Application of Parallel and Distributed Systems
Institute for Parallel and Distributed Systems
University of Stuttgart

Enhancing IoT Platforms for Autonomous Device Discovery and Selection

Jan Schneider, Pascal Hirmer

In: Service-Oriented Computing. 16th Symposium and Summer School (SummerSOC). Communications in Computer and Information Science, vol. 1603, pp. 24 – 44, Springer International Publishing, 2022.

BIBTEX:

```
@inproceedings{Schneider2022,
    author={Schneider, Jan and Hirmer, Pascal},
    title={Enhancing IoT Platforms for Autonomous Device Discovery and Selection},
    booktitle={Service-Oriented Computing. 16th Symposium and Summer School (SummerSOC).
    Communications in Computer and Information Science},
    volume={1603},
    year={2022},
    pages={24-44},
    publisher={Springer International Publishing},
    doi={10.1007/978-3-031-18304-1_2},
    isbn={978-3-031-18304-1}
}
```

Enhancing IoT Platforms for Autonomous Device Discovery and Selection

Jan Schneider and Pascal Hirmer

Institute for Parallel and Distributed Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{forename}.{surname}@ipvs.uni-stuttgart.de

Abstract. The Internet of Things (IoT) encompasses a variety of technologies that enable the formation of adaptive and flexible networks from heterogeneous devices. Along with the rising number of applications, the amount of devices within IoT ecosystems is constantly increasing. In order to cope with this inherent complexity and to enable efficient administration and orchestration of devices, IoT platforms have emerged in recent years. While many IoT platforms empower users to define application logic for use cases and execute it within an ecosystem, they typically rely on static device references, leading to huge manual maintenance efforts and low robustness. In this paper, we present an approach that allows IoT platforms to autonomously and reliably execute pre-defined use cases by automatically discovering and selecting the most suitable devices. It establishes loose coupling and hence does not impose major technical constraints on the ecosystems in which it is operated.

Keywords: Internet of things · IoT platforms · Device discovery

1 Introduction

In the world of tomorrow, digital ecosystems will be able to autonomously create synergies between so-called smart devices and let them jointly perform tasks for the benefit of the people in their environment. This way, a smart home may autonomously detect the presence of an internet-enabled heating system and a smart phone and use them to control the room temperature depending on the current location of its owner. Turning such scenarios into reality is one of the visions of the Internet of Things (IoT) [51]. It encompasses various technologies that enable the interconnection of virtual or physical internet-enabled devices ("things") and to constitute advanced services from them [24]. These devices typically comprise different capabilities and interfaces and are equipped with sensors and/or actuators, providing them the ability to perceive their environment and to interact with it [20]. Typical examples of IoT devices include micro-controllers, smart phones and other objects of the everyday life, as long as they possess basic computing and communication capabilities [35] that allow them to exchange data within a shared IP network.

Along with the rapidly growing number of world-wide available IoT devices [50] and their increasing affordability, also the individual IoT ecosystems

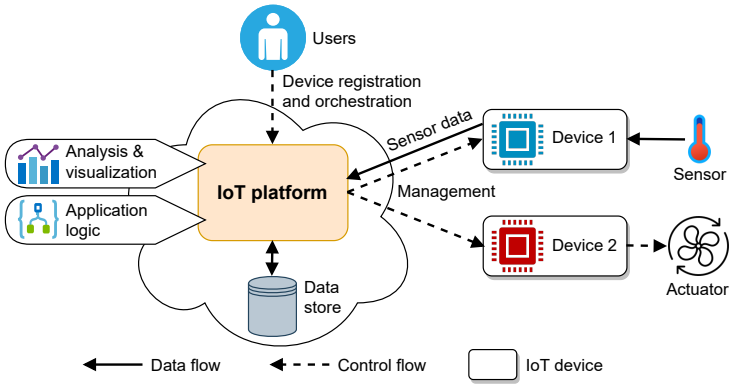

Fig. 1. Exemplary IoT ecoystem with integrated IoT platform and two devices.

become more and more complex: They can now comprise several thousands of heterogeneous devices [44], reaching from low-end micro-controllers to powerful backend services in the cloud. The high dynamics of IoT ecosystems, to which new devices may be added and from which existing devices may be removed at runtime, impose further challenges [11]. In order to cope with the resulting complexity, so-called IoT platforms [22] have emerged in recent years, offering tools for the administration and orchestration of IoT devices. As depicted in Fig. 1, they may also provide means for the definition of application logic, enabling the implementation of custom use cases within an ecosystem. However, these IoT platforms typically require their users[1] *a*) to manually select suitable devices for realizing the desired use cases and *b*) to register them at the IoT platform using IP addresses or other static references. This leads to huge maintenance efforts, since the employed devices may become less preferable or even unavailable during runtime and hence demand human intervention for reconfiguring and adjusting the IoT platform accordingly.

To address this shortcoming and progress towards the previously described vision of self-controlled and autonomously operating IoT ecosystems, dynamic approaches for the automatic discovery of devices can be employed [11]. Device and service discovery are broad and well-established fields in literature, suggesting hundreds of different methods for the discovery of resources within networks [41]. However, they typically exploit very particular protocols and formats and hence can only be applied to ecosystems that support exactly these technologies. Due the high heterogeneity of the IoT and its devices, IoT platforms implementing such specialised approaches become severely limited in their scope of application and thus more difficult to integrate into existing ecosystems. This contradicts common design goals of IoT platforms, which are typically not

[1] For the scope of this paper, we differentiate between a) non-expert users, who want to implement their use cases within IoT ecosystems by leveraging an IoT platform and b) administrators, who are responsible for maintaining the technical setup.

tailored towards individual ecosystems, but rather intend to represent universal solutions for a broad variety of application scenarios [8].

To overcome these issues, we propose *a*) a method allowing IoT platforms to autonomously and reliably execute pre-defined use cases within IoT ecosystems by automatically discovering and selecting the most suitable devices and *b*) a supporting architecture, which introduces an additional abstraction layer between the IoT platform and the ecosystem. As a result, loose coupling is achieved, which avoids that the IoT platform imposes major technological constraints on the ecosystems in which it is operated.

2 Related work and literature review

We conducted a comprehensive literature research in which we investigated many papers proposing various approaches for discovery within IoT ecosystems.

2.1 Method

Most relevant work in literature is available under the keyword *IoT discovery*. We searched for papers by using the online search engines of *Google Scholar*, *The Collection of Computer Science Bibliographies*, *ACM Digital Library*, *IEEE Xplore Digital Library* and *dblp* and refined our queries by adding the terms *device*, *sensor*, *platform*, *selection*, *query*, *directory* and *ranking* both individually and in various combinations. In addition, the references of the publications were followed up recursively. We selected the papers on the basis of *a*) their assumed relation to our work, *b*) their recency, *c*) their reputation within the scientific community, but *d*) also in a manner to cover a broad thematic and technological spectrum to do justice to the variety of concepts in this field. Publications that are not directly related to the IoT or do not address local, self-contained IoT ecosystems, are out of the scope of our work due the different problem dimensions and were thus only marginally considered. The same applies to proposals that are fundamentally unsuited for integration in cloud-operated IoT platforms, including solutions specifically designed for proximity-based technologies.

