Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network

Oana Iova, Fabrice Theoleyre, Thomas Noel

Using Multiparent Routing in RPL to Increase the Stability and the Lifetime of the Network ☆,☆☆

Oana Iova[a,*], Fabrice Theoleyre[a], Thomas Noel[a]

[a] *University of Strasbourg, ICube Laboratory, 300 Bv. Sébastien Brant, 67400 Illkirch France*

Abstract

Energy is a very scarce resource in Wireless Sensor Networks. While most of the current proposals focus on minimizing the global energy consumption, we aim here at designing an energy-balancing routing protocol that maximizes the lifetime of the most constraint nodes. To improve the network lifetime, each node should consume the same (minimal) quantity of energy. We propose the *Expected Lifetime* metric, denoting the *residual time* of a node (time until the node will run out of energy). We design mechanisms to detect energy-bottleneck nodes and to spread the traffic load uniformly among them. Moreover, we apply this metric to RPL, the *de facto* routing standard in low-power and lossy networks. In order to avoid instabilities in the network and problems of convergence, we propose here a multipath approach. We exploit the Directed Acyclic Graph (DAG) structure of the routing topology to probabilistically forward the traffic to several parents. Simulations highlight that we improve both the routing reliability and the network lifetime, while reducing the number of DAG reconfigurations.

Keywords: RPL; network lifetime; energy-balancing

1. Introduction

Routing in Wireless Sensor Networks (WSN) has been extensively studied in the last decade. These networks are highly unreliable, prone to multihop interference, and to a time varying link quality. Moreover, the devices composing them are very limited in terms of memory, processing power and battery [1]. In this type of environment, a *good* routing protocol should:

a) save energy, since most of the nodes are battery powered [2];

☆A short conference version of this paper has been submitted `https://clarinet.u-strasbg.fr/~theoleyre/tmp/rpl_multipath_conf_submitted_version.pdf`

☆☆This work was partially supported by the French National Research Agency (ANR) project IRIS under contract ANR-11-INFR-016.

*Corresponding author, tel. +33 3 68 85 44 01

Email addresses: `otiova@unistra.fr` (Oana Iova), `theoleyre@unistra.fr` (Fabrice Theoleyre), `noel@unistra.fr` (Thomas Noel)

b) deal with lossy links, for both control and data packets [2];

c) avoid routing loops, and enable fast convergence [3].

RPL is considered as a *de facto* routing standard for the Internet of Things [4]. It aims at optimizing the routing scheme for the convergecast traffic pattern (i.e., all the packets are sent to a collection of *border routers*, connected to the Internet). RPL is based on a Destination-Oriented Directed Acyclic Graph (DODAG) rooted at the border routers. The DODAG is constructed based on a *rank*, denoting the *virtual distance* of each node to the root. RPL reflects the current evolution of this research area: it introduces redundancy in the routing structure, to be fault-tolerant.

However, in our opinion, the current version of RPL presents two ways of improvement. First, the Roll working group has focused on efficiently constructing a routing structure. We have now to provide metrics and mechanisms to make RPL energy-efficient: the topology (i.e., the DODAG) should be constructed based on energy criteria. Second, a node selects one preferred parent to construct the DODAG without loops, and to compute its own rank. However, only this preferred parent is used for routing: the other ones have just a *backup* purpose. We are convinced we should rather exploit this diversity to distribute the traffic load in the network and to create energy-balanced paths.

While adding more sinks could help better distributing the traffic load [5], this also increases the deployments cost. We consider here only one sink per network. In this case, two main approaches to save energy exist in the literature. The first one, minimizes the global energy consumption: the ETX metric for instance, aims at selecting energy-efficient links [6]. However, the nodes with the best links will be chosen uppermost to route packets: they will deplete their energy faster. The second one, uses in priority the nodes with a large residual energy to forward most of the traffic [7]. Still, these nodes, with possibly bad links, will receive most of the traffic and will consequently run out of energy faster. Clearly, we should not have a small collection of nodes that forwards most of the traffic.

We take here an alternative approach that balances efficiently the energy among all the nodes, based on a multipath solution. Indeed, multipath routing has been widely used in the literature to improve the reliability [8], to be fault-tolerant [9], to balance the load for congestion avoidance [10], or for QoS improvement [11]. While QoS is an important issue [12, 13, 14], we aim here at rather splitting the load to balance the energy consumption, and improve this way the network lifetime. We think that each path should consume the same quantity of energy. More precisely, all the nodes with the lowest residual energy (denoted *bottlenecks* in this paper) should be equally energy-balanced.

We have proposed in [15] a novel routing metric that allows a node to estimate how much time it has to live before running out of energy. We define here this metric for multipath routing protocols, to create an energy balanced topology. The improvements brought by this paper are:

1. we extend the *Expected Lifetime* metric for the multipath scenario and we detail how to estimate it for each node;

2. we identify the bottleneck nodes in this new context, and we construct accordingly a Directed Acyclic Graph (DAG) that equalizes the *Expected Lifetime* among the weakest nodes;

3. we address the stability problem: the routing DAG should limit the number of reconfigurations, in order to avoid oscillations and to minimize the energy consumption. We highlight that multipath helps surpassing these problems. A node changes its preferred parent only when it becomes useless (i.e., it does not forward traffic anymore), avoiding this way the sudden routing reconfigurations. Less reconfigurations also mean less control packets, which minimizes the energy consumption;

4. we propose an algorithm to split the traffic among several paths, while balancing the energy equally among them.

2. Related Work

2.1. RPL: Routing Protocol for Low-power and Lossy Networks

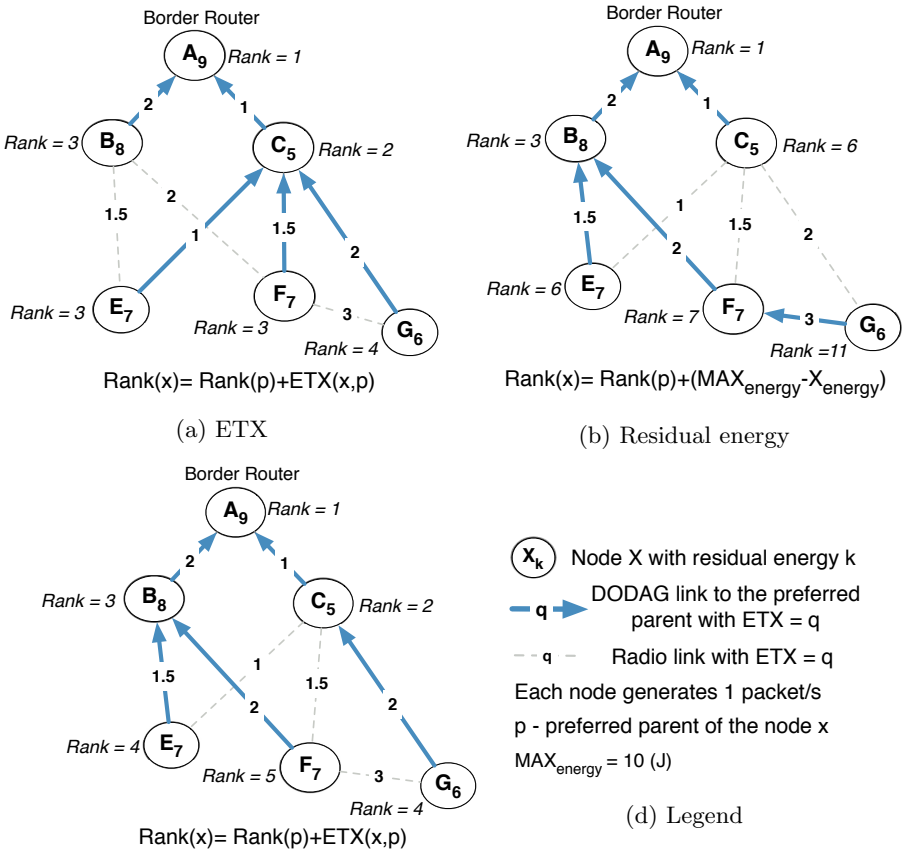
RPL is a distance vector protocol for low-power and lossy networks [4]. Starting from a border router, RPL constructs a Destination-Oriented Acyclic Graph (DODAG) using one or several routing metrics.

The DODAG construction is based on the *rank* of a node, which depicts its relative distance to the DODAG root. An *Objective Function* defines how a collection of routing metrics have to be combined to compute the rank. In order to have a loop-free topology, the rank must strictly monotonically increase from the root towards the leaves of the DODAG.

The construction and the maintenance of the DODAG are ensured by DODAG Information Object (`DIO`) messages periodically broadcasted by all the nodes. These packets contain information such as the *DODAG identifier*, the Objective Function, the rank of the node, the metrics used for path calculation, etc.

When a node receives a `DIO`, it inserts the emitter in the list of the possible successors, i.e., next hops to the border router. From all the successors in this list, the node will choose its preferred parent and send it all its traffic. It then computes its own rank with the Objective Function and starts broadcasting itself `DIO` messages. The simplest Objective Function – OF0 [16] – consists in choosing the preferred parent as the one advertising the lowest rank. Then, the node adds a small increment to the rank of its preferred parent to compute is own rank while maximizing the number of siblings.

The DODAG structure creates and maintains multiple routes towards the border router. However, RPL only uses a single path to route the packets (through the preferred parent). Pavkovic *et al.* [17] extended RPL to be used opportunistically with IEEE 802.15.4-2006: a node sends a data packet to the first available parent, instead of waiting for the preferred parent to be available.



(a) ETX

(b) Residual energy

(c) ELT

(d) Legend

Figure 1: DODAG construction using different routing metrics

However, their focus is on offering QoS for delay-sensitive packets and not on improving the network lifetime.

Hong *et al.* [18] proposed to choose the preferred parent using the hop count and then to select as the forwarding node the parent offering the best link quality. However, this can be equivalent to choosing the best parent among the worst available ones. Besides, a collection of nodes may still forward most of the traffic, and will run faster out of energy.

2.2. Energy Aware Routing

If we aim at minimizing the average energy consumption, ETX may take into account the link reliability to construct only energy-efficient routes [6]. However, we would not optimize the network lifetime, since a small number of nodes with a high ETX and close to the border routers may have to forward most of the traffic.

Let us consider the topology in Fig. 1a where a routing DAG is constructed based on the ETX metric [19]. E may choose either B or C as next hop. C is the most accurate choice, since it presents the lowest cumulative ETX towards the border router. However, if all the nodes generate the same amount of traffic, B should be preferred to balance the energy consumption.

