Heterogeneity-Aware Data Regeneration in Distributed Storage Systems

Yan Wang[+], Dongsheng Wei[+], Xunrui Yin[*], and Xin Wang[+]

[+]School of Computer Science, Fudan University, Shanghai, China
[*]Department of Computer Science, University of Calgary, Canada

***Abstract*—Distributed storage systems provide large-scale reliable data storage services by spreading redundancy across a large group of storage nodes. In such big systems, node failures take place on a regular basis. When a node fails or leaves the system, to maintain the same level of redundancy, it is expected to regenerate the redundant data at a replacement node as soon as possible. Previous studies aim to minimize the network traffic in the regeneration process, but in practical networks, where link capacities vary in a wide range, minimizing network traffic does not always mean minimizing regeneration time. Considering the heterogeneous link capacities, Li *et al.* proposed a tree-structured regeneration scheme, called RCTREE, to bypass the low-capacitated link encountered in direct transmissions. However, we find that RCTREE may rapidly lose data integrity after several regenerations.**

In this paper, we reconsider the problem of minimizing regeneration time in networks with heterogeneous link capacities. We derive the minimum amount of data to be transmitted through each link to preserve data integrity. We prove that building an optimal regeneration tree is NP-complete and propose a heuristic algorithm for a near-optimal solution. We further introduce a flexible regeneration scheme, which allows providers to generate different amount of coded data. Simulation results show that the flexible tree-structured regeneration scheme can reduce the regeneration time significantly.

I. Introduction

Large-scale distributed storage systems are widely used today to provide reliable data storage service, by spreading data redundancy over a large number of storage nodes. Users can access their data anytime anywhere. Such application scenarios include large data centers like Google File Systems [2], Total Recall [3], OceanStore [4] and peer-to-peer storage systems such as Wuala [5].

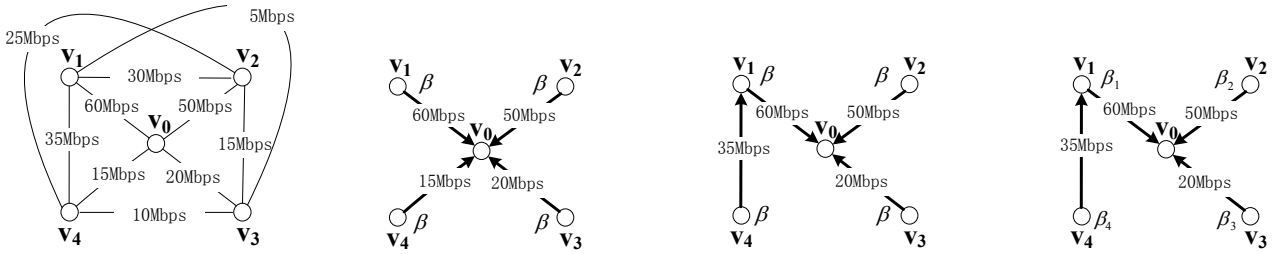
In such big systems, nodes fail frequently and the failures should be handled on a routine basis. After a node fails or leaves the system, the reliability degrades and the protected data becomes vulnerable. To maintain the same level of redundancy, it is important to regenerate the lost data at a replacement node, called *newcomer*, as soon as possible [2]. A direct way to solve this problem is to minimize the amount of data transferred during the repair process. Thus, Dimakis *et al.* proposed Regenerating Codes [6] to minimize the total repair bandwidth. They found that there is a trade-off between repair bandwidth and storage efficiency, with two extremal cases: the *minimum-storage regenerating* (MSR) point where each node stores the minimum amount of data, and the *minimum-bandwidth regenerating* (MBR) point where the storage efficiency is sacrificed for minimum repair bandwidth. For simplicity, they assumed each surviving node participating in the regeneration, called *provider*, sends the same amount of data to the newcomer.

However, minimizing repair bandwidth does not always mean minimizing regeneration time, especially in heterogeneous networks where link capacities vary in a wide range. The heterogeneous networks are common in distributed storage systems. For example, within a data center, servers are usually placed in racks, and servers in the same rack enjoy much higher bandwidth than those located in different racks [7]. Meanwhile, the available bandwidths among servers are different because of different background traffics, even if the link capacities are the same [8]. The difference of link bandwidths becomes even larger when using multiple geo-distributed data centers for distributed storage, which is a conventional practice for large companies, *e.g.* Google, to safeguard their users' data from the failure of an entire data center [9]. In these networks, the time of regeneration process is determined not only by the amount of data transmitted from providers to the newcomer, but also by the bandwidths of bottleneck links between them.

For fast regeneration in heterogeneous networks, Li *et al.* [1] have proposed a *tree-structured* topology data regeneration with regenerating codes, called *RCTREE*. They achieve high utilization of network links by building a tree-structured path to transfer data, in addition to reducing the regeneration traffic by coding data on the intermediate node of the tree. However, because of transmitting insufficient amount of data, their codes may lose data integrity, *i.e.*, the data collector may fail to reconstruct the original file. Specifically, a distributed storage system is usually described with parameters (n, k, d), where n is the number of storage nodes, k is the number of nodes accessed by data collector to reconstruct the file, and d is the number of providers participating in the regeneration. In literature, the ability to reconstruct the file from any k storage nodes is called *MDS property* [10].

In this paper, we study the problem of minimizing regeneration time in heterogeneous networks while maintaining the

Yan Wang is also with School of Software, East China Jiao Tong University, Nanchang, China. This work is supported in part by the National Science Foundation of China under Grant 61171074, Program for New Century Excellent Talents in University under Grant NCET-11-0113, Shanghai Municipal R&D Foundation under Grant No. 13511500400 and Projects of East China Jiaotong University(10RJ02). Xin Wang is the corresponding author.



(a) network model with one newcomer and four storage nodes

(b) conventional star-structured regeneration scheme (STAR)

(c) tree-structured regeneration scheme (TR)

(d) flexible tree-structured regeneration scheme (FTR)

Fig. 1. Examples for tree regeneration schemes: STAR, TR and FTR. The parameters are $n = 5$, $d = 4$, $k = 2$, $M = 480$Mb, $\alpha = M/k = 240$Mb, $\beta = \frac{\alpha}{d-k+1} = 80$Mb. The regeneration time of STAR, TR and FTR is 5.33s, 4s and 3s, respectively.

