On Existence of Latency Optimal Uncoded Storage Schemes in Geo-Distributed Data Storage Systems

Srivathsa Acharya, P. Vijay Kumar
Department of Electrical Communication Engineering,
IISc., Bangalore
Email: {srivathsaa, pvk }@iisc.ac.in

Viveck R. Cadambe
Department of Electrical Engineering,
Pennsylvania State University, USA
Email: viveck@psu.edu

2024 IEEE International Symposium on Information Theory (ISIT) | 979-8-3503-8284-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIT57864.2024.10619184

***Abstract*—We consider the problem of geographically distributed data storage in a network of servers (or nodes) where the nodes are connected to each other via communication links having certain round-trip times (RTTs). Each node serves a specific set of clients, where a client can request for any of the files available in the distributed system. The parent node provides the requested file if available locally; else it contacts other nodes that have the data needed to retrieve the requested file. This inter-node communication incurs a delay resulting in a certain latency in servicing the data request. The worst-case latency incurred at a servicing node and the system average latency are important performance metrics of a storage system, which depend not only on inter-node RTTs, but also on how the data is stored across the nodes. Data files could be placed in the nodes as they are, i.e., in uncoded fashion, or can be coded and placed. This paper provides the necessary and sufficient conditions for the existence of uncoded storage schemes that are optimal in terms of both per-node worst-case latency and system average latency. In addition, the paper provides efficient binary storage codes for a specific case where optimal uncoded schemes do not exist.**

I. Introduction

Distributed data storage systems are an integral part of modern cloud-computing infrastructure. Over the last decade, coding theory has played an integral role in ensuring cost-effective fault-tolerance for distributed data storage systems, for e.g., through the development of regenerating codes [1], [2], locally repairable codes [3], [4] and codes with availability [5] (see [6], [7] for a survey). In this paper, we study a coding formulation that is relevant for geographically distributed (or *geo-distributed*) cloud storage systems where the data is replicated primarily to enable low latency data access to clients across a wide geographic area. In fact, most major commercial cloud storage providers including Google Cloud [8], Amazon AWS [9], and Microsoft Azure [10] offer support for geo-distributed data storage.

Geo-distributed cloud storage systems consist of nodes (data-centers/servers) connected to each other through links having certain round-trip delays. Each node serves a specific set of clients, where each client can request for any data available in the system. One of the desired features of geo-distributed storage systems is to provide wait-free or low-latency access to data. Providing wait-free access requires every file to be replicated at every node, which is inefficient in terms of storage utilization, and also infeasible when the storage requirement is comparable to the total storage capacity of the system. Given that total replication of files at nodes is not possible, several schemes based on *partial replication* have been proposed in the literature, where each node stores only a subset of the data (see [11], [12] and references therein), which we refer to as *uncoded storage schemes*.

In uncoded storage schemes where nodes store only a subset of the data, clients may have to fetch data from remote nodes, and thereby incur a data access latency of the inter-node round-trip-time (RTT)[1]. In this paper, we study two latency metrics that are relevant to practice. First, we consider the worst-case latency incurred over the system - the maximum round trip time required to fetch an object from a node for a given storage scheme. The second metric is the average latency (measured across nodes and files), which determines the average throughput of the system as per Little's law[2] [12].

Instead of storing copies of data files across nodes, one could also store functions of files (for e.g., linear combination of files) in the nodes, which we refer to as *coded storage*[3]. Coded storage can be beneficial (in terms of latency) over uncoded storage schemes in certain cases, while in others, uncoded schemes work best. Both cases are illustrated below.

Example 1. *Consider a data storage system with 4 nodes $\{A, B, C, D\}$, each capable of storing one file. Suppose the system has to store 3 information files $\{W_1, W_2, W_3\}$. The nodes along with inter-node RTTs are depicted in Fig. 1. The figure shows two possible ways of storing the information files on the nodes. In the first method, uncoded files are placed whereas the second method uses coded storage on node D which stores a coded file that is bit-wise XOR of the 3 files. Denote the contents of nodes as $\{X_A, X_B, X_C, X_D\}$. Table I shows the encoding and decoding of the 3 information files at each node. The resulting per-node worst-case latencies and system-average latency[4] are also provided in the table. We see that the coded scheme outperforms the uncoded scheme*

This research is supported by SERB Grant No. CRG/2021/008479 and NSF Grant #2211045.

