Recipes for Faster Failure Recovery in Smart Grid Communication Networks

Oana Balmau[1], Dacfey Dzung[2], Yvonne Anne Pignolet[2]
[1] EPFL, Lausanne, Switzerland, [2] ABB Corporate Research, Baden, Switzerland

***Abstract*—The communication network supporting Smart Grid applications must minimize the time that messages cannot be delivered and thus find alternative paths rapidly. At the same time, the control overhead in normal operation should be limited and not interfere with the data traffic. Hence the routing protocol for such networks has to trade-off these two conflicting goals.**

One of the candidate routing protocols for Smart Grid networks is the IETF IPv6 routing protocol RPL which was designed for low-power and lossy networks. It strives to consume little bandwidth and energy with its traffic overhead and to be under-reactive to network changes. This design choice was made because in many of the technologies envisioned for such networks the bandwidth is scarce and the link quality may vary. Fluctuations should not trigger unnecessary instabilities, thus the system was designed to have inertia before deciding to modify communication paths. However, this robustness is in direct conflict with the agility that is required for fast recovery in case of communication failures. In this article we investigate the behavior of the *local repair* failover mechanism, then we propose and evaluate simple options to minimize the recovery time.

I. Introduction

To enable the intelligent management of the production and consumption of electrical energy with Smart Grid applications a suitable communication infrastructure has to be in place. Communication networks for Smart Grids are likely to consist of a multitude of legacy and new communication links, such as copper wires, optical fiber, wireless and power line communication [5], [13]. These networks will disseminate measurements and control commands to use synergies between different sectors, e.g., to optimize the interaction of renewable energy resources and electric vehicle chargers. In other words, monitoring and control applications covering large numbers of devices need to be connected to each other.

Recently, Internet protocols are being integrated in an increasing number of devices, including sensors and actuators necessary for Smart Grids. Additionally, IPv4 has started to be replaced by IPv6, due to the ubiquitous nature that the Internet has achieved, which led to a shortage of IPv4 addresses. Thus IPv6 is a natural candidate to form the basic protocol for a smart city communication network. High loss rates, low data rates and instability are specific to both wireless and power line networks and require routing protocols that address their characteristics. The IETF ROLL working group has designed an IPv6 routing protocol that satisfies the constraints and needs of Low Power and Lossy Networks, named RPL [18].

One of the key characteristics of RPL is its ability to adapt to unstable and changing environments. When a path becomes unusable, the protocol tries to find a suitable alternative. The mechanism coping with such environmental changes is called local repair. It allows a RPL node to attach to a different parent node to maintain connectivity, if possible. This self-healing procedure is defined in the protocol specification and behaves according to a set of parameter settings. These configurable parameters enable a deployment that is tailored to the application requirements and provides a trade-off between a fast reaction to changes and sluggish behavior to avoid unnecessary changes due to short-time variations in the channel properties.

While route switching latencies and global repair have been investigated in several studies, the recovery time of local repair in case of communication failures has not been examined much. For Smart Grid applications such as soft real-time status reporting from remote terminal units [1], communication outage times must be short (e.g. less than a minute). Hence, short recovery times are crucial to the applicability of RPL in the Smart Grid. In this paper we measure the *recovery time* from the time a link fails until the repair process has found a new valid route and we propose parameter settings and simple enhancement mechanisms to speed up the recovery process. Our mechanisms have been built into the RPL implementation of ContikiOS 2.5 [17], an open source operating system designed for the Internet of Things.

Contributions. This paper investigates the behavior of RPL in case of communication problems. Four simple mechanisms to minimize the recovery time are proposed and analyzed. A worst case scenario is studied with default parameter settings as a baseline. To shorten the time until messages can reach their destination again when a path becomes unavailable, first the influence of parameters determining the reactivity to network changes is analyzed and suitable settings are identified. Second, the benefit of measuring the link quality to all neighbors regularly is studied. Third, a more meticulous maintenance of the candidate list for the default route is devised. Finally, the introduction of a virtual sink for data collection traffic is proposed. The influence of each of these enhancements and their combination is then evaluated by simulations and recommendations for deployment are described.

