mRPL: Boosting mobility in the Internet of Things

Hossein Fotouhi

Daniel Moreira

Mário Alves

CISTER-TR-141103

11-22-2014

mRPL: Boosting mobility in the Internet of Things

Hossein Fotouhi, Daniel Moreira, Mário Alves

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: mohfg@isep.ipp.pt, dadrm@isep.ipp.pt, mjf@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

The 6loWPAN (the light version of IPv6) and RPL (routing protocol for low-power and lossy links) protocols have become de facto standards for the Internet of Things (IoT). In this paper, we show that the two native algorithms that handle changes in network topology –the Trickle and Neighbor Discovery algorithms– behave in a reactive fashion and thus are not prepared for the dynamics inherent to nodes mobility. Many emerging and upcoming IoT application scenarios are expected to impose real-time and reliable mobile data collection, which are not compatible with the long message latency, high packet loss and high overhead exhibited by the native RPL/6loWPAN protocols. To solve this problem, we integrate a proactive hand-off mechanism (dubbed smart-HOP) within RPL, which is very simple, effective and backward compatible with the standard protocol. We show that this add-on halves the packet loss and reduces the hand-off delay dramatically to one tenth of a second, upon nodes' mobility, with a sub-percent overhead. The smart-HOP algorithm has been implemented and integrated in the Contiki 6LoWPAN/RPL stack (source-code available on-line [1]) and validated through extensive simulation and experimentation.

ARTICLE IN PRESS

Ad Hoc Networks xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

mRPL: Boosting mobility in the Internet of Things

Hossein Fotouhi [*], Daniel Moreira, Mário Alves

Polytechnic Institute of Porto, CISTER/INESC-TEC, ISEP, Portugal

ARTICLE INFO

Article history:
Received 25 March 2014
Received in revised form 26 September 2014
Accepted 16 October 2014
Available online xxxx

Keywords:
Wireless sensor networks
Internet of Things
Mobility
RPL
Hand-off
Test-bed

ABSTRACT

The 6loWPAN (the light version of IPv6) and RPL (routing protocol for low-power and lossy links) protocols have become *de facto* standards for the Internet of Things (IoT). In this paper, we show that the two native algorithms that handle changes in network topology – the Trickle and Neighbor Discovery algorithms – behave in a reactive fashion and thus are not prepared for the dynamics inherent to nodes mobility. Many emerging and upcoming IoT application scenarios are expected to impose real-time and reliable mobile data collection, which are not compatible with the long message latency, high packet loss and high overhead exhibited by the native RPL/6loWPAN protocols. To solve this problem, we integrate a proactive hand-off mechanism (dubbed smart-HOP) within RPL, which is very simple, effective and backward compatible with the standard protocol. We show that this add-on halves the packet loss and reduces the hand-off delay dramatically to one tenth of a second, upon nodes' mobility, with a sub-percent overhead. The smart-HOP algorithm has been implemented and integrated in the Contiki 6LoWPAN/RPL stack (source-code available on-line mrpl: smart-hop within rpl, 2014) and validated through extensive simulation and experimentation.

1. Introduction

The next generation Internet, commonly referred as *Internet of Things* (IoT), depicts a world populated by an endless number of smart devices that are able to sense, process, react to the environment, cooperate and intercommunicate via the Internet. For over a decade, low-power wireless network research contested the complexity of the Internet architecture for sensor network applications. However, as the state-of-the-art progressed, academic and commercial efforts invented new network abstractions based on the Internet architecture. The *Internet Engineering Task Force* (IETF) designed some protocols and adaptation layers that allow IPv6 to run over the IEEE 802.15.4 link layer. The *IPv6 over Low-power Wireless Personal Area Networks* (6LoWPAN) working group [2] designed header compression and fragmentation for IPv6 over IEEE 802.15.4 [3]. The IETF *Routing Over Low-power and Lossy networks* (ROLL) working group designed a routing protocol, referred as RPL [4], which is the *de facto* standard routing protocol for 6LoWPAN. These standard IP-based protocols are thus a fundamental building block for the IoT.

6LoWPAN as an adaptation layer is able to support routing in the link layer and the network layer [5,6]. Two routing schemes of mesh-under and route-over are devised to support link and network layer routings respectively. In a mesh-under organization, the adaptation layer performs the mesh routing and forwards packets to the destination via multiple radio hops. The mesh-under design is suitable for single-hop networks, where all nodes are within the transmission range of each other. In a route-over scheme, the routing decision is taken at the network layer, where nodes act as IP routers. Each link layer hop is an IP hop and the IP routing forwards packets between these links.

* Corresponding author.
E-mail addresses: mohfg@isep.ipp.pt (H. Fotouhi), dadrm@isep.ipp.pt (D. Moreira), mjf@isep.ipp.pt (M. Alves).

http://dx.doi.org/10.1016/j.adhoc.2014.10.009
1570-8705/© 2014 Elsevier B.V. All rights reserved.

The route-over routing supports a multi-hop mesh routing communication, suitable for large-scale deployments.

Mobility support is becoming a requirement in various emerging IoT applications [7–9], including health-care monitoring, industrial automation and smart grids [10–12]. Many recent research projects and studies have considered the cooperation between mobile and fixed sensor nodes [13–16]. In clinical monitoring [17], patients have embedded wireless sensing devices that report data in real-time. In oil refineries, the vital signs of workers are collected continuously in order to monitor their health situation in dangerous environments [18]. In fact, many applications require timeliness and reliability guarantees for transmitting critical messages from source to destination, but providing *Quality of Service* (QoS) in low-power and mobile networks is very challenging.

In this work, we are considering a wireless clinical monitoring application that collects patients' vital signs. Patients are mobile nodes that generate traffic and freely move while maintaining their connectivity with the fixed nodes infrastructure. All nodes in our system model are simple sensor nodes featuring low-power CC2420 radio. Fig. 1 illustrates the system model, where a MN moves from the vicinity of Node 8 toward Node 7. We propose a hand-off mechanism that quickly detects mobile entities and locally updates the routing tree. *Hand-off is referred as the process of switching a MN from one point of attachment to another.* In this process, the standard RPL routing performs normally while the mobile nodes run a hand-off algorithm. We build the mobility enabled RPL based on smart-HOP, which is a hard hand-off mechanism that was designed and tested in a generic network architecture, in a protocol-agnostic way [19,20].

Two main mechanisms are employed in RPL and 6LoWPAN that partially cope with mobility. First, the periodic transmission of control packets, scheduled by the *Trickle* algorithm, can detect topological changes. During this process, RPL resumes a fast global routing update that causes a high overhead. Second, the *Neighbor Discovery* (ND – defined in RFC 4861) mechanism, assesses the neighbor reachability in a regular basis. At each activation, the ND protocol floods the entire network with router advertisements, also leading to a high overhead. A short activation interval (that reduces the overhead) leads to low responsiveness to network/topological changes. However, in the revised ND mechanism of 6LoWPAN, router advertisement packets are transmitted upon receiving router solicitation messages [21].

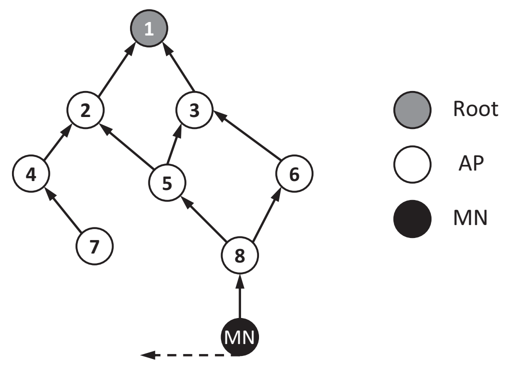


Fig. 1. An example of having mobile node within an RPL tree, where the MN moves from the vicinity of AP_8 toward AP_7.

Why smart-HOP? Hand-off has been widely studied in Cellular and wireless local area networks [22–25]. However, it has not received the same level of attention in low-power networks. Cellular networks perform centralized hand-off decisions typically coordinated by powerful base-stations. Contrarily to Cellular networks, WiFi networks have a distributed architecture where hand-off is triggered when the quality of the service degrades. In low-power networks, a centralized approach is not feasible as the access points are assumed to have scarce resources. smart-HOP [19,20] considers the main features of low-power networks, the link unreliability and the existence of a single low-power radio per node. It manages hand-offs in a distributed way and leads to very short disconnection times.

Why integrating smart-HOP in RPL? There are four main RPL features that motivated us to grant it with mobility support: (i) the proactive feature of RPL that generates and maintains stable routing tables. A periodic broadcast of control messages among all nodes maintains the paths and link states between them. In reactive routing protocols; such as AODV [26] and DSR [27], routes are established upon request, so they do not respond quickly to environmental changes due to mobility or link degradation. RPL maintains the route in the background with minimal overhead. Moreover, for an application with limited mobility and the requirement of an infrastructure, RPL is very suitable, (ii) unlike other proactive routing protocols (e.g. OSPF [28]), RPL exchanges local information among neighbors to repair routing inconsistencies, instead of globally advertising control messages, (iii) RPL runs a tree-based structure that is suitable for data collection WSN applications, and (iv) the IPv6-based addressing in RPL naturally performs the interoperability with other Internet devices.

Contributions. Building on our previous works [19,20], we provide fast and reliable mobility support in RPL. The proposed mobility solution keeps the standard RPL protocol unchanged while providing backward compatibility with the standard implementation, i.e. standard and smart-HOP-enabled nodes can coexist and inter-operate in the same network. The main contributions of this paper are:

1. Efficient hand-off mechanism for RPL with good performance, correctly delivering nearly 100% packets with at most 90 ms hand-off delay and $< 1\%$ additional overhead upon nodes' mobility in high traffic scenarios.
2. Smooth integration and backward compatibility with the standard RPL/6LoWPAN.
3. Collision avoidance mechanism (to avoid collision during the hand-off process while collecting packets from neighbor APs) and loop avoidance mechanism (to avoid closed loops in RPL routing upon mobility).
4. Simulation (Cooja) analysis and experimental validation with commodity hardware platforms in a reliable environment.

5. Implementation over a SOTA operating system (Contiki), for which the open source is freely available [1].

Organization. Section 2 explains the basics of RPL: the control messages, objective function and the process to maintain routes upon link dynamics. A brief background on the smart-HOP hand-off mechanism is presented in Section 3. Then, a general picture of the mRPL design is described in Section 4, which is further detailed in Section 5.[1] The simulation and experimental set-ups, followed by the results and discussion are presented in Sections 6 and 7 respectively. We categorize the related works on mobility support in IP-based low-power networks in Section 8. Some quantitative comparisons between mRPL and the related works are also provided. Finally, we conclude the paper and outline the most relevant findings in Section 9.

2. Relevant aspects of the RPL protocol

RPL is an IPv6 distance vector routing protocol that operates on top of the IEEE 802.15.4 Physical and Data Link Layers and is appropriate for low-power wireless networks with very limited energy and bandwidth resources. The data rate is typically low (less than 250 kbps) and the communication is prone to high error rates, resulting in low data throughput.

RPL organizes nodes in a *Destination Oriented Directed Acyclic Graph* (DODAG), depicted in Fig. 1. Each RPL router identifies a set of stable parents, each of which is a potential next hop on a path toward the "root" of the DODAG. The preferred parent is typically selected to be the one with the lowest rank among the candidate parents. A network may encompass several DODAGs, which are identified by the following parameters:

1. *RPLInstanceID*. This is used to identify an independent set of DODAGs that is optimized for a given scenario.
2. *DODAGID*. This is an identifier of a DODAG root. The *DODAGID* is unique within the scope of an *RPLInstanceID*.
3. *DODAGVersionNumber*. This parameter increments upon some specific events, such as rebuilding of a DODAG.
4. *Rank*. This parameter defines the node position with respect to the root node in a DODAG.

Each node in a DODAG is assigned a *rank* that increases in the downstream direction of the DAG and decreases in the upstream direction. For example, in Fig. 1, Node 8 has higher rank than Node 5, and Node 5 has higher rank than Node 3 and Node 2.

RPL control messages. The RPL control messages are the new type of *Internet Control Message Protocol version 6* (ICMPv6) – defined in RFC 2463. The RPL specification defines four types of control messages: (i) *DODAG Information Object* (DIO). The transmission of this message is issued by the root node and then multicast by other nodes. This message holds the main information for constructing and maintaining a tree, e.g. current rank of a node, RPL Instance and root address, (ii) *DODAG Information Solicitation* (DIS). A node that requires a DIO message from neighbors, requests it by multicasting DIS message, (iii) *Destination Advertisement Object* (DAO). Each node propagates a DAO message upward (along the DODAG). Thus, this message enables the downward traffic from the root through the DODAG to this node, and (iv) *Destination Advertisement Object Acknowledgment (DAO-ACK)*. This unicast message is sent by a DAO recipient to acknowledge its successful reception.

