Cluster Computing (2022) 25:1645–1664
https://doi.org/10.1007/s10586-021-03455-5

DNS-embedded service endpoint registry for distributed e-Infrastructures

Andrii Salnikov[1,2] · Balázs Kónya[2]

Received: 26 November 2020 / Revised: 13 October 2021 / Accepted: 19 October 2021 / Published online: 11 November 2021

Abstract
Distributed e-Infrastructure is a key component of modern BIG Science. Service discovery in e-Science environments, such as Worldwide LHC Computing Grid (WLCG), is a crucial functionality that relies on service registry. In this paper we re-formulate the requirements for the service endpoint registry based on our more than 10 years experience with many systems designed or used within the WLCG e-Infrastructure. To satisfy those requirements the paper proposes a novel idea to use the existing well-established Domain Name System (DNS) infrastructure together with a suitable data model as a service endpoint registry. The presented ARC Hierarchical Endpoints Registry (ARCHERY) system consists of a minimalistic data model representing services and their endpoints within e-Infrastructures, a rendering of the data model embedded into DNS-records, a lightweight software layer for DNS-record management and client-side data discovery. Our approach for the ARCHERY registry required minimal software development and inherits all the benefits of one of the most reliable distributed information discovery source of the internet, the DNS infrastructure. In particular, deployment, management and operation of ARCHERY is fully relying on DNS. Results of ARCHERY deployment use-cases are provided together with performance analysis.

Keywords e-Science · Distributed computing · Domain Name System · Information system · Service endpoint registry · ARCHERY · Advanced resource connector · WLCG

1 Introduction

Modern science heavily relies on distributed e-Infrastructures. The Worldwide LHC Computing Grid, WLCG [5] is a well-known example of a large scale geographically distributed e-Infrastructure that combines together thousands of computing and storage services as a single interoperable e-Science environment.

Nowadays, production scientific e-Infrastructures as WLCG, EGI [23] and Open Science Grid, OSC [30] rely on federated organizational and operational model where resources are grouped either geographically (country, region, sites, so on) or administratively (collaboration, tier, research group, so on). Within those units various service are hosted. Furthermore, the e-Infrastructure resources such as the computing and data storage services can be accessible via their endpoints. Figure 1 illustrates the general grouping concepts within a typical distributed e-Infrastructure.

In such environment, the ability to be able to discover service endpoints is a crucial functionality that enables infrastructure consumers to utilize the available resources as part of their distributed computing workflows. In particular, availability of service endpoint information is essential to implement load balancing techniques in the distributed computing [25]. In the past, various service registries, catalogues, endpoint indices, directory services and other similar solutions have been proposed, developed and deployed to serve as an information source for the service endpoint discovery. Both the centralized and the widely distributed approaches have been tried. These

✉ Andrii Salnikov
andrii.salnikov@maxiv.lu.se

Balázs Kónya
balazs.konya@hep.lu.se

1 Computer Engineering Department, Taras Shevchenko National University of Kyiv, 64/13 Volodymyrska str., Kiev 01601, Ukraine

2 Lund University, BOX 118, 221 00 Lund, Sweden

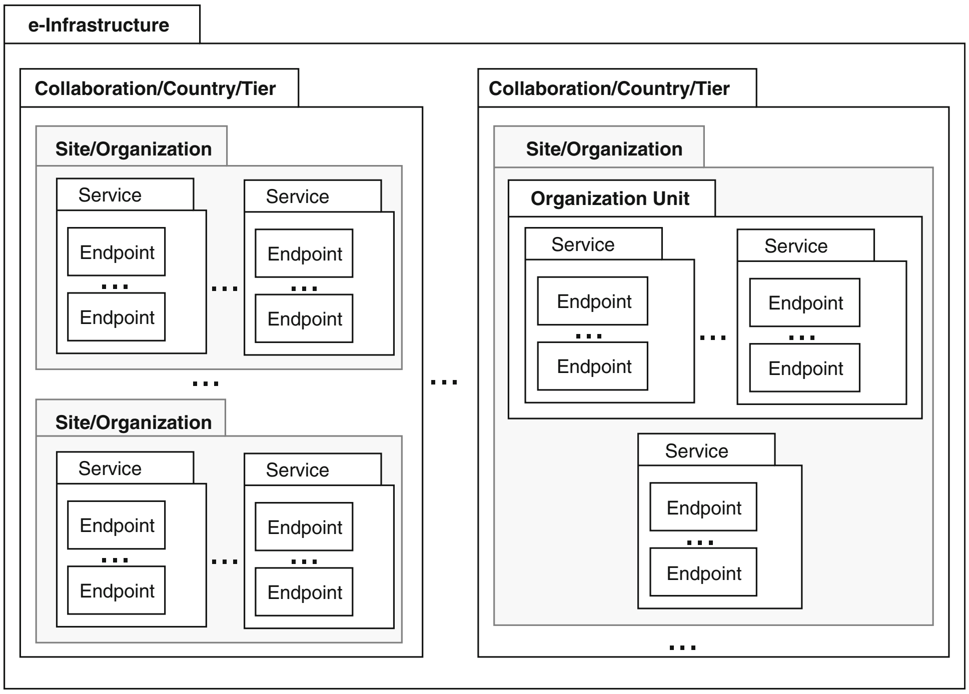

Fig. 1 Typical organizational structure of a distributed e-Infrastructure illustrating the grouping relations of various Collaborations, Sites, Services and Service Endpoint

catalogues have been storing information about either or both the internal administrative grouping of the infrastructure and the services as well.

EGI stores the grouping of services to sites and their grouping to the National Grid Infrastructures (NGI) in the Grid Operations Center Database, GOCDB [26]. WLCG groups resources based in Tier level, and provided services are stored in the WLCG REsource, Balance and USage (REBUS) topology database [6]. OSG in turn has its Own Information Management System (OIM) that manages Unites States resources [30]. On the particular scientific collaboration level information about available resources may be aggregated and stored in a centralized database, e.g. for the ATLAS CERN experiment a dedicated Grid Information System, AGIS [2] had been developed.

The very first attempt to provide a service catalogue was the Globus Meta computing Directory Service, MDS [10], an LDAP-based solution. The limitations and instabilities of the MDS implementation resulted in the development and deployment of various LDAP-based solutions such as the WLCG BDII [7] or the EGIIS service of NorduGrid ARC [14].

As of this writing the BDII service is widely deployed and used as the production information backbone of WLCG despite many of its well known limitations. BDII is a fully centralized system relying on LDAP. LDAP is being efficient on query level, but from the other hand is not optimized for frequent writes. This results in serous hardware load. There is also noticeable huge load on network caused by regular fetching of centralized entire database.

There were previous attempts to overcome the issues of the centralized LDAP-based BDII and introduce general purpose newly-developed service registries for e-Infrastructures. The distributed systems—EMI Registry service [15] and P2P indexing systems [1, 31] represent such incomplete projects that got stuck at the roll-out deployment phase.

All existing or attempted infrastructure topology databases and service registries used in WLCG so far had been developed as a standalone complex services with an overloaded data model with unclear separation of static and dynamic information. Furthermore, the deployment, operational and management aspects of such services are at least as important as the quality of the technical implementation.

There is a well-established information lookup system widely used all over the world: The Domain Name System (DNS), dated back to the ARPANET times, is an integral part of today's Internet and has been used for domain-name specific information discovery for ages. DNS primarily had been used for resolving network layer addresses of the hosts but it was designed as a general multi-purpose distributed hierarchical database that holds information about anything related to the target domain [28].

The ARC Hierarchical Endpoints Registry represent a novel DNS-based service endpoint registry for e-Science infrastructures. ARCHERY embeds the service endpoint information directly into the DNS database according to a proposed minimalistic data model. The reference implementation of the suggested method based on the ARC middleware [14] delivers the lightweight middle layer between the e-Infrastructure services and the DNS infrastructure. In contrast to the existing solutions the proposed ARCHERY registry does not require development and operation of additional services since it reuses well-known DNS infrastructure services with out-of-the-box integrity, fault-tolerance and network-level caching. The data model definition of ARCHERY takes into account the operational experience of existing e-Infrastructures and designed to be minimalistic as possible, significantly reducing the data rates.

The rest of the paper is organized as follows: Sect. 2 provides a brief overview of the benefits of relying on the DNS infrastructure and the possibility of embedding free-form information into DNS records. The paper continues with Sect. 3 introducing set of requirements for service endpoint registries. The main part of the work is delivered in Sects. 4, 5, and 6 describing the ARCHERY data model, security aspects, dynamic information and software layer implementation. The article concludes with Sects. 7 and 8 introducing typical ARCHERY deployment scenarios and deployment results.

2 DNS as an ultimate source of domain-specific information

DNS is a well-established integral part of today's Internet that used for information discovery. Dated back to the ARPANET times it was originally developed to solve the problem of domain to network layer address mapping but then continuously evolve to provide more types of information and features.

There are more than a hundred different RFC documents that describe various aspects of DNS data structures, operations, security, etc. [20]. The design goals of the DNS architecture itself cover many points that define how information is stored and made accessible. For the approaches proposed in this paper the following points are most relevant [28]:

- Database must be maintained in a distributed manner, with local caching to improve performance to achieve both scalability and manageability.
- The source of information controls the trade-off between update speed and cache validity, defining the accuracy.
- The worldwide distributed DNS infrastructure is not restricted to single usage pattern and generally useful for many different applications.

In the DNS all data associated with a domain name (that identify a **node**) is tagged with a **type**. Information in each node is stored inside the resource records (**RR**). Resource records are grouped in the Resource Records Set (**RRSet**) that is referenced by the name (**owner**) [17]. It is important to mention that the order of RRs itself is not preserved by name servers or resolvers.

2.1 Benefits of the DNS infrastructure

Current DNS infrastructure exhibits many capabilities that are particularly well-suited for the use case of an e-Infrastructure service endpoint registry:

- *Caching* DNS infrastructure implies caching on many levels in the network, distinguishing between caching name services and origin of information that defines the `TTL` value. Even dedicated software products (for example Unbound [35]) exists to target efficient caching implementation. On the client level, DNS responses are also cached by the operating system.
- *Integrity* DNS keeps track of the information origins, maintaining the so-called "Authoritative information" that is organized into units called **zones**. Each zone has *serial number* that define the revision of information and allows to control which instance has the most accurate information. When client gets response even from cached data, the response always includes "authority" section that can be used to contact information source directly if needed.
- *Resiliency* it is generally required that each zone is served by at least two different name servers. All name services are defined in the zone itself and also in the parent zone following the hierarchy. Name services for the same zone are configured to share the same data using the zone transfer requests.
- *Zone delegation* DNS was designed to be distributed. Each particular zone can be hosted by independent name servers located anywhere in the network. All zones form the common tree starting from the *root zone* that hosted by well-known *root servers* [19].
- *Aliases* the `CNAME` type of records can be used to redirect particular DNS request to another DNS name. RFC 6672 defines the `DNAME` type of records [32] that can be used to transparently map all names of particular DNS suffix to another part of the tree.
- *Dynamic updates* current DNS infrastructure also defines the protocol for "Dynamic Updates in the Domain Name System" that was first declared in the RFC 2136 [39] with further security-related modifications. It is possible to add or delete RRs or RRSets from a specified zone with an atomic UPDATE operation. Possibility to modify DNS dynamically opens it for much more use-cases, starting with DHCP integration.
- *Security* the DNS Security Extensions (DNSSEC) adds data origin authentication and data integrity to the DNS

[3]. It allows to build and verify the *Authentication Chain* using the cryptography algorithms to stand against possible information spoofing. To authenticate dynamic updates Secret Key Transaction Authentication for DNS (TSIG) [38] can be used on transactions level.

