Towards Green Distributed Storage Systems

Abdelrahman M. Ibrahim, Ahmed A. Zewail, and Aylin Yener

Wireless Communications and Networking Laboratory (WCAN)
Electrical Engineering Department
The Pennsylvania State University, University Park, PA 16802.
{*ami137,aiz103*}*@psu.edu* *yener@engr.psu.edu*

***Abstract*—We model a distributed storage system consisting of energy harvesting nodes which store multiple files. To investigate the performance of file retrieval and node repair, we formulate two optimization problems: maximizing the number of retrieved (repaired) files given a deadline, and minimizing the retrieval (repair) time of a number of stored files. We derive the necessary and sufficient conditions for the feasibility of retrieving (repairing) a number of files by a deadline. Utilizing these conditions, we reduce the aforementioned problems to a single feasibility problem, which is solved using forward and backward algorithms. Finally, the system performance is illustrated numerically.**

I. Introduction

Distributed Storage Systems (DSS) have garnered extensive research interest over the recent years. In particular, the development of regenerating codes [1], [2], where a file is encoded into n pieces, each with size α, has been a propeller of this area. In such systems, file pieces are stored in a distributed manner over a system consisting of n storage nodes, such that the whole file can be reconstructed from the pieces stored in any k nodes. Furthermore, if a node fails, the system can initiate a repair process in which a newcomer node generates the content of the failed node, by connecting to d alive nodes and downloading β bits from each of them. Regenerating codes achieve the fundamental trade-off between the node storage capacity α and the repair bandwidth $d\beta$ [1], [2].

The main goal in [1], [2] and the references therein is to design codes that achieve this fundamental trade-off and to investigate their characteristics. More recently, building on these results, systems with more dynamics have been considered. For example, references [3] and [4] consider the possibility of communication outage between the storage nodes and the newcomer node. The probability of successful repair is investigated under different channel conditions and several techniques are proposed to maximize this probability. Reference [5] considers the impact of network topology of the system, which is captured by the transmission cost during the repair process. Reference [6] considers a distributed storage system with multiple files and investigates the file retrieval latency under different scheduling schemes. In all the aforementioned effort, system and channel dynamics are utilized to design techniques that enhance the system performance.

Separately, communications using energy harvesting nodes have also been considered extensively in recent years, see for example, [7] and [8], without data placement considerations.

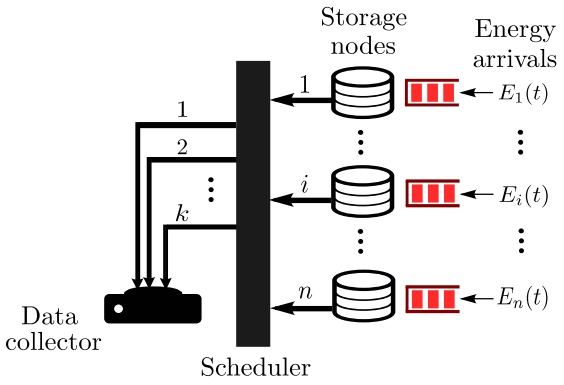

Fig. 1: Distributed storage with energy harvesting nodes.

These works mainly consider amortizing the harvested energy appropriately so as to maximize system throughput.

In this paper, we propose a distributed storage system with energy harvesting nodes. Our motivation behind considering energy harvesting storage nodes is towards building *green data centers* [9]. As, the electricity costs count for about 20% of the operational costs of data centers [9], envisioning energy self-sufficiency for these is a relevant objective.

