Abstract—In wireless distributed storage systems, storage nodes are connected by wireless channels, which are broadcast in nature. This paper exploits this unique feature to design an efficient repair mechanism, called broadcast repair, for wireless distributed storage systems in the presence of multiple-node failures. Due to the broadcast nature of wireless transmission, we advocate a new measure on repair performance called repair-transmission bandwidth. In contrast to repair bandwidth, which measures the average number of packets downloaded by a newcomer to replace a failed node, repair-transmission bandwidth measures the average number of packets transmitted by helper nodes per failed node. The storage system we considered can undergo an unlimited number of repair rounds. We obtain an upper bound on the maximum file size that can be supported by a cut analysis of a finite graph. The achievability is shown by codes constructed over a refined information flow graph, which is unbounded. In addition, the optimal storage-bandwidth tradeoff is obtained. The performance of broadcast repair is compared both analytically and numerically with that of cooperative repair, the basic repair method for wired distributed storage systems with multiple-node failures. While cooperative repair is based on the idea of allowing newcomers to exchange packets, broadcast repair is based on the idea of allowing a helper to broadcast packets to all newcomers simultaneously. We show that broadcast repair outperforms cooperative repair, offering a better tradeoff between storage efficiency and repair-transmission bandwidth.

Index Terms—Distributed storage systems, wireless cache networks, broadcast repair, centralized repair, min-cut value, repair-transmission bandwidth.

I. INTRODUCTION

In distributed storage systems (DSS), it is important that failed nodes can be repaired in an efficient manner. Traditional erasure codes, have high storage efficiency, but typically require a large amount of data exchange (or repair bandwidth) during node repair. The replacement node, commonly called the newcomer, needs to download the whole file from some or all of the surviving nodes, called helper nodes, to reconstruct the lost data in the failed node. In this case, the repair bandwidth is the whole file size. Dimakis et al. showed that repair bandwidth can be reduced at the expense of storage space [2]. Using a cut-based analysis on the information flow graph, which models the evolution of information flow in the storage system subject to arbitrary sequences of node failures/repairs, the maximum file size that can be stored, can be obtained by network coding [3]. Given a fixed file size, the fundamental tradeoff between storage efficiency and repair bandwidth has been derived. The tradeoff is optimal if the file size given is the maximum one that the system can support. Codes achieving optimal tradeoff are named regenerating codes. Specifically, codes attaining the best storage efficiency are called minimum-storage regenerating (MSR) codes whereas codes attaining the minimum repair bandwidth are called minimum-bandwidth regenerating (MBR) codes. Since the number of repair rounds should be unlimited, it is a challenge how to apply network coding over an unbounded graph with an unbounded number of receivers. In [4] and [5] optimal codes are constructed over a finite field whose size depends only on the maximum number of nodes at any instant, but independent of the number of failures/repairs.

The seminal work of [2] has stimulated a lot of study on efficient repair of failed nodes in DSS. Some works utilized the information flow graph to investigate the bandwidth-storage tradeoff under more complicated repair scenarios [6], [7]. Some works focused on code constructions. Basically, there are two kinds of code construction models, based on either functional repair or exact repair. In functional repair [2], [5]–[8], the newcomer is not required to store exactly the same symbols as the original failed node as long as the data collector is able to retrieve the file. In exact repair [9]–[13], the newcomer must reconstruct the same data stored on the failed node. Most of these works, however, focus on single-node repair, that nodes are assumed to fail one at a time and the repair process starts immediately when the node fails. In [14], it was observed that repair bandwidth per failed node can be reduced if the repair process starts only when the number of failed nodes reaches a predetermined threshold. In this case, newcomers first download some data from the surviving nodes, and then exchange some data among themselves. Such a repair process is termed cooperative repair. Coding for cooperative repair are proposed in [15] and [16].

Ping Hu, Chi Wan Sung, Senior Member, IEEE, and Terence H. Chan
In [17], results on cooperative repair are further extended to a more general scenario, and the fundamental tradeoff in a DSS with cooperative repair is derived. Exact cooperative regenerating codes at the minimum bandwidth cooperative regeneration point for all possible parameters are constructed explicitly in [18].

Nowadays due to the increasing use of wireless devices and the popularity of wireless sensor networks, the design and analysis of wireless distributed storage systems (WDSS) has become an emerging new area [19]–[27]. Recently, in the context of mobile cellular systems, it has been proposed to store or cache popular files in the wireless edge by deploying a number of small-cell base stations with large storage space [25], [28]–[30]. Since the storage space of modern smart phones or tablets is ever increasing, it is also possible to exploit this capability and store files directly in these devices. When a user requests a file, he or she can retrieve it by downloading data from neighboring devices through device-to-device (D2D) communications [26], [27], [31]–[33]. In such a scenario, it is necessary to repair lost data when a device becomes unavailable such as running out of battery or moving out of the area.

The repair problem in WDSS has been investigated in [32]–[40]. Among them, the works closest to ours are [37]–[39]. In [37] and [38], the fundamental storage-bandwidth tradeoff for single-node repair in DSS with erasure channels are established. In [39], it considered the repair problem when parts of stored packets in multiple nodes are lost. The channel for repair is assumed broadcast without interference. The work focused on one repair round and obtained the minimum transmitted packets for repair. For a special parameter setting, an exact repair code construction is proposed. Repair under multiple repair rounds is unclear for general parameter settings.

While designs for DSS can also be applied to WDSS, it is important to understand the fundamental difference between DSS and WDSS. The key difference between wireless and wired transmission, is the broadcast nature, which could be exploited during the repair process for multiple failed nodes. To illustrate, consider the example shown in Figure 1.

There are 4 storage nodes. The file, which is divided into $A_1, A_2, B_1,$ and $B_2$, can be retrieved from any two storage nodes. Suppose two of the nodes fail as indicated. To replace them, two newcomers join the system, and the two surviving storage nodes broadcast packets to them. As shown in the figure, by broadcasting 4 packets in total, the content originally stored in the failed nodes can be regenerated by the newcomers. The average number of packet transmissions needed per newcomer is 2. Suppose we repair the failed nodes one by one using unicast transmission. The packet would be counted twice, which is 8, in calculating repair bandwidth. Each newcomer requires 4 packets for repair. Furthermore, as shown in Fig. 2, if we repair the failed nodes by cooperative repair [17, Sec. I], the transmitted packets we need is 6, meaning that the average number of packet transmissions needed per newcomer is 3. From this example, we see that the number of packets transmitted over the air can be reduced by broadcast.

To reap the potential gain, we propose the concept of broadcast repair for WDSS with multiple node failures. In [1], we investigated the case where the number of repair rounds is finite and obtained the maximum file size supported by the system in that case. It remains unknown that when the repair continues to undergo, how the system performs. In this work, we remove the restriction that the number of repair rounds is finite. The storage capacity we defined in this work should hold for the system who can undergo infinite number of repairs. The maximum file size supported by a system with finite repairs, which is obtained by a cut analysis of the graph representation, serves as an upper bound of the storage capacity (under infinite repairs). We prove that this upper bound is achievable by a code with finite alphabet (even if there are infinite repair rounds) for functional repair. To quantify the benefit of broadcast repair, we compare our method with cooperative repair with unicast transmissions. An explicit form on the storage capacity is derived, and its superiority over cooperative repair for WDSS is shown analytically.

Finally, it should be noted that our broadcast repair model in our conference version [1] is equivalent to the
centralized repair model investigated in [41]–[45]. The work in [41] and [42] focuses on the minimum-storage point and the minimum-bandwidth point. It is shown that the minimum-storage point under centralized repair can be achieved by the same codes that achieve the minimum-storage point under cooperative repair. Besides, explicit code constructions at this point with exact repair are considered in [41], [42], [45], and [46]. At the minimum-bandwidth point, exact-repair codes for systems that satisfy a certain technical property are constructed in [41] and [42]. In [45], a general framework was proposed to convert single erasure minimum storage regenerating codes to ones that can achieve the minimum-storage point, hence proving that functional minimum-bandwidth point is not achievable for linear exact repair codes. Nevertheless, we are the first to propose the minimum-storage point, hence proving that functional minimum storage regenerating codes to ones that can achieve the minimum-storage point under cooperative repair. Besides, explicit code constructions at this point with exact repair are considered in [41]–[45]. The work in this repair round is denoted by \( R_d \).

