Computer Communications 112 (2017) 1–13

Contents lists available at ScienceDirect

Computer Communications

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

A reinforcement learning-based link quality estimation strategy for RPL and its impact on topology management

Emilio Ancillotti[a], Carlo Vallati[b,*], Raffaele Bruno[a], Enzo Mingozzi[b]

[a] *Institute for Informatics and Telematics (IIT) – CNR, V. Giuseppe Moruzzi 1, 56124 Pisa, Italy*
[b] *Dipartimento di Ingegneria dell'Informazione, University of Pisa, Via Diotisalvi, 2, 56122 Pisa, Italy*

A R T I C L E   I N F O

Article history:
Received 8 May 2017
Revised 8 August 2017
Accepted 10 August 2017
Available online 14 August 2017

Keywords:
RPL
Topology and mobility management
Link quality estimation
Experimental evaluation

A B S T R A C T

Over the last few years, standardisation efforts are consolidating the role of the Routing Protocol for Low-Power and Lossy Networks (RPL) as the standard routing protocol for IPv6-based Wireless Sensor Networks (WSNs). Although many core functionalities are well defined, others are left implementation dependent. Among them, the definition of an efficient link-quality estimation (LQE) strategy is of paramount importance, as it influences significantly both the quality of the selected network routes and nodes' energy consumption. In this paper, we present RL-Probe, a novel strategy for link quality monitoring in RPL, which accurately measures link quality with minimal overhead and energy waste. To achieve this goal, RL-Probe leverages both synchronous and asynchronous monitoring schemes to maintain up-to-date information on link quality and to promptly react to sudden topology changes, e.g. due to mobility. Our solution relies on a reinforcement learning model to drive the monitoring procedures in order to minimise the overhead caused by active probing operations. The performance of the proposed solution is assessed by means of simulations and real experiments. Results demonstrated that RL-Probe helps in effectively improving packet loss rates, allowing nodes to promptly react to link quality variations as well as to link failures due to node mobility.

1. Introduction

The future Internet of Things (IoT) foresees information systems seamlessly integrated with smart objects, i.e. daily objects empowered with computation and communication capabilities [1,2]. This vision will require sensors and actuators deployed on a large scale for ubiquitous sensing and remote control of physical systems. In this context, Wireless Sensor Networks (WSNs) will represent a key enabler to guarantee low-cost and rapid deployment of IoT devices exploiting multi-hop data delivery over wireless links. Efficiency and reliability of multi-hop data forwarding will be essential to support future IoT systems and guarantee their robustness [3].

In general, the main objective of WSN routing protocols is to enable reliable communication while minimising resource consumption [4]. This is particularly important in low-power and lossy networks (LLNs), because they are characterised by unreliable links whose quality may significantly fluctuate over time influenced by external interference or obstacles. In this context, the selection of the optimal path is however challenging, as gathering topology information requires communication and processing overhead that contrasts with the limited computational and energy capabilities available to constrained devices.

Recently, significant efforts were put into defining a common routing protocol for IP-based WSNs, which have led to the standardisation of RPL, a gradient-based routing protocol that aims at building a robust multi-hop mesh topology over lossy links with minimal state requirements [5]. However, several studies have demonstrated that RPL is affected by reliability issues. The root cause of this unreliability is the lack of responsiveness to variations of network conditions that arise in real-life scenarios due to node mobility, or environmental factors, such as interference, multi-path effects, irregular radios patterns among the others [6]. Hence, many extensions have been recently proposed for the original RPL specification to support routing optimisations and more efficient mechanisms for route discovery and topology repair, especially under mobility [7–14]. However, *tweaking routing procedures does not solve the core problem of how to proactively maintain up-to-date information about links quality and network routes in a highly efficient manner*.

More generally, the availability of a highly efficient, accurate, and adaptive link-quality estimation (LQE) framework is essential

* Corresponding author.
E-mail addresses: emilio.ancillotti@iit.cnr.it (E. Ancillotti), carlo.vallati@iet.unipi.it (C. Vallati), raffaele.bruno@iit.cnr.it (R. Bruno), enzo.mingozzi@iet.unipi.it (E. Mingozzi).

http://dx.doi.org/10.1016/j.comcom.2017.08.005
0140-3664/© 2017 Elsevier B.V. All rights reserved.

to allow any routing protocol to select the best routing path under time-varying network conditions. It is well known that transmitting data over links with high quality improves both the network throughput, by limiting packet loss, and the network lifetime, by minimising the number of retransmissions. However, link quality estimation also plays a crucial role for detecting the dynamic behaviours of the links and maintaining the stability of the topology. For instance, LQE is needed to identify high quality links that are short-lived or to predict short-term variations of the quality of links. Then, costly route re-selection procedures that are triggered by link failures could be avoided. Thus, LQE in wireless sensor networks has received significant attention over the past years [15,16]. Unfortunately, there are several limitations in using existing LQE techniques with RPL. In particular, broadcast-based probing is commonly adopted for LQE in low-power wireless networks because it incurs a lower overhead than unicast-based probing. However, RPL employs the Trickle algorithm [17] to dynamically control the dissemination of routing control information, which makes complicate the implementation of broadcast-based probing without affecting normal RPL operations. Furthermore, even with broadcast-based probing the measurement overhead increases almost linearly with the number of neighbours and the probing frequency, thus consuming precious energy resources and worsening network congestion. On the other hand, infrequent or on-demand probing would yield poor measurement accuracy and reduce routing ability to adapt to the link dynamics in real time. A few approaches have been proposed to support adaptive LQE in multi-hop wireless networks to minimise measurement overhead. Unfortunately, these existing techniques cannot be used in LLNs because they require the maintenance of large link-state tables or exploit cross-traffic overhearing [18].

To address the above issues, in this paper we propose a novel *lightweight link monitoring scheme*, called RL-Probe, which is designed to ensure routing reliability (i.e., minimal packet loss rates) with lower overhead than classical periodic link probing, and without degrading the responsiveness to variable network conditions. The salient features of RL-Probe can be summarised as follows. First, RL-Probe *combines synchronous and asynchronous monitoring mechanisms* to maintain up-to-date information about link quality and their temporal variations while promptly reacting to sudden and unpredictable changes in network and link conditions. Secondly, RPL-probe uses *a reinforcement learning technique based on the multi-armed bandit model* [19] to dynamically control the probing procedures in order to automatically minimise measurement overhead without affecting responsiveness to route changes. Thirdly, RL-Probe preserves backward compatibility with standard RPL, i.e., enabling the interoperability of standard and enhanced nodes in the same network. It is also important to point out that the objective of RL-Probe is not to provide a novel routing metric for RPL, but a new measurement methodology that can be applied to generic link metrics, such as ETX [20].

To evaluate the effectiveness of our solution, we have integrated RL-Probe within the Contiki RPL/6LoWPAN protocol stack. The proposed solution has been compared against the standard RPL configuration, in which no active probe is employed, and RPL with periodic probing, in which active probe traffic is exploited to monitor links and handle channel quality variations. In addition to static scenarios, experiments involving mobile nodes have been run to analyse the performance of RL-Probe in challenging conditions. Indeed, mobile nodes are characterised by rapidly-changing link quality conditions that are usually managed using ad-hoc extensions of RPL rather than standard LQE solutions. In order to offer a fair comparison, mRPL, a state-of-the-art RPL enhancement explicitly designed to cope with mobility, is also considered in the scenarios in which mobility is involved. We conduct both simulations and experiments in an indoor testbed in a broad range of static and dynamic scenarios. We show that legacy RPL can suffer from packet loss rates of up to 65% with mobile nodes, while RL-Probe achieves a packet loss rate lower than 15% in the same scenarios. Furthermore, our results indicate that RL-Probe performs similarly to mRPL in terms of routing reliability, but it generates up to 30% less control messages and consumes up to 20% less energy per successfully transmitted packet. Finally, RL-Probe is faster than mRPL to detect sudden link quality variations thanks to the analysis of link trends, which allows RL-Probe to anticipate changes of link characteristics.

The remaining of this paper is organised as follows. Section 2 provides an overview of related work. In Section 3, we present RL-Probe and we evaluate its performance in Section 4. Finally, in Section 5 we draw our conclusions.

2. Background and related work

In the following subsections, we first provide background information on RPL. Then, we overview the works that are the most relevant to our study. Specifically, we review both LQE approaches for RPL aimed at monitoring links quality, and mechanisms proposed to improve mobility support in RPL. We also discuss solutions that apply machine learning techniques to LQE.

2.1. Background on RPL

RPL is an IPv6-based routing protocol specifically designed for lossy environments and resource-constrained embedded devices [5]. Specifically, RPL employs a distance vector routing algorithm which builds a logical topology on top of the physical network. In particular, the topology is a *Destination Oriented Directed Acyclic Graph, DODAG* for short. The root node of the DODAG initialises the DODAG formation by emitting DODAG *Information Object* messages (hereafter *DIOs* for short). Non-root nodes listen for DIOs and use the included information to join a DODAG. As a node joins a DODAG, it starts advertising its presence through the emission of DIO messages. Each DIO message specifies the rank of the sender, which is a scalar measure of the distance of that node from the root.[1] More recently, content-based approaches for path computation have bene also integrate in RPL [22,23]. Note that RPL can virtually split the network into multiple RPL Instances, which transport each kind of data according to its particular objective function [24]. To avoid loops in the logical topology the rank must monotonically decrease along an upward path towards the DODAG root.

As DIO messages are received from the neighbours, each node updates its view of the topology. In particular, node's neighbours with lower rank are selected to form a *parent set* which is used for data forwarding. Among them, a *preferred parent* is selected to forward traffic towards the root. Other important RPL control messages are the: (i) *DODAG Information Solicitation* (*DIS*), which is a multicast message used to trigger the transmission of DIO messages from neighbours; and (ii) *Destination Advertisement Object* (*DAO*), which is propagated upward (along the DODAG) to build the downward routes. To reduce the overhead associated to routing signalling RPL use a Trickle-based strategy [17,25], an adaptive beaconing scheme that exponentially increases the transmission timers when the network conditions are considered stable. This implies that the Trickle communication rate is not periodic. As better explained later, this may negatively affect the ability of RPL to quickly detect topology changes using routing control packets. Note

[1] The rank of each node is computed on the basis of an *Objective Function* (*OF* for short), which also defines how nodes select parents. Although the rank is not meant as a path cost, it is typically obtained from path metrics that are somehow function of the distance from the root, e.g., number of hops or each-to-end packet delays [21].

that when a network inconsistency is detected (e.g., a network loop or preferred parent change) a repair procedure is triggered. This repair mechanism can be local (e.g., the node detaches from the DODAG and tries to reconnect) or global (the current DODAG is invalidated and a new DODAG is built). Finally, Contiki also supports the Neighbour Discovery Protocol (NDP, defined in RFC 4861 [26]), which implements four types of ICMPv6 messages for the purpose of router advertisements and neighbour unreachability detection. However, when RPL is activated the router advertisement procedures of NDP are disabled as DIO and DAO messages are already used to announce the presence of other nodes and their link parameters. Furthermore, when receiving DIO or DAO packets, RPL populates the NDP neighbour info table. Then Neighbour Solicitation (NS) and Neighbour Advertisement (NA) messages are generated by the Neighbour Unreachability Detection (NUD) mechanism that is included in NDP to refresh the table and to verify that a neighbour is still reachable through a cached link layer address.

