Peer-to-Peer Netw. Appl. (2016) 9:299–312
DOI 10.1007/s12083-015-0337-z

A novel hybrid P2P and cloud storage system for retrievability and privacy enhancement

Gi Seok Park · Hwangjun Song

Received: 21 October 2013 / Accepted: 23 February 2015 / Published online: 7 March 2015

Abstract This paper presents a novel fountain code-based hybrid P2P and cloud storage system. While most cloud storage systems guarantees data retrievability of high level, they may be vulnerable to data privacy since the stored data may be exposed to others by accident or design. On the other hand, P2P storage system keeps any peer from accessing the whole data by dividing the data into small pieces and distributing them to multiple participating peers, but may degrade data retrievability due to unstable peers. The proposed hybrid storage system attempts to enhance data retrievability and privacy by effectively distributing fountain encoded symbols to cloud server system and participating peers. It is demonstrated that the proposed hybrid storage system achieves the desired level of data retrievability with a short upload time, and enhance privacy by preventing others from reading the contents.

Keywords Hybrid P2P and cloud storage · Fountain codes · Data privacy · Hybrid storage system reliability · Data retrievability

1 Introduction

In recent years, remote data storage systems that are accessible over the Internet have attracted a huge amount of research interest [1]. Many storage services such as Amazon Glacier [2], Google Drive [3], and Microsoft SkyDrive [4] have already been successfully deployed over the Internet. Nowadays, these remote storage services are becoming more and more popular since they make ubiquitous data access possible [5]. For example, the number of storage customers for Dropbox [6] reached 100 million in 2012. In general, remote storage systems can be classified into two groups: cloud storage systems and P2P storage systems. In cloud storage systems, a high level of data retrievability can be guaranteed using a mirroring technique for data backup similar to replication. In this case, data retrievability means the probability that the user can obtain his/her data successfully [7]. However, private data may be exposed to other users since all data is stored at the server. Therefore, data privacy is one of the most important issues for cloud storage systems [8]. Storage scalability is also a growing concern in cloud storage since the number of cloud customers is increasing rapidly. According to the report in [9], the total number of cloud storage customers is likely to reach one billion in the near future. On the other hand, in P2P storage systems, some of the problems that are inherent in cloud storage can be solved. A unique characteristic of the P2P system is that peers share their resources with each other when they join a network. Therefore, as more peers participate in the system, P2P storage can be extended constantly. In addition, higher data retrieval download rates can be supported as compared to a single source since users retrieve their data from multiple peers simultaneously [10]. In the P2P storage system, when multiple copies of whole data are stored to peers, the privacy may be seriously degraded compared to cloud storage. On the other hand, when the data are divided into many small pieces and distributed to multiple participating peers, any peer cannot access the whole data. However, the relatively low data retrievability is a serious obstacle to the successful deployment of P2P storage [8]. Thus, data retrievability is considered as one of the most important performance measures for P2P storage systems. Mach research effort has been devoted to provide data retrievability equal to that of cloud storage systems [10–15]. Until now, the most efficient method for achieving high data retrievability for P2P storage is to use erasure protection codes such as LDPC (low-density

G. S. Park · H. Song (✉)
Department of Computer Science and Engineering, POSTECH, Pohang, Gyeongbuk, South Korea
e-mail: hwangjun@postech.ac.kr

parity-check) and LT (Luby transform) codes and to store encoded data in multiple peers until the required data retrievability is satisfied. However, this may increase the upload time.

In this paper, we propose a fountain code-based hybrid P2P and cloud storage system to provide data privacy, storage scalability, and data retrievability. Some of the unique features of the proposed algorithm are that it adopts rateless fountain codes and includes a packet distribution algorithm in order to guarantee the required level of data retrievability and privacy. Furthermore, data is stored with a short upload time. The rest of the paper is organized as follows. The background information related to storage systems is described in Section 2. The proposed hybrid P2P and cloud storage system is then presented in Section 3. Simulation results are provided in Section 4. Finally, concluding remarks are presented in Section 5.

2 Background

We survey research related to P2P and cloud storage systems in Section 2.1, and briefly review the characteristics of erasure protection codes in Section 2.2.

2.1 Remote storage systems

A remote storage system is one in which multiple clients share and access remotely located data as if they were accessing their own local storage system [16]. Two technologies for remote storage systems are reviewed in this section: the CIFS (common Internet file system) and the NFS (network file system) [17, 18]. CIFS is Microsoft's version of a distributed system. It is an extended version of the SMB (server message block) protocol that can provide shared access to files, printers, serial ports, and miscellaneous communication between workstations over the Internet. The CIFS protocol is based on a client–server service model, and is widely used in local area networks. The client requests a file from a server that is remotely located. In response to the client's requests, the server provides the appropriate file. On the other hand, NFS is a distributed file system protocol developed by Sun Microsystems. It is an open standard defined in RFCs. The NFS protocol is designed for communication between multiple workstations regardless of the kind of transport layer protocol, computer system, OS, or network architecture. It is used between geographically distributed workstations, and it allows multiple workstations to be operated as a single workstation. Since the remote files are connected to the client directory through the network, the clients can access not only their local file systems but also the local file systems of the other workstations as if they were their own local file systems. Hence, it is possible to save storage space.

In recent times, much research effort has been devoted to effectively provide reliable remote storage system. Many related studies have proposed ad hoc solutions using redundant data to build storage systems with high reliability. In [10], Li et al. propose an adaptive erasure resilient code scheme that uses an RS (Reed-Solomon) code to build a reliable P2P storage system. In this scheme, the original data is spilt into k original fragments, and coded fragments are then generated from these original fragments. The coded fragments are distributed to peers to achieve the required data retrievability. The appropriate fragment size is then adaptively chosen according to the different file sizes in order to reduce the overall network bandwidth needed to store the redundant data. When compared to replication (non-ERC), the authors verified that the adaptive ERC (erasure-resilient code) method improves the overall bandwidth efficiency. In [11], Gaidioz et al. explore the feasibility of implementing a distributed storage system using a redundancy scheme with LDPC codes. The authors conduct a comparison of the retrieval failure rates of original data between replication and LDPC. In addition, they express the retrieval failure rate as a function of the number of data chunks for different LDPC code rates. In [14], Kim et al. investigate the performance of erasure protection codes (LDPC, LT, and Raptor codes) for P2P storage. They show the retrieval failure rate with respect to the storage overhead. Kim et al. verify that LDPC codes have an inferior performance when compared to Raptor codes in the case of P2P storage, whereas they are superior to LT codes. In [19], Cao et al. design an LT code-based secure cloud storage service that addresses the hybrid storage system reliability issue with near optimal performance. To protect data confidentiality, existing encryption techniques or data access control schemes are utilized prior to the LT encoding process. This prevents the cloud server from prying into outsourced data.

2.2 Erasure protection codes

Here, we review some of the erasure protection codes such as RS codes, LT codes, and Raptor codes with Table 1.

Reed-Solomon codes RS codes can be used to develop reliable storage systems. RS codes belong to a class of MDS (maximum-distance separable) codes [20]. A message consisting of K symbols can be recovered after receiving K distinct encoded symbols. In practice, however, there are several complications. The field size imposes a limitation on the number of distinct encoded symbols that can be created. In addition, the decoding complexity of practical RS (K+r, K) codes is $O((K+r)^2)$, thus making them too complex for real-time applications.

LT codes LT codes were the first practical realization of rateless codes that can generate an unlimited number of

Table 1 Comparison of RS, LT, and Raptor codes

Codes characteristics	RS codes	LT codes	Raptor codes
Encoding complexity	Quadratic complexity	$O(\ln k_{encoded})$	Linear complexity
Decoding complexity	Quadratic complexity	$O(k_{encoded} \cdot \ln k_{encoded})$	Linear complexity
Decoding symbol overhead	Zero	$O\left(\sqrt{K_{sourece}} \ln^2 \left({}^{K_{sourece}}/_{\delta_{failure}}\right)\right)$ with error probability $\delta_{failure}$	close to zero
The number of encoding symbols that can be generated from a given set of source symbols	Limited	Rateless	Rateless

encoded symbols [21]. In general, the source symbols can be recovered with a high decoding success rate. This is made possible by using a slightly larger number of encoded symbols than the number of source symbols [22]. The code rate (c) plays an important role as it determines the amount of redundant data for error protection. That is

$$c = {}^{K_{source}}/_{k_{encoded}}, \quad (1)$$

where K_{source} and $k_{encoded}$ are the number of source symbols and encoded symbols, respectively. An encoded symbol is generated in the following manner:

1) Choose a degree d for the encoded symbol, according to a predetermined distribution.
2) Choose d distinct source symbols uniformly at random, and generate the encoded symbols by conducting a bitwise XOR of these d source symbols.