[1]Assuming that the system is well-provisioned, RTTs between the nodes, which can be relatively large (tens to a few hundreds of ms, see Sec. V) are a dominant component of user data access latency.

[2]The *throughput* of a data store - the average number of client requests that can be served per second - is an important metric in data store design.

[3]Coded storage is known as *erasure coding* in the existing literature.

[4]The latency terms are formally defined in Section II.

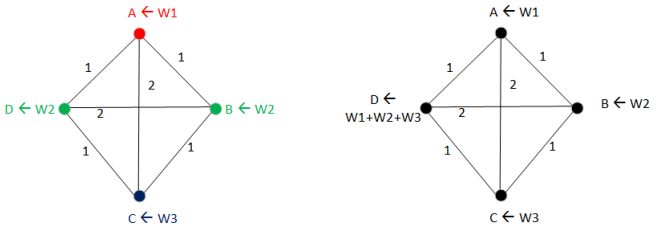


Fig. 1: Example 1: Data store with 4 nodes and 3 files with inter-node RTTs. Storage type - Left: Uncoded, Right: Coded.

TABLE I: Per-node worst-case latencies and system average latency for the coded and uncoded schemes shown in Fig. 1.

Scheme	Node/ Codeword $i(X_i)$	Decoding	Worst-case Latency $L^{(i)}_{max}$	Average Latency L_{avg}
Uncoded	$A(W_1)$		2	$\frac{10}{12}$
	$B(W_2)$		1	
	$C(W_3)$		2	
	$D(W_2)$		1	
Coded	$A(W_1)$	$W_3 = X_D \oplus X_A \oplus X_B$	1	$\frac{9}{12}$
	$B(W_2)$		1	
	$C(W_3)$	$W_1 = X_D \oplus X_C \oplus X_B$	1	
	$D(\oplus_{i=1}^{3} W_i)$	$W_2 = X_D \oplus X_A \oplus X_C$	1	

in both the latency metrics. It can be further verified that the coded scheme has the least latency over all uncoded schemes.

Example 2. *Consider again a 4-node 3-file system as in Example 1, but with different inter-node RTTs as given in Fig. 2. An uncoded storage scheme is also shown in the figure, with which it is possible for each node to obtain all the 3 files by contacting only its 2 least RTT neighbors. This results in minimum per-node worst-case latency and average latency which no coded storage scheme can beat.*

Thus, it is useful to know the class of storage systems where an uncoded scheme itself gives optimal worst-case and average latency. This is the focus of current paper. It is notable that in computer systems and performance analysis literature, there are several works that aim to optimize data placement in geo-distributed data storage systems by utilizing knowledge of inter-node RTTs [13]–[21]. These works develop optimization frameworks and solutions for data/codeword placement based on latency, communication cost, storage budget, and fault-tolerance requirements. However, even for the simpler objective of minimizing average and worst-case latencies, the best strategies are not known. In particular, for a given storage budget and worst-case latency, it is unclear when uncoded strategies obtain optimal average latency, or how erasure codes should be designed to minimize average latency. There are also works which provide latency analsysis based on MDS storage (as in [21], [22]), but as will be shown later, MDS codes are not suited well for average latency constraints. Notably, different from classical erasure codes, the erasure codes must be designed and codeword symbols must be placed on the nodes based on the RTTs to minimize latency. This paper makes progress on these problems.

The rest of the paper is organized as follows. Section II develops the system-model, provides formal definitions of storage codes and associated latencies. Section III gives the main contribution of the paper where the problem of optimal uncoded storage is converted to one of *vertex coloring* on a special subgraph called the *nearest-neighbor graph*, and provides necessary and sufficient condition for an optimal uncoded scheme to exist. Section IV provides coded storage schemes for some specific cases where optimal uncoded schemes do not exist. Section V gives an application of the main result on a hypothetical geo-distributed data-center network. The paper concludes with possible research directions for future in SectionVI.

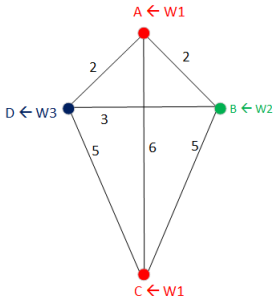


Fig. 2: Example 2: Data store with n = 4 nodes $\{A, B, C, D\}$, k =3 files $\{W_1, W_2, W_3\}$, and their inter-node RTTs.

