Trickle-L^2: Lightweight Link Quality Estimation through Trickle in RPL Networks

Emilio Ancillotti, Raffaele Bruno, Marco Conti
Institute for Informatics and Telematics (IIT)
Italian National Research Council (CNR)
Via G. Moruzzi 1, 56124 Pisa, ITALY
Email: {a.ancillotti,r.bruno,m.conti}@iit.cnr.it

Enzo Mingozzi, Carlo Vallati
Dipartimento di Ingegneria dell'Informazione
University of Pisa, Italy
Email: {e.mingozzi, c.vallati}@iet.unipi.it

***Abstract*—Lightweight link quality estimation is crucial in wireless sensor networks. Indeed, devices with limited capabilities shall trade off between consuming their resources to maintain a precise view of the neighbours' link quality and to build routes almost blindly. For instance, the Routing Protocol for Low-Power and Lossy Networks (RPL), which has been recently standardised by the IETF to enable IPv6-based sensor networks, only estimates the quality of the links used to deliver data packets. However, this solution has been demonstrated to cause periods of routing instability and reduced packet delivery rates since it estimates only the quality of utilised links. To address this issue in this work we propose a lightweight link estimation procedure that exploits Trickle-based topology maintenance techniques to simultaneously estimate link qualities and propagate routing information. Our proposed scheme has been integrated in the Contiki's RPL prototype implementation. Simulation results demonstrate that our proposal is capable of measuring the quality of the links to neighbours with small overhead, which results into better routing decisions and improved packet delivery rates.**

***Keywords*-RPL, LLN, Contiki OS, link quality estimation, probing, reliability, Cooja.**

I. Introduction

The idea of Wireless Sensor Networks (WSNs) as isolated groups of nodes that implement proprietary networking protocols and are incapable of communicating with the rest of the world has come to an end. The Internet of Things (IoT) vision foresees sensor networks seamlessly connected and integrated into the Internet together with a new generation of daily objects empowered with communication capabilities, referred as *smart objects* [1]. A key driver of this integration is the development of network architectures and standards that ensure a full interoperability of interconnected devices, and allow smart objects to communicate using IP, the universal communication protocol.

In the past decade it was often argued that a full IPv6 stack is unsuitable for the requirements of small embedded devices. Indeed, sensors and smart objects are usually inexpensive resource-constrained devices that are characterised by limited memory and low computational capabilities [2]. Furthermore, communications between IoT devices are usually performed through low-power wireless technologies, such as IEEE 802.15.4 [3]. On the one hand, those communication technologies guarantee limited energy consumption, which helps to maximise the lifetime of battery powered devices. On the other hand, they are typically characterised by low data rates, high packet loss rates, and instability. However, the increasing interest in connecting WSNs to existing Internet-based services and preliminary studies demonstrating the feasibility of using IPv6 in WSNs [4] has led the Internet Engineering Task Force (IETF) to create a Working Group with the goal of standardising IP-compatible protocols for the class of *Low-Power and Lossy Networks* (LLNs). One the main results of those standardisation efforts is a standard adaptation layer, called *6LoWPAN* [5], which enables IPv6 frames to be delivered on networks characterised by lossy wireless links with low data rate and short frames, such as the IEEE 802.15.4 technology. However, 6LoWPAN is only an adaptation layer, which specifies fragmentation and reassembly mechanisms and compression techniques. As far as fundamental communication aspects, such as routing, forwarding and device configuration, are concerned, another working group from IETF, called ROLL, has recently standardised the IPv6 Routing Protocol for Low Power and Lossy networks (*RPL*), [6].

RPL is designed with scalability and energy efficiency as main requirements. As a matter of fact, those features are essential in large and dense sensor networks in which the overhead necessary to spread routing information can be so high to saturate the limited bandwidth offered by the wireless medium and to drain the limited amount of power available to each device. Minimising this signalling overhead is of paramount importance to ensure the feasibility of such deployments. Among the various techniques employed in RPL to ensure routing scalability of particular relevance is the *Trickle* algorithm, which is used to control the dissemination of routing control messages within the network. Specifically, Trickle aims at minimising the amount of route updates that are broadcasted by each node by suppressing the transmission of some updates when unnecessary [7].

At the same time, link quality assessment is a critical functionality for multi-hop wireless networks. Accurate estimation of the quality of the links is a prerequisite for optimised routing. Typically, most multi-hop routing protocols for sensor networks aim at selecting network paths that minimise the overall number of (re)-transmissions needed to deliver a message to its destination in order to maximise network lifetime and network throughput. Without an accurate estimation of the link qualities, routing can only act blindly, thus selecting paths which might be far from the optimal choice. Furthermore, topology control mechanisms also need effective link quality estimation to select stable links that are not affected by excessive link quality fluctuations. In RPL, the link estimation procedures are left unspecified by the standard, i.e., different implementations can adopt different mechanisms. The classical approach adopted by most routing protocols for multi-hop wireless networks is to exploit routing signalling traffic. In other words, routing messages that are broadcasted periodically for topology construction and maintenance can also be used to monitor the quality of links and collect link measurements [8]. In RPL this approach can not be applied, since the transmission of routing messages is not periodic but irregular and unpredictable due to the use of Trickle-based transmission timers [7]. Alternative solutions already proposed in the literature involve the use of unicast probe packets that are sent over a specific link to measure its quality. Even though this approach can lead to accurate estimation, it causes an additional delay at the time of network formation, additional overhead as probe traffic and an additional complexity of the implementation [9]. In order to minimise overhead and complexity, passive link monitoring has been widely used in WSNs. In other words, existing data traffic is exploited to measure the link quality. The major advantages of this solution are its simplicity, i.e., no modifications to the original protocol are required, and small overhead. For these reasons, most RPL prototype implementations adopt such link estimation technique, and the quality of links is measured by collecting statistics on successful transmissions and failures for the links used by data traffic. However, this approach may be too conservative in the sense that it only evaluates the links that are currently being used. This provides a partial knowledge of the quality of the links to the neighbours, which is particularly critical in large WSNs since it is likely to have many poorly-connected neighbours [10]. Furthermore, the detection of alternative better forwarders is difficult because new neighbours are tested only in case of link failures. As a matter of fact, several studies have demonstrated, both using network simulators and real-world testbeds, that the interplay between data-driven link estimation and RPL topology maintenance procedures causes route instability and reduced packet delivery rates in existing RPL prototype implementations [7], [11], [12]

