International Journal of Mathematical, Engineering and Management Sciences
Vol. 7, No. 2, 299-311, 2022
https://doi.org/10.33889/IJMEMS.2022.7.2.020

Optimizing Regeneration Time by Node Selection in Group Repair Code

Swati Mittal
Department of Computer Science and Engineering,
ASET, Amity University Uttar Pradesh, Noida, India.
Corresponding author: ssingal@amity.edu

Anu Mehra
Department of Electronics and Communication Engineering,
ASET, Amity University Uttar Pradesh, Noida, India.
E-mail: amehra@amity.edu

Nitin Rakesh
Department of Computer Science and Engineering,
Sharda University, Greater Noida, India.
E-mail: nitin.rakesh@gmail.com

Rakesh Matam
Department of Computer Science and Engineering,
Indian Institute of Information Technology (IIIT), Guwahati, India.
E-mail: rakesh@iiitg.ac.in

(Received on September 11, 2021; Accepted on January 14, 2022)

Abstract
Distributed storage systems use network coding techniques like replication, erasure codes, local codes, regeneration codes, hybrid code, double code and group repair code to store data efficiently and provide speedy recovery of data during failures. The performance of these approaches is mainly compared on the basis of storage required and repair bandwidth. Out of these, Group Repair Codes is the one that has optimal repair bandwidth for regeneration of nodes. Traditionally, the cost of regeneration was considered to be dependent on the number of nodes participating in the process and the amount of data being transferred. There was not much discussion on the heterogeneity of the network and the capacity of the links between the nodes. In real-time, the nodes are connected to each other with different link capacities due to which the same amount of data takes different duration in reaching its destination. Selecting the node with higher link capacity helps in reducing the data transfer time. So, considering the heterogeneous nature of the network, this paper reduces the regeneration time for Group Repair Codes (GRC). The node selection algorithms for data regeneration have been proposed for GRC and the results of simulation show significant improvement in the regeneration time. Further, the network coding in heterogeneous systems may be explored for factors like network traffic, intermediate nodes, data routing etc.

Keywords- Group repair codes, Regeneration time, Heterogeneous link capacity, Node selection, Replication, Erasure codes

1. Introduction
Network coding has been widely used for storing the data in distributed and cloud storage systems to provide 100% availability of data. Replicating the data at multiple places makes it readily available to users and helps in recovering data in case of node failures. Replication and erasure codes are two techniques used for storing the data (Weatherspoon & Kubiatowicz, 2002). These were used in Google file system (Ghemawat et al., 2003), OceanStore (Kubiatowicz et al.,

2000), Facebook-HDFS (Xia et al., 2015). Replication stores the original data of M bytes on three different nodes. If any node fails, it is replaced by a new node and its data is recovered by transferring M bytes from other node. Problem with replication was huge storage requirements of $3M$ bytes and high cost of repair. The storage was optimized by network erasure codes that divides the data of M bytes into k blocks, adds r redundant parities to it ($r < k$) and then stores the data in $n = k + r$ nodes. Each node stores M/k bytes of data. To obtain the original data as well as for recovering the data of a failed node, any k out of n nodes are required. The erasure codes optimize the storage requirement, but the cost of repair was the same as that of replication.

The Regeneration Code (RC) (Dimakis et al., 2010) implemented in Amazon Cloud and NCCloud (Hu et al., 2012) reduces the cost of repair by reducing the amount of data transmitted during repair. It encodes the data in such a manner that the amount of data stored in each node is slightly greater than $\frac{M}{k}$ bytes. In case of failures, the data of a failed node is recovered by connecting to surviving nodes that transfers a part of its data rather than transferring its entire data, thereby reducing the amount of data transferred for repairing a node. Piggybacked-RS codes and PM-MBR, PM-RBT codes (Rashmi et al., 2015, 2017) are some codes based on erasure and regeneration codes with slight modifications in order to reduce the storage requirements, the repair bandwidth, data reads and to minimize the disk input/output.

Another concept of maintaining local parities of data for smaller set of nodes was applied in Windows Azure Storage (WAS) and Xorbus (Huang et al., 2012; Kolosov et al., 2020; Sathiamoorthy et al., 2013) for faster recovery of data. This method is the Local Regeneration or Repair Codes (LRC). A similar approach is the locally decodable code (Yekhanin, 2011), largely applicable in sensor network and reliable data communication for secret sharing or Private Information Retrieval (PIR). Locally Recoverable Codes (Tamo & Barg, 2014), describes $(n, k, r) LRC$ code that generates n length codewords from k information symbols and has r locality, such that $1 \leq r \leq k$. Here, locality means the number of nodes participating in the recovery process. All these local codes were beneficial in reducing the cost of repair.

Other than these, the hybrid code applied in DHT (Rodrigues & Liskov, 2005) and the double codes (Araujo et al., 2011; Mohan et al., 2015) also proved beneficial for storing the data and provide optimal repair bandwidth. These techniques used a combination of replication and erasure codes. The amount of data transferred for repairing failures is $\frac{M}{k}$ bytes, but due to replication, the amount of storage space required is very large. The Group Repair Codes (GRC) (Mittal et al., 2018) reduces the storage requirements and still provides the same optimal repair bandwidth as the hybrid and the double codes.