Chipara *et al.* proposed to dynamically adapt the transmission power of the sensors in order to reduce real-time communication delay [20]. Packets that do not have an urgent deadline are transmitted at a lower power, to reduce the energy consumption. However, even if this solution is energy efficient, it does not maximize the network lifetime.

If we focus on the network lifetime optimization, we should use nodes with a large residual energy. For instance, Yoo *et al.* proposed to use the residual energy depletion rate (REDR) [21] to avoid overloaded nodes. REDR is estimated through an exponential moving average of the depletion ratio per second, of the residual energy, during an interval of time. The next hop is chosen as the node advertising the smallest weighted sum between the maximum value of the REDR and the sum of the REDR on the path. In our opinion, to optimize rather directly the lifetime for each node is a better approach.

Chang *et al.* formulated the routing problem as a linear programming problem where the objective is to maximize the network lifetime [22]. However, they only presented a centralized algorithm to route the packets.

PWave on the other hand, adopted an analogy with potential fields (similar to a virtual distance) [23]. The protocol constructs multiple routes and balances the load proportionally to the inverse of the cumulative path cost. The authors consider they optimize the energy consumption if the protocol uses the residual energy as the routing metric. However, they only minimize the sum: we should rather take care of the most loaded node. Besides, this potential field concept is very similar to the rank of RPL.

Chang *et al.* combined linearly the residual energy and the ETX [24]. However, this weight is not directly related to the real lifetime of a node. Kamgueu *et al.* proposed to use the residual energy to construct the RPL DAG [7]. However, they do not consider the radio link quality and thus, the energy budget for a correct reception. Consequently, bad radio links may be used, which results in inefficient routes.

Let us take a look at the topology depicted in Fig. 1b where a routing DAG is constructed based on the residual energy. We can observe that G can choose either F or C as a next hop. Because the residual energy of F is greater, it will choose it as a parent, even though the corresponding link quality is very low (ETX=3). This will lead to the quick battery depletion of node G. C would be a more appropriate choice. Moreover, this will result in a more energy balanced topology, since B will not have to relay the whole traffic of the network.

As we can see, a lot of routing metrics that take into account the energy constraints of a WSN have been proposed in the literature. However, to the best of our knowledge, none of them manages to create a routing topology that maximizes the lifetime of the network in a distributed manner.

2.3. Multipath Routing

Multipath routing has been widely used in the literature to improve the fault-tolerance (reliability), to balance the load (congestion avoidance), or for QoS improvement [11]. Routes may be either node or link-disjoint. Since we consider here only energy savings, we focus on the former case. Besides, *braided* paths (i.e., partially overlapping) have been proved to reduce the maintenance cost while still balancing the load efficiently [25].

Chen *et al.* were the first to apply energy-balancing routing in WSN [26]. However, the authors proposed to compute the optimal paths in a centralized way, by collecting the whole radio topology at the base station. Since the radio topology may change dynamically, and the overhead is important, this approach may perform poorly in practice. Ming-hao *et al.* adopted the same objective, but they constructed reactively two paths per source [27]. After the first path has been discovered, the source tries to find another path using nodes non interfering with the previous path. However, this scheme does not exploit the convergecast traffic pattern. We have rather considered the proactive construction of an energy-balancing routing structure for convergecast.

Kacimi *et al.* proposed a load balancing solution where they formalized the lifetime optimization as a nonlinear problem with linear constraints [28]. Even though their method is defined for the convergecast traffic, they assume constant and uniform link quality, which is not true in real life deployments.

Yahya *et al.* proposed to take into account also the radio link quality by using the REER metric [29]. REER combines the residual energy, the buffer size and the Signal-to-Noise Ratio (SNR) into a weighted function to obtain an energy efficient metric. However, this combination assumes implicitly that the energy consumption depends linearly on these factors: this is not the case as highlighted in the Equation 5 further.

An opportunistic routing specific for WSNs has been presented by Ghadimi *et al.* [30]. Based on the number of duty-cycled wakeups until a packet has reached its destination, each node selects a number of potential forwarders. Finally, a single node is selected as a unique forwarder, using a coordination algorithm. However, such algorithms usually need a modification of the MAC layer, or at least, a cross-layer approach.

We can see that the multipath routing has been used previously for load balancing. We propose here to combine it with an energy-aware path selection algorithm, in order to improve the network lifetime, without using cross-layer techniques.

3. The Expected Lifetime

3.1. Problem Statement

To the best of our knowledge, no metric manages to create a routing topology that maximizes the lifetime of a WSN in a distributed manner. We need a routing metric that takes into account the quality of the links, while balancing

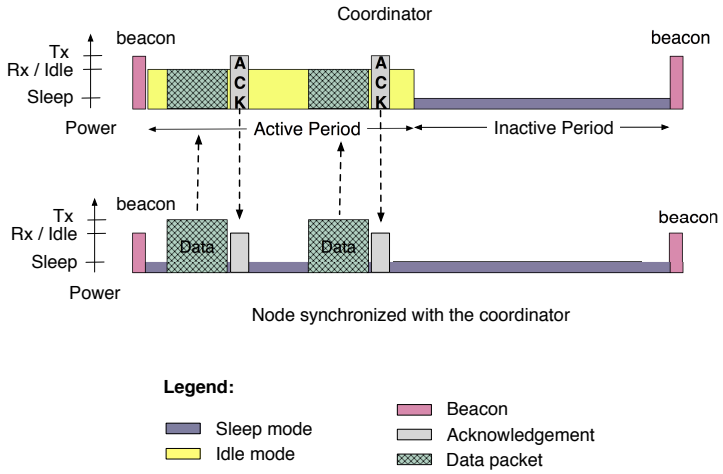


Figure 2: Communication between a node and its parent (i.e., a coordinator) in IEEE 802.15.4-2006

the energy consumption for all the flows. We aim here at constructing paths so that all the nodes share *fairly* the traffic load / energy.

In short, the routing metric should satisfy the following properties:

1. it must capture the variations of the link quality;
2. it has to maximize the end-to-end reliability by using energy efficient routes;
3. it must minimize the energy consumption of the nodes that consume the most energy.

Let us consider the scenarios described in Fig. 1, where the DODAG is constructed using RPL with different routing metrics. We can see in Fig. 1c how an energy-balanced routing topology should look like. A node selects its preferred parent so that it maximizes the lifetime of the most constraint nodes, without becoming itself the new constraint node.

3.2. Assumptions

In this paper, we consider the network lifetime as the time before the first node runs out of energy, since it is the most frequent definition [7, 21, 28, 31]. Moreover, a node should choose to send its traffic on the least energy-constraint path. Hence, if a node runs out of energy it will most surely disconnect the network, otherwise, its neighbors would not have chosen it as parent.

We consider a WSN with a periodic traffic pattern: all the sensors report periodically their measures to a border router [32, 33, 34].

We also consider that the energy consumed to receive a packet can be neglected. Indeed, in IEEE 802.15.4-2006 for example, a node transmits data packets to its parent with a convergecast traffic pattern (Fig. 2). The transmitter is able to turn off its radio during the backoff and the idle time. Consequently, the quantity of energy consumed by a transmitter is roughly proportional to the number of transmissions. On the contrary, the coordinator (i.e., the receiver) has to stay awake during the whole active period. Besides, the power of the radio chipset is often the same, no matter if a packet is received or not (e.g., Atmel AT86RF231). Consequently, a coordinator does not spend more energy to receive a packet: it has anyway to stay awake (idle listening) during the whole active period.

3.3. Contribution

In this paper, we present a routing metric that aims at maximizing the network lifetime for a multipath routing scheme. Moreover, through local decisions, but depending from each other, this metric enables the creation of an energy balanced topology.

We propose to take advantage of the DODAG structure created by RPL and to balance the traffic to all the parents, not just to the preferred one. We therefore manage to address the following problems:

1. *Even though RPL creates a DODAG, for routing it uses a tree topology, which does not allow load balancing.* Indeed, a node has to take a binary decision: each of its parents receives either all its traffic or none. Forwarding the traffic to several parents results in a more reliable and energy efficient routing protocol.

2. *Estimating the quality of a radio link consumes energy.* In order to save energy, a passive measurement technique is preferable. However, only the link quality to the neighbors with which a node exchanges data packets can be estimated passively. Since in RPL all the traffic is sent to the preferred parent, a node can only estimate the link quality to this node. However, by balancing the traffic to all the parents, a node can continuously estimate the quality of all these links.

3. *Frequent preferred parent changes can induce instability in the network and are energy consuming.* Indeed, the change of the preferred parent triggers the reset of the trickle timer, which causes a more frequent `DIO` transmission. The more control packets are sent, the more energy is consumed by the nodes. Moreover, frequent changes can induce instability in the network, which are exacerbated with larger topologies [35]. By using all the parents to route the traffic, the preferred parent can be changed only when it is not useful anymore, reducing the trickle timer resets, and the probability of creating instabilities.

Table 1: Notation used in the article

Notation	Meaning
ELT(X)	Expected lifetime of X (in seconds)
$\boldsymbol{E_{res}(X)}$	Residual energy of X (in Joules)
$\boldsymbol{P_{TX}(X)}$	Radio power in transmission mode (in Watts or Joules/s)
ETX(A,B)	ETX of the link $A \rightarrow B$
$\boldsymbol{T_X}$	Throughput of X (in bits/s)
$\boldsymbol{\alpha_P}$	Ratio of traffic sent to parent P
$\boldsymbol{\gamma}$	used for the computation of α_P
$\boldsymbol{r_{X,B}}$	Ratio of traffic forwarded by X to bottleneck B
$\boldsymbol{T_{gen}(X)}$	Traffic generated by X (in bits/s)
Children(X)	Children set of node X
Parents(X)	Parents set of node X
Bottlenecks(X)	Bottlenecks set of node X
DATA_RATE	The rate at which the data is sent (bits/s); All nodes transmit at the same rate

3.4. The Expected Liftetime (ELT) of a Node

We propose here the *Expected Lifetime* (ELT) metric. Instead of minimizing the sum of energy, or considering only the residual energy, ELT aims at maximizing directly the lifetime of the most constraint nodes, denoted *bottlenecks*.

ELT estimates the expected lifetime, i.e., the time before a node dies if it keeps on forwarding the same quantity of traffic. ELT helps quantifying the impact of a routing decision on the bottlenecks.