MDS property. Our contributions are composed of two parts.

First, we reconsider the tree-structured regeneration approach with regenerating codes, for which we develop the information flow graph method from [6]. Using this method, we derive the minimum amount of data to be transmitted on each link of a given regeneration tree, and formulate the problem of building an optimal regeneration tree to minimize the regeneration time. Unfortunately, we find this problem is NP-complete, mainly because the flow on each link exhibits a correlation with the number of providers using this link. We propose a heuristic algorithm, called Tree-structured Regeneration (TR), to find a near-optimal regeneration tree.

Second, we introduce a new approach to further reduce the regeneration time by flexibly determining the amount of data generated by each provider. In previous studies such as regenerating codes and RCTREE, they both assume that each provider generates the same amount of coded data to be transmitted to the newcomer. However, we find that this assumption is not necessary. If we allow the provider with low end-to-end bandwidth to generate less data, the regeneration time will be further reduced. Note that as a compromise, the provider with high end-to-end bandwidth will have to generate more coded data to maintain the MDS property. We derive a sufficient condition for determining the amount of data generated by each provider to preserve the MDS property, and combine it with the tree-structured regeneration scheme. The resultant regeneration scheme is called Flexible Tree-structured Regeneration (FTR).

These ideas can be intuitively illustrated with the example shown in Fig. 1. Consider the overlay network shown in Fig. 1(a), where v_0 is the newcomer, and v_1, v_2, v_3, v_4 are the $d = 4$ providers. The bandwidths are labeled on the links, which range from 5Mbps to 60Mbps. We assume that the redundancy is coded as an $(n = 5, k = 2)$-MDS code, such that any 2 out of 5 storage nodes are able to reconstruct the file. Suppose the size of the original file is $M = 480$Mb, and each storage node stores $\alpha = M/k = 240$Mb. In order to regenerate the lost data at v_0, regenerating codes require downloading $\beta = \frac{M}{k(d-k+1)} = 80$Mb data from each provider. Fig. 1(b) shows the conventional regeneration scheme (STAR) which uses the star-structured topology: v_0 receives data directly from the four providers. The regeneration time of STAR is determined by the slowest transmission which takes $\frac{\beta}{15Mbps} = 5.33$ seconds. Fig. 1(c) shows the regeneration tree used by TR and RCTREE. They coincide with the same tree in this example. In RCTREE, each provider generates β coded data with its local storage, encodes them with its received data (if any) and finally transmit β data to its parent node, which means v_1 only transmits β data to v_0 in the example. We find that this is not sufficient for the MDS property (refer to Sec. II for more details). According to our analysis, the minimum amount of data transferred on edge (v_1, v_0) is $2\beta = 160$Mb at least. Thus, it costs TR $\max\{\frac{2\beta}{60Mbps}, \frac{\beta}{20Mbps}\} = 4$ seconds to accomplish the regeneration. Fig. 1(d) shows the flexible tree-structured regeneration (FTR) scheme. Each provider is allowed to generate different amount of coded data. We find that if providers v_1, v_2, v_3, v_4 generate $\beta_1 = 90$Mb, $\beta_2 = 90$Mb, $\beta_3 = 60$Mb, $\beta_4 = 90$Mb, respectively, and v_1 transmits $\beta_1 + \beta_4 = 180$Mb to v_0, the MDS property is maintained. The regeneration time of FTR is reduced to $\max\{\frac{\beta_1+\beta_4}{60Mbps}, \frac{\beta_3}{20Mbps}, \frac{\beta_4}{35Mbps}, \frac{\beta_2}{50Mbps}\} = 3$ seconds.

To evaluate the performances of our algorithms, we implement all these regeneration schemes and run simulations with practical link capacities. Experimental results show that, according to the variance of link capacities, our proposed algorithm can reduce the regeneration time by $10\% \sim 90\%$, compared with conventional regenerating codes. And the FTR scheme even outperforms RCTREE in many scenarios.

The remainder of the paper is organized as follows. In Sec. II, we formulate the process of data regeneration in distributed storage systems and analyze the loss of MDS property in RCTREE by theory and practice. In Sec. III, we reconsider the tree-structured regeneration scheme by analyzing the corresponding information flow graph. In Sec. IV, we propose a flexible scheme based on tree-structured regeneration. We evaluate the performance of our two proposed regeneration schemes in Sec. V. We introduce the related works in Sec. VI and conclude the paper in Sec. VII.

II. Problem Formulation

Assume a file consisted of M blocks is encoded into $n\alpha$ blocks and stored in n storage nodes. Each node holds α blocks. It is required that the file can be reconstructed by

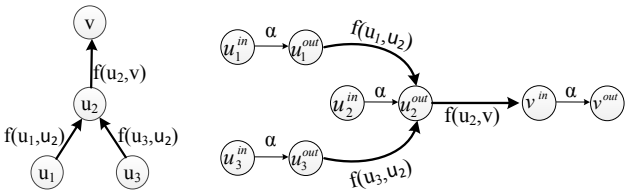
accessing any k storage nodes. After a storage node fails, d providers are accessed to regenerate α blocks at a newcomer. We use a complete graph $G(V, E)$ to represent the overlay network consisted of the d providers and the newcomer [1][11]. For any two nodes $u, v \in V$, let $c(u, v)$ denote the link capacity from node u to v.

With regenerating codes, each provider generates β coded blocks by encoding its own α blocks and directly transmits them to the newcomer. The d provider-to-newcomer flows form a star topology centered at the newcomer. However, in real world overlay networks, end-to-end links usually have different capacities, which may vary in one or two orders of magnitude [12]. Repairing with the star topology may suffer from the low link capacity between some provider and the newcomer.

