GOLDIE: Harmonization and Orchestration Towards a Global Directory for IoT

Luoyao Hao and Henning Schulzrinne

Department of Computer Science, Columbia University, New York, NY, USA
Email: {lyhao, hgs}@cs.columbia.edu

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488752

***Abstract*—To scale the Internet of Things (IoT) beyond a single home or enterprise, we need an effective mechanism to manage the growth of data, facilitate resource discovery and name resolution, encourage data sharing, and foster cross-domain services. To address these needs, we propose a GlObaL Directory for Internet of Everything (GOLDIE). GOLDIE is a hierarchical location-based IoT directory architecture featuring diverse user-oriented modules and federated identity management. IoT-specific features include discoverability, aggregation and geospatial queries, and support for global access. We implement and evaluate the prototype on a Raspberry Pi and Intel mini servers. We show that a global implementation of GOLDIE could decrease service access latency by 87% compared to a centralized-server solution.**

I. Introduction

As more devices are connected to the Internet and they acquire additional capabilities, we are moving from the notion of the Internet of Things (IoT) to the stage of Internet of Everything (IoE), a broader view coined by Cisco, integrating people, processes, data, and things [1].

Most current IoT systems are restricted to a single home or enterprise, often with devices made by a single hardware manufacturer or device controller, such as Google Home or Apple HomeKit. Even if different manufacturers all rely on similar data models (e.g., JSON) for IoT devices, they do not interoperate. However, new application scenarios depend on integrating sensor data across heterogeneous devices and across administrative domains. As an example, commercial HVAC systems are often managed both locally and through device management companies. Energy usage monitoring is useful both within a home and, in aggregate, for utilities to detect outages.

As one of the key functions in computing systems, directory services maintain network resources so as to be identified in a flexible and efficient way [2]–[4]. IoT devices, given their heterogeneity, benefit from being reachable not just by name, but also by properties (metadata), whether location or sensing and actuating capabilities. For discovery, remote access, and device management, IoT directories are deemed to be an essential need [5]. With a global directory maintaining metadata comprising identifier, name, address, credentials, etc., IoT devices can therefore be exposed to a larger network, allowing common users to query and browse data.

However, current name-based directory solutions fail for IoT scenarios since they lack any of the four key features: (1) some directories are designed for unique IoT scenarios or single administrated, while barely extensible for global access; (2) IoT devices often have geographic restrictions, as devices may only be accessible within certain areas and are often useful only based on their geographic location; (3) many devices are not entirely public and device owners may want to limit the visibility of devices beyond designated administrative domains or beyond a geographic region; (4) IoT applications benefit from semantically rich queries, such as querying for aggregated information instead of data from a single sensor or particular types of data. Here, we implement a directory service that meets these key requirements.

The body of work on global IoT resource directory is not large, unfortunately. Kafle et al. [5] present a scalable IoT directory architecture; it dynamically assigns computing resources of virtual machines and updates cache replicas to guarantee the lookup performance. Badii et al. [6] leverage a directory service to manage registry and ownership of devices for smart city applications. Shelby et al. [7] analyze the REST architecture and interfaces of the COAP protocol for severely resource-constrained IoT devices. Jin and Kim [8] build resource directories based on Domain Name System (DNS) for transparent access in heterogeneous network. However, none of these meets all the design requirements spelled out in the previous paragraph.

Thus, we propose and implement GOLDIE (for GlObaL Directory for Internet of Everything), a federated and geographically distributed global directory for managing the metadata for IoT and IoE. GOLDIE provides a variety of lookup functions such as aggregation queries, location-based queries, customized queries, metadata discoverability, and federated authentication. Basic directory operations such as registration, query, update, and deletion, are designed to fit IoT scenarios. The design and implementation of GOLDIE seeks to conform to existing standards and solutions, while avoiding catering to a specific application domain or service. The metadata format adopted by our system conforms to the W3C thing description [9], a working standard to formally organize metadata of IoT as JSON objects. To enable directories to validate user identities through trusted parties, user authentication in GOLDIE builds on OpenID Connect (OIDC) [10], a commonly-used federated identity management model. We have implemented a prototype on a Raspberry Pi and Intel mini servers and have evaluated its performance.

Evaluations regarding a real-world dataset of four different applications justify the merits of deploying such a directory system. Compared to a centralized solution, service latency is reduced by 9.7%, 51.4%, 79.9%, and 86.9% for each application under optimal conditions.

We make the following contributions:

- We analyze and propose a series of design principles for a global IoT directory, covering architecture, discoverability, query functions, and identity management.
- We design and implement GOLDIE, a federated and location-based global directory to manage metadata for IoT. GOLDIE provides federated identity management and various query functions, apart from basic registration, lookup, update, and delete operations.
- We deploy GOLDIE on commodity devices and measure our implementation against standard generated profiles as well as a real-world dataset. Extensive evaluation on various functions validates the efficiency of GOLDIE.

The rest of this paper is organized as follows. Section II introduces the background and design issues for directory services based on the features of IoT. Section III proposes GOLDIE, with an overview of structures and available services. Section IV presents the design and implementation of all the important features, federated identity management, and supported operations. Section V evaluates our prototype and demonstrates the efficiency with a real-world dataset. Section VI discusses potential concerns of the proposed model. Section VII concludes the paper.

II. Directory Services for IoT

A metadata directory for IoT devices and services offers numerous benefits as long as it follows a set of design principles, summarized below. With a focus on building a global directory, we evaluate existing solutions to see how well they meet those principles.

A. Requirements for IoT Directories

To discover metadata about resources on the Internet, one may be able to query a "/.well-known" URL on the HTTP server. The server network address has been obtained by resolving a domain name, while the URL itself is typically either guessed by brand ("amazon.com") or obtained via a content-based search engine. While the "well-known" URL could be supported by IoT devices reachable by HTTP(S), this mechanism does not allow applications to find suitable IoT devices, as traditional content-based search engines or brand-based guessing are ill-suited for IoT devices. It also fails for devices that use other protocols, such as MQTT, or for devices that are not meant to be accessible globally. Some IoT devices have constrained energy or computational resources, so answering directory queries imposes an additional burden or interferes with their sleep cycles [7], [11], [12]. Thus, a resource directory, hosted on well-resourced servers, makes discovery more efficient. IoT resources register and update their information as needed on these directory services.