II. System Model and Problem formulation

We model the data storage network of n servers and $k \leq n$ files by an undirected weighted complete[5] graph $\mathcal{G} = (\mathcal{N}, T)$, where $\mathcal{N} = \{1, 2, \ldots, n\}$ denote the n nodes, and $T = \{\tau(i,j) : 1 \leq i, j \leq n\}$ is the edge-weight matrix, representing the RTTs between a pair of nodes. RTT is same in either direction, i.e., $\tau(i,j) = \tau(j,i) \quad \forall (i,j) \in \mathcal{N} \times \mathcal{N}$. Also $\tau(i,i) = 0 \quad \forall i \in \mathcal{N}$. Let $W_1, \ldots W_k$ denote the k information files (each of unit file-size) to be stored in the storage network. Each node has capacity to store data worth 1 file-size [6]. We denote $[k] = \{1, 2, \ldots, k\}$ to represent the file index set. Let $X_1, \ldots X_n$ denote the data stored in each of the n nodes.

Note that the point-to-point single-hop model above can also be applied to a more general multi-hop scenario, where the communication between two nodes (i,j) traverses intermediate nodes. In this case, the sum total of RTTs of the links along the least RTT path between the nodes is taken as the equivalent edge weight $\tau(i,j)$ in our model.

A linear storage code with sub-packetization α can be defined as follows. Assume that each information file $W_j; j \in [k]$ can be split into α sub-packets $(W_{j1}, W_{j2}, \ldots, W_{j\alpha})$, with each sub-packet belonging to a finite field $\mathcal{F}$. Similarly, a file stored in node $i \in \mathcal{N}$, X_i, is composed of α sub-packets $(X_{i1}, X_{i2}, \ldots, X_{i\alpha})$, also belonging to $\mathcal{F}$. Denote

[5]A complete graph is one where an edge exists between every pair of nodes.

[6]In a general setting, each of the n servers can store $M \geq 1$ files, and the requirement is to store $kM(\leq nM)$ information files in the network. The paper addresses $M = 1$ case. The storage codes thus obtained can be extended to $M > 1$ case by partitioning each of the node contents into M *stripes* and then applying the code on each stripe.

$\underline{\mathrm{X}} = (X_{11}, X_{12}, \ldots, X_{1\alpha}, \ldots, X_{n1}, X_{n2}, \ldots, X_{n\alpha})^T$
$\underline{\mathrm{W}} = (W_{11}, W_{12}, \ldots, W_{1\alpha}, \ldots, W_{k1}, X_{k2}, \ldots, W_{k\alpha})^T$.

Definition 1 (Linear storage code)**.** *A linear storage code $\mathcal{C}$ on $\mathcal{G}$ is defined by a $(k\alpha \times n\alpha)$ generator matrix G such that*

$$\underline{X}^T = \underline{W}^T G \tag{1}$$

The column j in G comprises the linear weights of each of the $k\alpha$ sub-packets of $\underline{\mathrm{W}}$ that combine to make the jth coded sub-packet in $\underline{\mathrm{X}}$. We only consider the codes with $rank(G) = \alpha k$, the condition necessary for decoding all information files from the coded files.

An *uncoded storage scheme* is a special case of linear storage codes where there is no sub-packetization ($\alpha = 1$) and no coding across the files.

A. Average Latency and Per-node Worst-case Latency

Given a code $\mathcal{C}$ on $\mathcal{G}$, the *decoding process* and associated *latencies* are formally defined in Appendix F of the extended paper [23]. Here, we provide an intuitive definition of the latency as follows. For a certain wait-time L, a node i has access to the contents of those nodes t whose RTT satisfies $\tau(t,i) \leq L$. Using the contents from these nodes, certain raw files can be decoded. *Latency* at node i to decode a file W_j, denoted as $l_j^{(i)}$, is defined as the minimum wait-time L at node i needed to decode file W_j[7].

