Edge-supported Microservice-based Resource Discovery for Mist Computing

Arash Sattari, Rouhollah Ehsani, Teemu Leppnen, Susanna Pirttikangas and Jukka Riekki
Center for Ubiquitous Computing,
University of Oulu, Finland
{arash.sattari, rouhollah.ehsani, teemu.leppanen, susanna.pirttikangas, jukka.riekki}@oulu.fi

***Abstract*—Mist computing extends the Internet of Things computing infrastructures to the IoT devices at the edges of the networks. The dynamic characteristics of the IoT environments and resource limitations of the devices introduce challenges to the orchestration of the mist platform resources. In this paper, we present a hybrid resource discovery solution for mist, based on the IETF CoRE Resource Directories deployed as containerized Microservices to the supporting edge devices. This way the directory instances can be deployed on-demand as part of the edge platform, where each instance serves a mist network connected to the hosting edge device. This enables low latency resource queries at one-hop distance for the mist applications. At the edge layer, the directories form a distributed discovery infrastructure, connecting resources in disparate mist networks with each other and cloud and edge applications. A real-world prototype of such discovery infrastructure is implemented, based on low resource edge devices hosting the directory instances and low-power embedded devices as the mist resource servers and clients. The prototype is evaluated with latency and power consumption measurements, where the results show that discovery latency is as low as half a millisecond with a low power consumption.**

***Index Terms*—Distributed Resource Directory, Container, Energy Efficiency, Low Latency, Fog Computing**

I. Introduction

Mist computing [1] extends the Internet of Things (IoT) computational infrastructures from the cloud through the edge to the connected IoT devices at the extreme endpoints of networks. The cloud provides high availability of computational resources and large-scale data for IoT applications, whereas the edge layer, exemplified by fog computing platforms, add computational layers between the cloud and the data producing devices. This lowest layer, consisting of these resource-constrained IoT devices, is considered as the mist layer. Such a hierarchical three-tier IoT platform aims to reduce the massive-scale data transfer load on the networks and provides real-time capabilities in close proximity of the users. Data processing already at the data sources in the mist layer increases the energy efficiency of the mist devices and addresses privacy concerns, as only the relevant data is shared and transmitted upstream.

Challenges lie in the nature of the mist layer, built on constrained IoT devices, a mixture resources based on low-power embedded systems and mobile devices with integrated sensors and actuators. Such devices can only perform lightweight data processing and need to rely on different operational modes to reduce their power consumption. Moreover, the devices operate on constrained wireless networks with a variety of optimized low power communication technologies and protocols. In such an open environment, the role of a mist computing platform is to discover, integrate, assemble and manage the available resources for the provisioned IoT applications.

Resource discovery on the IoT device level has been addressed before [2], [3], with client-server solutions provided as system services and multicast-based direct discovery between devices. However, multicasting in these constrained device networks is challenging due to device limitations and increased energy consumption. In the client-server solutions, a limiting factor is the resource query processing overhead. Here, the Internet Engineering Task Force (IETF) has proposed Constrained RESTful Environments (CoRE) framework for IoT application development, including Resource Directory (RD) specifications, that address low power operations on the IoT devices [4].

This paper presents, as the first contribution, distributed resource discovery solution for mist computing platforms that employs edge devices to host IETF CoRE RD instances in the proximity of the mist devices. This way the edge devices, such as gateways and proxies that the mist devices inherently are connected to, realize low latency resource discovery without the limitations of the mist devices. The virtualized RD instances, implemented as containerized Microservices that follow the CoRE specifications, are deployed on-demand and managed as system services provided by the edge platform. Moreover, the RD instances at the edge layer form a horizontal distributed peer-to-peer (P2P) resource discovery infrastructure, which in turn enables the cloud and edge applications to discover mist resources. The second contribution of the paper is a real-world evaluation of the power consumption of the containerized Microservice-based RD instances. To the best of our knowledge, such evaluation has not been presented in the previous work on IoT resource discovery.

The rest of the paper is organized as follows. In Section 2,

This paper has been accepted for publication in the IEEE International Conference on Pervasive Intelligence and Computing (IEEE PICom 2020), August 17-21, Online, 2020.

the background in IoT device resource discovery is presented. In Sections 3 and 4, the proposed resource discovery solution is described in detail and a real-world prototype is evaluated with regard to the discovery latencies and power consumption at the edge devices. In Section 5, the evaluation results and implications of the presented solution are discussed. Section 6 concludes the contributions of the paper.

