On Designing Provably Correct DODAG Formation Criteria for the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)

Agnieszka Paszkowska
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw, Poland
E-mail: ap321142@students.mimuw.edu.pl

Konrad Iwanicki
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw, Poland
E-mail: iwanicki@mimuw.edu.pl

Abstract—**Apart from being standardized, an important industrial advantage of the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is the possibilities of completely adapting its route formation criteria. Through largely open routing metrics and route selection delegated to so-called objective functions, RPL's adopters are free to customize the protocol to individual applications, even ones with peculiar requirements. However, these possibilities also pose a major reliability risk: it is not clear whether a given combination of routing metrics, objective function, and RPL's remaining configuration parameters ensures that the resulting routes will be formed correctly when the routing metric values change dynamically. We study this problem here to derive conditions for routing metrics and objective functions under which routing path construction and maintenance can be proved correct. We start with a case study, based on an industrial deployment, that motivates the considered problem. We then formally model and analyze the dynamic behavior of RPL's route formation process in abstraction from particular objective functions and routing metrics. We derive conditions under which this process is provably correct. Finally, we attempt to translate these theoretical conditions into practical guidelines that can be utilized by RPL's adopters who aim at reliable systems.**

I. Introduction

The IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) [1] aspires to be a practical solution for industry. On the one hand, being standardized by IETF, it aims to facilitate interoperability and system composition. On the other hand, leaving a number of issues open, it enables per-application customizations. The latter possibility is particularly important as each real-world embedded system has its own application requirements, which are reflected in the design and optimization of its various networking stack components [2].

One of such customizable features, which has received a lot of attention from RPL's designers, is routing path formation. More specifically, routing paths in RPL are built so as to globally optimize an application-dependent *objective function*, computed over application-dependent node or link *routing metrics*. Two objective functions are in use today, OF0 [3] and MRHOF [4], and they seem to fulfill the requirements of industry. Nevertheless, developing a new one, tailored to a particular application need not be straightforward, as the specification of such a function leaves a lot of freedom for implementers. What is more, for routing metrics, RPL's design is deliberately even more general so as to accommodate as many conceivable ones as possible [5]. At the same time, only two metrics are supported by the protocol's popular implementations: hop count and estimated transmission count (ETX). To be exact, ContikiRPL provides serialization stubs also for a node energy metric but offers no implementation thereof. TinyRPL, in turn, does not even contain such stubs.

This situation is problematic for industrial applications that require different route formation criteria. The lack of mature implementations of many common metrics identified for industrial applications [5] forces RPL's adopters to develop such implementations on their own. However, this is not straightforward because of the protocol's aforementioned underspecification. Moreover, even if the implementations were available, employing them could be challenging as to work correctly, different metrics may require different objective functions and modifications to RPL's remaining configuration parameters. In other words, without guarantees that a particular combination of the protocol's settings, objective function, and routing metrics does not impair the correctness of route formation, employing RPL in real-world systems with custom route formation criteria may pose a considerable risk.

To derive solutions that can help managing such risks, in this paper we analyze RPL's route formation process under an abstract objective function and routing metrics. After giving a short background on RPL and surveying related work (Sect. II), to further motivate our research, we present a case study based on a past industrial delpoyment for which we considered an intuitive node energy metric instead of the default ETX (Sect. III). We show that such a simple metric leads to an emergent behavior resulting in gradual deformation and permanent disconnection of routing paths. To understand and prevent such behaviors we model the dynamics of RPL's routing path formation, based directly on the protocol's specification and in abstraction from particular objective functions and routing metrics (Sect. IV). Given the model, we try to formally ascertain whether RPL is guaranteed to build and maintain routing paths in a provably correct manner and what formal requirements routing metrics, objective functions, and the protocol's overall configuration have to satisfy for this to be true (Sect. V). Finally, we discuss how the derived theoretical requirements can be applied in practice by RPL's adopters, such as, for instance, in the previous case study (Sect. VI).

II. Background and Related Work

In RPL, the possible routing paths in a network of low-power wireless nodes form a so-called *DODAG* (destination-oriented directed acyclic graph), which at its sink normally has a more powerful border router node connecting the network to the Internet, the so-called DODAG *root* (see Fig. 1). To explain, each node keeps track of the wireless links to other nodes in its radio range: its *neighbors*. The maintained links are required to be symmetric. Some of them are selected to form routing paths to the root node. To this end, each node is given a *rank* that describes a cost of reaching the root from this node: the root itself has the lowest rank, and the further away a node is from the root, the higher its rank. The neighbors with lower ranks than the node's own one are the node's *parents*: forwarding a packet to a parent brings the packet closer to the root. Normally, however, a node forwards all packets to a single *preferred parent*. Globally, the nodes' links to preferred parents thus form a directed tree within the DODAG.

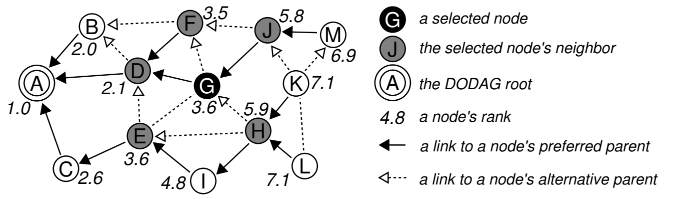
A
1.0
B
2.0
C
2.6
D
2.1
E
3.6
F
3.5
G
3.6
H
5.9
I
4.8
J
5.8
K
7.1
L
7.1
M
6.9
a selected node
the selected node's neighbor
the DODAG root
4.8
a node's rank
a link to a node's preferred parent
a link to a node's alternative parent

Fig. 1. An example of a DODAG.

The preferred parent and rank selection process is largely delegated by RPL to an application-dependent objective function that can utilize application-dependent routing metrics for nodes or links. The nodes assign themselves ranks so as to globally minimize the values of the objective function computed over the selected routing metrics. More specifically, a node's *potential rank* for a given parent is computed by the objective function based on the parent's rank and, optionally, the metric(s) of the link between the node and the parent, and the node's own metric(s). As its preferred parent, the node normally selects the parent that offers it the lowest potential rank, which then becomes the node's actual rank, but selecting the best parent is not obligatory. For instance, MRHOF involves *hysteresis* to limit control traffic [4]: a new parent is adopted only if the rank it offers is better at least by a configurable threshold than the node's present rank.

In contrast, what RPL's core specification does not delegate is the mechanisms for exchanging rank and metric values between neighbors. In essence, each node systematically broadcasts so-called DIO control messages to its neighbors. They contain its rank and other configuration parameters. A neighboring node receiving such a DIO records or updates an entry for the DIO sender in its local *neighbor set*. It then passes the set to the objective function, which selects its preferred parent and rank. The parent and rank are reselected whenever the neighbor set changes, notably when the routing metrics change. The details, including other messages (DISes, DAOs, and DAO-ACKs) not influencing the correctness of DODAG maintenance, can be found in RPL's specification [1].