Per-node worst-case latency of code $\mathcal{C}$ at node i is defined as

$$L_{max}^{(i)}(\mathcal{C}) = \max_{j\in[k]} l_j^{(i)} \tag{2}$$

Average latency of code $\mathcal{C}$ is defined as

$$L_{avg}(\mathcal{C}) = \frac{1}{kn} \sum_{i\in\mathcal{N}} \sum_{j\in[k]} l_j^{(i)} \tag{3}$$

Given a node i, let $\left(\lambda_0^{(i)} \leq \lambda_1^{(i)} \leq \cdots \leq \lambda_{(n-1)}^{(i)}\right)$ be the sorted list of RTTs to node i, i.e., $\left(\tau(j,i) : j \in \mathcal{N}\right)$, in ascending order. That is, $\lambda_m^{(i)}$ is the m^{th} least RTT value to node i from other nodes. By definition, $\lambda_0^{(i)} = \tau(i,i) = 0$.

Proposition 1. *For any code $\mathcal{C}$ on $\mathcal{G}$, per-node worst-case latency at any node i is lower-bounded as:*

$$L_{max}^{(i)}(\mathcal{C}) \geq \lambda_{(k-1)}^{(i)} \tag{4}$$

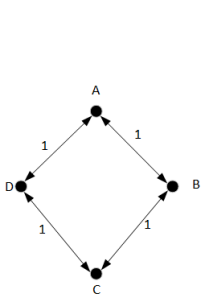
Further, the average latency $L_{avg}(\mathcal{C})$ is lower-bounded as:

$$L_{avg}(\mathcal{C}) \geq \frac{1}{kn} \sum_{i\in\mathcal{N}} \sum_{j\in[k]} \lambda_j^{(i)} \tag{5}$$

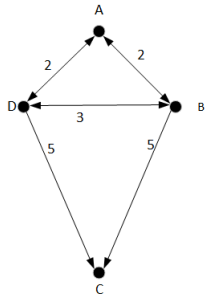
Proof. See Appendix A of the extended paper [23]. □

Definition 2. *Given a directed subgraph $\mathcal{D}$ of $\mathcal{G}$ with the same node set, a code $\mathcal{C}$ is said to be* admissible *on $\mathcal{D}$ if the decoding of any file at any node involves file transfers only along the directed edges of $\mathcal{D}$.*

[7]Strictly speaking, as users/clients request files from nodes, the latency should also include the delay between a user and its local node. But this delay can be neglected since it is dominated by inter-node RTT (see [14], [20]), and since the delay remains same for any choice of the storage code, thus not affecting the storage optimization.



(a) Example 1



(b) Example 2

Fig. 3: Nearest-neighbor graphs $\mathcal{G}_2$ for Section I examples.

III. Main Result

In this paper, we consider only the codes meeting the worst-case latency optimality constraint (4). Satisfying this constraint are the codes admissible on a special subgraph called the *nearest-neighbor graph*.

Definition 3. *A* nearest-neighbor graph *$\mathcal{G}_{k-1}$ is defined as a directed subgraph of $\mathcal{G}$ where each node i has incoming edges from $(k-1)$ other nodes having $(k-1)$ least RTT values to node i.*

Remark 1.

- *The incoming edges to a node i in $\mathcal{G}_{k-1}$ have, in ascending order, the weights $\lambda_1^{(i)}, \ldots, \lambda_{(k-1)}^{(i)}$.*
- *In this paper, we refer to **neighbors** of a node i in $\mathcal{G}_{k-1}$ as only the $(k-1)$ nodes from which there are incoming edges to node i. But, a node connected only via an outgoing edge from node i is **not** referred as its* neighbor *in $\mathcal{G}_{k-1}$.*
- *Multiple $\mathcal{G}_{k-1}$ are possible when multiple nodes share the same RTT value of $\lambda_{(k-1)}^{(i)}$ to a node i.*

$\mathcal{G}_{k-1}$ for Examples 1 and 2 of Section I are shown in Fig. 3. Note that there always exist codes that are admissible on $\mathcal{G}_{k-1}$, such as the MDS (Maximum Distance Separable) codes given below.

