# IPARO: InterPlanetary Archival Record Object for Decentralized Web Archiving and Replay

**Sawood Alam**
*Internet Archive*
*USA*
*sawood@archive.org*
*0000-0002-8267-3326*

**Abstract – We proposed a decentralized version tracking system using the existing primitives of IPFS and IPNS. While our description talks primarily about archived web pages, we proposed the concept of IPMT and namespacing so that it can be used in other applications that require versioning, such as a wiki or a collaborative code tracking system. Our proposed system does not rely on any centralized server for archiving or replay of the content. The system continues to allow aggregators to play their role from which both large and small archives can benefit and flourish.**

**Keywords – IPARO, IPFS, Decentralized Web, DWeb, Web Archiving**

**Conference Topics – WE'RE ALL IN THIS TOGETHER; DIGITAL ACCESSIBILITY, INCLUSION, AND DIVERSITY**

## I. Introduction and Background

Web archiving is the practice of temporal versioning and preservation of representations of resources on the web. An archival replay is the practice of the playback of the archived historical representation of web resources while maintaining the essence and fidelity of the web resource. Decentralized content-addressable file systems, such as InterPlanetary File System (IPFS) [1], offer the opportunity to preserve all the historical versions of files stored in them. However, their current implementations lack native support for versioning.

In 2016, our initial work on InterPlanetary Wayback (IPWB) was a successful exploration of the possibilities of web archiving in the early days of IPFS [2-4]. It gained a fair share of visibility in the relevant communities. However, it relied on a local index for the system to operate, which made its operations centralized, despite the archival data being decentralized. We address this shortcoming in this work to make web archiving truly decentralized.

During the IPFS Lab Day event in 2018, we discussed the limitations posed by the centralized index for decentralized web archives and laid out what was needed to address them [5]. We proposed that the InterPlanetary Name System (IPNS) be history-aware, but current implementations only keep the most recent mapping of a URI to its corresponding content hash using a Distributed Hash Table (DHT), leaving prior versions of a resource untracked when the IPNS mapping is updated. We later proposed IPNS-Blockchain (a decentralized blockchain-based approach) [6] and Trillian data store (an append-only federated solution) [7] to eliminate the need of local index. Since neither of these approaches attracted enough momentum to get implemented, we came up with another solution that operates within the existing IPFS primitives.

In this approach we embed references to prior versions of a resource in the new versions, forming a singly linked list for backward traversal. We then use IPNS to enable direct access to the most recent version and traverse the chain from there. Moreover, we propose media types and namespaces in IPFS to make local indexing of archival resources more efficient, a concept that is orthogonal to IPNS-based

iPRES 2023: The 19th International Conference on Digital Preservation, Champaign-Urbana, IL, US.

DOI: 10.1145/nnnnnnn.nnnnnnn

access. Finally, we describe ways to achieve deduplication by storing slices of data separately that are likely to be repeated many times. Our proposed system, InterPlanetary Archival Record Object (IPARO), can be implemented without the need of a centralized server and archival records can be played back from a client-side system like a web browser using Service Workers [8,32].

Recent developments of web archiving tools [9-11] have made the practice more accessible to individuals pursuing personal web archiving. However, discovery of and access to the archived web content is still limited to large institutions and organizations, while small efforts suffer from disuse. Making large-scale web archival data available to researchers is still a challenge [12-15]. Our proposed system, IPARO, makes the ecosystem more inclusive and accessible to everyone. Archived content preserved in the IPFS network would be discoverable and accessible to everyone, irrespective of the creators of IPAROs.

## II. Related Work

Content-addressing is well-established practice, especially, in the context of data shared over a network. Unlike location-addressing, in which Uniform Resource Locators (URLs) are used to find a resource, content-addressing uses technique like hashing to identify corresponding resources. Content on a URL may change over time, but a content-address is derived from the content itself, so it is guaranteed to maintain the integrity and fixity of the content for a given content-address. Content-addressing has traditionally been used in peer-to-peer and decentralized systems like Torrent and Git. More recently, the introduction of InterPlanetary File System (IPFS) [1] has also appreciated content-addressing in its protocol to leverage advantages like peer-to-peer discovery, replication, deduplication, and integrity. In an IPFS network any data can be added, which results in a content identifier (CID), using one of the many supported hashing algorithms. The CID can then be used to retrieve the data by performing a lookup in the peer-to-peer network. Alternatively, an InterPlanetary Name System (IPNS) entry can be added that maps a URL to its corresponding current CID, which would allow performing lookups using the URL, instead of the content digest. IPNS entries are broken into pieces and stored in a distributed hash table, so that there is no single point of failure, and the table does not grow too large on a single node.

While distributed computing, decentralized web, peer-to-peer networks, content-addressing, and web archiving are well explored fields individually, decentralized web archiving is still a fairly new and niche discipline. After the emergence of IPFS, we explored it for decentralized web archiving the first time in 2016 by introducing InterPlanetary Wayback (IPWB) [2-5]. We processed Web ARChive (WARC) files (an ISO standard file format, used to preserve archival data) [16] to go through HTTP response records, split them in headers and payload, store the two pieces, get the corresponding pair of content digests, and create a local index that maps that archived URL and archiving date-time to the corresponding IPFS hashes of the header and payload. This local index is queried during the replay time to retrieve corresponding data from the IPFS network. The header and payload are split (as opposed to storing them as a single record) before storage to leverage deduplication of payload as the headers can be unique even when the payload is the same in consecutive observations of the same or different URLs (primarily due to the presence of the HTTP "Date" header). We also implemented a Service Worker module, Reconstructive, to allow client-side archival replay with minimum rewriting [8]. The biggest limitation of our IPWB system is the need for a local index to keep track of archived URLs and their corresponding IPFS hashes (i.e., CIDs), which makes it partially decentralized.

