Efficient Distributed Monitoring in 6LoWPAN Networks

Abdelkader Lahmadi, Alexandre Boeglin, Olivier Festor

Efficient Distributed Monitoring in 6LoWPAN Networks

Abdelkader Lahmadi
Université de Lorraine, LORIA
Vandoeuvre-lès-Nancy, F-54506, France
Email: Abdelkader.Lahmadi@loria.fr

Alexandre Boeglin, Olivier Festor
Inria
Villers-lès-Nancy, F-54600, France
Email: {Alexandre.Boeglin,Olivier.Festor}@inria.fr

***Abstract*—Monitoring constrained, low power and lossy networks is essential to many operations including troubleshooting, forensics, performance management. The main challenge for the monitoring plane in these networks is to efficiently cope with both frequently changing topologies and constrained resources.**
We present a novel algorithm and the supporting framework that improves a poller-pollee based architecture. We empower the poller-pollee placement decision process and operation by exploiting available routing data to monitor nodes status. In addition, monitoring data is efficiently embedded in any messages flowing through the network, drastically reducing monitoring overhead. Our approach is validated through both simulation, implementation and deployment on a 6LoWPAN-enabled network. Results demonstrate that our approach is less aggressive and less resource consuming than its competitors.

I. Introduction

Low Power and Lossy Networks (LLNs) [13] are made of interconnected wireless devices with limited resources in terms of energy, computing and communication. The communication channels are low-bandwidth with high loss rate and their volatile wireless links are subject to failure over time. In addition, link layer frames have high constrains on their size and throughput is limited. These networks are used for many different applications including industrial automation, smart metering, environmental monitoring, homeland security, weather and climate analysis and prediction. A main issue to be addressed in those networks is optimal operation combined with strong energy preservation.

Monitoring, i.e the process of measuring sampled properties of nodes and links in a network, is a key technique in operational LLNs where devices need to be constantly or temporally monitored to assure their functioning and detect relevant problems. Traditional monitoring approaches and technologies like SNMP are natural candidates. Their implementations, messages format and supported functions can to a certain extend be adapted to meet devices constraints (memory, CPU and communication capacity) [1]. An adaptation effort is however, not sufficient in most cases to meet the monitoring requirements and infrastructure constraints of LLNs. A paradigm shift is needed to introduce new protocols and algorithms in how to monitor efficiently LLN networks and to meet their constraints in terms of limited nodes energy and the random nature of wireless links. This has not been addressed adequately in the existing proposals. A monitoring solution dedicated to LLNs needs to be lightweight to reduce its resource consumption; it must be robust and fault tolerant to fight hostile wireless channels effect; it should be distributed not relying on one or a subset of specific nodes, and it needs to be adaptive by dynamically changing its operation to support channels errors and nodes movements. Distributed solutions like for example the poller-pollee monitoring structure [2] hence need to be deployed. The main design criteria are thus to lower communication overhead and to facilitate monitoring nodes placement. Such design requirements can be met by self-organizing and distributed monitoring systems. In the literature, several efforts [3] address the problem of data forwarding algorithms and routing protocols in LLNs, specifically in Wireless Sensor Networks (WSNs), to reliably deliver sensing data under given energy and complexity constraints while being robust against topological changes. Such available protocols and algorithms provide valuable information and messages to drive and maintain a monitoring structure over a LLN. The generated traffic in LLNs is usually highly directed between many nodes and one or few data sinks. To reduce the communication overhead and preserve the channel to the network primary application, we could embed status reports into traveling packets through the network along routing paths where a set of monitoring nodes denoted as pollers are placed. Each poller makes monitoring decision based on the received embedded status reports from nearest pollees.

In this work, we present a novel monitoring framework for LLNs aiming at minimizing the monitoring communication cost and overhead and being robust and adaptive to frequent topology changes while efficiently locating poller nodes. Our approach relies on an assisted routing process to assign poller roles to nodes over a constructed and maintained routing layer by embedding pollers assignment information and monitoring data in packets traveling through the network. This paper makes the following main contributions. (1) We characterize the monitoring problem in LLNs and we propose monitoring approaches that use routing information to place a set of monitoring pollers over a constructed routing graph. (2) We present a piggybacking method that achieves energy saving by disseminating monitoring reports over standard packets traveling through the network. (3) We evaluate through both simulation and real measurement on substantial environments our proposed algorithms. Our results show that our monitoring framework

outperforms the state of the art monitoring approaches in terms of energy consumption and dynamic adaptation.

The remainder of the paper is organized as follows. Section II reviews related work. Section III provides a formulation of the monitoring problem in lossy networks. Section IV presents algorithms to assign poller roles to nodes over an established routing graph. Section V presents the design of a piggybacking technique to deliver monitoring reports. Section VI presents the simulation results. Section VII concludes the paper and presents future work.

