Exact Minimum-Repair-Bandwidth Cooperative Regenerating Codes for Distributed Storage Systems

Kenneth W. Shum and Yuchong Hu
Institute of Network Coding,
The Chinese University of Hong Kong.
Email: {wkshum, ychu}@inc.cuhk.edu.hk.

Abstract—In order to provide high data reliability, distributed storage systems disperse data with redundancy to multiple storage nodes. Regenerating codes is a new class of erasure codes to introduce redundancy for the purpose of improving the data repair performance in distributed storage. Most of the studies on regenerating codes focus on the single-failure recovery, but it is not uncommon to see two or more node failures at the same time in large storage networks. To exploit the opportunity of repairing multiple failed nodes simultaneously, a cooperative repair mechanism, in the sense that the nodes to be repaired can exchange data among themselves, is investigated. A lower bound on the repair-bandwidth for cooperative repair is derived and a construction of a family of exact cooperative regenerating codes matching this lower bound is presented.

Index Terms—Distributed Storage, Regenerating Codes, Erasure Codes, Repair-Bandwidth, Network Coding.

I. INTRODUCTION

Distributed storage systems such as Oceanstore [1] and Total Recall [2] provide reliable and scalable solutions to the increasing demand of data storage. They disperse data with redundancy to multiple storage nodes and the data can be retrieved even if some of nodes are not available. When erasure coding is used as a redundancy scheme in distributed storage, the task of repairing a node failure becomes non-trivial. A traditional way to repair a failed node is to download and reconstruct the whole data file first, and then regenerate the lost content (e.g., RAID-5, RAID-6). Since the size of the original data file may be huge, a lot of traffic is consumed for the purpose of repairing just one failed node.

In order to reduce the total traffic required for repairing, called repair-bandwidth, a new class of erasure codes, called regenerating codes [3], is presented and has a significantly lower traffic consumed in regenerating a failed node. The main idea of regenerating codes is to reduce repair-bandwidth from the survival nodes to a new node (called a newcomer), which regenerates the lost content in the failed node. Some constructions of minimum repair-bandwidth regenerating codes are given in [4], [5]. They are based on exact repair or called exact MBR codes, which means the lost content of the failed node are repaired exactly.

Most of the studies on regenerating codes in the literature are for the single-failure recovery or one-by-one repair. When the number of storage nodes becomes large, the multi-failure case is not infrequent, and we need to regenerate several failed nodes at the same time. In addition, in practical systems such as Total Recall, a recovery process is triggered only after the total number of failed nodes has reached a predefined threshold. These facts motivates the regeneration of multiple failed nodes jointly, instead of repairing in a one-by-one manner. A repair process in which the newcomers may exchange packets among themselves, called a cooperative repair or cooperative recovery, is first introduced in [6]. We will call the regenerating codes for multiple failures with cooperative repair cooperative regenerating codes. In [7], a special class of cooperative regenerating codes is proposed, in which the newcomers can select survival nodes for repairing flexibly. In [8], an explicit construction of cooperative regenerating code minimizing the storage in each node is given.

The tradeoff spectrum between repair-bandwidth and storage for cooperative regenerating codes is given in [8], [9]. Regenerating codes which attain one end of this spectrum, corresponding to the minimum storage, are considered in [6], [7]. In this paper, we focus on the other end of this spectrum. Codes which minimizes repair-bandwidth is called Minimal Repair-Bandwidth Cooperative Regenerating (MBCR) codes.

Main Results: After presenting a simple example and demonstrating the basic ideas in Section II, we derive in Section III a lower bound on the repair-bandwidth in cooperative recovery. An explicit construction of a family of exact MBCR codes matching this lower bound is given in Section IV.

II. AN ILLUSTRATIVE EXAMPLE

In this section, we introduce some notations and illustrate the basic idea of cooperative repair.