Considering the heterogeneous link capacity and the bottleneck effect, Li *et al.* [1] proposed a tree-structured regeneration scheme which transmits the regeneration traffic along a carefully selected tree spanning all providers. In this way, they may by-pass low capacitated links and perform coding at the intermediate nodes of the tree to save bandwidth. However, their method cannot maintain the MDS property. To illustrate this, we first extend the information flow graph for the tree-structured regeneration and then use its with a specific example to show the loss of MDS property caused by RCTREE.

A. Information Flow Graph for Tree-structured Regeneration

Information flow graph is first introduced by Dimakis *et al.* [6] to study the sufficient and necessary condition of the MDS property. As their method is based on the star topology, we need to generalize the construction of the information flow graph for arbitrary tree-structured regeneration topology.



(a) regeneration tree (b) corresponding part of information flow graph

Fig. 2. The part of information flow graph corresponding to the regeneration of storage node v (newcomer). Assume $d = 3$ and u_1, u_2, u_3 are the providers.

In particular, we construct the information flow graph in the following way. For each storage node u, create two nodes u^{in} and u^{out} and a link of capacity α from u^{in} to u^{out}. Create a source node s and for each initial storage node u, add a link from s to u^{in} with infinite capacity. For each regenerated storage node, we add links according to its regeneration procedure, which is specified by the regeneration tree T and the amount of data $f(u, v)$ transmitted on each link of the tree. During the regeneration, if provider u transmits $f(u, w)$ blocks to provider w, we add a link from u^{out} to w^{out} with capacity $f(u, w)$, and if provider u transmits $f(u, v)$ blocks to the newcomer v, we add a link from u^{out} to v^{in} with capacity $f(u, v)$. For example, Fig. 2(a) shows the regeneration tree consisted of 3 providers and a newcomer, the corresponding part of information flow graph is shown in Fig. 2(b).

For a data collector (DC) which connects to k storage nodes to reconstruct the file, we add k links from the k "out" nodes to DC with infinite capacity. As long as the min-cut separating source s from DC is no less than the file size M, DC can reconstruct the file [6].

B. Loss of MDS property in RCTREE

Assume a file of size $M = 480$Mb is stored on 5 storage nodes $v_1, v_2, \cdots, v_5$. Each node holds $\alpha = 240$Mb data and the MDS property requires that the file can be reconstructed by any two storage nodes. Suppose v_5 fails, v_0 is the newcomer and $v_1, \cdots, v_4$ are the $d = 4$ providers. Let the example network shown in Fig. 1(a) be the overlay network.

In this example, RCTREE will use the same regeneration tree as shown in Fig. 1(c), but transmit $\beta = \frac{M}{k(d-k+1)} = 80$Mb on each link. Assume the data collector connects to v_0 and v_3 to reconstruct the file. Fig. 3 shows the information flow graph. As noted in the figure, there is a cut of volume $2\beta+\alpha = 400$Mb, which is smaller than the file size. Therefore, the file cannot be reconstructed with storage nodes v_3, v_0.

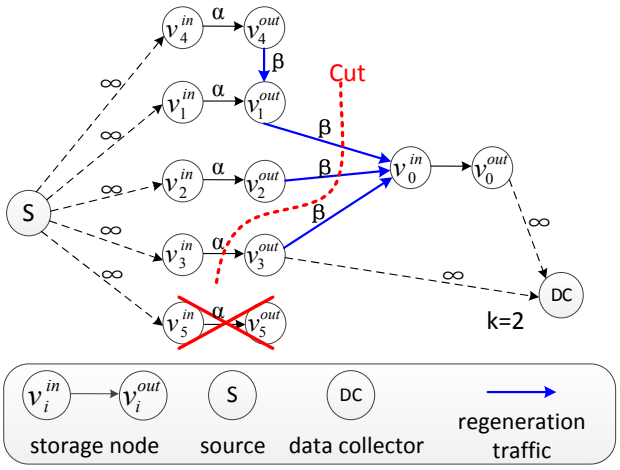


Fig. 3. An example of RCTREE and the corresponding information flow graph. The parameters are $n = 5, d = 4, k = 2, M = 480$Mb, $\alpha = M/k = 240$Mb, $\beta = \frac{M}{k(d-k+1)} = 80$Mb. A min-cut with capacity $2\beta + \alpha = 400$Mb is illustrated by a red dashed line, implying that a DC can not reconstruct the original file by connecting with $\{v_3, v_0\}$.

To find out how frequently such failures of reconstruction occur, we have implemented RCTREE scheme based on Random Linear Regenerating Codes(RLRC) and run simulations with practical parameters. Random linear coding is performed over a sufficiently large finite field $GF(2^{16})$, so that the probability of reconstruction failure caused by generating linearly dependent blocks can be ignored [13]. Fig. 4 shows the probability that data collector successfully reconstructes the file. From the results, we can see that data collector can hardly reconstruct the original file after 5 rounds of regenerations.

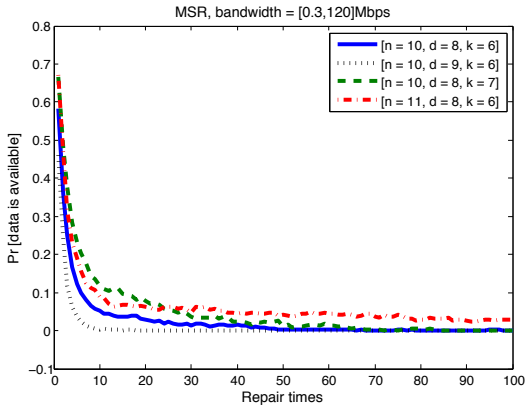


Fig. 4. The impact of repair times to the probability of successfully reconstructing the original file.

C. Summary

From the simulation results, we can see that it is necessary to reconsider the problem of optimizing regeneration time with heterogeneous link capacities and maintaining the MDS property at the same time.

Specifically, given the topology of the overlay network $G(V, E)$ and link capacities $c(u, v), (u, v) \in E$, let $f(u, v)$ denote the number of blocks transmitted along each link (u, v). Our target is to minimize the regeneration time. Considering that the coding operation are performed parallel with the data transmission, which dominates the regeneration time, we may simply represent the regeneration time as

$$\max\{\frac{f(u,v)}{c(u,v)}|(u,v) \in E\}$$

Our contributions consist of two parts. First, we reconsider the basic tree-structured regeneration scheme where each provider is assumed to generate the same amount of coded blocks. Second, we relax the assumption and further reduce the regeneration time by flexibly determining the amount of information generated by each provider.