Mobility detection in RPL. Mobility is indicated as one of the main sources of inconsistency in RPL [29]. Generally, there are two main approaches that help in detecting mobility; (i) the ICMPv6 packet transmission, controlled by the Trickle algorithm and (ii) the ICMPv6 packet transmission, controlled by the ND protocol, which are described below.

(i) RPL Trickle Algorithm. The traditional collection protocols in low-power networks typically broadcast control messages at a fixed time interval [30]. A small interval requires more bandwidth and energy. A large interval uses less bandwidth and energy but topological problems may occur due to the incapability to cope with the network dynamics. The basic idea of the Trickle algorithm (defined in RFC 6206) is to propagate beacons if there is a change in routing information.

RPL reduces the cost of propagating routing states by using a Trickle-based timer [31]. Trickle is an adaptive beaconing strategy aiming at fast recovery and low overhead. While the DIS packets are sent periodically from the routers until the first parent node is selected, a Trickle timer is used to schedule the transmission of DIOs. This timer allows the DIO intervals to exponentially increase when the network conditions are stable and quickly decrease to the minimum when noticeable changes in the network conditions are detected. The periodic Trickle timer t is bounded by the interval $[I_{min}, I_{max}]$, where I_{min} is the minimum interval defined in milliseconds by a base-2 value (e.g. $2^{12} = 4096$ ms), and $I_{max} = I_{min} \times 2^{I_{doubling}}$ is used to limit the number of times the I_{min} can double. Assuming $I_{doubling} = 4$, the maximum interval is simply calculated as $I_{max} = 4096 \times 2^4 = 65536$ ms.

The Trickle algorithm is able to maintain the topology update globally in a short period of time. A node that detects an inconsistency in a DIO message (e.g. imposed by node mobility), sets t to I_{min} and updates the tree. If the DODAG remains consistent, t is doubled each time a DIO transmission occurs until it reaches I_{max}, keeping that value constant. When the network is stable, the Trickle timer gradually converges to its maximum interval. Upon mobility, this large interval results in a very low network responsiveness. After detecting any inconsistency in the network, the DIO period of all nodes in the network exponentially decreases, affecting network overhead.

(ii) IPv6 neighbor discovery approach. RPL may use the IPv6 neighbor discovery approach [32] for detecting environmental changes. The low-power links exploit an optimized version of ND, which has been developed by the IETF as an adaptation of neighbor discovery for

[1] In the remainder of the paper, the terms "RPL", "standard RPL" and "default RPL" are used interchangeably. The same applies to the "mRPL", "smart-HOP-enabled RPL" and "mobility-enabled RPL" terms.

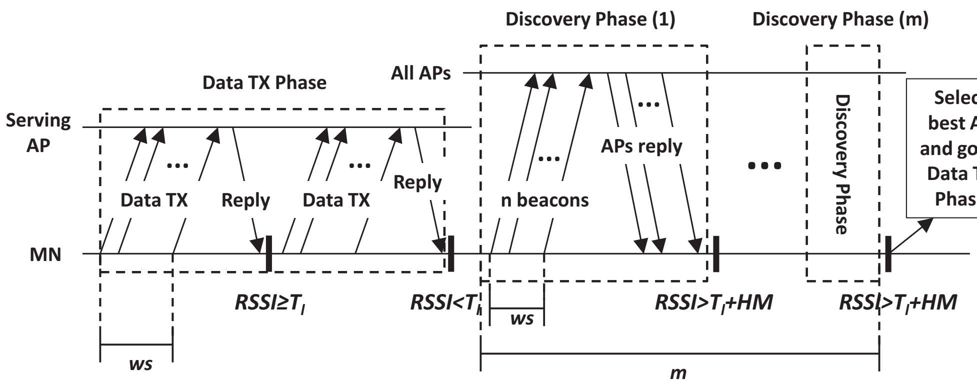


Fig. 2. Timing diagram of the smart-HOP mechanism.

6LoWPAN [21]. The ND protocol allows nodes to detect neighbor unreachability and to discover new neighbors. This protocol is supported by four ICMPv6 control messages: (i) *Neighbor Solicitation* (NS): it determines the link layer address of a neighbor and verifies if a neighbor is still reachable, (ii) *Neighbor Advertisement* (NA): it replies to the NS message and it is also sent periodically to announce link changes, (iii) *Router Solicitation* (RS): it requests from the host node (mobile node in our system model) to its router, asking for information, and (iv) *Router Advertisement* (RA): a router sends periodically and as a response to the RS message that advertises its (the router's) presence with the information of the link and the Internet parameters.

3. Background on smart-HOP

In this section, we provide a brief description on the design of smart-HOP and the main hand-off parameters involved. The smart-HOP algorithm has two main phases: (i) *Data Transmission Phase* and (ii) *Discovery Phase*. A timeline of the algorithm is depicted in Fig. 2. For the sake of clarity, let us assume that a node is in the *Data Transmission Phase*. In this phase, the mobile node (MN) is assumed to have a reliable link with an access point (AP), defined as *Serving AP* in Fig. 2. The mobile node monitors the link quality by receiving *reply* packets from the serving AP. Upon receiving *n* data packets in a given window, the serving AP replies with the average RSSI (ARSSI) or SNR of the *n* packets. If no packets are received, the AP takes no action. This may lead to disconnections, which are solved through the use of a time-out mechanism. It is important to notice that smart-HOP filters out asymmetric links implicitly by using reply packets at the Data Transmission and Discovery Phases. If a neighboring AP has no active links, that AP is simply not part of the process.

Hand-off parameters. The smart-HOP mechanism encompasses three main parameters[2] for fine-tuning:

Parameter 1: *window size (ws). ws* is the number of packets required to calculate the ARSSI over a specific time interval, as illustrated in Fig. 2. A small *ws* provides detailed information about the link, but increases the processing of reply packets, which leads to higher energy consumption and lower delivery rates. The packet delivery reduces as the MN opts for performing some unnecessary hand-offs. The hand-off is triggered by detecting low quality links, resulting from the signal strength reduction. On the other hand, a large *ws* provides coarse grained information about the link and decreases the responsiveness of the system, which is not suitable for mobile networks with dynamic link changes.

Parameter 2: *hysteresis margin (HM)*. This parameter is defined as the difference between the ARSSI threshold for starting the hand-off (T_l) and the ARSSI threshold for stopping the hand-off ($T_h \overset{\text{def}}{=} T_l + HM$). In WSNs, the selection of thresholds and *hysteresis margins* is dictated by the characteristics of the transitional region and the variability of the wireless link. The thresholds should be selected according to the boundaries of the transitional region. The transitional region is often quite significant in size and hence a large number of links in the network (higher than 50%) are unreliable [33,34]. Therefore, wireless nodes are likely to spend most of the time in the transitional region.

If the T_l threshold is too high, the node could perform unnecessary hand-offs (by being too selective). If the threshold is too low, the node may use unreliable links. The *hysteresis margin* plays a central role in coping with the variability of low-power wireless links. If the *hysteresis margin* is too narrow, the mobile node may end up performing unnecessary and frequent hand-offs between two APs (ping-pong effect). If the *hysteresis margin* is too large, hand-offs may take too long, which ends up increasing the network inaccessibility times, and thus decreasing the delivery rate.

Parameter 3: *stability monitoring (m)*. Due to the high variability of wireless links, the mobile node may detect an AP that is momentarily above T_h, but the ARSSI may decrease shortly after handing-off to that AP. In order to avoid this, it is important to assess the stability of the candidate AP. After detecting an AP with the ARSSI above T_h, the MN continues *m* further Discovery Phases to check the stability of that AP. As can be easily inferred, the *stability monitoring* and the *hysteresis margin* parameters are tightly coupled. A wide *hysteresis margin* requires a lower *m*, and vice versa. Ref. [20] shows that an appropriate

[2] We ignored the stability monitoring parameter at this stage, since it has no impact on the smart-HOP performance [19]. The stability monitoring is the number of times in sequence that the MN detects a high quality link from an AP in the *Discovery Phase*.

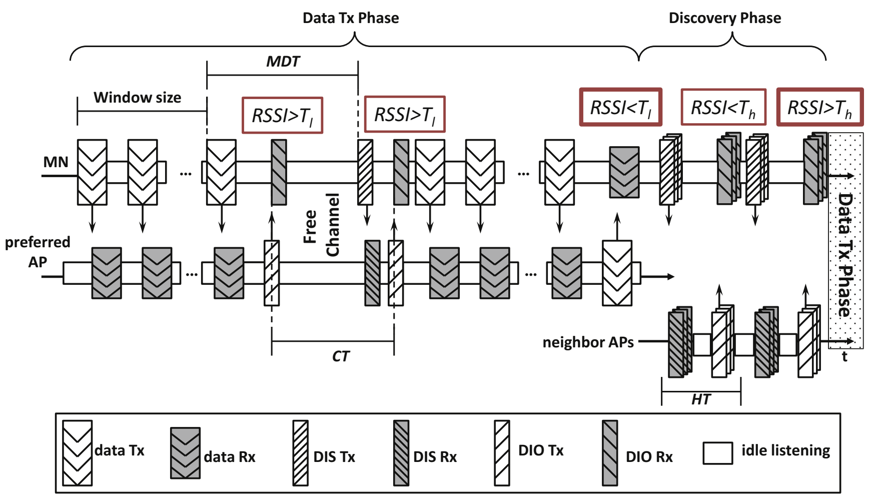


Fig. 3. mRPL timing diagram for the *Data Tx Phase* and *Discovery Phase*.

tuning of the *hysteresis margin* will lead to $m = 1$, which leads to a minimal overhead.

4. mRPL overview

As previously mentioned, we are integrating smart-HOP within RPL in a way that is very simple, effective and backward compatible with the standard protocol. In this model, the standard RPL protocol is unchanged while providing mobility support, i.e. standard and smart-HOP enabled nodes can coexist and inter-operate in the same network.

The general procedure of beacon and data exchanges in smart-HOP integrated in RPL (mRPL) is similar to the original smart-HOP design, except employing RPL control messages (DIS and DIO) as beacons and adding some timers to improve reliability and efficiency. The timeline of the algorithm is depicted in Fig. 3. In this approach, the MN gets a reply packet (unicast DIO message) immediately after transmitting a predefined number of data packets (window size). The DIO reply message (that holds the average RSSI level), implicitly filters out the asymmetric links.

Upon detecting a good quality link (from the average RSSI level in the DIO reply message), the MN continues the *Data Transmission Phase*. By observing the ARSSI degradation, MN starts the *Discovery Phase*. MN resumes the data communication with the serving AP until finding a better AP. After a successful hand-off, the nullifying process of the RPL algorithm is executed.[3]

To assess the potential parents, the MN broadcasts a burst of DIS control messages. Then, all neighbor APs reply to the MN in a non-conflicting basis (this will be discussed in detail in Section 5). The average RSSI level is embedded in the unicast DIO reply. Upon reception of each DIO reply, MN compares the ARSSI value with T_h level. If it is not satisfactory (ARSSI below T_h), MN continues broadcasting DIS bursts periodically (with respect to the *Hand-off Timer*). Upon detecting a high link quality (ARSSI above T_h), the *Discovery Phase* stops and the MN resumes regular data communication (with a new preferred parent) – *Data Transmission Phase*.

[3] Parent nullifying is a process in which the preferred parent is removed and the *rank* is set to infinity.

5. mRPL in detail

This section details the mRPL design. We first describe the additional timers that improve the efficiency and reliability of the hand-off process. The enhanced control message packets and the priority assignment of reply packets (DIO messages from neighbor APs) are then described. The Trickle timer setting (during the hand-off) and the parent selection are also discussed.

Timers. We have implemented four main timers to easily perceive the link degradation and the parent unreachability in a short period of time. The use of these timers within the *Data Transmission* and *Discovery Phase* is instantiated in Algorithms 1 and 2 and briefly described as follows.

(i) *Connectivity timer* (T_C). Mobile nodes constantly monitor the channel activity to detect any packet reception from their serving AP. Every MN runs a timer to increase the RPL routing responsiveness – the connectivity timer. The periodicity of the connectivity timer is set according to the maximum Trickle interval (I_{max}). During this period, the MN keeps listening to the channel and monitors the incoming packets from the serving parent. Upon elapsing T_C, if the MN observes a silent parent, then it starts the *Discovery Phase*. Upon detecting any packet reception from the serving AP (e.g. Trickle DIO, unicast DIO or a data packet), connectivity timer is reset.

