applied sciences

MDPI

Article

DiRPL: A RPL-Based Resource and Service Discovery Algorithm for 6LoWPANs

Luca Davoli [1,†], Mattia Antonini [2,3,†,‡] and Gianluigi Ferrari [1,*,†]

1 Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy; luca.davoli@unipr.it
2 OpenIoT Research Unit, FBK CREATE-NET, 38123 Povo, Italy; m.antonini@fbk.eu
3 Department of Information Engineering and Computer Science, University of Trento, 38123 Povo, Italy
* Correspondence: gianluigi.ferrari@unipr.it; Tel.: +39-0521-906513
† These authors contributed equally to this work.
‡ M. Antonini was with the Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Italy, at the time of writing.

Received: 26 October 2018; Accepted: 20 December 2018; Published: 22 December 2018

Featured Application: The proposed Resource Discovery (RD) and Service Discovery (SD) mechanism, denoted as DiRPL, is applicable to real deployments, where the tree-like network topology built by the IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) adaptation layer can be effective in guaranteeing reliable data collection through effective RD/SD (e.g., smart street lighting and water consumption monitoring in smart city scenarios). Smart city scenarios are typically characterized by: (i) large coverage areas; (ii) distributed sensing; and (iii) the need for human intervention in the presence of a fault. In this context, the proposed approach is very attractive as faulty devices can be automatically excluded from the network (which self-organizes) and newly installed IoT devices can be automatically self-discovered by the network itself.

Abstract: The Internet of Things (IoT) will bring together billions of devices, denoted as Smart Objects (SOs), in an Internet-like architecture. Typically, SOs are embedded devices with severe constraints in terms of processing capabilities, available memory (RAM/ROM), and energy consumption. SOs tend to be deployed in environments in which the human intervention is not suitable or needs to be minimized (e.g., smart city maintenance). They must adapt to the surrounding environment by self-configuring: to this end, several mechanisms have been proposed (e.g., UPnP, ZeroConf, etc.). In this paper, we focus on IEEE 802.15.4 networks with IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) adaptation layer, where IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is the routing protocol of choice. In this context, we propose a lightweight RPL-based mechanism to Resource Discovery (RD) and Service Discovery (SD), denoted as DiRPL. In particular, DiRPL exploits the RPL handshake to detect new nodes in the network; resources are then simply discovered with a Constrained Application Protocol (CoAP) request and can thus be published in a local resource directory. A very attractive feature of the proposed DiRPL approach is that it builds on well-defined and well-known standard protocols. The performance of the proposed system is investigated with WisMote nodes deployed inside the Cooja simulator, running the Contiki operating system. Practical application scenarios to large-scale smart city monitoring, such as smart lighting and large-scale water consumption monitoring, are investigated.

Keywords: RPL; Internet of Things; Service Discovery; IEEE 802.15.4; 6LoWPAN; Constrained Devices; Resource Discovery

1. Introduction

The Internet of Things (IoT) is envisioned to interconnect together billions of heterogeneous devices, denoted as Smart Objects (SOs), in a global Internet-like "network of networks" structure, thus allowing them to connect seamlessly to standard Internet hosts. In this way, new forms of interactions and applications among things and humans will be enabled, owing to the foreseen use of an IP-based protocol stack for IoT devices. The IoT will thus include nodes with different features, in terms of processing, communication capabilities, power supply, and energy consumption, spanning from SOs to smartphones, Internet hosts, up to the Cloud.

The majority of deployed IoT devices will likely operate in Low-power and Lossy Networks (LLNs). In this context, the IP protocol has been recognized as the true IoT enabler, as it allows full interoperability among heterogeneous objects and the canonical Internet. Some international standardization organizations, such as the Internet Engineering Task Force (IETF) and the Institute of Electrical and Electronics Engineers (IEEE), have contributed to the definition of protocols suitable to LLNs, such as the IETF ROLL Working Group [1] or the IETF 6LoWPAN Working Group [2]. In detail, the ROLL WG introduced the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [3], a lightweight routing protocol for LLNs, while the 6LoWPAN WG defined 6LoWPAN [4], a header compression mechanism that allows to send IPv6 frames over IEEE 802.15.4 networks.

A crucial aspect for constrained IoT scenarios is the nodes' capability to self-adapt and configure to the surrounding environment. This is a fundamental capability, since it avoids the need for external human intervention and configuration, especially for highly dynamic scenarios (e.g., smart city scenarios). Among different approaches to satisfy this requirement, Service Discovery (SD) and Resource Discovery (RD) play a fundamental role. They are representative of the ability of an IoT device to be aware of the presence of (i) resources/services in an IoT-oriented network and (ii) endpoints allowing to access the discovered resources/services. In particular, SD is suitable for networks in which auto-configuration capabilities are required (i.e., networks composed by a large number of nodes). In the presence of large amounts of IoT devices, RD can be seen as a relevant natural capability, as an IoT node is expected to expose several resources that can be/need to be discovered to fully interact with these nodes.

Among different application protocols that can be adopted in IoT scenarios, the most relevant one is represented by the (request/response) Constrained Application Protocol (CoAP) [5]. CoAP assures that IoT nodes (implementing a CoAP-based server) expose a particular resource, addressable at the Uniform Resource Identifier (URI) `/.well-known/core`. A unicast CoAP GET request to the URI obtains, as a response, a (formatted in CoRE Link format [6]) list containing all the CoAP resources available on the queried CoAP server. In this way, CoAP provides a simple and practical RD mechanism available for both constrained and powerful devices.

Several approaches to both RD and SD have been proposed, for both IoT-oriented scenarios and, in general, for the Internet. However, these approaches are not exempt from disadvantages, typically requiring heavy information exchange procedures, as well as sometimes being time- and energy-consuming during both RD and SD phases, thus being "lethal" for battery-powered constrained IoT devices. This underlines the requirement, for an IoT network, to rely on lightweight information exchange mechanisms, thus bringing also benefits to multi-hop communications among SOs, in terms of energy consumption and transmission delays. However, the need to adopt small footprint mechanisms is critical, due to the limited amount of memory available on SOs, to allow developers to deploy their applications without the need to sacrifice some features, abiding by practical memory constraints.

In this work, we propose a low-power and lightweight RD/SD mechanism, denoted as DiRPL, which builds upon control features already available in RPL protocol, namely Destination-Oriented Directed Acyclic Graph (DODAG) Information Object (DIO)/Destination Advertisement Object (DAO) messages exchange. Unlike other RD/SD mechanisms, DiRPL is particularly suitable for constrained and duty-cycled SOs operating in 6LoWPAN multi-hop networks and avoids the introduction of

redundant transmissions. It is targeted to Contiki OS-enabled SOs but, being based on an RPL-oriented approach, may be extended to other OSs, e.g., RIOT OS [7]. An experimental performance evaluation is carried out in the Cooja simulator, to evaluate the feasibility and efficiency, in terms of delay and energy consumption, of the proposed DiRPL RD/SD mechanism.

The rest of this paper is organized as follows. In Section 2, an overview of related works on SD and RD mechanisms for IoT systems is provided. In Section 3, a detailed description of the building blocks of the proposed architecture is given. Implementation details of the proposed architecture and experimental performance results are presented in Section 4. A brief discussion on the proposed RPL-based RD/SD mechanism, denoted as DiRPL, is done in Section 5. Finally, conclusions are drawn in Section 6.