We aim to utilize information about the energy arrivals to design transmission algorithms that ensure the success of the system operations, i.e., file retrieval and node repair. In particular, we consider a system consisting of n storage nodes equipped with energy harvesters and infinite capacity batteries, see Fig. 1. F files are stored over the system nodes. We design scheduling algorithms that optimize file retrieval and node repair performance. We formulate the problem of maximizing the number of retrieved (repaired) files by a deadline T. In addition, we formulate the problem of minimizing the retrieval (repair) time of M different files. Next, we derive the necessary and sufficient conditions to ensure the feasibility of retrieving (repairing) M files by a deadline T. Then, we reduce both optimization problems to a feasibility problem. Last, we propose two scheduling algorithms that provide feasible power allocation to ensure the success of the system operations.

The remainder of this paper is organized as follows. In Section II, we describe the system model and the main

assumptions. The problem formulation is introduced in Section III. Section IV contains the proposed algorithms, which are illustrated by numerical examples in Section V. Conclusions are presented in Section VI.

Notation: Throughout this paper, we adopt the following notation. Matrices and vectors are represented by boldface uppercase and lowercase letters, respectively. For a matrix $\boldsymbol{A}$, $A_i(l)$ denotes the element (i,l) and $\boldsymbol{a}(l)$ denotes column l.

II. System Model and Main Assumptions

Consider a system consisting of n storage nodes, represented by the set $\mathcal{N} = \{1,2,..,n\}$, which stores F files, represented by the set $\mathcal{F} = \{1,2,..,F\}$. Each file, with size U bits, is encoded using an (n,k,d,α,β) regenerating code [1], [2]. Each node in the system has storage capacity $\alpha_T = F\alpha$, and stores an encoded piece from each file. In addition, each node is equipped with an energy harvesting unit and an infinite capacity battery, as depicted in Fig. 1. We consider a time-slotted system, where energy arrives at the beginning of a time slot. For convenience, we assume unit length time slots and in turn, we use the terms power and energy interchangeably. We denote the energy arrival to node i at time slot l by $E_i(l)$. To capture different channel conditions, we consider a monotonically increasing energy cost function $v_i(.)$, i.e., node i consumes $v_i(b)$ energy units in the transmission of b bits per time slot. Moreover, we assume offline knowledge of the energy arrivals and the harvested energy is used only in data transmission. We have two modes of data transmission, *retrieval* and *repair* modes.

In the *retrieval mode*, a data collector (DC) joins the system to retrieve M files, given by the subset $\mathcal{M} \subset \mathcal{F}$. The file retrieval process is done file-by-file. To retrieve each file, a set of k nodes is required to transmit the stored data content related to this file, i.e., each of the k nodes transmits α bits. The transmission occurs only if there exists a set of k nodes, each with stored energy greater than or equal to $v_i(\alpha)$. Communications take place over one time slot via k orthogonal noiseless links. From the received $k\alpha$ bits the data collector is able to reconstruct the whole file. Similarly, the system repeats this process to retrieve the remaining files in the set $\mathcal{M}$. Note that at any time slot during the retrieval mode, we have only two possibilities, if there exist k or more nodes with sufficient energy to transmit α bits, then only k nodes transmit their data to the data collector to retrieve a certain file, otherwise all nodes remain silent.

In the *repair mode*, a newcomer joins the system to replace a failed node. Similar to the retrieval mode, the repair is done file-by-file. At a certain time slot, if there exists a set of d nodes, each of which has energy greater than or equal to $v_i(\beta)$, then these nodes transmit β bits each to the newcomer. With the received $d\beta$ bits, the newcomer generates a piece of size α that preserves the functionality of the system. Again, during the repair mode, the system either repairs the content related to one file by transmitting β bits from d nodes, or all nodes remain silent. Lastly, note that there is no overlap between the two modes of operation. We summarize the operational assumptions as follows.

(S1) A file is retrieved (repaired) by transmitting $k\alpha$ ($d\beta$) bits from k (d) nodes, and each node transmits α (β) bits.
(S2) At any time slot, during the retrieval (repair) mode either k (d) nodes transmit or all nodes remain silent.
(S3) The transmission of b bit costs $v_i(b)$ units of energy, for node i.