2.2 Embedding free-form information into DNS records

Worldwide DNS infrastructure was designed to be scalable and open for many types of information that can be identified by domain name. As declared in the RFC 1034 [28]:

> The costs of implementing such a facility dictate that it be generally useful, and not restricted to a single application. We should be able to use names to retrieve host addresses, mailbox data, and other as yet undetermined information.

Historically the first use-case of free-form information embedding into DNS was Hesiod [12], the Athena name server, aimed to provide naming for services and data objects in a distributed network environment. It was used to provide `passwd`, `group` and other databases via the network by means of DNS protocol. Even in 1980s it took milliseconds to get a responses from DNS-embedded database that held roughly 10k records of each type (3 MB of data) [12]. Hesiod introduced the `TXT` records type that is now one of the standard DNS RRs [29]. These records allows to store any arbitrary ASCII string in the DNS database.

There is also an experimental RFC 1464 dated back to 1993 that suggests to define the format of TXT RDATA fields as the "attribute name followed by the value of the attribute". However, despite this experimental RFC the `TXT` records are in practice used in arbitrary defined format and TXT RRs usage increased over the last years [37].

Some of the most common production TXT RR use-cases are:

- Defining Kerberos realm name in the DNS with TXT record using the `_kerberos` RR owner [27]. Value of TXT RR is an exact Kerberos realm name.
- Email security heavily relies on DNS. Sender Policy Framework, SPF [22], Domain Key Identified Mail (DKIM) or Domain-based Message Authentication, Reporting, and Conformance, DMARC [9] define the policies of e-mail acceptance and public keys for signature verification via `TXT` records.
- Domain ownership verification often uses `TXT` RR. For example Google GSuite verification process provides a token that should be published in the `TXT` RR for the domain in question.

2.3 Service discovery using DNS

There are several approaches that have been evolved over the time for service discovery using DNS infrastructure.

The SRV RRs were introduced to specify the location of the server(s) for a specific protocol and domain [16]. SRV records can only store a single service of any given host and port combination. Also, SRV records do not support the meta data for a specific service instance.

The other method based on RFC 6763 defines the DNS-Based Service Discovery (DNS-SD) mechanism that allows clients to discover a list of named instances of that desired service, using standard DNS queries [8] and has been developed to overcome SRV RRs limitation. DNS-SD builds upon the combined usage of `SRV`, `TXT` and `PTR` records. The SRV RRs give the target host and port. The DNS TXT RRs give additional information about particular instance, in a structured form using key/value pairs. The PTR RRs used to map service types to named service instances.

The SRV RRs and DNS-SD covered the use-case of getting particular known services within a certain domain (e.g. printers in the local area network). Service discovery for distributed e-Infrastructures necessitates hierarchical services topology and federation of services that are outside of the scope of the DNS-SD standard. The capability of delegated administration of a distributed topology is not covered by DNS-SD either.

3 Requirements for service endpoint registries

Based on more than a decade of operational experience and following the evolution of available and proposed service registry solutions, below we present a revisited set of requirements for e-Infrastructure service registries. Our use-case in focus is the distributed e-Infrastructure serving the WLCG community.

3.1 Minimalistic data model

A service registry should store only minimalistic information necessary for endpoint discovery. Furthermore, the information maintained inside the registries should be structured according to a mapping capturing the essential relation of the key e-Infrastructure concepts such as organization, site, services, endpoints, etc. (Fig. 1). It is expected that the data objects of the model represent mostly static or semi-static information. A service registry is not intended to store fast changing or unrelated and unnecessarily verbose data. The distributed nature of e-Infrastructures implies that "approaches that attempt to collect a consistent copy of the entire database will become more and more expensive and difficult, and hence should be avoided" [28]. The complex data model and related database update issues are some of the known limitations of the Top-BDII registry [7, 15].

3.2 Registry fault tolerance

Eliminating the single point of failure in a distributed infrastructure is a critical, therefore a registry service should offer fault tolerance and redundancy while providing a single transparent access point.

3.3 Flexible services grouping via federations

To reflect the topology of the e-Infrastructure resources, the registry should be flexible in terms of grouping of services and administrative units. For that, support for nested groups, shared objects and out-of-tree references should be available. By using these flexible service grouping options it will be possible to describe complex hierarchy, overlapping organization units in various federations, etc. (Fig. 1).

3.4 Query and network load efficiency

Worldwide-scale e-Infrastructures will have to deal with huge number of registry information consumers. The technical implementation should take into account query efficiency, server-side load and network usage during the registry operations. There is always a trade-off between up-to-date information availability and the load on the infrastructure.

3.5 Information integrity and security

Increasing number of attack vectors in current network makes information content integrity and spoofing protection a crucial part of any service. This applies to both authenticity and integrity of information itself and securing the information updates inside the registry.

3.6 Simple deployment and operation

An important and often overlooked aspect of service take-up is the cost of deployment, maintenance and operation. The deployment aspect is especially critical for distributed systems where service roll-out requires cooperation of many infrastructure actors. Therefore building the registry service upon trusted and well-known solutions is preferred. In the past, we witnessed promising registry service developments that got stuck in the deployment phase and never reached the production status [15]. Distributed operations, where different people are responsible for different parts of the e-Infrastructure should also be supported. Furthermore, a change in the topology of the e-Infrastructure should not require a complete reconfiguration of the registry.

The above collected e-Infrastructure registry requirements form the basis of a novel registry service—"ARCHERY" and the rest of the paper describes in detail how the requirements are fullfilled.

4 Embedding a service endpoint registry into the DNS infrastructure

We propose to fulfill the re-visited e-Infrastructure registry requirements (see Sect. 3) using the already established DNS infrastructure and embedding the necessary service endpoint information. In order to capture the relevant service information and e-Infrastructure topology an ARCHERY minimalistic data model has been designed (Sect. 4.1).

DNS as a well-proven and robust distributed information infrastructure offers numerous benefits for building a registry for e-Infrastructures (see Sect. 2.1). Registry fault tolerance will be guaranteed by DNS services resiliency, still providing a single entry point. Built-in zone-delegation and DNS aliases allows to carry out a distributed operational model of the registry. Information integrity and caching in DNS offers query efficiency and consistency of the registry data. The available built-in DNS security features can be transparently applied to the registry data.

Simple deployment and operation is achieved by developing a light-weight software layer (Sect. 6) on top of the NorduGrid ARC middleware [14].

4.1 ARCHERY data model

ARCHERY implies a minimalistic data model that is targeting mostly static service information to address the distributed e-Infrastructure resource grouping and service discovery needs (see Sect. 5 regarding non-static information). To represent the distributed e-Infrastructure concept (Fig. 1) the ARCHERY data model defines three type of objects: the ***Endpoint Object***, the ***Service Object*** and the ***Group Object*** (Fig. 2).

[*Endpoint object*] Endpoint object is used to capture information about a network location that can be used to access specific service functionality, including accessing information within ARCHERY registry service itself. For the later, special endpoint types were defined (see below). The Endpoint object is described with the following set of attributes:

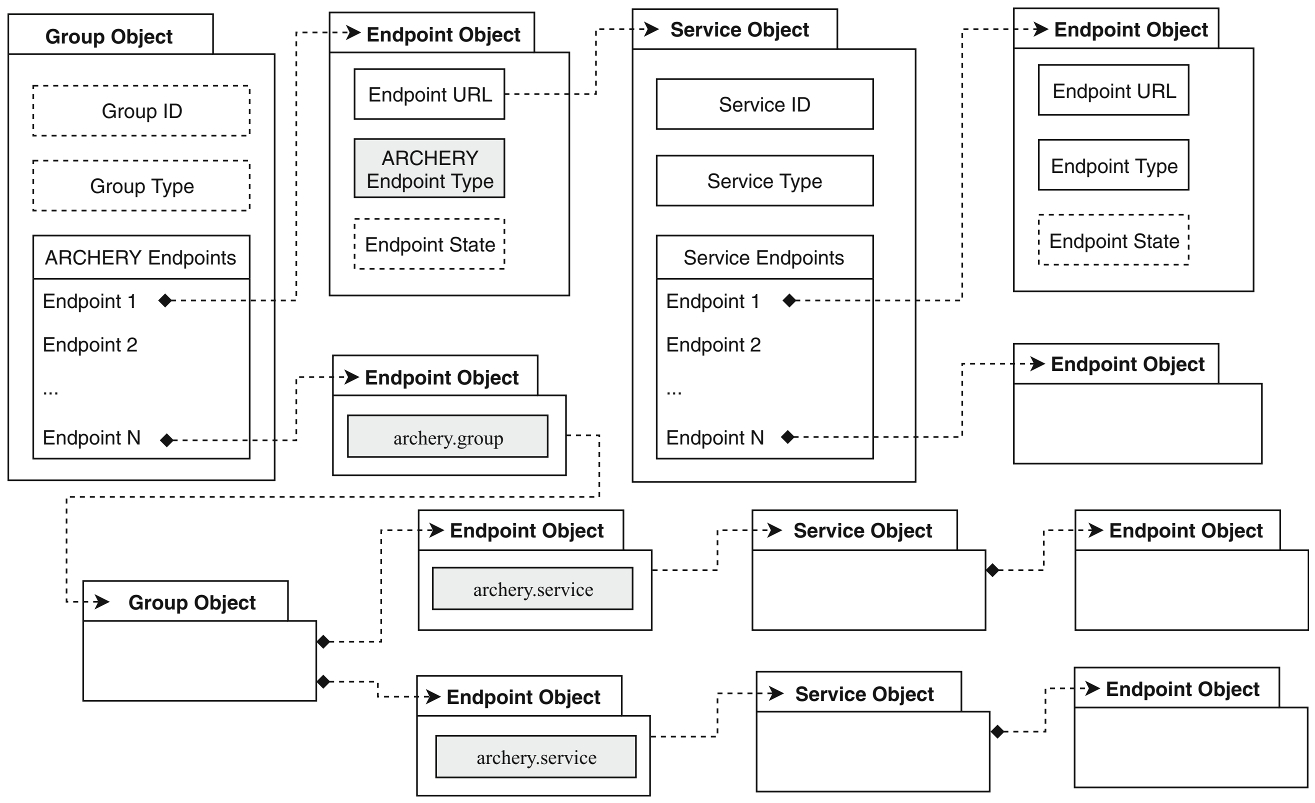

Fig. 2 ARCHERY objects, their attributes and relations in the proposed data model. Please observe the way how ARCHERY Endpoint objects are used to describe groupings in the topology

- *Endpoint URL* defines the network location by which the service functionality is accessible.
- *Endpoint Type* contains the value from the Endpoint type enumeration defined by the infrastructure operators. In addition to ordinary service endpoint types in the model we introduced two special ARCHERY endpoint types `archery.group` and `archery.service` for accessing the Group and Service objects within the ARCHERY.
- *Endpoint State* (***optional***) Boolean value that indicates the endpoint availability. In case of an ARCHERY service endpoint, if the endpoint state is false (unavailable) it is still used to represent e-Infrastructure topology, but should not be used during service endpoints discovery. Missing attribute is interpreted as true i.e. the endpoint is available.

Service object represents an e-Infrastructure service—Computing Cluster, Storage service or a User database, etc. The Service object is described with the following set of attributes:

- *Service ID* holds the service identifier as an arbitrary string. For example the ID can be derived from the hostname;
- *Service Type* contains the value from the Service type enumeration defined by the e-Infrastructure operators;
- *Service Endpoints* an array of Endpoint objects associated with the Service.

Group object is used to organize other objects such as ARCHERY Service or Group. The grouping was introduced to the data model so that infrastructure topologies (i.e. hierarchies or federations) can be represented inside the registry in a flexible way. The Group object is described with the following set of attributes:

- *Group ID* (***optional***) holds the group identifier as an arbitrary string.
- *Group Type* (***optional***) defines the grouping type based on organizational structure; example values could be: Site, Country, Tier, ExperimentA, etc.
- *ARCHERY Endpoints* an array of Endpoint objects of special defined ARCHERY endpoint types (`archery.group` or `archery.service`). These Endpoint objects describe the optional state and URL of the ARCHERY objects inside the registry.

