Using the RPL Protocol for Supporting Passive Monitoring in the Internet of Things

Anthéa Mayzaud, Anuj Sehgal, Rémi Badonnel, Isabelle Chrisment, Jürgen Schönwälder

Using the RPL Protocol for Supporting Passive Monitoring in the Internet of Things

Anthéa Mayzaud*, Anuj Sehgal†, Rémi Badonnel*, Isabelle Chrisment* and Jürgen Schönwälder†
*TELECOM Nancy, Université de Lorraine, LORIA UMR 7503,
Inria Nancy-Grand Est, Villers-lès-Nancy, F-54600, France
Email: {anthea.mayzaud, remi.badonnel}@inria.fr, isabelle.chrisment@loria.fr
†Computer Science, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany
Email: {s.anuj, j.schoenwaelder}@jacobs-university.de

***Abstract*—Most devices deployed in the Internet of Things (IoT) are expected to suffer from resource constraints. Using specialized tools on such devices for monitoring IoT networks would take away precious resources that could otherwise be dedicated towards their primary task. In many IoT applications such as Advanced Metering Infrastructure (AMI) networks, higher order devices are expected to form the backbone infrastructure, to which the constrained nodes would connect. It would, as such, make sense to exploit the capabilities of these higher order devices to perform network monitoring tasks. We propose in this paper a distributed monitoring architecture that takes benefits from specificities of the IoT routing protocol RPL to passively monitor events and network flows without having impact upon the resource constrained nodes. We describe the underlying mechanisms of this architecture, quantify its performances through a set of experiments using the Cooja environment. We also evaluate its benefits and limits through a use case scenario dedicated to anomaly detection.**

I. Introduction

The ability of networking low-cost embedded devices via wireless communication and interconnecting them to the Internet opens the way to multiple new applications. The capabilities of these embedded computing devices can range from simple environmental sensors to complex actuators in manufacturing plants. Their large-scale deployment leads to the emergence of so-called Internet of Things (IoT) [1], where these devices are integrating with services to support various application domains such as home automation, advanced metering infrastructures (AMI) for smart electricity grid, medical monitoring services and intelligent transport systems [2]. They are typically equipped with 8-bit or 16-bit microcontrollers that possess limited RAM, storage and computing capabilities. Low-power lossy radio communication channels, such as the IEEE 802.15.4 radio, are also likely to be used on such devices to conserve energy [3]. Even though several classes of devices can be employed, as described in Table I, the available computing resources are quite minimal, when compared to standard computing devices used in most applications today. This means that protocols for the Internet of Things must operate within the resource constraints implied by these devices.

Monitoring and managing the Internet of Things using IP-based protocols would be ideal to permit its straightforward integration with the traditional Internet infrastructure. However, previous analyses have already shown that existing protocols such as SNMP and NETCONF, can require significant resources on embedded devices [4]. As such, their usage pose resource and scalability issues with respect to the large scale deployments expected in the Internet of Things. Alternatively, passive monitoring provides an interesting strategy for minimizing the number of devices requiring to be instrumented. Monitoring solutions should therefore have a minimal impact on IoT infrastructures and take benefits from protocols designed for them. In particular, the Routing Protocol for Low-power Lossy Networks (RPL) has been introduced by the IETF [5] to address these devices. This protocol enables a distance-vector routing based on IPv6. The RPL devices are interconnected according to a specific topology, which combines mesh and tree topologies, called Destination Oriented Directed Acyclic Graphs (DODAG). A network can operate one or more RPL instances which consist of multiple DODAG graphs. This multi-instance mechanism allows establishing several routing topologies targeting different objectives within the same network. The properties of such a dedicated protocol should be exploited to support monitoring activities.

TABLE I: Classes of constrained devices used in the Internet of Things (IoT) [3]

Device Classes	RAM	ROM
C0	$<$ 10 KiB	$<$ 100 KiB
C1	$\sim$ 10 KiB	$\sim$ 100 KiB
C2	$\sim$ 50 KiB	$\sim$ 250 KiB

We propose in this paper a distributed and passive approach to monitor network traffic in the IoT. This one relies on an architecture composed of higher order dedicated monitoring nodes that passively monitor an Internet of Things infrastructure. The objective is to preserve the resources of regular nodes, and exploit the RPL protocol mechanisms to support monitoring activities in an efficient manner. Our main contributions are (1) the design of a distributed passive monitoring architecture for the Internet of Things, (2) the instantiation of this solution based on RPL protocol mechanisms, (3) the quantification of its performance through an extensive set of experiments with the Cooja environment, (4) a feasibility evaluation based on a use case scenario dedicated to anomaly detection.

The rest of the paper is organized as follows. Existing work related to monitoring architectures for the Internet of Things is presented in Section II. Section III introduces our

distributed and passive monitoring architecture, describes its main components and mechanisms based on the RPL protocol. Experimental results regarding to the architecture performance and cost, as well as a use case scenario dedicated to anomaly detection are given in Section IV. Finally, the paper draws conclusions and points out future research perspectives in Section V.

II. Related Work

Internet of Things infrastructures require lightweight methods and techniques to observe network devices and make sure services are properly running. Such monitoring information provides an important data source to identify failures, optimize the network functioning, and detect potential security attacks.

While traditional management protocols, such as SNMP and NETCONF, have been adapted to resource constrained environments, an analysis of these configuration monitoring protocols [4] shows their limits in the context of the Internet of Things. The NETCONF protocol is quite resource heavy due to its reliance on XML. SNMP performs relatively well, as long as authentication and encryption are not utilized since these tasks occupy most of the device resources [6]. Even though SNMP appears to be a good candidate for monitoring IoT networks, the integration of SNMP agents with their management information base (MIB) on resource constrained devices may take away valuable resources. This is especially true on C0 and C1 devices, where the amount of RAM available to nodes is quite restricted. It is important to note that these devices are likely to be the majority of deployed IoT devices [3].

