Linking Policies for the Permanent Web

Patrick Le
Drexel University
Philadelphia, USA
ptl46@drexel.edu

Quang Bui
Drexel University
Philadelphia, USA
qb42@drexel.edu

Alexander Grigorian
Drexel University
Philadelphia, USA
asg325@drexel.edu

Alexey Kuraev
Drexel University
Philadelphia, USA
ak4249@drexel.edu

Thiyazan Salman
Drexel University
Philadelphia, USA
ts3375@drexel.edu

Tu H. Nguyen
Drexel University
Philadelphia, USA
nguyenhoangtu.427@gmail.com

Mat Kelly
Drexel University
Philadelphia, USA
mkelly@drexel.edu

Sawood Alam
Internet Archive
San Francisco, USA
sawood@archive.org

***Abstract*—Earlier approaches such as InterPlanetary Wayback (IPWB) demonstrated how archived content could be stored in the InterPlanetary File System (IPFS) but still required a centralized index to resolve resources. We evaluate the feasibility of the InterPlanetary Archival Record Object (IPARO), a framework proposed to eliminate the dependency on local indexes in decentralized web archiving. IPARO addresses this limitation by embedding version-linked references directly into archival objects stored in IPFS and using the InterPlanetary Name System (IPNS) for discovery of the latest versions. This design enables traversal of a resource's version history entirely within the decentralized network. In this paper, we simulate the construction and resolution of IPARO chains to assess their practicality for replaying archived Web content without centralized infrastructure. Our evaluation examines integrity and availability under various conditions. Results show that IPARO retains the benefits of distributed storage while eliminating reliance on local indexes, offering a scalable and robust model for decentralized preservation.**

***Index Terms*—IPARO, IPFS, Decentralized Web, DWeb, Web Archiving, Evaluations**

I. Introduction

The InterPlanetary File System (IPFS) [1], [2] is a peer-to-peer content-addressable storage network that facilitates the distribution and persistence of data across decentralized nodes. This architecture offers desirable properties for long-term digital preservation, including inherent deduplication, cryptographic integrity, and elimination of single failure points. However, despite these advantages, existing web archives continue to rely on centralized infrastructure for indexing and retrieval, exposing them to risks associated with organizational continuity, funding constraints, and regional or political vulnerabilities. These challenges motivate the exploration of fully decentralized approaches to web archiving that do not depend on centralized repositories or index servers.

Building on this motivation, prior work has begun to explore decentralized web archiving, most notably with InterPlanetary Wayback (IPWB) [3]–[6] and, more recently, IPARO [7]. While IPWB demonstrated the feasibility of storing archival content in IPFS, it still depended on local indexes. IPARO was proposed to address this limitation by embedding version-linked references directly in the IPFS objects and using IPNS for discovery.

In this paper, we extend that prior work by simulating the construction and traversal of IPARO chains and evaluating their performance. Specifically, we examine the practicality of decentralized archival replay without local indexes by measuring the space and time costs under various linking strategies. In doing so, we provide a clearer understanding of the trade-offs and potential of IPARO for real-world deployment.

II. Background & Related Work

Decentralized approaches to web archiving are made feasible by recent advances in peer-to-peer storage and naming systems. In this section, we review the foundational technologies that underpin our work, including IPFS and IPNS, followed by prior efforts to decentralized web archiving via InterPlanetary Wayback (IPWB), and finally the recent design of InterPlanetary Archival Record Objects (IPAROs), upon which our work builds.

A. InterPlanetary File System (IPFS)

The InterPlanetary File System (IPFS) [1], [2] is a distributed content-addressable file system that identifies data using cryptographic hashes rather than a path or location of files. In contrast to traditional web infrastructure, where resources are retrieved based on their location (e.g., via DNS and HTTP), IPFS retrieves data by its content identifier (CID), ensuring immutability and verifiability. Data stored in IPFS are chunked, hashed, and stored in a Merkle Directed Acyclic Graph (DAG), allowing for deduplication, integrity checks, and efficient replication across nodes, making IPFS a compelling foundation for building resilient and distributed archival systems.

B. InterPlanetary Name System (IPNS)

The InterPlanetary Name System (IPNS) [8], [9] is a complementary protocol to IPFS that allows mutable, persistent naming over an immutable content-addressed network. Although IPFS CIDs are immutable, IPNS enables users to publish references to content that can be updated over time.

IPNS names are tied to cryptographic key pairs, and the most recent version of the content is published to a Distributed Hash Table (DHT) under the corresponding public key. This design allows IPNS to act as a mutable pointer mechanism rather than a source of human-meaningful identifiers, enabling the resolution of evolving resources while preserving the integrity guarantees of IPFS.

Unlike the Domain Name System (DNS) [10], which is hierarchical and centrally administered, IPNS is inherently flat and decentralized. In the context of web archiving, this allows IPNS to serve as an alternative to centralized index servers by enabling dynamic resolution of version chains or resource heads, maintained entirely through peer-to-peer infrastructure.

C. InterPlanetary Wayback (IPWB)

InterPlanetary Wayback (IPWB) [3]–[6] was an early prototype that explored the feasibility of integrating web archives with IPFS. In IPWB, HTTP response payloads were extracted from WARC [11] files and stored in IPFS, while a local CDXJ [12], [13] index was created to associate each URI and datetime pair with the corresponding IPFS hashes. IPWB extended the pywb replay system to retrieve content from IPFS during playback.

Although IPWB demonstrated that IPFS could be used to distribute and preserve archival content in a peer-to-peer manner, the system remained dependent on a centralized or local index for resource discovery and version navigation. This limitation constrained its decentralization and left the resolution process dependent on the availability of the index.

D. InterPlanetary Archival Record Object (IPARO)

The InterPlanetary Archival Record Object (IPARO) [7] was proposed to address the primary limitation of IPWB by eliminating the dependency on local or centralized indexes. Instead of external mapping, each IPARO embeds a set of metadata fields-including URI, timestamp, and references to prior versions-directly into the object stored in IPFS. These IPAROs form a resilient linked list [14] that can be traversed to reconstruct the version history of a resource.

As illustrated in Figure 1, each IPARO consists of a header with metadata, a payload that stores the archival content (e.g., a WARC record), and an optional nonce used to adjust the resulting CID. By embedding references to prior versions, IPAROs enable traversal across time without reliance on an external index.