Following the data model (Fig. 2) the service endpoint discovery process can start from the **entry point** Group Object and recursively querying the ARCHERY Endpoints. The proposed object attributes allow taking into account group, service and/or endpoint types as well as availability status during the recursive discovery process to query only the subset of objects.

4.2 Rendering the data model with DNS data structures

What follows we propose a rendering rules for mapping the ARCHERY data model into DNS structures.

4.2.1 ARCHERY objects within DNS

ARCHERY objects are rendered using TXT resource records (Sect. 2.2). These TXT RRs contain the space separated `key=value` pairs where keys correspond to the data model object attributes. Boolean object attributes have values specified as 0 or 1. The TXT RRs-based rendering was chosen as the most streamlined rendering that is capable to reflect the data model objects including their hierarchical relations.

We note that it is technically possible to render records by means of extending the DNS-SD approach (Sect. 2.3), however, as the ARCHERY objects contain more than the hostname and port information, using the TXT RRs would anyway be required, according to DNS-SD specification, in addition to the SRV and PTR RRs. Thus the resulting rendering would be much more complicated, containing more records in the DNS and not providing any benefits to the registry clients.

The **Endpoint Object** is rendered with a single TXT RR where Endpoint Object attributes—*Endpoint URL*, *Endpoint Type* and optional *Endpoint State* are represented by the `u`,`t`,`s` keys in this specific order within the space separated key-value pairs.

The **Service Object** of the ARCHERY data model is rendered by a dedicated RRSet identified by a unique domain name that can be used to access this object inside the registry. The service RRSet is composed of a single service object identity RR and one RR for every Endpoint objects associated with the service. The service object identity RR has the format of `o=service` followed by `t=<Service Type>` and `id=<Service ID>`. Figure 3 shows the rendering format of the Service object as an RRSet containing the Endpoint RRs inside.

```
<DNS name>    TXT    "o=service t=<Service Type> id=<Service ID>"
<DNS name>    TXT    "u=<Endpoint URL> t=<Endpoint Type> [s={0|1}]"
<DNS name>    TXT                        ...
<DNS name>    TXT    "u=<Endpoint URL> t=<Endpoint Type> [s={0|1}]"
```

Fig. 3 The DNS RRSet representing the ARCHERY Service Object including the associated Endpoint Objects as well. The first line of the RRSet is the service object identity RR

The **Group Object** of the ARCHERY data model is rendered by a dedicated RRSet identified by a unique domain name that can be used to access this object inside the registry. The Group RRSet is composed of a single Group object identity RR and one RR for every ARCHERY Endpoint object pointing to other Service or Group objects that are part of this specific Group. The Group object identity RR has the format of `o=group` followed by optional `t=<Group Type>` and optional `id=<Group ID>`. Figure 4 shows the rendering format of the Group object as an RRSet containing the ARCHERY Endpoint RRs.

4.2.2 Embedding ARCHERY object relations as part of DNS

In the registry rendering the entry point Group Object is distinguished by the predefined RRSet DNS name starting with the mandatory `_archery`. Any other objects may have arbitrary RRSet name in any DNS zone. Exact RRSets naming scheme does not affect service discovery process, but should be consistent and transparent from the operational point of view.

The child–parent Endpoint Object to Service Object relation is implicitly defined by means of grouping within the same RRSet (Fig. 3).

The ARCHERY Data model allows grouping of Service and Group Objects into Group objects via the intermediate Endpoint objects. This grouping in the DNS rendering is implemented by using dedicated ARCHERY Endpoint types (Fig. 4) that are contained as RR text inside the Group object's RRSet.

Both Group Object to Group Object and Group Object to Service Object relations rest upon the unique DNS name of the RRSet of the pointed object.

4.2.3 Rendering example

As an example for the rendering rules, we present an ARCHERY registry, embedded into `example.org` DNS zone. The registry describes an e-Infrastructure collaboration "COLLABORATION_EXAMPLE" consisting of a site and several services (Fig. 5).

The top level DNS RRSet in the DNS registry is the `_archery.example.org` which is a Grouping object on the *Collaboration layer*.

```
<DNS name>    TXT    "o=group [t=<Group Type>] [id=<Group ID>]"
<DNS name>    TXT    "u=<DNS URL> t=archery.{group|service} [s={0|1}]"
<DNS name>    TXT                     ...
<DNS name>    TXT    "u=<DNS URL> t=archery.{group|service} [s={0|1}]"
```

Fig. 4 The DNS RRSet representing ARCHERY Group Object including the associated Endpoint Objects pointing to other Group or Service objects inside the registry. The first line of the RRSet is the Group Object identity RR

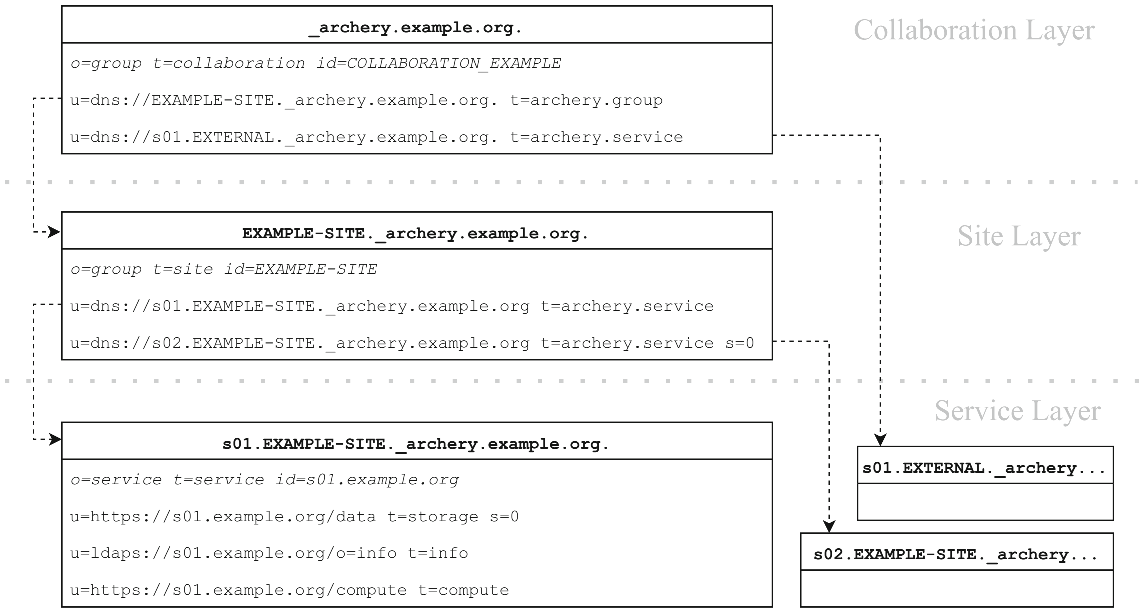
Collaboration Layer
_archery.example.org.
o=group t=collaboration id=COLLABORATION_EXAMPLE
u=dns://EXAMPLE-SITE._archery.example.org. t=archery.group
u=dns://s01.EXTERNAL._archery.example.org. t=archery.service
Site Layer
EXAMPLE-SITE._archery.example.org.
o=group t=site id=EXAMPLE-SITE
u=dns://s01.EXAMPLE-SITE._archery.example.org t=archery.service
u=dns://s02.EXAMPLE-SITE._archery.example.org t=archery.service s=0
Service Layer
s01.EXAMPLE-SITE._archery.example.org.
o=service t=service id=s01.example.org
u=https://s01.example.org/data t=storage s=0
u=ldaps://s01.example.org/o=info t=info
u=https://s01.example.org/compute t=compute
s01.EXTERNAL._archery...
s02.EXAMPLE-SITE._archery...

Fig. 5 ARCHERY DNS RRSets rendering an example e-Infrastructure collaboration topology

This top-level Group groups another Group Object (EXAMPLE-SITE._archery.example.org RRSet) on the *Site Layer* and a Service Object on the *Service Layer* (s01.EXTERNAL._archery.example.org RRSet). The specific service is an example of a Collaboration-level service (e.g. an outsourced external service).

Site Layer Group Object (EXAMPLE-SITE._archery RRSet) represents the EXAMPLE-SITE that provides two services. The services are referenced by the corresponding ARCHERY endpoints.

Furthermore, notice that the second ARCHERY Endpoint (s02.EXAMPLE-SITE._archery RRSet) pointing to the service marked as unavailable with the Endpoint State (s=0) and therefore this service will not be queried during endpoint discovery process.

On the *Services Layer* there are three services with the details of the s01.EXAMPLE-SITE._archery RRSet shown. This particular service offers three service endpoints of different types (e.g. t=compute). Also note that the storage endpoint is unavailable.

Notice that the RRSet naming scheme in this example implies the convenient choice of a new sub-domain for each further e-Infrastructure hierarchy layer.

4.3 Implications of DNS records size limits

There is a limitation of DNS response size which affects the ARCHERY implementation as well. For the UDP transport layer protocol the limit is 4096 bytes [11] (older limit is 512 bytes [24]) while for TCP protocol 65,535 bytes are allowed. Unless the transport protocol is defined explicitly DNS clients use UDP transport for queries by default. However, if the message does not fit into the UDP limit, the DNS server sends truncated flag in the response and the clients use TCP [4] transparently to the application. DNS updates and zone transfer requests always use TCP transport. Therefore, for ARCHERY operations, the 65,535 bytes TCP response size limit is the relevant limitation.

The TCP 65 kB limit constrains the size of a single ARCHERY object, thus the number of ARCHERY endpoints a single object can contain. In the presented e-Infrastructure deployments (see Sect. 8) the typical size of the rendered registry TXT RRs was approximately 100 bytes. The exact value depends on the object attribute values string representation length. This means that one RRSet representing either a Service or a Group Object can include nearly 650 Service or ARCHERY Endpoint Objects in accordance to the TCP response limit. Please note that the allowed 650 endpoints within an ARCHERY object is an order of magnitude larger than the typical real-life deployment numbers where sites or services usually have around a dozen endpoints (see Sect. 8.1). This means that even the more restrictive UDP limit of approximate 40 endpoints can be easily met, thus offering better query latency.

To conclude it can be noted that according to the typical hierarchical e-Infrastructure design (Fig. 1), that introduce grouping in accordance to operations model, the DNS response size limits provide enough headroom to store all registry data. Nevertheless, it is required that the ARCHERY software layer respects this constraint and introduce another grouping layer whenever DNS transport layer limit is reached.

5 Security aspects and dynamic information

5.1 ARCHERY security

Since the underlying DNS infrastructure implements a world-readable distributed system the service endpoint registry information stored in ARCHERY also becomes publicly available. Being worldwide readable the DNS infrastructure offers the **information authenticity** and **integrity verification**, implementing the technology called DNS Security Extensions (DNSSEC) [3].

At the time of this writing the top-level deployment of DNSSEC is 91% and growing [18] and deployment on the further hierarchy levels is progressing. It is advised to use DNSSEC for ARCHERY but it is not mandatory.

Everything on both client and server side of DNS, including DNSSEC verification is already in place, does not require re-implementation and does not change anything in ARCHERY way of DNS usage. For security reasons we recommend to have DNSSEC configured for ARCHERY DNS zones.

Inserting service registry information or modifying existing records within the DNS is carried out by following well-established DNS management procedures this way ARCHERY inherits the operation security of the DNS platform. When it comes to authenticated dynamic updates the Secret Key Transaction Authentication for DNS (TSIG) used on transactions level.

5.2 Dynamic information in ARCHERY

ARCHERY primarily stores static service endpoint information thus the attributes of the minimalistic data model are not expected to change frequently. The few occasional changes are to be handled by deleting the old record and inserting a new one. Such a typical rare change could be the reconfiguration of a service endpoint URL that would imply the need of modifying the corresponding ARCHERY record.