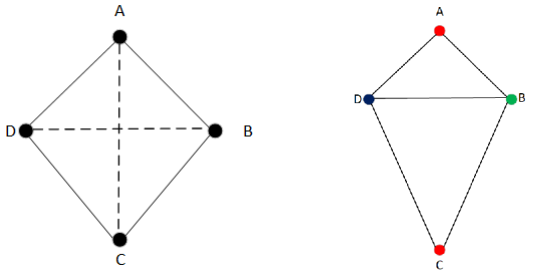
Example 3 (Scalar MDS Codes)**.** *Let $\alpha = 1$(no sub-packetization)* [8] *and G be a $k \times n$ matrix of a MDS code. From k information files, let n coded files be generated with this MDS matrix G as in* (1)*, and place one coded file at each node. Due to MDS property, any information file can be recovered at a given node if there are k coded files, which can be obtained by using the local data at the node and by contacting $(k-1)$ least RTT nodes. Hence this code is admissible on a $\mathcal{G}_{k-1}$.*

The reason for looking into admissible codes on $\mathcal{G}_{k-1}$ is because of their latency optimality properties as shown next.

Proposition 2. *Any admissible storage scheme $\mathcal{C}$ on $\mathcal{G}_{k-1}$ meets the per-node worst-case latency bound in* (4)*, i.e,*

$$L_{max}^{(i)}(\mathcal{C}) = \lambda_{(k-1)}^{(i)} \tag{6}$$

[8]*Storage code without sub-packetization* is known as *Cross-object erasure coding* in [11]. MDS codes with sub-packetization also exist (see [12]).



(a) Example 1 (b) Example 2

Fig. 4: Extended graphs for the examples in Section I.The dashed edges are those added on top of $\mathcal{G}_2$.

Proof. A given node i can only use the links available in $\mathcal{G}_{k-1}$, and the maximum weight of its incoming edges is $\lambda^{(i)}_{(k-1)}$. □

Even though any admissible code (such as MDS code) on $\mathcal{G}_{k-1}$ is worst-case latency optimal, it need not be average-latency optimal. However, if an **uncoded** admissible code exists on a $\mathcal{G}_{k-1}$, it is both worst-case latency and average-latency optimal as given below.

Proposition 3. *If there is an admissible **uncoded** scheme $\mathcal{C}$ on $\mathcal{G}_{k-1}$, then it meets the average latency lower bound in* (5)*:*

$$L_{avg}(\mathcal{C}) = \frac{1}{kn} \sum_{i \in \mathcal{N}} \sum_{j \in [k]} \lambda^{(i)}_j \tag{7}$$

Conversely, an optimal uncoded scheme in the original complete graph $\mathcal{G}$ that meets the latency bounds of (4) *and* (5) *exists only if it is admissible on* some $\mathcal{G}_{k-1}$ *of* $\mathcal{G}$*.*

Proof. See Appendix B of the extended paper [23] for proof of (7). The converse can be seen from the fact that every node is forced to communicate with precisely $(k-1)$ least latency neighbors in order to meet both the latency bounds. □

We thus look for existence of admissible uncoded storage schemes on $\mathcal{G}_{k-1}$. For this, we first convert the problem into that of vertex coloring [24] on a related undirected graph called the *extended graph*.

Definition 4. *Given a nearest-neighbor graph $\mathcal{G}_{k-1}$, its extended graph $\mathcal{H}$ is defined as an undirected graph on same node set formed by the following rules.*

- *If 2 nodes are connected by a (directed) edge in $\mathcal{G}_{k-1}$, connect them by an (undirected) edge in $\mathcal{H}$.*
- *If 2 nodes are neighbors*[9] *of same node in $\mathcal{G}_{k-1}$, then also connect them by an edge in $\mathcal{H}$.*

Fig. 4 shows the extended graphs for the examples of Section I.

Vertex coloring of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a map $\rho : \mathcal{N} \to S$ such that $\rho(v) \neq \rho(w)$ whenever v and w are adjacent. The elements of set S are called the *colors*. The smallest size of set S with which G can be vertex colored is known as its *chromatic number* denoted by $\chi(\mathcal{G})$. One result that we use is that if $\mathcal{G}$ has a complete subgraph of m nodes, then $\chi(\mathcal{G}) \geq m$.

