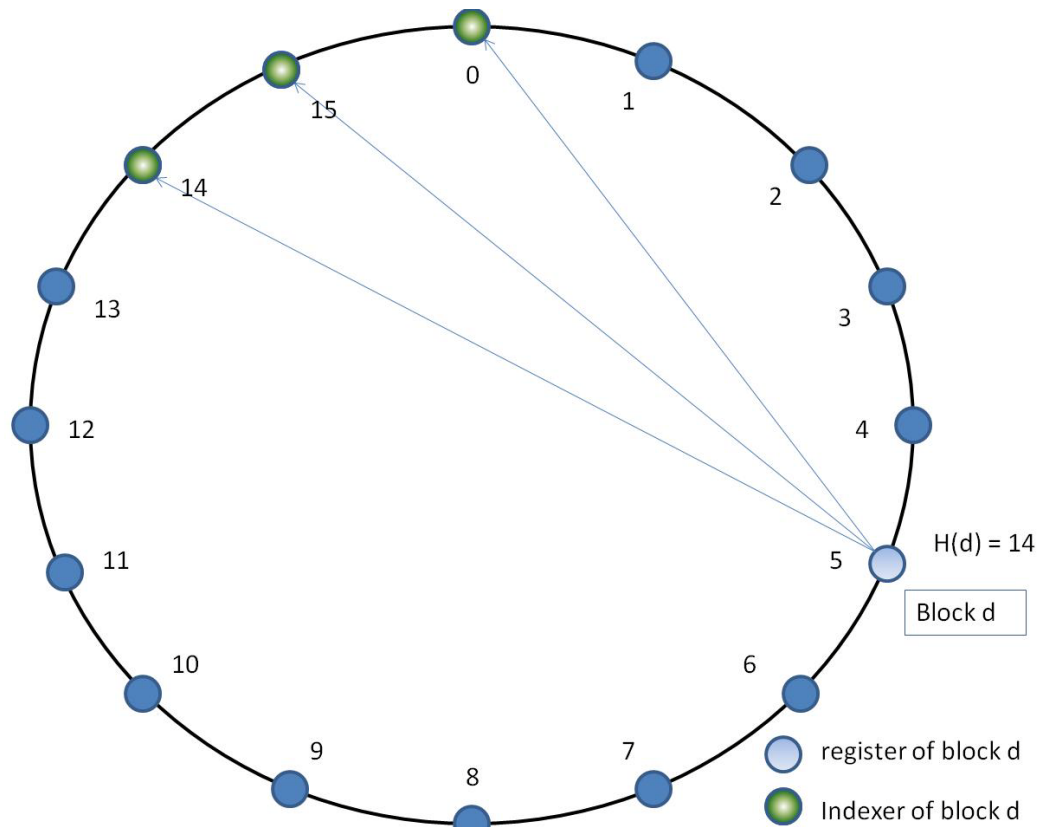


Figure 3-3 A peer registers the block d to 3 indexers

For the maintenance, each peer is assigned to manage the blocks of the some files according to the hash function $H(ID)$, and then it periodically evaluates the number of registered blocks in its index table. If the number is under the redundancy target, it invokes the event of generating new coding blocks to increase the availability of the file before communicating with other indexers and confirm that the redundancy actually is not enough.

Next, we explain how to request a file and how to create a new coding block. In the two operations, each peer will check both its database and LRU cache before request to other peer and each requested peer also replies the holding information of the requesting file through its database and LRU. The communication has two steps as a peer wants to request a file from the target peer. First, the peer requests a file to the target peer, and then the target peer compares all its owning blocks ID with the file ID

to produce a list of block information only containing the 49 coefficients of each block. Second, the requesting peer uses the list to get the blocks it does not have from the target peer and the transporting of the blocks occurs at this step. Owing to the two steps, the requesting peer can avoid downloading the owned blocks.

For the request, the indexers will always additionally index the peer for each file that last requested the file, and send the information of both the peer and registered peers to the requesting peer. When requesting a file d , the peer will choose the first indexer to get the peer information of the file. After getting back the peer list, the requesting peer will connect to the last requested peer first. The peer then follows the peer list to collect 7 independent blocks. If the information is still not enough, the peer attempts to look up other indexers. If all indexers can't provide sufficient information, it will try again later until its max lookup time. After collecting the 7 independent blocks successfully, the peer decodes the blocks and put them back to its LRU cache for the next access.

For the mechanism of creating a new coding block, we introduce another type of the indexed peers. It is similar to the above that the indexers always additionally index the peer for each file that last requested the file but not among the registered members. The reason is that we want to index a peer holding the most blocks to become the first choice to let it generate a new coding block by itself, and this can save the maintaining cost. When the indexers decide to create a new coding block, they notify the peer indexed above first and then randomly choose other peers not in the list of the registered members to generate the new coding blocks and the list of the registered members for the file is in the notification for the notified peers. The chosen peers with 7 coding blocks in LRU cache generate the coding block by themselves. If the coding blocks are not enough in LRU cache, they request the last accessed peer first. If that peer has 7 blocks, then the coding block is generated by

the peer. If not, they then choose the registered members, and have two choices which depend on the number of the gathered and useful peers for creating the coding block. If the number is equal or larger than 13, then it follows the Regenerating Code to create a coding block. If unfortunately the number is smaller than 13 then it turns to collect 7 blocks to reconstruct the file and create a new coding block. Finally, if the number is smaller than 7, like the request process, it tries to request the indexers again later to get more information and follow the steps before until time out occurs. The flow chart of the process appears in Figure 3-4 .

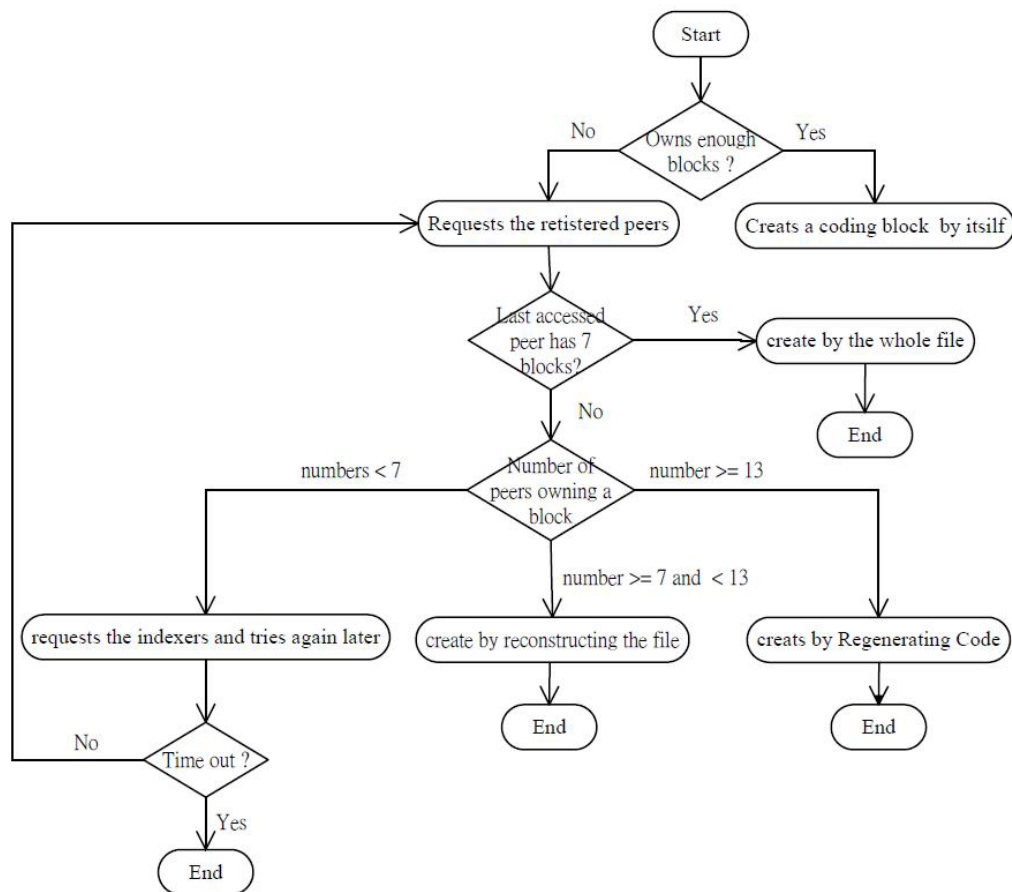


Figure 3-4 The flow char of creating a coding block

3.3. System analysis

Our analyses have two parts. One is about the request and the other is about the generating of a coding block. In the section, there are some figures which show the