2.2. LQE in RPL

There is a large body of research on LQE in WSNs and it is out of the scope of this paper to overview the several link quality metrics that exist in the literature (the interested reader is referred to the comprehensive survey in [16]).

From a general perspective, an LQE framework consists of three components: (*i*) *link monitoring*, which is the mechanism used to collect link measurements; (*ii*) *link measurement*, which specifies the information to be retrieved, and (*iii*) *metric evaluation*, which defines the metric to assess the link quality. First RPL implementations employed simple passive monitoring techniques, which leverage statistics of transmission failures for the links used by data traffic [27]. However, data-driven link estimation methods do not apply to idle links, and the reactivity of such methods heavily depends on the network traffic patterns. Furthermore, overhearing is required to monitor links to neighbouring nodes that are not the preferred parent [28]. Thus, recent RPL implementations prefer active monitoring to measure link quality through probes. The Contiki 3.0 RPL implementation, for instance, includes an optional probing mechanisms (dubbed RPL-PP), in which neighbours are probed periodically through unicast DIO messages. The target of the probe is selected following a round robin strategy, except the link to the preferred parent that pre-empts the other links to neighbouring nodes in the probing schedule if its quality has not been updated in a given interval.

It is generally agreed that unicast-based probing provides accurate link measurements [29]. The downside is that neighbours are assessed individually, which results into long convergence times for link quality estimation, especially when large or dense networks are considered. In order to reduce the measurement delay, a broadcast-based probing has been proposed in many RPL extensions. This is implemented in different ways, e.g. forcing the periodicity of routing control messages [30], or sending bursts of RPL control messages during specific phases of network operations (e.g. at network formation) or at the occurrence of certain events (e.g. during parent changes) [10]. Those schemes are sender-initiated, as the transmitting node generates the probe packets. One the other hand, some RPL variants use receiver-side probing schemes, which use information from received probes to estimate link quality. For instance, in [9] each node triggers the link monitoring by sending a multicast DIS message to its neighbours, which reply with a train of unicast DIS messages. The advantage is that multiple neighbours can be monitored in parallel even if unicast probes are used. One potential drawback of this approach is that the metric computation is performed on the opposite direction to data transmission. However, many experimental studies show the symmetry of wireless links in real use cases of WSNs [15,31].

All the above-described mechanisms adopt basic measurement schemes. However, there are a few examples of more sophisticated solutions for LQE, which use a hybrid approach by combining multiple complementary methods. For instance, a link-quality measurement framework is proposed in [18], which combines three measurement techniques: passive, cooperative, and active monitoring. However, this framework cannot be easily applied in low-power wireless networks because it requires the maintenance of large link-state tables and leverage cross-traffic overhearing. Differently from this prior work, RL-Probe proposes a lightweight framework that can be implemented in constrained devices in which different methods for LQE are adopted.

2.3. Machine learning for LQE

During the past decade, machine learning algorithms and computation intelligence methods have been increasingly applied in wireless sensor networks to improve network performance and solve a variety of networking problems [32,33]. In particular, prediction models that utilise different machine learning algorithms have been frequently proposed to automatically estimate link quality. For instance, a logistic regression classifier is presented in [34] that uses a combination of physical parameters, such as RSSI and SNR, to predict the success probability of the next transmission. An online learning algorithm based on artificial neural networks is described in [35] to predict short-term quality variations. Decision trees and classification rules for supervised learning of link quality are developed in [36]. Online channel quality estimation is modelled as a game of prediction with expert advice in [37]. Unsupervised feature selection is proposed in [38] to determine the dominant features for the performance of end-to-end links. Multi-arm bandit problems have been also proposed to deal with the channel exploration-exploitation dilemma in wireless networks. For instance, the authors in [39] formulate the channel selection problem in cognitive radio networks using a multi-arm bandit model, and propose a fast converging sampling method. The channel decision problem is modelled as a restless multi-arm bandit problem in [40]. In the context of computational intelligent algorithms for link quality prediction, fuzzy logic is frequently used. For instance, authors in [11,41] propose fuzzy rules to combine multiple link metrics while compensating for the uncertainties in the wireless channel conditions.

Our work differs from the above-mentioned studies, as we use machine learning techniques to increase the efficiency of link sampling and not to develop new predictive models of link performance. Multi-armed bandit problems have been extensively applied in the context of channel selection in the cognitive radio context. However, the channel probing problem is different from the link measurement problem considered in this paper, as in the former case the objective is to infer the binary channel state (unoccupied or busy), while in this study we focus on optimising the trade-off between probing frequency and responsiveness in detecting short-term link quality variations.

2.4. RPL mobility extensions

A survey of recent RPL extensions and modifications to improve mobility support is provided in [12]. Previous studies have shown that RPL provides a fast network setup but that mobility support is not adequate and it should be improved [42]. A few works have investigated neighbour discovery protocols that improve the capability of mobile devices to remain connected to the WSN as they move [43,44]. However, most of existing solutions for mobility management in RPL focus on modifying native RPL mechanisms to improve RPL ability of detecting link failures, or to speed up the handoff and local repair procedures [45].

In ME-RPL [7] nodes are separated as *mobile* nodes (i.e. nodes with unstable links) and *fixed* nodes, with mobile nodes forced to act only as leaf nodes to avoid network paths through them. Then, mobile nodes must advertise their mobility status in control messages and generate frequent DIS messages to update their parent set information. The time between DIS messages is computed based on the frequency of preferred parent changes using a simple multiplicative increase/multiplicative decrease algorithm.

MoMoRo [11] quickly reacts to packet transmission failures by immediately resending the lost packet, and starting a new route search (by broadcasting beacons and collecting updated routing information from the neighbours) if the second attempt also fails. Furthermore, MoMoRo uses a fuzzy estimator that combines different link metrics (ETX, average RSS, symbol error rate variance) to classify links based on their probability of disconnection.

As for ME-RPL, the authors in [8] propose that mobile nodes only connect as leaves in the DODAG. Furthermore, a reverse Trickle timer – the timer starts from the maximum value and halves the DIO sending intervals after each new DIO transmission – is used by the preferred parents of mobile nodes to trigger DIO messages. The implicit assumption of this approach is that mobile nodes' stability decreases with time. An adaptation of the Trickle timer algorithm for better mobility support is also proposed in mod-RPL [13]. Specifically, mod-RPL takes into account the trajectory and velocity of mobile nodes when selecting the sending interval of DIO messages, and it dynamically adapts the timer to the distance between the mobile node and its preferred parent.

KP-RPL is a position-based extension of RPL to provide mobility support [46]. Standard RPL is used for routing between fixed nodes, while mobile nodes make their routing decisions using positioning information obtained by applying a Kalman filter on RSSI measurements. Furthermore, mobile nodes generate a blacklist to discard neighbours that could not be reachable due to positioning errors.

In conclusion, all the above-described schemes are specifically designed to handle mobility through ad-hoc mechanisms that usually do not perform accurate LQE. Differently from such solutions, our proposed approach is not restricted to support only node mobility, but it can seamlessly manage node mobility through fine-grained LQE, without requiring modifications of standard RPL or a-priori knowledge of which nodes are mobile.

Overview of mRPL. We present mRPL [10] in more details as it will be used as benchmark in the following evaluation.[2]

Basically, mRPL integrates smart-Hop, a handoff scheme for low-power networks [48], in RPL. As in [7,8,46] mobile and fixed nodes are separated. Then, mobile nodes monitor the link quality by receiving DIO messages from their preferred parents. A mobile node disconnects from the preferred parent and enters a discovery phase if the average received signal strength is below a given threshold or if no packets are received by the parent before a connectivity timer (T_C) expiration. To avoid that the high variability of wireless links causes frequent handoffs a hysteresis mechanism is applied to this RSSI threshold. During the discovery phase, the mobile node multicasts DIS messages to all neighbouring nodes and collects their unicast DIO replies to decide which new preferred parent to select based on the average RSSI level. As an additional stability mechanism, a mobile node repeats the discovery procedure m times after switching to a different preferred parent to check the stability of that node.

mRPL introduces additional timers to increase handoff efficiency and reliability. In particular, the mobility detection timer (T_{MD}) is used to detect connection losses due to the existence of external objects (obstacles between the sender and the receiver). The handoff timer establishes the transmission period of DIS messages to the neighbouring parents. Finally, a reply timer (T_R) is used to select the time instant at which a parent should reply to the mobile node to reduce the probability of collision between control messages during the handoff process.

It is important to emphasise that mRPL is not as general as RL-Probe, since it requires to a priori configure nodes as either mobile or static (called APs) in order to perform different operations. In fact, mobile nodes are restricted to be leaf nodes in the RPL DODAG. This may cause inefficient routing in fully static networks, as also shown in Section 4.

[2] Note that a recent enhancement of mRPL, called mRPL+, has been presented in [47]. The core mechanisms of mRPL and mRPL+ are similar and our findings can be applied to mRPL+ as well.

3. The RL-Probe framework

This section describes RL-Probe in details. First, we provide background material on the multi-armed bandit model. Then, we explain the design rationale behind RL-Probe. Finally, we detail the main mechanisms and algorithms used in RL-Probe.

3.1. Multi-armed bandit background

Generally, a multi-armed bandit (MAB) model is used to describe a learning problem in which an agent must repeatedly choose among different options, or actions [19]. After each choice, the agent receives a reward chosen from an *unknown* stationary probability distribution that depends on the selected action. The objective of the agent is to maximise the expected total reward over a time horizon. The MAB model is so named by analogy to a slot machine that has multiple levers, with different but unknown probabilities of hitting the jackpot. Then, the player should try to find the best levers that maximise the winnings by repeatedly playing.