(ii) *Mobility detection timer* (T_{MD}). Periodic DIS beaconing of the MN requests a unicast DIO message from the serving AP. The MN reads the ARSSI level related to the DIO message to assess the reliability of the link. Moving a node or appearing an obstacle between two nodes may result in losing the request or reply packets. In this situation, the MN starts the *Discovery Phase* to find a new serving parent. The periodicity of the mobility detection timer is set according to the data generation rate at the MN.

(iii) *Hand-off timer* (T_{HO}). It is paramount to reduce the hand-off delay. This timer manages the periodicity of broadcasting bursts of DIS to the neighboring parents. This period should enable to accommodate transmitting bursts of DIS with the highest possible rate and receiving intermittent replies from neighbor nodes. The DIO replies are collected immediately after sending each burst. The sequence of sending replies by each parent is scheduled in such a way to reduce the probability of collision.

(iv) *Reply timer* (T_R). The serving parent is supposed to reply to the MN by unicasting a DIO control message at certain instants. Selecting a wrong moment to reply may cause a collision with the data packets, which in turn triggers the *Discovery Phase*. The parent node extracts relevant information from the packets that are received from the MN (e.g. data packet counter in each window size). The reply time is calculated by $(ws - C) \times T_{DIS}$, where C represents the counter of DIS packets within each window size (ws) and T_{DIS} indicates the DIS interval. This reply time is adaptively changing upon receiving new packets.

Algorithm 1. *Data Transmission Phase*

```
begin
    if received DIO packet then
        reset $T_C$;
        if $ARSSI < T_l$ then
            go to the Discovery Phase;
        else
            continue the Data TX Phase;
        end
    else if $T_{MD}$ expires then
        reset $T_{MD}$;
        unicast burst of DIS;
        go to the begin;
    else if $T_C$ expires then
        go to the Discovery Phase;
    end
end
```

Algorithm 2. *Discovery Phase.*

```
begin
    if received unicasted DIS message then
        store RSSI readings;
        store counter value $C$ of the latest DIS packet;
        reset $T_R$ with $(ws - C) \times T_{DIS}$;
        if $T_R$ expires then
            calculate average RSSI;
            send unicast DIO message with average RSSI;
        else
            continue Discovery Phase;
        end
    else
        continue the Data TX Phase;
    end
end
```

Enhanced control messages. To integrate the smart-HOP algorithm within RPL, we enhanced the RPL control messages rather than creating new ones. This approach guarantees backward compatibility with the standard RPL, i.e. standard RPL nodes can coexist and inter-operate with smart-HOP-enabled nodes in the same network.

RPL control messages are transmitted on a regular basis; however, during the hand-off process, they follow specific rules. In the *Data Transmission Phase*, the DIS is sent from the MN to the AP (unicast) and the preferred parent replies with a unicast DIO. The type of DIS and DIO is detected by reading a flag that reflects the status of each node (will be explained next). In the *Discovery Phase*, the MN multicasts DIS messages to all neighboring APs and receives unicast DIO replies.

Smart-HOP enables transmitting unicast DIS control messages to probe the serving AP in order to ensure the parent is reachable and reliable (RPL transmits multicast DIS and DIO packets). To distinguish between the mRPL DIS and the native RPL DIS, a one bit flag (*F-DIS*) is implemented – see Fig. 4(a). Initializing this field to "0" represents the multicast transmission of the RPL DIS. Instead, setting this field to "1" reflects the unicast mRPL DIS transmission. The additional two bits of "*C*" describe the counter of DIS messages within a window size. In mRPL with $ws = 3$, the counter increments to a maximum of 3.

The mRPL DIO message adds two fields: (1) *F-DIO* that stands for the flags and (2) *ARSSI* that holds the average RSSI reading at the potential parent node – see Fig. 4(b). The two bits of *F-DIO* distinguish three cases: (i) *F-DIO* = 0 corresponds to the RPL DIO, (ii) *F-DIO* = 1 indicates the mRPL DIO within the *Data Transmission Phase*, and (iii) *F-DIO* = 2 reflects the mRPL DIO within the *Discovery Phase*.

Priority assignment. In order to reduce the packet collision during the *Discovery Phase*, we prohibit some of the APs to reply to the MN; parents with $ARSSI < T_h$ are excluded from the possible parents' set and do not reply. To do this, each parent assigns a priority according to the average RSSI readings, as shown in Table 1. The priority assignment schedules the DIO transmissions in different slots. Since low-power networks are likely to operate in the transitional region, it is more likely that different parents choose the same slot. A timer schedules the DIO transmission (t_{offset}) after detecting a busy channel as follows.

$$t_{offset} = (ws - C) \times T_{DIS} + t_2 \times prio + rand(t_1, t_2) \qquad (1)$$

The first part of this equation, $(ws - C) \times T_{DIS}$, is the waiting time for receiving the complete DIS messages transmitted, which is similar to the *Data Transmission Phase* of the smart-HOP algorithm. We force the higher quality APs ($prio = 0$) to transmit earlier ($t_2 \times prio = 0$ ms) and the lower quality APs ($prio = 1$) transmit later ($t_2 \times prio = t_2$ ms). A random delay is also added to reduce the possibility of colliding the same priority level APs by $rand(t_1, t_2)$. It is important to note that with $t_2 \times prio$, lower quality links wait at most for t_2 ms, ($max((prio = 0) \times t_2 + rand(t_1, t_2)) = t_2$), which is measured by $(prio = 1) \times t_2 = t_2$ ms. The random value also reduces the possibility of collision between the replies from the lower quality and the higher quality APs. Random values are set to 10 and 15 ms, which

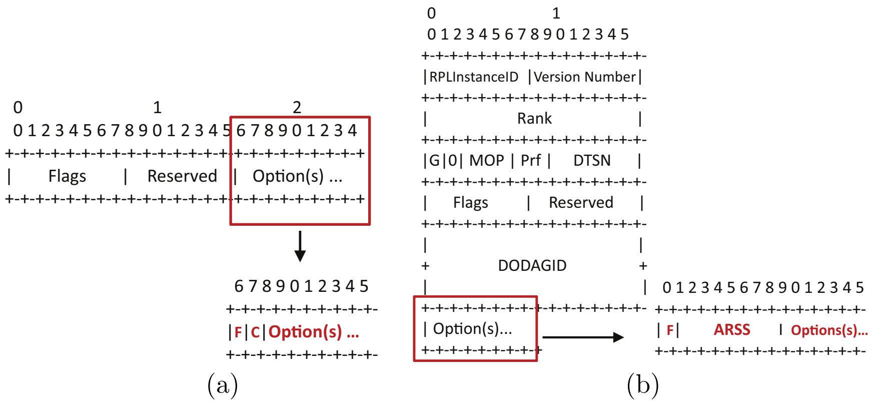


Fig. 4. (a) The modified DIS packet format. Two fields of *F-DIS* and *C* are added to the RPL DIS packet, and (b) the modified DIO packet format. Two fields of *F-DIO* and ARSSI are added to the RPL DIO packet. Additional bits are applied to the "Option(s)" part of the packet.

Table 1
The priority assignment.

Priority	Range of average RSSI readings
$prio = 1$	$-85 < ARSSI < -80$ dBm
$prio = 0$	$ARSSI \geqslant -80$ dBm

are above the maximum possible transmission rate.[4] Considering $ws = 3$ and $T_{DIS} = 15$ ms, in the worst case (i.e. $rand(t_1, t_2) = 15$ ms), it takes at most 75 ms for the MN to get all replies from the neighboring APs. In our system model, we are considering a wise deployment of APs in order to avoid very high or very low density of APs. Our tests provide minimal overlap between contiguous APs that would prevent the possibility of having multiple high quality APs in a region.

Trickle setting during mRPL. According to the Trickle algorithm, all nodes (roots/routers) broadcast messages (DIOs) to exchange information with the neighbor nodes. The transmission interval is bounded and enlarged upon network stability. When a node moves, it interferes with the network stability and hence the interval is set to its minimum value (I_{min}). We keep the Trickle interval unchanged during the hand-off process, while keeping the transmissions' schedules independent. As already mentioned, the *F* fields of the control messages (*F-DIS* and *F-DIO*) are added to distinguish between mRPL and RPL messages.

Loop avoidance mechanism. In RPL, when a node disconnects from its parent, the *rank* value sets to infinity. This enables the MN to connect to any neighboring node, even the ones with a lower *rank*. For instance, the MN may select a neighbor node that was previously the MN's child (before the hand-off) as the new parent. Since the neighbor has a lower *rank* compared with the infinity, according to the default RPL, the MN is allowed to choose it. As shown in Fig. 5, in DODAG 1, Node 5 has a parent (Node 2) and three children (Nodes 7, 8 and 9). Each node delivers data to a lower rank node (written besides each node). When Node 5 moves out from the range of Node 2, according to the RPL routing, DODAG 2 is established. In this case, first the MN's rank is set to infinity and then it picks a neighbor with the highest ARSSI level and the lower rank level (6). Thus, Node 7 (Node 5's previous child) is selected as the preferred parent. The data messages from Node 5 are forwarded to Node 7, and Node 7 forwards to Node 5 (its parent), which represents a closed loop.

RPL has some loop detection mechanisms; however, loops cannot be fully avoided and thus may still occur. To fix this, RPL performs global repairs where the routing tree is reconstructed, updating the rank of all nodes in a DODAG. This behavior is not efficient as a MN will need to start the whole process of finding a new parent again, which is highly time and energy consuming.

In this context, we devised a simple yet efficient loop avoidance mechanism. We analyzed two different approaches to avoid the loop effect. First, the MN gets replies from all neighboring APs and then ignores the messages from the previous children. Thus, after creating the set of alternative parents, the children are excluded from the set. Second, the children decline to reply the previous parent's request for joining. We select the latter approach as it leads to less communication, processing and overhead during a hand-off. DODAG 3 in Fig. 5 shows a scenario where Node 5 disconnects from Node 2 and connects to Node 6, avoiding to choose one of its previous children.

Memory overhead. The memory overhead of the standard RPL against mRPL is illustrated in Table 2. smart-HOP has been integrated with about 4 kB ROM and 1 kB RAM extra, representing just 10% of additional footprint.

6. Simulation analysis

We implemented and tested the protocol with a simulator that easily ports to the sensor hardware and provides

[4] We use Tmote Sky motes that are equipped with the Chipcon 2420 radio chip [35], operating at 2.4 GHz with 250 kbit/s data rate. The packet size depends on the data payload, which is added to the header and footer. Since RPL runs an IPv6 addressing strategy, we assume that the packet size is 127 bytes in the worst case. Considering the radio data rate and the packet size, the node is able to transmit at most 246 packets/s (1 packet every 4 ms). The propagation delay, modulation, demodulation, fragmentation and de-fragmentation extend this approximate transmission delay. In real world experiments, it is wise to pick intervals larger than 4 ms to ensure successful transmissions.

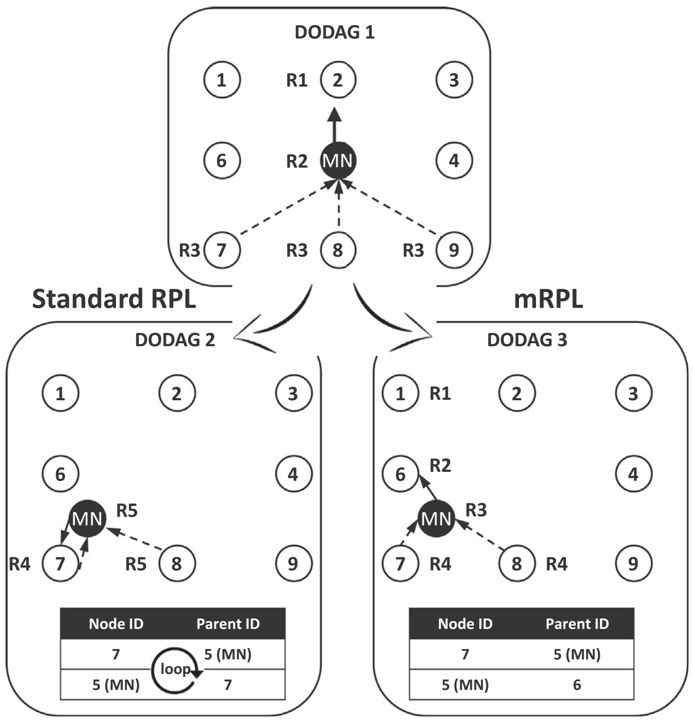


Fig. 5. The DODAG 1 updates upon mobility. The DODAG 2 updates by applying the standard RPL algorithm, increasing in a closed loop. The DODAG 3 updates according to mRPL, avoiding the closed loop.