III. Problem Formulation

In the following, we focus mainly on the retrieval mode, however, our results can be applied directly to the repair mode by replacing the parameters M, n, k and α with F, $n-1$, d and β, respectively as evident from the assumptions (S1)-(S3).

First, we introduce the notion of effective accumulated energy and the energy feasibility conditions that ensure retrieving M files by time slot T. Second, we formulate two optimization problems that capture the system operations.

Definition 1. *The **effective accumulated energy** at node i by time slot l, $A_i(l)$, represents the maximum energy that can be utilized at node i by time slot l, and is obtained by the following recursive relation in $z(l-r), r = 0,\dots,l-1$, which represents the maximum energy that can be utilized out of the energy arrivals $E_i(l-r),\dots,E_i(l)$.*

$$A_i(l) = z(1),\ z(l+1) = 0, \tag{1}$$

$$z(l-r) = \min\left\{z(l-r+1) + E_i(l-r),\ (r+1)v_i(\alpha)\right\}. \tag{2}$$

■

For example, the effective accumulated energy of node i at $l=2$ is $A_i(2) = \min\left\{E_i(1)+\min\left\{E_i(2),\ v_i(\alpha)\right\},\ 2v_i(\alpha)\right\}$. Note that this notion follows from the assumption that a node i consumes at most $v_i(\alpha)$ units of energy at any time slot, i.e., $\sum_{l=1}^{j} E_i(l) = A_i(j) + A_i^e(j)$, where $A_i^e(j)$ is excess energy saved for future use. Next, we present necessary and sufficient conditions on the harvested energy profiles to ensure the feasibility of retrieving M files by time slot T.

Lemma 1. *The following conditions for a sequence of time slots $\{l_m\}_{m=1}^{M}$, where $0 < l_1 < \ldots < l_M = T$, are necessary and sufficient for the existence of a feasible power allocation to retrieve M files by T.*

$$\sum_{i=1}^{n} \left\lfloor \frac{A_i(l_m)}{v_i(\alpha)} \right\rfloor \geq km, \tag{3}$$

$$\sum_{i=1}^{n} \left\lfloor \frac{A_i(l_m)}{v_i(\alpha)} \right\rfloor - \left\lfloor \frac{A_j(l_m)}{v_j(\alpha)} \right\rfloor \geq (k-1)m,\ j = \arg\max\nolimits_{i\in\mathcal{N}}^{(1)}\{A_i(l_m)\}, \tag{4}$$

$$\sum_{i\in\mathcal{S}_{l_m}} \left\lfloor \frac{A_i(l_m)}{v_i(\alpha)} \right\rfloor \geq m,\ \mathcal{S}_{l_m} = \arg\min\nolimits_{i\in\mathcal{N}}^{(n-k+1)}\{A_i(l_m)\}, \tag{5}$$

where $\mathcal{S}_{l_m}$ is the set of $n-k+1$ nodes with minimum effective accumulated energies at time slot l_m. That is, if we have $A_{i_1}(l_m) \leq \cdots \leq A_{i_{n-k+1}}(l_m) \leq \cdots \leq A_{i_n}(l_m)$, then, the set $\mathcal{S}_{l_m}$ and its complement $\mathcal{S}_{l_m}^c$ are defined as $\{i_1,\dots,i_{n-k+1}\}$ and $\{i_{n-k+2},\dots,i_n\}$, respectively.

Note that, at time slot l_m, (3) ensures that the storage nodes total energy is sufficient to retrieve m files; (4) ensures that the effective accumulated energy of any node contribute to at most m files. Lastly, (5) ensures that there exist at least k nodes with sufficient energy to transmit in m time slots.

