Optimization of Lifetime in Sensor Networks

Nikhil Bansal (Eindhoven University of Technology), David Bourne (Eindhoven University of Technology), Murat Firat (Eindhoven University of Technology), Maurits de Graaf (Thales), Stella Kapodistria (Eindhoven University of Technology), Kundan Kumar (Eindhoven University of Technology), Corine Meerman (Leiden University), Mihaela Mitici (University of Twente), Francesca R. Nardi (Eindhoven University of Technology), Björn de Rijk (Leiden University), Suneel Sarswat (National Institute of Securities Markets, India), Lucia Scardia (Eindhoven University of Technology)

Abstract

We consider the problem proposed by Thales Nederland at the SWI 2012 meeting. Thales Nederland is the Dutch branch of the international Thales Group. The company specializes in designing and producing professional electronics for defence and security applications, such as radar and communication systems. Moreover, Thales Nederland acts as a local point of contact for the complete portfolio of the Thales Group.

During the SWI 2012 meeting, on behalf of Thales Nederland, Dr. Maurits de Graaf posed several questions regarding the maximization of the lifetime of a wireless sensor network. We addressed these questions during the workshop and our most significant results are summarized as follows: We have proven that this lifetime maximization problem, even under the most simple constraints, is NP-complete, so it is not possible to find an algorithm that gives the optimal solution within polynomial time. Furthermore, we have constructed a counterexample to illustrate that the heuristic currently used by Thales can be asymptotically at least $\log n$ times worse than the optimal solution. Moreover, we have developed a new heuristic and have illustrated with several numerical examples that it performs better than the heuristic currently used by Thales. Finally, we have formulated the problem as a linear programming problem and used this to quantify how far the heuristic used by Thales is from being optimal.

KEYWORDS: ad hoc networks, network lifetime, multipoint relay selection, linear programming.

1 Introduction

One of the most important examples of wireless ad hoc networks are wireless sensor networks. Sensor networks can be dynamic, i.e. the topology of the wireless network can change over time (e.g. networks of wearable communication

systems), or static (e.g. sensors in a forest for detecting fires). Sensor networks occur in different real-life settings, for example, in the military, in emergency services or in radar systems (see, e.g., [2] and its references). In all the aforementioned situations it is of vital importance that all nodes in the network can always communicate with each other. We assume that each node in a wireless ad hoc network is equipped with an (omnidirectional) antenna and with a battery of limited capacity. The constraint of limited battery capacity is one of the most important features of sensor networks. Therefore it is essential to develop networking algorithms and protocols that are optimized for energy efficiency.

Usually these wireless ad hoc networks are spread over very large areas. Under this condition one has to employ routing techniques so that the out-of-range nodes can communicate with each other via intermediate nodes. The problem of routing in wireless networks is addressed using different routing protocols. These protocols are distinguished into reactive (finding the route to a destination when it is needed) and proactive (periodically exchanging control messages, which inform on the local or entire topology of the network). The proactive protocols immediately provide the required routes when they are needed, at the cost of bandwidth and battery consumption. Over the last decade the most commonly used proactive protocol is the Optimized Link State Routing (OLSR) (see, e.g., [3]). The OLSR protocol is an optimization strategy for wireless networks that reduces the number of control messages and minimizes flooding of this control traffic by using only a subset of the nodes to send the messages through the system. Each node in the network selects a set of nodes in its neighborhood to retransmit each message it broadcasts. This set of selected neighbor nodes is called a *multipoint relay set* (MPR-set) of this node. The neighbors of any node that are not in its MPR-set receive its messages but do not retransmit them. Each node v selects its MPR-set among its 1-hop neighbors in such a way that messages sent from v are relayed by its MPR-set to all nodes that are two hops away (in terms of radio range). The smaller the MPR-set, the more optimal the traffic control of the network. The MPR-set can change over time, in accordance to the changes of the network topology over time. For the sake of simplicity, in the rest of the manuscript we will consider a stationary network, i.e. we assume that the topology of the network does not change over time.

Many research papers aim at optimizing the selection of MPRs with a specific purpose in mind, e.g., to minimize the number of MPRs used, to keep paths with high Quality of Service, or to maximize the network lifetime (the time until the first node runs out of battery power or the first time at which a communication fails due to battery depletion). In this manuscript we will focus on the maximization of the network lifetime. We adopt the definition of network lifetime as the time until the first node fails due to battery depletion (see, e.g., [9] and its references). Our aim is two-fold: on the one hand to check the non-optimality of the existing algorithms that select relay nodes with the objective of maximizing

the network lifetime, and on the other hand to propose a better heuristic. We will carry out our analysis in two levels distinguishing whether we fall within the specifications of an OLSR network or not.

Consider a group of wireless static nodes randomly distributed in a region, where each node has its own battery supply used mainly for the transmission of messages. We assume that for each transmission (independently of whether this is the initialization of a transmission or a message forwarding) the battery level is reduced by a *fixed* amount. This linear battery model approach is a simplification compared to reality, where the batteries have a recovery time. The node initializing the data transmission is called the source. After the source has sent the message, which will be received first by its neighbors and eventually by all the other nodes in the network, another node becomes the source. We assume that this process continues until all the nodes have acted as a source, which means that a round has been completed. The order of the sources during a round may be prescribed or random.

This problem was brought to our attention during the SWI 2012 meeting on behalf of Thales Nederland, by Dr. Maurits de Graaf. The following directions of investigation were proposed regarding the maximization of the lifetime of a wireless sensor network:

Direction 1: What would be the optimal MPR selection algorithm? With a linear battery decrease model it should be possible to formulate this as a linear program. How much does the optimal solution differ from the known heuristics? Can we define easily a better heuristic than the Maximum Willingness heuristic algorithm used by Thales?

Direction 2: Assume additionally that a node can choose between different power levels. For a higher power level a node will have larger set of neighbors to choose its MPR-set from. Can we formulate the optimization problem and find some good heuristic to solve it (a solution being an assignment of transmit powers and MPR-sets)? What would be the impact on the network lifetime?

Direction 3: What is the effect on the network lifetime problem when using a battery model with a recovery effect?

The paper is organized as follows: In Section 2 we formulate the problem. In Section 2.1 we present the Maximum Willingness heuristic algorithm used by Thales. The rest of the paper is divided into two parts. In the first part (Section 3) we treat the problem of selecting the relays of a network having a general topology in order to maximize the network lifetime. In the second part (Section 4) we treat the same problem within the OLSR framework.

More specifically: In Section 3.1 we show that outside the OLSR framework

the problem under consideration is NP-complete, so it is not possible to find an algorithm that gives the optimal solution within polynomial time. We do this by reducing our problem to the Set-Cover problem, which is a known NP-complete problem. Furthermore, in Section 3.2 we construct a counterexample to illustrate that the Maximum Willingness heuristic does not provide the optimal solution to the problem, which was expected since the problem is NP-complete (unless P=NP). In Section 3.3 we present a new heuristic and in Section 3.4 we illustrate with several numerical examples that our algorithm performs better than the Maximum Willingness heuristic. Furthermore, in Section 3.5 we formulate the problem as a linear programming problem (LP). In Section 4 we work inside the OLSR framework. In Section 4.1 we show that the Maximum Willingness heuristic can be at least $\Omega(\log n)$ times worse than the optimal solution. Furthermore, in Sections 4.2 and 4.3 we present two LPs for optimizing the network lifetime. Finally, in Section 4.4, we conclude our analysis with some numerical results comparing the Maximum Willingness heuristic with the optimal solution. We close with discussions and extensions in Section 5.

2 Formulation of the Network Lifetime Problem

Let $G = (V, E)$ be a connected graph with n nodes, where V denotes the set of nodes and E the set of edges. A network transmission (a message circulated through the entire network) is defined as a time slot, in the sense that time is updated by 1 when a message is sent to all the $n-1$ nodes of the network starting from any given source, i.e. for every time r there is one single node acting as source, say $s_r \in V$, that sends the initial message. This initial message is then forwarded through the entire network. Hence, time $r \in \mathbb{N}_0$ represents the total number of different sources that have sent a message across to the entire network.

A battery level $B_v(r)$ is associated with every node v at time r. We assume that the battery levels are reduced by a fixed amount for every message sent. For simplicity this fixed amount is chosen to be 1.

If we denote by $R(r)$ the set containing the source at time r and all relay nodes that forward the message at time r, then the battery level evolution can be described as follows: for every $v \in R(r)$ we have $B_v(r+1) = B_v(r) - 1$. For all other nodes $B_v(r+1) = B_v(r)$.

Instead of only focusing on minimizing the battery consumption of the individual nodes, it is important, especially in emergency situations, to maximize the network lifetime l, i.e. the first time at which at least one of the batteries is flat:

$$l = \min\{t : B_v(r) = 0 \text{ for at least one } v \in V\}.$$

Note that l depends on the algorithm that is used to circulate the message through the entire network.

The problem can be formulated as follows: Given a connected graph G, a starting distribution of battery levels and a source assignment protocol, choose the relays used to circulate the messages so that the network lifetime is maximized:

$$L = \max\{\min\{r : B_v(r) = 0 \text{ for at least one } v \in V\}\}$$

subject to

$$B_v(r+1) = \begin{cases} B_v(r) - 1, & \text{if } v \in R(r) \\ B_v(r), & \text{otherwise.} \end{cases}$$

2.1 Thales Approach: Maximum Willingness Heuristic

In order to solve the problem stated above we need to have a heuristic that provides us with an optimal selection of relays at each time. Many algorithms address the network lifetime problem in general topology networks. The most relevant for this paper is the Maximum Willingness (MaxWill) algorithm. In this section we closely follow the notation and terminology of [9].

