Tree-structured Data Regeneration with Network Coding in Distributed Storage Systems

Jun Li, Shuang Yang, Xin Wang, Xiangyang Xue
School of Computer Science
Fudan University, China
{0572222, 06300720227, xinw, xyxue}@fudan.edu.cn

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto, Canada
bli@eecg.toronto.edu

***Abstract*—Distributed storage systems, built on peer-to-peer networks, can provide large-scale data storage and high data reliability by redundant schemes, such as replica, erasure codes and linear network coding. Redundant data may get lost due to the instability of distributed systems, such as permanent node departures, hardware failures, and accidental deletions. In order to maintain data availability, it is necessary to regenerate new redundant data in another node, referred to as a newcomer. Regeneration is expected to be finished as soon as possible, because the regeneration time can influence the data reliability and availability of distributed storage systems. It has been acknowledged that linear network coding can regenerate redundant data with less network traffic than replica and erasure codes. However, previous regeneration schemes are all star-structured regeneration schemes, in which data are transferred directly from existing storage nodes, referred to as providers, to the newcomer, so the regeneration time is always limited by the path with the narrowest bandwidth between newcomer and provider, due to bandwidth heterogeneity.**

In this paper, we exploit the bandwidth between providers and propose a tree-structured regeneration scheme using linear network coding. In our scheme, data can be transferred from providers to the newcomer through a regeneration tree, defined as a spanning tree covering the newcomer and all the providers. In a regeneration tree, a provider can receive data from other providers, then encode the received data with the data this provider stores, and finally send the encoded data to another provider or to the newcomer. We prove that a maximum spanning tree is an optimal regeneration tree and analyze its performance. In a trace-based simulation, the results show the tree-structured scheme can reduce the regeneration time by 75%-82% and improve data availability by 73%-124%.

***Index Terms*—Distributed Storage System, Linear Network Coding, Maximum Spanning Tree.**

I. Introduction

Distributed storage systems store data in a large number of storage nodes, either in the context of data centers in cloud computing systems, or in the context of peer-assisted online storage systems *e.g.,* [1]. Due to the inherent lack of reliability caused by node departures and hardware failures, data may become temporarily or permanently unavailable in such systems. Concerns about Quality of Service (QoS) in storage systems hinge upon two aspects: the *reliability* and *availability* of data. Data are reliable when data saved in the distributed storage system are sufficient to recover the original data. Data are available when there are enough active nodes in the distributed storage system so that the original data can be recovered at once. In order to provide high data reliability and availability, distributed storage systems usually use redundant data. The forms of redundant data include replica, erasure codes and linear network coding.

Redundant data can provide higher availability because there can be more active storage nodes for data recovery, when there may be nodes temporarily unavailable. However, when data are lost permanently in the distributed storage system, the number of storage nodes will decrease gradually. Therefore it is necessary to regenerate new redundant data to maintain data availability. Regeneration is the process that a node in the distributed storage system, referred to as a newcomer, receives data from active storage nodes, referred to as providers, and finally becomes a new storage node, so that the lost redundant data are regenerated.

To ensure data reliability and availability, we expect the regenerate time to be as little as possible. The less time regeneration costs, the more redundant data can be preserved in the distributed storage system with data loss. The newcomer or the provider may also leave the system even during the regeneration process, so less regeneration time can result in higher probability that the regeneration is finished before any node (newcomer or provider) leaves the system. The simplest way to reduce the regeneration time is to reduce the network traffic in the regeneration. Dimakis et al. [2] showed that linear network coding can incur less regeneration traffic and the corresponding encoding scheme is given in [3].

To our knowledge, previous regeneration schemes mainly focused on how to generate redundant data to reduce the regeneration traffic, but the bandwidth capacity between nodes has not been taken into account. In this paper, we propose a tree-structured regeneration scheme based on linear network coding from the perspective of bandwidth capacity. Conventional regeneration is a star-structured scheme, i.e. the newcomer downloads data directly from providers. Thus the regeneration time is limited by the path between the newcomer and the provider with the narrowest bandwidth, if the network of the storage system suffers from bandwidth heterogeneity. In our tree-structured scheme, we define a regeneration tree as a spanning tree covering the newcomer and all the providers. In the regeneration tree, the child node sends data to its parent node, and the parent node encodes the received data with the data it stores and then sends the encoded data to its

parent node. If the transmission is pipelined, the bandwidth bottleneck is the edge with the narrowest bandwidth in the tree. We prove a maximum spanning tree is an optimal regeneration tree.

In this paper, we present the tree-structured regeneration scheme and analyze its performance mathematically. We first show how the tree-structured scheme regenerates redundant data at the newcomer. Then we prove a maximum spanning tree is an optimal regeneration tree. By analysis based on probability theory and order statistics, we show our scheme can reduce the regeneration time by improving the transmission rate, and can improve the adaptability to the bandwidth heterogeneity, while not increasing the regeneration traffic. We evaluate our scheme by a trace-based simulation. The simulation results show that our scheme can reduce regeneration time by 75%-82% and improve data availability by 73%-124%.

The remainder of the paper is organized as follows. In Section II we introduce the related work. We introduce some basic concepts of distributed storage systems using linear network coding and present the network model in Section III. In Section IV, we present the tree-structured regeneration scheme and analyze its performance. We show the simulation results in Section V. Finally, Section VI concludes this paper.

II. Related Work

Many papers discussed how to improve data reliability from the perspective of redundant data. The forms of redundant data include replica, erasure codes and linear network coding. Some distributed storage systems use replica, such as BitVault [4]. In OceanStore [1], however, the original data are encoded at the source node by erasure codes. Lin et al. [5] investigated and compared some decentralized replication algorithms for improving file availability in P2P network. Compared with replica, erasure codes provide higher data availability, because in the storage systems using $(n,k)-$erasure codes, any k nodes of n storage nodes are sufficient to recover the original data. However, erasure codes incur more storage space at the source node than replica, when disseminating the encoded data [6]. What's more, Rodrigues et al. [7] pointed out that in some cases, the benefits from erasure codes might not be worth its disadvantages.

