Wireless Netw (2014) 20:2437–2453
DOI 10.1007/s11276-014-0749-3

Towards efficient distributed service discovery in low-power and lossy networks

Badis Djamaa · Mark Richardson · Nabil Aouf · Bob Walters

Published online: 15 June 2014

Abstract Low-power and Lossy Networks (LLNs) have been recognised as a promising technology to achieve ubiquity in the internet of things era. To realise this, service oriented architectures and the emerging IPv6 over low-power wireless personal area network (6LoWPAN) standard are identified as key paradigms. One of the main elements to succeed any service oriented approach is a proficient service discovery protocol. In this paper, we propose EADP: an efficient protocol to announce and discover services in 6LoWPAN networks. EADP adopts a fully distributed approach using an adaptive push–pull model to ensure fast discovery times, low energy consumption and low generated overhead with timely reaction to network dynamics. EADP achieves this by using context-awareness information, delivered by a trickle algorithm. EADP was implemented and evaluated in Contiki using the Cooja simulator. Simulation results show EADP's capability to realise fast discovery times with low cost in terms of energy and overhead. These achievements make EADP very suitable for pervasive LLN applications.

Keywords 6LoWPAN · Low-power and lossy networks · Service discovery · Trickle algorithm · Web/internet of things

B. Djamaa (✉) · M. Richardson · N. Aouf · B. Walters
Cranfield University, Shrivenham SN6 8LA, UK
e-mail: b.djamaa@cranfield.ac.uk

M. Richardson
e-mail: m.a.richardson@cranfield.ac.uk

N. Aouf
e-mail: n.aouf@cranfield.ac.uk

B. Walters
e-mail: c.r.walters@cranfield.ac.uk

1 Introduction

Low-power and Lossy Networks (LLNs) such as Wireless Sensor Networks (WSNs) have proven significant use in multiple application domains over the last decade. With the reach of the Internet Protocol (IP) to LLNs, new applications are emerging to create the so called Internet of Things (IoT) in which LLNs are no longer seen as isolated proprietary networks. Instead, they are considered as a main part of the IoT architecture which integrates sensors, actuators, computers, PDAs, smartphones…etc. As these devices need to interoperate, Service Oriented Architecture (SOA) appears as a promising paradigm. In SOA, available capabilities are modelled as services. A service is a self-contained functionality provided by a service provider and used by service consumers conforming to a service contract (service description) [1].

To allow usability of provided services, devices (nodes) must be able to dynamically share, discover and manage service descriptions in a seamless way. For this reason, automatic Service Discovery (SD) that allows entities (persons/devices) to get access to the right service at the right time, is seen as a fundamental requirement of any pervasive, distributed, open and dynamic environment such as LLNs [2]. However, being able to discover services without the need for central servers or human administration introduces many challenges. Thus, different announcements and discovery strategies result in different amounts of resource consumption in terms of memory, energy and bandwidth. This may be aggravated when IP is the underlying layer such as in IPv6-enabled LLNs standardised as 6LoWPANs (IPv6 over Low-power Wireless Personal Area Networks) [3].

Designed to be integrated in a seamless way with traditional IP networks, 6LoWPANs adopt the use of

standard, interoperable protocols to overcome heterogeneity problems, making the cross-layer design unsuitable. In addition, the very limited power, memory and bandwidth of LLNs makes existing protocols designed for other ubiquitous environments inadequate. The mobility of nodes, introduces more challenges in these networks. Thus, the frequency of devices joining/leaving the network implies that services should be announced/deleted frequently which consumes high resources.

These constraints imply the need for a novel approach to deal with SD in 6LoWPANs which is still immature [4–6]. The computing, communication and power constraints make it critical to reduce and balance the energy consumption of both the devices and the network, minimise the amount of generated overhead and more importantly propose cost-effective mechanisms to detect the availability/unavailability of services as soon as devices join/leave.

Therefore, we propose in this paper which is built upon our previous work [7], a fully-distributed Efficient Application-layer Discovery Protocol (EADP) which has been designed to satisfy the requirements of both static and dynamic[1] IPv6-enabled LLNs bearing in mind the need for seamless integration with traditional IP networks. The main additional contributions of this paper are:

- Whilst [7] presented the preliminary version of the proposed push algorithm, this paper introduces a general version including an *n-inconsistency* approach.
- This paper proposes and evaluates a new algorithm for vanishing services which can cope and adapt to both static and dynamic networks.
- With the above mechanisms, this paper provides design extension, implementation details and in-depth discussions along with an application scenario.
- Furthermore, this paper proposes a comprehensive worst case analysis of the final version of EADP regarding latency, energy consumption and scalability with the number of nodes and services.
- Finally, whilst [7] presented preliminary results of EADP's earlier version in static networks, this paper extends the evaluation to other aspects of the protocol such as false discoveries and evaluates its performance under new metrics including mobility, varying number of services, nodes' speeds and network densities.

The remainder of this paper is organised as follows: Sect. 2 will be devoted to review existing SD protocols and discuss their limitations to pervasive 6LoWPANs. This will be followed by presenting the motivating scenario in Sect. 3. Section 4 will introduce EADP, its design and main components. A formal worst case analysis of the proposed push algorithm will be presented in Sect. 5. Experimental evaluation, results and discussions will be the subject of Sect. 6. The paper ends with a conclusion and suggestions for future works.

2 Related work

To ensure proficient service discovery, traditional SD protocols such as Service Location Protocol (SLP) [8] and Universal Description, Discovery and Integration (UDDI) [9] rely on central directories to store information about all available services. While this trend is vital for an infrastructure based environment, it is hardly applicable in distributed, constrained and dynamic environments. Thus, distributed directories and directory-less approaches were explored by previous research in the field [5, 10–12].

Distributed directories approaches employ clustering techniques to form a distributed backbone of directories. These approaches generally assume the availability of some resource-rich nodes and need synchronisation between them, which implies a high maintenance overhead. This added to the high discovery overhead generated by re-clustering and re-registration makes the distributed directories approaches less suitable for dynamic IP-enabled LLNs [5].

2.1 Directory-less service discovery

In directory-less approaches, nodes use multicast/broadcast of service requests/advertisements to realise service discovery. Three approaches are considered in the literature namely: push models; pull models; and hybrid models. In push models, service providers make use of multicast to proactively send unsolicited advertisements of services, whilst clients just listen and select the most appropriate ones. For instance, DEAPspace [13] adopts a pure push model aimed at single-hop networks. Applying the push model makes the network aware of new services as soon as they appear, allowing clients to find them in less time, which gives more time for invoking selected services. However, while the push model reduces the latency, it introduces a large amount of traffic, making it unsuitable for highly dynamic networks [14]. By contrast, in pull models, clients issue requests to be propagated across the network on-demand for a service. Upon the reception of a request, a matching provider generates a reply message containing descriptions of matching services and sends it

[1] Dynamic LLNs encompass mobile networks, networks with frequent nodes and services *churn* and networks with unreliable links. Thus, networks with stable topologies, small *churn* and good links can be considered as static.

back to the client. For instance, the SLIM framework [10] incorporates a pull model discovery protocol allowing nodes to discover provided services. Where from a latency point of view, the pull model is less efficient, it is more suitable for dynamic environments as it generates less traffic. However, being inefficient in latency makes pull model protocols less reactive to highly dynamic environments. Thus a service may appear and upon discovering it, the service disappears.

For this reason, minimising the discovery latency is a key goal of EADP, bearing in mind that an uncontrolled push model affects the protocol capacities in reacting to dynamic environments. Thus, a hybrid adaptive push–pull protocol is needed to benefit from both aforementioned approaches. Many ad hoc protocols using hybrid mechanisms are proposed both in WSNs and Mobile Ad hoc Networks (MANETs) such as PDP [15], ADDER [14] and NanoSD [11]. In PDP, nodes maintain, in a local directory, service descriptions that were previously announced and advertise some of them (push-mode) when they issue requests (pull-mode). Also, PDP nodes forward replies using broadcast, instead of unicast, to enable other nodes to cache contained services. While this mechanism allows PDP to minimise push mode traffic, it may not assure optimising latencies. On the other hand, ADDER uses a hybrid approach in which nodes periodically advertise services in their vicinities and clients issue requests to locate services which are not available locally. While this exploits the benefits of the push mode in terms of latency, it generates high push overhead. In addition, ADDER has scalability issues w.r.t the number of services included in one advertisement. To deal with this, ADDER uses a fixed probability to decide which services should be included. This may compromise the performance realised in terms of latency. NanoSD is another hybrid push–pull SD protocol designed to discover services in dynamic, mobile and heterogeneous WSNs. NanoSD focuses on techniques to minimise packet and service description sizes. However, like ADDER, NanoSD does not propose advanced forwarding mechanisms to minimise push traffic.

The above hybrid protocols suffer from high generated overhead and/or latency issues. To address this, location-based SD protocols [16–19] make use of node locations to minimise the amount of generated traffic. Finally, it should be noted that for energy efficiency reasons, integrated discovery protocols have been investigated [20–24]. The basic idea of this class of SD protocols is to integrate discovery traffic with the routing one. However, binding a SD protocol to a specific routing protocol violates SOA, in general, where interoperability is preferred over optimisation [25] and the 6LoWPAN architecture, in particular, where the layered design is the main characteristic allowing seamless integration with traditional IP networks.