In addition to configuration monitoring, several passive solutions have been proposed for analyzing network activities in wireless sensor networks. For instance, Khan et al. [7] introduced a troubleshooting suite to facilitate the identification of anomalies in sensor applications. In the same manner, LiveNet proposed to reconstruct the complex behavior of a deployed sensor network using multiple passive packet sniffers collocated with the network [8]. Several other monitoring approaches such as [9], [10], [11] and [12] have shown the benefits of passive monitoring in sensor networks. In that case, the monitored nodes do not require to be instrumented. Security monitoring solutions have also been designed for the IoT. For instance, the SVELTE framework [13] consists in an intrusion detection system capable of rebuilding the topology based on node messages, identifying intrusion behaviors and protecting the network based on a distributed firewall. In the same manner, a specification based solution is proposed in [14] to detect attacks in an RPL-based network. These approaches present solutions exclusively dedicated to security aspects.

In such constrained networks, it is necessary to provide a monitoring solution with a minimal impact on the network resources. Solutions based on traditional management protocols require instrumentation of constrained devices, utilizing their scarce resources. Passive monitoring strategies from wireless sensor network rely on sniffers, which do not participate to the network. Some other solutions are clearly specifically dedicated to security and not suitable for other monitoring purposes. Since many IoT applications deployed on heterogeneous networks including constrained and higher order devices, we argue in favor of not instrumenting constrained nodes, but relying on those higher order devices to observe the network in a passive manner. In addition, such a strategy should exploit dedicated IoT protocol mechanisms, such as the RPL standardized protocol to perform an efficient monitoring.

III. Distributed Monitoring Architecture

We propose a distributed monitoring architecture for the Internet of Things that passively observes the network. It is based on dedicated nodes and relies on the RPL protocol mechanisms to perform monitoring operations. We describe both the main components of this architecture and the RPL-oriented features that are exploited to instantiate it on an IoT network.

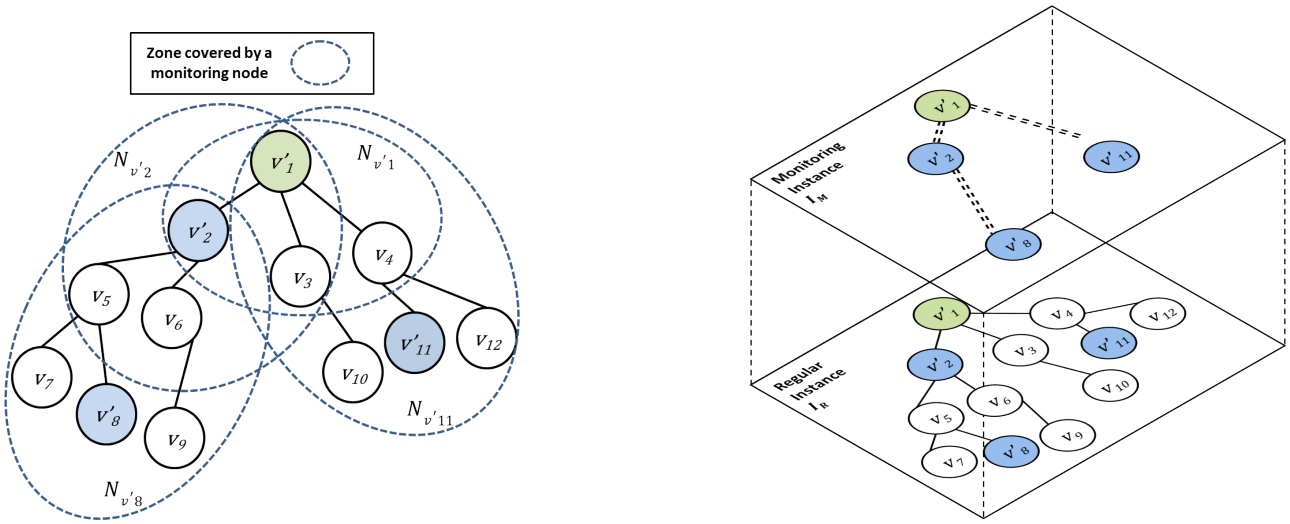
A. Overview and components

Our monitoring architecture described in Figure 1 is composed of two types of nodes participating in the network, *monitored* nodes, also called regular nodes, plotted in white, and *monitoring* nodes plotted in blue.

The monitored nodes are typically lower order devices that fit into the C0 or C1 class of constrained devices. Their primary function is to carry out their assigned sensing or actuation task and form the so-called regular network. They communicate with a sink/controller, where all collected sensing data is forwarded or from where actuation commands might be periodically received. This communication occurs over low-power lossy channels and a multi-hop mesh network might be formed in order to enable interconnection between all nodes.

The monitoring nodes are higher order devices that are at least C2 or better. As such, their monitoring activities will not have an effect upon their ability to serve their primary purpose of routing information in the regular network. These monitoring nodes should be capable of passively listening to the monitored nodes in their radio communication range, while also recording required information. Determining the optimal placement of monitoring nodes is beyond the scope of this paper, as such, we assume that the monitoring nodes are placed so that all regular nodes are covered by at least one monitoring node.