Theorem III.1 (Vertex Coloring)**.** *An admissible uncoded storage scheme exists on $\mathcal{G}_{k-1}$ if and only if the corresponding extended graph $\mathcal{H}$ has chromatic number $\chi(\mathcal{H}) = k$.*

Proof. By associating each file with distinct color, result is obtained. A detailed proof is in Appendix C of the extended paper [23]. □

Thus, the theorem along with Proposition 3 implies that a latency optimal uncoded scheme exists on the original graph $\mathcal{G}$ if and only if $\mathcal{H}$ of some $\mathcal{G}_{k-1}$ is k-colorable. Let us apply this result to $(n, k) = (4, 3)$ systems of Section I.

- **Example 1:** Extended graph is itself a complete graph of 4 nodes, and hence it needs $4 (> k)$ colors. So, from Theorem III.1, no latency optimal uncoded scheme exists, which reinforces the observation in Section I.
- **Example 2:** By assigning same color to the non-adjacent nodes A and C (in $\mathcal{H}$), $k = 3$ coloring is possible as shown in Fig. 4 . Thus, an optimal uncoded scheme exists as shown in Section I.

One consequence of the theorem is for special case of $k = 2$.

Corollary 1. *For any data storage system $\mathcal{G}$ with $k = 2$, there always exists an optimal uncoded scheme that meets the latency bounds of* (4) *and* (5)*.*

Proof. See Appendix D of the extended paper [23]. □

IV. Coded Storage Schemes on $\mathcal{G}_{k-1}$

We next look at networks where Theorem III.1 does not hold on any nearest-neighbor graph $\mathcal{G}_{k-1}$. As no optimal uncoded scheme exists, we need to look for coded schemes on $\mathcal{G}_{k-1}$ that are average latency optimal. This is an open problem. However, as a byproduct of Theorem III.1, we provide a family of admissible binary codes on $\mathcal{G}_{k-1}$ for the special case of $\chi(\mathcal{H}) = (k+1)$ as described below.

Consider a data storage network where an extended graph has $\chi(\mathcal{H}) = (k+1)$, i.e., $\mathcal{H}$ needs one more color than the no. of files. By replacing k of these colors by files, and then converting an extra color to an appropriate linear combination of files, it is possible to get an admissible coding scheme on $\mathcal{G}_{k-1}$, as described next.

$\chi(\mathcal{H}) = (k+1)$ implies a valid $(k+1)$ vertex-coloring map: $\rho : \mathcal{N} \to S := \{c_1, c_2, \ldots, c_{k+1}\}$. Associate some k of the $(k+1)$ colors directly with k file indices , and mark the remaining color as *coded*. That is, form a bijective function $f : S \to \{1, 2, \ldots, k, *\}$ where $*$ represents a coded color.

A. Encoding Algorithm at each node $i \in \mathcal{N}$

If i is mapped to an *uncoded* color, i.e, $f(\rho(i)) \in [k]$, then the file assignment is $X_i = W_{f(\rho(i))}$. On the other hand, if i is mapped to a coded color $f(\rho(i)) = *$, then:

- Let $\mathcal{R}(i) = \{j \in \mathcal{N} : (i, j) \in \mathcal{G}_{k-1}\}$ be the set of nodes that are adjacent to node i via outgoing edges from i in $\mathcal{G}_{k-1}$ (called as *receive* nodes).

[9] see Remark 1 for definition of neighbor in $\mathcal{G}_{k-1}$

Regions	Seoul	Mumbai	Ireland	London	California	Oregon
Seoul	0	120	230	240	138	126
Mumbai	120	0	121	113	228	220
Ireland	230	121	0	13	138	126
London	240	113	13	0	146	137
California	138	138	230	146	0	22
Oregon	126	220	126	137	22	0

Fig. 5: A sample data store with 6 nodes and their inter-node RTTs (in ms) measured as per AWS public cloud [11] [25].

- For each receive node $r \in \mathcal{R}(i)$, identify the index of the missing file $\mu(r)$ as follows. The node r and all of its $(k-1)$ neighbors in $\mathcal{G}_{k-1}$ except node i have uncoded colors. This is because these nodes are adjacent to i in $\mathcal{H}$ and hence cannot share the same color as i. Therefore, r has access to some $(k-1)$ uncoded files from its $(k-2)$ uncoded neighbors and itself. Hence, there is precisely one file which is not available with r that it wishes to get from i. Denote the index of this missing file as $\mu(r)$
- For node i, its $(k-1)$ neighbors in $\mathcal{G}_{k-1}$ have distinct, uncoded colors due to valid vertex coloring. Hence i has a missing file which needs to be provided by itself. Denote this missing file index as $\mu(i)$
- Assign the sum (bitwise-XOR) of missing files to node i as:

$$X_i = \sum_{f \in S} W_f \ , S := \left\{\mu(r) : r \in \{i\} \cup \mathcal{R}(i)\right\} \quad (8)$$