In Section V, we show that the upper bound obtained in Section IV is achievable under infinite repair rounds. The achievability is shown by the existence of generic storage code over the system. To show that, we introduce a refined information flow graph, and construct generic storage code based on it. We derive the explicit form of broadcast repair storage capacity and compare its performance with cooperative repair both analytically and numerically in Section VI. Section VII concludes this paper.

II. SYSTEM MODEL AND BROADCAST REPAIR

The system model is designed to capture the broadcast characteristic under the wireless scenario. It includes one source node, multiple storage nodes, and multiple data collectors. Each storage node can store \( \alpha \) packets at most. Storage nodes are not point-to-point connected, but fully connected by a wireless broadcast medium.

At the initial stage, the source node distributes a file into \( n \) storage nodes such that the data collectors can retrieve the file from any \( k \) nodes. We index these storage nodes by the set \( \mathcal{N} = \{1, 2, \ldots, n\} \). After the initialization, the source node becomes inactive and leaves the system. In other words, the initialization is simply writing data onto the storage nodes.

These \( n \) storage nodes are not reliable and can fail at times (becoming inactive). When the number of failed nodes accumulates up to a threshold \( r \), the repair process is triggered. We call this process one round of repair. During each repair round, \( r \) new nodes, or newcomers join the system to replace the failed nodes. To regenerate the lost data, any \( d \geq k \) active nodes can be chosen as helper nodes to broadcast packets to the newcomers. The number of packets broadcast by each helper is denoted by \( \beta \). To ensure enough helper nodes, we require

\[
n - r \geq d.
\]  

When a packet is broadcast during repair, we assume that it can be received successfully by all newcomers without error. We also assume that the helpers use orthogonal channels for transmissions so that there is no interference between them. We index the newcomers after the \( s \)-th repair round by \( R_s \triangleq \{ n + (s - 1)r + 1, \ldots, n + sr \} \). The set of helpers in this repair round is denoted by \( \mathcal{H}_s \).
Any data collector can join the system after the initialization stage or after any repair round. It can connect to any \( k \) active nodes via \( k \) orthogonal channels. We assume that the data collector retrieves all data stored in the nodes it connects to. In practice, it is possible that a data collector joins the system when some nodes are failed but the next repair round, say \( s + 1 \), has not been triggered or finished. Because of (1), there are always \( k \) or more active nodes in the system for the data collector to retrieve the file. Such a data collector can be regarded as joining the system right after the \( s \)-th round. Therefore, without loss of generality, we assume that all data collectors join the system right after the completion of a repair round.

For ease of presentation, we call the initialization stage the 0-th repair round. We denote the data collector which joins after the \( s \)-th repair round and connects to a set \( K \) of \( k \) active nodes by \( DC_{s,K} \). Since we have to ensure that the file can always be retrieved, we consider all possible arrivals (in terms of \( s \)) and connections (in terms of \( K \)) of a data collector.

We assume that the WDSS is used indefinitely, so that there is no restriction on the number of repair rounds. The above system is called a WDSS with parameters \( (n, k, d, r, \alpha, \beta) \). An instance of a WDSS is determined by the failure patterns, newcomers \( R_1, R_2, \ldots \), and the collection of helper sets \( H_1, H_2, \ldots \). The repair process described above is called broadcast repair. We denote the maximum file size that can be supported by \( C_{\text{storage}} \), which we named storage capacity. We also denote the maximum file size that can be supported by repair round \( T \) by \( C_{\text{storage}}^T \). It is clear that \( C_{\text{storage}} \leq C_{\text{storage}}^T \) for any \( T \).

In the literature of DSS, the total number of packets downloaded by a newcomer so as to repair a failed node is called repair bandwidth [2]. It is one of the key performance metrics in DSS, reflecting the amount of network traffic required in the repair process. The same concept can also be applied to multiple node failures with cooperative repair processes [14]–[17]. In a wireless environment, due to the broadcast nature, a more accurate and relevant measure should be the number of packets being transmitted by the helper node (instead of the number of packets received by the newcomer), especially when there are multiple node failures. To better reflect the use of frequency spectrum in a wireless environment, we introduce a new performance metric named repair-transmission bandwidth:

**Definition 1:** The repair-transmission bandwidth, \( \tau \), is defined as the number of packets the helper nodes transmitted per newcomer.

If all the packet transmissions are in unicast mode, then repair-transmission bandwidth is equal to repair bandwidth, since the total number of packets transmitted by the helpers is equal to the total number of packets received by the newcomers. They are different, however, when packets transmissions are in broadcast mode. For the WDSS model described above, we have

\[
\tau = \frac{d \beta}{r}. \tag{2}
\]

When \( r \geq k \), the \( d \) helpers should transmit the whole file to the \( r \) newcomers because if a data collector connects to \( k \) of these \( r \) newcomers, it should be able to retrieve the file. For this case, we can directly obtain

\[
\tau = \frac{C_{\text{storage}}}{r},
\]

no matter how large the storage space \( \alpha \) of a node is. Each newcomer can then reconstruct the original file and re-encode it in the same way as the source had done in the initial stage. This corresponds to exact repair, and obviously the system can sustain for infinite repair rounds.

When \( r < k \), it may not be necessary for the helpers to transmit the whole file to the \( r \) newcomers. In this paper, we consider only this non-trivial case. We will see that given a requirement on storage capacity \( C_{\text{storage}} \), there is a tradeoff between the per-node storage capacity, \( \alpha \), and the repair-transmission bandwidth, \( \tau \). For easy reference, we summarize our notation in Table I.

### III. Graph Representation

The repair dynamics of a WDSS can be represented by a directed acyclic graph (DAG) \( G = (V, E) \), where \( V \) is the vertex set and \( E \) is the edge set. When the system evolves as the nodes fail and are repaired, the corresponding graph becomes infinite. Each edge \( e(i,j) \in E \), which connects node \( i \) to node \( j \), is associated with a parameter \( u_{i,j} \), which denotes the capacity of the edge. The graph includes one source vertex \( S \), multiple storage nodes, and multiple data collectors \( DC_{s,K} \). Each storage node \( j \) is represented by two vertices, \( \text{in-vertex} \ \text{In}_j, \ \text{out-vertex} \ \text{Out}_j \), and a directed edge \( \text{In}_j \to \text{Out}_j \) with parameter \( \alpha \). In this paper, the terms “node” and “vertex” have different meanings. A node refers to a storage device in the WDSS while a vertex is an abstract entity in the graph.

In the initialization stage where data is first stored at the storage nodes, the source vertex \( S \) transmits packets to the storage nodes and then becomes inactive. This is modeled by adding the edges \( S \to \text{In}_j \), for all \( j \in N \), with capacity \( \infty \). Note that this does not mean that the actual transmission rate of the communication link between the source and each storage node is infinite; it only means that all the information in the source link are available in the in-vertices of each storage node, and each storage node can store only \( \alpha \) symbols as indicated by the edge capacity of \( \alpha \) between the in-vertex and the out-vertex of a storage node.

In the first repair round (i.e., \( s = 1 \)), node \( i \in H_1 \) broadcasts \( \beta \) packets to newcomer \( j \in R_1 \), which is again modeled by two vertices \( \text{In}_j, \text{Out}_j \), and a directed edge \( \text{In}_j \to \text{Out}_j \) with parameter \( \alpha \). Note that \( R_1 \) and \( N \) are disjoint, meaning that a newcomer has a new index, which is different from the index of the failed node being replaced by that newcomer. For each helper node \( i \in H_1 \), we add an auxiliary vertex, say \( h_1^i \). This auxiliary vertex is used to model the broadcast feature of the wireless channel. Auxiliary vertex \( h_1^i \) goes to \( \text{Out}_j \) by an edge with capacity \( \beta \). Edges with capacity \( \infty \) are added from vertex \( h_1^i \) to in-vertex \( \text{In}_j \) of every newcomer \( j \in R_1 \). Subsequent repair rounds are modeled in the same way.

