Tree-structured Data Regeneration in Distributed Storage Systems with Regenerating Codes

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

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

***Abstract*—Distributed storage systems provide large-scale reliable data storage by storing a certain degree of redundancy in a decentralized fashion on a group of storage nodes. To recover from data losses due to the instability of these nodes, whenever a node leaves the system, additional redundancy should be regenerated to compensate such losses. In this context, the general objective is to minimize the volume of actual network traffic caused by such regenerations. A class of codes, called *regenerating codes*, has been proposed to achieve an optimal trade-off curve between the amount of storage space required for storing redundancy and the network traffic during the regeneration. In this paper, we jointly consider the choices of regenerating codes and network topologies. We propose a new design, referred to as RCTREE, that combines the advantage of regenerating codes with a *tree-structured* regeneration topology. Our focus is the efficient utilization of network links, in addition to the reduction of the regeneration traffic. With the extensive analysis and quantitative evaluations, we show that RCTREE is able to achieve a both fast and stable regeneration, even with departures of storage nodes *during* the regeneration.**

***Index Terms*—Distributed Storage System, Data Regeneration, Regenerating Codes**

I. Introduction

Large-scale distributed storage systems are designed to provide reliable services of data storage, by storing a degree of data redundancy in a decentralized manner, across a large number of *storage nodes* in the system [1]. These storage nodes may be off-the-shelf cluster nodes in large-scale data centers, disk arrays in storage area networks, or even ordinary end hosts across the Internet, organized in a peer-to-peer fashion.

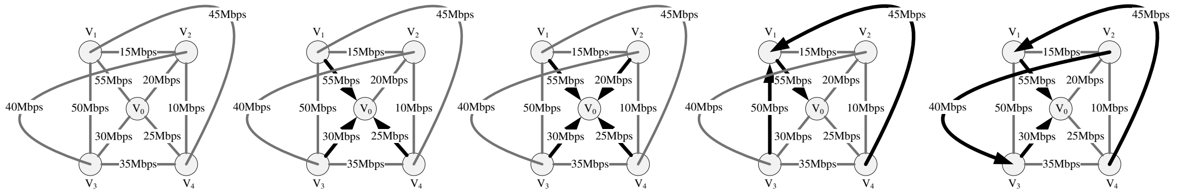
Regardless of their reliability, however, storage nodes in distributed storage systems may fail, leading to the data loss. In fact, large data centers are designed to treat storage node failures as the *rule*, not the *exception*. With the presence of node failures, it is desirable to maintain a degree of data *redundancy*, such that a subset of storage nodes is sufficient to recover the original data. When a storage node does fail, it is necessary to *regenerate* data in a replacement node, called a *newcomer*, in order to restore the required degree of data redundancy. How such regeneration is to be performed depends on the design objectives of codes to achieve redundancy.

If the objective is to minimize the storage space needed for redundancy, it has been shown that Maximum Distance Separable (MDS) codes are optimal for such minimum-storage regeneration [2]. It has been used in the literature to maintain a much smaller degree of redundancy than simple replication for the same reliability [3]. For example, a file of size $\mathcal{M}$ bits can be divided into k blocks, each of size $\mathcal{M}/k$, and then be encoded into n coded blocks with an (n, k) MDS code, to be stored in n distinct storage nodes, and any k blocks can be used to recover the original file.

However, if minimizing network bandwidth used to regenerate data on the newcomer becomes the objective instead, with MDS codes, a newcomer needs a minimum of k blocks, *i.e.* a total of $\mathcal{M}$ bits, to regenerate its new coded block of size $\mathcal{M}/k$ bits; while only $\mathcal{M}/k$ bits are required if replication is used. Dimakis *et al.* [2] and Wu *et al.* [4] have shown the surprising result that, deterministic linear network coding (defined over a sufficiently large finite field) can be used to design a class of *minimum-bandwidth regenerating* codes to minimize bandwidth required for the regeneration, as long as more than k storage nodes, called *providers*, can be contacted by the newcomer during the regeneration. Acedański *et al.* [5] has also evaluated the role of random linear coding, rather than deterministic linear codes, in distributed storage systems.

Though it is encouraging to design minimum-bandwidth regenerating codes to minimize bandwidth, the existing literature has not focused on the role of the *network topology*, within which the regeneration process takes place. It has conventionally been assumed that the regeneration process is performed on a simple *star-structured* topology, *i.e.*, the newcomer receives coded blocks directly from each of providers. In the overlay mesh connecting storage nodes, however, not all overlay links enjoy the same available bandwidth. If we take into account the heterogeneity of bandwidth on links between storage nodes, a *tree-structured* topology naturally ensues, in which providers are allowed to *relay* regeneration traffic to the newcomer. How should we construct such a tree to efficiently utilize available bandwidth on each link between storage nodes? How should we jointly consider the construction of regeneration trees and the design of regenerating codes?

In this paper, we consider the general case of constructing such tree-structured regeneration topologies, with a variable number of providers in a tree topology, as well as the use of regenerating codes to achieve the storage-bandwidth optimal trade-off curve. Our new design, referred to as RCTREE, is able to work effectively with the bandwidth heterogeneity. RCTREE even considers the case that the storage node may fail



(a) network model with one newcomer and four storage nodes (b) STAR with V_1,V_3 and V_4 as providers (c) RCSTAR with V_1,V_2,V_3 and V_4 as providers (d) TREE with V_1,V_3 and V_4 as providers (e) RCTREE with V_1,V_2,V_3 and V_4 as providers

Fig. 1: Examples of four regeneration schemes: STAR, RCSTAR, TREE and RCTREE.

during the regeneration process. With the extensive analysis and quantitative evaluations based on statistical data in PlanetLab, we are able to show that RCTREE helps to be one step closer towards a practical repair of failed storage nodes in distributed storage systems.

The remainder of the paper is organized as follows. Sec. II shows the advantages of RCTREE with an illustrative example. In Sec. III we introduce the network model, and present our extensive analysis on the regeneration tree. Sec. IV proposes RCTREE with detailed analysis. Sec. V analyzes the stability of RCTREE in comparison with some existing schemes. Sec. VI concludes this paper.

II. Beyond Regenerating Codes: A Motivating Example

We now introduce an illustrative example of data regeneration in the distributed storage system in Fig. 1. Fig. 1(a) shows the network model. There are five storage nodes, denoted by $V_0, V_1, V_2, V_3,$ and V_4. The bandwidth capacity of the link between two providers is heterogeneous. We assume that the redundancy is coded by a $(5, 3)$ MDS code, stored in $V_1, V_2, V_3, V_4,$ and a departing storage node. Each storage node stores a coded block of $\frac{\mathcal{M}}{3}$ bits, if the size of the original data is $\mathcal{M}$ bits. In order to regenerate the lost redundancy, V_0 is selected to be the newcomer. Since $(5, 3)$ MDS code is used, V_0 needs to receive redundancy from at least three providers.

Fig. 1(b) — Fig. 1(e) show illustrations of four regeneration schemes. For the conventional star-structured regeneration (STAR) in Fig. 1(b), if V_0 selects $V_1, V_3,$ and V_4 as providers, it receives data directly from the three providers, illustrated by the darkened edges. Considering the regeneration time, *i.e.*, the time that the newcomer spends on regenerating a new coded block, STAR costs $\frac{\frac{\mathcal{M}}{3}}{25\text{Mbps}}$ seconds to accomplish the regeneration, because the transmission is bottlenecked by the link between V_0 and V_4. We ignored the encoding time of MDS code because the processors usually perform encoding operations much faster than the network transmission, and the encoding can be performed simultaneously with the transmission.