Traditionally, the nodes in the system were considered to be homogeneous in nature. The cost of repair was considered to be dependent on the number of nodes participating in the process and the amount of data being transferred. Network topology, links capacity, network traffic, capacity of each node, load at a particular node etc. also play a vital role in time taken for transferring the data between the nodes. The heterogeneous nature of nodes in storage system has been investigated by various researchers. Researchers (Li et al., 2009) focused on the network topology of system, to efficiently utilize network links and to reduce the regeneration traffic. The optimal regeneration tree was produced by finding the maximum spanning tree using Prim's and Kruskal's algorithm. Research was also done for data regeneration for the regeneration codes (Li et al., 2010). For this RCTREE was proposed, which combines the advantage of regenerating codes with a tree-structured regeneration topology to achieve both fast and stable regeneration even with the departure of storage nodes during the regeneration. Concepts of repair topology and retrieval sets were applied for regeneration in a heterogeneous network (Yu et al., 2012). Researchers (Wang et al., 2014) had also proposed node selection algorithms for selecting a newcomer node and provider nodes from a set of nodes for heterogeneous distributed storage systems to optimize the cost of repair. The newcomer and provider selection has been investigated by different authors to optimize the cost of repair and minimize the regeneration time in the network that has different link capacities between the nodes for transferring the data (Gong et al., 2015; Jia et al., 2015; 2016).

Researcher (Zhang et al., 2016) also implemented tree topology for faster computing and improved repair throughput. Some researchers also focused on the repair degree and the amount of information downloaded from each helper node (Qu et al. 2018). The flexible reconstruction degree, where each node in the system has nonuniform repair bandwidth and nonuniform storage capacity was discussed by Benerjee and Gupta (2021). A hybrid genetic algorithm to determine the optimal repair tree was proposed by Ye et al. (2021) that considered the network topology and link bandwidth of storage nodes to reduce the delay of faulty node repair. Recently, replicas prioritized MobileRE was used to solve the fault tolerance problem in a mobile distributed system with fluctuating network bandwidth and high failure probability (Wu et al., 2021).

All these studies carried out for heterogeneous network motivated us to investigate the data regeneration for heterogeneous network. Since, the Group Repair Codes (GRC) have the minimum repair bandwidth and require less storage space, this paper focuses on further optimizing the data regeneration under heterogeneous network. A network that stores the data using the GRC codes was considered to have heterogeneous capacity of links between the nodes. When a node fails, it is replaced by a newcomer node and its data is recovered by transferring the data from the provider nodes in the network. By selecting the provider that has high link capacity to the newcomer node, the data will be transferred faster and the regeneration time will be reduced. The GRC has two different kinds of node- the data node and the parity node. This paper presents algorithms for selecting the provider and the newcomer nodes for optimizing the cost of regeneration for both these node failures. The results of these algorithms have been compared with random node selection in order to prove the efficiency of these algorithms.

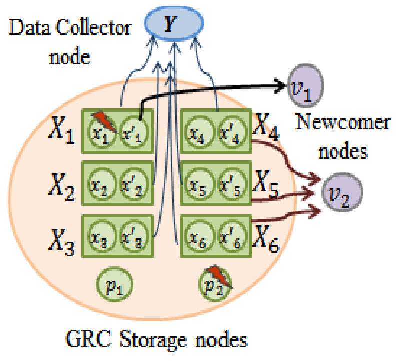
The remainder of this paper is structured as follows: The optimal Group Repair Codes (GRC) have been described briefly in Section 2 along with comparing its performance with other approaches like replication, erasure codes, local regeneration codes, hybrid and double coding. In Section 3, the effect of heterogeneity for GRC has been discussed. The system design and node selection algorithms for data regeneration have also been presented in Section 3. The results and analysis of the proposed algorithms have been discussed in Section 4. Finally, the conclusion and future research direction in this area is given in Section 5.

2. Overview of the Group Repair Codes

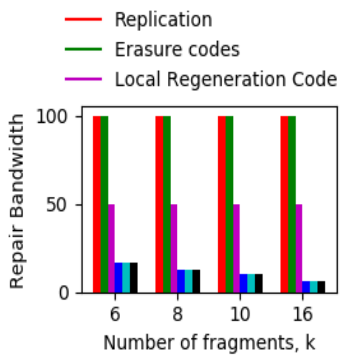
The Group Repair Codes (GRC) is a network coding technique that maintains distributed storage data by providing optimal repair bandwidth and occupies less storage space. The GRC divides the original data of size M bytes into k fragments, copies data of each fragment to two separate nodes, forming $2k$ data nodes. These k fragments are then grouped into l local groups, such that each group has $\frac{k}{l}$ original fragments stored over $\frac{2k}{l}$ nodes. The parity values are calculated for each local group to produce l local parity nodes. So, each group has $\frac{2k}{l}+1$ nodes. Each fragment is stored in two different nodes which help in high availability of data and faster recovery in case of node failures.