Consider the example shown in Fig. 3, where we draw the first 3 repair rounds of a WDSS with parameters
A WDSS instance $I$ is specified by the failed nodes and the helpers in each repair round. Given any instance $I$, we can construct a graph as described above, which corresponds to a multicast network problem with the single source $S$ and multiple destinations $\text{DC}_s, K$, where $s = 0, 1, \ldots$, and $K$ can be any legitimate choice of storage nodes after repair round $s$. 

To model the file retrieval process, after each repair round $s$ and for each possible choice of $K$, we add a data collector $\text{DC}_{s,K}$. Furthermore, a directed edge from each out-vertex of a node in $K$ to $\text{DC}_{s,K}$ with capacity $\infty$ is added. In Fig. 3, we show only one data collector, namely, $\text{DC}_{3,\{9,13,14\}}$, for simplicity.

$n = 8, k = 3, d = 4, r = 2$ (We have neglected the source vertex due to space limit). In this example, nodes 5 and 6 failed in the first repair round, and we have $R_1 = \{9, 10\}$ and $H_1 = \{1, 2, 3, 4\}$. Nodes 8 and 10 failed in the second repair round, and we have $R_2 = \{11, 12\}$ and $H_2 = \{9, 3, 4, 7\}$. Then, nodes 1 and 2 failed, and we have $R_3 = \{13, 14\}$ and $H_3 = \{9, 11, 12, 7\}$. 

Fig. 3. An example for cut-value in broadcast repair.

| Symbol | Definition |
|--------|------------|
| $n$    | number of storage nodes |
| $k$    | minimum number of storage nodes required for file reconstruction |
| $s$    | repair round index |
| $T$    | total number of repair rounds |
| $d$    | number of helpers in each repair round |
| $r$    | number of newcomers in each repair round |
| $\alpha$ | per-node storage capacity |
| $\beta$ | number of transmitted packets of each helper |
| $\tau$ | number of packets the helpers transmitted per newcomer |
| $R_s$  | the set of newcomers in repair round $s$ |
| $H_s$  | the set of helpers in repair round $s$ |
| $C_{\text{storage}}^T$ | the maximum file size that can be supported under $T$ repair rounds |
| $C_{\text{storage}}$ | storage capacity (the maximum file size that can be supported under unlimited number of repair rounds) |
Given a specific instance $I$, an $x$-$y$ cut $X$ is a subset of $V$ such that $x \in X$, $y \in \overline{X} \triangleq V \setminus X$ and there is at least one edge from $X$ to $\overline{X}$. The cut-set of a cut $X$ is $\{(u, v) \in E : u \in X, v \in \overline{X}\}$. The cut-value of $X$ within that instance $I$ is defined as:

$$C_I(X) \triangleq \sum_{i \in X, j \in \overline{X}} u_{ij}.$$  

(3)

Two examples of $S$-$DC_{3,9,13,14}$ cuts are denoted in Fig. 3 by dashed lines. For line 1, the cut-value is $8\beta$, which means that the information that can pass through this cut is at most $8\beta$. Note that this is just an upper bound, as the information actually transmitted over these eight edges can be correlated. Similarly, for line 2, the cut-value is $\alpha + 3\beta$.

IV. MIN-CUT UPPER BOUND OF THE STORAGE CAPACITY

Since the storage capacity $C_{storage}$ is defined for the WDSS which is required to tolerate an infinite number of repair rounds, it is clear that $C_{storage} \leq C_{T}$ for every fixed, finite value $T$. In this section, we investigate the WDSS limited to a fixed, finite $T$ repair rounds to provide an upper bound of the storage capacity $C_{storage}$ (who can withstand infinite repair rounds).

When there is limited number of repair rounds $T$, the graph representation is finite. According to the max-flow cut theorem [47] and network coding theory [3], [48], the min-cut value $U^T$ of the family of graphs with $T$ repair rounds, which is given by

$$U^T \triangleq \min_{I \in \text{DC}, X \in \text{S-DC cut}} \min_{T \in \text{S-DC cut}} C_I(X),$$  

(4)

can be achieved by random linear network coding, i.e., $C_{storage}^T = U^T$. Thus the values $U^T$ for every finite value $T$ serve as upper bounds of the storage capacity $C_{storage}$.

By definition, it is clear that $U^T$ is monotonic decreasing with $T$. Since zero is a lower bound, the sequence $U^1, U^2, \ldots$ converges to a limit, which we denote by $U$. In other words, we have $U^1 \geq U^2 \geq \ldots \geq U$. We call the value $U$, which is the limit point of $U^1, U^2, \ldots$, as the min-cut upper bound of the storage capacity $C_{storage}$.

Since we are interested in the value of $U$, we can neglect the case when $T$ is small. For large $T$, we will show that $U^T = U$ for $T \geq k$. This is difficult to obtain $U$ directly from (4). We obtain $U$ by two steps in this section. First we provide a lower bound $B$ of $U^T$ for $T \geq k$ in Theorem 1. Then in Theorem 2, we show the tightness of $B$, i.e., $U^T = B$ for $T \geq k$.

The following theorem is to obtain the lower bound $B$ of $U$ for $T \geq k$. We carefully examine all the cuts in the graphs with a fixed, finite $T \geq k$ repair rounds to make sure that every cut-value is no less than $B$. When $T \geq k$, given a DC, we can always find $k$ repair rounds, together with the initial round, contain all the storage nodes the DC connects to. Re-index these $k$ repair rounds as $1, 2, \ldots, k$. Let $x_0$ be the number of storage nodes the DC connects to in the initial round, and $x_s, s = 1, 2, \ldots, k$ be the number of nodes connected by the DC in re-indexed rounds $1, 2, \ldots, k$. For any cut, we have the following result:

**Theorem 1:** Consider a graph with $T$ repair rounds, where $T \geq k$. For any $S = DC$ cut $K$ in the graph, the cut-value $C(K)$ is bounded below by

$$B \triangleq \min_{x, t_1} \left\{ x_0 \alpha + \sum_{s \in T_1} x_s \alpha + \sum_{s \in T_2} (d - \sum_{i=0}^{s-1} x_i) \beta \right\},$$  

(5)

where the minimization is taken over $T_1 \subseteq \{1, 2, \ldots, k\}$ and

$$0 \leq x_0 \leq n,$$  

(6)

$$0 \leq x_s \leq r,\text{ for } s \in \{1, 2, \ldots, k\},$$  

(7)

$$x_0 + \sum_{1 \leq s \leq k} x_s = k.$$  

(8)

**Proof:** Consider an arbitrary instance of the WDSS. Regard the initialization stage as round 0 and let $V_0 \triangleq V$. For rounds $s = 1, 2, \ldots, T$, let $A_s \triangleq \{h_s^i : i \in H_s\}$ be the set of auxiliary vertices in round $s$, and $V_s \triangleq A_s \cup \{\text{In}_s, \text{Out}_j : j \in R_s\}$ be the set of all vertices in round $s$. Then $V_0 \cup V_1 \cup \cdots \cup V_T$ contains all the vertices except the source and the destinations in the graph. For $s = 0, 1, \ldots, T$, let $x_s$ be the number of out-vertices in $K$.

To obtain the cut-value of an arbitrary $S = DC$ cut $K$, we examine the in-edges of all the vertices in $V_0 \cup V_1 \cup \cdots \cup V_T$, and express the cut-value as a sum of $T + 1$ terms:

$$C(K) = \sum_{0 \leq s \leq T} C_{\Delta,s}(K),$$  

(9)

where

$$C_{\Delta,s}(K) \triangleq \sum_{i \in V_s \cap K, j \in V_s \cap K} u_{ij}$$

is called the cut-value contribution of the vertices in $V_s$. When there is no ambiguity, we may simply write it as $C_{\Delta,s}$. For example, in Fig. 3, the cut denoted by line 2 has cut-value equal to $C_{\Delta,0} + C_{\Delta,1} + C_{\Delta,2} + C_{\Delta,3} = 0 + \alpha + 0 + 3\beta$.