On the other hand, if more than three providers, for example, $V_1, V_2, V_3,$ and V_4, are used as providers in the regeneration, regenerating codes [2], [4] provide a way to reduce the bandwidth usage in the regeneration. Apart from *minimum-bandwidth regenerating* codes, Wu *et al.* also propose *minimum-storage regenerating* (MSR) codes, which cost the same storage space on storage nodes with MDS codes. For an (n, k) MSR code, each storage node stores $\frac{\mathcal{M}}{k}$ bits and only $\frac{\mathcal{M}}{k(r-k+1)}$ bits are transmitted on each link in the regeneration, where r is the number of providers. Different from MDS codes, the original file is divided into more than k blocks. The coded blocks are their deterministic [4] or random [6] linear combinations and each storage node stores more than one coded blocks.

Even though MSR codes are not able to reach the minimum regeneration traffic of regenerating codes, they cost the least amount of storage space in storage nodes. Other kinds of regenerating codes can further reduce the regeneration traffic with an increased storage cost, but MSR codes use the storage space most effectively. Therefore, we consider MSR codes in this paper. In Fig. 1(c), with the employment of MSR codes in STAR (RCSTAR), there will be only *half* of $\frac{\mathcal{M}}{3}$ bits transferred on each darkened edge. Therefore, the regeneration time can be reduced to $\frac{\frac{\mathcal{M}}{6}}{20\text{Mbps}}$ seconds.

STAR and RCSTAR, however, suffer from the bottleneck links of (V_0, V_4) and (V_0, V_2), respectively. If we consider the links between providers, we can utilize these links to bypass the slow bottleneck link in STAR. In Fig. 1(d), we show an example of the tree-structured regeneration (TREE), with three providers of $V_1, V_3,$ and V_4. A spanning tree, called *regeneration tree*, is constructed over $V_0, V_1, V_2,$ and V_4. V_1 receives data from V_3 and V_4, encodes the received data with the data it stores, and sends the encoded data to V_0. By streamlining the relay on V_1, *i.e.*, V_1 encodes the data byte-by-byte rather than after receiving the whole block, the regeneration time will be bottlenecked by the link between V_1 and V_4, and thus the regeneration time is $\frac{\frac{\mathcal{M}}{3}}{45\text{Mbps}}$ seconds only.

In our previous work [7], we have analyzed the bottleneck bandwidth that the tree-structured regeneration can achieve, yet with the constraint of exactly k providers, namely three providers in Fig. 1(d). In fact, as shown by Fig. 1(c), if there is only one storage node losing its data among a total of five storage nodes, there are four storage nodes available to be used as providers in the regeneration. In Fig. 1(e), we construct a regeneration tree with $V_1, V_2, V_3,$ and V_4 as providers, and use MSR codes in the system. As a result, the regeneration time can be further reduced to $\frac{\frac{\mathcal{M}}{6}}{30\text{Mbps}}$ seconds. Compared with STAR in Fig. 1(b), the regeneration time is reduced by 58.3% in RCTREE.

In this paper, we present an in-depth analysis of the general case that the number of providers is variable, and propose RCTREE, a combined scheme of regenerating codes and TREE.

In addition, we also considers node failures *during* the regeneration. We compare the stability of STAR, TREE and RCTREE from the perspective of lifetime of regeneration trees.

III. Constructing Regeneration Trees

In this section, we present an in-depth analysis of TREE, the tree-structured regeneration in the general case with a variable number of providers. We first introduce our network model for the regeneration process in distributed storage systems. Then we validate Prim's algorithm to obtain the optimal regeneration tree, analyze its bottleneck bandwidth, and show the strategy of deciding the number of providers.

A. Network Model

We assume that in a distributed storage system, redundant data is produced by an (n, k) MDS code, which divides the original file into k blocks, $F_1, F_2, \ldots, F_k$, and encodes them into n coded blocks $B_1, B_2, \ldots, B_n$. In the network, with respect to one file, there are n storage nodes, $V_1, V_2, \ldots, V_n$, storing the n coded blocks. We assume that B_i is stored in $V_i, i = 1, 2, \ldots, n$, for this storage space allocation scheme will lead to the optimal recovery rate [8]. Without loss of generality, assume that B_n gets lost. Another coded block will then be regenerated in a newcomer V_0. Assume d storage nodes are active in the regeneration, *e.g.* $V_1, V_2, \ldots,$ and V_d. In order to maintain the MDS property, V_0 should receive data from at least k nodes of the d active storage nodes, called providers, $k \leq d < n$. Let (V_i, V_j) be the undirected edge connecting V_i and V_j, and $\omega(V_i, V_j)$, the weight of (V_i, V_j), represent the bandwidth capacity of (V_i, V_j).

In this paper, we present the network model in the regeneration as an undirected complete graph $G(d; n, k) = \{V(d+1), E(d+1), \omega\}$, $k \leq d < n$, where $V(d+1) = \{V_0, V_1, \ldots, V_d\}$, and $E(d+1) = \{(V_i, V_j) | 0 \leq i < j \leq d\}$. V_0 is the newcomer and other nodes in $V(d+1)$ are storage nodes, at least k nodes of which should be selected as providers. We assume that the weight of each edge in $E(d+1)$ is different from other edges. In the wide-area network or Internet, this holds with high probability. Fig. 1(a) is an example of $G(4; n, k), n > 4 \geq k$.

B. The $(r, d)-$Regeneration Tree

Given the network model, the following definition describes the tree-structured regeneration.

Definition 1: In $G(d; n, k) = \{V(d+1), E(d+1), \omega\}$, an $(r, d)-$regeneration tree is a tree whose root is V_0 and covers r providers in $V(d+1)$, $k \leq r \leq d$.

In an $(r, d)-$regeneration tree, the non-leaf providers receive data from their children nodes, encode the received data with the data they store, and relay the encoded data to their parent nodes byte-by-byte. By the relay of providers, the newcomer will get a linear combination of r coded blocks of r providers, though it may probably connect to fewer than r providers directly. On each edge in the regeneration tree, $\frac{\mathcal{M}}{k}$ bits of data are transmitted. The bottleneck edge is the least weighted edge in the $(r, d)-$regeneration tree.

In order to get an optimal $(r, d)-$regeneration tree, which has the maximum bottleneck bandwidth, we can use Prim's algorithm, which constructs a maximum spanning tree starting from the root inductively. If the root has been selected, in the r^{th} step of Prim's algorithm, there are $r + 1$ nodes in the produced tree, whose bottleneck bandwidth is optimal among all $(r, d)-$regeneration trees in $G(d; n, k)$.

Theorem 1: After the r^{th} step, Prim's algorithm can produce an optimal $(r, d)-$regeneration tree in $G(d; n, k)$.

Proof: When $r = 1$, the proof is clear. The Prim's algorithm will select the maximum edge incident to V_0 at the first step. Suppose this statement is true when $r = k_0$, $0 < k_0 < d$. After the k_0^{th} step, an optimal $(k_0, d)-$regeneration tree T_{k_0} is produced. Assume that a $(k_0 + 1, d)-$regeneration tree T_{k_0+1} is produced after the $(k_0 + 1)^{th}$ step, and e is the selected edge at the $(k_0 + 1)^{th}$ step. If $\omega(e) \geq B(T_{k_0})$, T_{k_0+1} is an optimal $(k_0 + 1, d)-$regeneration tree, otherwise it will contradict with that T_{k_0} is an optimal $(k_0, d)-$regeneration tree.