This process is repeated until the last encoded symbol is generated. LT codes can generate each encoded symbol using an average of $O(\ln k_{encoded})$ symbol operations, and can recover the source symbols using an average of $O(k_{encoded} \cdot \ln k_{encoded})$ symbol operations.

Raptor codes Raptor codes are an extension of LT codes that achieve a linear encoding/decoding processing time. Raptor codes are a concatenation of pre-code and LT codes, as shown in Fig. 1b. Initially, Raptor codes pre-code the source symbols using a fixed high-rate systematic linear code. The resulting intermediate symbols are then encoded using an LT code, and a limitless sequence of encoded symbols can be generated dynamically. Decoding can be perfectly conducted after receiving any set of encoded symbols whose cardinality is only slightly greater than that of the source symbols. For these reasons, Raptor codes have been adopted by the 3GPP (3rd Generation Partnership Project) standard as a forward error correction scheme in MBMS (multimedia broadcast/multicast services) [23, 24].

3 Proposed hybrid P2P and cloud storage system

The main goal of the proposed hybrid P2P and cloud storage system is to minimize the upload time while satisfying the required data retrievability and supporting the privacy of user data stored on P2P and cloud storage systems. The proposed hybrid storage system adopts the use of fountain codes to achieve our goal. Thanks to the generic characteristics of fountain codes, we can achieve data privacy by controlling the number of encoded symbols stored on cloud storage and peers in P2P storage. The generic nature of fountain codes also helps to improve data retrievability by distributing a sufficient number of encoded symbols on cloud storage and among peers. Before presenting a detailed description of the proposed hybrid storage system, we define the following:

Definitions

- **Cloud storage and peer availability** means the probability that the user is able to access the data stored on cloud storage and peers for the given time interval.

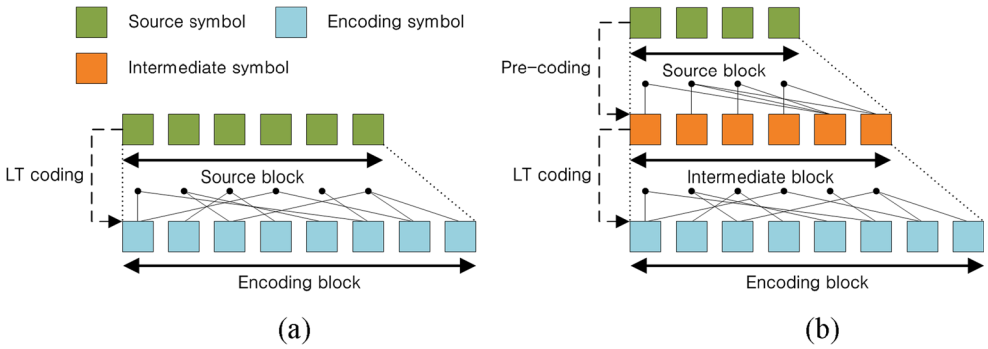


Fig. 1 Examples of fountain codes: (**a**) LT codes (**b**) Raptor codes

- **Storage system reliability** denotes the probability that the user retrieves more than the minimum number of encoded symbols required for successful fountain decoding.
- **Data retrievability** stands for the probability that user obtains his/her own data successfully without any errors.

3.1 System architecture

The symbol descriptions frequently used in the paper are summarized in Table 2 and the proposed hybrid storage system architecture is shown in Fig. 2. The proposed system consists of four main components: a node manager, a parameter control unit, a fountain encoder and an upload scheduler. The node manager receives the information of the server in cloud storage and peers in P2P storage from the bootstrap server, and estimates the bandwidth available with them. In the proposed system, the pathChirp algorithm [25] is employed to estimate the bandwidth. This algorithm is an estimation scheme based on probe packet arrival intervals. The parameter control unit determines the packet distribution vector (i.e. how much data should be stored on the cloud storage server and peers) and the code rate of the fountain encoder based on the feedback information. The fountain encoder divides a file into several blocks, and generates encoded symbols for every source block at the code rate determined by the parameter control unit. The upload scheduler allocates encoded symbols to each node based on the estimated bandwidth and packet distribution vector until the ACK message arrives. In the proposed system, the FFDH (first-fit decreasing height) algorithm

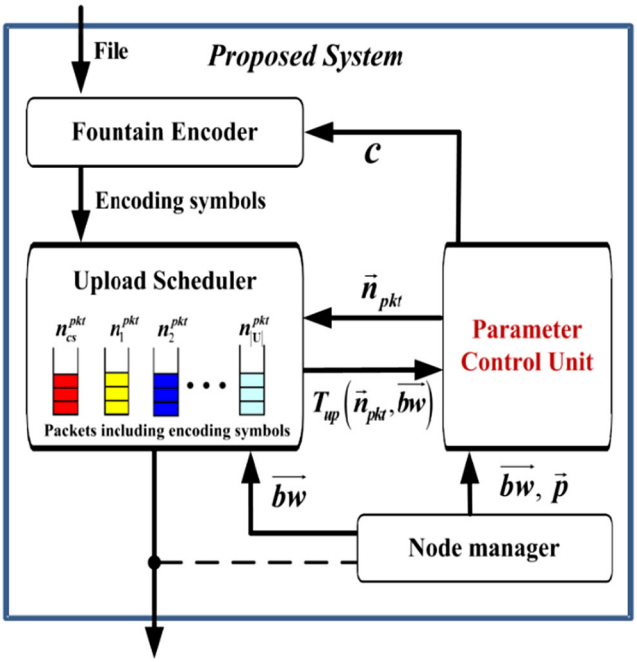


Fig. 2 Architecture of the proposed hybrid P2P and cloud storage system

Table 2 Symbol descriptions

Symbol	Description
$K_{\min}$	The minimum number of encoded symbols required for successful decoding (Refer to Eq. (6))
K_{source}	The number of source symbols
$k_{encoded}$	The number of generated encoded symbols
S_{symbol}	Symbols size
$c\left(\vec{n}_{pkt}\right)$	The ratio of K_{source} to $k_{encoded}$.
$U_{initial\_set}$	The initial peer set randomly selected by the bootstrap server
U_{add}	The additional peer set for maintenance
T_{req}	Time interval during which the user can requests data access
T_i^{stay}	Time that the i_{th} peer has stayed in a system
N_{ps}	The number of encoded symbols in a packet
$p_{retrieve}^{\min}$	The minimum required data retrievability
$\delta_{failure}$	Fountain decoding failure rate
$T_{up}\left(\vec{n}_{pkt}, \overrightarrow{bw}\right)$	The upload time
$\overrightarrow{bw} = \left(bw_{cs}, bw_1, bw_2, \ldots\ldots, bw_{\lvert U_{initial\_set}\rvert}\right)$	The available bandwidth vector, where bw_{cs} and bw_i indicate the estimated bandwidth available with the server in cloud storage and the i_{th} peer in P2P storage, respectively.
$\vec{n}_{pkt} = \left(n_{cs}^{pkt}, n_1^{pkt}, n_2^{pkt}, \ldots, n_{\lvert U_{initial\_set}\rvert}^{pkt}\right)$	The packet distribution vector, where n_{cs}^{pkt} and n_i^{pkt} denote the number of packets including the encoded symbols that the user allocates in cloud storage and the i_{th} peer in P2P storage, respectively.
$\vec{p}_{node} = \left(p_{cs}, p_1, p_2, \ldots, p_{\lvert U_{initial\_set}\rvert}\right)$	The node availability vector, where p_{cs} and p_i denote the probability of availability of cloud storage and the i_{th} peer during T_{req}, respectively.
h_i	The remaining storage space in the i_{th} peer
$SR\left(\vec{n}_{pkt}\right)$	The hybrid storage system reliability
$DR\left(\vec{n}_{pkt}\right)$	The data retrievability

[26, 27], which provides a near optimal solution for the scheduler design problem, is employed for an upload scheduler.

3.2 Problem description

In the proposed system, our control variables are c and $\vec{n}_{pkt}$. Generally, when S_{symbol} and $\delta_{failure}$ are fixed, the amount of overhead (i.e. $K_{\min}-K_{source}$) decreases, but the fountain coding complexity increases as K_{source} becomes larger. In this paper, for the sake of simplicity, it is assumed that K_{source} is appropriately pre-determined while taking into account the coding complexity and the overhead. Under this assumption, c has a strong dependency on $\vec{n}_{pkt}$ since $k_{encoded}$ is the sum of all elements in $\vec{n}_{pkt}$. At this point, we need to describe the hybrid storage system reliability and data retrievability because the fountain decoding process may fail with a very low probability even though $K_{\min}$ encoded symbols are available.