In many IoT scenarios, there is often a need to bundle multiple smart objects into a group or by function, e.g., all devices that make up a smart home. It is even possible to fabricate a virtual object, e.g., an intelligent logistics service or a data aggregation service or logging service that offers computed data or historical time series. To deal with such cases, rigid data storage formats and query supports are not helpful. A better solution is to let local directories reach out to end users, allowing users to register and manage their own IoT devices with related metadata.

Successful internet-scale directories such as DNS rely on a simple organizational and administrative hierarchy mapped to names, with usually three levels of hierarchy: top-level domain such as a country or category, delegated domain (example.com) and intra-organizational names (server1.example.com). Geographic location is usually irrelevant and often intentionally hidden, if the data is served by a CDN. This simple approach does not work well for IoT devices. Devices may be managed by multiple entities and are often best described by a geographic location or region and a specific function, leading to names that are unusable for searching by applications. Domain names can still be used as lower-level identifiers within directories, providing another layer of indirection and hiding changes in network attachment points from the directory. Often, the stability of such names is unpredictable, a new domain name may be assigned automatically by DHCP or the device itself when a device is replaced, even though it serves the same function as the old device it replaces. For example, an application gathering local temperature measurements may request data from thermometers in a polygon to average out measurement errors. Devices making up the ensemble to be queried are routinely added and removed, so an application should not rely on a domain name of unknown stability but rather encode a semantic description of the types of measurements or actuators that it needs.

Our proposed IoT directory acts as an intermediate that provides two facets of decoupling, shielding IoT programs from changes in device identities and unify access to aggregated data with access to individual devices. Therefore, it provides an opportunity to connect and leverage data as an ecosystem, as an experimental form of connecting devices towards the forthcoming IoE vision. With elaborated authorization management, device owners would be able to grant fine-grained access rights of resources to target users. Device owners might register date and designate access policies of their devices through user-friendly interfaces, so that their preferences on data exposure are followed.

To sum up, a global directory system has five responsibilities: (1) facilitate property-based resource discovery; (2) record grouped and virtual objects; (3) resolve names to individual devices and data sources; (4) foster data sharing; (5) facilitate inter-organizational services.

TABLE I
COMPARISON OF EXISTING DIRECTORIES. ● FULLY DESIGNED ○ PARTIALLY CONSIDERED

	Kafle et al. [5]	CoRE [7]	Jin et al. [8]	LDAP [13]	Sailhan et al. [14]	Hao et al. [15]	ThingsPage [16]	Thingweb [17]	GOLDIE
Designed for IoT	●	●	●		○	●	○	●	●
Easy to scale	●	●	●	●	●			●	●
Location-based queries	○	○				●			●
Aggregation query				●		○		○	●
Flexible updates	●	●			●	●	●	●	●
Global access		○	●	●	●		●	●	●
Support for federation				●					●

B. Design Principles

In the previous section, we described the responsibilities of a global IoT directory. An implementation has to meet

Easy to scale: New directory components, services, resources, or data, must be able to join the system without requiring manual updates elsewhere.

Flexible query functions: The global directory should support query types that reflect IoT application scenarios, such as queries based on geographic locations and shapes.

Dynamic content: Mobile IoT devices may update their location or other properties frequently, and may join and leave the directory.

Limited infrastructural changes: Existing network mechanisms and hardware should not change, i.e., the directory should provide added value, not impose constraints.

Globally accessible: Access to devices and services should be governed by policy, not by the limitations of the directory system.

A comparison among available directories and GOLDIE is shown in Table I. Aggregation queries refer to queries that ask for data that is computed across multiple sensors or across time, for example a current-time average or a time series. It might be provided by an edge computing service. Location-based queries may be described by a shape, typically a polygon with vertices described by longitude and latitude, and possibly altitude. In other cases, a civic address, such as "Room 520, 1154 Vine Street, North Chicago, IL" is a more convenient descriptor. Finally, the location may be described by location function, such as "kitchen" or "stairwell". Drawing on civic address standards for emergency calling, civic and functional locations can be uniquely represented by a set of hierarchical address facets and thus can be mapped into property-based query mechanism [18]. We will mainly discuss location-based queries relying on polygons.

In general, classic network directory designs based on LDAP, although they exhibit great read-optimized performance [13], [19], are not suitable for IoT applications since they do not support location-based queries and lack flexible update mechanisms. As the table shows, even directories designed for IoT fail to meet one or more requirements.

III. AN OVERVIEW OF GOLDIE

This section presents an overview of our system model and briefly introduces the various parts of the system, including the terminology, typologies, naming services, identity management processes, and our design philosophy.

A. Terminology

The federated global directory model is essentially a voluntary, bottom-up association of local directories. A set of local directories can be deployed on one or more servers that are maintained by the same organization, and then export data outside the local organization. To manage the names, things, users of its own entities, an organization should be able to easily structure its own local directories as an information tree.

Necessary terms and conceptions widely used in this paper and general directory systems are illuminated as follows.

Metadata Profile (MP): A metadata profile is a JSON object that describes the metadata of an IoT device. An MP is a digital profile of either a physical entity or a virtual entity, and the relationships between internal and external objects.

Physical entity: A physical entity is an object characterized by having a digital profile (i.e., something that identifies them in the digital environment, MP in this paper) for both the directory system and has a physical instance, such as a temperature sensor.

Virtual entity: A virtual entity is an object that has digital profile but does not have a physical sensor or actuator instantiation. Examples include a time series storage unit or data aggregator.

Sector: A sector manages a logical group of endpoints, where each endpoint hosts at least one local directory. A sector can be an organization, an institution, an administration, or any affiliation that owns and manages a number of IoT devices and metadata.

Resource owner: A resource owner maintains devices and their metadata. A resource owner can be either the owner of devices or the owner of a sector (administrators of the sector who operate the resources in the directories). In the directory system, we usually use the latter definition.

Local directory: A local directory is the minimum functional directory unit that performs basic create, retrieve, update, and delete operations. A local directory is physically located at a single computational node. There are two kinds of local directory: boarder directory and interior directory.