If $\omega(e) < B(T_{k_0})$, clearly $B(T_{k_0}) > B(T_{k_0+1})$. Assume that there exists a tree T'_{k_0+1} with root V_0, and $B(T'_{k_0+1}) > B(T_{k_0+1})$. Removing one of the leaf nodes except V_0 and the edge incident to this node in T'_{k_0+1}, we obtain another tree T'_{k_0}, and thus $B(T_{k_0}) \geq B(T'_{k_0}) \geq B(T'_{k_0+1}) > B(T_{k_0+1})$.

If the number of the nodes in $T_{k_0} \cup T'_{k_0}$ is more than $k_0 + 1$, then the weight of the bottleneck edge of the $(k_0 + 1, d)-$regeneration tree constructed by Prim's algorithm in $T_{k_0} \cup T'_{k_0}$ is no less than $B(T'_{k_0+1})$, and thus must be more than $B(T_{k_0+1})$. This contradicts with the fact that $B(T_{k_0+1})$ is constructed by Prim's algorithm.

If the number of the nodes in $T_{k_0} \cup T'_{k_0}$ is exactly $k_0 + 1$, the node set of T_{k_0} is the same with that of T'_{k_0}. Since the uniqueness of the edge weight leads to the uniqueness of the maximum spanning tree, we have $T_{k_0} = T'_{k_0}$. By Prim's algorithm, we thus have $B(T'_{k_0+1}) = B(T_{k_0+1})$. This leads to the contradiction. ■

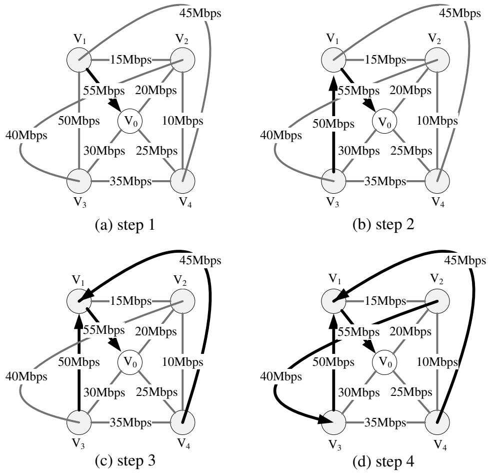
From the proof of Theorem 1, we can easily get the following corollary, which reveals the strategy of deciding the number of providers in TREE.

Corollary 1: Given $G(d; n, k)$, the bottleneck bandwidth of an optimal $(r + 1, d)-$regeneration tree is no better than an optimal $(r, d)-$regeneration tree.

Fig. 2 shows the output of Prim's algorithm on the network model in Fig. 1(a). The four steps correspond to an optimal $(r, 4)-$regeneration tree, $r = 1, 2, 3, 4$, whose bottleneck bandwidth are 55Mbps, 50Mbps, 45Mbps, and 40Mbps, respectively. We can see the employment of more providers will not improve the bottleneck bandwidth in TREE.

C. Bottleneck Bandwidth of the $(r, d)-$Regeneration Tree

Since we have obtained the optimal $(r, d)-$regeneration tree by Prim's algorithm, we analyze its bottleneck bandwidth in this section. We represent the bottleneck edge in the optimal $(r, d)-$regeneration tree by its sequential index in the edge set. Based on the probability of the sequential index and the expected bandwidth of the edge with the corresponding sequential index, we can obtain the expected bottleneck bandwidth.



(a) step 1 (b) step 2 (c) step 3 (d) step 4

Fig. 2: Regeneration trees after 1^{st} — 4^{th} steps of Prim's algorithm on the network model in Fig 1(a). The bottleneck bandwidth decreases with the increased number of providers.

Definition 2: Let e be the bottleneck edge of an optimal $(r,d)-$regeneration tree in $G(d;n,k)$, produced by Prim's algorithm. If e is the i^{th} maximum edge in $E(d+1)$, $\sigma_{\text{TREE}}(r, G(d;n,k)) = i$.

The following property gives the upper and lower bounds of $\sigma_{\text{TREE}}(r, G(d;n,k))$.

Property 1: $r \le \sigma_{\text{TREE}}(r, G(d;n,k)) \le M_{d+1} - d + 1$, where $M_d = \frac{d(d-1)}{2}$.

Proof: There are r edges in the $(r,d)-$regeneration tree, so $r \le \sigma_{\text{TREE}}(r, G(d;n,k)$. Because $G(d;n,k)$ is a complete graph, it is d-edge-connected. $G(d;n,k)$ will still be connected after removing $d-1$ edges, so $\sigma_{\text{TREE}}(r, G(d;n,k)) \le M_{d+1} - d + 1$. ■

The following lemma shows the probability of $\sigma_{\text{TREE}}(r, G(k;n,k))$, a special case of the network model that the number of providers is exactly k.

Lemma 1: [7] Let $Q(l,j)$ denote the number of the connected graphs which contain l labeled nodes and j edges, and $P(k+1,i)$ denote the probability that $\sigma_{\text{TREE}}(k, G(k;n,k)) = i$. Thus in $G(k;n,k) = (V(k+1), E(k+1), \omega)$, if $k < i < M_{k+1} - k + 1$,

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

and

$$Q(l,j) = \begin{cases} 0 & j < l-1; \\ C_{M_l}^{j} \sum_{i=l-1}^{j} P(l,i) & l-1 \le j \le M_l; \\ 0 & j > M_l. \end{cases} \quad (2)$$

Based on Lemma 1, we show the probability of σ_{TREE} in the general model, $G(d;n,k)$.

Theorem 2: Let $p(d+1, r+1; i)$ be the probability that $\sigma_{\text{TREE}}(r, G(d;n,k)) = i$. We have $p(d,r;i) =$

$$\frac{\frac{2}{d} \sum_{l=1}^{r-1} C_{d-2}^{l-1} R(d,r,i,l) + \frac{2(d-2)}{d} \sum_{l=2}^{r-1} C_{d-3}^{l-2} R(d,r,i,l)}{C_{M_d-1}^{i-1}}, \quad (3)$$

$r - 1 \le i \le M_d - (d-1) + 1$, where

$$R(d,r,i,l) = \sum_{i_1=l-1}^{i-1} Q(l,i_1) \sum_{t=r-l}^{d-l} C_{d-l-1}^{t-1} \cdot \sum_{i_2=t-1}^{i-1-i_1} Q(t,i_2) C_{M_{d-l-t}}^{i-1-i_1-i_2}. \quad (4)$$

Proof: In $G(d-1;n,k)$, given an optimal $(r-1, d-1)-$regeneration tree produced by Prim's algorithm, we assume that (V_a, V_b) is its bottleneck edge and it is the i^{th} maximum edge in $E(d), 0 \le a < b < d$. Removing all the edges in $E(d)$ whose weight is less than (V_a, V_b), we can see that $p(d,r;i)$ equals the probability that any i edges connecting nodes in $V(d)$ can form an $(r-1, d-1)-$regeneration tree. If the position of (V_a, V_b) has been determined, apparently the number of ways to select other $i-1$ edges in $E(d)$ is $C_{M_d-1}^{i-1}$.