In the following, we formulate the optimization problems that represent the file retrieval operation. In particular, we consider the problem of maximizing the number of retrieved files M, given a deadline T, which we refer to as the *throughput maximization* problem. We also consider the problem of minimizing the retrieval time T of M files, which we refer to as the *retrieval time minimization* problem. The throughput maximization problem is given by the following $0-1$ programming problem.

$$\textbf{\textit{O1}}: \max_{\Delta_i(l)} \quad \frac{1}{k}\sum_{i=1}^{n}\sum_{l=1}^{T}\Delta_i(l) \tag{6a}$$

$$\text{subject to} \quad \sum_{l=1}^{j}\Delta_i(l) \leq \Big\lfloor \frac{A_i(j)}{v_i(\alpha)} \Big\rfloor,\ \forall i \in \mathcal{N}, j=1,\ldots,T, \tag{6b}$$

$$\Delta_i(l) \in \{0,1\},\ \forall\ i \in \mathcal{N},\ l=1,\ldots,T, \tag{6c}$$

$$\sum_{i=1}^{n}\Delta_i(l) \in \{0,k\},\ l=1,\ldots,T. \tag{6d}$$

where $\Delta_i(l) = P_i(l)/v_i(\alpha)$ is the transmission indicator of node i at time slot l, and $P_i(l)$ is the transmitted power by node i at time slot l. Note that (6b) captures the energy causality constraints, while (6c) and (6d) represent the operational assumptions (S1)-(S3).

The retrieval time minimization problem is given by the following $0-1$ programming problem.

$$\textbf{\textit{O2}}: \min_{\Delta_i(l)} \quad T \tag{7}$$

$$\text{subject to} \quad \sum_{i=1}^{n}\sum_{l=1}^{T}\Delta_i(l) = kM,\ (6b),\ (6c),\ (6d).$$

Our solution methodology in Section IV is motivated by the following theorem.

Theorem 1. *[10, Theorem 1] The optimal solution of a linear integer program can be found in polynomial time if the optimal objective value can be identified in polynomial time.* □

IV. The proposed Algorithms

In this section, we provide algorithms that solve ***O1*** and ***O2*** in two steps. First, *Algorithm 1* and *Algorithm 2* reduce the problems ***O1*** and ***O2***, respectively, to the feasibility problem ***F1***. Each of these two algorithms checks the necessary and sufficient conditions provided in Lemma 1, sequentially, in order to obtain the optimal objective value.

$$\textbf{\textit{F1}}: \text{Find} \quad \boldsymbol{\Delta} \tag{8}$$

$$\text{subject to} \quad \sum_{i=1}^{n}\sum_{l=1}^{T}\Delta_i(l) = kM,\ (6b),\ (6c),\ (6d).$$

where $\Delta_i(l)$ is the element (i,l) in $\boldsymbol{\Delta}$. Note that the complexity of *Algorithm 1* is $O(nT)$. Next, we present two algorithms that solve the $0-1$ assignment problem in (8).

Algorithm 1 Finds the maximum number of files M to be retrieved by a deadline T.

Input: $A_i(l),\ \forall\ i \in \mathcal{N},\ l=1,\ldots,T.$
Output: M
1: $M \leftarrow 0$.
2: **for** $l=1$ **to** T **do**
3: **if** $\sum_{i=1}^{n}\lfloor \frac{A_i(l)}{v_i(\alpha)} \rfloor \geq k(M+1)$ **and** $\sum_{i=1}^{n}\lfloor \frac{A_i(l)}{v_i(\alpha)} \rfloor - \lfloor \frac{A_j(l)}{v_j(\alpha)} \rfloor \geq (M+1)(k-1),\ j = \arg\max^{(1)}_{i\in\mathcal{N}}\{A_i(T)\}$ **and** $\sum_{i\in\mathcal{S}_l}\lfloor \frac{A_i(l)}{v_i(\alpha)} \rfloor \geq (M+1)$ **then**
4: $M \leftarrow M+1$.
5: **end if**
6: **end for**

Algorithm 2 Finds the minimum retrieval time T of M files.