To compute its ELT, a node N (cf. notation in Table 1):

1. estimates the total traffic that it has to transmit by taking into account both the traffic that it generates and the incoming traffic from its children (i.e., the layer-3 load). By computing the throughput of a node in bits/s, we implicitly account for packets of different sizes:

$$T_N = T_{gen}(N) + \sum_{i \in Children(N)} T_i \tag{1}$$

2. multiplies the traffic to transmit (T_N) by the average number of retransmissions, given by:

   - the link reliability to each of its parents i.e., $\mathrm{ETX}(N, P)$, where $P \in Parents(N)$;
   - the ratio of traffic sent to each of its parents (α_P).

   This represents the average number of MAC transmissions:

$$T_N \times \sum_{P \in Parents(N)} \alpha_{\mathbf{P}} \times \mathbf{ETX}(\mathbf{N}, \mathbf{P}) \tag{2}$$

   where $\sum_{P \in Parents(N)} \alpha_P = 1$.

3. computes the ratio of time during which it uses the medium for its transmissions, by taking into account the rate at which the data is sent:

$$\frac{T_N \times \sum_{P \in Parents(N)} \alpha_P \times ETX(N,P)}{\mathbf{DATA\_RATE}} \tag{3}$$

4. computes the energy spent to transmit all the traffic by multiplying the ratio of time during which it uses the medium, with the transmission power of its radio:

$$\frac{T_N \times \sum_{P \in Parents(N)} \alpha_P \times ETX(N,P)}{\text{DATA\_RATE}} \times \mathbf{P_{TX}(N)} \tag{4}$$

5. finally, N computes its remaining lifetime as the ratio between its residual energy, and the energy spent to receive and to transmit its traffic:

$$ELT(N) = \frac{\mathbf{E_{res}(N)}}{T_N \times \frac{\sum_{P \in Parents(N)} \alpha_P \times ETX(N,P)}{\text{DATA\_RATE}} \times P_{TX}(N)} \tag{5}$$

Once we know how to compute the *Expected Lifetime* of a node, we need now to:

1. find the nodes that will be the first ones to run out of energy (i.e., the bottlenecks), and advertise them along the paths (Section 4);
2. construct an energy-balanced topology using multiple parents (Section 5);
3. balance the traffic to all the parents, while taking into account the lifetime of each bottleneck (Section 6).

4. ELT for a Path: Computation and Advertisement with RPL

Since we aim at maximizing the network lifetime, we need to focus on the bottlenecks i.e., the nodes that are most likely to be the first ones to run out of energy. Thus, the weight of a path is the minimum ELT between all the traversed nodes. For example, in Fig. 3, the bottleneck of the path $G - D - B - A$ is the node B: it has the minimum ELT of the path. To choose the path advertising the maximum lifetime, a node needs to:

1. estimate its own impact on the ELT of the bottlenecks of a path;
2. send the information about the bottleneck along the path in the control packets (i.e., `DIOs`) in a compact manner.

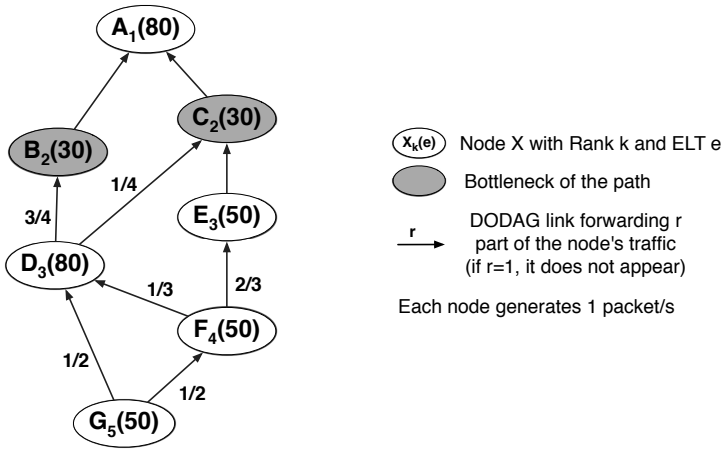


Figure 3: DODAG using multipath with ELT: repartition of loads

4.1. Path Computation

Let us consider that a node N has to associate with the DODAG. We will now describe how a node N estimates the impact of its traffic on the lifetime of a bottleneck.

If we take a look at Equation 5, we can notice that only the throughput (T_N) is dependent on the traffic injected by the new node. Hence, in order to estimate how N influences the lifetime of the bottleneck, we just add the traffic of N to the current throughput of the bottleneck. If B is the bottleneck, then:

$$new\_T_B = T_N + T_B \tag{6}$$

However, in a multipath scenario, we have to take into account that a node sends its traffic to several parents. Hence, only a part of its traffic will finally arrive at a specific bottleneck. We need to determine the ratio of traffic that a node forwards to a bottleneck.

Let $r_{N,B}$ be the ratio of traffic that N forwards to the bottleneck B. Given a parent P, N computes the ratio of traffic that will reach B through P as the product of the proportion of traffic that it sends to P (α_P) and the ratio of traffic that P forwards to the bottleneck B ($r_{P,B}$). Then, it will sum over all its parents these values. More formally:

$$r_{N,B} = \sum_{P \in Parents(N)} (\alpha_P \times r_{P,B}) \tag{7}$$

In the case when a node is itself the bottleneck, the ratio of the traffic that it forwards to the bottleneck is equal to 1 (i.e., $r_{B,B} = 1$).

Let us take an example. Consider node G and its parents D and F in Fig. 3. 3/4 of the traffic of D is forwarded through the bottleneck B. This is also the

case of $1/3 \times 3/4$ of the packets of F. Finally, the actual quantity of traffic of G that reaches B is: $1/2 \times 3/4 + 1/2 \times 1/3 \times 3/4$.

Thus, every node computes recursively the ratio of traffic forwarded to a bottleneck by summing over all the parents the ratio of traffic forwarded to it. Consequently, the new throughput of the bottleneck B can be estimated as:

$$new\_T_B = \mathbf{r_{N,B}} \times \mathbf{T_N} + T_B \tag{8}$$

Knowing this information, a node N can now estimate its impact on the lifetime of B:

$$ELT(B) = \frac{E_{res}(B)}{\mathbf{new\_T_B} \times \frac{\sum_{P \in Parents(B)} \times \alpha_P \times ETX(B,P) \times P_{TX}(B)}{\text{DATA\_RATE}}} \tag{9}$$

4.2. Compact `DIO` Advertisement

A node maintains a list of all its bottlenecks. This information has to be updated and included in each of its `DIOs`. Consequently, we are able to determine partially overlapping paths: several parents may lead to the same bottleneck.

However, just sending all the information needed to compute the ELT of one single bottleneck can be expensive in energy. We need to find a compact method to store it in the `DIOs`.

4.2.1. Compact packaging

We can adopt the same reasoning as for Equation 6. The only component that is dependent on the traffic added by a new node is the throughput. Hence, all the other elements can be compressed into a single constant, called *B_constant*.

In consequence, a `DIO` will contain a list of bottlenecks with the following information for each of them:

- **id:** the bottleneck id: IPv6 address (16 bytes), 6LowPAN address (4 bytes) or IEEE 802.15.4 short address (2 bytes);
- **ratio:** the normalized value of the ratio of traffic forwarded by the node to this bottleneck, which is computed recursively by each node: $r_{N,B}$ (1 bytes). We used a whole number of bytes because of its implementation simplicity;
- **existing_traffic:** the traffic forwarded by the bottleneck to the sink (normalized): T_B (1 bytes);
- **B_const:** the normalized value of the bottleneck constant (2 bytes) computed as:

$$B\_const = \frac{E_{res}(B)}{\frac{\sum_{P \in Parents(B)} \alpha_P \times ETX(B,P) \times P_{TX}(B)}{\text{DATA\_RATE}}} \tag{10}$$

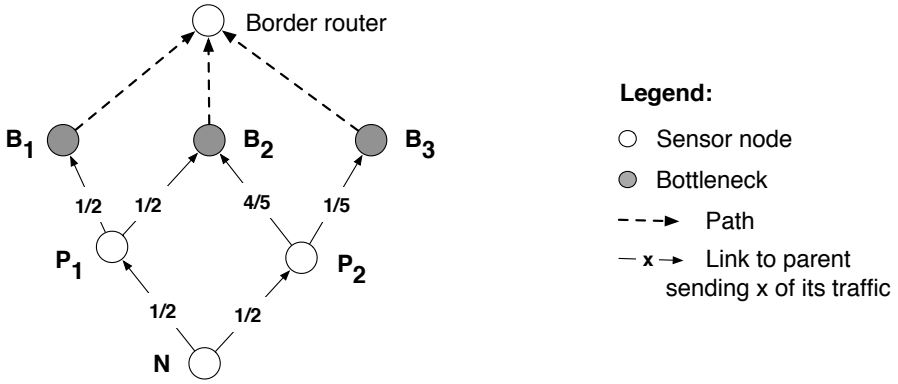


Figure 4: Number of bottlenecks

This constant, expressed in seconds, can have a very large range (from a few minutes to a few years). Thus, we chose to use a scientific notation: the precision is consequently relative. If we represent the significand on 13 bits, and the exponent on 3 bits (2 bytes in total), we are able to have a very good precision. Indeed, the minimum value represented is 1 second, while the maximum is 2590 years.

Practically, a tradeoff exists between the accuracy of the information and the overhead induced. On one side, if a node advertises too few bottlenecks, the lack of information could lead to less energy balanced paths. For example, in Fig. 4, if $P1$ only advertises $B1$ as a bottleneck, and $P2$ only $B3$, N could end up significantly reducing the lifetime of the bottleneck $B2$. On the other side, if the number of bottlenecks advertised is very large, it can induce more overhead in the network, and make the nodes consume more energy.

We will study this tradeoff in the performance evaluation section. We verified that advertising a limited number of bottlenecks is sufficient to practically balance the energy consumption in the network.

4.2.2. Worst case analysis

Let us now analyze the maximum number of bottlenecks that a node N can have. This will happen when all the bottlenecks are independent from each other, i.e., they are not shared by one or more nodes. The maximum number of bottlenecks will be reached when:

- all the bottlenecks are situated at one-hop from the border router and are distinct from each other (i.e., the furthest away possible from the nodes);
- all the nodes (but the bottlenecks) have the maximum number of parents k, and all the parents are distinct from each other. This condition

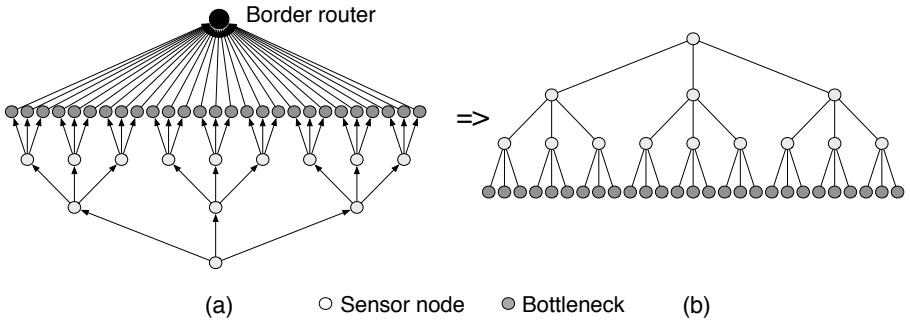


Figure 5: Maximum number of bottlenecks

makes a node advertise all the bottlenecks from all the parents as its own bottlenecks.