Kim et al. [2], in turn, provide an extensive, up-to-date survey of the volume of past work on RPL. In particular, the impact of the two standardized objective functions on the protocol's performance was explicitly evaluated by Brachman [6] and Iova et al. [7]. In contrast, Gaddour et al. [8] designed a new objective function that features parent selection based on fuzzy logic over multiple routing metrics. Novel metrics were in turn proposed by Kulkarni et al. [9], Landsiedel et al. [10], Thulasiraman [11], Kim et al. [12], and Iova et al. [13]. However, the performance of the vast majority of those new solutions was by and large evaluated experimentally; it has not been formally proved that they ensure the correctness of DODAG formation in the presence of changes to the actual routing metric values. In contrast, this paper focuses explicitly on provable guarantees for DODAG construction and maintenance under such routing metric dynamics, yet in abstraction from particular objective functions and routing metrics. As such, not only does it complement the past work but may also inspire novel objective functions and routing metrics that address specific industrial needs and, at the same time, guarantee a provably correct behavior. Such provable guarantees are important for reliable industrial applications, in particular, in the face of our recent findings demonstrating abnormal behaviors of RPL's state-of-the-art implementations under various types of network dynamics [14], [15], [16].

III. Motivating Case Study

Our research has been inspired by an industrial deployment, the experiences from which we compile here into an illustrative case study. To this end, consider a multi-hop network of battery-powered nodes. To prolong the network lifetime, such a system can employ the following node routing metric: the percentage of a node's energy budget already consumed (EPC). Accordingly, a node's rank can be defined as the sum of the node's preferred parent's rank and the node's own EPC value (appropriately increased if necessary to satisfy RPL's requirement for the rank step, `MinHopRankIncrease`, as explained in Sect. IV). Moreover, when selecting its preferred parent, a node considers only those parents with which it has sufficient-quality links (i.e., links with ETX below a threshold). Finally, as the objective function, let us assume MRHOF with an arbitrary (e.g., default) parent switch threshold.

Such a configuration should naturally prolong the system's lifetime. Globally, it tends to prefer routing paths with fewer hops and involving nodes that have consumed less energy than others. At the same time, the paths consist of links of a sufficient quality. Together, these properties thus aim to minimize and balance the energy spent when routing packets.

To experimentally evaluate the performance of this configuration, we implement it for the freshest version of TinyRPL with fixes from our earlier work [15]. We conduct the experiments in the TinyOS simulator, TOSSIM, in 11×11 grid networks with unit-disk connectivity. The choice of simulation, the network topology, and the connectivity model are deliberate so as to convince the reader that the observed behavior is due to RPL alone and not some uncontrollable external factors, such as unstable, low-quality links or heavy noise. Nevertheless, the same behavior occurs in the real world.

Similarly, we adopt a simple model of node energy consumption, in which all nodes have the same battery lifetime of 96h and the resolution of EPC is 1% (rounded up). In other words, each node's initial EPC is equal to 1% and grows by 1% approximately every hour. Although a node's energy consumption is in practice typically much lower, not linear in time, and depends on the placement of the node in the network, we have selected such a simple model here only for the ease of result explanation. Again, however, the behavior we illustrate with the model is also inherent in the real world.

The experimental application running at each node on top of RPL emulates a typical data collection application. It repeatedly generates a one-frame UDP packet to be routed in the network to the DODAG root. The inter-packet interval is selected at random between T and $2T$ time units, where T = 10s. The link layer underlying RPL, in turn, does not utilize any duty cycling. Nevertheless, like previously, the observed behavior does not differ when duty cycling is used.

Fig. 2 presents the first 8h of two representative 96h runs of the application with two different metrics: ETX and EPC. The top and middle plots show the evolution of the number of nodes with non-null preferred parents (i.e., nodes capable of packet routing): the top plot—for the ETX metric; the middle plot—for the EPC metric. The bottom plot shows for reference the energy consumed by the root's neighbor in any run.

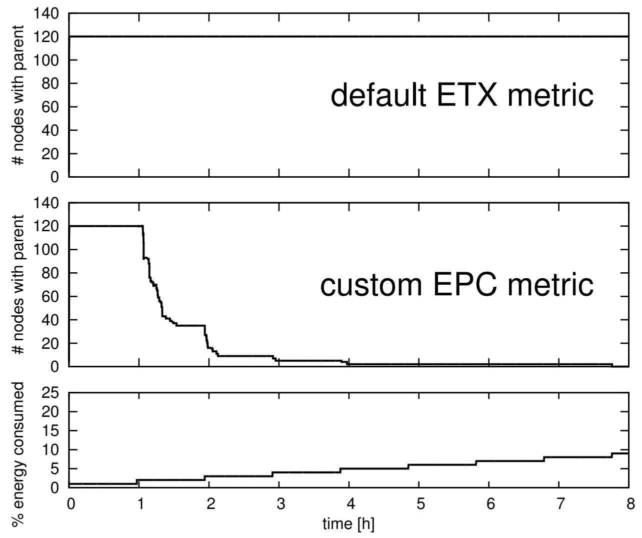

Fig. 2. A sample run illustrating an emergent behavior of the EPC metric.

With the default ETX metric (the top plot), the DODAG is formed within seconds and remains connected for the entire duration of the experiment: all 120 non-root nodes have their preferred parents non-null. With the EPC metric (the middle plot), initially, the DODAG behaves in the same way. However, as soon as the nodes have consumed an additional percent of their energy budgets (cf. the bottom plot), a large fraction of them forever loses their parents. The same situation occurs repeatedly, until the nodes have utilized 9% of their energy, starting from which none of them ever has a non-null preferred parent again. This means that despite having 91% of their energy available, the nodes are unable to route any data to the DODAG root: the network is useless. As mentioned previously, the same behavior can be observed in the real world, which suggests that it is not a simulation artifact.

IV. DODAG Formation Model

This emergent behavior can thus be due to an implementation bug or some features of RPL's design itself. To ascertain which is the case, we model RPL—based directly on its specification and in abstraction from particular objective functions and routing metrics—so that, in the next section, we can formally analyze its DODAG construction and maintenance dynamics. More specifically, we adapt our recent approach, used for proving RPL's behavior under network partitions [15] to make it support custom objective functions and routing metrics. As the underlying formalism, our approach employs linear temporal logic (LTL) [17], albeit in a textual form to facilitate presentation. For the same reason, the modeling and analysis process originally involved several iterations, so that the version presented here is succinct, yet accurate (i.e., not oversimplified), and hence illustrates the major conclusions.

A. System State

We model a RPL network as a graph of nodes connected with bidirectional links. The links may exhibit packet loss, duplication, and reordering, as we explain shortly. The nodes, in turn, run a program that, among others, incorporates RPL.