II. Background

IoT resource discovery solutions are categorized into directory-based and directoryless [2]. In the directoryless solutions, devices interact directly with each other over predefined patterns, e.g. with multicast/broadcast and discovery or peer-to-peer (P2P) protocols, or by introducing resource crawlers. Due to computing and connectivity limitations, it is difficult for IoT devices to handle a large number of direct resource queries. Furthermore, discovery becomes more complicated if the devices are mobile or rely on radio duty cycling or sleep modes to save power.

In the directory-based solutions, selected system components maintain online resource directory instances. These solutions have been further categorized into centralized and distributed [3]. The centralized solutions face the problem of maintaining (large-scale) directories in an inherently dynamic IoT system in real-time, without a priori knowledge of the resource demand and resulting network traffic. In the distributed solutions, multiple devices act as resource directories for location-specific real-time discovery, avoiding a single point of failure and facilitating, for example, self-configuration and -organization. Distributed solutions importantly address scalability, exemplified by P2P architectures for discovery such as [5]. However, the overhead of network traffic and query processing load still must be taken into account. A way to reduce the processing overhead is to use caching of query results, as in [5].

In addition, hybrid IoT resource discovery solutions, such as [7], vary the discovery method between the directoryless and directory-based approaches. If an initial centralized RD instance cannot be found in a location, the devices switch to multicast mode to advertize and discover resources. A benefit of the hybrid approach are extended deployment options and dedicated RD instances for specific purposes.

The IETF CoRE framework provides standardization for CoRE RD[1], targeted for low-power IoT devices in constrained networks, that maintains runtime information of available system resources. The devices communicate with Constrained Application Protocol (CoAP), where the resource descriptions and queries are based on the CoRE Web Link format[2]. The CoRE resource descriptions contain resource address (URI) and resource metadata, such as semantic resource type, media type, interface description and resource size.

The CoRE RD has been further optimized for constrained environments. The authors in [7] implement additional parameters into the CoRE resource description and utilize duty cycling in the IoT devices with the aim to improve energy efficiency. In [8]–[10], the CoRE resource descriptions and queries are extended with semantic and contextual information as additional search attributes. Moreover, the authors in [10] integrate resource discovery into the device network protocol stack atop CoAP and propose CoAP FETCH method to retrieve and filter resource descriptions. A structured document, based on the Open Connectivity Foundation (OCF) modeling language, is used for resource descriptions atop CoRE specifications in [11]. In [5], [6], [12], [13], distributed CoRE RD P2P overlays are presented. A hierarchical resource discovery scheme is proposed in [14], where resources are found by traversing the hierarchy of gateways presenting IoT devices. In large-scale distributed IoT applications, search engines and layered resource discovery frameworks, as in [15], [16], provide discovery services with structured document and linked open data. In [17], both centralized and distributed service discovery solutions utilize network routers as CoRE RD peers that also perform discovery load balancing. A comparison of the IoT resource discovery solutions is presented in Table I.

Although the resource discovery and orchestration in the cloud and edge platforms is out of the focus of this paper, solutions for service discovery have been presented for centralized cloud-based IoT systems, for virtualization infrastructures [18], and for Microservices [19], [20]. Commonly, the resource requirements in edge applications are considered known a priori [2], where scalability is achieved through deploying more service or application instances to the platform. However, mist and fog applications predominantly rely on the resources available in a physical location.

III. Hybrid Resource Discovery for Mist

Mist platforms rely on an assembly of a heterogeneous set of constrained devices, in terms of both computation and communication, that share their resources on an open and dynamic IoT environment. In such a dynamic environment, the resource registration and runtime updates, exemplified by periodic advertising, introduce inevitable communication and computation overhead for the platform operation. Therefore, small overhead of resource registration and discovery is required, where resource descriptions should be lightweight, for low power operations. In open IoT environments, standardized resource description formats and discovery protocols provide interoperability in large-scale.

Furthermore, particularly in mist computing, the platforms cannot rely solely on the local information for efficient operation. Support for device mobility and additional data sources, outside the local mist network and across the IoT system architecture, and for evolving application scenarios are needed. Here, domain-specific RD instances utilizing semantic information can be deployed to the infrastructure, as in [8], [9], [21].