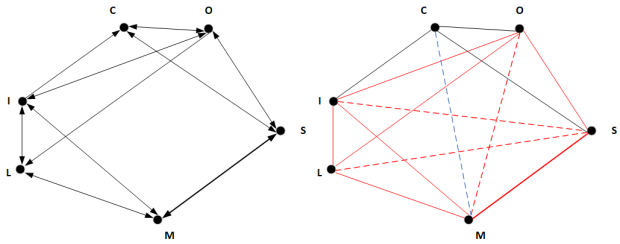
By construction, the above code is admissible as every node has at most one missing uncoded file, which can be obtained from its coded neighbor. For clarity, a *decoding* algorithm has been added in Appendix E of the extended paper [23].

Using the above algorithm, we can get multiple admissible codes, one for each choice of vertex coloring on $\mathcal{H}$ (unique up to color permutation), and for each choice of coded color. It is not known whether this family of codes contains a system average-latency optimal code on $\mathcal{G}_{k-1}$. Nevertheless, the codes are attractive from implementation perspective since they are worst-case latency optimal, binary-coded, and have just enough file additions to make the code admissible.

V. Application to a Data-Storage System

We illustrate the efficacy of Theorem III.1 on a sample geo-distributed data-center network of 6 nodes as shown in Fig. 5. The inter-node RTTs are taken from measurements as per Amazon AWS public cloud [11] [25]. Consider $k = 4$ files.

Non-existence of Optimal Uncoded Scheme: For $(n, k) = (6, 4)$, there is a unique nearest-neighbor graph $\mathcal{G}_3$ as shown in Fig. 6. The figure also shows the extended Graph $\mathcal{H}$. It can be



(a) (Unique) $\mathcal{G}_3$ (b) The extended Graph $\mathcal{H}$.

Fig. 6: AWS data-store (cities replaced by initials) for $k = 4$.

TABLE II: AWS data store with $(n, k) = (6, 4)$: A Binary-code obtained using 5-vertex coloring of $\mathcal{H}$. *: coded color.

Node Index i	Vertex Color $\rho(i)$	Code X_i	Decoding	File recovery latency (ms) W_1	W_2	W_3	W_4
(S)	$^*c_1$	$W_1+W_2+W_4$	$W_2 = X_S + X_M + X_O$	120	126	138	126
(M)	c_2	W_1	$W_4 = X_S + X_M + X_I$	0	121	113	121
(I)	c_3	W_2		121	0	13	126
(L)	c_4	W_3		113	13	0	137
(C)	c_4	W_3	$W_1 = X_S + X_I + X_O$	138	138	0	22
(O)	c_5	W_4	$W_1 = X_S + X_I + X_O$	126	126	22	0

seen that the induced sub-graph among nodes S, L, I, M, O is a complete graph $\mathcal{K}_5$, and hence $k = 4$ coloring is not possible. Thus, as per Theorem III.1 and Proposition 3, no optimal uncoded scheme exists (in terms of both average and per-node worst-case latency).

Admissible Binary Codes: From Fig. 6, we can see that $\mathcal{H}$ can be 5- colored by assigning same color to C and L. Since $\chi(\mathcal{H}) = (k+1)$, we can obtain admissible binary codes as described in Section IV.

Let us mark c_1 as the coded color and associate rest of the colors directly with uncoded files as given in Table II. Now, c_1 is mapped to node S(Seoul) which is a neigbhor of nodes M, C, and O in $\mathcal{G}_3$. The node S has a missing file W_2 while node M needs W_4 and nodes C, O both need W_1 from node S. Hence the codeword on node S as per (8) is $X_1 = W_1 \oplus W_2 \oplus W_4$. The decoding equations at each node and the corresponding latencies are also tabulated in Table II. The resulting average latency is $L_{avg}(\mathcal{C}) = 81.67$ *ms*, compared to the non-achievable lower bound of 76.37 *ms*.