2.2 Service discovery in 6LoWPANs

To deal with the problem of service discovery in 6LoWPANs, the Internet Engineering Task Force (IETF) has proposed a draft for a Simple Service Location Protocol (SSLP) [26]. SSLP relies mainly on a central directory to store service information, although it proposes a basic fully-distributed mechanism for small-size networks. The authors of [4] introduce the use of vicinity information to enhance service discovery in 6LoWPANs. The proposed protocol supports the same architecture used in SSLP and introduces the concept of Directory Proxy Agent (DPA) to manage user and service contexts. Network elements are organized in a hierarchical manner and multiple DPAs are considered to cache services information and context in their vicinity on behalf of the Directory Agent (DA). However, the proposed protocol provides a complex mechanism for accessing services from the Internet reducing its performance [5]. TRENDY [27] is a centralised service discovery protocol designed to discover services working over the Constrained Application Protocol (CoAP) [28]. TRENDY chooses the 6LoWPAN Border Router as DA and divides 6LoWPAN nodes to Group Leaders (GL) and Group Members (GM). In addition to TRENDY's unsuitability for dynamic 6LoWPANs, the approach of creating and maintaining GLs and GMs induces a high maintenance overhead. ENUM-based SD [5] aims to discover services from inside as well as outside a 6LoWPAN network. It employs a distributed-directory approach based on the idea of resource-rich master nodes holding information of services available in their clusters. For its functioning, ENUM-based SD requires the presence of powerful nodes to play the role of masters which cannot be assumed in all 6LoWPAN networks. NanoSLP is a fully-distributed discovery protocol for constraint networks developed within the NanoIP stack [12]. NanoSLP allows service discovery-delivery integration. Thus, it offers the possibility to get the requested value in the reply message which optimises the latency by saving one round-trip time. However, like ADDER and NanoSD, NanoSLP does not provide advanced forwarding mechanisms to minimise the amount of traffic. Recently, the IETF has proposed new discovery protocols for 6LoWPAN networks targeting mainly web of things applications. In this context, two main classes can be distinguished: (1) the class operating over CoAP where the IETF is working on the three following approaches: (a) a fully-distributed pull-only approach called resource discovery [29] which relies on the underlying IP-multicast routing; (b) a centralised approach called resource directory [30]; and (c) a distributed-directory based approach dubbed distributed resource directory [31]. 2) The second class being investigated is based on multicast Domain Name System (mDNS) [32] and

DNS-SD [33]. For instance, [34–36] looked at the feasibility of such an approach and identified points on adapting it to constrained networks. However, these protocols being proposed by the IETF are still works in progress and focus on static web of things applications.

Having shown the limitations of the existing SD protocols to pervasive 6LoWPANs, we designed EADP; an extensible adaptable service discovery protocol. EADP works at the Application layer, over UDP-IPv6, and it is intended to adapt to the whole span of 6LoWPAN networks. The main focus of EADP is to discover intra-6LoWPAN services.

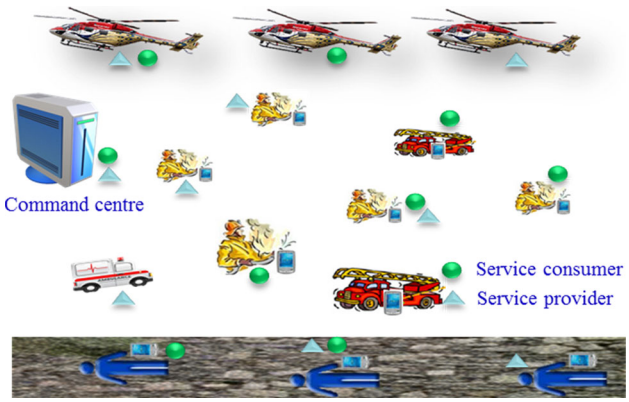

Fig. 1 Emergency response scenario

3 Motivating scenario

Our research scenario considers a LLN composed of a set of fixed and mobile nodes including sensor motes, actuators, 6LoWPAN communicators [37] and gateways toward a traditional IP network. A node in our system may consist of physically separated networked device (sensor, actuator…etc.) or be embedded in other devices (laptops, smartphones…etc.) and it may act as a service provider, service consumer or both. Many LLN applications including ubiquitous healthcare systems (e.g. the global healthcare monitoring system [38]), environmental monitoring (e.g. the farmyard application [11]) and smart logistics (e.g. the intelligent container project [39]) can fall under the above research scenario and hence may benefit from EADP. In the intelligent container, for instance, sensors attached to goods, shippers and receivers can use EADP to announce their services when they join a network, locate required services (e.g. gateways that provide access to other networks) and figure out the state of goods at any time.

Our specific scenario focuses on the ubiquitous emergency response application depicted in Fig. 1. During emergency response situations (evacuation of buildings, forest fires, field operations…etc.) decisions have to be made in a timely manner in order to save lives. In our scenario, unattended LLNs are deployed as static nodes to monitor the event and interact with mobile nodes attached to injured people, working teams, vehicles…etc. As nodes belong to different ownerships and have different dynamics, a discovery protocol which allows them to announce, discover and use available services at their current location is needed. The mobility of involved entities and the timely requirements of such an application, impose important technical issues on such a protocol that make building and distributing a global knowledge on the topology of the network unsuitable. This justifies our choice for a fully-distributed approach. In addition, such a SD protocol should be able to support all kind of LLN service interactions that can be categorized into four generic classes:

- Simple services: provide support for simple service interactions where a single data is needed at the moment of invocation (e.g. sensor readings: temperature, humidity…etc.). In this case, discovery and delivery can be integrated.
- Alert services: enable sending alerts when an abnormal situation occurs. The discovery is needed to search for services provided by gateways (e.g. translation, Internet access) in order to announce the event.
- Complex services: handle complex service interactions such as the history of a specific measure (e.g. temperature monitoring over the last hour) or complex sensor readings (e.g. multimedia services). This case requires the establishment of a communication session between the provider and the consumer and thus needs, in addition to discovery, an invocation phase.
- Multicast services: send a command to a group of nodes, configure the network…etc. In this case the discovery may not be required. A multicast invocation containing necessary attributes might be sufficient [10].

4 EADP description

EADP provides a service discovery mechanism targeting 6LoWPAN networks. In addition to its timely reaction to network dynamics, EADP has been designed to provide high discovery rates and fast discovery times with low network overhead and low energy consumption. To assure these qualities, EADP adopts a fully-distributed approach based on an adaptive push–pull model. The EADP architecture is made-up of three main components: a User Agent (UA) responsible for discovering services in the pull mode; a Service Agent (SA) responsible for registering and advertising services' information in the push mode; and a state maintenance mechanism responsible for managing nodes local directories and making the protocol react

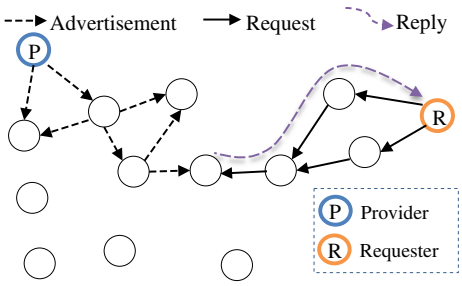

Fig. 2 EADP overview

seamlessly to network dynamics. For energy-efficiency reasons, EADP offers a fourth optional reverse-path routing mechanism for delivering unicast service replies. It should be noted that while EADP provides an adaptive mechanism for enabling/disabling the push mode, a possibility is always given to the end-user to decide to disable it. EADP overview is depicted in Fig. 2.

In EADP, when a new node joins a network, it starts by advertising its available services. Upon reception of such a packet, the receiving node's SA decides on the utility of each contained entry and adjusts, consequently, the push mode transmitting window using a trickle algorithm. At the transmission time, the SA includes in its outgoing advertisement useful local and remote stored services. To assure live-ness of stored services, EADP provides a state maintenance mechanism which deletes any stored service entry at the expiration of its TTL. In addition, the state maintenance mechanism provides an algorithm to forward explicit delete messages initiated by service providers. On the other hand, once a node needs to discover and use a network service, it calls its UA. The UA starts by inspecting the local directory in order to find the requested service; if found the discovery process finishes. Otherwise, a service request message will be generated and propagated across the network. Upon finding a service matching the requested criteria, corresponding node's UA generates a service reply message to be sent to the requester, as shown in Fig. 2. To deliver service replies, EADP proposes two mechanisms 1) use the underlying routing protocol or 2) exploit a reverse-path constructed when forwarding requests. Notice that the service description and matchmaking component of EADP is left generic. This way EADP is not forced to a single service description and query language but can adapt to multiple descriptions and languages.

In realising this process, EADP defines three mandatory packet types namely: advertisement, request, and reply plus an optional packet type: delete. All the packets share the same header, containing information about the protocol version and the message type. However, for the payload, while the request and delete messages can have fixed payload sizes containing, respectively, the necessary criteria about the requested service and the necessary information to uniquely identify a service, the advertisement and reply messages have variable payload sizes depending on the number and size of the service entries included in the message. Finally, and in addition to the trickle algorithm specific parameters [40], EADP defines the configuration parameters presented in Table 1.

Table 1 EADP configuration parameters

Configuration parameter	Meaning
REQUEST_DISK	The maximum number of hops a request is allowed to propagate. After this distance the request is aborted
ADVERTISEMENT_DISK	The maximum number of hops, from the provider, a service description is allowed to propagate
WAIT_RESPONSE_TIME	The time a requester waits for a reply. At its expiration, the requester may resend or abort its request
REQUEST_RETRANSMISSION_COUNTER	The maximum retries to resend a request. When it reaches zero, the request is aborted
TTL (Time to live)	The period of time a service entry is kept in a node's local directory. It should be a multitude of the push period (e.g. 2–5 times)

4.1 The user agent (UA)

When a node wants to use a service, it calls its UA. The UA checks firstly its local directory. If the service is found, the discovery is accomplished in a purely push mode. Otherwise, the UA initiates a service request and sends it out over the network using a limited flooding approach to assure fast and 100 % discovery. Simultaneously, the UA sets its request timer for WAIT_RESPONSE_TIME to wait for replies. At the expiration of the timer, if a response has not been received, the UA retransmits the same packet and decrements the REQUEST_RETRANSMISSION_COUNTER. When the counter reaches zero, the UA aborts the request and concludes that the service is either non-existent in the vicinity or unreachable. On the other hand, the UA is always listening for service requests, processing them and deciding to forward, abort or generate reply messages when necessary. Thus, upon reception of a request message, the UA matches it with the node's local directory entries. If a service matches the request, a reply is generated. Otherwise, the UA investigates the distance travelled by the entry and compares it

with REQUEST_DISK. Depending on the results, it decides whether to abort or forward the request. By adopting a limited flooding approach inspired by distance vector protocols, the UA minimises unproductive overhead and assures that a request will get forwarded at most once by an intermediate node. This is achieved thanks to loop free primitives based on the use of message sequence number and a request cache.

