

Capacity of dynamical storage systems

Ohad Elishco

Alexander Barg

Abstract

We introduce a dynamical model of node repair in distributed storage systems wherein the storage nodes are subjected to failures according to independent Poisson processes. The main parameter that we study is the time-average capacity of the network in the scenario where a fixed subset of the nodes support a higher repair bandwidth than the other nodes. The sequence of node failures generates random permutations of the nodes in the encoded block, and we model the state of the network as a Markov random walk on permutations of n elements. As our main result we show that the capacity of the network can be increased compared to the static (worst-case) model of the storage system, while maintaining the same (average) repair bandwidth, and we derive estimates of the increase. We also quantify the capacity increase in the case that the repair center has information about the sequence of the recently failed storage nodes.

I. Introduction

The problem of node repair based on erasure coding for distributed storage aims at optimizing the tradeoff of network traffic and storage overhead. In this form it was established by [9] from the perspective of network coding. This model was generalized in various ways such as concurrent failure of several nodes [7], heterogeneous architecture [2], [18], cooperative repair [14], and others. The existing body of works focuses on the failure of a node (or several nodes) and the ensuing reconstruction process, but puts less emphasis on the time evolution of the entire network and the inherent stochastic nature of the node failures. The static point of view of the system and of node repair leads to schemes based on the worst case scenario in the sense that the amount of data to be stored is known in advance, the amount of data each node transmits is known, and the repair capacity is determined by the least advantageous state of the network. Switching to evolving networks makes it possible to define and study the average amount of data moved through the network to accomplish repair, and may give a more comprehensive view of the system.

Several models of storage systems have been considered in the literature. The basic model of [9] assumes that the amount of data that each node transmits to the repair center is fixed. The analysis of the network traffic and storage overhead relies on [1] which quantifies the maximum total amount of data (or flow) that can arrive at a specific point, but does not specify the exact amount of data that each node should transmit at each time instant. To use the communication bandwidth more efficiently, we assume the amount of data that each node transmits changes over time, while the total amount of communicated information averaged over multiple repair cycles is fixed.

A similar idea appears, although not explicitly, in [17], where the authors propose to perform repair of several failed nodes within one repair cycle with the purpose of decreasing the network traffic. The decrease can occur if the information sent over a particular link can be used for repair of more than one node, thereby decreasing the repair bandwidth. This scheme, which the authors called “lazy repair,” views the link capacity as a resource in network optimization, which in general terms is similar to the underlying premises of our study. A related, more general model of storage that accounts for time evolution of the system, given in [16], attempts to optimize tradeoffs between storage durability, overhead, repair bandwidth, and access latency. Coding for minimizing latency has been considered on its own in a separate line of works starting with [13]. We refer to [4] for an overview of the literature where access latency is considered in the framework of queueing theory.

This paper was presented in part at the 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, July 2019.

O. Elishco is with Institute for Systems Research, University of Maryland, College Park, MD 20742, email ohadeli@umd.edu. His research is supported by NSF grant CCF 1814487.

A. Barg is with Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742 and also with Institute for Problems of Information Transmission (IITP), Russian Academy of Sciences, 127051 Moscow, Russia. Email: abarg@umd.edu. His research was supported by NSF grants CCF1618603 and CCF1814487.

To further motivate the dynamical model, recall that cloud storage systems such as Microsoft Azure or Google file system encode information in blocks. The information to be stored is accumulated until a block is full, and then the block is sealed: the information is encoded, and the encoded fragments are distributed to storage nodes [8], [15]. This implies that a storage node contains encoded fragments from several different blocks and that the sets of storage nodes corresponding to different blocks may intersect partially. Therefore, a storage node may participate in recovery of several failed nodes simultaneously, which implies that the capacity of the link between the node and the repair center can be considered as a shared resource.

In this work, we make first steps toward defining a dynamical model of the network with random failures. The prevalent system model assumes homogeneous storage under which the links from the nodes to the repair center all have the same capacity. We immediately observe that the dynamical approach does not yield an advantage in the operation or analysis of this model. For this reason we study storage systems such that the network is formed of two disjoint groups of nodes with unequal (average) communication costs, which was proposed in the static case in [2]. We show that, under the assumption of uniform failure probability of the nodes, it is possible to increase the size of the file stored in the system while maintaining the same network traffic. This means that, while in [2] the node transmits the same amount of data each time that there is a failure, in our model the same node will transmit the same amount of data in the average over a sequence of repair cycles (the time). In addition, we provide a simple scheme that increases the size of the stored file compared to the static model, and study state-aware dynamical networks in which the repair center has causal knowledge of the sequence of the failed nodes. The idea of time averaging is motivated by the assumption that the network exhibits some type of ergodic behavior whereby the expected capacity can be related to minimum cut averaged over time in a sample path of the network evolution. Since in our derivations we rely on the value of the minimum cut, in effect we are assuming “functional repair” of the failed nodes as opposed to the more stringent requirement of exact repair [5], [9].

In Section II, we present the dynamical model and give a formal definition of the storage capacity. The evolution of the network is formalized as a random walk on the set of node permutations. Using this representation, we argue that it suffices to limit oneself to discrete time. We also prove the basic relationship between the storage capacity of a continuous-time network and the time-average min-cut of the corresponding discrete-time network (Sec. II-D). This part relies on standard arguments related to the discrete-time Markov chain obtained by sampling a Poisson process at the times of change. The main results of this paper are collected in Sec. III where we derive estimates of the average capacity of the fixed-cost storage model. We examine two approaches toward estimating the capacity. The first of them is related to a specific transmission protocol while the second relies on an averaging argument. In Section IV we analyze state-aware networks and extend the ideas of the previous section to obtain a lower bound on their capacity. Finally, in Section V we consider the case of different failure probabilities of the nodes and establish a partial result regarding a lower bound on capacity.

II. Model Definition

In this section we define a storage network that evolves in time and describe the basic assumptions that characterize this evolution. We also define a sequence of information flow graphs, which enables us to define the capacity of a randomly evolving network.

A. Evolution of the network

A storage network is a set of data storage units that save information (“file”) with the purpose of being able to retrieve it at a later time. The file is partitioned into fragments placed on different storage drives or nodes of the system. Node failures occur regularly, and to maintain the integrity of the data, the file is encoded using an erasure-correcting code. This incurs a penalty in terms of both the storage overhead and increased network communication and delay in the course of repair of the failed nodes. Once a node has failed, the system initiates the reconstruction process in the course of which the centralized computing unit (CU) downloads information from a subset of functional storage nodes and performs the recovery of the data stored on the failed node. The amounts of data downloaded from the different helper nodes to the CU vary over time, and are selected with the objective of minimizing the repair bandwidth. Thus, the sequence of node repairs is a time-dependent process which accounts for the time evolution of the network in terms of the information flow graph.

Apart from node repair, the system also performs the operation of data collection (reading the file). This operation is performed by a Data Collector (*DC*) which contacts storage nodes that allow the retrieval of the data. Since the file is encoded with an erasure-correcting code, the *DC* can retrieve the file by contacting a subset of the storage nodes.

Let us give a formal description of our storage network model. A storage network is a pair (\mathcal{N}, β) where \mathcal{N} is a triple $\mathcal{N} = (V, DC, CU)$ in which V is a set of n nodes (storage units) $V = \{v_1, \dots, v_n\}$, DC is the data collector node, and CU is the centralized computing unit node. The real nonnegative vector $\beta = (\beta_1, \dots, \beta_n)$ gives the maximum average amount of data communicated from v_i for the node repair, and will be discussed in more detail below.

Every node v_i , $i \in [n] \triangleq \{1, 2, \dots, n\}$ has the ability to store up to α symbols over some finite alphabet F . To store a file of size M we encode the file using an (n, k) code \mathcal{C} . The coordinates of the codeword are vectors over F , and each coordinate is stored in its own storage node in V . To read the file, the *DC* accesses at least k nodes, obtaining the information stored in them, and retrieves the original file.

The time evolution of the storage network is related to a random process of node failures. We begin our study assuming that time is continuous starting at $t = 0$, when the encoded file is stored in the network. The time instances t_1, t_2, \dots indicate consecutive node failures. Let $s = (s_1, s_2, \dots) \in V^\infty$ be the sequence of failed nodes, where s_j is the node that fails at time t_j . We assume that in order to restore the data to a failed node (reconstruct the node), the *CU* contacts a group of storage nodes, called helper nodes, accesses some of the data stored on them, and uses this data to accomplish the recovery. In this work we assume that *CU* contacts all the nodes except the failed node, i.e., we assume that the number of helper nodes is $n - 1$. Further, we assume that the definition of the storage network includes a set of parameters β_i , $i = 1, \dots, n$, where β_i is the maximum amount of information that is downloaded from v_i to *CU* for node repair, averaged over the time instances t_i . Specifically, we define a sequence of functions $\{h_j\}_j$, where $h_j : \mathcal{A}_j \rightarrow \mathbb{N}$, that determines the number of symbols that each node transmits for the recovery of the node s_j . Thus, node v_i provides $h_j(v_i)$ symbols of F for the repair of node s_j . It is assumed that $\limsup_{j \rightarrow \infty} \frac{1}{j} \sum_{j=1}^j h_j(v_i) \leq \beta_i$ for all i . The case of $h_j(v_i) = \beta_i$ will play a special role, and we introduce a notation for it: Let

$$h_j^*(v_i) = \begin{cases} \beta_i & \text{for all } j : s_j \neq v_i \\ 0 & \text{for all } j : s_j = v_i. \end{cases} \quad (1)$$

We will also write h, h^* to refer to the infinite sequences $\{h_j\}_j, \{h_j^*\}_j$, respectively.

Note that by definition, the weight function h^* does not achieve β with equality. This is because $h_j^*(v_i) = 0$ whence $s_j = v_i$. Thus, it is possible to increase the maximum file size that can be stored by taking $h_j(v_i) = (1 + \frac{1}{n-1})h_j^*(v_i)$. However, this increment in the file size will also increase the repair bandwidth. Although, for simplicity, the results in this paper are compared to the constraints β , it is straightforward to compare the results to the constraints $(1 + \frac{1}{n-1})\beta$.

Given \mathcal{N} and a sequence $s = (s_1, s_2, \dots)$ of nodes, we define a sequence of directed graphs $\{\mathcal{X}_j^s\}_{j \in \mathbb{N}}$, called **information flow graphs**, where \mathcal{X}_j^s corresponds to $t \in [t_j, t_{j+1})$ and is a subgraph of \mathcal{X}_{j+1}^s . When no confusion occurs, we will write \mathcal{X}_j . The sequence of information flow graphs is a formalization of the notion of a (time-evolving) information flow graph that appears in the foundational paper [9]. For ease of description (and in accordance with [9]), we introduce a new node \tilde{v} which is called the source node.

Definition:

- 1: Let $V_0 = V \cup \tilde{v}$ and put $\mathcal{X}_0 = (V_0, E_0)$, i.e., all the nodes in \mathcal{N} and the source node \tilde{v} , with edges

$$E_0 = \{(\tilde{v}, v_i) : i \in [n]\}.$$

The nodes in the set $\mathcal{A}_0 := V$ are called the *active nodes* of the graph \mathcal{X}_0 . We define $\mathcal{A}_{-1} = \{\tilde{v}\}$.

- 2: Suppose that $s_1 = v_{i_1}$, $i_1 \in [n]$ and define a new node (newcomer) $v_{i_1}^1$. The superscript 1 implies that there was one failure and v_{i_1} is the node that failed, and the recovered node is $v_{i_1}^1$. The graph $\mathcal{X}_1 = (V_1, E_1)$ is formed as follows:

$$\begin{aligned} V_1 &= V_0 \cup \{CU_1, v_{i_1}^1\} \\ E_1 &= E_0 \cup \{(v_j, CU_1), j \in [n] \setminus \{i_1\}\} \cup (CU_1, v_{i_1}^1). \end{aligned}$$

The set of active nodes of \mathcal{X}_1 is defined as $\mathcal{A}_1 := (\mathcal{A}_0 \setminus \{v_{h_1}\}) \cup \{v_{h_1}^1\}$.

- 3: Suppose we are given the graph $\mathcal{X}_{j-1}, j \geq 2$. Suppose that $s_j = v_{i_j}$ and consider the corresponding node $v_{i_j}^{j'}$ in \mathcal{X}_{j-1} for some $j' < j$. The superscript j' means that the node v_{i_j} is the (j') th node that had been recovered (i.e., the superscript serves as a counter for the number of failures). Define a new node $v_{i_j}^j$ and define $\mathcal{X}_j(V_j, E_j)$ as follows:

$$V_j = V_{j-1} \cup \{CU_j, v_{i_j}^j\}$$

$$E_j = E_{j-1} \cup \{(u, CU_j) : u \in \mathcal{A}_{j-1} \setminus \{v_{i_j}^{j'}\}\} \cup (CU_j, v_{i_j}^j).$$

The set of active nodes of \mathcal{X}_j is defined as $\mathcal{A}_j = (\mathcal{A}_{j-1} \setminus \{v_{i_j}^{j'}\}) \cup \{v_{i_j}^j\}$. We refer to Figure 1 for an illustration.

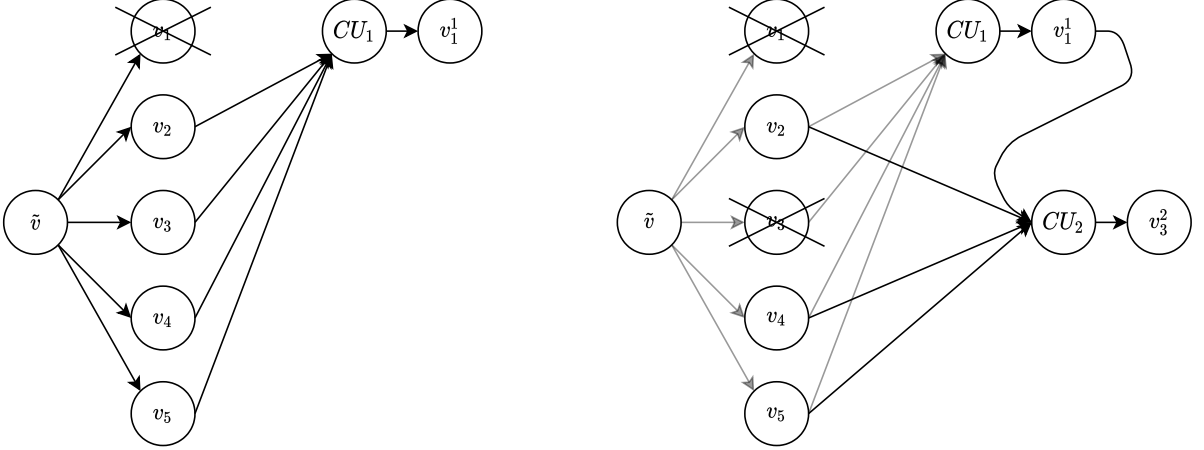


Figure 1. An illustration of \mathcal{X}_1^s (left figure) and \mathcal{X}_2^s (right figure) with $s = (v_1, v_3, \dots)$. The set of active nodes for $j = 1$ is $\mathcal{A}_1 = \{v_1^1, v_2, v_3, v_4, v_5\}$ and the set of active nodes for $j = 2$ is $\mathcal{A}_2 = \{v_1^1, v_2, v_3^2, v_4, v_5\}$.

The sequence of information flow graphs is an important tool used to represent the time evolution of the network. Each graph in the sequence accounts for a new node failure, and also records the information regarding all the past failures that occurred from the $t = 0$ time. For a given j , the information for the repair of s_j is communicated over the edges in the graph \mathcal{X}_j , wherein the edge (v_i^ℓ, CU_j) carries $h_j(v_i)$ symbols of F , the index $\ell < j$ corresponds to the last instance when the node v_i has failed.

We will sometimes write $(\mathcal{N}, \beta, s, \mathbf{t}, h)$ to denote a network (\mathcal{N}, β) with the sequence of failed nodes s , the sequence of failure times $\mathbf{t} = (t_1, t_2, \dots)$, and a sequence of functions $\{h_j\}_j$.