Since the higher order devices, instrumented as monitoring nodes, are expected to be deployed in many IoT applications, those nodes participate in the regular network. As such, they are able to intercept and analyze packets sent by regular nodes. A monitoring node can only monitor its own low-power lossy network neighborhood. However, network-level monitoring information is useful to track the topology and inconsistencies in the network, e.g. topological, security, etc. As such, these monitoring nodes must periodically forward the collected monitoring data towards a sink. To avoid using the resource of constrained nodes, the monitoring nodes form a second routing topology as illustrated by the upper part of Figure 1(b). This second network, known as the *monitoring network*, has access strictly limited to monitoring nodes. Two possibilities can be considered to build the monitoring network depending on the use case. If the monitoring nodes have an high-speed



(a) Monitoring nodes snooping packets transmitted by nodes in radio range.

(b) Building of two RPL instances.

Fig. 1: A passive monitoring architecture exploiting the RPL multi-instance feature, where all nodes except the root and monitoring nodes are resource constrained devices. Blue nodes represent the monitoring nodes and the root is green. (a) Circled areas represent the neighborhood of the different monitoring nodes. (b) The first instance, on the lower plane, is the regular IoT network that typically uses low-power lossy radio channels. The second instance, on the upper plane, is a network only composed of monitoring nodes. Both RPL networks may either be separate RPL instances within the same channel, or operate as independent networks using different channels.

high-bandwidth access network they can interlink with each other and with the sink, this can be the case in AMI (Advanced Measurement Infrastructure) deployments [15]. The second possibility is to share the same medium as the regular network, the interconnection is feasible using different radio ranges for monitoring nodes which is possible considering higher order devices. The monitoring network will form an overlay network.

Complementary to the data collected by a monitoring node based on packets it has to process, i.e. data and control messages that are legitimately transmitted to it, it may enable the promiscuous mode in order to cover a larger quantity of packets. The promiscuous mode allows a node to overhear packets, it is particularly useful for detecting anomalies and potential attacks by snooping data traffic not transmitted to them.

B. RPL-based mechanisms

This passive monitoring solution is instantiated based on the RPL protocol mechanisms. The RPL protocol forms loop-free topologies termed Destination Oriented Directed Acyclic Graphs (DODAGs), which organize nodes into a hierarchical structure of a single root, children and further descendants. This topology is presented in Figure 1(a) where node v'_1 is the root. Objective functions are used by the protocol to optimize the topology based on predefined goals, e.g. energy consumption, hop-count or link quality. Multiple instances of RPL, each being an execution of RPL with a specific objective function, can be run within a network, each with its own DODAGs [5] as illustrated in Figure 1(b) where the network is composed of two instances I_M and I_R with one DODAG each. While a node may be a member of multiple instances, it can only join a single DODAG in an instance at any point in time such as nodes v'_2, v'_8 and v'_{11} in Figure 1(b) which are part of both instances. If a node does not cope with the objective function it cannot join the graph. New ICMPv6 control messages are defined to build and maintain the topology. These are DODAG Information Solicitation (DIS), DODAG Information Object (DIO) and Destination Advertisement Object (DAO). The rank value of a node indicates its position with respect to the DODAG root. It is calculated using the objective code point advertised in DIO messages and on rank value advertised by neighbor nodes. The rank preserves the acyclic nature of the DODAG graph. RPL includes also global and local repair mechanisms. When the *global repair* mechanism is used, it initiates a rebuild of the entire DODAG by incrementing the version number of the DODAG [5]. The version number is carried in the DIO message; each node receiving a DIO compares its existing version number against the one received from its parent and in case the received version is higher, it must ignore its current rank information and initiate a new procedure to join the DODAG. To avoid rebuilding the entire DODAG when a parent node disappears, two *local repair* mechanisms are also provided [5]. The first allows nodes to temporarily route through neighbors of the same rank, while the other consists in switching parents. These approaches may be used in combination as well to avoid any loss of connectivity.

Applying the RPL protocol to a network leads to the building of a DODAG in a instance I noted D^I. We note as $D^I(V, E)$ the DODAG graph composed of V nodes linked using E edges. Every node participating in the DODAG has an access to the root or sink S using the E edges which are chosen among all the links available to cope with the objective function. We note N_{v_i} the neighborhood of a node v_i which is the set of nodes $\{v_k\}$ in the communication range of v_i. The neighboring nodes of v_i can be parents whose

rank is lesser than the rank value of v_i, children whose rank is greater or siblings with the same rank value. The rank value of a node v_i is noted r_i. We exploit the multi-instance feature of RPL to build two networks: a regular network and a *monitoring network*. An instance in RPL can be seen as a network optimized for specific metrics or constraints given by an objective function. The RPL multi-instance principle is an example of VRF (Virtual Routing and Forwarding): multiple instances of a routing table coexist on the same router at the same time. Those instances are completely independent which means if one network breaks down at some point because of node's failure or attacks the second network can operate normally. Therefore, two instances are running at the same time in our solution: one instance for the regular service noted I_R wherein the DODAG built is noted D^{I_R} composed of V nodes and one monitoring instance, I_M where the DODAG is noted D^{I_M} also called *monitoring network* as shown in Figure 1(b). Using RPL multi-instance feature presents two main advantages. First it allows us to preserve regular nodes' resources because monitoring nodes forward their data on their monitoring instance/network. Second, if the regular network malfunctions, the monitoring data will still be forwarded thanks to the monitoring network, which is independent of the regular one and does not rely on regular nodes.