Webrecorder has embraced IPFS and other decentralized file systems by adding support for them as storage engines. For example, using their ArchiveWeb.page tool, one can share archived data via IPFS. Similarly, their ReplayWeb.page tool allows playback of archived data directly from an IPFS address. However, these tools use IPFS as a file storage system and store entire WARC files (or the emerging WACZ files), as opposed to individual WARC records (as used by our IPWB system). References to such archived bundles are shared using out-of-band channels as there does not provide any built-in mechanisms for the discovery of the archived content in the IPFS network.

Bamboo is a cryptographically secure, distributed, single-writer append-only log that supports transitive partial replication and local deletion of data [17]. This log format creates a tree-

like linked list that has a logarithmic path length between any two entries (from newer to older). Each entry in this format has a link to the previous entry and another link to an older entry to allow long jumps in the chain. Our resilient linked list concept has similar inspiration, but we allow an arbitrary number of links from any node to its prior nodes. Our system also caters to the possibility of incorporating nodes that were left from being part of the chain earlier due to any race conditions or transient discovery issues.

## III. METHODOLOGY

We create an InterPlanetary Archival Record Object (IPARO) for every archival observation that is intended to be looked up and replayed independently. These objects contain an extensible set of headers followed by the data in any one of the many supported archival formats and optional trailers, stored as IPFS media types (a concept we introduce in section III-A), as shown in Fig. 1. The purpose of headers is to identify the media type of the data, to establish relationships with other objects, to describe interpretation of the data and any trailers, and to hold other associated metadata.

An IPARO is the basic building block of web archiving in IPFS. It is enough to have a working decentralized web archiving system just by storing IPAROs. Anyone can create and store these, while anyone can scan the entire IPFS network data in the future, identify IPAROs from all the stored objects, and interpret them for replay based on the self-contained information. However, this approach is impractical and inefficient. In the coming sections we address these limitations.

### *A. InterPlanetary Media Types (IPMT)*

Media types is not a new concept, but we propose a way to bring it to the IPFS ecosystem as InterPlanetary Media Types (IPMT), which is otherwise an opaque data storage system. In traditional file systems, applications often rely on file name extensions to identify the format or media types of files. In cases where those are absent or are not reliable, applications attempt content sniffing (such as, reading the first few bytes to find signatures of various media types) to interpret the file appropriately. It is worth noting that a media type is not only about the syntactic interpretation of bits, but also the semantics. For example, a JSON file can be interpreted by a JSON parser, but the contents of

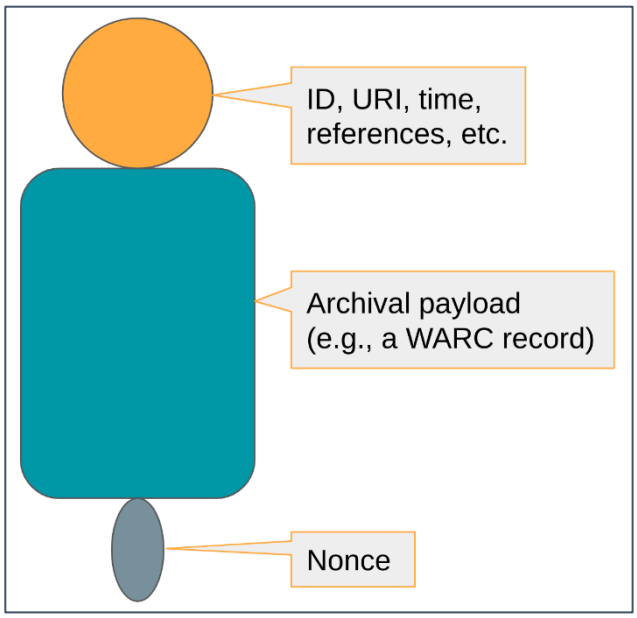


Fig. 1: InterPlanetary Archival Record Object (IPARO) Outline

the file can have configuration information of a system, state of a system, data records of an observation, etc. In Hypertext Transfer Protocol (HTTP) a server can convey the media type (and other clues for the correct interpretation) of a response with the help of response headers on-the-fly. We propose an HTTP-like mechanism for this in IPFS, but in this case the headers should be stored along with data (similar to how WARC records store metadata of each record). However, this means every record will have additional headers that many clients might not understand. One way to solve it is to use a proxy that converts IPMT headers to HTTP headers. Another approach would be to extend the IPFS API to optionally add/remove those headers depending on the need and capabilities of the client. These approaches require changes that might not be desired or feasible. In section III-D we discuss unobtrusive ways to keep the headers and data separate and combine them when needed. The IPFS ecosystem has the concept of InterPlanetary Linked Data (IPLD) [18] which facilitates interoperability among many hash-based systems. It allows external schema definitions to describe the outline of data types. Hence, IPLD can be used to describe IPMTs.

The concept of IPMT is generic so that various use cases and applications can leverage different IPMTs, but in this work we focus on IPMTs that can be interpreted by archival replay systems. Together the set of these IPMTs falls under IPAROs as described earlier. In sections III-A1 through III-A6 we describe primary IPARO candidate IPMTs, but this list can be

extended as more supported archival formats emerge.

*1) Memento* - We introduce a media type called memento, which in addition to IPMT headers, contains the original HTTP response headers and payload, corresponding HTTP request message (if available), and any additional metadata created by the crawler. This IPMT may also contain references to the corresponding archived versions of all the page requisites as a dependency graph. This media type holds an archival record of only one representation of a resource. Ideally, this can be part of WARC as an "exchange" record, but such a WARC type is not defined in the specifications yet, because the use-case was not realized before.

