An Optimized Discovery Mechanism for Smart Objects in IoT

Mohammed Mahyoub, Ashraf Mahmoud, Tarek Sheltami
Computer Engineering Department,
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
{*g201405280, ashraf, tarek*}*@kfupm.edu.sa*

***Abstract*—Smart objects (SOs) have been utilized widely to transform the physical environment around us to a digital world using the Internet of things (IoT) vision. Integrating a huge number of these devices into the Internet presents a significant necessity for an efficient discovery mechanism with high capability of an autonomous configuration and detection for theses devices and their provided services. Instead of developing new protocols to provide such functionalities, community resorted to adapting already existing protocols such as multicast Domain Name System (DNS) and DNS-Service Discovery (DNS-SD) as they are the most extended discovery protocols for Internet architecture nowadays. Since these protocols have been designed mainly for the ordinary computational devices without considering of the low capability constraints of SOs, it is not applicable to apply them directly on the SOs. To this end, this work proposes an optimization approaches to minimize the overhead caused by some operations of mDNS/DNS-SD protocols. Probing and advertisement suppression (PAS), discovery responses suppression (DRS) approaches and selection stage optimization (SSO), namely, are proposed and explained in this paper.**

***Index Terms*—IoT, LLNs, Discovery, Multicast, Smart Objects, Zero-configuration.**

I. Introduction

The smart objects (SOs), with varying levels of complexity, have been utilized to connect the physical environment around us to the Internet known as the Internet of Things (IoT) vision. Having said that, using tiny IP stack (uIP for short), IoT enables the SOs to connect, share information, perform jobs cooperatively with each other and with the connected ordinary computational devices [1].

This connectivity provided by IoT transforms the standalone applications into cooperative ones. Several applications therefore have been created in many different domains and use cases such as home automation, smart grids, mobile health-care, medical aids, elderly assistance, automotive, traffic management, industrial automation, and many others [2].

Having equipped SOs with sensors and usually deployed in large scale, they form networks known as Low Power and Lossy Networks (LLNs). LLNs is a class of network in which the devices and their interconnections are constrained in terms of the memory capacity, processing speed capabilities, energy as they are mostly a battery-based powered, low data rates, high faulty links, and instability such as IEEE 802.15.4 radio link [3].

The LLNs applications in an IoT vision can provide many important daily used services for the citizens and the communities. The most notable ones are those directed to monitoring and controlling purposes. Monitoring the body heath [4], buildings , atmospheric phenomena [5], [6] such as pollution levels, temperature and humidity, places noise [7], and smart parking [8] are some of these services.

The successful deployment of LLNs applications, especially in highly dynamic scenarios, requires two main important aspects, described as follows. One is the SOs capability to adapt and configure themselves automatically to the surrounding environment without any human intervention. The other one is the awareness and detection of the available services provided by the SOs in an autonomous fashion [9].

To deal with the two aspects mentioned above, the existing Internet infrastructure needs two key takeaways. The first one is a self-configuration process for network enabling to deal efficiently with the predicted huge number of SOs emphasized by the IoT vision. The second one is a standardized process to detect and discover both joining SOs and provided services autonomously while satisfying a backward compatibility with the existing Internets standards [10]. This work will call the second process and the combination of the above two aspects as a service discovery (SD) and a discovery mechanism, respectively.

A constrained application (CoAP) protocol operating at application layer has been developed to provide low message overhead, meeting the capabilities of constrained SOs. Unfortunately, the lacking of fully backward compatibility with the widely used Internet standards is the main drawback of this protocol as it highly depends on a special translation protocol, leading to a loss of end-to-end functionality, flexibility and scalability [10].

Instead of developing new protocols to handle the discovery mechanisms for SOs, it is favored to adapt already existing protocols to provide such functionalities. Nowadays, multicast Domain Name System (mDNS) [11] and DNS-Service Discovery (DNS-SD) [12] are the most extended discovery protocols for Internet architecture.

The mDNS protocol and its companion technology DNS-SD have been creating to provide an IP networking and participants discovering with ease-to-use and auto-configuration strategies. Furthermore, these protocols offer a good scalability, light memory footprint, and they are wide used since the DNS's programming interfaces, and operating semantics, packet formats are utilized. The re-use of the DNS protocol is beneficial in many-folds, the most notable of which are the followings: the deployment and configuration of a separate discovery system could be avoidable, DNS is widely used for many years in all sizes networks, and DNS has gained an exceptional popularity.