Let $G = (V, E)$ be a connected graph, where V denotes the set of nodes, E the set of edges and n is the number of nodes, i.e. $|V| = n$. Furthermore, let $N^m(v)$, $v \in V$, denote the strict m-hop neighborhood of node v, i.e. the set of nodes for which the shortest path to v has exactly m edges. A subset $M(v) \subset N^1(v)$ is called an MPR-set if $M(v)$ dominates $N^2(v)$, i.e. each node in $N^2(v)$ has a neighbor in $M(v)$. Furthermore, for a given MPR-set $M(v)$, we call each node in this set an MPR node of v. Finally we denote the set of all possible MPR-sets of v by $MPR(v)$.

The MaxWill MPR selection algorithm uses the following structure to calculate an optimal MPR-set $M(v)$ for node v:

1. Start with an empty MPR-set for node v and add nodes of $N^1(v)$ that are the only neighbors of some node in $N^2(v)$.

2. While there are still uncovered nodes in $N^2(v)$, select the nodes from $N^1(v)$ that cover at least one uncovered node and have the highest remaining battery level.

3. Optimize the MPR-set by attempting to remove a node from $M(v)$ and checking if $N^2(v)$ is still dominated. If this is the case, the node is removed from $M(v)$. Nodes are removed in the order lowest remaining battery level first.

Note that the MaxWill MPR-selection algorithm is a localized algorithm, since each node $v \in V$ selects an MPR-set $M(v)$ independently from the other nodes.

Thales uses the MaxWill algorithm and as demonstrated in [9] this algorithm performs better in most cases than other MPR-selection algorithms.

3 Our Approach: Outside the OLSR Framework

We investigate the impact of the selection of the relays on the lifetime of the sensor network. In this section we do not enforce the OLSR constraint that messages should be broadcast in layers (see Section 4 for more details about this constraint). In Section 3.1 we prove that the Network Lifetime problem is NP-complete. Our ideas are similar to those of [7] and [10], who consider related (but not exactly the same) problems. In Section 3.2 we emphasize the weaknesses of the MaxWill heuristic introduced by [9]. We show that for small networks the MaxWill heuristic is outperformed by a path-based heuristic, which is described in Section 3.3. The path-based heuristic attempts to maximize the network lifetime by avoiding using nodes with low battery level, if possible. In Section 3.4 we give simulation results for the path-based algorithm and the MaxWill algorithm. In Section 3.5 we formulate the relay selection problem as a LP.

3.1 Proof of NP-completeness

We will show that the problem of maximizing the network lifetime is not only NP-complete, but that it is also hard to approximate within a factor of $\Omega(\log n)$, where one writes $f(n) = \Omega(g(n))$ as $n \to \infty$ for two functions f and g if and only if there exists a positive real number M and positive integer n_0, such that $|f(n)| \geq M|g(n)|$, for all $n \geq n_0$.

We give a reduction to the so-called maximum domatic partition problem (see [5]), defined as follows. Given a graph $G = (V, E)$, a *dominating set* of G is a subset $S \subset V$ such that each node $v \in V$ is either in S or has a neighbor in S. The domatic number of G is the maximum number of dominating sets into which the vertices V can be partitioned. Let k be the maximum number of disjoint dominating sets contained in the graph G. [5] showed that for every $\epsilon > 0$, no polynomial times algorithm can approximate the domatic number problem within a $(1 - \epsilon) \log n$ factor, unless NP has a slightly superpolynomial-time algorithm, i.e. unless $NP \subseteq DTIME(n^{\log \log n})$. Hence, it is impossible to approximate the size of the maximum domatic partition to a factor better than $\Omega(\log n)$. In particular, for any n, there are instances such that no efficient algorithm can distinguish whether the domatic number is at least k or at most $O(k/\log n)$. In fact, [5] show the following stronger result (which is the one we need): No efficient algorithm can distinguish whether the domatic number is at

least k, or whether there is a dominating set of size $O((n \log n)/k)$. Note that this is a much stronger hardness result, as the domatic number must be at most $O(k/\log n)$ if the minimum dominating set has size $\Omega((n \log n)/k)$.

We will use the above problem to show the $\Omega(\log n)$ hardness of the network lifetime problem. Let G be a hard instance of the domatic partition problem, and n denote the number of vertices in G. We will construct an artificial graph $\hat{G}$ with $2n$ vertices, which is equivalent to G. The artificial graph is constructed as follows: For each vertex $i \in G$, we define two vertices $(i,1)$ and $(i,2)$. We think of the second index as defining a layer. So there are two layers. For every $1 \le i < j \le n$, we connect the vertices $(i,1)$ and $(j,1)$ by an edge (i.e. all vertices in the first layer form a clique). For each vertex $(i,2)$, we connect it to the vertex $(j,1)$ whenever (i,j) is an edge in G or if $j = i$. Note that a vertex in the second layer is not connected to any other vertex in the second layer. Hence, the graph $\hat{G}$ has k disjoint dominating sets. Let the initial battery levels of $\hat{G}$ be $(2n/k+3)B$ for layer 1 vertices and B for layer 2 vertices. When each vertex has had its turn being a source once, we call that a round.

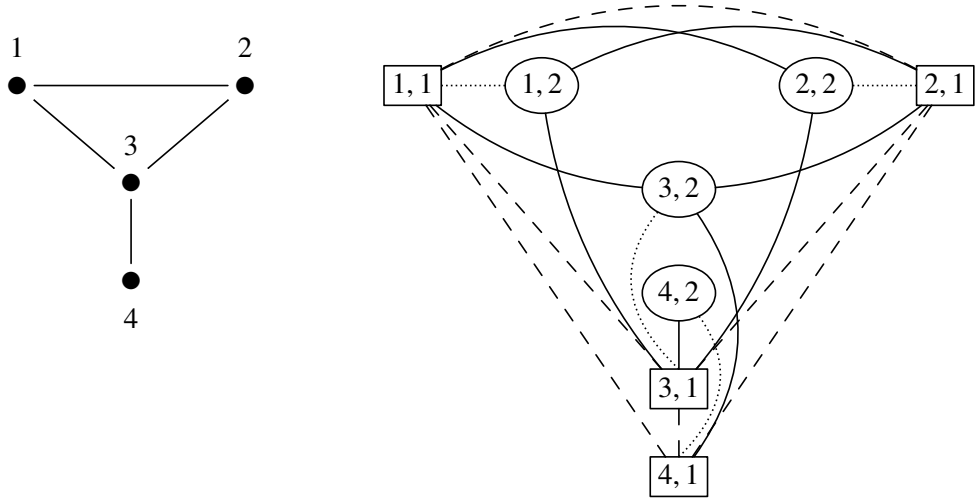

Figure 1: Example of the construction of $\hat{G}$. In this case $k = 2$ and $C_1 = \{3\}$, $C_2 = \{2,4\}$ are examples of disjoint dominating sets.

Claim 3.1. *If the domatic number is k, the source can transmit for B rounds (i.e. optimum lifetime is at least B).*

Proof. Let $C_1, \ldots, C_k$ denote the k disjoint dominating sets of $\hat{G}$. When the source is a layer 1 vertex, say $(i,1)$, it first directly transmits its message to all the layer 1 vertices. This can be done as layer 1 vertices form a clique. Then the nodes

$(j,1)$ such that $j \in C_{i'}$ where $i' = i \bmod k$ transmit the message to all the nodes in layer 2. Note that this is possible as each $C_{i'}$ is a dominating set. Next, the strategy when the source is a layer 2 vertex, say $(i,2)$, is to first transmit its message to $(i,1)$ and then the above strategy is followed.

Now note that during each round, each vertex $(i,2)$ in layer 2 transmits exactly once when it is the source. Each vertex $(i,1)$ in layer 1 transmits exactly once when it is the source, once when it is the relay for the corresponding node $(k,2)$ from layer 2, and at most $2n/k+1$ times as a relay in some dominating set. The latter is because the dominating sets are disjoint (they form a partition), and each collection $C_{i'}$ is used $\lceil 2n/k \rceil$ times. By the choice of our initial battery levels, it is easily checked that this strategy lasts for at least B rounds. □

We now show the other direction, i.e. if the lifetime is high, then there is a small dominating set. In particular,

Claim 3.2. *If the network lifetime is cB, for some constant $c \in [1, \lceil 2n/k \rceil]$, then the domatic number is at least $2n/(ck)$.*

Proof. Observe that a layer 2 vertex $(i,2)$ can receive a message only via some layer 1 node. Thus, no matter which node is the source, to transmit its message to all nodes, the relays on layer 1 must form a dominating set. So in each round, nodes from at least $2n$ dominating sets must transmit. Let s be the size of the minimum dominating set. If the network lifetime is at least cB, this means that the total battery power used by the layer 1 nodes is at least $2n \cdot s \cdot cB$. On the other hand, the total initial battery power of these nodes is $(2n/k+3)B \cdot n$. This implies that $2nscB \leq n(2n/k+3)B \leq 4n^2/kB$, and hence $s \leq 2n/(ck)$. □

Given the hardness result of [5] stated above (the stronger form) and the equivalence of the original graph G and the artificial graph $\hat{G}$, the two claims imply the following result.

Corollary 3.3. *The network lifetime cannot be approximated to within $O(\log n)$.*

3.2 Weakness of Maximum Willingness

Consider the example in Figure 2 of five nodes in a cyclic order, in which the batteries of nodes $1, 2, 4$ and 5 have initial capacity of 100, while the battery capacity of node 3 is only 10. Furthermore, consider the following deterministic transmission order protocol: node 1 is the first to transmit, then node 2 transmits and they continue in ascending order until all the nodes transmit a message. Then node 1 is again the first to transmit and they continue in this way until the first node fails due to battery depletion.