II. Related work

Locating and activating monitors in IP-based networks has been addressed in several research works [2], [4]–[6]. The goal was for all of them to minimize the number of monitors in order to reduce the overhead while maintaining a suitable quality of the collected data. The quality level of monitored data is management application dependent. These investigations were all conducted over fixed network topologies which are extremely different from low power and lossy networks, especially regarding the frequency of topology changes and instability of links. The monitoring problem of wireless sensor networks has been partially addressed only in a few works. Hsin et al [7] present a self-monitoring approach based on neighborhood monitoring. Each node monitors its neighbors using two-phase timers to detect node failures caused by internal events, e.g. device malfunction, energy depletion, ... Dong et al [8] present a local monitoring technique where a number of nodes are employed for observing communication links in a specific area of the network. These proposals are orthogonal to our approach. While they focus on improving the monitoring impact of a single active monitor (a poller in our framework), our solution provides an energy efficient strategy to assign pollers roles to a small subset of nodes over an available robust routing graph computed by a third party routing protocol and algorithm. Therefore, nodes could, once assigned a poller role apply neighborhood monitoring algorithms as proposed in [7] or their operation could be scheduled to apply local monitoring schemes as described in [8]. Conversely, the overall monitoring data dissemination overhead of the state of the art approaches can benefit from our piggybacking technique to reduce their traffic overhead. The closest research efforts to our is those presented in [9]. They propose a distributed algorithm using graph theory models to assign poller and pollee roles to nodes over a wireless sensor network. It relies on two distance parameters to control the geometrical distribution of pollers and pollees. Their algorithm builds the monitoring architecture by exchanging specific messages between nodes with respect to distance parameter values. However, their algorithm takes the wireless sensor network as given and fixed which introduces an important communication overhead that will compete with application data transfer. We propose an alternative approach based on locating poller nodes over an existing virtual structure constructed and maintained by a running standard routing process. We make use of a robust and distributed routing algorithm that is able to maintain a routing graph over lossy links and low power nodes. We are mainly focusing on, but not limited to, the RPL protocol [10] that has been proposed by the IETF as a suitable candidate to handle routing in LLNs. RPL mitigates the impact of lossy links by maintaining Destination-Oriented Direct Acyclic Graphs (DODAGs) which are constructed to optimize a set of quality of service metrics required by deployed applications.

III. Models and preliminaries

In this section, we first introduce a network model that captures the lossy nature of LLNs. We then provide a formulation of the monitoring problem for LLNs for which a monitoring layer has to be established to collect network status and provide alarms while preserving resources.

A. Network model

A physical LLN network can be modeled as a communication graph $G(V, E)$ comprised of a set of physical nodes V and E, the set of physical links. For each pair of vertices $u, v \in V, (u, v)$ and $(v, u) \in E$ if and only the nodes u and v are within a transmission range of each other. In this graph, we denote D as a special node defined as a destination node (sink). The set of nodes other than D is finite and denoted as V'. Let us consider a routing algorithm running on G and aiming the construction of an instance of a DODAG $G_D(V', E')$ which means that every node in V' has at least a directed forwarding path using edges $E' \subseteq E$ in the graph to the distinguished destination node D. A chain in G_D is a sequence of nodes $v_0, .., v_k$ such that for any i, $0 \leq i \leq k-1$, either (v_i, v_{i+1}) or (v_{i+1}, v_i) is a link of G_D. A path in G_D is a chain such that for any i, $0 \leq i \leq k-1$, (v_{i+1}, v_i) is a link of G_D. Due to the lossy property of physical links in E, the DODAG G_D is changing over time where directed routing edges between two nodes in V' are constructed according to a height function to be optimized regarding application requirements.

For a given node $v' \in V'$, at a construction step of the DODAG, we denote the set of neighbors of v' in G by $Nv(v')$. The routing process running on this node computes a set of nodes $Np(v') \subseteq Nv(v')$ where for each node $v'_i \in Np(v')$, a path exists from v'_i to the destination node D. We denote $Np(v')$, as the set of candidate parents of the node v'. To each node v'_i, we associate a function $height : Np(v') \times \Theta \times D \rightarrow \mathbb{R}$ that uses a set of metrics Θ to compute its height to the node D. The node v' selects a node $v'_p \in Np(v')$ as its preferred parent such that $height(v'_p) = MIN(height(v'_{\{i=1..k\}}))$. The node v' is thus a direct child or descendant of v'_p, if an edge $(v', v'_p) \in E'$ exists with $height(v'_p) < height(v')$. A neighbor node $v'_i \in Np(v')$ is selected as a candidate parent if $height(v'_i, D) \leq height(v', D)$. We consider $Ch(v'_p)$ as the set of direct children of a parent node v'_p. A node that is not selected as a preferred parent by any other node is a leaf node in G_D. The size k of the set of candidate parents $Np(v')$ available to a node $v' \in G_D$ is variable and it is at least equal to 1 to ensure that the graph G_D is connected. Thus, according to the value of k, each parent v'_p selected by a node v' has a relative importance for delivering data to the destination node D. The importance of a node in

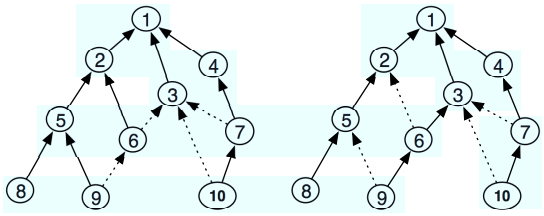


Fig. 1. A simple example of two versions of a DODAG G_D over time where nodes $(6, 9)$ choose new preferred parents to reach the destination node 1. Solid lines denote preferred parents and dashed lines denote candidate parents.

the graph G_D is called its criticality. We define a critical parent v'_c in a DODAG as a parent node that has at least a child who has no parent other than v'_c to reach the data sink node. Thus, a critical parent is a node that has at least a child v' where $\parallel Np(v') \parallel = 1$. We use this criticality measure as a criterion to identify and assign a monitoring role to nodes in the routing graph G_D constructed by a running routing process over the communication graph G. Figure 1 shows an example of two versions of a DODAG G_D where nodes $(6, 9)$ have respectively chosen as new preferred parents $(3, 6)$. Solid lines in the figure denote an edge between a node and its preferred parent and a dashed links denote an edge between a node and a candidate parent. We assume that the routing process propagates advertisement packets to build and maintain the DODAG. The message $ADV : v' \rightarrow Nv$ is a one-to-many message sent by a node to all its neighbors to advertise its *height* to the destination D.