VI. Conclusion and Future Work

We introduced the problem of latency optimal storage schemes on a geo-distributed data network having certain inter-node round-trip times. By modeling the storage network as a weighted complete graph, we showed that a latency optimal uncoded storage exists if and only if it is admissible on a subgraph called the nearest-neighbor graph. We then obtained vertex-coloring based condition for such an optimal uncoded scheme to exist. In the networks where the vertex coloring condition fails, our result provides justification for employing coded storage. Finding optimal codes for such networks is an open problem and is the direction of our future work.

Acknowledgment

We thank Dr. Sridhar Ramesh, MaxLinear Inc. for helpful comments on this work.

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] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *2009 IEEE International Symposium on Information Theory*. IEEE, 2009, pp. 2276–2280.

[3] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Transactions on Information theory*, vol. 58, no. 11, pp. 6925–6934, 2012.

[4] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4661–4676, 2014.

[5] A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," *IEEE Transactions on Information Theory*, vol. 62, no. 8, pp. 4481–4493, 2016.

[6] S. 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, 2018.

[7] V. Ramkumar, S. Balaji, B. Sasidharan, M. Vajha, M. N. Krishnan, and P. V. vk Kumar, "Codes for distributed storage," *Foundations and Trends in Communication and Information Theory*, vol. 19, no. 4, 2022.

[8] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, and W. Hsieh, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.

[9] Https://aws.amazon.com/rds/aurora/global-database/.

[10] Https://learn.microsoft.com/en-us/azure/cosmos-db/distribute-data-globally.

[11] V. R. Cadambe and S. Lyu, "Brief announcement: CausalEC: A causally consistent data storage algorithm based on cross-object erasure coding," in *2023 ACM Symposium on Principles of Distributed Computing*, 2023, pp. 374–377.

[12] ——, "CausalEC: A causally consistent data storage algorithm based on cross-object erasure coding," 2021, arXiv:2102.13310.

[13] M. S. Ardekani and D. B. Terry, "A Self-Configurable Geo-Replicated Cloud Storage System," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. Broomfield, CO: USENIX Association, Oct. 2014, pp. 367–381. [Online]. Available: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/ardekani

[14] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 292–308. [Online]. Available: https://doi.org/10.1145/2517349.2522730

[15] A. Jonathan, M. Uluyol, A. Chandra, and J. Weissman, "Ensuring reliability in geo-distributed edge cloud," in *2017 Resilience Week (RWS)*. Wilmington, DE, USA: IEEE, Sep. 2017, pp. 127–132.

[16] P. Shankaranarayanan, A. Sivakumar, S. Rao, and M. Tawarmalani, "Performance sensitive replication in geo-distributed cloud datastores," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Atlanta, GA, USA: IEEE, 2014, pp. 240–251.

[17] M. Abebe, K. Daudjee, B. Glasbergen, and Y. Tian, "Ec-store: Bridging the gap between storage and latency in distributed erasure coded systems," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2018, pp. 255–266. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICDCS.2018.00034

[18] M. Su, L. Zhang, Y. Wu, K. Chen, and K. Li, "Systematic data placement optimization in multi-cloud storage for complex requirements," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1964–1977, 2016.

[19] J. Matt, P. Waibel, and S. Schulte, "Cost- and latency-efficient redundant data storage in the cloud," in *2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA)*. Kanazawa, Japan: IEEE, Nov 2017, pp. 164–172.

[20] H. Zare, V. R. Cadambe, B. Urgaonkar, N. Alfares, P. Soni, C. Sharma, and A. A. Merchant, "Legostore: a linearizable geo-distributed store combining replication and erasure coding," *Proceedings of the VLDB Endowment*, vol. 15, no. 10, pp. 2201–2215, 2022.

[21] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.

[22] K. Lee, N. B. Shah, L. Huang, and K. Ramchandran, "The mds queue: Analysing the latency performance of erasure codes," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 2822–2842, 2017.

[23] S. Acharya, P. V. Kumar, and V. R. Cadambe, "On existence of latency optimal uncoded storage schemes in geo-distributed data storage systems," 2024, arXiv:2405.06641.

[24] R. Diestel, *Graph Theory*, 5th ed. Springer books, 2004.

[25] M. Leonhard, "Cloudping.info," 2017. [Online]. Available: https://www.cloudping.info/