2.2 Literature overview

Table 1 provides an detailed overview about the different reviewed discovery approaches and their characteristics that we were able to identify. These concepts can generally be divided into three overlapping categories, comprising *a*) proposals based on the paradigm of Service-oriented computing (SoC) [37], which pursue to apply holistic concepts, technologies and standards of Service-oriented Architecture (SoA) [30] to the IoT, *b*) suggestions for centralized solutions, in which one or multiple central repositories are used to store and query formal descriptions of devices and *c*) decentralized approaches, where the IoT devices directly communicate with each other in order to accomplish discovery, comparable to searches in graph data structures. Accordingly, the second column of Table 1

states if and how many central repositories are used within the approaches, while the third and fourth column indicate whether the devices decentrally interact with each other and whether they are based on SoA, respectively.

Some of the suggested solutions are able to discover new IoT devices joining the ecosystem automatically, while others require administrators to manually deposit device descriptions in corresponding repositories. Still others combine both approaches into one system. Therefore, the sixth column of Table 1 lists the entity which is responsible for initiating the first contact when a new device joins the ecosystem. In most cases, this is the device itself, i.e. it has to actively declare its presence against other components. The subsequent three columns indicate whether the approaches support *a*) keyword-based (free-text) queries for finding devices that are suitable for a certain use case, *b*) criteria-based queries which allow to specify mandatory capabilities for devices or *c*) automatic ranking of the search results according to the relevance of the devices to the query. As pointed out by Gomes et al. [19], the supported types of queries can be further classified as either synchronous or asynchronous. While synchronous queries are processed immediately by the discovery service and answered with a list of device descriptions matching the query, asynchronous requests allow other components to register subscriptions at the discovery service and be notified as soon as a change occurs in the result set of the issued query. Thus, asynchronous requests are particularly useful for IoT ecosystems that need to adapt themselves to changes at runtime, such as when a new device is added to the ecosystem or when an existing device is removed from it. In Table 1, the tenth column indicates the support for both types of queries. Finally, the last two columns provide information about *a*) the technology stacks on which the proposed solutions rely for the discovery of devices and the processing of queries, as well as *b*) the description languages that are used for formally modelling the IoT devices.

Two of the approaches listed in Table 1 are particularly related to our work: Papp et al. [38] propose a protocol which pursues to achieve interoperability among heterogeneous devices within a common ecosystem. For this purpose, it provides means for device abstraction and establishes an overlay on top of the Message Queuing Telemetry Transport (MQTT) protocol, which enforces a prescribed interaction model and, among other features, affords access to device capabilities. According to the protocol, the network and its devices have to follow a star topology with a so-called *IoT hub* in the center. Since this hub acts as a central controller and may implement application logic for the ecosystem, it can be considered as a kind of IoT platform. The concept includes a mechanism for the automatic detection of new devices joining the network based on Simple Service Discovery Protocol (SSDP), as well as a HTTP interface for manual registration. However, since the protocol relies on MQTT and prescribes a certain topic structure as well as message orders and formats, it can only be used with devices that actively support the particular protocol. As a result, IoT platforms that solely rely on this or another discovery approach from Table 1 inevitably dictate their communication technologies to the entire ecosystem, which renders them unusable for all other scenarios that are based on different technology

stacks. In our work, we propose an abstraction layer between the IoT platform and the ecosystem in order to overcome this issue.

Gomes et al. [18] suggest a centralized approach for a discovery service that involves multiple federated repositories. By using a web interface, administrators are able to insert semantic descriptions of devices into the available repositories and then register the latter at the service. Subsequently, applications may use to service to retrieve descriptions of devices showing certain characteristics by issuing criteria-based queries. To process such a query, the discovery service creates a corresponding request and forwards it to the endpoints of all registered repositories. Upon completion, the service consolidates their responses into a common result set, which is then returned to the application. Both synchronous and asynchronous queries are supported. Due to its centralized architecture with federated repositories, the proposed discovery service is suitable for integration into IoT platforms. Furthermore, the asynchronous queries allow the IoT platform to be notified when previously used devices become unavailable or when new, potentially more suitable devices, join the ecosystem. However, there are still shortcomings: a) The discovery service does not evaluate and rank the search results obtained from the different repositories and instead returns an unsorted result set, from which suitable devices must be selected manually, b) the endpoints of all repositories need to be manually registered at the discovery service, leading to additional maintenance efforts, c) administrators are required to manually manage the device descriptions within the repositories, as these basically act as databases, d) as no concepts for the handling of duplicates in search results are mentioned, it has to be assumed that redundant storage of device descriptions in multiple repositories is not allowed, which possibly leads to a loss of information in case of failures and e) working with semantic descriptions of devices may overwhelm users of IoT platforms who often lack sufficient technical expertise [8]. We adapt some of the concepts of Gomes et al. for the scope of IoT platforms and enhance them in order to overcome the mentioned issues and enable the autonomous execution of use cases within IoT ecosystems.

Achieving interoperability between different IoT resources is also a key objective of the Web of Things (WoT) [29], which pursues to transfer concepts of the web to the IoT. With WoT Discovery [6], it includes a flexible discovery approach, in which directories or the devices itself provide so-called *Thing Descriptions* in response to synchronous or asynchronous queries. However, the specification is currently still in draft state and thus subject to frequent changes. Despite this, some of the underlying web technologies may not be available or unreliable in certain scenarios due to resource restrictions, while concepts for failure-tolerant storage of device descriptions are not available yet. Furthermore, it is left open which entities are responsible for inserting device descriptions into the directories, as well as how based on the descriptions the most suitable devices can be selected for the use cases at hand. We address these open aspects for the scope of IoT platforms by proposing a corresponding architecture and method.

In prior work [36, 15], we investigated the modeling of complex context information that was managed in a world-wide scalable infrastructure through hier-

archical registers and extensible ontologies. In contrast, the concepts proposed in this paper provide a lightweight approach with reduced expressiveness to cater the specific requirements of IoT platforms and resource-limited ecosystems.

