SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2016, Vol. 59 012103:1–012103:15
doi: 10.1007/s11432-015-5299-5

BufferBank storage: an economic, scalable and universally usable in-network storage model for streaming data applications

Hongyi CHEN*, Zhigang SUN, Fei YI & Jinshu SU

School of Computer, National University of Defense Technology, Changsha 410000*, China*

Received October 17, 2014; accepted January 29, 2015; published online July 16, 2015

Abstract Large-scale streaming media distribution services impart unprecedented pressures and challenges to existing Internet architecture. Researchers have proposed CDN, P2P Cache, ICN and other solutions to alleviate the pressure of the core network, as well as improve the user experience (QoE). All these solutions achieve their goals by deploying storage resources close to the end users to cache the hot data. Based on the advantages of P2P service, which takes into account end-users resources, we proposed the BufferBank Storage (BBS). It is a new streaming media distribution-oriented storage model by aggregating end-users resources. This provides a novel approach for implementation of economic, scalable, dynamically deploy streaming media distribution applications. However, the dynamic character of the end-user behavior brings challenges to BBS designation. In our previous work, we have analyzed the basic principle of BBS and its feasibility. There is lack of substantial research on resources management and reliability assessment, which are the core issue of BBS implementation. After carefully analyzing the reliability and security issue in BBS deployment, this paper has proposed the implementation model of BBS and studied the performance of different buffer allocation mechanism through simulation. Our work mainly provides a new way of thinking for the dynamic, universal scalable storage system in the Internet, that suffers "weak reliability".

Keywords in-network storage, P2P Cache, streaming media, distributed resources management

Citation Chen H Y, Sun Z G, Yi F, et al. BufferBank storage: an economic, scalable and universally usable in-network storage model for streaming data applications. Sci China Inf Sci, 2016, 59(1): 012103, doi: 10.1007/s11432-015-5299-5

1 Introduction

In-network storage [1] is the primary method to accelerate the large-scale data distribution in the Internet, particularly for the streaming media distribution. Its main principle is to deploy the network storage in the edge network, caching the data which is frequently accessed to satisfy user requests in the edge network, reducing the core network bandwidth, decreasing the access latency and improving the user experience. And it has become a critical method for enhancing the efficiency of streaming media distribution, which is widely employed in the Content Delivery Network (CDN) [2], P2P Cache [3] and web cache. Information-Centric Network (ICN) [4], which is widely studied, embeds the storage into routing and switching, to

* Corresponding author (email: chychenhongyi@gmail.com)

Table 1 The attributes of different in-network storage systems

	Storage from ICP or ISP	Storage from end-user
Application-dependent	P2P Cache [3]	P2P
Application-independent	CDN, NDN [5]	Our work

maximize the efficiency of content distribution. Compared with the storage in cloud/data center and enterprise network, the In-network storage has following features:

• Lower reliability. Considering that the In-network storage is mainly the content cache, wherein data can be derived from the server if it is lost and will not incur fatal errors.

• Lower security. The content which is stored in In-network storage is mainly streaming media data. Even if these data is tampered or eavesdropped, it will not cause irreparable damage. While when the data is related to scientific computing and financial transaction, the tampered data would lead to a failure.

• Stronger adaptability to transmission flutter. Transmission flutter contains the flutter of data access bandwidth and latency. The cache-play mode of streaming media applications determines the strong adaptability to the transmission flutter. The clients of a media flow, with 2 Mbps bandwidth, only require 16 MB when cache 60 s media data. Hence, it can effectively facilitate the access to the storage network bandwidth and latency flutter.

With the features of low reliability and security in In-network storage, P2P system has achieved great success in the past decade. From the storage perspective, each peer in P2P system caches data on end-users' computers. The P2P applications query the indexing tables of peers to locate the position of cached data. Unfortunately, the distribution scheme which P2P adopts makes it quite difficult to query on combination with the physical location and reduces the querying efficiency. Further, P2P system lacks an efficient incentive model among network operators, content providers and end users, which hinders with its further development substantially.

Table 1 classified current In-network storage technology in two dimensions. One is storage source, and the other is application-dependency. On the dimension of storage source, if applications use ISP/ICP's storage, they can enjoy a reliable service however it is expensive. Using the End-user storage resources can save operation costs while they are not very dependable and trustworthy, such as P2P. With the other dimension, Application-dependent mode can optimize data model for specified applications and achieve high efficiency, but each application uses these storage resources separately. For instance, each P2P system controls mass storage resource, while using it independently. On the other hand, application-independent mode aggregates resources as a common infrastructure, which provides In-network storage services to different applications. Recent research shows that application-independent In-network storage will be prevalent, such as NDN which will embed In-network storage into the basic switching and routing functions of the network [5]. Actually, IETF has already setup Peer-to-Peer Streaming Protocol (PPSP) [6] and Content Delivery Networks Interconnection workshops to push P2P and CDN to become a common service for all applications.

After putting all popular In-network storage solutions in the right place in Table 1, it is surprisingly found that there are no application-independent solutions based on volunteers resources. Thus, our work can draw P2P's advantage and aggregate end-users resources to provide common storage services for streaming media applications.

Therefore, we proposed the BufferBank Storage (BBS). It aggregates storage resources in edge network to provide scalable storage services for streaming media distribution applications through effective storage management mechanisms. Similar to the banking transactions, the end-user who provides storage resources is like a depositor in the real bank system, so we call it BufferBank Depositor (BBD). BufferBank Manager (BBM) is like the bank, and streaming media application, which is called BufferBank Application (BBA) in BBS, is like an enterprise that asks for a loan from the bank.

In BBS, BBM provides profit to incite BBDs to deposit more resources, just like deposit allowance in the bank system. Meanwhile BBM charges BBA for the storage service, resembling interest on loan in the

bank system. After performing a survey on the idle resources in the campus network [7], we found that most hosts have idle rescouses when they are in normal work(such as processing a document). End-users have the ability to contribute most of the idle resources to BBS and obtain benefits. If there are 2000 users in the edge network, every one contributes 512 MB storage space; the total capacity could reach 1 TB. The article [8] mentioned that no proper cache size is given, typically considered in simulation are much smaller (2 GB [9]). They select cache sizes of 10 GB as a realistic case study. While BBS's born scalabel architecture could acheive greater capacity easily. After reorganization by BBM, these In-network storage can provide low-cost local storage service for different streaming media applications. The advantages of this storage service are: (1) little infrastructure investment, involving only the software installed on end-users and BBM; (2) the more end-users, prompting the larger storage capacity; (3) the performance of RAM-based system is higher than disk-oriented storage system.

In order to obtain a high performance, we collect RAM resources from distributed end-nodes and provide a unified access to customers (BBA), which has been studied in Data Center such as RAMCloud [10]. However, unlike the centralized managed servers in Data Center, the behavior of hosts in the Internet is uncontrollable. So BBS faces more challenges which are absent in Data Center, including:

• The dynamic management of storage resources. As the behavior of each end-user is uncontrollable, we must consider the user opting off-line due to sudden problems. The sudden off-line will result in data loss if the streaming media data is stored in the lost node. Therefore, the redundant and dynamic data migration must be designed.