As previously explained the set of monitoring nodes V' is included in the set of regular nodes V i.e. $V' \subset V$ so the monitoring nodes can participate to D^{I_R}. The sink S is also a monitoring node. A monitoring node, v'_k is able to collect information regarding its neighborhood $N_{v'_k}$ as illustrated in 1(a), the zone covered by a monitoring node is its neighborhood. The collected information allows it to monitor the network and also detect possible anomalies by implementing locally detection algorithm. In Figure 1(a), monitoring node v'_8 is able to monitor $N_{v'_8} = \{v_5, v_6, v_7, v_9\}$ using passive listening and overhearing. The monitoring supports the detection of local anomalies based on dedicated detection modules. Collected information as well as detection results may then be aggregated and forwarded to its monitoring neighbor v'_2. Since neighborhoods of nodes v'_8 and v'_2 overlap, node v'_2 checks if information gathered by node v'_8 matches its own information in order to refine the detection in a collaborative manner. Node v'_2 performs the same process as its predecessor: collect information, run detection algorithms, aggregate the different sources of data and report it to the next monitoring node which is here the sink. Since the sink collect data from the different monitoring nodes, it may detect inconsistencies only observable at a global level.

A monitoring node is able to record the following RPL statistics from intercepted messages:

- **Instance ID** observed in messages originating from monitored nodes.
- **DODAG ID** observed in messages originating from monitored nodes.
- **DODAG Root** destination address observed in all data packets from monitored nodes.
- **DODAG Version** observed from RPL control messages originating at each monitored node.
- **Node DODAG Rank** observed from RPL control messages originating at each monitored node.
- **Node Objective Function** observed from RPL control messages originating at each monitored node.
- **Delay between DAO messages** observed by timing the frequency of DAO message reception from each monitored node.
- **Local Repairs Triggered** the number of local repairs triggered by a node.
- **Global Repairs Triggered** the number of global repairs triggered by a node, i.e. higher DODAG version advertised by a non-root node.
- **Minimum Rank Increase** the option observed in control messages advertised by a non-root node.
- **Maximum Rank Increase** the option observed in control messages advertised by a non-root node.
- **Down Flag Set** number of packets observed from a monitored node with the Down flag set.
- **Forwarding-Error Flag Set** number of packets observed from a monitored node with the Forwarding-Error flag set.
- **Rank-Error Flag Set** number of packets observed from a monitored node with the Rank-Error flag set.
- **DAO Message Count** number of RPL DAO control messages observed from a node.
- **DIO Message Count** number of RPL DIO control messages observed from a node.
- **DIS Message Count** number of RPL DAO control messages observed from a node.

Those statistics allow detecting potential misconfigurations as well as misbehaviors in the RPL functioning. For instance, a node except the root is not allowed to increment the version number. Such inconsistencies can be detected based on global repairs statistics collected by our passive monitoring solution for the Internet of Things. It is also possible to record statistics about other used protocols in the stack.

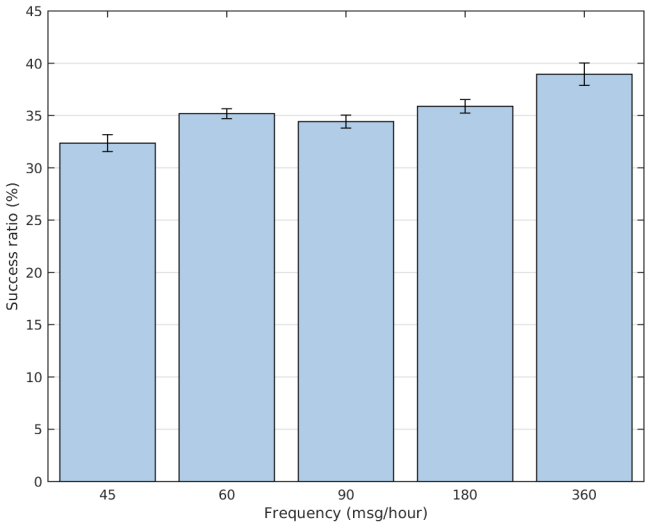
IV. Experimental Results

We have evaluated our monitoring architecture through a set of experiments. In particular, we have quantified the performance and the cost of the promiscuous mode, and have analyzed its exploitation in the context of anomaly detection.

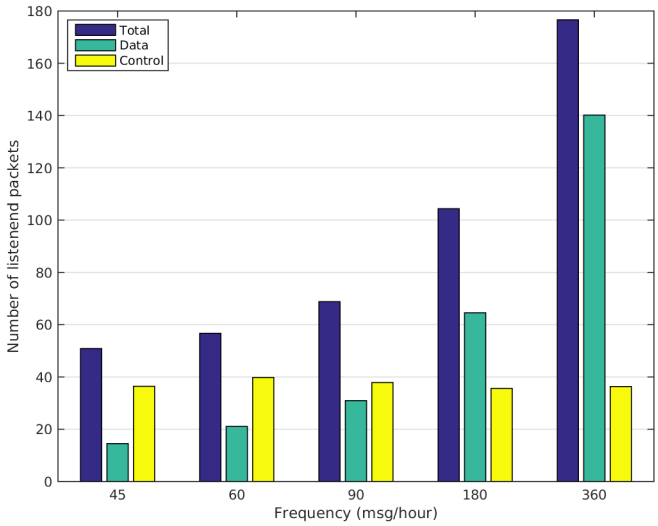
A. Overhearing evaluation

To evaluate the performance and the cost of the overhearing mode, we set up a simple scenario composed of three nodes: a root, a regular client configured to send data traffic periodically and directly to the sink and a monitoring node with the promiscuous mode enabled which was equidistant from the previous nodes. This scenario allows us to study three parameters separately and measure their potential influence on the overhearing capacity of a monitoring node: (i) the distance from the monitoring node to the node it monitors, (ii) the sending frequency of a monitored node and (iii) the number of neighbors of a monitoring node.