We now consider the number of regeneration trees. The probability of $a = 0$ is $\frac{2}{d}$. Divide V_d into two groups, $V_{(a)}$ and $V_{(b)}$. Let V_a belong to $V_{(a)}$ and V_b belong to $V_{(b)}$. Assume $V_{(a)}$ contains l nodes, $1 \le l \le r-1$. Since V_0 and V_b belong to $V_{(a)}$ and $V_{(b)}$, respectively, the number of ways to assign other $d-2$ nodes into the two groups is C_{d-2}^{l-1}. On the other hand, the probability of $a \ne 0$ is $\frac{d-2}{d}$. However, now there are two possibilities that V_a can belongs to either $V_{(a)}$ or $V_{(b)}$. Moreover, since now $V_a \ne V_0$, we have $2 \le l \le r-1$. As we need to assign other $d-3$ nodes into the two groups, there are C_{d-3}^{l-2} ways to achieve this.

Since we have determined the nodes in $V_{(a)}$ and $V_{(b)}$, we need to assign other $i-1$ edges except (V_a, V_b) into $V_{(a)}$ and $V_{(b)}$ to form an $(r-1, d-1)-$regeneration tree. Suppose there are $R(d,r,i,l)$ ways to achieve this, we get Eq. (3).

Now we consider $R(d,r,i,l)$. Without loss of generality, we assume that $V_0 \in V_{(a)}$. Assume there are i_1 edges in $V_{(a)}$ and $i-1-i_1$ edges in $V_{(b)}$. The i_1 edges in $V_{(a)}$ have to construct a spanning tree on $V_{(a)}$. By Eq. (1), the number of ways to achieve this is $Q(l, i_1)$. For the $i-1-i_1$ edges in $V_{(b)}$, an spanning tree has to be constructed over V_b and at least other $t-1$ nodes in $V_{(b)}$, $r-l \le t \le d-l$, otherwise the $(r-1, d-1)-$regeneration tree can not be constructed by Prim's algorithm. There are C_{d-l-1}^{t-1} ways to select the nodes covered by the spanning tree, since V_b has been selected. Assign i_2 edges into these t nodes, $t-1 \le i_2 \le i-1-i_1$, and there are $Q(t, i_2)$ possibilities by Eq. (1). For the remaining $d-l-t$ nodes and $i-1-i_1-i_2$ edges, just assign such number of edges as the edges connecting these $d-l-t$ nodes. and the number of ways is $C_{M_{d-l-t}}^{i-1-i_1-i_2}$. In summary, $R(d,r,i,l) = \sum_{i_1=l-1}^{i-1} Q(l,i_1) \sum_{t=r-l}^{d-l} C_{d-l-1}^{t-1} \sum_{i_2=t-1}^{i-1-i_1} Q(t,i_2) C_{M_{d-l-t}}^{i-1-i_1-i_2}$. ■

When $d = r = k$, Eq. (1) is a special case of Eq. (3).

Let $E_{(i:M_{d+1})}$ denote the expected bandwidth of the i^{th} maximum edge in $E(d)$. We can obtain the expected bottleneck bandwidth of the optimal $(r,d)-$regeneration tree in $G(d;n,k)$

by Eq. (5):

$$E_{\text{TREE}}(r, G(d; n, k)) = \sum_{i=r}^{M_{d+1}-d+1} p(d+1, r+1; i) E_{(i:M_{d+1})}. \quad (5)$$

IV. Regeneration with Regenerating Codes

A. Regenerating Codes

In Sec. III, we show a general analysis of the regeneration tree using (n, k) MDS codes. Though the bottleneck bandwidth can be improved if we have more storage nodes as the candidates of providers, employing more providers does not provide a substantial improvement. First, according to Corollary 1, the increased number of providers does not relieve the bottleneck further. Second, it incurs more network traffic in the regeneration, because more edges are employed in the $(r, d)-$regeneration tree and the traffic on each edge has not been reduced. Therefore, we propose RCTREE, combining *minimum-storage regenerating* (MSR) codes with the tree-structured regeneration (TREE).

Compared with MDS codes, MSR codes can reduce the regeneration traffic. Since the number of providers is variable in this paper, the coding scheme of MSR codes should adapt to this. For the regeneration with d providers, the file should be divided into at least $k(d-k+1)$ blocks to achieve the lower bound of regeneration traffic [6]. Assume that the original file are divided into L blocks, and d is the maximum integer that satisfies $k(d-k+1) \le L$. L should be large enough so that $d \ge k$, *i.e.*, there can be at least k providers in the regeneration. Each storage node stores $\lceil \frac{L}{k} \rceil$ coded blocks. In the regeneration with r providers, $k \le r$, for RCTREE, each provider encodes its $\lceil \frac{L}{k} \rceil$ coded blocks into $\left\lceil \frac{\lceil \frac{L}{k} \rceil}{r-k+1} \right\rceil$ blocks and then sends them to its parent node. Then the newcomer receives a total of $r \left\lceil \frac{\lceil \frac{L}{k} \rceil}{r-k+1} \right\rceil$ blocks and finally encodes them into $\lceil \frac{L}{k} \rceil$ blocks, so the newcomer has to receive data directly from at least $r-k+1$ providers. The traffic on each link is $\frac{\mathcal{M}}{k(r-k+1)}$ bits approximately if L is large enough.

Definition 3: An $(r, d, k)-$regeneration tree in $G(d; n, k)$ is a tree with root V_0 whose degree is at least $r-k+1$, and covers r providers in $V(d+1)$, $k \le r \le d$.

Algorithm 1 shows how to get an optimal $(r, d, k)-$regeneration tree. We first construct an optimal $(r, d)-$regeneration tree by Prim's algorithm (Line 1 – Line 5). If the degree of the $(r, d)-$regeneration tree is invalid, we adjust the edges in the tree by adding the edge in E_{root} inductively (Line 6 – Line 10).

B. Bottleneck Bandwidth of the Optimal $(r, d, k)-$regeneration tree

In this section, we discuss the bottleneck bandwidth of the optimal $(r, d, k)-$regeneration tree produced by Algorithm 1 . Similar to Definition 2, we give the definition of σ_{RCTREE} of the optimal $(r, d, k)-$regeneration tree in $G(d; n, k)$.

Algorithm 1 Find an optimal $(r, d, k)-$regeneration tree T in $G(d; n, k)$, $k \le r \le d$. Define $E_{root} = \{(V_0, V_i) | i = 1, 2, \ldots, d\}$, and $D(T) = |E_{root} \cap \{\text{edges in } T\}|$.

```
1:  $T \leftarrow \emptyset$
2:  for $i \leftarrow 1$ to $r$ do
3:      $e_i \leftarrow$ the largest edge making $T \bigcup \{e_i\}$ a rooted tree
4:      $T \leftarrow T \bigcup \{e_i\}$
5:  end for
6:  for $i \leftarrow D(T) + 1$ to $r - k + 1$ do
7:      $e_1 \leftarrow$ the largest edge $\in E_{root} - T$
8:      $e_2 \leftarrow$ any edge $\in T - E_{root}$ making $T \bigcup \{e_1\} - \{e_2\}$ a tree rooted by $V_0$
9:      $T \leftarrow T \bigcup \{e_1\} - \{e_2\}$
10: end for
```

Definition 4: Let e be the bottleneck edge of an optimal $(r, d, k)-$regeneration tree in $G(d; n, k)$, produced by Algorithm 1. If e is the i^{th} maximum edge in $E(d+1)$, $\sigma_{\text{RCTREE}}(r, G(d; n, k)) = i$.