Figure 1(a) represents GRC for k= 6 fragments. The nodes $x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6$ and $x'_1,\ x'_2,\ x'_3,\ x'_4,\ x'_5,\ x'_6$ are data replica nodes that are divided into $l=2$ groups, which will have 3 original fragment. The parity node $p_1$is calculated from first group $(x_1, x'_1, x_2, x'_2, x_3, x'_3)$ and parity p_2 from second group $(x_4, x'_4, x_5, x'_5, x_6,\ x'_6)$. For simplification, we have used X_i to represent $\{x_i, x'_i\}$ pair of replica data nodes. Thus, the total number of nodes used in GRC is $n = 2k+l$ (i.e. $2k$ data nodes and l parity nodes). Each node in the network stores $\frac{M}{k}$ bytes of data, so the total storage required is $\frac{M}{k}(2k+l)$ bytes.

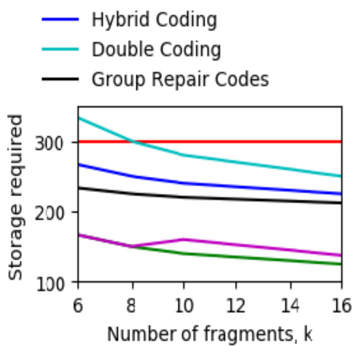
For reconstructing the original data, k data nodes having different fragments transfer their entire data to the data collector node, Y. In case of failures, the regeneration of node depends on the type of node that has failed i.e. the data node or the parity node. For repairing *the data node*, only one node that has its replica transfers its entire data to the newcomer node that takes its place. This is shown in Figure 1(a), where the failed data node x_1 has to be replaced by newcomer node v_1 and its data is recovered from its replica node x'_1. For repairing *the parity node*, it is recovered within the local group by connecting to $\frac{k}{l}$ nodes of the group containing different data fragments. Figure 1(a) also shows the recovery of parity node p_2. The newcomer node v_2 takes its place and its data is regenerated by calculating local parity of the group to which the parity node p_2 belong. It connects to any one of the two node in each data node pair X_4, X_5 and X_6.



a) GRC for k=6



b) Graph for repair bandwidth



c) Graph for storage required

Figure 1. Group repair codes (GRC)

In Figure 1(b) and 1(c), the GRC has been compared with replication, erasure, local regeneration, hybrid and double codes. Figure 1(b) represents the graph for the repair bandwidth against the different values of number of fragments k, when data of size M=100 Mb is to be stored in the system and the value of local groups $l=2$. It shows that the repair bandwidth for replication and erasure codes is the maximum and is same for all values of k. For hybrid code, double codes and GRC, the value of repair bandwidth is the minimum and decreases with increase in the value of k. For the number of fragments k=6, GRC is 83.3% better than replication and erasure codes and 66.6% better than local regeneration codes. Figure 1(c) represents the graph for the storage required by the system, plotted against the different values of number of fragments k. Replication occupies a fixed 300 Mb space for 3 replicas. The erasure codes occupy the minimum storage space. Among the optimal repair bandwidth codes i.e. the hybrid, double and GRC codes, the total storage when M=100 Mb and k=6, is 266.6 Mb, 333.3 Mb and 233.3 Mb respectively. It is hence seen that GRC has optimal repair bandwidth with comparatively less storage requirements. The next section describes how nodes selection can be beneficial in reducing the time taken for regeneration in GRC under a heterogeneous network with different link capacities between nodes.

3. Effect of Heterogeneity in Group Repair Code

In real time applications, the links in the network have different link capacities. Due to this, the same amount of data takes different time to reach the destination. If the link capacity between the nodes is less then it takes more time for transferring the data and if link capacity is high then data travels faster. This means the time for regeneration can be reduced by selecting the nodes with greater link capacities. If a process demands data transfer from multiple nodes, then the time for processing depends on the bottleneck link capacity between nodes represented by B_{min}. If each node stores $\frac{M}{k}$ bytes of data, the time for data transfer will be $\frac{M}{k.B_{min}}$. We, therefore implement GRC in a heterogeneous network and present the node selection algorithms to reduce the regeneration time. Table 1, gives all the notations used in this paper.

Table 1. Notations used for data regeneration in Group Repair Codes.

Notation	**Description**
M	Size (in bytes) of original data to be encoded and stored using GRC
k	Number of fragments in which original data is divided
l	Number of smaller groups(each group has a local parities)
n	Total number of data nodes on which data is stored using GRC
$G(V,E)$	A network/graph of nodes in the system
V	Set of all storage nodes
$E\ (u,v)$	Set of communication link capacities from node u to node v
V_G	Set of nodes containing data coded by GRC, $V_G = \{X_1, X_2, .., X_i, .., X_k, P\}$
V_{New}	Set of newcomer nodes
X_i	Set of two data replica nodes x_i, x'_i containing same data where $(i = 1..k)$
P	Set of l parity nodes $p_1, p_2, .., p_l$
V_{pr}	Candidates for provider set of nodes
V_p	Optimal provider set for data regeneration. $V_p \subset V_{pr}$
v_r	Randomly selected newcomer node from newcomer set V_{New}. Here $v_r \in V_{New}$
v_n	Optimally selected newcomer node that replaces the failed node. Here $v_n \in V_{New}$
B_{min}	Minimum bandwidth link capacity from selected nodes

3.1 System Design

Let us consider a completely connected graph $G(V,E)$ that represents the network of nodes, where V is a set of storage nodes in the network and E is a set of link capacities between the nodes. Some nodes of network $G(V,E)$ are utilized for storing data using GRC which form a set of nodes represented by V_G. The remaining nodes form a set of unused nodes represented by