This scenario was implemented in the Contiki 2.7 operating system [16]. The TelosB, also known as the TMote Sky, was chosen as a target platform for regular nodes. Its computational resources allow it to function as an RPL router node with the Contiki RPL implementation and is quite representative of C1 device types, and as such makes for a good platform to use for performing analysis on. The Cooja simulator provided by Contiki [17] was used during this analysis to execute the code



(a) Average success ratio.



(b) Average number of packets listened in one hour.

Fig. 2: Performance of the overhearing mode when the sending frequency varies from 45 messages per hour to 360 messages per hour.

TABLE II: Performance while varying the distance.

Distance (m)	15	20	30	40	50
Success ratio (%)	36	35.5	35.5	36.5	36
Number of listened data packets/hour	64	65	65	65	65

TABLE III: Performance while varying the neighborhood size.

Number of neighbors	2	4	6	8	10
Success ratio (%)	36.5	38.5	37.5	38.5	38
Number of listened data packets/hour	66	208	337	482	612

written for the TelosB platform. Since Cooja does not have access to a device model for any higher order devices, the TelosB platform was used not only as monitored nodes, but also as monitoring nodes. The monitoring node was configured to enable the promiscuous mode. Each simulation lasted for a lifetime of eight hours and was repeated six times for accuracy reasons.

The metrics used to evaluate the different parameters are: (i) the success ratio which is the number of overheard data packets over the number of data packets sent by regular nodes in percentage and (ii) the number of overheard packets. Through the simulations it was experienced that the monitoring node can overhear two types of packets: data packets and point-to-point control messages (not destined to itself) which were ICMPv6 Neighbor Solicitation and Neighbor Advertisement messages.

Performance analysis: In a first series of experiments the influence of the distance was studied. For this scenario, the position of the monitoring node varied from 15 to 50 meters from both the regular node and the sink. The regular node was configured to send data every 20s (180 msg/hour). Table II gathers results for different distances. Looking at the ratio which is around 36%, we see that a monitoring node cannot overhear all messages. This is because the monitoring node has to process its legitimate traffic (control messages) in priority and if its queue is already full so the new arriving overheard packets are dropped. Being able to overhear approximately 1/3 of data packets might seem quite low. However, the target platform used is a TelosB, a C1 class device, so it is expected that for a higher order device the overhearing success ratio would be higher. Also, as explained in Section III, overhearing is not the only source of monitoring data since a monitoring node can also gather data from packets that it legitimately forwards. In this scenario only monitoring data coming from the promiscuous mode is evaluated. As it will be presented in the next section, even if the success ratio may look low it is good enough to detect certain types of anomalies. As we can observe the number of overheard packets does not change over distance as the ratio which implies that the distance does not affect the overhearing in the Cooja environment.

A second series of experiments has focused on the sending frequency. For this study the same topology as before was setup, and the sending frequency of the regular node varied from 45 messages to 360 messages per hour. Since the distance has no influence on the overhearing performance the monitoring was placed at 25 meters. Figure 2(a) presents average success ratio of monitoring node for different sending frequences. Figure 2(b) shows the number of packets listened using the promiscuous mode by the monitoring node. From Figure 2(a), we can see that the ratio lays between 32% and 39% and is slightly increasing with the frequency. We can therefore conclude that the success ratio is relatively stable (low variation). Since the ratio is stable and the number of sent data packets is increasing, it means that the number

TABLE IV: Energy model for the CC2420 radio and MSP430F1611 microcontroller operating at 1 MHz on TelosB.

Operation	Current	Voltage	Part
Receive (R_x)	17.4 mA	2.2 V	CC2420

of overheard data packets also increases. This is confirmed by Figure 2(b) where the number of listened data packets increases with the frequency. We can also note that the number of listened control messages is stable (less than 40 messages), which makes sense because the number of nodes do not change over the simulations and the number of overall exchanged control messages is almost the same for each simulation. As a conclusion it can be said that for this environment the overhearing mode has slightly better results with a heavier load, although this improvement is limited (+7% when the traffic is multiplied by 8). As such, the overhearing success ratio can be considered as stable with the frequency.

Finally, the number of neighbors was analyzed in a third series of experiments. For this scenario the number of regular nodes varied from 2 to 10. The new neighbors were directly connected to the root in the range of the monitoring node. The sending frequency was set to 180 messages per hour. Table III gathers the different results regarding frequency and number of messages. We can see that the ratio stands between 36.5% and 38.5% which is quite stable. For these simulations we can explain the relative stability of the ratio by the fact that not only the number of data packets but also the number of control messages exchanged increases significantly with the number of neighbors. Indeed if we consider the number of control messages exchanged between two nodes as stable over the simulations and if we multiply the number of nodes, in that case the number of listened control messages is multiplied as well. The increase of listened data packets is also proportional to the number of neighbors: for 2 neighbors we have only one regular node which is sending data packets; if we multiply the number of listened data packets by the number of regular nodes we can see that we are close to the results given by the simulation. For instance, $66 \times 3 = 198$ which is close to 208, the number of listened data for 4 neighbors (3 regular nodes). This is also why we did not simulate for more neighbors because the results could be extrapolated.

The different results obtained on the performance of the promiscuous mode for TelosB platform in a Cooja environment are useful information. Indeed, even if the monitoring node can overhear slightly more than a third of transmitted data packets, thanks to the different results an estimation of the actual number of sent data packets can be achieved. Also those results can be helpful when developing detection algorithms. As we know, from the different studied scenarios, distance, sending frequency and neighborhood size do not affect much the success ratio, it is not necessary to take them into account. A similar study should be conducted in real environment in order to adapt the monitoring and detection algorithms.