Peer availability Until now, a large amount of research efforts have been devoted to characterize P2P system in terms of lifetime (session length) of peers. It is observed in [28–31] that the distribution of peer lifetime is generally heavy-tailed and the lifetime of peers depends on the time it stays in the P2P system. Thus peer availability is defined as follows:

$$p_i = P\left\{X \geq T_i^{stay} + T_{req} \middle| X \geq T_i^{stay}\right\} = \frac{P\left\{X \geq T_i^{stay} + T_{req}\right\}}{P\left\{X \geq T_i^{stay}\right\}}, \tag{2}$$

where a random variable X represents the lifetime of a peer. Eq. (2) gives the probability that a peer who has stayed for T_i^{stay} will survive longer than T_{req} in the system. Here, the lifetime of a peer can be modeled by a Pareto distribution since the distribution of peer lifetime is generally heavy-tailed in real P2P system [28–31]. In our proposed system, a Pareto distribution is adopted for modeling the lifetime of a peer.

Hybrid storage system reliability The hybrid storage system reliability is calculated based on $\vec{p}_{node}$. Two vector matrices are defined. The first matrix, the node combination matrix is characterized by

$$\begin{aligned} \mathbf{A} &= \left(\vec{a}_1 \quad \vec{a}_2 \quad \cdots \quad \vec{a}_i \quad \cdots \quad \vec{a}_{N_{row}}\right)^{\mathrm{T}}, \\ \vec{a}_i &= \left(a_{i,cs},\ a_{i,1},\ a_{i,2}, \cdots,\ a_{i,j}, \cdots,\ a_{i,|\mathbf{U}_{initial\_set}|}\right), \\ N_{row} &= 2 \cdot \sum_{q=1}^{|\mathbf{U}_{initial\_set}|} \binom{|\mathbf{U}_{initial\_set}|}{q}. \end{aligned} \tag{3}$$

In the above equation, the corresponding element in the available node status vector $\vec{a}_i$ is set to one if the cloud storage or peers are consistently available in the system for the time T_{req}; otherwise, it is fixed as zero. The second matrix, the event matrix can be characterized by

$$\begin{aligned} \mathbf{E} &= \left(\vec{e}_1 \quad \vec{e}_2 \quad \cdots \quad \vec{e}_i \quad \cdots \quad \vec{e}_{N_{row}}\right)^{\mathrm{T}}, \\ \vec{e}_i &= \left(e_{i,cs}\left(\vec{n}_{pkt}\right),\ e_{i,1}\left(\vec{n}_{pkt}\right),\ e_{i,2}\left(\vec{n}_{pkt}\right), \cdots,\ e_{i,j}\left(\vec{n}_{pkt}\right), \cdots,\ e_{i,|\mathbf{U}_{initial\_set}|}\left(\vec{n}_{pkt}\right)\right), \\ e_{i,cs}\left(\vec{n}_{pkt}\right) &= \begin{cases} p_{cs} & \text{if } N_{ps} \cdot \left(\vec{a}_i \bullet \vec{n}_{pkt}\right) \geq K_{\min} \text{ and } a_{i,cs} = 1 \\ 1-p_{cs} & \text{if } N_{ps} \cdot \left(\vec{a}_i \bullet \vec{n}_{pkt}\right) \geq K_{\min} \text{ and } a_{i,cs} = 0 \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{4}$$

$$e_{i,j}\left(\vec{n}_{pkt}\right) = \begin{cases} p_j & \text{if } N_{ps} \cdot \left(\vec{a}_i \bullet \vec{n}_{pkt}\right) \geq K_{\min} \text{ and } a_{i,j} = 1 \\ 1-p_j & \text{if } N_{ps} \cdot \left(\vec{a}_i \bullet \vec{n}_{pkt}\right) \geq K_{\min} \text{ and } a_{i,j} = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where • denotes the vector inner product. The multiplication of all components in $\vec{e}_i$ denotes the probability that more than $K_{\min}$ encoded symbols can be obtained when the available node status vector is $\vec{a}_i$. Now, $SR\left(\vec{n}_{pkt}\right)$ is represented by

$$SR\left(\vec{n}_{pkt}\right) = \sum_{i=1}^{N_{row}} \left(e_{i,cs}\left(\vec{n}_{pkt}\right) \cdot \prod_{j=1}^{|\mathbf{U}_{initial\_set}|} e_{i,j}\left(\vec{n}_{pkt}\right) \right). \tag{6}$$

Data retrievability As mentioned earlier, the fountain decoding process may fail even though $K_{\min}$ encoded symbols are obtained. In [23, 32], the relationship among $\delta_{failure}$, $K_{\min}$, and K_{source} is found as follows.

$$K_{\min} = K_{source} + 2 \cdot \ln\left(\frac{\omega \cdot \ln\left(\frac{K_{source}}{\delta_{failure}}\right) \cdot \sqrt{K_{source}}}{\delta_{failure}} \right) \cdot \omega \cdot \ln\left(\frac{K_{source}}{\delta_{failure}}\right) \cdot \sqrt{K_{source}}, \tag{7}$$

where ω is an extra parameter of the robust Soliton distribution with a value less than one. Consequently, in this paper, data retrievability is defined by combining Eqs. (6) and (7).

$$DR\left(\vec{n}_{pkt}\right) = \left(1-\delta_{failure}\right) \cdot SR\left(\vec{n}_{pkt}\right). \tag{8}$$

The above $DR(\overrightarrow{n}_{pkt})$ is the probability to obtain K_{source} source symbols successfully from the retrieved encoded symbols. Now, we can formulate the problem as follows.

Problem formulation Determine $\overrightarrow{n}_{pkt}$ to minimize the total upload time

$$T_{up}\left(\overrightarrow{n}_{pkt},\ \overrightarrow{bw}\right), \tag{9}$$

$$\text{subject to } DR(\overrightarrow{n}_{pkt}) \geq P_{retrive}^{\min}, \tag{10}$$

$$n_{cs}^{pkt} < \left\lceil \frac{K_{source}}{N_{ps}} \right\rceil \text{ and } n_{l}^{pkt} < \min\left\{ \left\lceil \frac{h_l}{S_{symbol} \cdot N_{ps}} \right\rceil, \left\lceil \frac{K_{source}}{N_{ps}} \right\rceil \right\}, \quad for\ 1 \leq l \leq |\mathbf{U}_{initial\_set}|, \tag{11}$$

where Eq. (10) is included in order to avoid the invasion of data privacy since the decoding process is not performed successfully with less than K_{source} encoded symbols. $\lceil x \rceil$ is the smallest integer no less than x.

3.3 Determining algorithm of control parameters

The proposed algorithm consists of two parts that are required to achieve a feasible solution: (1) the packet distribution process to determine the initial peer set and the packet distribution vector with low computational complexity, (2) on-the-fly maintenance process to update the packet distribution vector when there is a significant bandwidth change during transmission. First, we have to determine the initial peer set provided by the bootstrap server. Whenever a service request is received, the bootstrap server has to provide the user with the peer set in which a feasible solution to the given problem exists. However, it is quite complicated for the bootstrap server to always provide the initial peer set for satisfying the constraint in Eq. (10) since peer availability, available bandwidth, and the network environment should be considered. Thus, in the proposed system, a fixed number of randomly selected peers are provided by the bootstrap server. If the constraint in Eq. (10) cannot be satisfied with the given initial peer set,