• The reliability assessment of storage services. Although data loss will not bring fatal errors to streaming media applications, as a storage service provider, BBS must possess the storage reliability and provide distinguished reliability service for BBA according to the QoS.

• The scalable resources management. The number of BBDs changes dynamically in some networks such as the campus network, where end-users are relatively centralized. Sometimes it has large number of end-users available for the BBS system, while other times just a few. Moreover, each BBD has different properties, such as different sizes, different reliability and different contribute timings. Hence, it is a challenge to manage these resources effectively.

• End-users' stimulation and evaluation. Similar to P2P system, stimulation and evaluation in BBD is a critical issue in BBS. However, unlike P2P, BBM breaks the tight coupling between resource provider (BBD) and resource consumer (BBA). Therefore it has greater design space on BBD stimulation and evaluation in BBS.

We analyze the feasibility of BBS in [7], including statistics on remaining resources of end-users by preliminary comparison of the performance between BBS and the local storage service. In [11], we studied the BBD evaluation methods, and proposed a credit scoring model based on genetic algorithm. However, the first three challenges have not been studied.

To the best of our knowledge, BBS is currently the only mechanism that considers aggregating the end-users' distribution and unreliable storage resources to provide In-network storage services for various streaming media distribution applications. This paper described the core issue in BBS which contains dynamic management, reliability assessment, and storage management. The main contribution is as follows.

(1) We analyzed the principle of BBS and described the reliability, security and scalability issue in its deployment. We also designed a probabilistic-reliability storage system in the Internet against the feature of weak reliability. It provides novel approach for implementing a dynamically deployed and scalable local cache for streaming media applications.

(2) We showed a formal definition of distributed buffer management issue in BBS. Incidentally pointed out that the core issue of buffer management would be the various buffer allocation strategies through a simple example, and described buffer allocation algorithm in detail.

(3) We designed a BBS simulator named BB-SIM, and analyzed the performance of these three buffer allocation mechanisms. The results confirmed it having different effects.

The rest of this paper is organized as follows. Section 2 introduces the basic idea of our BufferBank storage model and describes some issues in BBS. Section 3 gives the distributed buffer management model

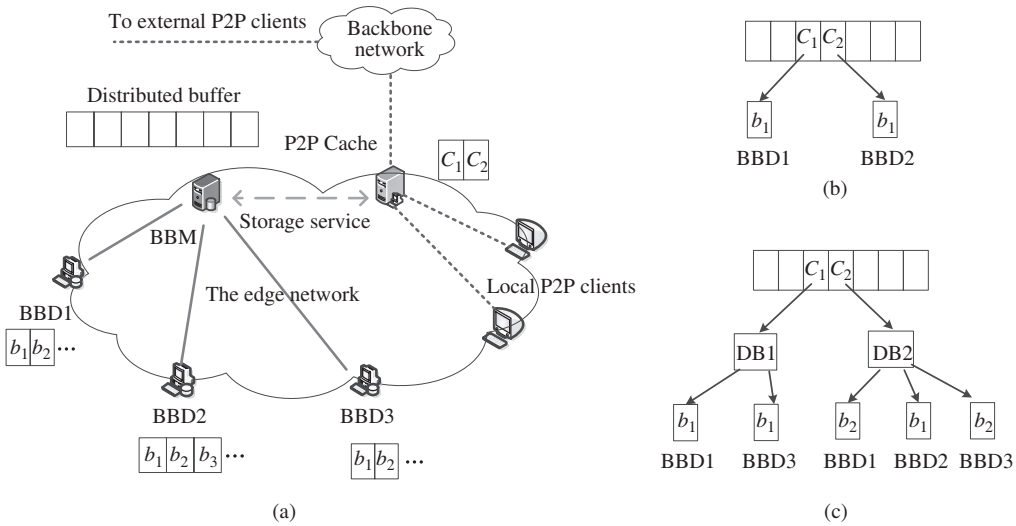

Figure 1 The deployment of BBS. In (a), the doted lines represent P2P systems, and the solid lines represent BufferBank system. P2P Cache is using BBS storage service. (b) Describes a simple resource mapping method. (c) Describes a resource mapping method that consider storage reliability, and now it is deployed in BBS.

in BBS, including buffer allocation algorithm and some organization mechanisms. Section 4 presents the simulation results and analyzes the performance of three organization mechanisms. Finally we have discussed some open issues in BBS and drawn a conclusion last.

2 Overview of BBS architecture

2.1 Basic idea of BBS

A BBS system contains one BBM and lots of BBDs, whose basic principle is shown in Figure 1. Suppose that in this scene, BBM manages 7 buffers contributed by 3 BBDs. The buffers are BBD1(b_1,b_2), BBD2(b_1,b_2,b_3) and BBD3(b_1,b_2). As an application in BBS, P2P Cache applies for two buffers (C_1, C_2), which are used to cache the local P2P traffic. A simple allocation method is shown in Figure 1(b). BBM maps C_1 to b_1 in BBD1, C_2 is mapped to b_2 of BBD2. When P2P Cache wants to read and write data in C_1, BBM will redirect the request to BBD1.

From the above example, we can see the advantages of BBS. On one hand, BBA, which is the P2P Cache in the previous example, can benefit BBS storage service without deploying its own local disk array and other hardware facilities. It reduces costs and simplifies device management. Meanwhile, BBA's existing code needs not change much, as BBS can provide an API similar to file access interface. On the other hand, BBD can obtain certain benefits according to contributed buffer size and number. At the same time, BBS can charge the BBAs, according to its usage, such as the time and size of buffers it demands. Therefore, the above scenario meets all requirements of a clear economic model.

However, BBS must resolve storage reliability problem in order to be practical. For example, P2P Cache's data C_1 is stored in BBD1(b_1). Due to the complex Internet environment and BBS also is unable to control the behavior of BBD1; it is possible that BBD1 shuts down the computer without notifying the BBM. This will lead to data C_1 loss. Therefore, it is imperative to solve the storage reliability problem in BBS.

2.2 Data reliability issue in BBS

Increasing the storage redundancy is the main way to improve the reliability of BBS. For example, to improve the reliability of C_1 and C_2, BBM distributed these data multi-replicas. C_1 was stored both in BBD1(b_1) and BBD3(b_1), and C_2 was stored in BBD1(b_2), BBD2(b_1) and BBD3(b_2). Obviously, the redundancy of C_1 was $1:2$, and C_2 was $1:3$. The reliability of C_2 was better than C_1, the data of C_2

would be lost only when all of BBD1, BBD2 and BBD3 were disconnected. But for C_1, the data would be lost just when both BBD1 and BBD3 were disconnected. Suppose the disconnected possibility of BBDi was P_i, and the data reliability of C_i in BBA was R_i. Then $R_1 = 1 - P_1$, $R_2 = 1 - P_2$ in Figure 1(b), and $R_1 = 1 - P_1 \times P_3$, $R_2 = 1 - P_1 \times P_2 \times P_3$ in Figure 1(c).