II. RPL Overview

RPL is the IETF IPv6 routing standard for routing in Low-Power and Lossy Networks (LLNs) [18]. One of the main contributions of RPL lies in its flexibility to optimize paths for different requirements. First of all, the very creation of the main data structures has at its core an optimization function,

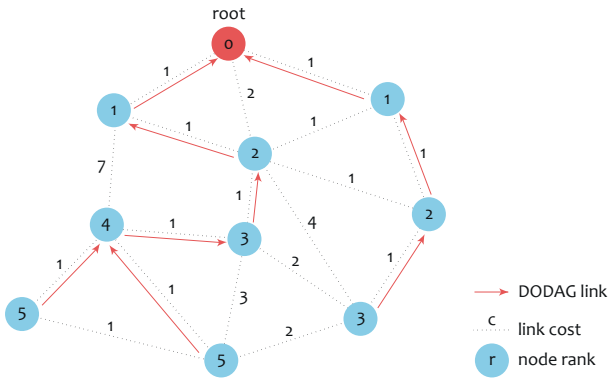

Fig. 1. DODAG example where the objective function minimizes the parent rank plus the link cost

which can be tailored for various application needs. Secondly, RPL control messages can be used to pass metrics information throughout the network, allowing optimization of the objective function at the node level.

The main elements of RPL architecture are instances and destination oriented directed acyclic graphs (DODAGs) i.e. DAGs with a single sink node (or root). A simple DODAG example is shown in Figure 1. A RPL instance consists of one (or more) DODAGs and each instance is associated to one routing objective, derived from one objective function (OF). The OF is used to establish rules on how nodes will route in order to satisfy the optimization objective. By exchanging ICMPv6 control messages, RPL enables nodes to distribute routing metrics in the network and to form DODAGs according to the OF rules. Each node joins one DODAG to have a path towards a sink (root) node.

RPL is designed to support Multipoint to Point, Point to Multipoint and Point to Point traffic flows. To achieve this, there are four types of control messages exchanged in a DODAG: DIOs (DODAG Information Object), used for DODAG discovery and maintenance, DISs (DODAG Information Solicitation Message), used to probe for DIO messages, DAOs (DODAG Destination Advertisement Object) used to propagate routing table entries towards the leaves, and DAO-ACKs, used to respond to the DAOs. More precisely, in the storing mode of point to multipoint RPL, a node chooses a preferred parent from its neighbors as a default route towards the sink according to the ranks received in so-called DODAG Information Object (DIO) messages and broadcasts DIOs containing its root, rank and metric information as well. To inform the nodes on the paths to its root (DODAG sink) a RPL node sends Destination Advertisement Object (DAO) messages to its preferred parent which adds an entry in its routing table and forwards the DAO in turn to its own preferred parent recursively.

A. Self-healing

One of the defining characteristics of Low Power and Lossy Networks is instability, often caused by link or node failures. RPL is a robust protocol, which was designed to consume few resources with traffic overhead and to be under-reactive to network changes. Since fluctuations in the network should not trigger DODAG oscillations, the system has certain inertia before deciding to change the underlying DODAGs. Self-healing was integrated in RPL's design through two mechanisms: global repair and local repair. When a global repair is performed, the whole DODAG is reconstructed from scratch. The inherent complexity of this operation makes it take up a considerable amount of time. Global repair can be triggered periodically by the root for maintenance purposes. On the other hand, local repair is meant to perform small, local changes. Nodes in the network can initiate local repair independently whenever their preferred parent is no longer deemed appropriate. In order to avoid oscillations, this mechanism was designed to be rather conservative as well. Consequently, local repair in its original form does not provide fast failure recovery. In the subsequent sections (Section III, Section IV), we will focus on techniques aimed at accelerating the local repair process whilst avoiding oscillatory behavior.