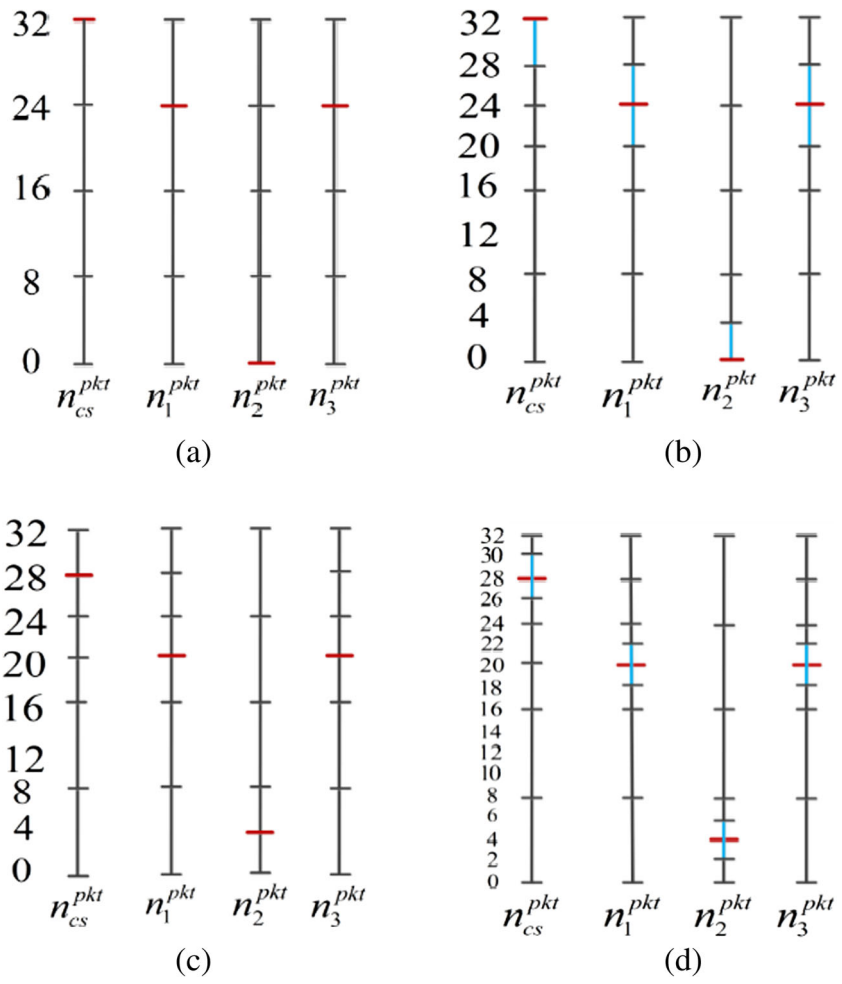


Fig. 3 Example of the dynamic step-size algorithm when N_{int} is 4

the user requests additional peers until an effective solution is obtained.

Next, we control the number of encoded symbols stored on cloud storage and each peer to minimize the upload time while satisfying data retrievability and privacy constraints. Basically, we adopt the **Branch and Bound algorithm (B&B)** [33] to obtain the optimal solution of the given problem with a low computational complexity. The Branch and Bound algorithm is a well-known method for solving optimization problem. It searches through a complete set of solutions to find the optimal one. In the state space tree, the bound value of a vertex is calculated to determine whether the vertex is promising or not. To avoid unnecessary searching, the data retrievability and upload time are calculated at each vertex. If the data retrievability is smaller than $P_{retrive}^{\min}$ or the upload time is longer than the current best solution, the sub-tree of the vertex is pruned. However, this still requires a considerable amount of computation. Thus, we propose the **Branch and Bound algorithm with Dynamic Step Size (B&B with DSS)** for the fast convergence: This algorithm addresses how to obtain a near optimal solution for $\overrightarrow{n}_{pkt}$ with low computational complexity compared to the traditional Branch and Bound algorithm. The B&B with DSS algorithm repeats the B&B algorithm with smaller step size in only the selected interval at the previous iteration. And thus it can significantly reduce the computational complexity by decreasing the number of searching points. To transmit the first

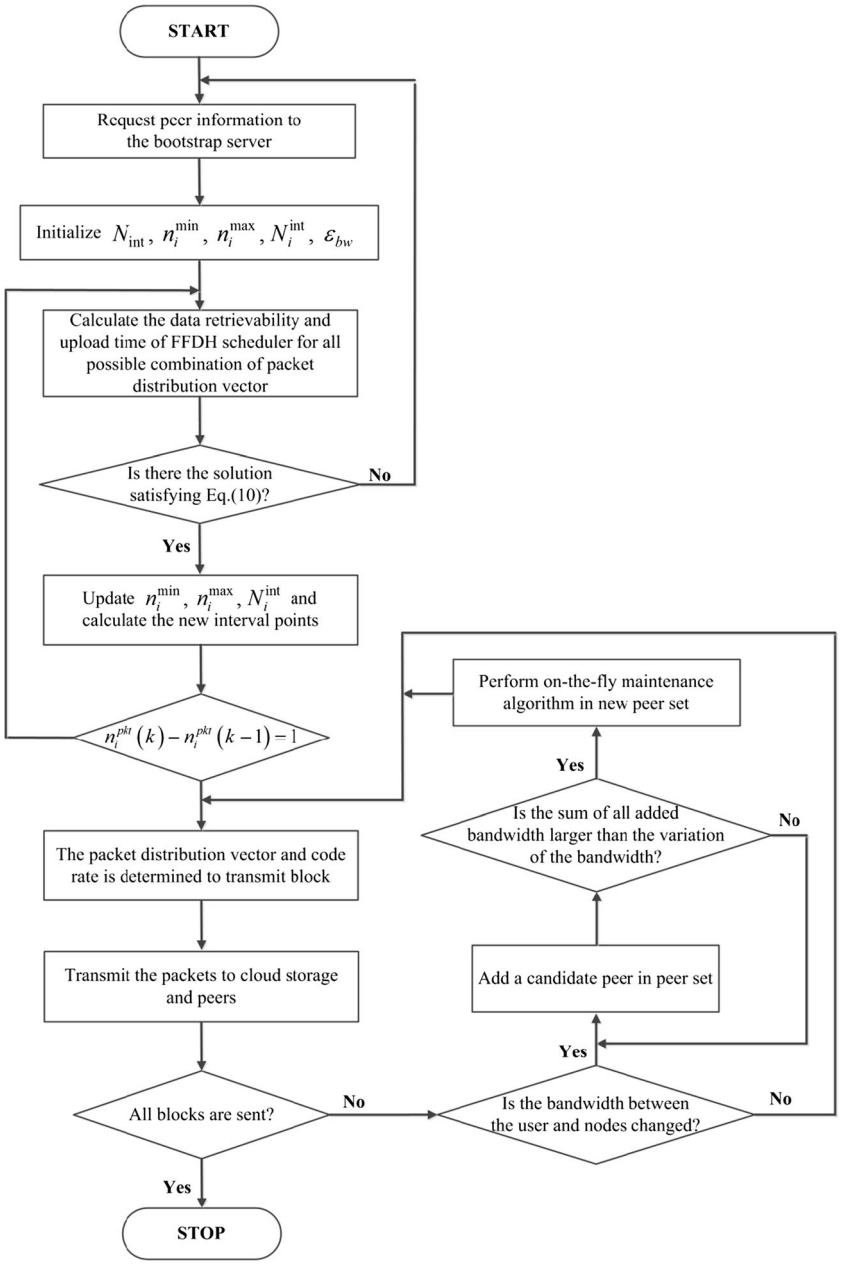


Fig. 4 Flow chart of the determining algorithm of control parameters

block of the file, the packet distribution algorithm is summarized as follows and its structure is shown in Fig. 3.

Step 0: First, set the number of intervals N_{int} to 2^m, where m is an integer between 0 and $\left\lfloor \log_2 \frac{K_{source}}{N_{ps}} \right\rfloor$. For $\forall i (0 \le i \le |\mathrm{U}_{initial\_set}|)$, initialize $n_i^{\min}=0$, $n_i^{\max} = \left\lfloor \log_2 \frac{K_{source}}{N_{ps}} \right\rfloor$, and $N_i^{\text{int}}=N_{\text{int}}$. Then, calculate the interval points using the equation

$$n_i^{pkt}(k) = n_i^{\min} + k \cdot \frac{n_i^{\max} - n_i^{\min}}{N_i^{\text{int}}} \quad \text{for} \quad 0 \le k \le N_i^{\text{int}}, \tag{12}$$

where $n_i^{pkt}(k)$ represents the number of packets that are transmitted to the i_{th} peer. The exception case, $n_0^{pkt}(k)$, represents the number of packets that are assigned to the cloud storage.

Step 1: Calculate the data retrievability and the upload time for all possible combinations of the packet distribution vector determined by Eq. (12) when the FFDH scheduler is adopted. If any combination does not satisfy the constraint in Eq. (10), request for additional peer information from the bootstrap server, and then go back to Step 0.

Step 2: When $\overrightarrow{n}_{pkt}^{cur} = (n_0^{cur}, n_1^{cur}, \ldots, n_i^{cur}, \ldots, n_{|\mathrm{U}_{initial\_set}|}^{cur})$ is the packet distribution vector with the minimum upload time that satisfies data retrievability among all the possible combinations in Step 1, the minimum point $n_i^{\min}$ and the maximum point $n_i^{\max}$ are updated by $\max\left\{0,\ n_i^{cur} - \frac{n_i^{\max}-n_i^{\min}}{2 \cdot N_i^{\text{int}}}\right\}$ and $\min\left\{ n_i^{cur} + \frac{n_i^{\max}-n_i^{\min}}{2 \cdot N_i^{\text{int}}},\ \ 2^{\left\lfloor \log_2 \frac{K_{source}}{N_{ps}} \right\rfloor} \right\}$, respectively. If n_i^{cur} is 0 or $2^{\left\lfloor \log_2 \frac{K_{source}}{N_{ps}} \right\rfloor}$, then N_i^{int} is set to 1; otherwise, N_i^{int} is set to 2.