In Figure 1(c), especially for C_2 the improvement of reliability in storage leads to lower the cost of inefficient use of buffer. Another way to improve the utilization of buffer was dynamic data migration. An example is the C_1 that was reserved in BBD1 and BBD3 as shown in Figure 1(c). If BBD1 was disconnected, the data would not be lost, but it would decrease the storage reliability to $1 - P_3$. Then we could redistribute BBD2(b_2) for C_1 and copy the data in BBD3(b_1) to BBD2(b_2). The result was that R_2 increased to $1 - P_2 \times P_3$. The method of dynamic data migration can improve the buffer efficiency, but the cost in BBM management and network bandwidth will increase simultaneously.

As the loss of data stored in grid will not bring a fatal error to streaming media applications, thus data replacement is necessary in streaming media buffering, such that storing with BBS, BBA can send a reliability requirement to BBS. The higher the reliability required, the greater is the cost to realize. In fact the higher BBA will pay for the cost. Hence, BBS can provide a buffer redundancy backup strategy with a certain probability according to different requirements from BBA.

2.3 Data security issue in BBS

As BBS stores data in user host which cannot be controlled, we should pay special attention to safety issues. From the view of BBD, the security problem is the potential risk of local agent programs, which will control local noncontributing resource, or invade the local system illegally more rampantly. However, we are not discussing it in BBS, as any third-party software installed in local systems, for example the P2P, will cause such a problem [12].

3 Distributed buffer management model

3.1 Buffer definition

Buffer is the basic unit of BBD contribution and BBA application in BBS system. The size of Buffer if too small will increase the management complexity of BBM, while too large Buffer will cause the wastage of storage resources. The optimal buffer size is not the focus of this paper(smaller than 10 KB would be an overkill [8]), and we suppose that BBD contributes and BBA can be applied in a uniform buffer size.

From the perspective of BBM, the attributes of each Buffer could be described by triples $\langle \mathrm{DT}, \mathrm{DF}, \mathrm{DR} \rangle$. DT refers to the contribution time. DR refers to the expected expire time. DR refers to the reliability property. Then the buffer ID i can be denoted by $b_i\langle \mathrm{DT}_i, \mathrm{DF}_i, \mathrm{DR}_i \rangle$. For example, an end user turned on the computer at 8: 00 pm, and expected to shut down at 11: 00 pm. Thus the user could start the BBD client application (client ID is i) and set the contributed buffer's property DT = 8: 00 pm, DF = 11: 00 pm. DR is the loss rate based on the client's historical information gathered by BBM.

From the perspective of BBA, as the application makes it difficult to determine when to release the buffer, so the applications only have a reliability requirement which is denoted by AR.

Logically, BBM contains two buffer sets. One is FBS (Free Buffer Set), which refers to the unused contributed buffers and the other is UBS (Free Buffer Set) that refers to the buffers having been used.

3.2 Buffer management in BBM

BBM buffer management contains the following four kinds of scenes. (1) BBD contributes buffer b. BBM will add b to FBS directly. (2) BBD withdraws buffer b (including BBD dropping due to network failure). BBM will calculate the income BBD gains and update the credit value. If b belongs to FBS, then remove it from the collection directly. While if it belongs to UBS, BBM should do data migration before removing it. For example, in Figure 1(c), when BBD1(b_1) was withdrawn, it would trigger BBM to migrate the data whose copy is stored in BBD3(b_1) to the newly allocated buffer BBD2(b_2). (3) BBA applies for

buffer c. BBM selects one or a set of buffers which could satisfy the reliability requirement of buffer c from FBS and removes these buffers from FBS, then adds them to UBS. The details are discussed in Subsection 3.3. (4) BBA free buffer c. BBM removes the corresponding buffer set from UBS and adds these buffers to FBS, then BBM calculates the cost.

Obviously, there are mainly four complex algorithms or processes in the above buffer management processes. (1) BBD income calculation algorithm; (2) BBD credit updating algorithm; (3) BBA cost calculation; (4) FBS buffer allocation algorithm; (1)–(3) algorithms will be studied later. In this paper, we discuss the FBS buffer allocation algorithm. The aim of the algorithm is to select a buffer set (BS), which contains s ($s \geqslant 1$) buffers, from FBS to satisfy the reliability requirement (AR) of BBA. The ith element's attributes in BS are $\langle \mathrm{DT}_i, \mathrm{DF}_i, \mathrm{DR}_i \rangle$, and satisfy the formula (1).

$$1 - \mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s > \mathrm{AR} \ \text{ i.e. } \ \mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s < 1 - \mathrm{AR}. \tag{1}$$

To ensure BBA buffers' reliability demands AR, data migration operation would be triggered when some buffers in BS were withdrawn. The buffer allocation issue involved in the migration operation can be described as: given the BBA buffer $c\langle \mathrm{AT}, \mathrm{AF}, \mathrm{AR} \rangle$ and the buffer set BS which contains s buffers ($s \geqslant 1$). The s buffers DR attributes meet formula (1): $1 - \mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s > \mathrm{AR}$. When the ith buffer was withdrawn, it was required to find m ($m \geqslant 1$) buffers in FBS to form a new buffer set BS', which is met in formula (2).

$$1 - \mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_{i-1} \times \mathrm{DR}_{i+1} \times \cdots \times \mathrm{DR}_s \times \mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m} > \mathrm{AR}. \tag{2}$$

If we multiply both side by DR_i, we get

$$\begin{aligned}
&\mathrm{DR}_i - \mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s \times \mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m} > \mathrm{AR} \times \mathrm{DR}_i,\\
&\mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s \times \mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m} < \mathrm{DR}_i(1 - \mathrm{AR}),\\
&\mathrm{DR}_1 \times \mathrm{DR}_2 \times \cdots \times \mathrm{DR}_s < \frac{\mathrm{DR}_i(1 - \mathrm{AR})}{\mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m}}.
\end{aligned}$$

As with formula (1), the issue can be transformed into finding m buffers to meet the requirements of formula (3). Comparing formula (3) with formula (1), we could conclude that the data migration issue can be transformed into finding m buffers to meet the reliability demands DR_i. It is equivalent to the buffer allocation problem.

$$\frac{\mathrm{DR}_i(1 - \mathrm{AR})}{\mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m}} < 1 - \mathrm{AR} \Rightarrow \mathrm{DR}_{s+1} \times \cdots \times \mathrm{DR}_{s+m} < \mathrm{DR}_i. \tag{3}$$

3.3 Buffer allocation algorithm

FBS buffer allocation algorithm is the core issue in distributed buffer management. The algorithm should be simple, besides meeting the reliability requirements. Different buffer allocation methods will lead to different data reliability as well as different system recovery traffic.

3.3.1 *An example of buffer allocation*

We describe the effect of different data allocation mechanisms on the system reliability and other aspects through a simple example. There are three allocation mechanisms that are based on: (a) the sequence of events; (b) the remaining survival time; (c) the reliability.

To describe more clearly, we use a three-tuple $\langle \mathrm{DT}, \mathrm{DF}, \mathrm{DR} \rangle = \langle \text{time}, \text{expired time}, \text{loss rate} \rangle$ to represent each contribution operation of BBDs. As shown in Figure 2, $T_i \sim T_{i+1}$ interval is a time unit, denoted as 1t in the timeline. BBD1, BBD2, BBD3 have a contribution operation at time point T_3, T_1, T_6 respectively. Incidentally BBA stored data at time point T_2, T_4, and T_7. The BBDs reliability is shown in Figure 2. Figure 3 (a)–(c) show the resource usage at the time point T_2, T_4, T_5 and T_7 under these three different data allocation mechanisms. The colored blocks represent the buffer that has been allocated.