For any $S = DC$ cut $K$, it is obvious that $DC \in K$. By definition, the DC has $k$ out-vertices as its parents, and these $k$ edges all have infinite capacity. If the cut-value is finite, then these $k$ out-vertices must be in $K$. We can always find $k$ repair rounds, together with the initial stage, which contain all these $k$ out-vertices. For ease of presentation, we re-index these repair rounds as $1, 2, \ldots, k$. We consider only these $k$ (re-indexed) repair rounds to obtain a lower bound of $C(K)$:

$$C(K) \geq C_{\Delta,0} + \sum_{1 \leq s \leq k} C_{\Delta,s}(K).$$  

(10)

From now on, the remaining $T - k$ repair rounds that are not re-indexed will not occur in our discussion. We will consider only the re-indexed repair rounds, with index set $\{1, 2, \ldots, k\}$.

In the initial stage, since there is no auxiliary vertex in repair round 0 and all in-vertices should be in $K$ if $C(K) \neq \infty$, we have $C_{\Delta,0} = x_0 \alpha$.

For the repair rounds re-indexed by $\{1, 2, \ldots, k\}$, we have two cases. First, consider the case where $s \in T_1$, where $T_1 \triangleq \{s \in \{1, 2, \ldots, k\} : A_s \cap K \neq \emptyset\}$. In other words, $s \in T_1$ if there exists at least one $h_s^i$ in $K$. We investigate the three
Now consider a special cut $K^*$, which is constructed from $K$ as follows. Initially, let $K^*$ be the same as $K$. For $s \in T_1$, move all $h_i^s$'s into $K^*$, and then $z_s$ becomes zero. Note that, since $h_i^s$'s child vertices are all in round $s$, moving all $h_i^s$'s into $K^*$ will not affect the cut-value contribution of other repair rounds. For $s \in T_2$, move all $l_n$'s into $K^*$, and $v_s$ becomes zero. Again, since $l_n$'s child vertex $Out_j$ is in the same repair round, moving $l_n$ will not affect the cut-value contribution of other repair rounds. For the example in Fig. 4 and Fig. 5, the corresponding $K^*$ are the vertices in the left side of the solid line. We have

$$C(K) \geq C(K^*) \geq x_0 \alpha + \sum_{s \in T_1} x_s \alpha + \sum_{s \in T_2} y_s \beta,$$

where $T_1$ is the index set of repair rounds whose auxiliary vertices and in-vertices are all in $K^*$, and $T_2$ is the index set of repair rounds whose auxiliary vertices and in-vertices are all in $\overline{K}^*$. The newcomers in round $s$ are connected to $d$ helpers, which are located in rounds 0, 1, ..., $s - 1$. Of these $d$ helpers, at most $\sum_{i=0}^{s-1} x_i$ have their out-vertices in $\overline{K}^*$. Therefore, we have

$$y_s \geq d - \sum_{i=0}^{s-1} x_i,$$

for $s \in T_2$.

Thus we obtain the lower bound (5).

Next, we consider the constraints. It is clear that (6) and (7) must hold. Since the initial stage and the $k$ repair rounds have $k$ out-vertices in $\overline{K}$, we must have

$$x_0 + \sum_{1 \leq s \leq k} x_s \geq k.$$

Now we show that the inequality in (12) can be replaced by an equality. To see this, suppose $(x_0^*, x_1^*, ..., x_{k}^*, T^*_1)$ is an optimal solution, which yields the minimum value $B^*$. Let $l$ be the first repair round after which $\sum_{s=0}^{l} x_s^* \geq k$. Suppose $l$ is not the last round (i.e. $l \neq k$). If $k \in T^*_1$, then $x_i^*$ must be equal to 0, for otherwise, we can reduce its value and $B^*$ cannot be the minimum. If $k \in T^*_2$, we move $k$ into $T^*_1$ and set $x_k^* := 0$. Since this does not change the value of $B^*$, we can assume $k \in T^*_1$ and $x_k^* = 0$. The same argument can be repeatedly applied to round $k - 1$, round $k - 2$ and so on, so that we can assume $s \in T^*_1$ and $x_s^* = 0$ for all $s > l$. Consider round $l$. If $l \in T^*_1$, then $x_l = 0$ must be equal to $k$, for otherwise we can reduce the value of $x_l^*$ to obtain a value lower than $B^*$. If $l \in T^*_2$, we can reduce the value of $x_l^*$ by $\sum_{s=0}^{l} x_s^* - k$ without changing the value of $B^*$. Hence, replacing the inequality in (12) by an equality does not affect the value of the lower bound.

Theorem 2: $U^T = B$ for all $T \geq k$.

Proof: Theorem 1 states that $B \leq U^T$ for all $T \geq k$. It remains to prove that the lower bound $B$ is tight. Let $(x^*, T^*_1)$ be an optimal solution to the minimization in Theorem 1. We construct an instance $I^*$ with a particular failure pattern, DC$^*$ and a cut $X^*$ such that the cut-value $C(X^*)$ is exactly $B$.

The instance $I^*$ is constructed as follows. Firstly, we specify the failure pattern. In stage 0, choose any $r$ nodes in $N$ and let them fail. For stage $1 \leq s \leq k$, choose any $x_s^*$ active nodes.
in \( N \) and any \( r - x^*_s \) nodes in \( R_s \). Let them fail right before stage \( s + 1 \). Since the accumulated number of failed nodes in \( N \) is always no more than \( n \), i.e.,
\[
    r + \sum_{1 \leq s \leq k} x^*_s \leq r + k \leq n,
\]
where the first inequality follows from (8) and the second inequality follows from the assumption in the system model, and there are \( r \geq r - x^*_s \) nodes in \( R_s \) for every \( s \), we can always find such a failure pattern.

Next, we specify the helpers for each repair round. Since the number of remaining active nodes in \( N \) is
\[
    n - (r + \sum_{1 \leq s \leq k} x^*_s) = n - (r + k - x^*_0) \geq x^*_0,
\]
we can select any \( x^*_0 \) active nodes from \( N \) and denote them by \( M_0 \). Denote the \( x^*_s \) active nodes in \( R_s \) by \( M_s \). The helpers for repair round \( i \), for \( i = 1, 2, \ldots, s \), are chosen first from \( M_0 \), then from \( M_1 \), and so on, until \( d \) helpers are chosen. If \( \sum_{i=1}^{s-1} |M_i| < d \), the remaining helpers are chosen arbitrarily from the active nodes in \( N \). There are always enough active nodes in \( N \) to serve as helpers because
\[
    n - r - \sum_{1 \leq s \leq k} x^*_s \geq n - r - |M_1| - \cdots - |M_{s-1}| \geq d = |M_0| - \cdots - |M_{s-1}|.
\]
where (13) is the number of active nodes in \( N \) after stage \( s - 1 \), (14) follows from the definition of \( M_s \), and (15) is the number of required helpers in \( N \). The inequality holds because \( n - r \geq d \) according to the system model.

Finally, consider DC, which comes after the repair round \( k \) and connects to \( M_0 \cup M_1 \cup \cdots \cup M_k \).