To focus our reasoning on DODAG construction and maintenance, we abstract RPL's state at a node into the following variables: *prefpar*, *rank*, *minrank*, *metrics*, and *neighborset*. *Prefpar* is the identifier of the node's preferred parent or *null* if the node does not have one. *Rank* is a positive integer equal to the node's rank or infinity if the node lacks a rank yet. *Minrank* is the minimal rank the node has ever had or infinity as for *rank*. To be precise, *minrank* is the node's minimal rank in a so-called DODAG *version* [1]. However, ordering a version change is outside RPL's specification and corresponds to a complete DODAG reconstruction. In contrast, to study DODAG formation, we need to consider only a single version. Therefore, although we did study DODAG version changes, they need not be modeled here. *Metrics* is in turn the node's local routing metric values. Importantly, we treat them "a black box" to account for the fact that they can be arbitrary entities. Finally, *neighborset* is a set of tuples, each corresponding to the node's neighbor and containing at least the following fields: *id*, *rank*, and *metrics*. Fields *id* and *rank* contain respectively the neighbor's identifier and rank (as known to the node, which may differ from the neighbor's current `rank`). Field `metrics` describes in turn the routing metric values for the neighbor and/or the link with the neighbor. These metrics complement the node's local metrics when it computes its `rank`. Again, we treat them as "a black box." To avoid confusion between the fields of a node's `neighborset` entry, `n`, and the node's local variable with the same name, we can prefix the variable name with `self`, for example, `n.metrics` denotes the metric values associated with the node's `neighborset` entry `n`, whereas `self.metrics` denotes the node's own local metric values.

We also model packets in transit, that is, on air or in communication queues anywhere in the network stack. More specifically, for the analysis of DODAG formation, we need to consider only packets carrying DIO messages. Each such packet contains at least fields `id` and `rank`, representing respectively the sender's identifier and rank (at the moment of sending the packet). Packets in transit over a link form a multi-set associated with the link: a packet may appear more than once in the multi-set, which allows us to model duplication.

B. System Properties

To sum up, the global system state is the values of all nodes' variables and all packets in transit. It is thus well defined formally, and we can formulate various properties describing it, such as "node *A*'s `rank` is greater than node *B*'s `rank`."

RPL's dynamic operation is in turn modeled as LTL's *computation*: an infinite sequence of global system states representing the flow of time. Given this, we can express properties that may be true only at some moments in time by using LTL's temporal operators: *always/never* and *eventually*. More specifically, property "always/never ϕ" holds in state i of a computation if and only if (*iff*) property "ϕ" holds in all/no states $j \geq i$ of the computation. By symmetry, property "eventually ϕ" holds in state i of a computation iff there exists some state $j \geq i$ in which property "ϕ" holds. A property is said to be satisfied for a computation iff it holds in the first state of the computation. For example, "node *A*'s `rank` is eventually always finite" means that, starting from some moment in time forever, node *A*'s `rank` has only finite values. In contrast, "node *B*'s `rank` is always eventually finite" means that for every moment, *B*'s `rank` is finite at this or some future moment. Yet, such moments may interleave with ones in which the `rank` is infinite. In other words, *B*'s `rank` is repeatedly finite but not necessarily permanently.

Note that such properties formulated in LTL do not specify precise timing. This is deliberate, though, as RPL is not a real-time protocol, and thus the correctness of its DODAG formation process must not be affected by the actual timings.

To describe the various facets of RPL's dynamic operation in terms of the model, we have formulated a number of properties—based directly on the protocol's specification. As mentioned previously, for brevity, here we present the final minimal self-contained subset of those properties that is sufficient for analyzing the considered problem.

The first family of the properties (properties OF1–OF5) describes RPL's rules for rank and parent selection, which must be obeyed by any objective function. The positive constant `MinHopRankIncrease` is the minimal delta between a node's and its preferred parent's `rank`s, which aims to prevent routing loops; it is also the DODAG root's `rank`. The constant `MaxRankIncrease` is in turn the maximal tolerated growth of a node's `rank` in the DODAG (actually, its version, similarly to the definition of `minrank`).

OF1. *A node's* `prefpar` *and* `rank` *change only as a result of reselection; otherwise, they remain unmodified.*

OF2. *A node's* `minrank` *is always equal to the minimal value of the node's* `rank` *so far.*

OF3. *Always, when reselecting its* `prefpar` *and* `rank`*, the root node adopts* `null` *and* `MinHopRankIncrease`*, respectively. These are also the initial values of the two variables at the root node when the node starts.*

OF4. *Always, when reselecting its* `prefpar` *and* `rank`*, a non-root node adopts* `null` *as* `prefpar` *iff it also adopts infinite* `rank`*. These are also the initial values of the two variables at the non-root node when the node starts.*

OF5. *Always, when reselecting its* `prefpar` *and* `rank`*, a non-root node adopts* `null` *and infinity, respectively, iff its* `neighborset` *contains no entry,* `n`*, for which a potential rank,* `r` = $\mathcal{R}$(`n.rank`, `n.metrics`, `self.metrics`)*, can be computed so that* `n` *and* `r` *satisfy all following constraints:*
(a) `n.rank` $<$ *infinity,*
(b) `r` $<$ *infinity,*
(c) `r` $\geq$ `n.rank` $+$ `MinHopRankIncrease`*,*
(d) `r` $\leq$ `minrank` $+$ `MaxRankIncrease`*.*
Otherwise, the node adopts `n.id` *and* `r` *as its* `prefpar` *and* `rank`*, respectively, for some* `n` *satisfying all cond. (a)–(d).*

The second family of properties (properties DIO1–DIO3) corresponds to the delivery guarantees for DIO messages. The properties imply no DIO fabrication (DIO1) and an arbitrary, albeit finite loss (DIO2) and duplication (DIO3). As such, they are extremely permissive to avoid oversimplifying the intricacies of real-world low-power wireless communication.

DIO1. *If a node receives a DIO message,* `d`*, then the message must have been sent earlier by the node's neighbor:* `d.id` *equals the neighbor's identifier and* `d.rank` *equals the neighbor's* `rank` *from the moment of sending* `d`*.*

DIO2. *A node always eventually receives a DIO message from each of its neighbors.*

DIO3. *If a node's neighbor sends a DIO message, then the node eventually never receives the message.*

The third family of properties (properties RA1–RA3) formalizes the management of the nodes' `neighborset`s. They state that each modification to a node's set causes `prefpar` and `rank` reselection (RA1) and specify how the set is modified (RA2 and RA3). Again, the properties aim not to oversimplify the reality.

RA1. *Always, if a node's `neighborset` changes, then the node eventually reselects its `prefpar` and `rank`.*

RA2. *If a node always eventually receives a DIO from a neighbor, then an entry for the neighbor is eventually always present in the node's `neighborset`.*

RA3. *Always, for each entry, `n`, in a node's `neighborset`, `n.rank` is equal to `d.rank`, where `d` is the last DIO message with `d.id` equal to `n.id` received by the node since the entry was added to the `neighborset`, or is infinite, if the node has received no such message yet.*

V. DODAG Formation Analysis

The presented model formalizes RPL's operation based on the specification, together with some additional properties on which the protocol relies (e.g., the guarantees on DIO message reception). Among others, it illustrates our previous argument that RPL's specification is largely open when it comes to objective functions and routing metrics. Therefore, to analyze what additional conditions the functions and the metrics should satisfy to guarantee the correctness of the DODAG construction and maintenance process, we will attempt to prove the correctness of this process given the current suite of properties. Any extra properties that we will add to successfully complete the proof will thus represent the additional conditions.