Table 2
Memory usage in standard RPL vs. mRPL.

Implementation	ROM (bytes)	RAM (bytes)
RPL (MN)	40,202	7660
RPL (AP)	40,336	7606
mRPL (MN)	44,348	8562
mRPL (AP)	44,022	8512

Table 3
Description of the RPL scenarios.

Scenarios	I_{min}	$I_{doubling}$	DIO_{min} (s)	DIO_{max} (s)
(12–8)	12	8	4.096	1048.576
(12–1)	12	1	4.096	8.192
(10–2)	10	2	1.024	4.096
(8–1)	8	1	0.256	0.512

the opportunity of analyzing different network conditions. Since low-power wireless links are very prone to external radio interference from other wireless technologies operating in the ISM band, simulators are usually unable to provide a very accurate radio interference model. Each indoor/outdoor environment exhibits specific link behaviors that are impossible to mimic the simulated environment. mRPL has been designed to perform well in networks with full AP coverage and minimum overlap between neighboring APs. In simulation, we are able to establish an environment that provides these requirements, but in real experiments, links may overlap differently (more or less). We will compare simulation and experimental results in Section 7 to show the necessity of performing experimental tests in order to enrich the radio propagation and interference models in simulation.

6.1. Simulation setup

In order to implement and evaluate mRPL, we opted for the Contiki 2.6.1 [36] operating system (OS), which supports the Cooja simulator. The main reasons for selecting Contiki are: (i) the availability of a RPL/6LoWPAN implementation that is reasonably mature and widely used, (ii) the ease of porting Cooja code to the hardware platforms, and (iii) the availability of a mobility plugin in Cooja [37], that enables to evaluate mRPL in a repeatable environment.[5]

In this section, we compare mRPL with different settings of the standard RPL, considering different topologies. Then, we study the impact of other parameters on the mRPL performance. The major parameters that impact the RPL performance are I_{min} and $I_{doubling}$ in the Trickle algorithm. We considered four RPL scenarios by varying the tuple $\langle I_{min}, I_{doubling}\rangle$ values, as defined in Table 3. The evaluation focuses on the impact that these Trickle parameters have on the following network metrics:

[5] By default, Cooja does not support mobility. Nevertheless, based on the fact that each deployed mote has its own location represented in a two-axis (x, y) system, a Cooja mobility plugin [37] was developed that is capable of loading specific mobility trace-files using the Interval Format.

Hand-off delay. It represents the average time required to perform the hand-off process with mRPL or the time spent to discover a new preferred parent in the standard RPL.

Total packet overhead. We identify all the non-data packets (control messages) as network overhead. RPL uses ICMPv6 based control messages (DIS, DIO and DAO) for building and maintaining DODAGs. The mRPL utilizes these control messages to detect the mobility and perform the hand-off process.

Packet delivery ratio (PDR). It is defined as the number of successfully received packets at all APs over the total number of packets sent from MNs. The successful delivery rate of mRPL is compared with different RPL scenarios in the presence of mobility.

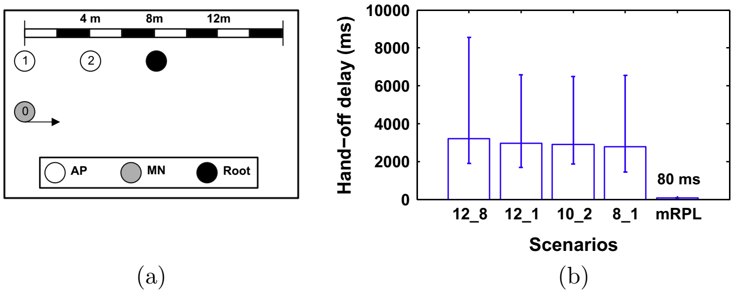
Total DAO packets. To establish downward routes, RPL nodes send unicast DAO message upward. The next hop destination of a DAO message is the preferred parent. After switching to the best parent, the child node informs the previous parent about its disconnection and the selected parent about its reachability. The total number of DAO packets is an indication for assessing the routing responsiveness and the number of hand-offs in a mobility-enabled network.

6.2. RPL vs. mRPL

To evaluate the proposed algorithm, we consider three network topologies: (1) with two APs, (2) with four APs deployed in a row, and (3) with eight APs deployed in two parallel rows. In the first deployment with two APs (Node 1 and Node 2, 10 m apart – see Fig. 6(a)), the MN travels 15 times between AP_1 and AP_2 with a constant speed ($v = 2$ m/s) and transmission power of -25 dBm, while generating data with the rate of 30 pkt/s. Similarly, in the two other deployments (Figs. 7(a) and 8(a)), the MN moves from one left corner to the right corner with the same constant speed and then returns back to the starting point.

Connectivity is guaranteed by providing a fast and reliable hand-off process. mRPL is able to detect and perform a hand-off within tens of milliseconds (80–83 ms), which is much faster than all RPL scenarios – see Figs. 6(b), 7(b) and 8(b). We have estimated the hand-off delay in the standard RPL as it does not have a hand-off mechanism. *The "hand-off" in RPL is assumed to start at the moment when packets start to get lost at the serving parent and to end when the new parent starts to successfully receiving data packets from the MN.* The high data generation rate accelerates the updating of the ETX metric that leads to a fast parent switching process during link degradation.

The hand-off delay of RPL scenarios fluctuates a lot, as the mobility detection mechanism depends on various conditions (e.g. data rate, Trickle timer and ND protocol) and the responsiveness to environmental dynamics is not guaranteed in RPL. The average hand-off delay of RPL scenarios varies from 2776 ms to 9776 ms in these three network topologies (Figs. 6(b), 7(b) and 8(b)). In RPL, the mobile node switches between parent nodes in its parent set. In order to update the parent set information, it uses the Trickle and ND algorithms. The Trickle algorithm (that schedules the control message exchanges) will enlarge intervals in a stable network. To detect mobility in this condition, the RPL node either waits for receiving the NA message or requests this message by multicasting the NS message to its neighboring APs. These messages are supported by the ND protocol to detect parent unreachability. The major drawbacks of RPL concerning network connectivity are: (i) the sudden changes due to nodes mobility



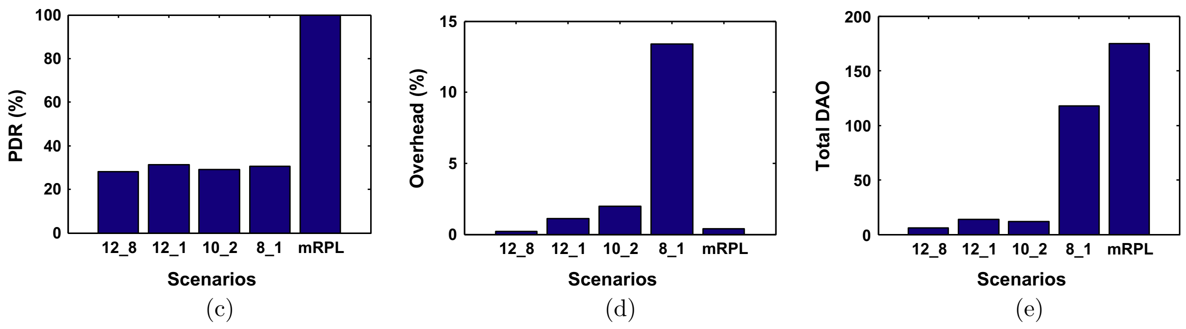
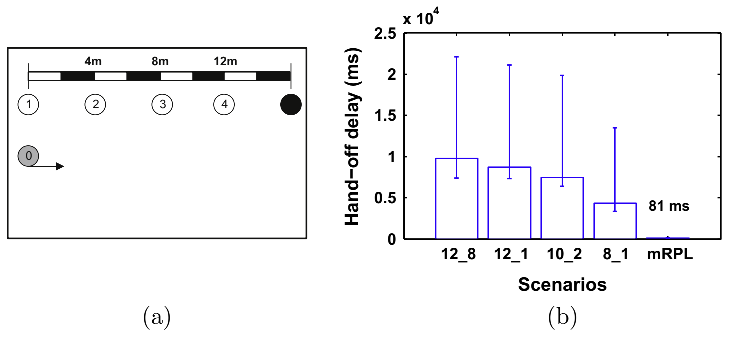


Fig. 6. Simulation results for a network topology with two APs. (a) Simulation scenario, (b) hand-off delay, (c) packet delivery ratio, (d) total overhead in terms of control messages, and (e) total number of DAOs.



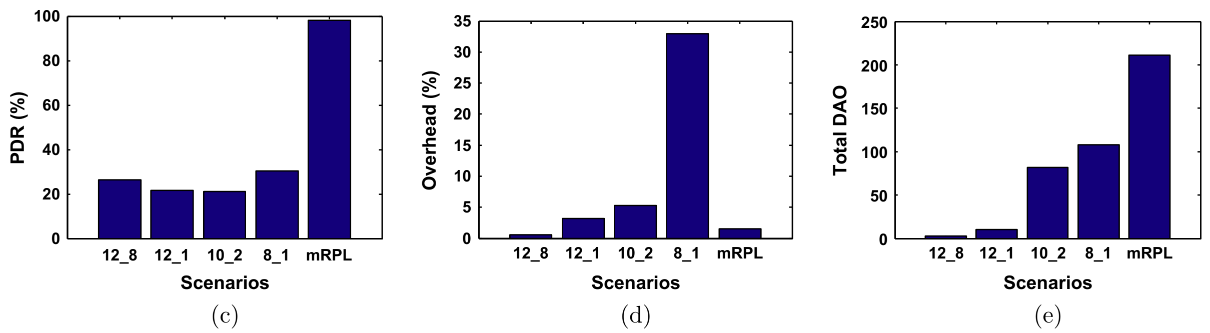


Fig. 7. Simulation results for a network topology with four APs. (a) Simulation scenario, (b) hand-off delay, (c) packet delivery ratio, (d) total overhead in terms of control messages, and (e) total number of DAOs.

are not quickly detected if the network has been stable for a while, (ii) the ND protocol is initiated at the parent side (like a passive hand-off), which enlarges the hand-off duration, and (iii) resuming the ND protocol (that reconstructs the routing trees) is very expensive.

mRPL is able to provide nearly 100% packet delivery ratio. A fast hand-off process enables transmitting most of the packets to the targeting access point. In RPL, the MN should wait for control messages from the nearest AP. A longer delay causes more packet losses as the MN is not connected to any AP. In mRPL, the MN is able to send data to the previous parent during the *Discovery Phase* until finding a new preferred parent. This mechanism increases the chance of delivering most of the data packets, as shown in Figs. 6(c), 7(c) and 8(c).

The control message overhead of mRPL is comparable with the RPL settings with minimum overhead. In RPL, after creating DODAGs during an initialization phase, if the network remains stable, the periodicity of control message exchanges will converge to its maximum value. For instance, according to Table 3, in the $\langle 12,8\rangle$ RPL scenario, the periodicity is 1048.576 s and with $\langle 8,1\rangle$ is 0.512 s. A higher message transmission rate increases the network overhead. In mRPL, the Trickle parameters are set according to the RPL scenario with lowest overhead ($\langle 12,8\rangle$). The additional control messages triggered by the hand-off are invoked on-demand. Hence, in a high data rate network, similar to our example (with 30 pkts/s), mRPL has a higher amount of overhead compared with RPL. Comparing different network topologies shows that adding more neighbor nodes (APs) increases the overhead of the network – see Figs. 6(d), 7(d) and 8(d). Adding more APs in the neighborhood of a MN would increase the number of reply packets in the *Discovery Phase*, eventually increasing the overhead.

mRPL is very responsive to network dynamics. The total number of DAOs is an indicator for showing the efforts for creating new connections. Since RPL does not have an explicit hand-off mechanism, a successful parent selection is identified by DAO transmissions. In Topology 1 (with two APs), mRPL has the greatest number of new connections, which shows an accurate hand-off during each trip. Adding more APs in Topology 2 results in creating more connections in both RPL and mRPL. In a denser deployment (Topology 3), there are more overlaps between links and hence the total number of DAOs reduces in RPL and mRPL. However, mRPL is still able to smoothly switch between APs with only 1.4% less hand-offs, while RPL reduces new connections up to 63% – see Figs. 6(e), 7(e) and 8(e).