analyses of the hybrid scheme (See Figure 3-5 shown below), the Regenerating Code scheme (See Figure 3-6 shown below) and the Regenerating Code scheme with LRU cache scheme (See Figure 3-7 (Case 1) and Figure 3-8 (Case 2) shown below). The setting of (n, k) is $(4, 2)$ below. We analyze the differences among these schemes to explain why we design our scheme. The scenario is that there are four peers, Peer 1 to Peer 4, store individually a coding block of the file. When the Peer 4 crashed, Peer 5 is notified to generate a coding block. In addition, Peer 6 shows how to requests the file in each scheme. We want to explain the difference among these schemes.

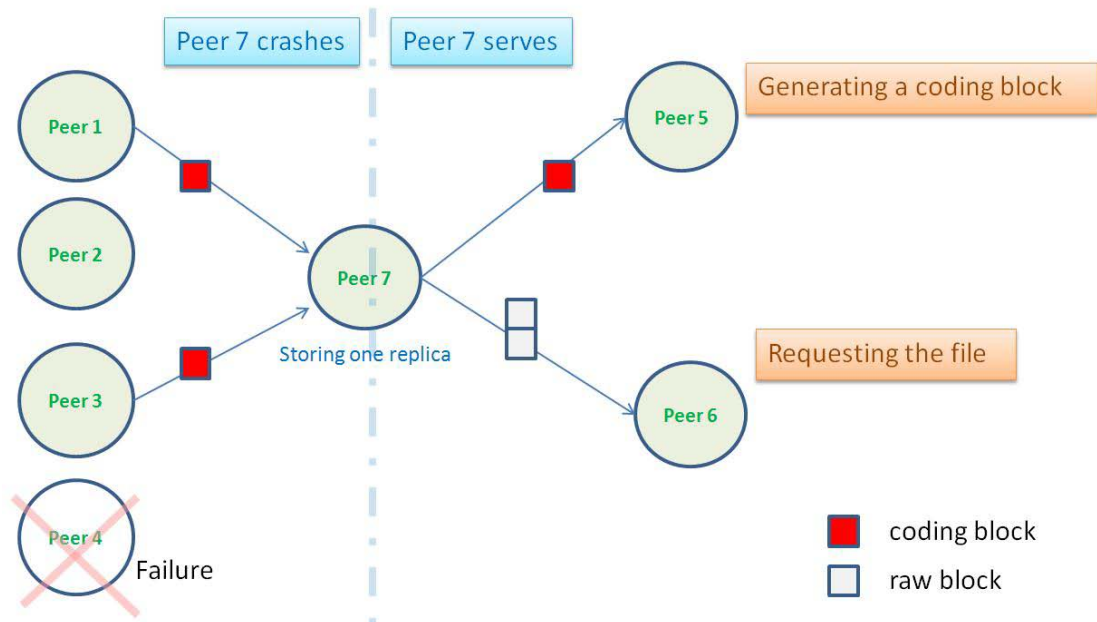


Figure 3-5 Analysis of the hybrid scheme

In the Figure 3-5, we show the analysis of the hybrid scheme. Peer 7 is an important peer storing a replica in the hybrid scheme. When generating a coding block, Peer 7 can directly forward a coding block to Peer 5, and when requesting the file, Peer 7 also can directly forward the whole file to Peer 6. If Peer 7 crashes, Peer 6 still can access the file through communicating with two of the first three peers. But Peer 5 can't get a coding block until the file is reconstructed by some peer. Usually the system can wait to index some peer accessed the file, like Peer 6, later; however, if

the waiting time is so long that some of the first three peers may crash in the period, then the redundancy in the system may be not enough to create a coding block. Therefore, sometimes the system has to maintain a whole replica additionally. The extra overhead is the drawback of the erasure coding scheme. The advantages of the scheme are that the access only requires connecting to one peer and the bandwidth cost of the generating is one coding block, a half of the file, as the replica is present.

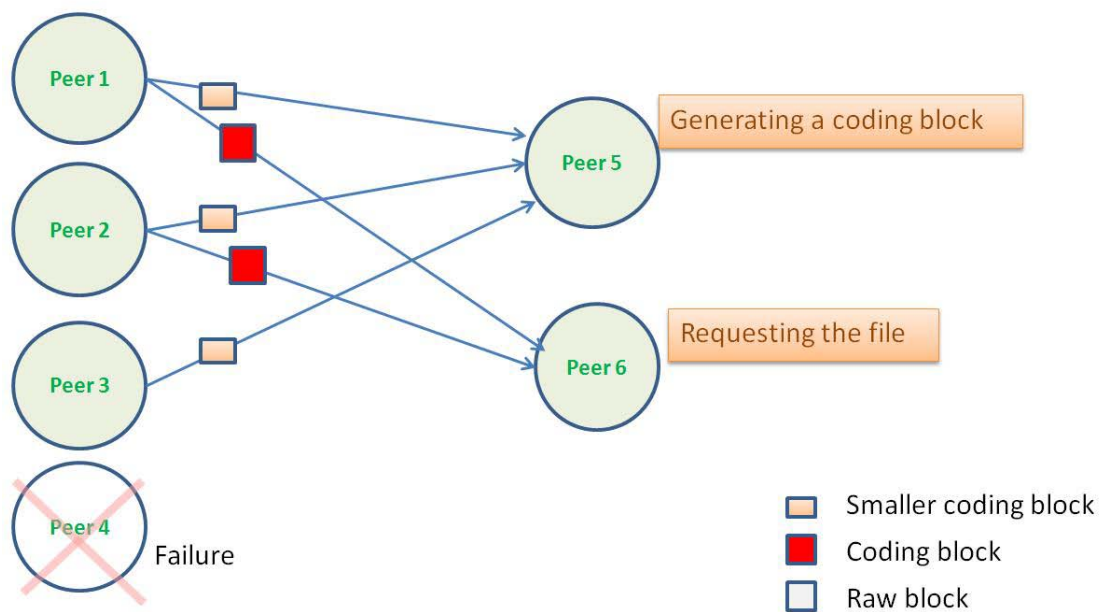


Figure 3-6 Analysis of the Regenerating Code

In the Figure 3-6, it is the analysis of the Regenerating Code. The replica is not maintained in the system. All the requesting and the generating are finished by collecting enough coding blocks. The needed number for the requesting is two, but the number for generating is different. Peer 5 gathers three smaller coding block, a coded packet with the size of a quarter of the file. Consequently, the bandwidth cost is three fourths of the file but the cost is higher than hybrid scheme at ideal case. In the example, the number d in the Regenerating Code is 3. If d is larger, for example, in our experiment the value is 13 and $(n, k) = (14, 7)$, then the bandwidth cost will be close to the hybrid scheme where the value of our setting is $\frac{13}{49}$ of the file and the

value of hybrid scheme is $\frac{1}{7}$ of the file. We can find that the cost is gradually close to the size of the coding block as d is larger, but still larger than hybrid scheme. Nevertheless, that the number d close to n is larger implies that there must be so many peers simultaneously existing in the system, which is a rigorous condition in dynamic environment. The probabilities of at least any d peers of the n peers are alive concurrently in the P2P environment at different peer availability are shown in Table 3-1 . Here the peer availability is independent and identically distributed. The last two rows are the setting in our experiment.