The cut \( X^* \) is constructed as follows: For \( s \in \{0\} \cup T^*_1 \), put \( \text{Out}_{t_s} \) for \( i \in M_s \), into \( X^* \), and all the remaining vertices in round \( s \) into \( X^* \). Vertices in these repair rounds contribute \( x^*_0 \alpha + \sum_{s \in T^*_1} x^*_s \alpha \) to the cut-value. For \( s \in T^*_2 = \{1, 2, \ldots, k\} \setminus T^*_1 \), put all vertices in round \( s \) into \( X^* \). Vertices in these repair rounds contribute
\[
    \sum_{s \in T^*_2} (d - x^*_0 - \sum_{1 \leq i \leq s} x^*_i) \beta
\]
to the cut-value. Summing up the cut-value contribution of all the vertices, we get \( B \), showing that the bound is tight. \( \square \)

We give an example to demonstrate the above result. Consider the storage system WDSS \((n = 8, k = 3, d = 4, r = 2, \alpha = 2, \beta = 1)\). We can obtain an lower bound \( B = 5 \) to the min-cut value \( U \) from Theorem 1. One of the optimal solution to the minimization problem is \( T^*_1 = \{1, 2\} \) and \( x^*_0 = 0, x^*_1 = 1, x^*_2 = 0, x^*_3 = 2 \). According to Theorem 2 and the above optimal solution, let \( M_0 = \emptyset \), \( M_1 = \{9\}, M_2 = \emptyset \), \( M_3 = \{13, 14\} \), and DC be connected to \( \{9, 13, 14\} \). We can obtain a cut indicated as Cut 2, in Fig. 3, which has value \( \alpha + 3\beta = 5 \).

\[
    (16)
\]

The result in (16) meets the lower bound \( B \) in Theorem 1. Combining the above results, we can conclude that the min-cut value of this WDSS with \( T \geq k \) is 5.

\[ \text{V. Achievability of the Min-Cut Bound} \]

In this section, we show that for a WDSS with infinite repair rounds, the min-cut upper bound obtained in the previous section can be achieved by linear network coding with a finite alphabet. In other words, we show that \( C_{\text{storage}} = U \). To do this, we follow the idea of generic storage codes [5], which is based on the concept of generic network coding [49], [50]. In the following, we first introduce the refined information flow graph of the WDSS. Then based on the refined information flow graph, we show the construction of generic storage code and its properties. Finally, the achievability is shown by the existence of a generic storage code for this particular network.

\[ \text{A. Refined Information Flow Graph} \]

Given any information flow graph for a WDSS, we construct a refined information flow graph by introducing the concept of repair stage. In regard to the refined information flow graph, the repair process of \( r \) nodes is called a repair stage. From \( S \) to \( S_0 \), \( s \) is called stage \(-1\). The original \( n \) storage nodes are said to be in stage 0. In stage \( s > 0 \), the out-vertex of each storage node, except the failed ones in stage \((s-1)\), is connected to an auxiliary out-vertex by a directed edge of capacity \( \alpha \). This is a distinctive feature of the refined graph, which differs from the original flow graph. In addition, we add stage stamps for the out-vertices and auxiliary out-vertices. The out-vertex of node \( i \) in stage \( s \) is denoted by \( \text{Out}^*_s \). For any stage \( s \), if node \( i \) is a helper, then a new auxiliary h-vertex \( h^*_i \) is added to \( \text{Out}^*_s \) by an edge with capacity \( \beta \). Newcomers in this stage is connected to these new auxiliary h-vertices. This construction is essentially the same as the original flow graph. For the out-vertices of the newcomers, we also need to add stage stamps, i.e., re-label the out-vertex of the newcomer \( t \) as \( \text{Out}^*_s \).

After constructing the vertices and connecting them by edges as described above, we adjust the capacities of some edges in a way which does not affect the capacity of the network. Since each storage node has capacity \( \alpha \), every edge from \( S \) to \( S_0 \) with infinite capacity is replaced by an edge of capacity \( \alpha \). Furthermore, since each auxiliary h-vertex's in-edge only has capacity \( \beta \), we change all its out-edges' capacity from infinite to \( \beta \). After these changes, all the edge capacities are finite. Finally, for each edge in the network, if its capacity is \( c \), we replace it by \( c \) parallel edges of unit capacity. The above changes ensure all edges in the graph are of unit capacity, which facilitate our analysis in the following way: First, some graph theoretic concepts, such as edge-disjointness, are applicable only to edges of unit capacity. Second, the relationship between edges and global encoding kernels becomes one-to-one correspondence. Third, the relationship of the global encoding kernels can be used to describe properties of a given edge set more conveniently.

Note that the refined information flow graph represents a single-source multicast acyclic network. An example with \( n = 8 \), \( d = 4 \), and \( r = 2 \) is shown in Figure 6. In this example, nodes 1 and 2 fail in stage 0 and nodes 3 and 4 fail in stage 1.
B. Generic Storage Codes

Based on the refined information flow graph, we can introduce the generic storage codes of the WDSS. Before that, we first give some concepts of network coding [50], which are building blocks of generic storage codes.

Consider a single-source acyclic communication network and its corresponding graph. Let the alphabet $\Sigma$ be the finite field $GF(q)$. Suppose the message to be transmitted from the source node is an $\omega$-dimensional column vector $x$ over $GF(q)$. We add $\omega$ imaginary edges terminating at $S$ and assign them distinct vectors of the $\omega$-dimensional standard basis. These vectors are referred to as the global encoding kernels of the imaginary edges. For each edge $e(i,j)$, we iteratively define its global encoding kernel by

$$g_e = \sum_{d \in I(i)} l_{d,e} g_d,$$

where $I(i)$ is the set of all incoming edges of $i$. The transmitted symbol on edge $e$ is $x^T g_e$. An $\omega$-dimensional linear network code consists of a scalar $l_{d,e} \in GF(q)$ for every adjacent pair of edges $< d, e >$ in the network as well as a column $\omega$-vector $g_e$ for every edge $e$.

For an edge set $P$, denote the set of the corresponding global encoding kernels by

$$\ker(P) \triangleq \{ g_e : e \in P \},$$

and the linear span of $\ker(P)$ by

$$\vspace(P) \triangleq \text{span}(\ker(P)).$$

For a vertex $i$, define

$$\vspace(i) \triangleq \text{span}(\ker(I(i))).$$

An edge-disjoint path if they do not have any edge in common. A set of edges is said to be path-independent if each edge in this set is on a path originating from an imaginary edge and these paths are edge-disjoint. An edge set $P$ is said to be regular with respect to a linear network code if the global encoding kernels in $\ker(P)$ are linearly independent.

Since a DC can only connect to nodes in the same stage in the refined information flow graph, to make sure all DC can retrieve the file, it is sufficient to ensure all path-independent sets of edges in the same stage are regular. A code that satisfies this requirement can be regarded as a restricted form of a generic network code [49], [51]. We call it a generic storage code. Denote the set of all the edges in stage $s$, except the incoming edges of data collectors, by $E_s$. Generic storage codes can then be formally defined below:

Definition 2: An $\omega$-dimensional linear network code on a refined information flow graph is said to be an $\omega$-dimensional generic storage code if every path-independent $\omega$-subsets of $E_s$ is regular, for any stage $s = 0, 1, 2, \ldots$.

C. Achievability

Theorem 2 says that the min-cut value of the network limited to $T \geq k$ repair rounds is $B$. This only implies that $B$ is an upper bound of $C_{\text{storage}}$, since the latter requires that the system can tolerate unlimited rounds of failures and repairs. This question now is whether $B$ is achievable, i.e., whether $C_{\text{storage}} = B$. Note that the classical multicast network coding result does not apply, since the graph is infinite. In the following, we show that $B$ is achievable by the use of generic storage codes.

Theorem 3: A file with size $\omega = B$ can be stored in a WDSS($n, k, d, r, \alpha, \beta, \infty$) by the use of an $\omega$-dimensional generic storage code over $GF(q)$, where $q > (\frac{n \alpha + d}{\omega - 1})$. 