4.2 The service agent (SA)

The SA is responsible for controlling EADP push mode where nodes periodically advertise own and remote services stored in their local directories. To do so, the SA proposes and implements a new variant of the trickle algorithm [[40], [41]] in order to minimise the number and size of advertisements.

4.2.1 Trickle algorithm

A node using the trickle algorithm periodically broadcasts its data unless it has recently heard identical ones. As long as nodes agree on what data they have, trickle exponentially increases the transmission window. When data disagreements are detected, trickle starts transmitting more quickly. To realise this behaviour, and as by [41] 's notations, the trickle algorithm maintains three variables namely: a consistency counter c, an interval I and a transmission time t within I. In addition, it defines three configuration parameters namely: the minimum interval size *Imin*, the maximum interval size *Imax* and a redundancy constant k. When trickle starts, it sets c to zero, I to a random value between $[Imin; Imax]$ and picks t from $[I/2; I)$. Piking t from the second half of the I interval allows for a listen-only period which avoids the *short-listen problem* [40]. Whenever a node hears identical data (dotted lines in Fig. 3), it increments c. At time t, a node transmits (dark box in Fig. 3) if and only if c is less than k. Otherwise, the transmission is suppressed (grey box in Fig. 3). When I expires, trickle doubles the interval length until *Imax*. Finally, if a node hears an *inconsistent* data and I is greater than *Imin*, I is set to *Imin*. Otherwise, trickle does nothing. Whenever I is set (a new interval begins), c is reset to zero and t to a random value in $[I/2; I)$.

Trickle is used in many applications such as routing control traffic and reliable broadcast/dissemination. In each use-case, the objective is different and so is the performance. In the data dissemination use-case of trickle, control messages are exchanged between nodes to assure what we call *artificial periodicity*. This is similar to the use of trickle in the Multicast Protocol for LLNs (MPL) [42]. However, our use of trickle comes from the *natural periodicity* of service descriptions exchanged in the push mode.

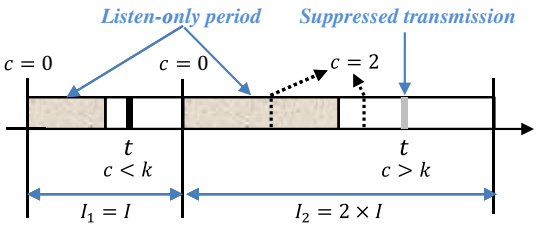

Fig. 3 Trickle algorithm over two intervals with k = 1

This is similar to its use in the Routing Protocol for LLNs (RPL) [43] and the Collect Tree Protocol (CTP) [44].

The trickle algorithm was originally designed to handle single data dissemination [45, 46]. To enable its use for many data items, protocols, in general, use two approaches[2]: the first establishes many parallel trickles while the second uses a single trickle that serially manages all data items. The two approaches have different characteristics and performance when compared with the original trickle. Parallel approaches, applied in [45, 48], introduce a control cost that increases linearly with the number of data items. Serial approaches, applied in [49], keep the control cost fix but make the latency increase linearly with the number of data items. However, the total cost of the two approaches scales linearly with the number of data items [47]. Another main criticism we address to the serial approach is its scalability w.r.t the size of a control packet. Thus in serial approaches a packet may exceed the Maximum Transmitting Unit (MTU) (only around 61 bytes are left at the application layer in 6LoWPANs). Finally, it should be noted that the mentioned usage of these approaches assumes a small number of data items [46].

Thus, we propose another variant to use trickle in hybrid push–pull protocols (Fig. 4). Unlike other trickle variants, our approach attaches the consistency counter to the data items (services). Thus, every data item in the node's local directory has a consistency counter which is updated following the rules defined by our approach. In addition, at the start of a new interval, a node only resets the consistency counters of its own services. While this does not assure a strict consistency, it fits well hybrid push–pull protocols and gives our algorithm very attractive proprieties. Thus, the main benefits of our algorithm are threefold: zero control overhead, constant maximum advertisement size and independent inconsistency detection latency. These benefits are realised thanks to the registration and advertising algorithms discussed below. Table 2 classifies our approach w.r.t to existing trickle variants.

[2] Other approaches combining trickle with other techniques are proposed in [46] and [47].

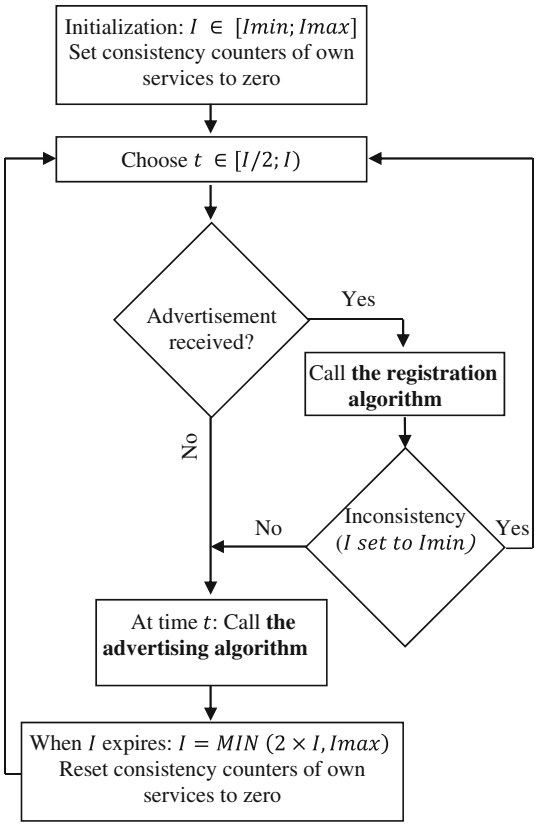

Fig. 4 The proposed SA's trickle algorithm

Table 2 Trickle usage in the literature

Data/periodicity	Natural	Artificial
Single data	[43, 44]	[40]
Multiple data	Our approach	[42, 45, 48, 49]

4.2.2 Service registration algorithm

Upon the reception of an advertisement message *adv_msg*, the receiver's SA starts the registration algorithm. Each entry in *adv_msg* represents a service description s appended with a metric m (distance in hops) and decorated with a sequence number f. The former of the two parameters is used to limit the entry's propagation; it is incremented by each forwarder, whilst the latter is used to assure loop-free transmissions and it is set and incremented only by the provider. Thus, an entry in *adv_msg* can essentially be presented by the vector (s,f,m).

The registration algorithm investigates the consistency of each entry (s,f,m) in *adv_msg*. A received service entry is considered as *consistent* when the corresponding service information is already in the node's local directory and considered as older, or announced as being far. Formally, a *consistent* entry verifies:

- The received entry (s,f,m) is already in the node's local directory, referred to as (s,f',m'), and has a lesser value of $f(f<f')$ OR;
- The received entry (s,f,m) is already in the node's local directory, referred to as (s,f',m') and has the same value of $f(f=f')$ and a greater or equal value of $m(m \geq m')$.

If an entry is identified as *consistent*, the registration algorithm only increments its consistency counter c in the node's local directory. Otherwise, the entry is *inconsistent* and hence feasible for registration (either the entry is not present in the node's local directory or the node received an update for an existing entry). The SA proceeds to the registration of such an entry for a TTL period, increments and updates its distance m and reinitialise its consistency counter c to zero.

For the *first inconsistent* entry in *adv_msg*, if the interval I is greater than *Imin*, the trickle timer I is set to *Imin*. This is so, to allow quick updates of the network about inconsistent services by quickly transmitting the next advertisement. It should be noted that applying the *first-inconsistency* approach to reset I assures a quick inconsistency detection time and allows for timely reaction to network changes. However, if minimising the cost is further more important than the application time requirements, we propose a more general *n-inconsistency* approach. In which, once receiving an *adv_msg*, the registration algorithm resets I if and only if n *inconsistent* entries are reached. To do so, the node keeps an inconsistency counter *ic* which is incremented every time an inconsistency appears. When *ic* reaches n ($ic = n$) and I is greater than *Imin*, I is set to *Imin*.

4.2.3 Advertising rules and protocol scalability

At time t (Fig. 4), the SA calls the advertising algorithm to form outgoing advertisements. The advertising algorithm includes in the outgoing advertisement message all entries whose distances are lesser than or equal ADVERTISEMENT_DISK and flagged as *inconsistent* in the node's local directory (with a consistency counter c equal to zero). These entries have not yet been made known to the network, thus their announcement is of great importance and hence they are prioritised. However, this could be insufficient to optimise discovery times in cases where the wireless transmission is unreliable (interferences, noise…etc.) and if the network is sparse or contains holes as can be seen in Fig. 5. In this figure, if node N_2 hears a consistent entry from N_1 and decides to suppress its transmission, a part of the network may not be updated quickly which may delay subsequent replies. Therefore, a redundancy constant k greater than one can be used to

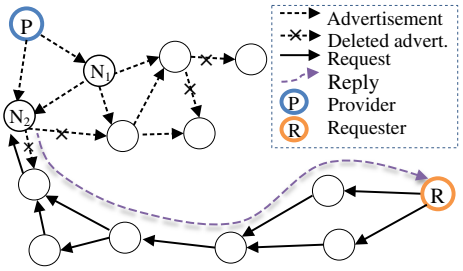

Fig. 5 EADP's service agent algorithm

include other estimated less useful entries. Hence, the outgoing message will be filled by other entries which verify, when sorted, the condition c is less than or equal to $k(c \leq k)$. Notice that this advertising algorithm not only minimises the number of advertisements, but also assures protocol scalability by controlling and minimising the number of services included in a single advertisement.