*2) WARC* - A Web ARChive (WARC) [16] file is an arbitrary set of records of HTTP request, response, metadata, and various other types of resources (often compressed individually) concatenated together. There are no explicit order, grouping, or limitations defined for the format. Direct access to specific records in a WARC file usually requires an external index (a CDX or CDXJ file [19]). It is possible to store a whole WARC file and optionally its index with many observations of many web resources (such as one or more web pages and all their page requisites) as an IPARO in IPFS, which can then be retrieved and replayed by a client that supports it.

*3) WACZ* - Web Archive Collection Zipped (WACZ) [20] is a Zip container for one or more WARC files, their CDXJ index, and various other metadata files as a single bundle. It is possible to store WACZ files as IPAROs to be retrieved and replayed by supporting clients.

*4) HAR* - HTTP Archive (HAR) [21] file format is commonly used in web browsers' developer console for network and performance analysis and debugging of web pages. It is a JSON-based file format that can be used for archival replay directly or after transforming it to something like a WARC file. This is yet another candidate for being an IPARO.

*5) Web Bundles* - Web Bundles or Web Packaging [31] is an evolving format to deliver multiple HTTP exchanges as a single resource by the primary origin or third-party aggregators/CDNs when signed. This too can be an IPARO.

*6) Annotations* - IPAROs are supposed to be immutable, so any annotations, contexts, additional metadata, or linking that are realized after the storage of an IPARO must be attached as a separate object (without altering the existing IPARO). An annotation is not necessarily one of the archival formats, but it is of interest in the context of archival replay, so it would be useful to include it in the set of IPMTs that fall under IPARO. The exact format details are yet to be determined, but semantics can be borrowed from existing specifications and practices related to annotations [22].

### *B. Immutable Linked List*

Web archives usually archive each web resource in their collection numerous times over time. As a result, they can report a list of all the observations (also known as a TimeMap [23]) of the resource representations of a given URI they have recorded. Moreover, they can resolve a specific version (i.e., a memento) of the resource close to a given time in the past (using a TimeGate resource) [24-25]. An archival replay system, when replaying a memento (an archived version of a representation of a resource), usually also reports first, last, previous, and next mementos in a "Link" response header using corresponding link relations. In traditional web archival replay systems, an index is used that is sorted primarily on a key, called, Sort-friendly URI Reordering Transform (SURT) [26], and secondarily on datetime (in `YYYYMMDDhhmmss` format). This ensures that all the observations of the same URI have spatial locality in the index, making it easier to list all the versions or locate a specific version close to a given date and time. Those index entries point to corresponding byte offsets and chunk sizes in corresponding WARC files where the archived data is stored (often in compressed format).

We used a similar idea in our IPWB system during our initial exploration of decentralized web archiving, but we loaded individual records in IPFS and replaced WARC references in the index file with corresponding IPFS CIDs instead. The downside of this approach was centralization of the index. We are changing it in this work by making IPAROs hold necessary references to prior versions for traversal as shown in Fig. 2.

By leveraging the "Link" header concept of the Memento protocol we can add references in the header part of an IPARO to a number of prior observations of the resource held by that IPARO. These references (i.e., IPFS CIDs) can point to the known immediate previous observation, the very

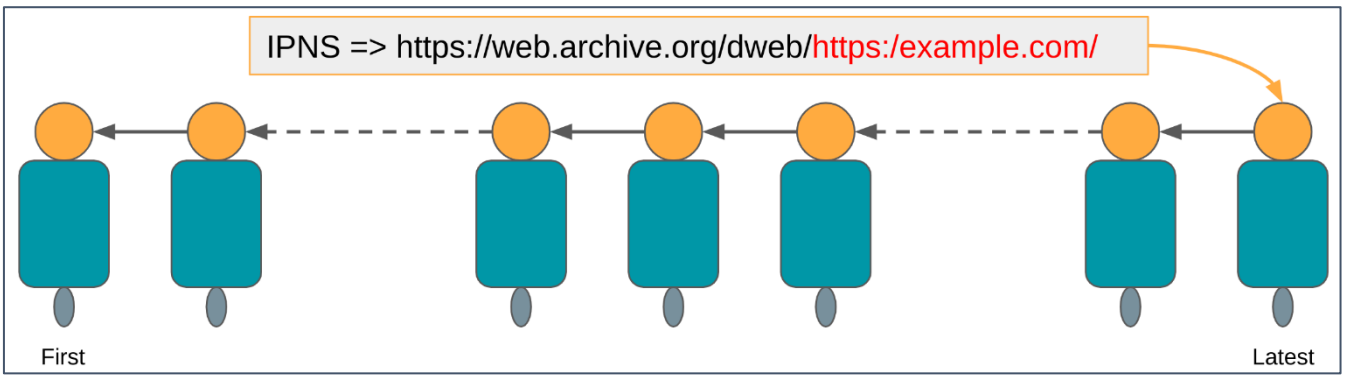


Fig. 2: An Immutable Linked List of IPAROs with an IPNS Entry Pointing to the Head