Input: M, and $A_i(l),\ \forall\ i \in \mathcal{N},\ \forall\ l.$
Output: T
1: $T \leftarrow 0$.
2: **for** $m=1$ **to** M **do**
3: $T \leftarrow T+1$.
4: **while** $\sum_{i=1}^{n}\lfloor \frac{A_i(T)}{v_i(\alpha)} \rfloor < km$ **or** $\sum_{i=1}^{n}\lfloor \frac{A_i(T)}{v_i(\alpha)} \rfloor - \lfloor \frac{A_j(T)}{v_j(\alpha)} \rfloor < m(k-1),\ j = \arg\max^{(1)}_{i\in\mathcal{N}}\{A_i(T)\}$ **or** $\sum_{i\in\mathcal{S}_T}\lfloor \frac{A_i(T)}{v_i(\alpha)} \rfloor < m$ **do**
5: $T \leftarrow T+1$.
6: **end while**
7: **end for**

A. Forward algorithm

In this algorithm, we have T time slots, in each stage l, we search for $\boldsymbol{\delta}(l) \in \mathcal{D}_l$, where $\boldsymbol{\delta}(l) = [\Delta_1(l),\ldots,\Delta_n(l)]^T$ and $\mathcal{D}_l$ is the transmissions feasibility set at stage l, defined by

$$\mathcal{D}_l = \Big\{\boldsymbol{\delta}(l) \in \{0,1\}^n : \Delta_i(l) \leq \Big\lfloor \frac{A_i(l)}{v_i(\alpha)} \Big\rfloor, \forall i \in \mathcal{N}, \sum_{i=1}^{n}\Delta_i(l) \in \{0,k\}\Big\}. \tag{9}$$

In order to guarantee the feasibility of retrieving the remaining files in the upcoming slots, we apply *Algorithm 2* on the updated effective accumulated energies $\hat{A}_i(l),\ i \in \mathcal{N}, l = 1,\ldots,T$, which represent the elements of the matrix $\hat{\boldsymbol{A}}$, as shown in steps 6 to 8. The advantage of the forward algorithm is that we retrieve a file a soon as we have sufficient energy and this transmission does not affect the feasibility of retrieving the remaining files. Hence, as a byproduct, the forward algorithm minimizes the average delay per file.

B. Backward algorithm

In this algorithm, we have M stages indexed by q, which correspond to time slots T to $T-M+1$, i.e., $l = T-q+1$. First, we capture the energy arrivals $\boldsymbol{E}$ by a transmission opportunity matrix $\boldsymbol{B}$ with $B_i(l)$ as its (i,l) entry. $B_i(l) \in \{0,1\}$

Algorithm 3 The forward algorithm for finding a feasible power allocation to retrieve M files by time slot T.

Input: T, M and $A_i(l),\ \forall\ i \in \mathcal{N},\ l{=}1,\ldots,T$.
Output: $\boldsymbol{\Delta}$

$\boldsymbol{\Delta} \leftarrow 0$.
$f \leftarrow 0$.
while $l \leq T$ **and** $f < M$ **do**
  repeat
    Choose $\boldsymbol{\delta}(l) \in \mathcal{D}_l$.
    $\hat{\boldsymbol{a}}(r) \leftarrow \boldsymbol{a}(r) - [v_1(\alpha)\Delta_1(l),\ldots,v_n(\alpha)\Delta_n(l)]^T$, $r = l, l{+}1,\ldots,T$.
    Run *Algorithm 2* with input: $\hat{\boldsymbol{a}}(r),\ r = l, l{+}1,\ldots,T$.
    $\hat{M} \leftarrow$ *Algorithm 2* output.
  until $\hat{M} = M{-}f{-}1$.
  if $\boldsymbol{\delta}(l) \neq \mathbf{0}$ **then**
    Update $\boldsymbol{\Delta}$ with $\boldsymbol{\delta}(l)$.
    $\boldsymbol{a}(r) \leftarrow \hat{\boldsymbol{a}}(r),\ r = l, l{+}1,\ldots,T$.
    $f \leftarrow f + 1$.
  end if
  $l \leftarrow l + 1$.
end while

indicates the time slots at which a node i is capable of a new transmission, i.e., when its accumulated energy increases by more than or equal to $v_i(\alpha)$. Also, we denote the accumulated transmission opportunity matrix by $\boldsymbol{G}$ with $G_i(l)$ as its (i,l) entry. *Algorithm 4* calculates $\boldsymbol{B}$ and $\boldsymbol{G}$.