B. Link Metric Calculation

Routing metrics are quantifiable characteristics that are used by RPL nodes to determine the best paths for packets according to an OF. An optimal path is defined as a path in the DAG that minimizes (or maximizes, respectively) the Rank value (an abstract scalar value computed using the objective function with metric values as inputs) between any given pair of source-destination nodes. In the IETF RFC 6551 [7] several metrics and OFs are described. The original version of Contiki RPL, provides an implementation of the link metric ETX (expected number of transmissions), which is a default and mandatory metric for RPL. An objective function having ETX as the single metric will attempt to minimize the nodes' rank values based on the ETX values on the path to the root. ETX represents how often the MAC layer tries to transmit a packet until the transmission is successful. ETX is updated upon packet sending, based on the status of the transmission and the number of the attempts necessary for performing the transmission. Thus, implicitly, ETX includes a measure for congestion (i.e. if many nodes close by attempt to send a message at the same time, the value of ETX will be high, because the channel would be busy). Finally, in order to take into account previous values of ETX and to avoid large oscillations, the Contiki RPL implementation applies a first order low-pass filter when the metric is computed (i.e. $\overline{ETX}(t) = 0.9 \cdot \overline{ETX}(t-1) + 0.1 \cdot ETX(t)$, where ETX is the instantaneous value reported by the MAC and $\overline{ETX}$ is the smoothed value used for routing).

C. Parameters

RPL is a highly configurable protocol. Apart from objective functions and metrics, it is also possible to alter various other parameters within the system. For instance, the frequency of global repairs, of RPL control messages, which are sent periodically, default route lifetime, or the maximum number of potential parents are among the most relevant parameters.

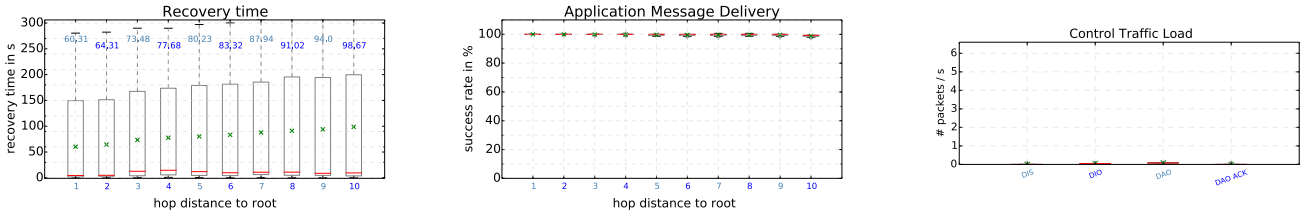

Fig. 2. Baseline of chain topology: boxplots of recovery time measurements (the average of each node is indicated at the top), delivery success probability of application layer messages and number of control traffic packets per second in chain network (see Section IV for a detailed scenario description). The maximum recovery time is 641.0s for the node 10 hops away from the root.

D. Recovery Baseline on Chain

Since our goal is to analyze the contributing factors of recovery without the influence of other mechanisms and events in the system, it is important to identify the most simple scenario that allows us to observe it. Otherwise, side effects and other mechanisms that could impact the recovery time would need to be considered as well.

To study local repair for multi-point-to-point traffic in detail, at least one alternative path to the destination must exist. In other words, in the most simple scenario at least one node v can find a new path if it cannot reach its root via the preferred parent anymore. The exact structure of the original DAG and of the new path do have an influence, but in its essence the local repair mechanism relies on i) the detection that the link to v's old preferred is not available or of too low quality and ii) the identification of a new suitable preferred parent for v. Thus it is sufficient to study a node v that can connect to two different parents to reach a sink node.

All the nodes that are further away from the root than v, which do not have an alternative path, have to wait until this node has established a new route before they can deliver messages successfully again. Thus the only difference between these nodes is the (hop) distance to v. The most simple scenario to study local repair is thus a chain network with two sinks. To observe both the behavior of v (node ID 1 in Figure 3) and of its children under moderate data traffic, we selected a chain topology containing 12 nodes: two sinks (nodes 11 and 12) and 10 non-sinks. The 10 non-sink nodes are placed in a chain, with the distance to two consecutive nodes large enough to ensure that a node in a chain would have exactly two neighbors. I.e., it can exchange messages only with the node just before it and the node right after it. To model a low-power and lossy network, the packet success rate of each link is set to a relatively small value, 64%. Due to MAC layer retransmissions the application layer message success probability is around 99%, even for messages that are passed along the whole chain.