Despite the mDNS/DNS-SD advantages mentioned above, indeed, both of them were initially designed and implemented for ordinary computational devices in local networks with nearly no limit of network bandwidth, memory storage, and processor speed capability. To this end, the existing implementation of mDNS/DNS-SD (sometimes called as Bonjour [13]) is not directly applicable to operate in networks of resource constrained devices such as SOs in LLNs [14].

To remedy this issue, the literature has introduced several lightweight versions of these protocols by re-implementing their codes to function on constrained SOs over LLNs. The most common implementation of these is uBonjour [15]; reimplementation of Bonjour code in an optimized way on top of uIP for the Contiki OS [16]. Some optimization approaches have been applied on uBonjour to meet the constrained capabilities of SOs in LLNs, but still some operations causing an overhead need to be optimized.

In this work, the basic operations of mDNS/DNS-SD protocols are investigated, and critical optimization approaches to reduce the overheads caused by these operations are proposed. Having provided SOs services over distributed devices in LLNs, the wireless communication significantly contributes in the energy consumption. For this, minimizing the network overhead is an indispensable requirement; as a result, the proposed approaches mainly focus on reducing the communications overhead to enhance the network traffic and power consumption.

The rest of this paper is structured as follows.Section II presents a detailed description of the mDNS/DNS-SD protocols and their operational phases and functions. The related work is shown in section III while the optimization classes and proposed optimization approaches are presented in section IV. The final remarks concluding this work are shown in section V.

II. An mDNS/DNS-SD Protocols

Many operating systems already have zero-configuration systems used to discover services such as print services, file share, ssh and web-servers advertised by network devices running mDNS. Examples of zero-configuration systems include Avahi [17], jmDNS [18] and Bonjour [13]. These zero-configuration systems provide the discovery mechanism of non-constrained devices (e.g., PCs, smartphones and netbooks) and their provided services communicating over IP networking. The zero-configuration systems mentioned above are implementation of mDNS/DNS-SD protocols for the Linux/BSD/OpenWRT/Android, Java, MAC/Windows/iOS OS, respectively. uBonjour [15], on the other hand, is a minimized Bonjour-based implementation of mDNS/ DNS-SD for constrained devices over Contiki OS with Internet protocol (IP) support.

Fundamentally, automatic fulfilling the network configuration and discovery management using mDNS/DNS-SD protocols can be accomplished by two phases known as a zero-configuration phase and discovery phase, respectively. Figure 1 shows the taxonomy of mDNS/DNS-SD operational phases and the operations performed by each of them. This section will discuss these phases and how they are working towards accomplishing the tasks of mDNS/DNS-SD protocols.

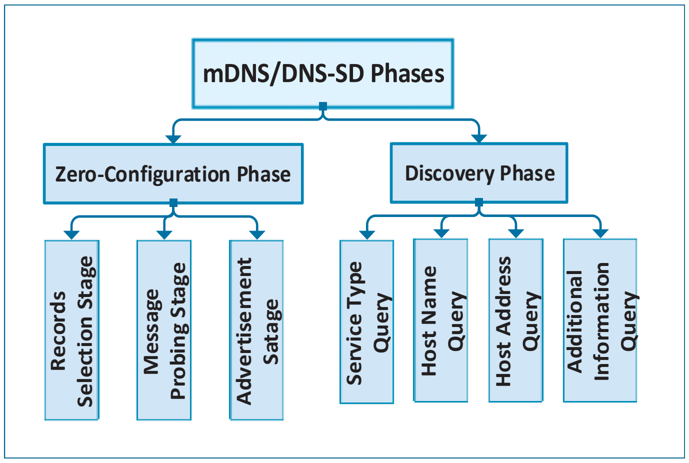

Fig. 1. An mDNS/DNS-SD Operational Phases

A. *The mDNS/DNS-SD Zero-configuration Phase*

The Zero-configuration phase of mDNS/DNS-SD takes care of enabling the networking in SO and assigning the unique names and descriptions for the services hosted by the SO.

Whenever an mDNS SO starts up, wakes up from sleep, receives an indication of a network interface ”link change” event, or has any other reason to believe that its network connectivity may have changed in some relevant way, it must perform the stages of this phase which are record selection, probing, and advertisement stage [12].