Table 3-1 The successful probabilities of the encoding of the Regenerating Codes

Peer availability	Value of n	Value of d	Probability
0.9	14	13	0.5846
0.65	21	20	0.0003
0.4	42	41	close to 0
0.65	21	13	0.5237
0.4	42	13	0.7589

We consider that LRU cache can reduce the bandwidth in the system without keeping the entire file, so our scheme additionally index two peers for each file, the last accessed peer and the peer which last accessed the file but not among the registered members, to exploit the cache more sufficiently. The two cases that the LRU cache can decrease the cost are shown in Figure 3-7 and Figure 3-8. One is the indexed peers have partial blocks, the other is they have enough blocks. Another point to use the data is that there is linear relationship between the coding blocks and raw blocks. The requested coding blocks in cache will be decoded for access and then are transformed into the raw blocks. If the raw blocks and the coding blocks are

independent, the two types of the blocks also can reconstruct the file. Furthermore, if there are some coding blocks in the LRU cache of the peer generating a coding block, the peer can use these blocks to create the coded packets by itself; therefore, the peer does not need to collect d coded packets.

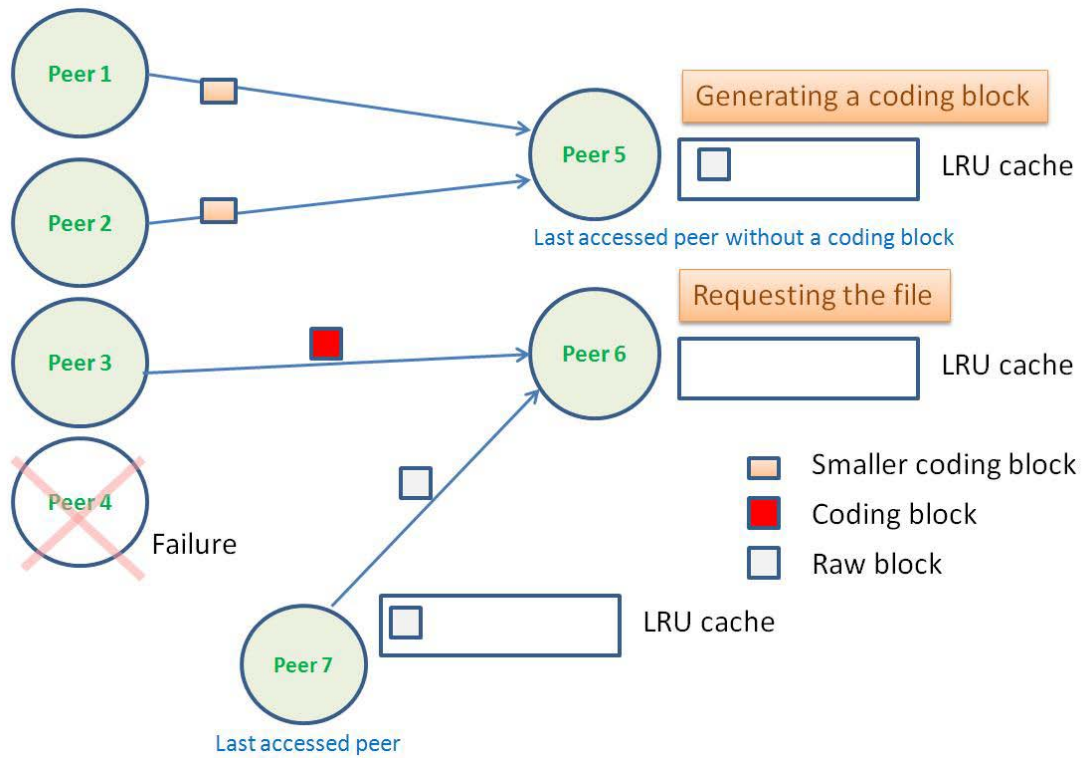


Figure 3-7 Analysis of the Regenerating Code with LRU cache (Case 1)

In the Figure 3-7, the indexed peers do not have enough blocks, this case is one block; as a result, they only have to communicate with fewer peers to finish their tasks. There is still another case like this, which is that the peer last accessed the file holds partial coding blocks but Peer 6 has no blocks. In our (14,7) setting, if it has 4 coding blocks, then Peer 6 only lack 3 coding blocks after just one connection. For the cost, when the connection times become fewer, the bandwidth is less too. Although LRU cache can bring the benefit, there is a problem when the peers use the coding blocks in LRU cache. That is the peer may collect the repeated blocks or dependent blocks which can't support the decoding, so sometimes not all the blocks in cache are helpful. The requesting peer has to check the coefficients of these blocks before

retrieving the blocks.

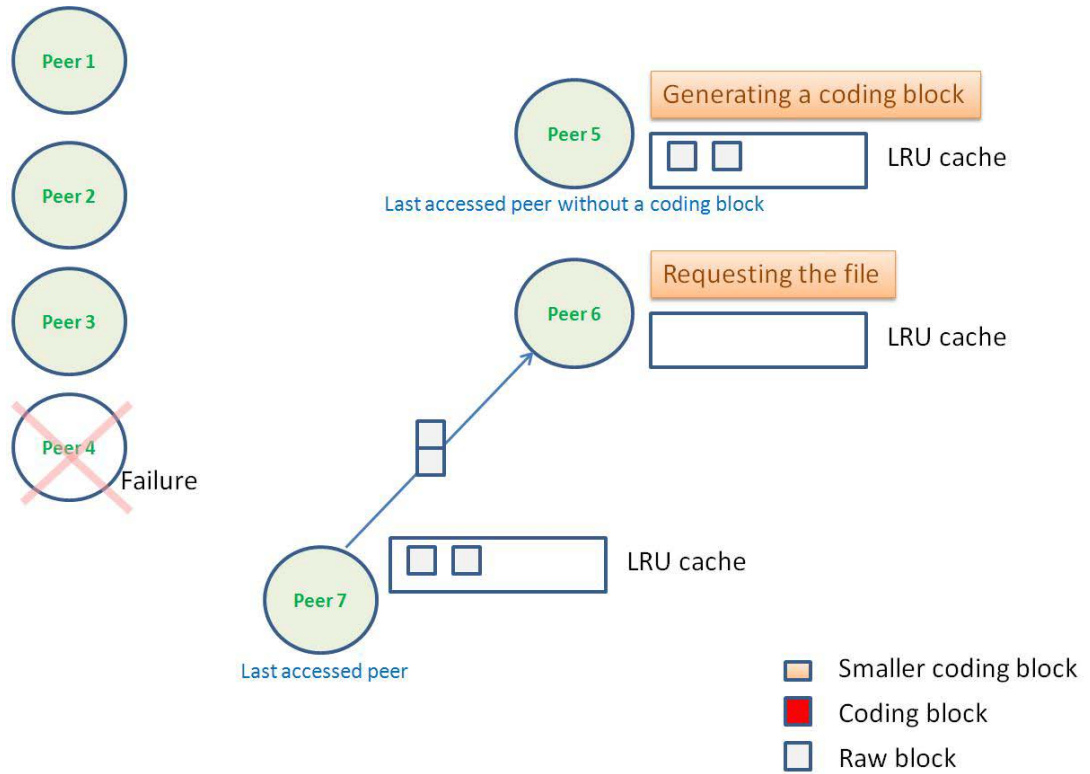


Figure 3-8 Analysis of the Regenerating Code with LRU cache (Case 2)

In the Figure 3-8, the indexed peers have entire file. Fortunately, Peer 5 can generate by itself, and Peer 6, like hybrid scheme, just connect to one peer to read the file.