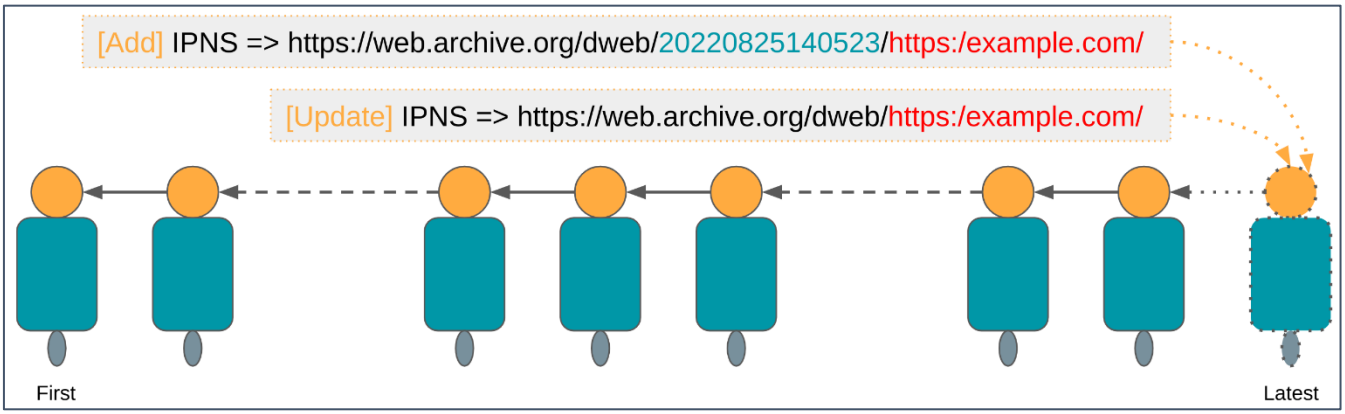


Fig. 3: Adding a New IPARO to an Existing Linked List with an Updated and an Added IPNS Entries

first observation, and number of other random IPAROs in the chain along with their corresponding observation dates and times. Reference to the previous observation of each IPARO would make a singly linked list, allowing linear traversal in the version history back in time as shown in Fig. 2. Adding reference to the first observation is useful because there is usually a significant interest in knowing when a resource was observed the very first time (e.g., to assess the age of a resource). A random set of prior version links are added for iteration efficiency and chain resilience (discussed in sections III-B3 and III-B4).

*1) Addition* - When a new observation of a resource is recorded, the corresponding new IPARO is appended to the head of the kinked list (if one exists for the given resource identifier). First, an IPNS query is performed to find the CID of the latest IPARO of the given URI (if exists). This CID is then added to the header of the newly created IPARO as the previous link reference (along with additional links like the first one of the list) and then the IPARO is added to IPFS. Finally, the IPNS entry is updated to point to the CID of the newly added IPARO as the latest observation. Moreover, an additional permalink IPNS entry is added with specific date and time to point to the newly added IPARO as shown in Fig. 3.

IPAROs are immutable objects, so adding links to future observations of a resource in prior versions is not possible. This means chain traversal will be unidirectional and easy to report the first and previous versions. However, in the next section we discuss how it is still possible to report the last and next versions of a given IPARO.

*2) Routing* - Traditional web archives, such as Wayback Machine, support the following templates of archival playback URIs (where URI-R is the original resource URI):

```
<BASEURL>/*/<URI-R>
    [TimeMap/Calendar view]
<BASEURL>/<URI-R>
    [Redirect to the last memento]
<BASEURL>/<DATETIME>/<URI-R>
    [Permalink to the DATETIME version]
```

To support the first of these three representations, we will need the list of CIDs of IPAROs of all the versions of a given original resource URI and their corresponding archival dates and times. We talk about collecting this data efficiently in the next section.

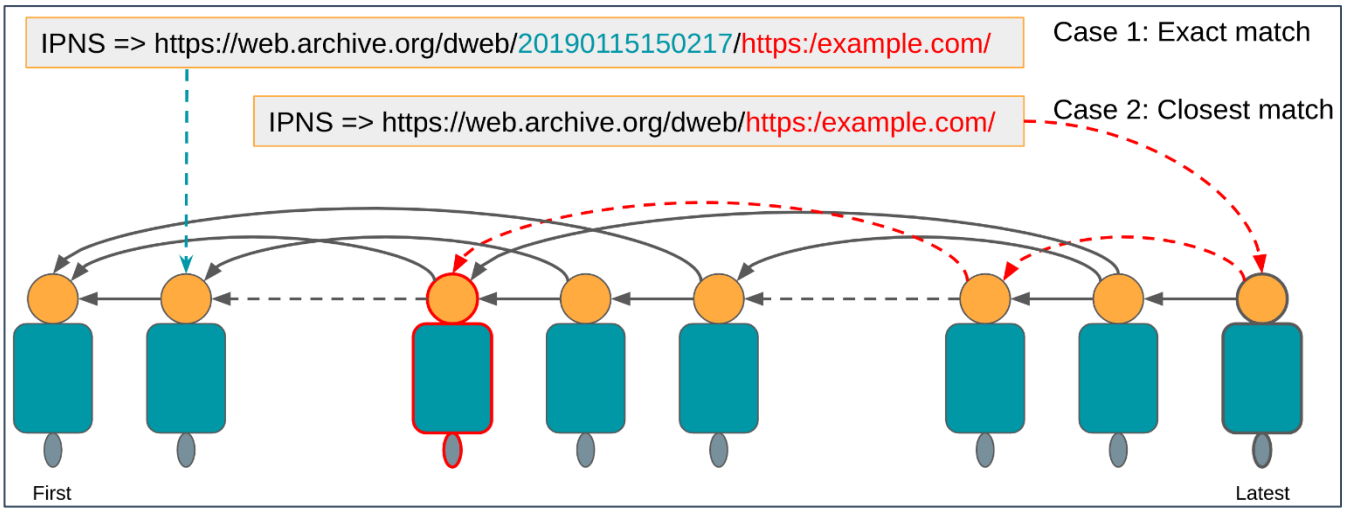


Fig. 4: Illustration of Discovering an Exact or Closest IPARO w.r.t. a Given Date and Time