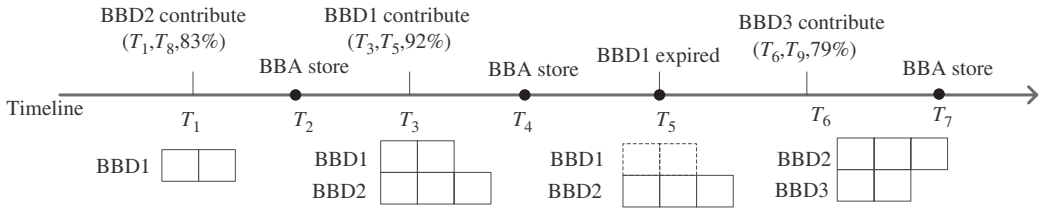

Figure 2 An example of resource management.

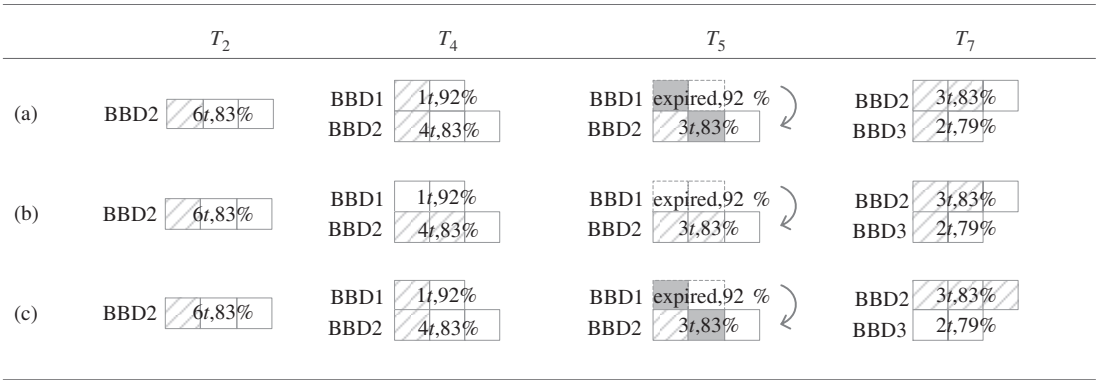

Figure 3 Three different allocation approaches.

Approach (a) in Figure 3 adopts the allocate mechanism based on the sequence of events. At time T_4, there are 2 BBDs and 5 buffers in the system. As the BBD2's contribution operation comes later than BBD1's, so BBM will allocate BBD2's blocks at time T_4 according to the allocation mechanism. Then BBM should do data migration operation when BBD2 expires at time T_5. For the same reason, BBM will allocate BBD3's block to BBA until exhausted at time T_7.

Approach (b) adopts the allocation mechanism based on remaining survival time, which means all those moments BBM stores data to those who has the longest survival time blocks. For example, BBD1 only commits to contribute $2t$ at time T_3 which will be expired at time T_5. However, BBD2's buffer will be expired at time T_8 ($8t > 2t$), so BBM will allocate BBD2's blocks to BBA at time T_4 according to the adopted approach (b). Compared to approach (a), BBD2 had stored nothing at time T_5, so it is futile to do data migration at that time. BBD3 joins to contribute $3t$ at time T_6, more than that of BBD1's remaining time $2t$. Then after time T_6 (for example at T_7 time), BBM will distribute the data on the BBD3.

Approach (c) adopts allocation mechanism based on the reliability. The best reliability buffer always prefers to be allocated. As reflected in Figure 3, BBM will allocate BBD1's block at time T_4 and BBD1's blocks at time T_7. Only when BBD2's buffers are used up will BBM consider allocating BBD3's buffers.

As showed in Figure 3, the allocation effect of these three allocation mechanisms is different, and has different data reliability as well. Using approach (a) and (c), then BBM needs to do data migration after BBD1 expires (at time T_5). While it is not necessary if it adopts approach (b). When approach (c) is adopted, the data is distributed by priority to the high reliability BBDs. And then when these BBDs expire, BBM needs to move a large amount of data. For example, at the time point T_7 in approach (c), all of BBD2 blocks have been allocated, so that the entire data stored on BBD2 migrates upon expiry at the time point T_8, resulting in a relatively large data migration traffic.

The example in Figure 3 actually does not take BBD abnormal loss into consideration. Considering the suddenly exit, BBM does not have time to do data migration. So BBM should have a backup strategy to ensure its reliability. This has been described in Subsection 2.2. It should be noted that different allocation mechanisms have different impacts on data reliability. If BBD3 dropped at time T_7, then the adoption of approach (c) will have the minimum data loss. In a real BBS system, BBD loss is a probability event, and the loss probability differs at all times. Moreover, it is difficult to use the same

data set to compare the performance of these three different allocation mechanisms, thus we implemented a simulator to generate a data set and save it to disk so that we can compare their performance using the same data set.

3.3.2 *Buffer allocation algorithm*

It has been mentioned in Subsection 3.2 that the buffer allocation algorithm is to find a set of buffers BS in FBS to meet the BBA's reliability needs (AR). In fact, it is not necessary for BBA to provide the specific value of AR. Further in order to provide a compatible API, we propose the concept of reliability level. The storage level determines what BBA needs to provide when it stores data in BBS. It is a flexible restriction of the reliability and refers to a reliability interval. BBS offers three storage levels to BBA, namely high, medium and low. Each storage level refers to a reliability interval. The three intervals are ($\mathrm{AR_{high}}$, 1), ($\mathrm{AR_{medium}}$, $\mathrm{AR_{high}}$) and (0, $\mathrm{AR_{medium}}$) respectively.

The buffers are also divided into three levels based on its DR attribute, namely high, medium, low. With their its corresponding DR intervals are ($\mathrm{DR_{high}}$, 1), ($\mathrm{DR_{medium}}$, $\mathrm{DR_{high}}$) and (0, $\mathrm{DR_{medium}}$). The three levels of buffers store the corresponding levels of BBA data. Only when the corresponding levels of buffers are used-up the mix-store can be accessed. When BBA selects the buffer of the same level to store data, BBM needs to distribute two replicas to meet the reliability requirements. That is to say the value of s in Formula (1) should be 2 if the data is stored on the same level. For example, BBA data c whose storage level is high, when it is stored in the high level buffer, BBA should look for two buffers satisfy $1 - \mathrm{DR}_1 \times \mathrm{DR}_2 > \mathrm{AR_{high}}$. Evidently, $\mathrm{DR_{high}}$ should meet (4).

$$\mathrm{DR_{high}} < \sqrt{1 - \mathrm{AR_{high}}}. \tag{4}$$

However, when a certain level buffer is not enough, BBA needs to store data on a low-level or high-level buffer, and the reliability should also meet formula (1). To achieve this and avoid the unnecessary reliability calculations every time BBA stores data, a simple algorithm is adopted. Each of the lower levels, distribute one time more, as showed in Procedure 1. If the low-level buffers are used-up then the higher-level buffers are opted. In this way $\mathrm{DR_{medium}}$ value must meet (5).

$$1 - \mathrm{DR_{medium}} \times \cdots \times \mathrm{DR_{medium}} > 1 - \mathrm{DR_{high}} \quad \text{i.e.} \quad \mathrm{DR_{medium}} < \sqrt{\mathrm{DR_{high}}}. \tag{5}$$

When allocating buffers, BBA should also avoid choosing the same BBD's buffer, because it is useless to distribute two replicas on the same BBD. If the BBD is offline without notification, both of the two replicas will be lost. ($\sigma(b_i)$ denotes the BBD b_i belongs to, $\sigma(Y)$ denotes the set of BBD which all b_i in set Y belongs to.) The algorithm is shown in Procedures 1 and 2.