Table 3 Parameters for our simulation

Parameter	Value
Uplink capacity of the user	100 [Mbps]
The size of initial peer set $\lvert\mathrm{U}_{iniial\_set}\rvert$	10
The size of additional peer set $\lvert\mathrm{U}_{add}\rvert$	100
Availability of cloud storage	0.999999
Source block size	33 [KB]
Symbol size	32 [Byte]
Packet payload size	1024 [Byte]
The number of source symbols in a source block	1056
The minimum number of encoded symbols	1120
Fountain decoding failure rate	9.10^{-10}
ε_{bw}	0.95

Table 4 Information of server and peers in the initial state

Nodes measures	Server	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5
Node availability	0.999999	0.9989	0.9992	0.9988	0.9987	0.9990
Bandwidth [Mbps]	51.21	51.31	47.81	52.01	52.54	48.33

Step 3: Repeat Steps 1~2 until $n_i^{pkt}(k) - n_i^{pkt}(k-1) = 1$.

Step 4: The current packet distribution vector $\overrightarrow{n}_{pkt}^{cur}$ is chosen to transmit the first source block and the code rate is determined by

$$c\left(\overrightarrow{n}_{pkt}^{cur}\right) = \frac{K_{source}}{N_{ps} \cdot \left(\sum_{i=0}^{|\mathrm{U}_{initial\_set}|} n_i^{cur} \right)}. \tag{13}$$

Next, we describe the on-the-fly maintenance process that updates $\overrightarrow{n}_{pkt}$ under time-varying network conditions. The upload time is time-varying when the network conditions change during transmission. The upload time of the proposed system strongly depends on the available bandwidth with the server and peers. However, it is inefficient to perform the packet distribution algorithm with all the participating peers whenever the estimated bandwidth is changed due to the computational complexity. To transmit all blocks except the first block of the file, the maintenance algorithm is summarized as follows.

Step 0: Monitor the available bandwidth vector $\overrightarrow{bw} = (bw_{cs}, bw_1, \ldots, bw_i, \ldots, bw_{|\mathrm{U}_{initial\_set}|})$ to transmit the next block.

Step 1: Check whether a significant bandwidth decrease occurs. If so, select more peers from $\mathbf{U}_{add}$, i.e. if $\frac{bw_{cs}}{bw_{cs}^{pre}} \le \varepsilon_{bw}$ or $\frac{bw_i}{bw_i^{pre}} \le \varepsilon_{bw}$ for $1 \le i \le |\mathbf{U}_{initial\_set}|$ where $\overrightarrow{bw}_{pre} = (bw_{cs}^{pre}, bw_1^{pre}, \ldots, bw_i^{pre}, \ldots, bw_{|\mathrm{U}_{initial\_set}|}^{pre})$ is the previous bandwidth vector and ε_{bw} is a constant ($0 < \varepsilon_{bw} < 1$), then additional peers are

Table 5 Performance comparison of packet distribution algorithms in the first block

Measures algorithms		The number of searching points	Upload time [msec]
Full search		1,291,467,969	5.618
Branch and bound algorithm		58,703,521	5.618
Branch and bound algorithm with dynamic step size	$N_{\text{int}=1}$	415	5.637
	$N_{\text{int}=2}$	796	5.620
	$N_{\text{int}=4}$	4264	5.620
	$N_{\text{int}=8}$	71,602	5.620
	$N_{\text{int}=16}$	1,900,253	5.618

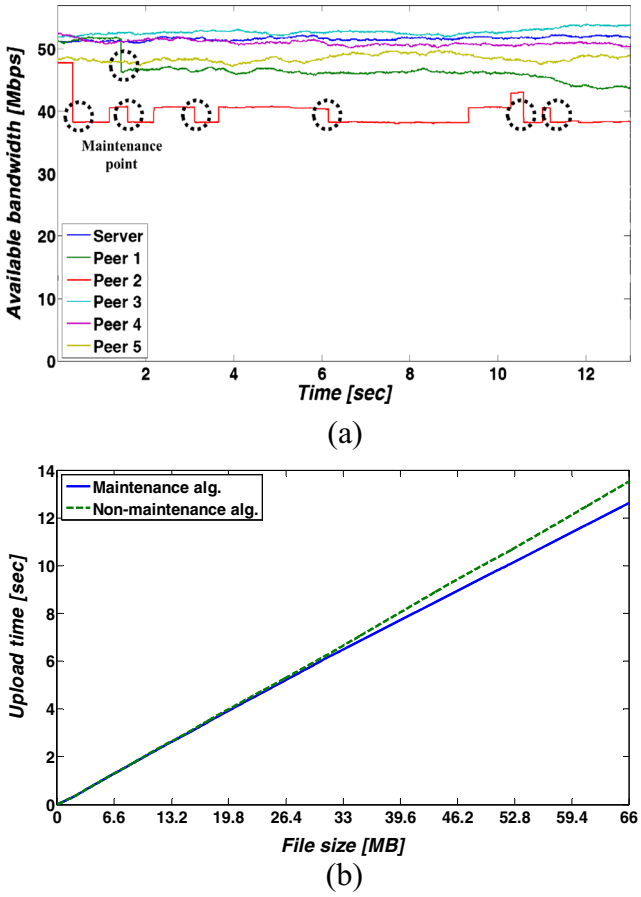


Fig. 5 Performance comparison of maintenance algorithm and non-maintenance algorithm: (**a**) available bandwidth (**b**) The upload time

selected from $\mathbf{U}_{add}$ to $\mathbf{U}_{initial\_set}$ until the sum of bandwidth between the user and additional peers becomes larger than the decreased bandwidth.

Step 2: The packet distribution vector is re-calculated using the B&B with DSS algorithm only for the additional peers and the above server or peers with a serious bandwidth reduction. Set $\overrightarrow{bw}_{pre}$ to $\overrightarrow{bw}$.

Step 3: Repeat Step 0~2 until all the blocks are transmitted.

Figure 4 shows the whole procedure for determining the control parameters.

4 Simulation results

The proposed hybrid storage system is implemented in JAVA to demonstrate the performance of the proposed system. The maximum uplink capacity of the user is set to 100 Mbps [34]. The sizes of $\mathbf{U}_{initial\_set}$ and $\mathbf{U}_{add}$ are set to 10 and 100, respectively. In general, commercial cloud storage systems such as Amazon S3 are designed for 99.9999 % reliability over a year, which means that the average loss rate of data stored is 0.0001 % [35–37]. During the simulation, the cloud storage availability is set to 0.999999 taking into consideration that of commercial cloud storage systems. It is shown in [28–31] that while most peers tend to stay in the system for very short time, some peers take part in the system for long time. Especially, it is observed in [30] that among the peers seen on the first day, about 45 % peers have a lifetime longer than 3 months. Thus, the peer availability is set up by taking into account the dynamics of peer participation: Case 1: only participating peers of high availability are selected and Case 2: a few of participating peers of low availability are also chosen. The peer bandwidth is set up based on the up-to-date OECD broadband statistics since all peers use different access networks, where DSL (Digital Subscriber Line), cable modem, and fiber/LAN occupy 55.8, 30.0, and 13.7 %, respectively [38]. As mentioned earlier, to support data privacy, the number of encoded symbols stored on cloud storage and on each peer is limited to less than the number of source symbols. K_{source} is set to 1056 and $K_{\min}$ is determined to be 1120 by setting $\delta_{failure}$ to $9 \cdot 10^{-10}$ and ω to 0.00165 [32]. The symbol size and the packet payload sizes are set to 32 and 1024 bytes (i.e., each packet consists of 32 encoded symbols) taking into consideration the

Table 6 The number of allocated packets on server and peers

Nodes maintenance time[sec]	Server	Peers					Additional peers							Data retrievability
		1	2	3	4	5	1	2	3	4	5	6	7	
0	32	3	30	0	0	2								0.99999911451
0.33769	32	3	30	0	0	2	0							0.99999911451
1.44874	32	2	30	0	0	2	0	1						0.99999999409
1.59792	32	2	29	0	0	2	0	1	1					0.99999999476
3.1103	32	2	29	0	0	2	0	1	1	0				0.99999999476
6.14707	32	2	24	0	0	2	0	1	1	0	5			0.99999999472
10.5437	32	2	24	0	0	2	0	1	1	0	5	0		0.99999999472
11.1444	32	2	24	0	0	2	0	1	1	0	5	0	0	0.99999999472

Table 7 Information of server and peers in the initial state

Nodes measures	Server	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5	Peer 6	Peer 7	Peer 8	Peer 9	Peer 10
Node availability	0.999999	0.749	0.767	0.760	0.790	0.769	0.728	0.698	0.782	0.750	0.689
Bandwidth [Mbps]	96.12	15.83	16.74	14.66	42.83	14.79	41.35	42.28	14.64	15.70	41.86

fountain coding complexity and the amount of overhead. For the execution of the maintenance algorithm, ε_{bw} is set to 0.95. The detailed simulation environment is set up as shown in Table 3.