---

Fig. 6. An example of a refined information flow graph. Each link with label $e$ represents $e$ parallel edges of unit capacity.
Proof: If \( q > \left( \frac{n\alpha + d\beta}{\omega - 1} \right) \), we can construct an \( \omega \)-dimensional generic storage code over \( \text{GF}(q) \) on a refined information flow graph of a WDSS. The existence of such a code is shown in Lemma 5 in Appendix A. Now we show that any data collector can retrieve the file based on the code. Recall that any data collector is connected to \( k \) out-vertices in the same stage. By Theorem 2, the value of a cut is at least \( B \), so there are at least \( B \) disjoint paths terminating at any \( k \) out-vertices, in every stage \( s \geq 0 \). Thus there are at least one path-independent set, say \( P \), with size \( B \) within the incoming edges of these \( k \) out-vertices. By the definition of generic storage codes, the dimension of \( \ker(P) \) is \( B \), and the file with size \( B \) can be decoded. \( \square \)

Algorithm 1 Assign Global Encoding Kernels for the New Auxiliary h-vertices and Newcomers

Input: \{\( g_e : e \in I(\text{Out}^s_i) \) for all \( i \in A \)\} and \( h_1, h_2, \ldots, h_d \)

Output: \{\( g_e : e \in I(h_i^{s+1}), i = 1, 2, \ldots, d \), or \( e \in I(\text{In}_i), I(\text{In}_{i+1}), \ldots, I(\text{In}_{i+r-1}) \), or \( e \in I(\text{Out}^s_i), I(\text{Out}^s_{i+1}), I(\text{Out}^s_{i+r-1}) \)\}:

1. \( B_0 := \{ e \in I(\text{Out}^s_i) \) for all \( i \in A \}\);
2. for \( i := h_1, h_2, \ldots, h_d \) do
3. for \( j := 1, 2, \ldots, \beta \) do
4. \( e := \) the \( j \)-th incoming edges of \( h_i^{s+1} \) from \( \text{Out}^s_i \);
5. Choose a vector \( x \in \text{vspace}(\text{Out}^s_i) \) such that \( x \not\in \text{vspace}(\zeta) \), where \( \zeta \) is any \( \omega - 1 \)-subset of \( B_0 \) such that \( \zeta \) is regular and \( \text{vspace}(\text{Out}^s_i) \) \( \not\subset \text{vspace}(\zeta) \);
6. \( g_e := x \) and \( B_0 := B_0 \cup \{ e \} \);
7. end for
8. end for
9. for \( i := h_1, h_2, \ldots, h_d \) do
10. for \( j := 1, 2, \ldots, \beta \) do
11. \( e := \) the \( j \)-th incoming edges of \( h_i^{s+1} \) from \( \text{Out}^s_i \);
12. for \( l := 1, t, t+1, \ldots, t+r-1 \) do
13. \( e' := \) the \( j \)-th outgoing edges of \( h_i^{s+1} \) to \( \text{In}_i \);
14. \( g_{e'} := g_e \);
15. end for
16. end for
17. end for
18. for \( i := t, t+1, \ldots, t+r-1 \) do
19. for \( j := 1, 2, \ldots, \alpha \) do
20. \( e := \) the \( j \)-th incoming edges of \( \text{Out}^s_i \);
21. Choose a vector \( x \in \text{vspace}(\text{In}_i) \) such that \( x \not\in \text{vspace}(\zeta) \), where \( \zeta \) is any \( \omega - 1 \)-subset of \( B_0 \) such that \( \zeta \) is regular and \( \text{vspace}(\text{In}_i) \) \( \not\subset \text{vspace}(\zeta) \);
22. \( g_e := x \) and \( B_0 := B_0 \cup \{ e \} \);
23. end for
24. end for

The generic storage code can be constructed by Algorithm 1 in Appendix A. In each stage, there are at most \( \left( \frac{n\alpha + d\beta}{\omega - 1} \right) \) subsets to be considered for assigning a global encoding kernel for each edge. Since the total number of edges to be processed in each stage is \( d\beta + r\alpha \), for a given value of \( \omega \), the complexity of the algorithm is polynomial time in \( n\alpha + d\beta \) in each stage.

VI. Comparison With Cooperative Repair

We compare broadcast repair with cooperative repair, assuming both repair processes are triggered after the number of failed storage nodes accumulates to \( r \). To simplify the analysis, in the following, we restrict \( r \) to be a divisor of \( k \), i.e.,

\[
r u = k,
\]

for some positive integer \( u \). The storage capacity of a WDSS can then be expressed as follows:

**Theorem 4:** If (17) holds, the storage capacity of a WDSS is

\[
C_{\text{storage}} = \sum_{j=1}^{u} \min \left\{ r\alpha, (d - (j - 1)r)\beta \right\}.
\]

**Proof:** Define

\[
c_{T_1}(x) \triangleq x_0\alpha + \sum_{s \in T_1} x_s\alpha + \sum_{s \in T_2} (d - \sum_{i=0}^{s-1} x_i)\beta,
\]

where \( x = (x_0, x_1, \ldots, x_u) \) satisfying (6) to (8). Based on Lemma 6 shown in Appendix B, there is an optimal solution vector \( x^* = (x_0^*, x_1^*, \ldots, x_u^*) \) for \( c_{T_1} \), whose components are either \( x_i^* = r \) or \( x_i^* = 0 \). Because of (8) and (17), there are \( u \) components that have the value \( r \).

Recall that \( C_{\text{storage}} = \min_{T_1 \subseteq K} c_{T_1}(x) \), where \( T_1 \subseteq K \) and \( x \) is subject to (6) to (8). Let \( T_1^* \) be an optimal solution and denote \( |T_1^*| \) by \( t_1 \). It can be seen from (19) that there exists an optimal \( x^* \) in the form of

\[
(r, r, \ldots, r, r, 0, 0, \ldots, 0).
\]

Hence,

\[
C_{\text{storage}} = \min_{t_1 \in \{1, \ldots, u\}} t_1 r\alpha + \sum_{j=t_1+1}^{u} (d - (j - 1)r)\beta,
\]

which can be re-written as (18). \( \square \)

Now we compare broadcast repair with cooperative repair.

In cooperative repair, the newcomer receives packets from helper nodes through individual channels, and then exchange the encoded packets to all the other newcomers.

From (18), we can obtain a tradeoff between storage capacity \( \alpha \) and repair-transmission bandwidth \( \tau \). We cannot obtain better tradeoff since the storage capacity is already the maximum file size the system supported. Thus the tradeoff is optimal. Consider the two points, minimum storage (MS) point, which corresponds to the best storage efficiency, and the minimum repair-transmission bandwidth (MT) point, which corresponds to the minimum repair-transmission bandwidth on the tradeoff curve between repair-transmission bandwidth and storage (see Fig. 7 for example). In cooperative repair, the repair-transmission bandwidth is equal to the repair bandwidth. According to [17], the MS point and the MT point for cooperative repair are

\[
(\tau_{\text{MSC}}, \omega_{\text{MSC}}) = \left( \frac{d + r - 1}{k(d + r - k)}, \frac{1}{k} \right).
\]
and

$$\left( \tau_{\text{MSB}}, \alpha_{\text{MSB}} \right) = \left( \frac{2d + r - 1}{k(2d + r - k)}, 1, 1 \right),$$

respectively. Note that the values in the above expressions are normalized so that the file size (or equivalently, the storage capacity) is normalized to 1.

Next we consider broadcast repair. To derive the MS point and MT point for broadcast repair, for the purpose of normalization, $C_{\text{storage}}$ is assumed to be 1. At the MS point, $\alpha$ is equal to $1/k$ so that $k$ nodes can recover the file. When $\alpha = 1/k$, every term of (18) should be $r\alpha$ such that the sum of $u$ terms should equal 1. It is required that

$$(d - (u - 1)r)\beta \geq r\alpha.$$

Then we can obtain the MS point for broadcast repair as follows:

$$\left( \tau_{\text{MSB}}, \alpha_{\text{MSB}} \right) = \left( \frac{d}{k(d + r - k)}, 1, \frac{1}{k} \right).$$

At the MT point, the total number of transmitted symbols is equal to the total number of stored symbols in the newcomers. Therefore, we have $\alpha = \tau$. Hence, $r\alpha = d\beta \geq (d - (j - 1)r)\beta$ for $j = 1, 2, \ldots, u$. According to (18), we have

$$\sum_{j=1}^{u} (d - (j - 1)r)\beta = 1,$$

$$u \left[ d - \frac{r(u - 1)}{2} \right] \beta = 1,$$

from which we can obtain the MT point for broadcast repair as follows:

$$\left( \tau_{\text{MTB}}, \alpha_{\text{MTB}} \right) = \left( \frac{2d}{k(2d + r - k)}, 1, 1 \right).$$

It is easy to see that broadcast repair outperforms cooperative repair in the two points for any $r > 1$. In Fig. 7, we plot the tradeoff curves of the two repair schemes with parameters $C_{\text{storage}} = 1, d = 9, k = 4, r = 2$. We have

$$\left( \tau_{\text{MSB}}, \alpha_{\text{MSB}} \right) = \left( 0.321, 0.25 \right),$$

and

$$\left( \tau_{\text{MTB}}, \alpha_{\text{MTB}} \right) = \left( 0.281, 0.281 \right).$$

As a benchmark, we also plot the single-node repair, in which the repair is triggered whenever there is a single node failure. As reported in [17], cooperative repair performs better than single-node repair due to the benefit of node cooperation. On the other hand, when applied to WDSS, it performs worse than broadcast repair, since it does not exploit the broadcast nature of the wireless medium.