V_{New}, that are the candidates for the newcomer node which may replace the failed node when required. Thus, V_G and V_{New} are both subsets of V such that:

$$V_G \cap V_{New} = \emptyset \quad \text{and}$$
$$V_G \cup V_{New} = V$$

The GRC stores k fragments of data which are replicated to form $2k$ data nodes. These replica nodes are named x_i and x'_i and together we name it as X_i. Thus, data replica nodes in GRC is represented as $X_i = \{x_i, x'_i\}$. GRC also contains l local parity nodes, one for each group which is represented as $P = \{p_1, p_2, .., p_l\}$. Combining all these nodes of GRC we represent: $V_G = \{X_1, X_2, .., X_i, .., X_k, P\}$.

3.2 Regeneration of Data

The GRC contains two types of nodes: the data nodes and the parity nodes. The procedure for regeneration of data in case of failure of these nodes is discussed below:

i) Data node failure

In case of failure of the data node, it has to be replaced by a newcomer node and its data is recovered from its corresponding replica node. So, here the provider node is fixed. The failed node is replaced by the newcomer node from the newcomer set V_{New}. If the new node is selected such that it has the maximum link capacity from the provider node to this newcomer node, then the data is transferred faster and it takes less time for regeneration. This is shown in Figure 2(a). Suppose, the data of size M=100 Mb is stored in the network and the data node x'_2 fails, then its corresponding data replica node x_2 becomes the provider. If a newcomer node, say v_r, is chosen randomly from newcomer set V_{New} that has a link capacity of 80 Mb/s, the regeneration time to transfer $\frac{M}{k} = \frac{100}{6}$ bytes of data for this random selection is 0.208s. On the other hand, if optimized node selection technique is applied to select the node from newcomer set V_{New} with maximum link capacity, of 130 Mb/s i.e. node v_1, the data transfers much faster and the regeneration time becomes 0.128s which is 38.4% improvement from random selection. The Algorithm 1 for optimized selection of node for data node failure from a given set of newcomers is given below:

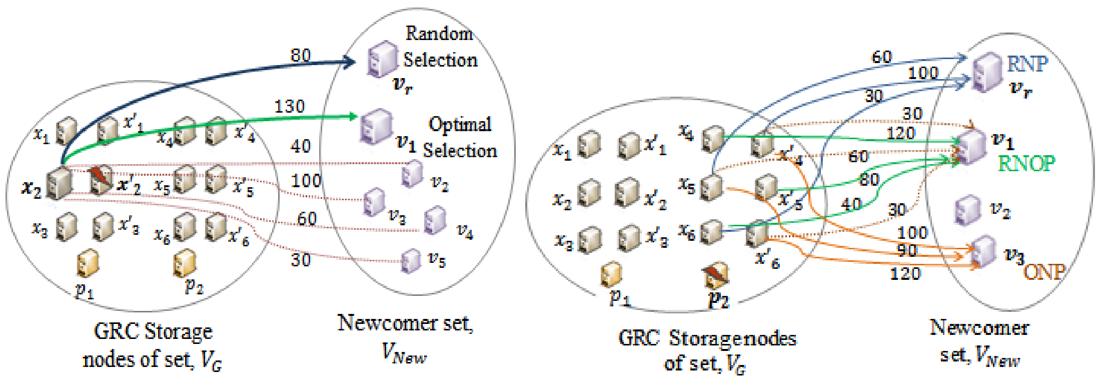
Algorithm 1: Optimized node selection for data node failures

Let there be t nodes in the newcomer set V_{New} denoted by v_j where $j = 1..t$. Thus, $V_{New} = \{v_1, .., v_j, .., v_t\}$. When a data node fails its corresponding replica node becomes the provider node, V_p. The aim of this algorithm is to select a node, v_n from a set of newcomer nodes with maximum link capacity to this provider node for faster regeneration.

Input: Graph $G(V, E)$, failed node, set of newcomer nodes V_{New}.

Output: selected node v_n that has maximum link capacity and value of maximum link capacity Max

```
1.  if node $x_i$ fails then $V_p = x'_i$
2.  if node $x'_i$ fails then $V_p = x_i$   //$V_p$ contains the provider node
3.  $Max = 0$
4.  //finding the maximum link capacity node from provider node to all nodes in newcomer set.
     for $j = 1..t$ nodes in $V_{New}$ do
         if $Max < E(V_p, v_j)$
           $Max = E(V_p, v_j)$ and $v_n = v_j$
    end for
5.  return $v_n$ and $Max$
```



a) Repairing data node failure b) Repairing parity node failure

Figure 2. Data regeneration for data node and parity node failure in Group Repair Codes.

ii) Parity node failure

When the parity node p_i fails in the GRC system, it has to be replaced by a new node and its data is recovered from the data nodes of the group to which the parity node belongs. Thus, $\frac{k}{l}$ data nodes of this group provide the data for its recovery, where k is the number of fragments and l is the number of groups. For example, if node p_2 fails, then the data is taken from three data node pairs of that group i.e. nodes X_4, X_5, X_6. This forms the candidates for the provider set of nodes, denoted by V_{pr}. Any one node from pair of two data nodes has to be selected. If for each pair of data nodes from these provider nodes, the optimized nodes are selected that have maximum link capacity then the regeneration time is reduced. The newcomer node v_n, that replaces the failed node can either be selected randomly or optimally. Based on newcomer selection, we have two algorithms for repairing parity node:-