Based on the system model introduced in [3] and [6], a file consisting of $B$ packets is encoded and distributed to $n$ nodes and a data collector can retrieve the file by downloading data from any $k$ of $n$ nodes. When $r$ nodes fails, $r$ newcomers are selected to repair the failed nodes. The repair process is divided into two phases. In the first phase, each of the $r$ newcomers connects to $d$ surviving nodes and downloads some packets. In the second phase, the newcomers exchange some packets among themselves. The objective is to minimize the total number of the packets transmitted (i.e., repair-bandwidth) in the two phases. Next we give an illustration of cooperative repair with parameters $n = 4$ and $d = k = r = 2$.
We initialize the distributed storage system by dividing a data file into eight data packets $A, B, \ldots, H$, and distribute them to four storage nodes. Each node stores five packets: four systematic and one parity-check (Fig. 1). The addition “+” is bit-wise exclusive-OR (XOR). The first node stores the first four packets $A, B, C, D$, skips the packet $E$, and stores the sum of the next two packets, $F + G$. The content of nodes 2, 3 and 4 can be obtained likewise by shifting the encoding pattern to the right respectively by 2, 4 and 6 packets. It is easy to verify that a data collector can rebuild the file from any two of four nodes in the illustrated code. For example, the data collector which connects to nodes 1 and 2 can reconstruct the eight data packets by downloading $A, B, C$ and $F + G$ from node 1, and $D, E, F$, and $H + A$ from node 2. Then it can solve for $G$ by subtracting $F$ from $F + G$, and $H$ by subtracting $A$ from $H + A$.

As for the repair process, the illustrated code costs ten packets per any two-failure recovery. Suppose that nodes 1 and 3 fail (see the first diagram in Fig. 1). Both newcomers 1 and 3 first download four packets from the survival nodes 2 and 4. Then newcomer 1 (resp. newcomer 3) computes the sum $B + C$ (resp. $F + G$) and sends it to newcomer 3 (resp. newcomer 1). Obviously, a total of ten packets, which are equal to the number of lines (including solid and dashed lines), are transmitted in the network. Similarly, should nodes 2 and 4 fail, the same repair-bandwidth is consumed for regeneration. Suppose that node 1 and 4 fail (see the second diagram in Fig. 1). Both newcomers 1 and 4 first download four packets from the survival nodes 2 and 3. Note that among the four downloaded packets, newcomer 1 (resp. newcomer 4) receives one encode packet $F + G$ (resp. $D + E$) from node 3 (resp. node 2). Then newcomer 1 solves for packet $B$ by subtracting $C$ from $B + C$, and transmits packet $B$ to newcomer 4. Also, newcomer 4 solves for packet $A$ and sends it to newcomer 1. Clearly, a total of ten packet transmissions are sufficient. Similarly, if any pair of two adjacent storage nodes fail, we can also repair them with ten packet transmissions, using the symmetry in the encoding for data distribution.

III. LOWER BOUND ON REPAIR-BANDWIDTH FOR MULTI-LOSS COOPERATIVE REPAIR

In this paper, we assume that the storage nodes are symmetric; for the storage cost, each node stores $\alpha$ packets, and for the repair-bandwidth, each newcomer connects to $d$ existing nodes and downloads $\beta_1$ packets from each of them, and then sends $\beta_2$ packets to each of the $r - 1$ other newcomers. In this paper, we only consider the case that $d \geq k$. The repair-bandwidth per newcomer, denoted by $\gamma$, is defined as the total number of the packets each newcomer receives, and thus is equal to

$$\gamma = d\beta_1 + (r - 1)\beta_2.$$ 

The aim of this section is to derive a lower bound on $\gamma$.

To formulate the problem, we draw an information flow graph as in [6]. Given parameters $n, k, d, r, \alpha, \beta_1$ and $\beta_2$, we construct an information flow graph $G = (V, E)$ as follows. The vertices are grouped into stages.