Notice that an $(r, d, k)-$regeneration tree is still an $(r, d)-$regeneration tree. By Property 1, $r \le \sigma_{\text{RCTREE}}(r, G(d; n, k)) \le M_{d+1} - d + 1$.

Now we show the probability of $\sigma_{\text{RCTREE}}(r, G(d; n, k))$. In Algorithm 1, some edges may be added into the optimal $(r, d)-$regeneration tree if the degree constraint of the root is not satisfied (Line 6 — Line 10). We first discuss whether this will decrease the bottleneck bandwidth of the regeneration tree.

Lemma 2: Let $C_n^{(k_1,k_2)} = \frac{n!}{k_1! k_2!}$, $0 \le k_1, k_2 \le n$. Let $p(d+1, r+1, c; i)$ be the probability that $\sigma_{\text{RCTREE}}(r, G(d; n, k)) = i$ and c edges in E_{root} have weights more than the i^{th} edge in $E(d+1)$. $p(d, r, c; i) =$

$$\frac{\frac{2(d-2)}{d} \sum_{l=2}^{r-1} C_{d-3}^{l-2} S(d, r, i, l, c) + \frac{2}{d} \sum_{l=1}^{r-1} C_{d-2}^{l-1} S(d, r, i, l, c-1)}{C_{M_d - 1}^{i-1}}, \quad (6)$$

$r - 1 \le i \le M_d - (d-1) + 1$, where

$$S(d, r, i, l, c) = \sum_{i_1 = l-1}^{i-1} Q'(l, i_1, c) \sum_{t=r-l}^{d-l} C_{d-l-1}^{t-1} \cdot \sum_{i_2 = t-1}^{i-1-i_1} Q(t, i_2) C_{M_{d-l-t}}^{i-1-i_1-i_2}, \quad (7)$$

and

$$Q'(l, j, c) = \begin{cases} 1 & l = 1, j = c = 0; \\ \displaystyle\sum_{i-l-1}^{j} \sum_{c'=\max\{1, c+i-j\}}^{c} p(l, l, c', i) \frac{C_{M_l}^{(l-1-c, j)} C_{j-i}^{c-c'}}{C_{M_l - i}^{l-1-c'}} & 1 \le c \le l-1, l-2 \le j-c \le M_{l-1}; \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Proof: Eq. (6) can be proved similarly with the proof of Eq. (3). In $G(d-1; n, k)$, given an optimal $(r-1, d-1, k)-$regeneration tree produced by Algorithm 1, its bottleneck edge (V_a, V_b) is the i^{th} maximum edge in $E(d)$, $0 \le a < b < d$. We remove all the edges in $E(d)$ whose weight is less than (V_a, V_b). Divide V_d into two groups, $V_{(a)}$ and $V_{(b)}$.

Let V_a belong to $V_{(a)}$ and V_b belong to $V_{(b)}$. Assume $V_{(a)}$ contains l nodes. Let $S(d,r,i,l,c)$ represent the number of ways of assigning $i-1$ edges into $V_{(a)}$ and $V_{(b)}$ to form an $(r-1,d-1,k)-$regeneration tree. Thus replacing $R(d,r,i,l)$ by $S(d,r,i,l,c)$ in Eq. (3), we can obtain the proof of Eq. (6).

Let $V_{(a)}$ and $V_{(b)}$ contain V_a and V_b, respectively. Without loss of generality, we assume that $V_{(a)}$ contains V_0. Note that V_a may equal V_0. We define $Q'(l,j,c)$ as the number of connected graphs on $V_{(a)}$ (l nodes with one given root V_0) with j edges in which there are c edges in E_{root} having weights larger than the i^{th} edge in $E(d)$. Thus Eq. (7) can be obtained from Eq. (4) by replacing $Q(l,i_1)$ with $Q'(l,i_1,c)$.

Now we prove Eq. (8). We define $Q'(l,j,c)=1$ when $l=1$ and $j=c=0$. Otherwise $Q'(l,j,c)>0$ if and only the following two conditions are both satisfied:

1) Since there are l nodes in $V_{(a)}$, there should be at most $l-1$ edges with weights no less than the i^{th} maximum edge in E_d. Meanwhile, in order to guarantee the connectivity of the $(r,d,k)-$regeneration tree, there is at least one edge in E_{root} with weight larger than the i^{th} maximum edge in E_d, *i.e.*, $1\le c\le l-1$.
2) Nodes in $V_{(a)}-V_0$ are connected. Thus $l-2\le j-c\le M_{l-1}$.

When the two conditions above are satisfied, all the l nodes are connected, so $l-1\le j\le M_l$. Assume that the bottleneck edge of the maximum spanning tree in such a graph satisfying the conditions above is the i^{th} largest edge among the j edges, and the degree of the root in the maximum spanning tree is c'. We have $\max\{1,c+i-j\}\le c'\le c$. The number of such kind of maximum spanning trees is $C^i_{M_l}p(l,l,c',i)$. There are still $j-i$ edges to be assigned into the graph. Among the $j-i$ edges, $c-c'$ edges should be assigned to connect the root ($l-1-c$ candidates of positions), and $j-i-c+c'$ edges to be assigned to connect the non-root nodes ($M_l-i-l+c'+1$ candidates of positions). Thus the number of graphs is $C^i_{M_l}p(l,l,c',i)C^{c-c'}_{l-1-c'}C^{j-i-c+c'}_{M_l-i-l+c'+1}$. Because $j-i-c+c'\ge 0$, $c'\ge c+i-j$, $Q'(l,j,c)=\sum\limits_{i=l-1}^{j}\sum\limits_{\max\{1,c+i-j\}}^{c}C^i_{M_l}p(l,l,c',i)C^{c-c'}_{l-1-c'}C^{j-i-c+c'}_{M_l-i-l+c'+1}=\sum\limits_{i-l-1}^{j}\sum\limits_{c'=\max\{1,c+i-j\}}^{c}p(l,l,c',i)\frac{C^{(l-1-c,j)}_{M_l}C^{c-c'}_{j-i}}{C^{l-1-c'}_{M_l-i}}$. ■

Based on Lemma 2, we can obtain the probability of $\sigma_{\text{RCTREE}}(r,G(d;n,k))$, by simply checking whether there are enough non-selected edges in E_{root} so that the bottleneck bandwidth will not be affected.

Theorem 3: Let $p^{\text{RCTREE}(k)}(d+1,r+1;i)$ be the probability that $\sigma_{\text{RCTREE}}(r,G(d;n,k))=i$ in $G(d;n,k)$.

$$p^{\text{RCTREE}(k)}(d,r;i)=\sum_{c=r-k}^{r-1}p(d,r,c;i)+\sum_{c=1}^{r-k-1}\sum_{j<i}p(d,r,c;j)\frac{C^{r-k-c-1}_{i-1-j}C^{d-1-r+k}_{M_d-i}}{C^{d-1-c}_{M_d-j}}. \quad (9)$$

Proof: In $G(d-1;n,k)$, according to Algorithm 1, if the optimal $(r-1,d-1)-$regeneration tree produced by Prim's algorithm (Line 1 – Line 5) is also an $(r-1,d-1,k)-$regeneration tree, or there are enough non-selected edges in E_{root} with weights larger than the i^{th} edges in E_d, the bottleneck edge of the optimal $(r-1,d-1)-$regeneration tree will still be the bottleneck edge of the optimal $(r-1,d-1,k)-$regeneration tree. These cases occur with probability $\sum\limits_{c=(r-1)-k+1}^{r-1}p(d,r,c;i)$.