III. Tree-structured Regeneration Scheme with Regenerating Codes

A tree-structured regeneration solution has two parts: the regeneration tree $T \subset E$ and the number of blocks $f(u, v)$ transmitted on each link of the tree. In this section, we first study the minimum $f(u, v)$ to preserve the MDS property with a given regeneration tree. Thereafter, we show that the problem of building an optimal regeneration tree is NP-complete. Finally, we conclude this section with a heuristic algorithm to construct a regeneration tree.

A. Determine $f(u, v)$ for a given regeneration tree

We use the information flow graph as constructed in Sec. II-A to determine the minimum flow $f(u, v)$ on each link that ensures the MDS property.

Theorem 1: For a given regeneration tree T rooted at the newcomer, in order to preserve the MDS property, the minimum number of blocks transmitted on each link $(u, v) \in T$ is

$$\min\{m_u\beta, \alpha\}$$

where m_u is the number of nodes in the sub-tree rooted at u. Here β is the number of blocks transmitted in the regenerating codes, which satisfies $\sum_{i=1}^{k} \min\{(d - i + 1)\beta, \alpha\} = M$.

Proof: Construct the information flow graph as described in Sec. II-A. The key is to compute the min-cut of this information flow graph. Let $[U, \bar{U}]$ denote a min-cut separating data collector DC from the source, where U is the set of nodes containing DC. Then U must contain at least k "out" nodes. Label the corresponding k storage nodes as $u_1, u_2, \cdots, u_k$ in topological order, such that u_i is a provider in regenerating u_j only if $i < j$. We divide the cut links into k sets. If $u_i^{in} \notin U$, the i-th set i contains only one link $u_i^{in} \to u_i^{out}$. Otherwise, the i-th set contains all the cut-links introduced in the repair of node u_i (Fig. 5). According to our scheme, a link $u \to v$ has flow rate $m_u\beta$ only if the subtree rooted at u has m_u nodes. As there are at most $i - 1$ providers in U repairing u_i, there are at most $d - i + 1$ providers not in U. Thus the total flow rate of the i-th set is at least $\min\{(d - i + 1)\beta, \alpha\}$.

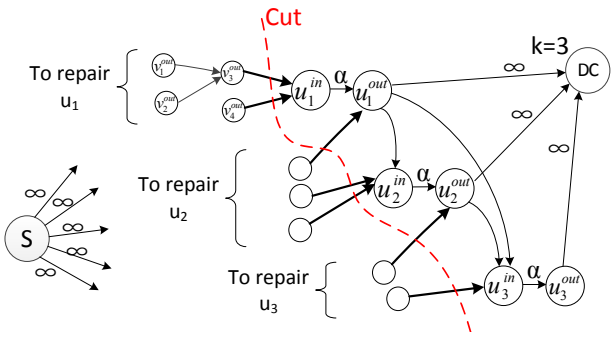


Fig. 5. An example of min-cut in an information flow graph in tree-structured regeneration process. The parameters are $d = 4, k = 3$. The red dotted line shows the min-cut and the darkened links are the cut-links.

Sum up the volume of each link set, the volume of the cut $[U, \bar{U}]$ is at least $\sum_{i=1}^{k} \min\{(d - i + 1)\beta, \alpha\} = M$. Thus, if $f(u, v) \geq \min\{m_u\beta, \alpha\}$, DC can construct the file by assessing any k storage nodes.

To show that we cannot reduce $f(u, v)$ further, we need to show that the min-cut of volume M is achievable. According to the information flow graph constructed in the proof, we can see that such a min-cut is achievable if we require each provider provides β blocks, which means the solution $\min\{m_u\beta, \alpha\}$ is optimal for a given regeneration tree. ■

B. Construction of the optimal regeneration tree

With the knowledge of how much information to be transmitted on each link, the remaining task is to build an optimal regeneration tree T to minimize the regeneration time. However, we find that the optimal regeneration tree problem, ORT

for short, is NP-complete. To demonstrate this, we start with the formal definition of the ORT problem.

Definition 1: For a given overlay network $G(V, E)$ with link capacity $c(u, v), (u, v) \in E$, the optimal regeneration tree problem is to find a spanning tree T such that its regeneration time $\max\{\frac{f(u,v)}{c(u,v)} | (u, v) \in T\}$ is minimized, where $f(u, v) = \min\{m_u\beta, \alpha\}$ and m_u is the number of nodes in the sub-tree rooted at u.

To study the complexity of ORT problem, it is equivalent to restate this optimization problem as a decision problem, which aims to determine whether the regeneration time of an optimal regeneration tree is no more than 1. The following theorem shows that this problem is NP-complete.

Theorem 2: The ORT problem is NP-complete.

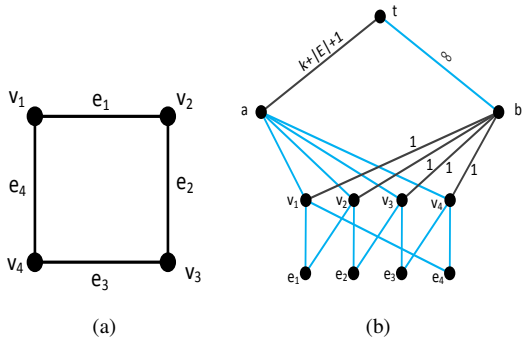
Proof: We first show that ORT $\in$ NP. Suppose we are given a graph $G = (V, E)$. The certificate we choose is the optimal regeneration tree T. The verification algorithm checks, for each edge $(u, v) \in T$, that $\frac{f(u,v)}{c(u,v)} \leq 1$. This verification can be performed in polynomial time.