- In stage $-1$, there is only one vertex $S$, representing the source node which has the original file.
- In stage $0$, there are $n$ vertices Out$_1$, Out$_2$, $\ldots$, Out$_n$, each of them corresponds to an initial storage node. There is a directed edge with capacity $\alpha$ from $S$ to each Out$_i$.
- For $t = 1, 2, 3, \ldots$, suppose $r$ nodes fail in stage $t$. Let the indices of these $r$ storage nodes be $S_t = \{j_1, j_2, \ldots, j_r\}$. For each $i \in S_t$, we put three vertices In$_i$, Mid$_i$, and Out$_i$ in stage $t$. There are $d$ directed edges, with capacity $\beta_1$ from $d$ “out” vertices in previous stages to each In$_i$. For each $i \in S_t$, we put a directed edge from In$_i$ to Mid$_i$ with infinite capacity, and a directed edge from Mid$_i$ to Out$_i$ with capacity $\alpha$. For each pair of distinct indices $i, j \in S_t$, we draw a directed edge from In$_i$ to Mid$_j$ with capacity $\beta_2$. The edges with capacity $\beta_1$ represent the data transferred from existing storage nodes to newcomers, and the edges with capacity $\beta_2$ represent the data exchanged among the newcomers.
- To a data collector, who shows up after $s$ repair processes have taken place, we put a vertex DC in stage $s$ and connect it with $k$ “out” vertices with distinct indices in stage $s$ or earlier. The capacities of these $k$ edges are set to infinity.

An example of information flow graph for $n = 4$ storage nodes and $d = r = k = 2$ is shown in Fig. 2.

To derive a lower bound on $d\beta_1 + (r - 1)\beta_2$ for a file of fixed size $B$ is equivalent to derive a upper bound of $B$ for given capacities $\beta_1$ and $\beta_2$ in the information flow graph. So we can apply a celebrated max-flow theorem in [10], which says the size of the data file $B$ cannot be larger than the max-flow from...
S to any data collector (DC). The max-flow is the maximal value of all feasible flows from S to DC. Here, a flow from S to DC, called an (S, DC)-flow, is a mapping \( F \) from the set of edges to the set of non-negative real numbers, satisfying (i) for every edge \( e \), \( F(e) \) does not exceed the capacity of \( e \), (ii) for any vertex \( v \) except the source vertex S and the terminal vertex DC, the sum of \( F(e) \) over edges \( e \) terminating at \( v \) is equal to the sum of \( F(e) \) over edges \( e \) going out from \( v \),

\[
\sum_{e=(v,u) \in E} F(e) = \sum_{e=(v,u) \in E} F(e).
\]

The value of an (S, DC)-flow \( F \) is defined as

\[
\sum_{e=(u,DC) \in E} F(e).
\]

From the max-flow-min-cut theorem, we can upper bound the value of a flow by the capacity of a cut. For a given data collector DC, an (S, DC)-cut is a partition \((U, \bar{U})\) of the vertices in the information flow graph, such that \( S \in U \) and DC \( \in \bar{U} \), where \( \bar{U} \) stands for the complement of \( U \) in the vertex set \( V \). The capacity of an (S, DC)-cut is defined as the sum of capacities of the edges from vertices in \( U \) to vertices in \( \bar{U} \). Next, we will use the fact that the value of any (S, DC)-flow is less than or equal to the capacity of any (S, DC)-cut, together with the max-flow theorem in [10], to prove the following theorem.

**Theorem 1.** If \( d \geq k \), the repair-bandwidth \( d\beta_1 + (r-1)\beta_2 \) is lower bounded by

\[
B(2d + r - 1) \frac{1}{k(2d + r - k)}
\]

and this lower bound can be met only when

\[
(\beta_1, \beta_2) = \left( \frac{2B}{k(2d + r - k)}, \frac{B}{k(2d + r - k)} \right).
\]

**Proof:** Consider a data collector which downloads data from \( k \) out of \( n \) storage nodes. By re-labeling the storage nodes, we can assume without loss of generality that the corresponding \( k \) “out” vertices be \( \text{Out}_1, \text{Out}_2, \ldots, \text{Out}_k \).