Border directory: A border directory is a local directory that serves as a publicly-accessible endpoint within a sector.

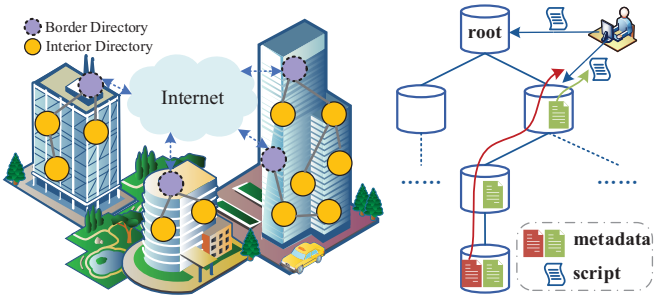
Interior directory: An interior directory is a local directory that is accessible only within a sector.

Directory Information Tree (DIT): A DIT is a tree-like structure that organizes a set of local directories. The DIT

is consulted to find a route to a designated entry.

Client: A client interacts with the directory server side by sending requests and receiving responses. A client can be part of a web browser, a mobile application or built into a server application.

B. Topological Structure



(a) A global view of federated directories. (b) A directory tree structure.

Fig. 1. Topological structure and system architecture of GOLDIE. Each sector maintains a couple of directory trees. Each directory tree consists of a bunch of local directories including one or more border directories.

The topology of our proposed directory system is a federation of tree-like structures. Unlike DNS, which grows top-down, our system grows bottom-up. As shown in Figure 1(a), there is at least one root node, a border directory, for each sector. Each node represents a local directory in charge of physical and virtual entities for a specific geographic area or administrative domain. In general, leaf directories are responsible for relatively smaller regions, and directories with shallower depths are responsible for larger regions. The location-based topology facilitates aggregation query and simplifies routing inside a topology tree. The number of target servers in the search space can be also decreased with the help of aggregated metadata when a certain type of IoT device ("outdoor temperature sensor") or a certain set of locations are requested. Besides, it paves the way to enhance complex access control mechanisms, since a user with permission to access one node might not be able to access the nodes of its subtrees.

This generic structure allows each sector to design its own tree-like typologies, storage principles, and access policies. For example, one can design a hierarchical tree where the metadata profiles can be mostly stored in leaf nodes, and the parent nodes store their aggregated data. As the size of the tree is usually not too large, the tree is not too deep as well. The sector by definition can possibly cover a huge geographic area, if the sector represents an enterprise with multiple branches or a large-scale service, e.g., the local directories of a metropolitan transit authority can be geographically distributed over a city or a state. Therefore, the local directories are not always close to each other.

In DNS, domain names are generally discoverable globally, with no formal provision for more limited visibility. Restricting visibility of names to within an enterprise requires careful DNS resolver design and is not supported by the protocol itself, other than through the `.local` domain. For IoT, restricting discoverability is crucial; GOLDIE supports visibility restrictions by geographic area and hierarchy, reflecting the preference of device or resource owners. Resource profiles may differ by scope, reflecting access restrictions or simply appropriate-use preferences. This feature is helpful when an MP is registered in a local directory, but should be visible to users across a wider geographic or administrative scope. For example, building-level data can be published to a campus directory, so that the data is visible to those access to the campus directory. As shown in Figure 1(b), a user registered two metadata profiles in the bottom local directory. The discoverability of the red one (on the left) is 0, which means that the metadata is stored locally and the access to the metadata is managed locally as well. The discoverability of the green one (on the right) is 2, which means that copies of the metadata are pushed to local directories that are within two levels above. Access management to the copied metadata also relies on those directories, and users who access to those directories would possibly access the copied metadata. The directory only hints at the availability of the data itself, and listing in the directory is not guaranteed to be congruent with actual data access, but well-designed directories will want to prevent unnecessary denied data queries to devices and services and thus align directory listings and access permissions.

Inside a sector, directory nodes may not have public IP addresses and are thus not reachable by external queriers. We manage local directories of each sector as a DIT. The border directory proxies external requests to interior directories.

C. Local Directory Architecture

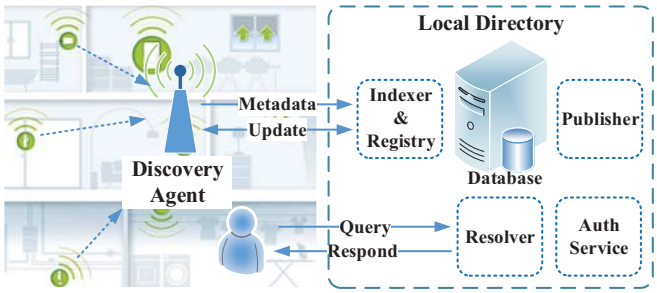


Fig. 2. Local directory architecture for name resolution. A local directory is a capable and functional unit that feasibly runs in edge servers.

Figure 2 illustrates the functional architecture of a local directory. It shows the various modules of each node in Figure 1(a). A local directory can run in an IoT hub, a dedicated server or as a service on a shared server. A discovery agent, running in an edge computational node or an IoT gateway, collects metadata useful for services from local IoT devices. The metadata can include, among other elements, device type, location, function, operational state, and access control information. The collected data is indexed by an indexer, stored in a database, and updated by the discovery agent. A query generated by a user or a remote IoT device is then resolved to the target metadata, with the help of the resolver. A local directory could authenticate users on its own or rely on the resource owner via techniques such as OAuth2 [20].

In principle, local discovery mechanisms adopted by each discovery agent should be orthogonal to directory services and left for the sectors to design. Well-studied active or passive data collection mechanisms [21]–[24] are likely applicable to propagate device metadata into directories. Additionally, each local directory can be configured by sectors to further synchronize data with other nodes or replicas via widely developed mechanisms, such as publish-subscribe, broadcast or multicast-based models [25]–[27].

Ideally, application services or local directories can be hosted on different servers for performance and robustness. They could also be sharded and replicated to improve scalability and availability by utilizing serverless computing platforms. Classic distributed system concerns (e.g., atomicity, consistency, isolation, durability) are of great importance, but beyond the scope of this paper.