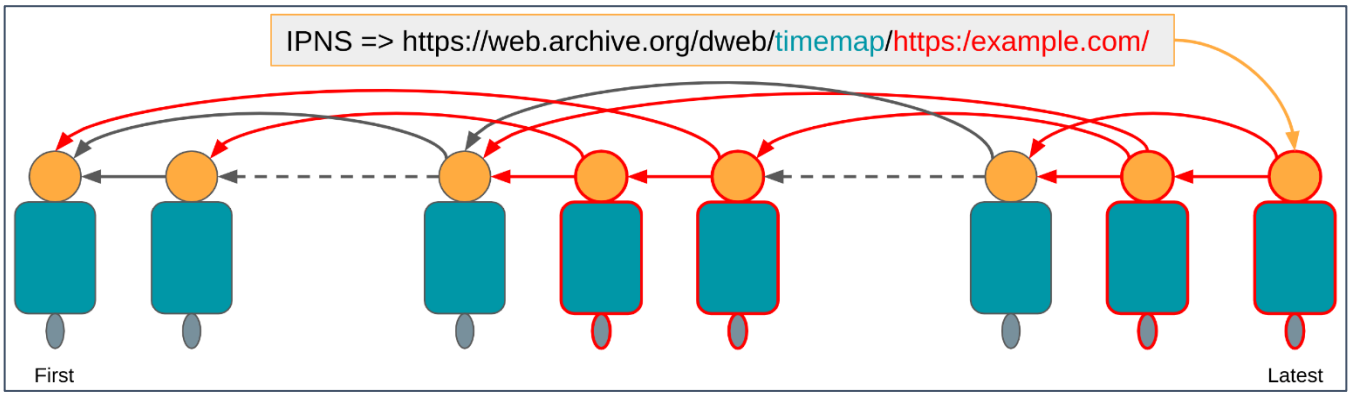


Fig. 5: An Efficient Approach to Construct a Complete TimeMap by Fetching the Linked List Only Partially

The second and third URI templates can be supported using IPNS. When an IPARO is stored in IPFS, an IPNS entry of the third URI form is created that points to the CID of the IPARO. This IPNS entry serves as a permalink to that version. If the newly created IPARO is also the most recent version of the resource, then an IPNS entry of the second URI form is created or updated (if one already exists) to point to it. This way, there is always a quick way to access the most recent version (or the head of the linked list) of every archived resource. Once we have the head of the linked list, we can access all the prior versions of the resource by following the chain via the "Link" header.

If an archived resource is not present at the exact `<DATETIME>` of the third form of the URI, traditional web archives do not treat it as a permalink. Instead, they try to find the closest archive to that datetime in their collections and redirect to a more appropriate permalink (with a different datetime). In order to support this use case, we will need some traversal of the chain, that we discuss in the next section.

In traditional web archives it is common to have more than one observation of a given resource with the same datetime, especially when the requests are redirected to a canonical form (e.g., http/https or www/apex origin changes) of the URI by the original host within a second. It is important to maintain the IPNS mapping of the third form of the URI and treat it as immutable. To minimize such collisions, we can use a finer temporal granularity, which is an information that is often recorded, but is not exposed in the URI or index of the traditional web archives for historical reasons (also, HTTP semantics support temporal granularity of one second). If we encounter a collision, despite a finer temporal granularity, we can link IPAROs with "see also" semantics, instead of overwriting the IPNS mapping.

It is worth noting that this decentralized model of archiving democratizes web archiving and makes it available as long as operators own a domain name for IPNS record mappings. It allows individual players

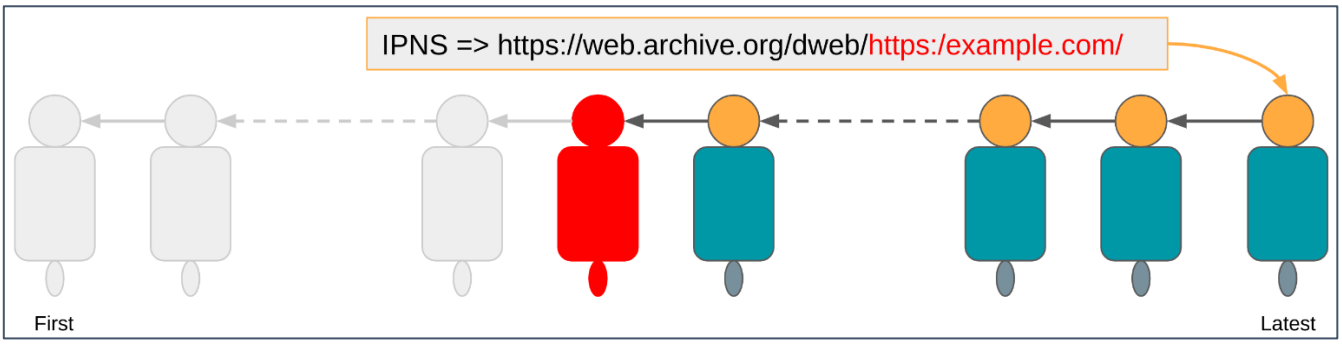


Fig. 6: An Incomplete Linked List Traversal Due to Inaccessible IPAROs with Only the Previous Links