In our model, the evolution of the network is random. Following earlier literature on storage networks, e.g., [16], we represent this evolution by assuming that the failure of each node is a Poisson arrival process with rate λ , and these arrivals occur independently for different nodes. The interarrival time between two failures of a specific node $v \in V$ is an exponential random variable with pdf $\lambda e^{-\lambda t}$. Since node failures are independent, the overall rate of node failures in the system is a Poisson process with parameter $n\lambda$. This implies that we can formulate the network time evolution as follows. Let (X_1, X_2, \dots) be a sequence of i.i.d. random variables with pdf $f_X(t) = n\lambda e^{-n\lambda t}$. Let $T = (T_1, T_2, \dots)$ be the sequence of failure times defined as $T_j = \sum_{i=1}^j X_i$ for $j \in \mathbb{N}$ and let $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \dots)$ be the sequence of failed nodes defined as a sequence of i.i.d. random variables distributed uniformly over $[n]$. Note that with probability zero the values T_j can be infinite. Denote by μ_1 the infinite direct power of the uniform distribution on V and by μ_2 the infinite power of the exponential distribution on $[0, \infty)$. We will assume that the sequence (\mathbf{S}, T) is distributed according to $\mu_1 \times \mu_2$.

B. Data retrieval and network capacity

To retrieve the file, the *DC* contacts k or more storage nodes and reads the information stored on them. Assume that the read request occurs at time $t \in [t_j, t_{j+1})$ for some $j \in \mathbb{N}$. In the information flow graph \mathcal{X}_j , the reading process amounts to introducing a new node DC_j with at least k incoming edges. Each edge

originates in an active node. The set of these in-neighbors of DC_j is denoted by $D_j \subseteq \mathcal{A}_j$, $|D_j| \geq k$. The edges $\{(v, DC_j) : v \in D_j\}$ have infinite weight. We denote by k' the minimal number of active nodes from which the entire file can be retrieved.

We are interested in the storage capacity of the network which is the maximum size of the file that can be stored in the network and be retrieved at any time while satisfying the average bandwidth constraints given by β . Before defining the storage capacity, we need the following definition.

Definition 1 Let $(\mathcal{N}, \beta, s, \mathbf{t}, h)$ be a storage network with a sequence of functions $\{h_j\}_j$. The h -**capacity** of \mathcal{N} , denoted by $\text{cap}_h(\mathcal{N})$, is the maximum file size that can be saved on the network \mathcal{N} and retrieved at any time.

Example 1 Let $(\mathcal{N}, \beta, s, \mathbf{t}, h^*)$ be a storage network and assume $\beta_i = \beta_0$ for all $i \in [n]$. Note that in this case when a node fails all the active nodes transmit exactly β_0 symbols for the recovery process. It was shown in [9] that the capacity in this case is equal to $\text{cap}_{h^*}(\mathcal{N}) = \sum_{i=1}^{k'} \min\{(n-i)\beta_0, \alpha\}$.

Note that the h^* -capacity expression contains a minimum. In order to simplify notation, we assume throughout that α is large enough, i.e., the storage nodes can contain any amount of information. This assumption allows us to remove the minimum in the capacity expression. In the previous example, we obtain that $\text{cap}_{h^*}(\mathcal{N}) = \sum_{i=1}^{k'} (n-i)\beta_0$.

We now define the storage capacity.

Definition 2 Let $(\mathcal{N}, \beta, s, \mathbf{t})$ be a storage network and let \mathcal{H} denote the set of all sequences of functions h that satisfy the constraints given by β . The **storage capacity** (or just **capacity**) of $(\mathcal{N}, \beta, s, \mathbf{t})$, denoted by $\text{cap}(\mathcal{N})$, is defined as

$$\text{cap}(\mathcal{N}) = \sup_{h \in \mathcal{H}} \text{cap}_h(\mathcal{N}).$$

The random evolution of the network makes the sequence of failed nodes a sequence of random variables which we henceforth denote by \mathbf{S} . This makes $\text{cap}(\mathcal{N})$ a random variable as well. As such, we will analyze the expected value of the storage capacity which is defined as follows.

Definition 3 Let $(\mathcal{N}, \beta, \mathbf{S}, \mathbf{t})$ be a (random) storage network. The **expected capacity** is defined as

$$\overline{\text{cap}}(\mathcal{N}) \triangleq \mathbb{E}[\text{cap}(\mathcal{N})].$$

For any realization s of \mathbf{S} , the storage capacity of a network can be calculated using the sequence of information flow graphs $\{\mathcal{X}_j\}_j$. Indeed, let $(\mathcal{N}, \beta, s, \mathbf{t}, h)$ be a storage network with a corresponding sequence of information flow graphs $\{\mathcal{X}_j\}_j$. For a time $t \in [t_j, t_{j+1}]$, $j \in \mathbb{N}$, let D_t denote a selection of k' nodes from \mathcal{A}_j (from which the entire file can be retrieved). As shown in [9], the storage capacity of the network is equal to the minimum weight of a cut between \tilde{v} and D_t . In other words, the maximum file size that can be reliably stored in the network and retrieved at time t is equal to the minimum weight of a cut between \tilde{v} and D_t . A cut between \tilde{v} and D_t is a partition of the vertices into two sets where \tilde{v} is contained in the first set and the k' nodes in D_t are contained in the second set. The weight of the cut is the sum of the weights of the edges from the first set to the second set. Therefore, if we can find a weight function h such that for every time instance t and for every selection D_t of k' active nodes a minimum cut between \tilde{v} to D_t is bounded below by a constant C , then C is the maximum file size that can be saved in the network. Our goal is to bound C by specifying a weight function h that obeys the restrictions given by the average bandwidth of the nodes.

In this work, we consider the time-average minimum cut (as defined below) and hence we define the minimum cut accordingly.

Definition 4 Let $C_t^h(D_t)$ denote the value of the minimum cut in \mathcal{X}_j between $(\bigcup_{i=-1}^{j-1} \mathcal{A}_i) \setminus \mathcal{A}_j$ and D_t under the weight assignment h . Further, let C_t^h denote the minimum cut over all selections D_t ,

$$C_t^h = C_t^h(\mathcal{A}_j) \triangleq \min_{D_t \subseteq \mathcal{A}_j, |D_t|=k'} \{C_t^h(D_t)\}. \quad (2)$$

When $h = h^*$, we will sometimes write C_t instead of $C_t^{h^*}$.

In this definition we again assume that the DC is not aware of the state of the network, i.e., the order of the failed nodes, and the minimum accounts for the worst case. If DC can choose which nodes to contact, the minimum should be replaced with a maximum.

Definition 5 Let $(\mathcal{N}, \beta, s, t, h)$ be a storage system. Define the **average cut** as

$$C_{\text{avg}}^h(\mathcal{N}) \triangleq \limsup_{t \rightarrow \infty} \frac{1}{t} \int_0^t C_\tau^h d\tau.$$

Note that the average cut is a function of s . Hence, if the network is random $s = \mathbf{S}$, then the average cut is a random variable. As shown in the following lemma, for a storage system $(\mathcal{N}, \beta, s, t, h^*)$, the average cut $C_{\text{avg}}^{h^*}(\mathcal{N})$ can be used to bound below the capacity of the network \mathcal{N} .

Lemma 1 Let $(\mathcal{N}, \beta, s, t)$ be a storage system. Assume that for any $n - k'$ nodes $v_{i_1}, \dots, v_{i_{n-k'}} \in V$ and all failure times $t = t_1, t_2, \dots$

$$\sum_{j=1}^{n-k'} \beta_{i_j} \geq \max_{D_t \subseteq \mathcal{A}_t, |D_t|=k'} |C_t^{h^*}(D_t) - C_{\text{avg}}^{h^*}|. \quad (3)$$

If every node fails infinitely often, then

$$\text{cap}(\mathcal{N}) \geq C_{\text{avg}}^{h^*}(\mathcal{N}). \quad (4)$$

Remark: Eq. (4) in effect states that there exists a weight assignment $h \in \mathcal{H}$ such that $\text{cap}_h(\mathcal{N})$ is at least the size of the average cut under h^* .

Proof: We prove the lemma by describing an algorithm for weight selection. In order to simplify the notation, for $t \in [t_j, t_{j+1})$ we use the subscript j instead of t , for example, we write C_j instead of C_t . Let j_1 be the first occurrence when $C_{j_1} < C_{\text{avg}}^{h^*}$. If j_1 is infinite, then there is nothing to prove; otherwise, for $j < j_1$ take $h_j = h_j^*$. Let v_i be the node that failed at time $t \in [t_{j_1}, t_{j_1+1})$. For every $v_\ell \in \mathcal{A}_j$ ($v_\ell \neq v_i$) define $h_{j_1}(v_\ell) = (1 + \varepsilon_0)h_{j_1}^*(v_\ell)$ for $\varepsilon_0 > 0$ such that the minimum cut $C_{j_1}^h(\mathcal{N}) = C_{\text{avg}}^{h^*}(\mathcal{N})$, i.e., every active node transmits more information to enlarge the minimum cut to $C_{\text{avg}}^{h^*}(\mathcal{N})$. For time t_{j_1+1} again find ε_1 such that for every $v_\ell \in \mathcal{A}_{j_1+1}$, taking $h_{j_1+1}(v_\ell) = (1 + \varepsilon_1)h_{j_1+1}^*(v_\ell)$ yields $C_{j_1+1}^h(\mathcal{N}) = C_{\text{avg}}^{h^*}(\mathcal{N})$. Note that ε_1 can be positive or negative. Continue this way to define h_{j_1+r} , and the respective values of ε_r , $r \geq 2$.

By construction, the average number of symbols that node v_i transmits is β_i . Moreover, at any time instance t_ℓ , the file can be reconstructed from any selection D_ℓ of k' storage nodes. Hence, the minimum cut between \tilde{v} and D_ℓ (the storage capacity) is at least $C_{\text{avg}}^{h^*}(\mathcal{N})$. ■

To explain assumption (3) note the following. Suppose a node was one of the nodes selected by DC_{t-1} , and suppose that it fails at time $t - 1$. If the recovered node is selected by DC_t , the weight of the cut between the source and the selected nodes is affected only by the $n - k'$ in-edges that connect the nodes not selected by DC_t to the failed node. Assumption (3) implies that for every selection D_t of the k' nodes and for every $r \geq 1$, it is possible to find an $\varepsilon_r \geq -1$, which ensures the consistency of the transmission. The physical interpretation of the assumption given in (3) is that any set of $n - k'$ nodes contain more “new information” than the information unavailable due to the node failure. Throughout the paper we assume that (3) holds true.

We will establish a more detailed lower bound on $\text{cap}(\mathcal{N})$ in terms of $C_{\text{avg}}^{h^*}(\mathcal{N})$ in Sec. II-D below.

C. Network evolution as a sequence of permutations

In this subsection we define a set of permutations related to the sequence of failed nodes $s = (s_1, s_2, \dots)$. Each node in \mathcal{A}_j is denoted by $v_{i_j}^{j'}$ for some $j' \leq j$, and it can be identified with the node $v_i \in V$. Therefore, if at some point t_{j_0} all the nodes have failed at least once, then for $j \geq j_0$ the order in which the nodes in \mathcal{A}_j have failed can be identified with a permutation of the set $[n]$. For example, if $\mathcal{A}_j = \{v_1^{j_1}, v_2^{j_2}, \dots, v_n^{j_n}\}$, then the corresponding permutation π is such that $\pi(i) \leq \pi(\ell)$ iff $v_i^{j_i}, v_\ell^{j_\ell} \in \mathcal{A}_j$ with $j_i \leq j_\ell$. Below we identify V and $[n]$ and consider π_t , $t \geq t_{j_0}$ as a permutation of either of these sets as appropriate. We denote by \mathfrak{S}_n the set of all permutations of $[n]$. The permutations π_t are associated with the sequence of the information flow graphs (\mathcal{X}_j) , and we call π_t the associated permutation (at time t). Note that the associated permutation $\pi_t \in \mathfrak{S}_n$ corresponds to the order of the n most recent node failures. Hence, for $t \in [t_j, t_{j+1})$ we will sometimes write

π_j instead of π_t to refer to the associated permutation at time t . Observe that the minimum cut $C_t(\mathcal{A}_j)$ is a function of the associated permutation π_t for every $t \in [t_j, t_{j+1})$ and $j \geq j_0$, so we can write $C_t^h(\mathcal{A}_j) = C_t^h(\pi_t)$.

It is possible to obtain $\pi_t, t \geq t_{j_0}$ from s by considering only the last appearance of each node as seen in the next example.

Example 2 Assume that $|V| = 5$ and assume that $s = (v_1, v_2, v_3, v_4, v_5, v_2, v_1, v_5, \dots)$ with $\mathbf{t} = (1, 2, 3, \dots)$. Then $\pi_t, t \in [1, 5)$ is not defined and $\pi_t = (v_1, v_2, v_3, v_4, v_5) = id$ for $t \in [5, 6)$ since all the nodes had failed by $t = 5$. At $t = 6$ the node v_2 fails again, hence the new order is given by $\pi_t = (v_1, v_3, v_4, v_5, v_2)$ for $t \in [6, 7)$, i.e., $v_{\pi_t(1)} = v_1, v_{\pi_t(2)} = v_3$ and so on. This is because the second node appears twice in s_1^6 and we consider only the last appearance. Following the same reasoning, $\pi_t = (v_3, v_4, v_5, v_2, v_1)$ for $t \in [7, 8)$ and $\pi_t = (v_3, v_4, v_2, v_1, v_5)$ for $t \in [8, 9)$.

We remark that if π_t is an associated permutation, then $v_{\pi_t(i)}$ denotes the node that appears in the i th location and $\pi_t^{-1}(i)$ denotes the location of the node v_i . In Example 2, we have $\pi_6(2) = 3$ since v_3 occupies the second position, and $\pi_6^{-1}(2) = 5$. Although π_t is a function from $[n]$ to $[n]$, for a node v_i we will often write $\pi^{-1}(v_i)$ to denote $\pi^{-1}(i)$. In Example 2, we have $\pi_6^{-1}(v_2) = 5$.

Now suppose that the evolution of the network is random and let t_{j_0} be the time by which all the nodes fail at least once. The next lemma shows that such j_0 exists almost surely and that the associated permutation $\pi_{t_{j_0}}$ is uniformly distributed on \mathfrak{S}_n .

Lemma 2 Let $(\mathbf{S}, T) = ((\mathbf{S}_i, T_i), i \geq 1)$ be an infinite sequence distributed according to $\mu_1 \times \mu_2$. Then almost surely, there exists a finite $t_0 \in \mathbb{R}$ such that all the nodes have failed at least once by t_0 . Moreover, π_{t_0} is distributed uniformly on \mathfrak{S}_n .

Proof: We denote by t_0 the first time instance when all the nodes have failed at least once and note that t_0 is a stopping time for the sequence (T_i) . Under our model, the failures of a node are independent of other nodes and defined as a Poisson arrival process. For each node v , the probability that the node has not failed up to time t is $e^{-\lambda t}$. Thus, we obtain

$$\Pr(t_0 \leq t) = \prod_{i=1}^n (1 - e^{-\lambda t}) = (1 - e^{-\lambda t})^n.$$

This proves the finiteness claim. The uniform distribution of π_{t_0} follows by symmetry. \blacksquare

Since $t_0 \leq \infty$ a.s., the time $t' = t - t_0$ is well defined. Consider a continuous-time Markov chain $X(t')$ with the state space \mathfrak{S}_n constructed as follows. Let $l \in [n]$ and let $\tau_l = (l, n, n-1, \dots, l+1)$ be a permutation (in cycle notation) that moves entry l to the last position, and shifts everything to the right of l one step to the left. Then $P(\pi \rightarrow \sigma) = \frac{1}{n}$ if and only if $\sigma = \tau_l \circ \pi$ for some l , and $P(\pi \rightarrow \sigma) = 0$ for all other pairs π, σ .