B. Monitoring models

We focus on a two tier monitoring structure as described in [9]. In one tier we place pollees; in the other tier we place pollers. The pollees report the values of the monitored attributes to a poller node along multihop paths [9]. Intermediate nodes may aggregate these reports to reduce overhead. Each poller node takes local decisions based on the received monitoring values and forwards the decisions towards a centralized monitoring station. Given the routing graph G_D, we formulate the monitoring problem with the objective to locate a set of poller nodes $M(G_D)$ and to deliver monitoring reports over the set of routing edges E' while energy consumption is minimized. Thus the set of pollees is $O(G_D) \subseteq V' \setminus M(G_D)$.

1) Pollee monitoring model: Each pollee has access to m real valued variables $X = \{x_1, ..., x_m\}$ measured at their respective monitoring periods $T = \{t_1, ..., t_m\}$. When a period t_i expires, one or several reports that contain the value of the variables $x_i(t)$ is sent to the poller. We model the running monitoring algorithm on a pollee as: $\Gamma : \Xi \rightarrow \Upsilon$, where $\Xi = \{X, T\}$ and $\Upsilon = \{Y, C\}, Y = \{x_i(t), \forall t = k \times t_i, k \in \mathbb{Z}\}$ is the set of variables to be sent and C = $\{c_i(t) \in \mathbb{Z}, \forall t = k \times t_i, k \in \mathbb{Z}\}$ is their associated communication cost.

2) Poller monitoring model: a poller node is running a monitoring algorithm $\Phi : \Lambda \rightarrow \Psi$ which decides to trigger or not an alarm. The algorithm input is $\Lambda = \{L, F, x_i^j(t), t \in T'\}$. The poller has access to p reports containing $x_i^j(t)$ sent by a set $L \subseteq O(G_D)$ of pollees where each pollee $o_j \in L$ has a routing path to the currently assigned poller node. F is a global function to be applied on received x_i^j values [11]. T' denotes the set of detection instants. The algorithm output is $\Psi = \{\psi_1, \psi_2\}$, with $\psi_1 \in \{True, False\}$ and $\psi_2 \subseteq \{1..n + p\}$. The monitoring algorithm Φ is correct if it always detects alarm conditions and is optimal if its cost is always lower than the cost of any other correct algorithm [11].

IV. Routing Assisted Pollers Placement

First, we consider the following problem: given a destination oriented DAG as modeled in section III, built using a gradient-based routing process, in our case the RPL protocol [10], identify a set of nodes to serve as pollers for other pollee nodes. In a constructed and maintained routing DODAG, we use the set of candidate parents $Np(v')$, available to a node v', as the set of candidate pollers. When a selected poller fails or its connectivity becomes unstable, the running routing protocol will react and select a new preferred parent which may be identified and assigned as a new poller if it satisfies a given selection criteria. We discuss two possible location and assignment solution which are easy to figure out. Although both of them have identified flaws, we discuss them to provide the necessary background for our monitoring protocol design.

A. Critical parents-based algorithm

An interesting first approach is to identify critical parents, as defined in section III, within a DODAG constructed by the routing process. The rationale of this choice is that those parents are located near nodes with limited connectivity in the sense that only a single path is available to reach the destination D through a single available parent. To identify critical parents, each node has to inform its selected parent about the number of its candidate parents and its current role. To do that, we need to extend the routing message sent by a node to its preferred parent to carry also this information. The algorithm 1 presents locating poller nodes using the critical parents criteria.

Algorithm 1 Critical-parent based pollers placement.

Input: N the number of directed children of the current node.
Input: $Np(v_i \in Nv)_{i \in 1..N}$ the set of candidate parents of each closed child v_i of the current node.
Function setRole (N, Np)
1: **if** $N = 0$ **then**
2: role := pollee
3: **else if** $\exists v_i \in Nv$ where $\parallel Np(v_i) \parallel = 1$ and role(v_i) = pollee **then**
4: role := poller
5: **else**
6: role := pollee
7: **end if**

Figure 2 depicts the result of applying the above algorithm on the DODAG presented in Figure 1. We observe that nodes 1 and 5 are identified and assigned as pollers. Node 5 is critical

for the child node 8, so it becomes a poller. We observe also that node 2 is critical for its child node 5. However, node 5 is a poller, so node 2 remains a pollee. Node 1 becomes a poller because it is critical for nodes $2, 3, 4$.

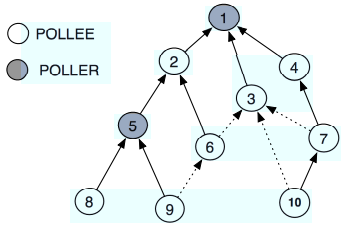


Fig. 2. An example of locating pollers using the critical-parent algorithm. Solid lines represent links to the selected parent and dashed lines represent backup links to candidate parents.

This strategy introduces a small overhead into the routing message where each node needs to inform his parent about its number of candidate parents and its associated role. We have to note that if in the routing graph no node is critical, the only poller will be the root node which has this role by default.