Table 1. Overview about the reviewed literature, sorted by year of publication. CR: *Reliance on central repositories;* D: *Decentralized discovery between devices possible;* S: *Exploitation of SOA-related concepts and technologies;* M: *Usage of semantic technologies;* IN: *Entity initiating first contact;* K: *Support for keyword-based queries;* C: *Support for criteria-based queries;* R: *Ranking of query results;* Q: *Query type, either synchronous (s), asynchronous (as) or both (b)*

	CR	D	S	M	IN	K	C	R	Q	Technologies	Device Description
[42]	single	✗	✗	✓	unspec.	✓	✗	✗	s	REST	SensorML
[52]	single	✗	✓	✗	device	✗	✓	✓	s	SOA stack	WSDL, QoS readings
[2]	single	✗	✗	✗	device	✗	✗	✗	-	CoAP, CoRE-RD	comm. behaviour
[17]	federated	✗	✗	✓	device	✓	✓	✗	s	DNS-SD	DNS names, TXT-RRs
[32]	per ecosys.	✓	✗	✗	device	✗	✓	✗	s	CoAP, CoRE-RD, P2P	(range) attributes
[12]	none	✓	✗	✗	device/P2P	✗	✓	✓	s	unstructured P2P	QoS attributes
[38]	single	✗	✗	✗	device or IoT hub	✗	✗	✗	-	SSDP/UPNP, HTTP, MQTT	JSON
[34]	single	✗	✓	✓	user	✓	✗	✓	s	REST	JSON-LD
[4]	per gateway	✗	✗	✗	device	✓	✗	✗	s	DNS-SD, mDNS, CoAP, 6LoWPAN	resource records
[6]	arbitrary	✓	✗	✓	device	✓	✓	✗	b	DNS-SD, mDNS, CoAP, server-sent events	WoT Thing Description
[10]	Blockchain	✗	✗	✓	provider	✗	✓	✗	s	SPARQL	SSN/SOSA extension
[18]	federated	✗	✗	✓	user	✗	✓	✗	b	MQTT, SPARQL	SSN, SAN, OWL-S
[43]	none	✗	✗	✗	scanner	✗	✗	✗	-	network/port scanner	open network ports
[40]	per sniffer	✓	✗	✗	device	✗	✗	✗	b	MQTT, multicast	JSON
[48]	none	✓	✗	✗	device	✗	✓	✗	s	unspecified	feature list
[13]	arbitrary	✓	✗	✗	device	✓	✓	✗	s	CoAP, CoRE-RD, P2P	e.g. CoRE Link Format
[39]	single	✗	✗	✗	device	✗	✗	✗	-	HTTP proxy	XML, JSON
[49]	per gatew.	✓	✗	✗	device	✓	✗	✗	b	CoAP, CoRE-RD, P2P	CoRE Link Format
[1]	federated	✗	✗	✗	device	✓	✗	✗	s	CoAP, CoRE-RD	JSON
[14]	per router	✓	✗	✗	device	✓	✗	✗	s	ICN, CoAP	CoRE Link Format
[27]	single	✗	✓	✗	device	✗	✗	✗	-	DPWS, multicast, MQTT	JSON
[5]	single	✗	✗	✓	user	✗	✓	✗	s	SPARQL	custom ontology
[26]	single	✗	✗	✗	device	✓	✓	✗	s	REST, Bluetooth, ZigBee, Wi-Fi	JSON
[19]	federated	✗	✗	✓	user	✗	✓	✗	b	SPARQL	SSN, SAN, OWL-S
[33]	single	✗	✓	✗	user/MW	✓	✗	✓	s	SOA middleware	XML
[46]	single	✗	✗	✓	device	?	✓	?	s	X-GSN, SPARQL	SSN extension
[9]	single	✗	✗	✗	device	✓	✗	✓	s	REST, LWM2M, CoAP	CoRE Link Format
[31]	none	✓	✗	✓	n.a.	✓	✓	✓	s	P2P Skipnet	context ontologies
[7]	per gateway	✓	✗	✗	device or gateway	✓	✗	✗	s	DNS-SD, mDNS, P2P, CoAP	CoRE Link Format or JSON-WSP
[28]	arbitrary	✓	✗	✗	device	✓	✗	✗	b	DNS-SD, mDNS, Bonjour	TXT-RR
[21] [3]	single	✗	✓	✗	device or disc. unit	✓	✓	✓	s	WS-Discovery, DPWS, multicast	WSDL
[47]	federated	✗	✗	✓	device	✓	✓	✗	s	AtomPub, mDNS	Atom Syndic. Format

2.3 Conclusion and challenges

Based on the literature review, we conclude that our work needs to address the following core challenges:

Limited resources Devices and networks of IoT ecosystems are often severely resource-constrained and hence possess only few computing and hardware capabilities, as well as low bandwidth [51]. Accordingly, when designing a discovery mechanism for IoT platforms, it must be ensured that the additional computional load for the devices is as low as possible. With respect to the network, the interaction model between the IoT platform and the devices should also minimize the exchange and size of messages and be robust against connection failures. Furthermore, it must be taken into consideration that the ecosystem might support only a small selection of technologies. For instance, many of the approaches listed in Table 1 rely on UDP multicast, which however may be unavailable in the underlying networks of certain ecosystems.

Need for fault-tolerance Many of the reviewed approaches are based on a single central repository (cf. Table 1), in which the formal descriptions of all devices are stored. However, such architectures are often unreliable, as the repositories represent single point of failures. Furthermore, they typically scale poorly with an increasing number of devices [16] and become a bottleneck as the system grows. For this reason, an architecture with federated repositories is to be preferred. In addition, it should allow to redundantly store the descriptions of devices in several repositories at the same time in order to avoid loss of information in case of failures of individual repositories.

High dynamics Most IoT ecosystems are highly dynamic, as devices can continuously join or leave them during runtime. Accordingly, devices that previously participated in a use case may be modified or become unavailable. In this case, the discovery component of the IoT platform must be able to detect these kind of changes and compensate for them, e.g. by transferring the affected tasks to other suitable devices of the ecosystem. In order to enable the efficient and timely detection of such events, the discovery mechanism should support asynchronous requests, as already offered by some of the approaches listed in Table 1. In addition, the high dynamics require that as many tasks as possible are performed automatically by the IoT platform and without human intervention, so that huge manual maintenance efforts are avoided. Thus, in contrast to some of the proposals in literature, the discovery approach should not necessarily require administrators to manually write, register and maintain descriptions of devices.

High flexibility IoT platforms are typically not tailored towards individual IoT systems and domains, but rather represent universal solutions that are supposed to be applicable in as many different contexts and ecosystems as possible. For this reason, they either employ technology-agnostic concepts or support a variety of different technologies. In contrast, the literature proposes a vast num-

ber of discovery approaches, which rely on different technology stacks and also vary strongly in their individual characteristics [25]. Furthermore, in the areas of enterprise systems, industry and home automation, numerous other discovery protocols are available that are individually optimized for their respective domains, but mostly unsuited for constrained IoT environments [40]. Due to the interoperability issue discussed earlier for the proposal of Papp et al. [38], it does not suffice to just integrate a single discovery approach into an IoT platform. On the other hand, the parallel integration of multiple different discovery approaches involves unreasonable implementation and maintenance efforts and still renders the IoT platform useless for ecosystems relying on other technologies. For this reason, a more flexible approach is needed that can be easily adapted to the characteristics and requirements of individual IoT ecosystems.

3 Method for autonomous execution of use cases

In order to enhance IoT platforms for the autonomous execution of pre-defined use cases within IoT ecosystems, we propose the method depicted in Figure 2. It assumes an ecosystem into which a discovery-enabled IoT platform has already been integrated. The process can be divided into two main parts: The first one is called *User Process* and is carried out only once by the users of the IoT platform. With the beginning of the *Discovery Process*, the IoT platform then takes over and autonomously performs various discovery-related tasks.