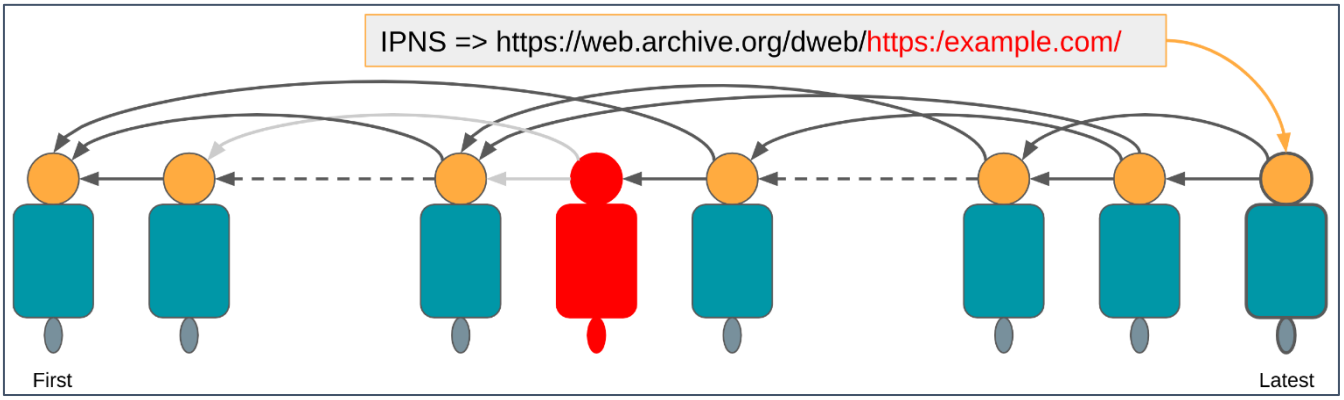


Fig. 7: A Resilient Linked List with Multiple Links to Prior Versions

to have their own chains of observations of the web while allowing memento aggregators [27-28] to mix and match consolidated web archives from many different sources either on-the-fly or by creating IPNS entries under their domains (or even by caching popular resource chains locally).

*3) Iteration* - So far, we have established that we can have quick access to the most recent archival version of a resource using IPNS and the very first version by accessing any version and reading the appropriate link reference. Moreover, we can also access any IPARO if we know its IPNS permalink as shown in Fig. 4 (Case 1). However, there are times when an application needs a list of all the versions of a resource or discover a version closest to a given date and time. This is possible by linear traversal of the linked list, starting from the most recent version and going backward, one version at a time, using the previous version references. However, this process would be slow for resources with too many archived versions. Also, if any IPARO in the linked list is inaccessible then prior versions will not be reachable.

A more performant approach would be to leverage all the memento link relations stored in IPAROs, not just the one that points to the previous version. For example, to construct the TimeMap (i.e., the list of all or most of the observations), we can: 1) start with the most recent IPARO of the given URI and fetch its headers, 2) add CIDs and corresponding date and times to a set of discovered versions, 3) pick a random CID, that is not yet fetched, and fetch its headers to discover more datetime/CID pairs to be added to the discovery set, and 4) repeat the process until growth of the discovery set stops or slows down significantly. Depending on how densely the "Link" header of IPAROs is populated with prior records, it may require fetching only a fraction of the chain to know all the datetime/CIDs pairs as shown in Fig. 5.

Similarly, to find a IPARO closest to a given date and time: 1) start with the most recent versions and collect all the CID/datetime pairs from its headers, 2) then select the smallest timestamp from the set that is greater than or equal to the desired timestamp and fetch headers of that IPARO, and 3) repeat the process until an exact match is found or all the timestamps discovered from the IPARO are smaller than the desired one. Either that IPARO or the one before it (referenced as previous link) will be the closest one as shown in Fig. 4 (Case 2). This is a greedy algorithm for the task, but there can be more efficient approaches depending on what strategies were used to populate the "Link" header of IPAROs at the time of their creation.

*4) Resilience* - It is not guaranteed in a peer-to-peer network that all the nodes containing historical IPAROs of a resource be available and accessible all the time. If the data is not replicated on enough nodes, it is possible that some IPAROs may not be reachable when iterating over the linked list backward using the previous version references as shown in Fig. 6. This is where a strategically selected set of references to various prior versions stored in every IPARO can play a critical role in avoiding roadblocks or broken chains. Those additional links will allow jumping past the missing link of the chain and continue the iteration as shown in Fig. 7, resulting in the loss of just a few IPAROs.

There is an inherent trade-off between storage overhead vs. resilience of the chain and speedy reconstruction of TimeMaps. On the one extreme, each IPARO may only store the reference to its predecessor to have small overhead but has the risk of broken chains. On the other extreme, each IPARO may store references to all the IPAROs prior to it for a given resource, making it extremely resilient, but having an $O(N^2)$ storage overhead. An optimal configuration can likely be achieved with constant (or amortized constant) reference storage overhead with strategic selection of candidates. Various policies should be evaluated in the future to see which ones work well.

### C. *Namespacing*

Namespacing is an alternate approach of web archiving and replay without an explicit local index. It is orthogonal to the approach established with the linked list and IPNS. This approach is designed to work on small collections, stored in local IPFS nodes (unless there is an API to perform query for CID prefixes in the peer-to-peer network).

*1) Media Type Namespace* - In section III-A we described the generic concept of IPMTs and discussed a few media types specific to web archiving. An IPFS node may store all sorts of data. In order to group the data of our interest, we can attach a namespace to them. In this case, we can assign a specific bit sequence at a specific part of the CID (e.g., first six bits be "*010011*") to be associated with certain media types of our interest as shown in Fig. 8. When creating an IPARO we add a nonce at the end of it and keep changing the value of the nonce until the generated CID of the IPARO meets the archival namespace bitmap. We add nonce at the end for efficient CID calculation, adding it in the header section of an IPARO would require hashing everything each time a nonce is changed. Similarly, for other applications and media types some other namespaces can be assigned. Now, when a web archiving system interacts with the IPFS store, its sample space is smaller.