Briefly, if the indexed peers own the whole file, the benefit is equal to save a replica the system. Even if they only have a partial file, the connection cost still goes down.

4. Experimental Result and Analysis

4.1. Experimental Setup

We attempt to evaluate the performance improved by the additionally indexed peers in P2P storage system. We implement our scheme in P2PSim [21], a discrete event packet level simulator that can simulate structured overlays only. The simulated network comprises 1024 peers, and each peer alternately crashes and rejoins in the network. The interval between successive events for each peer is exponentially distributed with a mean of given time. Each peer has a database, a LRU cache and an index table. When a peer crashes or leaves, all stored data and indexes are cleared. When a peer rejoins, it changes to a different IP and also uses different DHT id. There are 1000 different files with the same size in the system and all files have the same accessed probability. At the beginning, all files are randomly allocated to these peers. These peers will use the files to generate enough coding blocks to those randomly chosen peers.

The main two parameters are the peer availability and cache size. The values of each parameter are shown in Table 4-1 and Table 4-2. The goal of the file availability is 99.9%. We choose the peer availability first and then set different cache size. The experiment time is 6 hours when peer availability is 0.9. In order to let the number of lookup of the peers be 40, we increase the experiment time in the experiments with lower peer availability. We collect the data only during the second half of the experiment time.

Table 4-1 Individual parameters for each peer availability

Parameter	Value		
Peer availability	90%	65%	40%
Time (hours)	6	$\frac{90}{65} * 6$	$\frac{9}{4} * 6$
Data redundancy	2	3	6
Data redundancy (blocks)	14	21	42

Table 4-2 Common parameters for all peer availability

Parameter	Value
Cache size (blocks)	8, 16, 32, 64, 128
Average requested files per peer	40
Average requested blocks per peer	$40 * 7 = 280$
Target availability	99.9%

4.2. Evaluation and Analysis

4.2.1. Experimental results of access

When accessing a file, a peer will check its LRU cache first. If it does not own 7 blocks, it will request the indexed peer to reduce the connection times. If the indexed peer owns 7 blocks in its LRU cache, the accessing peer only requires one connection time to get the whole blocks rather than requires several connection times to collect 7 blocks. We show the access performance, connection times, improved by the indexed peers through three types of the request cases, the peers having 7 blocks, the peers having no block and the peers having partial blocks. In addition to the three types of

the request cases, there are some cases that the accessing peer has 7 blocks in its LRU cache and then does not need to request other peers, called other cases below. The three tables, Table 4-3 , Table 4-4 and Table 4-5 show the percentages of the four types among total request cases for each peer availability. Furthermore, the percentages of the type of the request cases that the indexed peers have 7 blocks for three peer availabilities are shown in Figure 4-1. The values below are the average of 5 trials.

Table 4-3 The access results for peer availability = 0.9

Cache size \ Block number	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	45.76%	28.44%	25.70%	0.10%
16	69.21%	16.29%	14.30%	0.20%
32	86.35%	7.41%	5.85%	0.39%
64	91.51%	4.62%	3.09%	0.77%
128	91.30%	4.55%	2.67%	1.48%

Table 4-4 The access results for peer availability = 0.65

Cache size \ Block number	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	38.72%	30.34%	30.83%	0.10%
16	60.69%	19.89%	19.21%	0.21%
32	80.87%	10.18%	8.58%	0.37%
64	89.06%	5.98%	4.20%	0.76%
128	89.74%	5.34%	3.41%	1.51%

Table 4-5 The access results for peer availability = 0.4

Cache size \ Block number	The indexed peers have 7 blocks	The indexed peers have partial blocks	The indexed peers have no block	Other cases
8	28.00%	37.96%	33.92%	0.13%
16	48.54%	26.47%	24.78%	0.21%
32	68.97%	16.64%	14.01%	0.38%
64	83.51%	9.18%	6.51%	0.81%
128	85.25%	8.10%	5.07%	1.58%

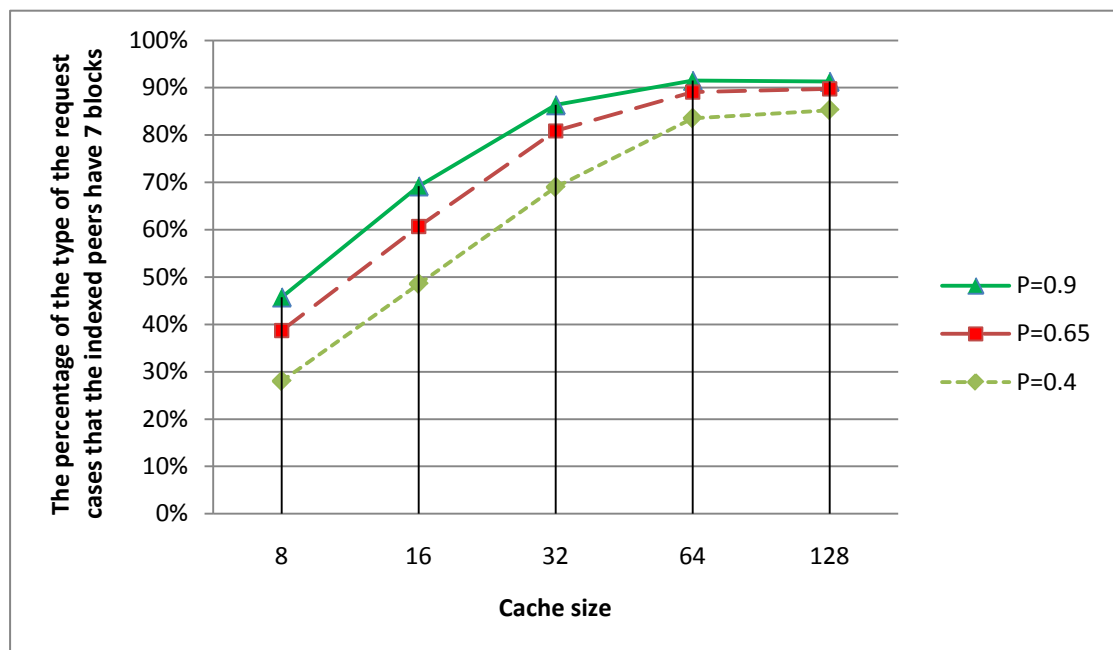


Figure 4-1 The access results of the case the indexed peers have 7 blocks for each peer availability

Through the results above, we can observe that the optimal choice of cache size is 64. The values are at least 83%.

In our experiments, we take advantage of the LRU cache to improve the access performance. Below, we compare the connection times of our scheme with the pure Regenerating Code scheme which only detects the database during the process of

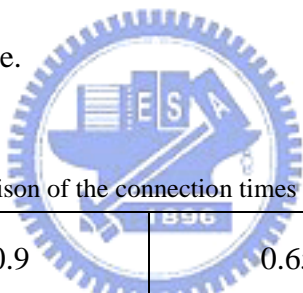
access. For the connection times, we just compare those connections the peer actually gets at least one blocks. We ignore the connections for searching the information of peers and blocks.

In the pure Regenerating Code scheme, if a peer owns a coding block of the file in database, it only requires 6 connection times to collect 7 blocks. If not, a peer then requires 7 connection times. The percentages of the cases of 6 and 7 in each comparison are according to the statistics in our experiments.

In our scheme, the indexed peers can help to reduce the connection times because they may own more than 1 block. If they own 7 blocks, then it is the ideal case that the peer only needs 1 connection.

In the Table 4-6, “Before” means the values of the pure scheme and “After” means the values of our scheme.