**Fig. 2.** An example of information flow graph.

Suppose that these \( k \) “out” vertices belong to stages 1 to \( s \), and for \( \nu = 1, 2, \ldots, s \), there are \( \ell_\nu \) “out” nodes in \( U \) in stage \( \nu \). By vertex re-labeling, we can assume without loss of generality that \( \text{Out}_1, \text{Out}_2, \ldots, \text{Out}_{\ell_1} \) belong to stage 1, \( \text{Out}_{\ell_1+1}, \text{Out}_{\ell_1+2}, \ldots, \text{Out}_{\ell_1+\ell_2} \) belong to stage 2, and so on. For notational convenience, we let \( \ell_0 = 0 \). Consider the (S, DC)-cut with \( U \) consisting of the vertices

\[
\bigcup_{j=\ell_0+\ell_1+\ldots+\ell_{\nu-1}+1}^{\ell_0+\ell_1+\ldots+\ell_{\nu}+1} \{ \text{In}_j, \text{Mid}_j, \text{Out}_j \}
\]

in stage \( \nu \), for \( \nu = 1, 2, \ldots, s \), and DC. We say that the cut thus defined is of type \((\ell_1, \ell_2, \ldots, \ell_s)\). An example of cut with type \((2, 1, 2)\) is shown in Fig. 3.

**Fig. 3.** A sample cut through the information flow graph.

We claim that the capacity of an (S, DC)-cut of type \((\ell_1, \ell_2, \ldots, \ell_s)\) can be as small as

\[
\sum_{\nu=1}^{s} \left( \ell_{\nu}(d - \sum_{j=1}^{\nu-1} \ell_j)\beta_1 + \ell_{\nu}(r - \ell_j)\beta_2 \right).
\]

In stage 1, there are \( d\ell_1 \) edges, each with capacity \( \beta_1 \), terminating at \( \text{In}_1, \text{In}_2, \ldots, \text{In}_{\ell_1} \).

There are \( \ell_1d \) edges, each with capacity \( \beta_1 \), terminating at \( \text{In}_1, \text{In}_2, \ldots, \text{In}_{\ell_1} \) in stage 1. Also, there are \( \ell_1(r - \ell_1) \) edges, each with capacity \( \beta_2 \), terminating at \( \text{Mid}_1, \text{Mid}_2, \ldots, \text{Mid}_{\ell_1} \). Hence, a total of \( \ell_1d\beta_1 + \ell_1(r - \ell_1)\beta_2 \) are contributed to the summation in (3). This is the summand corresponding to \( \nu = 1 \) in (3).

For the second group of \( \ell_2 \) storage nodes in stage 2, there may be \( \ell_1 \) links from the first group of storage nodes, which are not counted in the capacity of the cut. The sum of capacities of edges terminating at the “in” vertices in \( U \) in stage 2 could be as small as \( \ell_2(d - \ell_1)\beta_1 \). Together with the sum of capacities of the edges terminating at the “mid” vertices, a total of \( \ell_2(d - \ell_1)\beta_1 + \ell_2(r - \ell_1)\beta_2 \) are contributed to (3). This is the second summand in (3). The rest of the summands can be derived similarly. This finishes the proof of the claim.

For a data file of size \( B \), we should be able to construct a flow of value at least \( B \). Hence \( B \) is less than or equal to (3) for all type \((\ell_1, \ell_2, \ldots, \ell_s)\) with \( 0 < \ell_\nu \leq r \) for all \( \nu = 1, 2, \ldots, s \), and \( \ell_1 + \ldots + \ell_s = k \). After some algebraic manipulations, we have the following upper bound on \( B \),

\[
B \leq dk\beta_1 + rk\beta_2 - \beta_1 \sum_{i=1}^{s} \ell_i\ell_j - \beta_2 \sum_{i=1}^{s} \ell_i^2.
\]
We note that if we substitute $\beta_1$ and $\beta_2$ by $2B/(k(2d+r-k))$ and $\beta_2 = B/(k(2d+r-k))$ respectively, then we have equality in (4).