Algorithm 4 Finding the transmission opportunity matrices $\boldsymbol{B}$ and $\boldsymbol{G}$.

Input: $\boldsymbol{E}$.
Output: $\boldsymbol{B}$ and $\boldsymbol{G}$.

$\boldsymbol{B} \leftarrow 0$.
for $i = 1$ **to** n **do**
  for $r = 1$ **to** T **do**
    if $\sum_{l=1}^{r} E_i(l) - v_i(\alpha)\big(\sum_{l=1}^{r-1} B_i(l)\big) \geq v_i(\alpha)$ **then**
      $B_i(r) \leftarrow 1$.
    end if
    $G_i(r) \leftarrow \sum_{l=1}^{r} B_i(l)$.
  end for
end for

Second, at stage q, we schedule k nodes from the set of nodes with current transmission opportunity, $\mathcal{B}_q$, which is defined as

$$\mathcal{B}_q = \big\{i \in \mathcal{N} :\ B_i(T{-}q{+}1) = 1\big\}, \tag{10}$$

and its complement is denoted by $\mathcal{B}_q^c$. In case the number of nodes in the set $\mathcal{B}_q$ is $y{<}k$, we schedule the remaining $k-y$ nodes from the set $\mathcal{C}_q$ given by

$$\mathcal{C}_q = \arg\max\nolimits_{i\in\mathcal{B}_q^c}^{(k-y)} \big\{G_i(T{-}q{+}1)\big\}. \tag{11}$$

Finally, we update $\boldsymbol{B}$ and $\boldsymbol{G}$, by subtracting the allocated transmissions. The steps of the backward scheduling algorithm

Algorithm 5 The backward algorithm for finding a feasible power allocation to retrieve M files by time slot T.

Input: T, M, and $\boldsymbol{E}$.
Output: $\boldsymbol{\Delta}$.

$\boldsymbol{\Delta} \leftarrow 0$.
for $q = 1$ **to** M **do**
  if $\sum_{i=1}^{n} B_i(T{-}q{+}1) \geq k$ **then**
    $\Delta_{i_v}(T{-}q{+}1) \leftarrow 1$ for some $\{i_1,\ldots,i_k\} \subseteq \mathcal{B}_q$
  else
    $\Delta_i(T{-}q{+}1) \leftarrow 1, \forall\ i \in \mathcal{B}_q$.
    $y \leftarrow \sum_{i=1}^{n} B_i(T{-}q{+}1)$.
    $\mathcal{C}_q \leftarrow \arg\max\nolimits_{i\in\mathcal{B}_q^c}^{(k-y)} \big\{G_i(T{-}q{+}1)\big\}$.
    $\Delta_i(T{-}q{+}1) \leftarrow 1, \forall\ i \in \mathcal{C}_q$.
  end if
  for $i = 1$ **to** n **do**
    $r \leftarrow q$,
    $w_i \leftarrow \Delta_i(T{-}q{+}1)$.
    while $w_i > 0$ **do**
      if $B_i(T{-}r{+}1) > 0$ **then**
        $B_i(T{-}r{+}1) \leftarrow 0,\ \ w_i \leftarrow 0$.
      end if
      $r \leftarrow r + 1$.
    end while
  end for
  Update $\boldsymbol{G}$ with $\boldsymbol{B}$.
end for

are illustrated in *Algorithm 5*.