In the distributed e-Infrastructure, service availability is dynamically changing. In order to capture this sort of dynamic behaviour the ARCHERY data model introduced the **optional Endpoint State** attribute. The state values should be updated regularly based on the testing results and set to False if endpoint is not functional. Even though the state of service may change frequently, within the ARCHERY index this information is expected to be updated less frequently. An update interval of 10–15 min is still useful to reflect the outages on e-Infrastructure level, when the endpoint is unavailable for hours and days. Such a dynamically updated semi-static state information is very useful for optimizing client workload and avoiding unnecessary attempts to access the nonoperational services and prevent slow-down in the e-Infrastructure usage.

Updating the state attribute means solving several related problems: discovering the state of the service endpoint, indicating the validity of information in the registry and automating the dynamic information updates. All these are taken care by the ARCHERY software layer (see Sect. 6) that includes set of probes for state discovery, a module to set the TTL value for the RR to control DNS caching and a utility implementing the Dynamic DNS (DDNS) update requests [39] secured with TSIG [38] that ensures remote and secured way of registry content modification.

6 Implementation of the ARCHERY software layer

The proposed method of embedding service endpoint registry into DNS required minimal additional software development. The actual implementation of the lightweight software layer was carried out as part of the NorduGrid ARC middleware [14]. ARCHERY reuses the existing DNS infrastructure services thus eliminating the need to develop, deploy and operate new set of custom dedicated services. On the client-side there are numerous libraries and tools available for interacting with the DNS infrastructure in most of the programming languages and platforms, making ARCHERY client integration simple.

To simplify the process of rendering ARCHERY records (see Sect. 4.2) and injecting those to the DNS we developed the `archery-manage` information management tool (see Sect. 6.1). The `archery-manage` utility is capable modifying data in the DNS zone via dynamic DNS updates over the network eliminating the need to interact with DNS configuration itself. This approach also makes the deployment and access rights delegation simple, fully separating the DNS hosting itself and ARCHERY data management machine.

6.1 The `archery-manage` tool

The `archery-manage` tool had been designed to simplify common operations with ARCHERY, including registry initial bootstrap, data migration from the other service registries and keeping dynamic information up to date.

The idea behind the `archery-manage` is to provide a tool that can discover service endpoints by understanding e-Infrastructure topology. Based on the discovered information the tool generates DNS records for ARCHERY.

The various steps of the archery-manage operational workflow are illustrated on Fig. 6 and explained below.

Step 1. Define e-Infrastructure topology. Data describing topology defines how services are grouped within the e-Infrastructure. It comes either from a configuration file or from other databases that holds such information (including another ARCHERY instance). Support for already established databases (e.g. GOCDB) simplifies the integration and/or migration process.

Step 2. Fetch service data. The topology database provides the pointers to information services that can be used to query service data. During this step, the archery-manage tool discovers available endpoints and fetches service information.

Step 3. Filter endpoints. Set of discovered endpoints later passed to the filtering process. Based on the endpoint data (e.g. endpoint type), or additional testing (e.g. endpoint network availability check) endpoints that does not pass the filters are excluded. Filters are extensible by design.

Step 4a. Incremental DDNS Update. The target automation use-case is to push the discovered data to the DNS database. This is done automatically with Dynamic DNS updates [39] over the network. Comparing the data already available in the DNS with discovered information, archery-manage constructs the incremental update that only applies the difference.

Step 4b. Output data. In addition to automatic updating of the DNS database, we also target the manual operation use-case. For this, following the same processing chain, the tool prints out endpoint or service lists with their types or the ARCHERY DNS records that can be manually added to DNS zone configurations.

6.2 Registry information consumers

On the client level, ARCHERY benefits from the distributed DNS caching on many levels, making the registry scalable.

Since DNS client is an integral part of any operating system, obtaining and processing service endpoint information from ARCHERY comes down to parsing and interpreting the data obtained from the DNS in accordance to the ARCHERY data model rendering (see Sect. 4.2).

As part of the ARC middleware we provide several tools to fetch ARCHERY data. One of them is the archery-manage operation tool that itself can be used to query registry with ARCHERY service specified as the topology source and displaying the discovered endpoint information as a formatted output.

We have also developed a service endpoint retrieval plugin for the ARC middleware [14] that uses ARCHERY as an information service to discover available computing resources for computational job submission. Furthermore, an infrastructure monitoring web application [34] was also updated to be able to visualize realtime Computing Cluster information obtained via service endpoint discovery performed using ARCHERY.

7 ARCHERY deployment scenarios

Below we provide two boundary use cases for ARCHERY deployment—the most minimalist topology serving a small research group and a large scale example corresponding to the EGI e-Infrastructure. Deploying ARCHERY for particular projects in most cases will be something in between, depending on the organization and operation model.

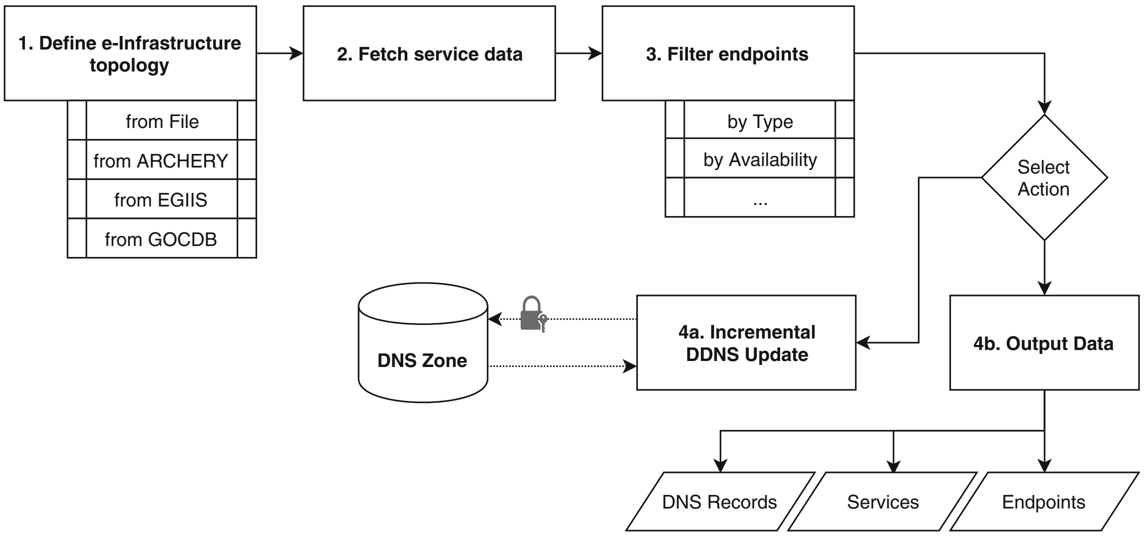

Fig. 6 The archery-manage data processing chain

In this section, we also present general guidelines for operating a DNS services since ARCHERY deployment relies on a DNS instance that contains the service endpoint registry records embedded.

7.1 Operating DNS services for ARCHERY hosting

DNS services configuration and maintenance are beyond the scope of the ARCHERY itself and already done by networking technicians at every organization. This is one of the main operations and deployment benefits of the proposed DNS-embedded service endpoint registry.

Embedding necessary data into the DNS database in the simplest case can be done by means of manually editing the records within the DNS zone configuration file. An automated typos-free way of rendering the records in accordance with the data model is—to use `archery-manage` data output capabilities (see Sect. 6.1). However, this method does not scale well beyond the smallest e-Infrastructure use cases.

It is recommended to use Dynamic DNS updates to bootstrap and further operating the ARCHERY data inside DNS database. From DNS services operations point of view this requires one time initial configuration—allow updates from `archery-manage` by means of specifying shared transaction signature key (TSIG) in DNS service configuration.

It is generally advised to store ARCHERY data in dedicated DNS zones keeping only ARCHERY information (e.g. use archery.mydomain.org for all the ARCHERY records). This approach will allow the isolation of the scope of update operations and eliminate the possibility of record corruption. DNS infrastructure allows distributed administration by means of DNS *zone delegation* (using NS type resource records and glue address records in the parent zone [29]). From the naming point of view, the zone itself is represented by the sub-domain. The `archery-manage` tool keeps possible zone delegations in mind and implies the DNS naming scheme that introduces new sub-domain for each grouping object in the lower layer of the hierarchy. This allows to distribute ARCHERY administration (delegate) on the various grouping levels in terms of both DNS hosting and modifying the data.

7.2 Use-case 1: single-group registry for a small research community

In the first example, we present a small research group that needs a minimalist solution to register the available services it makes use of. The MolDynGrid Virtual Laboratory [33] conducts interdisciplinary research in computational structural biology and bioinformatics by means of relying on e-Infrastructure for in silico calculations of molecular dynamics of biological macromolecules. MolDynGrid uses several computing and storage facilities of the Ukrainian National Grid Infrastructure.

ARCHERY had been deployed for MolDynGrid needs in the simplest possible manner (Fig. 7). The topology source for the MolDynGrid ARCHERY is a simple configuration file containing a list of computing resources. In this case, all the services are grouped under a single ARCHERY group object used as the entry point in the DNS registry.

All ARCHERY records are managed within a single dedicated DNS zone (`index.moldyngrid.org`). DNS records define one *Group Object* (`_archery` RRSet) referencing a set of *Service Objects* (e.g. `d91b974761._archery` RRSet) with endpoints data. Notice that DNS names for Service Objects are generated by `archery-manage` using hashing algorithm.

The Registry was populated manually by using the `archery-manage` tool with 1 day TTL value. In case the service information of the MolDynGrid community would change then the registry administrator would need to manually rerun the—`archery-manage` tool with the updated configuration file.

7.3 Use-case 2: a hierarchical registry for the EGI large scale e-Infrastructure

EGI represents an example of a world-wide large scale e-Infrastructure for scientific research [23]. The distributed research environment consists of thousands of services hosted by hundreds of sites organized by numerous administrative domains on various levels (Tiers).

Efficient management of information for entire EGI starts with proper topology design. Proposed ARCHERY grouping follows the existing EGI hierarchical organizational structure using the GOCDB [26] as a topology information source. Within EGI all services are grouped by Sites that in turn further organized in administrative groupings called the National Grid Infrastructures (NGIs).

In this specific deployment example we propose a possible way of organizing and representing the EGI NGI, Site and service topology within the ARCHERY registry. Furthermore, a full-scale test deployment corresponding to the current EGI service catalogue stored in GOCDB was carried out: a DNS-based ARCHERY instance with real-life EGI service data was setup and analyzed (see Sect. 8).