Cost analysis: Overhearing packets implies a cost for the monitoring node. From the energy model provided by Table IV, we calculated the cost for receiving monitored packets. Figure 3 shows the energy consumption of a monitoring node under the two scenarios used to analyse the frequency and number of neighbors parameters. Since the energy is proportional to the number of packets, the results are similar to the ones presented in the previous section. We can see in Figure 3(a) that the cost in total varies between 65 mJ for 45 msg/hour to 240 mJ for 360 msg/hour.Up to 90 msg/hour the monitoring node spends more energy to overhear control messages than data packets. We can observe from Figure 3(b) that both overhearing data cost and overhearing control messages cost are increasing with the size of the neighborhood as explained earlier. The cost in total varies from 140 mJ (for two neighbors) to 1250 mJ (for ten neighbors) which is up 5 times more than costs presented in Figure 3(a). We see that the cost increases linearly with the size of the neighborhood. In more realistic conditions it is unlikely that client nodes send so many data packets for their application (360 msg/hour represent one message every 10s) and have so many neighbors. We included extreme cases in our analysis.

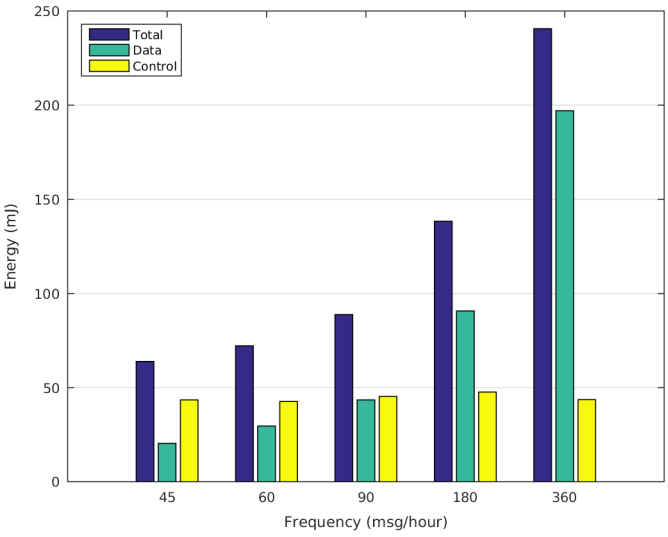
As usual in monitoring and security solutions, a tradeoff has to be made between cost and efficiency. For instance we can optimize the number of monitoring nodes to be deployed but it means that they have to cover more nodes and consequently it costs more in terms of energy. This is also why we proposed in this architecture that the monitoring tasks are supported by higher order devices so the energy is not as restrictive as it can be on usual C0/C1 devices.

B. Exploitation for anomaly detection

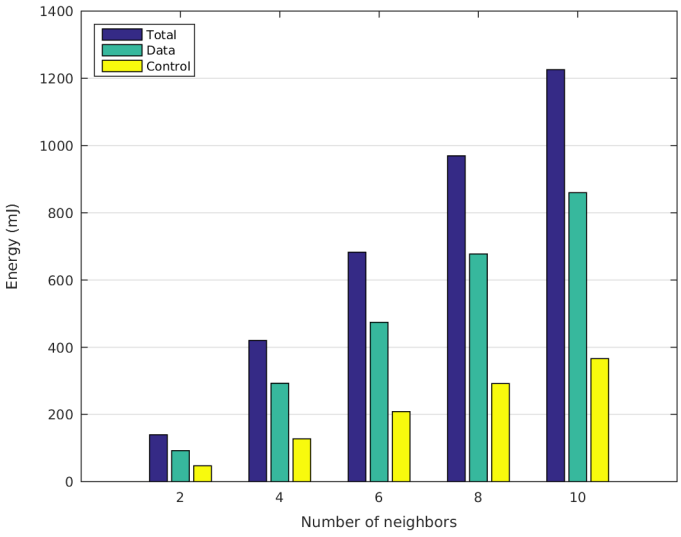
We have complemented the evaluation of our monitoring architecture by considering the use case of anomaly detection. More specifically, we considered an IoT infrastructure where monitoring nodes enable the promiscuous mode and implement a detection algorithm to identify unusual behaviors and potential attacks.

In that context, we have considered the DAG inconsistency attack that we have already studied in [18] and [19]. This attack is interesting because it targets data traffic, and in order to detect it the promiscuous mode has to be enabled on monitoring nodes. The DAG inconsistency attack exploits the data path validation feature of RPL which is used to avoid and detect possible loops within the network. The idea is to use packet information transported in an IPv6 option header. Three flags are defined: the down flag indicating the expected direction of a packet, the rank-error flag indicating if a mismatch occurred between the down flag and the actual direction of the packet and the forwarding-error flag used to indicate if a node cannot reach a destination. The attack consists in manipulating the flags in the IPv6 option header of regular data packets to introduce fake loops in the network. More precisely, an attacker can set the down flag to mismatch the rank relationship e.g. setting the down flag to 1 to an upward packet, and the rank error flag so an inconsistency is detected by the targeted node. Upon receiving such a packet, the targeted node drops it and reinitializes its trickle timer, increasing the number of control messages within the network. The trickle timer is used in RPL to control the sending frequency of control messages.

To detect such anomaly, we have implemented algorithm 1 where R_flag represents the rank error flag in data packets. A monitoring node v'_k tracks for each neighbor $v_i \in N_{v'_k}$ the



(a) Energy consumed under different frequencies.



(b) Energy consumed under different neighborhood sizes.