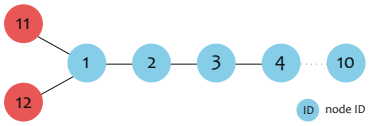

Fig. 3. 12-node Chain Topology

We define the *recovery time* of node x to comprise the time interval from the time of removing the link between node 1 and its root until it can deliver messages (now to the other root) again. We are mainly concerned with the recovery time of the node situated the closest to the roots, because it is the only one that has to perform a parent change. The recovery times of the other nodes are interesting to observe because they show the time it takes for the root change to propagate through the chain. Figure 2 depicts the measurements of the recovery times with the Contiki RPL default settings. Furthermore it shows the success probability of data messages sent if there are no communication problems in the network as well as the number of control messages emitted by all nodes per second. On average the number of bytes per second used for control traffic is 19.47 with a standard deviation of 1.37.

III. Recovery Enhancements

RPL self-healing is accomplished with global and local repair mechanisms, as presented in Section II. In this section, we discuss approaches for a *local repair speed-up*.

A. Tuning DIO Settings

A key component of RPL's local repair mechanism is the maintenance of upward routes. Nodes in a DODAG keep these routes up-to-date through the exchange of DIO messages, which are periodically broadcast link locally (only to neighbors situated one-hop away). The frequency of DIO broadcast is controlled by a trickle timer [15], initialized with a minimum time interval. When the timer goes off, if no preferred parent changes or DODAG changes occurred for the concerned node, the DIO broadcast interval doubles. Clearly, the minimum transmission interval and the maximum allowed number of doublings have a considerable influence on system recovery time: receiving DIOs more often allows nodes to become aware of failures more rapidly. However, frequent DIOs increase control traffic overhead, causing data messages to be dropped. Message loss occurs especially at nodes close to DODAG roots in multipoint-to-point traffic flows.

B. Probing

As presented in Section II, the current path metrics in our system are updated upon sending data or control messages. Therefore, in the multipoint-to-point traffic flow scenario, only the state of the link from a node to its preferred parent is updated, while links to potential parents remain in a stale state. For example, in Figure 4, suppose A

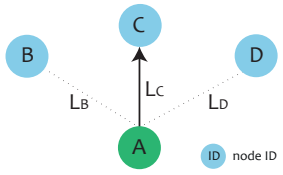

Fig. 4. Links L_B and L_D are not updated

chooses C to be its preferred parent. As a consequence, the ETX value corresponding to L_C is updated every time A sends a control or a data message to C, keeping L_C's state updated (as described in Section II-B). However, once A chooses its preferred parent, it does not send anything to B or D, causing L_B and L_D to be seen as having a stale (and most of the time high) ETX value. Because this occurs at initialization, A could be stuck with its first choice of preferred parent C (since it sees all the other links as poor) until it explicitly receives a No Path DAO, even if B or D could have become better options.

Being aware of the state of their surroundings is essential for all the nodes in the DODAG, enabling them to make sensible decisions when choosing their preferred parents. In order for the nodes to explore all available links, we have set up a probing system on top of RPL: periodically, a small message is sent by every node to all of its potential parents. The probes' purpose is to trigger the update of link metrics (ETX) for neighboring links. Probing is a simple and effective solution for providing more flexibility when choosing preferred parents. On the other hand, similarly to the previous subsection, one should consider the probe sending time interval, in order to avoid excessive control traffic overhead.