In the light of these challenges, in this Section we discuss how edge-supported resource discovery is beneficial for mist and present our hybrid resource discovery solution.

[1]https://datatracker.ietf.org/doc/draft-ietf-core-resource-directory/

[2]https://tools.ietf.org/html/rfc6690

TABLE I
COMPARISON OF IOT RESOURCE DISCOVERY SOLUTIONS

Reference	Model	Architecture	Deployment	Resource description	Optimization	Evaluation
[5], [6]	Distributed	P2P	Peer nodes	IETF CoRE	Latency	Real world
[7]	Centralized & distributed	Hybrid	Network devices	IETF CoRE Extension	No. of messages	Simulation
[8]	Centralized	Client-Server	Network devices	Semantic IETF CoRE	-	Real world
[9]	Centralized	Client-Server	Mobile devices	IETF CoRE Extension	No. of messages & latency	Simulation
[10]	Centralized & distributed	Layer	Protocol Stack	IETF CoRE Extension	-	Simulation
[11]	Centralized	Client-Server	Server	OCF RAML	-	Real world
[12]	Distributed	P2P	Network devices	IETF CoRE & JSON	-	Simulation & Real world
[13]	Distributed	P2P	Peer nodes	IETF CoRE & RELOAD	-	Simulation & Real world
[14]	Centralized	Hierarchy	Network devices	IETF CoRE Extension	-	-
[15], [16]	Distributed	Layer	Middleware	IETF CoRE & others	-	-
[17]	Centralized & Distributed	Client-Server	Network devices	IETF CoRE	-	Simulation
This paper	Centralized & distributed	Hybrid	Edge device clusters	IETF CoRE	Power consumption, latency & load balancing	Real world

A. Edge-supported Resource Discovery

The three-tier IoT system architectures rely on the devices on the edge layer to connect system components and transmit data upstream and control messages downstream. The edge devices are deployed as an integral part of the network infrastructure and co-located with powerful servers that form the edge computing infrastructure for edge applications. Now, from the mist layer perspective, it is natural to request support from the upper layer in hierarchy, where the role of an edge device can be extended to handle the resource discovery load.

Several advantages of such edge-supported mist resource discovery can be found. First, low latency for real-time discovery can be provided with RD instances placed at one-hop distance from the mist layer. Such RD instances, as a location-based service, provide on-demand resource discovery and facilitate resource provisioning and runtime reconfiguration of the mist platform functionality. In addition, connecting disparate mist networks, an important consideration for mist [1], becomes straightforward through the edge layer. The edge layer also has capacity to host additional domain- or application-specific RD instances.

Second, the edge servers have capacity for handling numerous resource queries collaboratively, with location-aware load balancing for discovery from all the tiers. Moreover, an edge device that connects mist devices to the edge layer, consequently encapsulates and represents the underlying mist platform resources and protects their security as a proxy.

Third, virtualized RD instances are straightforward to deploy and orchestrate as any other edge platform software component. When low overhead virtualization technologies, e.g. containers, are available, instances can be deployed also to the less powerful edge devices. Such loose coupling increases edge deployment scalability, flexibility and robustness. Importantly, low virtualization overhead of containers enables an additional on-demand deployment scheme with low-power edge devices that don't have the capacity to host full-size virtual machines.

B. Edge Service Implementation Considerations

For cloud, edge and fog application development, the Microservices paradigm has emerged from the need to modularize applications into a set of small distributed components, to the level of a single function, that then collectively implement the application logic [22].

The benefits of the paradigm are seen in the isolated development of the functionality, reuse, maintainability, extendability and straightforward life-cycle management [23]. Commonly, Microservices are deployed to the edge as self-contained package, where lightweight containers are the appropriate choice for running Microservices due to their low computational overhead and portability [24]. This way, the containerized Microservices provide more choice, flexibility and scalability for their deployment, e.g. can be hosted in devices with less capabilities, in comparison with virtual machine-based applications that typically require powerful servers. This is particularly beneficial for mist, as the layer is based on heterogeneous resource-constrained IoT devices that rely on optimized embedded operating systems, communication technologies and programming solutions.