By combining this version-linked structure with IPNS, which maintains a pointer to the latest IPARO for a resource, the system enables decentralized discovery and replay of archived content entirely through the IPFS network. This approach parallels other distributed, append-only data structures such as the Bamboo protocol [15] and contributes to the broader literature on resilient decentralized systems [16].

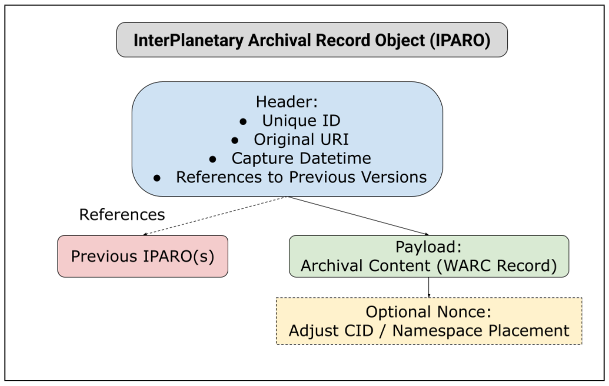

InterPlanetary Archival Record Object (IPARO)
Header:
• Unique ID
• Original URI
• Capture Datetime
• References to Previous Versions
References
Previous IPARO(s)
Payload:
Archival Content (WARC Record)
Optional Nonce:
Adjust CID / Namespace Placement

Fig. 1: Structure of an InterPlanetary Archival Record Object (IPARO). Each object includes a header, payload, and optional nonce, with version-linked references embedded for traversal.

III. Methodology

The IPARO Simulation Application[1] is a tool that is used to compare linking strategies in different environments. The setup uses Bash scripts to run the simulations concurrently on a Windows 11 PC with 32 GB of RAM allocated to the Windows Subsystem for Linux (WSL) [17] and takes about 10 hours to run on 8 cores.

A. Metrics

There are two types of costs that we are measuring. The first cost is the storage cost. The storage cost, in bytes, are calculated as follows:

$$S = b \cdot L + H \tag{1}$$

In Equation (1), S is the storage cost, L is the average number of links, b is the number of bytes per link, and H is the overhead storage cost.

The second cost is the time cost, which is described by the following equation:

$$T = t_N N + t_F F + t_W W + t_U U + C \tag{2}$$

As described in Equation (2), C stands for overhead cost, N represents the average number of IPNS reads, F represents the number of IPFS reads, W is the average number of IPFS writes, and U is the average number of IPNS updates. In addition, the variables t_N, t_F, t_W, and t_U describe the mean time it takes for each respective operation.

$$\bar{T} = \frac{1}{k}\sum_{i=1}^{k} T_k \tag{3}$$

$$\bar{S} = \frac{1}{k}\sum_{i=1}^{k} S_k \tag{4}$$

[1]https://github.com/johnnguyenn77/iparo

Equations 3 and 4 represent the average costs for time and storage operations. We use 10 samples for each operation on a chain length of 100. Equation 4 is applicable when dealing with temporal policies, as the number of links may vary between runs. Additionally, since we are only dealing with a specific collection of environments in Sections IV-B and IV-C, the results are not necessarily generalizable to other environments and other distributions.

B. Simulation Testing Environment

The testing environments used in the simulation have three parameters: version density (or distribution), version volume (or scale) and an operation to test. The terms may be used interchangeably for this paper.

Each testing environment contains a version density (of which 4 are currently tested).

- *Uniform*: In a uniform distribution with N versions, for a time interval T, the expected number of nodes at any time interval of length T_i in the interval is NT_i/T.
- *Linear*: For a time period T, if we take an arbitrary interval of Ti, then the number of versions of the document in that interval Ti would be $k \cdot i$ for some k.
- *Big Head Long Tail (BHLT)*: In this distribution, there are captures scattered throughout the whole time interval, but there are more captures in a short window of time.[2]
- *Multipeak*: The multipeak distribution is composed of two normal distributions.[3] This is effectively a Gaussian mixture model, a well-studied approach for representing multimodal distributions [18].

The simulation randomly generates a timestamp for each IPARO based on the version distribution, using the NumPy library [19] to sample from uniform, linear, log-uniform, and Gaussian mixture distributions. These timestamps are generated relative to the timestamp of the earliest IPARO. IPAROs in the IPFS are generally sorted from earliest to latest.

Second, each testing environment includes a specific operation to test. The operations are listed as follows:

- *Retrieve First*: Retrieval of the first IPARO.
- *Retrieve Latest*: Retrieval of the latest IPARO.
- *Add Node*: Adds one IPARO at the end of the existing chain and updates necessary references.
- *Retrieve by Sequence Number (RBSN)*: Retrieval of the Nth IPARO.
- *Retrieve by Time (RBT)*: Retrieval of an IPARO at time T.
- *List All*: List links to every IPARO. This operation should not be confused with retrieving all IPAROs, whose IPFS retrieval count is equal to N, where N is the chain length. In fact, listing all links often takes much less than N IPFS retrieves depending on the linking policy.

Each operation is tested based on the four basic operations, the IPNS "get," the IPNS "update," the IPFS "store," and the IPFS "retrieve" operations. Additionally, the "Add Node" operation is tested based on both the number of links and the number of IPARO link traversals.

Third, each testing environment contains a version volume. The version volumes used in the app are shown in Table I.

The data for huge volumes may be limited, in part, because of the storage constraints required to run the application.

TABLE I: The version volumes used for the IPARO Simulation Application.

Version Volume	Chain Length
Single	1
Tiny	2–9
Small	10–99
Medium	100–999
Large	1,000–9,999
Huge	10,000+

C. Linking Policies

There are 11 distinct linking strategies that we evaluate in our simulation framework. Throughout this paper, we refer to them consistently as *linking strategies*. The design of these strategies is influenced by distributed data structures and resilient peer-to-peer protocols that seek to minimize lookup cost under churn [15]. The linking strategies are listed as follows:

- *Single*: Only stores a single link to the previous version. This is same as the traditional singly linked list data structure.
- *Baseline*: Links to the immediate previous version and the very first version. It is a subset of every other strategy except the Single.
- *Comprehensive*: Links to all prior nodes.
- *K-Previous*: Links to the K consecutive previous versions and the first version.
- *K-Random*: Links to the K random prior versions, the previous version, and the first version.
- *Sequential K-Uniform*: Links to K prior versions (distributed uniformly across the sequence of versions), the immediate previous version, and the first version. This approach resembles skip list structures, which probabilistically maintains logarithmic traversal time through additional long-distance pointers [20].
- *Sequential S-Max-Gap*: Adds a necessary number of links to nodes between the immediate previous and the first versions that are no more than S hops away. It ensures that the time it takes to retrieve a node is no greater than $S + 1$ IPFS retrieve actions.
- *Base B Sequential Exponential*: Links in such a way that the version gap is a power of B, starting with the previous node. For instance, if B = 2, can link 1 version prior, 2 versions prior, 4, 8, 16, all the way to the first

[2]To be more precise, if we take any small enough interval of length i around a time t (measured in seconds), where $0 \leq t \leq T$, we would expect the number of documents in that interval to be equal to $i \cdot c/(1 + kt/T)$ for some c. For this setup, we use $k = 20$ and $T = 1000$ seconds.