To illustrate the functioning of the above algorithms, let's take the example presented in Fig. 6 which depicts a network constructed of three nodes x, y and z. The state of the network, particularly node y, before receiving an *adv_msg* is depicted in Fig. 6 (a) which shows node y's local directory containing three service entries S1, S2 and S3 with their respective sequence numbers and distances from their providers. Upon receiving an *adv_msg* containing three service entries S1, S2 and S4 with their respective values of f and m (Fig. 6 (b)), the registration algorithm investigates the consistency of each entry. S1 is already present in node y's local directory and received with the same sequence number and distance $(f = f' \text{and } m = m')$thus the registration algorithm only increments its c counter. S2 is already present in node y's local directory but received with a new sequence number $(f > f')$, the registration algorithm updates it and resets its c counter to zero. If the *first-inconsistency* approach is employed, the trickle timer I is reset to *Imin*. S4 is new; the registration algorithm creates an entry for it with c equal to zero. At time t, the advertising algorithm goes through node y's local directory and includes in the outgoing advertisement entries with c counters less than k. Thus if a constant $k = 1$ and an ADVERTISEMENT_DISK $= 4$ are used, the outgoing advertisement contains S2 and S4 as illustrated in Fig. 6 (c).

4.3 The state maintenance mechanism

EADP proposes a state maintenance mechanism aiming to timely react to network dynamics and hence prevent erroneous storage and advertisement of services when they are no longer available. The state maintenance mechanism provides two primitives to keep the network updated about intended and/or unintended departures of nodes/services.

The first primitive offers the possibility for a provider to announce services departures via the optional delete message. In order not to flood the network with delete-packets, EADP implements an algorithm allowing local analysis of these messages. In fact, EADP uses the trickle algorithm proposed for controlling service advertisements (Sect. 4.2) to manage delete-packets forwarding. Hence, once a provided service become unavailable (e.g. fault of the sensing/actuating components…etc.) or the provider deliberately becomes unavailable (e.g. provider decides to leave the network, device reboots…etc.), it initiates a delete-packet to be sent in the network. For optimisation reasons, two considerations can be taken: (1) The outgoing packet may contain one or more service entries to be deleted or even service entries to be advertised if there are some, (2) when a provider is going to leave the network, it can send a delete-packet with zero entries to inform the network to delete all its services. Upon receiving a delete-packet, nodes delete corresponding service information from their local directories (all the service entries provided by the provider if the delete-packet contains zero-entry) then apply the above trickle algorithm to manage its forwarding. Using trickle to manage delete-packets forwarding saves energy, throughput and more importantly keeps the network updated about intentional departures of nodes.

While in static networks, exactly the same algorithm can assure vanishing stale services from the network, in mobile networks; however, limiting delete-packet forwarding to the ADVERTISMENT-DISK is impractical as nodes randomly move. Thus, to be able to reach all nodes, network-wide trickle forwarding should be used. In addition, some nodes which are not aware of the service to be deleted may receive the delete-packet. In this case, those nodes cache the delete-packet and will be considered in forwarding it. This is to balance the transmission costs and provide the possibility of reaching the nodes storing a copy of the service in question that can only be reached via non-aware nodes. Note that the speed of clearing the network form unavailable services depends on the trickle minimum interval *Imin*. Thus, one might use another trickle timer to manage delete-packets.

However, because the above mechanism will not work in the case of unintended departures of nodes, EADP can enforce the TTL vanishing by exploiting the underlying neighbour discovery protocol to keep EADP updated about the disappearance of nodes. This needs more discussions and it will be further investigated in future works.

4.4 The reverse path construction

EADP provides an alternative reverse-path routing mechanism to reduce the protocol's resources consumption. This mechanism exploits the path being traversed by the

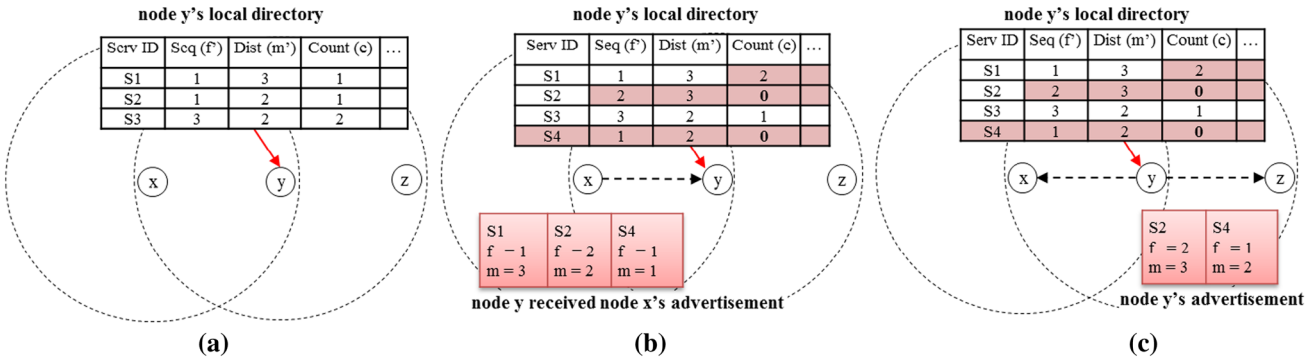

Fig. 6 An example of execution of the proposed push-algorithm **a** previous network state, **b** registration algorithm, **c** advertising algorithm

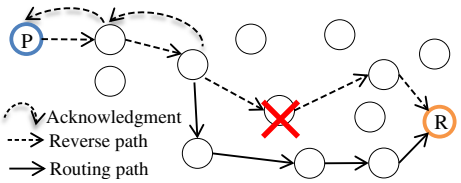

Fig. 7 The reverse-path routing

Table 3 Parameter used in the analysis

Parameter	Meaning
N	Number of nodes in the network
S	Number of services
Ns	Number of new services
k	Redundancy constant

requests, to route back the replies using a route-over [50] AODV-like routing mechanism [51]. The reverse-path provides three main advantages: (1) it eliminates the overhead generated by the underlying routing protocol; (2) it avoids delaying the responses when trying to establish routes, especially in the case of reactive routing protocols and; (3) it uses simple cost-effective primitives. To do so, it just implies the use of a limited-size routing table, having the following structure:

$< destination\_addr, next\_hop, distance >$

Additionally, when a link in the path is broken, the mechanism can detect it (using the acknowledgment mechanism) and calls the routing protocol to continue delivering the packet, as depicted in Fig. 7. In addition to its importance for resource saving, the reverse-path mechanism allows nodes constructing the path to seamlessly cache information about services contained in the replies and, hence, enhance subsequent requests' latencies. The reverse-path mechanism can always be disabled especially if the network provides a proactive routing protocol.

5 Formal analysis of the proposed push algorithm

In this section, we present a formal worst case analysis of the proposed push algorithm in the case of single-hop lossless networks. To generalise the conclusions to multi-hop unreliable networks, the methodology presented in [40] can be used. In this latter, it is proven that considering the imperfect nature of wireless links and the multi-hop nature of the network will add a cost that scales logarithmically with the number of nodes. In this study, we investigate the performance of the proposed push algorithm, over one transmitting window (interval), w.r.t: the number of exchanged advertisements, the size of an advertisement, the total amount of generated traffic and the inconsistency detection latency.

These four parameters aim to provide a comprehensive analysis of the time/cost behaviour of our solution. To do so, we state, in the following points, the problem and the assumptions taken. The parameters and notations used in this analysis are presented in Table 3.

- The analysis considers the *first-inconsistency* approach which obviously sends more messages than the *n-inconsistency* approach ($n > 1$).
- We assume that a provider provides at most one unique service. Without a loss of generality when a node provides more than one service. This assumption is taken to simplify the analysis.
- For ease of the analysis, we assume that all the services have the same size : *service_size*.

5.1 Worst case analysis

We present a worst case analysis of the number and maximum size of advertisements, the total amount of generated traffic and the detection latency of inconsistencies for our push approach, serial/parallel trickle

approaches, and fixed-period push algorithms such as the one used in ADDER.

In our solution and since nodes provide unique services, each node will send its new services even after it hears other nodes' *consistent* data. This makes our solution depend only on the number of new services ($O(Ns)$) occurring in an interval. This gives it a good scalability with the amount of exchanged messages as the number of new services in an interval is generally much less than the total number of services available in the network. In the serial and parallel approaches of trickle, the number of messages scale linearly with the number of services $O(S)$ [47]. Finally, in fixed-period push algorithms where each node transmits once in each interval, the number of exchanged messages scales linearly with the number of nodes, presenting thereby a worst messages scalability in $O(N)$.

With regards to the maximum size of an advertisement, our algorithm presents a very attractive characteristic. Thus, the maximum size of an advertisement generated by our approach is top-capped by a constant value equal to $(k + 1) \times service\_size$ i.e. the maximum number of entries included in an advertisement message is $k + 1$. This gives the proposed approach a good scalability with the size of an advertisement, in $O(1)$ which is in the same order of parallel trickle approaches [46]. However, using the ADDER advertising rule (probability = 1) where each advertisement contains all entries stored in a node's local directory, the size of an advertisement message might reach S. Thus, the size of an advertisement message in ADDER scales linearly with the number of service ($O(S)$). This creates scalability issues in handling large number of services. For serial trickle approaches, where the exchanged summary might contains all the information about the data stored in nodes' local directories, the size of a packet also scales linearly with the number of services ($O(S)$) [46].