Drawing from the case study, our goal is to prove that every non-root node eventually always has its `prefpar` non-`null`, so that it can forward packets. Because of space constraints in this paper, we do not prove that the packets will actually reach the DODAG root, which can also be done but requires extending the model and adding assumptions. From properties OF1 and OF4, a non-root node has its `prefpar` non-`null` iff its `rank` is finite. From OF1 and OF3, in turn, the root node's `prefpar` is always `null` whereas its `rank` is always finite. Therefore, for uniformity, we consider `rank`s instead of `prefpar`s. More specifically, we will try to prove Hypothesis 1, which is a formalization of those aspects of the nodes' ability to construct and maintain a DODAG that are relevant to reaching the aforementioned goal.

Hypothesis 1. *All nodes eventually always have their `ranks` finite.*

A. Basic Insights

An intuitive way of proving Hypothesis 1 would be induction on a node's hop count from the DODAG root. With its hop count 0, the root is the inductive base. Since from OF1 and OF3, its `rank` is always finite, the base case holds. What we would thus have to show is the inductive step: if all nodes with hop count i from the root eventually always have finite ranks, then a node with hop count $i+1$, $node_{i+1}$, eventually always has a finite rank too. Yet, let us first prove a weaker property, that $node_{i+1}$ ever sets its rank to a finite value:

Lemma 1. *If all nodes within i hops from the DODAG root eventually always have finite `ranks`, then a node within $i+1$ hops from the root, $node_{i+1}$, eventually has a finite `rank` (at least once but not necessarily always).*

Proof. $Node_{i+1}$ starts with its `rank` infinite (property OF4) and can change it only upon reselection (OF1). We thus have to show that a reselection of a finite rank eventually occurs.

We begin by observing that, having a hop count of $i+1$, $node_{i+1}$ must have at least one neighbor with a hop count of i. Let us denote one of such neighbors as $node_i$.

From property DIO2, $node_{i+1}$ always eventually receives a DIO message from $node_i$. From property DIO1, the `rank` fields of these messages are copied from $node_i$'s `rank` variable. Since eventually always this variable has a finite value and since, from property DIO3, any previous DIOs from $node_i$ with infinite `rank` fields eventually cease to be received by $node_{i+1}$, eventually always any DIO from $node_i$ received by $node_{i+1}$ has a finite value of the `rank` field and $node_{i+1}$ is guaranteed to always eventually receive such a message.

Consequently, from property RA2, an entry for $node_i$ is eventually always present in $node_{i+1}$'s `neighborset`. Likewise, from property RA3, the entry's `rank` field eventually gets a finite value for ever. At the moment when this happens, $node_{i+1}$'s `neighborset` changes. Therefore, from property RA1, a `rank` (and `prefpar`) reselection indeed eventually occurs at $node_{i+1}$. What thus remains to be shown is that the `rank` computed during this (or an earlier) reselection is finite.

To this end, let us observe that $node_{i+1}$ selects a finite rank `r` iff its `neighborset` contains an entry `n` such that `r` = $\mathcal{R}$(`n.rank`, `n.metrics`, `self.metrics`) is the potential rank offered by this entry, and together `n` and `r` satisfy all conditions (a)–(d) of property OF5. During the considered reselection, condition OF5(a) is satisfied by the entry for $node_i$. In contrast, the other conditions need not be satisfied by this and any other entries in $node_{i+1}$'s `neighborset`.

Let us start with conditions OF5(b) and OF5(c). We would be guaranteed they are satisfied for the entry corresponding to $node_i$ in $node_{i+1}$'s `neighborset` if $\mathcal{R}$, the function for computing potential ranks, fulfilled the following condition:

C1. *If `rank` is finite, then `rank`+`MinHopRankIncrease` $\leq$ $\mathcal{R}$(`rank`, $\bullet$, $\bullet$) $<$ infinity.*

To explain, condition C1 bounds the result of function $\mathcal{R}$ for a neighbor with a finite rank. The lower bound stems directly from RPL's minimal rank step, which aims to prevent routing loops [1]. The upper bound states in turn that a neighbor entry with a finite rank should not be excluded from the preferred parent selection process, irrespective of the routing metric values. There are indeed cases in which this requirement must be met; otherwise the nodes are unable to build a DODAG. In practice, however, one may want to slightly relax it by allowing for excluding extremely poor neighbors, such as ones with a high ETX or EPC approaching 100% in our examples. Nevertheless, in our proof, we assume the upper bound.

Let us return to the `rank` reselection at $node_{i+1}$. Given C1, conditions OF5(a)–OF5(c) are satisfied by the entry for $node_i$ in $node_{i+1}$'s `neighborset`. In contrast, condition OF5(d) may or may not be satisfied. In the first case, the entry and the potential rank it offers satisfy all conditions of property OF5, and hence $node_{i+1}$ adopts a finite value as its `rank`. In the second case, the fact that OF5(d) does not hold implies that *node*$_{i+1}$'s `minrank` is finite. However, from property OF2, this means that *node*$_{i+1}$'s `rank` must have been finite at least once. Consequently, in either case, *node*$_{i+1}$, at least once, must have its `rank` finite, which ends the proof of Lemma 1. □

We have proved that *node*$_{i+1}$'s `rank` will be finite *eventually* (e.g., once). To use induction for inferring that all nodes will eventually always have finite `rank`s, we would have to prove that *node*$_{i+1}$'s `rank` would *eventually always* be finite.

However, with the current approach, this need not be possible. In particular, the entry for *node*$_i$ in *node*$_{i+1}$'s `neighborset` may at some moment stop satisfying condition (d) of OF5 because of changes in routing metric values. Yet, induction on hop counts precludes assuming anything about *node*$_{i+1}$'s neighbors with hop counts larger than i. We thus have no means of showing that there will always be another entry in *node*$_{i+1}$'s `neighborset` satisfying all conditions of OF5 and thus eligible for *node*$_{i+1}$'s `prefpar`.

B. Problems with Changing Metric Values

For this reason, a different induction order needs to be considered. The main problem with the previous approach is that parent selection by the objective function depends on routing metric values, and thus need not reflect the nodes' hop counts from the root: if a neighbor with a larger hop count offers a node a better potential rank than a neighbor with a smaller hop count, then the node can select the former one as its preferred parent, like in the case of node H and its neighbors I and E in Fig. 1. Therefore, an intuitive induction order seems to be the one of the preferred parent relation.

Two problems arise, though. First, the nodes' links to preferred parents—in general—need not form a cycle-free graph, which is necessary to establish an order. Second, the nodes can change their preferred parents, whereas inductive reasoning requires a fixed order. Consequently, the preferred parent relation cannot be used as an order for induction.