The flowchart in Fig. 2, shows the operational flow performed by each new joining SO and by the other SOs already connected to the local link network as well. They are represented by *joining node* and *mDNS respondents* in the left and right side of that flowchart, respectively.

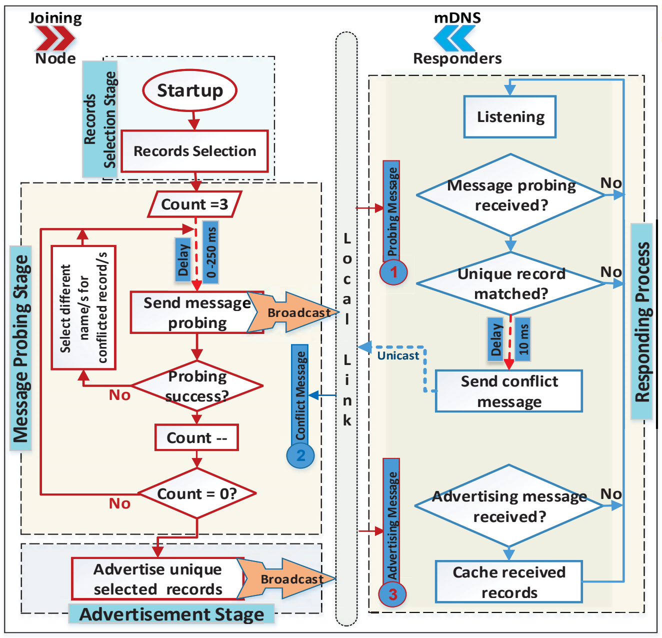

Fig. 2. An mDNS/DNS-SD Zero-configuration Phase

The first step of zero-configuration phase is that, for all those resource records (RR) that a joining mDNS SO desires to be unique on the local network, it selects them using whatever selection method such as that has been presented in RFC 4862 [19] and this step is called records selection stage in figure 2.

Secondly, the joining SO must sends an mDNS query asking for those RRs, to see if any of them are already in use by another SO in the local network since this step is called probing stage. The primary example of this is a hosts address records, which map its unique host name to its unique IPv4 and/or IPv6 addresses. When a DNS-SD service is advertised using mDNS, automatic name conflict and resolution will occur if there is already another service of the same type advertising with the same name.

Finally, If no conflicting responses received from the mDNS respondents for three consecutive probing times, the joining node advertises these unique records to the networks since this step known as advertisement stage. Otherwise it must select another names (e.g. append or increment a digit at the end of the name) for the conflicting RR and perform the probing stage again.

The respondents, on the other hand, store the advertised RRs about the new joining SO if there is an available capacity in their caches in case they need to contact that node later on.

B. The mDNS/DNS-SD Discovery Phase

Given a service's type and a domain name in which the SO is looking for a service, mDNS/DNS-SD allows the SO to discover a list of named instances of that desired service using DNS-SD queries. Figure 3 shows the needed operations to discover some services in the local network and the queries and their responses to accomplish these operations are shown as well.