If $k = 3$, the topology created following these conditions corresponds to what we can see in Fig. 5a. Now, if we eliminate the border router from the figure, we can notice that we have a perfect k-ary tree (Figure 5b, with $k = 3$). The problem of finding the maximum number of bottlenecks is equivalent to computing the total number of leaves in this tree. If k represents the number of children of a node and *depth* the depth of the tree, then the total number of leaves is equal to k^{depth}. If we go back to our topology (i.e., we add another level when we put back the border router), the total number of bottlenecks will be $k^{depth-1}$.

In conclusion, the maximum number of bottlenecks that a node N can have is the maximum number of parents that a node has in the network at the power $depth - 1$, that is : $\left(\max_{M \in Network} |Parents(M)|\right)^{depth-1}$.

However, since in most cases, the bottlenecks will be shared among one or more nodes, this scenario will rarely occur in real-life topologies.

5. DODAG Construction in RPL with Useful Multiparents

We now define how a node may choose its preferred parent and how to construct a loop-free topology using the ELT metric.

5.1. Preferred Parent Selection

When choosing its preferred parent, a node must consider both its own lifetime and the lifetime of the bottlenecks, in order to estimate which of them becomes the new bottleneck. However, it is not possible to know the ratio of traffic that will be sent to each of the parents before actually choosing them. Hence, it is challenging to accurately estimate both the lifetime of the node, and the lifetime of the bottlenecks.

Algorithm 1: Preferred parent selection

```
   Data: N
   Result: preferred_parent of N
1  max_elt ← 0;
2  for P ∈ Parents(N) do
       // all the traffic is sent to P
3      α_P ← 1;
       // track the minimum ELT (all bottlenecks & myself)
4      min_elt ← min_{B∈Bottlenecks(P)} {ELT(B)};
5      min_elt ← min{min_elt, ELT(N)};
       // is this parent the best one?
6      if max_elt < min_elt then
7          max_elt ← min_elt;
8          preferred_parent ← P;
9      end
       // test now the other parents
10     α_P ← 0;
11 end
12 return preferred_parent;
```

In order to balance more efficiently the energy consumption, we consider the worst case, i.e., a node sends all its traffic to a single parent. In consequence, during the preferred parent selection, we underestimate the lifetime of the most constraint bottlenecks. In this way, we choose as preferred parent the node maximizing the lifetime of all the bottlenecks.

We consequently propose the Algorithm 1 to select the preferred parent (using the notation of Table 1). For each possible parent (i.e., a neighbor advertising a rank smaller than itself) a node N:

1. computes the ELT of all the bottlenecks advertised by a parent P, as if it will send all its traffic to that parent and save the minimum ELT value among all of them (line 4);

2. computes its own lifetime when choosing this parent, and verifies if N becomes the new bottleneck (line 5);

3. removes the traffic to this parent to test the other ones: it will iteratively test all the parents before taking a decision (line 10);

4. chooses as preferred parent the node that maximizes the lifetime of the bottleneck with the minimum ELT, itself included (lines 6, 7, 8).

At the beginning of the simulation, a node does not know the link quality to its neighbors. To avoid choosing as preferred parent the sender of the first received `DIO`, a node waits a period of time, to be able to receive several `DIOs` before taking a decision. During this period, the node adds in its parents set all

the `DIO` senders. After a certain time, it triggers the selection of the preferred parent using the algorithm presented above.

5.2. Loop Freeness

Sobrinho [36] has proved that for a distance vector protocol like RPL to be loop free, the routing metric must be strictly monotonic. Strict monotonicity implies that the weight of the path does not decrease when prefixed or appended by another path.

In our case, the weight of a path represents the minimum ELT on that path, i.e., the ELT of the bottleneck. Let the path p be prefixed with the path q. The ELT of the path p may:

1. remain stable if p keeps on presenting the lowest ELT;
2. be smaller than the ELT of p if the ELT of q is less than or equal to that of p.

In other words, ELT is not strictly monotonic, and is susceptible to create loops.

Moreover, RPL specifies that to obtain a loop free DODAG, the rank of the nodes must be strictly monotonically increasing from the border router towards the leaves. However, its exact computation is left to the Objective Function. To fully exploit the flexibility of RPL, we propose here to separate the metric that we use to construct the DODAG, from the metric used to compute the rank.

In consequence, we propose that a node computes its rank by adding a step value (*Rank_increase*) to the rank of its preferred parent. This value is derived from the ETX of the link to its preferred parent:

$$\begin{aligned} Rank(N) &= Rank(P) + Rank\_increase \\ Rank\_increase &= ETX(N,P) \times \texttt{MinHopRankIncrease} \end{aligned} \tag{11}$$

where P is the preferred parent of N and `MinHopRankIncrease` the RPL parameter [4].

We chose to use ETX and not a simple metric like hop count, in order to increase the number of parents that a node can have. With hop count, the rank increases linearly, which reduces the number of neighbors that could be used as alternative parents.

Clearly, such metric is monotonic, guaranteeing a loop-freeness. The weight of the path is the cumulative ETX on that path. When prefixed or appended with another path, its weight cannot decrease.

RPL forbids a node to consider as next hop a neighbor with a higher rank than itself [4]. In order to keep the loop-freeness, a node:

1. chooses its preferred parent using Algorithm 1;
2. computes its own rank based on the rank of its preferred parent;
3. removes from the parents set all the neighbors having a rank higher than itself;

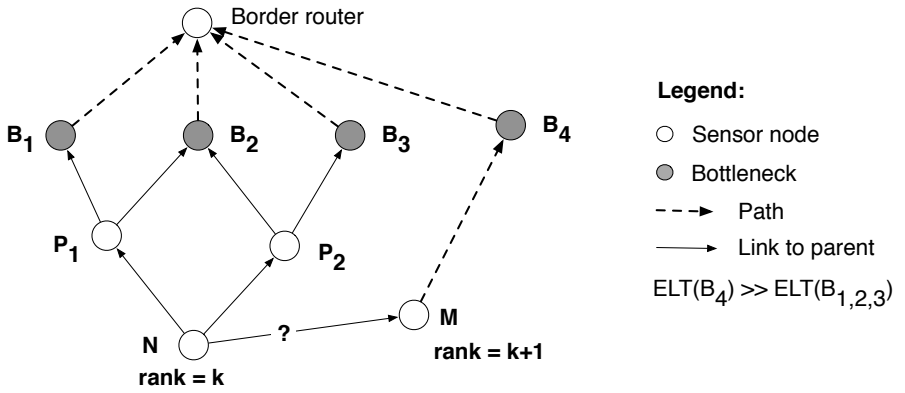


Figure 6: Discovering new paths

4. aggregates the bottlenecks, and updates the corresponding information in its `DIOs`;

5. ignores all `DIOs` from nodes advertising a higher rank than itself, in order to avoid the creation of loops.

5.3. Path Maintenance and Discovery of Better Bottlenecks

As we have seen, for the construction of the topology we use both ELT (to avoid the most constraint nodes in the network) and ETX (to ensure loop-freeness). One important question arises: what if there exists a path advertising a higher ELT, but the node will not receive this information, since the neighbor advertising it has a higher rank?

For example, in Fig. 6, the node M advertises the bottleneck B_4 with a lifetime greater than all the other bottlenecks. However, since $Rank(M) > Rank(N)$, N will never find this path, even though M is not situated in its sub-DODAG.

We propose here to allow a node to consider a `DIO` advertising a larger rank than itself, under the condition of maintaining a loop-free topology. However, we have to introduce some conditions to forbid such attachment if M is in the sub-DODAG of N.

Let us assume that M advertises a bottleneck B. If B is not one of the bottlenecks of N, this means the sub-DODAGs are at least partially disjoint: another path exist throughs the bottleneck B.

Hence, a node N considers M with *Rank(M)* $\geq$ *Rank(N)* as preferred parent if it satisfies the following conditions:

1. the bottlenecks advertised by M are not also bottlenecks of N:

$$Bottlenecks(M) \nsubseteq Bottlenecks(N)$$

2. M advertises at least one bottleneck $B_{new} \in Bottlenecks(M)$ who's lifetime is greater than the maximum ELT of all the bottlenecks of N:

$$ELT(B_{new}) > \max_{B \in Bottlenecks(N)} \{ELT(B)\} \quad (12)$$

Moreover, we need to make sure that the ELT of B_{new} will not become smaller than the maximum ELT of all the bottlenecks of N once the traffic of N will be sent to M. Equation (12) becomes:

$$ELT(B_{new}) > \max_{B \in Bottlenecks(N)} \{ELT(B) | r_{N,B} = 0\} \quad (13)$$

where $r_{N,B}$ represents the ratio of traffic forwarded by N to the bottleneck B (cf. Equation 7).

These conditions allow a node to consider a `DIO` from a neighbor situated deeper in the DODAG, and hence, find better paths, while maintaining the loop freeness of the topology.

6. Energy Balancing

After constructing the DODAG with useful multiparent, we now have to address the problem of the forwarding plane. A node must split its traffic among all the available parents, while taking into account the lifetime of each bottleneck. We have to propose a heuristic to determine the weights associated to each parent so that all the paths have the same lifetime.

Each node selects locally its set of parents, but focuses on maximizing the minimum lifetime of all the bottlenecks. These bottlenecks are often the 1-hop neighbors of the sink because of the well-known funneling effect [37]. Besides, it splits its traffic among all the bottlenecks to limit its impact on their lifetime. Thus, by applying a local rule, we reach a global objective which consists in maximizing the network lifetime (i.e. maximizing the minimum ELT among all the bottlenecks).

The first solution would consist in modeling the problem with linear inequalities:

- the weights of all parents represent the set of unknown variables;
- the node must compute the lifetime of each bottleneck, taking into account its own traffic;
- the optimal solution would consist in maximizing the minimal lifetime of all the bottlenecks.