According to the MaxWill algorithm if node 1 is the source then nodes 2 and 5 are the selected MPRs, if node x, $x \in \{2,3,4\}$, is the source then nodes $x-1$

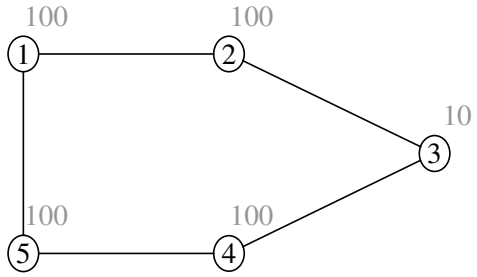

Figure 2: Example of the weakness of the MaxWill algorithm.

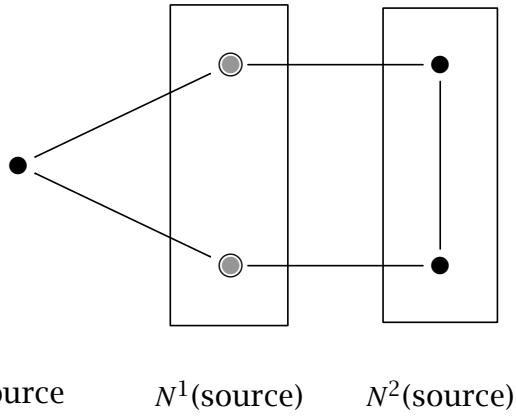

Figure 3: Selected MPRs according to MaxWill are indicated by a circled gray bullet.

and $x + 1$ are the selected MPRs, and finally if node 5 is the source then nodes 1 and 4 are the selected MPRs. Hence there will be in total 17 messages sent before the battery of node 3 is used up. However, one can easily verify that the best strategy to prolong the system's lifetime is to never use node 3, unless this node is the source. In this way there will be a total of 52 messages sent. The weakness of the MaxWill algorithm (and the OLSR framework in general) is that it only focuses on the 2-hop neighborhood, forcing in this example both nodes in the 1-hop neighborhood to be MPRs. A more efficient heuristic will be presented in the next section.

3.3 Path-Based Heuristic

We propose an algorithm that aims to prolong the network lifetime by *not* using the nodes with low battery levels as relays. The relay set determined by the heuristic chooses nodes with battery levels as high as possible.

Path selection. Given a source node $s \in V$, let the node $v' = \text{Argmin}_{v \in V \setminus \{s\}} \{B_v\}$. Our goal is to assign a neighbor node of v' as relay so that v' receives the transmission and hopefully is not assigned as relay. This is achieved by finding a path from s to v' (we know that such a path exists, since G is connected). In graph G we may have a number of such paths, so we choose one that uses relay nodes with the highest battery levels possible. The path selection continues by choosing a node with the second lowest battery level and by finding a path from s to it, and so on.

Relay assignments. Let $P_{s,v'} = \{s, \ldots, v'\}$ be the path found in the way explained above. We assign all nodes in $P_{s,v'} \setminus \{s, v'\}$ as relay nodes. Note that a low-battery level node may be assigned as relay if it is on a path from the source

Table 1: Some notation for the path-based heuristic

s	Source node, $s \in V$
B_v	Battery level of node v , $v \in V, B_v \in \mathbb{N}_0$
R_s	The set of relay nodes for source s, $R_s \subset V$
$G[V']$	Subgraph of G that is induced by $V', V' \subseteq V$
$N(R_s)$	Adjacent nodes to the set R_s of relay nodes
$SP(G, v_1, v_2)$	Shortest path from v_1 to v_2 in graph G

Note: We find all paths by assuming that all edges have same lengths (weights).

Path-Based Heuristic.

Input: See Table 1

$R_s \leftarrow \{s\}$;
while $R_s \cup N(R_s) \neq V$ **do**
  $v' \leftarrow \text{Argmin}_{v \in V \setminus (R_s \cup N(R_s))} \{B_v\}$;
  $V' \leftarrow \{s, v'\}$;
  $P_{s,v'} \leftarrow SP(G[V'], s, v')$;
  while $P_{s,v'} := null$ **do**
    $V' \leftarrow V' \cup \text{Argmax}_{v \in V \setminus V'} \{B_v\}$;
    $P_{s,v'} \leftarrow SP(G[V'], s, v')$;
  end
  $R_s \leftarrow R_s \cup P_{s,v'} \setminus \{s, v'\}$;
end

Output: The set R_s of relay nodes.

to another node. For example, cut-vertices must be used as relays no matter what their battery levels are. By a cut-vertex we mean one whose deletion makes the graph disconnected.

Path selection and relay assignment continues until each node in $V \setminus \{s\}$ is either a relay or is adjacent to a relay node. Table 1 introduces the necessary notation to describe our heuristic and the path-based heuristic is given afterwards.

In the path-based heuristic, in Step 1, we initialize the relay-set with the source s. Then we check, in Step 2, if the current relay set transmits to all nodes in G. If this is not achieved, we select, in Step 3, the lowest-battery level node, say v', among the ones not receiving any transmission. Next we try to find a (shortest) path from s to v' in the subgraph $G[V']$, where V' is defined in Step 4. If such a path does not exist, in Step 7 we update V' by adding the highest-battery level node in $V \setminus V'$ until the desired path is found. In Step 10 we update the set R_s of relay nodes by adding the intermediate nodes in the path $P_{s,v'}$. Finally, the algorithm terminates when all nodes receive transmissions from relay nodes.

Running time. Note that the while loop runs over all the nodes in the graph. So the path-based algorithm runs in $O(n^2 SP)$ time where SP denotes the time to find the shortest path in a graph with unit length (weight) edges. If the nodes in the inner while loop (Steps 6-9) are determined with a binary search, the running time of the algorithm can be improved to $O(n \log n SP)$.

3.4 Numerical Results: Comparing MaxWill with the Path-Based Heuristic

We denote by Algorithm 1 the MaxWill algorithm and by Algorithm 2 the path-based heuristic introduced above.

Let t_i be the number of messages sent using Algorithm i, for $i = 1, 2$, and let R denote the ratio t_2/t_1. The battery capacity B_v of each node v is uniformly distributed over $[\mu_B - \sigma_B, \mu_B + \sigma_B] \cap \mathbb{Z}$. For our simulation purposes we generated several graphs based on the binomial model of the Erdös-Rényi random graph. This model generates an edge between a pair of nodes with equal probability, say p, independently of the other edges. Therefore the lower p is, the sparser G is. If G turns out to be not connected after the construction, another graph is generated. Each time a message is sent, the source will be uniformly selected from the set of nodes V. For parameter values $n = 30, \mu_B = 15, \sigma_B = 10$ and $p = 0.1$, the results of 10.000 simulations in MATLAB are depicted in Figure 4.

Note that in all the simulations we obtained $R \geq 1$. This strongly suggests that, for the type of graph topologies that we have generated, $t_2 \geq t_1$ for all possible battery capacities $(B_v)_{v \in V}$ assigned to the nodes. Furthermore, we have

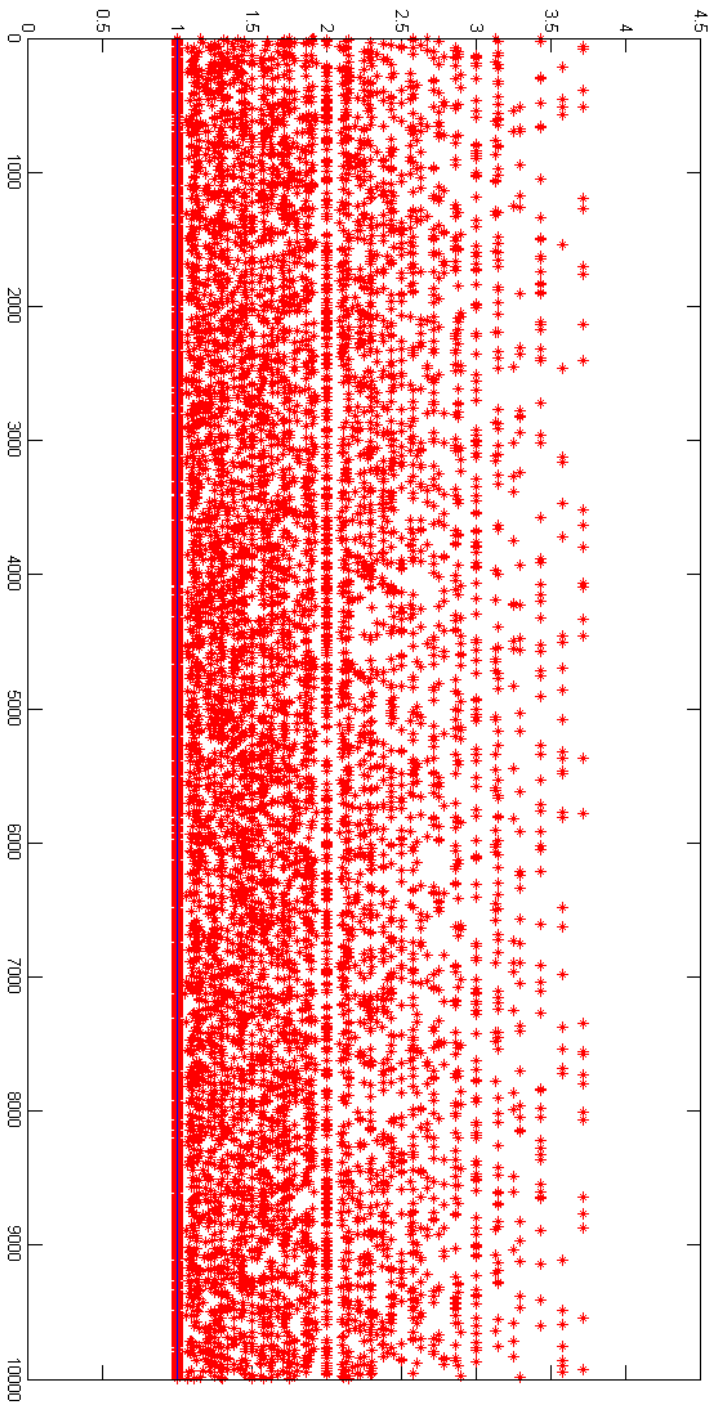

Figure 4: Comparison of the MaxWill and Path-Based algorithms for $n = 30, B_v \sim U[5, 25]$ and $p = 0.1$.

calculated the mean and standard deviation of R to be 1.642 and 0.639, respectively. So all together this suggests that Algorithm 2 performs overall better than Algorithm 1 and does this on average with ratio 1.642. Since at Thales they are dealing with various types of graphs we are interested in the effect of each of the variables n, μ_B, σ_B and p on the ratio R. The results are shown in Figures 5–8.