To address the above-described issues, in this paper we propose a lightweight link quality assessment methodology which exploits standard RPL features. The goal of our proposal is to provide an accurate link quality estimation with minimal modifications to the existing RPL implementations in order to facilitate the adoption on real devices. A salient feature of our solution is to dynamically select data-driven link monitoring and broadcast-based active link monitoring based on the status of the RPL router. Then, Trickle-based transmission timers are used to regulate the emission of broadcast probe packets during the active probing phases. Thus, probing and topology construction are performed simultaneously in order to minimise the time needed to form the topology. The proposed approach has been integrated in the Contiki's RPL prototype implementation. Simulation results demonstrate that the proposed probing mechanism significantly improves the quality of selected routes compared to the standard implementation with a small additional overhead. This leads to a noticeable improvement of network performance, especially in terms of packet delivery rates and network throughputs.

The rest of the paper is organised as follows. In Section II a description of the related work is provided. In Section III an overview of RPL and Trickle algorithms is given. Section IV illustrates our proposed scheme for link quality estimation, while Section V presents the performance evaluation of our proposal. Finally, in Section VI conclusions are drawn.

II. Related work

Radio link quality estimation is a topic widely investigated in the context of WSNs [9]. A fundamental aspect of link quality estimation is the *metric* used to measure the goodness of a link and/or path. Two types of link metrics exist: physical-layer metrics, which measure attributes directly related to the radio transceiver (e.g., RSSI or SNIR), and link-layer metrics (e.g., number of packet retransmissions). In this paper we focus on the latter type of link quality metrics. A well-known and widely used link-layer metric is the Estimated Transmission Count (ETX) metric [8], which is obtained by computing the inverse of the product of the forward delivery ratio and the backward delivery ratio of a link. Then, a key component of the link estimation technique is the strategy used to collect the statistics that are needed to compute such link metric. According to [9] three approaches exist for link monitoring in WSNs: *active*, *passive* and *hybrid*.

In *active* link monitoring approaches, nodes monitor the quality of their links by sending probe packets to neighbours. Probe packets can be encapsulated either into broadcast [8], or unicast [13] frames. Furthermore, probe packets are generally transmitted with a given periodicity. These approaches are demonstrated to produce accurate link quality measurements if probes are transmitted with sufficiently high frequency. However, the overhead associated to unicast probing may be excessive especially in WSNs. Thus, most network protocols in WSNs use broadcast probes, which incur a smaller overhead compared to unicast probing. Another advantage of broadcast probing is the possibility of exploiting routing signalling traffic, which is already needed to implement most routing functionalities. An example is the work presented in [8] in which the authors exploit the periodic emission of hello messages for neighbour discovery to update the estimation of the metric for each link.

Passive link monitoring, instead, exploits existing traffic without additional probing overhead. To this end, a node can use its own data traffic, i.e., the packets it transmits, or can overhear transmissions not addressed to it to evaluate the quality of the links to its neighbours. An example is provided in [14] in which the authors propose to overhear both data and acknowledgement messages from different neighbours to improve the accuracy of the estimation. This methodology has been widely adopted in WSNs because it is energy efficient since it does not involve the emission of additional packets. On the other hand [15] showed that overhearing consumes significant energy at the receivers, which can cancel out the energy saving due to no probing. A further main drawback of passive approaches is that they depend on the presence of data traffic. This can lead to measurement inaccuracies depending on the intensity of the data traffic and its spatial distribution.

Finally *hybrid* mechanisms combine both active and passive approaches in order to reach a convenient balance between accurate link measurements and energy efficiency. For instance, authors in [13] propose passive, cooperative, and active monitoring techniques, which are dynamically adopted by nodes over time based on the amount of egress/cross traffic.

Even though it is a well known issue, link estimation in RPL networks has not been deeply investigated. In [10] the authors propose a passive probing mechanism to estimate link quality in dense RPL networks. In the initial phase when the link quality is unknown, a node assigns optimistic estimation for the quality of the links to the newly discovered neighbours in order to rotate the selection of the preferred parent between all the nodes. In a previous work [16], a unicast-based probing technique is proposed in order to improve network stability. More precisely when a new neighbour is discovered a node triggers a link probing phase in which the quality towards each neighbour is estimated by means of a set of probe messages. However, all those enhancements require profound changes to the RPL standard.