However, if the degree of the root of the $(r-1,d-1)-$regeneration tree produced by Prim's algorithm is less than $r-k$, and there is no enough non-selected edge to be added into the tree (Line 7 — Line 9 in Algorithm 1), the bottleneck bandwidth of the optimal $(r-1,d-1,k)-$regeneration tree is less than that of the optimal $(r-1,d-1)-$regeneration tree. Assume that there are c edges in E_{root} with weights larger than the i^{th} maximum edge in E_d, $1\le c\le r-k-1$. Because of the degree constraint of the root, $r-k-c$ edges with weights less than the i^{th} maximum edge should be added into the optimal $(r-1,d-1)-$regeneration tree. If the minimum edge added is the j^{th} edge in $E_d, j>i$, this is equivalent to selecting $d-1-c$ edges, in which the $(r-k-c-1)^{th}$ edge is the j^{th} edge in E_{root}, from a total number of M_d-j edges. This probability is $\frac{C^{r-k-c-1}_{i-1-j}C^{d-1-r+k}_{M_d-i}}{C^{d-1-c}_{M_d-j}}$. Thus, the probability of this kind of cases is $\sum\limits_{c=1}^{r-k-1}\sum\limits_{j<i}p(d,r,c;j)\frac{C^{r-k-c-1}_{i-1-j}C^{d-1-r+k}_{M_d-i}}{C^{d-1-c}_{M_d-j}}$. ■

Similar to Eq. (5), we obtain $E_{\text{RCTREE}}(r,G(d;n,k))$, the expected bottleneck bandwidth of the optimal $(r,d,k)-$regeneration tree in $G(d;n,k)$:

$$E_{\text{RCTREE}}(r,G(d;n,k))=\sum_{i=r}^{M_{d+1}-d+1}p^{\text{RCTREE}(k)}(d+1,r+1;i)E_{(i:M_{d+1})}. \quad (10)$$

C. Quantitative Results

In this section, we compare the regeneration schemes of STAR, TREE, and RCTREE by a quantitative evaluation. We assume that in $G(d;k,n)=(V(d+1),E(d+1),\omega)$, ω, the weight of the edge in $E(d+1)$, satisfies a uniform distribution $U[0.3\text{Mbps}, 120\text{Mbps}]$, which reveals the bandwidth capacity between nodes in PlanetLab [9].

By the theory of order statistics [10], we obtain the value of $E_{(i:M_{k+1})}$ under the distribution of $U[a,b]$, where $a=0.3$Mbps, and $b=120$Mbps:

$$E_{(i:M_{k+1})}=\frac{(b-a)(M_{k+1}-i+1)}{M_{k+1}+1}+a. \quad (11)$$

Since the bottleneck edge of STAR with r providers in $G(d;n,k)$ should be the r^{th} maximum edge in E_{root}, we obtain its bottleneck bandwidth by Eq. (12):

$$E_{\text{STAR}}(r,G(d;n,k))=(b-a)\frac{d-r+1}{d+1}+a. \quad (12)$$

We compare $B(r,G(d;n,k))$, the virtual bottleneck bandwidth of STAR, TREE, RCSTAR (STAR with

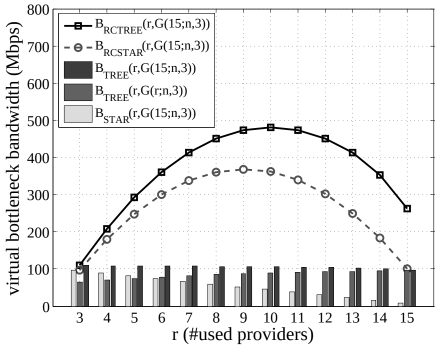


Fig. 3: Virtual bottleneck bandwidth of five regeneration schemes.

regenerating codes), and RCTREE. The regeneration time is $\frac{\mathcal{M}}{k \cdot B(r, G(d;n,k))}$. For STAR and TREE, $B_{\text{STAR}}(r, G(d;n,k)) = E_{\text{STAR}}(r, G(d;n,k))$ and $B_{\text{TREE}}(r, G(d;n,k)) = E_{\text{TREE}}(r, G(d;n,k))$. For RCSTAR and RCTREE, $B_{\text{RCSTAR}}(r, G(d;n,k)) = (r-k+1)E_{\text{STAR}}(r, G(d;n,k))$ and $B_{\text{RCTREE}}(r, G(d;n,k)) = (r-k+1)E_{\text{RCTREE}}(r, G(d;n,k))$, because the amount of transferred data on each edge in $G(d;n,k)$ is $\frac{1}{r-k+1} \cdot \frac{\mathcal{M}}{k}$ bits.

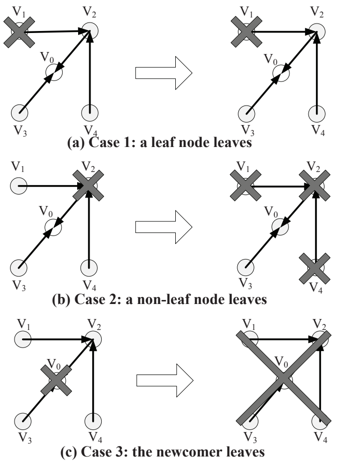
Fig. 3 shows the evaluation result in $G(15;n,3)$, $n \geq 15$. r is the number of providers. With the power of regenerating codes, the virtual bottleneck bandwidth of RCTREE and RCSTAR is improved significantly, compared with TREE and STAR. On the other hand, even though the network traffic on each edge is reduced by regeneration codes, the virtual bottleneck bandwidth of RCTREE and RCSTAR can not increase monotonically. For RCTREE, its topology is constrained by the degree of the root. For RCSTAR, moreover, its bottleneck bandwidth decreases with the increased number of providers, since it is based on STAR. When $r = 10(9)$, the curve of the virtual bottleneck bandwidth of RCTREE (RCSTAR) reaches its peak. When $r = 10$, the virtual bottleneck bandwidth of RCSTAR, TREE, and STAR are 75%, 22%, and 9% of RCTREE, respectively. *RCTREE outperforms all other schemes by combining tree-structured regeneration, which utilizes high-bandwidth links more efficiently, with regenerating codes, which reduces the regeneration traffic significantly.*

V. Lifetime of Regeneration Trees

A. Regeneration with Node Departures

We have analyzed the tree-structured regeneration and its combination with regenerating codes. However, we have not considered that nodes may leave during the regeneration. Fig. 4 shows some examples of node departures during the regeneration.

In Fig. 4, Case 1, 2, and 3 are three examples that a leaf node, a non-leaf node, and the newcomer in a regeneration tree leave the network, respectively. In Case 1, after the leaf node leaves the network, a regeneration tree with 3 providers remains. In Case 2, after the non-leaf node leaves the network, all its children nodes should be regarded as leaving the network, because the data can not be transferred to V_0 until another regeneration tree has been constructed. In Case 3, the newcomer leaves the system. Apparently the regeneration fails, since no data can be regenerated at the newcomer any more.