Containers are deployed into the edge devices by the platform orchestrators, typically in response to an user request or to perform a management function such as load balancing. Here, Kubernetes is a well-known open-source platform for managing and orchestrating distributed software deployments across clusters of computing nodes [18]. Each cluster, i.e. set of worker nodes running containers, is managed by a master node that provides system services and controls the cluster, selects the worker nodes to run containers and facilitates interactions with the application and service clients. Worker nodes are responsible for running the supporting components for container execution, checking health and reporting the state of their containers. Kubernetes provides functionality for service discovery and load balancing across its domain in the

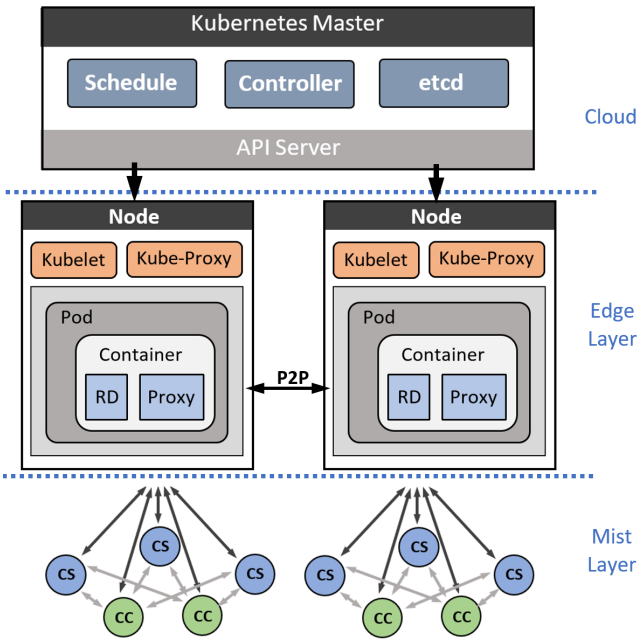

Fig. 1. Hybrid resource discovery architecture.

edge layer, but our focus is in the mist layer resource discovery under the edge devices.

Therefore, we expect that the Kubernetes worker nodes host the Microservice-based containerized RD instances as described in detail in the next Section.

C. Solution for Edge-supported Resource Discovery for Mist

Based on the discussion in previous Sections, we present a hybrid resource discovery solution for mist layer that relies on the support of the edge layer. The discovery architecture is hybrid, since both centralized and distributed approaches are considered simultaneously. Vertically in a bottom-up manner, an RD instance deployed into a local edge device at one-hop distance, such as network gateway or proxy, is a central point for resource discovery in the local mist platform. Similarly, in a top-down manner, the cloud and edge applications can discover the mist resources through the RD instances. Horizontally, the RD instances across the edge deployment form a distributed discovery infrastructure for the mist applications, enabling extended resource discovery and connections across disparate mist networks, where mist devices still use the services of a local RD instance. As in P2P systems, RD instances at the edge layer are capable of exchanging resource descriptions and queries, while balancing the discovery load in the edge platform is handled by the orchestrators.

Following the above ideas, we design the RD instances as containerized Microservices. Our starting point for low overhead resource registration and discovery is the IETF CoRE framework [4], including the CoRE RD and Proxy functionalities and CoAP protocol for interactions with the mist devices. To manage and deploy containerized RD instances, Kubernetes is employed as the edge platform management solution.

The Kubernetes-based deployment of the proposed hybrid discovery solution is illustrated in Figure 1. A Kubernetes master at the cloud layer configures and controls a cluster of worker nodes at the edge devices, which form the edge layer. Each worker node (a pod) hosts one RD and one Proxy instance containers. We assume that the deployed RD instances can interact with each other as a logical P2P network, as described in detail in our previous work [5]. At the mist layer, the devices are abstracted either as CoAP servers (CS) hosting resources and CoAP clients (CC) accessing the resources. The roles are interchangeable in each device for applications.

The resource registration and discovery process (Figure 2) is the following. First, the CS needs to locate a RD instance by sending a request, as CoAP GET message based on the CoRE Link Format, to the local edge node. The kube-proxy component at the edge node requests an instantiation of RD and Proxy Microservice instances. Once a local RD instance is available, the CS performs the registration for its resources with a CoAP POST request following the CoRE Link Format. Similarly, a RD instance can be located and instantiated on worker node by receiving a request from a CC.