[3]For our experimental setup, we set up a peak with a mean of 1,000 seconds and a standard deviation of 300 seconds, and another with a mean of 2,000 seconds and a standard deviation of 400 seconds. We discarded any timestamps outside of our test boundaries, such as negative timestamps.

TABLE II: The strategies chosen for analysis on Section IV-C. They were chosen for similar storage costs on the uniform distribution (see Figure 3), except for Single and Comprehensive strategies.

Linking Strategy	Chosen Parameters
Single	None
Comprehensive	None
K-Previous	$K = 4$
K-Random	$K = 4$
Sequential K-Uniform	$K = 4$
Sequential S-Max-Gap	$S = 16$
Sequential Base-B Exponential	$B = 3$
Temporal K-Uniform	$K = 4$
Temporal T-Min-Gap	$T = 160$s
Temporal Base-B Exponential	$B = 3$

version). Also links with the first version. This design is conceptually similar to navigable small-world networks that allow efficient decentralized search via logarithmic link placement [21].

- *Temporal N-Uniform*: Links to N prior versions (distributed uniformly across the window of time), the immediate previous version, and the first version.
- *Temporal T-Min-Gap*: Links in such a way that the maximum gap is a window of time T between captures if one such node is present, but otherwise extends the window, starting with the immediate previous version. Also links with the first version.
- *Base B Temporal Exponential*: Links in such a way that the time gap is growing exponentially with a power of B (i.e., the number of time units is closest to B^k for some integer k) starting with the previous node. For instance, if $B = 2$ and the time unit is days, can link 1 day prior, 2 days prior, 4, 8, 16, all the way to the first version. This strategy also links with the first version. For this paper, we set the time unit to 10 seconds.

To understand those linking strategies better, we use the link distribution plot for this paper.

The link distribution diagram shows how the sequence number of the source node gets plotted against the destination timestamps, showing how the strategy behaves not only temporally but also sequentially. The versatility comes not only from the fact that no source node can link to any future node, but also the fact that the timestamp of the source node is included as a reference timestamp.

See Figure 2 for visualizations. Temporal Exponential tends to require fewer links than Sequential Exponential under the big head long tail distribution as opposed to the uniform distribution since most of the nodes are concentrated towards the earlier endpoint (see Figures 2g, 2j–2l).

IV. Evaluation

A. Linked List Size

We analyze how different version volumes and different strategy parameters affect the linked list size of various policies. More specifically, we go over four non-trivial operations in this section: Retrieve By Sequence Number (RBSN), Retrieve By Time (RBT), Add Node, and List All. This section focuses more on the order of growth, particularly the effects of chain length N and the strategy parameters on the runtime and storage costs, than the effects of different environments on strategies. We note $\bar{T} = T^*/T$, $N' = N/K$, and $T' = T^*/K$, where T^* and T are described in more detail when we analyze the temporal strategies.

For the Single strategy, $O(1)$ links and $O(1)$ IPFS retrieves are required for Add Node, $O(N)$ IPFS retrieves are necessary for Retrieve by Sequence Number, Retrieve by Time, and List All operations. The Previous strategy has identical growth orders except for Retrieve First, which only requires $O(1)$ IPFS retrieves. The K-Previous and Sequential K-Uniform strategies both trade some storage with the retrieve time, generally requiring $O(K)$ links and $O(K)$ IPFS retrieves for K-Previous, but in exchange for only needing $O(N')$ IPFS retrieves for Retrieve by Sequence Number, Retrieve by Date, and List All operations.

On the opposite end, the Comprehensive strategy requires $O(1)$ IPFS retrieves for List All, Retrieve by Date, and Retrieve by Sequence Number operations, but $O(N)$ links and $O(1)$ IPFS retrieves for Add Node. The Sequential S-Max-Gap also attempts to reduce the storage cost to $O(N/S)$ links, in exchange for increasing the retrieve time to $O(S)$ for the other non-trivial operations. In effect, the sequential S-Max-Gap strategy is identical to Sequential (N/S)-Uniform.