With regards to the total amount of traffic generated in a transmitting window, which has a direct impact on the communication's energy consumption, our approach sends in a worst case approximately $(k + 1) \times Ns$. This comes in $O(Ns)$. In serial approaches where the number of messages is in $O(S)$, each contains in a worst case S entry, the amount of traffic generated in a transmitting window scales squarely with the number of services $O(S^2)$. The same analysis gives fixed-period push algorithms a total amount of generated push traffic in $O(N \times S)$. Finally, using the same analysis for parallel trickle approaches shows an amount of generated traffic in $O(S)$. Notice that a protocol which generates less amount of traffic saves more communication energy which is the main source of energy consumption in LLNs.

For inconsistency detection latency, the proposed approach can detect and react to an inconsistency in a fixed time span $O(1)$ which is in the same order as parallel trickle approaches and fixed-period push algorithms. However, the detection latency of serial approaches scale linearly with the number of services (O(S)) as shown in [47]. A summary of the performance is given in Table 4.

Table 4 comparative performance

Algorithm/parameters	# Messages	Max size	Traffic	Detection latency
Our approach	$\mathbf{O(Ns)}$	$\mathbf{O(1)}$	$\mathbf{O(Ns)}$	$\mathbf{O(1)}$
Parallel approach	$\mathbf{O(S)}$	$\mathbf{O(1)}$	$\mathbf{O(S)}$	$\mathbf{O(1)}$
Serial approach	$\mathbf{O(S)}$	$\mathbf{O(S)}$	$\mathbf{O(S^2)}$	$\mathbf{O(S)}$
Fixed-period push	$\mathbf{O(N)}$	$\mathbf{O(S)}$	$\mathbf{O(N \times S)}$	$\mathbf{O(1)}$

6 EADP performance evaluation

To consolidate and evaluate the mechanisms proposed in this paper, we implemented EADP in Contiki OS [52], an operating system dedicated to running IoT applications. In Contiki, a program can be directly run on a device, simulated or emulated. We emulated EADP behaviour using the Cooja simulator [53] and the emulated sky motes [54]. At the data link layer, we used ContikiMAC radio duty cycling [55] with a channel check rate of 8 Hz. To put EADP results in context, we compared it with ADDER [14]. This choice was driven by the fact that both EADP and ADDER adopt a hybrid approach working at the application layer over UDP-IPv6. However, while ADDER (Probability = 1) uses for its push mode a blind fixed transmitting window, EADP adjusts the window between $[Imin, Imax]$ and controls advertisement-size using the network context. Having in mind that the discovery time is proportional to the transmitting window, we compared EADP and ADDER having a transmitting window equal to *Imin*; the best window that EADP can reach (ADDER-b) and with ADDER with a transmitting window equal to *Imax*; the worst window that EADP reaches (ADDER-w). To see the impact of the push mode on EADP discovery times, we disabled it (EADP-d). By comparing EADP to a fixed-period algorithm using the best and the worst values of the EADP interval, we aim to cover the whole spectrum of its behaviour.

6.1 Experimental model and evaluation metrics

Since we are interested in ubiquitous emergency response applications, we used a reference scenario in which 100 sky motes were initially uniformly distributed in a square area of 350 m × 350 m. Twenty representative services, S1–S20, were distributed in the network. We considered a CBR traffic pattern in which a client (e.g. a member of the emergency team) generates a service request every 2 s looking for a service (e.g. location of an ambulance). To

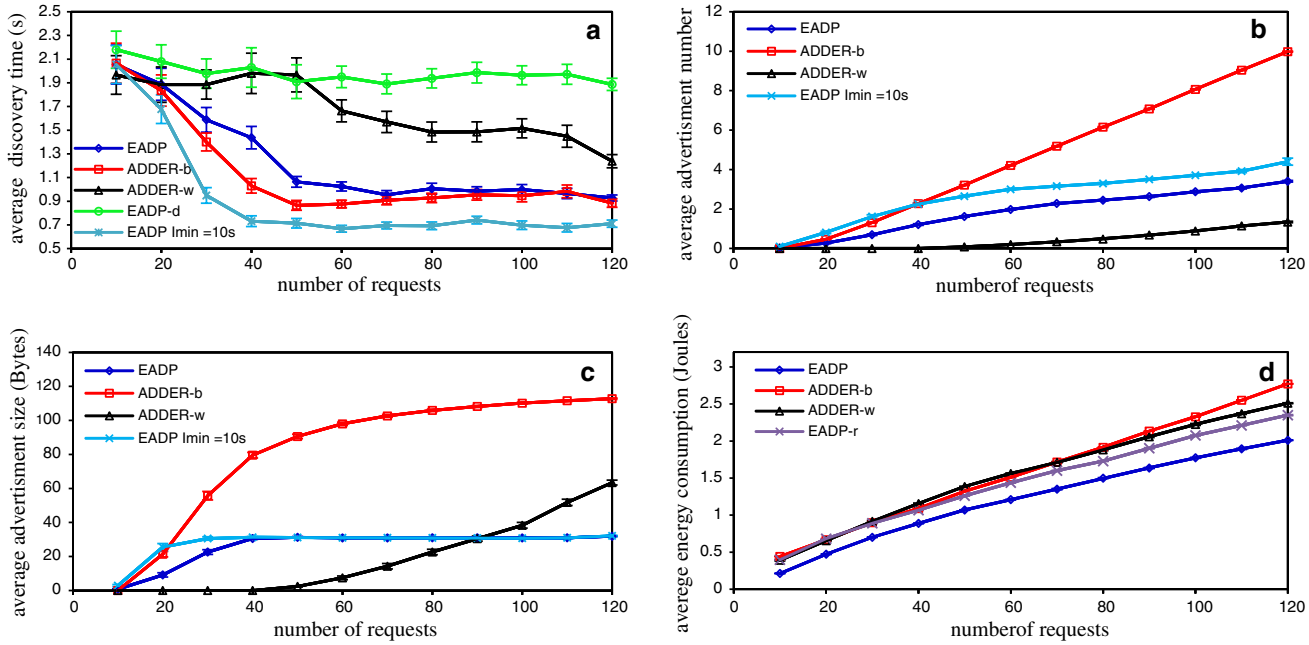

Fig. 8 EADP's time–cost performance

model the mobility of the entities involved in this scenario which includes humans, crew-cars slowly moving in the emergency area and static sensors deployed in and around the area, we used the random waypoint mobility pattern. Thus all the 100 nodes were mobile with a maximum speed of 4 m/s (minimum was 0 m/s) with random pauses between 2 and 10 s.

Starting from the above reference scenario, we analysed the performance of EADP under different conditions, changing, one by one: the execution time, the number of services, the network density, the maximum speed of nodes and the requests frequency. Our performance analysis focuses mainly on the trade-off between low push overhead and fast discovery times, thus we defined and measured the following metrics:

Average Discovery Time (*ADT*): ADT measures the latency realised by a discovery protocol to locate requested services. It is defined as the waiting time in milliseconds, averaged over all the requests, a client waits from transmitting a request to getting its first reply. In the mobile scenarios and because routing registered high packet loss ratios, we measured the average hit time instead.

Average Advertisements Number per node (*AAN*): AAN measures the capacity of a SD protocol to minimise the number of generated advertisements. It is defined as the ratio of the total number of advertisements sent during the simulation time to the number of nodes.

Average Advertisements Size per node (*AAS*): In EADP and ADDER the advertisement size is variable. A protocol which generates big advertisements not only increases the amount of traffic but also suffers from scalability issues when the number of services increases and risks to exceed the physical MTU. Thus, AAS is of great importance in such protocols and it is defined as the ratio between the sum of the average advertisement size for each node and the number of nodes.

Average Energy Consumed per node (*AEC*): AEC measures how much energy, in average, a node consumes in order to realise discovery tasks. It is calculated as the ratio between the total network energy consumed during the simulation time, and the number of nodes in the network. To measure AEC, we used Contiki power profiler [56].

The presented results are the average obtained by running each simulation 10 times, changing in each time the seed of the random number generator. Thus, we plot in Figs. 8, 9 and 10 the sample mean and its standard error. Table 5 summarizes the parameters used and varied in our simulations.

6.2 Results and discussions

6.2.1 EADP time–cost performance

To see the impact of the push mode on EADP time/cost performance with the course of time, we disabled the mobility and measured the discovery times, the number and size of generated push overhead and the consumed energy of EADP ADDER-b, ADDER-w and EADP-d (the EADP pure pull version). We also evaluated EADP with an $Imin = 10s$ in order to show its capacities to realise both the best times and costs compared to other evaluated protocols.