More formally, let us assume that time is discretised and time slots are numbered as $n = 1, 2, \ldots$. Then, let $\mathcal{U}$ be the set of actions (or arms). At round n, the arm pulled is $u^{(n)}$ and the reward received is $W^{(n+1)}$. We assume that the reward provided by arm u follows a random distribution $F_u(x)$, which is unknown. Now, let $s^{(n)}$ be the representation of the system state at round n and let $\mathcal{S}$ be the discrete set of possible states. The history of the system at a given stage is the sequence of decisions, observed states and collected rewards. For the sake of tractability, MAB models usually assume the *Markov property*, i.e., rewards depend only on the current state and the current action and not on the full history of previous actions and states. Then, the core of a MAB model is the policy π, namely the mapping function between states and actions, which should maximise the amount of rewards the agent receives over time. Several methods have been proposed in the literature to learn the optimal policy without requiring a model of the system behaviours, but leveraging only on the experience obtained by iteratively interacting with the system. As discussed more in depth in the following sections, these learning techniques have to cope with the *exploration/exploitation dilemma*, which implies balancing immediate gains (i.e., selecting the action with the maximum expected reward) with knowledge creation to make better decisions in the future (i.e., selecting actions that appear to be worse but could potentially be the best). Typically, a probabilistic learning strategy is defined that assigns a probability to each possible action in a state according to an estimation of the current state value.

3.2. Overview of RL-Probe

One of the main distinct features of RL-Probe over existing probing strategies for LLNs is that it adopts a *hybrid* approach to adaptively combine *synchronous* and *asynchronous* LQE techniques.

More precisely, the synchronous LQE technique relies on unicast probes to provide accurate link quality measurements. However, we design two novel mechanisms to make unicast-based probing more adaptive and responsive, without introducing excessive probing overhead. First, we formulate the selection of the probing period as a multi-armed bandit problem to dynamically adjust the probing frequency to the link variability in real time. Our learning-based approach provides a good trade-off between overhead and responsiveness to varying link quality. Secondly, we cluster neighbours into groups and we assign different probing priorities to each group. The clustering and the priority selection are based on the importance of each node in RPL route maintenance and recovery procedures. The rationale behind this approach is that wireless link correlation (e.g., due to cross-network interference under shared medium) has been observed in many recent studies [49]. Consequently, independent estimates of individual link quality can lead to redundant measurements, especially in dense networks.

Our asynchronous LQE mechanism is designed to efficiently handle sudden and disruptive link variations. To this end, we integrate in RL-Probe a receiver-side probing method first proposed in [50], which allows to rapidly assess the quality of the links from a node to all its neighbours with a sufficient accuracy. In principle, asynchronous probing could facilitate the isolation of faulty nodes/links, or the detection of preferred parent unavailability due to mobility. Clearly, asynchronous probing must be activated on-demand when it is most likely needed because it is costly in terms of energy and bandwidth consumption. Thus, we also propose specific triggering mechanisms for our asynchronous LQE technique, which are based on the trend of received signal strength indicator (RSSI) and link quality (ETX) values. Our solution reduces the cost of recovering RPL connectivity by ensuring quick detection of network disruptions without depleting the limited resources of the devices. Finally, it is worth pointing out that RL-Probe is specifically designed for single-channel MAC protocols. A number of multi-channel MAC protocols for low-power sensor networks, which perform channel hopping rather than allocating fixed channels to data collection trees, have been also studied [51]. However, how to implement efficient broadcast and unicast probing devices switches periodically between channels is still an open research issue.

For the sake of clarity, the pseudocode of the main mechanisms of RL-Probe are provided in Algorithms 1–4.

3.3. Asynchronous probing

As outlined above, the proposed asynchronous probing scheme exploits the measurements of RSSI values and ETX metric to detect sudden link variations. More precisely, whenever a node i receives a packet from a neighbour j, it obtains the RSSI value from the wireless transceiver (line 3 in Algorithm 1). A list of the most recent RSSI values is maintained for each link $l_{i,j}$ (line 5 in Algorithm 1), which is used to estimate the trend in RSSI variations (line 5 in Algorithm 1). For the sake of computational efficiency, in our implementation we estimate the RSSI trend using a simple moving average (SMA) filter over the last four measurements of RSSI differences between consecutively received packets. More formally, each node i maintains a list with the previous n RSSI samples it has received from neighbour j. If $rssi_{i,j}[k]$ denotes the last received RSSI sample, this list consists of the values $(rssi_{i,j}[k], rssi_{i,j}[k-1], \dots, rssi_{i,j}[k-(n-1)])$. Then, the RSSI trend is computed as the unweighted mean of the previous three RSSI differences. This can be written as:

$$rssiTrend_{i,j}[k] = \sum_{l=0}^{n-1} \frac{\big(rssi_{i,j}[k] - (rssi_{i,j}[k-l]\big)}{n}, \tag{1}$$

Algorithm 1 Main procedure of RL-Probe.

```
Require: $\mathcal{N}_i$                                ▷ set of neighbours
 1: loop
 2:     if (received packet $p$ from $j$) then
 3:         Get RSSI value ($rssi$) from packet $p$;
 4:         $rssi_{i,j}[].add(rssi)$;
 5:         Get RSSI trend ($rssiTrend_{i,j}$) for link $l_{i,j}$ from the last $n$ RSSI samples;
 6:         if ($p$ = ACK and $j = pp$) then
 7:             $\Delta rssi_{i,j} = \frac{rcvTh - rssi}{rcvTh}$;
 8:             if ($rssiTrend_{i,j} < 0$ and $\Delta rssi_{i,j} \le \alpha$) then
 9:                 do receiver-side probing;
10:                 for all $k \in \mathcal{N}_i$ do
11:                     update link stability ($cv_{i,k}$) for $l_{i,k}$;
12:                 end for
13:             end if
14:         else if ($p$ = NACK and $j = pp$ and $cv_{i,j}[].get(last) \le \beta$) then
15:             do receiver-side probing;
16:             for all $k \in \mathcal{N}_i$ do
17:                 update link stability ($cv_{i,k}$) for $l_{i,k}$;
18:             end for
19:         end if
20:     else if (timer $T_p$ expires) then
21:         update nodes in set $\mathcal{P}_i$ and $\mathcal{O}_i$;
22:         $x \leftarrow Uniform(0,1)$;
23:         if ($x \le \epsilon$) then                 ▷ exploitation phase
24:             $u \leftarrow \underset{x}{\operatorname{argmax}}\big(W^{(x)}[].get(last)\big)$;
25:         else                                      ▷ exploration phase
26:             $u \leftarrow random(D_1, D_2, D_3)$;
27:         end if
28:         if ($x = D_1$) then
29:             $j \leftarrow$ BestNode($\mathcal{P}_i$)
30:         else if ($x = D_2$) then
31:             $j \leftarrow$ BestNode($\mathcal{O}_i$)
32:         end if
33:         if ($x \ne D_3$) then
34:             do unicast probing to $j$;
35:             update link stability ($cv_{i,j}$) for $l_{i,j}$;
36:             UpdateUtility($i$, $j$);
37:         end if
38:         UdapteReward(i,j,u)
39:     end if
40:     if (sent data packet to $pp$) then
41:         update link stability ($cv_{i,j}$) for $l_{i,j}$;
42:         UpdateUtility($i$, $pp$);
43:     end if
44: end loop
```

Algorithm 2 Description of the function for utility update.

```
1: function UpdateUtility($i$, $j$)             ▷ $l_{i,j}$ probed link
2:     $\omega_{i,j}[].add\big(\mu_{i,j}[].get(last) + \sigma_{i,j}[].get(last)\big)$;
3:     $\Delta\omega_{i,j}[].add\big(\omega_{i,j}[].get(last) - \omega_{i,j}[].get(last-1)\big)$;
4:     if ($\Delta\omega_{i,j}.get(last) \cdot \Delta\omega_{i,j}.get(last-1) > 0$) then
5:         $U_{i,j} \leftarrow U_{i,j} + |\Delta\omega_{i,j}[].get(last)|$;
6:     else
7:         $U_{i,j}(n) \leftarrow 0$ ;
8:     end if
9: end function
```

Algorithm 3 Description of the function for reward update.

```
1: function UpdateReward($i, j, u$)    ▷ $u$ MAB action
2:     if ($u = D_1$) then
3:         $W_i^{(D_1)}[].add\left(\max\left[0, \max_{j \in P_i} U_{i,j}(n) - C_1\right]\right)$ ;
4:     else if ($u = D_2$) then
5:         $W_i^{(D_2)}[].add\left(\max\left[0, \max_{j \in O_i} U_{i,j}(n) - C_2\right]\right)$;
6:     else
7:         $W_i^{(D_3)}[].add\left(\max[0, G_{np} - U_{i,pp}(n)]\right)$;
8:     end if
9: end function
```

Algorithm 4 Description of the function for the selection of the best node.

```
1: function BestNode($\mathcal{A}$)    ▷ candidate node to be probed
2:     $x \leftarrow Uniform(0, 1)$;
3:     if ($x \le \epsilon$) then    ▷ exploitation phase
4:         Get node $i \in \mathcal{A}$ with the highest utility;
5:     else
6:         Get node $i \in \mathcal{A}$ randomly;
7:     end if
8: end function
```

with $n = 4$ in our implementation. A similar SMA filter is also maintained for the ETX measurements obtained from data packets and unicast probes. Then, our asynchronous probing has both a *proactive* and *reactive* phase. The proactive phase tries to anticipate topology changes and it is activated when a packet is successfully transmitted to the preferred parent *pp*[3] (i.e., the sender receives a MAC ACK from the preferred parent) but a set of simultaneous conditions occur that suggests a possible degradation of the link quality. Specifically, the following three conditions must be satisfied to trigger a receiver-side probing[4] (lines 9–11 in Algorithm 1) : *i*) the RSSI trend is negative; *ii*) the last RSSI sample is close to the receiver sensitivity *rcvTh*, and *iii*) the ETX trend is negative.

On the other hand, the reactive phase is designed to facilitate local repair operations after unexpected network disruptions. In this case, the receiver-side probing is activated when there is a transmission failure (i.e., the sender receives a MAC NACK from the preferred parent) on a link that has a stable link quality. The link quality stability is measured using the coefficient of variation of the link quality (defined as the ratio between the standard deviation $\sigma_{i,j}$ and the mean $\mu_{i,j}$). More formally, a link is considered stable if $cv_{i,j} \le \beta$ (line 14 in Algorithm 1). Note that the selection of threshold β is dictated by the variability of the wireless links. Links with $cv_{i,j}$ lower than one are considered low-variance. Summarising, both a successful transmission on a rapidly degrading link and a packet loss on a stable link activates the asynchronous probing for updating the link quality to all nodes in the neighbourhood (lines 15–17 in Algorithm 1).

