Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1085–1094
DOI: 10.15439/2017F114
ISSN 2300-5963 ACSIS, Vol. 11

# DNS as Resolution Infrastructure for Persistent Identifiers

Fatih Berber* and Ramin Yahyapour*‡
*Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen (GWDG), Germany
‡University of Göttingen
{fatih.berber, ramin.yahyapour}@gwdg.de

***Abstract*—The concept of persistent identification is increasingly important for research data management. At the beginnings it was only considered as a persistent naming mechanism for research datasets, which is achieved by providing an abstraction for addresses of research datasets. However, recent developments in research data management have led persistent identification to move towards a concept which realizes a virtual global research data network. The base for this is the ability of persistent identifiers of holding semantic information about the identified dataset itself. Hence, community-specific representations of research datasets are mapped into globally common data structures provided by persistent identifiers. This ultimately enables a standardized data exchange between diverse scientific fields.**

**Therefore, for the immense amount of research datasets, a robust and performant global resolution system is essential. However, for persistent identifiers the number of resolution systems is in comparison to the count of DNS resolvers extremely small. For the Handle System for instance, which is the most established persistent identifier system, there are currently only five globally distributed resolvers available.**

**The fundamental idea of this work is therefore to enable persistent identifier resolution over DNS traffic. On the one side, this leads to a faster resolution of persistent identifiers. On the other side, this approach transforms the DNS system to a data dissemination system.**

## I. Introduction

THE massive growth of digital data in many different areas including the scientific area, has driven a series of profound changes in the world. For commerce, this data deluge for example provides a door for new markets. By analyzing the purchase patterns of specific buyer groups, it is possible to subject them with pinpointed advertisements which ultimately could lead to a strong increase of the profits.

In the scientific area for instance, the increasing volume of research datasets has led to an increase of their importance. The overall goal in research data management is to provide a sustainable cross-disciplinary exchange between different scientific branches. This could ultimately help to enable the discovery of new insights in various scientific fields.

The concept of persistent identification is becoming a fundamental component for research data management. Its basic function is to provide a sustainable access to research datasets, which are currently retrievable by their locators. Even small technological changes in a research data repository could lead to many invalid locators. Hence for a long-term sustainable access, locators are highly inappropriate. Since research datasets are currently mapped into web resources, the locators are URLs. Thus, persistent identifiers currently provide an abstraction for URLs corresponding to individual research datasets.

However, with the explosive research dataset growth, the count of registered persistent identifiers is increasing enormously as well. Therefore, persistent identifier systems are increasingly subjected to high loads. Since persistent identifiers are an essential component for research data management, the performance of persistent identifier systems is highly critical for research datasets exchange. The focus of this work is therefore on the performance of the resolution procedure for persistent identifiers.

The importance of persistent identifiers for global research dataset exchange is comparable to the importance of the well-known DNS system for the current general Internet communication. The resolution procedure for both, persistent identifiers and domain names, is in principle a node traversal procedure, whereas the traversal starts at a specific root node and terminates at a responsible child node. Hence, the resolution time for domain names and persistent identifiers are generally composed of the network latencies between the traversed nodes. Moreover, the resolution procedure involves a specific proxy resolver which is tasked with the node traversal. A requesting application only submits a single resolution request to the proxy resolver, which then responses with the answer obtained from the traversal procedure. To reduce the traversal procedure, the answer of frequent resolution requests is usually cached at the proxy resolvers. In such a case, the resolution time only consists of the network latency between requesting application and proxy resolver.

For DNS, there are a myriad of publicly available proxy resolvers, such that an application can always choose a proxy resolver in its proximity. In contrast to that, the count of persistent identifier proxy resolvers is in comparison to the count of DNS proxy resolvers infinitesimally small. For the Handle System, which can be considered as the most important and established persistent identifier system, there are currently only five globally distributed proxy resolvers. Therefore, the fundamental idea of this paper is to enable the resolution of persistent identifiers over DNS proxy resolvers. Due to the global widespread of DNS proxy resolvers, with this approach, the resolution time for persistent identifiers can be significantly reduced.

The remainder of this paper is structured as follows. Section II gives an overview of the related work. In Section III we will analyze the core idea behind the concept of persistent identification. In addition, we will discuss the different possibilities to use DNS as resolution system for persistent identifiers. In Section IV, we will provide a realization of our proposed idea for the resolvability of Handle persistent identifiers over DNS traffic. Finally, in Section V we will do an evaluation of the proposed idea by means of an experimental setup. Ultimately, in Section VI will present our conclusions and give hints for future work.

## II. Related Work

Persistent identifiers are in principle widely acknowledged for research data management. However, their importance is significantly increasing due to the ability of imposing semantic information into the persistent identifier record itself. Therefore, the perception for persistent identifiers increasingly moves from a simple redirection mechanism towards a global data structure for research datasets, which ultimately enables information exchange between diverse research data repositories.

A prime example for an early perception for persistent identifiers is the work [1]. The authors consider the concept of persistent identification only as a redirection mechanism. Therefore, their focus is to devise a bridge from persistent identifiers to HTTP URIs, which are associated with semantic information about research datasets. But they do not take into account the possibility of persistent identifier records for holding semantic information.

The work [2] proposes *trusty URIs* for providing trust and reliability for scientific datasets. The focus is on using cryptographic hash values in URIs corresponding to identified scientific datasets, so called *trusty URIs*. A *trusty URIs* can then be used to determine whether the identified datasets has been subjected to manipulations. However, it becomes critical for this approach when the identified research datasets moves to another location.

One of the first works which considers persistent identifiers as a more complex concept than just a redirection mechanism is given by [3]. The basic approach is an ontological refinement of metadata sets contained in persistent identifier data records, which enables a common information set for the various persistent identifier systems.

The first work which considers persistent identifiers as a data structure for semantic information is provided by [4]. The focus is the integration of common abstract datatypes into persistent identifier data records, which can be consumed by machine actors. In contrast to [3], their core emphasize is to provide a standardized set of information entities about the identified dataset itself.