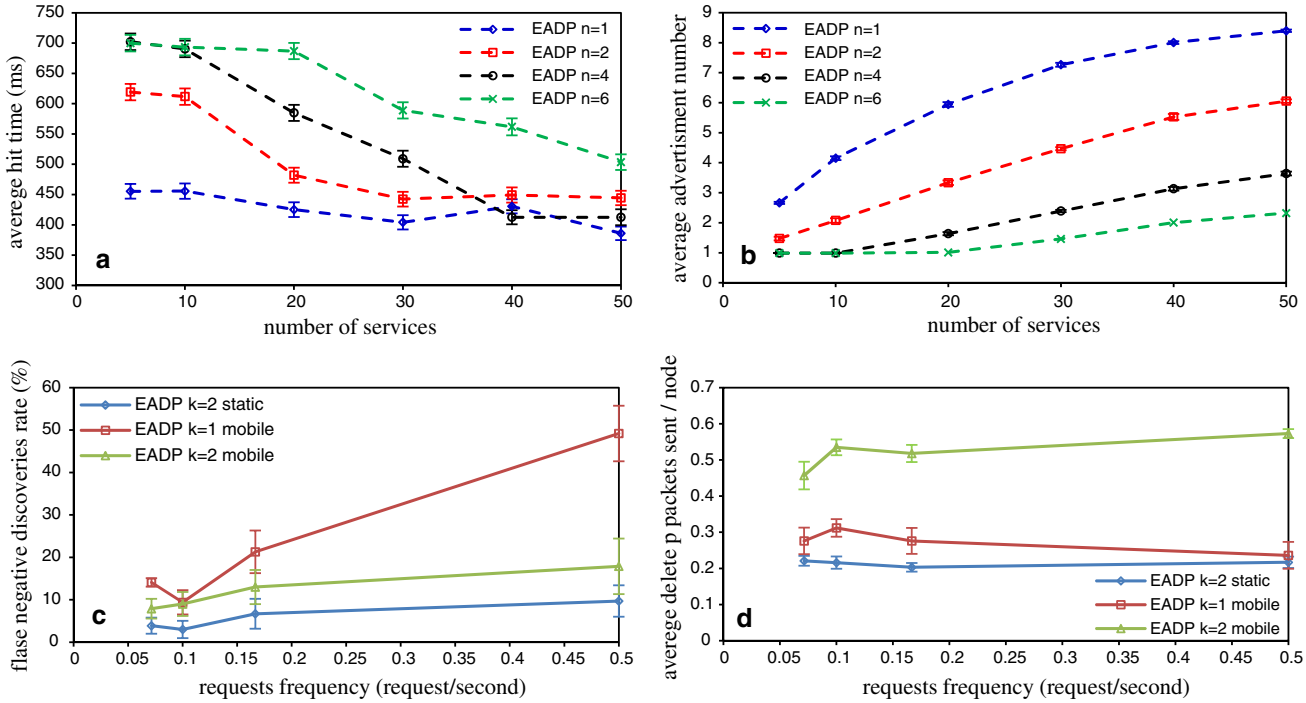

Fig. 9 Evaluation of the n-inconsistency approach and the sate maintenance mechanism

As can be seen from Fig. 8 (a), queries would, initially, traverse the network to find requested services, thus the average service discovery time was slow. This stays constant for EADP-d over the course of time (increasing number of requests) as a consequence of disabling the push mode. However, it gets faster in the other evaluated protocols. This is realised thanks to enabling the push mode, which allowed information about services to be propagated and stored throughout intermediate nodes enabling them to answer subsequent requests. Nevertheless, by using the largest window, ADDER-w presented slower response times. Compared with ADDER-b, EADP registered comparable results especially at the network convergence (more than 60 requests). However, if faster discovery times are further required, a smaller value of *Imin* can be used. Thus, EADP with $Imin = 10s$ plot presented the best discovery times. This is achieved thanks to exploiting information about the network-context to adjust the transmitting window which allows the necessary information to be propagated as soon as it appears.

In ADDER-b, achieving good discovery times came at a high number of generated overhead which scales linearly with the number of known services, as shown in Fig. 8 (b). Whilst EADP generated more messages than ADDER-w, it sent far fewer messages than ADDER-b and provided comparable discovery times. It even sent quite less messages when using an $Imin = 10s$ while presented the best discovery times. This was achieved thanks to adapting the transmitting window to the network-context and sending only utile services which minimised considerably the number of nugatory advertisements.

With regards to the impact of the number of known services on the size of an advertisement, Fig. 8 (c) shows that EADP presented a lower advertisement size which converged and stayed constant, at less than 40 bytes. This is even better than ADDER-w which sent fewer advertisements (Fig. 8 (b)). Thus in ADDER-w and since each advertisement blindly contains all known entries, its size kept increasing with the course of time and exceeded the one of EADP. This makes it less reactive to increasing number of known services although it does not achieve good discovery times. However, ADDER-b achieved good discovery times, but with the biggest average advertisement size, making it the worst in scalability terms as a result of sending big advertisements most often. Notice that the size of advertisements sent by EADP with an $Imin = 10s$ (which presented the best discovery times) also converged and stayed constant at the same size showed by EADP. This strengthens the analysis presented in Sect. 5.

Finally, to evaluate the impact of our reverse-path routing mechanism on EADP's energy consumption, we disabled it and used the proactive routing protocol RPL [43] to deliver unicast responses (EADP-r). As can be seen from Fig. 8 (d) the protocols using RPL (EADP-r, ADDER-b, ADDER-w) registered comparable energy consumption values at the beginning (less than 60 requests). This is explained by the fact that at the start of the network, the routing generated traffic was more important than the discovery one, which

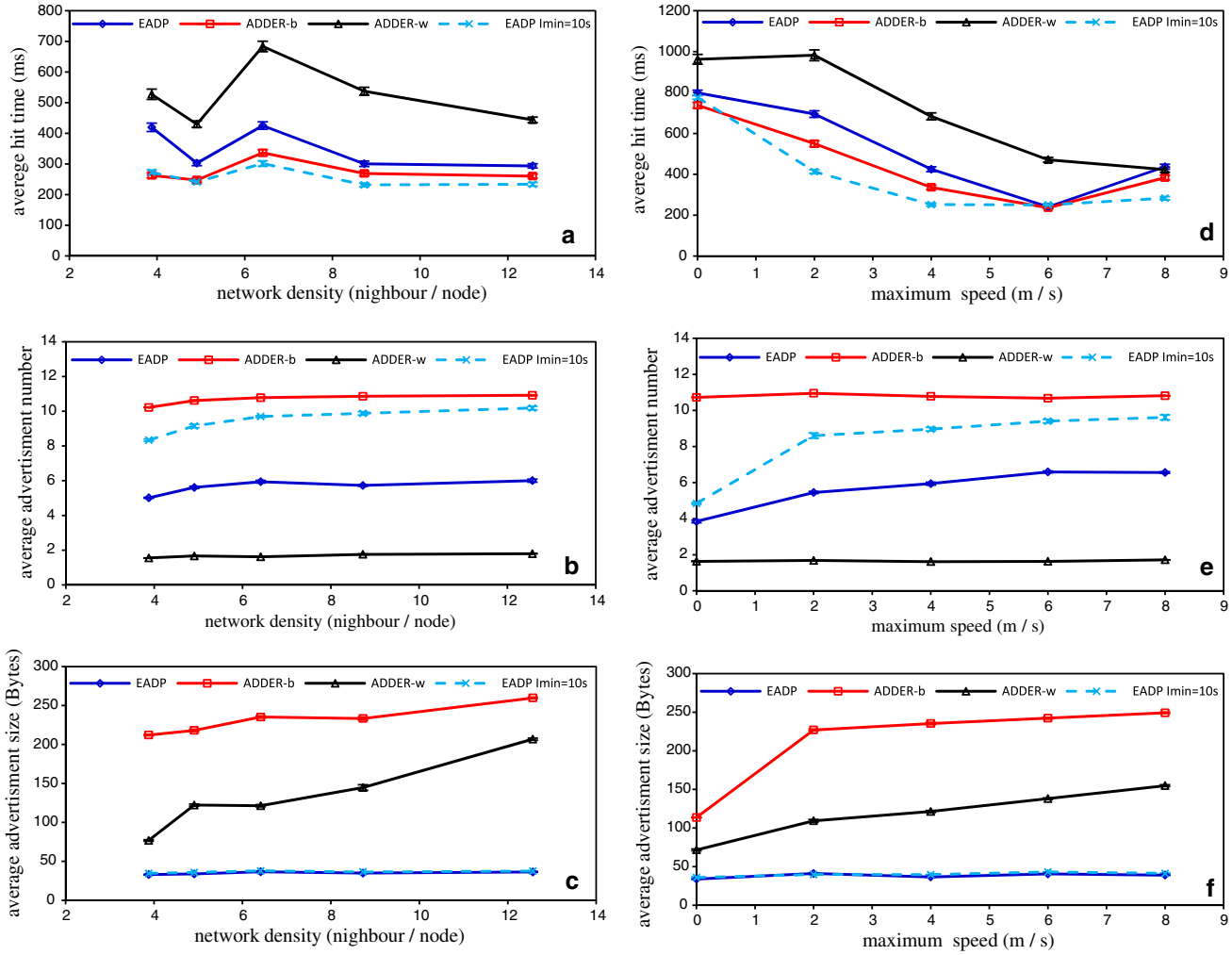

Fig. 10 Impact of nodes' speeds and density on EADP performance

made the protocols present approximately similar energy consumptions. However, after establishing the routes (values greater than 60 requests) and hence when the discovery traffic was more important than the routing control one, EADP-r presented the lowest energy consumption compared to ADDER-b and ADDER-w; this was done thanks to mechanisms which minimised both the number and the size of generated push traffic, which saved the energy that would be consumed by transmitting such packets. On the other hand, by comparing EADP using our reverse-path with its version using RPL (EADP-r), the reverse-path mechanism showed better energy savings. This is done thanks to avoiding routing control traffic.

6.2.2 *Evaluation of the n-inconsistency approach*

To evaluate the *n-inconsistency* approach, we varied in the reference scenario, the number of available services between 5 and 50 services. We measured the average hit times and the average number of generated advertisements, at the end of the simulation, for values of n equal to 1, 2, 4 and 6. Obtained results are depicted in Fig. 9 (a) and (b).

As can be seen from these graphs, the *first- inconsistency* approach realised better hit times thanks to transmitting inconsistent service information as soon as it is available. This came at the cost of more generated traffic (Fig. 9 (b)). For a small number of services (less than 20), the *first-inconsistency* approach realised the best hit times as it advertises as soon as an inconsistent service appears. The other approaches wait to gather *n inconsistent* services which takes time especially when there is less and dispersed services. For more than 20 services, as the environment became service-rich, the *2-incosistency* approach realised similar hit times to the first-inconsistency approach (Fig. 9 (a)) with a good gain in the number of advertisements (Fig. 9 (b)). This is done thanks to aggregating services in one packet. A similar behaviour can be seen with the *4-inconsitency* approach when the environment was more service-rich (more than 40 services).