B. k-distance based strategy

A major limitation of the the critical-parents strategy described above is that it may place the pollers far from pollees. This is because in this algorithm, we have no control on the maximum distance between a poller and a pollee. The strategy only ensures that a poller will be available on each critical parent node of the constructed routing DODAG. Figure 3(a)

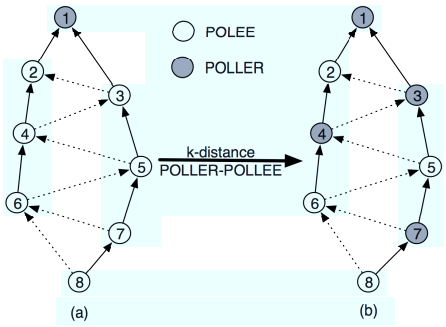


Fig. 3. Illustration of a set of pollers assigned using the proposed algorithm 1. In (a), the distance between the poller 1 and the pollee 8 is equal to 3. (b) represents the resulting placement after applying the *k*-distance poller-pollee strategy, where *k* is equal to 1.

depicts such a situation where we observe that the distance between the poller with the identifier 1 and the edge pollee 8 is equal to 3 hops. In such a situation, a false alarm may be raised since monitoring data may be lost or delayed in the path between an edge pollee and the poller. A false alarm denotes that the poller does not receive monitoring data from a node within a defined time window [9]. To overcome such a situation we define an expansion rule for the number of pollers along a routing path to minimize the probability of a false alarm by enforcing a maximum tolerated distance between a poller and a pollee. The idea is to propagate a predefined parameter k controlling the distance between pollers and pollees along the routing path. A counter c initialized with the parameter k, is piggybacked in any available routing control message. When a node along the routing path receives the control message it extracts the value of the counter. If the receiver node was previously tagged as a pollee using algorithm 1, it decreases the value of c. If the value of c is equal to 0, the node changes its role to poller and reinitializes the value of c to k before embedding it in any future outgoing control message. When a poller node receives a control message with an embedded distance counter, it reinitializes the value of the counter to k before embedding it in any outgoing control message. This strategy can be applied to expand the set of pollers assigned using algorithm 1. It can also be used in isolation to ensure a given distance between pollers and pollees in the constructed routing DODAG. The details of the $k-$distance strategy are described in Algorithm 2.

Algorithm 2 k-distance poller-pollee strategy.

Input: $role$ is the current monitoring role of the node
Input: k is the maximum distance between a poller and a pollee
Function setRole (role, k)
1: **if** $role$ = pollee **then**
2: piggyback in outgoing control message the value k
3: **end if**
4: **if** a message with an embedded distance value is received **then**
5: c = extract distance value from the control message
6: **if** $role$ = pollee **then**
7: c = c -1
8: **if** c = 0 **then**
9: role = poller
10: **end if**
11: **end if**
12: **if** $role$ = poller **then**
13: c=k
14: **end if**
15: piggyback in an outgoing control message the value c
16: **end if**

V. PIGGYBACKING ALGORITHM DESIGN

In this section, we describe the design and operations of a piggybacking algorithm to deliver monitoring reports in existing packets traveling through the network. This technique can reduce the number of monitoring packets and bytes sent across the network since it drastically reduces the number of monitoring packets competing with existing traffic. We propose to piggyback application data and control packets exchanged between nodes and their neighbors. If the monitoring process requires a report to be sent and if no data packet is available to server as the immediate piggy-backing carrier, our process attempts to piggyback the available report into control packets and beacons which are in a one hop scope. The receiver node

looks into both data and control packet to deliver the report upward until it reaches a poller.

A. Piggybacking basic operation

Assuming the monitoring placement is formed using the strategies described in the previous section, piggybacking monitoring data among nodes consists of two steps. (1) If the node is a pollee, it piggybacks its own generated monitoring reports on outgoing packets according to a scheduling strategy where a set of attributes are selected to be piggybacked. (2) When the piggybacked monitoring reports reach a poller node, the monitoring algorithm Φ is executed and possibly an alarm is sent to inform the monitoring station.

The piggybacking algorithm 3 of monitoring reports is executed on each pollee node. The algorithm is executed with the monitoring interval $T_m = gcd(T)$, where T is the set of periods of the monitoring attributes.

Algorithm 3 Monitoring data piggybacking algorithm.

Input: X is the set of monitoring attributes
Function MonitoringPiggyback(X)
  while TRUE **do**
    (p,r_{max}) = nextPacket()
    $\{x_{i\in\{1..k\leq r_{max}\}}\}^{j\in 1..n}$ = scheduleAttributes(r_{max})
    for $x_i^j \in \{x_{i\in\{1..k\leq r_{max}\}}\}^{j\in 1..n}$ **do**
      if $x_i^j(t)$ is available **then**
        piggyback($x_i^j(t)$,p)
      end if
    end for
    wait monitoring period $T_m = gcd(\{T_j\})$
  end while

The $nextPacket$ function returns an outgoing packet p with r_{max} available space to be piggybacked by one or several monitoring attributes. The $scheduleAttributes$ function constructs a schedule of n sets of monitoring attributes $x_i^{j\in 1..n}$ to be piggybacked with respect to the r_{max} available space in the current outgoing packet.

B. Piggybacking scheduling strategies

Piggybacking monitoring data requires the definition of a selection strategy for monitoring attributes to be piggybacked in the outgoing packet. There are two constraints that need to be considered when piggybacking packets. There is a limit on the volume of monitoring data that can be piggybacked into a packet based on the maximum message payload size in the sensor network radio layer. For example, wireless sensor networks using the 802.15.4 protocol stack have a frame size at the media access control layer limited to 127 octets. When using a 6LowPAN adapter, the size of the payload of application data is up to 97-70 octets in the most favorable case. The packet payload size determines the maximum number of monitoring reports it can contain. Piggybacking monitoring attributes together may affect their respective periods. Some of them have to be delayed to be piggybacked together into a single packet or have to wait until a data or control packet is available to be transported. Therefore, the piggybacking process must ensure that the average period of each monitoring attribute remains acceptable to avoid the occurrence of false alarms on the receiving side poller.