We now prove that the ORT problem is NP-hard by reduction from the VERTEX-COVER problem, which is known to be NP-complete. In particular, given an undirected graph $G = (V, E)$ and an integer k, the VERTEX-COVER problem asks whether all edges can be "covered" by k nodes, where node $u \in V$ can cover edge $e \in E$ only if they are adjacent. For an instance of the VERTEX-COVER problem, we construct a regeneration scenario in an overlay network $G' = (V', E')$, such that the regeneration time is less than 1 if and only if G has a vertex cover of size k.

We construct G' in the following way. G' has four layers of nodes. The first layer has only one node that is the root t. The second layer has two nodes, node a and node b. Both a and b are connected to the root. The link capacity of edge (a, t) is $k+|E|+1$ and the link capacity of edge (b, t) is unlimited. The nodes in the third layer correspond to the vertices in graph G, all of them are connected to both a and b. The link capacity of their edges connected to a is unlimited and the link capacity of edges connected to b is 1. The nodes in the last layer correspond to the edges in graph G. Each node in the last layer is connected to two nodes in the third layer, which is connected by the corresponding edge in graph G. The link capacity of these edges, which connect the nodes in the third layer to the nodes in the last layer, is unlimited. Links which are not mentioned in the construction are supposed to have zero capacity.

From the construction, graph G' can be constructed from G in polynomial time. Fig. 6 shows an example of this reduction. Fig. 6(a) is an instance $(G, k = 2)$ of the VERTEX-COVER problem. Fig. 6(b) is the graph G' constructed from (G, k).

We next show that this transformation of G into G' is a reduction. First, suppose that G has a vertex cover set $V' \subseteq V$, where $|V'| = k$. We claim that we can find a tree whose regeneration time is no more than 1. We construct this tree according to the vertex cover: for each node e_i of the fourth layer, let its parent be v_j which covers the edge e_i in the vertex cover of G. For each node v_i of the third layer, let its parent be a, if it belongs to the vertex cover V'. Otherwise, let its parent be b. Finally, let the parent of a and b be the root. It can be verified the regeneration time is 1.



(a) (b)

Fig. 6. Reducing VERTEX-COVER problem to Optimal Regeneration Tree problem. (a) An undirected graph $G = (V, E)$ with $k = 2$. (b) The graph produced by the reduction procedure.

Conversely, suppose that G' has a tree whose regeneration time is no more than 1. Then, we claim that the edges of G can be covered by no more than k nodes. Let $V' \subseteq V$ be the set of nodes that correspond to children of node a in the regeneration tree. First, V' is a vertex cover of G since otherwise, some nodes in the fourth layer will have to be connected to the root through b, causing some flow entering b larger than 1 and the regeneration time less than 1. Second, $|V'| \leq k$ because there are no more than $k + |E| + 1$ nodes transferring data from node a to root t, and as all the $|E|$ nodes in the fourth layer must transfer data through the link a to t, the number of third layer nodes with parent a is no more than k to ensure the flow on the link a to t is no more than $k + |E| + 1$. ■

C. The heuristic algorithm

In this section, we propose a heuristic algorithm to solve the ORT problem mentioned in Sec. III-B. The algorithm is inspired by Prim's algorithm [14] for the maximum weighted spanning tree problem. We start from a tree containing only the newcomer and iteratively add the remaining nodes to the regeneration tree until it spans all providers. In each iteration, we try all possible positions for each remaining provider and choose the best one and position to add it to the regeneration tree. The details are shown in Algorithm 1.

The most time consuming step is to test all possible positions for each provider, which has $|V|^2$ choices. Each test takes $O(|V|)$ to compute the regeneration time. Thus, the algorithm runs in polynomial time $O(|V|^3)$.

IV. Tree-structured Regeneration with Flexible End-to-End Traffics

In section III, we assume that each provider has the same amount of data to be transmitted to the newcomer. This assumption is actually not necessary. As the available bandwidth varies for different providers, we may repair faster if we allow providers to generate different number of blocks according to their bandwidths.

Algorithm 1 Find a regeneration tree T for a given network $G(V, E)$.

1: Input: Network topology $G(V, E)$, link capacity c, α, β
2: Output: Regeneration tree T
3: $T \leftarrow v_0$
4: $A \leftarrow V - \{v_0\}$
5: **while** $A \neq \varnothing$ **do**
6: **for all** $v \in A$ **do**
7: **for all** $u \in T$ **do**
8: Compute the regeneration time of tree $T \cup \{(v, u)\}$
9: If the regeneration time is better than previous choices, update $(v', u') \leftarrow (v, u)$
10: **end for**
11: **end for**
12: $T \leftarrow T \cup \{(v', u')\}$
13: $A \leftarrow A - \{v\}$
14: **end while**

In this section, we propose the Flexible Tree-structured Regeneration (FTR) scheme to further reduce the regeneration time. The idea can be illustrated by the example shown in Fig. 1. Fig. 1(c) shows a regeneration tree computed by algorithm 1, where each provider generates $\beta = 80$Mb coded data to be transmitted to the newcomer. We have computed that its regeneration time is 4s. Note that the bottleneck link is (v_3, v_0). If we require provider v_1, v_2, v_3, v_4 to generate 90Mb, 90Mb, 60Mb, 90Mb data, respectively. The actual data transmitted on each link will be $f(v_4, v_1) = f(v_2, v_0) = 90Mb, f(v_3, v_0) = 60Mb, f(v_1, v_0) = \min\{(90 + 90)Mb, \alpha = 240Mb\}$, and the regeneration time is further reduced to 3s, which saves 25% of regeneration time than TR in this example.

Let β_i denote the amount of information generated by the i^{th} provider. Note that if the providers of low-bandwidth generate less coded blocks, the high-bandwidth providers will have to generate more coded blocks to make sure the MDS property is maintained. So our first task is to analyze the explicit restrictions on β_i that ensures the MDS property.

For a regeneration tree and given $\beta_i, i = 1, 2, \cdots, d$, we require that the intermediate nodes perform coding on received blocks only when the number of blocks received from the child nodes plus the coded blocks generated by themselves is larger than α. In this way, the number of blocks transmitted on each link $f(u, v)$ will be

$$\min\{\sum_{v_i \in S(u)} \beta_i, \alpha\}$$

where $S(u)$ denotes the set of nodes in the sub-tree rooted at u. The following theorem provides a sufficient condition.