6.3. Further evaluations on speed, duty cycling and network density

At this stage, we are aiming at studying the impact of mobile node speed and network duty cycling on the performance in high and low data traffic in a more complicated network deployment. Human tends to walk at speeds from nearly 0 m/s to upwards of 2 m/s. In our simulations, we applied wider range of speeds from 0.5 to 4 m/s. We also assumed various data transmission periods of 50 ms, 100 ms, 500 ms, 1 s, 2 s, and 5 s. We employed a MN and 12 APs located in four rank levels as depicted in Fig. 9(a). MN starts its trip from the vicinity

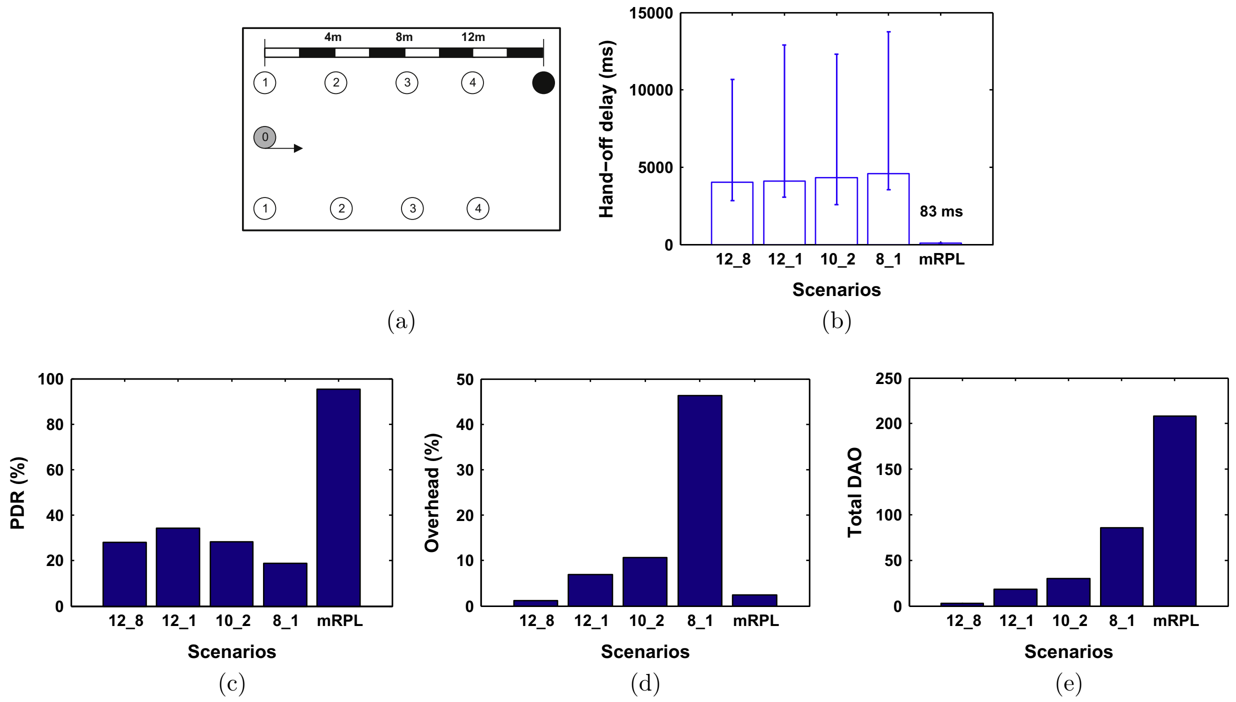


Fig. 8. Simulation results for a network topology with eight APs. (a) Simulation scenario, (b) hand-off delay, (c) packet delivery ratio, (d) total overhead in terms of control messages, and (e) total number of DAOs.

of AP_1 and travels all the network through the dotted lines, then pausing for 30 s at the initial position, while the simulation runs for two minutes.

mRPL is efficient for the range of normal human walk speeds. We define efficiency when having fast, light (low overhead) and reliable hand-off. The results in Fig. 9(b)–(d) show that the hand-off delay and network overhead are very low, in all scenarios. There are fluctuations in packet delivery ratio. In high traffic scenarios, by increasing mobile node speed, the packet delivery reduces. For instance, with 50 ms and 100 ms data periods, packet delivery ratio drops by $\approx 4\%$ and $\approx 6\%$ respectively, when increasing MN speed from 0.5 m/s up to 4 m/s. The main reason is the change in the starting and ending moments of the hand-off process. Some data packets in higher speed scenarios may drop due to these timing behaviors. By lowering the data transmission period, the Trickle timer and the mRPL timers reduce their periodicity to reduce network overhead. This effect causes some packet drops due to the delayed hand-off process. With low data periods, the packet delivery ratio with different mobile node speed has more fluctuations. Successful packet delivery depends on performing a hand-off before reaching an AP. The hand-off starting moment depends on the timeouts of the timers (T_C and T_{MD}). Tuning these timers to longer periods in low activity networks causes sudden packet drops. Lack of sufficient control messages in low traffic scenarios postpones the hand-off process. By lowering the data period from 100 ms to 5 s, the average packet delivery ratio drops by 24%.

mRPL has less overhead in low traffic networks. In mRPL, mobility detection is according to the link degradation (ARSSI), connectivity timer and mobility detection timer. We adapt the connectivity timer according to the larger Trickle interval and the mobility detection timer according to the data periods to reduce the network overhead. Fixed and low intervals of these timers in low traffic scenarios reduce network overhead. For instance, the overhead in a low data traffic scenario (e.g. transmitting data every 5 s) is 30% less than the high data traffic scenario (e.g. transmitting data every 50 ms).

Low traffic scenarios require data retransmissions to keep network reliability. By enlarging the mobility detection timer in low traffic scenarios, some of the data packets may drop. Upon data packet losses, hand-off process resumes that leads to parent switch. After a hand-off process, MN has good connectivity with the preferred parent. Therefore, we propose a data retransmission to the new AP immediately after the hand-off process to keep network reliability.

Hand-off delay is low regardless of network traffic and mobile node speed. Hand-off in mRPL is a process that requires a number of packet exchanges to assess neighbor APs. This process is very fast and takes about 85 ms with some fluctuations in various scenarios.

We also evaluated mRPL without existence of mobile node. Fig. 9(b) and (d) show that **a static node is able to successfully transmit almost all data packets to the fixed infrastructure.** The overhead of this experiment is the minimum, since additional control messages are not generated in a static environment.

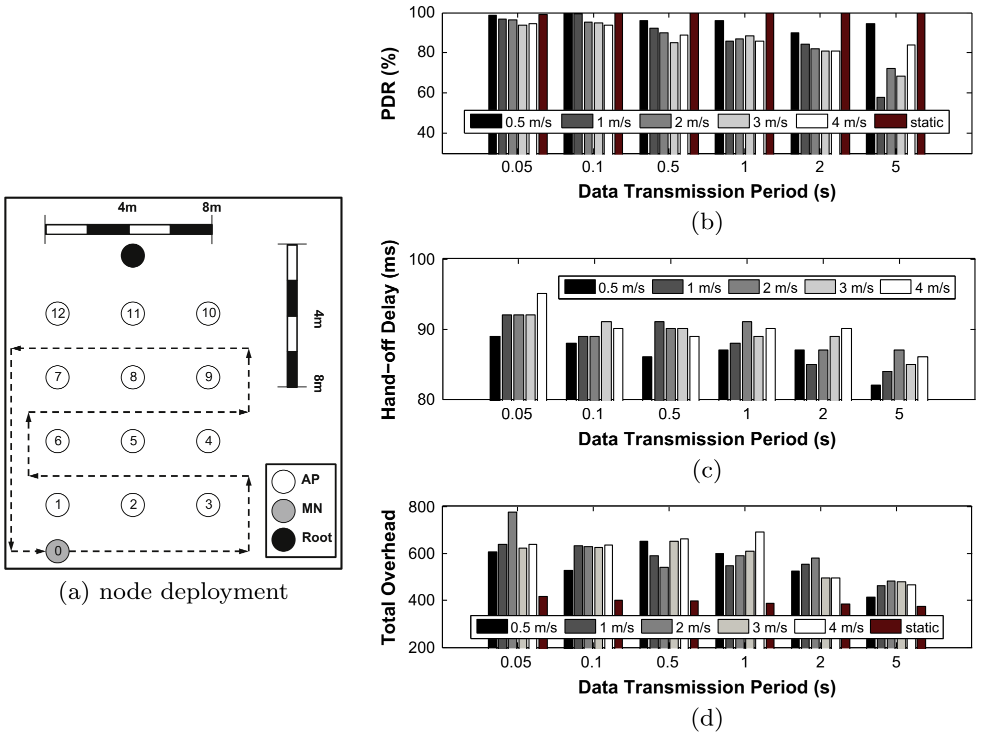


Fig. 9. Impact of MN speed on mRPL performance with different network traffics: (a) node deployment, and performance in terms of (b) packet delivery ratio, (c) average hand-off delay, and (d) total packet overhead.

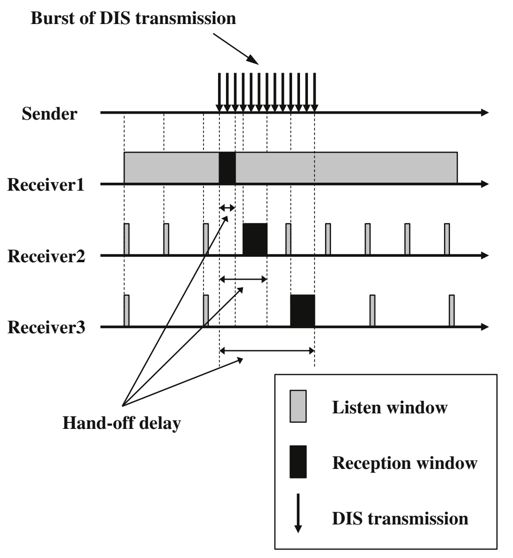
The duty cycling MAC design (ContikiMAC) reduces the energy consumption by periodic idle-listen periods. In ContikiMAC, if a packet transmission is detected during a wake-up period, the radio is kept on to receive the packet. After successfully receiving a packet, receiver sends a link layer acknowledgment. According to this behavior, in mRPL, MN keeps sending burst of DIS messages until receiving and replying by the neighbor AP as depicted in Fig. 10 (a). The radio of Receiver 1 is always on (NullMAC) and can immediately detect the packet transmissions from the MN, and thus, the hand-off process is the shortest possible. By applying a duty cycling approach, the request packets are detected later and the hand-off process takes longer (MN keeps sending burst of DIS messages until receiving a reply from a neighbor AP). Increasing the sleeping period worsens the performance in terms of responsiveness (compare Receiver 2 to Receiver 3).[6]

Increasing the listening period degrades the hand-off performance. We have analyzed the duty cycling approach by changing channel check rates (64, 32 and 8 Hz) and studied the network performance –see Fig. 10(b)–(d). Reducing the check rates increases the listening periods that enlarges the hand-off delay. By increasing the check rate from 64 Hz or 15.625 ms to 8 Hz or 125 ms (87.5% increase in listening period) with 1 (s) data transmission period, the hand-off delay increases from 115 ms to 156 ms (i.e. 26% increase), which is reasonable. Consequently, long hand-offs reduces the packet delivery ratio and increases the control message overhead. The trend of network performance degradation by increasing the listening period is similar in all scenarios with various data transmission periods.

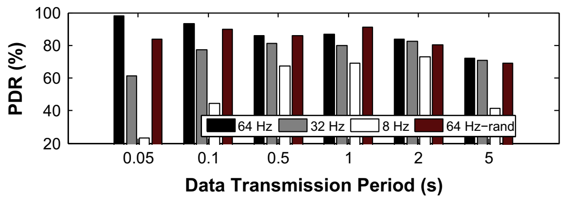
Hand-off delay is low in a random mobility pattern scenario. We created a scenario where the mobile node speed intermittently changes and the trajectory varies, while the mobile node moves within the deployment area. Comparing the random scenario with the constant speed scenario (2 m/s) with the 64 Hz check rate in Fig. 9 shows that the hand-off delay in all data traffics is below 100 ms. However, there are fluctuations in the PDR and the overhead of the random scenario compared with the constant speed scenario. The results indicate a PDR difference between 1.5% and 54% and the overhead difference from 0.2% to 14.2%. Generally, we observe that in a random speed scenario the performance with higher data traffics degrades more than the lower data traffic, when compared with the constant speed scenario.

