Ad Hoc & Sensor Wireless Networks, Vol. 0, pp. 1–23
Reprints available directly from the publisher
Photocopying permitted by license only

WiLE: Leader Election in Wireless Networks

ABHIMANYU VENKATRAMAN SHESHASHAYEE AND
STEFANO BASAGNI

Institute for the Wireless Internet of Things
Northeastern University
E-mail: abhi@ece.neu.edu, basagni@ece.neu.edu

Received: July 18, 2018. Accepted: March 29, 2019.

Numerous applications of multi-hop wireless networks benefit from or require the use of leaders, determined on the basis of some quantifiable and comparable criteria. In this paper, we propose an efficient scheme for electing the top K leaders of a multi-hop wireless network with $N \geq K$ nodes. The proposed protocol, called *WiLE* for Wireless Leader Election, is distributed and, by means of the sole exchange of packets among neighbors, terminates providing each node with the unique ID of the elected leaders and their rank. The correctness of the protocol is analytically proven. *WiLE* is implemented both on a physical testbed and in the GreenCastalia simulator. Our experimental evaluations demonstrate the effectiveness of *WiLE* in determining network leaders swiftly and efficiently. The performance of *WiLE* is compared to that of other previously proposed leader election protocols, *Vasudevan* and *Raychoudhury*. Results show that *WiLE* determines the top K global network leaders faster and consuming less energy than previous solutions. On average, *WiLE* is shown to complete in 17% of the time taken by *Vasudevan*, transmitting 12% of the bytes transmitted by that protocol. Against *Raychoudhury*, *WiLE* takes 34% of the time and transmits only 23% of the bytes transmitted by *Raychoudhury*.

1 INTRODUCTION

Multi-hop wireless networks, sometimes referred to as ad hoc networks or MANETs [3], are a communication paradigm that have found myriad of applications in different fields, ranging from wireless sensor networks

(WSNs) [27], to vehicular communications networks (VANETs) [5] and underwater networking [22]. In general, these wireless networks are characterized by repeatedly re-configuring topologies, due to mobility (e.g., MANETs and VANETs) or to the intermittent operations of the network nodes (e.g., WSNs). Wireless nodes are often battery powered, thereby requiring a suitable degree of energy efficiency in their operations, especially those concerning data transmission.

Certain classes of wireless network applications require the election of one or more leader nodes among the N nodes of the network. These leaders help with the performing of various special tasks, such as coordination, synchronization, etc. The suitability of a node as a leader is dependent on various factors, which could include available energy, reliability, fault tolerance, etc. Using some predetermined criterion for leadership, each node is weighted by how suitable it is to be leader. The network then attempts to elect the $K \leq N$ nodes with the highest weights to be the leaders. Solutions to this problem have been proposed that, as requested by the nature of multi-hop wireless networks, are primarily distributed protocols striving to optimize speed, data transmitted, energy consumption. The aim is to define election schemes that minimize the time taken to elect the top K leaders, the amount of data transmitted during the election process, and the overall energy consumed by the protocol. A review of solutions is provided in Section 4.

In this paper, we contribute to the problem of determining of the top K leaders of a multi-hop wireless network by proposing a new solution that assigns each node with a rank based on its suitability for leadership. The protocol is named *WiLE* (pronounced 'wily'), for *Wireless Leader Election*. Given a networks with N nodes, *WiLE* ranks all nodes, i.e., assigns to each one of them a unique number in the range 1 through N. The ranking is based on the node suitability for leadership, represented by the node *weight*, a real number. Through ranking the nodes by weight, *WiLE* allows the network to easily identify the nodes that are most suitable for leadership. Ranking is based on the local exchange of "progress view packets" among neighbors, thus being distributed and localized. These packets contain lists of nodes with their corresponding weights and current ranks. As the packet exchange proceeds, each node maintains a list of nodes and ranks, tracking the progress of the rank assignment process. Neighboring nodes reach a consensus on which rank they choose by cross referencing their progress views: If multiple nodes have the same rank, actions are taken to change conflicting ranks and the new assignment is communicated to the neighbors. Eventually, each node will have a view that reflects the state of the entire network. Once every rank on the list is unique, the nodes with ranks 1 to K are elected the leaders of the network, and the protocol terminates.

The effectiveness of *WiLE* in electing the network leaders is demonstrated by evaluating its performance when run on networks whose operations demand energy efficiency, specifically, in scenarios concerning WSN applications. We compare the performance of *WiLE* to that of two other leader election protocols each representing a different approach to selecting the top K leaders. In particular, we chose two solutions that have been shown to outperform all other solutions, namely, the protocols presented by Vasudevan et al. [31] and by Raychoudhury et al. [28]. *WiLE* and the other protocols are implemented in GreenCastalia [4], a freely available network simulator particularly geared towards wireless sensor networking, modeling energy consumption, as well as MAC and physical layer aspects of WSNs realistically. In scenarios with varied network sizes, network densities, and packet sizes, we investigate a few key metrics: total number of packets transmitted, protocol completion time, total number of bytes transmitted, and energy consumed per node. Our results show that *WiLE* outperforms the compared protocols in all metrics. Especially in large WSNs with high density (average number of neighbors per node > 25), we observe that *WiLE* completes the leader election process considerably faster than the compared protocols. This is due to the lower number of packets transmitted during its execution. Energy consumption is also kept at bay, and nodes running *WiLE* spend significantly less energy than nodes running the other protocols. This is achieved by transmitting fewer total bytes over the duration of the protocol.