This linear formulation may also be injected into a linear solver [38]. However, we consider that the extra memory and CPU consumed by this solution are inadequate for small sensor nodes.

Algorithm 2: Load balancing

Data: N, γ - the step of the load increase
Result: compute $\{\alpha_P\}_{P \in Parents(N)}$ — the ratio of traffic to send to each parent;

```
1  for i = 1 to γ^-1 do
2      max_elt ← 0;
3      for P ∈ Parents(N) do
           // test this parent P with its new weight
4          α_P ← α_P + γ;
           // track the min ELT with this new weight
5          min_elt ← min_{B∈Bottlenecks(P)} {ELT(B)};
6          min_elt ← min{min_elt, ELT(N)};
           // is this parent the best one?
7          if max_elt < min_elt then
8              max_elt ← min_elt;
9              parent_max ← P;
10         end
           // test each parent before taking a decision
11         α_P ← α_P − γ;
12     end
13     α_parent_max ← α_parent_max + γ;
14 end
```

Hence, we present here a greedy heuristic. A node N has to distribute the load to each parent so that it balances the expected lifetime of the corresponding bottlenecks. Consequently, a node divides its traffic in $\frac{1}{\gamma}$ equal fractions, and assigns sequentially each fraction to the parent which maximizes the minimum lifetime among all its bottlenecks.

Algorithm 2 defines more formally the heuristic:

1. First, N tries to find the best parent to send γ of its traffic by testing iteratively each parent (line 3):

   a) It computes the minimum ELT that would be obtained by increasing the weight of this parent by γ (line 4). It considers the lifetime of each bottleneck (line 5) and of its own (line 6);

   b) If this minimum value maximizes the network lifetime, it saves the current parent as the best one (line 7-10);

   c) N removes the hypothetical γ from the ratio of traffic to be sent to this parent (α_P), in order to test the other parents before definitively setting the new weight (line 11);

2. Finally, N assigns γ of the total traffic to the best parent (line 13) and re-starts with the next γ (line 1).

It is obvious that the smaller γ is, the closer to the optimal the solution will be. To help accomplish this, we put the condition for γ to be smaller than the

inverse of the maximum number of parents. Indeed, if we reconsider the example from Fig. 4. N has two parents: P_1 and P_2. The traffic load associated to each of the parents is $\frac{1}{2}$. If γ would be greater than the inverse of the maximum number of parents (i.e., $\gamma > \frac{1}{2}$), an optimal repartition of the weights would simply be impossible.

6.1. Complexity

For each parent, the algorithm searches the bottleneck having the smallest ELT. Since the minimum value in a list can be found in $O(n)$ and this value must be searched for each parent, the complexity of the greedy algorithm is $O(n * n) = O(n^2)$.

Some optimizations are possible in the implementation. In particular, for $i > 1$, a node has to recompute the minimum ELT (lines 4-11) only for the parent which was the best one at the previous iteration $(i - 1)$. Indeed, the possible weight of all the other parents has already been considered in the previous iteration.

In conclusion, a node has to execute the following number of ELT computations and comparisons:

$$nb\_bottlenecks * nb\_parents + nb\_bottlenecks * (\gamma^{-1} - 1) \tag{14}$$

where `nb_parents` denotes the number of parents and `nb_bottlenecks` the maximum number of bottlenecks to advertise. This time complexity is very reasonable (with e.g. 4 parents and 8 bottlenecks).

6.2. Correctness: $(1 + \gamma)$ - approximation

As previously said, the smallest γ is, the closest to the optimal the solution will be. However, there might be cases when the optimal values for the parent weights are not found.

Let us consider still the example in Fig. 4. P_2 has to send $\frac{4}{5}$ of its traffic to B_2 and $\frac{1}{5}$ to B_3. Let us take $\gamma = \frac{1}{2}$, the maximum value allowed (we put the condition that γ should be smaller than the inverse of the maximum number of parents). Algorithm 2 will first allocate $\frac{1}{2}$ of its traffic to parent B_2. Then, the optimal choice would be to allocate another $\frac{3}{10}$ of its traffic to B_2 and the rest of $\frac{1}{5}$ to B_3. However, it will allocate the last γ of its traffic $(\frac{1}{2})$ to B_2, hence, being sub-obtimal. Indeed, 100% of the traffic will be forwarded to B_2 and 0% to B_3.

Theorem 1. *The presented greedy algorithm is a $1 + \gamma$ approximation.*

Let us give an intuitive explanation. The full proof can be found in the Appendix A.

We first take the particular case when just one node in the network (N) has packets to send. The greedy algorithm makes the choice that seems optimal at each step, i.e., when each γ is distributed. Since γ is a constant, it means that the algorithm might have a problem distributing the last γ of the traffic.

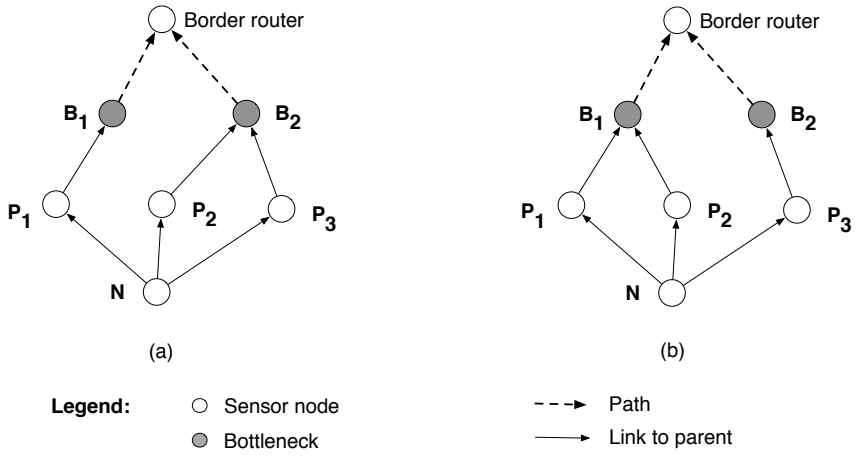


Figure 7: Correctness

Let us assume that the last γ of the traffic is sent by N to the parent P_1, which is not optimal. If we focus on the most constraint bottleneck (e.g., B_1), we can distinguish two cases:

1. P_1 is the only parent of N forwarding traffic to the bottleneck B_1 (Fig. 7a). B_1 will be overloaded with all the extra traffic from N. Hence the algorithm is a $(1+\gamma)$ - approximation.

2. All the traffic from P_1 plus some of the traffic from other parents (e.g., P_2) is forwarded to the bottleneck B_1 (Fig. 7b). This means that B_1 will be overloaded with the traffic sent from P_1. In the same time, less traffic than the optimal is sent on the other parents, and so, B_1 will be less loaded with that corresponding traffic.

In conclusion, the bottleneck B_1 will be overloaded with at most γ of the traffic of N, hence the algorithm is a $(1+\gamma)$ - approximation.

The reasoning holds also when there is more than just one node transmitting in the network. If each node is at $(1+\gamma)$ from the optimal, globally it will also be at $(1+\gamma)$ from the optimal.

6.3. Maintaining the Stability

Frequent preferred parent changes can induce instability in the network. Some would argue that these instabilities are beneficial since it shows that the network adapts to changes. However, DODAG reconfigurations have a strong impact on the performance of the network, especially on the end-to-end packet delivery ratio and on the energy consumption [35].

We address this problem as follows:

1. we change the preferred parent only when it is not useful anymore;

2. we gradually redirect the traffic from one parent to the others when re-computing the loads.

6.3.1. Changing the preferred parent

When a node changes its preferred parent, it triggers the reset of the trickle timer. This means that the node sends control packets more frequently and hence, consumes more energy. Besides, parent changes imply also traffic redirection. Since the metric depends on the traffic forwarded by the bottlenecks, all the nodes must in this case update their path metric.

We change the conventional meaning of the preferred parent (i.e., the parent offering the best path to the border router) to a parent that a node uses to compute its rank, without necessarily being the best one. We aim here at reducing the number of preferred parent changes while keeping up-to-date information about all the parents.

We propose to adopt a conservative approach in the maintenance of the parent list:

1. a node removes a parent P only if this parent is not useful anymore, i.e., the traffic that is actually forwarded to P (α_P) is smaller than a threshold value. If P was the preferred parent, the node re-executes the Algorithm 1 to select the new preferred parent and updates accordingly its rank;

2. any neighbor with a lower rank is inserted in the parent list. The node N updates dynamically the weight of all its parents (α_i, $i \in Parents(N)$) every time a new `DIO` is received.

6.3.2. Redirecting the traffic

A small variation in the link quality estimation is sufficient to change the preferred parent (local maximum among the neighbors). Since the inaccurate estimation of the radio link quality has a significant impact on the stability of RPL [35], we need to also take into account small and transient inaccuracies.

We propose here to progressively redirect the traffic when recomputing the weights for each parent, instead of completely replacing the old values with the newly computed ones. We set a maximum value for the increase/decrease of the parent weight between two estimations. We let the network adapt progressively to this change, while avoiding sudden redirections due to metric variations.

We propose Algorithm 3 for re-computing the parent weights:

1. For each parent P, a node N computes the difference between the old weight and the new one (line 2);

2. N adds to the old weight (old_α_P) the normalized value of the difference computed in the previous step ($diff_P$) (line 5).

This algorithm gradually redirects the traffic, reducing thus, useless oscillations in the network and saving energy.

Algorithm 3: Parent weight normalization

Data: N, α_{max} – the maximum increase/decrease in the parent weight,
$\{old\_\alpha_P\}_{P \in Parents(N)}$ – the current ratio of traffic for each parent,
$\{new\_\alpha_P\}_{P \in Parents(N)}$ – the newly computed ratio of traffic for each parent;

Result: $\{\alpha_P\}_{P \in Parents(N)}$ – the new ratio of traffic, after normalization;

1 **for** $P \in Parents(N)$ **do**
  `// compute the difference between the old weight and the new one`
2 $diff_P \leftarrow old\_\alpha_P - new\_\alpha_P$;
3 **end**

4 **for** $P \in Parents(N)$ **do**
  `// limit the weight change to` α_{max} `and normalize the values`
5 $\alpha_P \leftarrow old\_\alpha_P + diff_P \times \alpha_{max} \times \frac{1}{\max\limits_{P \in Parents(N)} diff_P}$;
6 **end**

6.4. Multipath to improve the reliability

When measures are reported periodically by a sensor, packets are very small and often do not need fragmentation, such as for the smart metering scenario [39]. In this case, multipath is an asset: packets losses may be considered independent among the different paths. In other words, while the first measure may be dropped, a second one may use a different path and delivered to the border router.