V. Numerical results

In this section, we present examples and numerical results to demonstrate the proposed algorithms. Example 1 illustrates the necessity and sufficiency of the conditions (3)-(5).

Example 1. *Consider the transmission of a file of size* $U = 2.4$ *Mbits under an* $(n = 4, k = 3, d = 3, \alpha = 0.8, \beta = 0.8)$ *minimum storage regeneration (MSR) code and* $v_i(\alpha){=}2, \forall i$. *The effective accumulated energy profiles of the storage nodes are given by*

$$\boldsymbol{A} = \begin{bmatrix} 2 & 4 & 4 & 4 \\ 1 & 2 & 4 & 4 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}. \tag{12}$$

1) *At* $l{=}1$, *none of the conditions (3)-(5) is satisfied.*
2) *At* $l{=}2$, *(3) is satisfied, while (4), (5) are not.*

$$\sum_{i=1}^{4} \Big\lfloor \frac{A_i(T)}{v_i(\alpha)} \Big\rfloor - \Big\lfloor \frac{A_1(T)}{v_1(\alpha)} \Big\rfloor = 1 < M(k{-}1) = 2,$$

$$\sum_{i\in\mathcal{S}_2} \Big\lfloor \frac{A_i(T)}{v_i(\alpha)} \Big\rfloor = 0 < M,\ \mathcal{S}_2 = \{3,4\}.$$

3) *At* $l{=}3$, *(3), (4) are satisfied, while (5) is not.*

$$\sum_{i\in\mathcal{S}_3} \Big\lfloor \frac{A_i(T)}{v_i(\alpha)} \Big\rfloor = 0 < M,\ \mathcal{S}_3 = \{3,4\}.$$

4) *At $l=4$, conditions (3)-(5) are all satisfied.*

Therefore, a feasible solution $\boldsymbol{P}$ to retrieve a file by $l=4$, is

$$\boldsymbol{P}=\begin{bmatrix}0&0&0&2\\0&0&0&2\\0&0&0&0\\0&0&0&2\end{bmatrix}. \tag{13}$$

■

Next, we present a scheduling example and solve it using the forward and backward algorithms.

Example 2. *Consider the transmission of $M=3$ files, each of size $U=1$ Mbits, in $T=5$ under an $(n=3,k=2,d=2,\alpha=0.5,\beta=0.5)$ code. Also, consider $v_i(\alpha)=1,\ \forall\, i$ and the following energy arrivals.*

$$\boldsymbol{E}=\begin{bmatrix}2&0&0&1&0\\0&0&0&1&0\\1&0&0&0&1\end{bmatrix},\ \boldsymbol{A}=\begin{bmatrix}1&2&2&3&3\\0&0&0&1&1\\1&1&1&1&2\end{bmatrix}. \tag{14}$$

The solutions obtained by the forward and backward algorithms are

$$\boldsymbol{P}_F=\begin{bmatrix}1&0&0&1&1\\0&0&0&1&0\\1&0&0&0&1\end{bmatrix},\ \boldsymbol{P}_B=\begin{bmatrix}0&0&1&1&1\\0&0&0&1&0\\0&0&1&0&1\end{bmatrix}. \tag{15}$$

■

We observe that in $\boldsymbol{P}_F$, the data collector retrieves the first file in the first time slot while in $\boldsymbol{P}_B$ the data collector retrieves the files in the last M time slots. Hence, the forward algorithm achieves a lower average delay per file.