Network density has a direct impact on the network overhead. Increasing the number of APs in a single broadcast domain increases the number of DIO replies in the *Discovery Phase* of a hand-off process, as depicted in Fig. 11(b). This also increases the network overhead of mRPL. A wise deployment of APs in a real experiment reduces the network overhead drastically.

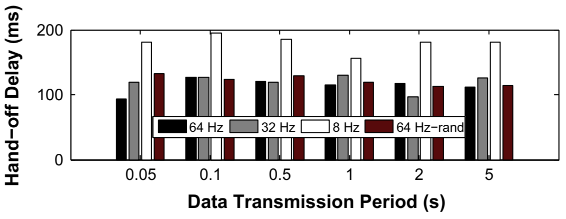
[6] In ContikiMAC it is required to obey a precise timing between transmissions. It uses *Clear Channel Assessment* (CCA) that reads the RSSI measurement to detect channel activity. The timing analysis in [38] shows that a minimum packet size of 23 bytes is required for the CCA mechanism to work properly. We respect this limitation in our simulations and experiments as the size of IPv6-based packet are normally much longer.



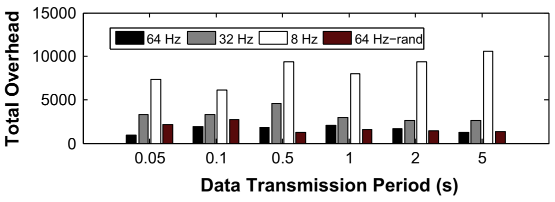
(a) Example of hand-off delay with different duty cycles.



(b)



(c)



(d)

Fig. 10. Impact of network duty cycling on mRPL performance with different network traffics: (a) an example of hand-off delay with different duty cycles, and performance in terms of (b) packet delivery ratio, (c) average hand-off delay, and (d) total packet overhead.

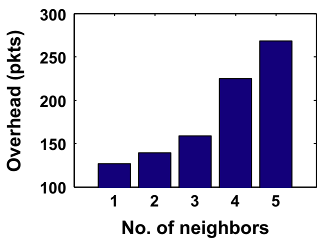


Fig. 11. Impact of network density on total overhead.

7. Experimental evaluation

In this section, we explain the experimental network setup in order to test and compare RPL and mRPL. The parameters' setting, topological configuration and the scenarios are described.

7.1. Network setup

In order to perform realistic experiments, we attached the mobile node to a person's body (Fig. 12(a)) and connected to the logging PC[7] to collect the information. The experiments were held in big room with 80 m^2 size and all nodes were running with their minimum transmission power (−25 dBm). Fig. 12(b) (Setup 1) shows a scenario where contiguous APs provide minimal overlap. This situation was achieved by selecting the lowest transmission power (power level = 1) and locating APs with a 0.3 m separation. In a more realistic scenario, Setup 2, we randomly deployed 9 APs in the room (as depicted in Fig. 12(c)). The APs were attached to walls at 1.5 m height from the ground (to guarantee a better connectivity). We will show the results later in this section.

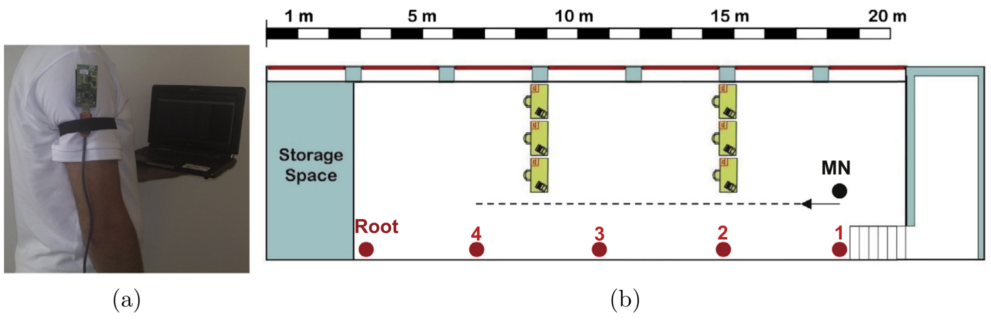
RPL configurations. In general, RPL devices play the role of a router or root node. In our experiments, we consider a single root that collects all data. The access points and the mobile nodes are routers; the MNs generate data and the APs forward them to the root.

In order to compare RPL with mRPL, we created the best possible RPL setting to switch fast between parents when a child node moves. Typically, in RPL, a child node needs to detect a high ETX value to trigger a parent switch. The frequency of ETX updates depends on the network traffic in terms of rate of data/control exchanges. We considered the highest possible data rate to increase the RPL routing responsiveness to network dynamics.

7.2. Results and discussion

Experimental Setup 1. We compare various RPL scenarios with mRPL in a simple network topology presented in Fig. 12(b), which provides minimum overlap between contiguous APs. All nodes run NullMAC (full-time on), which is more useful for comparing mRPL and mRPL

[7] At the beginning, we connected all APs to one laptop with passive USB cables and USB2.0 hubs. Then we observed some data loss during data transfer through the UART port. Adding more PCs did not solve the problem completely. Hence, we managed to get the data log from the MN with the cost of a person carrying a laptop during the experiment.



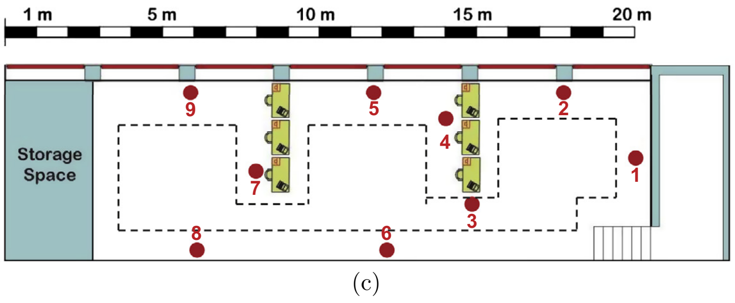


Fig. 12. Experimental evaluation, (a) the MN attached to the shoulder, (b) Experimental Setup 1 with 4 APs and a MN deployed in a row, and (c) Experimental Setup 2 with 9 APs distributed across the lab.

without the effect of packet losses and inherent delays in a duty cycling protocol.

First, we evaluate the packet delivery ratio of various RPL scenarios (previously defined in Table 3) with different data rates. Our analysis indicates that **higher data traffic leads to lower packet delivery ratio.** Smaller values of the Trickle timer and high data generation rate increase the network traffic, which in turn increase the chance of packet collision – see Fig. 13(a). The packet drops are more significant in larger Trickle timers (e.g. $\langle 12,8\rangle$), which results in nearly 46% packet drops when increasing the data rate from 1 to 30 pkt/s, while the lowest timer setting ($\langle 8,1\rangle$) exhibits nearly 29% drops.

Smaller values of the Trickle timer impose higher control packets overhead in RPL. The overhead is calculated according to the percentage of the ICMPv6 packets over the total number of packets (ICMPv6 packets + data packets). Fig. 13(b) shows that the overhead of RPL increases when choosing smaller Trickle values ($\langle 8,1\rangle$ scenario results in more control message exchanges compared with other scenarios). A lower data transmission rate results in a higher percentage of control messages with respect to the total number of packet exchanges.

Successful parent switching is based on the data rate and the Trickle setting. The number of DAOs corresponds to the number of new links created between a child (MN) and a neighboring parent. A high data transmission rate increases the ETX value updates. Fig. 13(c) shows that in all settings, the higher the packet rate, the lower the DAO transmissions. Additionally, smaller Trickle values increase the number of DAO packets.

mRPL performance is independent of the Trickle setting. In fact, mRPL uses RPL control messages as a backup mechanism. We compared mRPL with two extreme RPL scenarios ($\langle 12,8\rangle$ and $\langle 8,1\rangle$). Fig. 14(a) shows that regardless of the Trickle setting, mRPL copes with correctly

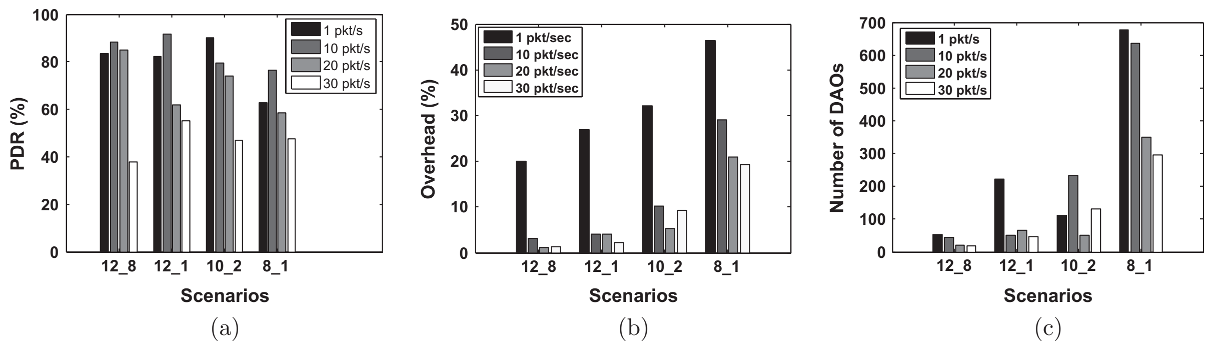


Fig. 13. Experimental Setup 1, comparing several RPL scenarios in terms of (a) packet delivery ratio, (b) overhead, and (c) number of DAOs.

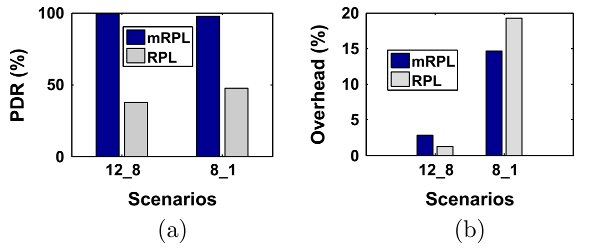


Fig. 14. Compare RPL and mRPL in terms of (a) PDR and (b) overhead in Experimental Setup 1.

delivering most of the data packets (nearly 100%). Note that reducing the Trickle timers decreases the mRPL packet reception rate by only 2%, as the data packets are more prone to collide with the control packets. The use of Trickle as a backup in mRPL raises the overhead of the algorithm in terms of additional control message exchanges. Hence, in mRPL it is recommended to use a low overhead Trickle setting (e.g. $\langle 12, 8\rangle$, as depicted in Fig. 14(b)).

Experimental Setup 2. We extended the tests by deploying APs as depicted in Fig. 12(b). All nodes were tuned to transmit power level 3 (−25 dBm), which created higher overlap between the neighbor APs. A root node was placed in the center of the room. The mobile node was attached to a person's arm (along the dashed path and it was generating packets at different rates).

A higher overlap of the wireless links increases the packet delivery ratio. Apparently, by creating more AP coverage overlapping, more packets have the possibility to reach the destination. However, RPL cannot support high transmission rates under mobility, as shown in Fig. 15(a). Contrarily, our experiments revealed that in an extreme condition with 30 pkt/s data rate, mRPL still forwards 99.7% of data packets.

Fig. 15(b) shows the overhead of RPL scenarios with different data rates. The trends are similar to the Experimental Setup 1. The best RPL setting with high data rate is $\langle 12, 8\rangle$, which leads to the lowest overhead. To update the routing information, RPL benefits from the data as well as control message exchanges. With the same setting in a high data rate application (30 pkt/s), mRPL resulted in 1% additional control messages overhead.

Higher links overlapping reduces the possibility of parent switching. The number of new links is smaller than for experimental Setup 1. By comparing the results in Figs. 13(c) and 15(c) in a high data rate condition, we conclude that the new links establishment (in a more realistic network topology) for $\langle 12, 8\rangle$ and $\langle 8, 1\rangle$ RPL scenarios reduces by 14% and 31%, respectively. Thus, we infer that higher links connectivity postpones the process of parent switching, and hence decreases the number of DAOs.

Standard RPL has no built-in hand-off mechanism. Therefore, it is hard to calculate the hand-off delay in RPL. In simulation, we have presented a rough estimation of the hand-off delay for different RPL scenarios. Empirical results show that mRPL has a very fast parent switching process with about 88 ms hand-off delay, leading to a very high packet delivery ratio even with high data transmission rates.