The procedure FindPhyBlock(L, K, Y) is a key process in the algorithm. It completes the function of finding k buffers in FBS[L], and adds the buffers that result in collection of Y. However, choice of b_j in procedure FindPhyBlock has great implications for the ultimate reliability. Generally, FBS[high], FBS[medium], FBS[low] are organized into a queue. Each time the selection starts traversing from the head of the queue, the diffident organizing method will result with different reliability.

3.4 Buffer organization in FBS

Different FBS organizing methods will lead to selection of different buffers during the b_j allocation process. According to the buffer's three attributes $\langle \mathrm{DT}, \mathrm{DF}, \mathrm{DR} \rangle$, we can get the following kinds of data organizing methods.

(1) Management mechanism based on the sequence of events.

Organize FBS according to the sequence of events (DT attibute). In this method, each time BBM selects the buffers which has just be contributed by BBDs to store BBA's data.

(2) Management mechanism based on the remaining survival time.

Organize FBS according to the remaining survival time (DF attribute). In this method, the data stored in the block can obtain the maximum storage time, i.e. the number of data movement caused by

Procedure 1 DataDistribute(L)

Input: L; {BBA storage level}
Output: Y; {block set $Y = b_1, b_2, \ldots, b_k$}
  $k = 2$; {default distribute 2 copies }
  if i = FindPhyBlock(L, k,Y) $< k$ **then**
    {k represents the k copies need to distribute}
    if j = FindPhyBlock($L-1, k-i+1, Y$) $< k-i+1$ **then**
      {there isn't enough blocks in low level BBDs}
      if FindPhyBlock($L+1, k-i+1-j, Y$) $< k-i+1-j$ **then**
        {If there isn't enough blocks in high level BBDs}
        Add the data to need replicate list;
      end if
    end if
  end if
  return Y

Procedure 2 FindPhyBlock(L, k, Y)

Input: L, k, Y;
Output: num; {the number of physical blocks was found}
  while num $<$ k and FBS[L] $\neq \emptyset$ **do**
    Choose a buffer b_j belongs to FBS[L];
    if $b_j \in \sigma(Y)$ **then**
      $j = j + 1$;
      continue;
    else
      $Y = Y \cup b_x$ and FBS[L] = FBS[L] $- b_j$;
      num $++$;
    end if
    if have traversal FBS[L] and num $< k$ **then**
      break;
    end if
  end while
  return num

expiring of the BBD buffer is smaller. If BBDs keep their promises, then the data migration pressure will be much smaller.

(3) Management mechanism based on the reliability.

Organize FBS in descending order according to buffers' DR attribute. In this method, the first choice is the lowest loss rate buffers, that would achieve the least number of data movements caused by BBD loss. But data movement caused by expired BBD is more than the second mechanism.

3.5 Dropped BBD detection strategy and data recovery strategy

Dropped detection strategy is divided into active and passive dropped detection methods. Active dropped detection means that BBM scans BBD Database regularly to determine if it is online. Passive dropped detection means that BBM does not perform periodic scans, however through the timeout event it interacts with BBD to determine whether this BBD is online. Active detection strategy can find dropped BBDs earlier, and perform data migration on time, rather than doing it until all the copies are all lost. So it can reduce the data loss probability.

For the purpose of data recovery, there are several strategies available as well. (1) Recover data immediately after having detected the dropped BBD. This approach momentarily leads to hundreds of MB traffic for losing just one BBD, which is a big pressure to the existing network. (2) Recover the block when BBA accesses it the next time. This approach may lead to failure in copying all the data, and result in data loss. (3) Recover the data when BBD's next heartbeat packets are sent. In this way traffic can be dispersed effectively.

4 Experiment evaluation

4.1 BB-SIM design

In this section, a BB-SIM was designed to simulate BBS and evaluate three buffer allocation strategies based on its organization mechanisms. The main function of BBM was implemented in BB-SIM, such as those of the distributed buffer management mechanism and buffer allocation algorithm in BBM as described in Subsection 3.3. For some complex communication process, we simulated them rather than their implementation, for instance, the communication protocols between BBM and BBDs.

For BBM, the interaction with BBD or BBA was actually a series of event processing. The events that BBM have to handle are (1) BBD contributing buffer; (2) BBD's buffer expiry; (3) BBD loss or logout; (4) BBA apply for buffer; (5) BBA free buffer. Take for instance, the contributing buffer event processing of BBM; BBD sends a message to BBM to allow contributing a buffer and the contribution time at a certain period, after receipt of the message, the BBM processes and sends back an ACK message to indicate successful protocol. If the network communication is always successful, then the handshake process can be ignored then. The BB-SIM was designed based on this idea.

BB-SIM consists of three simulation modules, BBM-Simulator, BBA-simulator and BBD-simulator. A variable i is used to represent the time slice and the increment of i indicates the time flow. In the simulation of BBD contributing buffer, BBD-simulator randomly chooses a start-time, and generate a contribution degree and a contribution time randomly, then these parameters are written in a specified file. By reading this file, BBM-simulator can get the process information. BBM and BBD can communicate with each other in this method.

Similarly, BBA chooses the start-time randomly, assigns the storage level, and writes in the sample file. The events mentioned above could be simulated by BBD-Simulator and BBA-Simulator adopting this method. A series of simulation events are written on the sample file, and then the BBM-simulator reads and processes the file off-line. Through the increment of timeline, BBM keeps on dealing with the coming events just as in a real system. Only sample file guarantees the fairness of different tactics and reliability of contrastive results.

4.2 BB-SIM parameters

In this experiment, BBD-simulator instantiated 30 BBDs. The buffer number of each contribution could be one of these three numbers 500, 1000, 1500. BBA-simulator imitated written operation 9475 times, the simulation lasted 10000 time slice, and the events were distributed in the 10000 time slice randomly. The backup strategy was backing-up data immediately when BBD exit was detected. And the experiment adopted passive off-line check strategy. That is to say BBM did not check BBD's online status all the time. The checking operation occurred only when reading or writing operation happened. Once the BBD off-line event had been checked, the simulator started to backup data immediately. The backup strategy and off-line checking strategy will have a great impact on the experimental results, such as the distributing of data movement.