The proposed DNS-embedded registry structure is illustrated on Fig. 8. Here, we assume that the e-Infrastructure owners have DNS management rights to certain DNS zones. A straightforward assumption is that every NGI or major administrative domain such as CERN are acting as DNS administrators of their DNS zones. For example, the top-level entry-point Group Object for EGI

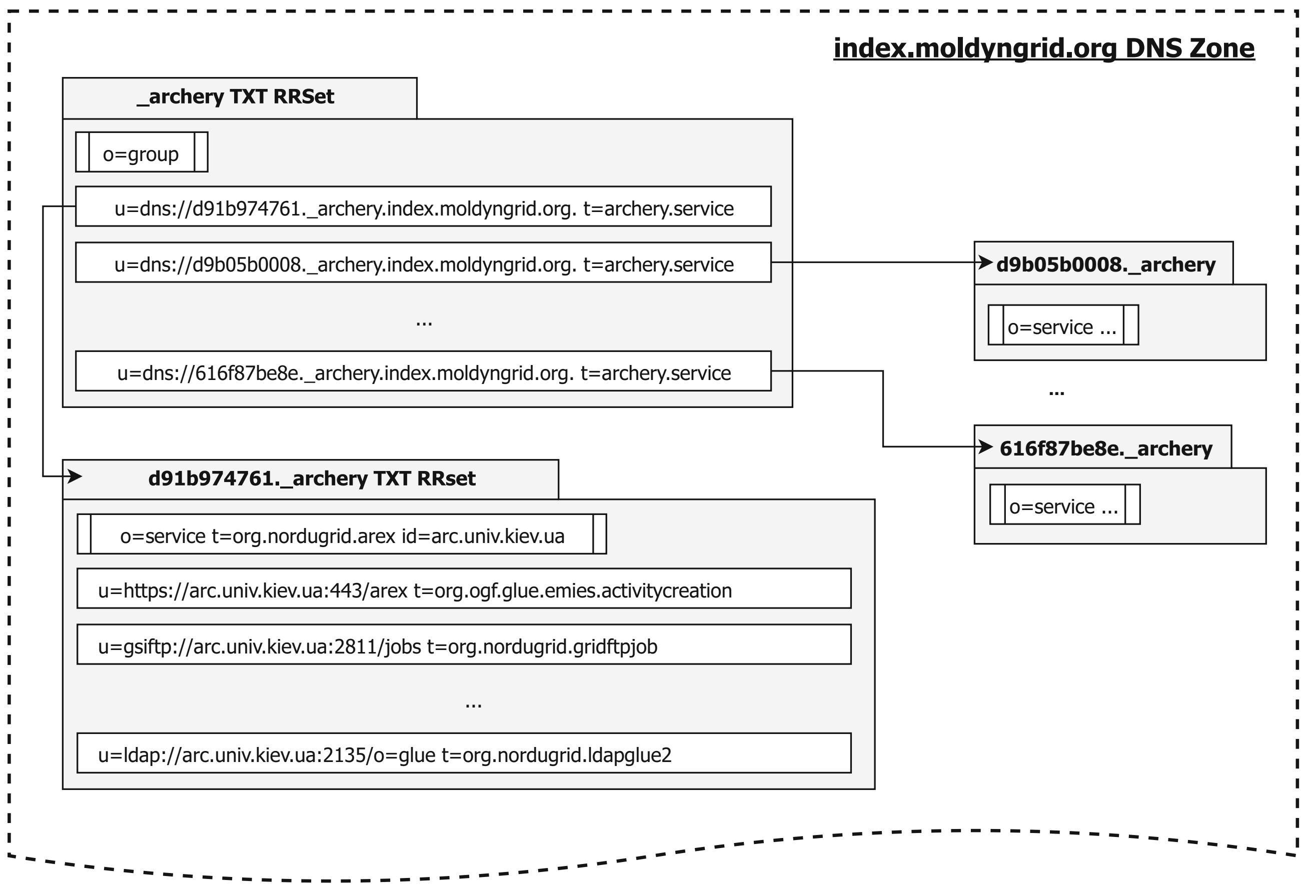
index.moldyngrid.org DNS Zone
_archery TXT RRSet
o=group
u=dns://d91b974761._archery.index.moldyngrid.org. t=archery.service
u=dns://d9b05b0008._archery.index.moldyngrid.org. t=archery.service
...
u=dns://616f87be8e._archery.index.moldyngrid.org. t=archery.service
d9b05b0008._archery
o=service ...
...
616f87be8e._archery
o=service ...
d91b974761._archery TXT RRset
o=service t=org.nordugrid.arex id=arc.univ.kiev.ua
u=https://arc.univ.kiev.ua:443/arex t=org.ogf.glue.emies.activitycreation
u=gsiftp://arc.univ.kiev.ua:2811/jobs t=org.nordugrid.gridftpjob
...
u=ldap://arc.univ.kiev.ua:2135/o=glue t=org.nordugrid.ldapglue2

Fig. 7 The MolDynGrid service registry records embedded in the index.moldyngrid.org zone. Please note that the minimalistic setup requires an ARCHERY group object to store all the available services. The service TXT RRSets themself are holding their endpoint information

e-Infrastructure stored within `_archery.egi.eu` RRSet in `egi.eu` DNS zone managed by the central EGI organization. This `_archery.egi.eu` object groups further grouping objects each of them representing NGIs. We assume a distributed operation model therefore we propose to introduce dedicated DNS zones for each NGI. A dedicated DNS zone for NGI can either use independent domain name e.g. `ndgf.org` zone managed by the NDGF organization or a sub-domain such as `ua.egi.eu` zone managed by Ukrainian NGI.

Authoritative name server for each zone can be located anywhere in the network. We propose to consider several operational cases for maintaining the NGI zone data:

(1) All NGI-based zones hosted on the central e-Infrastructure name server (e.g. EGI.eu). Each NGI receives TSIG key to update zone data.
(2) Each of the NGI-based zones are hosted on dedicated NGI-managed name servers. Each NGI defines own update procedures.
(3) Both central and NGI-managed approaches are used, depending on the particular NGI.

The presented use-case illustrates another grouping topology where certain sites of NGIs are organized in different Tiers. This Tier view can be used as classification mechanism of sites and their offered resources (Tier-0 is the central site/service, Tier-1 represents primary sites while Tier-2s are smaller resources). On the Fig. 8 the `_archery.wlcg.cern.ch` grouping object stored in the `cern.ch` DNS zone implements the above described Tier-based grouping. For example, the sites classified as Tier-1, such as NDGF-T1 stored in the `NDGF-T1.ndgf.org` RRSet, are grouped within the `T1.cern.ch` archery grouping object of type `t=wlcg.tier`. Please observe that there is no need to duplicate records describing the sites and services that are already present in the ARCHERY hierarchy describing the EGI topology. It is enough to add another ARCHERY Endpoint reference into tier-based Group Object.

The presented example showed that ARCHERY data model and rendering allows embedding several hierarchical overlapping e-Infrastructure topologies into the DNS registry without record duplication.

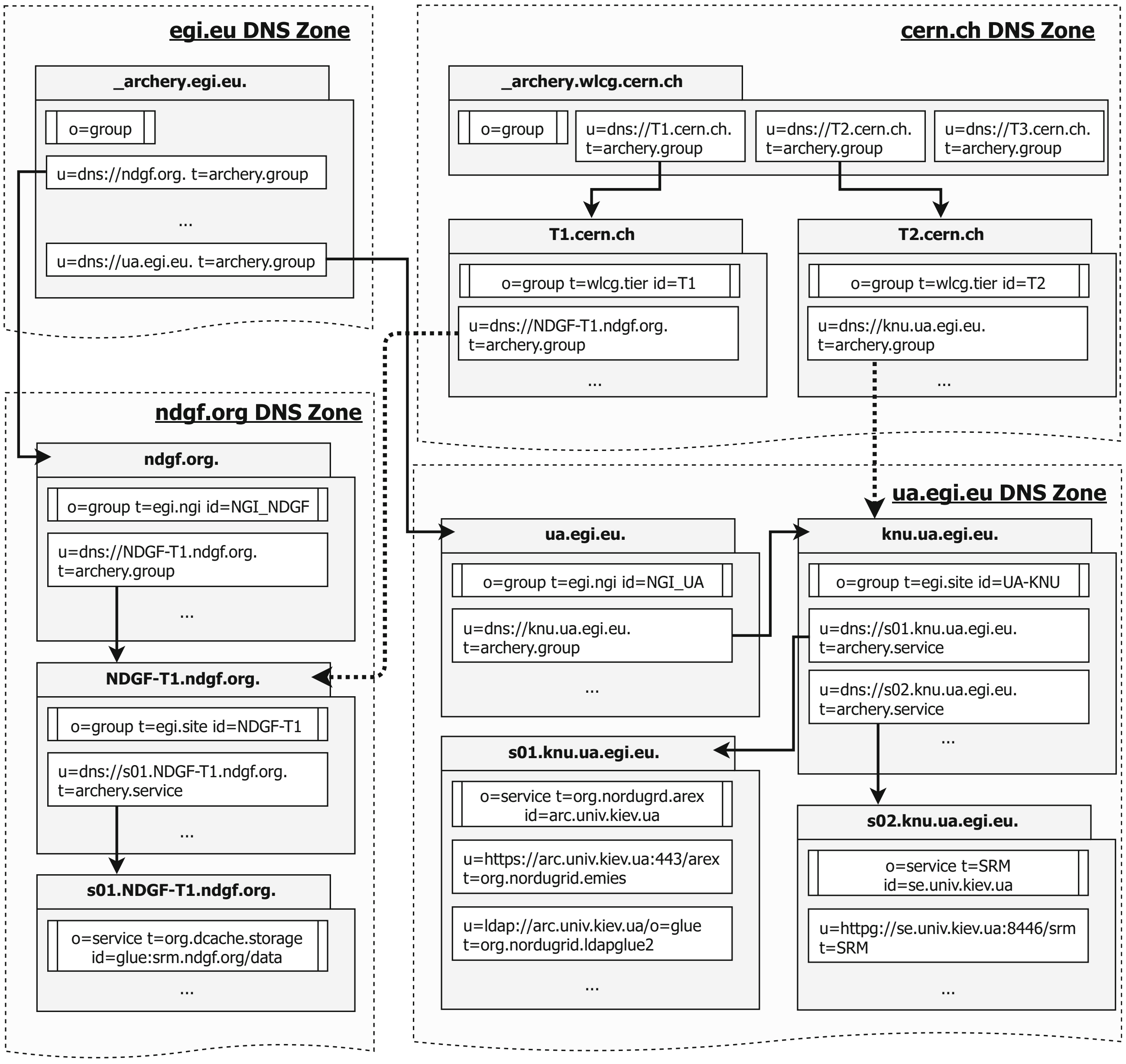
egi.eu DNS Zone
_archery.egi.eu.
o=group
u=dns://ndgf.org. t=archery.group
...
u=dns://ua.egi.eu. t=archery.group
cern.ch DNS Zone
_archery.wlcg.cern.ch
o=group
u=dns://T1.cern.ch. t=archery.group
u=dns://T2.cern.ch. t=archery.group
u=dns://T3.cern.ch. t=archery.group
T1.cern.ch
o=group t=wlcg.tier id=T1
u=dns://NDGF-T1.ndgf.org. t=archery.group
...
T2.cern.ch
o=group t=wlcg.tier id=T2
u=dns://knu.ua.egi.eu. t=archery.group
...
ndgf.org DNS Zone
ndgf.org.
o=group t=egi.ngi id=NGI_NDGF
u=dns://NDGF-T1.ndgf.org. t=archery.group
...
NDGF-T1.ndgf.org.
o=group t=egi.site id=NDGF-T1
u=dns://s01.NDGF-T1.ndgf.org. t=archery.service
...
s01.NDGF-T1.ndgf.org.
o=service t=org.dcache.storage id=glue:srm.ndgf.org/data
...
ua.egi.eu DNS Zone
ua.egi.eu.
o=group t=egi.ngi id=NGI_UA
u=dns://knu.ua.egi.eu. t=archery.group
...
knu.ua.egi.eu.
o=group t=egi.site id=UA-KNU
u=dns://s01.knu.ua.egi.eu. t=archery.service
u=dns://s02.knu.ua.egi.eu. t=archery.service
...
s01.knu.ua.egi.eu.
o=service t=org.nordugrd.arex id=arc.univ.kiev.ua
u=https://arc.univ.kiev.ua:443/arex t=org.nordugrid.emies
u=ldap://arc.univ.kiev.ua/o=glue t=org.nordugrid.ldapglue2
...
s02.knu.ua.egi.eu.
o=service t=SRM id=se.univ.kiev.ua
u=httpg://se.univ.kiev.ua:8446/srm t=SRM
...

Fig. 8 Hierarchical e-Infrastructure topologies embedded into several DNS zones. The use-case represents a proposal for storing EGI information in ARCHERY. The egi.eu and cern.ch zones are the top entries in the two overlapping hierarchies organizing sites either by NGIs or Tiers

8 ARCHERY deployments results

What follows, we present the results of two ARCHERY deployments. The first one is the production roll-out of an ARCHERY registry serving the Nordic e-Science community. This registry operates under the `nordugrid.org` entry point embedded in the DNS and replacing the previous LDAP-based EGIIS service.