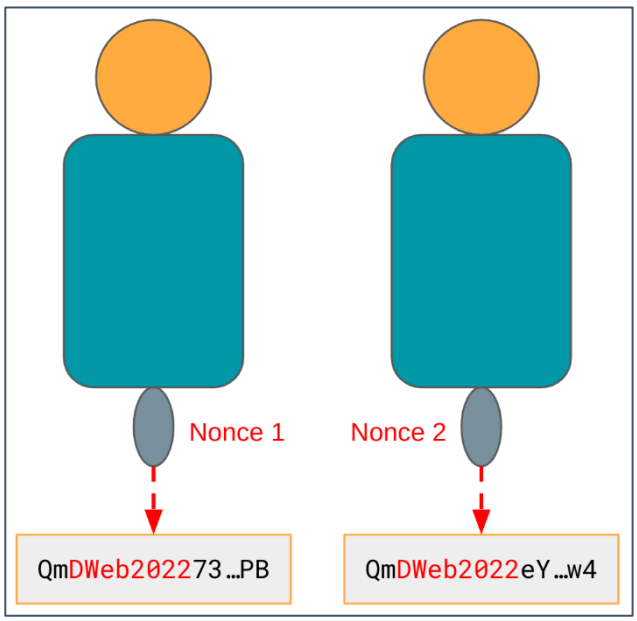


Fig. 8: Two IPARO CIDs With a Desired Common Substring

*2) URI Sub-Namespace* - To make the lookup space even smaller, we can further assign a few bits as a sub-namespace for original resource URIs. The resource URI is first canonicalized, then hashed, and finally the first few bits of the hash are selected to be used as the sub-namespace (e.g., a URI's hash might have the first ten bits as "*1110010110*"). In this case, the IPARO will be stored with a nonce value that causes the CID to have 16 bits of prefix of "*0100111110010110*".

At the time of replay, a similar calculation can be performed to identify the candidate namespace then the system can query for all the CIDs of the node that have the desired bitmap match. This will likely be a small set of nodes, which can then be inspected on-the-fly to discard any irrelevant IPAROs.

*3) Collisions* - Such bitmap-based namespacing is prone to collisions, especially, when the namespace is short and the IPFS node contains too many objects. A larger namespace will minimize the probability of collisions but will come with the added compute cost of finding the right nonce value at the time of IPARO creation. This compute cost would double, and the

probability of collisions would halve with each bit of increment in the namespace. Any remaining collisions must be identified and discarded on the fly at the playback time.

### *D. Composition and Decomposition*

So far, we have treated IPAROs as a blob containing some headers, the actual IPMT payload (described in section III-A), and some trailers like nonce (as described in section III-C). However, it might be more space-effective to split the blob in strategically identified chunks and store them in pieces in the IPFS store. At the time of playback, those pieces need to be put together to form the complete IPARO.

*1) Deduplication* - In our previous exploration, in IPWB, we split HTTP response records in two parts, headers and payload, and stored them separately. Consequently, our index had two CIDs for each observation. We did it because frequent observations of the same resource might result in the same payload or even the same payload can be seen across resources (such as soft-404 responses [29]). However, each HTTP response contains a "Date" header which indicates the time when the response was generated, which often changes every second. By splitting headers from payloads, we were able to achieve deduplication on payload while storing unique headers each time.

In this work we propose a flexible approach of chunking IPAROs. At the time of storing an IPARO, one may choose to store it as a single object or split it in as many chunks as desired. For example, one may choose to split a Memento IPARO in pieces like the IPMT header, request headers, any request payload, selected response headers that are often unchanged, remaining changing headers, response payload, and trailers as shown in Fig. 9. If the application identifies parts of the payload that are common across other responses, it can split the payload on those strategic points as well to achieve increased deduplication as the collection grows.

*2) Substitution by Reference* - In order to put the pieces of an IPARO together (if it is stored as chunks), one possibility is to use a templating language and replace extracted pieces in the container object with their corresponding CIDs. At the time of replay, first fetch the container object, process it to find any inline references, fetch those pieces, and replace them in-place to form the original complete IPARO.

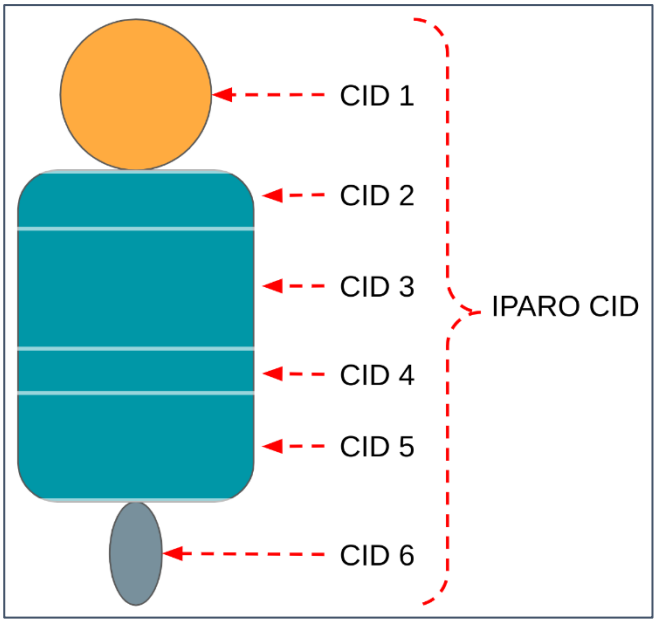


Fig. 9: Strategic Blocks for Efficient Merkle DAG Deduplication

This approach would require a rehydration step to process each IPARO and populate any inline references. Using a templating language will come with the additional overhead of placeholder markers, any escape sequences, and the space needed for CIDs.

*3) Concatenation* - This approach leverages a feature of IPFS in which it is possible to tell the system to create blocks of desired sizes instead of using the default block size to construct the Merkle DAG. To store data with strategically designated block sizes, the pieces of an IPARO are determined as described in section III-D1, then those pieces are stored one by one as independent objects as shown in Fig. 9 (CID 1-6). In the next step the whole IPARO is stored, but this time the system is told to have blocks of provided sizes (that align with the split points). This way, the system realizes that all the blocks already existed in the Merkle DAG, so no more data is stored, but a new CID is returned that represents the complete IPARO as shown in Fig. 9 (IPARO CID) [30]. At the time of replay, no additional post-processing or rehydration is needed as fetching the composite CID will return concatenated data as if it were a single piece of data in the IPFS store.