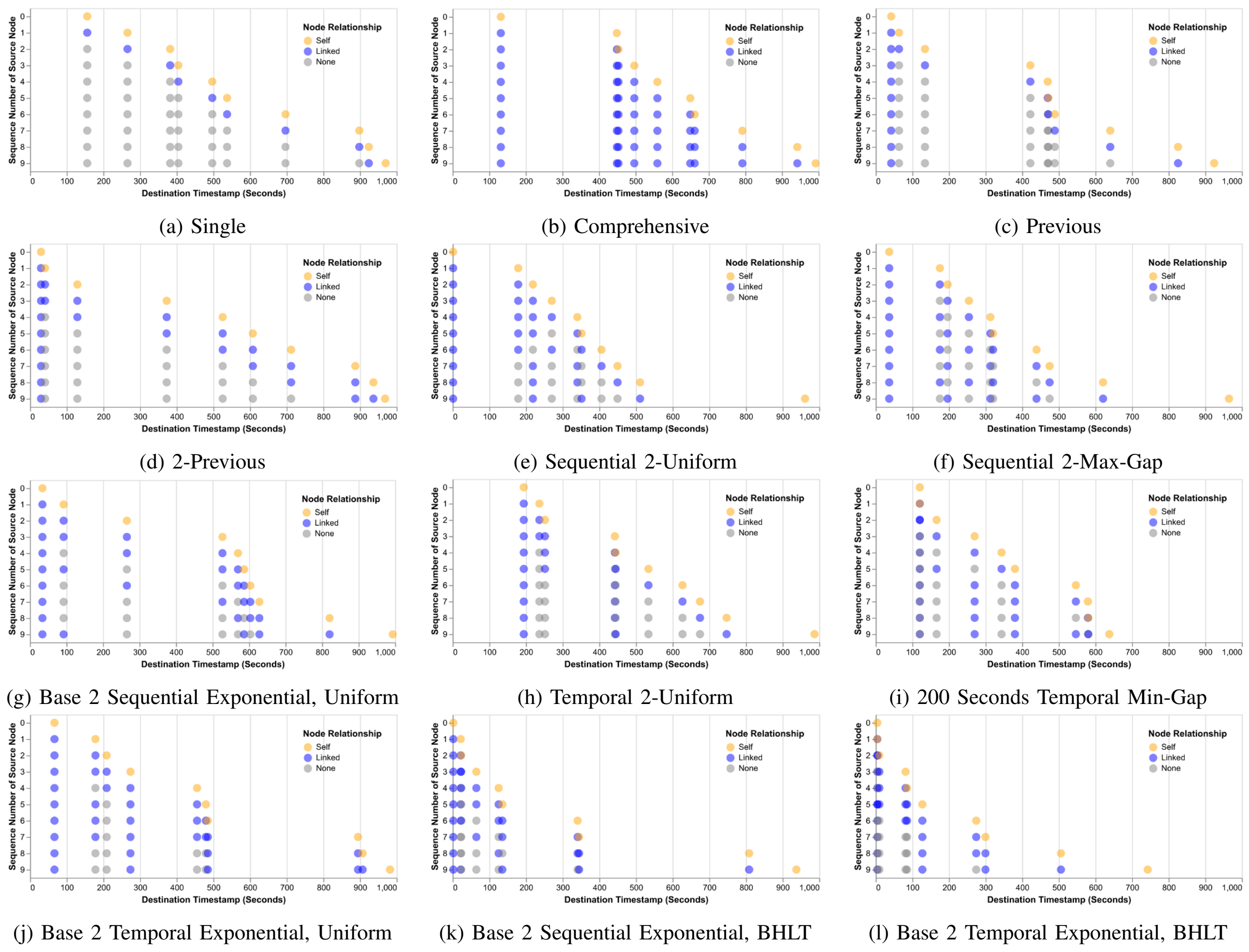
The K-Random strategy requires, on average, $O(\sqrt{N'})$ operations for Retrieve by Sequence Number. To prove that, we start with the event Z that there are zero links with sequence number between m and $N-2$ for K-random. Let $d = N - m$ be the sequence number distance between node N and node m. Denote this probability by $p(d, K)$. Since N can get arbitrarily large, we can approximate this probability with Equation (5).

$$P(Z) = p(d, K) \approx \left(1 - \frac{d}{N}\right)^K \tag{5}$$

Since the nodes in the linked list are numbered from 1 to N sequentially, d/N is roughly the probability that an arbitrary link is within a segment of length d. Therefore, the probability that a link is not in that segment is $1 - d/N$, which means that the probability that none of the K links are present in that segment is roughly $(1 - d/N)^K$.

When Z happens, then we can only travel by the previous link, so $E[D|Z] = 1$. Now let D be the distance traveled (i.e., the number of sequence numbers skipped) from one of the random links. Assuming that Z does not happen, D is uniformly distributed between 2 and d, and by Equation (6), we can calculate the expected value. Note that we assumed that there is only one link in the case that Z does not happen though in practice, there can be more than one link.

$$E[D|Z^c] = (d + 2)/2 \tag{6}$$



(a) Single (b) Comprehensive (c) Previous

(d) 2-Previous (e) Sequential 2-Uniform (f) Sequential 2-Max-Gap

(g) Base 2 Sequential Exponential, Uniform (h) Temporal 2-Uniform (i) 200 Seconds Temporal Min-Gap

(j) Base 2 Temporal Exponential, Uniform (k) Base 2 Sequential Exponential, BHLT (l) Base 2 Temporal Exponential, BHLT

Fig. 2: Link distributions of various linking strategies.

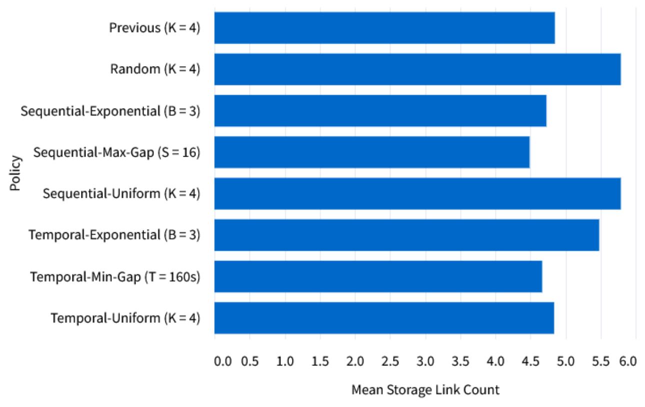


Fig. 3: The storage requirements for the chosen parameters for policies as used in Table II.

Now, we weigh the outcomes together by their probabilities, which means by Equation (6),

$$E[D] = 1 + (1 - p(d, K))\frac{d}{2} \tag{7}$$

It is important that we know when $E[D]$ is constant since the number of IPFS retrieve actions can be approximated by $d/E[D]$. In order for $E[D]$ to be constant, by Equation (7), we must have $p(d, K)$ equal to $1 - O(1/d)$. If we set $d = O(\sqrt{N'})$, we can apply the binomial expansion property [22] on Equation (5) to find out this probability: (By binomial/Taylor expansion or the approximation $(1 - \epsilon)^K \approx e^{-K\epsilon} \approx 1 - K\varepsilon$ for small ϵ, we obtain

TABLE III: Average-case space and time complexities for each operation. Every strategy has complexity $\Theta(1)$ for Retrieve Latest, and every strategy family listed besides the Single strategy (which has $\Theta(N)$ complexity) requires $\Theta(1)$ IPFS retrieves for Retrieve First. N_T denotes the number of IPAROs in a window of time T. For integer bases on Sequential Exponential, the average case complexity is bounded by $\log N$.