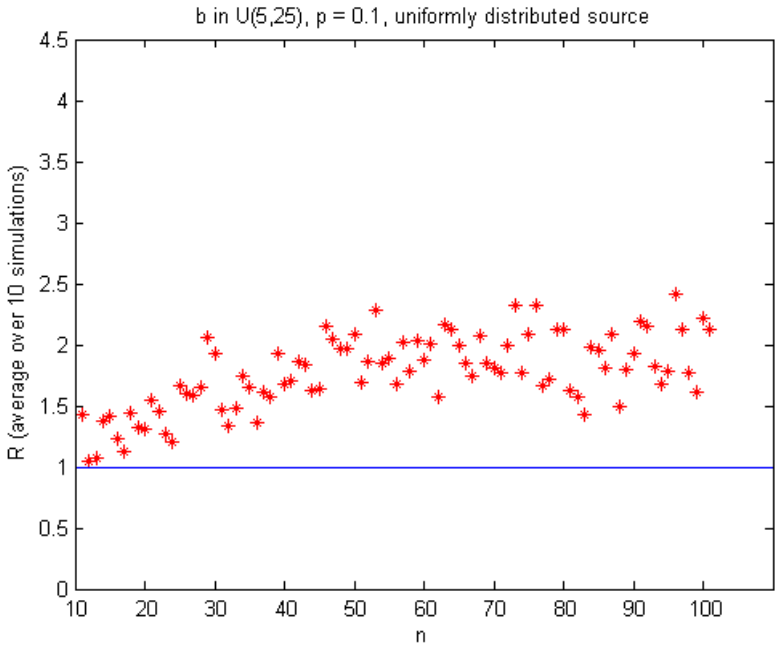

Figure 5: Effect of changing the number of nodes n on the ratio R.

In Figure 5 we see that if n is very small the value of R tends to be closer to one. This is due to the fact that for a randomly generated small graph the local approach of Algorithm 1 (looking at the 1-hop neighborhood) is in fact almost global.

In Figures 6 and 7 we see that R is particularly high if the initial battery values of the nodes differ a lot. Indeed, Figure 6 shows that when μ_B increases the ratio R decreases and Figure 7 shows that when σ_B increases the ratio R increases. So it is likely that Algorithm 2 performs significantly better if the variance of the initial battery values of the nodes is high. This might be due to the fact that Algorithm 2 tends to bring the battery values closer together with time, whereas, in an unlucky case, Algorithm 1 might use nodes with low battery capacities over and over again due to the local nature of the algorithm.

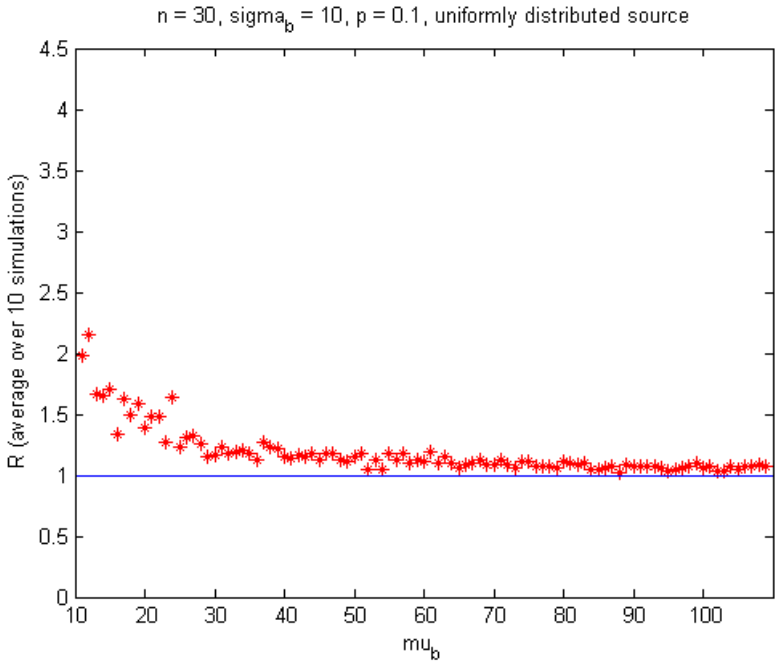

Figure 6: Effect of changing μ_B on the ratio R.

Finally, in Figure 8, we see that R is almost equal to 1 for both highly connected graphs and highly sparse graphs. This is because Algorithm 1 and Algorithm 2 perform optimally for both fully connected graphs and tree structured graphs (the most sparse graphs). Algorithm 2 does perform especially well if the graph is 'intermediately' sparse. This is because Algorithm 1 performs suboptimally when there are larger cycles in a non-fully connected graph (see Section 3.2).

In summary, it is likely that Algorithm 2 performs as well or better than Algorithm 1 for all graph topologies and all possible battery capacities assigned to the nodes. Furthermore, Algorithm 2 performs especially well in the case when the variance of the initial battery capacities is high and the graph is 'intermediately' sparse.

3.5 LP Formulation and Solution Approach

We can also design a linear programming formulation of the problem. We simply sketch the idea here since it is very similar to the one considered later in Sec-

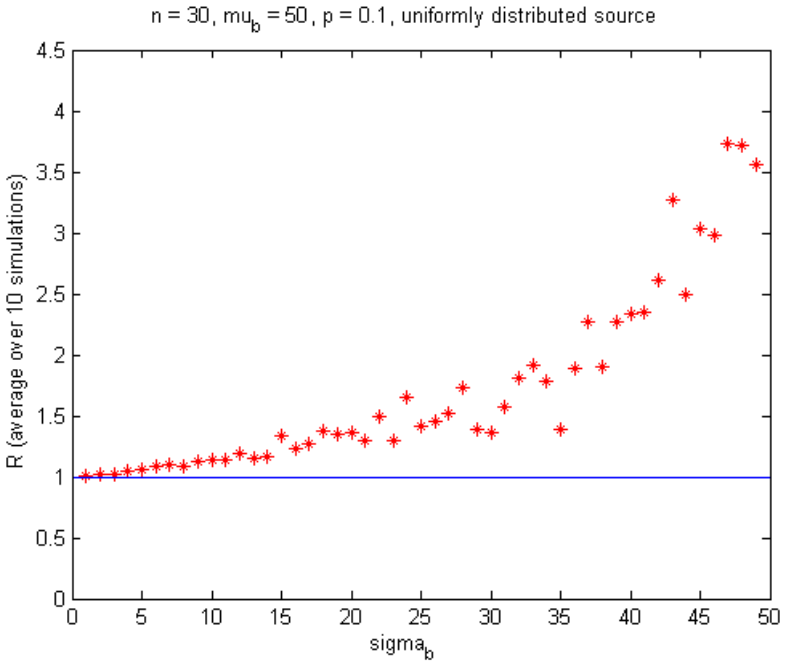

Figure 7: Effect of changing σ_B on the ratio R.

tion 4.2. Given a source s, we define variables $x_{s,S}$ for every subset S of nodes corresponding to a valid set of relays when s is chosen as the source (we always assume that S contains the source s). Given the battery levels B_v, one can thus write the following integer programming formulation:

$$\max r$$

$$\text{s.t.} \sum_s \sum_{S: v \in S} x_{s,S} \leq B_v \qquad \forall v$$

$$\sum_S x_{s,S} \geq r \qquad \forall s$$

$$x_{s,S} \in \{0, 1, 2, \ldots\} \qquad \forall s, S \text{ such that } S \text{ is a valid set of relays for source } s.$$

Let us see why this is a valid formulation. Here r denotes the number of rounds (i.e. where each node takes a turn being a source) that we wish to maximize. The first constraint says that the number of times vertex v is used as a relay is at most the initial battery level B_v. The second constraint says that for each source s we must determine at least r sets S that are valid sets of relays. Clearly, given any valid integer solution to this program, we can perform r rounds by choosing the set S as relays for $x_{s,S}$ rounds when s is the source.

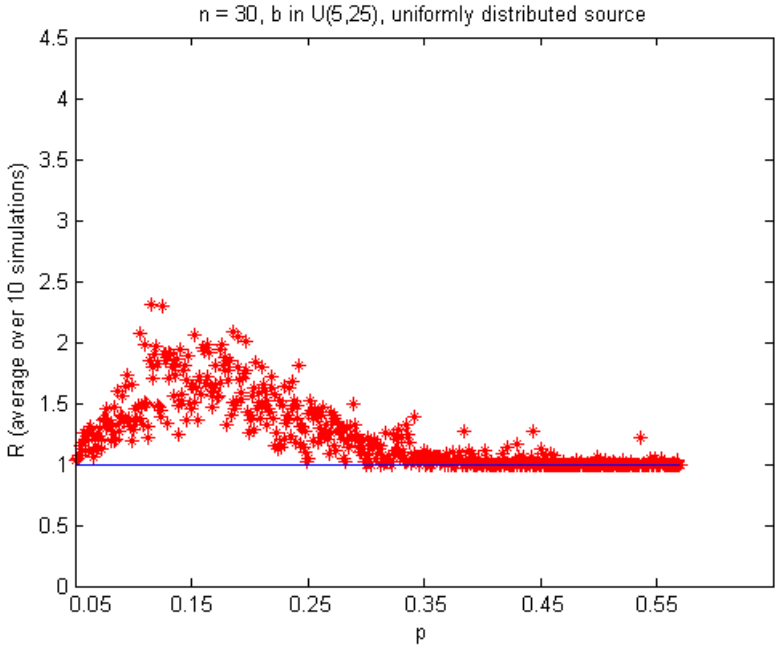

Figure 8: Effect of changing p on the ratio R.