We finish the proof by considering the two cases.

Case 1: $k \leq r$. Consider the cut of type $(1, 1, \ldots, 1, 1)$.

From (4), we obtain
\[ B \leq (dk - k(k-1)/2)\beta_1 + (rk - k)\beta_2 \quad (5) \]

From the cut of type $(k, 0, 0, \ldots, 0)$, we have the following constraint,
\[ B \leq dk\beta_1 + (r-k)k\beta_2 \quad (6) \]

If we multiply (5) by 2d, multiply (6) by $r-1$, and add the two resulting inequalities, we get
\[ (2d + r - 1)B \leq k(2d + r - k)(d\beta_1 + (r-1)\beta_2). \]

This proves the lower bound in (1) in Case 1.

To see that the lower bound can be met only when $\beta_1$ and $\beta_2$ are specified as in the theorem, we notice that (5) and (6) define an unbounded polyhedral region in the $\beta_1$-$\beta_2$ plane, with (2) as a vertex. If we want to minimize the objective function $d\beta_1 + (r-1)\beta_2$ over all $(\beta_1, \beta_2)$ in this region, the optimal point is precisely the point given in (2).

Case 2: $k > r$. Consider a cut of type $(r, r, \ldots, r, b)$, where $a = \lfloor k/r \rfloor$ and $b = k - ra$. The upper bound of $B$ in (4) becomes
\[ B \leq \left[ \frac{dk - r^2 a(a-1)}{2} - a br \right] \beta_1 + (rk - ar^2 - b^2)\beta_2 \quad (7) \]

Together with the constraint obtained from a cut of type $(1, 1, \ldots, 1)$, we set up a linear program and minimize $d\beta_1 + (r-1)\beta_2$ over all $(\beta_1, \beta_2)$ satisfying the inequalities in (5) and (7).

Let $L_1$ be the straight line in the $\beta_1$-$\beta_2$ plane by setting the inequality in (5) to equality. Let $L_2$ be the straight line consisting of point $(\beta_1, \beta_2)$ satisfying (7) with equality. We can verify that the intersection of $L_1$ and $L_2$ is the point in (2).

We investigate the slopes of $L_1$ and $L_2$. The slope of $L_1$ is equal to $-(d - (k-1)/2)/(r-1)$, which is larger than the slope of the objective function, $-d/(r-1)$. For the slope of $L_2$, we first check that
\[ dk - r^2 a(a-1)/2 - a br > db \]
\[ \iff dk - r^2 a(a-1)/2 - ab r > db \]
\[ \iff d > (r(a-1) + b) \iff d > k - r. \]

We have used several times that $k = ar + b$ in the above derivation. The last line holds by the assumptions $d \geq k$ and $r \geq 2$. Therefore,
\[ -\frac{dk - r^2 a(a-1)/2 - a br}{rk - ar^2 - b^2} < \frac{db}{rk - ar^2 - b^2} = -\frac{d}{r-1}. \]

The slope of $L_2$ is strictly less than the slope of the objective function. Thus, the optimal point of the linear programming problem is the vertex in (2). This completes the proof of Case 2. 

Note: Theorem 1 is obtained independently in [9].

We can now show that the regenerating code discussed in Section II is optimal, in the sense that given the parameters $B, d, k$ and $r$, the repair-bandwidth matches the lower bound in Theorem 1. We have $B = 8$ and $d = k = r = 2$ in the example. From Theorem 1, the repair-bandwidth cannot be less than $8 \cdot \frac{2.25}{2(2d+r-k)} = 5$. We have shown in Section II that the repair process requires exactly 5 packet transmissions per failed node, and therefore matches the optimal value.