Due to the versatility of persistent identifiers, they are also considered in various other fields. An example for that is the work [5]. The authors are discussing the usage of persistent identifiers in the field of Named Data Network (NDN). Their focus is to use existing persistent identifier concepts within NDN environment for delivering big datasets.

Another example is from the field of scientist identification, the authors in [6] introduce the concept of persistent identification for scientists. In their concept, an individual persistent identifier data record corresponds to a single scientist, which also contains the respective bibliography.

However, the versatility of the concept of persistent identification can also be important for the Internet-Of-Things paradigm. For IoT, persistent identifiers can be more than just a naming component, in fact, they are perfectly suitable as the global platform for device communication. The work [7] is an example for the need of a naming component in IoT. The main idea is in principle to provide a DNS-like system specifically targeted for IoT devices. However, this is de facto already existing: The Handle System, which can be considered as the most sophisticated persistent identifier system, is already productively in use. In addition, the data structure of the Handle System is generic enough to hold any kind of data including device information. Another major advantage of the Handle System is that there are guarantees for a long-term operation.

Ultimately, all the aforementioned research efforts do not specifically address the performance of the concept of persistent identification.

In a previous work [8], we have proposed a high-performance identification concept for huge research data repositories, which already provide a sophisticated data structure and immutable identifier scheme for its research datasets. However, in that work we did not emphasize on the performance of the resolution of persistent identifiers, which is the focus of this current work.

Persistent identifiers are very well comparable with domain names of the DNS system. In both systems, basically a name is registered once and resolved many times. Therefore, the concepts which are applied for accelerating domain name resolution are also suitable for persistent identifier resolution.

The resolution of domain names can be considered as a latency problem, which is caused by the traversal of the hierarchical architecture. Therefore, in order to shorten the traversal path, usually aggressive caching is applied. The authors [9] focus on analyzing the resolution performance from the client point of view. In addition, they analyze the effectiveness of caching.

A special caching strategy of DNS records is proposed by [10]. The authors introduce a concept for proactive caching of expired DNS records, whereas particular DNS records are unsolicited refreshed before clients queries them.

Usually, individual DNS zones are composed of multiple servers for ensuring high-availability and load-balancing. Since, the domain name resolution procedure can be considered as a node traversal problem, an optimal choice of the appropriate nodes can significantly improve the performance. Thus, the impact of the DNS server selection algorithms of popular DNS proxy server implementations is provided by the work [11].

Often, nameservers of top-level domain zones are also geographically distributed. Thus, in order to redirect a domain name resolution request to the nearest nameserver, usually the technique of anycast is applied. Hence, the work [12] analyses the effectiveness of anycast for DNS server selection, whereas it reveals the general usefulness of anycast.

However, in content distribution networks (CDNs) anycast alone usually does not suffices for efficiently redirecting user requests to appropriate CDN servers. Since in anycast the routing decision is done by the network, which is based on the static hop count between different autonomous systems, dynamic parameters such as the current load at individual servers are not covered by anycasting. Therefore, in CDNs the request routing is based on special DNS servers which are basically tasked with the collection of routing relevant parameters. The collected parameters are then used to determine the most appropriate CDN server. Thus the work [13] analyzes different server selection algorithms in CDNs which are based on DNS. A similar work is provided by [14]. It analyses the functioning the Akamai infrastructure, which is one the largest CDNs. In addition, by conducting a comprehensive measurement study it reveals the complexity in CDNs.

However, currently the fundamental problem of persistent identifier resolution is caused by the very few distribution of resolution systems. For DNS, in contrast, there are countless resolvers globally distributed. Hence, the fundamental idea of this paper is to make use of these DNS resolvers for persistent identifier resolution.

## III. The Concept of Persistent Identification

Since in the current Internet all resources are retrieved by their locators instead of their names, for a sustainable access temporary locators are highly inappropriate. This is especially true for research datasets. In order to prevent research datasets from being lost in the huge data deluge, research datasets are more and more interlinked with each other. In addition, data is increasingly consumed by machine actors instead of humans. For machine consumption it is necessary to employ a common data structure together with a common understanding for the elements in that data structure. Persistent identifiers initially have been conceived for providing an abstraction for locators of resources in the current Internet. The working principle of persistent identifiers is simply to map an opaque name, which is assigned to an individual research dataset, to its current locator. Another aspect of persistent identifiers is that they also enable the imposition of semantic information about research datasets. This aspect significantly increases the importance of persistent identifiers for research data management. Therefore, the registration procedure of research datasets at persistent identifier systems can be considered as a mapping from a community-specific into a standardized global representation. Whereas the global representation is used by various different machine consumers, which can access research datasets just by their globally unique names without further knowledge about their current Internet locations. In addition, these machine consumers can autonomously operate with research datasets based on the imposed semantic information.

In its fundamental working principle, the well-known DNS system is very much comparable to a persistent identifier system. In both systems, an information entity is first registered and secondly resolved. In the DNS system, the information entity is basically an IP-address of a computer host which then is assigned a human-friendly representation for its IP-address, namely a domain name. In the case of persistent identifiers, the information entity first of all consists of a locator for a research dataset. As mentioned earlier, the information entity also increasingly includes much more complex information about the identified research dataset. Therefore, the concept of persistent identification can also be considered as a technology, which enables the realization of a virtual global research data network, wherein the communication between the actors is realized by persistent identifiers and whereby the actions are derived from the corresponding information entities.