As is standard in approximation algorithms (see [8, 11]), we relax the integrality constraints on the variables $x_{s,S}$ above, and consider the linear programming relaxation (LP) where we only require that $x_{s,S} \geq 0$. This is done because linear programs can be solved to optimality in polynomial time in the number of variables and constraints (albeit at the expense of allowing the variables $x_{s,S}$ to take non-integral values).

However it is not immediately clear how to use this in our setting since the LP above has exponentially many variables (corresponding to the exponentially many possible sets S). To get around this, we note that while this LP has exponentially many variables, it has only polynomially many constraints. So we can consider the dual of this LP (which has polynomially many variables, and exponentially many constraints). Even though there are exponentially many constraints, this is not an major problem, as we can solve it using the Ellipsoid method (by adding constraints on the fly as needed), provided there is an approximate separation oracle for the dual separation problem. Recall that a separation oracle is a subroutine that, given a candidate solution x as input, either outputs a violating constraint, or certifies that x satisfies all the constraints and is a valid

solution. By the standard equivalence between optimization and separation (see for example [6] for more details) an α approximation algorithm to solve the dual separation problem implies an α approximate solution for the primal LP. It can be easily checked that in our setting, the dual separation problem is a weighted set cover problem and hence has an $O(\log n)$ approximation. Given such a solution, one can now apply randomized rounding as explained in Section 4.2. Again, similar ideas are discussed in section 4 and hence we omit the details here.

4 Our Approach: Within the OLSR Framework

As we have already mentioned in the introduction we are interested in selecting the MPRs in order to maximize the network lifetime while satisfying the OLSR constraint. The OLSR protocol relies on the selection of MPRs and calculates its routes to all known destinations through these nodes, i.e. MPR nodes are selected as intermediate nodes in a path. To implement this scheme each node in the network periodically broadcasts the information about its 1-hop neighbors. Upon receipt of this message each node calculates and updates its routes to each known destination. For more details on the OLSR protocol the interested reader is referred to the paper of [3].

Within the OLSR framework each node of the network selects its own MPR-set. This set is a subset of its 1-hop neighbors that covers all its 2-hop neighbors, i.e. the union of the neighbor sets of all MPR nodes contains the entire 2-hop neighborhood. For a given source s this results in a layered network: The first layer is the set of nodes that can be directly reached from the source s, i.e. there exists a directed edge (s, j) if s can directly transmit to node j. We denote the first layer by $L_s(1)$ and we set $L_s(1) = N^1(s)$, where $N^1(s)$ is the set of 1-hop neighbors of s. We denote $L_s(0) = s$. Then the k-th layer is denoted by $L_s(k)$ and is constructed recursively as follows

$$L_s(k) = \bigcup_{v \in L_s(k-1)} \left\{ N^1(v) \cap \left\{ V \setminus \cup_{i=0}^{k-1} L_s(i) \right\} \right\},$$

until all nodes of the graph are classified into layers. Among the nodes in layer $L_s(k)$, MPRs are selected to forward the message to the next layer, $L_s(k+1)$, so that all the nodes in the $(k+1)$-th layer receive the message.

Keeping in mind that each broadcast depletes the battery level of the nodes that are transmitting the message, we are interested in selecting the MPRs in the layered network so that the network lifetime is maximized. In Viennot [10] it was shown that the problem of finding an optimal set of MPRs is NP-complete. The authors showed that the Dominating Set Problem, which is known to be NP-complete, can be reduced to the Multi-point Relay Problem. In Coenen et al. [4], the authors consider the selection of master nodes that relay to their neighbors.

In Verbree et al. [9] the authors analyze the impact of the network topology on the selection of MPRs.

The rest of the paper is organized as follows: In Section 4.1 we give a network instance for which the MaxWill heuristic performs at least $\Omega(\log n)$ times worse than the optimal solution. In Section 4.2 we formulate the relay problem as a linear programming (LP) problem and we use randomized rounding to convert the LP solution into an integral solution. In Section 4.3 we present a second LP formulation and we implement it in Section 4.4 to compare the Maximum Willingness heuristic with the optimal solution.

4.1 Example of the Non-Optimality of Maximum Willingness

We consider here the simpler case of a single source. We construct an example of a graph G for which the MaxWill heuristic is at least $\Omega(\log n)$ times worse than the optimal solution.

Consider the following layer-structured graph of $n = 2k + 2^k - 1$ nodes, with $2k$ nodes in the first layer, and $2^k - 2$ nodes in the second layer. For the rest of the analysis k will be treated as a parameter. We view each node in layer 1 as the pair (i, b) where $i \in \{1, \ldots, k\}$ and $b \in \{0, 1\}$. The nodes in layer 2 are labeled $1, \ldots, 2^k - 2$. It is convenient to consider the binary representation of these labels. The edges between the layers are defined as follows. A node (i, b) is adjacent to all nodes in layer 2 whose i-th bit is b.

Suppose the initial battery of each node is $B_v = B$, for all $v \in V$. We claim that

Lemma 4.1. *The optimum solution can transmit kB messages.*

Proof. Note that for any i, if we pick $(i, 0)$ and $(i, 1)$ to be MPRs, then all the nodes in layer 2 can be covered. This is because each node has either a 0 or 1 in the i-th position. Suppose we pick the above solution with $i = 1, \ldots, k$. If we call this one round, then we can repeat this for B rounds until the batteries are exhausted. □

We now show that the MaxWill heuristic can perform badly. Since the battery levels are the same, it is possible that the nodes $(1, 0), \ldots, (k, 0)$ are chosen. Note that this collection covers all the nodes in layer 2 (since each label has a 0 in at least one position). Moreover, none of these sets are redundant. Indeed if some $(i, 0)$ were dropped, then the point which has 0 in the i-th position and 1 everywhere else would not be covered. After picking these nodes as MPRs, the battery level of these nodes is down to $B - 1$, so in the next step the algorithm will pick the nodes $(1, 1), \ldots, (k, 1)$ as MPRs. At this point, all the battery levels are $B - 1$, and the process repeats again. In this way the batteries are exhausted in $2B$ steps. This gives a gap of $k/2$. Hence in terms of the number of nodes in

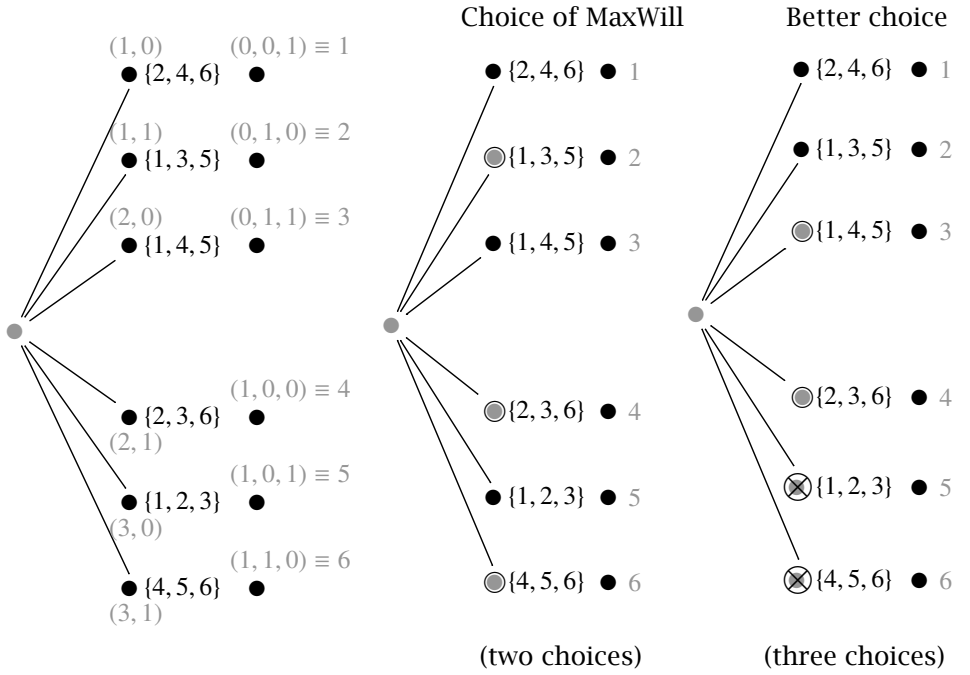

Figure 9: MaxWill heuristic can be at least $\Omega(\log n)$ times worse than the optimal solution.

the network, this gives a gap of $\Omega(\log n)$. See Figure 9.

Remark: It is possible that there are instances where the performance of the MaxWill algorithm is even worse.

4.2 First LP Formulation

In this section we present a linear program that might be useful when the number of nodes in each layer is not too large. We first give some notation. Let $L_s(i)$ denote the i-th layer in the graph when the source is s, $s \in V$. Call a subset $S \subset L_s(i)$ of nodes a valid MPR-set if it can cover all elements of $L_s(i+1)$. Let C_s^i denote the set of all valid MPR-sets in layer i when s is the source. Let $\ell_s = \max_{v \in V} d_G(s, v)$, where d_G is the graph distance in G. Furthermore, ℓ_S can be seen as the number of layers in the graph when s is the source.