Table 4-6 Comparison of the connection times for each availability



peer availability cache size	0.9		0.65		0.4	
	Before	After	Before	After	Before	After
8	6.99	4.22	6.97	4.61	6.92	5.14
16	6.98	2.70	6.97	3.18	6.90	3.85
32	6.99	1.69	6.97	1.97	6.89	2.60
64	6.98	1.44	6.97	1.57	6.88	1.89
128	6.98	1.39	6.97	1.45	6.88	1.70

In the Figure 4-2 , Figure 4-3 and Figure 4-4 , we individually show the comparison of connection times for each peer availability.

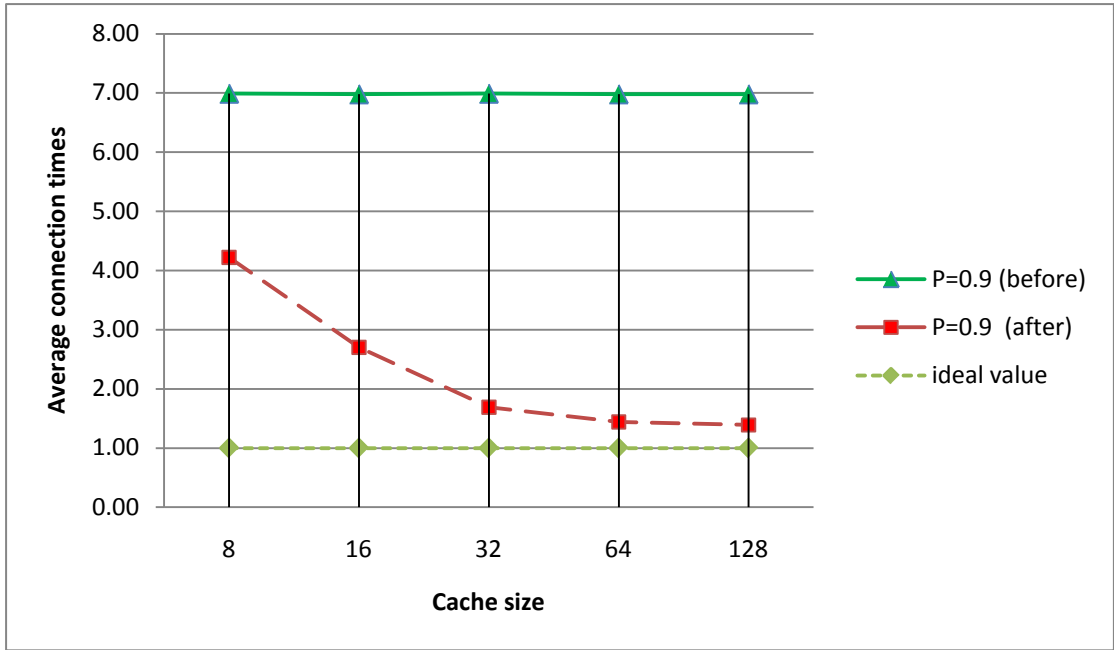


Figure 4-2 Comparison of the connection times for peer availability = 0.9

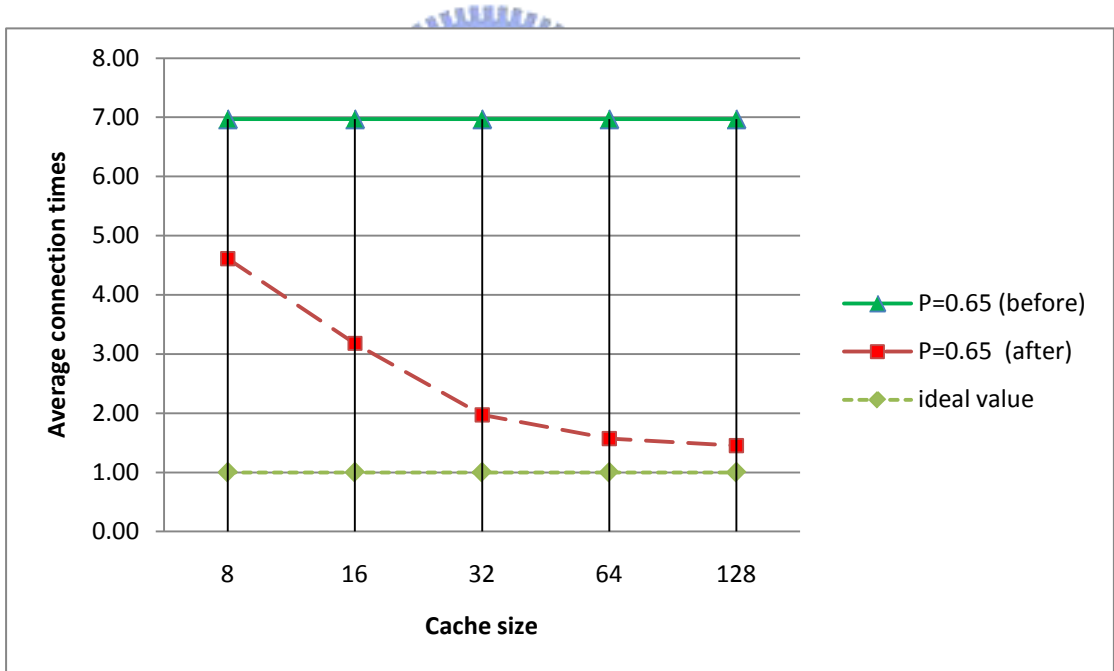


Figure 4-3 Comparison of the connection times for peer availability = 0.65

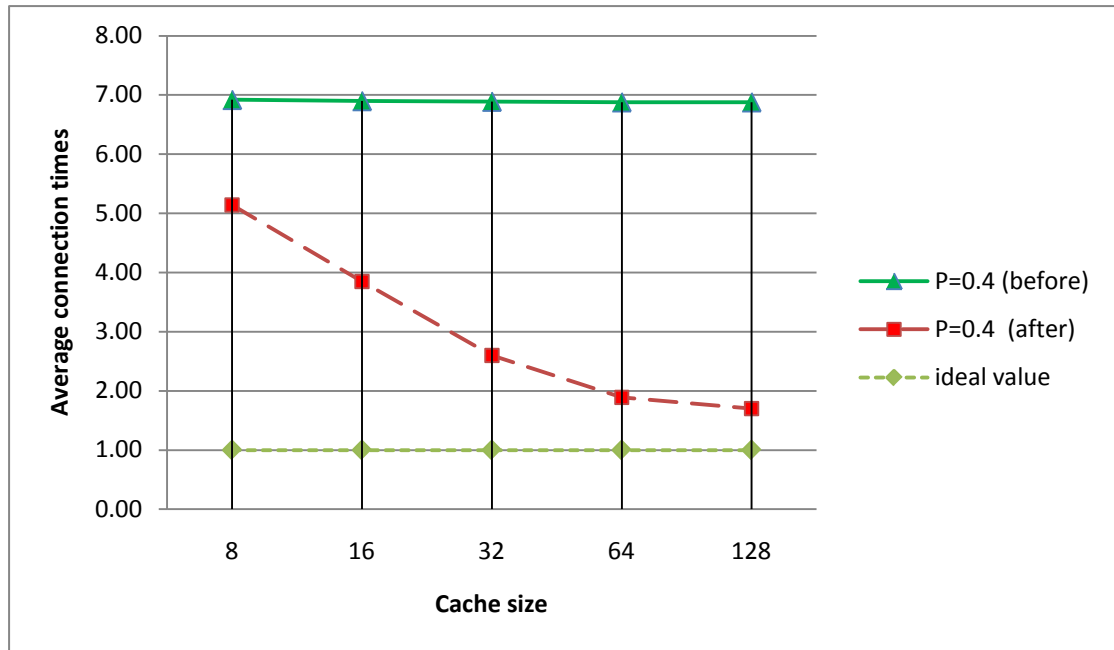


Figure 4-4 Comparison of the connection times for peer availability = 0.4

We can observe the connection times can reduce to less than 2 after the cache size is larger than 64. The value is very close to the ideal case.