As a second ARCHERY test deployment we have chosen to set up a dedicated ARCHERY registry to accommodate the complete service information from the EGI e-Infrastructure using the GOCDB [13] as the topology source. This second registry is operated under the `egi.grid.org.ua` endpoint and had been established in a dedicated DNS service for the purpose of various performance measurements. Please note that the distributed ARCHERY administration within different DNS zones as proposed in Sect. 7.3 was not configured but it is not affecting the performance measurements itself.

8.1 ARCHERY objects

As part of the two deployment cases the ARCHERY DNS service registries were populated with real-life data. The nordugrid.org production registry contains data describing typical Nordic e-Infrastructure services such as the ARC Computing Elements while the egi.-grid.org.ua test registry holds site, organization and service data for all the available EGI services. It is important to notice that the registries are populated with full-size real data.

This section contains the object size analysis for both ARCHERY deployments. Thanks to minimalistic data model the total size of ARCHERY database is small: 145 objects (107.15 kB in total) for the Nordic production deployment and 1831 objects (917.19 kB in total) for EGI test deployment. Please compare these numbers to the original 1980s Hesiod DNS-embedded deployment of 10k TXT records of 3 MB data [12].

Figures 9 and 10 show the object size analyses for the Nordic and test EGI deployments respectively.

For the Nordic deployment (Fig. 9) a typical NorduGrid ARC [14] service may contain up to dozens of endpoints. The most typical objects within the nordugrid.org ARCHERY correspond to ARC CEs part of the WLCG infrastructure. These computing services usually have about five endpoints and the corresponding service objects have the size between 500 and 800 bytes. The size of the service object depends on the number of endpoints and mostly affected by the length of the service endpoint URL.

In the case of the EGI test deployment (Fig. 10) most of the objects are describing various EGI services. The size of those service objects is typically less than 512 bytes. The small size of these service objects is due to the fact that a typical EGI service has only one endpoint. Interesting to note that the largest observed EGI object with a 22 kB size is also a service object (an unique storage service with around 150 endpoints). In EGI the Site grouping objects belong to the larger records with about 1 kb size. It is because the typical EGI sites are aggregating around a dozen services. All that said, the 75% of all EGI objects fit into the 512 bytes.

It is important to notice that the majority of the objects in both deployments fit to DNS response limit of the default UDP transport protocol. Furthermore, even the largest occurring object in EGI (of size 22 kB) is much smaller than the 65 kB TCP limit.

8.2 ARCHERY performance analysis

As we target the ARCHERY as a Top-BDII replacement service, we conducted the performance analysis to compare performance of these two systems.

We are not including the legacy EGIIS service of the NorduGrid e-Infrastructure into the performance comparison. Contrary to the Top-BDII and the ARCHERY, EGIIS does not perform any active information pulling. With no information pulling, EGIIS performance was never the limiting factor, while its limited data model with a single endpoint type and huge associated management overhead lead to its deprecation.

The testing environment contained three identical machines: Supermicro X10SLH with Intel Xeon E3-1220 CPU v3 @ 3.10 GHz, 8 GB of DDR3 RAM, 1 Gbps NIC connectivity. They are running CentOS Linux 7.7 as the operating system. The identical setup allowed us to focus on the performance comparison instead of absolute values.

Software layer deployed on the test servers:

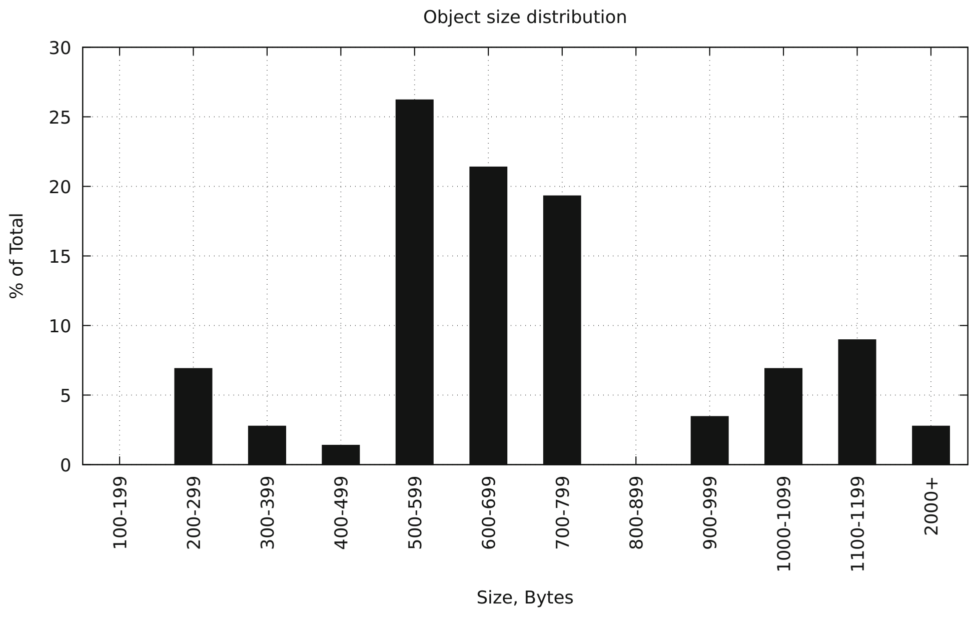

Fig. 9 ARCHERY Objects size distribution in the production Nordic deployment. The peak in the middle contains the ARC services within WLCG while the second peak on the right side contains the general ARC CE objects with typically more endpoints

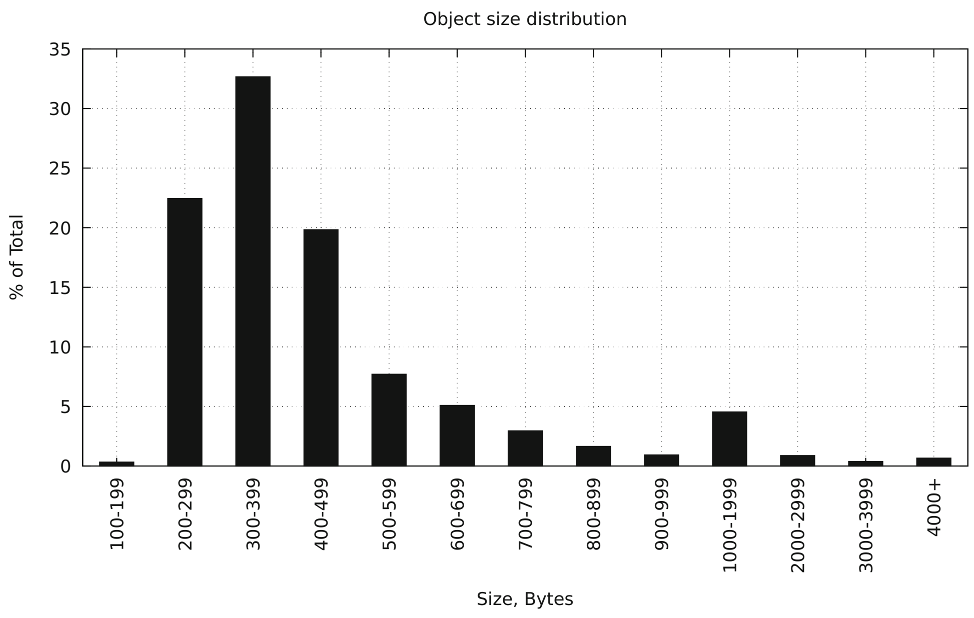

Fig. 10 ARCHERY Objects size distribution for the EGI test deployment. The large peak at 300–400 bytes correspond to service objects

(1) Top-BDII service (version 5.2.23 installed from Extra Packages for Enterprise Linux repository),
(2) ARCHERY registry embedded in a dedicated ISC BIND [21] DNS service (version 9.11.4 installed from OS repository),
(3) The `archery-manage` utility to populate the ARCHERY registry.

Because the Top-BDII service does not provide a security layer, DNSSEC is not enabled for ARCHERY zone in our test infrastructure for performance results comparison.

The `archery-manage` tool had been configured to run periodically by CRON service. Invocation interval is configured based on the data measured for Top-BDII. We used the `sysstat` (system statistics) performance monitoring tools on Linux to measure various system loads, including CPU activity, memory usage and network utilization simultaneously.

The time period shown on the following figures represents the same measurement window. Activity during this interval is representing several periods of Top-BDII and ARCHERY normal operation cycles. It continues exactly the same beyond shown time-frame.

We performed two sets of measurements corresponding the use-cases described in Sect. 7: the hierarchical EGI large-scale e-Infrastructure and a small research community with trivial flat topology. Results are presented below.

8.2.1 EGI e-Infrastructure

In the first use-case, both Top-BDII setup and ARCHERY represents the EGI e-Infrastructure with the same service set taken from EGI GOCDB [26] as a topology source. EGI is the largest existing distributed scientific e-Infrastructure thus with our test we conducted measurements on the largest available full-size production data. Information fetching performed every 10 min.

Network usage of Top-BDII, ISC BIND hosting the ARCHERY registry and `archery-manage` is shown on Fig. 11. The network usage is a known limitation of Top-BDII as clearly visible on the logarithmic-scale graph. Network usage by `archery-manage` is smaller by a factor of 100 during the service data fetching. The ARCHERY-DNS (ISC BIND) network traffic is also small because it needs to fetch only approximately 1 MB of data and then issue an even smaller incremental update.

We had also measured total traffic in addition to throughput: Top-BDII network consumption is near to 800 GB monthly (25.5 GB daily), `archery-manage` running on the same update rate is around 18 GB monthly (780 MB daily) and ARCHERY-DNS uses 1.6 GB monthly (240 MB daily).

Comparing the CPU load (Fig. 12) it is noticeable that Top-BDII uses lot of CPU power just after the data fetching period (3–6 min time interval on the network traffic figure Fig. 11) which is clearly visible as a CPU load peak in the 6–9 min time interval on the CPU load figure (Fig. 12). It is because of the data post-processing within the Top-BDII LDAP database update that is rather CPU intensive. As a comparison the `archery-manage` converts data to DNS rendering with minimal CPU usage thanks to minimal dataset. Please notice that the ARCHERY-DNS in ISC BIND consumes almost zero CPU therefore not visible on the figure. This is despite the fact that as part of the incremental registry update performed by the `archery-manage` tool the entire ARCHERY data is

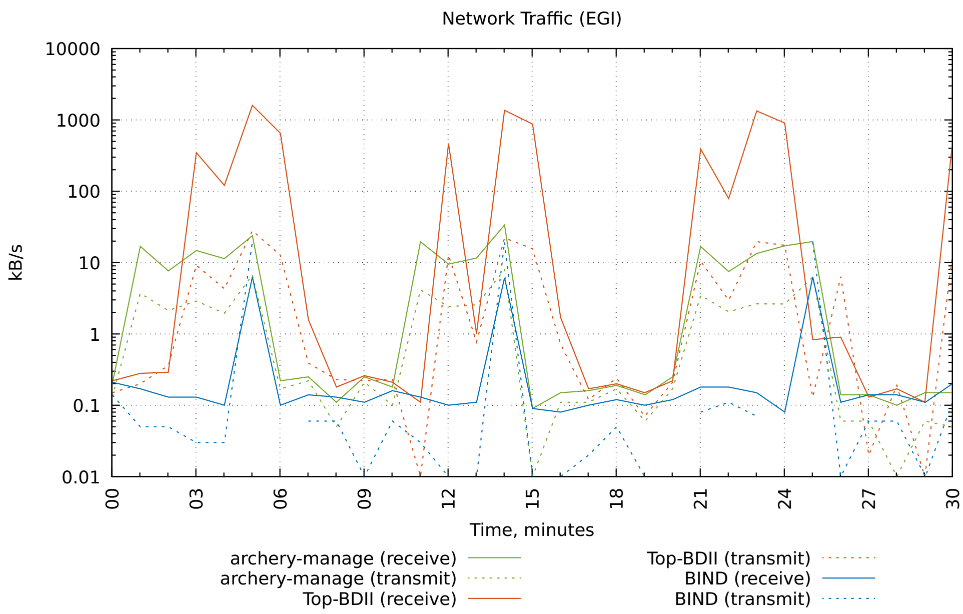

Fig. 11 Network traffic comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting ARCHERY registry for EGI e-Infrastructure. Please note that the periodic peaks correspond to the registries updates and the network traffic shown in a logarithmic scale