The scheduling of piggybacking monitoring data to be transported by outgoing packets largely depends on the period of monitoring attributes. For example, two attributes with the same reporting period or having crossed their respective thresholds close to each other will be able to be piggybacked into a single message if there is sufficient space available in the candidate packet. However, two attributes with different monitoring periods will be piggybacked in very few messages. In [12], authors propose a piggybacking approach to reduce gossiping overhead in sensor networks. They developed a scheduling strategy based on semblance graphs to build a piggybacking scheduler of different streams of messages. We adopted their approach to define a scheduling algorithm to piggyback monitoring attributes onto outgoing packets available in a node. We first define the relatedness of two monitoring attributes x_i and x_j with monitoring periods t_i and t_j respectively as follows:

$$R(x_i, x_j) = \frac{min(t_i, t_j)}{max(t_i, t_j)}$$

The relatedness defines the fraction of outgoing packets which would contain both the two attributes values. The piggybacking scheduler is aiming at building a scheduler of monitoring attributes using the relatedness metric.

Algorithm 4 The Greedy-Kruskal algorithm for piggybacking scheduling.

Input: $G = (X, E)$ is the semblance graph of the set X of monitoring attributes available on a node.
Function scheduleAttributes (r_{max})
1: I:= $\emptyset$
2: **if** F is empty **then**
3: F:= E
4: **end if**
5: **while** (size(I) $\leq r_{max}$) **do**
6: choose e $\in F$ of largest weight
7: F:= F - e
8: **if** $I \cup \{e\}$ is acyclic **then**
9: I:= $I \cup \{e\}$
10: **end if**
11: **end while**
12: **return** I

Monitoring attributes are maintained in a semblance graph where each vertex represents an attribute and edges represent the relatedness between them. Selecting a piggybacking scheduler is equivalent to partitioning this graph into connected subgroups such that the sum of weights is maximized while the number of vertices in any subgroup is less than r_{max}. From [12], we adapted a Greedy algorithm based on Kruskal's algorithm to build a Maximum-Weight Spanning Tree. In our model, the number of partitions which contain the grouped

piggybacked attributes, is dynamic and limited by r_{max} which is specific to the available outgoing packet in a node. The piggybacking scheduling is given in algorithm 4. The algorithm takes as input the semblance graph $G(X, E)$ of available monitoring attributes. It defines the function $scheduleAttributes$ which is called in algorithm 3 and takes r_{max} as the parameter related to the space available in an outgoing packet. The function starts by iterating over the set of non processed edges and selects the edge with the maximum weight (relatedness). Then it adds the selected edge-composing vertices to a subgroup I that obeys to the r_{max} constraint. The next calls to the function $scheduleAttributes$ continues on the set of edges F until all the edges are examined. When the set F is empty, it is re-initialized to the initial set E.

VI. Performance evaluation

In this section, we first present the implementation details of the proposed algorithms regarding pollers selection and assignment as well as monitoring data piggybacking. We then present our methodology and performance evaluation results from both simulation and experiments on a real wireless sensor network.

A. Implementation details

To deploy our approach on a real scenario and networks, we decided to instantiate it on a stack that combines 6LoWPAN and the RPL protocol implementations available in the open-source Contiki [13] operating system. We have implemented the proposed algorithms on the version 2.5 of Contiki OS where we extended its implementations of 6LowPAN and RPL.

The RPL protocol operates the DIO and DAO messages to propagate routing information between nodes. The DIO message broadcasts the height of a node to its neighbors. It is mainly used to build and maintain the DODAG. Every node uses this message to select its preferred parent. The DAO message is sent from a node to its selected parent node to inform him about available downstream routes. The number of candidate parents and the distance between a poller and a pollee required respectively by the algorithms 1 and 2 are piggybacked in the DAO message.

We have extended 6LoWPAN packets with a new IPv6 header option to embed monitoring reports carrying available measured values from pollees. Figure 4 depicts the details of the designed extension. Each piggybacked element is a sequence of type-length-value (TLV) structure associated with the emitter address. Once a monitoring report is available and needs to be sent, the monitoring process attempts to fill the monitoring extension in a 6LowPAN packet bound to a destination towards the poller. When a poller node along the routing path receives the piggybacking extension, it extracts the available measurement values and applies a monitoring algorithm.

B. Methodology

We conducted a set of simulations using the Cooja simulation tool [14] provided by the Contiki project to evaluate pollers placement algorithms. On each node, RPL is enabled as the routing protocol and 6LoWPAN at the network layer. We study the performance of the pollers placement algorithms under various 802.15.4 wireless network densities. Each simulation scenario includes N nodes distributed randomly in a 2D $C \times C$ square area. The communication range of each node is 1. We used the density of the network as $\lambda = \pi \times N/(C \times C)$. We varied the density of the network from $\frac{\pi}{4}$ to π. We choose a 10×10 squares area and nodes uniformly distributed in the area. The number of nodes is determined by the network density, e.g., 100 nodes when λ is π. In the selected wireless model, interfered packets are lost and the success ratio of the transmission and reception can be defined. We only varied the reception ratio RX in the set of values $\{1, 0.5, 0\}$. It means a packet reception probability equal to $1-(1-RX) \times \frac{distance^2}{range^2}$, where $distance$ is the euclidian distance between the emitter and the receiver, and $range$ is the transmission range of the emitter. While the loss model is simple, it does provide some insight into the robustness of the routing and placement algorithms in the presence of loss. The considered routing DODAG is built using the Expected Transmission Count (ETX) metric.