C. Parent List Purging

As discussed in the previous subsection, probing all potential parents can generate a substantial amount of control traffic. Therefore, the preferred parents list should not contain elements to which nodes would most likely not connect. In addition, a carefully selected parent list could help avoid loop-formation in the DODAG, which is directly linked to improved failure recovery times. Therefore, we have introduced a mechanism that periodically applies two filters to the nodes' parent lists. Firstly, it ensures locally that the children and the parent sets of a node are disjoint, since it would not be wise for nodes to connect to one of their children when their preferred parent becomes unsuitable. More precisely, all nodes for which a routing table entry exists due to DAOs are eliminated from the parent candidate list. In addition, we introduced a route expiry time for the entries in the routing table holding routes towards the children, to make sure that they stay fresh. Therefore, if one of the children becomes a suitable parent, its entry in the routing table will eventually expire, allowing the node to be a candidate for the parent set.

In the second filter, nodes whose rank is greater than the rank of the current node are removed from the parent list as well. This is done to enforce Rule 5, Section 8.2.1 of [18], which guarantees that nodes will only keep potential parents that are better placed in the DODAG than themselves. This is not currently enforced in the implementation of ContikiRPL. The periodical purging of the parent-lists was integrated with the already existing RPL periodic maintenance operations and does not induce any supplementary messages. It decreases the system-level complexity (by decreasing probe-related control traffic and loop forming), at the cost of introducing extra sequential complexity at node level.

D. Virtual Sinks

The last mechanism introduced in our system was inspired by anycast routing principles. Consider the following scenario, illustrated in Figure 5. B has A as its preferred parent, and A is one hop away from R_1, one of the two sinks. Both A and B are a part of R_1's DAG. Now, A chooses to switch to R_2, the other sink. However, it can happen that the message in which it announces its new DAG gets lost. If B sends a message destined to R_1 (because it thinks it is part of R_1's DAG), it will get forwarded upwards by A and it will reach R_2. Finally, R_2 will discard the message, because it was destined to R_1. This is not the desired behavior, since we want to use all data that reaches the roots; i.e., on an abstract level, we view all sinks as one unified sink, with different spatial representatives in the network. Hence we introduce virtual sinks, i.e. all sinks (i.e. DODAG roots) were enhanced with a common virtual IP address. Likewise, instead of sending messages to specific sinks, non-sink nodes make use of the unique virtual address.

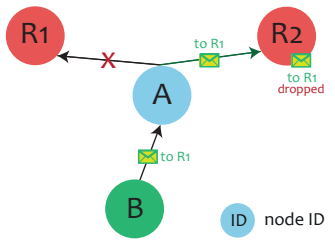

Fig. 5. R_2 does not process B's message because it was destined to R_1

The virtual sink address increases the overall message success rate, as fewer messages are lost during DODAG switches.

IV. Evaluation

In this section we present node recovery-time simulation results for the chain topology introduced in Section II-D. We showcase two approaches for local repair speed-up, described in Section III. The first approach is based on tuning already existing RPL DIO parameters. The second approach explores the effects of probing, parent list purging and virtual sinks. We will look at these three mechanisms in detail, evaluating each of them separately and collectively. Finally, we provide a comparison of the two approaches inSection IV-C.

Our implementation was built on top of ContikiRPL 2.5 [17] and the experiments were run in the COOJA simulator [12]. The data transmission rate was set to a relatively low value of 100 kb/s to mimic challenging conditions regarding traffic load or low-rate media. Every non-root node in the chain generates a data message (of 64 bytes) once every 10 seconds. Each scenario has been simulated 50 times, with distinct random seeds that influence the start up phase and CSMA back-off intervals. The DIO timers were set as follows: a minimum interval of approximately 4 seconds (2^{12} ms) and 8 allowed interval doublings, corresponding to the default RPL configuration and a minimum interval of approximately 2 seconds (2^{11} ms) and 2 doublings, corresponding to a tuned version of the DIO timer. When the probing mechanism is activated, probes are sent every 4s. After a stabilization period of 300s that allows the nodes to connect to a DODAG and reach large trickle time intervals, the root to which the nodes are connected stops communicating. The recovery time until