1) Random newcomer optimal providers (RNOP)
2) Optimal newcomer and providers (ONP)

Provider node set V_p are selected from candidate for provider node set V_{pr} and the newcomer node v_n is selected from newcomer node set V_{New}. Here, the set $V_p \subset V_{pr}$ and node $v_n \in V_{New}$. The algorithm for both these node selections is as follows:

Algorithm 2: Node selection for random newcomer optimized providers (RNOP)

Let P_f be the failed parity node where f is any variable between $1..l$, and v_n is the newcomer node which takes its place. For recovery of the data first we determine the group of the failed parity node and then select appropriate node from all X_i pairs of data nodes in that group that has maximum link capacity and add it to the provider set V_p, so that regeneration time is reduced.

Input: Graph $G(V,E)$, number of fragments k, local groups l, failed parity node P_f where f is any variable between $1..l$, all data node pairs $X_i = \{x_i, x'_i\}$ where $i = 1..k$ and the fixed newcomer node v_n.

Output: provider node set V_p

```
1. If node $P_f$ fails, then $s = \frac{k}{l}(f-1)+1$,    // determine group of failed parity node.
2. $V_p \rightarrow Null$
3. If node $P_f$ fails, then $s = \frac{k}{l}(f-1)+1$,    // determine group of failed parity node.
4. $V_p \rightarrow Null$
5. for each data node pair $X_i$ where $i = s$ to $i = s + \frac{k}{l} - 1$ do,  // all data node pairs in that group
```

```
            if $E(x_i, v_n) > E(x'_i, v_n)$
               then   $V_p \leftarrow V_p \cup x_i$
            else   $V_p \leftarrow V_p \cup x'_i$   //select maximum link capacity and add it to the provider set $V_p$,
         end for
6.    return $V_p$  //provider node set for repairing failed data node
```

Algorithm 3: Node selection for optimized newcomer and providers(ONP)

Let P_f be the failed parity node where f is any variable between $1..l$. There be t nodes in the newcomer set V_{New} denoted by v_j where $j = 1..t$. Thus, $V_{New} = \{v_1, .., v_j, .., v_t\}$. For recovering data of failed parity node, first we determine the group to which the failed parity node belongs. All data node pairs X_i of this group become providers for recovering the data. This provider set is denoted by V_{pr}. Then we select the most optimal newcomer v_n and provider node set V_p that connect with maximum link capacity such that the set $V_p \subset V_{pr}$ and node $v_n \in V_{New}$.

$Min[t]$ is an array of t nodes in a newcomer set that are used to save the minimum of maximum capacity using the min-max theorem in order to find the bottleneck link capacity for calculating the regeneration time.

Input: Graph $G(V, E)$, number of fragments k, local groups l, failed parity node P_f where f is any variable between $1..l$, nodes in the newcomer set V_{New} where $V_{New} = \{v_1, .., v_j, .., v_t\}$ and all data node pairs $X_i = \{x_i, x'_i\}$ where $i = 1..k$.

Output: the selected newcomer node v_n and $V_p(v_n)$ selected provider node set corresponding to the newcomer node v_n.

```
1.    $Min[t]$={INF}
2.    $V_p(j) = \emptyset$,  for $\forall j \in V_{New}$
3.    if node $P_f$ fails then  $s = \frac{k}{l}(f-1)+1$,    //to determine group of failed parity node
4.    for each data node pair $X_i$ where  $i = s$ to $i = s + \frac{k}{l} - 1$ do,
         for $j = 1$ to $t$ do
            if $E(x_i, v_j) > E(x'_i, v_j)$
               $MaxVal = E(x_i, v_j)$ and $V_p(j) \leftarrow V_p(j) \cup x_i$
            else
               $MaxVal = E(x'_i, v_j)$ and $V_p(j) \leftarrow V_p(j) \cup x'_i$
            if $Min[j] > MaxVal$
               $Min[j] = MaxVal$
         end for
      end for
5.    $max = 0$
6.    for $j = 1$ to $t$
         if $max < Min[j]$ then
            $max = Min[j]$ and $v_n = j$
7.    return $(v_n, V_p(v_n))$   // $v_n$ is the newcomer node in set $V_{New}$ and $V_p(v_n)$ is set of provider nodes corresponding to the
      newcomer node $v_n$.
```

As shown in Figure 2(b), if a node v_r is chosen randomly along with nodes x'_4, x_5, x_6 that has a link capacity of 30 Mb/s, 60 Mb/s and 100 Mb/s respectively to the randomly selected newcomer node v_r, then the bottleneck link capacity B_{min} will be of node x'_4(i.e. 30 Mb/s). If total size of the data is M=100 Mb, then each node has $\frac{M}{k}$ data. The regeneration time to transfer $\frac{M}{k} = \frac{100}{6}$ Mb data for this random selection (RNP) calculated using $\frac{M}{k.B_{min}}$ is 0.55s.