However, the increased importance of persistent identifiers has also led to a steadily increasing load at persistent identifier systems. To provide a reliable communication between various the actors, a performant resolution of persistent identifiers is therefore highly important. Due to the global distribution and the hierarchical architecture, the resolution of persistent identifiers is usually a latency problem. As it is the case for DNS, the resolution is accomplished by a node traversing, starting from the root node and ending at the responsible node for an individual persistent identifier. The mature DNS system is a fundamental component of the current Internet, therefore its performance is highly critical for the whole Internet communication. In order to ensure a reliable and performant functioning, for the DNS system several techniques are in use. A rough categorization of these techniques is given by the following:

a) Caching is primarily applied to hold the data in a faster storage, but in the particular case of DNS, it is applied to shorten the traversal path. Whereby the answer for frequent resolution requests is tried to be cached in a proximity node of a requestor.
b) In order to ensure high-availability and robustness an individual DNS zone usually consists of multiple nameservers, where upon the incoming resolution requests are distributed on. This is actually better known as load-balancing.
c) In addition to load-balancing, the technique of anycast is often used to redirect the requests to the nearest possible DNS server. Anycast is especially targeted for zones which consists of multiple geographically distributed nameservers. The top-level domain (TLD) zones are a typical example for that. The ".de" TLD zone for instance, consists of many nameservers which have been globally positioned. A request submitted by an application, which is hosted in Europe will be redirected to the European cluster of nameservers responsible for the ".de" zone. In contrast to that, a request originating

from the USA, will be answered by the nameservers located in the USA.

In anycast, multiple nodes are reachable by exactly the same IP-address. As an example, the public Google DNS resolver is reachable by the IP-address 8.8.8.8. However, since the public Google DNS resolver is globally distributed among the Google data centers, a request of an individual client will be redirected to the nearest Google DNS resolver. For anycast the request routing decision is made at the network switch level, which choose the path corresponding to the shortest hop count among a set of possible other paths.

However, for content distribution networks (CDNs), request routing based on anycast is not efficient enough. This is due to the fact that static routing decisions based on the hop count do not cover the dynamic load behavior in CDNs. Therefore, in CDNs special DNS servers have proven to provide a reliable request routing for redirecting an individual requestor to the current most efficient content server. These special DNS servers are equipped with probing information collected from previous requests which are used for the request routing.

To resolve a specific domain name into its IP-address, an application usually queries its operating system's stub resolver. The stub resolver in turn, redirects the resolution request to a pre-configured DNS proxy resolver. Such a proxy resolver then traverses the hierarchical global DNS system until it reaches a DNS server, which has the corresponding answer for the resolution request. This is depicted in Figure 1.

For the Handle System, which can be considered as the most important and elaborated persistent identifier system, the resolution of individual persistent identifiers, called Handles, is in principle very similar to the resolution procedure of domain names. In contrast to domain names, an individual Handle is composed of two parts: a prefix and a suffix part. Both parts are separated by the ASCII character '/'. The prefix part is comparable with a domain name as it consists of hierarchical labels separated by dots. The suffix part in turn, is a locally unique name assigned to an individual research dataset.

To resolve a Handle, an application has to submit the resolution request to the Global Proxy Resolver (hdl.handle.net). The Global Proxy Resolver in turn, starts at the Global Handle Registry (GHR) to find the responsible Local Handle Service (LHS). This information is stored in the so called Prefix Handle at the GHR. In the next step, the resolution request is sent to that LHS. Finally, the Global Proxy Resolver responds with the corresponding Handle Record received from the LHS. The Handle resolution procedure is depicted in Figure 2.

In summary, for DNS and Handle resolution, an application has to involve a particular proxy system. The fundamental difference between DNS and Handle resolution is the fact that for DNS resolution there a myriad of public DNS proxy resolvers available. In contrast to that, for Handle resolution, the Global Proxy Resolver currently consists of only five globally distributed servers.

Hence, due to caching at the Global Proxy Resolver, as it is applied at any DNS proxy resolver, the resolution time of Handles often consists to a great extend of the latency between the requesting application and the Global Proxy Resolver.

Therefore, to speedup the resolution time for Handles, it is essential to reduce the latency to the Global Proxy Resolver. The focus of this work will therefore be on the improvement of the resolution time for Handles. Since the Handle System is also a fundamental part of other important persistent identifier systems, such as the DOI System [15], an improvement of the Handle System will also affect these other persistent identifier systems. Before we will discuss the possibilities of reducing the latency between requestor and proxy system, we will first proceed with a brief comparison between DNS and Handle System.

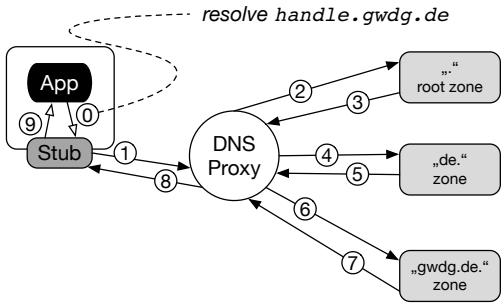


**Fig. 1:** DNS Resolution Procedure

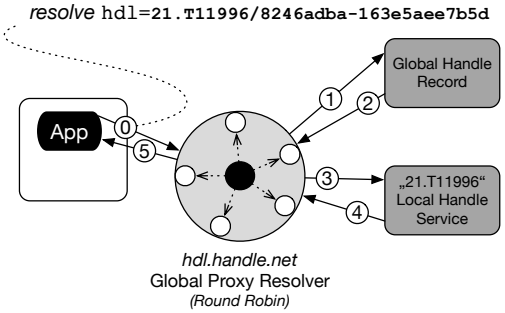


**Fig. 2:** Handle Resolution Procedure

### A. DNS and Handle System Comparison

The DNS, as well as the Handle System, can be considered as a hierarchical, distributed database. Both systems define a specific protocol to ensure the global functioning. However, in contrast to DNS, the Handle protocol offers a much richer set of operations for the management of individual Handles. In addition, the Handle protocol is equipped with its own authentication and authorization mechanism, which enables the management (provided that an individual user is authorized) of any Handle Record stored at any Local Handle Service in the world. Whereas management of DNS records is still a manual process involving an administrator's action. Another strength of the Handle System is its data structure. In DNS, the data is structured as Resource Records (see

Figure 4), whereas the *type* field and the *rdata* field are the most important fields. Obviously the *type* field denotes the type of the data in the *rdata* field. In the Handle System, an individual Handle consists of multiple Handle Values which form a Handle Record (see Figure 3). Similar to a Resource Record, a Handle Value is most importantly composed of a type and a data field. However, the essential aspect of Handle Values is that there is no restriction on a set of permissible data types as it is the case for Resource Records [16]. Thus, a Handle Value can hold any type of data, which makes the Handle System also attractive for the area of the Internet-Of-Things. The Handle System can act as a global registry for any device for storing device specific information. This in turn could enable various different devices to interact with each other by means of exchanging their corresponding Handle Records.