III. RPL Background

RPL is a routing protocol specifically designed for lossy environments and resource-constrained embedded devices. The protocol takes into account the unreliable nature of the communications and the limited available power of the devices by minimising the memory requirements and the complexity of the routing functionalities and reducing the routing signalling overhead. Typically, most applications for LLNs generate periodic traffic that is destined to one *root* node, which acts as border router for data collection. On the contrary, traffic originated from the root node is assumed sporadic, while node-to-node communication is rare. RPL design reflects those assumptions on common traffic patterns. Specifically, RPL is a mixed proactive-reactive algorithm which pro-actively construct upward routes for nodes-to-root traffic through the periodic emission of control messages, while it handles root-to-nodes traffic through on-demand downward route formation. Node-to-node traffic is not directly supported and can be delivered only indirectly: the source node sends the message to the root which reroutes the message downward to the destination.

As fas as network path selection is concerned, 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 initialise 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. The rank must monotonically decrease on each path identified by the DODAG towards its root. Such property is used to avoid loops in the logical topology.

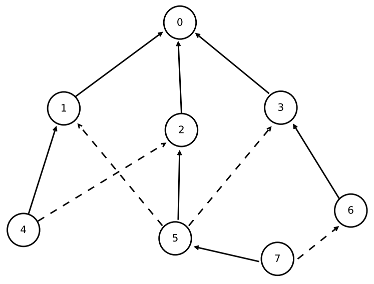

Fig. 1. DODAG example

As DIO messages are received from the neighbours, each node updates its view of the topology. In particular, a set of neighbours with lower rank is 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. Figure 1 shows an example of one DODAG. Each node selects a parent set (dashed and solid lines); among them a preferred parent is selected for upward traffic forwarding (solid line).

The rank is computed according to an *Objective Function* (*OF* for short). 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., packet delay or number of hops. Several OFs have been defined, such as the OF Zero [17], and the *Minimum Rank with Hysteresis Objective Function* (*MRHOF*) [18], which aims at reducing route flaps caused by small metric fluctuations. Note that RPL strictly defines the general rules that any OF should adhere to when updating the node rank value or selecting the parent set. Specifically, in order to avoid routing loops a node can not take a rank greater than its current value and it is not allowed to select as parent a node with a greater rank. As a DIO message with a smaller rank is received, the node updates its rank and adds the sender to the parent set. In this case all the nodes in the parent set whose rank is greater than the new rank are removed. Finally, RPL expects an external mechanism to be triggered during the parent selection phase in order to verify link properties and neighbour reachability.

A. The Trickle algorithm

The *Trickle* [19] algorithm, originally designed for polite gossiping in multi-hop wireless networks, has been adopted by RPL to regulate DIO broadcast transmissions so as to reduce nodes' energy consumption.

The rationale behind Trickle is as follows. A DIO broadcast transmission is scheduled by each node on a periodic basis.

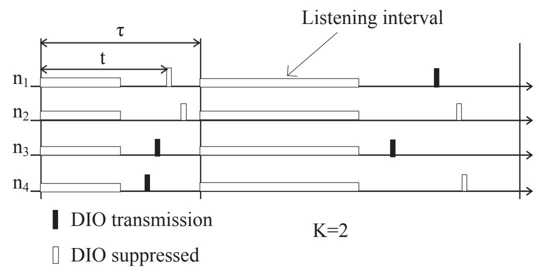

Fig. 2. Example of Trickle operation with four nodes, $k = 2$.

However, a node can suppress the transmission in a given period if enough DIO messages have been already overheard in the recent past from its neighbours. Moreover, the period is dynamically adapted depending on the 'steadiness' of the information carried by received DIO messages: if the information is deemed "consistent" with the current route configuration, the period duration is exponentially increased. Otherwise, if it is not consistent, the period is reset to the minimum value to boost spreading of the new information. In this manner, promptness in routing updates is preserved while the number of DIO messages transmitted overall is highly reduced, which results in significant energy saving.

More in details, Trickle algorithm operates in each node as follows. Each transmission period is halved into two subsequent time intervals: the *listening* interval and the *transmitting* interval, respectively. At the beginning of the period, a DIO message transmission is scheduled at a random time t in the *transmitting* interval. Up to t, the node keeps track of received messages by incrementing a *redundancy counter* c each time a new DIO message is received. At time t, if c is below a *redundancy threshold* k, the node actually transmits the DIO message, otherwise the transmission is suppressed. Finally, let τ denote the length of the current transmission period. At the end of the period, if only consistent messages were received, τ is doubled, i.e., the length of the next period is doubled with respect to the current one, until a maximum value I_{max} is reached. Otherwise, if an inconsistent message was received, τ is reset to a minimum value I_{min}. In any case, at the end of the interval, c is reset. Figure 2 illustrates an example of Trickle operation with DIO transmission suppression in the case of four nodes and $k = 2$.

IV. Trickle-based Link-Quality Estimation

According to the RPL specification, every non-root node at the time of network *bootstrap* listens for DIO messages. When at least one DIO is received, the node *immediately joins* the DODAG: first it selects one node as preferred parent among its neighbours and evaluates its rank, then starts broadcasting its own DIOs to advertise its presence. At the time of the reception of the first DIO, the quality of the link is still unknown because without using a probing mechanism, a node can measure the quality of a link only when data traffic is delivered over that link. For this reason when the first DIO is received, the node is unaware of the quality of the link to the neighbour that had sent that DIO message and it selects the preferred parent almost blindly by initialising the quality of the link to an arbitrary value[1].

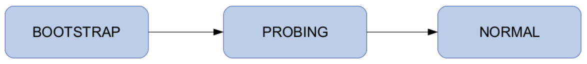

Fig. 3. Timeline of node operations.