2. Related Works

Several mechanisms for SD and RD have been proposed in the literature, from both academic and industrial worlds. The Service Location Protocol (SLP) [8,9] and the Universal Plug and Play (UPnP) [10] protocol refer to discovery and auto-configuration of existing services in Local Area Networks (LANs). Regardless of their features, neither are suitable to constrained IoT devices, due to high computation requirements, energy consumption, and reduced node lifetime.

In the area of SD and RD mechanisms for IoT systems, a few approaches have been proposed. In [11], Kovacevic et al. propose a protocol, denoted as NanoSD, suitable for mobile, highly dynamic and large-scale Wireless Sensor Networks (WSNs). It is based on multicast and broadcast messaging schemes to keep the list of resources and services available at neighboring nodes updated. In [12], an adaptive and context-aware RD/SD protocol, denoted as TRENDY and based on CoAP RESTful Web Services, is proposed. TRENDY splits a network into groups based on location tags, hierarchically organizing them, and assigning each node a specific role inside its own group. Although TRENDY leads to a scalable architecture, in which discovery is performed with a registration on a specific register entity, it suffers from a single point of failure, because of the central role of this register entity. In [13], a hybrid RD/SD mechanism for local and large-scale IoT scenarios is proposed, adopting a Peer-to-Peer (P2P)-based architecture for large-scale and geo-distributed scenarios, and the ZeroConf protocol [14] for local 6LoWPAN scenarios. The latter protocol reuses the Domain Name System (DNS) message-based semantics, denoted as DNS-SD [15–17], over IP multicast, to provide name resolution and service discovery/advertisement. In this way, while in single-hop LLNs broadcasting can be used, in multi-hop networks multicast can be implemented using the Multicast Protocol for LLNs (MPL) [18]. It is also possible to use RPL-based techniques, such as Stateless Multicast RPL Forwarding (SMRF) [19] or Lightweight Multicast Forwarding Protocol (LMFP) [20].

Owing to the characteristics of CoAP, other SD and RD mechanisms have been proposed. An example is represented by IoTivity [21], an open-source interoperability framework running over various types of radio connectivity (e.g., IEEE 802.3, Bluetooth, BLE, ZigBee, IEEE 802.15.4, Z-Wave, NFC, etc.) and using the IP/UDP/CoAP stack. Issuing a multicast CoAP request for a specific resource type, a CoAP client will receive a response only from the nodes that host these resources.

3. DiRPL: RPL-Based Resource and Service Discovery for Constrained IoT Devices

The RD/SD mechanism proposed in this work, denoted as DiRPL, relies on RPL, as highlighted in Section 1, in turn exploiting the handshake among a constrained IoT node newly joining a RPL-based network, and a sink node, referred to as Border Router (BR), handling the coordination tasks of the RPL-based network. We will refer to the joining IoT node as Last Entered Node (LEN). In the following, a description of the technique through which a new node discovers and joins an existing RPL-based DODAG is provided in Section 3.1, while Section 3.2 introduces our novel RD/SD mechanism, detailing how it exploits the RPL handshake phase in order to discover nodes, resources, and services. Hence, it is possible to separate the proposed DiRPL approach in two phases, both relying on a key behavior of CoAP, namely, the possibility to *observe* a CoAP resource (here defined as the "special" CoAP

resource /network/last_entered_node through the OBS option), thus avoiding continuous polling of appearing constrained nodes and to their resources and services. As will be better detailed in the following, this behavior helps in reducing the traffic among the IoT constrained network handled by the BR, together with a simplification of the tasks to be carried out by each LEN, which reduce to the need for being discovered by the BR itself and to reply with a list of the maintained CoAP resources once the joining phase has completed. It follows that this approach scales well and can be applied to several scenarios in which a direct human intervention could be prevented, e.g., in smart city scenarios [22] where technicians have to install/substitute SOs quickly without having to care about the interactions among themselves.

3.1. Node Joining Technique in an RPL DODAG

When a constrained node appears within the coverage area of a 6LoWPAN-based network, it tries to discover its neighbors carrying out a Neighbor Discovery (ND) task. First, the LEN sends, as a multicast request, a Router Solicitation (RS) packet to all neighboring nodes. Upon receiving the RS packet, the nodes reply to the originator node with a unicast Router Advertisement (RA) packet containing: (i) the prefix of the IPv6 address (inside a Prefix Information Object, PIO); (ii) the adopted compression technique (Context Option, CO); and (iii) the Border Router Option (Authoritative Border Router Option, ABRO). Upon receiving the RA packet, the LEN elects one router as its default router (also denoted as "parent"), typically choosing, among all its possible neighbors, the first node which replies to its solicitations. Hence, the LEN computes the global IPv6 address, based on the PIO address, and sends a unicast Neighbor Solicitation (NS) packet, containing the Address Registration Option (ARO), to its parent node, asking for the Link Layer address of the parent node and its reachability. Once the NS packet is received, the parent node adds an entry into its Neighbor Cache and replies with a Neighbor Advertising (NA) packet, containing the status of the address registration procedure: 0—Success, 1—Duplicate Address, or 2—Neighbor Cache Full. The reception of the NA packet completes the ND phase and denotes the beginning of the RPL registration task.

The LEN sends a DODAG Information Solicitation (DIS) message to its parent, which replies with a DIO message containing: (i) the rank of the LEN, (ii) the routing metrics, and (iii) the PIO object. Once the DIO message is received, the upward path to the BR is established. Then, the LEN sends a DAO message to its parent node, in turn forwarded upward by the parent itself, until the DAO reaches the BR. Since RPL uses DAO messages to create a downward path to the LEN, the BR replies, upon receiving the DAO message, with a DAO-ACK message to the LEN. When the DAO-ACK message is received by the LEN, the network registration task completes, with the LEN correctly discovered and listed in the routing table of the BR.

In this work, the proposed experimental scenario is composed by a multi-hop 6LoWPAN network managed by a BR running on a TelosB mote [23]. Among its active processes, the BR executes a CoAP server maintaining an observable resource (through the CoAP-provided OBS option), denoted as /network/last_entered_node, which: (i) is internally updated when a new joining node enters in the RPL DODAG and (ii) returns the IP address of the LEN itself. The TelosB mote is, in turn, plugged (through the USB port) into a Raspberry Pi (RPi) device, which runs two processes: (i) a CoAP client observing the CoAP resource /network/last_entered_node (through the CoAP OBS option); and (ii) a CoAP server, which maintains a list of discovered IoT end-nodes and CoAP resources.

Since the proposed scenario is CoAP-based, it can easily be integrated in IoT networks adopting this application protocol, such as the IoT-oriented testbed hosted at the IoT Lab of the Department of Engineering and Architecture of the University of Parma [24] and denoted as "Web of Things Testbed (WoTT)" [25], as well to an extended testbed such as the one characterizing a smart city, where several WoTTs may be integrated in order to monitor the urban environment and provide a feedback on the life style of the involved citizens. The integration procedure works as follows. When the RPi hosting the BR boots up the CoAP server, it also starts an advertisement process that publishes the RPi inside the IoT network, in detail using the DNS-SD protocol and advertising the rpibr._coap._udp.local. SRV

record [26]. Through this advertisement mechanism, the module of the WoTT that is in charge of the discovery of newly connected IoT nodes, denoted as IoTHub, can detect the CoAP server maintained by the RPi. Hence, after a CoAP GET request on its CoAP resource `/.well-known/core`, the IoTHub becomes aware of the resources known by the BR and adds them to the resource directory of the WoTT itself.