In step ① of this method, the users of the IoT platform define use cases that are supposed to be executed within their ecosystem. Such use cases typically involve the sensing of the environment, the recognition of situations and the intended reactions to those. The description "when a fire is detected in the factory building, the extinguishing system should be activated." could be considered as a simple example. While the use cases may be described either formally or informally at this point, they are always device-independent, meaning that they do not explicitly specify which devices of the ecosystem are supposed to be used for their implementation. As depicted in Fig. 2, the designed use cases represent the resulting artifact of this method step.

Based on the first step, the users translate the use cases into corresponding application logic for the IoT platform in step ②. As described in Section 1, IoT platforms typically support the definition of application logic either in a model-based manner or in the form of "if-then" rules. However, in this method, the application logic remains device-independent and does not refer to specific devices of the IoT ecosystem. Hence, the users do not need to manually register devices at the IoT platform by providing distinct identifiers, such as IP addresses, and link them with the application logic. Instead, they define the application logic with placeholders, which the IoT platform will automatically replace at runtime with those devices of the ecosystem that appear to be the most suitable

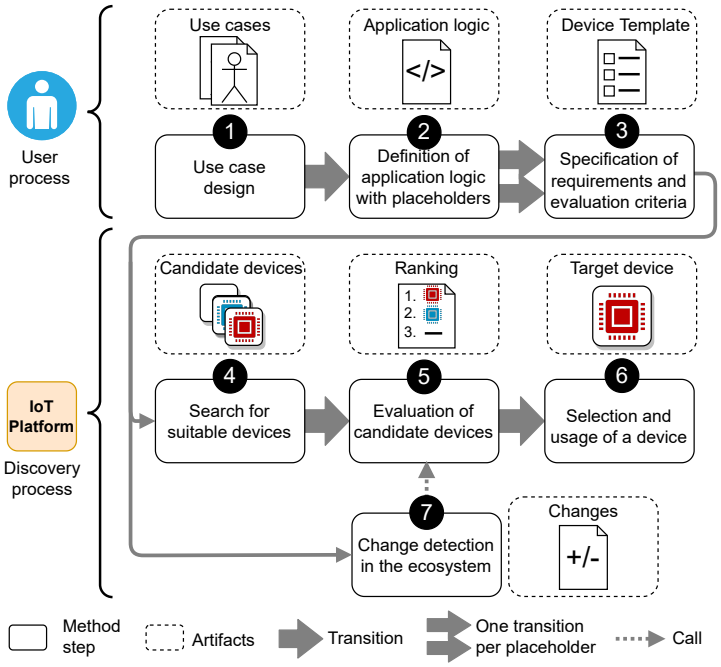

Fig. 2. The proposed method for the autonomous execution of use cases in IoT ecosystems by employing automatic device discovery and selection.

ones for realizing the use cases. Ultimately, the specified application logic with the device placeholders forms the resulting artifact of this step.

While the first two steps are performed only once within an iteration of the method, the following steps are executed for each device placeholder that is part of the previously specified application logic. In step ③, the users define criteria for each placeholder by describing the characteristics a device of the IoT ecosystem must possess to be considered a possible fill-in. In order to support this in a modular manner and to allow the reuse of criteria for multiple placeholders, the users register so-called *Device Templates* at the IoT platform and attach them to one or multiple placeholders of the application logic. Device templates can be considered as blueprints for the devices that are supposed to be discovered by the IoT platform within the ecosystem and selected as fill-ins for the placeholders. Each device template consists of *a*) a set of requirements, which correspond to search criteria and describe the hardware and software characteristics that a device must possess in order to become a possible fill-in for the respective placeholder, as well as *b*) a set of evaluation criteria specifying a scheme by which the devices meeting all requirements are supposed to be evaluated in order to determine the device that appears to be the most suitable fill-in. For example, a device template may prescribe that the devices must necessarily be located in a specific room and must be equipped with a temperature sensor in order to

qualify as fill-in for a certain placeholder. On the other hand, the evaluation criteria of the device template may define that among all the devices fulfilling these requirements always the device with the highest measurement accuracy should be selected as substitute for the placeholder. The user-defined device template for each placeholder forms the resulting artifact of this step.

With step ④, the method transitions to the discovery process. Here, the IoT platform searches the IoT ecosystem for devices that fulfill the requirements of the device templates. While in principle one or multiple of the discovery approaches discussed in Section 2 could be used for this purpose, we introduce a loosely-coupled architecture in Section 4 that allows to do this in a more flexible and technology-agnostic manner. During this procedure, the IoT platform verifies each device template that has been attached to a placeholder and retrieves the formal descriptions of all devices of the ecosystem that fulfill the specified requirements. Since these devices are potentially suitable candidates for filling in a specific placeholder, they are called *candidate devices*. If no candidate devices can be found for a placeholder, it will currently not be possible to implement the corresponding use case and hence the process terminates at this point. However, it can be resumed as soon as suitable devices become available (cf. step ⑦). The descriptions of the candidate devices form the artifact of this step.

In step ⑤, the IoT platform assesses the previously retrieved candidate devices for each device template. For this purpose, a score is calculated and assigned to the candidate devices by applying the evaluation criteria of the device template to the device descriptions. The resulting score reflects how well the respective device suits as fill-in for the placeholder in comparison to the other candidate devices. After all candidate devices for the device template have been evaluated, they are ranked based on their scores in descending order, such that the device with the highest score is listed at the top. This ranking can be considered as a recommendation in terms of which devices should preferably be used as fill-in for the placeholder and represents the resulting artifact of this step.

Based on the ranking, the IoT platform in step ⑥ selects a so-called *target device* among the associated candidate devices for each placeholder. Ideally, the decision is made in favor of the first device in the ranking, but the IoT platform must also ensure that this device is currently available in the IoT ecosystem and thus can actually be used as a fill-in. In case the device is unavailable, e.g., due to a technical issue, the next device in the ranking is selected and checked for its readiness. The finally selected target device is the output of this step.

After the start of the discovery process, step ⑦ becomes active in parallel to step ④. It makes use of asynchronous queries as introduced in Section 2, so that the IoT platform is notified about changes within the ecosystem. This way, the IoT platform is able to quickly detect *a*) when a new device that also satisfies the requirements of a user-defined device template joins the ecosystem, *b*) when an existing device satisfying the requirements of a device template becomes unavailable within the ecosystem, or *c*) when an existing device satisfying the requirements of a device template is modified, such that its capabilities change. Each of these three cases might have an impact on the candidate and target

devices that were previously determined for a device template. For example, it could turn out that a new device joining the ecosystem receives a higher score in step ⑤ than the devices that were formerly evaluated and thus represents a more suitable fill-in for the placeholder in the application logic. On the other side, a device that was selected in step ⑥ might become unavailable or change its capabilities over time, such that it does not fulfill the requirements of its device template anymore and hence can no longer be considered a candidate device. In order to cope with these situations, step ⑦ re-initiates the execution of the steps ⑤ and ⑥ as soon as it is notified about a modification in the ecosystem that actually affects the candidate devices of at least one placeholder. As part of the invocation of step ⑤, a formal description of the observed changes is passed, such that the set of candidate devices can be updated and the ranking re-calculated accordingly. As a result, step ⑦ allows the IoT platform to adapt to changes within highly dynamic ecosystems at runtime by re-evaluating the candidate devices and, if necessary, switching the fill-in devices for placeholders.