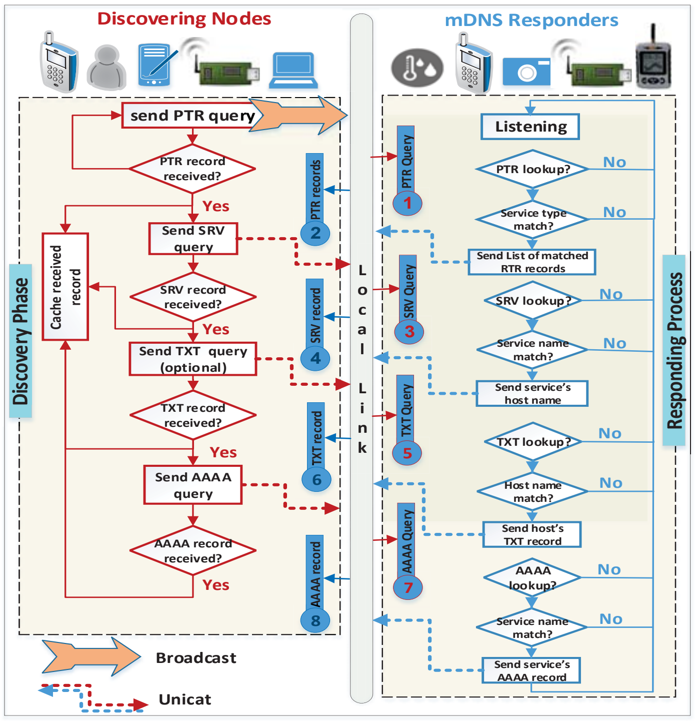

Fig. 3. An mDNS/DNS-SD Discovery Phase

The SOs can discover DNS-SD service instances by querying the PTR records with a name of the service type. The result is a DNS response message containing a set of zero or more PTR records listing matching service instance names. The services' names contained in the response for the PTR lookup are listed for the user to choose one or possibly more.

If a discovering SO decides to contact a particular service, it sends a query for its SRV record. The SRV record gives back the port number, service type, and hostname of the hosting SO where the service resides. It also contains a The priority and weight parameters, to give preference if the same service is hosted in multiple SOs, are contained on SRV record as well.

The additional service information is conveyed via the TXT record in a key-value pair format in which the exact key-value pairs are protocol dependent. For instance, a Uniform Resource Identifier (URI) path of a CoAP resource might be included in the TXT record if DNS-SD is used to discover CoAP services. Finally, to resolve the hostname to its IP address, a query for the A/AAAA record is made.

III. Related Work

uBonjour [15] enables a standardized SD of SOs with the same Internet mechanisms which are already used for ordinary computational devices. The general features of uBonjour for application protocols and developers are the support of self-configuration instead of hard-coded addresses, registration, removal, the discovering, and updating of services, and the resolving of hostnames.

Some approaches to optimize the uBonjour are discussed by Klauck and Kirsche, the authors of uBonjour themselves, in [20]. The optimization approaches they have proposed and implemented were the following.

The first approach is the *DNS Message Compression*, as recommended by the IETF's mDNS RFC 6762 [11], in which the size of a DNS's RR is reduced by shortening the names included in RRs using pointers to a prior occurrence of the same name.

The another approach is *Class Code and TTL Field Compression* since the class code and TTL values occur repeatedly when a single DNS message includes multiple DNS RRs. For this, the pointer can be used to indicate the prior occurrence of both fields to save some bytes.

Another optimization to reduce the DNS message overhead is published theoretically by Jara et al. [14]. The redefining of the TXT RR format is proposed to reduce the number of IEEE 802.15.4 frames per DNS RR. To do so, multiple TXT entries have been summarized into a single entry and the Constrained RESTful Environments (CoRE) link format [21] and Lempel-Ziv algorithm have been used for description and data compression, respectively.

The drawbacks of Jara et al [14] work are: (i) The incompatibility with DNS standard where a single record of information is expected per TXT record introducing the necessity for SO's specific code, and (ii) Its dependency on the external algorithm for the compression purpose producing a high memory usage which is not suitable for constrained devices such as SOs.

In addition to the optimization approaches mentioned above, some other approaches have not been implemented yet. Table I summarizes the un-implemented and non-evaluated proposed optimization approaches for mDNS/DNS-SD protocols.

TABLE I
Summery of Literature Unimplemented Proposed Optimization

Unimplemented Proposed Optimization	Work
Multipacket Known-Answer Suppression	RFC 6762 [11]
Duplicate Question Suppression	RFC 6762 [11]
Duplicate Answer Suppression	RFC 6762 [11]
Redundant Information Filtering	Klauck et al. [20]

IV. Proposed Optimization

A. *The General Optimization Classes for SOs in LLNs*

Optimization task for the traditional Internet protocols enables them to be operated efficiently on SOs over LLNs. To do so, the developers may use different forms accomplishing this task. In this section, we define the most important optimization categories.

● **MEmory Footprint-based Optimization (MEFO)**
Having SOs provided with a few Kbytes of memory (i.e. RAM and ROM), efficiently utilizing the available memory is one of the most important class of optimization. Reducing the application's code size installed in the SO memory is the main purpose of MEFO, in which either the unwanted operations causing an overhead is disabled as possible, or the coding way itself is managed efficiently. This kind of optimization enables the other applications to reside in the limited memory along with the operating system installed in the SOs, resulting in better utilization for these objects.