3.2. RPL-Based Resource Discovery

As previously anticipated, in this work a novel approach to RD and SD, denoted as DiRPL and based on the RPL handshake and composed of three main phases (initialization, node discovery, and resource identification), is proposed. While the RPL-based handshake helps in organizing a constrained network in a DODAG, facilitating the management and the data collection of the network itself, it does not provide an RD mechanism that matches well with most of the adopted IoT-oriented protocols, such as CoAP, thus forcing the LEN to self-advertise. This task, however, is carried out by a powerful device—namely the BR—and this is one of the reasons that motivates the approach proposed in this work.

The initialization phase acts as follows: after the BR module on the TelosB mote boots up, the RPi which the TelosB mote is attached to: (i) loads the CoAP server (as highlighted in Section 3.1) hosting the local resource directory; (ii) loads a CoAP client in charge of observing the CoAP resource `/network/last_entered_node`; and (iii) powers on the local caching processes. Finally, the CoAP client on the RPi starts observing the CoAP resource itself exposed by the TelosB.

Once the initialization phase has completed, the 6LoWPAN-oriented network is operational and can discover every new appearing IoT node (i.e., a LEN). As previously mentioned, the node discovery procedure exploits the RPL handshake for network registration. More precisely, the message exchange, shown in Figure 1, can be detailed as follows (numbers in Figure 1 refer to the step numbers below).

(1) When a joining node (LEN) appears in the coverage area of the 6LoWPAN network, it starts the ND procedure and sends a DIS message to the node that has been elected as its default parent.
(2) The parent node receives the DIS packet and replies with a DIO message. In this way, the LEN gathers knowledge about the existence of the BR in the 6LoWPAN network, and the DOGAG governing the network itself.
(3) The LEN generates a DAO packet that will be forwarded up to the BR through the DODAG.
(4) The reception of the DAO packet by the BR invokes the creation of a corresponding routing entry in the routing table of the BR itself and, since the BR is hosted on the TelosB mote, this routing insertion triggers an update of the CoAP observable resource `/network/last_entered_node`.
(5) Recalling the fact that, during the network boot up phase, the CoAP client (internal to the RPi) has started an observing relation on the CoAP resource `/network/last_entered_node` exposed by the TelosB mote, the updates of this CoAP resource are received and propagated by the BR to its upper layers, for further processing tasks.

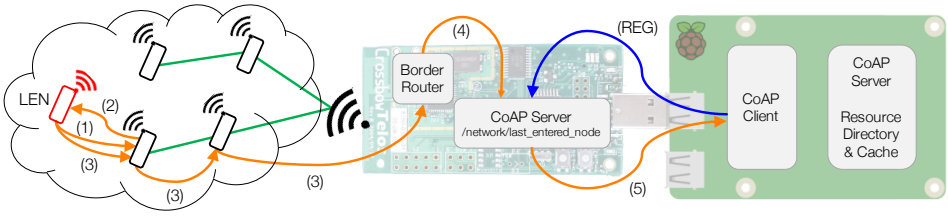

Figure 1. Node discovery procedure embedded in the RPL discovery handshake. The numbers in the figure refer to the node discovery phases.

Once the previously detailed steps have been taken, thus completing the node discovery phase, the following RD/SD procedure, shown in Figure 2, is ready to be executed (with numbers in Figure 2 referring to the step numbers below).

(6) The CoAP client running on the RPi, triggered by the CoAP Observe notification, sends a CoAP GET request to the CoAP resource `/.well-known/core` of the LEN.

(7) Since the LEN runs an internal CoAP server, it thus can handle the incoming CoAP GET request on its `/.well-known/core` resource. It replies (in CoRE Link format) to the CoAP client on-board of the RPi with the list of its maintained CoAP resources and services.

(8) Upon reception of the list of CoAP resources hosted by the LEN, the CoAP client running on the RPi requests the CoAP server, hosted on the same platform, to create several CoAP resources equal to received resource list's size.

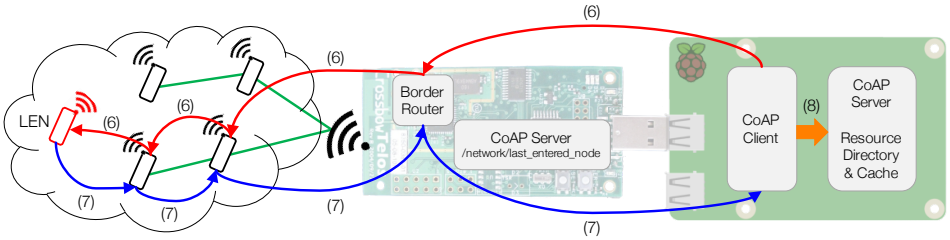

Figure 2. Resource identification procedure embedded in the RPL discovery handshake. The numbers in the figure refer to the resource identification phases.

According to the original properties of each CoAP resources on each LEN, the values of the local CoAP resources copy, maintained in the resource directory of the RPi, will be updated with the following strategies: (i) observing update, pursuing the `OBS` option of the CoAP request; or (ii) polling update, periodically issuing a CoAP request to the specific CoAP resource on the specific LEN.

4. Experimental Implementation and Performance Evaluation

4.1. Implementation Details

To demonstrate the applicability of the approach presented in Section 3, real application scenarios of interest refer to use cases in which the overall resulting 6LoWPAN needs to be energy efficient and/or self-adapting. In particular, a first use case is a smart home scenario with battery-powered sensing nodes. This scenario may be characterized by different devices (e.g., fans, light bulbs, etc.) and resources (e.g., remote control, appliance status, etc.) that should be discovered without any human intervention, with, in principle, various possible topologies. Another interesting use case is a smart city lighting scenario, which can be inherently modeled as a tree, thus being characterized by large coverage areas (compared to the smart home scenario), device homogeneity (street lamps), and with the need for a human intervention in the presence of a faulty lamp. In particular, the possibility of avoiding a reconfiguration by the technician on each repaired lamp is an attractive feature of DiRPL. In fact, each new lamp is self-discovered by the BR managing the street lighting architecture, while faulty lamps can be removed from the resource directory after a fixed inactivity period. A third interesting use case is given by a smart city water consumption monitoring system, which can be composed by both (i) networks federating domestic users and (ii) urban networks. In the case of domestic users, due to the possibly large distance among houses, the water meter can be battery-powered (this being a constraint) or it may directly use water flow as power source (exploiting small actuators converting the flow into energy). In both these cases, the possibility to limit the energy consumption and to exploit self-discovery mechanisms in the presence of a water meter fault is a strict requirement. In the case of an urban monitoring network, the density of water meters could be higher than in the domestic use case, and the need to minimize technician interventions supports the adoption of a DODAG-oriented approach which exploits resource and service's self-discovery. Hence, the proposed mechanism is preferable with respect to systems in which a centralized admission control (for a particular area) is required. In fact, the latter requires a manual intervention in updating the system status in the presence

of system changes, whereas DiRPL allows to automatically maintain the overall infrastructure status updated. Hence, these considerations motivate the applicability of the proposed mechanism to various practical scenarios.