VII. Conclusions

In this work, we show that by exploiting the broadcast nature of the wireless channel, the performance of a WDSS can be improved. Based on the graph representation of a WDSS with finite number of repair rounds, we obtain the min-cut upper bound for the storage capacity of a WDSS with unlimited number of repair rounds. This upper bound is shown to be tight by the use of generic storage codes over the refined information flow graph. The fundamental tradeoff between the repair-transmission bandwidth and storage amount shows that broadcast repair for $r$ nodes outperforms the naive way of repairing these $r$ nodes one by one. We also compare broadcast repair with cooperative repair, both of which are specifically designed for repairing multiple nodes. While cooperative repair works very well in wired DSS, it is outweighed by broadcast repair in wireless environments.

We have shown that the optimal tradeoff under functional repair can be achieved by generic storage codes. Except for some special cases, it remains unknown whether exact-repair codes exist on the whole curve. It is theoretically challenging and practically important to construct such codes. We hope that our work can stimulate more studies on this interesting topic.

In this work, our model assumes that all the wireless storage nodes are within radio coverage of one another. In reality, the network topology may not be fully connected, and even if it is fully connected, the channel gains of different links can be different. Our model, however, provides a baseline study for performance evaluation of systems with other network topologies and more sophisticated physical-layer techniques of using the broadcast channel. Finally, we remark that our model is not limited only to WDSS. As it is equivalent to the centralized repair model considered in [41] and [42], our results can be directly applied to a DSS which performs repairs at a central location. This, for example, includes the rack-based architecture, under which when there is a rack failure, all the nodes within the rack need to be replaced and centralized repair can be performed at a leader node in the rack.

APPENDIX A

Lemma 5: Let $G$ be a refined information flow graph of a WDSS such that there is at least one path-independent set
of edges of size $\omega$ in each stage. An $\omega$-dimensional generic storage code on $G$ over $GF(q)$ can be constructed, provided that $q > \binom{n\alpha + d\beta}{\omega - 1}$. 

Proof: Let $q$ be a prime power greater than $\binom{n\alpha + d\beta}{\omega - 1}$. We prove the statement by mathematical induction on the number of repair stages in the refined information flow graph. We want to maintain the inductive invariant that, in any stage, any path-independent set of edges is regular.

Consider a refined information graph in stage $-1$. First, note that any $\omega$-subset of the edges in stage $-1$ is path-independent. We claim that there exists a linear code such that all these $\omega$-subsets are regular. For the first $\omega$ edges in stage $-1$, it is clear that they can be assigned linearly independent global encoding kernels. For each of the subsequent edges in stage $-1$, we can pick a vector $x \notin \text{vspace}(\zeta)$, where $\zeta$ is any ($\omega - 1$)-subset of edges that have already been assigned global encoding kernels. This can be done by picking a generator matrix of an $\omega$-dimensional Reed-Solomon code of length $n\alpha$. We can also assign the global encoding kernels sequentially, since

$$\big| \bigcup_{\zeta} \text{vspace}(\zeta) \big| \leq \binom{n\alpha}{\omega - 1} q^{\omega - 1} - q^w.$$ 

In stage 0, let the global encoding kernels of $\alpha$ outgoing edges of every $\text{In}_i, i = 1, 2, \ldots, n$ be the same as those of its $\alpha$ incoming edges. Since in stage $-1$, any $\omega$-subset of the $n\alpha$ edges is regular, so is any $\omega$-subset of the $n\alpha$ edges in stage 0.

Assume that for any refined information graph with $s \geq 0$ repair stages, (i.e., $s + 1$ stages including stage 0), a generic storage code has been constructed. By definition, any path-independent $\omega$-subset of $\text{E}_s$ is regular with respect to the constructed network code.

In stage $s + 1$, there are $n - r$ auxiliary out-vertices of the surviving nodes in stage $s$, $d$ new auxiliary in-vertices of the helpers, and $r$ newcomers. Let the set of indices of the $n - r$ surviving nodes be $A$. For $i \in A$, $\text{Out}_i^0$ has $\alpha$ incoming edges and $\alpha$ outgoing edges connecting to $\text{Out}_i^{s+1}$. Let the global encoding kernels of these $\alpha$ outgoing edges be the same as those of the $\alpha$ incoming edges. Let the $d$ helpers be indexed by $h_1, h_2, \ldots, h_d \in A$, where $h_1 < h_2 < \cdots < h_d$, and the index of the newcomers be $t, t + 1, \ldots, t + r - 1$.

It remains to determine the global encoding kernels for all the incoming edges $h_i^{s+1}, i = 1, 2, \ldots, d$, $\text{In}_t, \text{In}_{t+1}, \ldots, \text{In}_{t+r-1}$, and $\text{Out}_t^{s+1}, \text{Out}_{t+1}^{s+1}, \ldots, \text{Out}_{t+r-1}^{s+1}$ in such a way that any path-independent $\omega$-subset of $\text{E}_{s+1}$ is regular. This can be done by Algorithm 1, which is adapted from [50, Algorithm 19.34]. While the algorithm in [50, Algorithm 19.34] considers all edges in a graph, Algorithm 1 only needs to consider the edges within the same stage. Besides, in Algorithm 1, the global encoding kernels of some edges are directly obtained from previous stages, which is different from the algorithm in [50, Algorithm 19.34].

By construction, it can be seen that any path-independent $\omega$-subset of $\text{E}_{s+1}$ is regular. In the algorithm, the vector $x$ in line 5 can always be found. To see this, notice that there are at most $\binom{n\alpha + d\beta}{\omega - 1}$ edges in $B_0$ and the number of possible choices of $\zeta$ is at most $\binom{n\alpha + d\beta}{\omega - 1}$. Denote the dimension of $\text{vspace}(\text{Out}_{i}^{s+1})$ by $\nu$.

Since $\text{vspace}(\text{Out}_{i}^{s+1}) \nsubseteq \text{vspace}(\zeta)$, the dimension of $\text{vspace}(\text{Out}_{i}^{s+1}) \cap \text{vspace}(\zeta)$ is less than or equal to $\nu - 1$. Thus,

$$\big| \text{vspace}(\text{Out}_{i}^{s+1}) \cap \bigcup_{\zeta} \text{vspace}(\zeta) \big| \leq \binom{n\alpha + d\beta}{\omega - 1} q^{\nu - 1} - q^\nu = |\text{vspace}(\text{Out}_{i}^{s+1})|.$$ 

Likewise, the vector $x$ in line 21 can also be found. 

APPENDIX B

Lemma 6: If $ru = k$ for some positive integer value $u$, the problem min $c_{T_1}$ has an optimal solution vector, whose components are either 0 or $r$.

Proof: Let $x \triangleq (x_0, x_1, \ldots, x_k)$ be an optimal vector, which must exist since there are only finite possible choices of $x$. Since $r$ is a divisor of $k$ and the sum of all $x_i$’s is equal to $k$, $x$ cannot have exactly one component whose value is positive and strictly less than $r$. Assume $x$ has two or more components which are positive and strictly less than $r$, and we show that we can reduce the number of such components without increasing the value of the objective function.