4.1 Performance verification of the proposed algorithms

In this section, we examine the performance of packet distribution and the maintenance algorithm for determining control parameters. During the simulation, $P_{retrive}^{\min}$ is set to 0.999999 considering the data retrievability that commercial cloud storage systems provide. First, we show that the B&B with DSS algorithm provides a near optimal solution with a low computation complexity compared to the optimal solution as a benchmark. Since the computational complexities of a full search and the B&B algorithm increase exponentially as the number of peers increases, the simulation is performed with a relatively small number of five peers as shown in Table 4. Table 5 shows that the B&B with DSS algorithm has much smaller searching points in comparison with a full search and the B&B algorithm. Compared to the conventional B&B algorithm, the number of searching points is decreased by approximately 96.75 % when the number of intervals N_{int} is set to 16. Moreover, the computational complexity of the B&B with DSS algorithm generally decreases as the number of intervals N_{int} decreases. On the other hand, the upload time of the B&B with DSS algorithm monotonically increases as the number of intervals N_{int} decreases. Consequently, it can reduce the number of searching points at the expense of a little extra upload time by selecting an adaptive initial interval.

Next, the performance verification of the maintenance algorithm is provided. Figure 5a represents the time-varying wired network states and the operating point for the maintenance algorithm. Table 6 shows the execution results of the maintenance algorithm at the maintenance points in Fig. 5a. The maintenance algorithm adaptively selects additional peers and allocates the number of packets to the selected additional peers as well as peers whose available bandwidth significantly decreases in order to avoid an increase in upload time. As shown in Table 6, $P_{retrive}^{\min}$ is always satisfied in the maintenance process. Figure 5b compares the upload time for the non-maintenance algorithm with that of the maintenance algorithm. The results of the comparison show that the

Table 8 The amount of stored data based on the information in the initial state

Algorithms nodes	Proposed algorithm	LTCS algorithm [19]	Adaptive ERC algorithm [10]	Replication algorithm [13]
Server	32	32	24.75	33
Peer 1	13	6	8.25	
Peer 2	3	6	8.25	
Peer 3	3	6	8.25	
Peer 4	5	6	8.25	
Peer 5	3	6	8.25	
Peer 6	13	6	8.25	
Peer 7	3	6	8.25	
Peer 8	3	6	8.25	
Peer 9	5	6	8.25	
Peer 10	5	6	8.25	

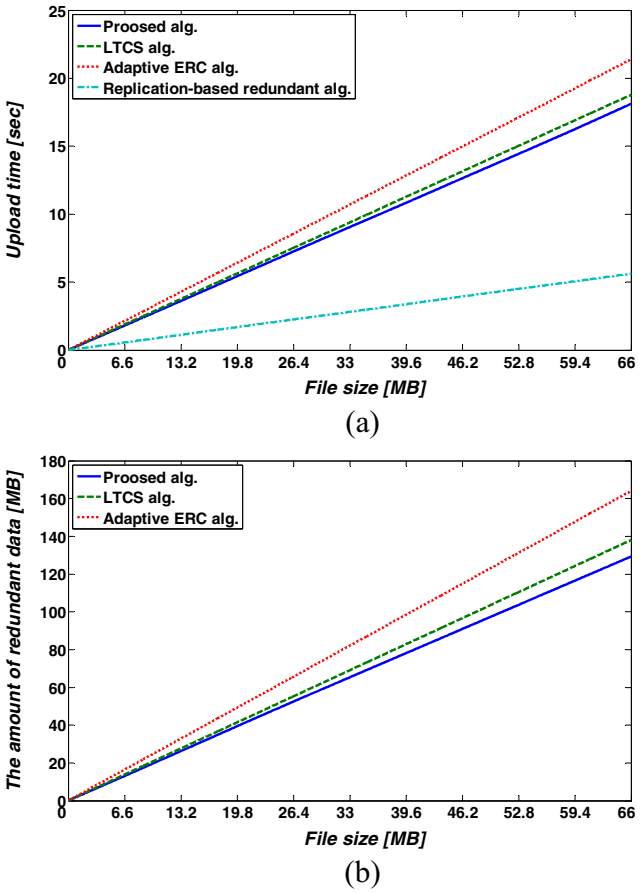
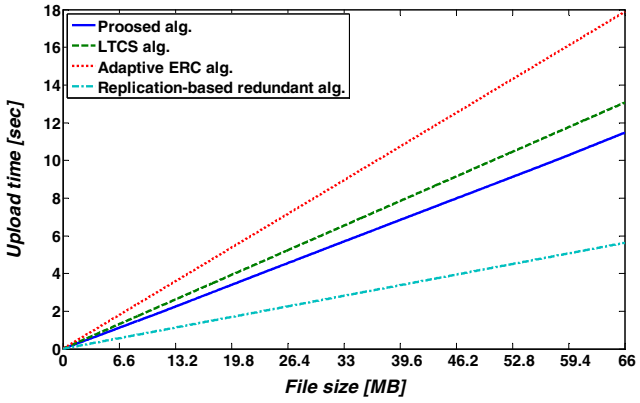
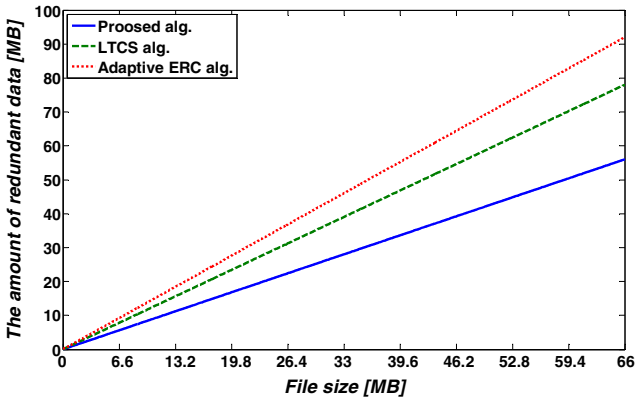


Fig. 6 Performance comparison between the proposed algorithm and other existing algorithms for the environment in Table 7 when $P_{retrive}^{\min}$ is 0.999999: (**a**) the accumulated upload time (**b**) the amount of redundant data



(a)



(b)

Fig. 7 Performance comparison between the proposed algorithm and other existing algorithms for the environment in Table 9 when $P_{retrive}^{\min}$ is 0.999999: (**a**) the accumulated upload time (**b**) the amount of redundant data

maintenance algorithm is effective in decreasing the upload time under time-varying network conditions.

4.2 Performance comparison with other algorithms

In this section, we compare the proposed algorithm with existing algorithms: the LTCS (LT code-based secure cloud storage service) algorithm [19], the adaptive ERC algorithm [10, 12], and a replication-based redundant algorithm [13]. The existing algorithms are briefly summarized below.

- **LTCS algorithm:** To recover all the source symbols from the distributed encoded symbols that can be retrieved from any m combination of peers, ${}^{K_{\min}}/_{m}$ encoded symbols are uniformly distributed to each storage device.
- **Adaptive ERC algorithm:** The appropriate fragment size of an RS code is adaptively chosen to efficiently store data in P2P storage. The encoded fragments are uniformly distributed to multiple participating peers.
- **Replication-based redundant algorithm:** The peer is selected and the replicated data is stored in the selected peer until the data retrievability is satisfied. The user can retrieve the original data from only one replica.

Since these existing algorithms are proposed for P2P or cloud storage systems, they are simply modified for the purpose of conducting a fair comparison with the proposed hybrid storage system. Thus, in LTCS and Adaptive ERC algorithm, the cloud storage stores the maximum amount of encoded data that the original data cannot be perfectly decoded without peer support in order to avoid the invasion of data privacy.

During the simulation, $P_{retrive}^{\min}$ is set to 0.999999 and 0.9999999 to verify the enhancement of data retrievability. The upload time and the amount of redundant data are measured to provide a comparison of the performance with that of existing algorithms.