On the basis of these use cases, in order to test the behavior of the proposed discovery technique, being able to collect performance indicators prior to really deploy such scenarios in real world, a Contiki-based implementation has been developed, with performance evaluation carried out by simulating WisMote nodes [27] in the Cooja simulator [28]. In detail, WisMote devices are equipped with 16 KB RAM and, depending on the specific type of platform, with 128 KB, 192 KB or 256 KB ROM. All the experiments have been performed on an Ubuntu 12.04-based virtual machine, equipped with a 2.2 GHz Intel i7-3632QM processor and with 4 GB RAM. Finally, the Contiki OS software stack (Contiki OS, uIPv6, RPL, and ContikiMAC) running on each node has been configured to fit memory constraints required by the adopted nodes.

DiRPL has been tested on 6LoWPANs composed by Contiki-enabled nodes and arranged in linear topologies, as shown in Figure 3 in an illustrative case with $N = 8$ nodes. Besides being representative of relevant use cases (e.g., smart lighting and water consumption monitoring), the choice of evaluating the performance of the proposed RPL-based RD/SD algorithm on linear topologies is due to the simulation implementation efficiency of this type of topologies. Furthermore, linear topologies allow to reduce the simulation time of DiRPL without limiting the applicability to other topologies. We remark that the simulation time required to generate the results presented in this paper is approximately 240 hours. Moreover, regardless of the complete network topology, each multi-hop path (e.g., toward the BR) can be modeled as a linear "subnetwork," making the obtained results applicable (as a theoretical performance benchmark) to a general case.

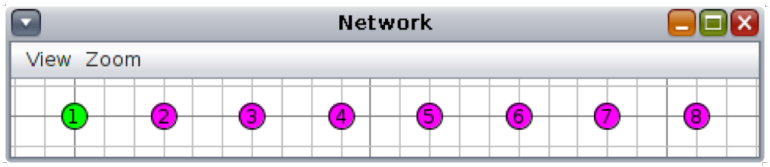

Figure 3. Network with linear topology ($N = 8$ nodes) modeled in the Cooja simulator.

In the following, topologies with various values of the number of nodes N, from 1 to 12, have been considered. In particular, node n_1 corresponds to the 6LoWPAN Border Router (6LBR), acting as root of the RPL DODAG and notifying the admission of new LENs in the RPL-based network. The other nodes represent resource owners exposed to external CoAP requests through CoAP servers built with the Erbium library [29]. For each performance evaluation campaign, the distance between consecutive nodes is selected so that packets are forced to follow a multi-hop path, in which the number of hops is equal to the number of LENs. This forces each constrained node to communicate only with its direct neighbors.

The simulation-based investigation aims at evaluating the time and overall network energy consumption needed to perform an RD/SD campaign using the proposed RPL-oriented approach. All the results shown in the following have been obtained by performing 100 discovery runs for each configuration. The performance metrics are the time required and the energy consumed for: (i) joining the RPL DODAG; (ii) propagating a DAO message from the LEN up to the BR; and (iii) transmitting a CoAP request for RD/SD from the BR to the LEN. A brief description of the employed performance metrics is provided in Table 1.

Table 1. Performance metrics.

Metric	Description	Unit
T-Join	Joining Time: time needed by the LEN to receive information about the DODAG from its parent	(ms)
EN-Join	Joining Energy: energy spent by the LEN and its parent during the joining phase	(mJ)
T-DAO-EBR	Time DAO-EBR: time interval between the generation instant of the DAO packet by the LEN and the creation instant of the route in the routing table of the BR	(ms)
EN-DAO-EBR	Energy DAO-EBR: overall energy spent by the network during the propagation of the DAO packet from the LEN to the BR	(mJ)
T-SCR-RES	Time SCR-RES: time needed for a RD/SD on the LEN originated by the BR	(s)
EN-SCR-RES	Energy SCR-RES: overall energy spent by the network during a RD/SD from the LEN to the BR	(mJ)

Being the proposed experimental analysis based on several simulation rounds, the energy consumption has been calculated as follows. The output logging messages provided by Cooja during simulation execution contain several informations. In particular, among the available data, there is the energy harvested in four different phases of the constrained device's life cycles: transmission phase—`TX`; reception phase—`RX`; processing phase—texttttCPU; waiting phase—`IDLE`. These data are recorded every 125 ms and used to estimate the values of the performance metrics, as explained in the following. Our results could be compared with those obtained with a RD-oriented architecture similar to the one proposed in this work, as the one proposed in [12]. However, this is out of the scope of the proposed work and we leave it to future investigations.

The performance results depend on the Channel Check Ratio (CCR), which corresponds to the rate (dimension: [Hz]) at which a node switches on the radio transceiver to check the activity over the channel. The higher the CCR, the finer the channel state monitoring, but the higher the energy wasted in useless channel checks. The value of the CCR should be properly optimized, taking into account the inherent trade-off between energy consumption and desired performance. Usually, small values of CCR are adopted in LLNs, with the default value set to 8 Hz. In our analysis, two different values are considered: 8 Hz and 16 Hz: the first value has been chosen because it is the default and mostly used one in Contiki-based networks; the second value has been selected to further validate the proposed DiRPL approach.

4.2. Performance Analysis with CCR set at 8 Hz

The overall network energy consumption (in logarithmic scale for the sake of visualization clarity) during the three different phases of the proposed discovery mechanism is shown in Figure 4. As expected, the highest energy consumption is due to the RD/SD activity (phase SCR-RES): this corresponds, as previously introduced, to a confirmable CoAP request to the `/.well-known/core` resource of the LEN. As shown in Figure 5, EN-SCR-RES is a quadratically increasing function of the number of nodes in the network (i.e., the network depth with a linear topology). This is because the number of packets/fragments is a quadratically increasing function of the number of traversed nodes.

In Figure 5, it is also shown how the energy consumption distributes at different nodes: blue bars are associated with the BR's energy consumption, whereas yellow ones are associated with the overall energy consumption of CoAP nodes. From the results in Figure 4, it can be concluded that the Join phase is, from an energetic point of view, the least demanding task. In fact, this energy consumption turns out to be constant, i.e., it does not depend on the number of nodes N. This is expected, as the consumed energy depends only on a LEN and its parent.

In Figure 6, the energy consumed during the DAO-EBR phase (i.e., the metric EN-DAO-EBR) is shown as a function of the number of nodes N, highlighting the energy partitioning among BR and CoAP nodes. It can be observed that EN-DAO-EBR is a linearly increasing function of N. By comparing the numerical values of the energies in Figure 6 with those in Figure 5, it can be concluded that the energy consumption in the second phase is much lower than in the first one.

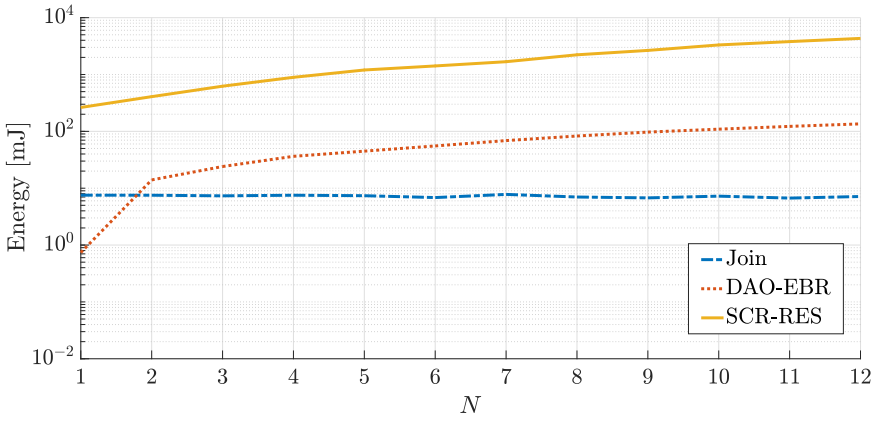

Figure 4. Comparison among energies (log scale) spent by the overall network during different phases of the proposed discovery approach, with CCR set to 8 Hz.