In summary, these characteristics underline the potential of the Handle System.

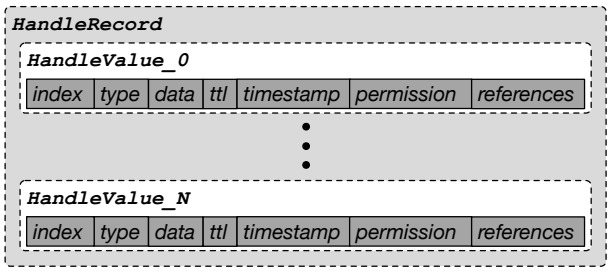


**Fig. 3:** Handle Record made of a set of Handle Values

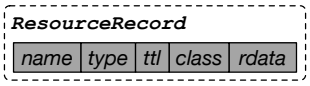


**Fig. 4:** Resource Record

### B. Proxy System Latency Reduction

To reduce the latency between an application and the Global Proxy Resolver for resolving Handles, first of all it is necessary to increase the count of globally distributed proxy servers. This could be done by a manual setup, which is theoretically possible but associated with high costs and administration efforts. Another option would be to make use of a content distribution provider such as Akamai, where the Handle proxy system could be spread on the provider's global infrastructure. However, since in CDNs special DNS servers are tasked with the request routing, for the relative simple Handle resolution operation the request routing based on DNS would again cause a considerable overhead into the overall Handle resolution time. In addition, the usage of CDNs is also associated with additional costs. The next option is to make use of an already publicly available and globally widespread infrastructure for Handle resolution, whereby the DNS infrastructure constitutes a perfect candidate for such an endeavor. The great benefit of this option is that one can make use of the myriad of globally available public DNS resolvers without additional costs and administrative overheads. However, the downside of this option is that it requires a translation process between the Handle and DNS protocol. For the translation process, there are in principle the two following options:

**(A) Translation at the DNS system:** In this case, the public DNS resolvers have to be extended by the Handle protocol in order to be able to communicate with the Handle System.

**(B) Translation at the Handle System:** Whereas in this case, both the Global Handle Registry as well as the Local Handle Service have to be extended by the DNS protocol in order to be able to communicate with the requesting DNS resolver.

The main obstacle for option (A) is based on the fact that there are various different implementations of DNS resolvers. For the realization of our proposed idea, it is necessary that all publicly available DNS resolvers have to be upgraded by a Handle protocol module, which can hardly be realized.

In contrast to that, all components of the Handle System are based on a common library set, called the Handle libraries [17], which are publicly available. The Handle libraries are in principle a reference implementation of the Handle protocol. Since our fundamental idea is to make use of publicly available DNS resolvers for Handle resolution, with option (B) there are no modifications in these DNS resolvers required, which is a major benefit of option (B). Therefore, in the remaining we will focus on the realization of option (B).

### C. Summary

Since the resolution procedure of Handles is very much comparable to the one of domain names, the resolution time mainly consists of the network latency between the different nodes which have to be traversed until the responsible node is found. For both domain names and Handles it is necessary to instruct a specific proxy resolver, which is tasked with the traversal of all the necessary nodes. However, for Handle resolution, currently there are only five proxy resolvers globally distributed. In contrast to that, for the domain names there are a myriad of publicly available DNS resolvers. This means that the resolution time of Handles suffers from the relative low density of the global distribution of the Handle proxy resolvers. Therefore, the network latency between the requesting client and the Handle proxy resolver has generally a high contribution to the overall Handle resolution time. To reduce this latency between client and Handle proxy resolver, the fundamental idea of this work is to make use of publicly available DNS resolvers for resolving Handles. This is based on the fact that the global distribution of DNS resolvers has a much higher density than the global distribution of Handle resolvers.

## IV. Implementation

In this section, we will consider the implementation of option (B) for enabling Handle resolution over DNS traffic.

The realization of this approach covers the following four parts:

(1) Handle server with DNS interface
(2) Mapping of Handle Values into DNS Resource Records
(3) Representation of Handles as domain names
(4) Appropriate Resolution Procedure

Therefore, in this section we will provide insight into all these parts.

### A. Handle server with DNS interface

Since our fundamental idea is basically to embed the Handle System into the DNS system, for the proposed solution it is necessary to extend the individual Handle servers with a DNS interface. A DNS interface enables the communication with Handle servers by means of the DNS protocol. It should be noted that the Handle libraries already include a DNS interface, which can be enabled in Handle servers for listening on DNS traffic. However, this DNS interface is intended for Handle servers to act as ordinary DNS servers. It does not enable the resolution of ordinary Handle Records over DNS traffic, which is our objective. This stems from the fact that DNS Resource Records are limited on a permitted set of data types, whereas Handle Records can contain Handle Values with any type of data. In this built-in DNS interface, the mapping direction is from DNS Resource Records into Handle Values. In addition, only Handle Values containing real DNS Resource Records can be resolved. In contrast to that, our approach is to enable the resolution of any type of Handle Values over DNS traffic. The key challenge for this endeavor is to transform Handle Values containing any type of data into Resource Records, which are actually limited on a specific set of data types. Hence, in contrast to the built-in mechanism, for our approach the mapping direction is from Handle Values into Resource Records. However, we make use of the already built-in DNS interface for the request and response encoding and decoding. The algorithm in the Handle server, however, for requests received at the DNS interface has been modified in order to realize the resolution of Handle Records over DNS traffic, which will be discussed in the following subsections.