As BBS is to provide differentiated storage service, we should divide BBDs to different classes according to its loss rate. Define $\mathrm{AR}_{\mathrm{high}} = 99\%$, then we could calculate that $\mathrm{DR}_{\mathrm{high}} = 0.1$ according to (4). So $\mathrm{DR}_{\mathrm{medium}}$ would be less than $\sqrt{\mathrm{DR}_{\mathrm{high}}} = \sqrt{0.1} = 0.32$ according to (5). In this experiment, let $\mathrm{DR}_{\mathrm{medium}}$ equals 0.3. The BBDs whose failure rate is less than $\mathrm{DR}_{\mathrm{high}} = 0.1$ are assigned to the high-level, storing BBA's high-level data; failure rate is less than $\mathrm{DR}_{\mathrm{medium}} = 0.3$ greater than $\mathrm{DR}_{\mathrm{high}} = 0.1$ are assigned to the medium-level, storing BBA's medium-level data. While the rest BBDs are assigned to the low-level.

4.3 Preliminary simulation results

BB-SIM simulates the BBS system, and the following figures described the simulation results. Figure 4 reflects the variation of total buffer number. The inflection point indicated that some events happened and had influenced the status of the system. For instance, the event of BBD contributing buffers, expiring of buffers or the logout event of BBDs, with either of these occuring, the system's buffer number alters.

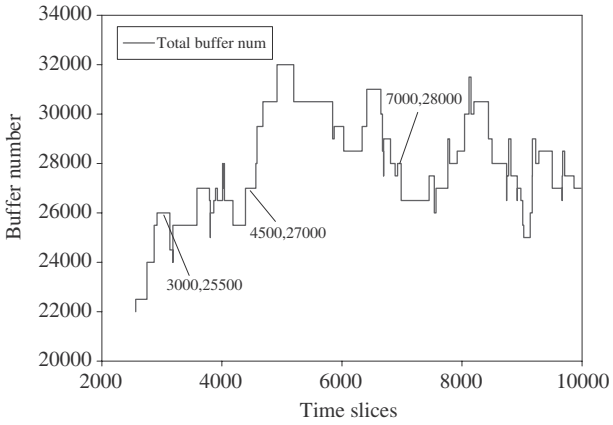

Figure 4 Variation of total buffer number.

Table 2 BB-SIM parameters

Items	Value
Number of BBDs	30
Contributed size	One of 500, 1000, 1500
BBA write buffer number	9475
BB-SIM run time	10000
Storage level	$\mathrm{AR_{high}} = 99\%,\ \mathrm{AR_{medium}} = 91\%$
Loss rate	$\mathrm{DR_{high}} = 0.1,\ \mathrm{DR_{medium}} = 0.3$

Before time slice 3100 we can see that, it belongs to the system warming-up process, and almost didn't lose data. While after time slice 3100, downward lines began to appear, which indicated that buffer gradually expires or drops. Since we assumed that every time BBD contributed a buffer set (described in Table 2). When a buffer was expired, the buffers belongs to the same buffer set were expired simultaneously. This phenomenon can be observed from the big span of every inflection point in Figure 4.

The simulation applied the same sample and same distributing strategy but in different buffer allocation mechanisms, so we could analyze the performance divergence objectively. There are three allocation mechanisms: (a) allocation mechanism based on the sequence of events; (b) allocation mechanism based on the remaining survival time; (c) allocation mechanism based on the reliability.

We compared these mechanisms in the following three aspects: the number of data movement, which reflects the system maintain flow of BBS; data loss rate, which reflects the data reliability; the payload of each BBD. BBM should try to balance the load so as to avoid the great data recovery traffic when a BBD was lost.

(1) Data movement and loss rate. According to Table 2 managing buffer based on the remaining survival time could achieve lowest data movement times; however, it may cause significant burst of traffic reflected by Figure 4. The third method's total number data movement times are highest, while its data loss rate is the lowest. Conjunction with Table 2 and Figure 4 we can also see the first method obtains relatively small number of data movements, and an average maintain flow. The reason would be that the third method ensures data reliability priority every time, so the failure rate can be guaranteed. However the data may be distributed to about to expire BBDs, leading to a situation that just completed the data distribution and will perform the data migration immediately, so that the number of data movement would be more. On the contrary, in the second method, data is stored for the longest time, so the total number of data movements are the smallest, but data reliability cannot be guaranteed.

Figure 5(a) reflects the accumulative data movement in the 10000 time slices. It can be seen that there was not any data movement before time slice 3100, which refers to the warming-up progress. The data begun to move around time slice 3400, which means buffers began to lose. We can see the data moving intensively near time 3500, 6000–8000, 9000. The reason would be buffer expired or BBD lose in these

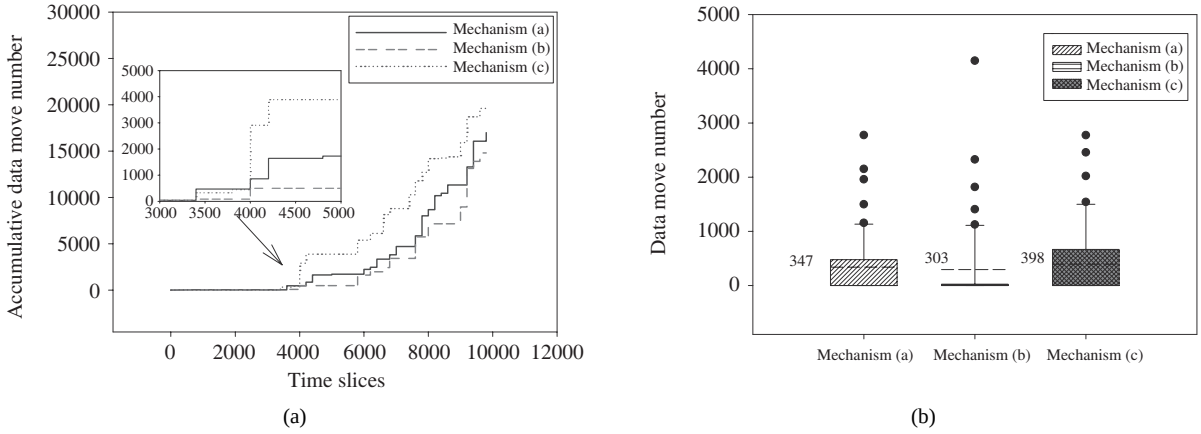

Figure 5 The data move number of three allocation mechanisms. (a) Accumulative data move number; (b) box plot of the data move number.

Table 3 The general performance of these three mechanisms

	Data move number after 10000 time slices	Data loss rate (%)
Mechanism (a)	17000	0.36
Mechanism (b)	14813	1.12
Mechanism (c)	19617	0

points. And we can also draw this conclusion from Figure 4. There are significant descending lines in Figure 4 near these points.

Although mechanism (c) has the largest total data move number which is shown in Figure 4. However, when comparing the usual data move, mechanism (c) may not be the worst. The usual data move number reflects the overall recovery traffic. Figure 5(b) is a Box Plot of data move number, and can reflect the overall data distribution. We can see that most of the time the data move number of mechanism (b) was relatively small. But mechanism (b) had a large outlier value which deviated from the highest point, causing a great traffic burst. Figure 5(b) also reflects that the overall performance between mechanism (a) and mechanism (c) is almost the same. The median value of mechanism (a) is 347 and smaller than mechanism (c)'s median 398. It means the average performance of data movement is slightly better.