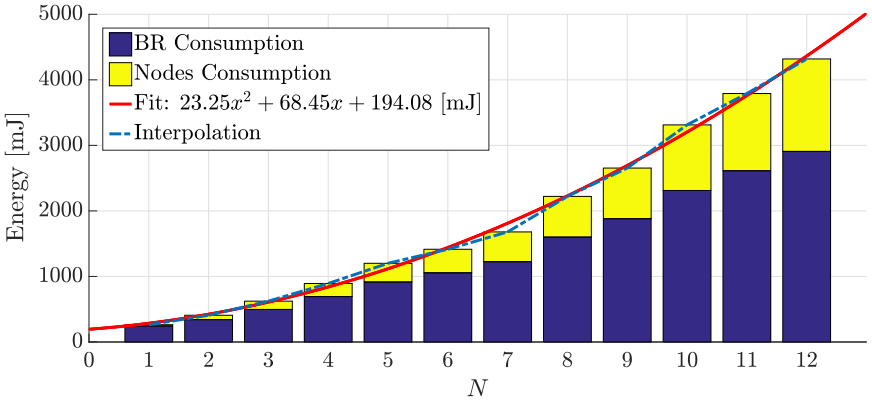

Figure 5. Energy spent by the overall network during the SCR-RES phase with CCR set to 8 Hz (metric EN-SCR-RES). Bars show the amount of energy spent by BR and nodes.

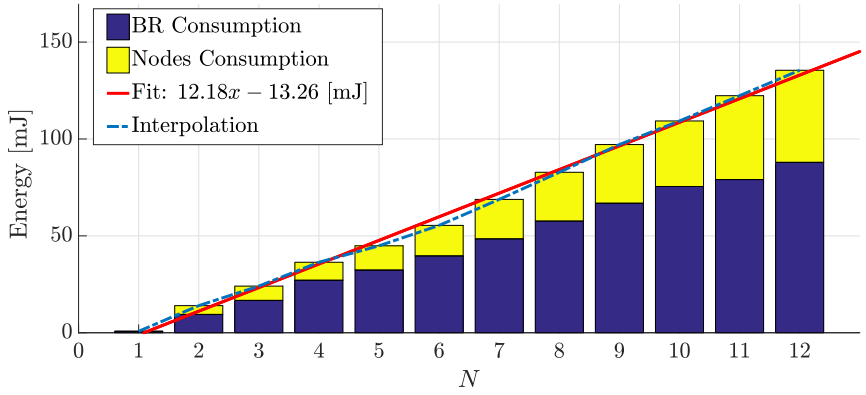

Figure 6. Energy spent by the overall network during the DAO-EBR phase with CCR set to 8 Hz (metric EN-DAO-EBR). Bars show the amount of energy spent by BR and Nodes.

Another important performance metric is the time required by each phase of the discovery mechanism. In Figure 7, a comparison among these times during different phases is carried out. In particular, the times replicate the same behaviors of energies: the time required by the Join phase (metric T-Join) is constant with respect to the number of nodes; it grows linearly (metric T-DAO-EBR) and quadratically (metric T-SCR-RES) during the DAO-EBR (Figure 8) and SCR-RES (Figure 9) phases, respectively.

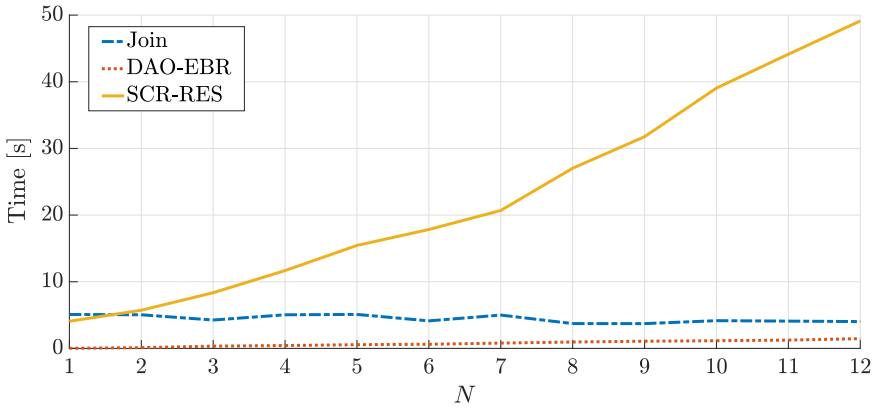

Figure 7. Comparison among times required by the different phases of the discovery procedure, with CCR set to 8 Hz.

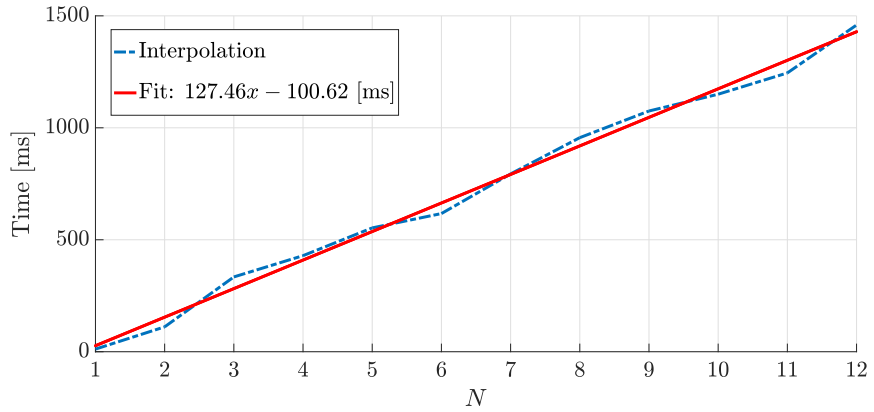

Figure 8. Time required to complete the DAO-EBR phase with CCR set to 8 Hz (metric T-DAO-EBR).

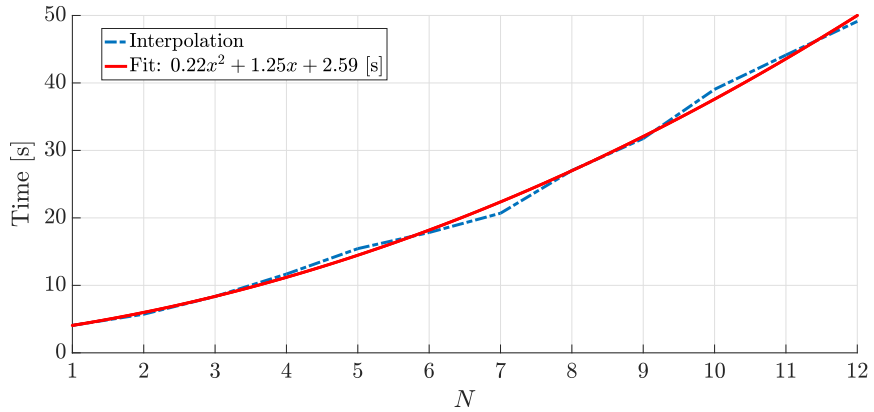

Figure 9. Time required to complete the SCR-RES phase with CCR set to 8 Hz (metric T-SCR-RES).