Ahlswede et al. [8] introduced the idea of network coding that the intermediate nodes can encode the data they have received and send out the encoded data. It has been proved that network coding can utilize the network resource optimally. Yang et al. [9] presented a file sharing scheme based on network coding, which used the combination network as the network topology. Taking $(n,k)-$linear network coding for example, the data are divided into k blocks, $F_i, i = 1, 2, \ldots, k$ and n encoded blocks, $B_1, B_2, \ldots, B_n, n > k$, are generated as linear combinations of $F_1, F_2, \ldots, F_k$ on Galois Field $\mathbb{F}_q$, where q is the size of the Galois Field. $B_j = \sum_{i=1}^{k} \alpha_{ji} F_i$, $\alpha_{ji} \in \mathbb{F}_q$, where $(\alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jk})^T$ is a coefficient vector, $j = 1, 2, \ldots, k$. When a node wishes to access the original data, it has to receive m encoded blocks, $m \geq k$. Then decoding becomes solving a linear system with k unknowns and m equations. The m encoded blocks can be decoded if and only if the linear system is solvable, i.e. k of the m coefficient vectors are linear independent. Random linear coding [10] is a form of linear network coding, which encodes data at the intermediate node linearly using randomly generated coefficient vector. If all the coefficient vectors are randomly generated, more than k encoded blocks may be required to decode. However, when q is large enough, any k encoded blocks are sufficient to decode with high probability [11]. Accendanski et al. [6] compared the performance of different forms of redundant data, including replica, erasure codes and random linear coding. They showed random linear coding provided data availability no worse than erasure codes, but saved storage cost at the source node when disseminating data into the network.

For different forms of redundant data, the regeneration mechanisms are different. For replica, the newcomer only needs to download one replica from one active storage node. Chun et al. [12] proposed a Carbonite replication algorithm to schedule the regeneration of new replica. For erasure codes and linear network coding, every bit of new data is encoded from the data stored in the providers, so it will incur more network traffic than replica. The simplest way is to recover the original data from providers and encode the original data into a new block. Duminuco et al. [13] proposed a new class of erasure codes, aiming to achieve the tradeoff between regeneration traffic and data reliability. Dimakis et al. showed that linear network coding can reduce the network traffic in the regeneration than erasure codes [2]. They proposed Regeneration Codes, a new form of linear network coding, which achieved the optimal tradeoff between storage cost and network traffic. Wu et al. [3] showed further analysis of the relation between storage cost and network traffic, and presented a construction method of Regeneration Codes.

Previous works mainly considered the form of redundant data and tried to reduce the regeneration traffic, but did not take the bandwidth capacity between two nodes into account. Lee et al. [14] proposed a bandwidth-aware routing scheme in overlay networks, which measured bandwidth capacity between hosts in the overlay networks and selected the best paths so as to bypass the problematic path in the networks. In this paper, we will consider the bandwidth heterogeneity and propose a tree-structured regeneration scheme to reduce the regeneration time and hence to improve data availability. The primary part of our work can be found in [15].

III. Preliminaries

A. Node and Redundant Data

A distributed storage system provides its service based on a distributed network containing a large number of nodes, which may play different roles in the system. A source node is a node which sends data into other nodes, and a storage node is a node which stores data for source nodes. In some distributed storage systems, one node may function as a source node as well as a storage node at the same time. When a source node wishes to

save data into the storage system, it generates redundant data and sends them to one or more storage nodes. For replica, the storage node stores one replica of the original file. For erasure codes, the original file is divided into a number of blocks. Redundant blocks are generated at the source node by erasure codes, such as Reed-Solomon codes and fountain codes. Each storage node stores one redundant block.

In a distributed storage system using linear network coding, each block saved in the distributed storage system is generated by linear network coding. We take $(n,k)-$linear network coding for example. The source node divides the original data into k blocks, $F_1, F_2, \ldots, F_k$, and encodes them into n encoded blocks, $B_1, B_2, \ldots, B_n$, which are all linear combinations of $F_1, F_2, \ldots, F_k$. The coefficient vector of B_i is $(a_{i1}, a_{i2}, \ldots, a_{ik})^T$, $a_{ij} \in \mathbb{F}_q$, $j = 1, 2, \ldots, k$, $i = 1, 2, \ldots, n$, where q is the size of the Galois field $\mathbb{F}_q$. Thus we can get

$$\begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{pmatrix} \cdot \begin{pmatrix} F_1 \\ \vdots \\ F_k \end{pmatrix} = \begin{pmatrix} B_1 \\ \vdots \\ B_n \end{pmatrix}. \quad (1)$$

The coefficient vectors form an encoding matrix C,

$$C = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{pmatrix}. \quad (2)$$

A download node is a node which wishes to access data saved in the distributed storage system. For replica, the download node needs to download data from only one storage node. For $(n,k)-$erasure codes or $(n,k)-$linear network coding, the download node can recover data as soon as it has received k redundant blocks or k linear independent encoded blocks respectively. For linear network coding, we assume the k encoded blocks are $B'_1, B'_2, \ldots, B'_k$, $\{B'_1, B'_2, \ldots, B'_k\} \subset \{B_1, B_2, \ldots, B_n\}$. Let C' be the encoding matrix formed by the coefficient vectors of $B'_1, B'_2, \ldots, B'_k$. Then decoding becomes a linear transformation as follows:

$$\begin{pmatrix} F_1 \\ \vdots \\ F_k \end{pmatrix} = C'^{-1} \begin{pmatrix} B'_1 \\ \vdots \\ B'_k \end{pmatrix}. \quad (3)$$

If the coefficient vectors are randomly generated, i.e. the system uses random linear coding, the encoding matrix C' is non-singular with high probability when q is large enough [11]. Conventionally $q = 2^8$, so the encoded block can be generated byte by byte, and it is guaranteed with very high probability that any k encoded blocks are sufficient to decode.

B. Regeneration

In distributed storage systems, redundant data can improve data availability and provide data reliability. However, it cannot guarantee data reliability and availability forever. Data saved in a storage node may get lost due to accidental deletions, hardware failures, or permanent node departures. Therefore if data loss is detected in the storage system by a data loss detection mechanism, such as Carbonite algorithm proposed in [12], the distributed storage system will generate new redundant data and save them into another node, referred to as a newcomer.