## IV. Conclusion and Future Work

We have proposed a decentralized version tracking system using the existing primitives of IPFS and IPNS. While our description talks primarily about archived web pages, we have proposed the concept of IPMT and namespacing so that it can be used in

other applications that require versioning, such as a wiki or a collaborative code tracking system. Our proposed system does not rely on any centralized server for archiving or replay of the content. The system continues to allow aggregators to play their role from which both large and small archives can benefit and flourish. Future plans include a proof-of-concept implementation and evaluations of various aspects.

## Acknowledgments

We thank Juan Benet from Protocol Labs, Dr. Mat Kelly from Drexel University, and Dr. Michael Nelson from Old Dominion University for their initial reflections on the idea and reviews. We thank Ilya Kreymer from Webrecorder Software for timely mention of the possibility of storing data in IPFS with custom block sizes. Also, we thank Irakli Gozalishvili from Protocol Labs for the reference to the Bamboo data structure. This work is supported in part by Protocol Labs and Filecoin Foundation, we thank Dietrich Ayala for the coordination.

## REFERENCES

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

[2] S. Alam, M. Kelly, and M. L. Nelson, "InterPlanetary Wayback: The Permanent Web Archive," in Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries, ser. JCDL '16, 2016, pp. 273–274.

[3] 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, 2016, pp. 411–416.

[4] 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.

[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, "IPNS Blockchain," https://github.com/oduwsdl/IPNSBlockchain, 2018.

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

[8] S. Alam, M. Kelly, M. C. Weigle, and M. L. Nelson, "Client-Side Reconstruction of Composite Mementos Using ServiceWorker," in Proceedings of the 17th ACM/IEEE-CS Joint Conference on Digital Libraries, ser. JCDL '17, 2017, pp. 237–240.

[9] M. Kelly, M. L. Nelson, and M. C. Weigle, "Making Enterprise-Level Archive Tools Accessible for Personal Web Archiving," https://www.slideshare.net/matkelly01/makingenterpriselevel-archive-tools-accessible-for-personal-webarchiving, 2013.

[10] J. A. Berlin, M. Kelly, M. L. Nelson, and M. C. Weigle, "WAIL: Collection-Based Personal Web Archiving," in Proceedings of the IEEE/ACM Joint Conference on Digital Libraries (JCDL), 2017, pp. 340–341.

[11] I. Kreymer, "Webrecorder Tools," https://webrecorder.net/tools, 2020.

[12] J. M. Patel, "Introduction to Common Crawl Datasets," pp. 277– 324, 2020.

[13] M. Phillips and S. Alam, "Hosting the End of Term Web Archive Data in the Cloud," https://netpreserve.org/ga2022/wac/abstracts/#Session_9, 2022.

[14] 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, vol. 17, no. 3, pp. 223–238, 2016.

[15] 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/IEEECS Joint Conference on Digital Libraries, ser. JCDL '19, 2019, pp. 172–181.

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

[17] A. Meyer, "Bamboo," https://github.com/AljoschaMeyer/bamboo, 2019.

[18] IPLD, "InterPlanetary Linked Data (IPLD)," https://ipld.io/docs/, 2021.

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

[20] I. Kreymer and E. Summers, "Web Archive Collection Zipped (WACZ)," https://specs.webrecorder.net/wacz/1.1.1/, 2021.

[21] J. Odvarko, A. Jain, and A. Davies, “HTTP Archive (HAR) Format,” https://w3c.github.io/webperformance/specs/HAR/Overview.html, 2012.

[22] R. Sanderson, P. Ciccarese, and B. Young, “Web Annotation Data Model,” https://www.w3.org/TR/annotation-model/, 2017.

[23] 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.

[24] H. Van de Sompel, M. L. Nelson, R. Sanderson, L. L. Balakireva, S. Ainsworth, and H. Shankar, “Memento: Time Travel for the Web,” Tech. Rep. arXiv:0911.1112, 2009. [Online]. Available: https://arxiv.org/abs/0911.1112

[25] M. L. Nelson and H. Van de Sompel, “Adding the Dimension of Time to HTTP,” in The SAGE Handbook of Web History, 2018.

[26] K. Sigurðsson, M. Stack, and I. Ranitovic, “Heritrix User Manual: Sort-friendly URI Reordering Transform,” http://crawler.archive.org/articles/user_manual/glossary.html#surt, 2006.

[27] 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.

[28] R. Sanderson, H. Van de Sompel, and M. L. Nelson, “IIPC Memento Aggregator Experiment,” http://www.netpreserve.org/sites/default/files/resources/Sanderson.pdf, 2012.

[29] L. Meneses, R. Furuta, and F. Shipman, “Identifying ‘Soft 404’ Error Pages: Analyzing the Lexical Signatures of Documents in Distributed Collections,” in Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries, ser. TPDL ’12, vol. 7489, 2012, pp. 197–208.

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

[31] S. Alam, M. C. Weigle, M. L. Nelson, M. Klein, and H. Van de Sompel, “Supporting Web Archiving via Web Packaging,” Exploring Synergy between Content Aggregation and the Publisher Ecosystem (ESCAPE) Workshop, 2019. [Online]. Available: https://arxiv.org/abs/1906.07104.

[32] A. Potsides, M. Rataj, S. Loeppky, D. Norman, and E. Lee, “State of IPFS in JS,” https://blog.ipfs.tech/state-of-ipfs-in-js/, 2022.