Let $N(t')$ be the number of nodes that failed until time t' . This is a Poisson counting process with rate $n\lambda$, i.e., $N(t') \sim \text{Poi}(n\lambda)$. At time $t' = 0$, $X(0)$ is chosen uniformly at random. For $t' \geq 0$ define $X(t') = \pi_{N(t')}$, where π with an integer index is defined above before Example 2. Due to the memoryless property of the exponential distribution, we obtain that $X(t')$ is indeed a Markov chain.

Next note that $X(t')$ is positive recurrent since the discrete-time chain on \mathfrak{S}_n defined by the kernel P is recurrent and the expected return time to a state in $X(t')$ is finite for any state in \mathfrak{S}_n . For a positive recurrent continuous-time Markov chain, the limiting probability distribution μ is unique, exists almost surely, and is given by

$$\mu(\pi) = \lim_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau \mathbb{1}_\pi(X(t')) dt' = \frac{1}{n\lambda \mathbb{E}[(\pi \rightarrow \pi)]} \quad (5)$$

where $(\pi \rightarrow \pi)$ is the time to return to state π starting from π (See, for example, [11, p. 332]). In our model, $\mathbb{E}[(\pi \rightarrow \pi)]$ does not depend on π . In words, (5) implies that for t large enough, the time that the network spends in each state is almost the same. We use this fact next to find an upper bound for the capacity.

Lemma 3 Let $(\mathcal{N}, \beta, \mathbf{S}, \mathbf{t}, h)$ be a storage network, where \mathbf{S} is a (random) sequence of failed nodes and h is a weight function satisfying the constraints given by β . Assume also that h_j is a function of the last failed node, i.e., if $\mathbf{S}_j = v_\ell$ then $h_j = h_{v_\ell}$. Then almost surely

$$\text{cap}(\mathcal{N}) \leq \frac{k'(2n - k' - 1)}{2} \frac{1}{n} \sum_{i=1}^n \beta_i. \quad (6)$$

Proof: The capacity of \mathcal{N} is equal to the minimum weight under h of a cut between \tilde{v} and DC_t where DC_t can connect to any set of k' nodes from \mathcal{A}_t . Assume that the set of weight functions is given by $\{h_v\}_{v \in V}$. Since there is a weight function for every node v , we will denote by $h_v(u)$ the weight that h_v assigns to the edge (u, CU) . Let t_{j_0} be the first time instance by which all the nodes have failed at least once. According to Lemma 2, t_{j_0} is almost surely finite. By (5) we may assume that all permutations appear as associated permutations with equal probability.

Let $D := \{v_{i_1}, \dots, v_{i_{k'}}\} \subset V$ and assume that the associated permutation π is such that $\pi^{-1}(v_{i_1}) \leq \pi^{-1}(v_{i_2}) \leq \dots \leq \pi^{-1}(v_{i_{k'}})$. Then the weight of the cut between \tilde{v} and D is at most [9]

$$C_t^h(D) \leq \sum_{\ell=1}^{k'} \left(\sum_{v \in V \setminus v_{i_\ell}} h_{v_{i_1}}(v) - \sum_{r=1}^{\ell-1} h_{v_{i_\ell}}(v_{i_r}) \right). \quad (7)$$

Since $\text{cap}(\mathcal{N})$ is the minimum weight value of a cut, we can bound it above by the average weight:

$$\text{cap}(\mathcal{N}) \leq \frac{1}{n! \binom{n}{k'}} \sum_{\substack{D \subseteq V \\ |D|=k'}} \sum_{\pi_t \in \mathfrak{S}_n} C_t^h(D), \quad (8)$$

For the moment let us fix D and consider how many times the term $h_v(u)$ appears on the right-hand side of (8) as we substitute $C_t^h(D)$ from (7) and evaluate the sum on π_t . If both $u, v \in D$ then this term appears for those π_t in which v appears after u and does not (is canceled in (7)) if v precedes u . Thus, overall this term appears $n!/2$ times. If $v \in D$ and $u \notin D$ then no cancellations occur, and the term $h_v(u)$ appears $n!$ times. Further, there are $\binom{n-2}{k'-2}$ choices of D for the first of these options and $\binom{n-2}{k'-1}$ for the second one of them. Thus, for each pair of nodes $u, v \in V$ the term $h_u(v)$ appears in (8)

$$\binom{n-2}{k'-2} \frac{n!}{2} + \binom{n-2}{k'-1} n! = \binom{n-2}{k'-2} \frac{n!}{2} \frac{2n-k'-1}{k'-1}$$

times. Substituting this into (8) and performing cancellations, we obtain that

$$\text{cap}(\mathcal{N}) \leq \frac{k'(2n-k'-1)}{2n(n-1)} \sum_{i=1}^n \sum_{j \neq i} h_{v_i}(v_j).$$

Since h is a weight function and since the nodes fail with equal probability, we obtain that $\sum_{i \neq j} h_{v_i}(v_j) = (n-1)\beta_j, j = 1, \dots, n$. Thus,

$$\sum_{i=1}^n \sum_{j \neq i} h_{v_i}(v_j) = (n-1) \sum_{j=1}^n \beta_j$$

and the result follows. ■

Remark 1 Lemma 3 holds also for h_j that is a function of the current associated permutation, i.e., $h_j = h_{\pi_{t_j}}$.

It is intuitively clear (and is confirmed by Lemmas 3 and 1) that if $\beta_i = \beta_0$ for every $i \in [n]$, the fact that \mathbf{S} is random does not affect the storage capacity, which implies that $\text{cap}(\mathcal{N})$ is equal to the minimum cut. Hence, in the case that $\beta_i = \beta_0$, both $\text{cap}(\mathcal{N})$ and its expected value are given in [9].

D. Discrete Time Evolution

In this subsection we define a discrete-time storage network, which will enable us to simplify the analysis of the average cut in the information flow graph and of network capacity. A discrete-time storage network is a network $(\mathcal{N}, \beta, s, \mathbf{t}, h)$ with $\mathbf{t} = (1, 2, \dots)$. For such a network with weight function h , the average cut is defined as

$$C_{\text{avg}}^h(\mathcal{N}) = \limsup_{l \rightarrow \infty} \frac{1}{l} \sum_{t=1}^l C_t^h(\mathcal{N}).$$

For a discrete-time network $(\mathcal{N}, \beta, s, \mathbf{t}, h)$ we will sometimes omit the time notion \mathbf{t} . Also, when a weight function is not specified we will omit the weight function notion and write (\mathcal{N}, β, s) . The following lemma

shows that for a random discrete-time storage network with $h = h^*$, the limit superior in this definition is almost surely a limit.

Lemma 4 *Let $(\mathcal{N}, \beta, \mathbf{S}, h^*)$ be a random discrete-time storage network with $\mathbf{S} = (\mathbf{S}_i, i \geq 1)$ a sequence of independent RVs uniformly distributed on $[n]$. Then*

$$\mathbb{E}[C_{\text{avg}}^{h^*}(\mathcal{N})] = \lim_{l \rightarrow \infty} \frac{1}{l} \sum_{t=1}^l \mathbb{E}[C_t^{h^*}(\mathcal{N})].$$

Proof: Let t_0 be the first time instance by which all the nodes have failed at least once. Note that t_0 is a stopping time and each failed node is chosen uniformly and independently. Referring to the Coupon collector's problem [12, p.210], we obtain that $\Pr(t_0 \geq cn \log n) \leq n^{1-c}$, for every $c \geq 1$. Thus, t_0 is finite almost surely.

By symmetry, π_{t_0} is distributed uniformly on the set of all permutations. Moreover, since \mathbf{S}_i is chosen uniformly and independently, for $t \geq t_0$ we have that $\Pr(\pi_t = \pi | \pi_{t-1}^{t-1}) = \Pr(\pi_t = \pi | \pi_{t-1})$, so the sequence $\{\pi_t\}$ is a Markov chain, which is irreducible and aperiodic. Because of this, a limiting distribution μ exists, and is unique and positive. Hence, as t grows, $\Pr(\pi_t) \rightarrow \mu(\pi_t)$. Together with the fact that $C_t^{h^*}$ is uniformly bounded from above for all t , we obtain that the limit $\lim_{l \rightarrow \infty} \frac{1}{l} \sum_{t=1}^l \mathbb{E}[C_t^{h^*}(\mathcal{N})]$ exists.

Now define $X_t = \frac{1}{t} \sum_{i=1}^t C_i^{h^*}(\mathcal{N})$ and note that X_t is a function of \mathbf{S} . Following the previous discussion, for almost every \mathbf{S} , the sequence X_t converges. Since X_t is non-negative and upper bounded for every t , by the dominated convergence theorem we have $\lim_{t \rightarrow \infty} \mathbb{E}[X_t] = \mathbb{E}[\lim_{t \rightarrow \infty} X_t]$ (the last limit exists a.s.), which is the desired result. ■

Since t_0 is almost surely finite and since π_t is an ergodic Markov chain, defining the initial state to be $\pi_0 = \text{id}$ does not affect the expected capacity. Hence, from now on we assume $\pi_0 = \text{id}$.

The problem of finding the limiting distribution of our Markov chain on \mathfrak{S}_n is similar to the classic question of the mixing time for the card shuffling problem called *Top in at random shuffle*. We use the following result from [3, Thm.1].

Theorem 1 (Aldous and Diaconis) *Consider a deck of n cards. At time $t = 1, 2, \dots$ take the top card and insert it in the deck at a random position. Let Q_t denote the distribution after t such shuffles and let U be the uniform distribution on the set of all permutations \mathfrak{S}_n . Then for all $c \geq 0$ and $n \geq 2$, the total variation distance satisfies*

$$\|Q_{n \log n + cn} - U\|_{TV} \leq e^{-c}. \quad (9)$$

To connect this result to our problem, we note that choosing the next failed node uniformly at random corresponds to selecting a random card from the deck and putting in at the bottom. The mixing time of this chain is stochastically equivalent to the mixing time of the *Top in at random shuffle*, and we obtain the following lemma.

Lemma 5 *Let \mathcal{N} be a storage network with $|V| = n \geq 2$ nodes and let \mathbf{S} be a random sequence of failed nodes. Consider the sequence of associated permutations $(\pi_t, t \geq 0)$ where $\pi_0 = \text{id}$. Then for any $c \geq 0$, $n \geq 2$ and any $\pi \in \mathfrak{S}_n$,*

$$\left| \Pr(\pi_{n \log n + cn} = \pi) - \frac{1}{n!} \right| \leq e^{-c}.$$

Proof: Let $T \geq 1$ be a time instant, and consider a realization $(\pi_t, t = 0, 2, \dots, T)$ of the Markov chain of failed nodes. Let $\tilde{\pi}_t, t \geq 0$ denote a realization of the card permutations in the top in at random shuffle. Then

$$\Pr(\pi_T = \tau | \pi_0 = \text{id}) = \Pr(\tilde{\pi}_T = \text{id} | \tilde{\pi}_0 = \tau)$$

for any $\tau \in \mathfrak{S}_n$. Taking $T = n \log n + cn, c \geq 0$ and using the definition of the total variation distance, we obtain the claimed result from (9). ■

The next lemma, whose proof is given in Appendix A, relates the values of average cut in the storage networks with continuous and discrete time.

Lemma 6 *Let $(\mathcal{N}_1, \beta, \mathbf{S}, t, h^*)$ be a continuous-time storage network and let $(\mathcal{N}_2, \beta, \mathbf{S}, h^*)$ be a discrete-time storage network. Then*

$$C_{\text{avg}}^{h^*}(\mathcal{N}_1) \stackrel{a.s.}{=} C_{\text{avg}}^{h^*}(\mathcal{N}_2).$$

As a result, we obtain the following statement which forms a basis of our subsequent derivations.

Theorem 2 *Let $(\mathcal{N}_1, \beta, \mathbf{S}, t)$ be a continuous-time storage network. Then $((\mu_1 \times \mu_2)\text{-a.s.})$*

$$\text{cap}(\mathcal{N}_1) \geq \frac{1}{n!} \sum_{\pi_t \in \mathfrak{S}_n} C_t^{h^*}(\pi_t).$$

Proof: From Lemma 1 we have that for any realization s such that every node fails infinitely often, $\text{cap}(\mathcal{N}_1) \geq C_{\text{avg}}^{h^*}(\mathcal{N}_1)$. According to Lemma 2, there exists a finite t_0 by which all the nodes have failed at least once and by (5) the stationary distribution of the permutations is uniform. This implies that almost surely, all the nodes fail infinitely often. According to Lemma 6, if $(\mathcal{N}_2, \beta, \mathbf{S}, h^*)$ is a discrete-time storage network, almost surely $C_{\text{avg}}^{h^*}(\mathcal{N}_1) = C_{\text{avg}}^{h^*}(\mathcal{N}_2)$. From Lemmas 4 and 5, $C_{\text{avg}}^{h^*}(\mathcal{N}_2)$ is almost surely a constant, which is equal to $\frac{1}{n!} \sum_{\pi_t \in \mathfrak{S}_n} C_t^{h^*}(\pi_t)$. Hence, almost surely, $C_{\text{avg}}^{h^*}(\mathcal{N}_2) = \mathbb{E}[C_{\text{avg}}^{h^*}(\mathcal{N}_2)] = \frac{1}{n!} \sum_{\pi_t \in \mathfrak{S}_n} C_t^{h^*}(\pi_t)$. Altogether these statements imply the claim of the theorem. ■

From this point, unless stated otherwise, we restrict ourselves to discrete-time networks.

III. The Fixed-Cost Model

In this section, we define the fixed-cost storage model and derive lower bounds on the storage capacity. Suppose that the set of nodes is $V = U \cup L$, where $U = (v_1, \dots, v_{n_1})$ and $L = (v_{n_1+1}, \dots, v_{n_1+n_2})$ are disjoint non-empty subsets. Suppose that the repair bandwidth of the node v_i is given by

$$\beta_i = \begin{cases} \beta_1 & \text{if } v_i \in U \\ \beta_2 & \text{if } v_i \in L \end{cases}.$$

where $\beta_1 \geq \beta_2 > 0$. Let C be the minimum cut of \mathcal{N} in the static case (i.e., the worst-case weight of the cut):

$$C \triangleq \min_{\pi \in \mathfrak{S}_n} \{C_t^{h^*}(\pi)\} = \min_{\substack{t \geq 0, \\ s \in V^\infty}} \{C_t^{h^*}\}. \quad (10)$$

Let $a \triangleq k - n_1$ and let us assume that $a > 0$ because otherwise the file reconstruction problem is trivially solved by contacting k nodes in U . The minimum cut is given by the following result from [2] (we cite it using our assumptions of $a > 0$ and large α).

Lemma 7 *Let $(\mathcal{N}, \beta, s, h^*)$ be a fixed-cost storage network. Then*

$$C = \frac{n_1(n_1 - 1)}{2} \beta_1 + \left(n_2(n_1 + a - 1) - \frac{a(a + 1)}{2} \right) \beta_2. \quad (11)$$

In this section we consider a dynamical equivalent of the above model, where the sequence of node failures \mathbf{S} is random. Note that if $n_2 = 1$ then $k = n$ which implies that no coding is used in the storage network, so we will assume that $n_2 \geq 2$. To avoid boundary cases, we will also assume that $n_1 > 1$ (the case of $n_1 = 1$ is not very interesting and can be handled using the same technique as below).

Expression (11) gives the size of the minimum cut in the static model of [9] and it also gives a lower bound for the cut $C_t^{h^*}$ for all t and s in the dynamical model. We shall now demonstrate by example that by controlling the transmission policy it is possible to increase the storage capacity of the $(\mathcal{N}, \beta, \mathbf{S}, h)$ network compared to (11).