3.4. Synchronous probing

Differently from conventional unicast-based probing schemes, our synchronous probing does not use a fixed probing interval, which would cause unacceptable convergence delays for the link quality estimation, especially in dense networks. On the contrary, RL-Probe clusters nodes into separate groups and adaptively adjusts the probing period for each group. Different approaches for such link clustering can be devised, taking also advantage of cross-layer information from the network layer. For instance, in this study we define a link clustering that considers the importance of a neighbour to maintain good RPL connectivity. More precisely, let us denote with $\mathcal{N}_i$ the set of neighbours of node *i*. We define the set $\mathcal{P}_i$ that contains the best m_p parents of node *i*, i.e. parents that have the lowest path cost to the sink if selected as next hop by node *i*. Similarly, we define the set $\mathcal{O}_i$ that contains up to m_o nodes from the set $\mathcal{N}_i \setminus \mathcal{P}_i$, which have the lowest path cost to the sink if selected as next hop by node *i*. The m_p and m_o parameters should be selected as a trade-off between reliability and responsiveness. Large m_p and m_o values provide coarse grained information about the links and decrease the responsiveness of the system. On the other hand, low m_p and m_o values reduce the possibility to discover good alternative paths in case of loss of the preferred parent. In general, $m_o > m_p$ as it is more critical to have detailed information on nodes of the parent set.

The decision-making process that determines which set to probe and how frequently to probe it is formalised through a MAB model. Specifically, we define three possible actions as follows:

D_1: probe a node in $\mathcal{P}_i$;
D_2: probe a node in $\mathcal{O}_i$;
D_3: skip the probing.

Each of the above decisions corresponds to an independent arm in the MAB problem. The reward associated to each probing decision, say $W^{(x)}$ with $x \in \{D_1, D_2, D_3\}$, corresponds to the potential *gain* provided by probing a node in a specific group. For instance, the gain can be a measure of the improved network responsiveness to link quality variations. Details on reward estimation are provided later in this section.

First of all, let us explain which is the policy used by node *i* to select the probing action given the knowledge of the average rewards. In this study, we adopt the well-known ϵ-*greedy* algorithm, which selects with probability ϵ the action with the maximum accumulated reward, or *greedy action* (lines 23–24 in Algorithm 1), and with probability $(1 - \epsilon)$ selects a random action (lines 25–26 in Algorithm 1). By properly tuning the ϵ parameter it is possible to balance exploration and exploitation phases to fast converge to the optimal action selection. If action D_1 or D_2 are selected it is necessary to decide which neighbour to probe in the set $\mathcal{P}_i$ and $\mathcal{O}_i$, respectively. As anticipated above, we exploit a measure of the utility of a node for the RPL topology maintenance procedure. Note that this utility measure is also used in the reward computation. In the following we formalise the algorithms used in RL-Probe to compute the utility and reward metrics.

Utility and reward computation. In RL-Probe, the reward function is mainly used to estimate the trends in link quality variations (e.g., quality degradation for an interfered link). To this end, we follow the same approach as in [52] and we use the *mean* and the *standard deviation* of the link quality metric. More formally, let us assume that each node maintains a list of the estimated values of the mean $\mu_{i,j}$ and standard deviation $\sigma_{i,j}$ of link $l_{i,j}$.[5] We introduce an aggregate measure $\omega_{i,j}$ of the link quality variability, as the sum of $\mu_{i,j}$ and $\sigma_{i,j}$ (line 2 in Algorithm 2). We can easily compute the incremental variation of the link quality variability as the difference of two consecutive samples of $\omega_{,j}$ (line 3 in

[3] As introduced in Section 2.1, the preferred parent of a node on a path towards the DODAG root is the parent with the lowest rank value.

[4] We recall that during a receiver-side probing phase a node sends a multicast DIS message and its neighbouring nodes reply with a train of unicast DIS messages.

[5] An exponential moving average (EMA) filter with smoothing factor 0.8 is used to update these estimates.

Algorithm 2). Intuitively, it might be appropriate to monitor more frequently links that are showing a clear *trend*, in order to timely identify a link that is quickly degrading (e.g., due to an external interference) or improving. Thus, we associate a positive utility to links that showed the same trend of link quality variation in the last two probes (lines 4–5 in Algorithm 2), while we assign a null utility to links that are not characterised by a steady (positive or negative) trend (line 7 in Algorithm 2). Clearly more sophisticated utility functions can be designed to utilise a large number of previous samples and more complex decision rules. However, our main objective is to determine the feasibility of our approach and we only check the trend of the last two samples for the sake of implementation simplicity. Now, we can also clarify how the BestNode($\mathcal{A}$) function chooses the neighbour to probe in set $\mathcal{A}$. In the simplest case, it could select the node with the highest utility. However, this would make impossible to check, even infrequently, links with small utilities (i.e., more variable links). Following the same line of reasoning of the above-discussed ϵ-greedy exploration strategy, the BestNode($\mathcal{A}$) function selects the node with the highest utility with probability ϵ (line 4 in Algorithm 4), while a random link in the set $\mathcal{A}$ in the other cases (line 6 in Algorithm 4).

Commonly, reward functions for learning problems should include a positive term and a negative term to be well specified. The positive term measures the gain of performing that action. For the case of actions D_1 and D_2 the gain is given by the highest utility value in the group (lines 3 and 5 in Algorithm 3). Thus, the reward of a link cluster is high if there is at least one link in the set with a consistent variability pattern for its link quality. Intuitively, the cost for actions D_1 and D_2 should be a measure of the cost of a unicast probe. Since we want to give higher priorities to probing parents than other neighbours, we have that $C_1 \geq C_2$. For the same reason, we assume that skipping a probing phase provides a gain G_{np}, in terms of saved node and network resources (line 7 in Algorithm 3). As a cost for the action D_3 we use the utility of the link with the preferred parent because if the link with the preferred parent is not stable a node should keep looking for alternative links.

Group management. As explained above link clustering is controlled using the path cost. It is important to point out that link quality variations, especially for neighbours with intermediate link quality, may yield to frequent changes in the cluster composition. However, this can negatively affect the convergence of the learning algorithm. Therefore, we define a *hysteresis margin* for the sojourn time of a node in the set $\mathcal{P}_i$. Specifically, when a node j in the set $\mathcal{P}_i$ is not anymore among the best m_p neighbours for node i it would be removed only if this condition persists for at least a time t_{hyst}. This check is implemented in the function UpdateClusters($\mathcal{P}_i, \mathcal{O}_i$) (line 21 of Algorithm 1).

Preferred parent monitoring. . In RL-Probe the link quality to the preferred parent is estimated by passively monitoring the data traffic, as in legacy RPL. For this reason, the preferred parent is not part of the set $\mathcal{P}_i$. However, the link quality measurements are still used to update the utility estimates for the preferred parent (line 42 of Algorithm 1).

4. Performance evaluation

In order to implement and evaluate RL-Probe, we opted for the Contiki 3.0 operating system (OS). The main reasons for selecting Contiki are: (i) the support of Cooja simulator, which allows to easily port the software on real hardware, (ii) the availability of a standard RPL implementation that is widely used, and (iii) the availability of several plugins that already implement mobility models, interference models and probing techniques. Table 1 summarises the RL-Probe parameters used in the algorithms described in Section 3, which have been fine-tuned with extensive simulations.

Table 1
RL-Probe protocol parameters.

Parameter	value
α	3%
β	1
m_p/m_o	3 / 10
C_1/C_2	1 / 5
G_{np}	10
ϵ	0.7
t_{hyst}	10 min
T_p	1 min

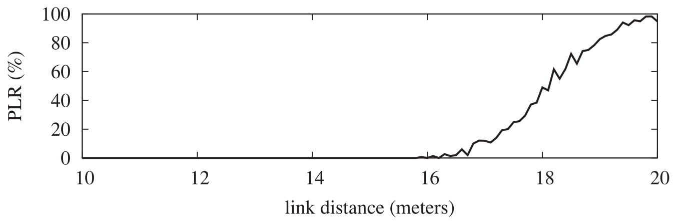


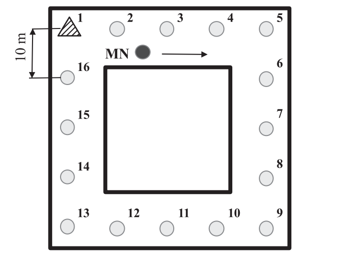
Fig. 1. Link characterisation in Cooja simulator under the MRM model.

In this section, we evaluate RL-Probe against standard implementations of legacy RPL, RPL-PP and mRPL using both simulations and experiments. Specifically, we consider a basic RPL implementation that measures the quality of links through passive monitoring of the links used by data traffic [27]. Secondly, we consider RPL-PP, included in Contiki 3.0 RPL implementation, as a solution specifically designed to handle link quality variations. Finally, mRPL is also considered as a term of comparison, to verify the efficacy of RL-Probe in handling mobility. To this aim, we ported the mRPL implementation described in [10] and available for Contiki 2.6.1 to the latest version of the Contiki OS. It is important to recall that mRPL needs to differentiate between mobile nodes and fixed nodes (called Access Points), and mobile nodes are forced to act as leaf nodes in the routing tree. For this reason, we compare mRPL with RL-Probe only in scenarios in which mobility is involved, or there are many unstable links.

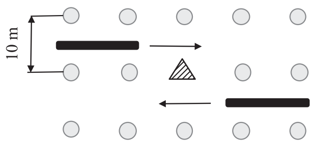
Metrics. We consider three main performance metrics in our evaluation. First, we measure the packet loss ratio (PLR) at the sink, defined as the percentage of failed packet transmissions over the total number of packets sent by a node. Secondly, we consider the packet overhead measured as the sum of the RPL control messages, including probe packets. Thirdly, we measure the normalised energy consumption per successfully received packet at the sink. We argue that this ensures a fairer comparison between scenarios affected by different PLR values than considering the total energy consumed per unit of time.

4.1. Simulation analysis

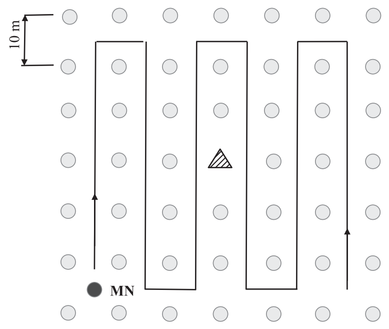
A simulation study is needed to investigate the performance in controllable and easily reproducible network conditions. Thus, we use Cooja to simulate Tmote Sky nodes. In WSNs, a radio duty cycling (RDC) mechanism is typically implemented to switch on and off the radio transceiver in order to save energy. ContikiMAC is the default RDC scheme used in our tests [53]. To model realistic radio propagation and interference we use the Multipath Ray-tracer Medium (MRM), which supports multi-path effects [54]. We have configured MRM parameters to achieve a 100% success rate at 10 m and an interference range of 20 m. Fig. 1 shows the packet loss rate as a function of node distance for a link under the MRM model.