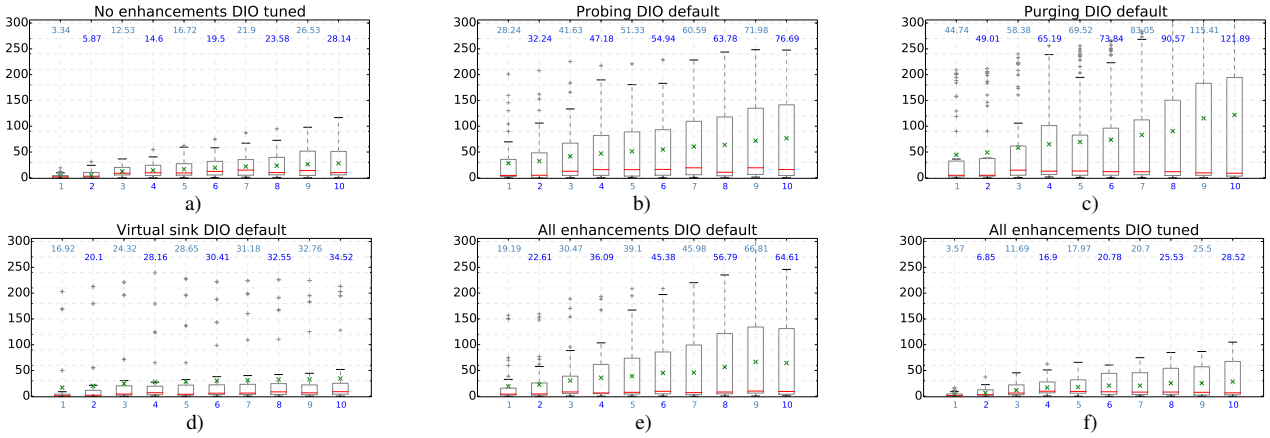

Fig. 6. Recovery time measurements in s for DIO Tuning, Probing, Purging, Virtual Sink, all enhancements combined with default and tuned DIO settings.

a node is able to send messages to the root successfully is measured for each node. Furthermore, the number of control traffic packets is counted and the success rate of application data messages under stable conditions is stored.

A. Tuning DIO Settings

Comparing the recovery time plots with and without DIO tuning (Figures 2 and 6.a) shows that reducing the minimum time interval to 2 seconds and allowing at most two doublings leads to a large cut in the recovery time. In particular, the first node can choose a different parent in about 3 seconds on average compared to roughly 60s without DIO tuning. For nodes which are situated further from the failure point (in our case the root), the parent switch is propagated starting from node 1, hop by hop, reaching node 10 in less than a third of the original time. The tuned average number of DIOs sent per second is about 3.2 for the whole network compared to roughly 0.4 with the default settings (Figure 7). The average number of control bytes sent per second in this setting is 512.87, with a standard deviation of 2.43. The average success rate still exceeds 90% for all nodes, but we can see that the deviation increases and that increased control traffic leads to more collisions and hence slightly lower application layer message success rates. We conducted some further experiments with different DIO settings. Restricting the number of doublings even more does not decrease the recovery time significantly, but leads to more overhead even in ideal conditions. Reducing the minimum interval length to 1 second does help in this scenario, but as soon as the traffic load is slightly increased or if the data rate is decreased, this adjustment leads to too many collisions and thus a lower success rate. Therefore, our recommended parameter settings for fast failure recovery are 2 and 8 seconds for the minimum and maximum interval respectively. Clearly this has to be checked for every deployment and may need to be modified for different applications and requirements.

B. Single Mechanisms on Chain

1) Probing (Fig. 6.b): Sending a probing message to all neighbors regularly and thus having a good estimate on (asymmetric) link qualities is able to shorten the recovery time significantly too. For nodes close to the point of failure, the speed up is in the order of a factor of two. Nodes further away are updated more slowly, because of the default DIO settings, which prevent the parent change to propagate to far away nodes quickly. Since the probing messages are much shorter than the DIOs, they occupy less of the bandwidth and thus allow a better utilization of the channel for data traffic. On the other hand, the time until nodes far away from the failure have been updated takes longer, since this depends mostly on the DIOs. By varying the time interval between subsequent probes, one can observe a trade-off between the recovery time of nodes close to a failure and the amount of the control traffic.