Theorem 3: The MDS property is maintained if in each regeneration, we choose $\beta_i, i = 1, 2, \cdots, d$ that satisfy

$$\sum_{l=1}^{d-k+j} \beta_{i_l} \geq \min\{(d - k + j)\beta, \alpha\} \quad \forall j = 1, 2, \cdots, k$$

where i_l is a permutation of $1, 2, \cdots, d$ such that $\beta_{i_1} \leq \beta_{i_2} \leq \cdots \leq \beta_{i_d}$.

Proof: We need to prove that any min-cut $[U, \bar{U}]$ (DC $\in U$) has volume at least M.

As the capacity from a storage node to the data collector DC is set to infinity in the information flow graph, we only need to consider the case that U contains at least k storage nodes. Let $v_1, v_2, \cdots, v_k$ be the first k storage nodes of U in the topological order. If it is an output node, the storage link of capacity α will be included in the cut. If it is an input node, set of the links repairing this node will be included in the cut.

For the j-th storage node, the number of its provider not in U will be at least $d - i + 1$. According to the way how we determine the flow on the regeneration tree, the total capacity of the cutting repair links will be at least

$$\sum_{l=1}^{d-k+j} \beta_{i_l} \geq \min\{(d - k + j)\beta, \alpha\}$$

Therefore, the volume of the cut $[U, \bar{U}]$ will be no less than M. ■

According to the analysis above, we propose an algorithm to calculate the amount of blocks β_i generated by each provider. The algorithm is based on the regeneration tree constructed in the previous section. In general, the algorithm has two steps: 1) calculate the available repair bandwidth $b_i, i = 1, 2, \cdots, d$ for each provider and rearrange the labels of providers such that $b_1 \leq b_2 \leq \cdots \leq b_d$; 2) solve the following linear programming problem for β_i:

$$\begin{aligned} \min \quad & \lambda \\ \text{subject to:} \quad & \lambda \geq \beta_i / b_i, \quad \forall i = 1, 2, \cdots, d \\ & \textstyle\sum_{l=1}^{d-k+j} \beta_l \geq \min\{(d - k + j)\beta, \alpha\}, \quad \forall j = 1, 2, \cdots, k \\ & 0 \leq \beta_1 \leq \beta_2 \leq \cdots \leq \beta_d \leq \alpha \end{aligned} \tag{1}$$

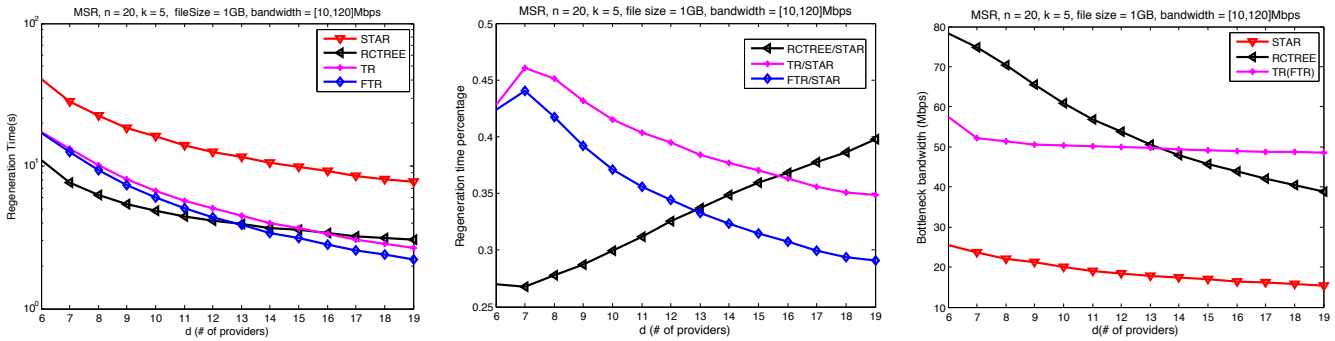
Here the variable λ is introduced for representing $\max_{1 \leq i \leq d}\{\beta_i / b_i\}$, which is ensured by the first d constraints.

The details are shown in Algorithm 2. Note that we must be careful calculating the available end-to-end bandwidth b_i from each provider to newcomer, as the provider-to-newcomer traffics may share a common link in the regeneration tree. For a link shared by m providers, we assume each provider occupies $1/m$ of the link bandwidth (Line 3 - 11).

In Algorithm 2, the estimated bandwidth b_i can be computed in time $O(|V|^3)$. The end-to-end traffic β_i can be computed by solving a linear programming problem of $d + 1$ variables and $2d + k + 1$ constraints, which can be solved in polynomial time as well.

V. Evaluation

In this section, we present simulation results to verify the effectiveness of our proposed algorithms: Tree-structured Regeneration (TR) and Flexible Tree-structured Regeneration (FTR). Our most concern is the regeneration time, which is measured as the time that the newcomer spends on regenerating the coded blocks. We ignore the the encoding time on



(a) Regeneration time

(b) Regeneration time of RCTREE, TR and FTR normalized by the regeneration time of STAR.

(c) Bottleneck bandwidth

Fig. 7. Performance of four regeneration schemes for different number of providers (d). The parameters are $n = 20, k = 5$ and $M = 1$GB.

Algorithm 2 Determine the amount of coded blocks that each provider should generate.

Input: n, k, d, M, α, a regeneration tree T with root v_0.
Output: $(\beta_1, \cdots, \beta_d)$
for $i = 1$ to d **do**
  for all edge e from v_i to v_0 **do**
    $a \leftarrow$ the bandwidth of edge e
    $m \leftarrow$ the number of nodes in the subtree rooted at edge e
    $s \leftarrow$ the smallest value of a/m
  end for
  $b_i \leftarrow s$
end for
sort $(b_1, \cdots, b_d)$ such that $b_1 \leq b_2 \leq \cdots \leq b_d$
solve the linear programming problem (1) for β_i

each provider and the decoding time on newcomer, because the encoding and decoding operations can be performed simultaneously with the transmission [1].