4.2.2. Experimental results of generating redundant data

In our scheme, there are four ways to generate a coding block. The main way to improve the cost of maintenance is that the coding blocks are generated by the indexed peers. Besides, there are the three ways to generate a coding block, generating by itself, generating through Regenerating Code and generating by reconstructing the file. The three tables, Table 4-7, Table 4-8, Table 4-9, show the percentages of the four ways among total generating cases. Furthermore, the percentages of the way of generating a coding block by the indexed peers for three peer availabilities are shown in Figure 4-5. Also, all values are the average of 5 trials.

Table 4-7 The results of generating redundancy for peer availability = 0.9

Cache size \ Generating type	By indexed peers	By Regenerating Code	By reconstructing the file	By itself
8	44.95%	52.21%	0.38%	2.47%
16	68.92%	28.12%	0.20%	2.77%
32	86.08%	10.96%	0.07%	2.89%
64	91.59%	5.15%	0.04%	3.22%
128	91.58%	4.41%	0.03%	3.98%

Table 4-8 The results of generating redundancy for peer availability = 0.65

Cache size \ Generating type	By indexed peers	By Regenerating Code	By reconstructing the file	By itself
8	35.99%	57.88%	0.01%	6.12%
16	58.03%	35.30%	0.01%	6.66%
32	78.57%	14.81%	0.00%	6.61%
64	86.67%	6.52%	0.00%	6.81%
128	87.58%	5.08%	0.00%	7.34%

Table 4-9 The results of generating redundancy for peer availability = 0.4

Cache size \ Generating type	By indexed peers	By Regenerating Code	By reconstructing the file	By itself
8	27.57%	67.38%	0.01%	5.04%
16	50.50%	45.49%	0.00%	4.01%
32	72.18%	24.74%	0.00%	3.08%
64	87.85%	9.65%	0.00%	2.50%
128	90.16%	6.90%	0.00%	2.93%

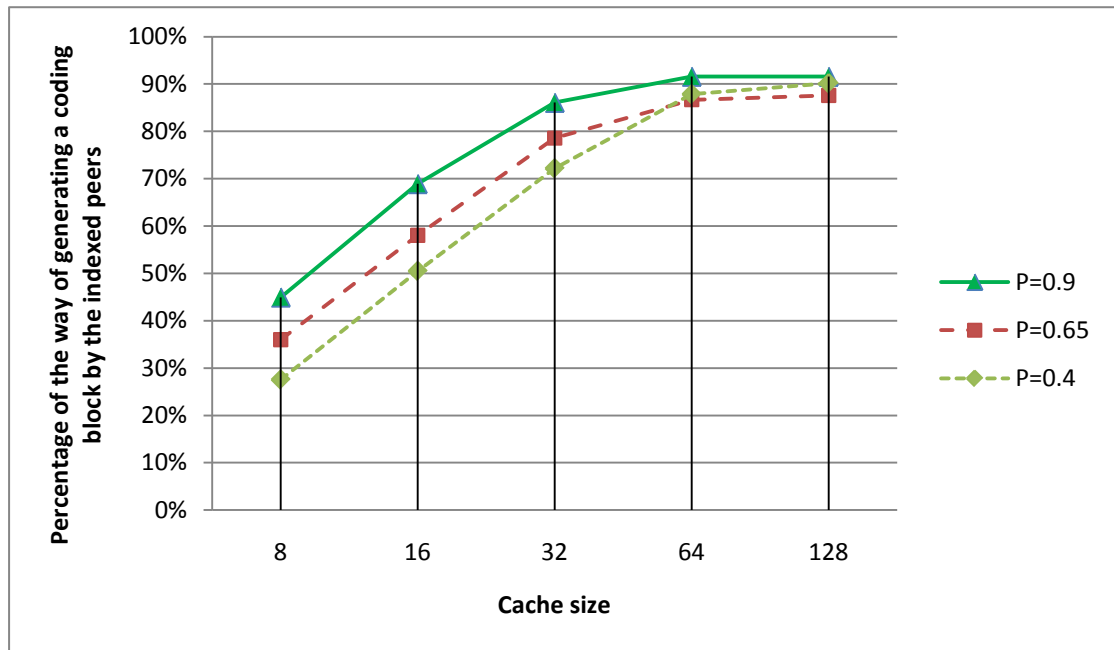


Figure 4-5 The results of generating redundancy by the indexed peers for each peer availability

The values in Figure 4-5 are the values of the first column of the three tables above. We can find that the optimal cache size is 64 where the value is at least 86% and the values of the case that peer availability is 0.65 are lower than the case that peer availability is 0.4 as cache size is larger than 64. The reason is that the percentage of another type of the cases of generating redundant data that the data is

generated by the peer itself is higher as peer availability is 0.65.

The target of the comparison of generating redundancy is the same as the comparison of access. Assume the file size is M and then the block size is $\frac{1}{7}M$. That the coding block is generated by another peer owning a whole file is the ideal case whose bandwidth cost is $\frac{1}{7}M$. We analyze the pure scheme first. It is according to the theoretical value of the Regenerating Code in our setting. The value is $\frac{13}{49}M$.

In our scheme, there are four types of cases:

- The coding block is generated by itself. The cost is 0.
- The coding block is generated by the indexed peer. The cost is $\frac{1}{7}M$.
- The coding block is generated by the process of Regenerating Code. The cost is $\frac{13}{49}M$.
- The coding block is generated by the process of reconstructing the file. The cost is M .



We calculate the average cost as below.

$$P_1 * 0 + P_2 * \frac{1}{7}M + P_3 * \frac{13}{49}M + P_4 * M \quad (11)$$

where P_1 , P_2 , P_3 and P_4 are the percentages of each cases in our experiments.

In the Table 4-10, “Before” means the values of the pure scheme and “After” means the values of our scheme. The unit is M and ideal value is $\frac{1}{7}$ which is close to 0.143.

Table 4-10 Comparison of the cost of maintenance for each peer availability

peer availability cache size	0.9		0.65		0.4	
	Before	After	Before	After	Before	After
8	0.265	0.207	0.265	0.205	0.265	0.218
16	0.265	0.175	0.265	0.177	0.265	0.193
32	0.265	0.153	0.265	0.152	0.265	0.169
64	0.265	0.145	0.265	0.141	0.265	0.151
128	0.265	0.143	0.265	0.139	0.265	0.147

In Figure 4-6 , Figure 4-7 and Figure 4-8 , we individually show the comparison of the average cost of maintenance for each peer availability.