During the regeneration, the newcomer must receive data from one or more existing storage nodes to become a new storage node. We define providers as storage nodes providing data for the newcomer in the regeneration. For replica, the newcomer needs only one provider. For erasure codes, the newcomer must recover the original data from providers and then encode the original data into a new redundant block. For $(n,k)-$linear network coding, the newcomer also needs to receive data from at least k providers. However, the newcomer can directly generate a new encoded block. We assume there are k providers. The k encoded blocks they store are $B'_1, B'_2, \ldots, B'_k$. The encoding matrix formed by the coefficient vectors of $B'_1, B'_2, \ldots, B'_k$ is C'. Similar to decoding, generating a new encoded block is also a linear transformation, if C' is non-singular. Let the coefficient vector of the new encoded block is $(\sigma_1, \sigma_2, \ldots, \sigma_k)^T$, $\sigma_j \in \mathbb{F}_q$, $j = 1, 2, \ldots, k$. We assume that

$$(\sigma_1 \ \cdots \ \sigma_k) \begin{pmatrix} F_1 \\ \vdots \\ F_k \end{pmatrix} = (r_1 \ \cdots \ r_k) \begin{pmatrix} B'_1 \\ \vdots \\ B'_k \end{pmatrix}, \quad (4)$$

where $r_j \in \mathbb{F}_q$, $j = 1, 2, \ldots, k$. According to Eq. (3),

$$(r_1 \ \cdots \ r_k) = (\sigma_1 \ \cdots \ \sigma_k)\, C'^{-1}. \quad (5)$$

If C' is non-singular, $(r_1 \ \cdots \ r_k)^T$ is a random vector if and only if $(\sigma_1 \ \cdots \ \sigma_k)^T$ is a random vector. For random linear coding, $(r_1 \ \cdots \ r_k)^T$ can be randomly generated rather than computed according to Eq. (5). Therefore the newcomer can encode $B'_1, B'_2, \ldots, B'_k$ directly into a new encoded block by the coefficient vector $(r_1 \ \cdots \ r_k)^T$.

Dimakis et al. [2] and Wu et al. [3] analyzed the lower bound of network traffic in the regeneration for distributed storage systems using linear network coding. If the size of the original data is M bytes, and each storage node stores $\frac{M}{k}$ bytes, the minimal regeneration traffic is $\frac{d}{k(d-k+1)}M$ bytes if d providers are required, otherwise the new encoded block will not be equivalent to other encoded blocks in decodability. It is clear that the regeneration scheme showed in Eq. (4) and Eq. (5) achieves the optimal regeneration traffic when the number of providers is k.

C. Network Model

In this paper, we focus on how to regenerate new redundant data quickly using linear network coding. The distributed storage system uses $(n,k)-$linear network coding, $n > k$. Thus the newcomer requires at least k providers in the regeneration. It will become more difficult to find more providers in order to start the regeneration, and the regeneration is more likely to be interrupted by node departures, when it requires more providers. Therefore in this paper, we only discuss the case that the regeneration scheme requires k providers. A regeneration scheme can transfer data from k providers to the

newcomer, regenerate new redundant data and save them at the newcomer. Different from conventional schemes, we also consider bandwidth heterogeneity in the network.

Assume that one file has been saved in the distributed storage system. The size of the original file is M bytes. Each storage node stores an encoded block of $\frac{M}{k}$ bytes. Our network model focuses on the regeneration in the system. In one regeneration, we assume k active storage nodes are required as providers. The node set $V = \{V_0, V_1, \ldots, V_k\}$, where V_0 is the newcomer, and $V_1, \ldots, V_k$ are providers. V_0 receives data from the k providers. Node departures are ignored, i.e. the newcomer and providers are assumed to be stable during the regeneration process. Edge set $E = \{(V_i, V_j)|i, j = 0, 1, \ldots, k, i < j\}$. $\omega(V_i, V_j)$ denotes the bandwidth capacity between V_i and V_j. Thus the weighted undirected complete graph $G_k = (V, E, \omega)$ denotes the network model of the regeneration, where k is the number of providers, i.e. $k = |V| - 1$.

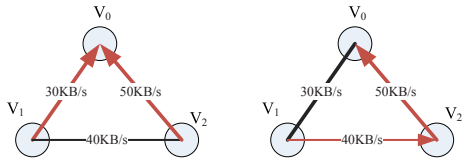


Fig. 1. Comparison between the star-structured and the tree-structured regeneration scheme in an example of the network model containing 3 nodes.

Fig. 1 shows an example of the network model described above. $(n, 2)-$linear network coding is employed in this model, $n > 2$. When a regeneration starts, the newcomer receives data from 2 storage nodes. Conventionally, the newcomer receives data from each provider directly. Fig. 1(a) illustrates the conventional regeneration scheme. In this regeneration scheme, the topology of newcomer and providers is like a star, so this scheme is referred to as a star-structured regeneration scheme in this paper. In Fig. 1(a), the newcomer V_0 receives encoded blocks directly from V_1 and V_2 and then encodes them again to obtain an encoded block with a new coefficient vector. In the star-structured regeneration scheme, the regeneration time depends on the minimal edge connecting to the newcomer V_0. In Fig. 1, $\omega(V_0, V_1), \omega(V_0, V_2), \omega(V_1, V_2)$ is 30KB/s, 50KB/s and 40KB/s respectively, so the bandwidth bottleneck is (V_0, V_1) and the available bandwidth capacity, i.e. the actual transmission rate during the regeneration process is 30KB/s.

In this paper, we propose a tree-structured regeneration scheme, which constructs a spanning tree in the network model G_k. Our regeneration scheme does not incur more regeneration traffic than the star-structured scheme, but it can improve available bandwidth capacity and thus reduce the regeneration time. Each node in the model, no matter newcomer or provider, can receive data from other nodes. To prevent from increasing the regeneration traffic and thus aggravating the bandwidth bottleneck, we assume each node can receive data from multiple nodes, but can send data to only one node. Encoding operation can be executed on the newcomer and the providers, and the encoding delay is ignored, since the transmission delay is usually much more critical. Fig. 1(b) is an example of the tree-structured regeneration scheme. V_1 sends its data to V_2. V_2 encodes the received data with the data it stores and sends the encoded data to V_0. As we will show in Section IV, the bandwidth bottleneck is (V_1, V_2), and the available bandwidth capacity is $\omega(V_1, V_2) = 40$KB/s. We can see the tree-structured regeneration scheme can regenerate new redundant data faster.

IV. Tree-structured Regeneration Scheme

In this section, we present our tree-structured regeneration scheme, based on the network model above. First, we show how the tree-structured scheme can regenerate new redundant data at the newcomer and prove that a maximum spanning tree is an optimal regeneration tree. Then we give the encoding scheme for linear network coding, especially for random linear coding. We analyze the available bandwidth capacity of the tree-structured and star-structured regeneration scheme based on probability theory and order statistics. We compare the available bandwidth capacity of the two schemes at last.