For default settings, we use the same experiment setup as [1], where redundant data is produced as an $(n = 20, k = 5)$-MDS code. The original file size is set to 1GB. The link capacities between these nodes obey the uniform distribution U[10Mbps,120Mbps], according to the measurement of real world networks [12].

A. *Effect of the number of providers* d

The number of providers d is a key parameter in the distributed storage system. According to [6], the total bandwidth consumed in the regeneration process decreases as d grows. Because of this, the theoretical optimal value of d is $n - 1$. But accessing a large number of providers will introduce extra communication overhead. Thus, all feasible values of d may appear in practice. In the evaluation, we vary d from $k+1$ to $n-1$, in order to find out how the factor affect the performance of each regeneration scheme. We consider the MSR point, where each node stores $M/k = 1\text{GB}/5 = 200\text{MB}$ data. The results are shown in Fig. 7.

The traditional regeneration schemes (STAR) proposed by Dimakis *et al.* in [6] is implemented as a benchmark. We also test the regeneration time of RCTREE proposed by Li *et al.* in [1], although their algorithm cannot preserve the MDS property. Fig. 7(a) illustrates the regeneration time of these schemes. We can see that the regeneration time of every schemes decreases when d increases, because total repair bandwidth decreases, and the total repair traffic is dispersed to more providers.

In Fig. 7(b), we divide the regeneration time of TR, FTR and RCTREE by the regeneration time of STAR to show the relative improvement. In general, our algorithms reduce $50\% \sim 70\%$ of regeneration time compared with STAR. As d grows, the performances become better. This is because the star topology has a large chance to include a low capacitated provider-to-newcomer link, which can be bypassed by a tree-structured regeneration scheme.

In previous sections, we have illustrated that, in RCTREE, the amount of data transferred through the regeneration tree is not enough for the MDS property, and we fix this problem by proposing the TR scheme, which transmits more data. Thus one might intuitively regard the regeneration time of RCTREE as a lower bound of the regeneration time of TR. However, in Fig. 7(b), we observe that TR algorithm performs even better than RCTREE when d is large. The reason is that TR can find a better regeneration tree than RCTREE.

To show this, we also measure the bottleneck bandwidth of the regeneration tree computed by the three algorithms: STAR, RCTREE and TR. The results are presented in Fig. 7(c). The bottleneck bandwidth of RCTREE and TR is improved significantly, compared with STAR. Since STAR always uses the d providers-to-newcomer links, but RCTREE and TR can freely select a spanning tree with edges chosen from $\binom{d+1}{2}$ links. Moreover, the bottleneck bandwidth of TR becomes better than RCTREE when d is large. The main reason is that RCTREE has a particular restriction on the regeneration tree, which requires $d - k + 1$ providers directly connected to newcomer, while TR has no such restrictions. When d is large, the regeneration tree constructed by RCTREE is more like a star-structured topology.

The performance of FTR is always better than TR in any

case. Since the process of flexibly determining β is done after TR algorithm. We also find that FTR algorithm can save more regeneration time than TR algorithm when d grows.

B. Effect of the bandwidth variance

In order to show the impact of network bandwidth variance, we run simulations with 5 different link capacity distributions: $U_1[0.3, 120]$Mbps, $U_2[3, 120]$Mbps, $U_3[30, 120]$Mbps, $U_4[60, 120]$Mbps, $U_5[90, 120]$Mbps. The number of providers d is set to 10. Results are shown in Fig. 8. In general, the performance of our algorithms is better when the variance of network bandwidth is large. For uniform distribution $U_1[0.3, 120]$Mbps, both TR and FTR can reduce 90% of regeneration time compared with the traditional STAR scheme. When the variance of network bandwidth becomes small, for example at $U_4[60, 120]$Mbps and $U_5[90, 120]$Mbps, TR has the same regeneration time as STAR, but FTR still reduces the regeneration time by $10\% \sim 20\%$.

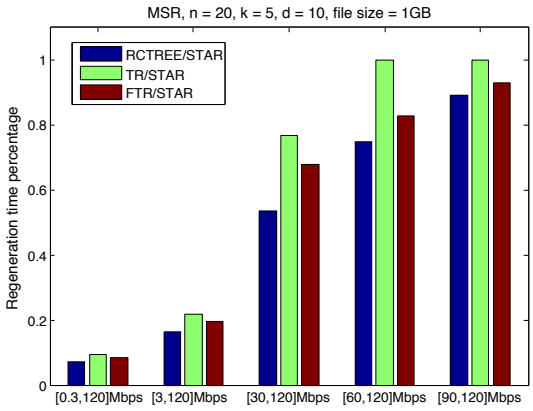


Fig. 8. Regeneration time of RCTREE, TR and FTR normalized by STAR for different bandwidth. The parameters are $n = 20, k = 5, d = 10$ and $M = 1$GB.

C. Effect of the storage capacity per node α

Our previous tests mainly focus on the MSR point, which achieves the optimal storage efficiency. However, as shown by Dimakis *et al.* [6], the repair bandwidth can be reduced by storing more data on each storage node. Specifically, the MBR point achieves the minimum repair bandwidth. We vary the storage α from the MSR point to the MBR point to test its impact on our algorithms.

Fig. 9 shows the regeneration time for different α. We can see that every regeneration schemes repair faster as α grows to the minimum repair bandwidth point and the tendencies of different repair schemes are the same. Compared with STAR, we find that the ratio of reduced regeneration time does not change much for different values of α. This implies that our previous simulation results and conclusions for the MSR case also apply to different storage α.

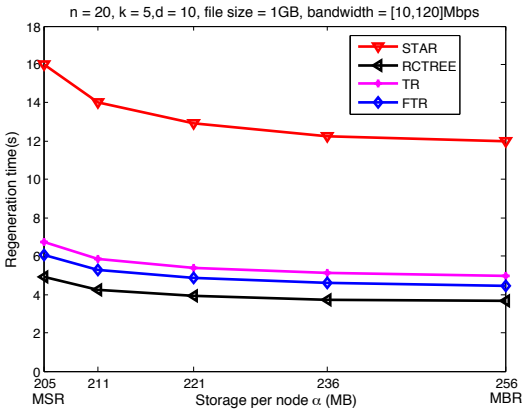


Fig. 9. Regeneration time of STAR, RCTREE, TR and FTR vs. storage capacity of each node (α), where α vary from MSR point to MBR point. The parameters are $n = 20, k = 5, d = 10$ and $M = 1$GB.