Considering Algorithm 2, *RNOP*, assume that newcomer node v_1 is randomly selected. The Figure 2(b) shows the link capacity of all the data nodes of a group to the node v_1. For data node pair X_4 (x_4 and x'_4), the link capacity is 120 Mbps and 30 Mbps respectively. So, node x_4 is selected (shown with green line) and added to the provider set V_p. Similarly node x'_5 and x_6 have optimal values from their node pairs X_5 and X_6. So, these nodes are also added to the provider set V_p. The algorithm returns the provider set $V_p = x_4, x'_5, x_6$. Among these three optimally selected nodes, the minimum bandwidth value $B_{min}$is 40 Mbps for node x_6, so the regeneration time is 0.41s which is 25.5% better than RNP.

For Algorithm 3, *ONP*, we select the optimized newcomer node from the newcomer set V_{New} that has the maximum capacities for all pairs of data nodes in provider node set V_{pr}. The algorithm returns the optimized newcomer node v_n with its corresponding optimized provider set $V_p(v_n)$. In Figure 2(b), the optimized newcomer node is v_3 and provider nodes are x'_4, x_5, x'_6 with link capacities 100 Mbps, 90 Mbps and 120 Mbps respectively. The regeneration time for this node selection is 0.185s which is 66.3% better than RNP and 54.8% better than Algorithm 2, RNOP.

4. Result Analysis

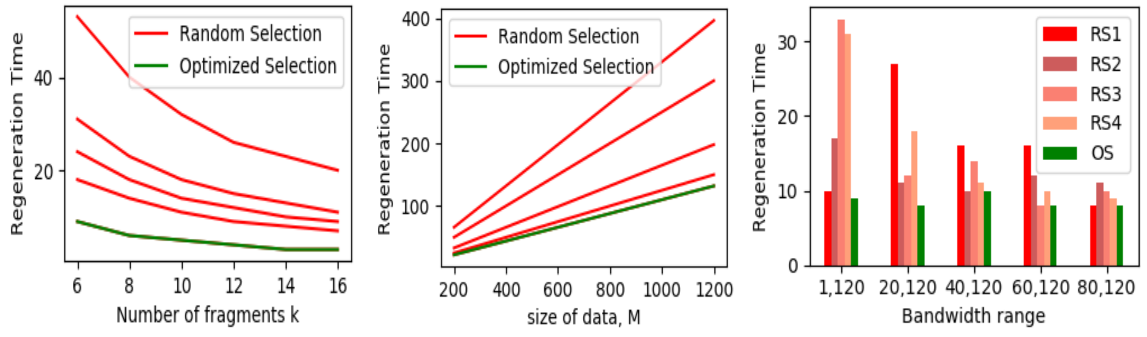
In this paper, the GRC approach that has minimum repair bandwidth and also has less storage requirement has been explored for further reducing the regeneration time under heterogeneous network having different link capacities between the nodes. Figure 3 and Figure 4, show the results obtained by simulating the node selection algorithm presented in Section 3 for repairing the data node and the parity node failures respectively. The size of data M=1000 Mb is divided into k=6 fragments and then stored in a network with bandwidth range (20 to 120) Mb/s. The comparison has been made on various parameters like number of fragments k, size of data M and bandwidth range.

Figure 3, shows the graphs of regeneration time for repairing the failed data node. When data node fails, the provider will be the corresponding replica node and the newcomer node has to be taken from newcomer pool. The regeneration time has been plotted against i) number of fragments k, ii) the data of size M stored in a network and iii) the different bandwidth range. In these graphs, four randomly selected newcomer nodes out of 100 nodes from the newcomer pool are compared with the optimized selected node using Algorithm 1, *Optimized node selection for data node failures*. The simulation results obtained are as follows:

a) **Regeneration time vs. number of fragments, k**: In Figure 3(a), the regeneration time is plotted against the gradually increasing number of fragments, k = 6, 8, .., 16. For the case given, at point k = 6, the regeneration time for optimized selected node is 55.5% better than the best randomly selected node and 85.1% better than the worst randomly selected node. The regeneration time decreases on increasing the value of k.

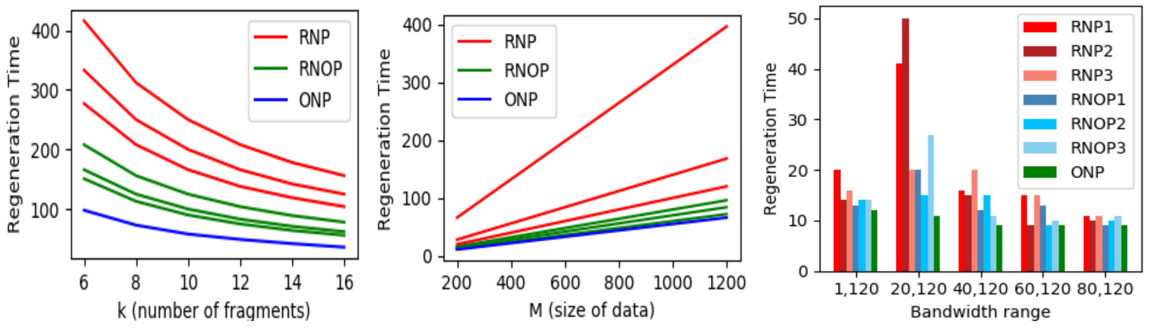
b) **Regeneration time vs. size of data, M**: In Figure 3(b), the regeneration time is plotted against the gradually increasing data size, M = 200, 400, .., 1200. For the case given, at M=1200 Mb, the regeneration time for optimized selected node is 14.2% better than the best randomly selected node and 66.7% better than the worst randomly selected node. The regeneration time increases with increasing the value of M.