### B. DNS Resource Records Types

The mapping of Handle Values into DNS Resource Records constitutes the core part of our approach. Initially, DNS has been conceived for providing human-friendly addressing for computer hosts, which are actually addressed by their IP-addresses. Therefore, most DNS Resource Records contain data which is specifically related to computer hosts. Among them, the Resource Record with the type ”A” can be considered as the most used type since the corresponding data field contains an IP-address of an individual computer host. As already mentioned, Resource Records are limited on a particular set of permitted data types [16], however, none of them is targeted for holding descriptive information about digital datasets (such as research datasets) or IoT devices. The ability of providing meta information would transform DNS from a simple resolution system into a data dissemination system. To realize this approach, either the permitted set of Resource Records has to be extended by additional types for the dissemination of datasets or one has to exploit specific types of the permitted set. The core problem of extending the set of permitted Resource Record types is based on the requirement that public DNS resolvers also have to support these new types. This is quite challenging since many public DNS resolvers even do not support all of the currently permitted types. Therefore, we concentrate on the exploitation of already permitted data types. For that, we have identified specific types of Resource Records, which we will analyze in the following:

**NAPTR Resource Records:** NAPTR typed Resource Records have been introduced to enable DNS to act as a rule database for the Dynamic Delegation Discovery System (DDDS) [18]. DDDS is basically an abstract concept for applications to enable resource access through applying rules on input strings. The rules are provided by the NAPTR Resource Records, which are then applied by the requesting DDDS applications to compose the locators of resources. Hence, DDDS applications are compelled to implement a specific algorithm in order to apply the rules contained in NAPTR Resource Records. In contrast to that, in ordinary persistent identifier systems the locator of a resource is retrieved directly through a resolution process.

**SRV Resource Records:** SRV Resource Records are used to describe available services on hosts. It allows to specify the service, its protocols and the corresponding ports on which the service is listening on the host.

**URI Resource Records:** The URI Resource Record is an alternative to the SRV Resource Record. It does only hold the URI for a resource. However, in order to access a digital datasets, it is often also necessary to specify the port number of the repository system.

**TXT Resource Records:** The TXT Resource Record is intended to transfer arbitrary textual information with DNS traffic. However, in principle it is possible to transfer any kind of data encoded as a character string. As such, it is pretty well suitable for transferring Handle Values.

Among the listed Resource Records, the TXT Resource Record is the most flexible one. Another aspect which is quite beneficial for our approach is given by the fact that this Resource Record is also widely supported by public DNS resolvers. The problem with the remaining Resource Records is that for them there are already defined standard procedures for their consumption, which is not the case for TXT. The general idea of using the TXT Resource Record to store key-value pairs is not new, as can be seen in [19], however, it was never considered in conjunction with persistent identifiers. Ultimately, for the realization of our approach, the TXT Resource Record is the most reasonable choice. Thus, in the next subsection we will provide the actual mapping from a Handle Value into a TXT typed Resource Record.

### C. Mapping of Handle Values into DNS Resource Records

Each Handle Value is mapped into a Resource Record with a TXT typed field. The *data* field of the Resource Record is composed of the Handle Value *type* field together with its corresponding *data* field, whereas the equal character ('=') is used as delimiter. In addition, the Handle Value *ttl* field, which is used for caching, is mapped into the *ttl* field of the Resource Record. An important aspect at this mapping procedure is that Handle Records may also contain Handle Values, which are not allowed to be consumed by the public. Hence, all Handle Values with restricted reading rights, which is recognized by the *permission* field, are discarded at the mapping procedure. Since a Handle Record can contain multiple Handle Values, the *index* field is used as an unique internal identifier within the Handle Record. This is not necessary for Resource Records and therefore the index is not considered at the mapping. The *name* field of the Resource Record is replaced by the domain name representation of the corresponding Handle identifier string. Finally, the overall mapping procedure is depicted in Figure 5.

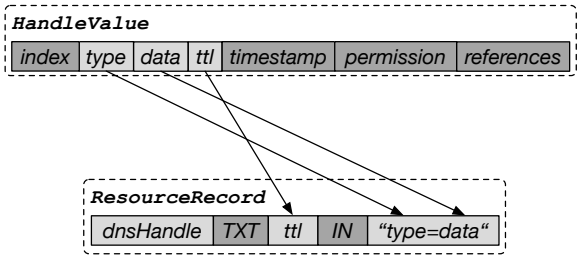


**Fig. 5:** Mapping of Handle Value into TXT typed Resource Record

### D. Representation of Handles as Domain Names

Another fundamental aspect for the resolution of Handles through DNS traffic is their representation as domain names. This is discussed in the following three points:

(1) **Character Set:**
In principal, a Handle may consist of any character from the Unicode character set. In contrast to that, a domain name can only contain letters from A to Z, the digits from 0 to 9, a hyphen (-) and a dot (.) as separator. Hence, in order to represent each possible Handle as a domain name there is an additional encoding algorithm necessary. However, for internationalized domain names (IDNs), which also consist of Unicode characters, there is already an algorithm available for converting them into regular domain names. This conversion algorithm could be extended to encode a general Handle as a regular domain name. Since, Handle strings are usually composed of known identifiers, such as UUIDs, which are representable as regular domain names without additional encoding efforts, in this work we will only focus on such Handles identifiers. For general Handles there is future research necessary in order to deduce an appropriate conversion algorithm.

(2) **Namespace Hierarchy:**
Domain names are composed as hierarchically dot separated labels. From an individual label point of view the label next to the right denotes its parent label. Handles in turn are composed of a prefix and a suffix. As for domain names, the prefix consists of hierarchically dot separated labels. However, in contrast to domain names, in a Handle prefix the rightmost label denotes the lowest level of the hierarchy. The prefix is followed by the slash ('/') character, which separates the suffix from the prefix. The suffix in turn is basically a locally unique identifier. In summary, this means that a Handle identifier incorporates two different separators: dots for the prefix and a slash to distinguish the suffix. In order to represent the a Handle as a domain name, it is first of all necessary to replace the slash by a character which is allowed for domain names. Hence, a slash is replaced by a dot. In addition, the resulting Handle identifier has to reversed to ensure the right traversing order at the resolution process by a DNS resolver.