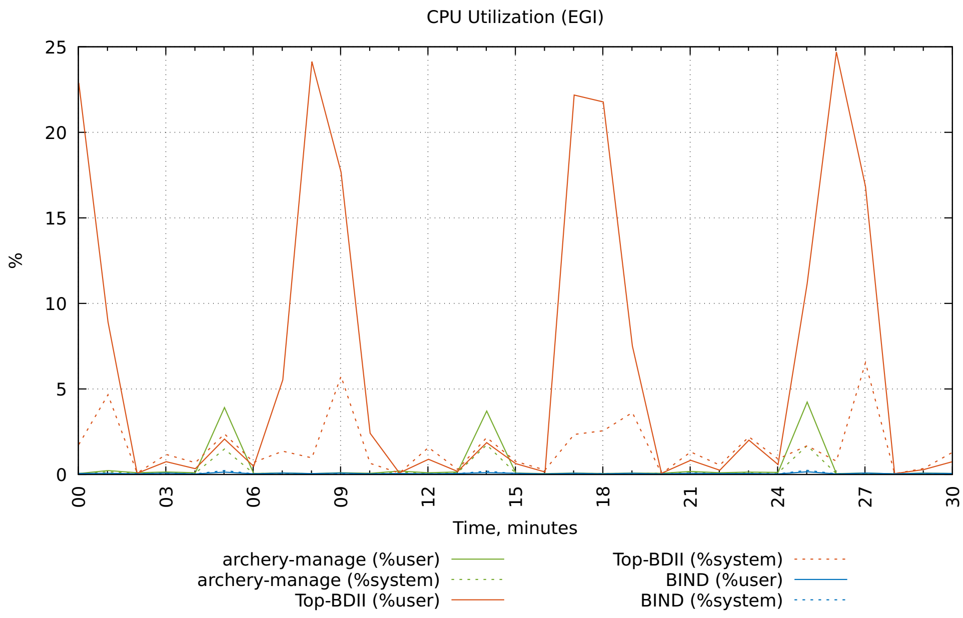

Fig. 12 CPU Utilization comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting ARCHERY registry for EGI e-Infrastructure. The generated load of BIND is almost zero therefore not visible

fetched from the DNS in every update period. This also serves as a proof for the client-side query scalability of ARCHERY.

Memory usage comparison is shown on Fig. 13. The average Top-BDII memory consumption is approximately 4.5 GB with an additional 500 MB per update cycles. Both `archery-manage` and ARCHERY-DNS does not demonstrate serious memory usage during update cycles and requires less than 1 GB RAM to run the service.

Performance metrics for large hierarchical e-Infrastructure use-case shows significant advantage of using ARCHERY from resource usage point of view.

8.2.2 Small research community

For a small research community use-case, the MolDynGrid setup was analysed (Sect. 7.2). We performed another set of measurement to collect resource usage metrics of dynamic endpoint information updates for the flat trivial topology of the MolDynGrid. Configuration of Top-BDII and ARCHERY has been altered to fetch information from 10 sites serving the community. The fetching frequency is increased to run the processes every 2 min.

The network usage of analysed services is shown on Fig. 14. Despite the amount of information to fetch is significantly smaller compared to EGI use-case, it is still

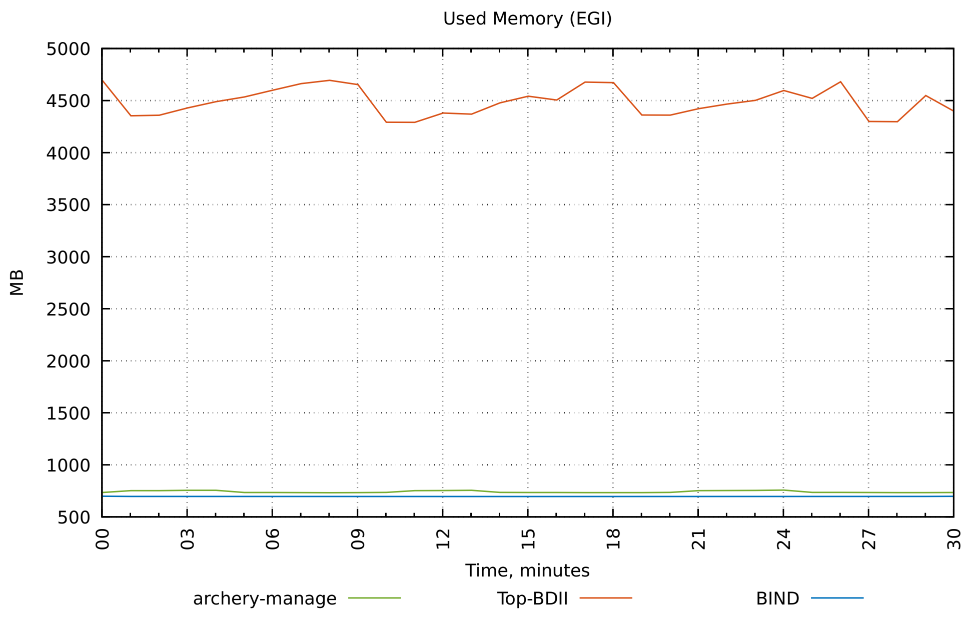

Fig. 13 Memory Utilization comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting the ARCHERY registry for EGI e-Infrastructure. Note the approximately 500 MB periodic memory load increase of Top-BDII

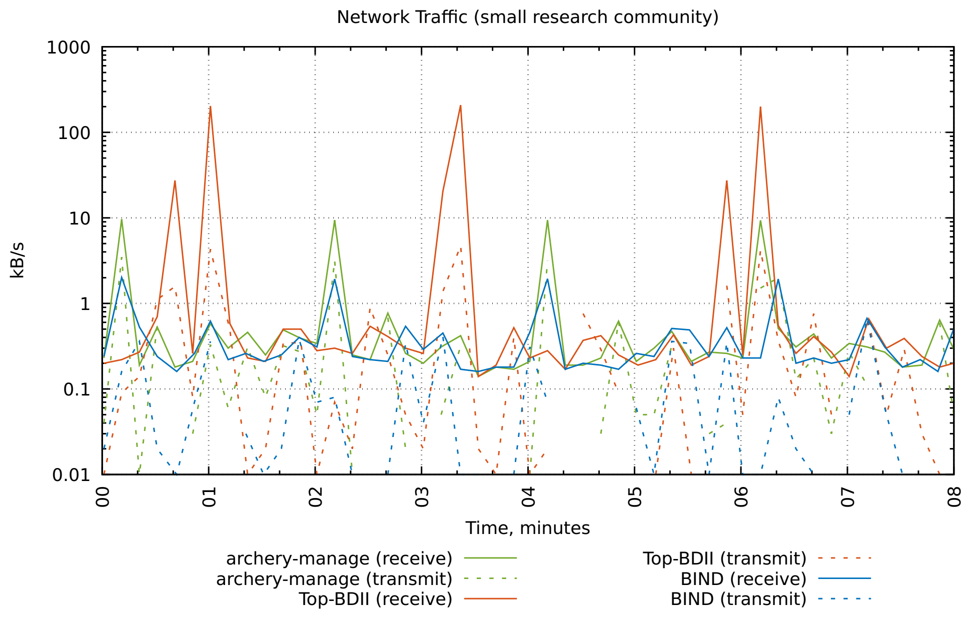

Fig. 14 Network traffic comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting ARCHERY registry for small research community. Please note the network traffic shown in a logarithmic scale

noticeably less traffic required for ARCHERY operation. It is also noticeable that `archery-manage` shows more stable network usage metrics compared to Top-BDII.

The CPU load for ARCHERY and Top-BDII in the small community use-case (Fig. 15) is comparable, but contrary to ARCHERY, Top-BDII has noticeable system CPU time, indicating additional handling of intensive disk writing.

Please also note, that ARCHERY resource consumption remains almost on the same level for both large e-Infrastructure (Fig. 12) and small community use-case (Fig. 15), proving the good scalability of the `archery-manage`.

Looking into memory consumption (Fig. 16), it is noticeable that Top-BDII even in the small 10-sites use-case have approximately 2 GB of RAM footprint, while `archery-manage` as well as ISC BIND holding ARCHERY database keeps low memory usage even for large EGI e-Infrastructure.

To conclude, compared to the Top-BDII setup, ARCHERY saves memory and network resources even in the simple small research community use-case.

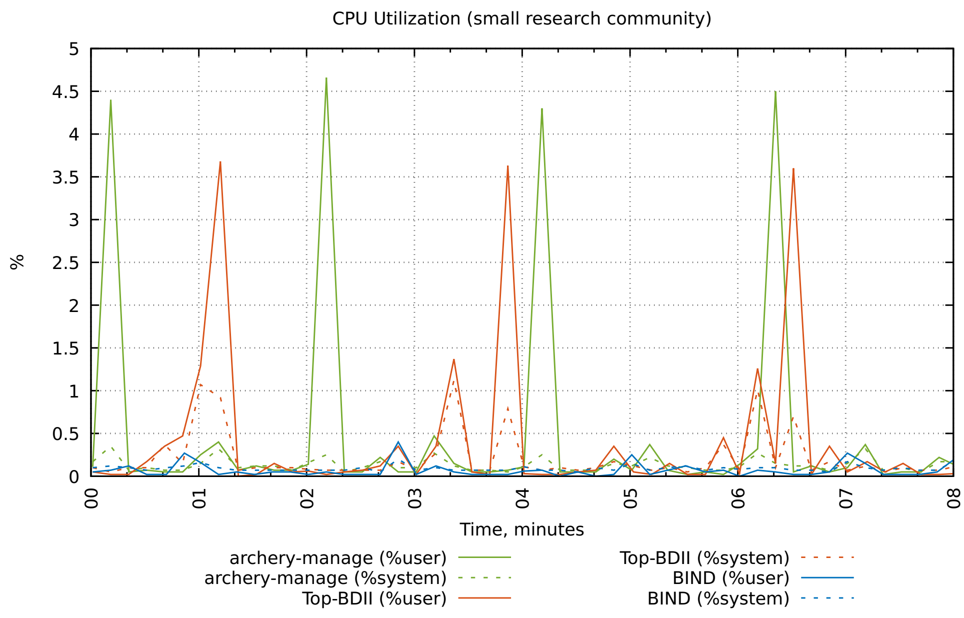

Fig. 15 CPU Utilization comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting ARCHERY registry for small research community

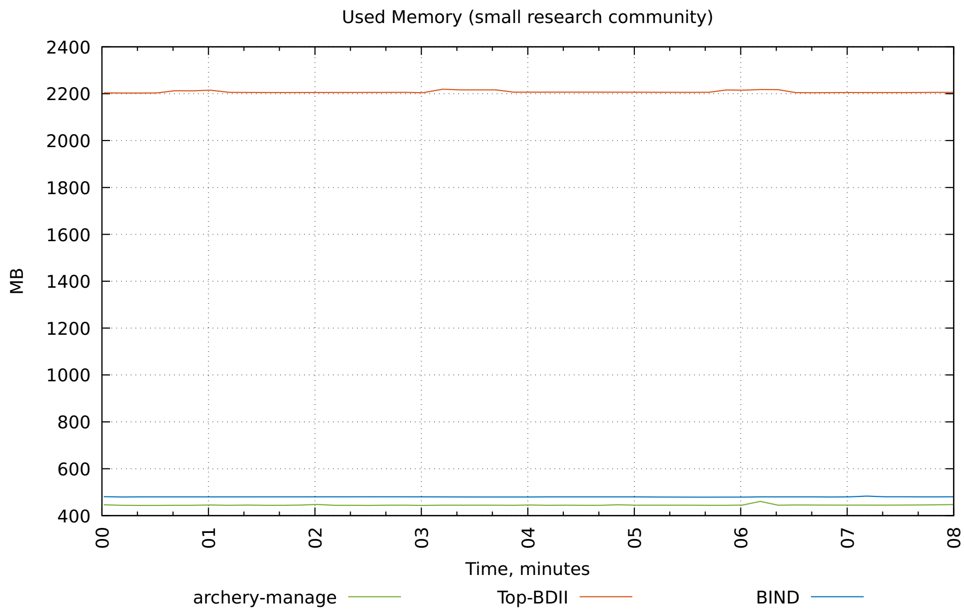

Fig. 16 Memory Utilization comparison for hosts running: archery-manage, Top-BDII and ISC BIND hosting the ARCHERY registry for small research community

9 Further work

We will continue the deployment of ARCHERY registries for various e-Infrastructures. As part of the deployment process we will collect feedback from various communities and review the ARCHERY service model based on the deployment experience. If necessary, an extension of the service description may take place by introducing additional minimalistic information such as Virtual Organization affiliation.