● **COMputational-based Optimization (COMO)**
Most SOs have limited computational power [22], therefore, heavy computation procedures cause an overhead in terms of delay due to lack of processing speed and increasing in power consumption as well. To mitigate this overhead, the computations load by SO's processing units needs to be reduced as much as possible.

● **NEtwork Traffic-based Optimization (NETO)**
Low data rate and faulty links are among the main features of LLNs where the SOs are deployed. In LLNs, the number of messages transferred between nodes have to be minimized as much as possible to avoid consuming the limited bandwidth resulting in high loss rate which in turn increases re-transmission to guarantee packet delivery which aggravates the situation even worst. More transmitted data, more power drained out causing in decrease the overall network life time. NETO is working towards decreasing the unneeded packets transmissions either by suppressing multiple packets (i.e. query or response messages in case of mDNS/DNS-SD protocol) in less packets as much as possible or cancelling some insignificant operations causing an overhead.

B. *Proposed Optimization Approaches for mDNS/DNS-SD*

■ **Selection Stage Optimization**
In the original procedure of mDNS/DNS-SD protocols, upon every start-up, SOs must select their unique RRs

such as services names, additional information and IPs addresses. Performing the selection process frequently causes two overheads of which are the delay due to the lack of computational power of SO processing unit, and the power consumption for computational needs.

One optimization way to remedy these overheads is using the unique RRs selected on the previous start-up. We call this approach as a selection stage optimization (SSO) since this kind of optimization is belonging to the COMO optimization category. The SSO method requires that the SO's records to be stored in a persistent storage, so the selection process is not needed on every start-up unless the message probing step get failed. Figure 4 shows a sketch of optimized RRs selection stage.

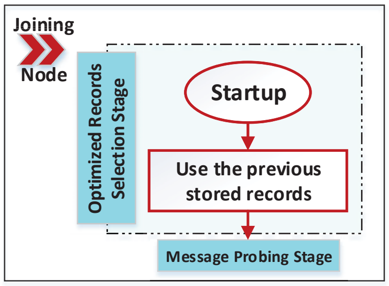
Joining
Node
Optimized Records Selection Stage
Startup
Use the previous stored records
Message Probing Stage

Fig. 4. Optimized DNS-SD RRs selection

■ Probing and Advertisement Stages Optimization
As mentioned above, in zero-configuration phase section described in II-A, four broadcasting messages are needed in which three of them for the message probing and one for the advertisement process. To mitigate the overhead caused by these two stages, as proposed in this work, the respondents can use the RRs propagated by massage of probing stage to keep the information about the new joining SO instead of sending a separate message for the announcement purpose. The optimization proposed in this section is categorized under NETO optimization class. Figure **??** shows a combined probing and advertisement stages optimization where the proposed Probing and Advertisement Suppression (PAS) algorithm is performed by the respondents to achieve a proper functional combination.

Whenever the respondent receives a probing message, either the conflicting message is replied back and then the proposed PAS is performed if a match occurs or otherwise the PAS algorithm is performed directly. Algorithm 1 and Table II show a pseudo code for PAS algorithm and the meaning of the used symbols in that algorithm, respectively.

PAS algorithm works as the following, firstly, the SO's cache is checked for availe cache entries to store the new joining SO's RRs received by probing message ($Line1$).