A. Regeneration Tree

Lemma 1: Any spanning tree T in $G_k = (V, E, \omega)$, whose root is V_0, corresponds to one and only one regeneration scheme in which V_0 is the newcomer.

Proof: Given a spanning tree T, we can build a regeneration scheme as follows. For any node in T, it receives data from its children if it is not a leaf node, encodes the received data with the data it stores, and sends the encoded data to its parent node if it is not the newcomer. In this case, the newcomer can get the data or its linear combination of the providers, and then become a new storage node.

Given a regeneration scheme of $G_k = (V, E, \omega)$, we can build a graph $T = (V, E')$, where $(V_i, V_j) \in E'$ when data are transferred on (V_i, V_j), $i, j = 0, 1, 2, \ldots, k$, $i < j$. For each edge in T, it can be mapped to one and only one provider which sends out data on this edge, since one node can send data to only one node and the newcomer does not send data to other nodes. Because there are k providers, $|E'| \leq k$. On the other hand, because the newcomer can receive encoded blocks or their linear combinations from all providers, T is a connected graph. So $|E'| \geq k$. Because $|E'| = k$ and T is a connected graph, T is a spanning tree of G_k. ■

Notice that T and G_k are both undirected graphs, but the transmission is always directed. From the proof of Lemma 1 we can see that for each edge in T, the transmission direction is from the child node to the parent node. In this sense, all edges in T can be regarded as "directed". The incoming edge of a node is the edge whose other endpoint is the child of this node, and the outgoing edge of a node is the edge whose other endpoint is its parent node.

Lemma 1 shows that we can use a spanning tree to represent a regeneration scheme of G_k. However, it does not show how

to encode the data at each provider. In Section IV-B, we will discuss this question.

Definition 1: A regeneration tree is a spanning tree in G_k.

Lemma 2: For each edge in the regeneration tree T, the amount of transferred data on it is $\frac{M}{k}$ bytes, where M is the size of the original file.

Proof: According to the proof of Lemma 1, each edge in T corresponds to one and only one provider, so we give the proof from the perspective of the providers.

For each leaf node in the regeneration tree, the size of data it sends is $\frac{M}{k}$ bytes, because the size of the data it stores is $\frac{M}{k}$ bytes.

Assume for each non-leaf node except the newcomer, the traffic on each incoming edge is $\frac{M}{k}$ bytes. The amount of data it stores is also $\frac{M}{k}$ bytes. So after linear encoding, the traffic on its outgoing edge is $\frac{M}{k}$ bytes.

Since the outgoing edges of all the providers have the same traffic on them, we can say that the traffic on each edge is uniform and is equal to $\frac{M}{k}$ bytes. ■

From Lemma 2, we can see the regeneration traffic of a tree-structured regeneration scheme is M bytes, so the optimal regeneration traffic shown in Section III-B has been achieved.

Lemma 3: For each regeneration tree T in G_k, the regeneration time depends on the edge with the minimal weight.

Proof: We have known that the weight of each edge in T, $\omega(V_i, V_j)$, $i, j = 0, 1, 2, \ldots, k$, $i < j$, denotes the bandwidth capacity between V_i and V_j. According to Lemma 2, the traffic on each edge in T is uniform. If a node sends data after it has received all the data from its children, it will waste a substantial amount of time. The optimal transmission method is to use the principle of pipelining. The node encodes and sends data to its parent node immediately after it has received one byte/packet from all of its children. So the bandwidth bottleneck is the minimal edge in the regeneration tree. ■

From the proof of Lemma 3, we give the definition of the available bandwidth capacity of a regeneration tree.

Definition 2: The available bandwidth capacity of a regeneration tree T in G_k is the weight of the minimum edge in T.

Lemma 4: [16] In a weighted undirected graph, a minimum (maximum) spanning tree is a bottleneck spanning tree, i.e. the weight of whose largest (smallest) edge is the minimum (maximum) over all spanning trees in this graph.

Theorem 1: A maximum spanning tree in G_k is a regeneration tree with the maximal available bandwidth capacity.

Proof: The proof is clear according to Lemma 3 and Lemma 4. ■

Theorem 1 shows how to find an optimal regeneration tree in G_k. We can see the star-structured regeneration scheme is a special form of the tree-structured regeneration scheme and sometimes it is the optimal. However, the tree-structured regeneration scheme is always no worse than the star-structured scheme.

Since the bandwidth capacity is time-sensitive, its measurement should be triggered before each regeneration, after which the regeneration tree can be constructed. However, because the regeneration tree is spanned over the newcomer and all the providers, the bandwidth measures are made between these nodes rather than all nodes in the network and thus are limited.

Theorem 2: Let $B(T)$ be the available bandwidth capacity of a regeneration tree T in G_k. Then the regeneration time is $\frac{M}{kB(T)}$, where M is the size of the original file.

Proof: According to Lemma 2, the amount of traffic on each edge in T is $\frac{M}{k}$ bytes. From Lemma 3, we know the regeneration time depends on the minimal weighted edge in T. According to the definition of $B(T)$, the regeneration time is $\frac{\frac{M}{k}}{B(T)} = \frac{M}{kB(T)}$. ■

B. Encoding Scheme

In G_k, assume that the encoded block stored in V_i is B'_i, $i = 1, 2, \ldots, k$. According to Eq. (4) and Eq. (5), if $(\sigma_1, \sigma_2, \ldots, \sigma_k)^T$ is a random vector, $(r_1, r_2, \ldots, r_k)^T$ is also a random vector. Thus $(r_1, r_2, \ldots, r_k)^T$ can be generated randomly on the encoding nodes in a distributed fashion. V_i is responsible to generate r_i randomly, $i = 1, 2, \ldots, k$. In one regeneration tree, if V_i does not receive data from other nodes, it sends $r_i B'_i$. If V_i receives data from $V_{i_1}, V_{i_2}, \ldots, V_{i_{in(V_i)}}$, where $in(V_i)$ is the indegree of V_i, assuming the data received from V_{i_j} is B''_j, it sends $r_i B'_i + \sum_{j=1}^{in(V_i)} B''_j$. Therefore, the newcomer can get $\sum_{i=1}^{k} r_i B'_i$, which is equal to $\sum_{i=1}^{k} \sigma_i F_i$.