(a) Case 1: a leaf node leaves

(b) Case 2: a non-leaf node leaves

(c) Case 3: the newcomer leaves

Fig. 4: Examples of node departures in a regeneration tree.

Since STAR can be regarded as a special form of TREE, and RCTREE constructs a regeneration tree with the degree constraint of the root, we discuss the continuous transmission time in the regeneration tree with node departures. Assume that in a regeneration tree with r providers, V_0 is still connected with r_0 providers after a node leaves the network. The connected component containing V_0 is a subtree of the original regeneration tree. For RCTREE, the subtree also satisfies the degree constraint of the root because the degree of V_0 decreases by one at most. If $r_0 \geq k$, the regeneration tree is still alive, because the newcomer can still receive data from at least k providers.

Definition 5: The lifetime of a regeneration tree in $G(d;n,k)$ is the time between when the tree is constructed and when less than k providers remain in the subtree.

We assume that nodes do not leave simultaneously in the regeneration. In the regeneration tree, we assume that all the nodes may leave the network with the same probability. All the nodes are aware of the departures of their children nodes. Specifically, when the parent node does not receive data from one of its children nodes, it regards this node as a terminated node and stops the data transmission. If the redundancy is coded by regenerating codes, the encoding coefficients of the regenerating codes may change with the departures of the providers. Dividing coded blocks into generations with suitable size may solve this problem.

B. Lifetime of STAR and TREE

Assume that in a regeneration tree with r nodes, t nodes remain after one node leaves the network. This occurs with probability $Pr(r,t)$. The lifetime is the time the regeneration tree keeps stable plus the lifetime of the remaining subtree with t nodes. Then we obtain a recursion of lifetime. First we consider the value of $Pr(r,t)$.

Lemma 3: For STAR,

$$Pr^{\text{STAR}}(r,t) = \begin{cases} \frac{1}{r} & t = 0; \\ 0 & 0 < t < r-1; \\ \frac{r-1}{r} & t = r-1. \end{cases} \tag{13}$$

Proof: When the newcomer leaves the network, $t = 0$. Since any node may leave the network with the same probability, $Pr^{\text{STAR}}(r, 0) = \frac{1}{r}$. Otherwise, t can only become $t - 1$, because any providers are leaf nodes in STAR. Thus $Pr(r, r-1) = 1 - \frac{1}{r}$. ■

Lemma 4: For TREE,

$$Pr^{\text{TREE}}(r,t) = \begin{cases} \frac{1}{r} & t = 0; \\ \frac{r-1}{r} \cdot \frac{C_{r-2}^{r-t-1} t^{t-1} (r-t)^{r-t-2}}{r^{r-2}} & 0 < t < r. \end{cases} \quad (14)$$

Proof: When $t = 0$, the proof can be seen in Lemma 3. When $0 < t < r$, let e be the edge connecting the leaving node with its parent node. The departure of this node can be regarded as removing e from the regeneration tree. Removing e will divide the regeneration tree into two subtrees. There are C_{r-2}^{r-t-1} ways to select t nodes in the subtree containing the newcomer V_0, because V_0 and the leaving node have been selected. By Cayley's formula [11], the number of spanning trees on n labeled nodes is n^{n-2}. Thus the number of regeneration trees which will have t providers remaining after removing e is $t^{t-2}(r-t)^{r-t-2} \cdot t$. Since the number of regeneration trees over r nodes is r^{r-2}, $Pr^{\text{TREE}}(r,t) = \frac{r-1}{r} \cdot \frac{C_{r-t-1}^{r-2} t^{t-1} (r-t)^{r-t-2}}{r^{r-2}}$. ■

Comparing Theorem 3 with Theorem 4, we can see STAR can be more stable than TREE, because if the newcomer does not leave, the regeneration tree will only lose at most one provider in STAR, but it may lose more than one providers in TREE, if a non-leaf provider leaves.

Now we give the recursion of lifetime for STAR and TREE.

Theorem 4: Let $L(r)$ be the expected lifetime of a regeneration tree with r providers. $E(r)$ is the expected time that all the r nodes remain in the regeneration tree. In $G(d; n, k)$, for STAR,

$$L^{\text{STAR}(k)}(r-1) = E(r) + \sum_{t=k+1}^{t=r-1} Pr^{\text{STAR}}(r,t) L^{\text{STAR}(k)}(t-1). \quad (15)$$

For TREE, replace $L^{\text{STAR}(k)}(r-1)$ and $Pr^{\text{STAR}}(r,t)$ with $L^{\text{TREE}(k)}(r-1)$ and $Pr^{\text{TREE}}(r,t)$, respectively.

Proof: For a regeneration tree in STAR or TREE with r nodes ($r - 1$ providers), the expected time that the tree keeps stable is $E(r)$. When a node leaves, the expected lifetime of the subtree is $L(t-1)$ if t nodes remain, $k + 1 \leq t < r$. We get the expected lifetime by adding $E(r)$ with the expected lifetime of the remaining subtree. ■

C. Lifetime of RCTREE

Now we discuss the lifetime of RCTREE. We also find a recursion of lifetime, by discussing how many providers remain after the node departure. We first introduce a lemma as follows.

Lemma 5: [11] Over n labeled nodes in which k node have been designated as roots, the number of forests containing k rooted trees is kn^{n-k-1}.

Corollary 2: Over n labeled nodes in which one node has been designated as root, the number of spanning trees in which the degree of the root is k, is $T(n,k) = C_{n-2}^{k-1}(n-1)^{n-k-1}$, $1 \leq k < n$.

Proof: Given a spanning tree in the statement, we can remove the root node to make it become a forest with k trees. Thus the number of spanning trees in the statement equals the number of ways to select k nodes from $n - 1$ nodes, multiplying the number of such forests by Lemma 5, *i.e.*, $C_{n-1}^{k} \cdot k(n-1)^{n-1-k-1} = C_{n-2}^{k-1}(n-1)^{n-k-1}$. ■

Specifically, if $n = 1$, let $T(n,k) = 1$ when $k = 0$, and $T(n,k) = 0$ otherwise.

Lemma 6: In $G(d; n, k)$, given an $(r-1, d, k)-$regeneration tree in which the degree of V_0 is c, after a node departure, a subtree with t nodes remains. The degree of V_0 is still c with probability $Pr_0^{\text{RCTREE}}(r,t,c)$, and become $c-1$ with probability $Pr_1^{\text{RCTREE}}(r,t,c)$. When $0 < t < r$,

$$Pr_0^{\text{RCTREE}}(r,t,c) = \frac{r-1}{r} \cdot \frac{C_{r-2}^{r-t-1}(t-1)T(t,c)(r-t)^{r-t-2}}{T(r,c)}. \quad (16)$$

$$Pr_1^{\text{RCTREE}}(r,t,c) = \frac{r-1}{r} \cdot \frac{C_{r-2}^{r-t-1} T(t,c-1)(r-t)^{r-t-2}}{T(r,c)} \quad (17)$$

when $c = t = 1$ or $1 < c \leq r - 1$, and equals 0 otherwise.

Proof: To prove this lemma, we refer to the proof of Lemma 4. The proofs of Eq. (16) and Eq. (17) are the same with the proof of Eq. (14) except for three points. First, since V_0 in the remaining subtree after removing e satisfies the degree constraint, the numbers of such subtrees are $T(t,c)$ and $T(t, c-1)$, respectively. Second, for $Pr_0^{\text{RCTREE}}(r,t,c)$, e can connect to any $t - 1$ nodes in the subtree, but for $Pr_1^{\text{RCTREE}}(r,t,c)$, e can only connect to V_0. Third, over r labeled nodes, there are $T(r,c)$ regeneration trees where the degree of V_0 is c.