Once the CS resources have been registered, a CC can perform discovery based on the Core Link Format to describe the query parameters, e.g. a sensor type or location. If a resource is found, its description is returned. With this information, the CC is able to directly interact with the CS by using CoAP. To discover additional system resources or resources in other mist networks, the request of a CC can be forwarded to other RD instances through the P2P overlay on the edge layer. Then, to interact with these remote resources, CC needs to send the CoAP request to a local Proxy that connects with another Proxy on the remote edge device. The CoRE Proxy is also capable for translating the request between protocols, e.g. CoAP to HTTP and vice versa, if required.

IV. Evaluation

To evaluate the presented resource discovery solution, a mist platform and a Microservice-based RD instance are implemented in a real-world prototype, managed by Kubernetes. To address the outlined mist computing challenges, the focus of the evaluation is on the discovery latencies and power consumption with both low resource mist and edge devices.

As the edge devices hosting the RD instances, and Kubernetes master and worker nodes, two Raspberry Pi (RPi, version 3 model B+) single-board computers are utilized. Such devices exemplify resource-constrained edge and fog devices, since the edge platforms are commonly based on powerful servers with tens of cores and tens of gigabytes of memory. Both edge devices were installed with Kubernetes (v1.16.1) and Docker (v18.09.7). The devices were connected through 10/100 Mbps router-switch. The CoRE RD instance was implemented in Python using the CoAPthon library [25] and MongoDB database for resource data storage. The corresponding Docker container image uses ready-made MongoDB image and installs python 2.7 along with required open-source libraries (e.g. pymongo and pyserial) that are needed to run

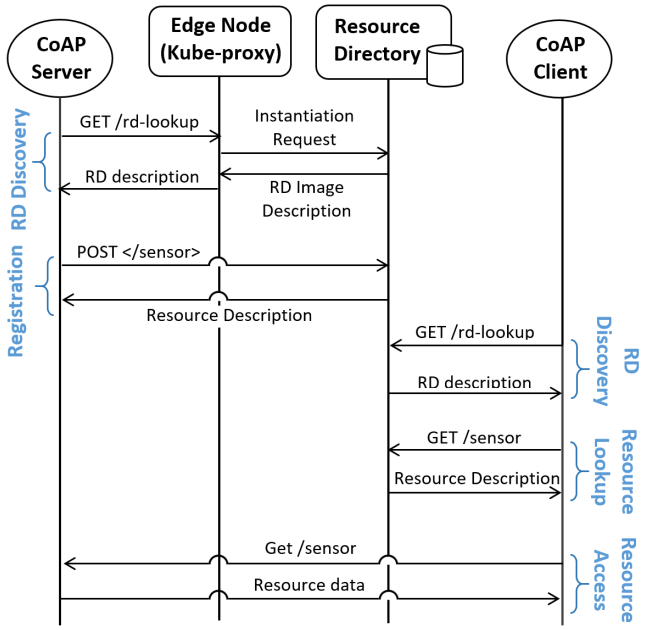

Fig. 2. Resource registration and discovery sequence.

the instance, resulting in a total container image size of 814 MB.

As the mist devices, ESP8266 embedded devices (clocked at 80 MHz with 4 MB of flash memory and 50 kB of RAM) operate as both CoAP servers and clients. The devices communicate atop a public WiFi network with UDP as expected in the CoRE framework. The servers provide inertial measurement sensor data as their resource.

A. Real-world experiments

First, to evaluate the latencies of resource discovery atop the presented real-world prototype, a CC sent 100-500 successive resource queries (as CoAP messages) to the RD instance running in a Kubernetes worker node in a RPi. Figure 3 illustrates the average response time as a function of the number of consecutive queries. The results show that the response time is less than half a millisecond for up to 500 successive requests per second. However, during the measurements, a query and response message loss from 2 to 9% was observed at the rates from 200 to 500 requests per second. In the experiments, the RD instance implementation is not optimized in any way, but the software components were installed into the container image as is.

Second, to evaluate the power consumption of running containerized RD instance on a RPi, an additional set of experiments were conducted. First, the power consumption of the worker node running a RD instance Docker image (Idle RD) without any requests was measured. The results are shown in Figure 4 and at Table II. It was observed that the power consumption overhead of an idle RD instance is insignificant in relation to the base RPi power consumption.