(3) **Handle DNS Zone:**
For the resolution of both, domain names and Handles, it is necessary to traverse to the responsible node. For domain names, the traversal path is given by its composing labels, whereas for Handles it is given by the prefix. The Handle prefix is controlled by the Global Handle Registry, which holds the addresses of the responsible Handle servers for each individual prefix. Hence, the resolution procedure of Handles over DNS traffic has to involve the Global Handle Registry in order to find the addresses for the responsible Handle servers. For the representation of Handles as domain names, this means that they have to include a common DNS zone name, which is composed of the servers forming the Global Handle Registry: The Handle Zone. However, currently there is only the "hdl.handle.net." zone, which is composed of the Global Proxy Resolvers. For the realization of our approach it is instead necessary to register an additional zone for the Global Handle Registry, e.g. "handle.pid.".

To summarize this discussion, Figure 6 provides an example for the transformation of an individual Handle into the corresponding domain name. The corresponding *name* field of the Resource Record is denoted as *dnsHandle* (see Figure 5).

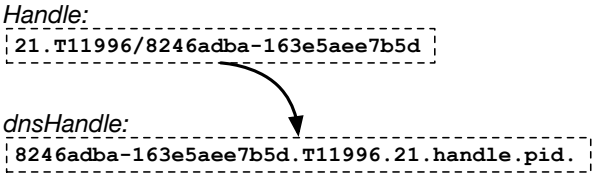


**Fig. 6:** Domain Name Representation of a Handle

### E. Resolution Procedure

The appropriate resolution procedure is depicted in Figure 7. To resolve a Handle over DNS, the Handle has to be transformed into its domain name representation, which is done at step 0. For a Java-based application we have implemented a transformation module (*HandleDNSResolver*) on top of an already existing DNS module for Java (dnsjava) [20]. The method `getTypesByName(handle)` returns all type-data pairs as a JSON string. In addition, the method `getTypeValueByName(handle, type)` returns only the type-value pairs for a specified type.

The steps 1 to 5 are required to be traversed until the actual Handle DNS zone is reached. In these steps, the DNS resolver communicates with ordinary DNS servers in order to reach the Handle DNS zone. At step 6, the DNS resolver communicates with the DNS interface of the Global Handle Registry to find the authoritative Handle server for a specific prefix. The Global Handle Registry in turn, responses with a referral to the responsible Handle server. At step 8, the DNS resolver queries the responsible Handle server through DNS traffic about a specific domain name. At the responsible Handle server the queried domain name is transformed into a Handle identifier for which the Handle Record is retrieved from the database. After applying the mapping procedure (Figure 5) on the retrieved Handle Record, the Handle server responses with multiple TXT Resource Records (step 9). Finally, the response is forwarded to the requesting application.

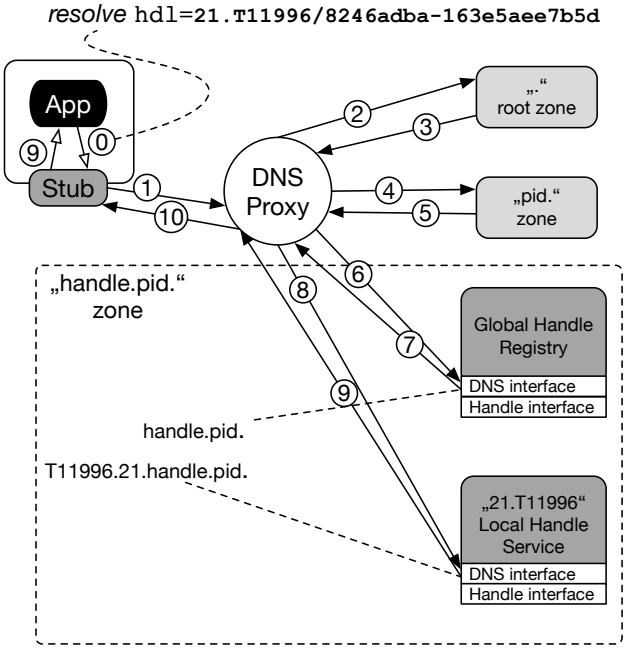


**Fig. 7:** Resolution Procedure for Handles for DNS

## V. Evaluation

In this section, we will evaluate our approach of resolving Handles over DNS traffic. Important parts of our experimental setup, which we will describe in the next subsection, are based on the GWDG infrastructure. GWDG is the service provider for Max-Planck Society of Germany and the University of Göttingen. Moreover, the persistent identifier systems offered by GWDG are based on the Handle System. In addition, GWDG is also member of the DONA Foundation [21], which is controlling the global Handle System infrastructure.

### A. Experimental Setup

Our experimental setup consists of the following components:

- **Imitation of Global Handle Registry:**
  For the evaluation we have setup an imitation of the GHR equipped with a DNS interface. This imitated GHR consists of two servers, whereas the primary is hosted at GWDG and the mirror on an Amazon EC2 instance. In addition, these two servers are the nameservers for our experimental Handle DNS zone ”hx.gwdg.de.”, which is a sub zone under the GWDG zone ”gwdg.de.”.
- **Local Handle Service:**
  Also the LHS consists of a primary and a mirror Handle server, whereas both are hosted at GWDG and equipped with our DNS extension. This LHS is responsible for Handles under the prefix ”21.T11996”. In addition, the Handle servers of this LHS are at the same time the nameservers for the ”T11996.21.hx.gwdg.de.” zone.
- **Load Generator Application:**
  The load generator is a Java-based application hosted on an Amazon EC2 instance located in Frankfurt. This application is equipped with three different modules: The first module is used to resolve Handles directly by means of the Handle libraries, without involving a specific proxy system. The second module is used to resolve Handles through the Global Proxy Resolver (hdl.handle.net), which is the current standard way for resolving Handles. The third module uses our *HandleDNSResolver* module to resolve Handles over DNS traffic.
- **Handle Proxy:**
  Although, the resolution procedure through the Global Proxy Resolver will also involve proxy servers outside of Europe (due to DNS round-robin), for our evaluation we will only consider the two proxy servers in located in Europe. One of the European proxy servers is hosted at an Amazon EC2 instance located in Ireland, whereas the other one is hosted at GWDG. Currently, within the Handle System infrastructure efforts are being made to replace the DNS round-robin based proxy server selection by anycast server selection. The result of that endeavor will be that in the near future requests made in Europe will be answered by the European proxy servers. Hence, in our evaluation we will already cover the Handle System infrastructure of the near future by only considering the European proxy servers.
- **DNS Proxy:**
  To resolve Handles over DNS traffic, we made use of two different DNS resolvers. The first one is an

internal Amazon DNS resolver preconfigured in the EC2 instances. This can be considered to reflect the situation of institutions which make use of their internal DNS resolvers.