D. Naming Service

One of the challenging problems for name-based directory services is managing potential name collisions and the rights to parts of the name space. DNS had to create a whole new administrative entity, ICANN, to handle potential conflicts with trademarks and to manage the entities that have exclusive control over sub-trees of the overall domain name space. Naming systems such as DNS also either require resource owners to purchase a name or become "tenants" of a domain, losing their identity if they decide to switch providers. Our IoT directory avoids these problems as devices are identified by self-deconflicting characteristics, such as geographic location, that reflect the cyber-physical nature of IoT devices. Thus, it can be seen as a more facet-based version of named data network [28], albeit with an out-of-band resolution mechanism.

GOLDIE provides the query-to-metadata resolution, where a query can include an identity, a location, a thing type, or a polygon. Directory entries represent objects in the name spaces, which contain arbitrarily many references to subordinate entries (device type, metadata, soft links, and pointers to other directories). Metadata profile entries in leaf nodes represent a terminal of the name space. Unifying the naming convention is not a focus of this paper. A naming convention is likely feasible, as long as each entity is identified in the directory information tree. In other words, X.500 standard [13] is compatible when naming a directory entry, e.g., /C=US/O=FOO/OU=PEOPLE/CN=ROBERSON.

In principle, naming and directory services can also support mobility [29]. Indeed, cellular voice networks have relied on home and visitor location registers to track mobile nodes from their earliest generations. But most internet-based directory services are not well-suited to propagate changes, relying instead on time-to-live (TTL) mechanisms and re-queries. By being insensitive to underlying device mobility, our interest-based query model provides an angle to partially address this problem, but a complete solution will be discussed in a separate paper.

E. Federated Identity Service

Most directory services do not consider authentication to the directory. If they need to restrict access to data, it is largely by hiding the server, e.g., from queries beyond the enterprise network. We believe that this is a mistake – directories should be able to return tailored responses that can reflect the identity or affiliation of the querier. Beyond the local domain, often only the affiliation of the querier is relevant, not its individual identity, such as "employee of a university", "located in New Jersey," or "paying user of service X". Thus, we integrate a single-sign on (SSO) mechanism into the directory system and avoid the need for users to create login credentials at a wide variety of directory services.

F. Design and Implementation Philosophy

Harmonization of information systems first attracted interest in the 1990s [30]. It becomes a general consensus that such convergence for global standards would translate into benefits for consumers, operators, and manufacturers [31]. During our efforts on realizing the global directory system, we are aware of the necessity of having IoT management systems be harmonized, as more vendor-specific or application-specific directories are problematic towards the forthcoming IoE vision. GOLDIE cannot overcome the balkanization of control protocols, but provides a common reference to devices regardless of the data layer protocols and could even include references to protocol gateways. The data models, communication protocols, and application program interfaces adopted by GOLDIE are all widely recognized and codified as standards.

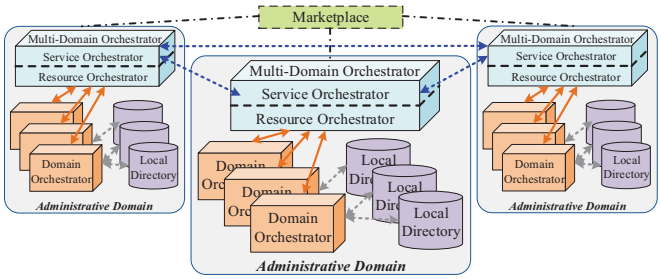


Fig. 3. A global directory system across multiple administrative domains from an understanding of network orchestration.

The process of assembling and developing an interorganizational network is referred as network orchestration. Orchestrations for IoT benefit creating more flexible services, reducing the probability of correlated failures between application components [32]. The multi-domain orchestration view of a global directory system is depicted in Figure 3. GOLDIE orchestrates a multi-domain system to discover resources beyond the boundaries of administrative domains. The exchange of messages and services are coordinated by a multi-domain orchestrator that exposes the correlated interfaces and data. Two essential components included in the multi-domain orchestrator are service orchestrator and resource orchestrator. The former carries high-level service orchestration, and the latter collects resources through local directories of the administrative domain, joint with underlying domain orchestrators

that expose metadata profiles stored in each local directory. When processing a request, local orchestrators have no idea about resources and topologies of other providers. In the proposed directory system, multi-domain orchestrators and domain orchestrators are applied to border directories and interior directories respectively.

IV. Prototype Functionality and Implementation

In this section, we dive into the details and implementation of our prototype[1]. We focus on major features of our prototype and introduce the points of implementation behind them. Then, the federated identity management by our system is covered and all supported services are discussed.

A. Application Program Interfaces

We implement a set of Application Program Interfaces (APIs) for requests processing and fulfillment. These primary services of directories are designed in terms of IoT-related features discussed in Section III. An overview of modules and functionalities is depicted in Figure 4.

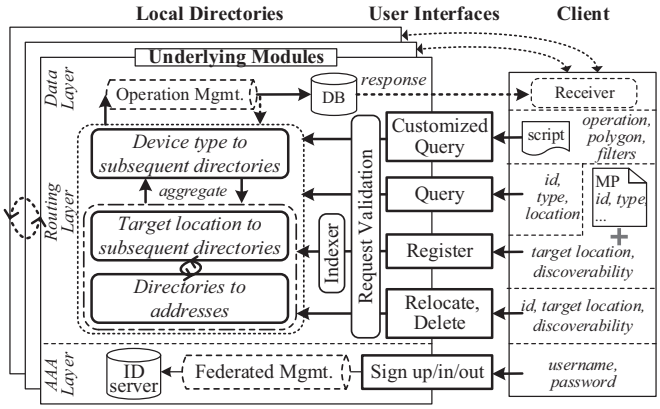


Fig. 4. An overview of modules and functionalities in GOLDIE. The underlying modules are divided into three layers: AAA (short for Account, Authentication, and Authorization) layer, routing layer, and data layer.

GOLDIE is implemented with the Python Flask framework to be deployable as a web application. We select MongoDB as the data store based on the observation in [15], which demonstrates that MongoDB is a favorable choice for storing metadata of IoT devices. The device type and spatial coordinates of metadata profiles are indexed by B-tree and 2dsphere. GOLDIE exposes RESTful APIs in support of the operations and federated authentication, which are listed below.