4 Architecture supporting discovery for IoT platforms

To accommodate the application of the previously presented method in IoT ecosystems, we propose the architecture illustrated in Fig. 3. According to it,

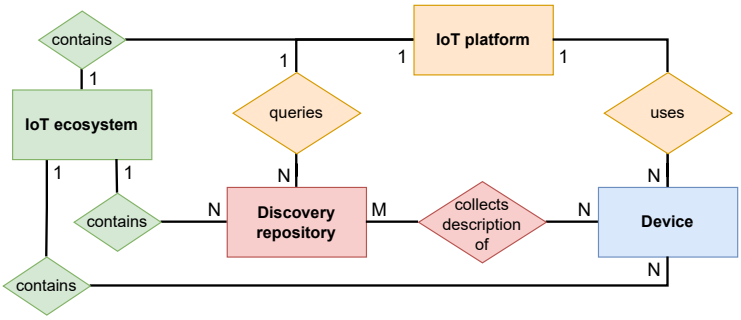

Fig. 3. ER model of the components that are involved in the architecture.

an IoT ecosystem consists of at least three different types of components: *a*) an IoT platform, *b*) an arbitrary number of devices and *c*) at least one so-called *discovery repository (DR)*. The device entities in this architecture do not solely refer to physical IoT devices, but may also encapsulate possibly necessary IoT gateways [53] for mediating between different networks and communication technologies. DRs are self-contained software components that collect formal descriptions of the devices within an ecosystem and provide them to the IoT platform on request via a prescribed interface. Since they serve as an additional abstraction layer between the IoT platform in a cloud environment and the devices at the edge, loose coupling can be achieved.

4.1 Discovery Repositories

Discovery repositories (DRs) are explicitly developed and deployed for an IoT ecosystem by its administrators. They are similar to the repositories proposed

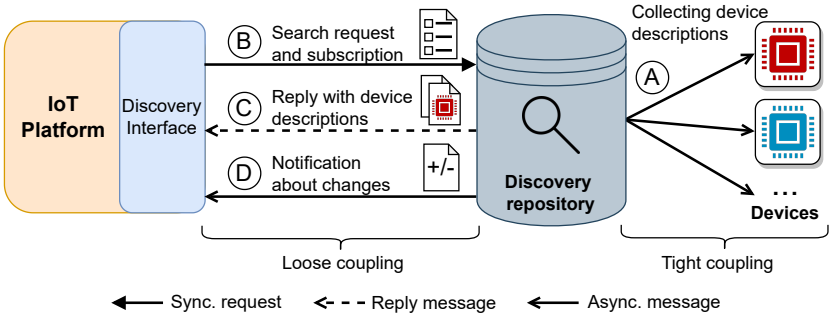

Fig. 4. Interactions between an IoT platform and a discovery repository.

by Gomes et al. [18], but more flexible and independent in their design and they undertake a broader field of responsibilities. They basically pursue three tasks:

Collection of device descriptions: The main task of a DR is to search, collect and manage the formal descriptions of the devices that are available within the IoT ecosystem, as depicted in Fig. 4 (A). This includes the storage of new descriptions when devices join the IoT ecosystem, the deletion of descriptions when devices leave or become unavailable, and the updating of descriptions in case device characteristics change over time. The selection of technologies and approaches that are employed for carrying out this task is however left to the individual implementation of the DRs. In the simplest case, the administrators of an IoT ecosystem can manually manage a DR by adding, deleting and updating the descriptions of their devices as needed. Here, the DR can be considered a simple database. In more complex systems though, the DRs may implement one or even multiple of the discovery approaches that are proposed in literature (cf. Section 2), which then enable the automatic discovery of devices and the collection of their descriptions. Since individual requirements tend to change from ecosystem to ecosystem, the administrators are encouraged to decide which approaches they consider most suitable for the DRs in their present scenario. In case they plan to employ several DRs, these may also implement different techniques.

Retrieval of device descriptions: As shown in Figure 4 (B), each DR provides an interface through which the IoT platform is able to issue search queries for devices. This interface is prescribed by the IoT platform implementing the method as introduced in Section 3, such that the DRs can be developed against this interface. The queries contain the requirements that were previously de-

fined by a user at the IoT platform as part of a device template. When a DR receives such a request, it is expected to search its collected device descriptions and eventually return a response to the IoT platform (cf. Ⓒ) that contains the set of all device descriptions meeting the given requirements. This way, the IoT platform can use the DRs in order to obtain information about the devices of the ecosystem that embody candidate devices for placeholders.

Notification about changes: As part of a search query for devices (cf. Ⓑ), the IoT platform has also the option to register a subscription at the DR, such that it is asynchronously notified by the DR about changes that affect the result set of the query during runtime. As a result, the DR will inform the subscribed IoT platform with a message (cf. Ⓓ) as soon as it detects that *a*) a new device became available in the IoT ecosystem whose description also satisfies the requirements of the query, *b*) a device whose description originally satisfied the requirements of the query became unavailable in the IoT ecosystem, or that *c*) the description of a device that either previously or now satisfies the requirements of the query was modified with respect to its capabilities.

Optionally, a DR may also be used to transform already existing descriptions of devices, e.g. as provided by the device manufacturers, into formats that are supported by the IoT platform. In this case, the DR additionally acts as a message translator [23] that allows the reuse of device descriptions and thus eases the integration of the IoT platform into existing IoT ecosystems.

The DRs do not necessarily need to be realized as stand-alone applications; instead, they can also be implemented on top of existing components, such as IoT gateways [53]. This applies in particular to ecosystems in which the gateways wrap larger groups of devices and even perform discovery tasks themselves.

In summary, the DRs encapsulate all ecosystem-specific aspects and hide them from the IoT platform, which can then use the DRs through a prescribed interface. As a result, the DRs become tightly coupled with the ecosystem and its contained devices, but potentially loosely coupled with the IoT platform.