To address these problems, we introduce a concept of *best ever parents* (*BEP*s). A node's BEP is the node's current or former preferred parent, whose selection gave the node the smallest rank ever (in a DODAG version) for the first time. Under our assumptions, this is the `prefpar` whose selection led to the last drop in the node's `minrank` (see Fig. 3).

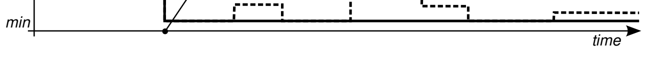

Fig. 3. An illustration of the BEP concept.

The links reflecting the BEP relation induce a subgraph of the connectivity graph: the DODAG root is always a member of the subgraph; a non-root node is a member iff it has a BEP or is another node's BEP. The BEP subgraph is thus actually a directed tree with a sink at the DODAG root, as formalized in Lemma 2, as such being suitable for induction, as explained next. Although the validity of the lemma is rather intuitive, the proof is technical, and hence can be found in Appendix A.

Lemma 2. *Nodes' links to BEPs eventually stop changing and form a directed tree with the sink at the DODAG root.*

Using the BEP relation, we will prove the property described by Lemma 3. In general, such a proof considers the system after the BEP tree has stabilized. The target property is first proved for the DODAG root, which constitutes the inductive base. Then, assuming that it holds for a node's BEP, the property is shown to also hold for the node itself.

Lemma 3. *Each member of the BEP tree eventually always has its* `rank` *finite.*

Proving Lemma 3 is relatively easy if we assume that conditions C2 and C3 hold.

C2. $\mathcal{R}$*(*`n.rank`*,* `n.metrics`*,* `self.metrics`*)* = `n.rank` + ρ*(*`n.metrics`*,* `self.metrics`*), where* ρ *depends solely on* `n.metrics` *and* `self.metrics`*.*

C3. *For each entry* `n` *in a node's* `neighborset`*,* ρ*(*`n.metrics`*,* `self.metrics`*) will eventually always be equal to its smallest value ever.*

Condition C2 is an interpretation of RPL's recommendation that the function for computing potential ranks, $\mathcal{R}$, should be additive. It aims at maximal simplicity, the advantage of which will become apparent shortly. At the same time, through function ρ, it leaves a lot of freedom in how routing metrics contribute to the computed rank. As such, it captures both OF0, for which ρ(•, •) = *MinHopRankIncrease*, and MRHOF with the two metrics utilized in this paper: for ETX, ρ(`n.metrics`, •) = ETX(`n.metrics`) × *MinHopRankIncrease*; for EPC, ρ(•, `self.metrics`) = EPC(`self.metrics`) × *MinHopRankIncrease*. In general, it captures all configurations in RPL's both aforementioned implementations: ContikiRPL and TinyRPL. Given this form of function $\mathcal{R}$ and the fact that routing metrics are modeled as "black boxes," in our subsequent reasoning, whenever we refer to changes in routing metric values, we in fact mean changes in the corresponding values of ρ(•, •).

Condition C3, in turn, states that however the routing metrics grow—or, recalling the previous explanation, however the corresponding values of ρ(•, •) grow—eventually they return to their minimal values and remain such forever. This ensures that a node's BEP will eventually always be eligible for the node's preferred parent, notably the potential rank it offers the node will be below the node's rank limit, as in OF5(d). In practice, however, this assumption is too strong to be satisfiable: instead of remaining minimal, routing metrics usually change dynamically. Therefore, for our work to have any practical implications, we must weaken condition C3.

Weakening C3 implies that we allow routing metrics to grow. Yet, if the allowed growth were unlimited, then a node would not be guaranteed to always have a neighbor satisfying OF5(d). The growth must thus be bounded, that is, for a neighbor `n`, we assume that ρ(`n.metrics`, `self.metrics`) is always within some limit, g, from its minimal value.

While not stated in RPL's specification, this requires `MaxRankIncrease` to depend on this metric growth limit, g. What may be even more unexpected, `MaxRankIncrease` should also depend on the network topology.

To explain, consider Fig. 4, where `MaxRankIncrease` $= L$. The N non-root nodes set their `rank`s when their $\rho(\bullet, \bullet)$ metric values are minimal. Now, suppose that the metrics grow by g for ever. If $N \times g > L$, the rightmost node will forever have no parent as its sole neighbor will never again satisfy OF5(d).

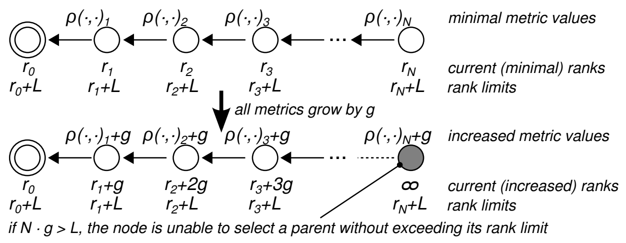

Fig. 4. A counter-example for local `MaxRankIncrease`.

`MaxRankIncrease` should thus indeed depend *globally* on the network topology; otherwise, a connected DODAG may never be formed or may suffer from disconnections. To emphasize this observation, let us formalize it as Proposition 1

PROPOSITION 1. *If the value of RPL's* `MaxRankIncrease` *does not depend on the number of nodes in the network, then RPL may not be able to form a connected DODAG.*

Such disconnections are precisely what can be observed in the middle plot of Fig. 2. In the experiment, all nodes started with their EPCs equal to 1 and these values grew by 1 in a bit less than an hour. This means that every hour a node's rank grew by the number of hops, i, from the node to the DODAG root. Since the default `MaxRankIncrease` was set to 7: after the first hour only the 35 nodes that were up to 7 hops away from the root were able to keep a preferred parent; after the second our, only the 9 nodes up to 3 hops away; and so on, so that after 8 hours even the root's direct neighbors were unable to select their preferred parent not to violate OF5(d). However, RPL's specification does not mention this global dependency of `MaxRankIncrease` on the possible metric value growth. Worse yet, it may suggest the opposite, stating that `MaxRankIncrease` is for *local* DODAG repairs. This is the reason for emphasizing our observations as Proposition 1.

C. Accounting for Imperfections of Objective Functions

Furthermore, our reasoning reveals how to weaken condition C3 while also taking this dependency into account. More specifically, C3 can be replaced with the weaker C3$'$.

C3$'$. *For each entry* `n` *in a node's* `neighborset`, $\rho($`n.metrics`, `self.metrics`$)$ *eventually always does not exceed its smallest ever value plus* `MaxRankIncrease` */ D, where D is the depth of the BEP tree.*

Under this weaker assumption, we can prove Lemma 3$'$, from which Lemma 3 follows. The D value in both C3$'$ and Lemma 3$'$ is the maximal path length in the BEP tree, which is bounded by the number of nodes. Put differently, we give an explicit dependency between `MaxRankIncrease` and both the tolerated metric growth and the network topology.