Fig. 3: Energy consumed by a monitoring node for one hour in different scenarios. (a) Sending frequency of a regular node varying from 45 to 360 packets per hour with 2 neighbors. (b) Number of neighbors varying from 2 to 10 (including the sink) while the sending frequency is set to 180 packets per hour.

foreach *data packets received from* $N_{v'_k}$ **do**
  if *R_flag is set* **then**
    identify sender v_i ;
    $count\_R_i++$;
    if $count\_R_i == THRESHOLD$ **then**
      $alone = 1$;
      foreach $v_j \neq v_i$ *in* $N_{v'_k}$ **do**
        if $count\_R_j > 0$ **then**
          $alone = 0$;
        end
      end
      if $alone == 1$ **then**
        anomaly detected;
      end
    end
  end
end

Algorithm 1: Detection algorithm implemented on monitoring nodes $\{v'_k\}$, $k \in \{1, 2, 8, 11\}$ to detect DAG inconsistency attacks

number of rank error flags $count\_R_i$ they have set. If this counter reaches the threshold value then the monitoring node has to check if v_i is the only node which has sent 'R' flag packets. In this case the monitoring node detects an anomaly otherwise it is considered a legitimate loop. In order to allow nodes to send 'R' flag packets in case of real loop without being detected as anomalous, the different $count\_R_i$ values are reset every hour and the threshold value has been set to twice the maximum number of neighbors of a monitoring node (cf. Table V). It can be envisioned that the threshold could be set dynamically according to each monitoring node configuration.

The topology shown in Figure 1(a) was setup in Cooja using the same configuration as presented previously. To emulate the behavior of monitoring nodes, Cooja was setup to ensure that they all (i.e., nodes v'_1, v'_2, v'_8 and v'_{11}) were within radio range of each other. On the other hand, all monitored nodes were configured such that their radio range was shorter than the monitoring nodes. Furthermore, the RPL implementation provided in Contiki was extended to support multiple RPL instances in a network. The monitoring nodes and the root were then configured to run two instances of RPL. This ensured that we had two separate topologies, as shown in Figure 1(b), built using the same nodes. Across all experiments, node v'_1 is the DODAG root and also the sink, nodes v'_1, v'_2, v'_8 and v'_{11} are monitoring nodes. Client nodes v_i were configured to send data packets to the sink every 20 seconds. The attacker is designed to send attack messages (packets with down and rank error flags set) every 5 seconds in average to its preferred parent which means that the attacker is very aggressive. This corresponds to the extreme case shown in [19]. This frequency was chosen to study the ability of our architecture to deal with aggressive situation. The location of the attacker has been varied within the network replacing a regular node in order to study the detection performance of our monitoring architecture. Attacks start after two minutes of simulation time, so that the network has enough time to settle. Each simulation lasts for a lifetime of two hours and were repeated three times.

TABLE V: Neighborhood of the different monitoring nodes

Monitoring nodes	1	2	8	11
Monitored neighborhood	{3,4}	{3,5,6}	{5,6,7,9}	{3,4,10,12}

Figure 4 depicts the average detection time for the different

TABLE VI: Targeted node for the different location of the attacker

Attacker's position	3	4	5	6	7	9	10	12
Target	1	1	2	2	5	6	3	4

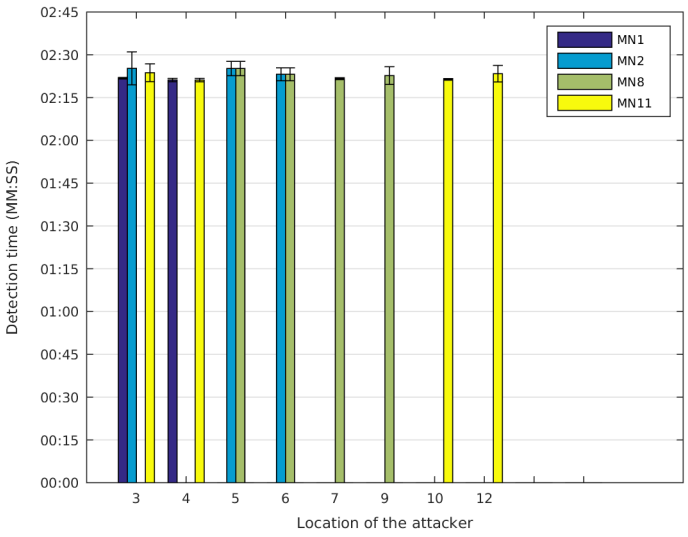


Fig. 4: Detection time for different location of the attacker (with threshold equal to 8). No bar means that the corresponding node could not detect the attack. Error bars indicate standard deviation. MN stands for monitoring node.

positions of the attacker where the threshold value is set to 8. Table V summarizes the regular nodes monitored by the different monitoring nodes as it is shown in Figure 1. The table VI shows for each location of the attacker which node was the target as it can be seen in Figure 1. According to Figure 4 and Table V it can be seen that the anomaly was successfully detected in every case; which means that all monitoring nodes detected the attacker when it was in their neighborhood. In general the detection time stands between 2min20s and 2min25s, since the attack started after 2 min it means that it was detected in less than 30s. However it can be seen that when the monitoring node was directly targeted (cf. Table VI), the detection time is shorter or equal, for example when the attacker is located at 3, node v'_1 is the target, the detection time is 2min21 while it is 2min25 for node v'_2 and 2min23 for node v'_{11}. When a node is targeted the overhearing mode is useless because the attack packets were directly addressed to the monitoring node. A lesser or a bigger threshold value results in detecting the anomaly quicker or slower. However, it is important to keep in mind that low value for threshold might also impact the repair of genuine loop conditions. The aggressiveness of the attack also influences the detection time. Indeed if the attacker was less aggressive the detection time would be higher. While our previous work [19] was focused on mitigation solution deployed on each node, this study is focusing on the possibility to detect the same anomaly without implementing a detection algorithm on each regular nodes.