(a) Mobile node (MN) in a corridor

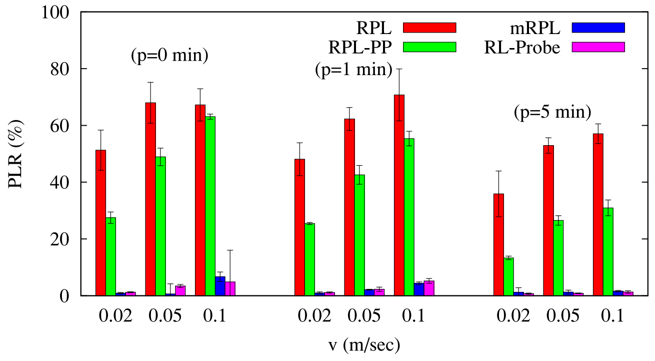


(b) Mobile obstacles

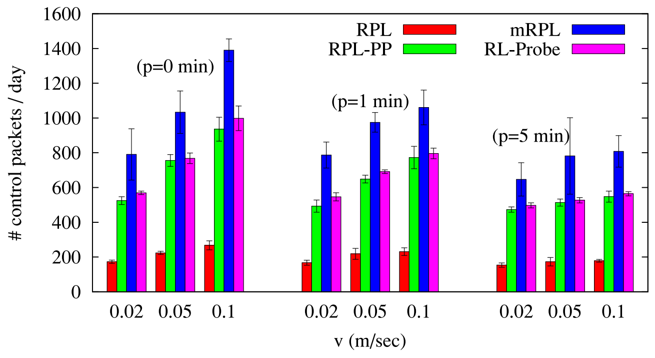


(c) Mobile node (MN) in a grid

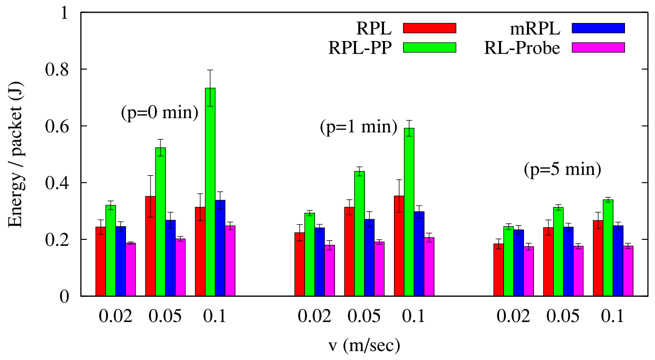
Fig. 2. Simulation scenarios (the triangle is the root node).



(a) PLR



(b) Packet overhead



(c) Normalised energy consumption

Fig. 3. Simulation analysis for the scenario shown in Fig. 2c for different speed values and pause times.

The traffic flows generate a Constant Bit-Rate (CBR) traffic consisting of small UDP messages (40 bytes) sent every one minute from all the nodes to the sink. We select a CBR traffic model as it well represents the period traffic generated by monitoring applications, which is still one the predominant use cases of wireless sensor networks. We simulate 24 h of network operations and 95% confidence intervals are computed by replicating each simulation five times with different random seeds. Confidence intervals are shown as error bars in the following diagrams.

To evaluate the proposed scheme, we consider three network scenarios. The first one is depicted in Fig. 2a, and it exemplifies a corridor monitored with fixed and mobile nodes. Specifically, 16 sensor nodes are deployed following a square layout at a distance of 10 m each. Then, a mobile node moves at a constant speed v from one corner to the following one, and every time it reaches the location of a fixed node it pauses for p min. The second scenario is depicted in Fig. 2b, and it exemplifies sensor nodes deployed in a challenged industrial environment (e.g., an assembly line) with large moving obstacles that can impair wireless communications. Specifically, we have three parallel rows of 5 sensor nodes each with one large obstacle that moves from the left to the right corners and back, and another large obstacle that moves in the opposite direction. We assume that each obstacle moves to the next node in the row and remain fixed for a time p. Furthermore, each obstacle is able to completely filter out wireless transmissions between adjacent nodes. The last scenario is depicted in Fig. 2c, and it exemplifies a dense sensor deployment. Specifically, 50 sensors are deployed in a regular grid layout and a mobile node moves at a constant speed following the trajectory shown in the figure. This last scenario is used to demonstrate the performance of the proposed solution in situations in which the mobile node may have many neighbours with high-quality links to choose while changing the preferred parent.

It is important to point out that in a network in which there are many mobile nodes, or many nodes experiencing unstable links as in the case illustrated in Fig. 2b, it might be difficult to build an optimal network topology using mRPL. Indeed, mobile nodes are forced to act only as leaf nodes and they cannot forward traffic from other nodes. Thus, mRPL could lead to less efficient, or even disconnected, network topologies if there is a large number of nodes configured as mobile/nomadic. In the network scenario depicted in Fig. 2b we have configured the four corner nodes as mobile in mRPL.

4.1.1. Results with mobile nodes

In this section, we report the results for the scenario illustrated in Fig. 2a and c.

Corridor topology. Fig. 3a shows the average packet loss rate of the mobile node for various speeds and pause times for the scenario illustrated in Fig. 2a. We recall that each time the mobile node reaches the location of a fixed node it stops for p min. We refer to this stop period as the pause time of the mobile node. Important conclusions can be drawn from these results. First, as expected packet loss rates increase when increasing speed and decrease when increasing pause times for all considered schemes. This is due to more frequent handoffs. Secondly, passive monitoring is unable to promptly cope with topology changes and packet loss rates range from 35% to 65% in the considered scenarios. Thirdly, unicast-based probing improves RPL ability to detect handoffs but packet loss rates still range from 18% to 55%. On the contrary, RL-Probe and mRPL have similar performance and they dramatically improve communication reliability, with packet loss rates that now range from 2% to 12%. An in-depth explanation of the root cause of such improvement is provided later in this section.

Fig. 3b shows the protocol overhead in terms of the total number of RPL control messages sent during 24 h. Clearly, there is a significant increase in packet overhead when active probing is used. As expected, the higher the speed and/or the shortest the pause time (i.e., the faster the network dynamics), the higher the protocol overhead. Interestingly, mRPL generates the highest protocol overhead among the considered routing schemes, while RL-Probe has similar overhead performance as RPL-PP. Analysing more in details the results we found out that the adaptive beaconing of RL-Probe reduces the number of unicast-based probing that are generated with respect to RPL-PP. However, these protocol overhead savings are compensated by the receiver-side probing, which generates trains of consecutive probes. On the other hand, handoff process in mRPL is quite aggressive as it generates long trains of multicast DIS messages to neighbouring nodes.

It may be argued that an increase in protocol overhead would severely affect the node energy consumption. To verify this conjecture, Fig. 3c shows the normalised energy consumption as estimated by the EnergyTest module provided with Cooja. We observe that RPL-PP consumes the highest amount of energy per successfully transmitted packet among the considered protocols. In highly dynamic scenarios (e.g. high speeds) RPL-PP consumes up to 100% more energy than RL-Probe. Interestingly, RPL and mRPL achieve similar normalised energy consumptions, while RL-Probe consumes up to 30% less energy than the other protocols. This counterintuitive result can be explained by observing that retransmissions have a great impact on the energy consumption. Thus, avoiding the use of lossy links can balance the additional energy consumption due to active probing.

To explain more in details the key advantages of RL-Probe, Figs. 4 show the sequence of handoff events and packet losses for the case $v = 0.02$ m/s and $p = 5$ min (the best case for both RPL and mRPL). The results indicate that standard RPL with passive link monitoring frequently changes the preferred parent, and it rarely selects the best parent. We define as the best parent the node that would be selected as preferred parent for uplink traffic if each node had a perfect knowledge of the link quality. RPL-PP is able to follow more closely the best parent. However, handoffs are mainly triggered by packet losses and round-robin periodic probing in RPL-PP causes a burst of packet losses before discovering the new optimal parent. A similar issue is also observed in mRPL, which triggers the discovery phase after a packet loss. Furthermore, the handoff delays in mRPL also cause short disconnections of the mobile node (preferred parent id equal to 0). On the contrary, *the analysis of link trends allows RL-Probe to anticipate changes of link characteristics* and to timely switch to a better preferred parent. Finally, it is interesting to note that a higher number of packet losses occurs when the mobile node is close to the sink. This can be explained by observing that an inaccurate ETX estimation of link quality to neighbours has a greater effect on the rank computation of nodes close to the sink than on nodes far from the sink.

Grid topology. Fig. 5a shows the average packet loss rate of the mobile node for various speeds and pause times for the scenario illustrated in Fig. 2c. The main noticeable difference with respect to related results shown in Fig. 3a is that PLR values are lower for the grid topology than the corridor topology. For instance, with legacy RPL packet loss rates range from 5% to 55% in the considered scenarios, and not from 18% to 65% as in the corridor topology. This is due to the fact the grid topology has a higher node density and each node has neighbours with good and intermediate quality links. Nevertheless, general trends are confirmed. First, packet losses increase as the node speed increases or the pause times decreases. Second, conventional RPL experiences packet losses that are one order of magnitude higher than the other schemes. Finally, RL-Probe performs mostly the same as mRPL in all considered cases, and they both outperform RPL-PP.