LEMMA 3$'$. *Each member of the BEP tree will eventually always have its* `rank` *finite, not exceeding its* `minrank` *by more than i* $\times$ `MaxRankIncrease` */ D, where i is the node's depth in the tree.*

Proof. The proof proceeds by induction on the final BEP tree. The base case for the root (with depth 0) follows directly from properties OF1 and OF3. To show the inductive step, in turn, let $node_i$ be a non-root node with depth i in the tree and let $node_{i-1}$ be $node_i$'s BEP in the tree, with a depth of $i-1$.

We start by showing that $node_i$'s `rank`—with $node_{i-1}$ as `prefpar`—eventually never exceeds $node_i$'s `minrank` by more than $i \times$ `MaxRankIncrease` / D, that is, that Inequality 1 eventually always holds

$$\begin{aligned}\mathcal{R}(n_{i,i-1}.\mathtt{rank},\, n_{i,i-1}.\mathtt{metrics},\, \mathtt{self}_i.\mathtt{metrics}) \\ \leq \mathtt{minrank}_i + i \times \mathtt{MaxRankIncrease} \,/\, D, \quad (1)\end{aligned}$$

where $n_{i,i-1}$ is the entry for $node_{i-1}$ in $node_i$'s `neighborset`, and the remaining symbols with subscript i correspond to $node_i$'s respective variables.

From condition C2, Inequality 1 can be transformed into

$$\begin{aligned}\Delta n_{i,i-1}.\mathtt{rank} + \Delta\rho(n_{i,i-1}.\mathtt{metrics},\, \mathtt{self}_i.\mathtt{metrics}) \\ \leq i \times \mathtt{MaxRankIncrease} \,/\, D, \quad (2)\end{aligned}$$

where $\Delta n_{i,i-1}$`.rank` denotes the difference between a given and minimal ever value of field `rank` of the entry for $node_{i-1}$ in $node_i$'s `neighborset`, and $\Delta\rho(n_{i,i-1}$`.metrics`, `self`$_i$`.metrics`$)$ has an analogous meaning for the $\rho(\bullet, \bullet)$ metric values.

From the inductive assumption, eventually $node_{i-1}$'s `rank` never exceeds its `minrank` by more than $(i-1) \times$ `MaxRankIncrease` / D. $\Delta n_{i,i-1}$`.rank` is thus also upper-bounded by this value. From C3$'$, in turn, $\Delta\rho(n_{i,i-1}$`.metrics`, `self`$_i$`.metrics`$)$ never exceeds `MaxRankIncrease` / D: Inequalities 2 and 1 eventually always hold. $Node_i$'s `rank` thus eventually never exceeds its `minrank` by more than $i \times$ `MaxRankIncrease` / D but only as long as $node_{i-1}$ is $node_i$'s `prefpar`.

However, $node_i$ may select another `prefpar`, which could potentially increase its `rank` above its `minrank` $+\ i \times$ `MaxRankIncrease` / D. To address this issue, we formalize the rules of `prefpar` selection, as in condition C4.

C4. *If a node selects a finite* `rank` *and non-*`null` `prefpar`*, then there does not exist an entry in the node's* `neighborset` *that satisfies all conditions (a)–(d) of OF5 and that offers the node a potential rank lower than the selected* `rank` *by more than* `ParentSwitchThr`.

Such a formulation of the selection rules is highly flexible in that it allows for the aforementioned hysteresis: a node may select as its `prefpar` a neighbor that offers a `rank` that is suboptimal up to a threshold, `ParentSwitchThr`.

In this way, the rules cover both existing objective functions: OF0 and MRHOF. For a moment, however, let us assume that `ParentSwitchThr` equals zero, that is, a node always selects the neighbor that offers it the lowest `rank`.

In this case, $node_i$ never selects as its `prefpar` a neighbor offering it a greater `rank` than $node_{i-1}$ does. Consequently, since $node_i$'s `rank` with $node_{i-1}$ as the `prefpar` is eventually always at most $node_i$'s `minrank` $+\ i\ \times$ `MaxRankIncrease` / D, so is $node_i$'s `rank` with any other selected parent. This ends the proof of Lemma $3'$ for the case where `ParentSwitchThr` equals zero.

In contrast, if we do allow hysteresis, that is, we allow `ParentSwitchThr` to be greater than zero, then we also need to generalize condition C$3'$, replacing it with C$3''$.

C$3''$. *For each entry* `n` *in a node's* `neighborset`, ρ*(*`n.metrics`*,* `self.metrics`*) eventually always does not exceed its smallest ever value plus max(0,* `MaxRankIncrease` */ D −* `ParentSwitchThr`*).*

Compared to condition C$3'$, under C$3''$ the tolerated limit on the growth of $\rho(\bullet, \bullet)$ metric values is lower, which accounts for suboptimal parent choice under C4. More specifically, because of hysteresis, $node_i$ can select and keep forever a `prefpar` that offers it a `rank` worse than the one offered by $node_{i-1}$ by at most `ParentSwitchThr`. Consequently, a limit lower by `ParentSwitchThr` than `MaxRankIncrease` / D accommodates such suboptimal selections. In effect, despite hysteresis, $node_i$'s `rank` is eventually always at most $node_i$'s `minrank` $+\ i\ \times$ `MaxRankIncrease` / D, which ends the proof of Lemma $3'$. □

From Lemma $3'$, we know that the *BEP tree members* eventually always have finite ranks. Hypothesis 1 is in turn that *all nodes* eventually always have finite ranks. This discrepancy is addressed by Lemma 4, with a simple proof in Appendix B.

LEMMA 4. *Every node is eventually always a member of the BEP tree.*

We are now ready to formulate this paper's main theorem.

THEOREM 1. *Hypothesis 1 is true.*

Proof. Follows directly from Lemmas 2, $3'$, and 4. □

VI. CONCLUDING DISCUSSION

By proving Theorem 1, we formally ascertained the correctness of RPL's DODAG construction and maintenance in abstraction from particular objective functions and routing metrics. To this end, however, in addition to the properties describing RPL's operation based directly on the specification, we had to introduce several additional conditions, notably C1, C2, C$3''$, and C4. These formal conditions provide important insights into the requirements on objective functions and routing metrics, which are not mentioned in RPL's specification but fulfilling which is crucial for the proper operation of the protocol in the presence of dynamic changes to routing metric values. To conclude, we compile these theoretical insights into practical guidelines that can be utilized by RPL's adopters:

Use a simple function, $\mathcal{R}$, for potential rank computation.
To compute a potential rank a neighbor, `n`, offers a node, one may employ an arbitrary function $\mathcal{R}$. Yet, using $\mathcal{R}$ of the form $\mathcal{R}$(`n.rank`, `n.metrics`, `self.metrics`) $=$ `n.rank` $+\ \rho$(`n.metrics`, `self.metrics`), where ρ depends solely on `n.metrics` and `self.metrics` (condition C2), greatly simplifies tracking the contribution of a single ancestor node/link in a DODAG to the rank of another node: each ancestor node has a chance to contribute equally to the node's rank. In contrast, if one deviates from the scheme even slightly, the contributions may be biased and tracking them need not be trivial. For example, for $\mathcal{R}$(`n.rank`, $\bullet, \bullet) = 2\ \times$ `n.rank` $+\ \rho(\bullet, \bullet)$, the contribution of an ancestor node i hops away from the node that computes $\mathcal{R}$ is proportional to 2^i.