Suppose $0 < x_t < r$ and $0 < x_j < r$ for some $l, j$ such that $0 < l < j < k$. Denote $\delta$ as any integer which satisfies

$$0 < \delta \leq \min\{r - x_t, x_t - x_j, x_j\}.$$ 

Let $x'$ be the same as $x$ except that its $l$-th and $j$-th components are different from those in $x$. We have

$$c_{T_1}(x) = \sum_{s \in \{0\} \cup T_1} x_s \alpha + \sum_{s \in T_2, l < i < s} (d - s - 1) x_i \beta + \sum_{s \in T_2, l < i < s} (d - 1) x_i \beta,$$

and

$$c_{T_1}(x') = \sum_{s \in \{0\} \cup T_1} x_s' \alpha + \sum_{s \in T_2, l < i < s} (d - s - 1) x_i' \beta + \sum_{s \in T_2, l < i < s} (d - 1) x_i' \beta.$$ 

In the following three cases: (i) $l, j \in \{0\} \cup T_1$, (ii) $l, j \in T_2$, or (iii) $l \in T_2, j \in \{0\} \cup T_1$, let $x_i' = x_i + \delta$ and $x_j' = x_j - \delta$. Since in these cases $\sum_{s \in \{0\} \cup T_1} x_s \alpha \geq \sum_{s \in \{0\} \cup T_1} x_s' \alpha$,

$$\sum_{s \in T_2, l < i < s} (d - s - 1) x_i \beta = \sum_{s \in T_2, l < i < s} (d - s - 1) x_i \beta,$$

$$\sum_{s \in T_2, l < i < s} (d - 1) x_i \beta \geq \sum_{s \in T_2, l < i < s} (d - 1) x_i' \beta,$$

and

$$\sum_{s \in T_2, j < i} (d - s - 1) x_i \beta = \sum_{s \in T_2, j < i} (d - s - 1) x_i' \beta.$$
Therefore,
\[ c_{T_1}(x) \geq c_{T_1}(x'). \]

Hence, in these cases, if there are two positive components whose values are strictly less than \( r \), the value of \( c_{T_1}(x) \) can be reduced by increasing the first component and decreasing the second by the same amount. This procedure can be repeated until the first component rises to \( r \) or the second one drops to 0. By repeating the argument, we can obtain an optimal solution that has no more than two components that are positive and strictly less than \( r \).

In the remaining case where \( l \in \{0\} \cup T_1 \), \( j \in T_2 \), we have
\[ c_{T_1}(x) - c_{T_1}(x') = (x_i - x_i') \alpha - \sum_{i \in T_2 \backslash \{j\}} (x_i - x_i') \beta. \]

If \( \alpha \geq \sum_{i \in T_2 \backslash \{j\}} \beta \), we can set \( x_i = x_i - \delta \) and \( x_i' = x_i' + \delta \); otherwise, we can set \( x_i' = x_i + \delta \) and \( x_i' = x_i - \delta \). Thus, we obtain \( c_{T_1}(x) \geq c_{T_1}(x') \). By repeating the same argument as above, we can also conclude that there is an optimal solution that has no more than two components that are positive and strictly less than \( r \).

\[ \square \]

REFERENCES
[38] M. Gerami, M. Xiao, J. Li, C. Fischione, and Z. Lin, “Repair for distributed storage systems with packet erasure channels and dedicated nodes for repair,” IEEE Trans. Commun., vol. 64, no. 4, pp. 1367–1383, Apr. 2016.

[39] M. Gerami, M. Xiao, and M. Skoglund, “Partial repair for wireless caching networks with broadcast channels,” IEEE Wireless Commun. Lett., vol. 4, no. 2, pp. 145–148, Apr. 2015.

[40] M. Dai, C. W. Sung, H. Wang, X. Gong, and Z. Lu, “A new zigzag-decodable code with efficient repair in wireless distributed storage,” IEEE Trans. Mobile Comput., vol. 16, no. 5, pp. 1218–1230, May 2017.

[41] A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, “Centralized repair of multiple node failures,” in Proc. IEEE Int. Symp. Inf. Theory, Barcelona, Spain, Jul. 2016, pp. 1003–1007.

[42] A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath. (Mar. 2016). “Centralized repair of multiple node failures with applications to communication efficient secret sharing.” [Online]. Available: https://arxiv.org/abs/1603.04822

[43] M. Zorgui and Z. Wang, “Centralized multi-node repair in distributed storage,” in Proc. Allerton Conf. Commun. Control Comput., Monticello, IL, USA, Sep. 2016, pp. 617–624.

[44] M. Zorgui and Z. Wang, “Centralized multi-node repair for minimum storage regenerating codes,” in Proc. IEEE Int. Symp. Inf. Theory, Aachen, Germany, Jun. 2017, pp. 2213–2217.

[45] M. Zorgui and Z. Wang. (Jun. 2017). “Centralized multi-node repair regenerating codes.” [Online]. Available: https://arxiv.org/abs/1706.05431

[46] M. Ye and A. Barg, “Explicit constructions of high-rate MDS array codes with optimal repair bandwidth,” IEEE Trans. Inf. Theory, vol. 63, no. 4, pp. 2001–2014, Apr. 2017.

[47] L. K. Ford, Jr., and D. R. Fulkerson, Flows in Networks. Princeton, NJ: Princeton Univ. Press, 1962.

[48] T. Ho et al., “A random linear network coding approach to multicast,” IEEE Trans. Inf. Theory, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[49] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding,” IEEE Trans. Inf. Theory, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[50] R. W. Yeung, Information Theory and Network Coding. Boston, MA: Springer, 2008.

[51] M. Tan, R. W. Yeung, S.-T. Ho, and N. Cai, “A unified framework for linear network coding,” IEEE Trans. Inf. Theory, vol. 57, no. 1, pp. 416–423, Jan. 2011.

Ping Hu received the B.Eng. degree in telecommunications engineering from Xidian University, Xi’an, China, in 2006, and the M.Phil. and Ph.D. degrees from the City University of Hong Kong, Hong Kong, in 2009 and 2016, respectively. She was with Alcatel-Lucent Shanghai Bell, Shanghai, China, from 2010 to 2012. She is currently with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. Her research interests include wireless communications, information-theoretic security, and distributed storage systems.

Chi Wan Sung (M’98–SM’96) was born in Hong Kong. He received the B.Eng., M.Phil., and Ph.D. degrees in information engineering from The Chinese University of Hong Kong in 1993, 1995, and 1998, respectively. After graduation, he was an Assistant Professor at The Chinese University of Hong Kong. He joined the faculty at the City University of Hong Kong in 2000, and is currently an Associate Professor with the Department of Electronic Engineering. His research interest is on coding, communications, and networking, with emphasis on algorithm design and complexity analysis. He has served as TPC members for various international conferences. He was an Associate Editor of the Transactions on Emerging Telecommunications Technologies from 2013 to 2016. He is currently on the editorial boards of the ETRI Journal and Electronics Letters.

Terence H. Chan received the B.Sc. (Mathematics), master’s, and Ph.D. degrees in information engineering from The Chinese University of Hong Kong in 1996, 1998, and 2000, respectively. In 2001, he was a Visiting Assistant Professor with the Department of Information Engineering, The Chinese University of Hong Kong. From 2002 to 2004, he was a Post-Doctoral Fellow at the Department of Electrical and Computer Engineering, University of Toronto. He was an Assistant Professor with the University of Regina from 2004 to 2006. He is currently an Associate Professor with the Institute for Telecommunications Research, University of South Australia. He was a recipient of the Croucher Foundation Fellowship and the Sir Edward Youde Fellowship in 2002 and 2000, respectively. He is the IEEE Information Theory Society, Joint South Australia/ACT/VIC/NSW/QLD Sections Chapter Chair. He serves as the Technical Co-Chair for the 2011 and 2015 IEEE International Symposium on Network Coding.