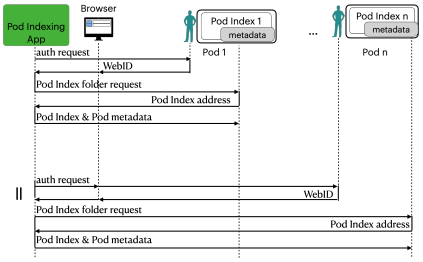

Figure 1: Indexing Process in ESPRESSO

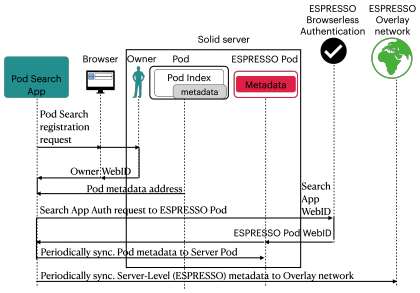

Figure 2: The process of registering a Pod for Search

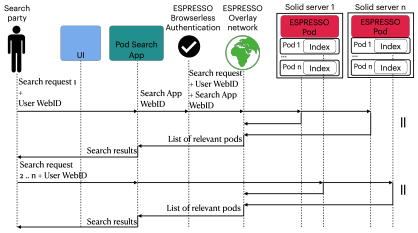

Figure 3: Search Process when overlay returns relevant pods

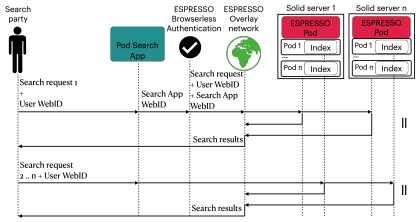

Figure 4: Search Process when overlay returns results

1 APPENDIX A

In this appendix, we provide details on how the main components of the ESPRESSO prototype system will work together to enable decentralized search over personal data stores distributed across multiple Solid servers. This discussion lays the groundwork for understanding why the ESPRESSO experimentation environment and its components are well-suited to supporting a decentralized search solution, as well as how different stakeholders could effectively engage with it.

1.1 Pod Indexing App.

The **Pod Indexing App** manages the indexing of unstructured text data in pods. Figure 1 shows a sequence diagram of the process of indexing pods. To perform this task, the Pod Indexing App requires access permission to the

pod's data, which it obtains by prompting the pod owner to log in with their WebID and pod credentials. Once authenticated, the app is granted write access to the pod's index container or directory, which it creates if necessary. The app then crawls all containers within the pod and indexes their text data. Additionally, it creates and stores a *metadata profile* inside the pod.

1.2 Pod Registration for Search

After completing the indexing process, pod owners have the option to register their pods, making them searchable by others. Figure 2 illustrates a sequence of steps in this process.

If the pod owner opts to make their pod searchable, the Search App WebID is granted access to the metadata profile within the pod, which is added to the *Access Control List* (ACL) of the metadata. This metadata structure, described in Section **??**, aggregates metadata information from all the server pods' indexed profiles, detailing which UUIDs (WebIDs) have access to specific keywords based on their visibility scopes for files within those pods. Notably, this metadata, maintained at each Solid server in a special pod (called the *ESPRESSO Pod*) about the hosted pods, can only be accessed by the ESPRESSO Search App WebID that is granted access to the pods' metadata.

1.3 Metadata Manager

The Metadata Manager will periodically retrieve and consolidate the metadata about pods from each server, integrating it with the pods' metadata collected from other servers. This integrated metadata is stored within the ESPRESSO Pod. This will enable us to maintain System-Level Metadata about servers, enabling efficient search operations across decentralized pods. The Metadata Manager will use the Search App's authorized WebID to periodically access and copy those servers' meta-information from ESPRESSO Pods at each server and store the locations of the Server-Level Metadata in the logical tables of the Overlay Network. Notably, metadata maintained in the Overlay Network logical tables will not reveal any sensitive information, as the tables relate the WebIDs to the ESPRESSO Pod URLs that contain Server-Level Metadata information accessible to those WebIDs. The ESPRESSO Pod URLs can only be accessed by the ESPRESSO Search App WebID.

1.4 Pod Search App.

The Pod Search App is the second ESPRESSO Solid-compatible application that enables the search party to conduct searches across Solid pods registered in the Overlay Network. Search parties - users, including agents or bots - initiate searches through a dedicated User Interface (UI), submitting a search request (keyword) along with a WebID. The Search App processes these requests by utilizing the Overlay Network to perform searches across Solid servers, ensuring that results include only accessible content associated with the User's WebID.

The app employs two primary alternative methods to utilize the support of the GaianDB network.