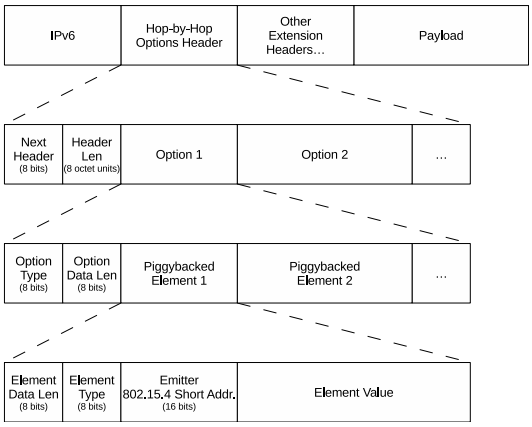


Fig. 4. The structure of the new IPv6 header option to piggyback monitoring data.

We used the performance metrics related to the overhead and the quality of our proposed algorithms for role placement and data piggybacking. We measured the communication cost and overhead associated to the monitoring overlay maintenance and setup. It also includes the communication overhead of monitoring reports in terms of transmitted bytes within a packet. We also measured the coverage in terms of the distribution of the selected pollers over a constructed DODAG. This metric is composed of two sub-metrics. The first metric denotes the number of selected nodes as pollers. The second sub-metric measures the distances between pollers and pollees.

C. Evaluation of the placement algorithms

The communication overhead of our poller selection algorithms is minor, since they use the messages available in the RPL protocol to piggyback an additional role information into

the DAO messages regarding the number of candidate parents of each node. The running cost of a placement algorithm is therefore equal to the cost of setting up and maintaining the routing layer. Several recent studies [15], [16] have investigated the performance of the RPL protocol and its cost, mainly through simulation. It was shown that when the nodes connectivity is stable, the constructed DODAG becomes stable and the number of control packets is low. The communication overhead becomes important when the DODAG needs to be globally built or locally repaired. Thus, when the DODAG is stable, the set of assigned pollers on critical parents is stable.

We evaluated the critical parent strategy as described in algorithm 1 both with and without reception loss for each network density scenario.

Figure 5 shows snapshots of the poller-pollee distribution after running the critical-parent algorithm on respectively 25, 50 and 100 nodes with a reception ratio RX equal to 0. Figure 5(a) shows that with a low density and lossy network, several nodes can become isolated and the routing process is not able to establish routes to the sink (node 1). We also observe that several nodes could not be properly designated as pollers because the DAO routing messages are lost and the relevant information is not available for their election. However, when the network becomes more dense, as depicted in Figure 5(b,c), the problem is partially fixed and more pollers are elected and the network becomes connected to the destination node. However due to packets reception loss, few nodes (e.g., 14,11,22,16,36,41,2,24) are still isolated as depicted in Figure 5(b).

The fraction of designated pollers over all the available nodes is depicted in Figure 6. We observe that when both the network density and loss are low, the fraction of elected pollers remains the same and reaches 0.44. However, when the network is lossy, the fraction of elected pollers remains stable between 0.33 and 0.36. This is due to the loss of routing messages and the partial disconnection of the constructed DODAG. When the network becomes dense, the fraction of pollers decreases since several nodes have multiple candidate parents, but remains between 0.26 and 0.31 even under a medium or high loss probability.

In [9], authors study the effect of distance between a poller and a pollee on the false alarm rate which increases as the distance increases. According to their results, a distance of 3 guarantees a low false alarm rate. We used this value to assess the coverage metric of our pollers assignment algorithm. Figure 7 shows the average distance which is calculated by averaging the sum of the distances between assigned pollers and their nearest pollees. It can be seen that when both network density and loss are low, the average distance is stable and remains close to 1. As shown in Figure 8(a),(b) the distance between pollees and their nearest pollers is bounded by 2 hops and 3 hops when the network size is 25 and 50, respectively. However, when the reception loss becomes important the average distance increases and becomes close to 2 hops.

When the network becomes dense the average distance remains in the interval $]1..2[$. However, in a dense network

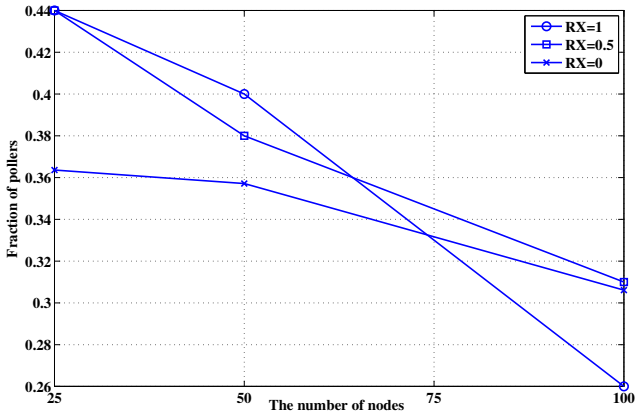


Fig. 6. The fraction of assigned pollers with the critical parents algorithm running on different network densities and under different RX values.