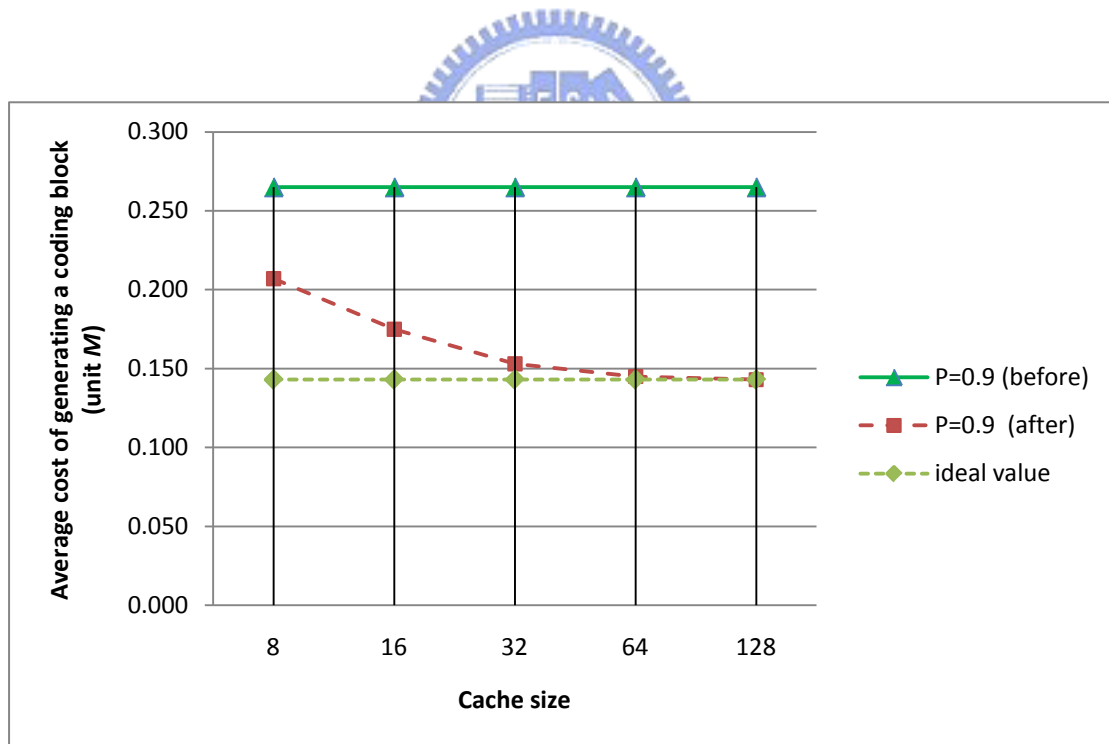


Figure 4-6 Comparison of the average cost of generating a coding block for peer availability = 0.9

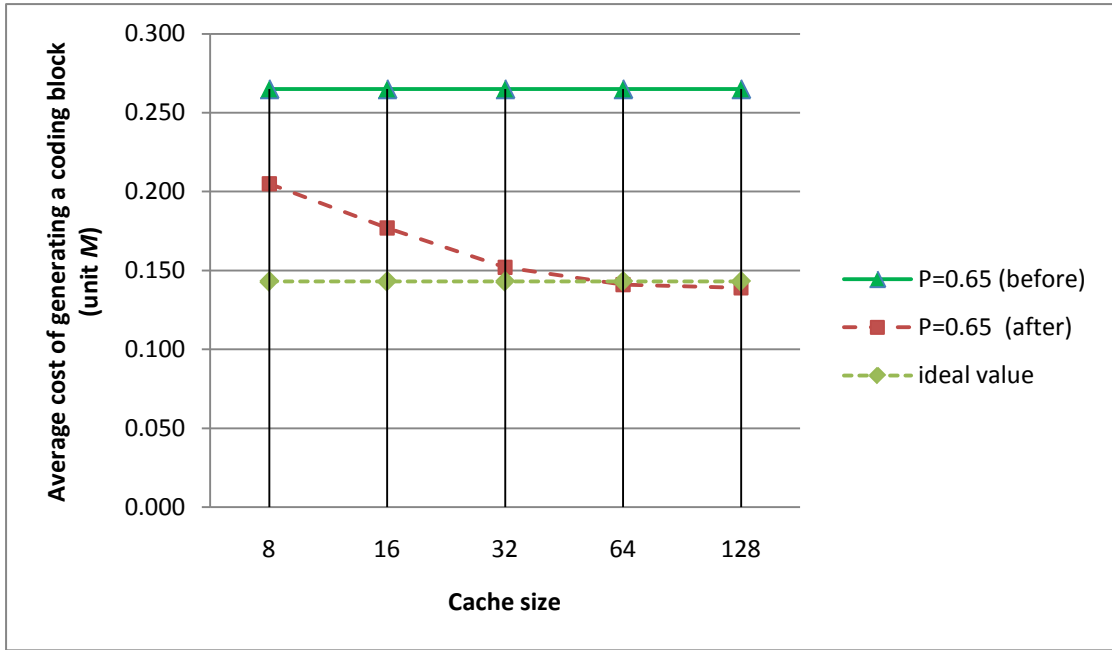


Figure 4-7 Comparison of the average cost of generating a coding block for peer availability = 0.65

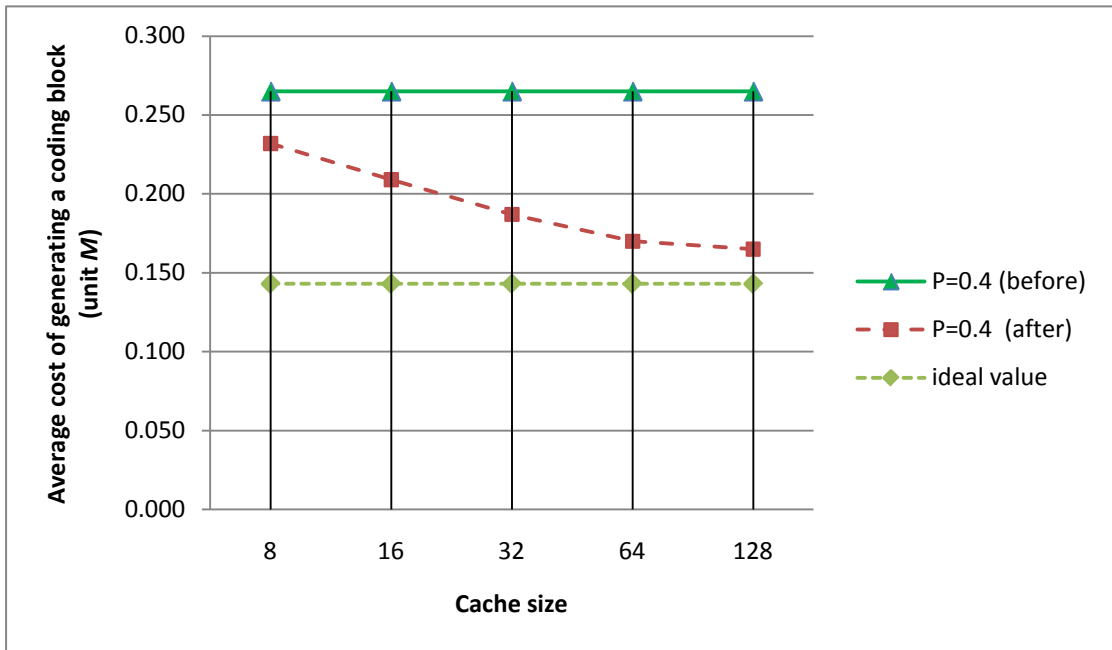


Figure 4-8 Comparison of the average cost of generating a coding block for peer availability = 0.4

5. Discussion

In the chapters above, we had discussed the design of P2P storage system. Here we discuss how the design is mapped on to the content distributed components in mCDN [33] . There are four related components for content service, CDN origin servers called CDN_Origin for short, CDN edge servers called CDN_EDS for short, optimal content placement called OCP for short and the content provider service called CPS for short, and a related architecture, Peer-to-Peer Search Middleware called P2P_SMW for short, in the mCDN architecture. In the Table 5-1 we list the description of these components.

Table 5-1 The description of the content distributed components in mCDN

Component	Description
CPS	CPS triggers the CDN_Origin to take the appropriate actions for the distribution of the content entity
CDN_Origin	CDN_Origin fetches the content entity together with the corresponding metadata from the CPS and distributes it according to the schedule
CDN_EDS	CDN_EDS stores the content entity
OCP	OCP supports the evaluation of the optimal placement of new content and the evaluation of the location of the “closest” content in the content retrieval scenario
P2P_SMW	P2P_SMW is a supporting P2P infrastructure used by the CDN_EDS and OCP components during the content location discovery

The scenario is that the content is scheduled for distribution from the content origin server (CDN_Origin) to the edge servers (EDSs). First, a message is broadcasted announcing the content transmission to the edge servers (OCP components). The OCP component in the EDS decides whether to store the content, when it broadcasts, on the basis of different parameters (e.g., number of replica in the CDN) and context conditions (e.g., high number of requests from clients).