The idea of the example is as follows. Assume that s_j is a failed node that needs to be recovered. Recall that in the static case, every active node transmits a fixed number of symbols for the recovery of s_j , namely, the nodes in U transmit β_1 symbols and the nodes in L transmit β_2 symbols. In the dynamic case we can choose how many symbols each node transmits for the recovery of s_j as long as the average constraint is satisfied. We change the number of symbols that node v_j transmits for the recovery of s_j depending on whether each of them is in U or L (the exact expressions are given in the example below). We then show that the average constraint is satisfied, and that this yields an increase of the capacity of the system.

Example 3 Let $(\mathcal{N}, \beta, \mathbf{S}, h)$ be a storage network with $n = 20$, $k' = 13$, $U = (v_1, \dots, v_{10})$, $L = (v_{11}, \dots, v_{20})$, and $\beta_1 = 2\beta_2$. Assume that α is large enough (in this case taking $\alpha \geq 33.5\beta_2$ suffices). By (11), the value of

the minimum cut with $h = h^*$ is $214\beta_2$, and thus the maximum file size that can be stored in the static case is $M = 214\beta_2$. The task of node repair is accomplished by contacting 19 nodes.

Now we will show that, under the dynamic model, it is possible to increase the file size by using the weight function h defined as follows. Suppose that at time t (recall that time is discrete) a node $v \in U$ has failed, i.e. $\mathbf{S}_t = v$ where $v \in U$, and define

$$h_t(v_i) = \begin{cases} \beta_2 & v_i \in L \\ \beta_1 + \frac{1}{20}\beta_2 & v_i \in U \setminus v \\ 0 & v_i = v. \end{cases}$$

If $\mathbf{S}_t = v$ where $v \in L$, define

$$h_t(v_i) = \begin{cases} \beta_2 & v_i \in L \setminus v \\ \beta_1 - \frac{9}{200}\beta_2 & v_i \in U \\ 0 & v_i = v. \end{cases}$$

A straightforward calculation of the minimum cut yields that

$$\min_{\pi \in \mathfrak{S}_{10}} \{C_t^h(\pi)\} = (214 + 2.25)\beta_2 \quad (12)$$

and it is obtained when $\pi = id$ and the active nodes selected are $D_t = (v_1, v_2, \dots, v_{13})$. This shows an increase over the static case estimate (11).

We now calculate the expected number of symbols a node transmits under h . Recall that in the random model, each node has the same probability of failure which in this case equals to $\frac{1}{20}$. Let t_0 denote the first time instance by which all the nodes have failed. For every $t \geq t_0$ we have that if $v_i \in U$ then

$$\mathbb{E}[h_t(v_i)] = \frac{9}{20}(\beta_1 + \frac{1}{20}\beta_2) + \frac{10}{20}(\beta_1 - \frac{9}{200}\beta_2) < \beta_1$$

and if $v_i \in L$ then

$$\mathbb{E}[h_t(v_i)] = \frac{9}{20}\beta_2 + \frac{10}{20}\beta_2 < \beta_2.$$

Therefore, the average amount of symbols each node transmits satisfies the constraints given by β .

The above simple procedure is not optimal in terms of the file size M : As we show below, it is possible to construct a different transmission scheme which allows for storage of a larger-size file. Note also that the upper bound (6) gives $\text{cap}(\mathcal{N}) \leq 235.5\beta_2$, while the improvement of (12) over (11) is relatively minor.

Example 3 provides a procedure to construct the weight function h such that the maximum file size can be increased. Below we generalize this idea and also explore other ways of using time evolution to increase the storage capacity of a fixed-cost network

A. A protocol to increase capacity

In this section we construct a weight function that increases the storage capacity and analyze the increase. The next theorem states the increase explicitly. In order to state the theorem, we need the following assumptions. For $\epsilon_1 > 0$, assume that

$$\beta_1 - \beta_2 \geq \frac{n(n_1 - 1)}{n_2} \epsilon_1. \quad (13)$$

Note that $\frac{n(n_1 - 1)}{n_2} \geq 1$ and that this assumption is satisfied in Example 3 above.

We now prove the following theorem which quantifies the increase of the average storage capacity over the static case.

Theorem 3 *Let $(\mathcal{N}, \beta, \mathbf{S})$ be a fixed-cost storage network. For any $\epsilon_1 \geq 0$ such that assumption (13) is satisfied, the storage capacity is bounded below by*

$$\text{cap}(\mathcal{N}) \stackrel{\text{a.s.}}{\geq} C + \frac{n_1(n_1 - 1)}{2} \epsilon_1$$

where C is the static storage capacity given in (11).

To prove the theorem we define a weight function along the lines of Example 3. Let us put $h_t(v_i) = h_U(v_i)$ if $s_t \in U$ and $h_t(v_i) = h_L(v_i)$ if $s_t \in L$ where

$$h_U(v_i) = \begin{cases} \beta_1 + \varepsilon_1 & v_i \in U \setminus s_j \\ \beta_2 & v_i \in L \\ 0 & v_i = s_j, \end{cases} \quad (14)$$

and

$$h_L(v_i) = \begin{cases} \beta_1 - \frac{n_1-1}{n_2}\varepsilon_1 & v_i \in U \\ \beta_2 & v_i \in L \setminus s_j \\ 0 & v_i = s_j \end{cases} \quad (15)$$

and $0 \leq \varepsilon_1 \leq \beta_1$.

We now show that the weight function h satisfies the constraints given by β .

Lemma 8 *Let $(\mathcal{N}, \beta, \mathbf{S}, h)$ be a fixed-cost storage network with h as defined above. Then h satisfies the average constraints given by β .*

Proof: Fix a node $v_i \in U$ and for each time instance t , let us calculate the expected number of symbols v_i transmits. Recall that $v_{\pi_t(n)}$ denotes the node that failed at time t . Recall that the failures of the nodes are uniformly distributed, so we obtain

$$\Pr(v_{\pi_t(n)} = v_j) = \begin{cases} \frac{1}{n} & \text{if } j = i \\ \frac{n_1-1}{n} & \text{if } j \in [n_1] \setminus i \\ \frac{n_2}{n} & \text{otherwise.} \end{cases}$$

Hence, the expected number of symbols that the node v_i transmits is

$$\frac{n_1-1}{n} h_U(v_i) + \frac{n_2}{n} h_L(v_i) = \frac{n_1-1}{n} (\beta_1 + \varepsilon_1) + \frac{n_2}{n} \left(\beta_1 - \frac{n_1-1}{n_2} \varepsilon_1 \right) < \beta_1.$$

If $v_i \in L$ we have

$$\Pr(v_{\pi_t(n)} = v_j) = \begin{cases} \frac{1}{n} & \text{if } j = i \\ \frac{n_1}{n} & \text{if } j \in [n_1] \\ \frac{n_2-1}{n} & \text{otherwise.} \end{cases}$$

In this case the expected number of transmitted symbols equals

$$\frac{n_1}{n} h_U(v_i) + \frac{n_2-1}{n} h_L(v_i) = \frac{n_1}{n} \beta_2 + \frac{n_2-1}{n} \beta_2 < \beta_2.$$

Thus, on average the number of symbols is within the allotted bandwidth. ■

The next two lemmas are used in the proof of Theorem 3 in order to estimate the minimum cut. The first lemma shows that the minimum cut for any permutation π_t , $t \geq t_0$ is obtained when $D_t \supseteq U$. The second lemma shows that the minimum cut is obtained for $\pi_t = id$.

Lemma 9 *Let $(\mathcal{N}, \beta, s, h)$ be a network with h as defined above. If assumption (13) is satisfied, then for $t > t_0$, the value $C_t^h(\mathcal{N})$ is attained when $D_t \supseteq U$.*

Proof: We formulate our question as a dynamic programming problem and provide an optimal policy for node selection. Assume that π_t is a fixed permutation that represents the order of the last n failed nodes. We will consider the information flow graph \mathcal{X}_t and show that the cut is minimized when all the nodes from U are selected.

Consider a k' -step procedure which in each step selects one node from \mathcal{A}_t . Each step entails a cost as explained next. Let $t' \leq t$ and assume that node $v_{t'}^{t'} \in \mathcal{A}_t$ was selected. The cost is defined as the added weight values of the in-edges of $CU_{t'}$ that are not out-edges of previously selected nodes. Our goal is to choose k' nodes that minimize the total cost and hence minimize the cut between $\left(\bigcup_{j=-1}^{t-1} \mathcal{A}_j \right) \setminus \mathcal{A}_t$ and DC_t .

In order to simplify notation, we write $\pi_t = (u_1, u_2, \dots, u_n)$, i.e., $u_l = v_{\pi_t(l)}$ is the storage node that appears in the l th position in π_t . Moreover, with a slight abuse of notation, if u_j failed at time t' we will write $h_j(u_i)$

instead of $h_{t'}(u_i)$ to denote the number of symbols that node u_i transmits for the recovery of u_j . For $\kappa \leq k'$ consider the sub-problem in step $\kappa - 1$, where the DC_t has already chosen $\kappa - 1$ nodes $(u_{i_1}, \dots, u_{i_{\kappa-1}})$ and we are to choose the κ th node. Assume that the chosen nodes are ordered according to their appearance in the permutation, i.e., $i_1 \leq i_2 \leq \dots \leq i_{\kappa-1}$. Let $u_{j_1}, \dots, u_{j_m} \in U$ be nodes that were not selected up to step $\kappa - 1$, i.e.,

$$\{u_{j_1}, \dots, u_{j_m}\} \cap \{u_{i_1}, \dots, u_{i_{\kappa-1}}\} = \emptyset,$$

and assume also that $j_1 \leq j_2 \leq \dots \leq j_m$. We refer to Figure 2 for an illustration. We show that choosing u_{j_1} accounts for the minimum cut. First, we claim that choosing u_{j_1} minimizes the cut over all other nodes from U . Denote by $C_{\kappa-1}$ the total cost (or the cut) in step $\kappa - 1$. Fix $2 \leq \ell \in [m]$ and note that since $j_1 \leq j_\ell$, we can write

$$i_1 \leq \dots \leq i_{r_1} \leq j_1 \leq i_{r_1+1} \leq \dots \leq i_{r_\ell} \leq j_\ell \leq i_{r_\ell+1} \leq \dots,$$

where the set of indices $\{i_1, \dots, i_r\}$ can be empty. Let $C(j_1)$ be the value of the cut once we add u_{j_1} in the κ th step. The change from $C_{\kappa-1}$ is formed of the following components. First, we add the values of all the edges from $U \setminus \{u_{j_1}\}$ to u_{j_1} and from L to u_{j_1} , accounting for $(n_1 - 1)(\beta_1 + \varepsilon_1) + n_2\beta_2$ symbols. Further, we remove the values of all the edges from the nodes $u_{i_1}, \dots, u_{i_{r_1}}$ to u_{j_1} and all the edges from u_{j_1} to $u_{i_{r_1+1}}, \dots, u_{i_{r_\ell}}$. Overall we obtain

$$C(j_1) = C_{\kappa-1} + (n_1 - 1)(\beta_1 + \varepsilon_1) + n_2\beta_2 - \sum_{q=1}^{r_1} h_{j_1}(u_{i_q}) - \sum_{q=r_1+1}^{\kappa-1} h_{i_q}(u_{j_1}). \quad (16)$$

Similarly, let $C(j_\ell)$ be the value of C_κ if in step κ we select the node u_{j_ℓ} , $\ell \geq 2$. Following the same argument as in (16), we obtain

$$C(j_\ell) = C_{\kappa-1} + (n_1 - 1)(\beta_1 + \varepsilon_1) + n_2\beta_2 - \sum_{q=1}^{r_\ell} h_{j_\ell}(u_{i_q}) - \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}(u_{j_\ell}).$$

Since $h_{j_\ell}(u_i) = h_{j_1}(u_i)$ and $h_i(u_{j_1}) = h_i(u_{j_\ell})$ for all $i \in [n]$, we have

$$C(j_1) - C(j_\ell) = \sum_{q=r_1+1}^{r_\ell} (h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1})).$$

For $u_{i_q} \in U$, we obtain

$$h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1}) = \beta_1 + \varepsilon_1 - (\beta_1 + \varepsilon_1) = 0.$$

For $u_{i_q} \in L$, we obtain

$$h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1}) = \beta_2 - \left(\beta_1 - \frac{n_1 - 1}{n_2} \varepsilon_1 \right)$$

which is nonpositive by assumption (13). Therefore,

$$C(j_1) - C(j_\ell) \leq 0.$$

Now we show that u_{j_1} minimizes the cut over a selection of any node u_{j_ℓ} from L . We divide the argument into 2 cases:

- 1) Assume that $j_\ell < j_1$. Denote by $(i_1, \dots, i_{r_\ell}, j_\ell, i_{r_\ell+1}, \dots, i_{r_1}, j_1, \dots)$ the indices of the selected nodes and let $C(j_\ell), C(j_1)$ be the cut values if we choose u_{j_ℓ}, u_{j_1} , respectively. We have

$$C(j_\ell) = C_{\kappa-1} + n_1 \left(\beta_1 - \frac{n_1 - 1}{n_2} \varepsilon_1 \right) + (n_2 - 1)\beta_2 - \sum_{q=1}^{r_\ell} h_{j_\ell}(u_{i_q}) - \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}(u_{j_\ell}).$$

On account of (16) and (13) we now obtain

$$\begin{aligned}
C(j_1) - C(j_\ell) &= -(\beta_1 - \beta_2) + \frac{n(n_1 - 1)}{n_2} \varepsilon_1 + \sum_{q=1}^{r_\ell} h_{j_\ell}(u_{i_q}) - \sum_{q=1}^{r_1} h_{j_1}(u_{i_q}) \\
&\quad + \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}(u_{j_\ell}) - \sum_{q=r_1+1}^{\kappa-1} h_{i_q}(u_{j_1}) \\
&\leq \sum_{q=1}^{r_\ell} (h_{j_\ell}(u_{i_q}) - h_{j_1}(u_{i_q})) - \sum_{q=r_\ell+1}^{r_1} h_{j_1}(u_{i_q}) + \sum_{q=r_\ell+1}^{r_1} h_{i_q}(u_{j_\ell}) + \sum_{q=r_1+1}^{\kappa-1} (h_{i_q}(u_{j_\ell}) - h_{i_q}(u_{j_1})) \quad (17)
\end{aligned}$$

Our goal is to show that the right-hand side of (17) is nonpositive. Let $1 \leq q \leq r_\ell$. For $u_{i_q} \in U$ we have

$$h_{j_\ell}(u_{i_q}) - h_{j_1}(u_{i_q}) = \beta_1 - \frac{n_1 - 1}{n_2} \varepsilon_1 - (\beta_1 + \varepsilon_1) \leq 0$$

and for $u_{i_q} \in L$ we have

$$h_{j_\ell}(u_{i_q}) - h_{j_1}(u_{i_q}) = \beta_2 - \beta_2 = 0.$$

Now let $r_{\ell+1} \leq q \leq \kappa - 1$. For $u_{i_q} \in U$ we have

$$h_{i_q}(u_{j_\ell}) - h_{i_q}(u_{j_1}) = \beta_2 - (\beta_1 + \varepsilon_1)$$

and for $u_{i_q} \in L$ we have

$$h_{i_q}(u_{j_\ell}) - h_{i_q}(u_{j_1}) = \beta_2 - (\beta_1 - \frac{n_1 - 1}{n_2} \varepsilon_1),$$

both of which are non-positive by assumption (13).

The remaining terms in (17) contribute $\sum_{q=r_\ell+1}^{r_1} (h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q}))$ to the value of the cut. As before, for $u_{i_q} \in U$ we have

$$h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q}) = \beta_2 - (\beta_1 + \varepsilon_1) \leq 0$$

by (13), and for $u_{i_q} \in L$ we have

$$h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q}) = \beta_2 - \beta_2 = 0.$$

Thus, $C(j_1) - C(j_\ell) \leq 0$.

2) Assume that $j_\ell > j_1$. This case is symmetric to the case $j_\ell < j_1$ and the analysis is similar.