Consider the following formulation. For each valid MPR-set S in C_s^i, we have

a variable $x_S \in \mathbb{R}^+$ which indicates how many times this set is chosen,

$$\begin{aligned} &\max r \\ \text{subject to } &\sum_{S \in C_s^i} x_S \geq r, \qquad \forall i = 1, \ldots, \ell_s - 1, \\ &\sum_{S : v \in S} x_S \leq B_v, \qquad \forall v \in V. \end{aligned}$$

Note that in an exact integer programming formulation of the problem we require that $x_S \in \{0, 1, \ldots\}$. Here, however, we have relaxed these integrality constraints to obtain a linear program which is tractable. The first constraint above says that for each source and each layer, we must choose at least one set per round (and hence the total number of sets chosen from a layer should be at least r). The second constraint says that the total number of times a node v is ever used as a MPR is at most B_v.

If the size of each layer is at most k, then this LP has a most $2^k \cdot n \cdot n$ variables (as there can be at most 2^k subsets per layer, at most n sources and at most n layers per graph). So, this LP might be solvable if k is not too large, say around 10 (of course, one needs to further explore this experimentally to determine up to what values of k this might scale to).

4.2.1 Rounding

The above LP gives a fractional solution for the number of times that the sets are chosen (because $x_S \in \mathbb{R}^+$). However, we need an integral solution $x_S \in \mathbb{N}_0$. To do this, we can use the standard randomized rounding approach (see for example [8, 11]). Here, given any accuracy parameter $\epsilon > 0$, we can convert the fractional LP solution into an integral one with $(1 - \epsilon)r$ rounds, provided every initial battery capacity B_v and the number of rounds r are $\Omega(\log n / \epsilon^2)$, where n denotes the number of nodes in the graph. That the battery and the number of rounds are modestly large, in the sense above, is perhaps a reasonable assumption in practice.

Before we state and prove the result formally, we recall some standard probabilistic tail bounds.

Lemma 4.2 (Chernoff Bounds, Theorems A.1.12 and A.1.13 in [1])**.** *Let X_i, $i = 1, \ldots, n$ be independent* 0-1 *random variables with mean $E[X_i] = \mu_i$. Let $X = \sum_i X_i$ and $\mu = E[X]$. Then for any $\epsilon > 0$, it holds that*

$$Pr[X \leq (1 - \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 2},$$

$$Pr[X \geq (1 + \epsilon)\mu] \leq \left[\frac{e^\epsilon}{(1 + \epsilon)^{1+\epsilon}} \right]^\mu.$$

Theorem 4.3. *Given any $0 < \epsilon \leq 1/2$, the LP solution can be rounded to a feasible integral solution where the number of rounds is at least $(1-2\epsilon)r$, provided that $r \geq 10(\log n)/\epsilon^2$ and $B_v \geq 10(\log n)/\epsilon^2$ for each node v, where n is the number of nodes.*

Proof. For each set S such that $x_S > 0$ in the LP solution, let $y_S = (1-\epsilon)x_S$, and let $f_S = y_S - \lfloor y_S \rfloor$ be the fractional part of y_S (note that $f_S \in [0,1]$). Let z_S denote the 0-1 random variable that is 1 with probability f_S and 0 otherwise. To round the LP solution, for each set S, we independently (of other sets) choose $\lfloor y_S \rfloor + z_S$ copies of S. In other words, we choose $\lceil y_S \rceil$ copies of S with probability z_S and $\lfloor y_S \rfloor$ copies otherwise.

Let us bound the probability that the battery capacity is exceeded for any node v (this is precisely the event $\sum_{S:v\in S}(\lfloor y_S \rfloor + z_S) > B_v$, for some v). To this end, consider a particular node v,

$$\begin{aligned}
\Pr\left[\sum_{S:v\in S}(\lfloor y_S \rfloor + z_S) > B_v\right] &= \Pr\left[\sum_{S:v\in S} z_S > B_v - \left(\sum_{S:v\in S}\lfloor y_S \rfloor\right)\right] \\
&= \Pr\left[\sum_{S:v\in S} z_S > B_v - \sum_{S:v\in S}(y_S - f_S)\right] \\
&\leq \Pr\left[\sum_{S:v\in S} z_S > \epsilon B_v + \sum_{S:v\in S} f_S\right] \\
&= \Pr\left[\sum_{S:v\in S} z_S > (1+\epsilon')\sum_{S:v\in S} f_S\right] \text{ with } \epsilon' = B_v/\left(\sum_{S:v\in S} f_S\right) \\
&\leq \left[\frac{e^{\epsilon'}}{(1+\epsilon')^{1+\epsilon'}}\right]^{\mu} \text{ with } \mu = \sum_{S:v\in S} f_S.
\end{aligned}$$

Above, the first inequality follows since the LP satisfies, $\sum_{S:v\in S} x_S \leq B_v$ and hence $B_v - \sum_{S:v\in S} y_S \geq \epsilon B_v$.

Consider two cases: $\epsilon' < 2e-1$ and $\epsilon' \geq 2e-1$. If $\epsilon' \geq 2e-1$, then we can bound the expression above by

$$(1/2)^{\epsilon'\mu} = (1/2)^{B_v} \leq 1/n^{10}.$$

If $\epsilon' < 2e-1$, then by using Taylor expansions for $e^{1+\epsilon'}$, we can bound this by $e^{-\epsilon'^2/4\mu} \leq 1/n^2$. Thus, we see that the probability of this event is at most $1/n^2$. Taking the union over all the n nodes, the probability of the battery running out for any node is at most $1/n$.

Similarly, since the LP has r rounds, the expected number of rounds in the rounded solution is at least $(1-\epsilon)r$ since the original LP constraint is $\sum_{S\in C_s^i} x_{s,S} \geq$

r, and hence $\sum_{S \in C_s^i} y_{s,S} \geq r$ (and the expected number of copies of S that we choose in our rounded solution is precisely $\lfloor y_S \rfloor + E[z_S] = \lfloor y_S \rfloor + f_S = y_S$). Again, direct application of Lemma 4.2 implies that the probability that there are fewer than $(1-2\epsilon)r$ rounds is at most $1/n^2$. So the overall probability of any of the bad events happening is at most $1/n + 2/n^2 \leq 2/n$. This algorithm can be derandomized using standard techniques, see for example [1]. □

4.3 Second LP Formulation

Here we consider an alternative formulation of the linear program. The binary variable $x_{r,s,v} \in \{0,1\}$ is defined for every round $r \in \{1,\ldots,\mathcal{R}\}$, for every source $s \in V$ and for every node $v \in V$. We take $V = \{1,2,\ldots,n\}$. We set

$$x_{r,s,v} = \begin{cases} 1 & \text{when node } v \text{ is broadcasting in round } r \text{ and node } s \text{ is the source,} \\ 0 & \text{otherwise.} \end{cases}$$

For a node v we denote by B_v its battery level and p_v the battery depletion after the broadcasting of a message. This means that each time it transmits a message (as a MPR or source) its battery level reduces by p_v. So if a node originally has battery level B_v it can broadcast at most $\lfloor B_v/p_v \rfloor$ messages. Note that up until now we have always assumed that $p_v = 1$.

For a source s and for two nodes u and v we define $P_{uv}^s \in \{0,1\}$ as follows: We set $P_{uv}^s = 1$ if the node u is a *predecessor* of the node v when s is used as source, namely if the node u is in the layer before that of v, and if the nodes u and v are connected. Clearly $P_{us}^s = 0$, since the source has no predecessors. The matrix $\boldsymbol{P}^{(s)} = (P_{uv}^s)_{u,v}$, for a fixed source s, will be referred to as the *s-predecessor matrix*.

To better understand the meaning of the predecessor matrix, we compute it for a simple, concrete example, namely for the graph in Figure 10.

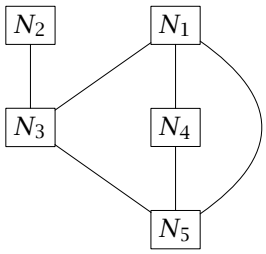

Figure 10: The Network.

Before we can compute P^s_{uv} we need to sort the graph into layers according to which node is the source. See Figures 11-15. Then we see, e.g., that matrix $\boldsymbol{P}^{(1)}$ (which corresponds to $s = 1$, Figure 11) is

$$\boldsymbol{P}^{(1)} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \tag{1}$$

indeed since N_1 is the predecessor of N_3, N_4 and N_5 the corresponding entries P^1_{13}, P^1_{14} and P^1_{15} are equal to one; also $P^1_{32} = 1$ since N_3 is a predecessor of N_2.

We formulate the linear program as a feasibility problem with linear constraints. More precisely, we first fix a number of rounds $\mathcal{R}$ and we ask whether our network can transmit messages for $\mathcal{R}$ *whole* rounds. Then we optimize over $\mathcal{R}$ to find the maximum number of feasible rounds. When we implement the algorithm in the following section we take as a first guess of the number of feasible rounds the result given by the MaxWill algorithm, which is clearly a lower bound. Then we keep incrementing $\mathcal{R}$ by 1 until the problem becomes infeasible.

More precisely, for a fixed $\mathcal{R}$, we consider the following feasibility problem with linear constraints:

(IP) Find $x_{r,s,v} \in \{0,1\}$, where $r \in \{1,\dots,\mathcal{R}\}$, $s, v \in \{1,\dots,n\}$, such that

(1) $x_{r,s,s} = 1 \quad \forall\, s \in \{1,\dots,n\},\ \forall\, r \in \{1,\dots,\mathcal{R}\};$

(2) $\sum_{u=1}^{n} x_{r,s,u} P^s_{uv} \geq 1 \quad \forall\, s \in \{1,\dots,n\},\ \forall\, v \in \{1,\dots,n\},\ \forall\, r \in \{1,\dots,\mathcal{R}\};$

(3) $\sum_{r=1}^{\mathcal{R}} \sum_{s=1}^{n} x_{r,s,v} \leq \frac{B_v}{p_v} \quad \forall\, v \in \{1,\dots,n\}.$

Constraint (1) is the trivial constraint that node s transmits a message when it is the source; constraint (2) ensures that every message is received by every node; and constraint (3) ensures that the battery level of each node remains nonnegative. Note that since the inequality in constraint (3) is not strict, it is possible for the battery level of one or more nodes to be zero or to go down to zero (but not below zero) in a feasible round, provided that every node receives the message.