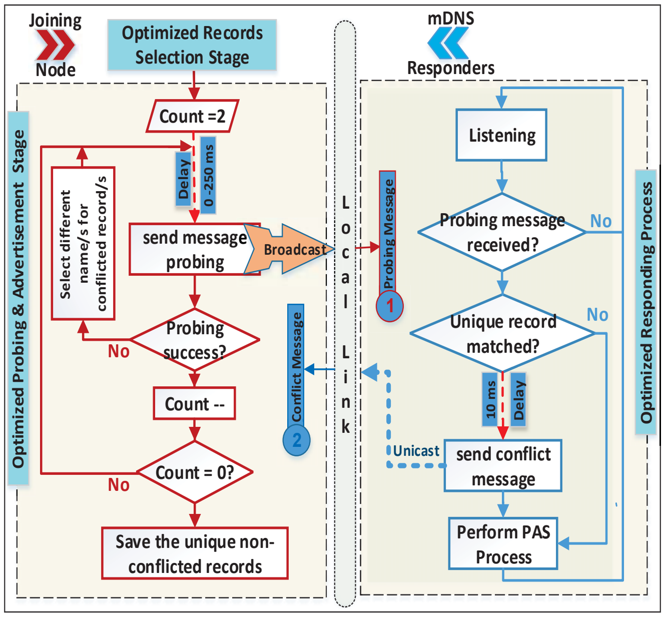

Fig. 5. An optimized probing and advertisement stages

Algorithm 1 Probing and Advertisement Suppression

Input: S, N, C, P
Output: Service's records resolution and caching

```
    when new probing message is received :
1:  if $N < S$ then
2:    for all $id$ in $C$ do
3:      if $(P_{qid} = C_{id})$ then
4:        modify $C_R$ corresponding to $C_{id}$
5:      else
6:        add $P_R$ to $C$
7:        $N++$
8:      end if
9:    end for
10: end if
```

TABLE II
SYMBOLS AND MEANING

Symbol	Meaning
C	Cache entries array
S	Cache size (Entry unit)
N	Number of entries in C
P	Probing message
C_R	Records of specific entry in C
P_R	Probing message records

If the cache is not full, probing message's query ID is searched to match one of the IDs already stored in the cache ($Line3$). If there is a match, that means a previous SO's probing get failed, then the respondent needs to modify the query ID-related RRs in its cache ($Line4$), otherwise the received RRs will be added to the cache silently ($Line6$). With PAS, the probing message is used by respondent to keep the new joining SO's information instead of need to broadcast a separate message by joining SO for that purpose. In addition to the optimization of combining the probing and advertisement stage, the number of messages used by probing stage is proposed to be reduced to two messages, of course, in case of successful probing. The author in paper [14] proposed to perform the probing stage only one time and they get good results. Two times probing is more enough to guarantee that all the SOs in LLN get the probing message.

■ Discovery Phase Optimization

As mentioned in section II-B, four queries and four responses are needed whenever the SO wants to discover the contact information related to specific service provided in the surround network. Two overheads raised due to these transactions are: (i) memory footprint (MF) overhead in which handling DNS records query and generating the responses consume a most size of mDNS/DNS-SD source code, and (ii) the overhead related to the limited bandwidth where these SOs are deployed.

To overcome these overheads, it is proposed to suppress multiple responses as one message. After the discovering SO getting the PTR RRs response including the list of services related to a the interested specific type, all the needed information related to the interested service (i.e. SRV, TXT and AAAA records) is provided by the service's hosting node as one message instead of asking each record individually.

Figure 6 shows the operations performed in optimized mDNS discovery phase.

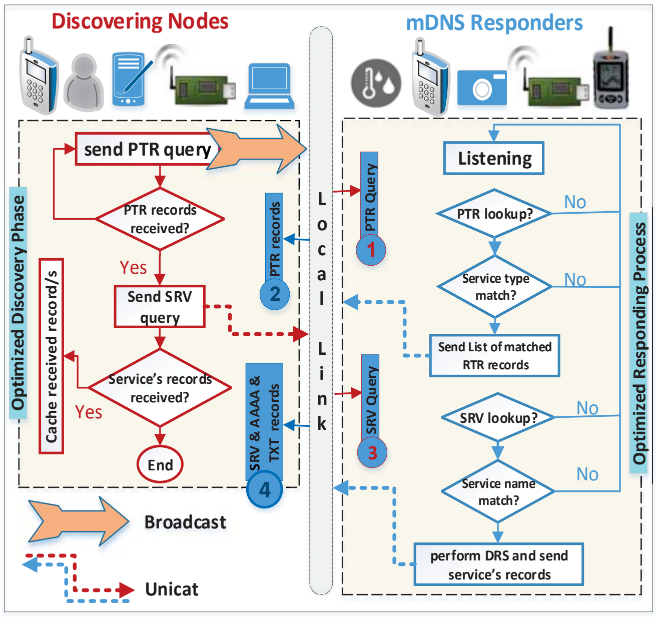

Fig. 6. An optimized mDNS/DNS-SD discovery phase

This optimization approach is defined as a discovery responses suppression (DSR) in which multiple records are combined to be sent in one message and this kind of optimization satisfy the MEFO and NETO optimization classes. This principle was being inspired by the discovery system of CoAP protocol [23], in which the CoAP server replies with all needed information for the request issued by the CoAP client (i.e. discoverer). On the other hand, the compression strategy recommended by mDNS's IETF RFC enables proposed DRS to fit multiple RRs in one message.