For non-cooperative or one-by-one repair, it is proved in [3] that the minimum repair-bandwidth per failed node is $2dB/(k(2d+1-k))$, which turns out to be the same as the left hand side of (1) with $r$ set to 1. If we apply a non-cooperative regenerating code to a distributed storage system with parameters as in Section II, the minimum repair-bandwidth is $8 \cdot \frac{2.25}{2(2d+r-k)} = 5.333$. From this simple example, we can see that repair-bandwidth can be further reduced if some data exchange of data among the newcomers is allowed.

The lower bound of repair-bandwidth in Theorem 1 in fact holds via random linear coding with field size large enough. The tightness of the lower bound is established in [11], by showing the existence of MBCR codes which match the lower bound. Thus, the minimum repair-bandwidth for MBCR is indeed equal to $B(2d + r - 1)/(k(2d + r - k))$.

IV. AN EXPLICIT CONSTRUCTION OF A FAMILY OF OPTIMAL MBCR CODES

We construct in this section a family of exact MBCR code with parameters $d = k$ and $n = d + r$. In fact, the illustrated code in Section II is a special case in this family.

The whole file is first divided into stripes. Each stripe consists of $B = k(2d + r - k) = kn$ data packets, considered as elements in $GF(q)$, where $q$ is a prime power. In each stripe let the $kn$ data packets be $x_0, x_1, \ldots, x_{kn-1}$. We divide them into $n$ groups. The first group consists of $x_0, x_1, \ldots, x_{k-1}$, the second group consists of $x_k, x_{k+1}, \ldots, x_{2k-1}$, and so on. For notational convenience, we let $x_j = [x_{(j-1)k} x_{(j-1)k+1} \ldots x_{(j-1)k+k-1}]$ be the vector of the data packets in the $j$th group ($1 \leq j \leq n$).

For $i = 1, 2, \ldots, n$, we construct the content of node $i$ as follows. We first put the $k$ data packets in the $i$-th group $x_i$ into node $i$ and then $n-1$ parity-check packets
\[ X_{i\oplus 1} \cdot v_1, X_{i\oplus 2} \cdot v_2, \ldots, X_{i\oplus (n-1)} \cdot v_{n-1} \]

into node $i$, where “$\cdot$” is the dot product of vectors and $\oplus$ is modulo-$n$ addition defined by
\[ x \oplus y := \begin{cases} x + y & \text{if } x + y \leq n, \\ x + y - n & \text{if } x + y > n. \end{cases} \]

Here $v_j (j = 1, 2, \ldots, n-1)$ are column vectors in a $k \times (n-1)$ generating matrix, $G = [v_1 \ v_2 \ldots \ v_{n-1}]$, of a maximal-distance separable (MDS) code over $GF(q)$ of length $n-1$ and dimension $k$. By the defining property of MDS code, any $k$ columns of $G$ are linearly independent of $GF(q)$.
As for the file reconstruction processing, suppose without loss of generality that a data collector connects to nodes 1, 2, ..., k. The systematic packets $x_0, x_1, \ldots, x_{k-1}$ in the first $k$ groups can be downloaded directly, because they are stored in node 1 to node $k$ uncoded. The $i$th group of data packets (with $i > k$) (the components in vector $x_i$) can be reconstructed from $x_j \cdot v_{j-1}, x_j \cdot v_{j-2}, \ldots, x_j \cdot v_{j-k}$, by the MDS property. A data collector connecting to any other $k$ storage nodes can decode similarly.

As for the cooperative repair processing, suppose without loss of generality that nodes $k + 1$ to $n$ fail at the same time. The repair process proceeds as follows.

**Step 1:** For $i = 1, 2, \ldots, k$, node $i$ computes $x_i \cdot v_{n+i-j}$ and sends it to newcomer $j$, for $j = k+1, k+2, \ldots, n$.