Table 5 Simulation parameters

Parameter	Values
Duration of a one simulation	300 s
Number of Iterations	10
Number of nodes	100
Medium/Transmission range/Throughput	UDGM/50 m/250 kbps
EADP's $[Imin, Imax]$	[20 s; 80 s]
ADDER probability/EADP constant k	1/1
Network area (x, y)	350 m × 350 m,
Mobility model	Random way-point
Min–max speed/min–max pause periods	[0 m/s; 4 m/s]/[2 s; 10 s]
Traffic pattern/Requests frequency	CBR/0.5 request/second
REQUEST_RETRANSMISSION_COUNTER	0
REQUEST_DISK/ADVERTISEET_DISK	6/4
PHY and MAC	IEEE 802.15.4/CSMA
Adaptation layer	6LoWPAN

6.2.3 Evaluation of the state maintenance mechanism

To evaluate the state maintenance's delete-packets forwarding algorithm, we performed an experiment in which the client sent 150 requests with different frequencies (1 request each 2, 6, 10 and 14 s). We triggered a delete-packet for an already advertised service and we measured the false negative percentage as the ratio of the number of hits counted after triggering the delete-packet to the number of sent requests (Fig. 9 (c)); and the cost as the average number of generated packets per node (Fig. 9 (d)). We did this, following the specifications introduced in Sect. 4.3 for the initial static network (using a redundancy constant k equal to 2) and for the reference scenario, with $k = 1$ and $k = 2$.

As can be seen from Fig. 9 (c), the false discovery percentage increased with increasing request frequencies. The algorithm registered better results with a redundancy constant $k = 2$ which allows nodes that did not receive the delete-packet in the first transmission to get it in the second one. Thus, while in the static environment the mechanism registered its best performance with less than 10 % false in the highest request frequency tested (1 request each 2 s), in the mobile scenario, the mechanism registered a value of about 15 % negative false discovery with the highest request frequency with $k = 2$ (the lowest was about 7 %). This performance was realised with an average cost of about half a packet per node (Fig. 9 (d)). This is half the cost of a flooding algorithm, which theoretically can assure zero false negative. However, practically it suffers from the *broadcast storm problem* [57]. In addition, its implementation requires more memory and computing resource and adds to the complexity of the protocol.

6.2.4 The impact of network density on EADP

In this experiment, we show the impact of the network density on EADP performance when compared to ADDER-b and ADDER-w. As the density can be defined by: $density = N \times (\pi \times communication\_range^2/area)$ [58] and instead of varying the number of nodes, we vary in the reference scenario the side length of the square area in steps of 50 m from 250 to 450 m. Obtained results are depicted in the first column of graphs in Fig. 10.

As can be seen from Fig. 10 (a), EADP showed comparable hit times to ADDER-b with about half the number of advertisements generated by ADDER-b (Fig. 10 (b)) each containing less than 20 % of data (Fig. 10 (c)) in a density of about 3 neighbours per node. This is equivalent to the cost (number by size of advertisements) realised by ADDER-w which showed the worst hit times. With increasing densities, ADDER showed a high increase in the size of exchanged advertisements (Fig. 10 (c)). For instance, in a network of 12.5 neighbours per node, ADDER-w and ADDER-b advertisement sizes were, respectively, more than 6 and 7 times the ones of EADP which kept the size of advertisements constant at less than 40 bytes. This suggests that EADP resists increasing network density while ADDER might not. This conclusion is also confirmed by EADP with $Imin = 10s$ plots. Thus, these plots show that using a minimum interval size of $10seconds$, allows EADP to present the best performance in hit times and costs (in terms of the number by the size of exchanged advertisements) while kept the size of an advertisement constant at the same value realised by EADP with increasing node densities.

6.2.5 The impact of nodes' speeds on EADP

In this experiment we study the impact of nodes' speeds on the performance of EADP, ADDER-b and ADDER-w. Thus, we changed in the reference scenario the maximum speed by steps of 2 m/s from 0 to 8 m/s. Obtained results are depicted in the second column of graphs in Fig. 10.

With increasing speeds, the cost generated by the evaluated protocols increased. However, while the number of ADDER advertisements (Fig. 10 (e)) remained approximately constant as nodes only send once per period, it increased slightly in EADP, but only measured about half the one generated by ADDER-b at a speed of 8 m/s. Nevertheless, the size of an advertisement highly increased in ADDER. Thus with increasing speeds, nodes quickly carry services form one place to another making them stored in many nodes. Since ADDER blindly includes all stored

Table 6 Practical discovery rate

Protocol	EADP-d (%)	EADP (%)	ADDER-w (%)	ADDER-b (%)
Discovery rate	74.16	87.5	78.33	85

entries in its advertisements, their sizes keep increasing to contain all available services towards the end of the simulation (Fig. 10 (f)). EADP on the other hand, kept the size of its advertisements constant with increasing speeds while showed comparable hit times to ADDER-b (Fig. 10 (d)). For instance, at a speed of 6 m/s, EADP sent about half the number of ADDER-b advertisements, each containing about 6 times less data. This cost is equivalent to the one realised by ADDER-w which registered the worst hit times. Notice that with an $Imin = 10s$, EADP can assure better hit times than ADDER-b with about 7 times less cost in a speed of 8 m/s. This suggests that EADP robustly resists increasing speeds while ADDER risks, in addition to wasting resources, to exceed the transmission buffer.

6.2.6 The practical discovery rate

Whilst, theoretically, the discovery rate of flooding pull protocols is expected to be 100 %, practically this is not always the case, as shown in Table 6 which presents the average practical discovery rates of EADP, ADDER-b, ADDER-w and EADP-d. This can be explained by losing requests/replies packets caused by collisions as a result of high traffic circulating for long distances. Thus, enabling the push mode allows service information to be proactively propagated and stored by intermediate nodes which increases the practical discovery rate. From Table 6, EADP-d registered the worst rate because of disabling the push mode making requests and replies travel long distances increasing thereby the loss probability. ADDER-w showed the second worst discovery rate as it slowly propagated service information. EADP measured the best rate compared to ADDER-b as in addition of propagating services fast, it minimises the number and size of the push mode messages, which reduces network load, thereby allowing more requests and replies to be delivered.

7 Conclusion and future works

In this paper we proposed, designed and evaluated EADP; an energy-efficient, context-aware application-layer discovery protocol targeting IPv6-enabled LLNs. EADP registered good performance, especially in realising a trade-off between fast discovery times and low overhead while assuring a 100 % discovery of available services. However, many enhancements can be added to EADP, especially in controlling the pull mode based on limited flooding which in addition to consuming resources, generates many responses for the same request. Enhancements in this context should be done with respect to the discovery time and rate of the protocol. Using more context and usage information (e.g. allowing context-aware decisions on the utility of each service entry) can constitute an important enhancement to the advertising algorithm. Finally, to make EADP able to meet worldwide SD in the internet of things era, a service description fitting the constraints of 6LoWPANs is required. Our future work consists of developing an advanced pull mode forwarding mechanism, and then tackles the service description and matchmaking component of EADP.

References

1. Erl, T. (2008). *Soa: Principles of service design* (Vol. 1). Upper Saddle River: Prentice Hall.
2. Tianfield, H. (2011). Context-aware service discovery in pervasive environments: A survey. *International Transactions on Systems Science and Applications, 7*(3/4), 314–338.
3. Kushalnagar, N., Montenegro, G., & Schumacher, C. (2007). IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals. http://www.rfc-base.org/txt/rfc-4919.txt.
4. Chauhdary, S. H., Cui, M., Kim, J. H., Bashir, A. K., & Park, M.-S. (2008) A context-aware service discovery consideration in 6LoWPAN. In *Third international conference on convergence and hybrid information technology*. ICCIT'08, 2008, vol. 1, pp. 21–26.
5. Anwar, F. M., Raza, M. T., Yoo, S.-W., & Kim, K.-H. (2010). ENUM based service discovery architecture for 6LoWPAN. In *2010 IEEE wireless communications and networking conference (WCNC)*, pp. 1–6.
6. Wang, X., & Huang, H. (2013) A service model for 6LoWPAN wireless sensor networks. *International Journal of Distributed Sensor Networks*.
7. Djamaa, B., & Witty, R. (2013). An efficient service discovery protocol for 6LoWPANs. *Science and Information Conference (SAI), 2013*, 645–652.
8. Guttman, E., Perkins, C., Veizades, J., & Day, M. (1999) Service location protocol, version 2. http://www.ietf.org/rfc/rfc2608.txt.
9. Bellwood, T., Clément, L., & von Riegen, C. (2003) UDDI Version 3.0.1. http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.
10. Cugola, G., & Margara, A. (2010). SLIM: Service location and invocation middleware for mobile wireless sensor and actuator networks. *International Journal of Systems and Service-Oriented Engineering, 1*(3), 60–74.
11. Kovacevic, A., Ansari, J., & Mahonen, P. (2010) NanoSD: A flexible service discovery protocol for dynamic and heterogeneous wireless sensor networks. In *2010 Sixth international conference on mobile ad-hoc and sensor networks (MSN)*, pp. 14–19.
12. Jardak, C., Meshkova, E., Riihijarvi, J., Rerkrai, K., & Mahonen, P. (2008) Implementation and performance evaluation of nanoIP protocols: Simplified versions of TCP, UDP, HTTP and SLP for wireless sensor networks. In *IEEE wireless communications and networking conference, 2008. WCNC 2008*, pp. 2474–2479.
13. Nidd, M. (2001). Service discovery in DEAPspace. *IEEE Personal Communications, 8*(4), 39–45.