2) Purging (Fig. 6.c): Cleaning the candidate parent list and avoiding to connect to your children does not improve the recovery time in chain networks, if used on its own. However, when combined with probing and/or virtual sinks other mechanisms, it is able to reduce the recovery time significantly (Figure 6.e).

3) Virtual Sink (Fig. 6.d): As expected virtual sinks do not help the nodes that are very close to the failure. However, for nodes further away they do decrease the median recovery time significantly when combined with probing. Moreover, they help increase the success rate rapidly and allow messages to be delivered to a sink, even when the DODAG root is changed.

C. Tuning DIO Settings vs. All Enhancements

Not surprisingly we obtain the best results with the default DIO setting when combining all three mechanisms, see Fig. 6.e). The first hop node can recover in less than 20 seconds, while the scenario with default settings and no enhancements used more than 60 seconds. The averages of the other nodes are also much lower. While we do not exhibit the plots of all the combinations of two enhancements, we want to emphasize that the benefit of using two of them is lower than applying all three. Compared to the scenario with tuned DIO parameters, the first few nodes profit more from the enhancements, while the nodes further away need frequent DIOs to react quickly. On the other hand, the control traffic overhead mostly dues to the higher DIO rate can be prohibitive

and does influence the success rate, as can be seen in Fig. 7. Tuned DIO settings together with all enhancements does not

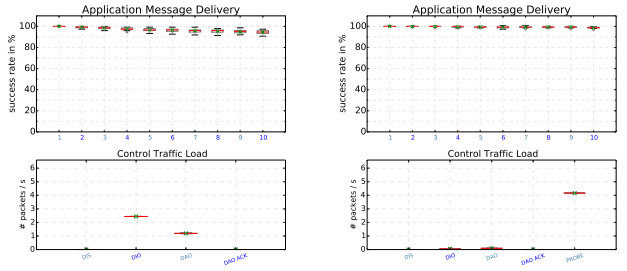

Fig. 7. Data packet delivery rate(*top*), number of messages emitted per second for each control traffic type (*bottom*) with DIO tuning (*left*) or applying all enhancements (*right*). The average number of control traffic bytes per second is 512.87 (2.43 std) with tuned DIO settings, while applying all enhancements results in an average of 204.05 (1.44 std).

lead to another reduction of the recovery time, see Fig. 6.f). This is mostly due to the fact that the control traffic is around 660 bytes per second in this case. This demonstrates that a careful analysis of the scenarios before a deployment can help to find good parameter settings.

V. Related Work

Numerous studies on the performance of the RPL protocol in general have been conducted. Some of them focus on the performance of RPL on (simulated) wireless networks [4], [6], [16] and power line communication [2], [3]. Others study RPL's implementations interoperability [10], the performance when nodes move [9], as well as multipath extensions [14].

Up to now, to the best of our knowledge, an in-depth analysis of the failure recovery speed of RPL's local repair mechanism has not yet been performed. In an extensive performance evaluation of RPL, Tripathi, Oliveira and Vasseur touch upon the problem of connectivity loss and the associated recovery time [16]. However, the local repair mechanism is inhibited and only global repair is considered. Our work complements this study, focusing on local repair. Moreover, Kermajani and Gomez examine the route change latency in the context of the 6LoWPAN neighbor discovery protocol, but do not show a direct relationship between the route change latency and RPL specific mechanisms, as we will exhibit in the following sections [8]. Another analysis of the RPL repair process, using ContikiRPL was conducted by Korte et al [11]. An important point presented in their work is the apparent independence of failure recovery times and DIO timers. In contrast, our study shows that the recovery time is closely linked to the DIO sending frequency. The discrepancy between our conclusions may come from the experimental setup (more precisely, the point in time when the node was switched off) and the fact that only the DIO timer minimum interval was varied in [11]. If the nodes are switched off when the DODAG has stabilized, DIOs are sent rarely even if the DIO timer interval was small initially, because of the trickle mechanism. Thus, the recovery time in [11] is not only affected by the DIO sending frequency.