Then, the power consumption of resource queries was measured with different discovery request rates. Here, 10

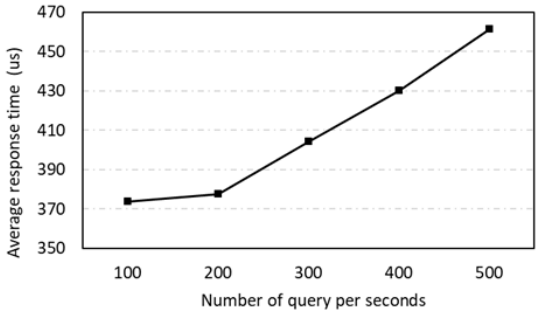

Fig. 3. Resource query latencies.

TABLE II
EDGE DEVICE POWER CONSUMPTION MEASUREMENTS

Power Consumption (mW)		
Measurement	**Container**	**Bare-metal**
Idle RD	2148	2102
Active RD (10 req/sec)	2278	2164
Active RD (100 req/sec)	2361	2283

and 100 queries per second exemplify a relatively busy RD instance. As shown in Figure 4 and in Table II, a significant difference in power consumption is observed.

Lastly, to see the overhead of hosting an RD instance with and without Kubernetes, similar set of measurements was repeated with RD implementation on a "bare-metal" RPi running Debian linux.

B. Analysis of results

The evaluation with low resource edge devices shows that latency (i.e. response time) of less than half a millisecond is achieved up to 500 requests per second when both the CC and CS are in the same mist network. Ideally, such request latency means that the RD instance is capable of handling around two thousand queries per second. However, due to the constrained networks, a significant message loss is experienced (up to 9%), which reduces the request handling throughput in the prototype.

In comparison with the results presented in the previous studies with IoT resource discovery, the results demonstrate lower query latencies. In [5], with CoRE RD-based resource discovery in P2P network, latencies of different types of queries between 20 - 120 ms were reported in a real-world prototype. In addition, the effect of caching of query results was studied in [5], [6], resulting 97% reduction (from 640 ms to 20 ms) in query latency. Web browser based CoRE RD semantic resource discovery latencies in the range of 10 ms were reported in [8]. With contextual resource queries, latencies from 200 to 2500 ms based on simulation were reported in [9]. The authors in [10] evaluated both centralized and distributed schemes by simulation with extended CoAP and reported latencies between 15 to 150 ms with request rates from one to 20 per minute. With structured document-based resource descriptions, query latencies around 400 ms were reported in a real-world system [11]. In [13] resource

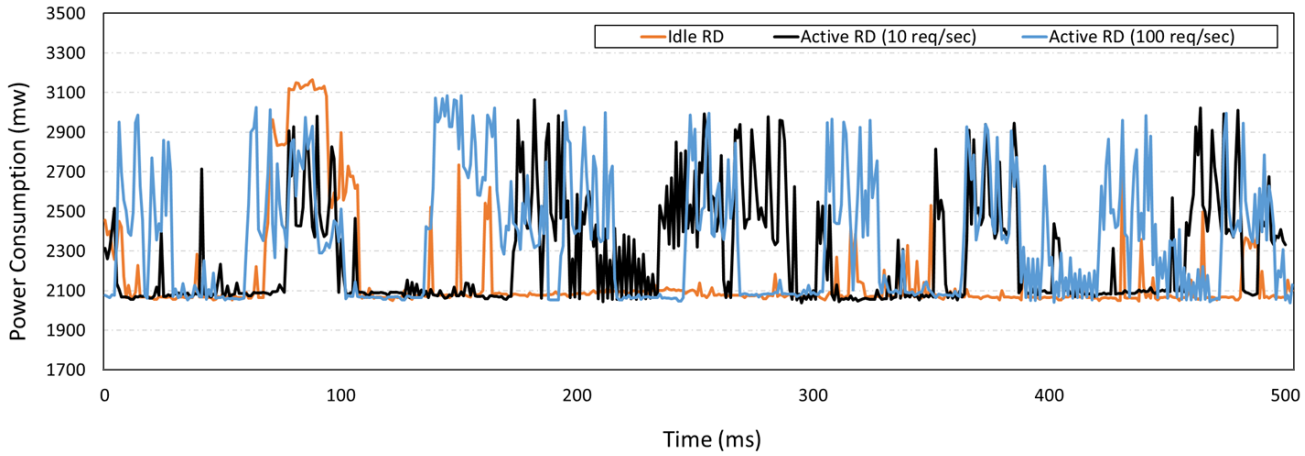

Fig. 4. Power consumption of the edge device (i.e. Kubernetes worker node) in different scenarios.

lookup delays in the range of several seconds in a real-world prototype were reported, but including the communication setup time. Lastly, simulated query latencies above one second were reported in [17].