c) **Regeneration time vs. bandwidth range**: In Figure 3(c), the regeneration time is plotted against different bandwidth range (1 to 120), (20 to 120), .., (80 to 120). For the bandwidth range (1 to 120) in figure, the regeneration time for optimized selected node is 20% better than the best randomly selected node and 75.7% better than the worst randomly selected node.



a) Number of data fragments ***k*** b) Size of data ***M*** c) Bandwidth range

Figure 3. Graph for regeneration time for repairing data node failure for randomly selected newcomer node and optimized newcomer node selection.



a) Number of data fragments ***k*** b) Size of data ***M*** c) Bandwidth range

Figure 4. Graph for regeneration time for repairing parity node failure for RNP, RNOP and ONP

Figure 4, shows the graphs of regeneration time for repairing the failed parity node. In these graphs, three random selections (RNP) are compared with three selections of Algorithm 2, *Node selection for random newcomer optimized providers* (RNOP) and one optimized selection obtained from Algorithm 3, *Node selection for optimized newcomer and providers* (ONP). The results are as follows:

a) **Regeneration time vs. number of fragments, *k***: In Figure 4(a), the regeneration time is plotted against the gradually increasing number of fragments, k = 6, 8, .., 16. For the case given, at k = 6, the regeneration time for ONP is 70.9% better than average of three RNP and 42.8% better than average of three RNOP.

b) **Regeneration time vs. size of data, M**: In Figure 4(b), the regeneration time is plotted against the gradually increasing data size, M = 200, 400, .., 1200. For the case given in figure, for M=1200 Mb, the regeneration time for ONP is 70% better than average of three RNP and 17.6% better than average of three RNOP.

c) **Regeneration time vs. bandwidth range**: In Figure 4(c), the regeneration time is plotted against different bandwidth range = (1 to 120), (20 to 120), .., (80 to 120). For the case given, for bandwidth range = (1 to 120), the regeneration time for ONP is 26.4% better than average of three RNP and 14.3% better than average of three RNOP.

These simulations were done hundreds of times and it was found that the regeneration time for the optimized node selection for repairing data node and node selections, RNOP and ONP applied for repairing parity node is always better than the randomly selected node.

5. Conclusions

The group repair codes (GRC) was one technique which had optimal repair bandwidth and occupy less storage. In this paper, we optimized the regeneration time for repairing the failures in GRC under heterogeneous network that had different link capacities. By selecting the optimal nodes with maximum link capacities between providers and newcomer nodes, the regeneration time required for the repair is reduced. The node selection algorithms for data node failure and parity node failure are proposed in the paper. These algorithms have been simulated and compared with random node selection. The effect of varying the number of fragments k, data size M and bandwidth range have been discussed in detail and shown graphically. The results show that by the optimal selection of the provider nodes and the newcomer node, the regeneration time can be reduced considerably.

The GRC approach can be applied in distributed storage, cloud and archival storage, and in data communication applications for error correction and repairing data loss due to failures. Since GRC has optimal repair cost, node selection algorithms helps in reducing regeneration time under heterogeneous networks. Node selection mechanism can be explored for other existing network coding approaches to reduce the cost of regeneration. The network with characteristics like tree topology; network with intermediate nodes, routers and switches; network traffic etc. may also affect the system's performance and thus may interest the readers for further research.

Conflict of interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

The authors would like to express their sincere thanks to the referee and there valuable suggestions for the improvement of this paper.

References

Araujo, J., Giroire, F., & Monteiro, J. (2011, September). Hybrid approaches for distributed storage systems. In *International Conference on Data Management in Grid and P2P Systems* (pp. 1–12). Berlin, Heidelberg: Springer.

Benerjee, K. G., & Gupta, M. K. (2021). Trade-off for heterogeneous distributed storage systems between storage and repair cost. *Problems of Information Transmission 57*(1), 33–53.

Dimakis, A. G., Godfrey, P. B., Wu, Y., Wainwright, M. J., & Ramchandran, K. (2010). Network coding for distributed storage systems. *IEEE Trans Inform Theory, 56*(9), 4539–4551.

Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). The Google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (pp. 29–43).

Gong, Q., Wang, J., Wei, D., Wang, J., & Wang, X. (2015, September). Optimal node selection for data regeneration in heterogeneous distributed storage systems. In *2015 44th International Conference on Parallel Processing* (pp. 390–399). IEEE.

Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J., Yekhanin, S. (2012). Erasure coding in windows azure storage. In *2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12)* (pp. 15–26).

Hu, Y., Chen, H. C., Lee, P. P., & Tang, Y. (2012, February). NCCloud: Applying network coding for the storage repair in a cloud-of-clouds. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST 12)* (pp. 21).