Registration adds an MP in the directory. The discoverability of MP can be specified while registering (by default as zero, i.e., local only).

Query searches for metadata by directory name, device ID, device type or any combination of them. The system responds with all visible MPs by default.

Customized query searches for metadata by customized JSON scripts. Users can customize aggregated operations, geographic polygons, and filters specified in MPs.

[1]https://github.com/Halleloya/GOLDIE

Relocation moves an MP from one local directory to another. It reflects the change of locations when devices move from one area to another.

Deletion removes an MP that was previously registered.

Authentication manages user identities and federated identities among directories, discussed in Section IV-C.

There is not always a clear distinction between metadata and produced data. The metadata we refer to in this paper is generally any concise, descriptive data that can be put into a short profile. We do not intend to replicate streaming databases to search time series data, for example. The MPs stored and exchanged through interfaces of GOLDIE are formatted as a W3C thing description [9], a working standard that allows to formally organize the metadata of physical or virtual IoT devices as JSON objects.

To give some examples of customized script query, one can write a script to request: the number of public transit buses inside a polygon (action: SUM, type: bus, filter: inside a polygon); the number of lights that are on (action: SUM, type: lights, filter: status=on); the temperature in NYC (action: AVG, type: thermometer; location=NYC).

B. Aggregated and Distributed Knowledge

Data privacy is naturally a concern of device owners and resource owners. Sometimes metadata can be considered sensitive and the owners only want to expose it to certain selected groups of users. In addition to reflecting the exposure preference as discoverability when registering, aggregated data provides an additional safeguard. Each directory should not store specific MPs of its lower-level directories, unless those are explicitly permitted by setting discoverability.

In GOLDIE, each local directory maintains a mapping from device types to child directories, a notion that which subsequent directories should further reach, rather than the detailed knowledge of MPs. Thus, each local directory is only aware of whether or not subsequent directories have such type of MPs, without any details. The query model, specifically for query by device type, relies on such aggregated knowledge to traverse local directories. To achieve this, a type information is aggregated to above directories along with a successful registration of a new type, and a notice to delete a type is passed up if none of this type left in the directory. The internal aggregated information leads to the efficiency of external aggregation query.

Each single directory preserves only distributed knowledge of location and address information. For each directory, there is a mapping from directory names to addresses that is configured in advance. It only maintains the names and addresses of root, parent, and direct children directories. Another mapping from target directory names to subsequent directories provides a knowledge on which subsequent directory leads to the target directory. In other words, the local directory again only knows which child directories should consult, rather than where is the target directory.

As shown in Figure 4, those three mappings constitute the routing module inside a local directory. Any queries specified

with a target directory, if not reachable from the responsive directory by consulting the mappings, are forwarded to the root directory for further routing and process, as any valid directories are reachable from the root node.

To sum up, there are three types of knowledge for routing of each local directory: (1) addresses of its directly reachable nodes; (2) which children nodes should reach, if a query targets a deeper directory; (3) which children nodes should reach, if a query asks for a certain type of things. The first two types of knowledge are configured in advance as current implementation. The third type of knowledge is automatically refreshed when a new MP is being registered or deleted.

C. Federated Authentication

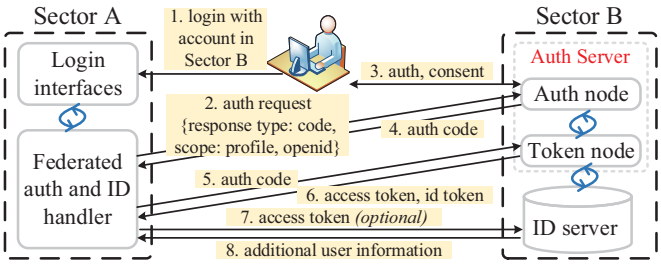


Fig. 5. Federated identity management with OIDC.

GOLDIE achieves federated user authentication by building on OpenID Connect (OIDC) [10], the de facto standard in the OpenID family. The process is depicted in Figure 5, where *sector A* relies on *sector B* for authentication of the user. It would commonly happen when a user doesn't have a valid credential to a directory. The trust relationship between *sector A* and *sector B* is established in advance, and an interface is exposed to allow users to login with listed third-party identity providers. In this example, the user has a valid account with *sector B*. Then to complete the login process to *sector A*, it issues an authentication request to the *authentication node* of *sector B* that authenticates the user and get the consent to provide relevant information. Whereafter, an authorization code is returned to the *sector A*, which is then forwarded to the *token endpoint* of *sector B* to exchange an ID token and an access token. ID token contains the basic information of the user, while additional user information can be optionally retrieved from *ID server* of *sector B* using the access token.

V. Performance Evaluation

This section presents the performance evaluation for our prototype system. We first introduce the evaluation scheme and testbed configurations. Then we run the evaluation and analyze the results based on standard metadata and a real-world dataset.

A. Factors Affecting the Performance

In the following evaluations, we measure response time as the critical indicator of the system performance, which is governed by several factors [33]. Since local directories are arranged as a tree structure, a query might go through several directories until the required information is reached. Besides, not all the users and target directories are at the same network, since a global directory is intended for multiple administrative sectors. Thus, the distribution of local directories as well as the propagation delays among directories and users both count.

There are some other factors that impact the response time, considering the implementation of directories. For instance, with more aggregation knowledge retained into a directory, fewer directories need to be consulted. Moreover, network conditions also make a difference, when queries are processed with bandwidth or traffic load ranges.

B. Configurations and Settings

We implement a proof-of-concept system and investigate the performance upon it. A sketch of the system is shown in Figure 6, which consists of a switch, a Raspberry Pi and two Intel NUCs. Intel NUC (short for Next Unit of Computing) is a mini server designed for small space and low power. It is well-suited for deploying directories in IoT and edge computing scenarios. The Raspberry Pi represents common IoT devices that perform operations on the client side. Table II shows configurations of the utilized hardware.

The directory deployed as a five-level binary tree topology for our testing goal. Directories in odd levels are deployed in one NUC server and directories in even levels are deployed in the other. In this way, we ensure that the traverse between any two directories goes through the network interface cards of NUCs and the switch. The parameters of network interfaces can be therefore adjusted by traffic control commands of the Linux systems. By default, we set the link delay between two directories to 20 ms, and the link delay between client and server to 30 ms. Each test of our evaluation is performed 100 times, and all the presented data are averages.