In contrast to that, for the second one, we replaced the IP-address of the preconfigured DNS resolver by the address of the public Google DNS resolver, which is reachable via the IP-address 8.8.8.8.

In summary, we have utilized five different resolvers:

- Built-in resolver based on Handle libraries.
- Handle proxy server located in Ireland.
- Handle proxy server located in Germany at GWDG.
- Amazon DNS resolver.
- Google DNS resolver.

### *B. Measurements*

Figure 8 finally illustrates the measurements for each of the five resolution approaches in the same order as listed above. The measuring process consists of two runs of 25,000 subsequent resolution requests for each approach. The mean resolution times of the first runs are depicted by the white left-sided bars. The right-sided black bars in turn, depict the mean resolution times of the second runs, which is intended to reveal the effect of caching. For all approaches, the bars show the resolution times from the perspective of the load generator application. Moreover, for the resolution through the Handle proxy servers, we were able to retrieve the response times from the logging files. Hence, the contribution of the Handle proxy server and the LHS to the overall resolution time is marked by hatches on the respective bars.

Ultimately, the measurements allow the following interpretations:

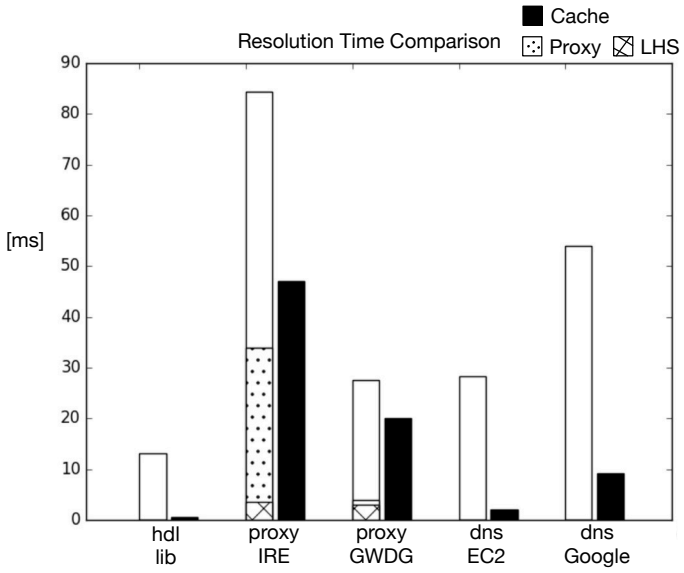


**Fig. 8:** Comparison of resolution times for different resolution approaches

(1) In both runs, the resolution with the Handle libraries is the fastest method. Since this approach directly communicates with the LHS, there is no overhead caused by the involvement of any proxy system. The disadvantage of this approach is the requirement for the implementation of the Handle libraries into the application, which could cause considerable amount of redesign of the application. The intention behind this approach in this evaluation is to provide a measure for the fastest resolution technique.

(2) The bars corresponding to the resolution through the Handle proxy servers show clearly the high contribution of the latency between the requesting application and the proxy server. This is especially visible by the second runs, whereby the Handle Record is cached at the proxy servers. Since in the second run, there is no communication between the proxy server and the LHS, the resolution time consists of the latency between the application and the proxy server only.

For the Handle proxy server located in Ireland the impact of the latency between proxy server and LHS to the overall resolution time is significant as well. However, by means of caching at the proxy server the impact of the latency between proxy and LHS can be minimized for frequently resolved Handles, which can be seen from the second run.

(3) The measurements for the Handle proxy hosted at GWDG and the preconfigured internal Amazon DNS resolver are in terms of network latency are quite comparable. The resolution via the GWDG Handle proxy consists to a great extend of the latency between the Amazon EC2 instance and the proxy server. Since the LHS is hosted at GWDG as well, the latency between this LHS and the GWDG Handle proxy is almost negligible. The database lookup at the LHS has still a small impact onto the overall resolution time, which can be seen from the LHS hatch on the bar. In contrast to that, the resolution time with the Amazon DNS resolver primarily consists of the latency between the LHS and the DNS resolver. As can be seen from the second run, the latency between the application and the DNS resolver is vanishingly small.

Moreover, the second run also reveal the real benefit of the approach of resolving Handles through DNS traffic: Since the DNS resolver is in close proximity of the application, the resolution time of cached Handles is even shorter than the resolution time achieved by a direct communication with the LHS (first run of resolution via Handle libraries). This is quite beneficial for institutions which are heavily working with Handles.

Another fundamental aspect which can be deduced from these two measurements is that Handle resolution via DNS does not cause a significant overhead due to the transformation effort needed to map Handle Values into the DNS resource records. Therefore, the resolution times for both resolution techniques are almost identical for similar network latencies between the involved components.

(4) Although the resolution time with the public Google DNS resolver in the first run is significantly higher than with the GWDG Handle proxy, in the second run, it is the opposite case. Furthermore, the measurements corresponding to the public Google DNS resolver enable to deduce a general strategy to reduce the resolution time for Handles: Each application uses the DNS resolver in its proximity to minimize the latency to the resolver. By means of caching, the communication overhead between proxy resolver and LHS could be eliminated, which ultimately could result in a significant resolution time reduction for frequently resolved Handles.
Although, caching is also applied at Handle proxy servers, the fundamental difference is based on the fact that for an individual application there is always a DNS resolver in its close proximity, which is not the case for Handle proxy servers.