A good news is that, the compression method is already discussed by Klauck and Kirsche in [20] and also implemented by integrate the message compression as an enhancement for uBonjour in uIP stack over Contiki OS.

V. Final Remarks

Since the smart objects are mostly networked in various environments and adapt themselves automatically, centralized and pre-configuration networking techniques are not effective. Toward this end, distributed and autonomous discovery mechanism for the SOs is one of the important aspects in IoT paradigm. The existing widely used discovery mDNS/DNS-SD protocols require an optimization to fit and operate efficiently given limited resource capabilities of SOs. In this work, the mDNS/DNS-SD operational phases are investigated and explained. Additionally, three approaches are proposed to minimize the overhead caused by some of mDNS/DNS-SD operations, which are PAS, DRS, and SSO. With PAS approach, probing and advertisement stages are proposed to be merged as one stage without sacrificing the correctness of their functions. The DRS approach proposes suppressing multiple responses to be sent as one message instead of sending them individually. Finally, SSO proposes to use, at the first attempt, the unique records selected from the previous start-up and if this attempt fails then the default method of mDNS/DNS-SD protocols is used.

References

[1] K. Ashton, "That 'Internet of Things' Thing," *RFID Journal*, 2009.
[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.
[3] T. Winter, P. Thubert, A. R. Corporation, and R. Kelsey, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Tech. Rep., 2012.
[4] J. P. Lynch and K. J. Loh, "A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring," *The Shock and Vibration Digest*, vol. 38, no. 2, pp. 91–128, 2006.
[5] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.
[6] A. R. Al-Ali, I. Zualkernan, and F. Aloul, "A mobile GPRS-sensors array for air pollution monitoring," *IEEE Sensors Journal*, vol. 10, no. 10, pp. 1666–1671, 2010.
[7] E. Maria, N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, L. Steels, and E. Maria, "Citizen Noise Pollution Monitoring," *The Proceedings of the 10th International Digital Government Research Conference*, pp. 96–103, 2009.
[8] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," *2008 International Symposium on Collaborative Technologies and Systems*, pp. 48–57, 2008.
[9] W. Edward and C. Richard, "What Can I Do Here ? IoT Service Discovery in Smart Cities," 2016.

[10] F. Mattern and C. Floerkemeier, “From the internet of computers to the internet of things,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6462 LNCS, pp. 242–259, 2010.
[11] “Multicast DNS, RFC,” 2013.
[12] S. Cheshire and M. Krochmal, “DNS-Based Service Discovery, RFC,” 2011.
[13] G. Kaindl, “Bonjour/Zeroconf with Arduino.” [Online]. Available: https://developer.apple.com/bonjour/
[14] A. J. Jara, P. Martinez-Julia, and A. Skarmeta, “Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the web of things,” *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*, pp. 731–738, 2012.
[15] R. Klauck and M. Kirsche, “Bonjour Contiki: A case study of a DNS-based discovery service for the internet of things,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7363 LNCS, pp. 316–329, 2012.
[16] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors,” *IEEE EMNETS, Tampa, Florida, USA*, 2004.
[17] The Avahi team, “Avahi-daemon.” [Online]. Available: https://www.avahi.org/doxygen/html/
[18] A. van Hoff, “jmDNS.” [Online]. Available: http://jmdns.sourceforge.net/
[19] “IPv6 Stateless Address Autoconfiguration, RFC 4862,” 2007.
[20] R. Klauck and M. Kirsche, “Enhanced DNS Message Compression - Optimizing mDNS / DNS-SD for the Use in 6LoWPANs,” no. March, pp. 1–6, 2013.
[21] Z. Shelby, “Constrained RESTful Environments (CoRE) Link Format, IETF, RFC 6690,” Tech. Rep., 2012.
[22] J. Chen, S. Li, H. Yu, Y. Zhang, R. Dipankar, R. Ravindran, H. Gao, L. Dong, G. Wang, and H. Liu, “EXPLOITING ICN FOR REALIZING SERVICE-ORIENTED COMMUNICATION IN IOT,” *IEEE Communications Magazine Communications Standards Supplement*, no. December, pp. 24–30, 2016.
[23] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP), RFC,” Tech. Rep., 2014.