C. Available Bandwidth Capacity

We analyze the available bandwidth capacity of tree-structured and star-structured regeneration scheme by order statistics. First, we introduce a basic theorem of order statistics in Lemma 5.

Lemma 5: [17] Assume $X_1, X_2, \ldots, X_n$ are n independent random variables, for each of which the cumulative distribution function is $F(x)$ and the probability density function is $f(x)$. Let $f_{(r:n)}(x)$ denote the probability density function of the r^{th} variable $X_{(r:n)}$, $X_{(1:n)} \geq X_{(2:n)} \geq \cdots \geq X_{(n:n)}$. If X_i is with continuous distribution,

$$f_{(r:n)}(x) = \frac{n! F^{n-r}(x)[1 - F(x)]^{r-1} f(x)}{(n-r)!(r-1)!}. \tag{6}$$

Let $E = \{e_1, e_2, \ldots, e_{\frac{k(k+1)}{2}}\}$ in $G_k = (V, E, \omega)$, where $\omega(e_1) > \omega(e_2) > \cdots > \omega(e_{\frac{k(k+1)}{2}})$. The bandwidth capacity on each edge is assumed to be different from each other, as it realistically reflects real-world networks with high probability.

Definition 3: MST(G_k) $= r$ if and only if the minimal edge in the maximum spanning tree of $G_k = (V, E, \omega)$ is the r^{th} maximal edge of E.

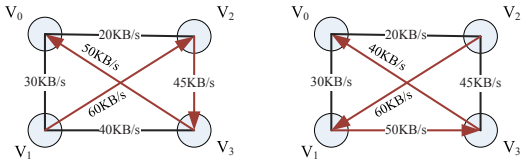
Property 1: $k \leq \text{MST}(G_k) \leq M_{k+1} - k + 1$, where $M_k = \frac{k(k-1)}{2}$.

Proof: Let $E_1 = \{e_1, e_2, \ldots, e_{k-1}\}$. If (V, E_1) is connected, it must be a maximum spanning tree of G_k. Since no other spanning tree in G_k whose minimal edge is larger than (V, E'), we can say $r \geq k$.

Because G_k is a complete graph, it is k-edge-connected. Thus it is always connected after removing $k - 1$ edges. Let

$E_2 = \{e_1, e_2, \ldots, e_{M_{k+1}-k+1}\}$. A maximum spanning tree of (V, E_2) is also a maximum spanning tree of G_k. Thus $r \leq M_{k+1} - k + 1$. ■

Fig. 2 illustrates the best case and the worst case of the maximum spanning tree in G_3. In Fig. 2(a), the bandwidth bottleneck is (V_2, V_3), which is the 3^{rd} largest edge in G_3. On the other hand, the bandwidth bottleneck in Fig. 2(b) is at (V_0, V_3), which is the $(M_{3+1} - 3 + 1 =)4^{th}$ largest edge.



(a) Best case: MST(G_3)=3 (b) Worst case: MST(G_3)=4

Fig. 2. Best and worst case of regeneration tree in G_3.

Let $E_{tree}(G_k)$ be the expected value of the available bandwidth capacity of the tree-structured regeneration scheme of G_k. Assume the probability density function of the distribution of the weight of the edge in E is $f(x)$ and $F(x)$ is the cumulative distribution function. Especially, $F(0) = 0$ because the bandwidth capacity is never negative in real world. According to Lemma 5, the probability density function of $\omega(e_i)$ is

$$f_{(i:M_{k+1})}(x) = \frac{M_{k+1}! F^{M_{k+1}-i}(x)(1-F(x))^{i-1} f(x)}{(i-1)!(M_{k+1}-i)!}. \quad (7)$$

Let $E_{(i:M_{k+1})}$ be the expected value of $\omega(e_i)$,

$$E_{(i:M_{k+1})} = \int_0^{+\infty} x f_{(i,M_{k+1})}(x) dx. \quad (8)$$

Let $p(k+1, i)$ be the probability that MST(G_k) $= i$. Then the expected value of the tree-structured regeneration scheme in G_k is

$$E_{tree}(G_k) = \sum_{i=k}^{M_{k+1}-k+1} p(k+1, i) E_{(i:M_{k+1})}. \quad (9)$$

Let $E_{star}(G_k)$ denote the expected value of the available bandwidth capacity of the star-structured regeneration scheme of G_k. Since $E_{star}(G_k)$ is the expected value of the weight of the minimal edge in $\{(V_0, V_i) | i = 1, 2, \ldots, k\}$,

$$E_{star}(G_k) = \int_0^{+\infty} x f_{(k:k)}(x) dx \quad (10)$$

$$= k \int_0^{+\infty} x(1 - F(x))^{k-1} f(x) dx. \quad (11)$$

D. Distribution of MST(G_k)

As stated in Section IV-C, $p(k+1, i)$ is the probability that MST(G_k) $= i$ in $G_k = (V, E, \omega)$. According to Property 1, $p(k+1, i) = 0$, when $i < k$ or $i > M_{k+1} - k + 1$.

Lemma 6 (Cayley's formula): [18] There are k^{k-2} spanning trees in a complete graph containing n nodes.

Theorem 3: $p(k+1, k) = \frac{(k+1)^{k-1}}{\binom{M_{k+1}}{k}}$ in $G_k = (V, E, \omega)$.

Proof: In G_k, $|V| = k+1$. According to Lemma 6, there are $(k+1)^{k-1}$ spanning trees in G_k. If we select k edges from E randomly, the probability that k edges form a spanning tree of G_k is $\frac{(k+1)^{k-1}}{\binom{M_{k+1}}{k}}$. Because MST($G_k$) $= k$ if and only if $(V, \{e_i | i = 1, 2, \ldots, k\})$ is a spanning tree of G_k, this lemma is proved. ■

Theorem 4: $p(k+1, M_{k+1}-k+1) = \frac{k+1}{\binom{M_{k+1}}{k}}$ in $G_k = (V, E, \omega)$.

Proof: MST(G_k) $= M_{k+1} - k + 1$ if and only if $e_{M_{k+1}-k+i}, i = 1, 2, \ldots, k$ are attached to one node in V. Because there are $k+1$ node in V, this lemma is proved. ■

Theorem 3 and Theorem 4 give two special cases of $p(k+1, i)$. For the general case, we can get $p(k+1, i)$ recursively.