Problem (IP) is a binary integer program and so it is NP-complete. Note, however, that the number of variables equals $\mathcal{R}n^2$. In particular it grows polynomially in n, in contrast with the linear program presented in the Section 4.2, where the number of variables grows exponentially in n. Note that the number of constraints also grows polynomially in n and $\mathcal{R}$.

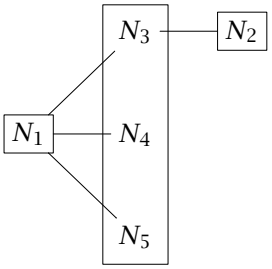

Figure 11: Source N_1.

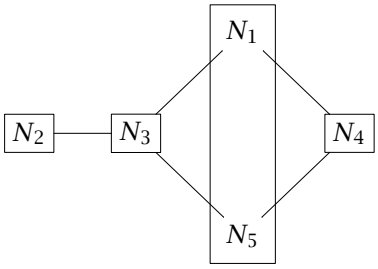

Figure 12: Source N_2.

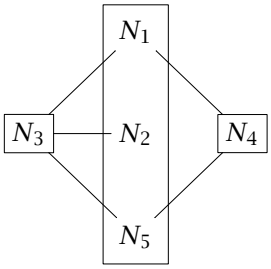

Figure 13: Source N_3.

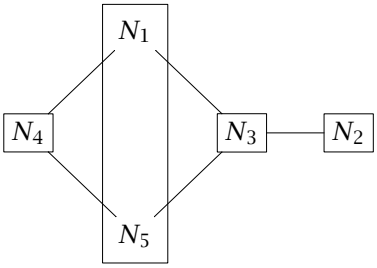

Figure 14: Source N_4.

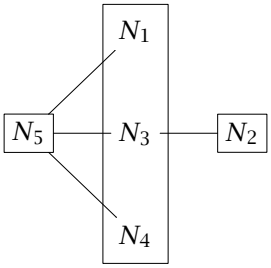

Figure 15: Source N_5.

4.4 Numerical Results: Comparing MaxWill with the Optimal Solution

In this section we implement the binary integer program (IP) introduced in the previous section and compare it with the MaxWill algorithm. The implementation was done in MATLAB using the function *bintprog* (with the cost vector taken to be zero).

Performance ratios. For a node $v \in V$ with battery level B_v and transmission power p_v, let R_{IP} denote the optimal number of rounds, i.e., let R_{IP} be the maximum value of $\mathcal{R}$ such that the binary integer program (IP) is feasible. Let R_{MW} denote the number of rounds that is possible if the MaxWill algorithm is used to select the relay nodes. In Figure 16 we plot the ratio $R_{\text{IP}}/R_{\text{MW}}$ for 100 different simulations, for $n = 5, 10, 15$. For each simulation we generated a binomial model of the Erdös-Rényi random graph, with probability $p = 0.5$ of two nodes

being connected. The battery levels B_v were also random, selected uniformly from the integers in the interval $[20, 30]$. We took the transmission battery depletion to be equal to one, $p_v = 1$, for all nodes in every simulation.

From Figure 16 we see that for $n = 5$ the MaxWill algorithm gives the optimum number of rounds in almost every simulation, 98 out of 100. As the number of nodes is increased the performance of the MaxWill algorithm decreases: For $n = 10$ the MaxWill algorithm is optimal 58% of the time, and for $n = 15$ only 45% of the time. Moreover, the ratios $R_{\text{IP}}/R_{\text{MW}}$ increase as n increases: For $n = 5$ the maximum ratio is 1.29, while for $n = 15$ it is 1.67.

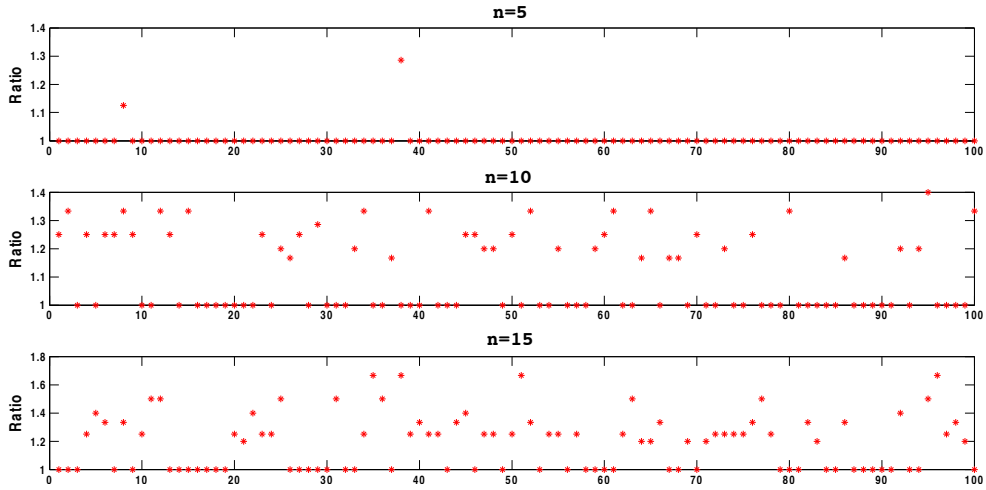

Figure 16: Performance ratios: The optimality of the MaxWill algorithm was tested for 100 different random graphs, with $n = 5, 10, 15$ nodes. The x-axes correspond to the simulation number. The y-axes correspond to the performance ratios $R_{\text{IP}}/R_{\text{MW}}$ of the optimal number of rounds divided by the number of rounds that are achieved using the MaxWill algorithm to select the MPRs.

In Figure 17 we assess the performance of the MaxWill algorithm when the nodes are assigned different power levels. We consider three cases: Power level 1 where $p_v = 1$ for all nodes; Power level 2 where, for each v, p_v is randomly selected from the set $\{1, 2\}$; Power level 3 where, for each v, p_v is randomly selected from the set $\{1, 2, 3, 4\}$. Once chosen, the power levels are fixed for all rounds. For each simulation, we randomly generate a graph with 10 nodes and with probability $p = 0.5$ of having an edge between two nodes. We randomly assign battery levels and power levels as described above.

We see that for the Power level 3, the performance ratio is statistically close to 1. For Power level 1 and Power level 2 MaxWill performs worse. In the case

of Power level 1 MaxWill performs slightly worse than in the case of Power level 2.

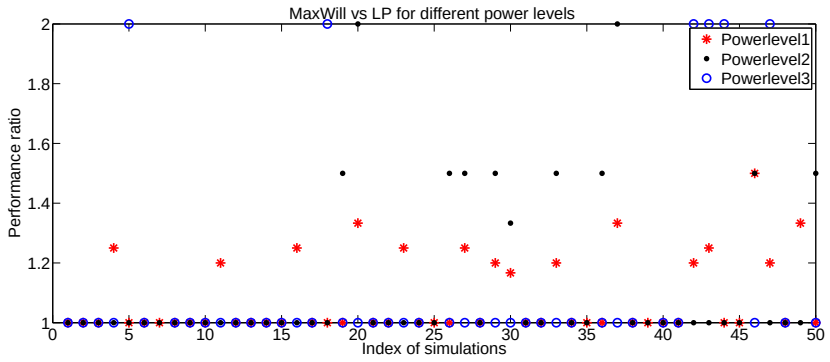

Figure 17: Performance ratios: The optimality of the MaxWill algorithm was tested for 50 different random graphs, with $n = 10$ nodes for different power levels. The x-axes correspond to the simulation number. The y-axes correspond to the performance ratios $R_{\text{IP}}/R_{\text{MW}}$ of the optimal number of rounds divided by the number of rounds that are achieved using the MaxWill algorithm to select the MPRs.

In Figure 18 we study the performance of MaxWill for graphs that have been generated with different probabilities. The different probabilities measure the sparseness of the graph. We took $p = 0.1, 0.3, 0.5, 0.7$ and performed 50 simulations for each case. The plot suggests that the performance indeed depends on the sparseness of the graph. The sparser the graph, the better the performance of MaxWill.

Runtimes. We compare the runtime of the MaxWill algorithm with the runtime needed to find the optimal number of rounds R_{IP} (which was found by solving the binary integer program (IP) for $\mathcal{R} = R_{\text{MW}}$ to $R_{\text{IP}} + 1$). See Table 2, which shows the average runtimes in seconds for the simulations given in Figure 16. Note that the runtimes depend partly on our implementation of the algorithms and no attempt was made to optimize the code. For interest we also present the runtimes of solving the linear programming problem (LP) that is obtained by relaxing the condition $x_{r,s,v} \in \{0, 1\}$ of (IP) to allow $x_{r,s,v}$ to take any value in the interval $[0, 1]$.

As expected, since (IP) is NP-complete, for small n the runtimes of the three algorithms are similar but for large n the binary integer program (IP) can take far longer. Note the big difference between the mean and the median runtime for (IP) when $n = 15$.