First, we compare the proposed algorithm with existing algorithms when $P_{retrive}^{\min}$ is 0.999999. Table 7 shows the initial bandwidth and the availability of server and peers. Further, Table 8 shows the amount of data stored in the first block based on the information given in Table 7. As shown in Table 8, we can see that the original data cannot be perfectly decoded at any node when fountain codes are adopted since the amount of stored data at each node is smaller than the source block size of 33 KB.

Figure 6 shows the upload time and the amount of total redundant data stored on the server and peers for the environment given in Table 7. As shown in Fig. 6, the proposed algorithm exhibits the best performance among all the algorithms with respect to upload time and storage efficiency except for replication algorithm. As the amount of transmitted file grows, the performance improvement of the upload time becomes more obvious. The upload time is reduced by about 3.5 and 15.3 % compared to LTCS and adaptive ERC algorithm, respectively. The amount of redundant data is also reduced by 6.3 and 21.2 % compared to LTCS and adaptive ERC algorithm, respectively. Data privacy cannot be supported by the replication-based redundant algorithm because the original data is stored entirely on the cloud storage. In addition, redundant data is not required in the replication algorithm since data retrievability is provided by the cloud storage itself.

Table 9 Information of server and peers in the initial state

Nodes measures	Server	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5	Peer 6	Peer 7	Peer 8	Peer 9	Peer 10
Node availability	0.999999	0.902	0.933	0.920	0.905	0.904	0.874	0.899	0.925	0.899	0.923
Bandwidth [Mbps]	96.12	15.83	16.74	14.66	42.83	14.79	41.35	42.28	14.64	15.70	41.86

Table 10 Information of server and peers in the initial state

Nodes Measures	Server	Peer 1	Peer 2	Peer 3	Peer 4	Peer 5	Peer 6	Peer 7	Peer 8	Peer 9	Peer 10
Node availability	0.999999	0.155	0.933	0.920	0.294	0.904	0.298	0.178	0.925	0.150	0.923
Bandwidth [Mbps]	96.12	15.83	16.74	14.66	42.83	14.79	41.35	42.28	14.64	15.70	41.86

Figure 7 represents the performance comparisons as peer availability increases, when $P^{\min}_{retrive}$ is 0.999999. As shown in Table 9, the network conditions are the same as in the previous environment (Table 7). As the peer availability increases, the amount of redundant data decreases, since $P^{\min}_{retrive}$ can be satisfied by a relatively lower amount of redundant data. This implies that the performance of storage efficiency improves as the peer availability increases. As shown in Fig. 7a, the upload time is also less than that of the previous results since the amount of transmitted redundant data decreases. The upload time is decreased by 12.2 and 35.9 % compared to LTCS and adaptive ERC algorithm, respectively. The amount of redundant data is also decreased by 28.2 and 39.1 % compared to LTCS and adaptive ERC algorithm, respectively.

As shown in Table 10, a few of peers in Table 9 are replaced with peers of low availability. Figure 8 shows the upload time and the amount of total redundant data with the participating peers in Table 10. It is apparently observed that both upload time and the amount of redundant data are increased compared with Fig. 7. In this case, the upload time is reduced by 8.4 and 19.6 % compared to LTCS and adaptive ERC algorithm, respectively. The amount of redundant data is also reduced by 15.1 and 28.5 % compared to LTCS and adaptive ERC algorithm, respectively. In summary, as peer availability increases, the upload time and the amount of redundant data decrease, and vice versa. When only peers of low availability are participating as an extreme case, the gain of the proposed system may be decreased. However, in a community having the same interest and purpose, peers are willing to stay at the session for long time and share their resources with other peers. In this situation, the achievement of the proposed system can be significantly improved.

Next, we examine the performance of the proposed algorithm as $P^{\min}_{retrive}$ increases. By efficiently utilizing the P2P storage system, the proposed hybrid storage system can guarantee a higher data retrievability than that of the cloud storage system. $P^{\min}_{retrive}$ is set to 0.9999999, which is ten times lower than the average loss rate of a commercial cloud storage system. The network conditions and the peer availability are the same as given in Table 9. The results are summarized in Fig. 9. Since a higher amount of redundant data is required to satisfy higher data retrievability, both the upload time and the amount of redundant data increase as compared to Fig. 7. Compared to the Figs. 7 and 9, the performance of the replication algorithm deteriorates in the current scenario. This is because the replicated data has to be stored on the peers as well as on cloud storage. As shown in the Fig. 9, the upload time is decreased by 6.3, 34.2, and 64.7 % compared to LTCS, adaptive ERC, and replication-based redundant algorithm, respectively. The amount of redundant data is decreased by 24.5 and 32.7 % compared to LTCS and adaptive ERC algorithm, respectively.

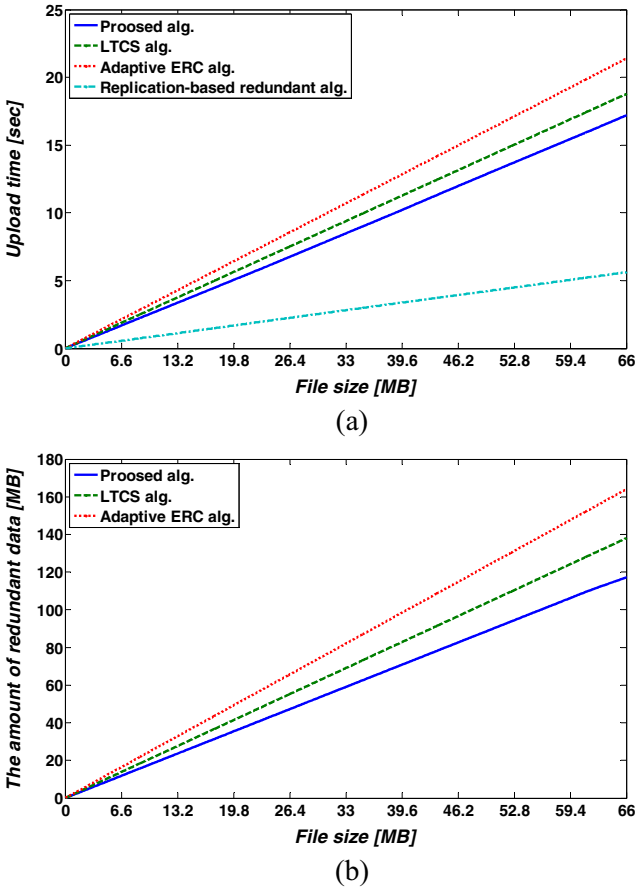


Fig. 8 Performance comparison between the proposed algorithm and other existing algorithms for the environment in Table 10 when $P^{\min}_{retrive}$ is 0.999999: (**a**) the accumulated upload time (**b**) the amount of redundant data

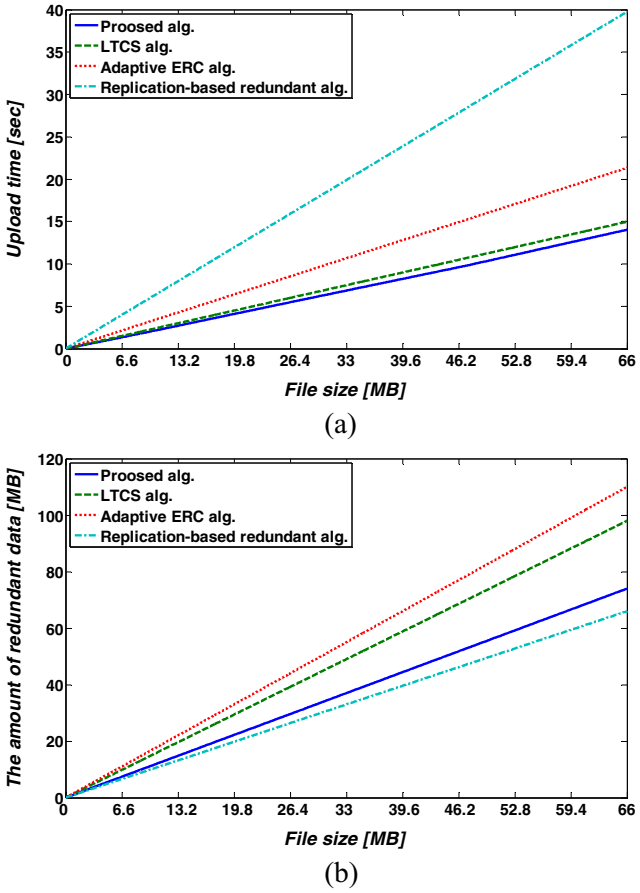


Fig. 9 Performance comparison between the proposed algorithm and other existing algorithms for the environment in Table 9 when $P_{retrive}^{min}$ is 0.9999999: (**a**) the accumulated upload time (**b**) the amount of redundant data

5 Conclusion

An effective hybrid P2P and cloud storage system has been proposed in this paper. There are several significant advantages with hybrid P2P and cloud storage system. The proposed system can satisfy a higher data retrievability requirement than that of the conventional cloud storage by controlling the amount of distributed fountain encoded symbols to cloud storage system and participating peers. And the proposed system is able to prevent others from reading the data by individually regulating the amount of fountain encoded symbols stored at cloud storage system and participating peers. Furthermore, the proposed system can decrease the data upload time by considering the network conditions. It has been observed during the simulation that the proposed algorithm enables the fast completion of data upload transmission while satisfying the required data retrievability and enhancing the privacy of user data, i.e. in the given simulation environments, the upload time of the proposed algorithm is reduced by up to 12.2, 35.9, and 64.7 % compare to LTCS, adaptive ERC, and replication-based redundant algorithm, respectively.