Theorem 5: Define $Q(l, j)$ as the number of the connected graphs which contain l nodes and j edges, and $P(k+1, i)$ as the probability that MST(G_k) $= i$. Thus in $G_k = (V, E, \omega)$, if $k < i < M_{k+1} - k + 1$,

$$P(k+1, i) = \frac{\sum_{l=1}^{k} \binom{k-1}{l-1} \sum_{j=0}^{i-1} Q(l, j) Q(k+1-l, i-1-j)}{\binom{M_{k+1}-1}{i-1}}, \quad (12)$$

and

$$Q(l, j) = \begin{cases} 0 & j < l-1 \\ \binom{M_l}{j} \sum_{i=l-1}^{j} P(l, i) & l-1 \leq j \leq M_l \\ 0 & j > M_l \end{cases} \quad (13)$$

Proof: Assume (V_a, V_b), $a, b = 0, 1, 2, \ldots, k$, $a < b$, is the i^{th} maximal edge in G_k. We divide V into 2 groups, to make V_a and V_b not belong to the same one. Let $V_{(a)}$ be the group containing V_a, and $V_{(b)}$ be the group containing V_b. As $|V| = k+1$, the number of cases that $V_{(a)}$ contains l nodes is $\binom{k-1}{l-1}$. MST(G_k) $= i$ if and only if $(V, \{e_1, \ldots, e_i\})$ is connected and $(V, \{e_1, \ldots, e_{i-1}\})$ is not connected. Thus (V_a, V_b) is the only edge in $\{e_1, \ldots, e_i\}$ connecting $V_{(a)}$ and $V_{(b)}$. If the number of edges in $V_{(a)}$ is j and the number of edges in $V_{(b)}$ is $k-1-j$, the number of cases that $V_{(a)}$ and $V_{(b)}$ are all connected is $Q(l, j)Q(k+1-l, i-1-j)$. Because the number of cases that selecting $k-1$ edges from $M_{k+1}-1$ edges is $\binom{M_{k+1}-1}{i-1}$,

$$P(k+1, i) = \frac{\sum_{l=1}^{k} \binom{k-1}{l-1} \sum_{j=0}^{i-1} Q(l, j) Q(k+1-l, i-1-j)}{\binom{M_{k+1}-1}{i-1}}. \quad (14)$$

Now we consider how to get $Q(l,j)$, which is the number of the connected graphs which contain l nodes and j edges. It is clear that when $j < l-1$ it is impossible that the graph is connected. It is also impossible when $j > M_l$, because there are at most M_l edges in the complete graph containing l nodes. When $l-1 \leq j \leq M_l$, the graph is connected if and only if MST(G_{l-1})$\leq j$, Thus selecting j edges from M_l edges randomly, the probability that the graph is connected is $\sum_{i=l-1}^{j} P(l,i)$. Therefore $Q(l,j) = \begin{pmatrix} M_l \\ j \end{pmatrix} \sum_{i=l-1}^{j} P(l,i)$. ■

Theorem 5 gives the formula of $p(k+1,i)$, which is easy to be implemented by dynamic programming. Fig. 3 illustrates the distribution of $p(k+1,i)$.

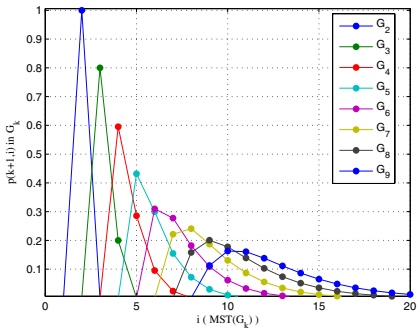


Fig. 3. Distribution of p(k+1,i) in G_k.

E. Tree vs. Star

We compare the available bandwidth capacity of the star-structured regeneration scheme and our tree-structured regeneration scheme, based on the analysis above, in the scenario that the weight of edge in $G_k = (V, E, \omega)$ is with uniformly distribution. If $\omega(V_i, V_j)$ is with uniformly distribution $U[a,b]$,

$$E_{(i:M_{k+1})} = \frac{(b-a)(M_{k+1}-i+1)}{M_{k+1}+1} + a. \quad (15)$$

$$E_{tree}(G_k) = \sum_{i=k}^{M_{k+1}-k+1} p(k+1,i) E_{(i:M_{k+1})}. \quad (16)$$

$$E_{star}(G_k) = \frac{(b-a)}{k+1} + a. \quad (17)$$

We assign randomly generated value with U[0KB/s, 56KB/s], U[8KB/s, 48KB/s], and U[16KB/s, 40KB/s] respectively, to the weight of each edge in G_k.

Fig. 4 shows the analysis results of the available bandwidth capacity in G_k, according to Eq. (15), Eq. (16) and Eq. (17). The X-axis shows k, the number of providers in the network model G_k. The Y-axis shows $E(G_k)$, the available bandwidth capacity of the corresponding regeneration scheme in G_k. Because $E_{star}(G_k) = \frac{(b-a)}{k+1} + a$, the expected value of the available bandwidth capacity of the star-structured regeneration scheme decreases and converges to a, the lower bound of uniformly distribution $U[a,b]$, with the increasing of k. However, the expected value of the available bandwidth capacity of the tree-structured regeneration scheme in G_k increases with k. Because a maximum spanning tree in G_k contains only k edges, while there are totally M_{k+1} edges, it is probable that the weight of the minimal edge in a maximum spanning tree is larger than at least half of all the edges. Thus it is easy to understand why $E_{tree}(G_k)$ increases with k.

We notice that when the bandwidth heterogeneity, i.e. the variance of the bandwidth distribution increases, the expected value of the available bandwidth capacity of the tree-structured regeneration scheme increases, but the expected value of the available bandwidth capacity of the star-structured regeneration scheme decreases. Since the tree-structured regeneration scheme has more chance to get the edges with higher bandwidth capacity in the networks with stronger bandwidth heterogeneity, the tree-structured regeneration scheme has stronger adaptability to the bandwidth heterogeneity.

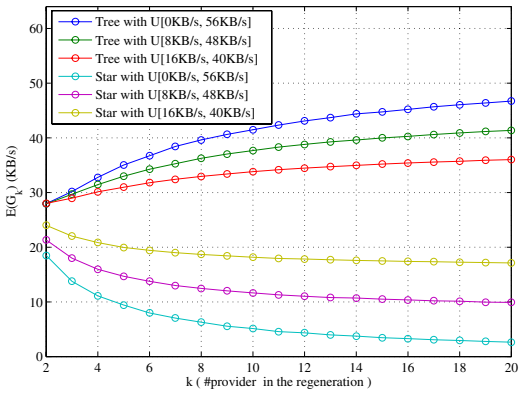


Fig. 4. Expected value of available bandwidth capacity in G_k with uniformly distributions.