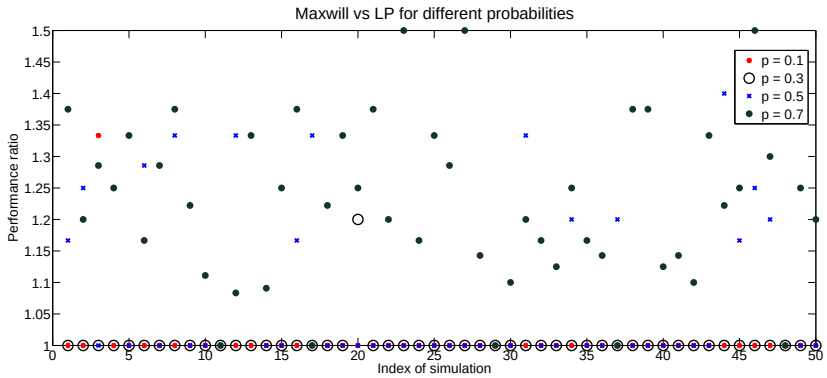

Figure 18: Performance ratios: The optimality of the MaxWill algorithm was tested for 50 different random graphs with $n = 10$ nodes, for different probabilities used for generating the graphs. The x-axes correspond to the simulation number. The y-axes correspond to the performance ratios $R_{\text{IP}}/R_{\text{MW}}$ of the optimal number of rounds divided by the number of rounds that are achieved using the MaxWill algorithm to select the MPRs.

n	MaxWill		IP		LP	
	mean	median	mean	median	mean	median
5	0.06	0.05	0.14	0.12	0.09	0.08
10	0.13	0.13	7.02	0.54	0.27	0.25
15	0.21	0.21	651.66	7.81	0.84	0.71

Table 2: Runtimes: Average runtimes in seconds of the simulations given in Figure 16, plus also the runtimes of the (LP) relaxation of the binary integer program (IP).

This shows that in practice, except for networks with few nodes (around 10 or less), the main role of the binary integer programming formulation (IP) is to assess the performance of the MaxWill algorithm and other heuristics. If the network is small and if the number of desired rounds to be broadcast is known in advance, then (IP) could be solved to choose the MPRs optimally. (Note that (IP) can only be used if you know a priori that the nodes will broadcast in rounds for a given, fixed number of rounds.)

It is interesting to note that, if the number of nodes is small, then the (LP) relaxation of (IP) tends to give the same number of feasible rounds, i.e., $R_{\text{IP}} = R_{\text{LP}}$ when n is small: For $n = 5$ we found that $R_{\text{IP}} = R_{\text{LP}}$ for every simulation, for $n = 10$ we found that $R_{\text{IP}} = R_{\text{LP}}$ for 98 % of the simulations, and for $n = 15$ we found that $R_{\text{IP}} = R_{\text{LP}}$ for 75 % of the simulations.

5 Extensions and Concluding Remarks

In this section we recall the questions posed by Thales Nederland and based on these questions we make some conclusions and recommendations for Thales and discuss possible extensions.

Direction 1: What would be the optimal MPR selection algorithm? With a linear battery decrease model it should be possible to formulate this as a linear program. How much does the optimal solution differ from the known heuristics? Can we define easily a better heuristic than the Maximum Willingness heuristic algorithm used by Thales?

Outside the OLSR framework. In the first part of this paper, Section 3, we relaxed the constraint of Thales that messages have to be sent in layers. We showed in Section 3.1 that the problem of selecting the relay nodes optimally to maximize the lifetime of the network is NP-complete (this is also the case with the layer constraint). In Section 3.3 we introduced a polynomial-time algorithm for choosing the relay nodes. We demonstrated numerically (Section 3.4) that this algorithm out-performs the MaxWill algorithm, in some cases significantly. Therefore if it were possible for Thales to work outside the OLSR framework, Thales could benefit greatly from the use of this new algorithm instead of the MaxWill algorithm.

Inside the OLSR framework. In Section 4 we introduced the constraint of Thales that messages should be broadcast in layers (the OLSR framework). In Section 4.1 we gave a (non-generic) example to show that the MaxWill algorithm can perform arbitrarily badly. To see how it performs in general, two binary linear programs were derived to choose the MPRs optimally. See Sections 4.2 and 4.3. In Section 4.4 the second linear program was implemented to show quantitatively how far the MaxWill algorithm is from being optimal. Thales could use this to decide whether to continue using the MaxWill algorithm or whether to search for a better heuristic. Our linear programming formulations are mainly intended for comparison purposes (to test the optimality of heuristics) but could also be used as algorithms for choosing the relay nodes when the network is small.

Finally we tie together Sections 3 and 4 by commenting on how the performance of MaxWill depends on whether or not the layer constraint is included, for the case of sparse and dense graphs.

Figure 18 shows that, within the OLSR framework, the MaxWill algorithm performs nearly optimally for sparse graphs. On the other hand, from Figure 8 we see that the Path-Based algorithm, which lies outside the OLSR framework, far out-performs the MaxWill algorithm for sparse (but not too sparse) graphs. This

suggests that for sparse graphs a significant improvement in the lifetime of the network could be achieved by working outside the OLSR framework.

Conversely, in the case of dense graphs, Figure 8 shows that the Path-Based algorithm does not perform better than MaxWill (even though the layer constraint is relaxed). From Figure 18 we see that also within the OLSR framework the MaxWill algorithm is far from being optimal. This suggests that for dense graphs it is worth looking for a better polynomial-time algorithm even within the OLSR framework.

Direction 2: Assume additionally that a node can choose between different power levels. For a higher power level a node will have larger set of neighbors to choose its MPR-set from. Can we formulate the optimization problem and find some good heuristic to solve it (a solution being an assignment of transmit powers and MPR-sets)? What would be the impact on the network lifetime?

A further direction of investigation proposed by Thales Nederland consists in allowing the nodes the freedom of choosing the power levels at which they transmit a message (the transmission, however, will still proceed layer by layer). This additional degree of freedom means that the topology of the network changes with every transmission according to the power level each node selects. Hence the set of edges connecting the nodes is essentially an unknown of the problem as well.

We notice that, although the transmission has still to respect the layered structure of the graph, since we can change the topology of the graph before sending the message, examples of the type in Figure 2 showing the weakness of the MaxWill algorithm can be ruled out by properly tuning the power levels. In fact, in the graph in Figure 2, it is possible to turn node 3 into a leaf by suitably choosing its power level. Hence a possible strategy for optimizing the lifetime of the network is to tune the power levels of the nodes so that, in the corresponding graph, the transmission path generated by the Path-Based algorithm becomes admissible within the OLSR framework.

For example, in the case of only two power levels, we could choose the power level of each node using a threshold argument: we could define a threshold battery level such that all nodes with battery level higher than the threshold will broadcast with the highest power level, while nodes with battery levels lower than the threshold will select the lowest power level. This heuristic prevents all “low battery" nodes from transmitting with the high power level, and hence might prolong the lifetime of the network.

Here is another strategy. Assume again that all nodes can use two trans-

mission powers ($p_{\min}$ and $p_{\max}$). First assume that the source uses power $p_{\min}$, provided that it can transmit to at least one node. All 1-hop neighbors of the source will be ordered according to their battery levels in a descending order. Furthermore, consider the two extreme cases where all the nodes transmit with the minimum power $p_{\min}$ or all the nodes transmit with the maximum power $p_{\max}$. Let $L_{\min}$ and $L_{\max}$ denote the corresponding number of relays used according to the MaxWill algorithm. If $L_{\min} \leq L_{\max}$ then all nodes are assigned $p_{\min}$. Otherwise, define the fraction of nodes transmitting with power $p_{\max}$ as follows:

$$R = \begin{cases} (L_{\max} * p_{\max} - L_{\min} * p_{\min})/(L_{\max} * p_{\max}), & \text{if } L_{\max} * p_{\max} - L_{\min} * p_{\min} \geq 0, \\ 1, & \text{otherwise.} \end{cases}$$

We choose which nodes transmit with power $p_{\max}$ according to their battery level, starting from the highest battery level. Having assigned the transmission powers we can now use the MaxWill algorithm to choose the relays in the 1-hop neighborhood of the source. We continue in this way until we cover the entire graph. Finally, we repeat this algorithm for the case where the source uses $p_{\max}$ and decide between the two options.

Of course, these approaches require further investigation and are not complete, but could be the start of a sequel paper.

Direction 3: What is the effect on the network lifetime problem when using a battery model with a recovery effect?

Although Direction 3 is very interesting, due to time limitations we were not able to extend our analysis in this direction. A very frivolous approach to the recovery effect could be to assume that the battery is depleted by a fixed amount for each message transmission and replenished by a fixed amount for each round that it remains idle. Of course this is simply a first thought on tackling the problem, but we would be very much interested to look into it in the future.

Acknowledgements

We would like to thank Dr. Maurits de Graaf for introducing the sensor lifetime maximization problem to us. Moreover, we would like to express our gratitude to Thales Nederland for sharing their problem with us and to the SWI 2012 committee for the wonderful opportunity to work on such an interesting and exciting problem. We are grateful to Prof. Remco van der Hofstad for many useful suggestions and improvements regarding this manuscript.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. New York: John Wiley, 2000.

[2] S.-H. Cheong, K.-I. Lee, Y.-W. Si, and L. H. U. Lifeline emergency ad hoc network. *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pages 283 –289, 2011.

[3] T. Clausen and P. Jacquet. RFC 3626: Optimized link state routing protocol (OLSR). Internet draft, `http://www.ietf.org/rfc/rfc3626.txt`, 2003.

[4] T. Coenen, J.-K. van Ommeren, and M. de Graaf. Routing versus energy optimization in a linear network. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–6. VDE, 2010.

[5] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, 2002.

[6] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms And Combinatorial Optimization*. Springer, 1993.

[7] T. Moscibroda and R. Wattenhofer. Maximizing the lifetime of dominating sets. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005.

[8] V. V. Vazirani. *Approximation Algorithms*. Springer, 2010.

[9] J.-M. Verbree, M. de Graaf, and J. Hurink. An analysis of the lifetime of OLSR networks. *Ad Hoc Networks*, 8(4):391–399, 2010.

[10] L. Viennot. Complexity results on election of multipoint relays in wireless networks. Report RR-3584, INRIA, 1998.

[11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.