However, if fragmentation is required, for instance to retrieve a whole data interval, or for a remote diagnostic, we should implement one of the following solutions:

1. use network coding to allow the recovery of some packet losses [40];
2. ensure the same path is always used for the same flow. Instead of considering all the packets independently, we may guarantee the same parent is selected for a given pair source/destination address.

7. Performance Evaluation

We simulated RPL using WSNet, an efficient event-driven simulator dedicated to WSN, which has been extensively evaluated [41]. The results are averaged over 30 simulations with different random topologies: the sink is situated in the center, while the location of each node is chosen randomly in the simulation area. For the traffic, we considered usual CBR convergecast flows.

At the PHY layer, we used the path-loss shadowing model, calibrated with the scenario FB6 (indoor real deployment) presented in [42]: shadowing, path loss = 1.97, standard deviation = 2.0, $Pr(2m) = -61.4dBm$, when the environment is static but accurately modeled.

We configured RPL as illustrated in Table 2. We compared the following variants:

Table 2: Simulation parameters

Parameter	Value
Simulation duration	$3600s$
Number of nodes	50
Number of bottlenecks advertised	10
Load balance step γ	0.1 (10%)
Simulated area	300m x 300m
Traffic type, rate	CBR, 1 pkt/min
Data packet size	127 bytes (incl. MAC headers)
RPL	MinHopRankIncrease = 128
Trickle	$Imin = 2^7 ms$, $Imax = 16$, $k = 10$
MAC layer	802.15.4 beacon mode
MAC parameters	BO=7, S=2
Energy consumption	CC2420 datasheet

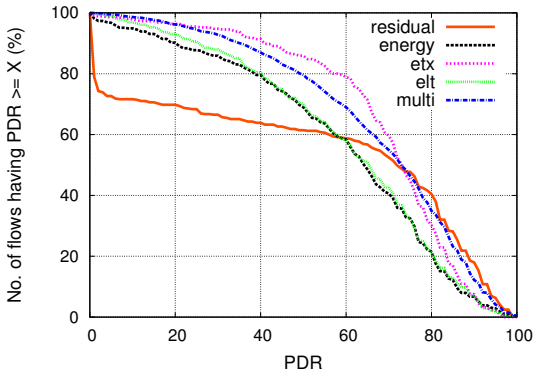
1. a standard version of RPL: only the preferred parent is used to forward packets;
2. multipath version of RPL: a node forwards the traffic fairly to all its parents, based on the lifetime of their bottlenecks. This version is possible only when using the ELT metric.

We also compared the following routing metrics:

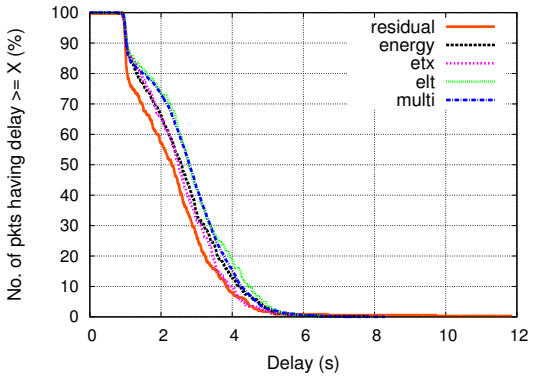
1. residual energy ([7]): the remaining energy of a node (node metric). We chose this metric since it helps avoiding the most energy constraint nodes.
2. ETX using Hysteresis Objective Function ([43]): the average number of transmissions for an acknowledged packet (link metric). We chose this metric since it constructs energy-efficient routes.
3. energy ([24]): a linear combination of ETX and the residual energy. This metric combines in a simplistic manner the residual energy and the quality of the links.
4. ELT: our metric, which directly exploits the remaining lifetime with the same conditions.

We measured the following metrics:

- Packet Delivery Ratio (PDR): ratio of packets received by the sink;
- End-to-end delay: delay between the packet generation and its reception by the sink (considered only for delivered packets);
- Stability: the number of preferred parent changes;
- Energy consumption: the energy consumed by the nodes during the whole simulation period;
- Network lifetime: time before the first node dies when considering only the energy consumed by the CC2420 chipset (differentiating the idle / sleep / tx and rx states).



(a) CCDF of the end-to-end PDR



(b) CCDF of the end-to-end delay

Figure 8: Reliability

7.1. Comparison of the Different Metrics and RPL Versions

7.1.1. Packet Delivery Ratio

Fig. 8a illustrates the complementary cumulative distribution function (CCDF) of the PDR for all the flows.

The residual energy presents the worst reliability: some nodes with bad radio links are selected only because they have a large residual energy. These bad links have a negative impact on the packet delivery ratio.

ETX presents the highest reliability: only the best links are used to route the packets. The metric *energy* represents, as expected, a tradeoff between the residual energy and the ETX metrics.

Finally, our multipath version of ELT achieves almost the same reliability as ETX. Our metric selects routes with the largest residual energy, without impacting negatively the reliability.

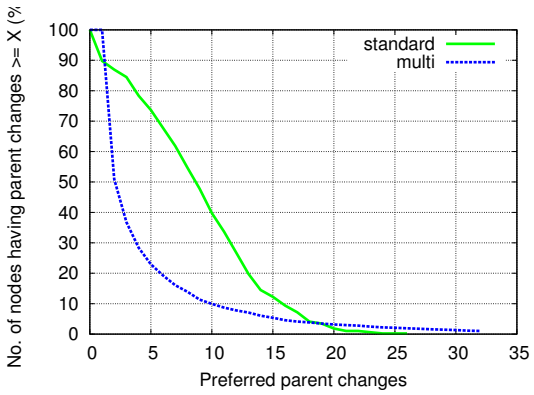


Figure 9: CCDF of the number of preferred parent changes during 1 hour – ELT metric

7.1.2. End-to-end Delay

Fig. 8b illustrates the CCDF of the end-to-end delay for of all the received packets. This delay is quite insensitive to the metric used for routing.

Indeed, the limited number of retransmissions at the MAC layer makes that bad radio links tend to reduce the reliability (i.e., the packet is dropped) while having a marginal impact on the delay. In particular, the retransmission delay is practically much shorter than the buffering delay: since the network operates at low duty-cycle ratios, a packet is buffered a long time before the node wakes up and sends it to one of its parents.

7.1.3. Routing Stability

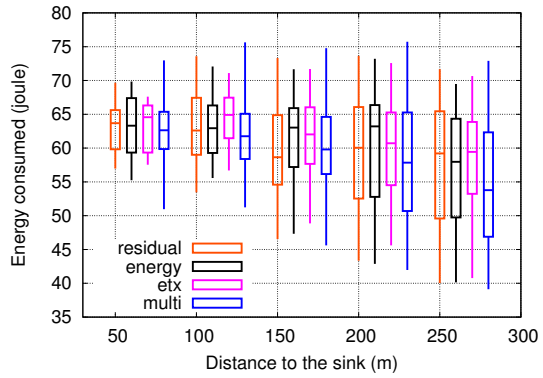
We have measured the stability of the network as the number of preferred parent changes (Fig. 9). We count the number of parent changes during 1 hour of simulation, and plot the associated CCDF.

The standard version exhibits high instability: most of the nodes change frequently their parents. During the simulation, one half of the nodes change at least 8 times their preferred parent. For each change, the trickle timer is reset and the `DIOs` are transmitted more frequently.

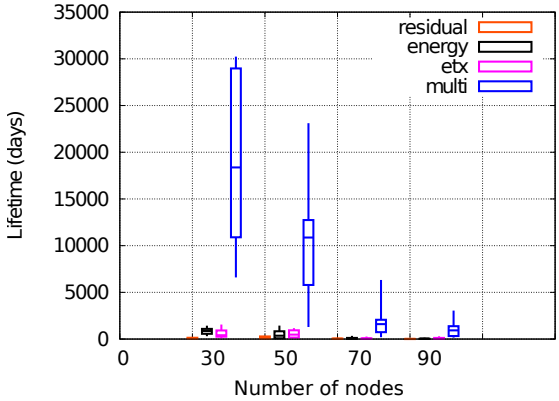
On the contrary, the multipath version improves significantly the stability. By forbidding sudden parent weight changes, a node smooths the traffic redirection. More than 80% of the nodes change at most 4 times their preferred parent.

7.1.4. Energy efficiency

Fig. 10a represents the box plot of the average energy consumed by the nodes during the whole simulation, against their physical distance from the sink (i.e., the border router).



(a) Energy consumption of the nodes in function of their physical distance from the sink



(b) Network Lifetime (time until the first node dies) in function of the density

Figure 10: Energy Efficiency

The residual energy and the multipath ELT improve both the energy efficiency. However, the residual energy presents also the worse reliability. Since less packets are forwarded, they also consume less energy.

Finally, we evaluated the network lifetime in function of the density. We increased the number of nodes within the same simulation area to isolate the impact of the network diameter and that of the density. The depth of the DODAG varies from 2 in high densities (with 30 nodes), to 7 in lower densities (90 nodes).

We can observe in Fig. 10b that the multipath ELT clearly outperforms the standard RPL. Multipath routing helps balancing more accurately the energy: routing decisions are not binary, and the traffic is spread to all the bottlenecks. Besides, our strategy of re-allocating the traffic to each parent based on their

lifetime, is conservative and avoids sudden redirections.

We can notice that even thought the multipath ELT has the biggest energy consumption in the worst case scenario (Fig. 10a), it still manages to have the best lifetime (Fig. 10b). This is due to the fact that the algorithm consumes more energy during the bootstrap period, which gains more lifetime on the long run.

7.2. Tuning the Parameters

7.2.1. Number of Bottlenecks Advertised

We have here investigated the impact of the number of bottlenecks included in the `DIO`. A too small number means a node under-estimates the impact of its traffic on the other bottlenecks.

In Fig. 11a we plotted the CCDF of the end-to-end PDR for all the flows, in function of the maximum number of bottlenecks advertised by a node. We can see that the PDR is almost the same, no matter how many bottlenecks a node advertises. Indeed, a node takes into account both its ELT and the ELT of the bottlenecks. Since ELT accounts for the ETX, it implicitly takes into account the reliability. The number of bottlenecks has rather an impact on the energy consumption.

In Fig. 11b we plotted the energy consumed by the nodes during the whole simulation, against their physical distance from the sink (i.e., the border router). No matter the number of bottlenecks advertised, our solution manages to construct an energy balanced topology. Our solution is hopefully not too sensitive to the exact number of bottlenecks to advertise. A small number of bottlenecks is sufficient to even balance the energy consumption in the network.