Comparing the loss rate among these mechanisms, there is no doubt that mechanism (c) achieved the lowest loss rate, which is shown in Table 3. However, mechanism (c) also has the largest overall data move number. Theoretically, according to Table 2, the loss rate should be less than $1 - 91\% = 9\%$. Results in Table 3 are in accordance to the theoretical value.

In general, mechanism (a) can get a relatively small data move number, and a relatively small average recovery traffic, while mechanism (c) has the best reliability performance. The reason is that mechanism (c) selects the BBDs with the best reliability to store data so that the loss rate can be guaranteed. Meanwhile, this strategy could also bring about a situation that distributes data to the BBDs which are about to expire, causing data migration. Thus, the total data move number is higher than the other two mechanisms. In contrast, mechanism (b) selects the buffer with longest surviving time so that it has the least data move number, but the data reliability cannot be guaranteed.

(2) Payload comparison between BBDs. Pick a few representative time points to show the payload of each BBD. We selected time points 3000, 4500, 7000. Because time point 3000 is a point that warming-up process has just ended and did not have much data loss, so it can reflect the running state after system warming-up. Time point 4500 can reflect the BBD payload situation where buffer begins to lose just after the warming-up progress. Time point 7000 can reflect the payload situation under BBS stable status. The colored histograms in Figure 6 represent the used blocks of each BBD, and the grey histograms represent the total blocks on that BBD.

As can be seen from the Figure 6, mechanism (a) can distribut blocks to different BBD compared to

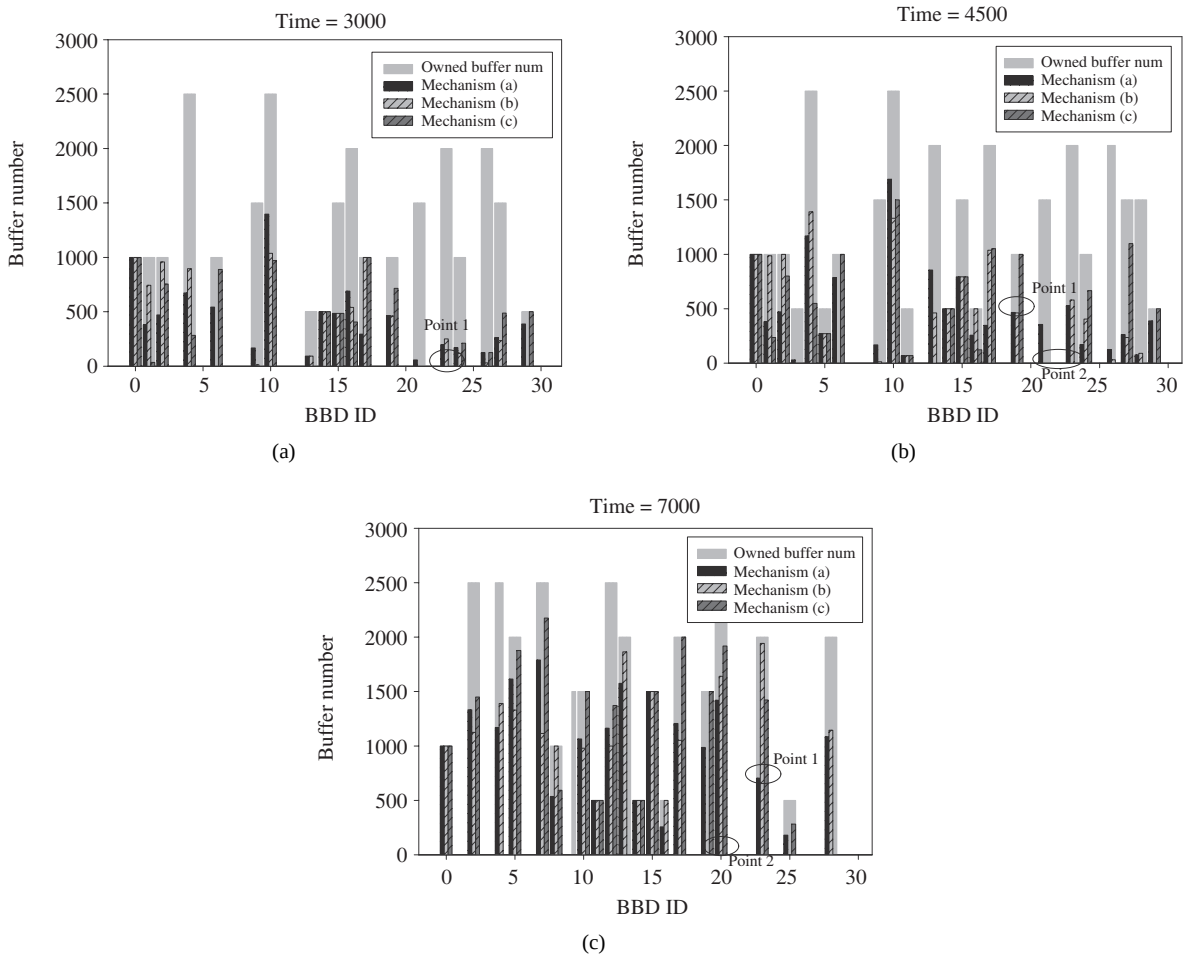

Figure 6 The payload of three allocation mechanisms at three time point. (a) At time point 3000; (b) at time point 4500; (c) at time point 7000.

the other two mechanisms. However, the situation that some BBDs were overload and some BBDs were idle existed in both the other two mechanisms. Take point 1 in Figure 6(a) whose BBD ID is 23 for example, it can be seen that it is empty, while the BBD 17 is full load in the case of mechanism (c). Similarly, at the other two time points, there occurs such a situation. For mechanism (c) in Figure 6(b), the BBD at point 2 is idle, while the BBD at point 1 is full load. For mechanism (b) in Figure 6(c), the BBD at point 2 is idle, while the BBD at point 1 has a heavy load.

Overall, mechanism (a) could spread the system maintain traffic throughout the whole running period and has a relatively small traffic burst as well. The payload of mechanism (a) is also better than the others two. Although the metric of reliability is not as good as mechanism (c). But in view of the streaming media applications, the reliability is acceptable. Thus, mechanism (a) performed better than the other two mechanisms.

5 More discussions about BBS

Data security issue. Similar to the P2P system, BBS have security issues, such as content identification issue, malicious code issue, etc. However, we believe that the introduction of BBS did not let the security situation become critical, and does not introduce new security issues. The existing P2P security solution to these issues can be applied in BBS.

Scalability bottleneck. Early P2P systems, such as Napster[1)], adopted the centralized control mode.

1) Napster L. Napster. https://en.wikipedia.org/wiki/Napster. 2001.

Napster encountered scalability bottlenecks while excution. Under the scenario of many P2P users, Napster server is unable to meet the massive demands. In BBS system, this problem is well resolved, as BBS deployed on the edge network, only provides service to thousands of users in the edge network, rather than that Napster serve the whole Internet users, BBS has a relatively smaller scale and the scalability bottleneck issue would not appear.