By the principle of optimality in dynamic programming, which states that every optimal policy consists only of optimal sub-policies [6, Ch. 1.3], we now conclude that the minimum cut is formed by first taking all the nodes from U and then take the remaining nodes from L . ■

Remark 2 Suppose that in forming the cut, we have added all the nodes from U , and there are a more nodes (from L) to select. To minimize the value of the cut, these nodes should be taken to be the a most recently failed nodes from L . This is because choosing the most recently failed node $v_{\pi(n)}$ assures that as few as possible of the previously selected nodes contain information from $v_{\pi(n)}$.

To justify this formally, consider the proof of Lemma 9. Indeed, if $u_{j_1}, u_{j_\ell} \in L$ with $j_1 < j_\ell$, then

$$C(j_1) - C(j_\ell) = \sum_{q=r_1+1}^{r_\ell} h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1})$$

which is non-negative by assumption (13).

Before stating the second lemma, we need the following notation. Let $\pi \in \mathfrak{S}_n$ and let $D \subset V, |D| = k'$. For a node $v_j \in D$, denote by $f_\pi(v_j)$ the number of nodes in $D \cap L$ that appear before v_j in π , i.e.,

$$f_\pi(v) := \sum_{i=1}^n \mathbb{1}_{D \cap L}(v_i) \cdot \mathbb{1}_{[1, \pi^{-1}(v_j)]}(\pi^{-1}(v_i)).$$

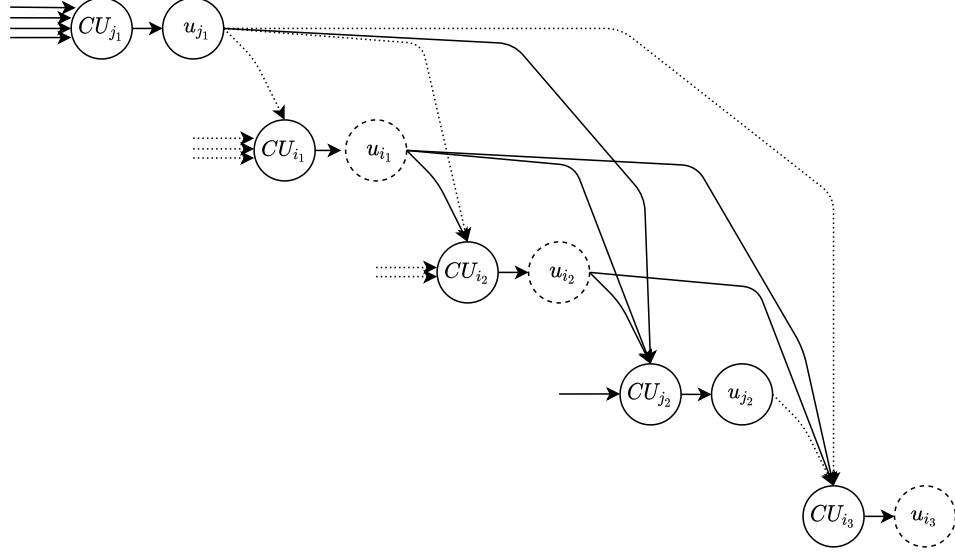


Figure 2. An illustration of the procedure in the proof of Lemma 9 with $\kappa = 4$, where the 3 dashed nodes were selected by the DC_t , and we are to select the next node out of u_{j_1}, u_{j_2} . The current cost of the procedure equals the sum of the weights of the dotted edges. Suppose that node u_{j_2} is selected next. Then the weight of the edge from u_{j_1} to CU_{j_2} (which equals $h_{j_1}(u_{j_2})$) and the weight of the solid in-edge of CU_{j_2} will be added to the cost, while the weight of the edge from u_{j_2} to CU_{i_3} (which equals $h_{j_2}(u_{i_3})$) will be deducted. Otherwise, if node u_{i_1} is selected, the weights of the solid in-edges of CU_{i_1} will be added and the weights of all the dotted out-edges of u_{j_1} will be deducted.

Let $T_j, j = 1, \dots, n-1$ be an *adjacent transposition* of π , i.e., $T_j \circ \pi$ exchanges $\pi(j)$ and $\pi(j+1)$.

Lemma 10 Let $(\mathcal{N}, \beta, \mathbf{S}, h)$ be a fixed-cost storage network with h as defined above. Let π_t be a permutation obtained at time $t > t_0$ and let D_t be a set of k' active nodes selected by the DC_t . If assumption (13) is satisfied, then

$$C_t^h(\pi_t) = C_t^h(id) + \sum_{v \in D_t \cap U} f_{\pi_t}(v)(\beta_1 - \beta_2) - \sum_{v \in D_t \cap U} f_{\pi_t}(v) \frac{n_1 - 1}{n_2} \varepsilon_1,$$

where $C_t^h(id)$ is the minimum cut for $\pi_t = id$.

Proof: Recall that by assumption (13), $\beta_1 - \beta_2 - \frac{n_1 - 1}{n_2} \varepsilon_1 \geq 0$. We start with showing that for every permutation $\pi_t \in \mathfrak{S}_n$ and for any $j \in [n]$,

$$C_t(T_j(\pi_t)) \in \left\{ C_t(\pi_t) + (\beta_1 - \beta_2) + \frac{n_1 - 1}{n_2} \varepsilon_1, C_t(\pi_t), C_t(\pi_t) - (\beta_1 - \beta_2) - \frac{n_1 - 1}{n_2} \varepsilon_1 \right\}.$$

First, observe that if π_t and σ_t are two permutations such that

$$\{\pi_t^{-1}(v_i) : i \in [n_1]\} = \{\sigma_t^{-1}(v_i) : i \in [n_1]\},$$

i.e., if the nodes from U occupy the same positions in π_t as in σ_t , then $C_t(\pi_t) = C_t(\sigma_t)$.

Let $\pi_t \in \mathfrak{S}_n$ and let D_t denote the k' storage nodes selected. Assume that $v_{\pi_t(j)} \in U$, $v_{\pi_t(j+1)} \in L$ and that $\{v_{\pi_t(j)}, v_{\pi_t(j+1)}\} \subseteq D_t$ for some $j \in [n-1]$. It is easy to see that

$$C_t^h(T_j(\pi_t)) = C_t^h(\pi_t) + (\beta_1 - \beta_2) - \frac{n_1 - 1}{n_2} \varepsilon_1.$$

On the other hand, if $v_{\pi_t(j)} \in L$, $v_{\pi_t(j+1)} \in U$ and $\{v_{\pi_t(j)}, v_{\pi_t(j+1)}\} \subseteq D_t$, then

$$C_t^h(T_j(\pi_t)) = C_t^h(\pi_t) - (\beta_1 - \beta_2) + \frac{n_1 - 1}{n_2} \varepsilon_1.$$

Recall that according to Lemma 9, the set D_t which yields the minimum cut contains U . Hence, for id , the minimum cut is given by $C_t(id)$ and is obtained by selecting the first k' nodes, $D_t = \{v_i : i \in [k']\}$.

Moreover, every permutation $\pi_t \in \mathfrak{S}_n$ can be obtained from id by repeated applications of T , such that at each application, the size of the minimum cut is not decreased. ■

Note that Lemma 7 is an immediate corollary of Lemmas 9 and 10. Indeed, taking the weight function $h = h^*$ implies that $\varepsilon_1 = 0$ which satisfies assumption (13). Hence, the minimum cut is obtained when $\pi_t = id$ and $D_t \supseteq U$, and is equal to C .

Let us prove Theorem 3.

Proof of Theorem 3: From Lemma 9 we obtain that there exists $\varepsilon_1 > 0$ such that assumption (13) is satisfied and such that at each time t , the selection D_t that minimizes the cut, contains U . Lemma 10 implies that the minimum cut is obtained for $\pi_t = id$. Taking $\pi_t = id$ and $D_t = \{v_1, \dots, v_{k'}\}$, it is straightforward to check that

$$C_t^h(D_t) = \sum_{j=1}^{n_1-1} j(\beta_1 + \varepsilon_1) + n_1 n_2 \beta_2 + \sum_{j=1}^a (n_2 - j) \beta_2 = C_t^{h^*}(\pi_t) + \frac{n_1(n_1 - 1)}{2} \varepsilon_1 \quad (18)$$

which together with Theorem 2 concludes the proof. ■

Remark 3 The function h can be defined with an additional parameter $0 \leq \varepsilon_2 \leq \beta_2$ such that when a node $v_i \in U$ ($v_i \in L$) fails, nodes from L transmit $\beta_2 - \varepsilon_2$ (resp., $\beta_2 + \varepsilon_2$) symbols instead of β_2 symbols. This change may increase the storage capacity even more, but requires additional assumptions on the parameters and can be developed along the same ideas.

In conclusion, we have shown that the maximum file size that can be stored in a dynamical fixed-cost storage network is always greater than its static counterpart. While it is always possible to choose ε_1 so that (13) holds true (e.g., $\varepsilon_1 = \frac{n_2}{n(n_1-1)}(\beta_1 - \beta_2)$), the capacity increase is relatively small because the allowable values of ε_1 are small as a proportion of $\beta_1 - \beta_2$. In the next section we take an alternative approach to bounding the average capacity.

B. The average min-cut bound on cap

We consider the same storage model as in the previous section and prove the following result.

Theorem 4 Let $(\mathcal{N}, \beta, \mathbf{S}, h^*)$ be a fixed-cost storage network. Then almost surely,

$$\text{cap}(\mathcal{N}) \geq C + \frac{\beta_1 - \beta_2}{2} \frac{a n_1}{n} (a + 1 + \frac{n_1 - 1}{n - 1} (a - 1)). \quad (19)$$

We will need the following two lemmas.

Lemma 11 Let $(\mathcal{N}, \beta, \mathbf{S})$ be a storage network, let $0 \leq \ell \leq \min(n_1, a)$ and denote by P_t^ℓ the probability that π_t contains ℓ nodes from U in the last $a = k' - n_1$ positions. As $t \rightarrow \infty$,

$$P_t^\ell \rightarrow \binom{n_1}{\ell} \binom{n_2}{a - \ell} \binom{n}{a}^{-1}.$$

Proof: Assume that π_t is distributed uniformly over \mathfrak{S}_n . We have $P_t^\ell = \frac{\binom{n_1}{\ell} \binom{n_2}{a - \ell} \binom{n}{a}^{-1}}{\binom{n}{a}}$. By Lemma 5, the distribution of π_t converges to the uniform distribution exponentially fast (after a certain time, the TV distance decreases by a factor of $1/e$ every n time units). By the definition of the total variation distance, for every ℓ

$$\left| P_t^\ell - \binom{n_1}{\ell} \binom{n_2}{a - \ell} \binom{n}{a}^{-1} \right| \leq e^{\log n - \frac{t}{n}}$$

which implies the lemma. ■

For the next lemma we need the following notation. Let \mathfrak{S}_n^ℓ be the set of all permutations over $[n]$ with exactly ℓ numbers from U in the last a positions, i.e.,

$$\mathfrak{S}_n^\ell \triangleq \{ \pi \in \mathfrak{S}_n : |\{ \pi(n - a + 1), \dots, \pi(n) \} \cap [n_1]| = \ell \}.$$

Given $\pi = (i_1, \dots, i_{n-a}, i_{n-a+1}, \dots, i_n) \in \mathfrak{S}_n$, let $\pi^c := (i_1, \dots, i_{n-a}, i_n, \dots, i_{n-a+1})$.

Lemma 12 Let $(\mathcal{N}, \beta, \mathbf{S}, h^*)$ be a fixed-cost storage network. Let π_t be the permutation at time t and for every ℓ , define $\mu_\ell(\pi_t) := \Pr(\pi_t | \mathfrak{S}_n^\ell)$. Then,

$$\lim_{t \rightarrow \infty} \mathbb{E}_{\mu_\ell}[C_t^{h^*}(\mathcal{N})] \geq C + \frac{1}{2}\ell(a + \ell)(\beta_1 - \beta_2)$$

where C is given in Lemma 7.

Proof: As above, let t_0 be time by which all the nodes have failed at least once, and recall that $P(t_0 < \infty) = 1$. Therefore, π_t (and hence, μ_ℓ) is well defined almost surely. From Lemma 5, we obtain that for every $\epsilon > 0$, there exists $t_\epsilon > t_0$ large enough such that $\left| \mu_\ell(\pi_t) - \frac{1}{|\mathfrak{S}_n^\ell|} \right| \leq \epsilon$ and therefore the limit exists almost surely.

For $t > t_\epsilon$ consider

$$\sum_{\pi_t \in \mathfrak{S}_n^\ell} \Pr(\pi_t | \mathfrak{S}_n^\ell) C_t(\pi_t) \geq \left(\frac{1}{|\mathfrak{S}_n^\ell| - \epsilon} \right) \sum_{\pi_t \in \mathfrak{S}_n^\ell} C_t(\pi_t) \geq \frac{1}{|\mathfrak{S}_n^\ell|} \sum_{\pi_t \in \mathfrak{S}_n^\ell} C_t(\pi_t) - \epsilon R,$$

where $R = \max_{\pi_t \in \mathfrak{S}_n} C_t(\pi_t)$. To bound this sum below we fix the last a entries of the permutation. Since for $h = h^*$ (i.e., $\varepsilon_1 = 0$), assumption (13) is satisfied. Thus we can use Lemma 9, according to which $C_t(\pi_t)$ is minimized if $n_1 - \ell$ entries from U appear in the first $n_1 - \ell$ positions, followed by $n_2 - a + \ell$ entries from L (in any order). Fix the first $n - a$ entries. Again according to Lemma 9, the minimum cut will be obtained when all the ℓ nodes from U are in positions $n - a + 1, n - a + 2, \dots, n - a + \ell$, and according to Lemma 10 it is equal to $C_{\min} := C + \ell^2(\beta_1 - \beta_2)$. Also, the maximum cut will be obtained when all the ℓ nodes from U are located in the last positions. This yields $C_{\max} := C + \ell a(\beta_1 - \beta_2)$.

Let $\pi_t \in \mathfrak{S}_n^\ell$ be any permutation with $v_{\pi_t(i)} \in U$ for $i \in \{1, \dots, n_1 - \ell\}$. We claim that

$$C_t(\pi_t) + C_t(\pi_t^c) = 2C + \ell(a + \ell)(\beta_1 - \beta_2) = C_{\min} + C_{\max}. \quad (20)$$

Indeed, assume $\pi_t = \pi$ and let D be a selection of k active nodes that minimizes the cut. By Lemma 9 if there is at least one node from U in the last a places, the minimum cut will be obtained by selecting the last a places as a part of D . Moreover, if $v_i \in U$ with $\pi^{-1}(v_i) = n - a + m$ for some $m \in [a]$, and $f_\pi(v_i) = b$ then $|\{v_{\pi(1)}, \dots, v_{\pi(n-a+m)}\} \cap (D \cap L)| = b$. Together with the fact that $|D \cap L| = a$, this implies that $|\{v_{\pi(n-a+1)}, \dots, v_{\pi(n-a+m)}\} \cap (D \cap L)| = b - \ell$. For π^c , we obtain that $(\pi^c)^{-1}(v_i) = n - m + 1$ and $|\{v_{\pi^c(n-m+1)}, \dots, v_{\pi^c(n)}\} \cap L| = b - \ell$ which means that $|\{v_{\pi^c(1)}, \dots, v_{\pi^c(n-m+1)}\} \cap (D \cap L)| = a - (b - \ell)$.