As underlined above, the results shown in Figures 7–9 have been obtained deploying various numbers of nodes in networks with linear topologies. This kind of topologies allows to extract the distance (in terms of number of hops or node depth) between the BR and the LEN in a very simple way. In a network with a generic topology, the distance between BR and LEN may not be constant (as it depends on the position where the LEN attaches) and may change over time. Without loss of generality, our approach can be extended to more general topologies. As anticipated above, considering a network

with a general topology, every path between the BR and the LEN can be interpreted as a subnetwork with linear topology with additional neighboring interfering devices (especially because of multiple access interference) that do not belong to this path. As intuitively expected, these devices will likely cause collisions, thus introducing additional delays and increasing energy consumption in the network. Therefore, our experimental results (in terms of energy consumption and delay) can be interpreted as lower bounds for the performance of networks with general topologies, provided that N is interpreted as the number of hops of the path between a node of interest (for instance, the LEN) and the BR, i.e., as the node's depth.

4.3. Performance Analysis with CCR set to 16 Hz

As mentioned before, we have conducted our analysis also with CCR set to 16 Hz. The obtained results are trend-wise similar to those with CCR set to 8 Hz. In particular, quadratically increasing metrics, such as EN-SCR-RES (Figure 10) and T-SCR-RES (Figure 11), tend to linearize for increasing values of N.

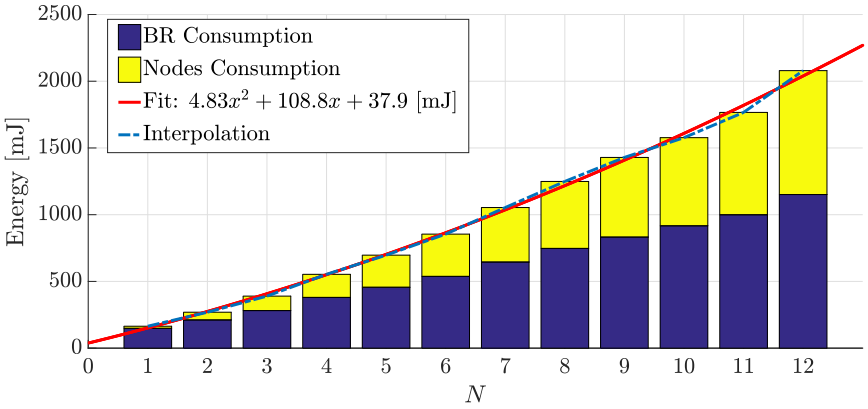

Figure 10. Energy spent by the overall network during the SCR-RES phase with CCR set to 16 Hz (metric EN-SCR-RES). Bars show the amount of energy spent by BR and Nodes.

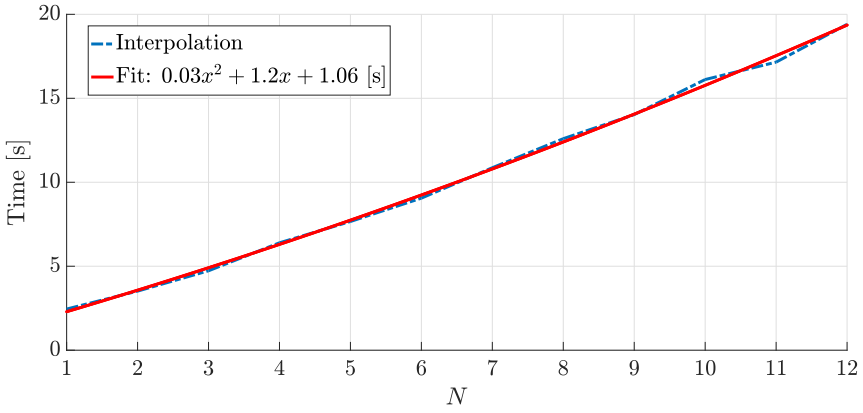

Figure 11. Time required to complete the SCR-RES phase with CCR set to 16 Hz (metric T-SCR-RES).

As discussed at the end of Section 4.2, the experimental results with the CCR set to 16 Hz in the presence of a linear topology can be interpreted as lower bounds for the system performance of networks with more general topologies.

5. Discussion

In this section, we discuss on the benefits brought by the adoption of a RD/SD mechanism exploiting the normal behavior of RPL, such as DiRPL, rather than using a RD/SD protocol which is operating at upper ISO/OSI layers.

With reference to the architecture proposed in Section 3, it is clear that, by exploiting the basic operations of the RPL protocol (namely, transmission of control messages for DODAG maintenance), the RD/SD approach embedded in DiRPL does not insert, by default, any additional operation and, thus, any additional delay. More precisely, DiRPL does not introduce any additional overhead in the DIO/DAO messages exchange among entering nodes and the RPL Border Router during the DODAG topology construction. This can be inherently considered as an energy efficient approach, since the energy consumption for the constrained nodes joining the DODAG is unavoidable, while the energy consumption of the Border Router is likely not a problem, since the Border Router is expected to be plugged in into an electrical network. Hence, the constrained nodes do not incur any supplementary energy and time consumption beyond those required by the RPL protocol during its normal operations.

Moreover, if compared with other RD/SD mechanisms (e.g., DNS-SD, mDNS [30–32], Bonjour [33], HyperCat [34], AllJoyn [35], DLNA [36], SSDP [37], Physical Web [38], NetBIOS [39], WS-Discovery [40]), the RD/SD approach proposed in DiRPL is less time-consuming, due to the fact that DiRPL works at the ISO/OSI L3 layer, while other approaches operate at higher layers (e.g., L4, L5, and L7). This can be denoted in Figure 12, in which some RD/SD mechanisms are highlighted with respect to their proper ISO/OSI layer, and in which is possible to note that, in a RPL-based network, all these protocols should work on top of RPL itself, it this way increasing the time required to complete the RD/SD procedure.