BBD behavioral modeling. In the simulator, BBD joins and exits the system randomly. This simple method does not describe BBD's behavior accurately. BBDs join and exit is a probability event, and the probability is changing all the time. However, it remains the same all the time in current simulator. The backup strategy adopted in current simulator is restore immediately if BBM has detected BBD loss. If we adopt lazy backup strategy, then it will be necessary for us to analyze the data reliability during the delayed time to ensure reliable storage service. Then we need to model the BBD behavior to prove its reliability. In fact, BBD's join and exit can be modeled by birth-and-death process. In our future work, we may consider modeling BBD's behavior to reflect some regularity problems.

Incentive issue. The intensive access to storage resources in end-users' computers via edge network may cause excessive bandwidth consumption to each individual user, which may prevent users from continuing contributing their resources. It relates to the incentive mechanisms in BBS. We find this problem too negligible to hinder the development of BBS. In fact end-users could get interested by depositing storage resources in BBS and get the reimbursement from BBA in the form of value-added services. All it requires is cooperation with these BBAs. One possible approach is to provide a virtual monetary system. BBD can get money through depositing storage resources and can use this money to get BBA value-added services. For the BBA, the income has decreased which reflects in two aspects. On one hand, the money BBA obtains from a single user decreases as a result of partial mortgage by BBS virtual currency. On another hand, BBA spends money to pay for BBS services. However, this occurs at the exchange of not upgrading the existing servers, as well as increasing subscribers. For instance, if the original service is 10/month, a BBD mortgaged 5 for the virtual money acquired by continuously depositing storage resource. Thereby the member price reduced to 5/month. This price will be very attractive. Then both the number of BBA service subscriber and the number of BBDs will increase, resulting in a win-win situation.

6 Related work

In the domain of In-network storage, researchers have proposed a variety of applications covering different aspects of the Internet. As it was mentioned in Section 1, In-network storage resources could be divided into two types. One is storage from ICP or ISP; another is the end-users' storage resources. Apart from the two usage patterns in Table 1, the third usage pattern is a hybrid mode, using both ICP/ISP and end-users' storage. It is called hybrid CDN + P2P method in article [13]. Indeed, it could save server bandwidth. Lots of researches have shown that, such as online file hosting system (FS2You [13]), P2P file sharing system (Napster[1)]), live video streaming and VoD system (Novasky [14], PPLive [15]) etc. Although the same storage usage pattern exists, they still have different design, focuses and issues.

P2P online file hosting systems (FS2You [13]) focus on the allocation of limited storage to potentially enormous amount of data files so that semi-persistent file availability may be achieved. VoD systems and live video streaming systems, such as Novasky [14], PPLive, require a real-time delivery. In particular, the key design objective of a VoD system is to guarantee continuous playback and short buffering delays after a random seek or an initial startup, which is not an issue in online hosting systems.

However, BBS is different from the entire above said hybrid CDN + P2P systems. BBS appears to be a storage infrastructure in the Internet and decouples the storage from enormous applications. That is to say the storage resources aggregated by BBS including end-users' storage and ISPs' storage, can be used by a variety of advanced applications. For instance, when FS2You [13] which is a P2P online hosting system adopts BBS service, it treats BBS as a replication server, which is closer to the edge network and has a greater storage capacity. In this use scenario, FS2You system is a BBA from the perspective of BBS. BBS, as an infrastructure, can serve other applications simultaneously, such as the

VoD application, Novasky [14]. In Novasky, BBS could be treated as a super peer, which has a tremendous uploading bandwidth as well as huge storage resources.

7 Conclusion and future work

This paper carefully discusses the implementation mechanism of a new storage model, compares three buffer management mechanisms, and analyzes the performance of these three methods by simulation. Since it is a preliminary simulation, hence we did not evaluate the system throughput, and delay. In our future work we would implement the complete system, in order to compare its performance with that of the existing P2P Cache. Existing data distribution algorithm seems rather simple, and the load balancing algorithm and other algorithms should be added to the system, and then a comprehensive comparative analysis could be performed.

Acknowledgements
This work was supported by National Basic Research Program of China (973 Program) (Grant No. 2012CB3159-06), Program for New Century Excellent Talents in University (Grant No. 4345113151) and National High-tech R&D Program of China (863 Program) (Grant Nos. 2011AA01A10, 2012AA01A50606).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Alimi R, Yang Y, Rahman A. A survey of in-network storage systems. RFC 6392. 2011. http://www.rfc-base.org/rfc-6392.html
2 Berkheimer A D, Dilley J A. Method and system for tiered distribution in a content delivery network. US Patent, Patent US7133905. 2006
3 Hefeeda M, Hsu C H, Mokhtarian K. Design and evaluation of a proxy cache for peer-to-peer traffic. IEEE Trans Comput, 2011, 60: 964–977
4 Xylomenos G, Ververidis C, Siris V, et al. A survey of information-centric networking research. IEEE Commun Surv Tutor, 2014, 16: 1024–1049
5 Zhang L, Estrin D, Burke J, et al. Named data networking (NDN) project. Relatrio Tcnico NDN-0001, Xerox Palo Alto Research Center-PARC, 2010
6 Zhang Y, Zong N, Camarillo G, et al. Problem statement of P2P streaming protocol (PPSP). IETF PPSP BoF, 2008
7 Huang B, Sun Z, Chen H, et al. BufferBank: a distributed cache infrastructure for peer-to-peer application. Peer-to-Peer Netw Appl, 2014, 7: 485–496
8 Rossini G, Rossi D. Evaluating ccn multi-path interest forwarding strategies. Comput Commun, 2013, 36: 771–778
9 Muscariello L, Carofiglio G, Gallo M. Bandwidth and storage sharing performance in information centric networking. In: Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking. New York: ACM, 2011. 26–31
10 Ousterhout J, Agrawal P, Erickson D, et al. The case for RAMClouds: scalable high-performance storage entirely in DRAM. ACM SIGOPS Oper Syst Rev, 2010, 43: 92–105
11 Chen H, Sun Z, Huang B, et al. Who is more reliable? An evaluation method for distributed-memory aggregation in the Internet. In: Proceedings of the 2012 ACM Conference on CoNEXT Student Workshop. New York: ACM, 2012. 41–42
12 Kim J T, Park H K, Paik E H. Security issues in peer-to-peer systems. In: Proceedings of the 7th International Conference on Advanced Communication Technology (ICACT 2005). Phoenix Park, 2005. 1059–1063
13 Sun Y, Liu F, Li B, et al. Fs2you: peer-assisted semi-persistent online storage at a large scale. In: Proceedings of the 28th Conference on Computer Communications, Rio de Janeiro, 2009. 873–881
14 Liu F, Shen S, Li B, et al. Novasky: cinematic-quality vod in a P2P storage cloud. In: Proceedings of the 30th IEEE International Conference on Computer Communications, Shanghai, 2011. 936–944
15 Huang Y, Fu T Z J, Chiu D M, et al. Challenges, design and analysis of a large-scale P2P-VoD system. ACM SIGCOMM Comput Commun Rev, 2008, 38: 375–388