By Lemma 10 we have

$$C_t(\pi) = C + \sum_{v \in D \cap U} f_\pi(v)(\beta_1 - \beta_2) \geq C + \sum_{\substack{v \in D \cap U \\ \pi^{-1}(v) \in \{n-a+1, \dots, n\}}} f_\pi(v)(\beta_1 - \beta_2).$$

For π^c we obtain

$$\begin{aligned} C_t(\pi^c) &\geq C + \sum_{\substack{v \in D \cap U \\ (\pi^c)^{-1}(v) \in \{n-a+1, \dots, n\}}} f_{\pi^c}(v)(\beta_1 - \beta_2) \\ &= C + \sum_{\substack{v \in D \cap U \\ (\pi^c)^{-1}(v) \in \{n-a+1, \dots, n\}}} (a - (f_\pi(v) - \ell))(\beta_1 - \beta_2). \end{aligned}$$

This implies that

$$C_t(\pi) + C_t(\pi^c) \geq 2C + \ell(a + \ell)(\beta_1 - \beta_2).$$

Note that for every $\pi_t \in \mathfrak{S}_n^\ell$, the permutation $\pi_t^c \in \mathfrak{S}_n^\ell$ and that $(\pi_t^c)^c = \pi_t$. Thus, for every $\epsilon > 0$, there exists $t_\epsilon > t_0$ such that for every $t > t_\epsilon$

$$\sum_{\pi_t \in \mathfrak{S}_n^\ell} \mu_\ell(\pi_t) C_t(\pi) \geq \frac{1}{|\mathfrak{S}_n^\ell|} \frac{1}{2} \sum_{\pi_t \in \mathfrak{S}_n^\ell} C_t(\pi_t) + C_t(\pi_t^c) - \epsilon R \geq C + \frac{1}{2}\ell(a + \ell)(\beta_1 - \beta_2) - \epsilon R,$$

which concludes the proof. ■

We can now complete the proof of Theorem 4.

Proof of Theorem 4: From Lemma 4 and Lemma 5, we have

$$C_{\text{avg}}^{h^*}(\mathcal{N}) \stackrel{\text{a.s.}}{=} \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{r=1}^t \mathbb{E}[C_r(\mathcal{N})] \stackrel{\text{a.s.}}{=} \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{r=t_0}^t \sum_{\pi \in \mathfrak{S}_n} \Pr(\pi_r = \pi) C_r(\pi).$$

Observe that $(\mathfrak{S}_n^\ell)_\ell$ partitions the set \mathfrak{S}_n , and we can continue as follows:

$$\begin{aligned} C_{\text{avg}}^{h^*}(\mathcal{N}) &\stackrel{\text{a.s.}}{=} \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{r=t_0}^t \sum_{\ell=0}^{\min\{a, n_1\}} \sum_{\pi \in \mathfrak{S}_n^\ell} \Pr(\pi_r = \pi | \mathfrak{S}_n^\ell) \Pr(\pi_r \in \mathfrak{S}_n^\ell) C_r(\pi) \\ &= \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{r=t_0}^t \sum_{\ell=0}^{\min\{a, n_1\}} \Pr(\pi_r \in \mathfrak{S}_n^\ell) \mathbb{E}_{\mu_\ell}[C_r(\mathcal{N})] \end{aligned}$$

By Lemma 11, for every $\epsilon > 0$, there is $t_\epsilon > t_0$ such that $|\Pr(\pi_r \in \mathfrak{S}_n^\ell) - \frac{\binom{n_1}{\ell} \binom{n_2}{a-\ell}}{\binom{n}{a}}| \leq \epsilon$. Hence, for every $\epsilon > 0$,

$$C_{\text{avg}}^{h^*}(\mathcal{N}) \stackrel{\text{a.s.}}{\geq} \lim_{t \rightarrow \infty} \frac{1}{t} \left(\sum_{r=t_\epsilon}^t \sum_{\ell=0}^{\min\{a, n_1\}} \frac{\binom{n_1}{\ell} \binom{n_2}{a-\ell}}{\binom{n}{a}} \mathbb{E}_{\mu_\ell}[C_r(\mathcal{N})] - \sum_{r=t_0}^{t_\epsilon} n_1 R \right),$$

where $R = \max_{\pi_t \in \mathfrak{S}_n} C_t(\pi_t)$. Together with Lemma 12 this yields

$$C_{\text{avg}}^{h^*}(\mathcal{N}) \stackrel{\text{a.s.}}{\geq} C + \sum_{\ell=0}^{n_1} \frac{\binom{n_1}{\ell} \binom{n_2}{a-\ell}}{\binom{n}{a}} \frac{\ell(a+\ell)(\beta_1 - \beta_2)}{2}. \quad (21)$$

By Lemma 1, the right-hand side of this inequality gives a lower bound on capacity. It can be transformed to the expression on the right-hand side of (19) by repeated application of the Vandermonde convolution formula. \blacksquare

Thus, we have proved that the average minimum cut (and thus, the capacity) is almost surely bounded below by an expression which is strictly greater than C , and accounting for the dynamics of the fixed-cost network enables one to support storage of a larger file than in the static case of [2].

To summarize the results of this section, we have proved that

$$\text{cap}(\mathcal{N}) - C \stackrel{\text{a.s.}}{\geq} \max \left\{ \frac{n_1(n_1-1)}{2} \epsilon_1, \frac{\beta_1 - \beta_2}{2} \frac{a n_1}{n} \left(a + 1 + \frac{n_1 - 1}{n - 1} (a - 1) \right) \right\}, \quad (22)$$

where the first of the bounds on the right is valid under assumption (13). To give numerical examples, let us return to Example 3. Applying Theorem 4 to Example 3 yields $\text{cap}(\mathcal{N}) \geq 214\beta_2 + 3.7\beta_2$. At the same time, Theorem 3 states that the storage capacity is bounded below by $214\beta_2 + \frac{9}{4}\beta_2$, showing that the choice of h is not always optimal. Generally, the lower bound on capacity of Theorem 3 is $C + \frac{n_1 n_2}{2n} (\beta_1 - \beta_2)$ and the bound of Theorem 4 is approximately $C + \frac{n_1 a^2}{2n}$. Therefore, Theorem 4 provides a better bound on the storage capacity when a is roughly above $\sqrt{n_2}$.

Since the storage capacity can be increased while the average amount of symbols each node v_i transmits is at most β_i , after a long period of time (for large enough t), the total bandwidth that was used for repair in the dynamical model is equal to the total bandwidth that was used for repair in the static model.

To conclude this section, we address the question regarding the accuracy of the derived bounds on $\mathbb{E}[C_{\text{avg}}^{h^*}(\mathcal{N})]$. In the next proposition we derive an upper bound on this quantity.

Proposition 1 *Let $(\mathcal{N}, \beta, \mathbf{S}, h^*)$ be a storage network. We have (μ_1 -a.s.)*

$$C_{\text{avg}}^{h^*}(\mathcal{N}) \leq C + \frac{a n_1 (a + n_1)}{2n} (\beta_1 - \beta_2).$$

Proof: Given $\pi \in \mathfrak{S}_n^\ell$, denote by $\bar{\pi}$ the permutation in which the first $n_2 - a + \ell$ positions contain nodes from L , the next $n_1 - \ell$ positions contain nodes from U , and the last a positions are the same as π . Lemma 9 and Remark 2 imply that $C_t(\bar{\pi}_t) \geq C_t(\pi_t)$. By (20) and by Lemma 10 we obtain

$$C_t(\bar{\pi}_t) + C_t(\bar{\pi}_t^c) = 2C + \ell(a + n_1)(\beta_1 - \beta_2).$$

Hence,

$$C_t(\pi_t) + C_t(\pi_t^c) \leq C_t(\bar{\pi}_t) + C_t(\bar{\pi}_t^c) = 2C + \ell(a + n_1)(\beta_1 - \beta_2)$$

which implies that

$$\begin{aligned} \sum_{\pi_t \in \mathfrak{S}_n} \Pr(\pi_t) C_t(\pi_t) &= \sum_{\ell=0}^{\min\{a, n_1\}} \sum_{\pi_t \in \mathfrak{S}_n^\ell} \Pr(\pi_t | \mathfrak{S}_n^\ell) \Pr(\mathfrak{S}_n^\ell) C_t(\pi_t) \\ &= \sum_{\ell=0}^{\min\{a, n_1\}} \Pr(\mathfrak{S}_n^\ell) \sum_{\pi_t \in \mathfrak{S}_n^\ell} \Pr(\pi_t | \mathfrak{S}_n^\ell) C_t(\pi_t). \end{aligned}$$

For $t \rightarrow \infty$, $\Pr(\pi_t | \mathfrak{S}_n^\ell)$ is uniform. Hence,

$$\begin{aligned} C_{\text{avg}}^*(\mathcal{N}) &= \sum_{\ell=0}^{\min\{a, n_1\}} \Pr(\mathfrak{S}_n^\ell) \left(\frac{1}{|\mathfrak{S}_n^\ell|} \sum_{\pi_t \in \mathfrak{S}_n^\ell} C_t(\pi_t) \right) \\ &\leq \sum_{\ell=0}^{\min\{a, n_1\}} \Pr(\mathfrak{S}_n^\ell) \left(\frac{\sum_{\pi_t \in \mathfrak{S}_n^\ell} (C_t(\bar{\pi}_t) + C_t(\bar{\pi}_t^c))}{2|\mathfrak{S}_n^\ell|} \right) \\ &= \sum_{\ell=0}^{\min\{a, n_1\}} \Pr(\mathfrak{S}_n^\ell) \left(C + \frac{1}{2} \ell(a + n_1)(\beta_1 - \beta_2) \right) \\ &= C + \sum_{\ell=0}^{n_1} \frac{\binom{n_1}{\ell} \binom{n_2}{a-\ell}}{\binom{n}{a}} \frac{\ell(a + n_1)(\beta_1 - \beta_2)}{2}, \end{aligned}$$

where the last equality follows from Lemma 11. By Vandermonde's identity we obtain that almost surely

$$C_{\text{avg}}^*(\mathcal{N}) \leq C + \frac{an_1(a + n_1)}{2n} (\beta_1 - \beta_2). \quad \blacksquare$$

Proposition 1 and Theorem 4 jointly result in the following (a.s.) inequalities for the average cut of the fixed-cost storage network:

$$\frac{an_1(\beta_1 - \beta_2)}{2n} \left(a + 1 + \frac{n_1 - 1}{n - 1} (a - 1) \right) \leq C_{\text{avg}}^*(\mathcal{N}) - C \leq \frac{an_1(\beta_1 - \beta_2)}{2n} (a + n_1) \quad (23)$$

where C is given in Lemma 7. For the above example, we obtain for the gap between $C_{\text{avg}}^*(\mathcal{N})$ and C an upper bound of 9.75. Generally, the difference between the upper and lower bounds (discounting the common multiplier) is $\frac{(n-a)(n_1-1)}{n-1}$. Of course, this does not directly result in an upper bound on capacity of \mathcal{N} , which appears to be a difficult question (a loose upper bound was obtained in (6), which in the example gives a gap of at most 21.5).

IV. Networks With Memory

Let us assume that the data collector DC_t in the dynamical fixed-cost model is aware of the state of the network; specifically, we assume that it selects the set D_t of k' active nodes for data retrieval with full knowledge of the permutation π_t . Under this assumption, DC_t can choose the nodes that *maximize* the cut between itself and D_t .

With this in mind, we give the following definition. Let $(\mathcal{N}, \beta, \mathbf{S}, h)$ be a storage network and let $C_t^h(D_t)$ denote the cut at time t for the selection of D_t active storage nodes. Let

$$C_t^{\max, h}(\mathcal{N}) = C_t^{\max, h}(\mathcal{A}_t) \triangleq \max_{D_t \subseteq \mathcal{A}_t, |D_t|=k'} \{C_t^h(D_t)\}. \quad (24)$$

(cf. (2)). Although the memory property does not affect the storage capacity when $\beta_i = \beta_0$ for all $i \in [n]$, using our idea of controlling the transmission policy enables us to increase the storage capacity. As a main result of this section, we show that the capacity of the network can be increased over the non-causal model.

Recall our notation $[n] = U \cup L$, where $|U| = n_1$, $|L| = n_2$. Throughout this section we denote $\hat{a} \triangleq k' - n_2 \geq 0$. The following lemma is a natural minimax analog of Lemma 7.

Lemma 13 *Let $(\mathcal{N}, \beta, s, h^*)$ be a static fixed-cost storage network and let*

$$C' \triangleq \min_{\pi \in \mathfrak{S}_n} \{C^{max, h}(\pi)\} = \min_{\substack{t \geq 0, \\ s \in V^\infty}} \{C_t^{max, h}\}. \quad (25)$$

Then

$$C' = \sum_{i=1}^{\hat{a}} n_1 \beta_1 + n_2 \beta_2 - i \beta_1 + \sum_{j=1}^{n_2} (n_1 - \hat{a}) \beta_1 + n_2 \beta_2 - j \beta_2. \quad (26)$$

Lemma 13 can be obtained from the next lemma which is a modified version of Lemma 9, together with the fact that every permutation appears as an associated permutation in $(\mathcal{N}, \beta, \mathbf{S})$ μ_1 -almost surely.

Lemma 14 *Let $(\mathcal{N}, \beta, s, h^*)$ be a storage network. For $t > t_0$, $C_t^{max, h^*}(\mathcal{N})$ is obtained when $D_t \supseteq L$.*

The proof of Lemma 14 is similar to the proof of Lemma 9 and is given in the appendix. Note that according to Lemma 9, the selection that minimizes the cut at time t is the node from U that has failed before the other nodes in U .

Remark 4 Similarly to Remark 2, from the proof of Lemma 9 it follows that after choosing the nodes in L , we should choose the remaining \hat{a} nodes in the order reversed from the order of their failure, starting with the most recently failed node.

For a network with memory $(\mathcal{N}, \beta, \mathbf{S})$ we denote the average (maximum) cut and the storage capacity by $C_{avg}^{max, h}$, $\text{cap}^m(\mathcal{N})$, respectively. The main result of this section is stated in the following theorem.

Theorem 5 *Let $(\mathcal{N}, \beta, \mathbf{S})$ be a (random) storage network with memory. We have (μ_1 -a.s.)*

$$\text{cap}^m(\mathcal{N}) \geq C' + \frac{\beta_1 - \beta_2}{2} \frac{n_1 n_2 \hat{a}}{n} \left(2 - \frac{\hat{a} - 1}{n - 1}\right).$$

In this section we denote by $\hat{\mathfrak{S}}_n^\ell$ the set of all permutations over $[n]$ with exactly ℓ elements from U in the last \hat{a} positions. To prove Theorem 5 we need the following lemma.

Lemma 15 *Let $(\mathcal{N}, \beta, \mathbf{S}, h^*)$ be a storage network with memory. Let π_t be the permutation at time t and assume that π_t is distributed uniformly over $\hat{\mathfrak{S}}_n^\ell$. We have*

$$\mathbb{E}[C_t^{max, h^*}(\mathcal{N})] \geq C' + \frac{1}{2} \ell (2n_2 - \hat{a} + \ell) (\beta_1 - \beta_2),$$

where C' is given in (25).

Proof: For any permutation $\pi \in \hat{\mathfrak{S}}_n^\ell$, let $\bar{\pi} \in \hat{\mathfrak{S}}_n^\ell$ be a permutation in which the first $n_1 - \ell$ positions contain only nodes from U , and the last \hat{a} positions are exactly as in π . Then by Lemma 14 and Remark 4 we have $C_t^m(\pi_t) \geq C_t^m(\bar{\pi}_t)$. This implies that by fixing the last \hat{a} positions in π_t , we can bound $C_t^{max, h^*}(\pi_t)$ below by $C_t^{max, h^*}(\bar{\pi}_t)$. We claim that

$$C_t^{max, h^*}(\pi_t) + C_t^{max, h^*}(\pi_t^c) \geq C' + \frac{1}{2} \ell (2n_2 - \hat{a} + \ell) (\beta_1 - \beta_2).$$

Note that if $\pi_t \in \hat{\mathfrak{S}}_n^\ell$ then $\pi_t^c \in \hat{\mathfrak{S}}_n^\ell$ as well. Hence, $C_t^{max, h^*}(\pi_t) + C_t^{max, h^*}(\pi_t^c) \geq C_t^{max, h^*}(\bar{\pi}_t) + C_t^{max, h^*}(\bar{\pi}_t^c)$. By Lemma 10 we obtain

$$\begin{aligned} C_t^{max, h^*}(\bar{\pi}_t) &= C' + \sum_{v \in D_t \cap U} f_{\bar{\pi}_t}(v) (\beta_1 - \beta_2) \\ &= C' + \sum_{\substack{v \in D_t \cap U \\ (\bar{\pi}_t)^{-1}(v) \in \{n - \hat{a} + 1, \dots, n\}}} f_{\bar{\pi}_t}(v) (\beta_1 - \beta_2) \end{aligned}$$

and the same holds for $\bar{\pi}_t^c$.