The three major components related to P2P storage system are CDN_Origin, CDN_EDS and OCP. These components are responsible for evaluating optimal content locations, distributing the content to corresponding edge servers and discovering optimal content locations matching user requests.

Following the description above, it is clear to map the components of mCDN onto those of P2P storage system. The difference is that the indexers in mCDN are responsible for more tasks, such as the evaluation of the location, as OCP does.

Table 5-2 The mapping results

Content distributed components in mCDN	Components of the P2P storage system
CDN_Origin	The peers own the files
OCP	The indexers
CDN_EDS	The peers for storing the files

6. Conclusion and Future Work

6.1. Conclusion

In the thesis, we extra index two types of the peer information, one is the information of the peer last accessed a file and the other is the information of the peer last accessed a file but not among the registered members, to improve the access performance and to reduce the maintenance cost. The experiment results indicate that there are two factors showing dramatic improvement, peer availability and cache size. In addition, although the influence of the second type of the peer information is low, the bandwidth cost in the case is zero. Finally, the first type of the peer information successfully achieves our goal, and the optimal choice of cache size is 64.

On the other hand, based on our research, we consider that it is possible that P2P storage systems do not need to additionally maintain a full file. Although the authors of Regenerating Code want to use the Regenerating Code to achieve the goal, the access performance of the scheme is not acceptable for content service. In our research, we overcome the drawback and then the goal is possible.

6.2. Future Work

For the theoretical improvement, in our scheme, the improvement of the indexed peers last accessed but not among the registered members is not clear, within 7%. We will abandon it and attempt to index the two peers last accessed a file to reduce the requirement of the cache size and to make more improvement.

For the application of P2P storage systems, our practical future work will focus on how to follow the mCDN architecture to implement a P2P storage system. The

work will implement P2P storage systems and some APIs of mCDN.



7. References

- [1] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE TRANSACTIONS ON INFORMATION THEORY*, VOL. 52, NO. 10, OCTOBER 2006
- [2] M. Médard, R. Koetter, "An Algebraic Approach to Network Coding", *IEEE/ACM TRANSACTIONS ON NETWORKING*, VOL. 11, NO. 5, OCTOBER 2003
- [3] S.-Y. R. Li, R. W. Yeung and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*," IT-49: 371-381, 2003.
- [4] C. Fragouli, J.-Y. L. Boudec, and J. Widmer, "Network Coding: An Instant Primer,"
- [5] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1204–1216, 2000.
- [6] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "On the benefits of random linear coding for unicast applications in disruption tolerant networks," *Second Workshop on Network Coding, Theory, and Applications (NETCOD)*, 2006
- [7] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. IT-51, pp. 1973–1982, 2005..

- [8] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin O. Wainwright and Kannan Ramchandran “Network coding for distributed storage systems” arXiv:0803.0632v1 [cs.NI] 5 Mar 2008
- [9] P. Sander, S. Egner, and L. Tolhuizen, “Polynomial time algorithms for network information flow,” in Symposium on Parallel Algorithms and Architectures (SPAA), (San Diego, CA), pp. 286–294, ACM, June 2003
- [10] Lin W, Chiu D, Lee Y. “Erasure code replication revisited” In: Proc. of the 4th Int’l Conf. on Peer-to-Peer Computing. 2004. 90-97.
- [11] C. Fragouli and E. Soljanin, “Network Coding Fundamentals” now Publishers, 2007.
- [12] C. Fragouli and E. Soljanin, “Network Coding Applications” now Publishers, 2008.
- [13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. “A scalable content addressable network” In Proc. ACM SIGCOMM, 2001.
- [14] A. Rowstron and P. Druschel. “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems” In Proc. Middleware, 2001.
- [15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. “Chord: A scalable peer-to-peer lookup service for internet applications” In Proc. ACM SIGCOMM, 2001.
- [16] Sit, Emil, “Storing and managing data in a distributed hash table” 2008.
- [17] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes" Communications Surveys & Tutorials, IEEE, vol. 7, pp. 72-93, 2005.

- [18] P. R. C. Gkantsidis, J. Miller, “Anatomy of a P2P content distribution system with network coding,” Proceedings of IPTPS, 2006.
- [19] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. “Wide-area cooperative storage with CFS” In Proc. ACM SOSP, 2001.
- [20] F. Dabek, J. Li, E. Sit, J. Robertson, F. Kaashoek, and R. Morris. “Designing a DHT for low latency and high throughput” In Proc. NSDI, 2004.
- [21] T. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling. p2psim: A simulator for peer-to-peer protocols. <http://www.pdos.lcs.mit.edu/p2psim/>
- [22] Hongxing LI, Guihai Chen “Data Persistence in Structured P2P Networks with Redundancy Schemes” The Sixth International Conference on Grid and Cooperative Computing(GCC 2007)
- [23] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker. “Total recall: System support for automated availability management” In Proc. NSDI, 2004.
- [24] C. Blake and R. Rodrigues. “High availability, scalable storage, dynamic peer networks: Pick two” In Proc. HotOS, 2003.
- [25] Rodrigo Rodrigues and Barbara Liskov. “High availability in DHTs: Erasure coding vs. replication” In Proc. IPTPS, 2005.
- [26] H. Weatherspoon and J. Kubiatowicz. “Erasure coding vs. replication: A quantitative comparison” In Proc. IPTPS, 2002.
- [27] Fan Wu, Tongqing Qiu, Yuequan Chen and Guihai Chen. “Redundancy schemes for high availability in DHTs” In Proceedings of ISPA, 2005. Journal version will appear in Journal of Supercomputing, 2007..

- [28] John Kubiawicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, HakimWeatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. “OceanStore: An architecture for global-scale persistent storage” In Proceedings of ACM ASPLOS. ACM, November 2000.
- [29] Guangping Xu, Gang Wang, Jing Liu. “A hybrid redundancy approach for data availability in structured P2P network systems” in Proc. of the 13th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'07) , Melbourne, Victoria, AUSTRALIA, December, 2007.
- [30] C. Gkantsidis and P. Rodriguez, “Network coding for large scale content distribution” Proceedings of IEEE Infocom, 2005.
- [31] A. Rowstron and P. Druschel. “Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility” In Proc. SOSP, 2001.
- [32] Tian J, Dai Y F. “Study on durable Peer-to-Peer storage techniques” J Softw (in Chinese), 2007, 18(6): 1379—1399
- [33] Sokol Joachim, Eckert Klaus-Peter : “MCDN: Multimedia Content Discovery and Delivery” , 2007. dans Eighth International Symposium on Autonomous Decentralized Systems (ISADS '07), 21-23 Mars 2007, pages : 411 – 420
- [34] M. Merzbacher and D. Patterson. “Measuring end-user availability on the web: Practical experience” In Proc. IPDS, 2002.
- [35] Min Yang Yuanyuan Yang, “Peer-to-peer File Sharing Based on Network Coding. Distributed Computing Systems” 2008. ICDCS '08. The 28th International Conference

- [36] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed fountain codes for networked storage," *Acoustics, Speech and Signal Processing, ICASSP 2006*, May 2006.
- [37] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," *IEEE/ACM Transactions on Networking (TON)*, June 2006.
- [38] A. Kamra and V. Misra, "Growth Codes: Maximizing Sensor Network Data Persistence," in *Sigcomm06*, Pisa, Italy, 2006. [52] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks
- [39] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *Proceedings of the 26th IEEE INFOCOM 2007*, Anchorage, Alaska, May 6–12 2007.
- [40] S. Reed and G. Solomon. "Polynomial codes over certain finite fields" In *J. SIAM*, 1960.
- [41] Bhagwan, R., Savage, S., Voelker, G.: "Understanding availability" In: *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, California (2003)