We plan to consider the integration of ARCHERY with some monitoring tools so that the endpoint state in ARCHERY would reflect actual functional testing results and not only network availability. Along the same line we will investigate other approaches to efficiently actualize the endpoint state in the DNS database, such as DNS-based Blackhole List (DNSBL).

The idea behind ARCHERY, that is storing e-Infrastructure metadata in DNS records, could be applied to implement a software environment catalogue. In such a system the DNS records would be enriched with software environment information.

10 Summary

In this paper we proposed a novel approach to utilize the DNS infrastructure as an information source for distributed computing service endpoints. The underlying DNS infrastructure out-of-the-box provides integrity, fault-tolerance and network-level caching. Furthermore, the native DNS delegation processes allow to create federations of service endpoints managed under different DNS zones. Using incremental dynamic DNS in updates allows us to keep information up to date and apply filtering based on service availability monitoring.

The proposed approach, including the data model definition was implemented as the ARC Hierarchical Endpoints Registry (ARCHERY) system and got integrated with the NorduGrid ARC middleware. The software-layer integration did not require development of additional services since ARCHERY reuses well-known DNS services that can be simply queried with any DNS library or command line client, simplifying the client-side integration.

Successful deployments of the ARCHERY system were presented in the paper including performance comparison to existing endpoint registries. In particular, ARCHERY has been successfully deployed for the Nordic High Energy Physics community and is used in production and demonstrates stable operation.

The DNS-based registry specification and the ARCHERY implementation is general enough to be used for other distributed e-Infrastructures as it was shown in the proposed EGI use-case scenario.

Acknowledgements This publication is part of research work at Lund University during 2018–2020, funded by a Swedish Institute scholarship [36]. We thank Dmytro Karpenko (University of Oslo) and Petter Urkedal (Niels Bohr Institute) for the first deployments of ARCHERY for the NorduGrid e-Infrastructure.

Funding Open access funding provided by Lund University.

References

1. Andrzejak, A., Xu, Z.: Scalable, efficient range queries for grid information services. In: Second International Conference on Peer-to-Peer Computing, 2002 (P2P 2002). Proceedings, pp. 33–40. IEEE (2002)
2. Anisenkov, A., Di Girolamo, A., Klimentov, A., Oleynik, D., Petrosyan, A., Collaboration, A., et al.: AGIS: the ATLAS grid information system. J. Phys. Conf. Ser. **513**, 032001 (2014)
3. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033. IETF (2005). http://tools.ietf.org/rfc/rfc4033.txt. Accessed 13 Oct 2021
4. Bellis, R.: DNS Transport over TCP—Implementation Requirements. RFC 5966. IETF (2010). http://tools.ietf.org/rfc/rfc5966.txt. Accessed 13 Oct 2021
5. Brun, R., Carminati, F., Carminati, G.G.: From the Web to the Grid and Beyond: Computing Paradigms Driven by High-Energy Physics. Springer, Berlin (2012)
6. CERN: The WLCG REsource, Balance and Usage (REBUS). https://wlcg-rebus.cern.ch/apps/topology/. Accessed 13 Oct 2021
7. CERN: Top-BDII. http://gridinfo.web.cern.ch/introduction. Accessed 13 Oct 2021
8. Cheshire, S., Krochmal, M.: DNS-Based Service Discovery. RFC 6763. IETF (2013). http://tools.ietf.org/rfc/rfc6763.txt. Accessed 13 Oct 2021
9. Crocker, D., Hansen, T., Kucherawy, M.: Domain Keys Identified Mail (DKIM) Signatures. RFC 6376. IETF (2011). http://tools.ietf.org/rfc/rfc6376.txt. Accessed 13 Oct 2021
10. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, 2001. Proceedings, pp. 181–194. IEEE (2001)
11. Damas, J., Graff, M., Vixie, P.: Extension Mechanisms for DNS (EDNS(0)). RFC 6891. IETF (2013). http://tools.ietf.org/rfc/rfc6891.txt. Accessed 13 Oct 2021
12. Dyer, S.P.: The Hesiod name server. In: USENIX Winter, pp. 183–189 (1988)
13. EGI service provided by STFC: Grid Configuration Database (GOCDB). https://goc.egi.eu/. Accessed 13 Oct 2021
14. Ellert, M., Grønager, M., Konstantinov, A., Konya, B., Lindemann, J., Livenson, I., Nielsen, J., Niinimäki, M., Smirnova, O., Wäänänen, A.: Advanced resource connector middleware for lightweight computational grids. Future Gen. Comput. Syst. **23**(2), 219–240 (2007). https://doi.org/10.1016/j.future.2006.05.008
15. Field, L., Memon, S., Márton, I., Szigeti, G.: The EMI Registry: discovering services in a federated world. J. Grid Comput. **12**(1), 29–40 (2014). https://doi.org/10.1007/s10723-013-9284-1
16. Gulbrandsen, A., Vixie, P., Esibov, L.: A DNS RR for Specifying the Location of Services (DNS SRV). RFC 2782. IETF (2000). http://tools.ietf.org/rfc/rfc2782.txt. Accessed 13 Oct 2021
17. Hoffman, P.E., Sullivan, A., Fujiwara, K.: DNS Terminology. RFC 8499. IETF (2019). http://tools.ietf.org/rfc/rfc8499.txt. Accessed 13 Oct 2021
18. ICANN Research: TLD DNSSEC Report. ICANN Research (2018). http://stats.research.icann.org/dns/tld_report/. Accessed 13 Oct 2021
19. Internet Assigned Numbers Authority: Root Servers. https://www.iana.org/domains/root/servers. Accessed 13 Oct 2021
20. Internet Systems Consortium: DNS RFCs. https://www.isc.org/community/rfcs/dns/. Accessed 13 Oct 2021
21. Internet Systems Consortium, Inc., et al.: ISC BIND. Internet Systems Consortium, Inc. (2005)

22. Kitterman, S.: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1. RFC 7208. IETF (2014). http://tools.ietf.org/rfc/rfc7208.txt. Accessed 13 Oct 2021
23. Kranzlmüller, D., de Lucas, J.M., Öster, P.: The European Grid Initiative (EGI). In: Remote Instrumentation and Virtual Laboratories, pp. 61–66. Springer, New York (2010)
24. Lottor, M.: Domain Administrators Operations Guide. RFC 1033. IETF (1987). http://tools.ietf.org/rfc/rfc1033.txt. Accessed 13 Oct 2021
25. Mahato, D.P., Sandhu, J.K., Singh, N.P., Kaushal, V.: On scheduling transaction in grid computing using cuckoo search-ant colony optimization considering load. Clust. Comput. **23**(2), 1483–1504 (2020). https://doi.org/10.1007/s10586-019-03016-x
26. Mathieu, G., Richards, A., Gordon, J., Novales, C.D.C., Colclough, P., Viljoen, M.: GOCDB, a topology repository for a worldwide grid infrastructure. J. Phys. Conf. Ser. **219**, 062021 (2010)
27. MIT Kerberos Consortium: MIT Kerberos Documentation: Realm Configuration Decisions. https://web.mit.edu/kerberos/. Accessed 13 Oct 2021
28. Mockapetris, P.: Domain Names—Concepts and Facilities. RFC 1034. IETF (1987). (http://tools.ietf.org/rfc/rfc1034.txt).Accessed 13 Oct 2021
29. Mockapetris, P.: Domain Names—Implementation and Specification. RFC 1035. IETF (1987). (http://tools.ietf.org/rfc/rfc1035.txt). Accessed 13 Oct 2021
30. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., et al.: The Open Science Grid status and architecture. J. Phys. Conf. Ser. **119**, 052028 (2008)
31. Puppin, D., Moncelli, S., Baraglia, R., Tonellotto, N., Silvestri, F.: A grid information service based on peer-to-peer. In: European Conference on Parallel Processing, pp. 454–464. Springer (2005)
32. Rose, S., Wijngaards, W.: DNAME Redirection in the DNS. RFC 6672. IETF (2012). http://tools.ietf.org/rfc/rfc6672.txt. Accessed 13 Oct 2021
33. Salnikov, A., Sliusar, I., Sudakov, O., Savytskyi, O., Kornelyuk, A.: MolDynGrid Virtual Laboratory as a part of Ukrainian academic grid infrastructure. In: IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2009. IDAACS 2009, pp. 237–240. IEEE (2009)
34. Smirnova, O.: The Grid Monitor. http://www.nordugrid.org/documents/monitor.pdf. Accessed 13 Oct 2021
35. Stichting NLnet Labs: Unbound. https://nlnetlabs.nl/projects/unbound/about/. Accessed 13 Oct 2021
36. Swedish Institute: The Visby Programme. https://si.se/en/apply/visby-programme/. Accessed 13 Oct 2021
37. van der Toorn, O., van Rijswijk-Deij, R., Fiebig, T., Lindorfer, M., Sperotto, A.: TXTing 101: finding security issues in the long tail of DNS TXT records. In: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), pp. 544–549 (2020). https://doi.org/10.1109/EuroSPW51379.2020.00080
38. Vixie, P., Gudmundsson, O., Eastlake III, D., Wellington, B.: Secret Key Transaction Authentication for DNS (TSIG). RFC 2845. IETF (2000). http://tools.ietf.org/rfc/rfc2845.txt. Accessed 13 Oct 2021
39. Vixie, P., Thomson, S., Rekhter, Y., Bound, J.: Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136. IETF (1997). http://tools.ietf.org/rfc/rfc2136.txt. Accessed 13 Oct 2021

Andrii Salnikov is a Systems Engineer and a Researcher within a field of Distributed Computing and Networking. He earned his PhD in the area of Engineering Sciences and Computer Systems researching the integration of Virtual Organizations in the computational grid infrastructures. He has many year of experience of developing and operating large-scale systems designed for e-Science communities. Andrii is one of the most active contributor of the Advanced Resource Connector middleware used within WLCG e-Infrastructure.

Balázs Kónya is a Senior Researcher at Lund University, Sweden and is working as the Technical Coordinator of the NorduGrid Collaboration. He has been involved in research, design, development and operation of various distributed systems targeting the needs of large scale e-science infrastructures such as the WLCG system. He received a PhD in Theoretical Physics from the University of Debrecen, Hungary.