Here, we see that adverting 8 bottlenecks represents the optimal value. Without increasing too much the `DIO` size, we manage to not underestimate the consumption of the *secondary* bottlenecks.

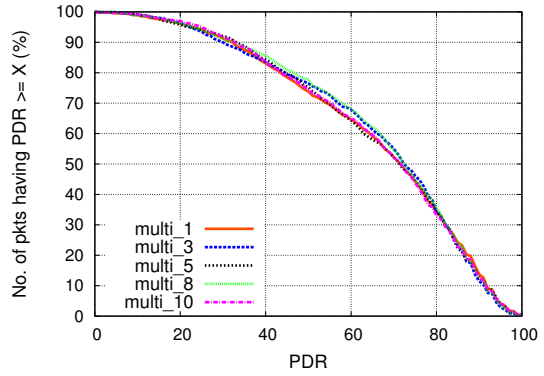
Finally, Fig. 11c illustrates the impact of the number of bottlenecks on the network lifetime. With only one bottleneck we manage to estimate quite accurately the network lifetime. Besides, only one bottleneck means we reduce the number of variables to maintain to estimate the ELT of each path: we consequently reduce the instability.

However, increasing the number of bottlenecks tends to also balance more finely the energy consumption along all the paths. This finer estimation tends to counter-balance the number of parent changes. 8 represents the optimal value for the number of bottlenecks to include in the `DIOs`.

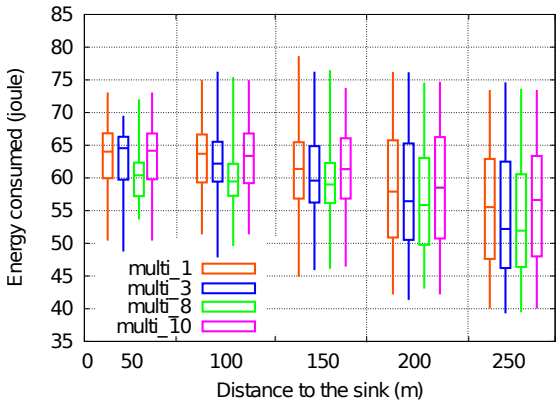
7.3. Impact of γ on the Network Performance

We finally studied the impact of the load balancing parameter γ which is used to compute the traffic to send to each parent.

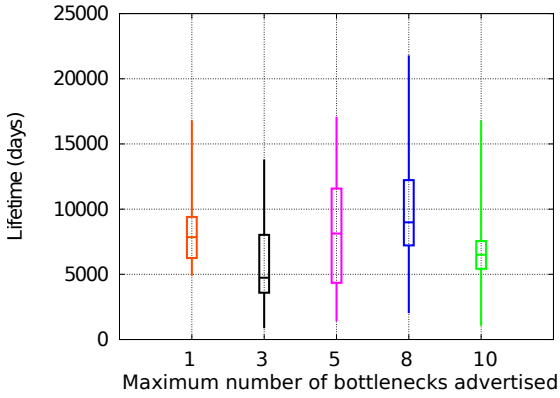
We can see in Fig.12a that it has no impact on the end-to-end packet delivery ratio. Indeed, the multipath ELT succeeds to find reliable routes, even if the energy is not so well balanced (too small γ value).



(a) CCDF of the end-to-end PDR in function of the maximum no. of bottlenecks advertised (n)

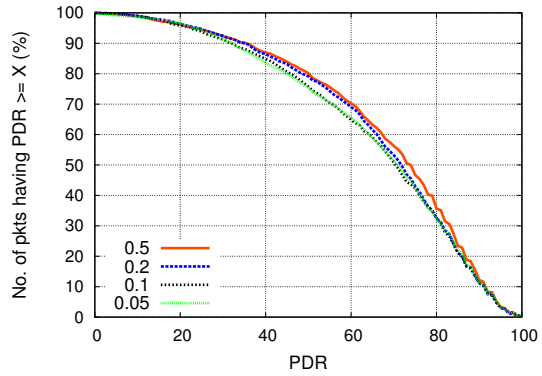


(b) Average energy consumption per node

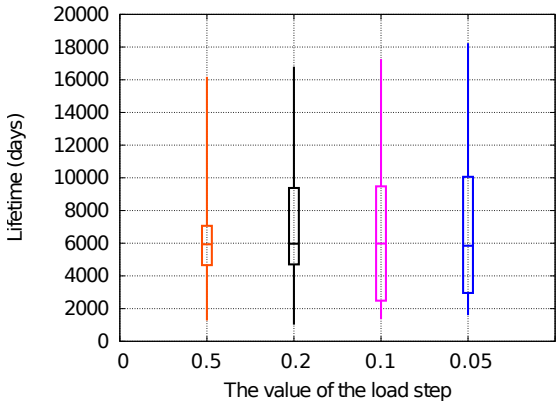


(c) Expected Lifetime (time until the first node dies)

Figure 11: Impact of the nb. of bottlenecks to include in the `DIO`



(a) CCDF of the end-to-end PDR



(b) Expected Lifetime
(time until the first node dies)

Figure 12: Impact of the parameter γ

Moreover, when γ is sufficiently small, it has no impact on the lifetime of the network. We can notice in Fig.12b that when $\gamma = 0.5$ the lifetime is slightly smaller. Since in most of the cases a node has more than two parents, it will be more difficult for the algorithm to optimally distribute the last 0.5% of the traffic. This sub-optimal distribution negatively impacts the lifetime.

8. Conclusion and Perspectives

We proposed here an energy-balanced version of RPL. First, we constructed a DAG based on the ELT metric, which accurately estimates the lifetime of all the routes toward the border router. By selecting as parents the nodes with the strongest paths (i.e., larger ELT), we improve the network lifetime.

Second, we proposed a multipath approach to fully exploit the DAG structure. A node exploits all its parents, assigning a weight of traffic to each of them. In this way, a node distributes fairly the energy consumption among all the paths toward the border router. Since each node receives fairly a quantity of traffic to forward, energy consumption is well balanced.

Finally, a preferred parent is removed only when it becomes useless (i.e., it forwards no traffic). We also dealt efficiently with inaccuracies in the metric estimation. Indeed, the radio link quality is stochastic, and the routes constructed by RPL should not change if the radio link quality has not *significantly* changed. We manage to limit the number of parent changes and thus, to reduce the energy consumption of the nodes.

In a future work, we plan also to experimentally evaluate this new multipath energy-balancing version of RPL, to verify it operates efficiently *in vivo*. In an experimental environment, we could deal with a faster varying radio channel, and asymmetrical links. Indeed, a faster varying radio channel could have an impact on the convergence of the ELT routing metric, since it uses the current traffic conditions to compute the expected lifetime of a node. However, we conjecture that the metric will converge in a reasonable amount of time.

Regarding the presence of asymmetrical links, ELT exploits ETX for the estimation of the link quality and thus, should be able to handle them. We recall that ETX accounts for the PDR in both directions of a link. Hence, it suffices to have an accurate implementation of ETX.

Also, since we only considered the case without packet fragmentation, we could relax this hypothesis. In an environment with frequent fragmentation we would have to improve our solution by including a network coding technique to improve the reliability of the network.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Computer Networks 38 (4) (2002) 393–422. doi:http://dx.doi.org/10.1016/S1389-1286(01)00302-4.

[2] T. Watteyne, et al., From MANET To IETF ROLL Standardization: A Paradigm Shift in WSN Routing Protocols, Communications Surveys Tutorials, IEEE 13 (4) (2011) 688–707. doi:10.1109/SURV.2011.082710.00092.

[3] P. Karkazis, et al., Design of primary and composite routing metrics for rpl-compliant wireless sensor networks, in: TEMU, IEEE, 2012, pp. 13–18. doi:10.1109/TEMU.2012.6294705.

[4] T. Winter, et al., RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks, RFC 6550, IETF (2012).

[5] K. Staniec, G. Debita, An optimal sink nodes number estimation for improving the energetic efficiency in wireless sensor networks, Elektronika ir

elektrotechnika 19 (8) (2013) 115–118. doi:http://dx.doi.org/10.5755/j01.eee.19.8.5407.

[6] D. D. Couto, et al., A high-throughput path metric for multi-hop wireless routing, in: MobiCom, ACM, 2003. doi:10.1145/938985.939000.

[7] P. Kamgueu, et al., Energy-Based Routing Metric for RPL, Research Report RR-8208, INRIA (2013).
URL http://hal.inria.fr/hal-00779519

[8] M. Conti, et al., Reliable and efficient forwarding in ad hoc networks, Ad Hoc Networks 4 (3) (2006) 398 – 415. doi:http://dx.doi.org/10.1016/j.adhoc.2004.10.006.

[9] L. Reddeppa Reddy, S. Raghavan, SMORT: Scalable multipath on-demand routing for mobile ad hoc networks, Ad Hoc Networks 5 (2) (2007) 162 – 188. doi:http://dx.doi.org/10.1016/j.adhoc.2005.10.002.

[10] O. Banimelhem, S. Khasawneh, GMCAR: Grid-based multipath with congestion avoidance routing protocol in wireless sensor networks, Ad Hoc Networks 10 (7) (2012) 1346 – 1361. doi:http://dx.doi.org/10.1016/j.adhoc.2012.03.015.

[11] M. Radi, et al., Multipath routing in wireless sensor networks: Survey and research challenges, Sensors 12 (1). doi:10.3390/s120100650.

[12] R. Malekian, A. H. Abdullah, N. Ye, A novel resource reservation based on cross layer design over mobile internet protocol version 6, Journal of Internet Technology 15 (3) (2014) 391–404.

[13] R. Malekian, A. H. Abdullah, Traffic engineering based on effective envelope algorithm on novel resource reservation method over mobile internet protocol version 6, International Journal of Innovative, Computing, Information and Control 8 (9) (2012) 6445–6459.

[14] R. Malekian, A. H. Abdullah, N. Ye, Novel packet queuing algorithm on packet delivery in mobile internet protocol version 6 networks, Applied Mathematics and Information Sciences 7 (3) (2013) 881–887.

[15] O. Iova, F. Theoleyre, T. Noel, Maximizing the lifetime of wireless sensor networks with rpl, in: Wireless and Mobile Networking Conference (WMNC), IEEE/IFIP, Vilamoura, Algarve, Portugal, 2014.

[16] JP. Vasseur et al., Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL), RFC 6552, IETF (2013).

[17] B. Pavković, et al., Multipath Opportunistic RPL Routing over IEEE 802.15.4, in: MSWiM, ACM, 2011. doi:10.1145/2068897.2068929.