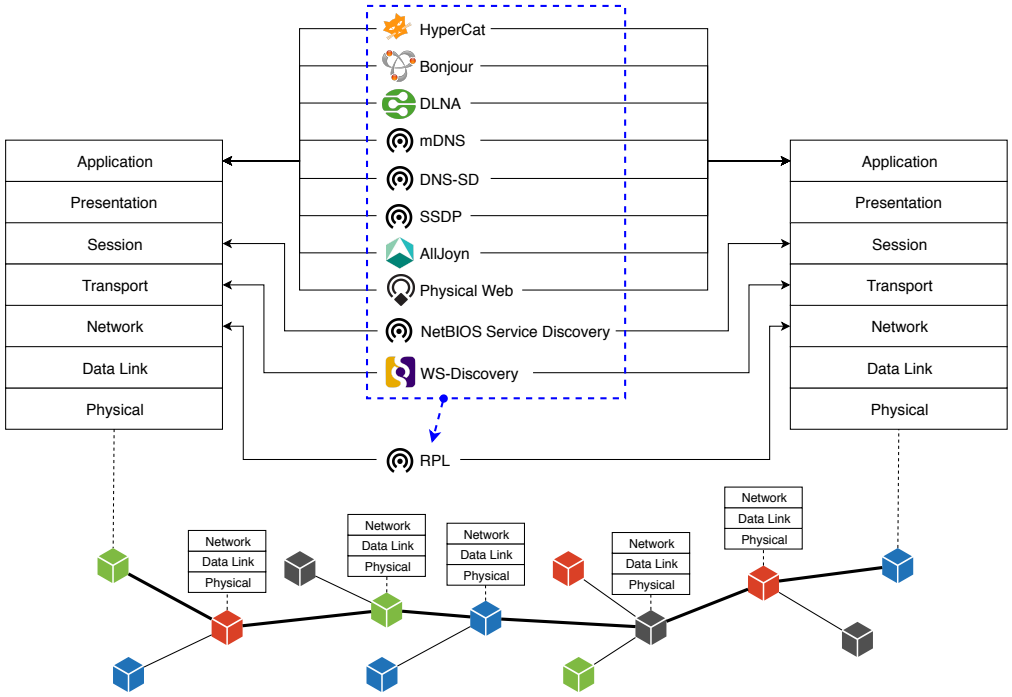

Figure 12. Comparison among different RD/SD protocols operating at various ISO/OSI layers. An illustrative multi-hop path is shown: intermediate nodes use their protocol stacks up to L3.

On the contrary, adhering to a RD/SD paradigm such as the one in DiRPL, it is possible to restrain the time required to complete these operations, as well as to avoid both time and energy consumption caused by packets having to go up to upper ISO/OSI layers, as shown in Figure 12. Hence, as depicted in Figure 13, the DiRPL approach allows to perform RD/SD only exploiting the normal operation of RPL, with packets flowing, into each DODAG-handled node, till L3 of the ISO/OSI protocol stack.

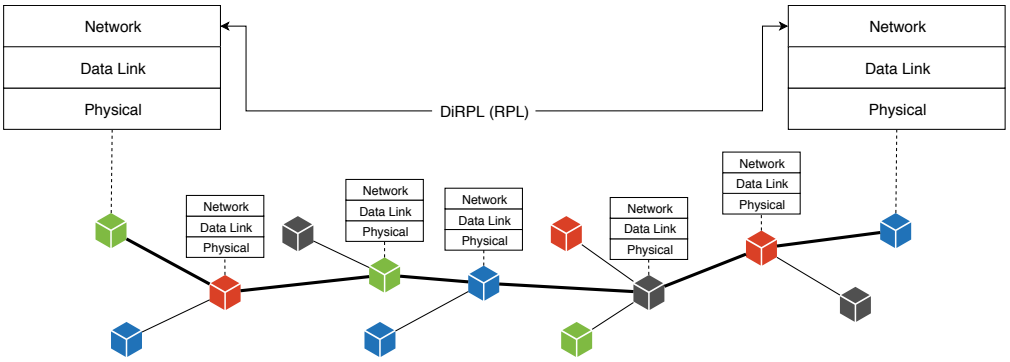

Figure 13. Detail on the ISO layers to be traversed by traffic flows with the proposed DiRPL mechanism.

For what concerns the performance evaluation, the choice of evaluating the performance of DiRPL on a linear network topology is due to the simulation implementation efficiency of this type of topology: the simulation time required to generate the proposed results is approximately 240 h on a commercial PC (as shown in Section 4, equipped with Ubuntu 12.04, equipped with a 2.2 GHz Intel i7-3632QM processor and with 4 GB RAM). Moreover, the performance of a linear network can be considered as a performance benchmark for a multi-hop path in a more general (e.g., tree-like) network topology. The application of DiRPL to other scenarios is an interesting research direction, possibly comparing the obtained performance with various simulated constrained devices (e.g., TelosB, Tmote Sky, Z1, all characterized by different RAM/ROM availability). We also remark that, regardless of the complete network topology, each multi-hop path (e.g., toward the BR) of the proposed IoT scenarios can be interpreted as a linear "subnetwork," making the obtained results applicable (as a theoretical performance benchmark) to a general case. Nevertheless, we plan to move to a more efficient simulation platform, such as a cluster farm composed by more powerful servers, specifically dedicated to simulation purposes. This will allow us to simulate DODAGs with a larger number of constrained nodes in an acceptable time. Moreover, this should allow also to evaluate heterogeneous RPL-based networks, composed not only by a unique type of constrained devices, but by heterogeneous devices with different features, such as a smart grid scenario similar to the one proposed in [41], which can be considered as a challenging scenario, due to the large number of PLC-based nodes that can be simulated and to their heterogeneity.

6. Conclusions

In this paper, we have introduced a low-power and lightweight RD/SD mechanism, denoted as DiRPL, particularly suited for constrained and duty-cycled SOs forming IEEE 802.15.4-based multi-hop IoT networks. First, we have described the main components of DiRPL and the functionalities that these elements must implement to perform SD and RD. Then, we have proposed a solution for automatic local RD/SD that allows (i) to discover resources advertised and available in constrained SOs and (ii) to publish them into the local resource directory (e.g., at the BR) with no need for any prior configuration. An extensive performance evaluation of the proposed RD/SD mechanism characterizing DiRPL has been performed, using Contiki-based nodes organizing in 6LoWPANs with RPL in the Cooja simulator. The obtained results show that the time required for RD/SD is a quadratically increasing function of the number of hops in the path between a joining SO (i.e., a LEN) and the BR. Moreover, for increasing values of the CCR, delay and energy tend to become linear functions of the number of hops between the LEN and the BR.

Author Contributions: Conceptualization, L.D., M.A. and G.F.; Data curation, L.D., M.A. and G.F.; Formal analysis, L.D., M.A. and G.F.; Investigation, L.D., M.A. and G.F.; Methodology, L.D., M.A. and G.F.; Supervision, G.F.; Validation, L.D., M.A. and G.F.; Writing—original draft, L.D., M.A. and G.F.; Writing—review & editing, L.D., M.A. and G.F.

Funding: The work of L.D. and G.F. is partially funded by the European Commission H2020 Framework Program, under Grant No. 783221, AFarCloud project - "Aggregate Farming in the Cloud." The work of L.D. is also funded by the University of Parma, under "Iniziative di Sostegno alla Ricerca di Ateneo" program, "Multi-interface IoT sYstems for Multi-layer Information Processing (MIoTYMIP)" project. The work reflects only the authors' views; the European Commission is not liable for any use that may be made of the information contained herein.

Conflicts of Interest: The authors declare that there is no conflict of interest regarding the publication of this paper.