After joining the DODAG, a node can perform a more accurate estimation of the link towards its preferred parent by collecting statistics for the data packets flowing over that link. However, until an accurate estimation is obtained, the network remains in a transitory phase in which the topology is unstable as nodes change frequently their preferred parents. This issue highlights the need for a more accurate link measurement at the time of DODAG formation in order to improve the ability of quickly constructing a stable and optimal network topology.

Our proposal overcomes this issue introducing a *probing* phase that allows nodes to collect an accurate link quality estimation *before* joining the DODAG. Specifically, each node in turn executes its own probing phase by sending broadcast probe messages, while nodes within the transmission range of the broadcasting node exploit these messages to estimate the quality of the link with that node. Figure 3 illustrates the timeline of the phases performed by each node: $i)$ *bootstrap*, the node collects information on its neighbours, $ii)$ *probing*, the node joins the DODAG and emits probe messages to allow its neighbours to measure the quality of the links, and $iii)$ *normal*, the node has joined the DODAG and executes regular operations as described in the standard. In the following we describe in detail the tasks performed by a node in the different phases of the link estimation procedure.

As defined in the RPL standard, a node starts from the bootstrap status, in which it listens for DIOs. The criteria which drive a node to leave the bootstrap status to join the DODAG is left unspecified, and the RPL specification requires only to discover at least one neighbour. In our implementation we introduce a timer to periodically check the status of a set of conditions, called *join conditions*, and only when at least one of those conditions is satisfied the node can leave the bootstrap phase to transit to the probing phase. More precisely, at the reception of the first DIO a timer called *join timer* is set as follows:

$$t_{join} = \tau + \epsilon \,, \tag{1}$$

where τ is the period on which a node must check the join conditions, while ϵ is a uniform random variable, which represents a random back-off introduced to avoid nodes within the same neighbourhood to start executing the procedure simultaneously, hence leading to the so-called broadcast storm problem[2]. The parameter τ can be tuned to achieve a trade-off between the

[1]For instance, Contiki's RPL implementation adopts pessimistic link estimation. Specifically, it assumes that any unknown link is bad and it initialises the ETX metric for that link to 6, which is highest possible value.

[2]In our tests we have selected ϵ in the range $[0; \frac{\tau}{10}]$.

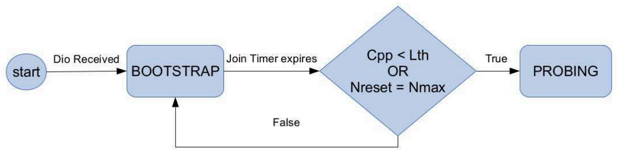

Fig. 4. Node operations during bootstrap.

delay for topology construction, the completeness of collected information and the energy consumption due to link estimation. Note that when the timer expires, if none of the join conditions are satisfied the node extends the bootstrap phase by resetting the t_{join} timer.

The join conditions defined in our implementation are the following: *minimum link quality* and *maximum number of timer resets*. The first condition requires that the links discovered during the bootstrap phase have a minimum quality, i.e. the link quality through the candidate preferred parent, say L_{cpp}, has to be greater than the *minimum link quality* threshold, say L_{th}. The second condition introduces a maximum number of times N_{max} the join timer can be reset. Specifically, when the number of resets N_{reset} is equal to N_{max}, the node leaves the bootstrap phase regardless of the quality of the discovered links. This condition is necessary to guarantee a maximum sojourn time in the bootstrap phase, which cannot be longer than $(N_{max} + 1) \cdot (\tau + \frac{\epsilon}{2})$. The parameters τ and N_{max} can be tuned jointly to represent the maximum time a node is willing to delay its operational status to obtain better routes. The operations performed by a node during the bootstrap phase are summarised in Figure 4.

When the node leaves the bootstrap phase, it first joins the DODAG by selecting a preferred parent and computing its rank value according to the adopted OF. Then, the node starts the active probing phase in which it broadcasts probe messages. More precisely, during the active probing phase a node broadcasts a set of N_p probe messages, with the objective of allowing the neighbours to evaluate the link quality. When the train of broadcast messages is completed the node can enter in the normal status in which the normal RPL operations are performed.

In order to ensure that our link estimation technique can be easily adopted in the RPL standard a slightly modified version of DIO messages (called *probe DIOs*) is adopted as broadcast probe. Specifically, probe DIOs have the same structure as regular DIOs except for the following modifications:

- *Probe flag*: added to the flag field of the message, if set indicates that the DIO belongs to a set of probe DIOs
- *Probe sequence number*: the sequence number of the message within the set of probe DIOs broadcast by one node. This field is added within the ”Reserved” field of the message. The sequence number is used to compute the number of probe DIOs that are lost.

Finally, it is worth to highlight that the probe DIOs are fully backward compatible with the RPL standard, i.e., a RPL router not implementing probing functionalities can process the DIO message gathering routing information, and discarding probe information.

The emission of probe DIOs is regulated through the Trickle algorithm, the same algorithm adopted by RPL to schedule the emission of DIO messages during normal operations. The adoption of Trickle timers to also regulate the emission of probe DIOs has the following benefits: the probe procedure can be easy implemented since it does not require extra complexity, the Trickle suppression policy can be exploited to reduce the collision probability of probe messages within each neighbourhood, so that nodes running regular RPL operations and nodes broadcasting probe messages can coexist at the same time.