Strategy	Links Per Node	Add Node	RBT	RBSN	List All
Single	1	1	N	N	N
Previous	1	1	N	N	N
Comprehensive	N	1	1	1	1
K-Previous	K	K	N'	N'	N'
K-Random	K	$\sqrt{NK}$	$\sqrt{N'}$	$\sqrt{N'}$	N'
Sequential K-Uniform	K	K	N'	N'	N'
Sequential S-Max-Gap	N/S	N/S	S	S	S
Sequential Base B Exponential	$\log N$	$\log^2 N$	$\log N$	$\log N$	N
Temporal K-Uniform	K	N'	$N_{T'}$	$N_{T'}$	N
Temporal T-Min-Gap	$\bar{T}$	$\bar{T}$	$\bar{T} + N_T$	$\bar{T} + N_T$	N
Temp-Exp Base B (Time Unit T)	$\log(\bar{T})$	$N_T \log \bar{T}$	N_T	N_T	$N_T \log \bar{T}$

$\left(1 - O\left(\sqrt{1/(KN)}\right)\right)^K = 1 - O\left(\sqrt{K/N}\right)$). Then,

$$p(d, K) = \left(1 - \frac{\sqrt{N'}}{N}\right)^K \quad (8)$$
$$= \left(1 - O\left(\sqrt{1/(KN)}\right)\right)^K \quad (9)$$
$$= 1 - O\left(\sqrt{K/N}\right) \quad (10)$$

Plugging in Equation (10) to Equation (7) yields

$$E[D] = 1 + O\left(\sqrt{K/N}\right) \cdot \frac{\sqrt{N'}}{2} = O(1). \quad (11)$$

Therefore, we can conclude that the number of IPFS retrievals required for the K-Random strategy is $O(\sqrt{N'})$. In the worst-case scenario, we can have all nodes linked to the first K nodes. In that case, the amount of time required can go up to $N - K$ if searching for the $(K+1)^{\text{st}}$ node. To add a node, we need to do K retrievals which means the average complexity is $\Theta(\sqrt{NK})$.

The average amount of time required for Retrieve by Time and Retrieve by Sequence Number for sequential K-uniform is at most N' and on average takes roughly that much for a greedy search algorithm. To list all nodes, this would require $O(N')$ steps.

The base B sequential exponential strategy requires $\Theta(\log_B N)$ links to add a new node for a chain length of N, which follows from the fact that

$$\sum_{i=1}^{N} \log_B i = \Theta(N \log_B N) \quad (12)$$

This can be derived by approximating the sum of logarithms with an integral and applying Stirling's formula [23]. After applying Stirling's formula and dividing by the chain length N to get the average number of links per node required, which is $\Theta(\log_B N)$. This is also the upper bound on the required storage space for sequential exponential, since no more than $\log_B N$ links are created for each node. Additionally, the worst-case bound on the retrieve operations is $(B-1)\log_B N + O(1)$. By traveling through one of the links, the search space is cut down from N to N/B at most. However, each substep costs up to $B - 1$ IPFS retrieves since we must know which section of size N/B to search. Therefore, our recurrence relation is:

$$T(n) = \begin{cases} 1, & n \leq 1 \\ T(n/B) + B - 1, & n > 1 \end{cases} \quad (13)$$

Solving this recurrence relation, we have our worst-case bound of $(B-1)\log_B N + O(1)$. Because it takes on average about $B/2$ IPFS retrieves per step on $O(\log_B N)$ steps, the average-case bound is also roughly $O((B-1)\log_B N)$, which we can write as $\Theta(\log N)$ because B is a constant.

For any integer base $B \geq 2$, it takes $(B-1)\log_B N + O(1)$ IPFS retrievals to store a new node. To see why, we can use induction. The base case when $N < B$ is trivial since only the links to the previous and first nodes are valid links. For the inductive case (when $N \geq B$), we can view B^n as the telescoping sum:

$$B^n = 1 + \sum_{i=0}^{n-1} (B-1)B^i = 1 + (B-1)\sum_{i=0}^{n-1} B^i \quad (14)$$

In other words, we can hop to the previous node to get the first link and take the link to its previous node $B - 1$ times to get the second link, then hop to the node B sequence numbers away repeatedly for $B-1$ times for the third link, and then hop to the node B^2 sequence numbers away repeatedly for $B - 1$ times, etc., while the resulting sequence number is greater than 0. The number of iterations it takes is therefore $\log_B N + O(1)$, with each iteration (except for the first) taking $B - 1$ hops.

However, if the base B is a noninteger, it takes $O((B - 1)\log_B^2 N)$ IPFS retrieves to store a new node. As B is constant, we can instead write the complexity as $O(\log^2 N)$. To prove that, we can multiply the number of links, which is $O(\log N)$, by our worst-case bound for the Retrieve by Sequence Number operation, which is $O(\log N)$, to get our worst-case bounds for the number of required IPFS retrieves for storage. This "Add Node" cost can be rather prohibitive when talking about the "medium" or smaller volumes, but is less so for larger volumes.

For List All, we split the analysis into two parts. For bases $B > 2$, we must traverse all links until we reach a link with the original node, which means that we must use at least $B-1$ IPFS retrieves to get $2B$ nodes. If we let $T(N)$ be the time it takes for Base-B Sequential Exponential to list N links, we have $T(N) = T(N-2B) + B + O(1)$, implying that $T(N) = O(N)$.

But for $B \leq 2$, the number of consecutive nodes linked is bounded below by the largest number x such that $B \leq 1+1/x$. Solving for x, we get $x = 1/(B-1)+1 = B/(B-1)$. Dividing N by x gives us the number of IPFS retrieves it takes for x consecutive nodes, which is N/x, or $N(B-1)/B = N(1-B^{-1})$. The reason the number of consecutive nodes is important for Sequential Exponential is because it can perform the List All operation at a much greater pace by mirroring the performance of K-Previous on that operation. Therefore, we have an upper bound of $N(1-B^{-1}) + O(1)$ IPFS retrieves for the List All operation on Sequential Exponential.

The bounds for temporal strategies (Temporal Exponential, Temporal Uniform, and Temporal Min-Gap) depend on the distribution of nodes over time and can affect the runtime complexity in a way that would complicate the ability to analyze the temporal strategies, at least in the current system we are working with. However, it is reasonable to assume the worst-case bounds generally do not have a higher-order growth rate than $O(N)$. For instance, the worst case for Base B Temporal Exponential on storage would happen if the time unit is T and each IPARO is exactly B^kT seconds away from the source node, thereby requiring links to all the prior nodes. On the other hand, one can maximize the retrieve time for Base B Temporal Exponential by linking to only the previous and the first node (i.e., behave like the K-Previous strategy) when the time unit is greater than the interval between the first node and the latest node.

For tighter bounds, we can switch to a new unit system. Let T^* be the interval between the first node and the latest node, and let T be the time unit. Additionally, let N_T be the maximum number of links in a given window of time t.