Let $v \in D_t \cap U$ with $(\bar{\pi}_t)^{-1}(v) \in \{n - \hat{a} + 1, \dots, n\}$, meaning that v is in one of the last \hat{a} positions. Let

$$b := |L \cap \{n - \hat{a} + 1, \dots, (\bar{\pi}_t)^{-1}(v)\}|.$$

Using definition of $\bar{\pi}_t \in \hat{\mathfrak{S}}_n^\ell$ and Lemma 14, we now observe that $f_{\bar{\pi}_t}(v) = n_2 - (\hat{a} - \ell) + b$. For $\bar{\pi}_t^c$ we have

$$f_{\bar{\pi}_t^c}(v) = n_2 - (\hat{a} - \ell) + (\hat{a} - \ell) - b = n_2 - b.$$

Overall we obtain

$$\begin{aligned} C_t^{\max, h^*}(\bar{\pi}_t) + C_t^{\max, h^*}(\bar{\pi}_t^c) &= 2C' + \sum_{v \in D_t \cap U} (f_{\bar{\pi}_t}(v) + f_{\bar{\pi}_t^c}(v)) (\beta_1 - \beta_2) \\ &= 2C' + \ell(2n_2 - \hat{a} + \ell) (\beta_1 - \beta_2) \end{aligned}$$

which in turn implies that

$$C_t^{\max, h^*}(\pi_t) + C_t^{\max, h^*}(\pi_t^c) \geq 2C' + \ell(2n_2 - \hat{a} + \ell) (\beta_1 - \beta_2).$$

We conclude the proof by noticing that

$$\begin{aligned} \mathbb{E}[C_t^{\max, h^*}(\mathcal{N})] &= \sum_{\pi_t \in \hat{\mathfrak{S}}_n^\ell} \Pr(\pi_t) C_t^{\max, h^*}(\pi_t) \\ &= \frac{1}{|\hat{\mathfrak{S}}_n^\ell|} \frac{1}{2} \sum_{\pi_t \in \hat{\mathfrak{S}}_n^\ell} (C_t^{\max, h^*}(\pi_t) + C_t^{\max, h^*}(\pi_t^c)) \\ &\geq C' + \frac{1}{2} \ell(2n_2 - \hat{a} + \ell) (\beta_1 - \beta_2). \end{aligned}$$

■

We can now prove Theorem 5.

Proof of Theorem 5: Consider $\mathbb{E}[C_t^{\max, h^*}(\mathcal{N})]$ and note that since $(\hat{\mathfrak{S}}_n^\ell)_\ell$ partition the set \mathfrak{S}_n we have

$$\begin{aligned} \mathbb{E}_\mu[C_{\text{avg}}^{\max, h^*}(\mathcal{N})] &= \sum_{\pi_t \in \mathfrak{S}_n} \Pr(\pi_t) C_t(\pi_t) \\ &= \sum_{\ell=0}^{\min\{\hat{a}, n_1\}} \sum_{\pi_t \in \hat{\mathfrak{S}}_n^\ell} \Pr(\pi_t | \hat{\mathfrak{S}}_n^\ell) \Pr(\hat{\mathfrak{S}}_n^\ell) C_t(\pi_t) \\ &= \sum_{\ell=0}^{\min\{\hat{a}, n_1\}} \Pr(\hat{\mathfrak{S}}_n^\ell) \sum_{\pi_t \in \hat{\mathfrak{S}}_n^\ell} \Pr(\pi_t | \hat{\mathfrak{S}}_n^\ell) C_t(\pi_t) \\ &\geq \sum_{\ell=0}^{\min\{\hat{a}, n_1\}} \Pr(\hat{\mathfrak{S}}_n^\ell) \left(C' + \frac{\ell(2n_2 - \hat{a} + \ell)\beta_1}{2} + \frac{\ell(2n_2 - \hat{a} + \ell)\beta_2}{2} \right) \\ &= C' + \sum_{\ell=0}^{\hat{a}} \frac{\binom{n_1}{\ell} \binom{n_2}{\hat{a}-\ell}}{\binom{n}{\hat{a}}} \frac{\ell(2n_2 - \hat{a} + \ell)(\beta_1 - \beta_2)}{2}, \end{aligned}$$

where the inequality follows from Lemma 12 and the last equality follows from Lemma 11 (with \hat{a}) and the fact that the stationary distribution of π_t is the uniform distribution. The final expression is obtained by repeated use of the Vandermonde convolution formula. The average cut bounds the storage capacity below since we can follow the same arguments as in Lemma 1 with C_t^{\max, h^*} instead of $C_t^{h^*}$. ■

For a numerical example we return to Example 3. If at each time t , DC_t chooses the k' nodes which yield the maximum cut, by Theorem 5, the storage capacity is $\text{cap}^m(\mathcal{N}) \geq C' + 13\frac{1}{3}\beta_2$, where $C' = 269\beta_2$. This is much greater than the lower bound computed earlier for the non-causal case, and in fact even breaks above the static-case upper bound of (6).

As seen from Theorem 5, if $\beta_1 = \beta_2$ the bound below is equal to the storage capacity of the static model. This comes as no surprise since the network is invariant under permutations of the storage nodes.

V. Extensions: Different failure probabilities

The dynamical model that we studied so far assumes that all the storage nodes have the same probability of failure. In reality, this may not be the case. An interesting extension of the above results would address a fixed-cost model in which the failure probability depends on the node (or a group to which the node belongs). We immediately note that the assumption of different failure probabilities does not affect the storage capacity of the static fixed-cost model.

Switching to the dynamical models, let us first assume that nodes from L fail independently with probability p and nodes from U fail (independently) with probability q . It is possible to adjust the weight function used in Section III to prove a lower bound on the network capacity. As before, let $(\mathcal{N}, \beta, \mathbf{S})$ be a fixed-cost storage network with n storage nodes and let $|U| = n_1, |L| = n_2$. For $s_t \in U$ we put $h_t(v_i) = h_U(v_i)$ and for $s_t \in L$ we put $h_t(v_i) = h_L(v_i)$, where

$$h_U(v_i) = \begin{cases} \beta_1 + \varepsilon_1 & v_i \in U \setminus s_j \\ \beta_2 & v_i \in L \\ 0 & v_i = s_j, \end{cases}$$

and

$$h_L(v_i) = \begin{cases} \beta_1 - \frac{q(n_1-1)}{pn_2} \varepsilon_1 & v_i \in U \\ \beta_2 & v_i \in L \setminus s_j \\ 0 & v_i = s_j \end{cases}$$

and $0 \leq \varepsilon_1 \leq \beta_1$. By a calculation similar to Lemma 8 it is straightforward to check that the constraints given by β are satisfied. Moreover, the proof of Theorem 3 does not use the fact that the stationary distribution of the associated permutations is uniform. Thus, from Lemma 9 and Lemma 10 we obtain the following statement.

Theorem 6 *Let $(\mathcal{N}, \beta, \mathbf{S}, h)$ be a fixed-cost storage network with weight function h as defined above. Assume that the failure probability of a node from U is $q > 0$ and of a node from L is $p > 0$. Fix $\varepsilon_1 > 0$ such that $\beta_1 - \beta_2 \geq \frac{qn(n_1-1)}{pn_2} \varepsilon_1$. The storage capacity is bounded below by*

$$\text{cap}(\mathcal{N}) \stackrel{\text{a.s.}}{\geq} C + \frac{n_1(n_1-1)}{2} \varepsilon_1, \quad (27)$$

where C is given in Lemma 7.

For a numerical example consider Example 3 with $q = \frac{1}{40}$ and $p = \frac{3}{40}$. Let us choose $\varepsilon_1 = \frac{pn_2}{qn(n_1-1)} \beta_2 = \frac{1}{6} \beta_2$. From (27) we now obtain

$$\text{cap}(\mathcal{N}) \geq (214 + 7.5) \beta_2,$$

where as above, $C = 214\beta_2$ is the value of the min-cut in the static case. As above in this paper, the assumption on ε_1 introduced in the theorem limits the increase of the network capacity. Lifting the assumption suggests following the path taken in Theorem 4 of Sec. III-B. To implement this idea, we need to find the stationary distribution of the Markov random walk on \mathfrak{S}_n that arises under our assumption. This is however not an easy task, and the classic (asymptotic) results such as in [10] seem not to be of help here. We have succeeded to perform the analysis in the simple case of $n = n_2 + 1$, i.e., of the "upper" set formed of a single node $U = \{u\}$, and we present this result in the remainder of this section.

Suppose that the failed nodes in the sequence \mathbf{S} are chosen independently and that $\Pr(\mathbf{S}_i = v) = p$ if $v \in L$ and $\Pr(\mathbf{S}_i = v) = q$ if $v \in U$. Assuming that $p, q \neq 0$, almost surely there exists a finite time t_0 such that all the nodes have failed at least once by t_0 . Choosing the next failed node gives rise to a permutation on \mathfrak{S}_n , and the conditional probabilities $\Pr(\pi_t | \pi_{t-1})$ between the permutations are well defined and can be found explicitly. The probabilities $\Pr(\pi_t | \pi_{t-1})$ define an ergodic Markov chain with a unique stationary distribution ν .

Define a partition of \mathfrak{S}_n into n blocks $P_i, i \in [n]$. Let $\pi \in P_i$ if and only if $\pi^{-1}(u) = i$ where u is the (unique) node in U . The partition (P_i) defines an obvious equivalence relation on \mathfrak{S}_n , and $|P_i| = (n-1)!$ for all i .

It turns out that the stationary probabilities of equivalent permutations are the same, i.e., $\nu(\pi)$ depends only on the block $P_i \ni \pi$. The distribution ν is given in the next lemma.

For any real number r and natural number k we define $\binom{r}{k} = \frac{r(r-1)\dots(r-k+1)}{k!}$, and put $\binom{r}{0} = 1$.

Lemma 16 Let $(\mathcal{N}, \beta, \mathbf{S})$ be a dynamical storage network with $n = n_1 + n_2$ nodes, where $n_1 = 1$. Let $0 < q \leq p$ and suppose that $\mathbf{S}_i, i = 1, 2, \dots$ are independent random variables with $\Pr(\mathbf{S}_i = v) = p$ if $v \in L$ and $\Pr(\mathbf{S}_i = v) = q$ if $v \in U$. Let $\pi \in P_i$ and define the distribution

$$\nu(\pi) = \frac{1-q}{(n-1)!} \binom{\frac{1}{p}-1}{n-2}^{-1} \binom{\frac{1}{p}-n-1+i}{i-1}.$$

Then ν is the stationary distribution of the Markov chain with state space \mathfrak{S}_n .

Proof: 1. We first note that for any t , $\Pr(\pi_{t+1}|\pi_t) = q$ if $\pi_{t+1} \in P_n$ and $\Pr(\pi_{t+1}|\pi_t) = p$ otherwise. This implies that for a fixed π_t ,

$$(n-1)p + q = \sum_{i=1}^n \Pr(\pi_{t+1} \in P_i|\pi_t) = 1.$$

Hence, $\frac{1}{p} \geq n-1$ which implies that all the binomial coefficients in $\nu(\pi)$ are positive. Moreover, since $(n-1)p = 1-q$ we obtain that if $\pi \in P_n$ then the expression for $\nu(\pi)$ simplifies as follows

$$\begin{aligned} \nu(\pi) &= \frac{1-q}{(n-1)!} \binom{\frac{1}{p}-1}{n-2}^{-1} \binom{\frac{1}{p}-n-1+n}{n-1} \\ &= \frac{1-q}{(n-1)!} \frac{\frac{1}{p}-n+1}{n-1} \\ &= \frac{q}{(n-1)!}. \end{aligned} \tag{28}$$

2. Let us check that ν is a probability vector. As already remarked, $\nu(\pi) > 0$ for all $\pi \in \mathfrak{S}_n$. Obviously, if $\pi, \sigma \in P_i$ for some i , then $\nu(\pi) = \nu(\sigma) = \frac{1}{(n-1)!} \nu(\{P_i\})$.

By the definition of ν we have that $\nu(\{P_{i+1}\}) = \nu(\{P_i\})(1 + \frac{1-pn}{pi})$ for all $i \leq n-1$. Therefore,

$$\begin{aligned} \sum_{\pi \in \mathfrak{S}_n} \nu(\pi) &= \sum_{i=1}^n \nu(\{P_i\}) \\ &= \sum_{i=1}^{n-1} \nu(\{P_i\}) + \nu(\{P_n\}) \\ &= \nu(\{P_1\}) \left(1 + \sum_{j=1}^{n-2} \prod_{i=1}^j \left(1 + \frac{1-pn}{pi} \right) \right) + q. \end{aligned}$$

Note that

$$\prod_{i=1}^j \left(1 + \frac{1-pn}{pi} \right) = \binom{j + \frac{1-pn}{p}}{j}$$

which implies that

$$\sum_{j=1}^{n-2} \prod_{i=1}^j \left(1 + \frac{1-pn}{pi} \right) = \binom{\frac{1}{p}-1}{n-2} - 1.$$

Since for $\pi \in P_1$, $\nu(\{P_1\}) = (n-1)! \nu(\pi)$ and $\nu(\pi) = \frac{1-q}{(n-1)!} \binom{\frac{1}{p}-1}{n-2}^{-1}$, we have

$$\sum_{\pi \in \mathfrak{S}_n} \nu(\pi) = (n-1)! \left(\binom{\frac{1}{p}-1}{n-2} \nu(\pi) + q \right) = 1.$$

Finally let us show that ν is a stationary vector of the transition matrix. Fix t and consider the sum $\sum_{\pi \in \mathfrak{S}_n} \nu(\pi) \Pr(\pi_{t+1} = \sigma | \pi_t = \pi)$. For $\sigma \in P_i$, this sum has exactly n non-zero terms, of which i are for $\pi \in P_{i+1}$ and $n-i$ for $\pi \in P_i$. Therefore, if $\sigma \in P_i$, we obtain

$$\sum_{\pi \in \mathfrak{S}_n} \nu(\pi) \Pr(\pi_{t+1} = \sigma | \pi_t = \pi) = \frac{pi}{(n-1)!} \nu(\{P_{i+1}\}) + \frac{p(n-i)}{(n-1)!} \nu(\{P_i\}).$$

Since $\nu(\{P_{i+1}\}) = \nu(\{P_i\})(1 + \frac{1-pn}{pi})$ for $i \leq n-1$, we have

$$\begin{aligned} \sum_{\pi \in \mathfrak{S}_n} \nu(\pi) \Pr(\pi_{t+1} = \sigma | \pi_t = \pi) &= \frac{p}{(n-1)!} \nu(\{P_i\}) \left(i \left(1 + \frac{1-pn}{pi} \right) + (n-i) \right) \\ &= \frac{1}{(n-1)!} \nu(\{P_i\}) \\ &= \nu(\sigma). \end{aligned}$$

Now assume that $\sigma \in P_n$. We obtain

$$\begin{aligned} \sum_{\pi \in \mathfrak{S}_n} \nu(\pi) \Pr(\pi_{t+1} = \sigma | \pi_t = \pi) &= \frac{1}{(n-1)!} \sum_{i=1}^n q \nu(\{P_i\}) \\ &= \frac{q}{(n-1)!} \left(\sum_{i=1}^{n-1} \nu(\{P_i\}) + \nu(\{P_n\}) \right). \end{aligned} \tag{29}$$

Using the fact that $\sum_{i=1}^{n-1} \nu(\{P_i\}) = 1 - q$ jointly with (29), we conclude that

$$\sum_{\pi \in \mathfrak{S}_n} \nu(\pi) \Pr(\pi_{t+1} = \sigma | \pi_t = \pi) = \frac{q}{(n-1)!} (1 - q + q),$$

recovering (28). This concludes the proof. \blacksquare

We can now bound the storage capacity below. Recall that \mathfrak{S}_n^ℓ denotes the set of all permutations on $[n]$ with ℓ numbers from $[n_1]$ in the last a positions. Let us use Lemmas 10 and 16 to calculate the expected minimum cut. In the below calculation we are taking some liberty in dealing with the conditional distribution $\nu(\pi_t | P_i)$ which operationally is the limiting (conditional) probability on \mathfrak{S}_n . A more rigorous approach requires defining a conditional distribution for a finite time t and arguing that it approaches $\nu(\pi_t | P_i) = \frac{1}{(n-1)!}$. At the same time, the final answer below is correct as written. We proceed as follows:

$$\begin{aligned} \sum_{\pi_t \in \mathfrak{S}_n} \nu(\pi_t) C_t^{h*}(\pi_t) &= \sum_{i \in [n]} \sum_{\pi_t \in P_i} \nu(P_i) \nu(\pi_t | P_i) C_t^{h*}(\pi_t) \\ &= \sum_{i \in [n-a]} \sum_{\pi_t \in P_i} \nu(P_i) \nu(\pi_t | P_i) C_t^{h*}(id) \\ &\quad + \sum_{i=n-a+1}^n \frac{(1-q)}{(n-1)!} \left(\frac{\frac{1}{p}-1}{n-2} \right)^{-1} \binom{\frac{1}{p}-n+i-1}{i-1} \sum_{\pi_t \in P_i} C_t^{h*}(\pi_t) \\ &= (1-q) \left(\frac{\frac{1}{p}-1}{n-2} \right)^{-1} \binom{\frac{1}{p}-a}{n-a} C \\ &\quad + \sum_{i=n-a+1}^n (1-q) \left(\frac{\frac{1}{p}-1}{n-2} \right)^{-1} \binom{\frac{1}{p}-n+i-1}{i-1} (C + (i-n+a)(\beta_1 - \beta_2)) \end{aligned}$$

where C is given (11). To argue that this expression can be used in the lower bound on $\text{cap}(\mathcal{N})$ similar to the bound in Theorem 4 (or in (21)) we can repeat the arguments used in the proof of Lemma 12. Then a modified version of (21) together with the above expression for ν gives a lower bound on the capacity.