Regardless of its current state, each node processes in background DIO messages to collect information on the topology. If the probe flag is set, also the sequence number is recorded. When the last probe DIO from one node is received, the node evaluates the number of lost messages and consequently measures the link quality in terms of Packet Reception Rate (PRR). In order to handle the case when the last probe message is lost, the node set a timer T_{probe} which is reset every time a probe DIO is received. When a link quality estimation is completed the cost of the neighbour is updated. If necessary a new preferred parent is selected.

V. Performance Evaluation

In this section we present the results of the performance comparison between the legacy Contiki's RPL implementation and our implementation with the proposed Trickle-based link-quality estimation proedure. First, we quickly describe the simulation set-up and the evaluation methodology, then we show the obtained results.

A. Simulation scenario and settings

The proposed approach has been implemented in Contiki, an open source operating system designed for constrained devices [20]. Contiki includes implementations of the 6LoWPAN header compression and of the RPL protocol, called *ContikiRPL*, which implements all the fundamental mechanisms described in the standard. One of the benefits of using Contiki is that the code can be run in an emulator called Cooja which is available as part of the Contiki distribution [21]. We used Cooja to emulate a Tmote Sky platform, an MSP430-based board with an IEEE 802.15.4-compatible CC2420 radio chip. To simulate a realistic interference we adopted the Multipath Ray-tracer Medium (MRM) model, a propagation model that implements ray-tracing techniques with various propagation effects, e.g. multi-path, refraction, diffraction, etc. The basic CSMA/CA scheme shipped with the Contiki kernel is adopted as MAC layer.

The objective function adopted to evaluate the rank of nodes is the *Objective Function Zero* (OF0), [17], in which we use the ETX metric to compute the *step_of_rank* parameter. In order to improve stability and avoid unnecessary parent switches in response to metric changes, a simple hysteresis mechanism is added to the implementation of OF0, in line with what recently proposed in [22]. In details, a new parent is selected as preferred parent only if its minimum cost path is shorter than the current by at least 0.5 (in terms of ETX).

In our simulations we considered a network with a single sink node placed at the center of a squared area with a side of 600 meters. A fixed number of 100 nodes are placed randomly following a uniform distribution over the simulated area. The transmit power of each node is set to the minimum value that ensures successful transmission within a distance of 80 meters. A low intensity traffic is simulated through a Constant Bit Rate data source installed on each node. Each source generates one 30-byte long message every minute. Trickle parameters are summarised in Table I. As can be seen, two different sets of parameters are used for the trickle algorithm: a default setting for normal operations and a modified one for probing, which corresponds to the generation of one probe DIO every 64 milliseconds. While the investigation of the network dynamics subject to the variations of the trickle parameters is out of scope in this paper, trickle setting during the probing phase can affects the performance of the proposed solution. For this reason the selection of the trickle parameters for probing has been performed after a preliminary analysis, not included here for the sake of brevity.

In order to asses the performance of the network we define the following metrics:

- *Bottleneck link stretch*, the ratio between the ETX of the bottleneck link of a path over the ETX of the bottleneck link of the optimal path (i.e., the path with minimum overall ETX),
- *Join time*, the time instant (in minutes) when a node joins the DODAG,
- *Overall ETX path stretch*, the ratio between the overall ETX of a path over the overall ETX of the optimal path,
- *Configuration time to ETX=c*, in minutes, the time needed by RPL to obtain a topology in which every node has a preferred parent with link quality lower or equal to c,
- *Control and Data traffic overhead*, number of (normal and probe) DIO and data messages sent by a node per minute,
- *Average packet loss percentage*, percentage of the application packet loss

All results presented are averaged over five independent simulations runs. Each experiment simulates two hours of network operations. Statistics can be collected at the end of the simulation time or at the time of the network join, in other words when the last node of the network joins the DODAG. Metrics are shown in boxplot graphs in order to show not only the average value but also its distribution[3].

B. Simulation results

In this first set of simulations we evaluate the performance of the proposed solution varying N_p, the number of probe DIOs that are sent by each node during its probing phase.

[3]The bottom and top of the box are the 25th and 75th percentile, the band in the box is the median (50th percentile), the ends of the whiskers represent the minimum and 95th percentile, the square and the triangle represent the maximum and average, respectively.

TABLE I
RPL PARAMETERS

Parameter	Value
I_{min}	8ms
$I_{min\_probing}$	64ms
I_{max}	2.3hours
$I_{max\_probing}$	64ms
K	10
$K_{probing}$	1

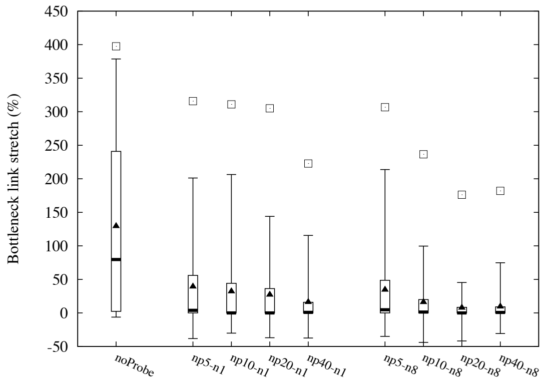

Fig. 5. Bottleneck link stretch at the time when a node joins the DODAG.

Each simulation scenario is repeated with four values of N_p (5, 10, 20 and 40), and two values of N_{max} (1 and 8). As far the L_{th} and τ parameters we set them to 1.5 and 1 second respectively. The results obtained with our implementation are compared with the results obtained without probing, which is the approach adopted by Contiki in its original implementation.