VI. Related Work

Li *et al.* [15] first considered the heterogeneity of network bandwidth in data regeneration process and proposed a tree-structured regeneration scheme to reduce the regeneration time. They also proposed a scheme of building parallel regeneration trees to further reduce the regeneration time in the network with asymmetric links [11]. However, they only discussed the case that the regeneration scheme requires k providers, which means the minimal regeneration traffic is equal to the size of original file M. To further reduce the regeneration time, they considered the regenerating codes in the tree-structured regeneration scheme and proposed RCTREE in [1]. They employ a minimum-storage regenerating (MSR) codes in RCTREE, which means the minimal regeneration traffic is $\frac{d}{k(d-k+1)}M$ bytes, *i.e.*, in a regeneration with d providers, each provider sends $\frac{\alpha}{d-k+1}$ blocks to its parent node. To make sure that the newcomer has enough information to restore α blocks, it has to receive data directly from at least $d-k+1$ providers. The details of how to construct an optimal regeneration tree can be found in Algorithm 1 in their paper [1]. Although they have done this in their algorithm to ensure that the degree of newcomer is at least $d - k + 1$, the MDS property can not be preserved after data regeneration.

Sun *et al.* [16] considered the scenario of repairing multiple data losses, and proposed two algorithms based on tree-structured regeneration to reduce the regeneration time. However, they assumed the amount of data transferred between providers and newcomer is designed to be the same as β in regenerating codes. According to our analysis, their regeneration schemes can not preserve the MDS property either.

Some researches, such as [17, 18], considered the heterogeneity of nodes availability and optimized the erasure code deployment to reduce the data redundancy. Some researches, such as [19–23], jointly considered the repair-cost and heterogeneity of communication(download) cost on each links. They flexibly determine the amount of data to minimize the total repair cost, which is different from the regeneration time.

VII. Conclusion

We reconsider the problem of how to reduce the regeneration time in networks with heterogeneous link capacities. We analyze the minimum amount of data to be transmitted on each link of the regeneration tree and prove that the problem of building optimal regeneration tree is NP-complete. We propose a heuristic algorithm to construct a near-optimal regeneration tree and further reduce the regeneration time by allowing non-uniform end-to-end repair traffics. Simulation results show that our regeneration schemes are able to maintain the MDS property and reduce the regeneration time by 10% $\sim$ 90%, compared with traditional star-structured regenerating codes. With the non-uniform end-to-end repair traffics, we can flexibly determine the amount of coded data generated by each provider. The proposed Flexible Tree-structured Regeneration scheme performs even better than RCTREE.

References

[1] J. Li, S. Yang, X. Wang, and B. Li, "Tree-structured Data Regeneration in Distributed Storage Systems with Regenerating Codes," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, 2010.

[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

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

[4] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer *et al.*, "Oceanstore: An architecture for global-scale persistent storage," *ACM Sigplan Notices*, vol. 35, no. 11, pp. 190–201, 2000.

[5] Wuala-Secure Cloud Storage. [Online]. Available: http://www.wuala.com/

[6] A. G. Dimakis, P. B. Godfrey, M. J. W. Y. Wu, and K. Ramchandran, "Network Coding for Distributed Storage System," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[7] B. Gastón, J. Pujol, and M. Villanueva, "A realistic distributed storage system: the rack model," *arXiv preprint arXiv:1302.5657*, 2013.

[8] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10. ACM, 2010, pp. 267–280.

[9] Google Datacenters. [Online]. Available: http://www.google.com/about/datacenters/inside/data-security/index.html

[10] P. P. C. L. Yuchong Hu, Henry C. H. Chen and Y. Tang, "Nccloud: applying network coding for the storage repair in a cloud-of-clouds," in *Proceedings of USSENIX Conference on File and Storage Technologies(FAST)*, 2012.

[11] J. Li, S. Yang, and X. Wang, "Building parallel regeneration trees in distributed storage systems with asymmetric links," in *2010 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2010.

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

[13] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *29th IEEE International Conference on Distributed Computing Systems(ICDCS'09)*. IEEE, 2009.

[14] Prims algorithm. [Online]. Available: http://en.wikipedia.org/wiki/Primsalgorithm

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

[16] W. Sun, Y. Wang, and X. Pei, "Tree-structured parallel regeneration for multiple data losses in distributed storage systems based on erasure codes," *Communications, China*, vol. 10, no. 4, pp. 113–125, 2013.

[17] L. Pamies-Juarez, P. Garcia-Lopez, and M. Sanchez-Artigas, "Heterogeneity-aware erasure codes for peer-to-peer storage systems," in *International Conference on Parallel Processing (ICPP)*, 2009.

[18] G. Xu, S. Lin, G. Wang, X. Liu, K. Shi, and H. Zhang, "Hero: Heterogeneity-aware erasure coded redundancy optimal allocation for reliable storage in distributed networks," in *International Performance Computing and Communications Conference (IPCCC)*, 2012.

[19] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "Cost-bandwidth Tradeoff in Distributed Storage Systems," *Computer Communications*, vol. 33, no. 17, pp. 2105–2115, 2010.

[20] M. Gerami, M. Xiao, and M. Skoglund, "Optimal-cost Repair in Multi-hop Distributed Storage Systems," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, 2011, pp. 1437–1441.

[21] S. Akhlaghi, A. Kiani, and M. Ghanavati, "A fundamental trade-off between the download cost and repair bandwidth in distributed storage systems," in *Proceedings of IEEE International Symposium on Network Coding (NetCod)*, 2010.

[22] C. Armstrong and A. Vardy, "Distributed storage with communication costs," in *Proceedings of Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011.

[23] N. Shah, K. V. Rashmi, and P. Kumar, "A flexible class of regenerating codes for distributed storage," in *Proceedings of IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2010.