References

1. IETF Routing over Low-Power and Lossy Networks (ROLL) Working Group. Available online: https://datatracker.ietf.org/wg/roll/ (accessed on 23 October 2018).
2. IETF IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Working Group. Available online: https://datatracker.ietf.org/wg/6lowpan/ (accessed on 23 October 2018).
3. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.; Alexander, R. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*; RFC 6550; Internet Engineering Task Force: Fremont, CA, USA, 2012.
4. Montenegro, G.; Kushalnagar, N.; Hui, J.; Culler, D. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*; RFC 4944; Internet Engineering Task Force: Fremont, CA, USA, 2007.
5. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; RFC 7252; Internet Engineering Task Force: Fremont, CA, USA, 2014.
6. Shelby, Z. *Constrained RESTful Environments (CoRE) Link Format*; RFC 6690; Internet Engineering Task Force: Fremont, CA, USA, 2012.
7. Baccelli, E.; Hahm, O.; Wählisch, M.; Günes, M.; Schmidt, T. *RIOT: One OS to Rule Them All in the IoT*; Research Report RR-8176; INRIA: Le Chesnay, France, 2012.
8. Guttman, E.; Perkins, C.; Veizades, J. *Service Location Protocol, Version 2*; RFC 2608; Internet Engineering Task Force: Fremont, CA, USA, 1999.
9. Guttman, E. *Vendor Extensions for Service Location Protocol, Version 2*; RFC 3224; Internet Engineering Task Force: Fremont, CA, USA, 2002.
10. UPnP Standards & Architecture. 1999. Available online: https://openconnectivity.org/developer/specifications/upnp-resources/upnp (accessed on 23 October 2018).
11. Kovacevic, A.; Ansari, J.; Mahonen, P. NanoSD: A Flexible Service Discovery Protocol for Dynamic and Heterogeneous Wireless Sensor Networks. In Proceedings of the IEEE Sixth International Conference on Mobile Ad-hoc and Sensor Networks, Hangzhou, China, 20–22 December 2010; pp. 14–19. [CrossRef]
12. Butt, T.A.; Phillips, I.; Guan, L.; Oikonomou, G. TRENDY: An Adaptive and Context-aware Service Discovery Protocol for 6LoWPANs. In Proceedings of the Third International Workshop on the Web of Things (WoT'12), Newcastle, UK, 19 June 2012; pp. 2:1–2:6. [CrossRef]
13. Cirani, S.; Davoli, L.; Ferrari, G.; Leone, R.; Medagliani, P.; Picone, M.; Veltri, L. A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things. *IEEE Internet Things J.* **2014**, *1*, 508–521. doi:10.1109/JIOT.2014.2358296. [CrossRef]
14. ZeroConf Suite. Available online: http://zeroconf.org/ (accessed on 23 October 2018).
15. Cheshire, S.; Krochmal, M. *DNS-Based Service Discovery*; RFC 6763; Internet Engineering Task Force: Fremont, CA, USA, 2013.
16. Stolikj, M.; Verhoeven, R.; Cuijpers, P.J.L.; Lukkien, J.J. Proxy support for service discovery using mDNS/DNS-SD in low power networks. In Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, Australia, 19 June 2014; pp. 1–6. [CrossRef]
17. Klauck, R. Seamless Integration of Smart Objects into the Internet Using XMPP and mDNS/DNS-SD. Ph.D. Thesis, Brandenburg University of Technology Cottbus-Senftenberg, Senftenberg, Germany, 2016.
18. Hui, J.; Kelsey, R. *Multicast Protocol for Low-Power and Lossy Networks (MPL)*; RFC 7731; Internet Engineering Task Force: Fremont, CA, USA, 2016.

19. Oikonomou, G.; Phillips, I. Stateless multicast forwarding with RPL in 6LowPAN sensor networks. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Lugano, Switzerlan, 19–23 March 2012; pp. 272–277. [CrossRef]
20. Antonini, M.; Cirani, S.; Ferrari, G.; Medagliani, P.; Picone, M.; Veltri, L. Lightweight multicast forwarding for service discovery in low-power IoT networks. In Proceedings of the 2014 IEEE 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 17–19 September 2014; pp. 133–138. [CrossRef]
21. IoTivity. Available online: https://www.iotivity.org/ (accessed on 23 October 2018).
22. Davoli, L.; Belli, L.; Cilfone, A.; Ferrari, G. Integration of Wi-Fi mobile nodes in a Web of Things Testbed. *ICT Express* **2016**, *2*, 96–99. doi:10.1016/j.icte.2016.07.001. [CrossRef]
23. TelosB Mote Platform. Available online: https://goo.gl/ENwrcX (accessed on 23 October 2018).
24. Web of Things Testbed (WoTT). Available online: http://iotlab.unipr.it/wott (accessed on 23 October 2018).
25. Belli, L.; Cirani, S.; Davoli, L.; Gorrieri, A.; Mancin, M.; Picone, M.; Ferrari, G. Design and Deployment of an IoT Application-Oriented Testbed. *Computer* **2015**, *48*, 32–40. doi:10.1109/MC.2015.253. [CrossRef]
26. Gulbrandsen, A.; Vixie, P.; Esibov, L. *A DNS RR for Specifying the Location of Services (DNS SRV)*; RFC 2782; Internet Engineering Task Force: Fremont, CA, USA, 2000.
27. WisMote. Available online: https://goo.gl/PceaQ3 (accessed on 23 October 2018).
28. Roussel, K.; Song, Y.Q.; Zendra, O. Using Cooja for WSN Simulations: Some New Uses and Limits. In Proceedings of the EWSN 2016—NextMote Workshop, Graz, Austria, 15–17 February 2016; Roemer, K., Ed.; Junction Publishing: Graz, Austria, 2016; pp. 319–324.
29. Kovatsch, M.; Duquennoy, S.; Dunkels, A. A Low-Power CoAP for Contiki. In Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011), Valencia, Spain, 17–22 October 2011.
30. Cheshire, S.; Krochmal, M. *Multicast DNS*; RFC 6762; Internet Engineering Task Force: Fremont, CA, USA, 2013.
31. Klauck, R.; Kirsche, M. Enhanced DNS message compression—Optimizing mDNS/DNS-SD for the use in 6LoWPANs. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), San Diego, CA, USA, 18–22 March 2013, pp. 596–601. [CrossRef]
32. Siljanovski, A.; Sehgal, A.; Schönwälder, J. Service discovery in resource constrained networks using multicast DNS. In Proceedings of the European Conference on Networks and Communications (EuCNC), Bologna, Italy, 23–26 June 2014; pp. 1–5. [CrossRef]
33. Bonjour. Available online: https://support.apple.com/bonjour (accessed on 8 December 2018).
34. Lea, R. HyperCat: An IoT Interoperability Specification. 2013. Available online: http://eprints.lancs.ac.uk/id/eprint/69124 (accessed on 20 December 2018).
35. AllJoyn Open Source Project. Available online: https://goo.gl/CiCM9U (accessed on 8 December 2018).
36. Heredia, E.A. *An Introduction to the DLNA Architecture: Network Technologies for Media Devices*, 1st ed.; Wiley Publishing: Hoboken, NJ, USA, 2011.
37. Goland, Y.Y.; Cai, T.; Leach, P.; Gu, Y.; Albright, S. *Simple Service Discovery Protocol/1.0 Operating without an Arbiter*; Internet-Draft draft-cai-ssdp-v1; Internet Engineering Task Force: Fremont, CA, USA, 1999.
38. Physical Web. Available online: https://google.github.io/physical-web/ (accessed on 8 December 2018).
39. Haugdahl, J.S. *Inside NetBIOS*; Architecture Technology Corp.: Minneapolis, MN, USA, 1990.
40. Web Services Dynamic Discovery (WS-Discovery). Available online: https://goo.gl/bGJMmH (accessed on 8 December 2018).
41. Davoli, L.; Veltri, L.; Ferrari, G.; Amadei, U., Internet of Things on Power Line Communications: An Experimental Performance Analysis. In *Smart Grids and Their Communication Systems*; Kabalci, E., Kabalci, Y., Eds.; Springer: Singapore, 2019; pp. 465–498._13. [CrossRef]