VI. Conclusion

We investigated the recovery of RPL from communication failures, identified the influencing factors and proposed possible ways for improvement. More specifically, apart from choosing suitable parameter settings, we devised and analysed three enhancement mechanisms. Our findings show that there is a tradeoff between the simplicity of changing the DIO settings and amount of control traffic (but keep in mind that the three enhancements are also easy to implement) as illustrated in Fig. 7. If a deployment features long paths it is better to tune the DIO timer parameters while short paths benefit greatly from the other enhancements. By combining all enhancements one can achieve fast recovery times for all nodes. The improved recovery times by our proposed enhancements were verified by further simulations of RPL deployments in realistic medium voltage power line communication scenarios such as those described in [1]. As a simulation cannot replace a real deployment these results may not be accurate in practice but they serve as a first indication for the suitability of RPL for smart grid applications. Moreover they can help to choose suitable parameter configurations and enhancements.

References

[1] O. Balmau, D. Dzung, K. Karaagac, V. Nesovic, A. Paunovic, Y. A. Pignolet, and N. Tehrani. Evaluation of RPL for Medium Voltage Power Line Communication. *IEEE SmartGridComm 2014*.

[2] L. Ben Saad, C. Chauvenet, B. Tourancheau, et al. Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: Two Cases Studies. In *SENSORCOMM*, 2011.

[3] C. Chauvenet, B. Tourancheau, D. Genon-Catalot, P. Goudet, and M. Pouillot. A Communication Stack over PLC for Multi Physical Layer IPv6 Networking. In *Smart Grid Communications*, 2010.

[4] T. Clausen and U. Herberg. Comparative study of rpl-enabled optimized broadcast in wireless sensor networks. In *ISSNIP*, pages 7–12, 2010.

[5] S. Galli, A. Scaglione, and Z. Wang. Power Line Communications and the Smart Grid. In *IEEE SmartGridComm*, 2010.

[6] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, A. Terzis. Evaluating the performance of rpl and 6lowpan in tinyos. In *IPSN*, 2011.

[7] JP. Vasseur et al. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. *IETF RFC 6551*, 2012.

[8] H. R. Kermajani and C. Gomez. Route change latency in low-power and lossy wireless networks using rpl and 6lowpan neighbor discovery. In *ISCC*, 2011.

[9] Kevin C. Lee et al. Rpl under mobility. In *CCNC*, 2012.

[10] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, A. Terzis, A. Dunkels, and D. Culler. ContikiRPL and TinyRPL: Happy Together. In *IPSN*, 2011.

[11] K. D. Korte, A. Sehgal, and J. Schönwälder. A study of the rpl repair process using contikirpl. In *Dependable Networks and Services*, 2012.

[12] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, IEEE Conference on*, 2006.

[13] A. Patel, J. Aparicio, N. Tas, M. Loiacono, and J. Rosca. Assessing communications technology options for smart grid applications. In *IEEE SmartGridComm*, 2011.

[14] B. Pavković, F. Theoleyre, and A. Duda. Multipath opportunistic rpl routing over ieee 802.15. 4. In *Modeling, analysis and simulation of wireless and mobile systems*, 2011.

[15] Ph. Levis et al. RFC 6206: The Trickle Algorithm. *IETF*, 2011.

[16] J. Tripathi, J. de Oliveira, and J. Vasseur. A performance evaluation study of RPL: Routing Protocol for Low power and Lossy Networks. In *CISS*, 2010.

[17] N. Tsiftes, J. Eriksson, and A. Dunkels. Low-Power Wireless IPv6 Routing with ContikiRPL. In *IPSN*, 2010.

[18] T. Winter, P. Thubert, A. Brandt, T. H. Clausen, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. *IETF RFC 6550*, 2012.