Try to keep routing metric values bounded.
Under the previous assumption on $\mathcal{R}$, the values of ρ(`n.metrics`, `self.metrics`) must be at least `MinHopRankIncrease` to fulfill RPL's requirement of a minimal rank difference between two nodes on a routing path. They should also be finite (condition C1). In contrast, infinity should be used only to exclude a given neighbor from the parent selection process, for instance, when the ETX of the link with the neighbor is so high that sending a packet to the neighbor would likely fail.

Explicitly configure the tolerated rank growth.
RPL tolerates only a limited growth of a node's rank, configurable as `MaxRankIncrease`, which is crucial for failure handling [14], [15]. This limit turns out to depend not only on the maximal allowable growth of the $\rho(\bullet, \bullet)$ routing metric values but also on the network diameter (condition C$3''$). Therefore, when deploying RPL, one should estimate both these quantities and configure `MaxRankIncrease` based on condition C$3''$; otherwise, the resulting DODAG may get disconnected.

Watch out for hysteresis in objective functions.
Although hysteresis in an objective function makes a DODAG more stable, it reduces the tolerated rank growth by the parent switch threshold, `ParentSwitchThr`, for every hop (condition C$3''$ vs. C$3'$). One should thus revisit RPL's configuration whenever hysteresis is employed. What is more, implementations of objective functions with hysteresis must pay special attention to enforce the threshold (condition C4); otherwise, again, the DODAG may get disconnected.

Stay in control of the growth of routing metric values.
Certain routing metrics, such as EPC, may grow continuously, exceeding the tolerated rank growth limit and, thereby, leading to DODAG disconnections (as in Fig. 4). To ensure that a DODAG remains connected, such a growth must be controlled. This can be done by first scaling the metrics' granularity and adjusting their initial values, and then having the nodes continuously monitor to what extent their present ranks deviate from the ones they had for their BEPs. Whenever the deviation approaches

`MaxRankIncrease`, a new DODAG version can be ordered, so that the ranks are assigned afresh. Such an approach employs DODAG versioning, which today is virtually unused in RPL's implementations, and, importantly, does not impair handling various types of failures.

In particular, by following these guidelines, we have solved the problems with EPC illustrated in the case study. First, by scaling EPC by 0.1, treating such scaled values as fixed-point numbers, and rounding the initial value to 1 to ensure a minimal delta of `MinHopRankIncrease`, we reduced the pace of rank growth. Second, by configuring `ParentSwitchThr` and `MaxRankIncrease` based on condition C3$''$, monitoring the rank growth through BEPs, and ordering DODAG version changes whenever the growth approaches the tolerated limit, we eliminated disconnections. In the case of our simulations, employing these solutions makes the plot for EPC resemble the one for ETX in Fig. 2 (the figure is thus omitted for brevity).

In general, by applying the presented research, one can truly improve the reliability of RPL's deployments, obtaining provably correct solutions not only for custom but also the common routing metrics and objective functions. Such a degree of reliability is important in many industrial applications.

What is more, the presented reasoning gives additional insights into RPL's operation, notably the issues the protocol leaves open. For example, the fact that we are able to prove correct DODAG formation and maintenance under such a weak requirement on DIO message delivery as formalized in property DIO2 implies that medium access control protocols suitable for RPL need not strive at highly reliable broadcasting and can even resort to simple, albeit energy-conserving probabilistic solutions, like narrowcasting [18].

Finally, our results and techniques stretch beyond RPL. Many of them can be applied to other multi-objective routing protocols utilizing multiple routing metrics and multi-constrained optimization functions.

At the same time, we are aware that the theoretical conditions and practical guidelines we derived for the considered problem of provably correct DODAG formation and maintenance are by no means the ultimate ones. Some of them can likely be relaxed or generalized so as to cover even a broader range of objective functions and routing metrics, even if such constructs would be purely theoretical. Furthermore, there are also other facets of RPL's behavior for which deriving provable correctness guarantees would be beneficial for industry [16]. RPL's reliability thus indeed involves exciting research problems with a considerable practical impact.

Acknowledgments

This work was supported by the National Center for Research and Development (NCBR) in Poland under grant no. LIDER/434/L-6/14/NCBR/2015. K. Iwanicki was additionally supported by the Polish Ministry of Science and Higher Education with a scholarship for outstanding young scientists.

References

[1] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, and R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks," RFC 6550, 2012.

[2] H.-S. Kim, J. Ko, D. Culler, and J. Paek, "Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 4, pp. 2502–2525, 2017.

[3] P. Thubert, "Objective function zero for the routing protocol for low-power and lossy networks (RPL)," RFC 6552, 2012.

[4] O. Gnawali and P. Levis, "The minimum rank with hysteresis objective function," RFC 6719, 2012.

[5] J.-P. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing metrics used for path calculation in low-power and lossy networks," RFC 6551, 2012.

[6] A. Brachman, "RPL objective function impact on LLNs topology and performance," in *NEW2AN'13*, 2013.

[7] O. Iova, F. Theoleyre, and T. Noel, "Stability and efficiency of RPL under realistic conditions in wireless sensor networks," in *PIMRC'13*, 2013.

[8] O. Gaddour, A. Koubâa, N. Baccour, and M. Abid, "OF-FL: QoS-aware fuzzy logic objective function for the RPL routing protocol," in *Proc. WiOpt '17)*, 2014.

[9] P. Kulkarni, S. Gormus, and Z. Fan, "Tree balancing in smart grid advanced metering infrastructure mesh networks," in *Proc. GreenCom '12*, 2012.

[10] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proc. IPSN '12*, 2012.

[11] P. Thulasiraman, "RPL routing for multigateway AMI networks under interference constraints," in *Proc. ICC '13)*, 2013.

[12] H. S. Kim, J. Paek, and S. Bahk, "QU-RPL: Queue utilization based RPL for load balancing in large scale industrial applications," in *Proc. SECON '15*, 2015.

[13] O. Iova, F. Theoleyre, and T. Noel, "Using multiparent routing in RPL to increase the stability and the lifetime of the network," *Ad Hoc Networks*, vol. 29, pp. 45 – 62, 2015.

[14] K. Iwanicki, "RNFD: Routing-layer detection of DODAG (root) node failures in low-power wireless networks," in *Proc. IPSN'16*, 2016.

[15] A. Paszkowska and K. Iwanicki, "The IPv6 routing protocol for low-power and lossy networks (RPL) under network partitions," in *Proc. EWSN '18*, 2018.

[16] A. Paszkowska and K. Iwanicki, "Failure handling in RPL implementations: An experimental qualitative study," Book chapter accepted for publication in H. M. Ammari (Ed.) "The Philosophy of Mission-Oriented Wireless Sensor Networks," available upon request, 2018.