The mock metadata profiles as the source of entries are generated and validated based on the W3C working standard [9]. The generated data are exploited to evaluate the performance of functions and interfaces. To further analyze the performance enhancement under specific circumstances, we leverage a real-world dataset, Lysis dataset [34], which collects more than 11,000 queries of IoT applications over seven months in 2017.

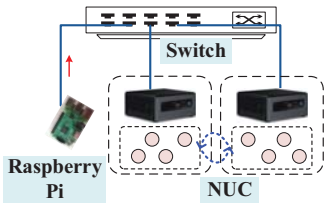


Fig. 6. Proof-of-concept system.

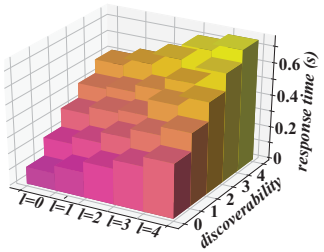


Fig. 7. Performance of registration.

C. Evaluation of Directory Operations

We evaluate the performance of various operations: registration, query, customized query, relocation, and deletion. For space constraints, we exclude the evaluation of relocation, as it is essentially a deletion followed by a registration. Given that an operation could act at not only the responsive local

TABLE II
CONFIGURATIONS OF THE TESTBED

Category	Specification
CPU (Raspberry Pi)	ARMv7 Processor, 4 cores, 1.2GHz
System (Raspberry Pi)	Raspbian GNU/Linux 9
Storage (Raspberry Pi)	1GB RAM, 32GB microSD
CPU (NUCs)	Intel CORE i3-8109U, 2 cores, 3.0GHz
System (NUCs)	Ubuntu 18.04.2 LTS
Storage (NUCs)	16GB RAM, 256GB SSD
Database	MongoDB 3.6.3 [35]

directory, but any directories as well, we denote the number of levels that an operation recursively traverse as l. If a query only operates at the responsive directory it accesses, then $l = 0$.

Figure 7 shows the evaluation of registering an MP. A registration takes two parameters along with the MP: a local directory name and an integer of discoverability. The response time is generally in direct proportion to both l and discoverability. As l increases, it takes more time to locate the target directory. It also takes more time to aggregate information and push up MPs, while the discoverability becomes higher.

In Figure 8, we measures three basic query functions: query by id, query by device type, and query by default (to show all abstract information of MPs in this location). There are four types of things in our generated MPs for evaluation: transit bus, thermometer, light, and thing (a default type if a thing is not categorized to a particular type). We store 1,000 MPs of each type beforehand. Then we perform our tests for queries. Query by identity takes less time and the increase with l going up is relatively moderate compared with others. The result makes sense, since the data volume transmitted for query by identity is much less then others.

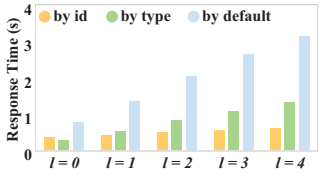


Fig. 8. Performance of generic queries with three basic functions.

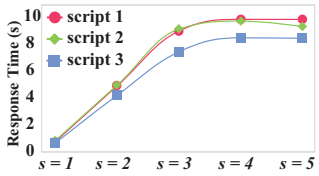


Fig. 9. Performance of customized query with three scripts.

For customized query, there are actually a lots of options one can customize through JSON scripts. Here we demonstrate three kinds of query scripts for the generated MPs: (1) count the number of transit buses running inside a polygon; (2) fetch the temperature of this sector; (3) count the number of lights that are left on. For the first script, the directory essentially performs polygon queries to determine if the coordinates of each bus is inside the input polygon. For the second script, it collects all the displayed data of thermometers and takes the average of them. For the third one, it filters the status of lights that are on. It seems less complicated than the other two, since less time is needed compared to polygon query and a half of MPs are filtered out by the status condition.

The results are shown in Figure 9, where script 1, 2, and 3 correspond to the three scripts respectively that are illustrated above. Since such queries require to process at all possible local directories, we denote how scattered the MPs are located as s. From $s = 1$ to $s = 5$, the MPs are located evenly from bottom level only to all five levels of directories. As s increases, more directories are needed to process. The curve becomes flat, since more processing happens in directories that are close to the responsive directory (i.e., root directory in this evaluation). Nearly identical results are achieved when $s = 4$ and $s = 5$, because an aggregation happens at the root directory no matter if there is any MPs residing or not.

Figure 10 shows the evaluation of deleting an MP with respect to l and discoverability. When deleting a thing, the system checks the discoverability of the associated MP. If the discoverability is greater than zero, the copies in above levels also need to be erased. Meanwhile, local directories needs to monitor and update aggregated information about whether the successive directories still hold these types of things. Hence, deleting an MP with higher l or discoverability requires more efforts.

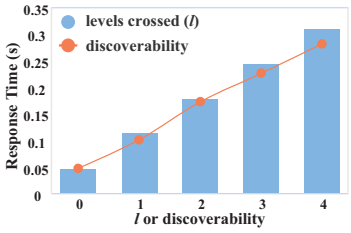


Fig. 10. Performance of deletion.

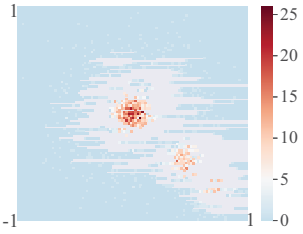


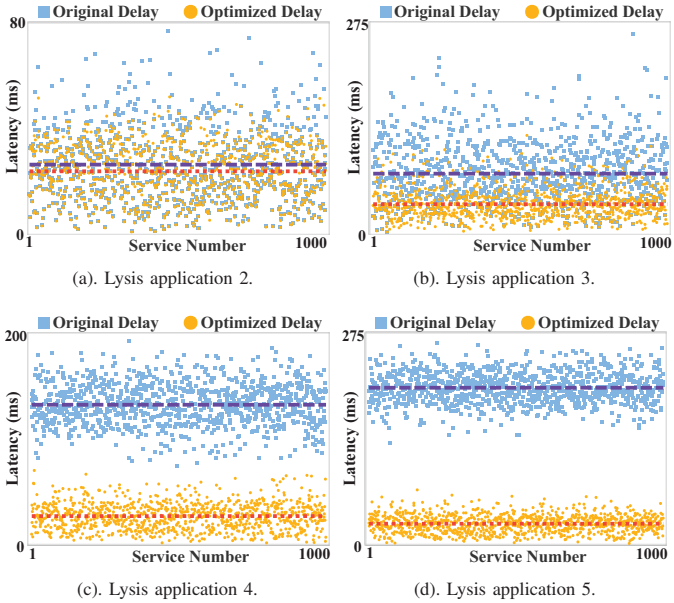
Fig. 11. Distribution of queries.