Regarding the power consumption measurements, the following can be said. With 10 requests per second (Table II), the power consumption of an active container-based RD instance increases around 6% in comparison with an idle RD instance. With 100 requests per second, the increase is around 10% in comparison with the idle RD instance. The management overhead Kubernetes, for an active containerized RD instance in a RPi in comparison with a bare-metal RD instance, increases its power consumption by 3.4% (78mW) as shown in Table II when handling 100 requests per second. In the previous work, only the authors in [7] evaluated and reported energy savings, based on optimized advertising of resource descriptions and duty cycling of the device radio, but similarly with a loss in the discovery ratio of the system resources.

Overall, in the presented solution, the resource discovery latencies are very low with high request rate and the total power consumption of an active RD instance is around 2.36W. Such results show that the solution is feasible for resource discovery in a mist computing platform even with low resource edge devices hosting the RD instances.

V. Discussion

In this paper, an edge-supported hybrid resource discovery solution for mist computing, based on containerized Microservices and standardized IETF CoRE framework, is presented. The RD instances form a P2P overlay at the edge layer that enables resource lookups between disparate mist networks and information sharing for applications both horizontally and vertically across the deployment. Such RD instances, implemented as containerized Microservices, improve scalability as on-demand deployment of the RD instances in edge and fog deployments is straightforward with existing edge computing platforms, such as Kubernetes. In comparison with existing IoT resource discovery solutions, virtualization of the RD functionality avoids the issues of fixed physical deployments and offloads the computation and communication load of the resource discovery from mist devices to the more powerful edge devices and servers, at one-hop distance.

The presented solution was evaluated with a real-world prototype. The results show that low resource edge devices, exemplified by RPi's, are able to host such containerized RD instances with very low query latencies while serving several hundred queries per second. In addition, power consumption measurements of the active RD instance were conducted in real-world settings, providing results that have not yet been presented in the previous work.

For the edge platform hosting the RD instances, the P2P interaction overhead and latencies on the core network can be considered negligible, as providing sufficient computational and networking capacity for services on the edge layer is the fundamental rationale for edge computing.

However, when mist platforms scale up, it remains an open question how well such Microservice-based collaborative solutions are able to handle queries from thousands of devices in real-time. Moreover, it can be envisioned that the resource discovery load largely fluctuates in relation to the user movement and resulting online application relocation (instantiation and termination) across the edge and fog deployments.

VI. Conclusion

Resource discovery solutions for IoT systems have attracted attention in recent years. However, a novel distributed computing paradigm for IoT, mist computing, lacks solutions that address the challenges in dynamic resource availability, where a variety of low power technologies are to be utilized with mist devices. This paper shows that energy efficient low latency resource discovery solutions, targeted for mist computing, can be achieved with the support of the edge layer that facilitates both vertical and horizontal interactions for applications across the IoT system deployments. In general, the presented hybrid solution is also applicable for fog computing platforms, where

low resource devices are utilized at the layers of the fog hierarchy.

Edge computing is currently considered an integral part of the 5G standardization efforts, where our future work aims to the development of massive-scale lightweight resource discovery solutions in the context of the foreseen mist computing paradigm.

ACKNOWLEDGEMENTS

This work is supported by the Academy of Finland 6Genesis Flagship (grant 318927), the Infotech Oulu Research Institute and the Jane and Aatos Erkko Foundation and the Technology Industries of Finland Centennial Foundation.

REFERENCES

[1] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[2] F. Marino, C. Moiso, and M. Petracca, "Automatic contract negotiation, service discovery and mutual authentication solutions: A survey on the enabling technologies of the forthcoming iot ecosystems," *Computer Networks*, vol. 148, pp. 176–195, 2018.

[3] C. N. Ververidis and G. C. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, 2008.