[17] M. Ben-Ari, *Mathematical Logic for Computer Science*, 3rd ed. Springer-Verlag London, 2012.

[18] T. Pazurkiewicz, M. Gregorczyk, and K. Iwanicki, "NarrowCast: A new link-layer primitive for gossip-based sensornet protocols," in *Proc. EWSN '14*, 2014.

Appendix A
Proof of Lemma 2

Proof. We start by showing that the nodes' `minrank`s eventually stop changing, even in the presence of continuous changes to routing metric values. To this end, let us observe that the only values a node's `minrank` can attain are the values of the node's `rank` (property OF2). In effect, initially, the node's `minrank` is infinite or equal to `MinHopRankIncrease` (properties OF4 and OF3) and can change only to the values the node selects for its `rank` (property OF1). What is more, any such change can only be a decrease (property OF2).

Since the two variables are integer, any such decrease is at least by one. From Lemma 5 (see Appendix C), however, the node's `rank`, and thus `minrank`, is lower-bounded by `MinHopRankIncrease`. Therefore, at some moment, the node's `minrank` must stop decreasing forever (if it has been decreasing at all). If the node has a preferred parent at that time, the parent remains the node's BEP forever (cf. Fig. 3). The nodes' links to BEPs thus indeed are eventually fixed forever, which proves the first part of the lemma. To prove the second part, we will show that they form a directed tree with a sink at the DODAG root.

To this end, let $node_k$ be a node with a non-`null` BEP, $node_j$: when $node_k$'s `minrank` changed for the last time, $node_j$ was selected as $node_k$'s `prefpar` from $node_k$'s `neighborset`. From OF5(c), at that moment, $node_k$'s `rank` exceeded the `neighborset`'s entry's `rank` field for $node_j$. It was thus also greater than $node_j$'s `minrank`. This was because, first, whenever a node broadcasts a DIO, the node's `minrank` is at most the node's `rank` (property OF2), second, the `rank` is copied to the DIO's `rank` field (property DIO1) and then copied to a corresponding neighbor entry's `rank` field (property RA3), and third, as we have just shown, the node's `minrank` can only decrease, in particular, when DIOs containing its value are in transit. All in all, $node_j$'s `minrank` is smaller than $node_k$'s `minrank` when $node_j$ is $node_k$'s BEP.

From this, we can infer that the BEP subgraph has no cycles. By contradiction, if there existed nodes $n_1, n_2, n_3, \ldots, n_k$ whose links to BEPs formed a cycle, then we would have n_1's `minrank` $<$ n_2's `minrank`, and n_2's `minrank` $<$ n_3's `minrank`, and $\ldots$, and n_k's `minrank` $<$ n_1's `minrank`. This, however, is impossible: contradiction! In other words, the BEP subgraph is indeed cycle-free.

We will now show that all paths in the subgraph end at the DODAG root. First, let us observe that since the subgraph contains a finite number of nodes and no cycles, each path is finite as well: it must end at some node. Suppose, by contradiction, that some path ends at a non-root node. As such, the node must have had its `rank` finite at some moment; otherwise, the entry for the node in the `neighborset` of the node's predecessor on the path would never satisfy OF5(a), and the node would have never become the predecessor's `prefpar`, and hence BEP. Since, not being the root, the node started with its `rank` infinite (property OF4), it must have changed the `rank` to a finite value during `prefpar` and `rank` selection (property OF1). From OF5, the selection yields a finite rank only with a non-`null` `prefpar`. However, if the node has ever had a non-`null` `prefpar`, then it has a BEP. As such, it cannot end the considered path in the BEP subgraph: a contradiction!

To sum up, the BEP subgraph is cycle-free, all its paths end at the DODAG root, and, by definition, it contains no isolated nodes (i.e., ones that are not on some path), apart from the case when it consists only of the DODAG root. The subgraph is thus a tree with the sink at the DODAG root, which proves the second part of the theorem. Note, however, that the theorem does not state whether *all* nodes belong to the BEP tree. This issue is addressed in Lemma 4, which is proved next. □

Appendix B
Proof of Lemma 4

Proof. The proof proceeds by induction on a node's hop count from the DODAG root in the connectivity graph, which is connected and covers all nodes. As the base, we take the DODAG root itself, which by definition belongs to the BEP tree. As the inductive step, in turn, we need to show that if all nodes with hop count i from the root are eventually always members of the tree, then a node with hop count $i+1$, $node_{i+1}$, is eventually always a member as well.

Since all nodes within i hops from the DODAG root are (eventually always) members of the tree, then from Lemma 3′ their `rank`s are eventually always finite. Consequently, from Lemma 1, $node_{i+1}$ eventually has its `rank` finite at least once. Since $node_{i+1}$ is a non-root node, from properties OF1 and OF4, its `rank` can be set to a finite value only when a non-null `prefpar` is selected. Consequently, since $node_{i+1}$ has a preferred parent at least once, it eventually always has a BEP and is thus eventually always a member of the BEP tree. □

Appendix C
Supplementary Lemmas and Their Proofs

Lemma 5. *The nodes' ranks in the system, that is, all nodes'* `rank` *variables and fields* `rank` *of DIOs in transit and of the entries in the nodes'* `neighborset`*s, are always greater than or equal to* `MinHopRankIncrease`*.*

Proof. The proof follows a standard method for showing safety properties of concurrent programs [17]: induction on the sequence of states representing a computation. To this end, consider an arbitrary computation.

The inductive base is the first state, in which: `rank` is `MinHopRankIncrease` for the root (property OF3) and infinity for all other nodes (property OF4), each node either has no entry in its `neighborset` or the field `rank` of any such entries is infinite (property RA3), and there are no DIOs in transit (property DIO1). All ranks in the system are thus indeed at least `MinHopRankIncrease`.

For the inductive step, we assume that all `rank` variables and fields `rank` of DIOs in transit and of entries in the nodes' `neighborset`s are at least `MinHopRankIncrease`. We will show that no operation at any node will produce a rank value smaller than `MinHopRankIncrease`.

Sending a DIO does not produce a lower rank because the DIO's field `rank` is copied from the sender's `rank` variable (property DIO1). Similarly, receiving a DIO boils down to copying its `rank` field to the `rank` field of the entry in the receiver's `neighborset` that corresponds to the DIO's sender (property RA3), whereas if the DIO is lost, no new rank value appears in the system. Rank reselection is in turn the only operation that affects a node's `rank` variable (property OF1). At the root node, it returns `MinHopRankIncrease` (property OF3); at a non-root node—either infinity (property OF4) or a value higher than the value of field `rank` of some entry in the node's `neighborset` (property OF5(c)). Therefore, in all cases, no value below `MinHopRankIncrease` is assigned to the node's `rank` variable. Finally, new entries inserted into a node's `neighborset` have their `rank` fields infinite (property RA3), while removing an entry does not produce a new rank value. All in all, indeed no operation generates in the system a rank value lower than `MinHopRankIncrease`, which ends the proof. □