To give a numerical example, assume that we have $n = 20$ with $n_2 = 19$, $p = \frac{4}{95}$ and $q = \frac{1}{5}$. Assume also that $\beta_1 = 2\beta_2$ and $k' = 13$ (which implies that $a = 12$). According to Lemma 7, the capacity in the static model is $C = 150\beta_2$. Using the results in this section, we obtain that a.s.

$$\text{cap}(\mathcal{N}) \geq (0.022 \cdot 150 + 155.4)\beta_2 = 158.7\beta_2.$$

Lemma 3 implies that $\text{cap}(\mathcal{N}) \leq 177.45\beta_2$ and thus in the above example we have obtained the capacity increase of more than 30% of the gap between the bounds.

VI. Concluding Remarks

In this work we introduced a dynamical model for distributed storage systems. We provided lower bounds on the capacity for the fixed-cost model with no memory and with memory. For the memoryless network, we also provided a simple transmission protocol that increases the storage capacity of the network over the static case. We did not manage to optimize the weight assignment, which is left as an open question for future work, or to provide explicit code families that support reliable file storage while accounting for the time evolution of the storage network. Another extension that we did not address is to consider more than two clusters of nodes with different values of (average) repair bandwidth. It is also possible to argue that once the node has been repaired, it is less likely to fail for a certain period of time. At this point we do not have an approach to the analysis of this general question.

Appendix

A. Proof of Lemma 6

First note that Lemmas 4 and 5 imply that if $(\mathcal{N}_2, \beta, \mathbf{S}, h^*)$ is a random discrete-time storage network, then $C_{\text{avg}}^{h^*}(\mathcal{N}_2)$ is almost surely a constant, which is equal to $\frac{1}{n!} \sum_{\pi \in \mathfrak{S}_n} C_t(\pi)$. On the other hand, if $(\mathcal{N}_1, \beta, \mathbf{S}, t, h^*)$ is a continuous-time storage network, then we can write

$$\begin{aligned} C_{\text{avg}}^{h^*}(\mathcal{N}_1) &= \limsup_{\tau \rightarrow \infty} \frac{1}{\tau} \int_0^\tau C_t^{h^*}(\mathcal{N}_1) dt \\ &= \limsup_{\tau \rightarrow \infty} \frac{1}{\tau} \int_{t_0}^\tau \sum_{\pi \in \mathfrak{S}_n} \mathbb{1}_\pi(\pi_t) C_t^{h^*}(\mathcal{N}_1) dt \\ &= \limsup_{\tau \rightarrow \infty} \frac{1}{\tau} \sum_{\pi \in \mathfrak{S}_n} \int_{t_0}^\tau \mathbb{1}_\pi(\pi_t) C_t^{h^*}(\pi) dt \end{aligned} \quad (30)$$

where t_0 is the first time instance by which all the nodes have failed. Moreover, since $C_t^{h^*}(\pi)$ is a function of π and not a function of t we denote $C_t^{h^*}(\pi)$ by $C^{h^*}(\pi)$, and obtain

$$\begin{aligned} C_{\text{avg}}^{h^*}(\mathcal{N}_1) &= \sum_{\pi \in \mathfrak{S}_n} \limsup_{\tau \rightarrow \infty} \frac{1}{\tau} \int_{t_0}^\tau \mathbb{1}_\pi(\pi_t) C^{h^*}(\pi) dt \\ &= \sum_{\pi \in \mathfrak{S}_n} \lim_{\tau \rightarrow \infty} \frac{1}{\tau} C^{h^*}(\pi) \int_{t_0}^\tau \mathbb{1}_\pi(X(t')) dt \\ &\stackrel{a.s.}{=} \sum_{\pi \in \mathfrak{S}_n} \frac{C^{h^*}(\pi)}{n\lambda \mathbb{E}[(\pi \rightarrow \pi)]} \end{aligned}$$

where the last equality follows from (5). Since $\mathbb{E}[(\pi \rightarrow \pi)]$ does not depend on $\pi \in \mathfrak{S}_n$ we obtain that $\frac{1}{n\lambda \mathbb{E}[(\pi \rightarrow \pi)]} = \frac{1}{n!}$, which in turn implies that $C_{\text{avg}}^{h^*}(\mathcal{N}_1) = C_{\text{avg}}^{h^*}(\mathcal{N}_2)$ almost surely.

B. Proof of Lemma 14

Assume that π_t is a fixed permutation and consider the information flow graph \mathcal{X}_t . We consider a k' -step procedure which in each step selects one node from \mathcal{A}_t . Let $t' \leq t$ and assume the node $v_{i_t}^t \in \mathcal{A}_t$ was selected. The cost it entails is defined as the added weight values of the in-edges of CU_t that are not out-edges of previously selected nodes. Our goal is to select k' nodes that maximize the cut for π_t .

In order to simplify notation, we write $\pi_t = (u_1, u_2, \dots, u_n)$, i.e., $u_l = v_{\pi_t(l)}^t$ is the storage node that appears in the l th position in π_t . Moreover, with a slight abuse of notation, if u_j failed at time t' we will write $h_j(u_i)$ instead of $h_{t'}(u_i)$. For $\kappa \leq k'$, consider the sub-problem at step $\kappa - 1$, where the DC_t has already chosen $\kappa - 1$ nodes $(u_{i_1}, \dots, u_{i_{\kappa-1}})$ and we are to choose the last node. Assume that the chosen nodes are ordered according to their appearance in the permutation, i.e., $i_1 \leq i_2 \leq \dots \leq i_{\kappa-1}$. Let $u_{j_1}, \dots, u_{j_m} \in L$ be nodes that were not selected up to step $\kappa - 1$, i.e.,

$$\{u_{j_1}, \dots, u_{j_m}\} \cap \{u_{i_1}, \dots, u_{i_{\kappa-1}}\} = \emptyset,$$

and assume also that $j_1 \leq j_2 \leq \dots \leq j_m$. We show that choosing u_{j_1} accounts for the maximum cut.

First, we show that choosing u_{j_1} maximizes the cut over all other nodes from L . Denote by $C_{\kappa-1}$ the total cost (or the cut) in step $\kappa - 1$. Fix $2 \leq \ell \in [m]$ and note that since $j_1 \leq j_\ell$ we may write

$$i_1 \leq \dots \leq i_{r_1} \leq j_1 \leq i_{r_1+1} \leq \dots \leq i_{r_\ell} \leq j_\ell \leq i_{r_\ell+1} \leq \dots,$$

where j_1 could also be 1. Let $C(j_1)$ be the cut value if DC_t chooses u_{j_1} in the κ th step, respectively. The change from $C_{\kappa-1}$ is formed of the following components. First, we add the values of all the edges from U to u_{j_1} and from $L \setminus \{u_{j_1}\}$ to u_{j_1} , accounting for $(n_1 - 1)(\beta_1 + \varepsilon_1) + n_2\beta_2$ symbols. Next, for each node u_{i_q} with $r_1 < q \leq \kappa - 1$, we subtract $h_{i_q}^*(u_{j_1})$ from the cut value. Overall we obtain

$$C(j_1) = C_{\kappa-1} + n_1\beta_1 + (n_2 - 1)\beta_2 - \sum_{q=1}^{r_1} h_{j_1}^*(u_{i_q}) - \sum_{q=r_1+1}^{\kappa-1} h_{i_q}^*(u_{j_1}). \quad (31)$$

Following the same steps for $C(j_\ell)$, $\ell \geq 2$ we obtain

$$C(j_\ell) = C_{\kappa-1} + n_1\beta_1 + (n_2 - 1)\beta_2 - \sum_{q=1}^{r_\ell} h_{j_\ell}^*(u_{i_q}) - \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}^*(u_{j_\ell}).$$

Since $u_{j_1}, u_{j_\ell} \in L$, we obtain that $h_{j_\ell}^*(u_i) = h_{j_1}^*(u_i)$ and $h_i(u_{j_1}) = h_i(u_{j_\ell})$ for all $i \in [n]$, we have

$$C(j_1) - C(j_\ell) = \sum_{q=r_1+1}^{r_\ell} (h_{j_\ell}^*(u_{i_q}) - h_{i_q}^*(u_{j_1})).$$

For $u_{i_q} \in U$, we obtain $h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1}) = \beta_1 - \beta_2 \geq 0$ and for $u_{i_q} \in L$, we obtain $h_{j_\ell}(u_{i_q}) - h_{i_q}(u_{j_1}) = \beta_2 - \beta_2 = 0$, and so

$$C(j_1) - C(j_\ell) \geq 0.$$

Now we show that u_{j_1} maximizes the cut over the selection of any node u_{j_ℓ} from U . We divide the argument into 2 cases:

- 1) Assume that $j_\ell < j_1$. Denote by $(i_1, \dots, i_{r_\ell}, j_\ell, i_{r_\ell+1}, \dots, i_{r_1}, j_1, \dots)$ the selected nodes and let $C(j_\ell), C(j_1)$ be the cut values if we choose u_{j_ℓ}, u_{j_1} , respectively. We have

$$C(j_\ell) = C_{\kappa-1} + (n_1 - 1)\beta_1 + n_2\beta_2 - \sum_{q=1}^{r_\ell} h_{j_\ell}^*(u_{i_q}) - \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}^*(u_{j_\ell})$$

Subtracting $C(j_\ell)$ from $C(j_1)$ and using (31), we obtain

$$\begin{aligned} C(j_1) - C(j_\ell) &= \beta_1 - \beta_2 + \sum_{q=1}^{r_\ell} h_{j_\ell}^*(u_{i_q}) - \sum_{q=1}^{r_1} h_{j_1}^*(u_{i_q}) + \sum_{q=r_\ell+1}^{\kappa-1} h_{i_q}^*(u_{j_\ell}) - \sum_{q=r_1+1}^{\kappa-1} h_{i_q}^*(u_{j_1}) \\ &\geq \sum_{q=1}^{r_\ell} (h_{j_\ell}(u_{i_q}) - h_{j_1}(u_{i_q})) - \sum_{q=r_\ell+1}^{r_1} h_{j_1}(u_{i_q}) \\ &\quad + \sum_{q=r_\ell+1}^{r_1} h_{i_q}(u_{j_\ell}) + \sum_{q=r_1+1}^{\kappa-1} (h_{i_q}(u_{j_\ell}) - h_{i_q}(u_{j_1})) \\ &\geq \beta_1 - \beta_2 + \sum_{q=r_\ell+1}^{r_1} (h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q})) \end{aligned}$$

Since $u_{i_q} \in U$ we have $h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q}) = 0$ and for $u_{i_q} \in L$ we have $h_{i_q}(u_{j_\ell}) - h_{j_1}(u_{i_q}) > 0$, we conclude that $C(j_1) - C(j_\ell) \geq 0$.

- 2) Now assume $j_\ell > j_1$. This case is symmetric to the case $j_\ell < j_1$ and relies on the same analysis. We omit the details.

According to the principle of optimality [6, Ch. 1.3], every optimal policy consists only of optimal sub-policies, and therefore we first need to choose all the nodes from U and then choose nodes from L . This completes the proof.

References

- [1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
- [2] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "Cost-bandwidth tradeoff in distributed storage systems," *Computer Communications*, vol. 33, no. 17, pp. 2105–2115, 2010.
- [3] D. Aldous and P. Diaconis, "Shuffling cards and stopping times," *The American Mathematical Monthly*, vol. 93, no. 5, pp. 333–348, 1986.
- [4] A. Badita, P. Parag, and J.-F. Chamberland, "Latency analysis for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 4683–4698, 2019.
- [5] S. B. Balaji, M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan, and P. V. Kumar, "Erasure coding for distributed storage: an overview," *Science China Information Sciences*, vol. 61, no. 10, p. 100301, 2018.
- [6] D. P. Bertsekas, *Dynamic programming and optimal control*. Belmont, MA: Athena Scientific, 2005.
- [7] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymptotic interference alignment for optimal repair of MDS codes in distributed storage," *IEEE Trans. Inf. Theory*, vol. 59, no. 5, pp. 2974–2987, 2013.
- [8] B. Calder, J. Wang, A. Ogun, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci *et al.*, "Windows Azure Storage: a highly available cloud storage service with strong consistency," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 143–157.
- [9] A. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.
- [10] L. Flatto, A. Odlyzko, and D. Wales, "Random shuffles and group representations," *The Annals of Probability*, vol. 13, no. 1, pp. 154–178, 1985.
- [11] R. G. Gallager, *Stochastic processes: theory for applications*. Cambridge University Press, 2013.
- [12] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford Univ. Press, 2001.
- [13] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *Proc. 50th Annual Allerton Conf. Commun. Control Comput.*, 2012, pp. 326–333.
- [14] A. M. Kermarrec, N. L. Scouarnec, and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," in *Int. Symp. on Network Coding (NetCod)*. IEEE, 2011, pp. 1–6.
- [15] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads," in *Proc. 2012 USENIX Conf. on File and Storage Technology (FAST)*, 2012, 14pp.
- [16] M. Luby, R. Padovani, T. Richardson, L. Minder, and P. Aggarwal, "Liquid cloud storage," *ACM Transactions on Storage*, vol. 15, no. 1, 2019, Article No. 2, 49 pp.
- [17] M. Silberstein, L. Ganesh, Y. Wang, L. Alvisi, and M. Dahlin, "Lazy means smart: Reducing repair bandwidth costs in erasure-coded distributed storage," in *Proceedings of International Conference on Systems and Storage*. ACM, 2014, pp. 1–7.
- [18] J. Y. Sohn, B. Choi, S. W. Yoon, and J. Moon, "Capacity of clustered distributed storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 1, pp. 81–107, 2019.