Figure 5 illustrates the results of the bottleneck link stretch at the time when a node joins the DODAG. As can be seen, our proposal significantly outperforms the original implementation, which selects paths with bottleneck links significantly higher than the optimal value. This gap can be explained considering the fact that the original implementation performs the path selection blindly only considering the neighbour from which has received the first DIO message. As a consequence, the initial selection of the preferred parent is performed almost randomly as demonstrated by the large variance of the results which vary in the range between 0% (the path selected is the optimal) and 400% (the path selected has a bottleneck with a quality four times higher than the bottleneck of the optimal path). If we look at the effect of the number N_p of probe messages on system performance, we can observe that this parameter noticeably impact the bottleneck performance. As expected, the higher the number of probe messages sent by a node, the lower the bottleneck link stretch. This can be explained by considering the influence of N_p on the accuracy of the measurements: low values of N_p result in short probe periods which are more subjects to channel fluctuations. On the other hand, when large values of N_p are adopted each node can collect more samples of the link quality, which results in more robust estimation against channel quality variation or collision of a single packet. The results also show that larger

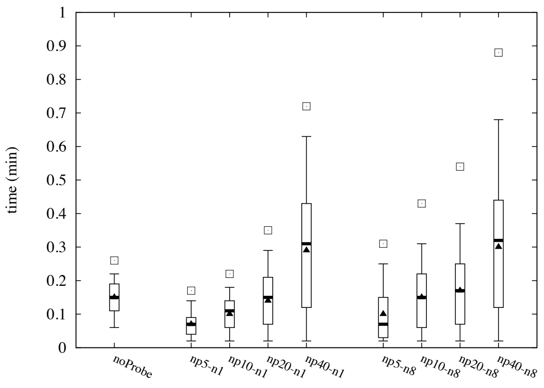

Fig. 6. Average join time in a network of 100 nodes.

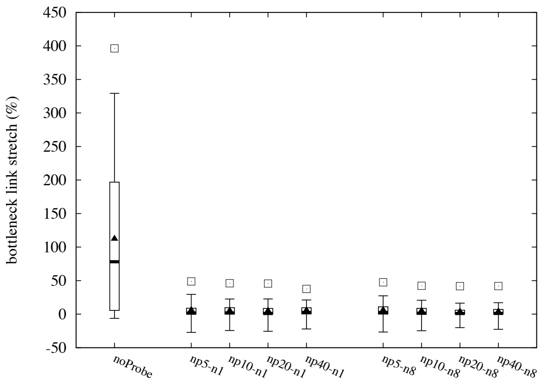

Fig. 7. Bottleneck link stretch averaged over the entire simulation time (two hours).

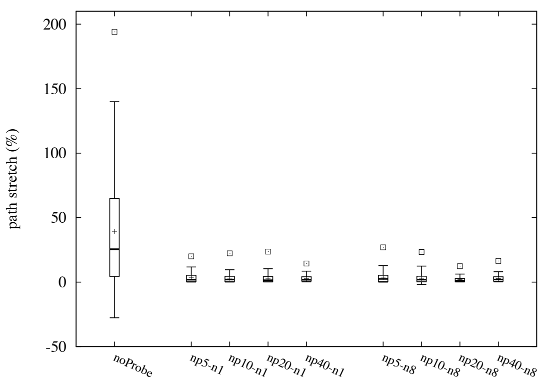

Fig. 8. Overall ETX path stretch averaged over the entire simulation time (two hours).

N_{max} values delay the bootstrap period in which each node waits for neighbours with a link quality less or equal than L_{th}, resulting in a better quality at the joining time.

The results of the average join time are shown in Figure 6. As can be seen the time needed by our solution to generate a complete DODAG topology is comparable with the time needed by the standard implementation. These results show that the additional time spent by each node in the probing phase is negligible compared with the time needed to propagate the routing information throughout the network. If we consider different N_p values we can notice that the join time increases with N_p as expected because the higher the number of probe DIOs the longer the duration of the probing phase. Finally, it is worth pointing out that in some scenarios, e.g., N_p equal to 5 or 10, the join delay of our proposal is even lower than the join time obtained with the standard implementation. This can be explained by considering that specific setting of the Trickle parameters during the probing phase, in particular $I_{max} = I_{min}$. In other words, during the probing phase the Trickle interval I is fixed, while in the standard implementation it is always doubled, as regular Trickle operations require. For this reason the probing phase facilitates the diffusion of the routing information which is delivered along with probe DIOs. Our results also show that the N_{max} parameter slightly influences the network formation time, since higher values of N_{max} may result in an increase of the number of times the join timer expires.

Figure 7 shows the bottleneck link stretch averaged over the whole simulation. When comparing these results with the ones shown in Figure 5 the performance improvement is remarkable. Specifically, our RPL implementation is capable of improving the quality of the routes over time, and it converges quite quickly to almost optimal routes. This can be explained by observing that when new neighbours with better rank values are discovered a node changes its preferred parent, consequently improving the quality of the bottleneck link. On the contrary, ContikiRPL is pretty slow in converging towards optimal routes and even after running the network for two hours the bottleneck link stretch is still high. This is a consequence of the fact that using data-driven link estimation techniques a node can evaluate the quality of only one link at a time, the link towards the preferred parent, while the quality of links towards other neighbours remains undiscovered until there is a parent switch.