V. Simulation

In this section, we compare the tree-structured regeneration scheme with the star-structured regeneration scheme in scenarios that emulate real-world distributed storage systems, based on the availability trace of PlanetLab [19] and the bandwidth distribution measured in the PlanetLab network.

Our simulation is based on an event-driven simulator, which simulates peers' actions from an availability trace file and bandwidth capacity between peers from bandwidth statistics measured in the PlanetLab network. The trace is provided by [20]. In the trace file, a node is considered to be up at t time if and only if at least half pings in the batch of pings immediately prior to t are sent to the node successfully. Pings were sent every 15 minutes between all pairs of 200-400 PlanetLab nodes from Jan. 2004 - Jan. 2005. However, in some periods, almost all the nodes go down, probably due to system upgrades or measurement errors, so a cleaned trace is provided, in which these periods are removed. Lee et al. [21] measured the bandwidth capacity between nodes in PlanetLab. Table I shows the end-to-end bandwidth capacity statistics we use in the simulation. The simulator simulates the bandwidth capacity between nodes according to Table I.

TABLE I
END-TO-END BANDWIDTH CAPACITY DISTRIBUTION IN [21].

Capacity(C)(Mb/s)	Number of paths	Percentage(%)
$0.3 \leq C < 20$	6733	30.8
$20 \leq C < 50$	1910	8.74
$50 \leq C < 80$	1303	5.96
$80 \leq C < 120$	11744	53.72
$120 \leq C < 200$	139	0.64
$200 \leq C < 500$	21	0.096
$500 \leq C < 682.9$	11	0.05

TABLE II
SIMULATION PARAMETERS

Tr	name of the trace file (*.avt)
T_s	start time (sec.)
T_f	finish time (sec.)
N_{all}	all #node
N_{on}	mean #node online
T_{up}	average up time per node per day (sec.)
N_{down}	average #down per node per day
N_{up}	average #up per node per day
r_B	bandwidth utilization rate
N_r	repeated times
M	file size (KB)
r	redundancy rate

Table II shows the parameters used in the simulation. The time of a trace file Tr starts from 0. The simulator starts the simulation at T_s and ends at T_f. The trace provides the time of nodes' join/departure. During the time, there're totally N_{all} nodes and N_{on} nodes online on average. The bandwidth capacity between each pair of N_{all} nodes is generated according to Table I. For one node on average, it joins the network N_{up} times, leaves the network N_{down} times, and stays in the network for T_{up} time every day. The storage system in the simulation uses $(n,k)-$random linear coding and the redundancy rate $r = \frac{n}{k}$ is fixed. We assume that a M KB file is saved into n nodes randomly in the network immediately before the simulation. Each storage node stores $\frac{M}{k}$ KB.

In the simulation, in order to prolong regeneration time and to enhance the impact of node instability relatively, we set a parameter r_B, the utilization rate of the bandwidth capacity between all pairs of nodes. If $r_B = 0.1$, the utilized bandwidth of each path is at most 10% of the bandwidth capacity between two nodes. What's more, we sort the nodes in the trace file according to their up/down times. In the simulation, we select N_{all} most unstable nodes to attend the simulation. Thirdly, we use a radical data loss detection mechanism in the simulation. If a node leaves the network, we assume data in this node will be permanently unavailable even though it returns some time later. Whenever a node joins the network, it will be considered as a stranger node as if it has never joined the network and never saved any data.

When a node storing at least one encoded block leaves the network, a regeneration is triggered. However, a regeneration can start when two conditions are satisfied: (i) there are at least k active storage nodes; (ii) there is at least one node to be the newcomer, while the existing storage node can not be a newcomer. If the two conditions are not satisfied, the regeneration has to wait to start. If there are more than k active storage nodes, the simulator selects k of them with widest bandwidth to the newcomer as providers. For $(n,k)-$linear network coding, the tree-structured and star-structured scheme both require k providers, so the waiting time of regeneration will not be different between the two schemes. If the regeneration is interrupted by node departures, it will be regarded as a failed regeneration and another regeneration will be triggered.

The simulation is repeated for N_r times and the results are average values. We measure the performance of regeneration schemes from three aspects: (i) regeneration time: how much time is spent from the start of a regeneration to the end; (ii) probability of the successful regeneration: the probability that a regeneration finishes successfully, not interrupted by the node departures; (iii) data availability: the probability that a file is available. According to the analysis in Section IV, the regeneration time is $\frac{M/k}{r_B \cdot B}$, where B is the available bandwidth capacity of the regeneration scheme. The probability that data are available is the ratio of the available time to the total time in the simulation. For $(n,k)-$linear network coding, data are available when there are at least k active storage nodes.

We run the simulation for the tree-structured and the star-structured regeneration scheme respectively. For the tree-structured scheme, the regeneration tree is produced by Kruskal's algorithm. Table III shows the value of parameters used in the simulation. We use the cleaned trace file *pl-app-cleaned.avt*. The simulation lasts for 10^7 seconds (115 days 17 hours 46 minutes 40 seconds). We select 100 most unstable nodes in the trace. The redundancy rate is 1.5, so the system uses $(1.5k,k)-$random linear coding. We observe the performance of the two regeneration schemes with the increasing of k, and k is selected as $2, 4, \ldots, 20$ respectively.

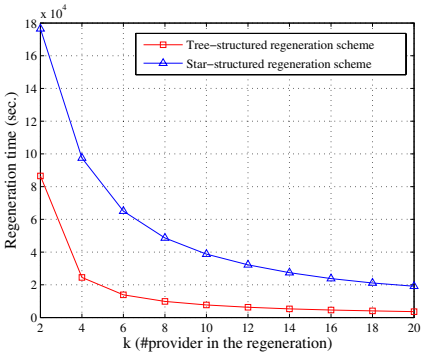


Fig. 5. Regeneration time with parameters in Table III.