D. Deploy Directories near End Users

For a city-scale or even larger sector, local directories are ideally deployed near end users. Distributed local directories, to be deployed on edge computational nodes, provide an opportunity to push services near users and therefore reduce service latency. In our model, a local directory responds to queries without involvement of other directories, if the coming request is resolvable by the responsive directory only. If this is the case, a local directory reduces the search space as well as propagation delay. Although deployment issue is not a primary focus of this paper, we run simulations to briefly explore the potentials of local directories that cache remote data to reach end users.

Here we simulate the deployment problem utilizing the Lysis dataset [34]. The geographic distribution of queries is analyzed in Figure 11, which normalizes the distance between client and server to $[-1, 1]$. It shows that queries are geographically distributed but clustered at some areas.

In the simulation, we evenly deploy local directories as a four-level quadtree and compare it to the original solution of a single centralized directory. For the four applications with the needed distance information (application 2–5 in the dataset), we simulate and generate the service access latency. From Figure 12(a) to Figure 12(d), the straightforward deployment turns out to be more and more feasible, reflected by if it pushes more data near end users through local directories. Due to space constraints, we show the first 1,000 queries in time sequence for each application in Figure 12, regardless of how many entries in total. The optimal condition is given by the assumption that all the queries are processed at the



(a). Lysis application 2. (b). Lysis application 3.

(c). Lysis application 4. (d). Lysis application 5.

Fig. 12. Potentials of caching remote data to reach end users. The horizontal purple dashed line and red dotted line present the average latency of all entries for each application in the dataset.

nearest local directory. Compared to the centralized solution of a single directory, the average service latency is reduced by 9.7%, 51.4%, 79.9%, and 86.9%, respectively, with the location-based directory solution.

VI. Discussions

Authorization: Although authorization problem is beyond the scope of this paper, it plays a critical role in a practical directory system. Naturally, the federated system relies on each resource owner (or administration of each local directory) to manage access rights and determine their preferred access control mechanisms. However, if the directory wants to achieve desired fine-grained access control on exposing what data to whom, traditional Access Control List (ACL) and Role-Based Access Control (RBAC) both lack flexibility. From our perspective, two finer-grained schemes, Attribute-Based Access Control (ABAC) and Capability-Based Access Control (CapBAC) [36], [37] are potential solutions.

Search process: In general, there are two types of search models, iterative search and recursive search, for a tree-like topology. We are not particular about which one to choose, as it might touch more on cache design or invoke additional privacy concerns. In spite of those, we also measure performance difference between the recursion and iteration implementations of query mechanisms in Figure 13. Initially, servers would likely be distant to clients as they reside in cloud computing centers, which makes client-server communications more time-consuming than inter-server communications. It makes sense given that our result seems to favor recursive search model a bit more. It is not always the case, as directories might be pushed near end users in the upcoming edge computing era.

Scalability: Scalability can be a reasonable concern for many globally accessible systems. We would like to identify that the proposed system, although each DIT is still a single

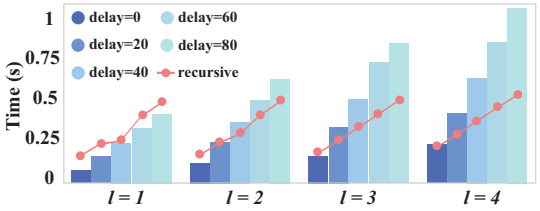


Fig. 13. Comparison of recursive and iterative search model. Link delay between client and responsive directory is set at intervals of 20ms.

tree structure, works as a forest with multiple endpoints. Each local tree owned by a single sector essentially runs like a web application, and thus the scalability challenges are not beyond those discussed in [14], [36], [38], [39]. Besides, as each DIT is organized through a bottom-up approach, and each local directory only maintain distributed knowledge, adding or deleting nodes incurs moderate changes. It does not hinder sharding or replica mechanisms. Therefore, the proposed framework unlikely raises huge scalability challenges.

Location-based queries: In Section II, we propose three kinds of location-based directory queries. GOLDIE achieves query by geographic polygon through database extensions and query by location property through filters of related metadata fields. For query by civic address, an applicable solution is to let the directory consult a trusted party that provides a civic address to polygon mapping. An alternative but more challenging idea is to provide a location to service mapping [18], which forwards queries to a nearby directory for further civic address processing. To meet more location-based needs is a future work of GOLDIE.

VII. Conclusion

In this paper, we design and implement GOLDIE, a federated global directory architecture for IoT and IoE. GOLDIE meets a series of proposed design principles that reflect the needs of IoT systems. Aware of harmonization and orchestration, it implements generic operations including registration, query, relocation, and deletion. GOLDIE implements essential features such as discoverability, aggregated information, location-based query, and customized search. In addition, we include a federated authentication mechanism using OIDC. Extensive evaluations on the system deployed on Intel NUC servers, against generated standard metadata and Lysis dataset, validate the performance of various operations and show the merits of the system compared to a centralized-server solution.

In the future, we may investigate access control mechanisms and directory pollution problems for the proposed system.

Acknowledgement

This work is supported by the National Science Foundation under grant CNS 19-32418. The authors would like to acknowledge Yifan Yang for his contribution on the initial prototype implementation. The authors would also like to thank Jan Janak, Krishan Sabnani, Artiom Baloian, and Pekka Karhula for helpful discussions.

REFERENCES

[1] "The Internet of Everything," https://www.cisco.com/c/dam/global/en_my/assets/ciscoinnovate/pdfs/IoE.pdf.