4.2 Request-reply interactions

In our proposed architecture, publish-subscribe-based messaging [23] via a message broker facilitates the communication between the IoT platform and the DRs. Accordingly, we assume that the DRs subscribe to so-called *request topics* at this message broker, under which they expect to receive query messages from the IoT platform. Similar to the concept of Gomes et al. [18], the administrators need to register the endpoints of the available DRs at the IoT platform. However, in our approach, they do this by specifying the request topics instead of network addresses. As a result, a one-to-many relationship between topic registrations and DRs is achieved, because an arbitrary number of DRs may be accessible under the same request topic due to the publish-subscribe paradigm. This is an important step towards loose coupling. Furthermore, in order to avoid the generation of individual request messages for each DR, our approach employs the

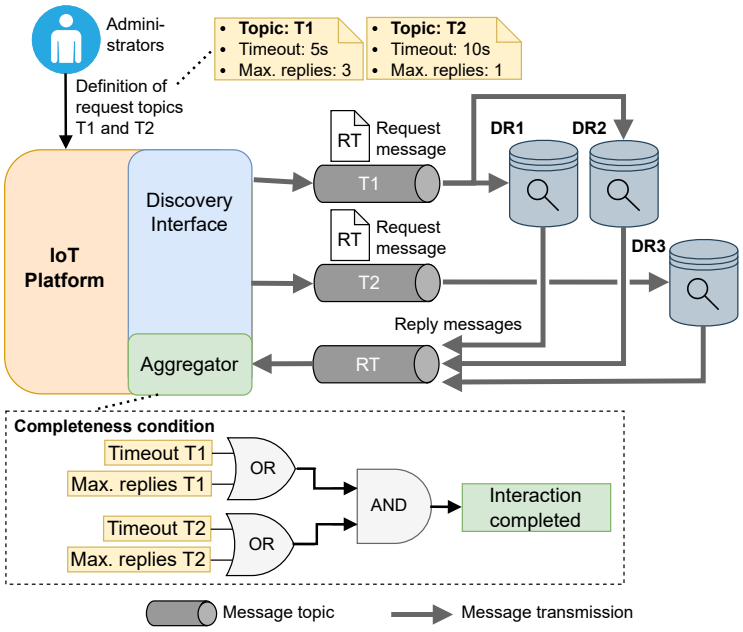

Fig. 5. Application of the scatter gather messaging pattern for achieving loose coupling between the IoT platform and the discovery repositories.

scatter gather messaging pattern [23], which allows to broadcast a single request message to multiple DRs at once and to subsequently collect and combine their responses. Fig. 5 illustrates how this pattern is applied to search queries: In the given ecosystem, three DRs are available, of which `DR1` and `DR2` are subscribed to the same request topic `T1`, while `DR3` is subscribed to `T2`. Both request topics have already been registered at the IoT platform by the administrators. In case a search query for device descriptions is supposed to be issued against all DRs according to (B) in Figure 4, the IoT platform creates a corresponding request message and publishes it under the request topics `T1` and `T2` at the message broker. This way, the request message is broadcasted to all available DRs and processed by them. It is worth noting that for the delivery of the request message the IoT platform does not need to know how many DRs are actually subscribed to each request topic. Next to the requirements of a device template, the request message also contains a so-called *reply topic* `RT`, which was previously generated by the IoT platform and subscribed by it at the message broker. It serves as a return address [23] for the DRs. Accordingly, after the individual DRs processed the request message, they publish their reply messages under this topic, so that the message broker can deliver them back to the IoT platform. Here, a software component called *aggregator* [23] receives the replies and aggregates them into a common data structure, which can then be further processed in correspondence with the method steps presented in Section 3. Since all device descriptions are

allowed to be redundantly available in multiple DRs at the same time and even in different versions, the aggregator is also responsible for eliminating duplicated device descriptions from the result set. This is done by *a*) verifying unique device identifiers that are embedded in the device descriptions, such as MAC addresses, as well as by *b*) comparing timestamps that are part of the device descriptions in order to ensure that always the most recent version of a device description is used among all versions that were received for the pertaining device.

Due to the one-to-many relationship, the IoT platform does not know how many DRs are currently available in the ecosystem and will send a reply in response to a request message. Hence, the aggregator must decide in a different way when a scatter gather interaction can be considered complete. For this purpose, a completeness condition is applied, which consists of two types of parameters that are provided by the administrators during the registration of the request topics (cf. Fig. 5): While the *max. replies* value specifies the maximum number of reply messages that are expected to be received from DRs for a given request topic, the *timeout* defines the maximum period of time the IoT platform is supposed to wait for incoming replies. As soon as at least one of both events occur for all request topics, the completeness condition is fulfilled, indicating that the IoT platform can start to aggregate and process the received messages.

As a consequence of this approach, loose coupling is achieved, which allows administrators to add new DRs to the ecosystem or remove DRs anytime, without needing to explicitly update or reconfigure the IoT platform. The only prerequisite is that the new DRs make use of already registered request topics.

5 Prototype and Discussion

As proof of concept, we integrated the method as presented in Section 3 into the Multi-purpose Binding and Provisioning Platform (MBP)[2] [45], an open source IoT platform that allows the definition of rule-based application logics. For testing purposes, we developed an exemplary DR[3] that implements the tasks as described in Section 4 and is able to communicate with the MBP via MQTT in accordance with the proposed interaction scheme. The software architecture of this DR is depicted in Fig. 6. At its core, it consists of a Spring Boot application with a REST interface that allows administrators to manually manage the descriptions of the devices that are available within the IoT ecosystem. Accordingly, they can add descriptions of devices joining the ecosystem, remove descriptions of leaving devices and update descriptions of modified devices. The application does not store the device descriptions itself, but instead uses an instance of Elasticsearch for this purpose, which treats and indexes the device descriptions as documents. This way, the DR is able to efficiently process incoming search requests of the IoT platform by translating them into corresponding queries for Elasticsearch. Furthermore, the DR supports subscriptions and is thus able to asynchronously notify the IoT platform

[2] MBP on GitHub: https://github.com/IPVS-AS/MBP
[3] DR on GitHub: https://github.com/IPVS-AS/MBP-DiscoveryRepository

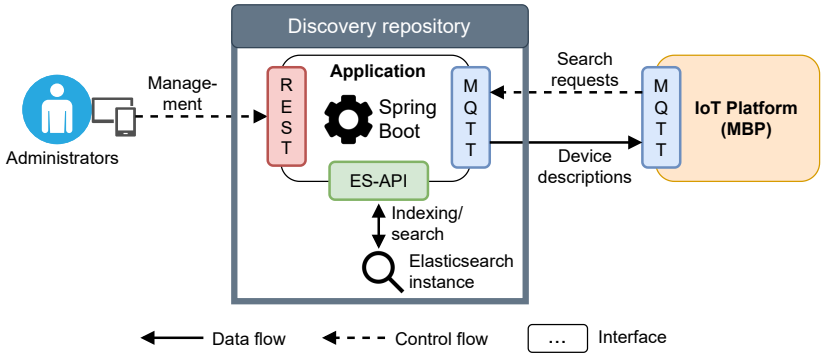

Fig. 6. Architecture of our discovery repository that serves as proof of concept.

about changes that affect the result sets of preceding search requests. By using the discovery-enabled MBP and running multiple instances of this DR in parallel, we were able to practically test and evaluate our concepts.