Temporal T-Min-Gap requires $O(\bar{T})$ links to store a new node and $O(\bar{T} + N_T)$ retrieves to store for a uniform time distribution. To retrieve a node, either by sequence number or by time, it would require $O(\bar{T} + N_T)$ time.

Temporal K-Uniform requires $K+2$ links and $O(N')$ IPFS retrieves to add a new node, but needs $O(N_{T'})$ IPFS retrieves to retrieve a node by time or by sequence number.

The Base B Temporal Exponential requires $O(\log_B(\bar{T})[N_T + \log_B(\bar{T})])$ IPFS retrievals and $O(\log_B(\bar{T}))$ links on average to store a node, but in exchange requires $O(N_T + \log_B(\bar{T}))$ IPFS retrieves on average to retrieve a node by time or by sequence number.

B. Storage Cost Analysis

We only consider storage costs in this section, so the time cost for Add Node is deferred to Section IV-C. To date, all of the heatmap visualizations have confirmed our findings from Section IV-A. In particular, Sequential Exponential beats out Comprehensive but takes up more space than the Single and 2-Previous strategies. In general, higher bases for both Temporal and Sequential Exponential require fewer links. Conversely, for $B = 2$, Temporal Exponential can require up to 10.7 links for the linear time density and as few as 5.9 links on BHLT, as compared to Sequential Exponential which requires from 6.65 links. This may be attributed to the skewness of the Linear version density towards later timestamps and the BHLT version density towards earlier timestamps. The Single and Previous strategies require only one link and two links per node, respectively.

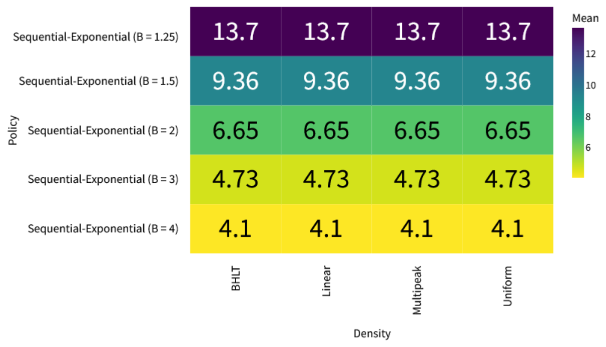
C. Time Cost Analysis

Retrieving the latest node only requires one IPNS lookup and one IPFS lookup for all policies. For retrieving the first node, all policies besides the Single policy require only one IPNS lookup and two IPFS lookups, one to fetch the latest node and the other to fetch the first node from its list of links. The single policy, on the other hand, requires $N-1$ IPFS retrieves. For this section, only average-case times are considered; for worst-case times, see Section IV-A. Moreover, we analyze the operations described in Section IV-A.

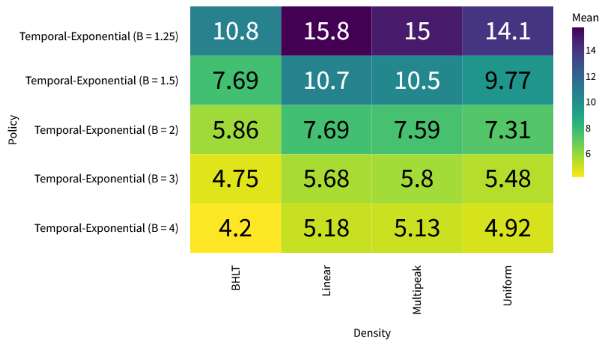
1) Retrieve by Sequence Number (RBSN): Figure 5a shows the number of IPFS retrievals required to Retrieve by Sequence Number. Sequential Exponential requires very few operations for RBSN. Since Sequential Exponential behaves identically on all cases for the sequence number, we can assume that the mean number of actions for the policy to perform an RBSN is equal no matter the version distribution. For Temporal Exponential, we start with a side-by-side comparison with the Sequential and Temporal Exponential policies. On one hand, Temporal Exponential performs better on a linear environment than other environments because the linear environment has a higher number of nodes concentrated on the later versions. On the other hand, the BHLT version density has more nodes concentrated towards the first node. Since the links are relatively sparse on those areas, it follows that the BHLT version density requires more operations from the Temporal Exponential policy than the other densities tested. Multipeak version density can also work against the temporal policies, especially for higher bases, since there are nodes concentrated both near the left and the right endpoints, meaning that the IPFS must navigate through only the previous link for most sequence numbers or timestamps.

2) Retrieve by Time: The Retrieve by Time (RBT) operation, as mentioned in Section III-B, deals with picking the closest IPARO to a uniformly-distributed timestamp. According to Figure 5b, Comprehensive performs the best at two IPFS operations. However, Base 3 Temporal Exponential and Base 3 Sequential Exponential perform similarly, with Temporal Exponential doing slightly better in Multipeak and Linear than in Sequential Exponential. 4-Random is also competitive with Sequential and Temporal Exponential strategies, and Temporal Min-Gap can be versatile in a Multipeak setting.

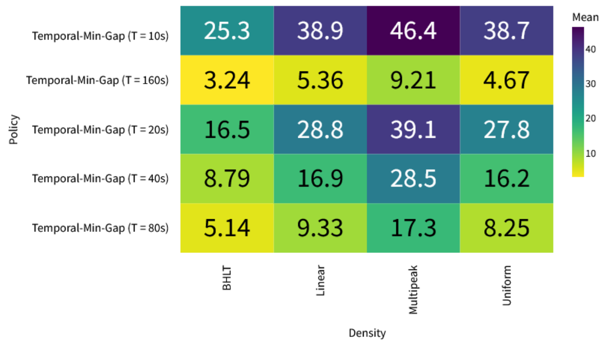
3) Storage Retrieve Time: Adding a node incurs a one-time cost of the number of IPFS retrieves, but one-time



(a) Strategy: Sequential Exponential



(b) Strategy: Temporal Exponential



(c) Strategy: Temporal Min-Gap

Fig. 4: Storage Costs of Various Strategies Under Different Distributions

costs are more impactful especially when there are only a few operations are performed after storing. Figure 5c shows the cost of adding a node. Storing a node requires IPFS retrieves in addition to the three other actions (one IPNS write, one IPNS read, and one IPFS store). Comprehensive and Single are tied for the best IPFS Retrieve count complexity (0.99 for both). However, 4-Previous comes in third at 1.93 IPFS retrievals. Sequential Exponential is next at 7.49 IPFS Retrieves on average. 4-Random and Sequential 16-Max-Gap are next at 11.7 and 14.3 IPFS retrieves, respectively, for the Uniform distribution. The exact number of IPFS retrieve actions required to add a node for 4-Random slightly fluctuates depending on the distribution.