Acknowledgement This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2013R1A1A2006732) and the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014.

References

1. Caceres J, Vaquero LM, RoderoMerino L, Polo A, Hierro JJ (2010) Service scalability over the cloud. Handb Cloud Comput 357–377
2. Amazon Glacier. Available: http://aws.amazon.com/ec2/reserved-instances/#3. Accessed 18 Oct 2013
3. Google Drive. Available: http://drive.google.com/. Accessed 18 Oct 2013
4. Microsoft SkyDrive. Available: http://skydrive.live.com/. Accessed 18 Oct 2013
5. Baca S (2010) Cloud Computing: What it is and what it can do for you (pp. 1–6). www.globalknowledge.com/. Accessed 18 Oct 2013
6. Dropbox. Available: http://dropbox.com/. Accessed 18 Oct 2013
7. Harihara SG, Janakiram B, Chandra MG, Aravind KG, Kadhe S, Balamuralidhar P, Adiga BS (2010) SpreadStore: a LDPC erasure code scheme for distributed storage system. Int Conf Data Storage Data Eng 154–158
8. Spoor R, Peddemors A (2010) Cloud storage and peer-to-peer storage. [Online]. Available: http://www.surf.nl/binaries/content/assets/surf/en/knowledgebase/2010/EDS-3R+Cloud+and+p2p+storage-v1.1.pdf/. Accessed 18 Oct 2013
9. iSuppli. Available: http://isuppli.com/. Accessed 18 Oct 2013
10. Li J, Huang Q (2006) Erasure resilient codes in peer-to-peer storage cloud. IEEE Int Conf Acoust Speech Signal Process 4:4
11. Gaidioz B, Koblitz B, Santaos N (2007) Exploring high performance distributed file storage using LDPC codes. Parallel Comput 33:264–274
12. Li J (2006) Adaptive erasure resilient coding in distributed storage. IEEE Int Conf Multimedia Expo 561–564
13. Rodrigues R, Liskov B (2005) High availability in DHTs: erasure coding vs. replication. Peer-to-Peer Syst IV 3640:226–239
14. Kim S, Lee S (2009) Rateless erasure resilient codes for content storage and distribution in P2P networks. 11th International Conference on Advanced Communication Technology 1:444–446
15. Ji W, Jian Z, Tong W, Qian S (2011) Study on redundant strategies in peer to peer cloud storage systems. Appl Math Inf Sci Int J 5-2S: 235S–242S
16. Park S, Moon B, Park M (2004) Design, implementation, and performance analysis of the remote storage system in mobile environment. 2nd International Conference on Information Technology for Application
17. Hertel CR (2003) Implementing CIFS: the common internet file system. Prentice Hall, Englewood Cliffs
18. Radkov P, Yin L, Goyal P, Sarkar P, Shenoy P (2004) "A Performance comparison of NFS and iSCSI for IP-networked storage", 3rd USENIX conference on file and storage technologies. USENIX Assoc 3640:101–114
19. Cao N, Yu S, Yang Z, Lou W, Hou YT (2012) LT code-based secure and reliable cloud storage service. IEEE INFOCOM 693–701
20. Blömer J, Kalfane M, Karpinski M Karp R, Luby M, Zuckerman D (1995) An XOR-based erasure-resilient coding scheme. ICSI Technical Report No. TR–950048
21. Luby M (2002) LT codes. Ann Symp Found Comput Sci 271–280. doi:10.1109/SFCS.2002.1181950. Accessed 18 Oct 2013

22. Han S, Joo H, Lee D, Song H (2011) An end-to-end virtual path construction system for stable live video streaming over heterogeneous wireless networks. IEEE J Sel Areas Commun 29:1032–1041
23. Shokrollahi A (2006) Raptor codes. IEEE Trans Inf Theory 52(6): 2551–2567. doi:10.1109/TIT.2006.874390. Accessed 18 Oct 2013
24. Xu Q, Stanković V, Xiong Z (2007) Distributed joint source-channel coding of video using Raptor codes. IEEE J Sel Areas Commun 25: 851–861
25. Ribeiro V, Riedi R, Baraniuk R, Navratil J, Cottrell L (2003) pathChirp: efficient available bandwidth estimation for network paths. Passive and Active Measurement Workshop. doi:10.2172/813038. Accessed 18 Oct 2013
26. Ntene N, Vuuren JHV (2009) A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem. Discret Optim 6:174–188
27. Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: a survey. Eur J Oper Res 141:241–252
28. Bustamante FE, Qiao Y (2004) Friendships that last: peer lifespan and its role in P2P protocols. Int Work Web Content Caching Distrib 233–246
29. Stutzbach D, Rejaie R (2006) Understanding churn in peer-to-peer network. The 6th ACM SIGCOMM Conference on Internet Measurement, pp. 189–202
30. Steiner M, En-Najjary T, Biersack E (2009) Long term study of peer behavior in the KAD DHT. IEEE/ACM Trans Netw 17(6):1371–1384. doi:10.1109/TNET.2008.2009053. Accessed 18 Oct 2013
31. Leonard D, Yao Z, Rai V, Loguinov D (2007) On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. IEEE/ACM Trans Networking 15:644–656
32. Mackay DJC (2005) Fountain codes. IEE Proc Commun 152:1062–1068
33. Lawler EW, Wood DE (1966) Branch–and–bound method—a survey. Oper Res 14:669–719
34. Lunttila T, Lindholm J, Pajukoski K, Tiirola E, Toskala A (2007) EUTRAN uplink performance. Int Symp Wirel Pervasive Comput. doi:10.1109/ISWPC.2007.342658. Accessed 18 Oct 2013
35. Turner WP, IV, Seader JH, Renaud V, Brill KG (2006) Tier classifications define site infrastructure performance. White Paper, The Uptime Institute
36. Ping L, Ge X, Wang Y, Fu J (2010) Cloud storage as the infrastructure of cloud computing. Intell Comput Cogn Inform380–383
37. Yang Y, Yuan D (2011) A novel cost-effective dynamic data replication strategy for reliability in cloud data centers. IEEE Dependable Auton Secure Comput
38. OECD Broadband Portal (2012). http://www.oecd.org/sti/ict/broadband. Accessed 18 Oct 2013

Gi Seok Park Gi Seok Park received the B.S. degree in Electrical Engineering from Dongguk University in 2010 and the M.S. degree in IT Convergence Engineering from POSTECH (Pohang University of Science and Technology) in 2013. Currently, he is a Ph. D. student in Division of IT Convergence Engineering, POSTECH. His research interests include P2P networking, cloud computing and future network.

Dr. Hwangjun Song received the B.S. and M.S. degrees from Dept. of Control and Instrumentation (EE), Seoul National University, Korea in 1990 and 1992, respectively, and Ph.D. degree in Electrical Engineering-Systems, University of Southern California, Los Angeles, CA, USA in 1999. From 1995 to 1999, he was a research assistant in SIPI (Signal and Image Processing Institute) and IMSC (Integrated Media Systems Center), Univ. of Southern California. From 2000 to 2005, he was an assistant professor/vice dean of admission affairs at Hongik University, Seoul, Korea. Since Feb. 2005, he has been with Dept. of Computer Science and Engineering, POSTECH (Pohang University of Science and Technology), Korea. He received Haedong Outstanding Paper Award from Korean Institute of Communication Science in 2005. He is an editorial board member of Journal of Visual Communication and Image Representation and an associate editor of Journal of Communications and Networks, and served as an editorial board member of International Journal of Vehicular Technology and a guest editor of Special Issue on "Network technologies for emerging broadband multimedia services" in the Journal of Visual Communication and Image Representation and Special Issue on "Wireless & Mobile networks" in the International Journal of Ad Hoc and Ubiquitous Computing His research interests include multimedia signal processing and communication, image/video compression, digital signal processing, network protocols necessary to implement functional image/video applications, control system and fuzzy-neural system.