Further insight into Experimental Setup 2. Indoor experiments impose some limitations on the overall performance. The location of APs, furniture, people and the external interference affect the hand-off performance. In Fig. 16, the arrows correspond to the parent switching (from one AP to another). The thickness of the arrows

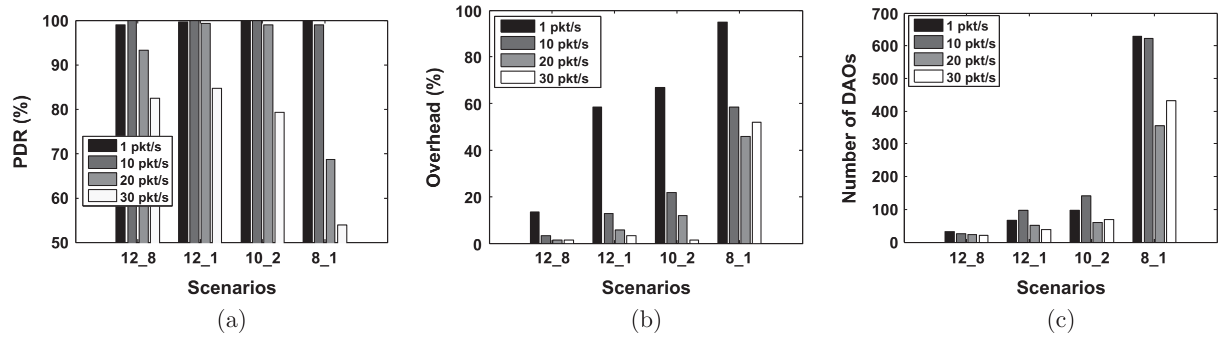


Fig. 15. Experimental Setup 2, comparing various RPL scenarios in a more realistic network topology in terms of (a) packet delivery ratio, (b) overhead, and (c) number of DAOs.

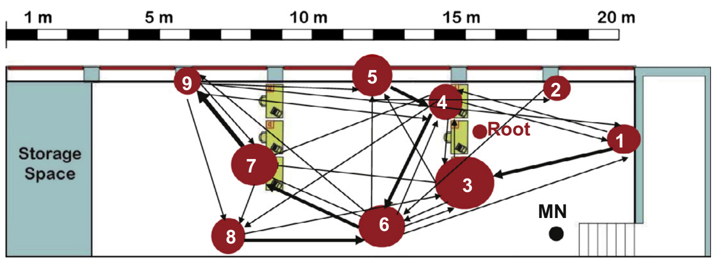


Fig. 16. Mobile node movement representation in Experimental Setup 2.

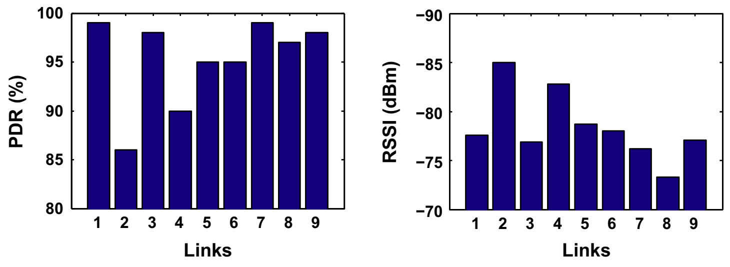


Fig. 17. Packet delivery ratio and the average RSSI measurement of each link in Experimental Setup 2.

indicates the amount of hand-off/s in the correspondent link. Note that hand-offs are not always performed between the closest APs. This means that the high variability of low-power wireless links and the dynamic behavior of the mobile network may dictate not choosing the closest APs. Fig. 16 also illustrates (with circles) the amount of packet exchanges with mRPL at each AP. Larger circles means more packets received (Nodes 3, 5, 6 and 7). Nodes in a good connectivity region (central location) can maintain the connection longer than the ones on the right and left sides of the room (Nodes 1, 2, 8 and 9); hence more packets are successfully received by “central” APs.

Fig. 17 shows the packet delivery ratio and the average RSSI at each AP (links 1–9 from the MN to the APs). There is a correlation between the average RSSI and the PDR in each link (higher ARSSI leads to higher PDR). This means that a hand-off triggered within the transitional region of the wireless link can result in a very good performance. Keeping the average RSSI and the PDR high requires a very careful decision on the moments of starting and ending a hand-off: closer to the lower threshold of the transitional region would reduce the packet delivery drastically. Nodes 3 and 7 are more benefited as they are placed in more strategic places with better ARSSI.

7.3. Simulation vs. experimental results

We compared the results of the simulation with the experimental tests in the two network setups: (i) Experimental Setup 1 as depicted in Fig. 12(b) and (ii) Experimental Setup 2 as depicted in Fig. 12(c). The results in Table 4 show that the performance in terms of packet delivery and hand-off delay in the experimental tests are relatively better than the simulation. The overhead in Setup 2 reduces drastically as the link overlap between the neighbor APs is higher than in the simulation. Moreover, the radio model of the simulator is inaccurate (mainly based on the distance between the nodes). However, in the real world, various physical and environmental parameters affect the radio model.

Table 4
mRPL: simulation vs. experimental results.

Scenarios	Sim. results	Exp. results

Setup 1 (Fig. 12(b))	PDR = 98.12%
	Overhead = 18.8%
	Delay = 81 ms
		PDR = 99.77%
		Overhead = 7.63%
		Delay = 92.7 ms

Setup 2 (Fig. 12(c))	PDR = 99.56%
	Overhead = 2.85%
	Delay = 86.21 ms
		PDR = 99.68%
		Overhead = 2.43%
		Delay = 88.2 ms

8. Related works

Mobility support in IP-based low-power networks is a recent research topic. The 6LoWPAN has been introduced to facilitate transmission of IPv6 packets over low powered networks. It incorporates an adaptation layer between the network and data link layers. Two routing schemes are used in 6LoWPAN, depending on the layer that makes the decision: (i) mesh-under and (ii) route-over [5,39]. Mobility solutions are applied on these routing schemes. The mesh-under routing supports communication in a single broadcast domain, where all nodes can reach each other by sending a single IP datagram. This scheme requires link-layer routing, since the multi-hop topology is abstracted by employing IPv6 support. The route-over routing supports a multi-hop mesh communication, where only immediate neighbors are reachable within a single link transmission.

In the following subsections, we address some of the related works on mobility support that focus on the mesh-under and route-over schemes. We summarize these works and their main features in Table 5,[8] including the solution we propose in this paper – mRPL – for the readers' convenience.

8.1. Mobility solutions within 6LoWPAN

In [40,41] a light version of Mobile IPv6 over 6LoWPAN is evaluated. In Mobile IPv6, movement detection is based on neighbor discovery, which is optional in 6LoWPAN [32]. In this work, the authors proposed Mobinet that relies on overhearing in the neighborhood of a mobile node. By detecting any changes in the neighborhood, the mobile node sends router solicitation in order to resume the neighbor discovery. The overhearing requires receiving all unnecessary packets by neighbor APs, which increases the network overhead and consequently the energy consumption.

Listening to the mobile node neighborhood activities increases MIPv6 responsiveness for detecting the mobility event. The authors claim that the hand-off delay with the light MIPv6 design is $\approx$ 130 ms ($\approx$ 85 ms in mRPL), while the solicitation timeout for periodic router solicitation is

[8] The high/low performance of the mobility enabled IP-based algorithms are rough estimations compared with the mRPL implementation. The explanations on these estimations are provided in Sections 8.1 and 8.2.

Table 5
Mobility solutions in IP-based low-power networks.

Reference	Routing mechanism	Mobility detection	Mobility solution	Additional hardware	Performance
Light MIPv6 [40,41]	Mesh-under	Overhearing	Light MIPv6	No	High overhead High energy High responsive
LowMOB [42]	Mesh-under	Periodic Beaconing	MIPv6	Yes	High overhead High energy High responsive
NoP [18]	Mesh-under	Periodic Beaconing	MIPv6	Yes	High overhead High energy High responsive
RPL for VANETs [43,44]	Route-over	Fixed DIO	Immediate ETX update	No	High overhead High energy High responsive
ME-RPL [45]	Route-over	Trickle	Adaptive DIS	No	Low overhead Low energy Low responsive
MoMoRo [46]	Route-over	Packet loss	Immediate Beaconing	No	High overhead Low energy Low responsive
mRPL	Route-over	Timers + Trickle + data packets	Immediate Beaconing	No	Low overhead Low energy High responsive

1 s. In mRPL, the periodicity of the mobility detection timer that guarantees network connectivity varies according to the network activities. In low connected networks, the time out enlarges, which will consequently reduce the overhead.

In LowMOB [42], the mobility detection is based on frequent beacon transmissions from the static nodes. A mobile node joins the AP with the highest RSSI level. The mobility support is based on conventional mobile IPv6. It also considers duty cycled APs, where the radios are turned off intermittently. By observing a low link quality at the current AP, it activates the next appropriate AP. To do so, the current AP performs a localization mechanism by using additional nodes, called mobility support points (MSPs), to find the direction of the MN.

The minimum hand-off delay of LowMOB in a single-hop scenario is ≈ 100 ms, which is comparable with the hand-off delay in mRPL. Using additional hardware for maintaining the mobility support is one of the disadvantages of LowMOB. In mRPL, we enable hand-off process on mobile nodes without adding any hardware or touching the default RPL mechanism. The conventional MIPv6 and the localization approach that are used in LowMOB require many packet exchanges in order to detect mobility and maintain the routing. This approach requires additional hardware and high processing capabilities. These two features are the main limitations of low-power networks.

The network of proxies (NoP) [18] provides a mobility support without interfering with the normal WSN behavior. The authors employ additional devices, which are called proxies. NoP devices are resource-unconstrained and handle the hand-off procedure (on behalf of sensor nodes). Similar to the LowMOB design, NoP requires additional hardware. This is one of the main disadvantages of the NoP design.

Proxies are responsible for monitoring the RSSI from MNs and share this information with other proxies. By analyzing this information, proxies decide for the next best parent of the MN. The mobile node is programmed to send periodic ICMP packets to the proxies. Then proxies communicate with each other to maintain network connectivity by performing hand-off process during mobility. The need for periodic signaling between mobile node and a proxy and also among proxies increases network overhead and energy consumption. In mRPL, the beaconing stops if there exists some activity in the medium. Moreover, neighbor APs reply only if there is a need for hand-off. The minimum hand-off delay in NoP is ≈ 117 ms, which is higher than the mRPL hand-off delay.

8.2. Mobility solutions within RPL

In [43,44], the authors focus on mobility support in RPL. The system model assumes a fixed set of nodes, while MNs get access to the fixed nodes directly or via multiple hops through other MNs. The mobility detection is obtained by employing a fixed timer (instead of the Trickle timer). The authors concluded that with higher DIO transmission, the connectivity increases, at the expense of additional overhead. To increase network responsiveness, DIO packets are transmitted more frequently. This method increases network overhead and energy consumption. In mRPL, the periodicity of DIO transmission is tuned based on the data transmission period to minimize the overhead.

Upon finding a new neighbor, immediate probing updates the ETX value to select the preferred parent in a timely fashion. To avoid loops after hand-off, the child nodes are discarded from the parent set. The proposed model has high overhead in terms of fixed and periodic beaconing. Moreover, it disables the Trickle timer, which is very useful in the absence of mobility. The evaluations are performed on a vehicular ad hoc network (VANET) setting with 24 Mbps transmission rate and a minimum of 25 mil/h speed. The results indicated the packet delivery of about 80% for a scenario with one MN and one AP (compared with $\approx 100\%$ in mRPL), which increased to $\approx 100\%$ for up to 10 MNs. Increasing the number of mobile nodes in a region increases network connectivity.

ME-RPL [45] assumes that mobile nodes are identified within static RPL nodes. By enabling a learning algorithm, nodes that change their parent more often query their neighbors with lower DIS intervals. It means that the DIS interval is dynamic according to the network inconsistencies. This strategy is also used in mRPL to dynamically adjust the beaconing interval and reduce network overhead and eventually the energy consumption. The MNs select static nodes with high quality links as their best parents.

In ME-RPL, sudden movements are not detected in real-time, since a learning algorithm is used. Thus, the problem of low responsiveness of the RPL routing to cope with environmental changes and inconsistencies still exists. In mRPL, a timer assures the timely detection of sudden movements (i.e. mobility detection timer). Tuning this timer guarantees network responsiveness. In ME-RPL, the maximum packet delivery ratio is $\approx 85\%$, which is outperformed by mRPL (roughly $\approx 100\%$ PDR).