Fig. 2 shows the average minimum retrieval (repair) time $\mathbb{E}[T]$ of M files, at the two extreme points on the storage and repair bandwidth trade-off curve [1], i.e., minimum storage regenerating (MSR) and minimum bandwidth regenerating (MBR) codes. In particular, we consider an $(n=10,k=5,d=9,\alpha=0.2,\beta=0.04)$ MSR and an $(n=10,k=5,d=9,\alpha=0.257,\beta=0.0268)$ MBR, for file size $U=1$ Mbits. Node i is assumed to have Binomial energy arrivals with maximum arrival $\theta=0.5$ energy units and harvesting rate h_i, i.e., $10\times E_i(t)\sim B(5,h_i)$, $\forall i$. Also, $v_i(b)=2^{2b}-1,\ \forall\, i$, where b is expressed in Mbits. From Fig. 2, we observe that for $h_i=0.1, i=1,...,8$ and $h_9=0.01$ retrieving the whole file requires less time on average compared with the repair process of node 10 at the two extreme points MSR and MBR.

VI. Conclusions

In this paper, we have investigated the operations of a distributed storage system with multiple files and energy harvesting nodes. We have formulated two optimization problems for the performance of the file retrieval and node repair operations. In order to solve these optimization problems, first we have characterized the necessary and sufficient energy feasibility conditions to retrieve M files by time slot T. Second, using these conditions, we have developed two algorithms

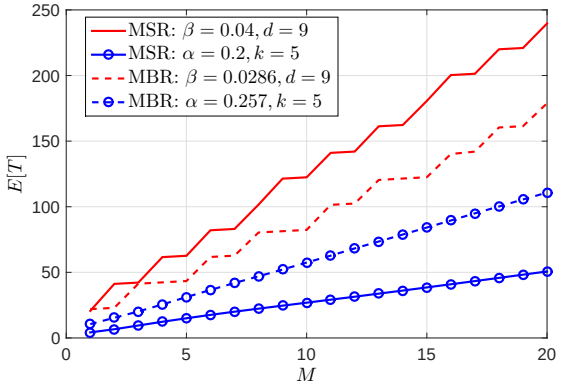

Fig. 2: The average minimum file retrieval (repair) time $\mathbb{E}[T]$ of M files for $h_i=0.1, i=1,...,8$ and $h_9=0.01$.

that reduce the optimization problems to a feasibility problem. We have proposed forward and backward algorithms to solve this feasibility problem. We have observed that a code that minimizes the repair bandwidth may not minimize the average repair time.

This work is a first attempt to model and investigate the dynamics of distributed storage systems under the intermittently available harvested energy. Many future directions remain and include online policies, file retrieval with priorities, and heterogeneous systems, e.g., files with different sizes or different encoding parameters.

References

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

[2] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[3] M. Gerami and M. Xiao, "Repair for distributed storage systems with erasure channels," in *IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 4058–4062.

[4] D. Karpuk, C. Hollanti, and A. Barreal, "Node repair for distributed storage systems over fading channels," in *International Symposium on Information Theory and its Applications (ISITA)*. IEEE, 2014, pp. 383–387.

[5] M. Gerami, M. Xiao, and M. Skoglund, "Optimal-cost repair in multi-hop distributed storage systems," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*. IEEE, 2011, pp. 1437–1441.

[6] S. Chen, Y. Sun, U. C. Kozat, L. Huang, P. Sinha, G. Liang, X. Liu, and N. B. Shroff, "When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds," in *Proceedings IEEE INFOCOM*. IEEE, 2014, pp. 1042–1050.

[7] J. Yang and S. Ulukus, "Optimal packet scheduling in an energy harvesting communication system," *IEEE Transactions on Communications*, vol. 60, no. 1, pp. 220–230, 2012.

[8] K. Tutuncuoglu and A. Yener, "Optimum transmission policies for battery limited energy harvesting nodes," *IEEE Transactions on Wireless Communications*, vol. 11, no. 3, pp. 1180–1189, 2012.

[9] Y. Zhang and N. Ansari, "Green data centers," *Handbook of Green Information and Communication Systems*, p. 331, 2012.

[10] J. B. Orlin, A. P. Punnen, and A. S. Schulz, "Integer programming: optimization and evaluation are equivalent," in *Algorithms and Data Structures*. Springer, 2009, pp. 519–529.