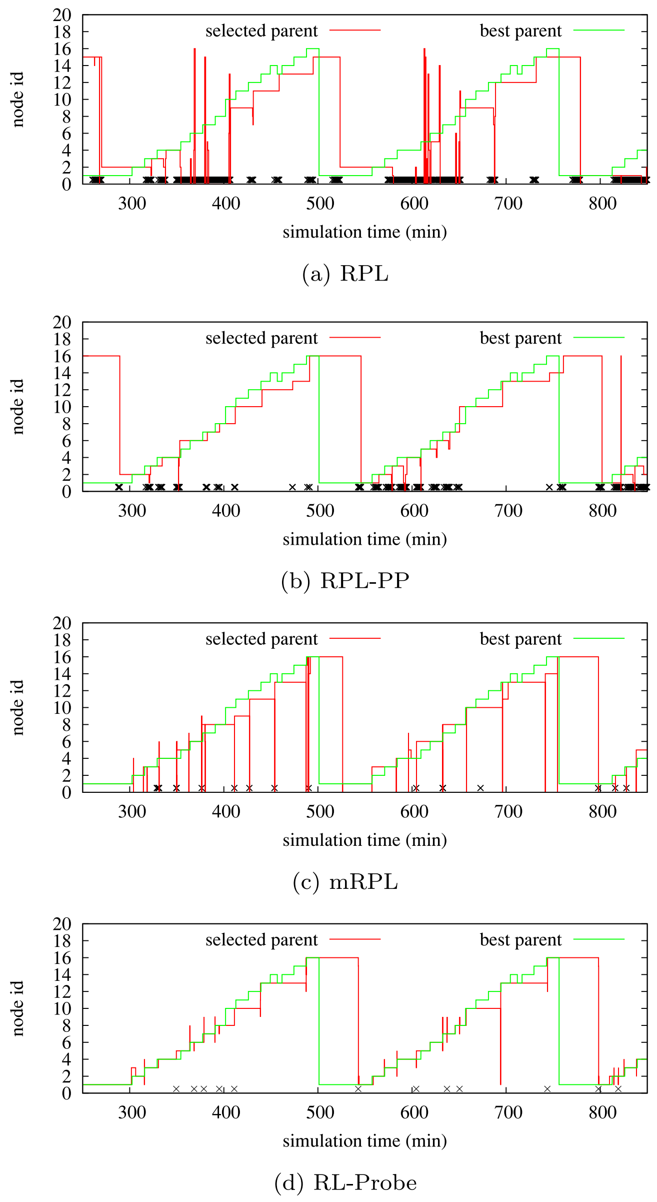
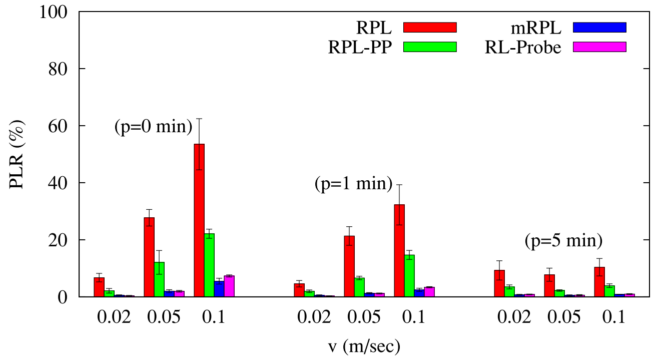
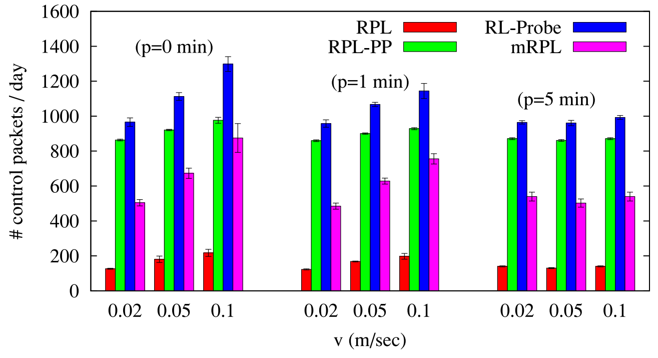


Fig. 4. Handoff events and packet losses (crosses) for MN when $p = 0.01$ m/s and $p = 5$ min.

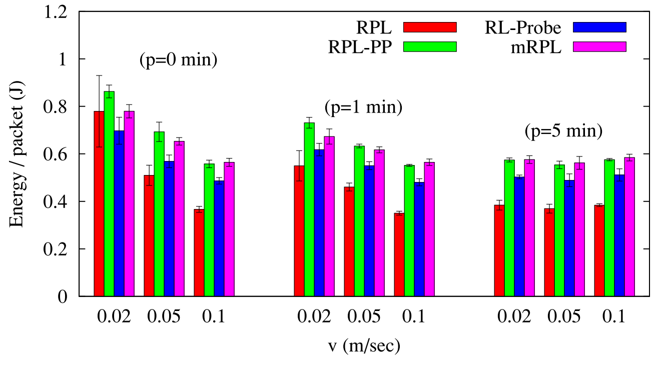
Fig. 5b shows the protocol overhead in terms of the total number of RPL control messages sent during 24 h. Conventional RPL has a packet overhead that is from three to four times lower than the other schemes that uses active probing. As expected, the higher the speed and/or the shortest the pause time (i.e., the faster the network dynamics), the higher the protocol overhead. As also shown



(a) PLR

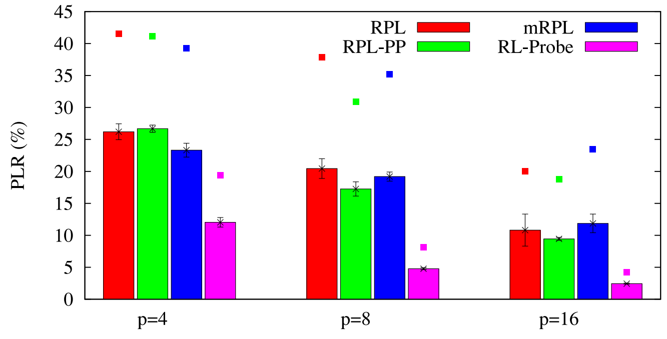


(b) Packet overhead

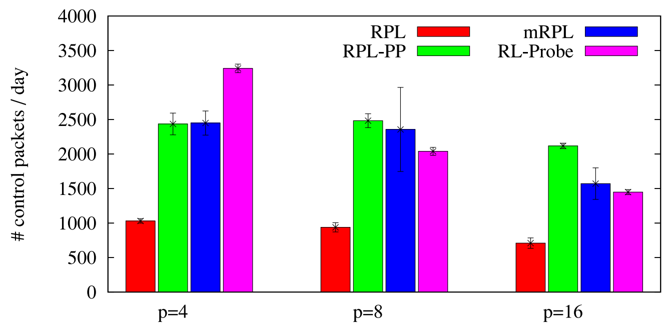


(c) Normalised energy consumption

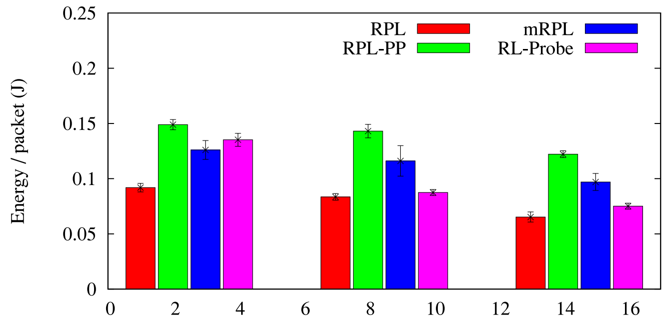
Fig. 5. Simulation analysis for the scenario shown in Fig. 2c for different speed values and pause times.



(a) PLR (bars represent the average values and filled squares the maximum value)



(b) Packet overhead



(c) Normalised energy consumption

Fig. 6. Simulation analysis for the scenario shown in Fig. 2b for different pause times.

in Fig. 3b mRPL generates the highest protocol overhead among the considered routing schemes, while RL-Probe has similar overhead performance as RPL-PP.

Fig. 5c shows the normalised energy consumption as estimated by the EnergyTest module provided with Cooja. We observe that RPL consumes the least amount of energy per successfully transmitted packet as it uses only passive techniques for link quality monitoring. RPL-PP and mRPL behaves similarly in all considered scenarios while mRPL consumes up to 20% less energy than RPL-PP and mRPL.

4.1.2. Results with mobile obstacles

In this section, we report the results for the scenario illustrated in Fig. 2b, where network topology changes are due to variations of link conditions caused by mobile obstacles and not handoffs. Fig. 6a shows the average (bars) ad maximum (squares) PLR of *all* nodes for different pause times ($p = 4, 8, 16$ min). Results indicate that RL-Probe achieves a three-fold decrease of both average and peak PLRs with respect to the other considered schemes, including mRPL. On the contrary, mRPL performs similarly to RPL and RPL-PP. Several factors contribute to mRPL inefficient behaviour. First, the hysteresis margin in mRPL assumes that the transitional region of links is quite wide [48]. However, this decreases the ability of mRPL to detect sudden changes of link quality that occur within the transitional region. Furthermore, if the quality of the link to the preferred parent is stable mRPL does not trigger discovery phases, which are needed to quickly detect if the quality of the links to neighbouring nodes is suddenly improved (e.g., because an obstacle has moved). Fig. 6b shows the protocol overhead in terms of RPL control messages. We can observe that RL-Probe rapidly limits its protocol overhead as the network conditions become less variable (i.e., pause times increase). RL-Probe generates higher protocol overhead than mRPL only for $p = 4$ min, but lower overhead for $p = 8$ and $p = 16$ min. The same trend can be observed also for the total energy consumption (see Fig. 6c). Summarising, in case of link

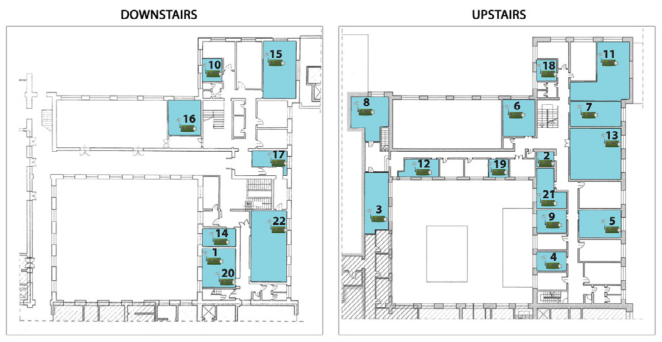


Fig. 7. Map of the testbed.

Table 2
p-th percentiles of the average ETX of the wireless links in the testbed.

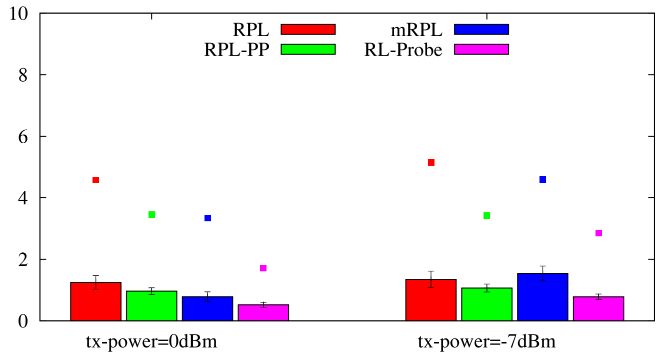
	p value				
	10%	25%	50%	75%	90%
Average ETX	1.0	1.0	1.025	1.245	2.776

quality variability due to changes in network conditions RL-Probe outperforms mRPL in terms of communication reliability with similar network overhead and energy waste.