14. Oikonomou, G., Phillips, I., Guan, L., & Grigg, A. (2011) ADDER: Probabilistic, application layer service discovery for MANETs and hybrid wired-wireless networks. In *Communication networks and services research conference (CNSR), 2011 Ninth Annual*, 2011, pp. 33–40.
15. Campo, C., García-Rubio, C., López, A. M., & Almenárez, F. (2006). PDP: A lightweight discovery protocol for local-scope interactions in wireless ad hoc networks. *Computer Networks, 50*(17), 3264–3283.
16. Kaur, M., Bhatt, S., Schwiebert, L., & Richard, G. G. (2008) An efficient protocol for service discovery in wireless sensor networks. In *2008 IEEE GLOBECOM workshops*, pp. 1–6.
17. Li, X., Santoro, N., & Stojmenovic, I. (2009). Localized distance-sensitive service discovery in wireless sensor and actor networks. *IEEE Transactions on Computers, 58*(9), 1275–1288.
18. Gasparovic, B., & Mezei, I. (2012) Improving iMesh based service discovery by agents in multi-hop wireless sensor and actuator networks. In Telecommunications Forum (^{TEL}FOR), 2012 20th, pp. 611–614.
19. Lukic, M., & Mezei, I. (2012). Distributed distance sensitive iMesh based service discovery in dense WSAN. In X.-Y. Li, S. Papavassiliou, & S. Ruehrup (Eds.), *Ad hoc, mobile, and wireless networks* (pp. 435–448). Berlin: Springer.
20. Heni, M., & Bouallegue, R. (2011) Adaptive service discovery and proactive routing protocol for Mobile Ad hoc Network. In *Mediterranean microwave symposium (MMS)*, 2011 11th, pp. 193–196.
21. Raghavan, B., Harju, J., & Silverajan, B. (2008). Service discovery framework for MANETs using cross-layered design. In *Proceedings of the 6th international conference on wired/wireless internet communications*, Berlin, Heidelberg, pp. 152–163.
22. Ververidis, C. N., & Polyzos, G. C. (2009). A routing layer based approach for energy efficient service discovery in mobile ad hoc networks. *Wireless Communications and Mobile Computing, 9*(5), 655–672.
23. Sailhan, F., & Issarny, V. (2005) Scalable service discovery for MANET, presented at the international conference on pervasive computing and communications: PerCom 2005, pp. 235–244.
24. Fernandes, P., & Rocha, R. M. (2008). Information networking. In T. Vazão, M. M. Freire, & I. Chong (Eds.), *Towards ubiquitous networking and services* (pp. 396–405). Berlin: Springer.
25. SOA Manifesto. http://www.soa-manifesto.org.
26. Kim, K.-H., Baig, W., Mukhtar, H., Yoo, S., & Park, S. Simple service location protocol (SSLP) for 6LoWPAN. http://tools.ietf.org/html/draft-daniel-6lowpan-sslp-02.
27. Butt, T. A., Phillips, I., Guan, L., & Oikonomou, G. (2012) TRENDY: An adaptive and context-aware service discovery protocol for 6LoWPANs. In *Proceedings of the third international workshop on the web of things*, Newcastle, UK, pp. 2:1–2:6.
28. Bormann, C., Castellani, A. P., & Shelby, Z. (2012). CoAP: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing, 16*(2), 62–67.
29. Shelby, Z. (2012) Constrained RESTful Environments (CoRE) Link Format.
30. Shelby, Z., Krco, S., & Bormann, C. *CoRE resource directory*. http://tools.ietf.org/html/draft-ietf-core-resource-directory-00.
31. Jimenez, J., Liu, M., & Harjula, E. (2013) *A distributed resource directory (DRD)*. Pearson Education.
32. Cheshire, S., & Krochmal, M. (2013) Multicast dns. Work Prog.
33. Cheshire, S., & Krochmal, M. (2013) DNS-based service discovery. Work Prog.
34. Jara, A. J., Martinez-Julia, P., & Skarmeta, A. (2012) *Light-weight multicast DNS and DNS-SD (lmDNS-SD): IPv6-based resource and service discovery for the web of things*, pp. 731–738.
35. Klauck, R., & Kirsche, M. (2012) Bonjour contiki: A case study of a DNS-based discovery service for the internet of things. In *Proceedings of the 11th international conference on ad hoc, mobile, and wireless networks*, Berlin, Heidelberg, pp. 316–329.
36. Klauck, R., & Kirsche, M. (2013). Enhanced DNS message compression: Optimizing mDNS/DNS-SD for the use in 6LoWPANs. In *2013 IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops)*, 2013, pp. 596–601.
37. Sarwar, U., Sinniah, G. R., Suryady, Z., & Khosdilniat, R. (2010) Architecture for 6LoWPAN mobile communicator system. In *International multi conference of engineers and computer scientists*, Hong Kong, vol. II.
38. Singh, D., Tiwary, U. S., Lee, H.-J., & Chung, W.-Y. (2009) Global healthcare monitoring system using 6Lowpan Networks. In *Proceedings of the 11th international conference on advanced communication technology–volume 1*, Piscataway, NJ, USA, 2009, pp. 113–117.
39. Intelligenter Container: Home. http://www.intelligentcontainer.com/en/home.html.
40. Levis, P., Patel, N., Culler, D., & Shenker, S. (2004) Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the first USENIX/ACM symposium on networked systems design and implementation* (NSDI, 2004), pp. 15–28.
41. Levis, P., Clausen, T., Hui, J., Gnawali, O., & Ko, J. (2011) *The trickle algorithm*. Internet Eng. Task Force RFC6206.
42. Hui, J., & Kelsey, R. (2013) *Multicast protocol for low power and lossy networks (MPL)*.
43. Brandt, A., Designs, S., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J. P., & Alexander, R. (2012). *RPL: IPv6 routing protocol for low-power and lossy networks*.
44. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., & Levis, P. (2009) Collection tree protocol. In *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pp. 1–14.
45. Hui, J. W., & Culler, D. (2004). The dynamic behaviour of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on embedded networked sensor systems*, pp. 81–94.
46. Lin, K., & Levis, P. (2008) Data discovery and dissemination with dip. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 433–444.
47. Dang, T., Bulusu, N., Feng, W.-C., & Park, S. (2009) Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks. In *Wireless sensor networks*, Springer, pp. 327–342.
48. Tolle, G. & Culler, D. (2005) Design of an application-cooperative management system for wireless sensor networks. In *Wireless sensor networks, 2005. Proceedings of the second European workshop on*, 2005, pp. 121–132.
49. Gnawali, O., Jang, K.-Y., Paek, J., Vieira, M., Govindan, R., Greenstein, B., Joki, A., Estrin, D., & Kohler, E. (2006) The tenet architecture for tiered sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 153–166.
50. Chowdhury, A. H., Ikram, M., Cha, H.-S., Redwan, H., Shams, S. M., Kim, K.-H., & Yoo, S.-W. (2009). Route-over vs Mesh-under Routing in 6LoWPAN. In *Proceedings of the 2009 international conference on wireless communications and mobile computing: Connecting the world wirelessly*, pp. 1208–1212.
51. Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc on-demand distance vector (AODV) routing*. http://www.ietf.org/rfc/rfc3561.txt.
52. Dunkels, A., Gronvall, B., & Voigt, T. (2004) Contiki-a lightweight and flexible operating system for tiny networked sensors.

In *Local computer networks, 2004. 29th Annual IEEE international conference on*, 2004, pp. 455–462.
53. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *Local computer networks, proceedings 2006 31st IEEE conference on*, 2006, pp. 641–648.
54. Polastre, J., Szewczyk, R., & Culler, D. (2005) Telos: enabling ultra-low power wireless research. In *Fourth international symposium on information processing in sensor networks, 2005. IPSN 2005*, pp. 364–369.
55. Dunkels, A. (2011) The contikimac radio duty cycling protocol.
56. Dunkels, A., Osterlind, F., Tsiftes, N., & He, Z. (2007) Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on embedded networked sensors*, pp. 28–32.
57. Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., & Sheu, J.-P. (2002). The broadcast storm problem in a mobile ad hoc network. *Wireless Networks, 8*(2–3), 153–167.
58. Bulusu, N., Estrin, D., Girod, L., & Heidemann, J. (2001) Scalable coordination for wireless sensor networks: Self-configuring localization systems, in *(ISCTA'01)*, Ambleside, UK, 2001, pp. 1–6.

Badis Djamaa is a Ph.D. researcher in wireless networks at Cranfield University, UK. After graduation with honours from the national preparatory school of engineering in 2008, he attended the Polytechnic School of Algiers (EMP) where he received a State Engineering Degree (with distinction) in computer science in 2011. His current research activities include wireless sensor networks, mobile ad hoc networks, internet and web of things and service oriented architectures.

Mark Richardson has over 30 years experience in the defence field in both industry and UK academia and has written over 200 classified and unclassified papers. He is Professor of Electronic Warfare and is currently Director of Academic Resources at Cranfield University, Defence Academy of the UK, Shrivenham.

Nabil Aouf received the Ph.D. degree from the Department of Electrical Engineering, McGill University, Montreal, Canada, in 2002. He is a Reader at the Department of Informatics and Systems Engineering, Cranfield University, UK. His research interests are Aerospace and defence systems, information fusion and vision systems, guidance and navigation, wireless networks and autonomy of systems. He is an Associate Editor of the *International Journal of Computational Intelligence in Control*. He has more than 80 publications of high calibre in his domains of interest.

Dr Bob Walters graduated from the University of Leeds with an honours degree in Electrical and Electronic Engineering and then worked for the British Aircraft Corporation as a microwave engineer, working on the design and development of radomes for military and commercial. A change of career direction into education followed, joining the Royal Navy as an Instructor Officer, taking up various teaching posts at undergraduate and postgraduate level on communications, signal processing and underwater acoustics. On leaving the Navy, Dr Walters became a lecturer at Cranfield University, completing his Ph.D. in passive sonar tracking. Lecturing and research interests are focussed on military communications, signal processing and sonar. He is now a Senior Lecturer in military communications and has held various academic and administrative roles at the university.