In the first approach, the Search App utilizes GaianDB as a distributed storage layer for storing the System-Level Metadata about servers, maintained across GaianDB nodes within the Overlay Network logical tables. Thus, the Overlay Network responds by providing a curated list of relevant servers to the Search App. Subsequently, the Search App accesses the Server-Level Metadata about pods, through their respective servers, to get the relevant pods. For each relevant pod, the Search App will access the WebID-specific Pod Index, and get the results accessible by that WebID. Thus, the Search App needs to conduct subsequent HTTP GET requests to the relevant Pod Index folders, utilizing permissions granted to the search party WebID. This process allows the Search App to retrieve results based on the specified search party WebID, facilitating efficient information retrieval across the network.

For the second search approach, the Search App leverages the Overlay Network to transmit both the query and the WebID to the GaianDB node. From there, the GaianDB node proceeds to distribute and propagate these requests across the Overlay Network nodes. The GaianDB Overlay Network then aggregates the results of these queries and returns them directly to the Search App user interface. This method streamlines the process of querying and retrieving information by centralizing the aggregation of results through the GaianDB node, enhancing efficiency in data retrieval for the Search App.

2 APPENDIX B

In this Appendix, we discuss the proposed approach ensures the conservativity of metadata stored in the ESPRESSO pod.

2.1 Conservativity property of Metadata

A set of users'/applications' known WebIDs: $(W_1, \ldots, W_u, E_I, E_S)$, where E_I is the WebID of the ESPRESSO Indexing App and E_S is the WebID of the ESPRESSO Search App. A set of known Solid Servers: $(S_1, \ldots, S_v)$.

A Solid Server $S \in \{S_1, \ldots, S_v\}$ contains a set of pods $P_1, \ldots, P_n$ identified by URLs $URL_1, \ldots, URL_n$ that the ESPRESSO Indexing App has permission to index.

The ESPRESSO Search App has access permission to a subset of these pods $\{P_1, \ldots, P_m\} \subseteq \{P_1, \ldots, P_n\}$; specifically, those pods that were registered as searchable.

Consider a search query (w, q) where q is the query and w is the WebID of the search party, $w \in \{W_1, \ldots, W_u\}$. When this query (w, q) is submitted by the Overlay Network node N_S local to the Solid server U to the ESPRESSO Search App running on U, it returns to N_S the set of URLs $\{URL_1, \ldots, URL_k\}$ corresponding to the pods $\{P_1, \ldots, P_k\} \subseteq \{P_1, \ldots, P_m\}$ that (i) match the search query (i.e., Keywords) and (ii) to which WebID w has access permission.

The ESPRESSO Search App consults the Pod Indexes corresponding to the pods $\{P_1, \ldots, P_m\}$ (to which it has access permission). Let $\text{Result}(w, q, S)$ denote the result returned by the ESPRESSO Search App running on a Solid server U to the local Overlay Network node N_S for a search query (w, q).

Consider a set of queries $(w_1, q_1), (w_2, q_2), \ldots, (w_r, q_r)$ submitted by N_S to the ESPRESSO Search App running on U. These will return a set of results as follows:

$$\text{Result}(w_1, q_1, S), \text{Result}(w_2, q_2, S), \ldots, \text{Result}(w_r, q_r, S)$$

2.1.1 Server-Level Metadata

Herein, the meta-information data structures that are created and maintained by ESPRESSO on a Solid Server U can be computed by a function

$$f_S\left((w_1, q_1, \text{Result}(w_1, q_1, S)),\right.$$

$$\left.(w_2, q_2, \text{Result}(w_2, q_2, S)), \ldots, (w_r, q_r, \text{Result}(w_r, q_r, S))\right)$$

In other words, no data is maintained in the ESPRESSO meta-information data structures on U that cannot be computed by submitting a set of search queries to U and applying a computable function to the set of results.

For example, for the pods' metadata, the metadata computation function can be as follows:

$$f_S = \{(w_1, q_1, n_1), (w_2, q_2, n_2), \ldots, (w_r, q_r, n_r)\}$$

where $n_i = |(\text{Result}(w_i, q_i, S)|$ represents the count of servers containing the pod associated with WebID w_i for query q_i. Notably, the function f_S can vary for different servers U.

2.1.2 Overlay Network Metadata

Also, the claim on the non-local meta-information stored by the Overlay Network N at each Solid Server U at time t is a function

$$f_{N,S}\left(t, MS_1(t_1), \ldots, MS_v(t_n)\right)$$

is the aggregate of local meta-information stored at Solid servers MS_i at time $t_j \leq t$. Notably, the function $f_{N,S}$ may be different for different servers U on a given network N and may vary across different networks N (i.e., it is dependent on N and U as well as on the local meta-information stored throughout the network).