[18] K.-S. Hong, L. Choi, DAG-based multipath routing for mobile sensor networks, in: ICTC, IEEE, 2011. doi:10.1109/ICTC.2011.6082593.

[19] J. Vasseur, et al., Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks, RFC 6551, IETF (2012).

[20] O. Chipara, et al., Real-time power-aware routing in sensor networks, in: Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on, 2006, pp. 83–92. doi:10.1109/IWQOS.2006.250454.

[21] H. Yoo, et al., GLOBAL: A Gradient-based Routing Protocol for Load-balancing in Large-scale Wireless Sensor Networks with Multiple Sinks, in: ISCC, IEEE, 2010. doi:10.1109/ISCC.2010.5546784.

[22] J.-H. Chang, L. Tassiulas, Maximum lifetime routing in wireless sensor networks, IEEE/ACM Trans. Netw. 12 (4) (2004) 609–619. doi:10.1109/TNET.2004.833122.
URL http://dx.doi.org/10.1109/TNET.2004.833122

[23] H. Liu, et al., PWave: A multi-source multi-sink anycast routing framework for wireless sensor networks, in: Networking, IFIP, Springer-Verlag, 2007.

[24] L.-H. Chang, et al., Energy-efficient oriented routing algorithm in wireless sensor networks, in: SMC, IEEE, 2013. doi:10.1109/SMC.2013.651.

[25] D. Ganesan, et al., Highly-resilient, energy-efficient multipath routing in wireless sensor networks, SIGMOBILE Mobile Computing Communications Review. 5 (4) (2001) 11–25. doi:10.1145/509506.509514.

[26] Y. Chen, N. Nasser, Energy-balancing multipath routing protocol for wireless sensor networks, in: QShine, ACM, 2006. doi:10.1145/1185373.1185401.

[27] T. Ming-hao, et al., Multipath routing protocol with load balancing in WSN considering interference, in: ICIEA, IEEE, 2011. doi:10.1109/ICIEA.2011.5975744.

[28] R. Kacimi, et al., Load Balancing Techniques for Lifetime Maximizing in Wireless Sensor Networks, Ad Hoc Networks 11 (8) (2013) 2172 – 2186. doi:http://dx.doi.org/10.1016/j.adhoc.2013.04.009.

[29] B. Yahya, et al., REER: Robust and Energy Efficient Multipath Routing Protocol for Wireless Sensor Networks, in: GLOBECOM, IEEE, 2009. doi:10.1109/GLOCOM.2009.5425587.

[30] E. Ghadimi, et al., Opportunistic routing in low duty-cycle wireless sensor networks, ACM Transactions on Sensor Networks 10 (4) (2014) 67:1–67:39. doi:10.1145/2533686.

[31] K. Lee, et al., Satisfying the target network lifetime in wireless sensor networks, Computer Networks 65 (0) (2014) 41 – 55. doi:http://dx.doi.org/10.1016/j.comnet.2014.03.001.

[32] L. Mo, et al., Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest, in: SenSys, ACM, 2009. doi:10.1145/1644038.1644049.

[33] G. Barrenetxea, et al., Sensorscope: Out-of-the-box environmental monitoring, in: IPSN, IEEE, 2008. doi:10.1109/IPSN.2008.28.

[34] X. Mao, et al., Citysee: Urban CO2 monitoring with sensors, in: INFOCOM, IEEE, 2012. doi:10.1109/INFCOM.2012.6195530.

[35] O. Iova, et al., Stability and efficiency of RPL under realistic conditions in wireless sensor networks, in: PIMRC, IEEE, 2013. doi:10.1109/PIMRC.2013.6666490.

[36] J.-L. Sobrinho, Network routing with path vector protocols: theory and applications, in: SIGCOMM, ACM, 2003. doi:10.1145/863955.863963.

[37] G.-S. Ahn, et al., Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks, in: SenSys, ACM, New York, NY, USA, 2006, pp. 293–306. doi:10.1145/1182807.1182837.
URL http://doi.acm.org.gate6.inist.fr/10.1145/1182807.1182837

[38] I. Kadayif, et al., An integer linear programming-based tool for wireless sensor networks, Journal of Parallel and Distributed Computing 65 (3) (2005) 247 – 260. doi:http://dx.doi.org/10.1016/j.jpdc.2004.04.004.

[39] W. Luan, D. Sharp, S. Lancashire, Smart grid communication network capacity planning for power utilities, in: Transmission and Distribution Conference and Exposition, IEEE, 2010. doi:10.1109/TDC.2010.5484223.

[40] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. Harris, M. Zorzi, Synapse: A network reprogramming protocol for wireless sensor networks using fountain codes, in: Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), IEEE, 2008, pp. 188–196. doi:10.1109/SAHCN.2008.32.

[41] E. Ben Hamida, et al., On the Complexity of an Accurate and Precise Performance Evaluation of Wireless Networks using Simulations, in: MSWiM, ACM, 2008. doi:10.1145/1454503.1454570.

[42] Y. Chen, A. Terzis, On the implications of the log-normal path loss model: an efficient method to deploy and move sensor motes, in: SenSys, ACM, 2011. doi:10.1145/2070942.2070946.

[43] Gnawali, O. and Levis, P., The Minimum Rank with Hysteresis Objective Function, RFC 6719, IETF (2012).

Appendix A. Correctness: $(1+\gamma)$ - approximation

Theorem 2. *The presented greedy algorithm is a $1+\gamma$ approximation.*

Proof. Let t_{NB} be the traffic from node N to bottleneck B, k the maximum number of parents that a node can have, and $\overline{x}$ the notation for the real value of x and x_{opt} the notation for the optimal value for x, $\forall x$. Table 1 p. 9 depicts the rest of the notation.

The fact that the greedy algorithm is a $1+\gamma$ approximation means that the traffic that reaches a bottleneck is at most the optimal_value * $1+\gamma$. Hence, the theorem is equivalent to:

$$\forall B_i \in Bottlenecks : \sum_{\forall N_j \in Network} \overline{t_{N_j B_i}} \leq (1+\gamma) \sum_{\forall N_j \in Network} t_{N_j B_i\ opt} \quad \text{(A.1)}$$

If N is the only node in the network generating packets, the Equation A.1 is equivalent to:

$$\forall B_i \in Bottlenecks : \overline{t_{NB_i}} \leq t_{NB_i\ opt} + \gamma \times T_N \quad \text{(A.2)}$$

Let us assume that the last γ of the traffic is sent by N to the parent P_1, which is not optimal. If we focus on the most constraint bottleneck (e.g., B_1), we distinguish the following cases:

1. There are other parents of N, besides P_1, that forward traffic to B_1:
   $\exists P_i \in \{Parents(N) \setminus P_1\}$ s.t. $r_{Pi,B1} > 0 \Rightarrow$

   $$\overline{t_{N_1 B_1}} = \sum_{P_i \in Parents(N)} \overline{\alpha_i} \times r_{P_i,B_1} \times T_N \quad \text{(A.3)}$$

   Let $Parents^*(N) = \{Parents(N) \setminus P_1 \mid r_{Pi,B1} > 0\}$. Then:

   $$\overline{t_{N_1 B_1}} = \overline{\alpha_1} \times r_{P_1,B_1} \times T_N + \sum_{P_i \in Parents^*(N)} \overline{\alpha_i} \times r_{P_i,B_1} \times T_N \quad \text{(A.4)}$$

   Since we add to parent P_1, γ traffic w.r.t. the optimal and we remove from the rest of the parents the ratio of traffic that is not sent compared ti the optimal case (which is smaller than $\frac{\gamma}{k-1}$), we obtain the following relations:

   $$\overline{\alpha_1} \leq \alpha_{1\ opt} + \gamma \text{ and } \sum_{P_i \in Parents^*(N)} \overline{\alpha_i} \ \leq \sum_{P_i \in Parents^*(N)} (\alpha_{i\ opt} - \frac{\gamma}{k-1})$$

Now, if we replace the values of $\overline{\alpha_1}$ in Equation A.4 we obtain an inequality:

$$\overline{t_{N_1B_1}} \leq (\alpha_{1\ opt}+\gamma)\times r_{P_1,B_1}\times T_N + \sum_{P_i \in Parents^*(N)} (\alpha_{i\ opt} - \frac{\gamma}{k-1})\times r_{P_i,B_1}\times T_N \tag{A.5}$$

$$\overline{t_{N_1B_1}} \leq \sum_{P_i \in Parents(N)} \alpha_{i\ opt}\times r_{P_i,B_1}\times T_N + \left(r_{P_1B_1} - \sum_{P_i \in Parents(N)} \frac{1}{k-1} \times r_{P_i,B_1} \right) \times \gamma \times T_N \tag{A.6}$$

$$\overline{t_{N_1B_1}} \leq t_{N_1B_1opt}+\gamma\times T_N \tag{A.7}$$

Since N is the only node in the network generating packets, we obtain c.f. Equation A.1:

$$\sum_{\forall N_j \in Network} \overline{t_{N_jB_i}} \leq (1+\gamma) \sum_{\forall N_j \in Network} t_{N_jB_i\ opt} \tag{A.8}$$

2. P_1 is the only parent of N forwarding traffic to the bottleneck B_1:
   $\forall P_i \in \{Parents(N) \setminus P_1\} : r_{P_i,B_1} = 0 \Rightarrow$

   $$\overline{t_{N_1B_1}} = \overline{\alpha_1} \times r_{P_1,B_1} \times T_N = \overline{\alpha_1} \times T_N \tag{A.9}$$

   Now, if we replace $\overline{\alpha_1}$ by $\alpha_{1\ opt} + \gamma$ in Equation A.9 we obtain:

   $$\overline{t_{N_1B_1}} \leq (\alpha_{1\ opt} + \gamma) \times r_{P_1,B_1} \times T_N \leq t_{N_1B_1opt} + \gamma \times T_N \tag{A.10}$$

   Agin, since N is the only node in the network generating packets, we obtain c.f. Equation A.1:

   $$\sum_{\forall N_j \in Network} \overline{t_{N_jB_i}} \leq (1+\gamma) \sum_{\forall N_j \in Network} t_{N_jB_i\ opt} \tag{A.11}$$

The reasoning holds also for when there is more than just one node generating packets in the network. If each node is at $(1+\gamma)$ from the optimal, globally it will also be at $(1+\gamma)$ from the optimal. For reasons of too many variables (which leads to complicated notations), we decided to not present this part of the proof. Moreover, it is very similar to the case of a single node generating packets.

□