Jia, C., Wang, J., Zhu, Y., Wang, X., Lu, K., Wang, X., & Wen, Z. (2015, November). On the optimal provider selection for repair in distributed storage system with network coding. In *International Conference on Algorithms and Architectures for Parallel Processing* (pp. 506–520). Cham: Springer.

Jia, C., Wang, J., Zhu, Y., Wen, Z., & Jia, J. (2016, August). Joint selection of providers and the new Comer for optimal repair in distributed storage network with network coding. In *2016 IEEE Trustcom/BigDataSE/ISPA* (pp. 1567–1574). IEEE.

Kolosov, O., Yadgar, G., Liram, M., Tamo, I., & Barg, A. (2020). On fault tolerance, locality, and optimality in locally repairable codes. *ACM Transactions on Storage (TOS), 16*(2), 1–32.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., ... & Zhao, B. (2000). Oceanstore: An architecture for global-scale persistent storage. *ACM SIGOPS Operating Systems Review, 34*(5), 190–201.

Li, J., Yang, S., Wang, X., & Li, B. (2010, March). Tree-structured data regeneration in distributed storage systems with regenerating codes. In *2010 Proceedings IEEE INFOCOM* (pp. 1–9). IEEE.

Li, J., Yang, S., Wang, X., Xue, X., & Li, B. (2009, July). Tree-structured data regeneration with network coding in distributed storage systems. In *2009 17th International Workshop on Quality of Service* (pp. 1–9). IEEE.

Mittal, S., Rakesh, N., Matam, R., & Adhikari, A. K. (2018). An optimal storage and repair mechanism for Group Repair Code in a distributed storage environment. *Intelligent Decision Technologies, 12*(4), 441–451.

Mohan, L. J., Harold, R. L., Caneleo, P. I. S., Parampalli, U., & Harwood, A. (2015, June). Benchmarking the performance of hadoop triple replication and erasure coding on a nation-wide distributed cloud. In *2015 International Symposium on Network Coding (NetCod)* (pp. 61–65). IEEE.

Qu, S., Zhang, J., & Wang, X. (2018, May). Asymmetric regenerating codes for heterogeneous distributed storage systems. In *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)* (pp. 1–8). IEEE.

Rashmi, K. V., Shah, N. B., & Ramchandran, K. (2017). A piggybacking design framework for read-and download-efficient distributed storage codes. *IEEE Transactions on Information Theory, 63*(9), 5802–5820.

Rashmi, K. V., Nakkiran, P., Wang, J., Shah, N. B., & Ramchandran, K. (2015). Having your cake and eating it too: Jointly optimal erasure codes for i/o, storage, and network-bandwidth. In *13th {USENIX} Conference on File and Storage Technologies ({FAST} 15)* (pp. 81–94).

Rodrigues, R., & Liskov, B. (2005, February). High availability in DHTs: Erasure coding vs. replication. In *International Workshop on Peer-to-Peer Systems* (pp. 226–239). Berlin, Heidelberg: Springer.

Sathiamoorthy, M., Asteris, M., Papailiopoulos, D., Dimakis, A. G., Vadali, R., Chen, S., & Borthakur, D. (2013). Xoring elephants: Novel erasure codes for big data. *Proceedings of the VLDB Endowment 6*(5), 325–336.

Tamo, I., & Barg, A. (2014). A family of optimal locally recoverable codes. *IEEE Transactions on Information Theory*, 60(8), 4661–4676.

Wang, Y., Wei, D., Yin, X., & Wang, X. (2014, April). Heterogeneity-aware data regeneration in distributed storage systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications* (pp. 1878–1886). IEEE.

Weatherspoon, H., & Kubiatowicz, J. D. (2002, March). Erasure coding vs. replication: A quantitative comparison. In *International Workshop on Peer-to-Peer Systems* (pp. 328–337). Berlin, Heidelberg: Springer.

Wu, Y., Liu, D., Chen, X., Ren, J., Liu, R., Tan, Y., & Zhang, Z. (2021). MobileRE: A replicas prioritized hybrid fault tolerance strategy for mobile distributed system. *Journal of Systems Architecture*, *118*, 102217, 1–14.

Xia, M., Saxena, M., Blaum, M., & Pease, D. A. (2015). A tale of two erasure codes in {HDFS}. In *13th {USENIX} Conference on File and Storage Technologies ({FAST} 15)* (pp. 213–226).

Ye, M., Qiu, H., Wang, Y., Zhou, Z., Zheng, F., & Ma, T. (2021). A method of repairing single node failure in the distributed storage system based on the regenerating-code and a hybrid genetic algorithm. *Neurocomputing*, *458*, 566–578.

Yekhanin, S. (2011, June). Locally decodable codes. In *International Computer Science Symposium in Russia* (pp. 289–290). Berlin, Heidelberg: Springer.

Yu, Q., Sung, C. W., & Chan, T. H. (2012, June). Repair topology design for distributed storage systems. In *2012 IEEE International Conference on Communications (ICC)* (pp. 7009–7013). IEEE.

Zhang, H., Li, H., & Li, S. Y. R. (2016). Repair tree: Fast repair for single failure in erasure-coded distributed storage systems. *IEEE Transactions on Parallel and Distributed Systems*, *28*(6), 1728–1739.