The proposed method and its underlying architecture can support IoT ecosystems that involve such an IoT platform at improving their **availability**: Since the DRs might be able to monitor the devices within the ecosystem and notify the IoT platform asynchronously about detected changes, the IoT platform can quickly react to device failures at runtime by re-evaluating the candidate devices and selecting the next most suitable device for the affected placeholder. On the other hand, the use of multiple, federated, but potentially also differently implemented DRs avoids the establishment of single point of failures. Furthermore, the aggregator component (cf. Fig. 5) is able to deal with duplicates, which allows the redundant storage of device descriptions within multiple DRs and hence leads to higher robustness. The federated architecture gives also rise to **scalability**, because an increasing number of devices can be countered by the deployment of additional DRs. The prerequisite for this is that the DRs share their responsibilities of discovering and monitoring the devices, such that not all DRs need to assess all available devices. This can be achieved e.g. by the usage of subnets. Due to the loose coupling, new DRs can be flexibly added to or removed from the ecosystem without needing to reconfigure the IoT platform, which eases horizontal scaling. With this approach, almost arbitrary numbers of devices and DRs can be inserted into an IoT ecosystem, until the IoT platform, which is a centralized component by nature, becomes a bottleneck itself. The proposed concepts can also be considered as **efficient** in terms of resource consumption, as they *a*) do not necessarily put additional tasks or load onto the typically resource-constrained devices, albeit this also depends on the specific implementation of the DRs, *b*) apply the scatter gather messaging pattern for the interaction between the IoT platform and the DRs, which can also be used on top of lightweight messaging protocols such as MQTT and avoids overhead in the generation of request messages and *c*) allow the developers of the DRs to select the most suitable and most efficient technologies for the discovery of devices

within their ecosystem, based on the application scenario at hand and by considering the actually available resources. In contrast to the proposals in literature (cf. Section 2), the main benefit of our approach is its **flexibility**. It allows to integrate the enhanced IoT platforms into different kinds of IoT ecosystems and to use them in a wide range of application scenarios. The foundations for this are provided by the DRs, since they serve as an additional abstraction layer that encapsulates and hides all ecosystem-specific aspects from the IoT platform. At the same time, they offer a prescribed interface through which the IoT platform can issue criteria-based queries for devices, as well as register subscriptions for asynchronous notifications about changes within the ecosystem. This way, the IoT platform remains technology-agnostic and can co-operate with both manually managed IoT ecosystems, as well as highly dynamic ones that need to make use of specific, potentially custom-tailored discovery technologies. In addition, the DRs also support the reuse of already existing device descriptions. Consequently, such an IoT platform provides high interoperability and can be integrated in various kinds of IoT ecosystems without having to adapt the IoT platform itself. On the downside, the additional abstraction layer may cause overhead in terms of development efforts, latency and resource consumption. However, this highly depends on the individual implementations of the DRs.

6 Conclusion

In this paper, we presented a method that allows IoT platforms *a*) to assist their users in the specification of device-independent application logic for use cases by employing placeholders, *b*) to let their users define device templates, which prescribe requirements and evaluation criteria for devices that should be selected as fill-ins for the placeholders, *c*) to autonomously discover and search for devices within an IoT ecosystem that fulfill the user-defined requirements and thus constitute candidate devices for the placeholders, *d*) to score and rank candidate devices with respect to the user-defined evaluation criteria, *e*) to select the most suitable candidate devices as fill-ins for the placeholders and *f*) to detect changes in highly dynamic IoT ecosystems at runtime and to cope with them by re-evaluating the candidate devices of the affected placeholders. To ease the application of this method, we also introduced a supporting architecture for IoT ecosystems. It comprises so-called discovery repositories, which are tailored towards the specific needs of the application scenario at hand and serve as an additional abstraction layer between the IoT platform and the ecosystem to establish loose coupling. As a result, IoT platforms implementing our method can remain technology-agnostic and thus be applied to a wide range of different scenarios without needing to adapt the IoT platform itself. Furthermore, also the availability and scalability of the encompassing IoT systems can be improved.

In future work, we pursue to conduct tests in larger IoT ecosystems in order to empirically verify our assumptions. Moreover, we plan to consider options for the automatic selection of suitable criteria for device templates as well as for including availability predictions into the evaluation of candidate devices.

References

1. Barreto, F.M., Duarte, P.A.d.S., Maia, M.E., et al.: Coap-ctx: A context-aware coap extension for smart objects discovery in internet of things. In: 2017 IEEE 41st Annual Computer Softw. and Applic. Conference. vol. 1, pp. 575–584 (2017)
2. Baykara, C., Şafak, I., Kalkan, K.: SHAPEIoT: Secure handshake protocol for autonomous IoT device discovery and blacklisting using physical unclonable functions and machine learning. In: 13th International Conference on Network and Communications Security (NCS 2021) (09 2021)
3. Chirila, S., Lemnaru, C., Dinsoreanu, M.: Semantic-based IoT device discovery and recommendation mechanism. In: 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP). pp. 111–116 (2016)
4. Chiu, Y.H., Liao, C.F., Chen, K.: Transparent web of things discovery in constrained networks based on mdns/dns-sd. In: 2021 International Conference on Platform Technology and Service (PlatCon). pp. 1–6. IEEE (2021)
5. Chun, S., Seo, S., Oh, B., et al.: Semantic description, discovery and integration for the internet of things. In: Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015). pp. 272–275. IEEE (2015)
6. Cimmino, A., McCool, M., Tavakolizadeh, F., et al.: Web of Things (WoT) Discovery. W3c working draft, World Wide Web Consortium (W3C) (Jul 2022)
7. Cirani, S., Davoli, L., Ferrari, G., et al.: A scalable and self-configuring architecture for service discovery in the internet of things. IEEE internet of things journal **1**(5), 508–521 (2014)
8. da Cruz, M.A.A., Rodrigues, J.J.P.C., et al.: A reference model for internet of things middleware. IEEE Internet of Things Journal **5**(2), 871–883 (2018)
9. Datta, S.K., Bonnet, C., Nikaein, N.: An IoT gateway centric architecture to provide novel m2m services. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). pp. 514–519. IEEE (2014)
10. Dawod, A., Georgakopoulos, D., Jayaraman, P.P., et al.: An IoT-owned service for global IoT device discovery, integration and (re) use. In: 2020 IEEE International Conference on Services Computing (SCC). pp. 312–320. IEEE (2020)
11. Del Gaudio, D., Hirmer, P.: Fulfilling the IoT Vision: Are We There Yet? In: IoTBDS. pp. 367–374 (2020)
12. Demir, K.: A qos-aware service discovery and selection mechanism for IoT environments. Sādhanā **46**(4), 1–13 (2021)
13. Djamaa, B., Kouda, M.A., Yachir, A., et al.: Fetchiot: Efficient resource fetching for the internet of things. In: 2018 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 637–643. IEEE (2018)
14. Dong, L., Ravindran, R., Wang, G.: Icn based distributed IoT resource discovery and routing. In: 2016 23rd International Conference on Telecommunications (ICT). pp. 1–7. IEEE (2016)
15. Dürr, F., Hönle, N., Nicklas, D., Becker, C., Rothermel, K.: Nexus - a platform for context-aware applications. Roth, Jörg, editor **1**, 15–18 (2004)
16. Evdokimov, S., Fabian, B., Kunz, S., et al.: Comparison of discovery service architectures for the internet of things. In: 2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. pp. 237–244 (2010)
17. Fernandez, S., Amoretti, M., Restori, F., et al.: Semantic identifiers and dns names for IoT. In: 2021 International Conference on Computer Communications and Networks (ICCCN). pp. 1–9. IEEE (2021)