[2] R. E. Droms, "Access to heterogeneous directory services," in *IEEE Conference on Computer Communications*, 1990.

[3] D. Comer and R. E. Droms, "Uniform access to internet directory services," *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 4, pp. 50–59, 1990.

[4] R. J. Clark, K. L. Calvert, and M. H. Ammar, "On the use of directory services to support multiprotocol interoperability," in *IEEE Conference on Computer Communications*, 1994, pp. 784–791.

[5] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and H. Harai, "Directory service for mobile IoT applications," in *IEEE Conference on Computer Communications Workshops*, 2017, pp. 24–29.

[6] C. Badii, P. Bellini, A. Difino, and P. Nesi, "Privacy and security aspects on a smart city IoT platform," in *IEEE International Conference on Advanced and Trusted Computing*, 2019, pp. 19–23.

[7] Z. Shelby, M. Koster, C. Bormann, and P. van der Stok, "Core resource directory," IETF, Internet-Draft 6749, 2020. [Online]. Available: https://www.ietf.org/id/draft-ietf-core-resource-directory-24.html

[8] W. Jin and D. Kim, "Improved resource directory based on DNS name self-registration for device transparent access in heterogeneous IoT networks," *IEEE Access*, vol. 7, pp. 112 859–112 869, 2019.

[9] "W3C web of things (WoT) thing description," https://www.w3.org/TR/wot-thing-description/, 2020.

[10] D. N. Sakimura, J. Bradley, and M. Jones, "Openid connect standard 1.0-draft 21," 2012.

[11] H. Cai and T. Wolf, "Self-adapting quorum-based neighbor discovery in wireless sensor networks," in *IEEE Conference on Computer Communications*, 2018, pp. 324–332.

[12] L. Hao, C. Jin, X. Gao, L. Kong, F. Wu, and G. Chen, "QoE-aware optimization for SVC-based adaptive streaming in D2D communications," in *IEEE International Performance Computing and Communications Conference*, 2017, pp. 1–8.

[13] W. Yeong, T. Howes, and S. Kille, "X.500 Lightweight Directory Access Protocol," IETF, RFC 1487, 1993. [Online]. Available: http://tools.ietf.org/rfc/rfc1487.txt

[14] F. Sailhan and V. Issarny, "Scalable service discovery for MANET," in *IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 235–244.

[15] L. Hao and H. Schulzrinne, "When directory design meets data explosion: Rethinking query performance for IoT," in *IEEE International Symposium on Networks, Computers and Communications*, 2020, pp. 1–6.

[16] "Thingspage," https://www.thingspage.com/.

[17] "Thingweb," http://www.thingweb.io/.

[18] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig, "LoST: A Location-to-Service Translation Protocol," IETF, RFC 5222, Aug. 2008. [Online]. Available: http://tools.ietf.org/rfc/rfc5222.txt

[19] X. Wang, H. Schulzrinne, D. Kandlur, and D. Verma, "Measurement and analysis of LDAP performance," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 232–243, 2008.

[20] D. Hardt, "The OAuth 2.0 Authorization Framework," IETF, RFC 6749, 2012. [Online]. Available: http://tools.ietf.org/rfc/rfc6749.txt

[21] K. Lynn, S. Cheshire, M. Blanchet, and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions," IETF, RFC 7558, Jul. 2015. [Online]. Available: http://tools.ietf.org/rfc/rfc7558.txt

[22] M. Stolikj, R. Verhoeven, P. J. Cuijpers, and J. J. Lukkien, "Proxy support for service discovery using mdns/dns-sd in low power networks," in *IEEE international symposium on a world of wireless, mobile and multimedia networks*, 2014, pp. 1–6.

[23] V. Daza, R. Di Pietro, I. Klimek, and M. Signorini, "Connect: Contextual name discovery for blockchain-based services in the IoT," in *IEEE International Conference on Communications*, 2017, pp. 1–6.

[24] M.-O. Pahl and S. Liebald, "A modular distributed IoT service discovery," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management*, 2019, pp. 448–454.

[25] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, "Dynamic policies in Internet of Things: enforcement and synchronization," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2228–2238, 2017.

[26] X. Gao, A. Song, L. Hao, J. Zou, G. Chen, and S. Tang, "Towards efficient multi-channel data broadcast for multimedia streams," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2370–2383, 2019.

[27] A. Ros and S. Kaxiras, "Callback: Efficient synchronization without invalidation with a directory just for spin-waiting," in *ACM/IEEE International Symposium on Computer Architecture*, 2015, pp. 427–438.

[28] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, p. 66–73, 2014.

[29] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 247–258, 2014.

[30] S. Braman, "Harmonization of systems: The third stage of the information society," *Journal of Communication*, vol. 43, no. 3, pp. 133–140, 1993.

[31] M. Zeng and A. Annamalai, "Harmonization of global third generation mobile systems," *IEEE Communications Magazine*, vol. 38, no. 12, pp. 94–104, 2000.

[32] N. F. Saraiva de Sousa, D. A. Lachos Perez, R. V. Rosa, M. A. S. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Elsevier Computer Communications*, vol. 142, pp. 69–94, 2019.

[33] P. Barker, "Providing the X.500 directory user with QoS information," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 3, pp. 28–37, 1994.

[34] C. Marche, L. Atzori, V. Pilloni, and M. Nitti, "How to exploit the social Internet of Things: Query generation model and device profiles' dataset," *Computer Networks*, p. 107248, 2020.

[35] "Mongodb," https://www.mongodb.com/.

[36] Q. Zhou, M. Elbadry, F. Ye, and Y. Yang, "Heracles: Scalable, fine-grained access control for Internet-of-Things in enterprise environments," in *IEEE Conference on Computer Communications*, 2018, pp. 1772–1780.

[37] S. Gusmeroli, S. Piccione, and D. Rotondi, "A capability-based security approach to manage access control in the Internet of Things," *Elsevier Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1189–1205, 2013.

[38] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *IEEE/ACM International Symposium on Cluster*, 2013.

[39] S. Taherizadeh and V. Stankovski, "Auto-scaling applications in edge computing: Taxonomy and challenges," in *ACM International Conference on Big Data and Internet of Thing*, 2017.