## VI. Conclusion

Persistent Identifiers are becoming more and more important which is correlating with the explosive growth of research datasets. To ensure a sustainable access to research datasets, they are increasingly registered at persistent identifier systems to be assigned a globally unique and location-independent identifier. However, as the count of persistent identifiers increases, the performance of the corresponding resolution systems is becoming increasingly critical for research data management. In principle, persistent identifiers are quite comparable to domain names, both are registered once and resolved multiple times. For the resolution of domain names there a myriad of globally distributed DNS resolvers. In contrast to that, for persistent identifiers the count of resolvers are only very few. For the Handle System, which can be considered as the most important and established persistent identifier system, there are currently only five geographically distributed resolvers available. Hence, the resolution time for persistent identifiers generally consists to a great extend of the latency between requesting application and the particular resolver. The fundamental idea in this work is therefore to use DNS resolvers for resolving persistent identifiers.

For a realization of this idea, we have first extended the Handle servers with a DNS interface and secondly conceived an algorithm to map the Handle data model into DNS Resource Records. The result of this endeavor is the resolvability of Handle persistent identifiers over ordinary, unmodified DNS resolvers.

Furthermore, by means of an evaluation, we have proofed that the resolution time for Handles could be significantly reduced with DNS resolvers. This is especially significant for Handles which are cached at DNS resolvers. Although, frequently resolved Handles are cached at the Handle resolvers as well, the overall resolution time suffers from the small number of globally distributed Handle resolvers, which is not the case with the myriad of DNS resolvers. In addition, the evaluation has also proofed that there is no performance loss due to the mapping from the Handle data model into DNS Resource Records.

Further research is needed, to conceive a standardized DNS Resource Record, which is specifically tailored to hold meta information about digital datasets. This would ultimately enable DNS to move from a simple resolution system, which resolves human-friendly labels into IP-addresses towards a real data dissemination system.

## References


[1] H. V. de Sompel, R. Sanderson, H. Shankar, and M. Klein, "Persistent identifiers for scholarly assets and the web: The need for an unambiguous mapping," *IJDC*, vol. 9, no. 1, jul 2014. doi: 10.2218/ijdc.v9i1.320

[2] T. Kuhn and M. Dumontier, "Making digital artifacts on the web verifiable and reliable," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2390–2400, Sept 2015. doi: 10.1109/TKDE.2015.2419657

[3] E. Bellini, C. Luddi, C. Cirinnà, M. Lunghi, A. Felicetti, B. Bazzanella, and P. Bouquet, "Interoperability knowledge base for persistent identifiers interoperability framework," in *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*. IEEE, 2012. doi: 10.1109/SITIS.2012.130 pp. 868–875.

[4] T. Weigel, S. Kindermann, and M. Lautenschlager, "Actionable persistent identifier collections," *Data Science Journal*, vol. 12, no. 0, pp. 191–206, 2014. doi: 10.2481/dsj.12-058

[5] A. Karakannas and Z. Zhao, "Information centric networking for delivering big data with persistent identifiers," 2014.

[6] A. E. Evrard, C. Erdmann, J. Holmquist, J. Damon, and D. Dietrich, "Persistent, global identity for scientists via orcid," *arXiv preprint arXiv:1502.06274*, 2015.

[7] C. H. Liu, B. Yang, and T. Liu, "Efficient naming, addressing and profile services in internet-of-things sensory environments," *Ad Hoc Networks*, vol. 18, pp. 85 – 101, 2014. doi: http://doi.org/10.1016/j.adhoc.2013.02.008

[8] F. Berber, P. Wieder, and R. Yahyapour, "A high-performance persistent identification concept," *2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, vol. 00, pp. 1–10, 2016. doi: 10.1109/NAS.2016.7549387

[9] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "Dns performance and the effectiveness of caching," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 589–603, Oct. 2002. doi: 10.1109/TNET.2002.803905

[10] E. Cohen and H. Kaplan, "Proactive caching of dns records: Addressing a performance bottleneck," *Comput. Netw.*, vol. 41, no. 6, pp. 707–726, Apr. 2003. doi: 10.1016/S1389-1286(02)00424-3. [Online]. Available: http://dx.doi.org/10.1016/S1389-1286(02)00424-3

[11] Y. Yu, D. Wessels, M. Larson, and L. Zhang, "Authority server selection in dns caching resolvers," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 2, pp. 80–86, Mar. 2012. doi: 10.1145/2185376.2185387. [Online]. Available: http://doi.acm.org/10.1145/2185376.2185387

[12] S. Sarat, V. Pappas, and A. Terzis, "On the use of anycast in dns," in *Proceedings of 15th International Conference on Computer Communications and Networks*, Oct 2006. doi: 10.1109/ICCCN.2006.286248. ISSN 1095-2055 pp. 71–78.

[13] J. Pan, Y. T. Hou, and B. Li, "An overview of dns-based server selections in content distribution networks," *Comput. Netw.*, vol. 43, no. 6, pp. 695–711, Dec. 2003. doi: 10.1016/S1389-1286(03)00293-7

[14] A. J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind akamai: Inferring network conditions based on cdn redirections," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1752–1765, Dec 2009. doi: 10.1109/TNET.2009.2022157

[15] "Doi handbook data model," http://www.doi.org, accessed: 2017-03-23.

[16] "Dns resource record types," http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml, accessed: 2017-04-04.

[17] "Handle software package," http://www.handle.net/download_hnr.html, accessed: 2017-03-31.

[18] "Dynamic delegation discovery system (ddds)," https://tools.ietf.org/html/rfc3401, accessed: 2017-11-23.

[19] "Dns txt resource record," https://tools.ietf.org/html/rfc1464, accessed: 2017-04-06.

[20] "Handlednsresolver source code," http://hdl.handle.net/11022/0000-0003-88B2-A, accessed: 2017-04-06.

[21] "Dona foundation," http://www.dona.net, accessed: 2016-11-23.