The same behaviours highlighted in Figure 7 are also confirmed in Figure 8 which illustrates the overall ETX path stretch. In other words, the total ETX along the selected paths is close to that of the optimal paths. It is also evident that having a good topology from the beginning helps in discovering better routes, which will positively influence the performance all over the network lifetime. To further confirm the inefficiency of the legacy RPL implementation in finding neighbours with good link quality and its slowness in converging towards optimal paths, in Figure 9 we show the time needed by RPL to obtain a topology in which every node has a preferred parent with link quality lower or equal to 2. When considering the maximum value of this metric we can notice that our solution provides one order of magnitude improvement with respect to ContikiRPL. This is mainly due to the fact that our proposal reaches the goal at the time of the network join, since on almost all the cases neighbours with good links are

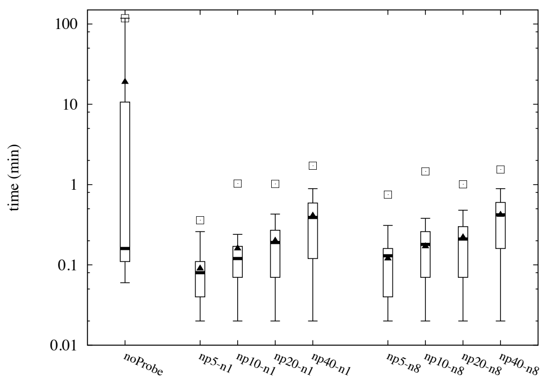

Fig. 9. Configuration time to ETX=2.

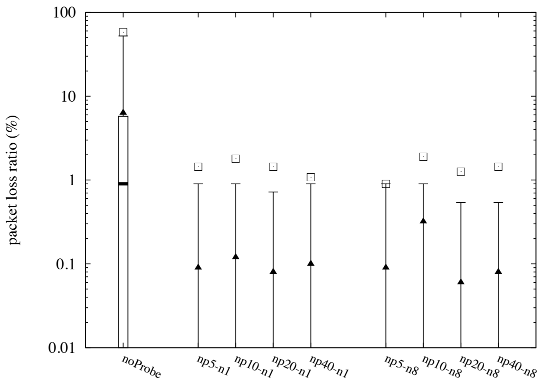

Fig. 12. Average data loss rate per node.

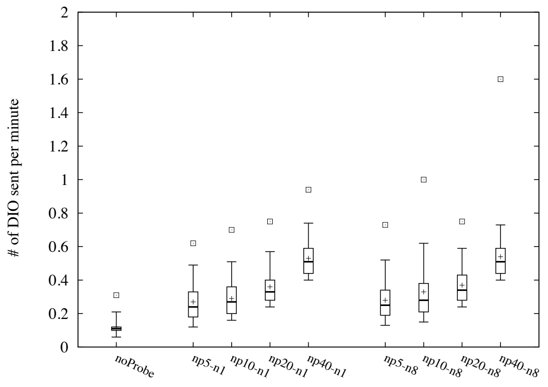

Fig. 10. Average number of DIO messages that are sent by a node every minute.

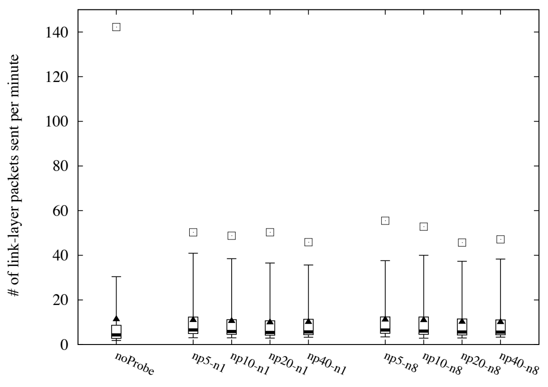

Fig. 11. Average number of packets (measured at the link-layer) that are sent by a node every minute.

discovered during the probing phase.

It is intuitive to observe that one of the drawbacks of our solution is that additional overhead is generated by the active probing. To quantify these additional overheads, in Figure 10 we show the average number of DIO messages (both normal and probe DIOs) sent by each node every minute. As expected, a more accurate estimation of the quality of the links to neighbours comes at the cost of a higher number of DIO messages sent by a node. However, Figure 11 shows that if focus on to the total number of packets transmitted by the wireless interface, the cost of probing messages is mitigated by the lower number of retransmission of unicast packets.

It is essential to note that the benefits of having better routes clearly impact on the quality of data forwarding. Figure 12 shows the distribution of the average packet loss rates over all nodes. It is important to consider the distribution because it makes easier to observe that there is a non negligible fraction of RPL routers that may suffer high packet loss rates, mainly due to the selection of a poorly connected neighbour. Again the performance gain is up to two orders of magnitude for our RPL implementation.

We conclude this section by analysing the influence of the remaining parameters on the system performance. More precisely, in this second set of simulations we will vary N_{max} and L_{th} (N_{max} is set to 0, 1, 4 and 8 while L_{th} to 1.5 and 4, respectively), while N_p is set to 20 and τ to 1 second. Figure 13 illustrates the join time versus different scenarios resulting in different configurations of N_{max} and L_{th}. When a stringent value of L_{th} is adopted, we can notice that the join time is influenced by the value of N_{max}: the higher N_{max}, the longer the join time, as higher N_{max} values allow nodes to extend more freely the length of the bootstrap phase. On the other hand, if large values of L_{th} are considered, we can notice that the join time is no longer influenced by the value of N_{max}, as no particular trend is observed. This can be explained by considering the fact that a large L_{th} forces a node to stop the bootstrap phase as soon as a neighbour is discovered regardless of the link quality. Finally, Figure 14 shows that N_{max} and L_{th} parameters slightly affect the quality of paths towards the sink node.