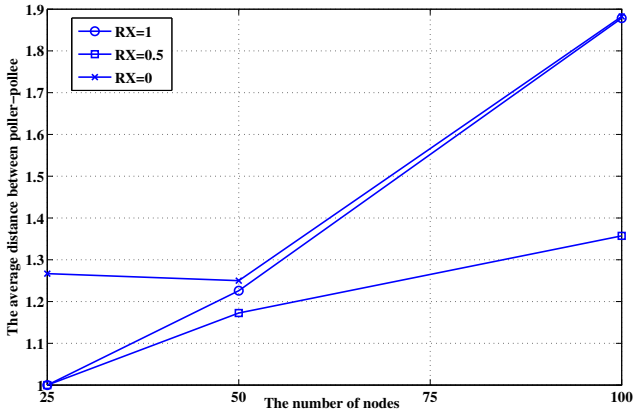
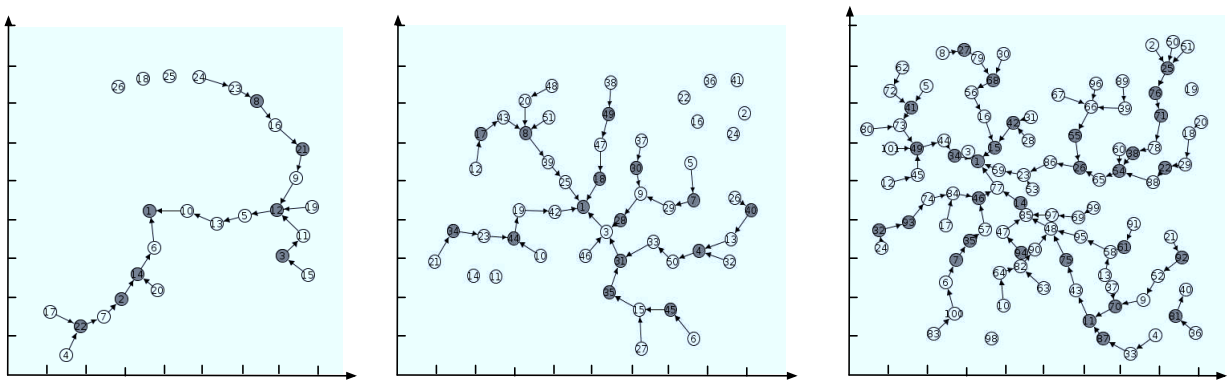


Fig. 7. The average distance between pollers and pollees using the articulation parents algorithm running on different network densities and under different RX values.

as shown in Figure 8(c) 9% of pollees are more than 3 hops away from their nearest poller. Thus, the effect of loss on the distance between pollers and pollees is more important when the network is dense and it is reduced when the network is sparse because more nodes are selected as pollers.

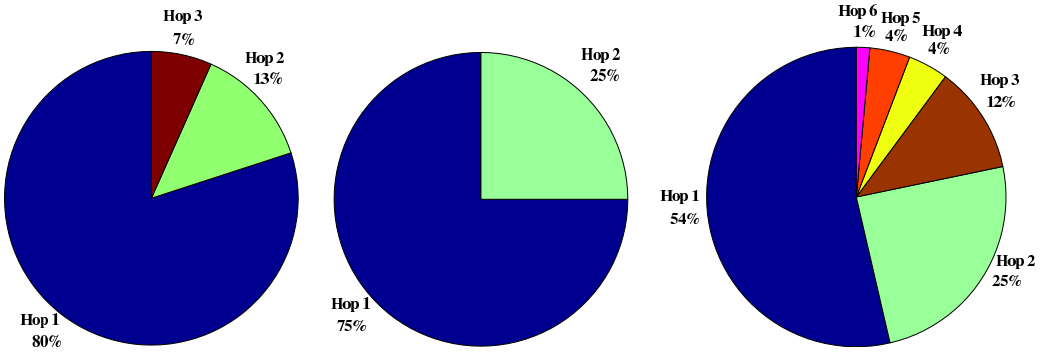
D. Piggybacking evaluation

We have instantiated in a network of 100 nodes a simple UDP application where a packet with a payload of 30 bytes is sent by each node within the interval of 60 seconds to the sink node where a server is running. We have evaluated the impact of piggybacking monitoring data in outgoing packets on each node including application layer packets and ICMPv6 control messages. Each node maintains a single monitoring attribute of 4 bytes value to be piggybacked when an outgoing packet is available. Figure 9 shows the impacts of piggybacking on the distribution of frames length sent by all the nodes in the network. We observe that frame length values become variable with a piggybacking activity since embedded monitoring data



(a) 25 nodes/RX=0 (b) 50 nodes/RX=0 (c) 100 nodes/RX=0

Fig. 5. Snapshot of pollers (gray nodes) distribution for (a) 25, (b) 50 and (c) 100 nodes with a RX value equal to 0.



(a) 25 nodes/RX=0 (b) 50 nodes/RX=0 (c) 100 nodes/RX=0

Fig. 8. Distribution of distance between pollers and pollees under different network densities with RX value equal to 0.

increases packets length and several packets are fragmented. However, without piggybacking, frames length is clustered around 105 bytes and 65 bytes. As seen, a piggybacking activity introduces an overhead, but it is more efficient than using dedicated packets to send monitoring data. We have observed that when using piggybacking, the fraction of fragmented packets from overall packets is around 1.2%.

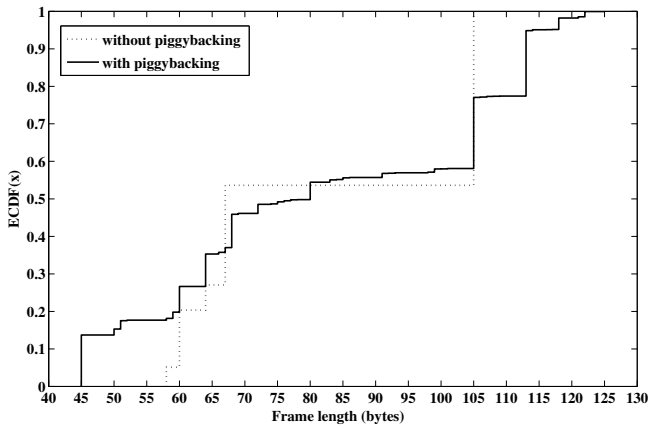


Fig. 9. The ECDF of frames length in a network of 100 nodes with and without monitoring data piggybacking.