Fig. 5, Fig. 6 and Fig. 7 show the simulation results. In

TABLE III
PARAMETER'S VALUE IN THE SIMULATION

Tr	T_s	T_f	N_{all}	N_{on}	T_{up}
pl-app-cleaned.avt	2×10^6	1.2×10^7	100	49.92	43135.08
N_{down}	N_{up}	r_B	N_r	M	r
0.137	0.135	0.01	10^4	10^7	1.5

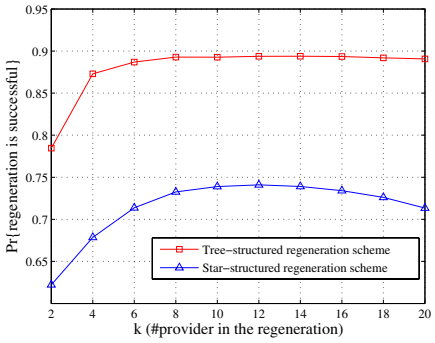


Fig. 6. Probability of successful regeneration with parameters in Table III.

Fig. 5, we show that the tree-structured regeneration scheme can save regeneration time by at least 75% when $k \geq 4$, and by 82% at most when $k = 20$. The regeneration time also decreases with k, because the network traffic of each path is $\frac{M}{k}$ bytes. Since the regeneration time is reduced, it is more likely to finish the regeneration successfully. Fig. 6 shows the tree-structured scheme can increase the probability of successful regeneration by 21%-29%. With the increasing of k, the regeneration is more possible to be interrupted. However, the regeneration time is reduced when k increases, so the probability of successful regeneration does not change a lot. Fig. 7 shows when $k > 2$, the tree-structured scheme can improve data availability by 73%-124% (73% when $k = 20$ and 124% when $k = 10$). What's more, when $k \geq 10$, the data availability of the tree-structured regeneration scheme is always more than 90%, while the availability of the star-structured scheme is less than 60%.

VI. Conclusion

In this paper, we present a tree-structured regeneration scheme for the distributed storage systems using network coding. Our mathematical analysis shows that the tree-structured regeneration scheme can improve the available bandwidth capacity and the adaptability to the bandwidth heterogeneity, compared with the conventional star-structured regeneration scheme. Our simulation results show that our scheme can reduce regeneration time and improve data availability.

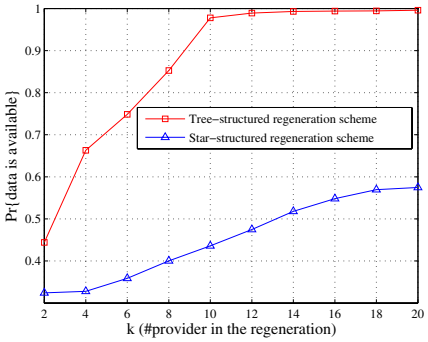


Fig. 7. File availability with parameters in Table III.

Acknowledgment

We thank the anonymous reviewers for their helpful advices. This work was supported in part by NSFC under Grant No. 60702054, 863 program of China under Grant No. 2006AA01Z203, Shanghai Municipal R&D Foundation under Grant No. 07dz15004-1, the Shanghai Rising-Star Program under Grant No. 08QA14009, and State Key Laboratory of Integrated Service Networks under Grant No. ISN-9-06.

References

[1] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, *Oceanstore: an architecture for global-scale persistent storage,* SIGPLAN Not., vol. 35, no. 11, pp. 190-201, 2000.

[2] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran, *Network coding for distributed storage systems,* 26th IEEE International Conference on Computer Communications, pp. 2000-2008, 2007.

[3] Y.Wu, R. Dimakis, and K. Ramch, *Deterministic regenerating codes for distributed storage,* in Allerton Conference on Control, Computing, and Communication, Urbana-Champaign, IL, 2007.

[4] Z. Zhang, Q. Lian, S. Lin, W. Chen, Y. Chen, and C. Jin, Bitvault: a highly reliable distributed data retention platform, *SIGOPS Oper. Syst. Rev.,* vol. 41, no. 2, pp. 27-36, 2007.

[5] W. Lin, C. Ye, and D. Chiu, Decentralized replication algorithms for improving file availability in p2p networks, *Fifteenth IEEE International Workshop on Quality of Service,* pp. 29-37, June 2007.

[6] S. Acedanski, S. Deb, M. Medard, and R. Koetter, How good is random linear coding based distributed networked storage? in *Proc. 1st Workshop on Network Coding,* Riva del Garda, Italy, Apr. 2005.

[7] R. Rodrigues and B. Liskov, High availability in DHTs: Erasure coding vs. replication, *Proceedings of 4th International Workshop on Peer-to-Peer Systems (IPTPS'05).*

[8] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, Network information flow, *IEEE Transactions on Information Theory,* vol. 46, no. 4, pp. 1204-1216, Jul 2000.

[9] M. Yang and Y. Yang, Peer-to-peer file sharing based on network coding, *ICDCS '08. The 28th International Conference on Distributed Computing Systems,* pp. 168-175, June 2008.

[10] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, The benefits of coding over routing in a randomized setting, *Proceedings. IEEE International Symposium on Information Theory,* p. 442, 2003.

[11] C. Cooper, On the rank of random matrices, in *Random Structures and Algorithms,* vol. 16, no. 2, 2000, pp. 209-232.

[12] B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, Efficient replica maintenance for distributed storage systems, in *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation.* Berkeley, CA, USA: USENIX Association, 2006, pp. 4-4.

[13] A. Duminuco and E. Biersack, Hierarchical codes: How to make erasure codes attractive for peer-to-peer storage systems, *Eighth International Conference on Peer-to-Peer Computing,* pp. 89-98, 2008.

[14] S.-J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu, Bandwidth-aware routing in overlay networks, *INFOCOM 2008. The 27th Conference on Computer Communications.* IEEE, pp. 1732-1740, April 2008.

[15] J. Li, and X. Wang, *Redundancy Maintenance in P2P Storage Networks*, to appear in Journel of Computer Research and Development, Vol. 46(Suppl.), 2009.

[16] S. Skiena, *The Algorithm Design Manual*, Springer-Verlag, 1997.

[17] H. A. David and H. N. Nagaraja, Order Statistics. Wiley-Interscience, 3 edition, August 4, 2003.

[18] R. Balakrishnan and K. Ranganathan, A Textbook of Graph Theory. Springer, 1999.

[19] Planetlab. [Online]. Available: http://www.planet-lab.org/

[20] J. Stribling. Planetlab all pairs ping. [Online]. Available: http://infospect.planet-lab.org/pings.

[21] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, Measuring bandwidth between planetlab nodes, 2005, pp. 292-305. [Online]. Available: http://www.springerlink.com/content/p9uy6mgwn7k2fabw