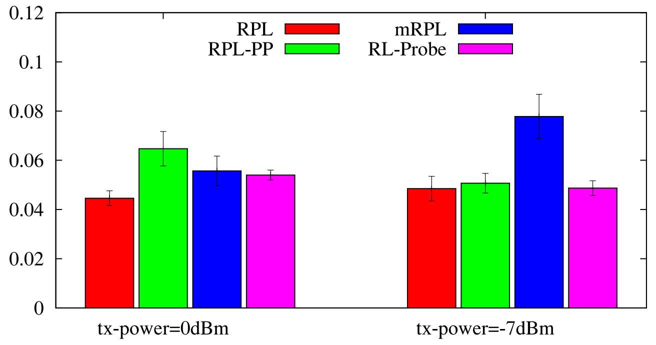
4.2. Experimental evaluation

In this section, we report the results obtained from real experiments conducted in an indoor IoT testbed [55]. Specifically, our low-power wireless network is composed of 23 wireless sensor nodes deployed in office spaces, student labs, and corridors on two floors in the Department of Information Engineering of the University of Pisa. Fig. 7 shows the layout of the testbed. Sensor nodes are TelosB motes, equipped with an MSP430 micro-controller that can run a wide range of Operating Systems for sensors. Thus, the same Contiki code used for Cooja simulations is loaded on the testbed. IEEE802.15.4 connectivity is provided through the cc2420 wireless chip equipped with an external 5dBi antenna. The maximum number of active links in the network is 178. Table 2 reports the main percentiles of the average ETX across all links, as measured by unicast-based probing without any data or control traffic in the network.

The first set of results is obtained considering a static scenario in which there are no mobile nodes. However, we emphasise that our testbed is deployed in a dynamic environment and the experiments have not been run at special times to avoid interference. Thus, our testbed is susceptible to changes in radio channel conditions due to interference (e.g., from other 802.15.4 radios and from 802.11 radios), and this interference is highly time-varying. Furthermore, we replicated each test using two radio transmission powers: 0 dBm and −7 dBm. The first value is the maximum transmission power that is supported by the CC2420 RF transceiver, which clearly maximises the network density and the number of high-quality links. The latter value is used to evaluate the performance in a configuration in which links with intermediate quality also exist. Note that farther reducing the transmission power may lead to a partitioned network topology. Finally, all nodes are configured to generate a CBR traffic consisting of 40-bytes UDP messages sent every minute to node 1. The underlying MAC protocol is CSMA and ContikiMAC is used as RDC layer. Fig. 8a shows the average packet loss ratio and energy consumption measured for the different RPL variants during experiments that last three hours. The results confirm that adding active probing techniques is beneficial to improve the routing reliability because it enables faster routing adaptation to channel fluctuations. In addition, in the analysed scenarios RPL does not experience high PLR values. Reducing the transmission power has only a small impact on the overall PLRs. Fig. 8b shows the normalised energy consumption for the different strategies. The results demonstrate that energy consumptions of RPL variants are basically equivalent.



(a) PLR (bars represent the average values and filled squares the maximum value)



(b) Normalised energy consumption

Fig. 8. Experimental analysis of the static scenario for different transmission powers.

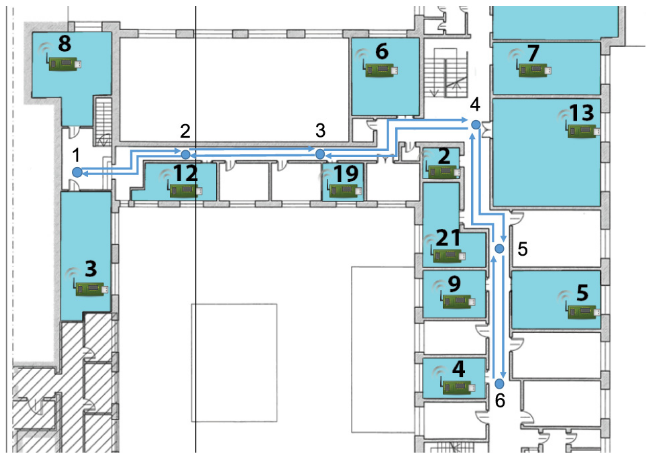


Fig. 9. Trajectory followed by the mobile node during the mobility experiments.

The second set of results is obtained in a scenario in which there is a mobile sensor node. Fig. 9 illustrates the trajectory of this mobile node. Specifically, after an initial set-up phase of 5 min, the node moves at 0.5 m/s from one specified point to another, pausing at each location for 2 min. The path is covered round-trip, i.e. from point 1 to 6 and then back from 6 to 1. Traffic is only orig-

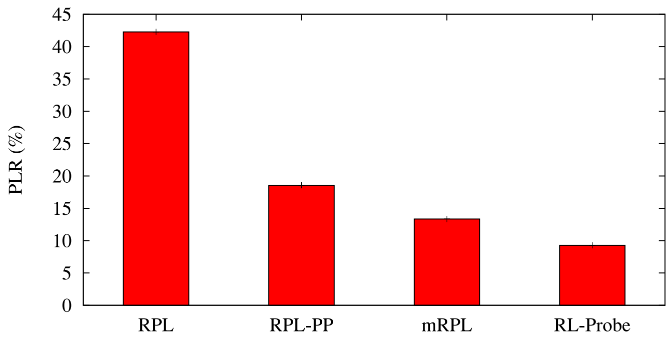


Fig. 10. Experimental analysis: average packet loss for the mobile scenario.

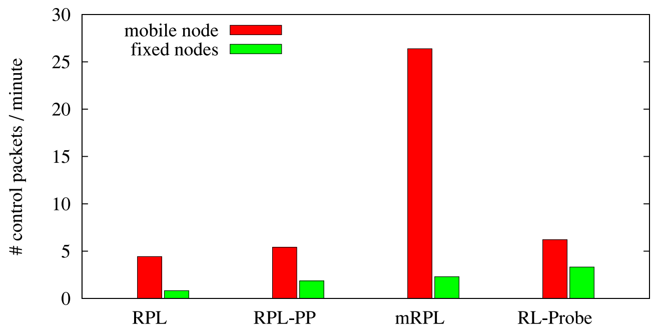


Fig. 11. Experimental analysis: control message overhead for the mobile scenario.

inated by the mobile node towards node number 1, which is selected as sink and RPL root node. A CBR traffic is employed i.e. a 40-byte UDP message is emitted every 10 s. In these tests, the radio transmission power is set to −7 dBm to use a sparser network topology. This guarantees that each movement of the mobile node results into a change of the parent node. Each experiment lasts 30 min.

Fig. 10 shows the average packet loss experienced by the mobile node with different strategies. We can observe that RL-Probe slightly outperforms mRPL, which confirms the effectiveness of multicast probing in obtaining a rapid assessment of the link quality when multiple neighbours appear/disappear at the same time. On the other hand, standard RPL experiences many packet losses every time the mobile node changes its location, due to the lack of an active strategy for LQE. RPL-PP achieves better performance than basic RPL but it is less efficient than both mRPL and RL-Probe. This can be explained by considering that unicast probing assesses links individually and therefore more time is needed to discover a better preferred parent when moving.

Fig. 11, instead, quantifies the overhead produced by each strategy. Specifically, the figure reports the average number of RPL control packets (including both probe and response packets) per second generated by the nodes in the network. We distinguish between the overhead generated by static nodes and the overhead generated by the mobile node. As expected, legacy RPL is the strategy characterised by the least overhead, as nodes only transmit RPL control packets for topology discovery without any probe packet. RPL-PP, instead, shows a slight increase in the overhead as a light unicast probe traffic is employed. Both mRPL and RL-Probe are characterised by the highest overhead due to the active probe traffic generated by each node. However, the overhead generated by the mobile node using mRPL is four times the overhead provided by the same mobile node when using RL-Probe. This clearly shows that the responsiveness of mRPL to node mobility is obtained at the cost of introducing frequent probing. On the contrary, RL-Probe does not penalise the mobile node, who requires a minimal additional overhead with respect to fixed nodes to detect link failures.

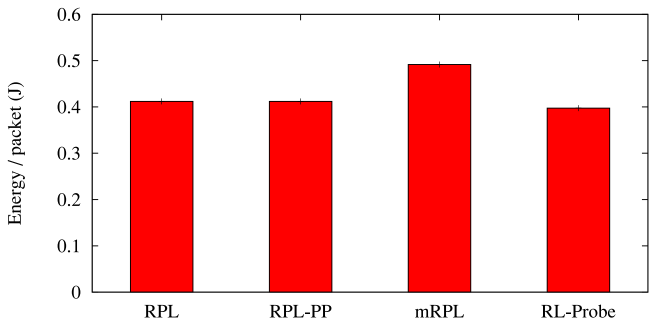


Fig. 12. Experimental analysis: normalised energy consumption for the mobile scenario.

Finally, Fig. 12 shows the normalised energy consumption for each strategy. Interestingly, we can observe that the energy consumed per successfully transmitted packet by each strategy is equivalent. This can be explained by considering that PLR for standard RPL is higher than the one of the other schemes and this compensates for the overhead increase. Since mRPL generates the highest overhead, it is also characterised by the highest normalised energy consumption.

5. Conclusion

In this paper, we have proposed RL-Probe, link quality estimation strategy for RPL-based WSNs. RL-Probe employs synchronous and asynchronous monitoring schemes to maintain up-to-date information on link quality towards the neighbours and react to sudden topology changes. RL-Probe achieves a trade-off between a low probing overhead and responsiveness to changing network conditions by leveraging on a lightweight reinforcement learning technique to control the active probing operations. This is crucial to minimise energy consumptions of tiny, resource-constrained devices. Furthermore, we have integrated our solution in the RPL implementation that is included in the Contiki operating system for embedded devices. A performance evaluation based on both simulations and real-world experiments has been carried out, demonstrating how the proposed approach guarantees better performance with respect to state-of-the-art LQE techniques for RPL. In particular, results show that the proposed approach does not only properly react to link quality variations, but it is also effective to handle topology variations due to mobility.

As future work, we plan to investigate how to improve RL-Probe performance in interference-limited scenarios in which link variations are due to external interference sources. One possible approach is to leverage opportunistic communications and to design a LQE techniques for cognitive radio [56]. Furthermore, RL-Probe can be extended to cater for more efficient learning policies than the greedy approach. Finally, RL-Probe is designed under the assumption that links are symmetric, thus exploiting eventual link asymmetry links is an open issue. A possible solution to identify the link asymmetry would be to measure the quality of each link in both directions of the link, as in [18], or to leverage cooperative approaches as in [57].

Acknowledgments

We are thankful to the editor and reviewers for their very constructive comments that helped us to significantly improve the paper quality and clarity.

This work was partially funded by the SIGS project. This project has received funding from the Regione Toscana through the PAR FAS 2007–2013 funds.