[4] Z. Shelby, "Embedded web services," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 52–57, 2010.

[5] M. Liu, T. Leppnen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, and T. Ojala, "Distributed resource directory architecture in machine-to-machine communications," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013, pp. 319–324.

[6] M. Liu, T. Leppänen, E. Harjula, Z. Ou, M. Ylianttila, and T. Ojala, "Distributed resource discovery in the machine-to-machine applications," in *10th International Conference on Mobile Ad-Hoc and Sensor Systems*. IEEE, 2013, pp. 411–412.

[7] B. Djamaa, A. Yachir, and M. Richardson, "Hybrid coap-based resource discovery for the internet of things," *Journal of Ambient Intelligence and Humanized Computing*, vol. 8, no. 3, pp. 357–372, 2017.

[8] Y. Wang and G. Wei, "An implementation of coap-based semantic resource directory in californium," in *International Conference on Artificial Intelligence and Security*. Springer, 2019, pp. 211–222.

[9] F. M. Barreto, P. A. d. S. Duarte, M. E. Maia, R. M. d. C. Andrade, and W. Viana, "Coap-ctx: A context-aware coap extension for smart objects discovery in internet of things," in *IEEE 41st Annual Computer Software and Applications Conference*, vol. 1, 2017, pp. 575–584.

[10] B. Djamaa, M. A. Kouda, A. Yachir, and T. Kenaza, "Fetchiot: Efficient resource fetching for the internet of things," in *Federated Conference on Computer Science and Information Systems*. IEEE, 2018, pp. 637–643.

[11] W. Jin and D. Kim, "Consistent registration and discovery scheme for devices and web service providers based on raml using embedded rd in ocf iot network," *Sustainability*, vol. 10, no. 12, 2018.

[12] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, and L. Veltri, "A scalable and self-configuring architecture for service discovery in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 508–521, 2014.

[13] J. Mäenpää, J. Bolonio, and S. Loreto, "Using reload and coap for wide area sensor and actuator networking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, p. 121, 2012.

[14] I. Ishaq, J. Hoebeke, J. Rossey, E. De Poorter, I. Moerman, and P. Demeester, "Facilitating sensor deployment, discovery and resource access using embedded web services," in *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2012, pp. 717–724.

[15] S. K. Datta and C. Bonnet, "Search engine based resource discovery framework for internet of things," in *4th Global Conference on Consumer Electronics*. IEEE, 2015, pp. 83–85.

[16] S. K. Datta, R. P. F. Da Costa, and C. Bonnet, "Resource discovery in internet of things: Current trends and future standardization aspects," in *IEEE 2nd World Forum on Internet of Things*, 2015, pp. 542–547.

[17] R. Ferdousi, M. Helaluddin, A. Akther, and K. M. Alam, "An empirical study of coap based service discovery methods for constrained iot networks using cooja simulator," in *20th International Conference of Computer and Information Technology*. IEEE, 2017, pp. 1–6.

[18] C. Tracey and B. Burns, *Managing Kubernetes*. O'Reilly, 2018.

[19] C. Rotter, J. Illés, G. Nyíri, L. Farkas, G. Csatári, and G. Huszty, "Telecom strategies for service discovery in microservice environments," in *20th Conference on Innovations in Clouds, Internet and Networks*. IEEE, 2017, pp. 214–218.

[20] F. Montesi and J. Weber, "Circuit breakers, discovery, and api gateways in microservices," *arXiv preprint arXiv:1609.05830*, 2016.

[21] C. Perera and A. V. Vasilakos, "A knowledge-based resource discovery for internet of things," *Knowledge-Based Systems*, vol. 109, pp. 122–136, 2016.

[22] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," in *21st International Conference on Emerging Technologies and Factory Automation*. IEEE, 2016, pp. 1–6.

[23] H. Kang, M. Le, and S. Tao, "Container and microservice driven design for cloud infrastructure devops," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2016, pp. 202–211.

[24] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures–a technology review," in *3rd International Conference on Future Internet of Things and Cloud*. IEEE, 2015, pp. 379–386.

[25] G. Tanganelli, C. Vallati, and E. Mingozzi, "Coapthon: Easy development of coap-based iot applications with python," in *2nd World Forum on Internet of Things*. IEEE, 2015, pp. 63–68.