This scenario showed that our monitoring architecture can be used to detect anomalies in a RPL network. In this scenario the algorithm was implemented for a local detection but the architecture can be exploited so the monitoring nodes can exchange data to provide a collaborative detection and when all monitoring data are gathered in the sink, the architecture would be able to perform a global detection. This algorithm can be seen as one detection module deployed on each monitoring node, we can envision to implement much more modules for different attacks so the architecture can be used as an IDS.

V. Conclusions

We have proposed in this paper a distributed passive monitoring architecture for Internet of Things. The architecture relies on the RPL standardized routing protocol to monitor the network in a lightweight manner for the monitored nodes. We have described its main components and how they interact based on the RPL protocol. It uses higher order monitoring nodes that are able to passively listening the network while participating to its operation. Its instantiation takes benefits from the RPL protocol mechanisms such as the multi-instance feature in order to establish two separated routing topologies: a first DODAG instance corresponding to the regular network, and a second DODAG instance supporting the monitoring activity. The monitored nodes do not require to be instrumented nor to dedicate resources to the monitoring tasks which are operated by higher order nodes.

We have evaluated our approach through sets of experiments. In particular, we have quantified the performance and cost of the promiscuous mode in that context, considering different distances, sending frequencies and neighborhood sizes. Experimental results with the Cooja environment have shown the feasibility of the proposed monitoring approach with respect to traffic load and neighborhood. We have also considered a use case scenario where we measured the solution performance for supporting anomaly detection. As the approach is passive and does not rely on regular nodes, it permits to minimize the impact on the Internet of Things infrastructure.

As future work, we are interested in performing complementary experiments in a real infrastructure with additional classes of devices implementing the RPL protocol and in comparing our approach to other passive monitoring solutions. We are also planning to evaluate the exploitation of this passive distributed architecture for other use cases, such as attack and intrusion detections [20]. The detection results may then be integrated to a risk management framework to assess network risks and determine mitigation mechanisms and countermeasures to be activated over the Internet of Things infrastructure.

Acknowledgments

This work was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Program.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, October 2010.

[2] M. Ersue, D. Romascanu, J. Schönwälder, and A. Sehgal, "Management of Networks with Constrained Devices: Use Cases," *IETF RFC 7548*, May 2015.

[3] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," *IETF RFC 7228*, May 2014.

[4] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, "Management of Resource Constrained Devices in the Internet of Things," *Communications Magazine, IEEE*, vol. 50, no. 12, pp. 144–149, December 2012.

[5] 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," *IETF RFC 6550*, March 2012.

[6] S. Kuryla and J. Schönwälder, "Evaluation of the Resource Requirements of SNMP Agents on Constrained Devices," in *5th Conference on Autonomous Infrastructure, Management and Security (AIMS 2011), Springer LNCS 6734*, Nancy, France, June 2011.

[7] M. M. H. Khan, L. Luo, C. Huang, and T. Abdelzaher, "SNTS: Sensor Network Troubleshooting Suite," in *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems*, ser. DCOSS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 142–157.

[8] B.-r. Chen, G. Peterson, G. Mainland, and M. Welsh, "LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, S. Nikoletseas, B. Chlebus, D. Johnson, and B. Krishnamachari, Eds. Springer Berlin Heidelberg, 2008, vol. 5067, pp. 79–98.

[9] M. Ringwald and K. Romer, "Deployment of Sensor Networks: Problems and Passive Inspection," in *Intelligent Solutions in Embedded Systems, 2007 Fifth Workshop on*, June 2007, pp. 179–192.

[10] A. Awad, R. Nebel, R. German, and F. Dressler, "On the Need for Passive Monitoring in Sensor Networks," in *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*, Sept 2008, pp. 693–699.

[11] X. Xu, J. Wan, W. Zhang, C. Tong, and C. Wu, "PMSW: a passive monitoring system in wireless sensor networks." *International Journal of Network Management*, vol. 21, no. 4, pp. 300–325, 2011.

[12] F. P. Garcia, R. M. C. Andrade, C. T. Oliveira, and J. N. de Souza, "EPMOSt: An Energy-Efficient Passive Monitoring System for Wireless Sensor Networks," *Sensors*, vol. 14, no. 6, p. 10804, 2014.

[13] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, November 2013.

[14] A. Le, J. Loo, Y. Luo, and A. Lasebae, "Specification-based IDS for securing RPL from topology attacks," in *Wireless Days*, Niagara Falls, ON, Canada, October 2011, pp. 1–3.

[15] Cisco Systems, "Routing in The Internet of Things – M2M Networks," *BRKSPG-1661*, 2013.

[16] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *29th Annual IEEE International Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, November 2004.

[17] F. Osterlind, A. Dunkels, J. Eriksson, and T. Finne, N.and Voigt, "Cross-Level Sensor Network Simulation with Cooja," in *31st Annual IEEE International Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, November 2006.

[18] A. Sehgal, A. Mayzaud, R. Badonnel, I. Chrisment, and J. Schonwalder, "Addressing DODAG inconsistency attacks in RPL networks," in *Global Information Infrastructure and Networking Symposium (GIIS), 2014*, Sept 2014, pp. 1–8.

[19] A. Mayzaud, A. Sehgal, R. Badonnel, I. Chrisment, and J. Schönwälder, "Mitigation of topological inconsistency attacks in rpl-based low-power lossy networks," *International Journal of Network Management*, 2015.

[20] A. Mayzaud, R. Badonnel, and I. Chrisment, "A Taxonomy of Attacks in RPL-based Internet of Things," *International Journal of Network Security*, vol. 18, no. 3, pp. 459 – 473,, May 2016.