References

[1] E. Borgia, The Internet of Things vision: key features, applications and open issues, Comput. Commun. 54 (1) (2014) 1–31.
[2] E. Khorov, A. Lyakhov, A. Krotov, A. Guschin, A survey on IEEE 802.11ah: an enabling networking technology for smart cities, Comput. Commun. 58 (2015) 53–69.
[3] J.A. Stankovic, Research directions for the Internet of Things, IEEE Internet Things J. 1 (1) (2014) 3–9.
[4] M. Zhao, I. Ho, P.H.J. Chong, An energy-efficient region-based rpl routing protocol for low-power and lossy networks, IEEE Internet Things J. 3 (6) (2016) 11319–11333.
[5] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, RPL: IPv6 routing protocol for low-power and lossy networks, 2012. (IETF RFC 6550).
[6] E. Ancillotti, R. Bruno, M. Conti, RPL routing protocol in advanced metering infrastructures: an analysis of the unreliability problems, in: Proceedings of IFIP SUSTAINIT'12, 2012, pp. 1–10.
[7] I.E. Korbi, M.B. Brahim, C. Adjih, L.A. Saidane, Mobility enhanced RPL for wireless sensor networks, in: Proceedings of IEEE NOF'12, 2012, pp. 1–8.
[8] C. Cobârzan, J. Montavont, T. Noël, Analysis and performance evaluation of rpl under mobility, in: Proceedings of IEEE ISCC'14, 2014, pp. 1–6.
[9] E. Ancillotti, R. Bruno, M. Conti, Reliable data delivery with the IETF routing protocol for low-power and lossy networks, IEEE Trans. Ind. Inform. 10 (3) (2014) 1864–1877.
[10] H. Fotouhi, D. Moreira, M. Alves, mRPL: boosting mobility in the Internet of Things, Ad Hoc Netw. 26 (2015) 17–35.
[11] J. Ko, M. Chang, MoMoRo: providing mobility support for low-power wireless applications, IEEE Syst. J. 9 (2) (2015) 585–594.
[12] A. Oliveira, T. Vazão, Low-power and lossy networks under mobility: a survey, Comput. Netw. 107 (2016) 339–352.
[13] F. Gara, L.B. Saad, E.B. Hamida, B. Tourancheau, R.B. Ayed, An adaptive timer for RPL to handle mobility in wireless sensor networks, in: Proceedings of IEEE IWCMC'16, 2016, pp. 678–683.
[14] M. Zhao, P.H.J. Chong, H.C. Chan, An energy-efficient and cluster-parent based RPL with power-level refinement for low-power and lossy networks, Comput. Commun. 104 (17) (2017) 17–33.
[15] L. Tang, K.C. Wang, Y. Huang, F. Gu, Channel characterization and link quality assessment of IEEE 802.15.4-compliant radio for factory environments, IEEE Trans. Ind. Inform. 3 (2) (2007) 99–110.
[16] N. Baccour, A. Koubâa, L. Mottola, M.A. Zúñiga, H. Youssef, C.A. Boano, M. Alves, Radio link quality estimation in wireless sensor networks: a survey, ACM Trans. Sens. Netw. 8 (4) (2012) 34:1–34:33, doi:10.1145/2240116.2240123.
[17] P. Levis, T. Clausen, J. Hui, O. Gnawali, J. Ko, The Trickle algorithm, 2011. (IETF RFC 6206).
[18] K.-H. Kim, K.G. Shin, On accurate and asymmetry-aware measurement of link quality in wireless mesh networks, IEEE/ACM Trans. Networking 17 (4) (2009) 1172–1185, doi:10.1109/TNET.2008.2008001.
[19] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.
[20] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of ACM MobiCom'03, 2003, pp. 134–146.
[21] P.D. Marco, G. Athanasiou, P.-V. Mekikis, C. Fischione, MAC-aware routing metrics for the Internet of Things, Comput. Commun. 74 (2016) 77–86.
[22] M. Amadeo, O. Briante, C. Campolo, A. Molinaro, G. Ruggeri, Information-centric networking for M2M communications: design and deployment, Comput. Commun. 89 (2016) 105–116.
[23] Y. Jin, S. Gormus, P. Kulkarni, M. Sooriyabandara, Content centric routing in IoT networks and its integration in {RPL}, Comput. Commun. 89—90 (2016) 87–104.
[24] M. Barcelo, A. Correa, J.L. Vicario, A. Morell, Cooperative interaction among multiple RPL instances in wireless sensor networks, Comput. Commun. 81 (2016) 61–71.
[25] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in: Proceedings of USENIX NSDI'04, 2004.
[26] T. Narten, E. Nordmark, W. Simpson, S. H., Neighbor discovery for IP version 6 (IPv6), 2007. (IETF RFC 4861).
[27] S. Dawans, S. Duquennoy, O. Bonaventure, On link estimation in dense RPL deployments, in: Proceedings of IEEE SenseApp'12, 2012.
[28] H. Zhang, L. Sang, A. Arora, Comparison of data-driven link estimation methods in low-power wireless networks, IEEE Trans. Mob. Comput. 9 (11) (2010) 1634–1648.
[29] H. Zhang, A. Arora, P. Sinha, Link estimation and routing in sensor network backbones: beacon-based or data-driven? IEEE Trans. Mob. Comput. 8 (5) (2009) 653–667.
[30] O. Gaddour, A. Koubaa, R. Rangarajan, O. Cheikhrouhou, E. Tovar, A. Abid, Co-RPL: RPL routing for mobile low power wireless sensor networks using corona mechanism, in: Proceedings of IEEE SIES'14, 2014, pp. 200–209.
[31] K. Srinivasan, P. Dutta, A. Tavakoli, P. Levis, An empirical study of low-power wireless, ACM Trans. Sens. Netw. 6 (2) (2010) 16:1–16:49, doi:10.1145/1689239.1689246.
[32] M.A. Alsheikh, S. Lin, D. Niyato, H.P. Tan, Machine learning in wireless sensor networks: algorithms, strategies, and applications, IEEE Commun. Surv. Tutorials 16 (4) (2014) 1996–2018.
[33] R.V. Kulkarni, A. Forster, G.K. Venayagamoorthy, Computational intelligence in wireless sensor networks: a survey, IEEE Commun. Surv. Tutorials 13 (1) (2011) 68–96.
[34] T. Liu, A.E. Cerpa, Data-driven link quality prediction using link features, ACM Trans. Sens. Netw. 10 (2) (2014) 37:1–37:35.
[35] T. Liu, A.E. Cerpa, Temporal adaptive link quality prediction with online learning, ACM Trans. Sens. Netw. 10 (3) (2014) 46:1–46:41.
[36] Y. Wang, M. Martonosi, L.-S. Peh, Predicting link quality using supervised learning in wireless sensor networks, SIGMOBILE Mob. Comput. Commun. Rev. 11 (3) (2007) 71–83.
[37] D. Marinca, P. Minet, On-line learning and prediction of link quality in wireless sensor networks, in: Proceedings of IEEE GLOCOM'14, 2014, pp. 1245–1251.
[38] A. Panousopoulou, M. Azkune, P. Tsakalides, Feature selection for performance characterization in multi-hop wireless sensor networks, Ad Hoc Netw. 49 (C) (2016) 70–89.
[39] V. Toldov, L. Clavier, V. Loscrí, N. Mitton, A Thompson sampling approach to channel exploration-exploitation problem in multihop cognitive radio networks, in: Proceedings of IEEE PIMRC'16, 2016, pp. 1–6.
[40] K. Wang, L. Chen, Q. Liu, W. Wang, F. Li, One step beyond myopic probing policy: a heuristic lookahead policy for multi-channel opportunistic access, IEEE Trans. Wirel. Commun. 14 (2) (2015) 759–769.
[41] N. Baccour, A. Koubâa, H. Youssef, M. Ben Jamâa, D. do Rosário, M. Alves, L.B. Becker, F-LQE: a fuzzy link quality estimator for wireless sensor networks, in: Proceedings of EWSN'10, 2010, pp. 240–255.
[42] N. Accettura, L.A. Grieco, G. Boggia, P. Camarda, Performance analysis of the rpl routing protocol, in: Proceedings of IEEE ICM'11, 2011, pp. 767–772.
[43] F. Wu, G. Sun, G. Chen, On multipacket reception based neighbor discovery in low-duty-cycle wireless sensor networks, Comput. Commun. 75 (2016) 71–80.
[44] S. Montero, J. Gozalvez, M. Sepulcre, Neighbor discovery for industrial wireless sensor networks with mobile nodes, Comput. Commun. 111 (2017) 41–55.
[45] M. Bouaziz, A. Rachedi, A survey on mobility management protocols in wireless sensor networks based on 6LoWPAN technology, Comput. Commun. 74 (2016) 3–15.
[46] M. Barcelo, A. Correa, J.L. Vicario, A. Morell, X. Vilajosana, Addressing mobility in RPL with position assisted metrics, IEEE Sens. J. 16 (7) (2016) 2151–2161.
[47] H. Fotouhi, D. Moreira, M. Alves, P.M. Yomsi, mRPL+: a mobility management framework in RPL/6LoWPAN, Comput. Commun. 104 (2017) 34–54.
[48] H. Fotouhi, M. Alves, M.Z. Zamalloa, A. Koubâa, Reliable and fast hand-offs in low-power wireless networks, IEEE Trans. Mob. Comput. 13 (11) (2014) 2620–2633.
[49] S. Wang, A. Basalamah, S.M. Kim, G. Tan, Y. Liu, T. He, A unified metric for correlated diversity in wireless networks, IEEE Trans. Wirel. Commun. 15 (9) (2016) 6215–6227.
[50] E. Ancillotti, R. Bruno, M. Conti, E. Mingozzi, C. Vallati, Trickle-L2: lightweight link quality estimation through Trickle in RPL networks, in: Proceedings of IEEE WoWMoM'14, 2014, pp. 1–9.
[51] B.A. Nahas, S. Duquennoy, V. Iyer, T. Voigt, Low-power listening goes multi--channel, in: Proceedings of IEEE DCOSS'14, 2014, pp. 2–9.
[52] C.E. Koksal, H. Balakrishnan, Quality-aware routing metrics for time-varying wireless mesh networks, IEEE J. Sel. Areas Commun. 24 (11) (2006) 1984–1994, doi:10.1109/JSAC.2006.881637.
[53] A. Dunkels, The ContikiMAC Radio Duty Cycling Protocol, Technical Report T2011:13, SICS, 2011.
[54] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with COOJA, in: Proceedings of IEEE LCN'06, 2006, pp. 641–648.
[55] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, G. Anastasi, Interplay of link quality estimation and RPL performance: an experimental study, in: Proceedings of ACM PE-WASUN '16, 2016, pp. 83–90.
[56] P. Rawat, K.D. Singh, J.M. Bonnin, Cognitive radio for M2M and Internet of Things: a survey, Comput. Commun. 94 (1) (2016) 1–29.
[57] L. Sang, A. Arora, H. Zhang, On link asymmetry and one-way estimation in wireless sensor networks, ACM Trans. Sens. Netw. 6 (2) (2010) 12:1–12:25.