18. Gomes, P., Cavalcante, E., Batista, T., et al.: A semantic-based discovery service for the internet of things. Journal of Internet Services and Applic. **10**(1) (2019)
19. Gomes, P., Cavalcante, E., Rodrigues, T., et al.: A federated discovery service for the internet of things. In: Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT. pp. 25–30 (2015)
20. Gubbi, J., Buyya, R., Marusic, S., et al.: Internet of things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems **29**(7), 1645–1660 (sep 2013)
21. Guinard, D., Trifa, V., Karnouskos, S., et al.: Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. IEEE transactions on Services Computing **3**(3), 223–235 (2010)
22. Guth, J., Breitenbücher, U., Falkenthal, M., et al.: Comparison of iot platform architectures: A field study based on a reference architecture. In: 2016 Cloudification of the Internet of Things (CIoT). pp. 1–6. IEEE (2016)
23. Hohpe, G., Woolf, B.: Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional (2004)
24. ITU: ITU-T Recommendation Y.2060: Overview of the Internet of things. Tech. rep., International Telecommunication Union (Jun 2012)
25. Khaled, A.E., Helal, S.: Interoperable communication framework for bridging restful and topic-based communication in IoT. Future Generation Computer Systems **92**, 628–643 (2019)
26. Khodadadi, F., Dastjerdi, A.V., Buyya, R.: Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT. In: 2015 Internat. Conference on Recent Advances in Internet of Things. pp. 1–6. IEEE (2015)
27. Kim, S.M., Choi, H.S., Rhee, W.S.: IoT home gateway for auto-configuration and management of mqtt devices. In: 2015 IEEE Conference on Wireless Sensors (ICWiSe). pp. 12–17. IEEE (2015)
28. Klauck, R., Kirsche, M.: Bonjour contiki: A case study of a dns-based discovery service for the internet of things. In: International Conference on Ad-Hoc Networks and Wireless. pp. 316–329. Springer (2012)
29. Kovatsch, M., Matsukura, R., Lagally, M., et al.: Web of Things (WoT) Architecture. W3c recommendation, World Wide Web Consortium (W3C) (Apr 2020)
30. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: service-oriented architecture best practices. Prentice Hall Professional (2005)
31. Li, J., Zaman, N., Li, H.: A decentralized locality-preserving context-aware service discovery framework for internet of things. In: 2015 IEEE International Conference on Services Computing. pp. 317–323. IEEE (2015)
32. Li, Z., Yao, J., Huang, H.: A coap-based decentralized resource discovery for IoT network. In: 2021 6th International Conference on Communication, Image and Signal Processing (CCISP). pp. 398–402. IEEE (2021)
33. Lunardi, W.T., de Matos, E., Tiburski, R., et al.: Context-based search engine for industrial IoT: Discovery, search, selection, and usage of devices. In: 2015 IEEE 20th Conference on emerging technologies & factory automation. pp. 1–8 (2015)
34. Madjarov, I., Slaimi, F.: A graph-based web services discovery framework for IoT ecosystem. Open Journal of Internet Of Things **7**(1), 1–17 (2021)
35. Miorandi, D., Sicari, S., De Pellegrini, F., et al.: Internet of things: Vision, applications and research challenges. Ad hoc networks **10**(7), 1497–1516 (2012)
36. Nicklas, D., Mitschang, B.: On building location aware applications using an open platform based on the nexus augmented world model. Software and Systems Modeling **3**(4), 303–313 (2004)

37. Papazoglou, M.P., Georgakopoulos, D.: Introduction: Service-oriented computing. Communications of the ACM **46**(10), 24–28 (2003)
38. Papp, I., Pavlovic, R., Antic, M.: WISE: MQTT-based protocol for ip device provisioning and abstraction in IoT solutions. Elektronika ir Elektrotechnika **27**(2), 86–95 (2021)
39. Pêgo, P.R., Nunes, L.: Automatic discovery and classifications of IoT devices. In: 12th Iberian Conference on Inform. Systems and Technol. pp. 1–10. IEEE (2017)
40. Pereira, E.M., Pinto, R., dos Reis, J.P.C., Gonçalves, G.: MQTT-RD: A MQTT based resource discovery for machine to machine communication. In: IoTBDS. pp. 115–124 (2019)
41. Pourghebleh, B., Hayyolalam, V., Aghaei Anvigh, A.: Service discovery in the internet of things: review of current trends and research challenges. Wireless Networks **26**(7), 5371–5391 (2020)
42. Raghu Nandan, R., Nalini, N., Hamsavath, P.N.: IoT-CBSE: A search engine for semantic internet of things. In: Emerging Research in Computing, Information, Communication and Applications, pp. 265–271. Springer (2022)
43. Riggs, C., Patel, J., Gagneja, K.: IoT device discovery for incidence response. In: 2019 Fifth Conference on Mobile and Secure Services. pp. 1–8. IEEE (2019)
44. Sharma, M., Pant, S., Kumar Sharma, D., et al.: Enabling security for the industrial internet of things using deep learning, blockchain, and coalitions. Transactions on Emerging Telecommunications Technologies **32**(7), e4137 (2021)
45. Franco da Silva, A.C., Hirmer, P., Schneider, J., et al.: MBP: Not just an IoT platform. In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). pp. 1–3. IEEE (2020)
46. Soldatos, J., Kefalakis, N., Hauswirth, M., et al.: Openiot: Open source internet-of-things in the cloud. In: Interoperability and open-source solutions for the internet of things, pp. 13–25. Springer (2015)
47. Stirbu, V.: Towards a restful plug and play experience in the web of things. In: 2008 IEEE International Conference on Semantic Computing. pp. 512–517 (2008)
48. Sunthonlap, J., Nguyen, P., Wang, H., et al.: SAND: A social-aware and distributed scheme for device discovery in the internet of things. In: 2018 Internat. Conference on Computing, Networking and Communications (ICNC). pp. 38–42. IEEE (2018)
49. Tanganelli, G., Vallati, C., Mingozzi, E.: Edge-centric distributed discovery and access in the internet of things. IEEE IoT Journal **5**(1), 425–438 (2017)
50. Transforma Insights: Global IoT market to grow to 24.1 billion devices in 2030, generating $1.5 trillion annual revenue. https://transformainsights.com/news/iot-market-24-billion-usd15-trillion-revenue-2030 (May 2020), accessed: 07.02.2022
51. Vermesan, O., Friess, P.: Internet of things: converging technologies for smart environments and integrated ecosystems. River publishers (2013)
52. Wang, R., Lu, J.: Qos-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in IoT. Wireless Personal Communications pp. 1–14 (2021)
53. Zhu, Q., Wang, R., Chen, Q., Liu, Y., Qin, W.: Iot gateway: Bridging wireless sensor networks into internet of things. In: 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. pp. 347–352 (2010)