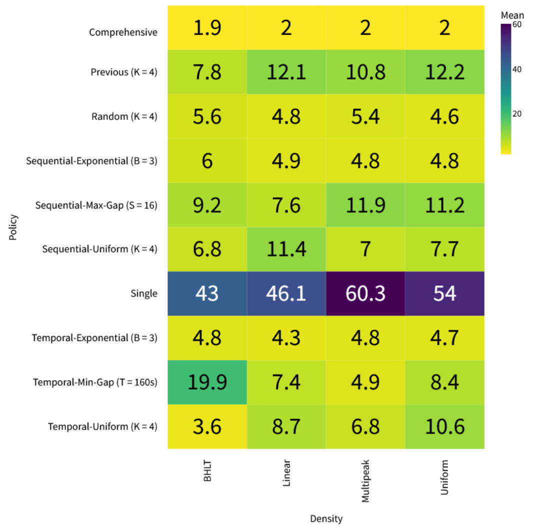
4) List All: Finally, the "List All" operation (see Figure 5d) asks for the number of IPFS retrieves required to get all links. For that purpose, Sequential 16-Max-Gap took only 16 IPFS retrieves despite taking up slightly less storage space than Sequential Exponential and Temporal Exponential, which took 36 and 40 IPFS retrieves respectively on the Uniform distribution, though Sequential 16-Max-Gap is less scalable due to linear storage complexity. The 4-Previous strategy comes second, at 25 IPFS retrieves, and is a more scalable alternative if List All operations are frequent (i.e., at least equal to 10 percent of the total operations). In fact, we could also use only $K = O(\log N)$ links for K-Previous, and it would take $O(N/\log N)$ IPFS retrieves to list all links while only using a comparable amount of storage space to Sequential and Temporal Exponential.

V. Conclusions and Future Work

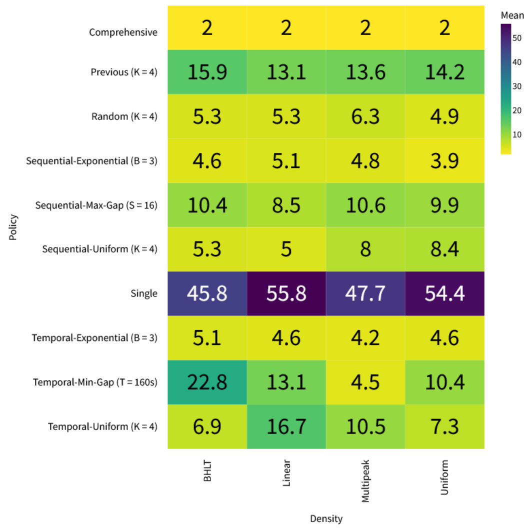
Out of the 11 linking policies surveyed, the Comprehensive strategy is versatile if the storage constraints are not as stringent as the retrieve constraints, and nodes are not being constantly added over time. This is because the Comprehensive strategy requires only up to two IPFS lookups for a "retrieve by number" operation, and up to three IPFS lookups for a "retrieve by date" operation. For that reason, Comprehensive works best for volumes that are in the "medium" range or smaller. However, in many real-world applications, storage constraints and retrieval operation times are equally important, and Sequential Exponential and Temporal Exponential, especially with fine-tuned parameters and medium or larger volumes, can help if there are not many "List All" operations. However, Temporal Exponential is more sensitive to version distributions than Sequential Exponential and may take longer in some distributions than others. If the number of "List All" operations is relatively large, then the K-Previous strategy with $K = O(\log N)$ is preferred. Even though Temporal Min-Gap may have some potential in the Multipeak distribution as it competes with both Sequential and Temporal Exponential for node retrieval, the upfront Add Node cost would offset that advantage from the Retrieve by Time and Retrieve by Sequence Number operations.

For our future work, we would like to expand not only the strategies to test but also provide a more in-depth survey on how other distributions, such as the real-world temporal distribution based on TimeMaps [24]–[29] of a few URLs, affect the performance of various linking strategies. Additionally, we would like to explore which strategies are the most resilient when some nodes are unavailable.

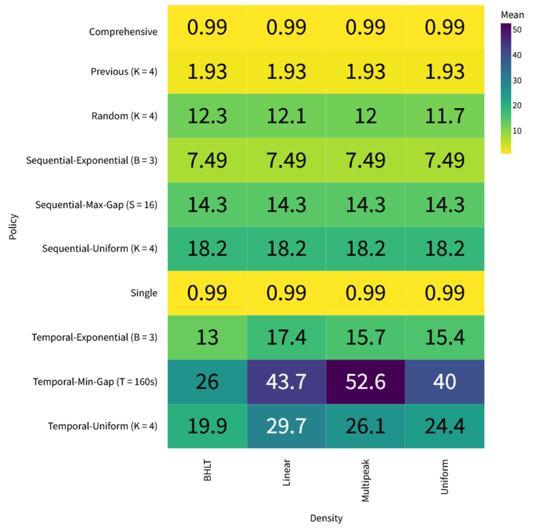
Moreover, it is helpful to search for parameters that minimize not only the link storage cost but also the cost of retrievals and the IPFS retrieve count for the "add node" operation.



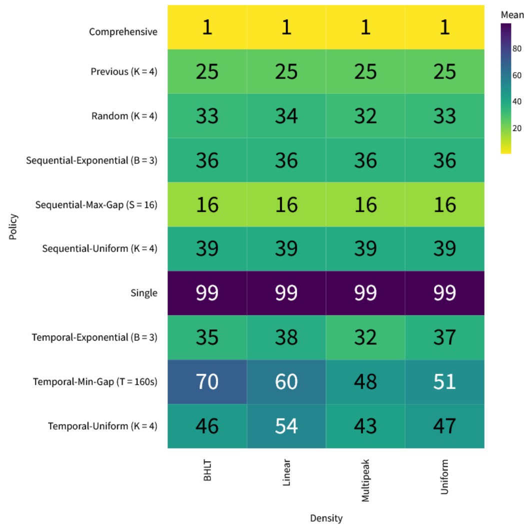
(a) Retrieve by Sequence Number



(b) Retrieve by Time



(c) Add Node



(d) List All

Fig. 5: Time Costs of various operations on chosen strategies

VI. Acknowledgments

This work was completed as part of a senior capstone project at Drexel University, with valuable feedback and support from faculty and peers during the development process. The authors also acknowledge the open-source communities behind IPFS, IPNS, and related tools that made the experiments and simulations in this paper possible.