VI. Conclusion

In this paper we have proposed a lightweight mechanism to estimate the quality of links in WSNs adopting RPL as routing protocol. The proposed approach exploits the Trickle algorithm, which is originally proposed in RPL to regulate the flooding of routing signalling messages, to also control

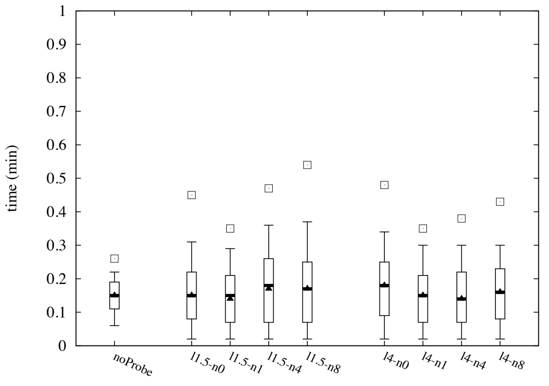

Fig. 13. Impact of N_{max} and L_{th} parameters on the average join time.

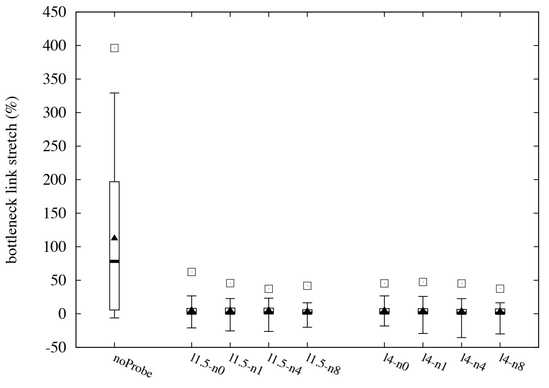

Fig. 14. Bottleneck link stretch.

the transmission of broadcast probe messages at the time of topology setup. In order to ensure backward compatibility of our proposed link estimation technique such probe packets are based on regular DIO messages. This also allows to simultaneously assess the link quality while disseminating routing information.

Results obtained using Cooja emulator indicate that our link estimation technique is capable of measuring link qualities with a decent accuracy and small additional overheads, and without introducing a significant delay in the time needed by the network to form the topology. Furthermore, our proposed scheme allows RPL to quickly converge towards optimal routes, which significantly improves the stability and reliability of the data forwarding process. In future works, we plan to investigate the impact of duty-cycling mechanisms on the accuracy of the probing. In addition, we will investigate how to extend the proposed techniques to operate in dynamic scenarios (e.g., nodes joint the network, or running out of batteries, etc.). It would also be interesting to validate the efficiency of the proposed scheme in real-world testbeds. Finally, a comparison with alternative measurements techniques will also be carried out as part of future work.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.05.010

[2] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the Internet of things," *IEEE Internet Computing*, vol. 14, no. 1, pp. 44–51, 2010.

[3] IEEE, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), pp. 1–314, 2011.

[4] J. Hui and D. Culler, "IP is dead, long live IP for wireless sensor networks," in *Proc. of ACM SenSys'08*, 2008, pp. 15–28.

[5] N. Kushalnagar, G. Montenegro, and Culle, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," Tech. Rep. 4944, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4944.txt

[6] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[7] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," in *Proc. of IEEE WiMob'11*, 2011, pp. 365–372.

[8] D. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-throughput Path Metric for Multi-hop Wireless Routing," in *Proc. of ACM MobiCom '03*, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 134–146.

[9] N. Baccour, A. Koubaa, L. Mottola, M. Zuniga, H. Youssef, C. Boano, and M. Alves, "Radio Link Quality Estimation in Wireless Sensor Networks: a Survey," *ACM Transactions on Sensor Networks*, vol. 8, no. 4, November 2012.

[10] S. Dawans, S. Duquennoy, and O. Bonaventure, "On Link Estimation in Dense RPL Deployments," in *Proc. of IEEE SenseApp'12*, 2012.

[11] E. Ancillotti, R. Bruno, and M. Conti, "RPL Routing Protocol in Advanced Metering Infrastructures: an Analysis of the Unreliability Problems," in *IFIP SUSTAINIT'12*, 2012.

[12] K. Heurtefeux and H. Menouar, "Experimental Evaluation of a Routing Protocol for Wireless Sensor Networks: RPL under study," in *Proc. of IFIP WMNC'13*, 2013.

[13] K.-H. Kim and K. G. Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *Proc. of ACM MobiCom'06*, 2006, pp. 38–49.

[14] P. Jiang, Q. Huang, J. Wang, X. Dai, and R. Lin, "Research on Wireless Sensor Networks Routing Protocol for Wetland Water Environment Monitoring," in *Proc. of ICICIC '06*, vol. 3, 2006, pp. 251–254.

[15] D. Lai, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian, "Measurement and characterization of link quality metrics in energy constrained wireless sensor networks," in *Proc. of IEEE GLOBECOM '03*, vol. 1, 2003, pp. 446–452.

[16] E. Ancillotti, R. Bruno, and M. Conti, "On the interplay between RPL and address autoconfiguration protocols in LLNs," in *Proc. of IEEE IWCMC'13*, 2013, pp. 1275–1282.

[17] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552, 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6552.txt

[18] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," RFC 6719(Proposed Standard), Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6719.txt

[19] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, "The Trickle Algorithm," RFC 6206 (Proposed Standard), 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6206.txt

[20] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, 2004, pp. 455–462.

[21] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proc. of IEEE LCN'06*, 2006, pp. 641–648.

[22] O. Gnawali, "The minimum rank with hysteresis objective function," 2012.