We have also measured monitoring and user application delays between successive collected values on the sink node. We selected the sink as a single poller in the network to collect monitoring values.

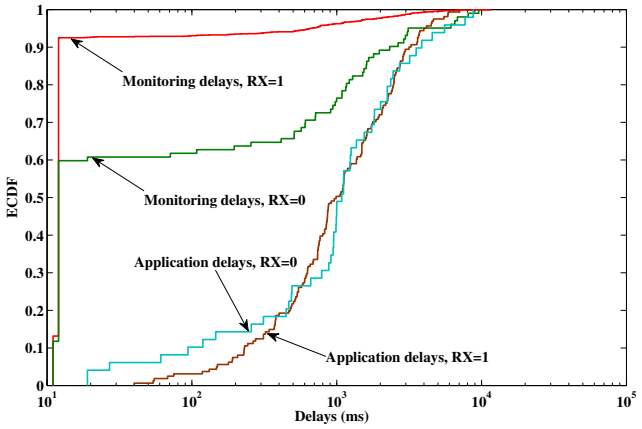


Fig. 10. The ECDF of monitoring and application delays in a network of 100 nodes.

Figure 10 shows the distribution of measured delays on the sink node. It can be observed that around 92% of monitoring delays do not exceed 12 ms with a reception ratio $RX = 1$

and decreases to 60% when the network becomes more lossy ($RX = 0$). We observe also that lower and upper application delays remain close to 40 ms and 7 seconds respectively. Monitoring delays are therefore associated to piggybacked control packets rather than application data packets. Thus, an opportunistic piggybacking strategy where monitoring values available in a node are embedded in every outgoing control and data packet decreases the monitoring delays and provides better detection times automatically reducing the false alarm rate.

VII. Conclusions and future work

In this paper, we have addressed the problem of efficiently monitoring a low-power and lossy-network while meeting their energy saving requirements and deal with the lossy nature of their links. We have designed algorithms to assign monitoring poller roles to nodes in the routing graph constructed using a standard running routing protocol. Through modeling, simulation, implementation and measurement, we demonstrate that our approach, where the poller assignment process is assisted by routing information, provides acceptable distance between pollers and pollees that guarantees a low false alarm rate. Our piggybacking approach ,where monitoring attribute values are scheduled and embedded in packets traveling over the network greatly reduces both the communication cost of monitoring activities and preserves the energy of the sensors for their applications. Future work will be pursued in three directions. First, we will develop new management applications that exploit the designed scheme. The first ones on which we are already working are security monitoring applications relying on the developed poller-pollee structure to detect malicious attacks and nodes. Second, we will investigate multi-objective heuristics from available information on the protocol stack of a node to select pollers. Third, distributed scheduling of piggybacking activities to carry monitoring, control and application data have to be addressed to meet monitoring timeliness while saving energy.

Acknowledgment

The authors were partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] S. Kuryla and J. Schönwälder, "Evaluation of the resource requirements of snmp agents on constrained devices," in *5th International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2011, Nancy, France, June 13-17, 2011. Proceedings*, ser. Lecture Notes in Computer Science, vol. 6734. Springer, 2011.

[2] E. L. Li, M. Thottan, B. Yao, and S. Paul, "Distributed network monitoring with bounded link utilization in ip networks," in *INFOCOM 2003. he 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, San Franciso, CA, USA, March 30 - April 3, 2003*. IEEE, 2003.

[3] T. Watteyne, A. Molinaro, M. G. Richichi, and M. Dohler, "From manet to ietf roll standardization: A paradigm shift in wsn routing protocols," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 4, pp. 688–707, 2011.

[4] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran, "Reformulating the monitor placement problem: optimal network-wide sampling," in *Proceedings of the 2006 ACM CoNEXT conference*, ser. CoNEXT '06. New York, NY, USA: ACM, 2006, pp. 5:1–5:12.

[5] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: complexity, heuristics, and coverage," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 1. IEEE, 2005, pp. 351–361.

[6] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices," in *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, ser. CoNEXT '05. New York, NY, USA: ACM, 2005, pp. 71–82.

[7] C. Hsin and M. Liu, "Self-monitoring of wireless sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 462 – 476, 2006.

[8] D. Dong, X. Liao, Y. Liu, C. Shen, and X. Wang, "Edge self-monitoring for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 514–527, Mar. 2011.

[9] C. Liu and G. Cao, "Distributed monitoring and aggregation in wireless sensor networks," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 2097–2105.

[10] J. Hui and J. Vasseur, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Requests for Comment, RFC Editor, Fremont, CA, USA, Tech. Rep. 6550, Mar. 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6550.txt

[11] M. Dilman and D. Raz, "Efficient reactive monitoring," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2001, pp. 1012–1019.

[12] E. Ucan, N. Thompson, and I. Gupta, "A piggybacking approach to reduce overhead in sensor network gossiping," in *Proceedings of the 2nd international workshop on Middleware for sensor networks*. ACM, 2007, pp. 19–24.

[13] J.-P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[14] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, nov. 2006, pp. 641 –648.

[15] J. Tripathi, J. de Oliveira, and J. Vasseur, "A performance evaluation study of rpl: Routing protocol for low power and lossy networks," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, march 2010, pp. 1 –6.

[16] U. Herberg and T. Clausen, "A comparative performance study of the routing protocols load and rpl with bi-directional traffic in low-power and lossy networks (lln)," in *Proceedings of the 8th ACM Symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ser. PE-WASUN '11. New York, NY, USA: ACM, 2011, pp. 73–80.