The authors received no external funding for this research.

References

[1] J. Benet, "IPFS - Content Addressed, Version, P2P File System," Tech. Rep. arXiv:1407.3561, 2014.

[2] I. Kreymer, "IPFS Composite File Utilities," https://github.com/webrecorder/ipfs-composite-files, 2022.

[3] S. Alam, M. Kelly, and M. L. Nelson, "InterPlanetary Wayback: The Permanent Web Archive," in *Proceedings of the IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, June 2016, pp. 273–274.

[4] M. Kelly, S. Alam, M. L. Nelson, and M. C. Weigle, "InterPlanetary Wayback: Peer-to-Peer Permanence of Web Archives," in *Proceedings of the 20th International Conference on Theory and Practice of Digital Libraries (TPDL)*, 2016, pp. 411–416.

[5] S. Alam, M. Kelly, M. C. Weigle, and M. L. Nelson, "InterPlanetary Wayback: The Next Step Towards Decentralized Web Archiving," IPFS Lab Day '18, 2018.

[6] S. Alam, M. Kelly, M. C. Weigle, and M. L. Nelson, "InterPlanetary Wayback: A Distributed and Persistent Archival Replay System Using IPFS," DWeb Summit '18, 2018.

[7] S. Alam, "IPARO: InterPlanetary Archival Record Object for

Decentralized Web Archiving and Replay," in *iPRES 2023: The 19th International Conference on Digital Preservation*, 2023, pp. 1–11. [Online]. Available: https://hdl.handle.net/2142/121090

[8] S. Alam, "IPNS Blockchain," https://github.com/oduwsdl/IPNS-Blockchain, 2018.

[9] S. Alam, "Implementing History Aware IPNS Via Certificate Transparency Log Data Structure Trillian," https://discuss.ipfs.tech/t/implementing-history-aware-ipns-via-certificate-transparency-log-data-structure-trillian/5756, 2019.

[10] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, p. 123–133, Aug. 1988. [Online]. Available: https://doi.org/10.1145/52325.52338

[11] ISO 28500:2017, "WARC File Format," https://iso.org/standard/68004.html, 2017.

[12] Internet Archive, "CDX File Format," http://archive.org/web/researcher/cdx_file_format.php, 2003.

[13] S. Alam, M. L. Nelson, H. Van de Sompel, L. L. Balakireva, H. Shankar, and D. S. H. Rosenthal, "Web Archive Profiling Through CDX Summarization," *International Journal on Digital Libraries (IJDL)*, vol. 17, no. 3, pp. 223–238, 2016.

[14] R. E. Tarjan, *Data structures and network algorithms*. SIAM, 1983.

[15] S. C. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proceedings of the General Track: 2004 USENIX Annual Technical Conference*. USENIX, 2004, pp. 127–140. [Online]. Available: http://www.usenix.org/publications/library/proceedings/usenix04/tech/general/rhea.html

[16] P. Smith, D. Hutchison, J. P. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, "Network resilience: a systematic approach," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 88–97, 2011.

[17] Microsoft, "Windows Subsystem for Linux," https://github.com/microsoft/WSL.

[18] C. M. Bishop and N. M. Nasrabadi, "*Pattern Recognition and Machine Learning*," *J. Electronic Imaging*, vol. 16, no. 4, p. 049901, 2007. [Online]. Available: https://doi.org/10.1117/1.2819119

[19] C. R. Harris, K. J. Millman, S. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nat.*, vol. 585, pp. 357–362, 2020.

[20] W. W. Pugh, "Skip lists: A probabilistic alternative to balanced trees," *Commun. ACM*, vol. 33, no. 6, pp. 668–676, 1990. [Online]. Available: https://doi.org/10.1145/78973.78977

[21] J. M. Kleinberg, "The small-world phenomenon: an algorithmic perspective," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, F. F. Yao and E. M. Luks, Eds., 2000, pp. 163–170.

[22] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms, Volume 1*. Addison-Wesley Professional, 1997.

[23] D. Romik, "Stirling's approximation for n!: The ultimate short proof?" *The American Mathematical Monthly*, vol. 107, no. 6, pp. 556–557, 2000. [Online]. Available: http://www.jstor.org/stable/2589351

[24] H. Van de Sompel, M. L. Nelson, and R. Sanderson, "HTTP Framework for Time-Based Access to Resource States – Memento," RFC 7089, Internet Engineering Task Force, 2013.

[25] S. Alam and M. L. Nelson, "MemGator - A Portable Concurrent Memento Aggregator: Cross-Platform CLI and Server Binaries in Go," in *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '16, 2016, pp. 243–244.

[26] M. Kelly, L. M. Alkwai, S. Alam, M. L. Nelson, M. C. Weigle, and H. Van de Sompel, "Impact of URI Canonicalization on Memento Count," in *Proceedings of the 17th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '17, 2017, pp. 303–304.

[27] S. Alam and M. Graham, "CDX Summary: Web archival collection insights," in *Proceedings of the 26th International Conference on Theory and Practice of Digital Libraries*, ser. TPDL '22, Istituto Sant'Antonio Dottore. Padua, Italy: Springer, Sep. 2022, pp. 297–305. [Online]. Available: https://doi.org/10.1007/978-3-031-16802-4_25

[28] S. Alam, M. L. Nelson, H. Van de Sompel, L. Balakireva, H. Shankar, and D. S. H. Rosenthal, "Web archive profiling through CDX summarization," *International Journal on Digital Libraries*, vol. 17, no. 3, pp. 223–238, Sep. 2016. [Online]. Available: https://doi.org/10.1007/s00799-016-0184-4

[29] S. Alam, M. C. Weigle, M. L. Nelson, F. Melo, D. Bicho, and D. Gomes, "MementoMap framework for flexible and adaptive web archive profiling," in *Proceedings of the 19th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '19, University of Illinois. Urbana-Champaign, Illinois, USA: IEEE, Jun. 2019, pp. 172–181, (Extended preprint: arXiv:1905.12607). [Online]. Available: https://doi.org/10.1109/JCDL.2019.00033