**Step 2:** For $j = k+1, k+2, \ldots, n$, newcomer $j$ downloads $k$ packets $x_j \cdot v_{j-1}, x_j \cdot v_{j-2}, \ldots, x_j \cdot v_{j-k}$ from nodes 1 to $k$.

**Step 3:** For $j = k+1, k+2, \ldots, n$, newcomer $j$ can solve for the systematic packets in $x_j$. Then node $j$ sends $x_j \cdot v_{j+i}$ to node $n+i-j$, for $i = 1, 2, \ldots, n-j$, and sends $x_j \cdot v_i$ to node $j-i$, for $i = 1, 2, \ldots, j-k-1$.

In steps 1 and 2, a total of $2k(n-k)$ packets are transmitted. In step 3, each newcomer transmits $r-1$ packets. The total number of packets required in the whole repair process is $2kr + r(r-1) = r(2d + r - 1)$. The number of packets per failed node is therefore $2d + r - 1$. According to Theorem 1, the repair-bandwidth is no less than

$$B \geq \frac{2d + r - 1}{k(2d + r - k)} = 2d + r - 1.$$

Thus, this regenerating code is optimal.

**Remark:** If $n = q + 2$ for some prime power $q$, we can use an extended Reed-Solomon (RS) code of length $q + 1$ in the construction. The alphabet size could be as small as $n - 2$. We refer the reader to [12] for the construction of extended RS code.

**Example:** An example for $n = 5$, $d = k = 3$ and $r = 2$ is shown in Fig. 4. A stripe of file data is divided into 15 packets $x_0, x_1, \ldots, x_{14}$. Let $q = 2$ and $G$ be the generating matrix

$$G = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

of a triply-extended Reed-Solomon code over $GF(2)$ [12]. The $i$th row of the array in Fig. 4 indicates the content of node $i$. For example, node 4 stores six systematic packets, $x_9, x_{10}, x_{11}$ in $x_4$, $x_1 \cdot v_2 = x_0, x_2 \cdot v_3 = x_4, x_3 \cdot v_4 = x_8$, and one parity-check packet $x_5 \cdot v_1 = x_{12} + x_{13} + x_{14}$.

Suppose that nodes 4 and 5 fail. In the first step, node 1 sends $x_1 \cdot v_2$ and $x_1 \cdot v_1$ to newcomers 4 and 5 respectively. Similarly, node 2 sends $x_2 \cdot v_3$ and $x_2 \cdot v_2$, and node 3 sends $x_3 \cdot v_4$ and $x_3 \cdot v_3$. In the second step, node 1 transmits $x_4 \cdot v_3$ and $x_5 \cdot v_4$ to newcomers 4 and 5 respectively. Likewise, node 2 transmits $x_4 \cdot v_2$ and $x_5 \cdot v_5$, and node 3 transmits $x_4 \cdot v_1$ and $x_5 \cdot v_2$. In the third step, newcomer 4 reconstructs $x_4$, and sends $x_4 \cdot v_4$ to newcomer 5. Also, newcomer 5 reconstructs $x_5$, and sends $x_5 \cdot v_1$ to newcomer 4. Lastly, the lost packets in nodes 4 and 5 are regenerated in newcomer 4 and 5. The total number of packet transmissions in the whole repair process is equal to 14. The repair-bandwidth per failed node is $7$. It matches the theoretic lower bound $15(2 \cdot 3 + 2 - 1)/(3(2 \cdot 3 + 2 - 3)) = 7$.

**V. Conclusion**

We give a construction of a family of exact and optimal MBCR codes for $d = k$ and $n = d + r$. The constructed regenerating code has the advantage of being a systematic code. For example, if we want to look at the content of one particular packet, we only need to contact the node which has a copy of this packet and download the packet directly. Another advantage of this construction is that the requirement of finite field size grows linearly as a function of the number of storage nodes.

**References**