MoMoRo [46] supports mobility in a sparse traffic network that runs RPL. It creates an additional layer between the data link and network layers to handle mobility detection. After a packet transmission failure, MoMoRo makes one more attempt to reach the destination by transmitting a unicast packet. If it fails again, MoMoRo starts searching for a new route by broadcasting beacons and collecting replies from neighbors. The results of MoMoRo evaluations show that it improves the packet delivery ratio up to 85%. However, it increases network overhead drastically, from 6 pkt/min in RPL to 65 pkt/min in MoMoRo. In mRPL, we achieved $\approx 100\%$ packet delivery ratio with only 22% more overhead compared with RPL routing.

In MoMoRo model, mobility detection depends only on the packet loss. This passive approach makes the network very low responsive to topological changes caused by mobility. Moreover, in low-power networks, it is very usual for the links quality to drop temporarily, causing packet loss, so a hand-off decision based on packet losses imposes unnecessary route maintenance that consequently increases network overhead.

9. Conclusion

This paper proposes a very simple yet effective solution to cope with mobility as one of the challenging issues for future IoT applications. We extend RPL – the standard routing protocol for the low-power networks in the IoT architecture – with fast and reliable mobility support.

We smoothly integrated a hand-off mechanism (dubbed smart-HOP) within RPL in a way to keep backward compatibility with the standard protocol. The smart-HOP hand-off mechanism [19,20] was applied to mobile nodes (MNs) by managing the schedules of the control messages within the Trickle algorithm (DIS, DIO and DAO). A MN selects a new preferred parent according to the average RSSI (ARSSI). Neighboring nodes within the child set of the MN are ignored, to prevent routing loops.

We implemented some timers to increase hand-off efficiency by reducing hand-off delays and network congestion. We considered the low-power link characteristics and the limitations of the IPv6 architecture to tune the schedules. We applied priorities to APs (according to the ARSSI levels) to minimize the probability of packet collisions during the hand-off process.

The integration of smart-HOP within RPL (dubbed mRPL) was tested, fine-tuned and validated through extensive simulations and experiments. The results we obtained indicate that an inaccurate radio propagation model in the simulator impacts results related to the hand-off performance: radio links overlap more significant in real experiments (the simulator creates a minimum overlap between neighbor APs).

We also found that the best setting of RPL parameters for mobile applications leads to a huge control message exchange overhead. Instead, mRPL is able to keep a low overhead while being responsive to network changes (≈ 85 ms hand-off delay). Moreover, nearly 100% packet delivery rate is achieved upon mobility.

We studied the impact of varying network traffic, duty-cycling and mobile node speed on the mRPL hand-off performance. The results indicated that in low traffic networks, the hand-off process is less responsive. Moreover, enlarging the listening periods affects the performance by increasing the hand-off delay. However, the variation of mobile nodes speed (within the range of human walking speed) does not affect the overall performance.

We implemented and integrated our hand-off mechanism in the RPL/6LoWPAN stack in Contiki, a widespread operating system for low-power wireless networks. Importantly, we made the source code freely available to the international community [1].

Acknowledgments

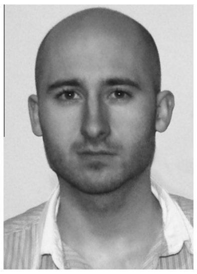
This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within projects FCOMP-01-0124-FEDER-037281 (CISTER), FCOMP-01-0124-FEDER-014922 (MASQOTS), and by the EU ARTEMIS JU funding, within project ARTEMIS/0001/2012, JU Grant Nr. 332987 (ARROWHEAD); also by FCT and by ESF (European Social Fund) through POPH (Portuguese Human Potential Operational Program), under PhD Grant SFRH/BD/71823/2010.

References

[1] mrpl: smart-hop within rpl, 2014. <http://www.cister.isep.ipp.pt/masqots/sources.html>.
[2] G. Mulligan, The 6loWPAN architecture, in: Proceedings of the 4th Workshop on Embedded Networked Sensors, ACM, 2007.
[3] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets Over IEEE 802.15. 4 Networks, Internet Proposed Standard RFC 4944.
[4] T. Winter, Rpl: IPv6 Routing Protocol for Low-Power and Lossy Networks, 2012.
[5] A.H. Chowdhury, M. Ikram, H.-S. Cha, H. Redwan, S.M.S. Shams, K.-H. Kim, S.-W. Yoo, 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, IWCMC '09, ACM, 2009, pp. 1208–1212.

[6] J. Hui, D. Culler, Extending ip to low-power, wireless personal area networks, IEEE Internet Comput. 12 (4) (2008) 37–45, http://dx.doi.org/10.1109/MIC.2008.79.
[7] Final Report from the NSF Workshop on Future Directions in Wireless Networking, November 2014. <http://ecedha.org/docs/nsf-nets/final-report.pdf?sfvrsn=0>.
[8] European Commission, Factories of the Future 2020: Multi-Annual Roadmap for the Contractual PPP Under Horizon 2020, Research Roadmap Produced by the European Factories of the Future Research Association (EFFRA), 2014. <http://www.effra.eu/attachments/article/129/Factories%20of%20the%20Future%202020%20Roadmap.pdf>.
[9] P.J. Marron, D. Minder, S. Karnouskos, The Emerging Domain of Cooperating Objects, Springer, 2012.
[10] J. Caldeira, J. Rodrigues, P. Lorenz, Toward ubiquitous mobility solutions for body sensor networks on healthcare, IEEE Commun. Mag. 50 (5) (2012).
[11] D. Zuehlke, Smartfactory–towards a factory-of-things, Ann. Rev. Control 34 (1) (2010).
[12] G. Wu, S. Talwar, K. Johnsson, N. Himayat, K. Johnson, M2M: from mobile to embedded Internet, IEEE Commun. Mag. 49 (4) (2011).
[13] GINSENG: Performance Control in Wireless Sensor Networks, 2014. <http://www.ucc.ie/en/misl/research/previous/ginseng/>.
[14] FASyS: Absolutely Safe and Healthy Factory, 2014. <http://www.fasys.es/en/proyecto.php>.
[15] Flexware: Flexible Wireless Automation in Real-time Environments, 2014. <http://www.flexware.at/>.
[16] J. Martinez-de Dios, A. Jimenez-Gonzalez, A. San Bernabe, A. Ollero, Conet integrated testbed architecture, in: A Remote Integrated Testbed for Cooperating Objects, SpringerBriefs in Electrical and Computer Engineering, Springer International Publishing, 2014, pp. 23–39. This work has been carried out within the Cooperating Objects European Network of Excellence. <http://www.cooperating-objects.eu/>.
[17] O. Chipara, W.G. Griswold, A.N. Plymoth, R. Huang, F. Liu, P. Johansson, R. Rao, T. Chan, C. Buono, WIISARD: a measurement study of network properties and protocol reliability during an emergency response, in: MobiSys, ACM, 2012.
[18] R. Silva, J.S. Silva, F. Boavida, A proposal for proxy-based mobility in WSNs, Elsevier J. Comput. Commun. 35 (10) (2012).
[19] H. Fotouhi, M. Zuniga, M. Alves, A. Koubaa, P. Marrón, Smart-hop: a reliable handoff mechanism for mobile wireless sensor networks, in: EWSN, 2012.
[20] H. Fotouhi, M. Alves, M. Zuniga, A. Koubaa, Reliable and fast hand-offs in low-power wireless networks, IEEE Trans. Mob. Comput. (2014).
[21] Z. Shelby, S. Chakrabarti, E. Nordmark, Neighbor Discovery Optimization for Low Power and Lossy Networks (6loWPAN), Work in Progress, IETF draft-ietf-6lowpan-nd-18.
[22] S. Cho, E. Jang, J. Cioffi, Handover in multihop cellular networks, IEEE Commun. Mag. 47 (7) (2009).
[23] I. Ramani, S. Savage, SyncScan: practical fast handoff for 802.11 infrastructure networks, in: INFOCOM, 2005.
[24] A. Mishra, M. Shin, W. Arbaugh, An empirical analysis of the IEEE 802.11 MAC layer handoff process, in: SIGCOMM 2003.
[25] M. Shin, A. Mishra, W. A. Arbaugh, Improving the latency of 802.11 handooffs using neighbor graphs, in: MobiSys, 2004.
[26] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: IEEE Workshop on Mobile Computing Systems and Applications, 1999.
[27] D.B. Johnson, D.A. Maltz, J. Broch, et al., DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks, Ad Hoc Netw. 5 (2001).
[28] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, S. Kozdrowski, On open shortest path first related network optimisation problems, Perform. Eval. 48 (1) (2002).
[29] T. Watteyne, A. Molinaro, M.G. Richichi, M. Dohler, From MANET to IETF ROLL standardization: a paradigm shift in WSN routing protocols, IEEE Commun. Surv. Tutorials 13 (4) (2011).
[30] A. Woo, T. Tong, D. Culler, Taming the underlying challenges of reliable multihop routing in sensor networks, in: ACM SenSys, 2003.
[31] P.A. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self regulating algorithm for code propagation and maintenance in wireless sensor networks, in: NSDI, 2004.
[32] T. Narten, W.A. Simpson, E. Nordmark, H. Soliman, Neighbor discovery for IP version 6 (IPv6), RFC 2461, December 1998.
[33] M. Zuniga, B. Krishnamachari, Analyzing the transitional region in low power wireless links, in: IEEE SECON, 2004.
[34] M.Z. Zamalloa, B. Krishnamachari, An analysis of unreliability and asymmetry in low-power wireless links, ACM TOSN 3 (2) (2007).
[35] T. Instruments, Cc2420 datasheet, 2007.
[36] A. Dunkels, B. Gronvall, T. Voigt, Contiki-a lightweight and flexible operating system for tiny networked sensors, in: IEEE LCN, 2004.
[37] Mobility Cooja Plugin, 2014. <https://github.com/contiki-os/contiki/wiki>.
[38] A. Dunkels, The contikimac Radio Duty Cycling Protocol, SICS Technical Report, 2011.
[39] J. Vasseur, N. Agarwal, J. Hui, Z. Shelby, P. Bertrand, C. Chauvenet, RPL: The IP Routing Protocol Designed for Low Power and Lossy Networks, Internet Protocol for Smart Objects (IPSO) Alliance.
[40] J. Montavont, D. Roth, T. Noël, Mobile IPv6 in Internet of things: analysis, experimentations and optimizations, Ad Hoc Netw. 14 (2014) 15–25.
[41] D. Roth, J. Montavont, T. Noel, Performance evaluation of mobile IPv6 over 6loWPAN, in: Proceedings of the 9th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN '12, ACM, 2012, pp. 77–84.
[42] G. Bag, M.T. Raza, K.-H. Kim, S.-W. Yoo, LoWMob: intra-PAN mobility support schemes for 6loWPAN, Sensors 9 (7) (2009) 5844–5877.
[43] K. Lee, R. Sudhaakar, L. Dai, S. Addepalli, M. Gerla, RPL under mobility, in: Consumer Communications and Networking Conference (CCNC), 2012 IEEE, 2012, pp. 300–304.
[44] K.C. Lee, R. Sudhaakar, J. Ning, L. Dai, S. Addepalli, J. Vasseur, M. Gerla, A comprehensive evaluation of RPL under mobility, Int. J. Veh. Technol. (2012).
[45] I. Korbi, M. Ben Brahim, C. Adjih, L. Saidane, Mobility enhanced RPL for wireless sensor networks, in: Third International Conference on the Network of the Future (NOF), 2012, pp. 1–8.
[46] J. Ko, M. Chang, MoMoRo: providing mobility support for low-power wireless applications, IEEE PP Syst. J. (99) (2014) 1–10.

Hossein Fotouhi is a Ph.D. student at CISTER/INESC-TEC Research Unit and University of Porto, working in the areas of sensor networks, mobile computing and Internet of Things. In 2009, he completed Master of Science at University Putra Malaysia on Communication Network Engineering. In 2004, he obtained Bachelor of Science at University of Guilan on Electrical Electronics Engineering. He worked 3 years afterward in industry as a network engineer.



Daniel Moreira received his degree in Electrical and Computer Engineering from the Polytechnic Institute of Porto – School of Engineering. He completed his MSc in Telecommunications in the same institute. He has been working in Wireless Sensor Networks with an emphasis in mobility management and hand-off mechanisms using commercial-off-the-shelf technologies to improve quality-of-service.

Mário Alves has a PhD (2003) in ECE at the University of Porto, Portugal. He is a Professor in ECE at Politecnico do Porto (ISEP/IPP) and a Research Associate at the CISTER/INESC-TEC Research Unit, a top ranked Portuguese research centre. His current research interests are mainly devoted to quality-of-service (QoS) in low-power wireless networks, particularly concerning timeliness and reliability issues.