Moreover, for $Pr_1^{\text{RCTREE}}(r,t,c)$, when $c = 1$, it is impossible that $t > 1$, because if the only node connecting V_0 leaves, only the newcomer will remain in the subtree. ■

Similar to Theorem 4, we obtain the following lemma.

Lemma 7: Let $L^{\text{RCTREE}(k)}(r,c)$ be the lifetime of an $(r,d,k)-$regeneration tree in which the degree of V_0 is $c, c \geq r - k + 1$.

$$\begin{aligned} L^{\text{RCTREE}(k)}(r-1,c) = E(r) + \\ \sum_{t=k+1}^{t=r-1} [Pr_0^{\text{RCTREE}}(r,t,c) L^{\text{RCTREE}(k)}(t-1,c) + \\ Pr_1^{\text{RCTREE}}(r,t,c) L^{\text{RCTREE}(k)}(t-1,c-1)]. \quad (18) \end{aligned}$$

Theorem 5: Given an optimal $(r,d,k)-$regeneration tree produced by Algorithm 1 in $G(d; n, k)$, its lifetime is

$$\begin{aligned} L^{\text{RCTREE}(k)}(r) = \frac{\sum_{c=1}^{r-k} T(r+1,c)}{\sum_{c=1}^{r} T(r+1,c)} L^{\text{RCTREE}(k)}(r, r-k+1) \\ + \sum_{d=r-k+1}^{r} \frac{T(r+1,d)}{\sum_{c=1}^{r} T(r+1,c)} L^{\text{RCTREE}(k)}(r,d). \quad (19) \end{aligned}$$

Proof: In Algorithm 1, if the degree of V_0 in the optimal $(r,d)-$regeneration tree produced by Prim's algorithm (Line 1

– Line 5) is invalid, this happens with probability $\frac{\sum_{c=1}^{r-k} T(r+1,c)}{\sum_{c=1}^{r} T(r+1,c)}$. Then the edges in E_{root} will be added to the tree until the degree of the root is $r-k+1$, so the lifetime is $L^{\text{RCTREE}(k)}(r, r-k+1)$. On the other hand, if the degree of V_0 is d, $r-k+1 \leq d \leq r$, which happens with probability $\frac{T(r+1,d)}{\sum_{c=1}^{r} T(r+1,c)}$, the lifetime is $L^{\text{RCTREE}(k)}(r, d)$. ■

D. Comparison

If the active time of a node in the network between one join and one departure satisfies an exponential distribution $\exp(1/\lambda)$, $E(r) = \frac{\lambda}{r}$. We let $\lambda = 690584.29149$ seconds according to the user behaviors in PlanetLab [12], and then obtain the expected lifetime of STAR, TREE, and RCTREE in $G(15; n, 3), n \geq 15$, illustrated by Fig. 5.

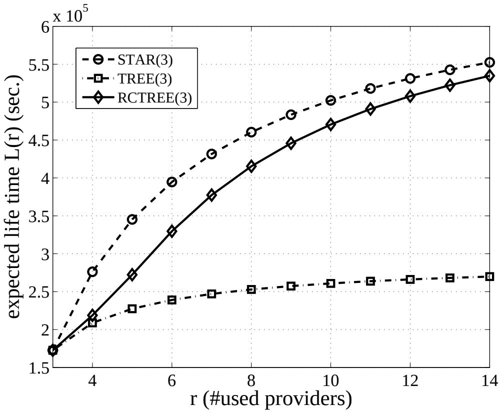


Fig. 5: Expected lifetime of STAR, TREE, and RCTREE in $G(15; n, 3), n \geq 15$.

In Fig. 5, the lifetime of all three schemes increases with r, because more providers can resist better towards node departures. STAR has the best lifetime, because the departure of one provider will not incur the loss of any other providers. However, for TREE, since the departure of one provider usually leads to the loss of some other providers in the regeneration tree, its lifetime is less than 60% of STAR when $r \geq 7$. Due to the degree constraint of V_0, RCTREE is much more stable than TREE. When r is large enough, RCTREE is quite similar to STAR from the perspective of the degree of V_0. When $r \geq 8$, the lifetime of RCTREE is more than 90% of STAR and still continues approaching STAR. Since the curve of the virtual bottleneck bandwidth of RCTREE reaches its peak in Fig. 3 when $r = 10$, and the virtual bottleneck bandwidth is improved significant by RCTREE compared with STAR, we can ignore the minor improvement of lifetime of STAR in practice.

VI. Conclusion

In this paper, we address challenges in constructing the regeneration tree in distributed storage systems with regenerating codes. We first analyze the constructive algorithm and its bottleneck bandwidth of the tree-structured regeneration with a variable number of providers (TREE). Based on this analysis, we discuss the tree-structured regeneration combined with regenerating codes (RCTREE) and analyze its bottleneck bandwidth. Considering the node churn in distributed storage systems, we make an analysis of the lifetime of TREE, RCTREE, and the conventional star-structured regeneration (STAR). Our analysis results show that RCTREE is not only the fastest scheme, but also a very stable scheme. Therefore, RCTREE is suitable for distributed storage systems, especially for the system with a substantial degree of bandwidth heterogeneity. In our future work, we will validate the theoretical advantage of RCTREE by real-platform based simulations and experiments.

Acknowledgment

We thank the anonymous reviewers for their helpful advices. This work was supported in part by NSFC under Grant No. 60702054, Shanghai Municipal R&D Foundation under Grant No. 09511501200, the Shanghai Rising-Star Program under Grant No. 08QA14009, NSERC Discovery Grant RGPIN 238994-06 and NSERC Strategic Grant STPGP 364910-08. Xin Wang is the corresponding author.

References

[1] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: system support for automated availability management," in *Proc. of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*. Berkeley, CA, USA: USENIX Association, 2004.

[2] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *in Proc. of INFOCOM*, pp. 2000–2008, May 2007.

[3] R. Rodrigues and B. Liskov, "High availability in dhts: Erasure coding vs. replication," *in Proc. of 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[4] Y. Wu, R. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," *Allerton Conference on Control, Computing, and Communication*, 2007.

[5] S. Acedański, S. Deb, M. Médard, and R. Koetter, "How good is random linear coding based distributed networked storage?" *in Proc. of 1st Workshop on Network Coding, WiOpt*, Apr. 2005.

[6] A. Duminuco and E. W. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," *in Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2009.

[7] J. Li, S. Yang, X. Wang, X. Xue, and B. Li, "Tree-structured data regeneration with network coding in distributed storage systems," *in Proc. of 17th IEEE International Workshop on Quality of Service (IWQoS)*, 2009.

[8] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocation problems," *in Proc. of Fifth Workshop on Network Coding, Theory and Applications (NetCod)*, 2009.

[9] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between PlanetLab nodes," *Passive and Active Network Measurement*, pp. 292–305, 2005.

[10] H. A. David and H. N. Nagaraja, *Order Statistics*, 3rd ed. Wiley-Interscience, Aug. 2003.

[11] A. Cayley, "A theorem on trees," *Quart. J. Math. 23 (1889)*, pp. 376–378.

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