In order to demonstrate the reliability of the simulation results, *WiLE* is also implemented on a physical testbed comprised of 11 wireless network motes. The testbed topology is recreated within the simulation environment, so to obtain similar set ups. When *WiLE* is evaluated in both the physical testbed and its simulated counterpart the testbed results are found to agree with the simulation.

The remainder of the paper is organized as follows. Section 2 describes *WiLE* in detail and proves its correctness. In Section 3, the performance of *WiLE* is evaluated on a testbed with 11 nodes and through simulations on networks with increasing size and density. Section 4 summarizes previous works on leader election for wireless networks. Finally, in Section 5 we conclude the paper.

2 PROTOCOL DESCRIPTION

In this section we provide a detailed description of our protocol for leader election. We start by defining relevant notation (Section 2.1). We then describe protocol operations through algorithms, a flowchart and an example

(Section 2.2). The section terminates with proof of the protocol correctness (Section 2.3).

2.1 Notation

WiLE is described by using the following notation:

- The set V is the set of the $N = |V|$ nodes in the network. It is assumed that N is known to every node (v) in the network.[1]
- For each $v \in V$, $I(v)$ indicates the unique identifier of node v. This can be any number, in any suitable base. It is assumed that each node v knows $I(v)$.
- For each $v \in V$, $W(v)$ indicates the weight of node v. The weight of a node is the scalar measure of how eligible said node is to be the leader of the network. This value is assumed constant during the execution of the protocol.
- By the end of the execution of *WiLE* every node $v \in V$ is assigned a unique rank $R(v) \in [1, N]$.
- The unique ID, weight, and rank of any node $v \in V$ are stored in the tuple $T(v) = \big(I(v), W(v), R(v)\big)$.
- The *progress view* $P(v)$ of node v stores the set of tuples $\big\{T(v), T(u_1), T(u_2), \cdots\big\}$ received so far from its neighbors. This set represents the node's view of the network: A map of all nodes that have communicated with node v, their weight and their rank.
- The operation **transmit**$\langle m \rangle$ represents a node broadcasting a packet whose payload is m. We assume that these packets are correctly delivered to all the neighbors of the sender by a finite, yet unpredictable, time. Communications are asynchronous, i.e., nodes are not supposed to be synchronized. They broadcast their packets as soon as the protocol generates them.

2.2 *WiLE* Operations

The protocol is executed at each node v in V. Upon starting its operations, node v executes the following initialization procedure *InitLE* (Algorithm 1).[2]

Each node v initially gets the rank 1 and initializes its set $P(v)$ to its own current view, i.e., to the tuple $(I(v), W(v), R(v))$. It is now ready to share its view with its neighbors, and therefore broadcasts $\langle P(v) \rangle$.

[1] This assumption can be relaxed at the cost of applying schemes for determining the size of a multi-hop wireless network [13].

[2] It is unlikely that all the nodes will start the execution of the protocol, or in general, that they will be "turned on" all at the same time. It is however realistic to assume that all nodes are up and running within a limited amount of time from when the first node comes alive. Furthermore, although customary in many wireless network applications, in this case it is not necessary that each node becomes aware of the identity and number of its neighbors. This might be useful, however, when exchange of packets at the MAC layer is implemented through one-to-one communication.

Algorithm 1 *InitLE*() (executed by node v upon starting its operations)

1: $R(v) = 1$
2: $P(v) = \{(I(v), W(v), R(v))\}$
3: **transmit**$\langle P(v) \rangle$

Algorithm 2 *ReceiveProgress*($\langle P(u) \rangle$) (executed by node v upon receiving $\langle P(u) \rangle$ from a neighbor u)

1: $P(v) = MergeProgress(P(v), P(u))$
2: $sameRankSet = \{T(z) \in P(v) : (z \neq v) \wedge (R(z) = R(v))\}$
3: **if** $sameRankSet \neq \emptyset$ **then**
4: $\quad lowestW = \min(W(z) : T(z) \in sameRankSet)$
5: $\quad$ **if** $W(v) < lowestW$ **then**
6: $\quad\quad R(v) = \max(R(z) : T(z) \in P(v)) + 1$
7: $\quad\quad P(v) = MergeProgress(P(v), \{T(v)\})$
8: **if** $P(v)$ has changed **then**
9: $\quad$ **transmit**$\langle P(v) \rangle$
10: **if** $(|P(v)| = N) \wedge (\forall u, z \in V : u \neq z \implies R(u) \neq R(z))$ **then**
11: $\quad$ EXIT

The rest of the algorithm is triggered by the reception of a packet containing $\langle P(u) \rangle$ from a neighbor u, as detailed in the following Algorithm 2, *ReceiveProgress*.

ReceiveProgress begins with the current node v merging its progress view with the progress view just received from node u. To this purpose it uses the function *MergeProgress*. Given two progress views, *MergeProgress* returns a progress view where there are no two tuples with the same node ID. Specifically, if the union of two progress views contains multiple tuples with the same node ID $I(z)$ and different values of $R(z)$, $z \in V$, *MergeProgress* retains only the tuple with the greatest value of $R(z)$, and discards the rest.

After this, a new set $sameRankSet$ is created (line 2). This is the set of all nodes in the newly merged progress view that have the same rank value of node v. If there is at least one node in this set, then node v has to decide whether to change its own rank, since all ranks must be unique. To this purpose, the lowest weight $lowestW$ among the nodes in $sameRankSet$ is selected and compared to the weight of node v. In the event that multiple nodes have the same weight, the tie is broken using node IDs. If $W(v)$ is lower than $lowestW$, then the current node proceeds to change its rank, setting it to the largest rank it has seen so far, increased by 1. The progress view is updated accordingly (line 7). If the progress view for node v has changed over the course of *ReceiveProgress* up to this point, then it broadcasts

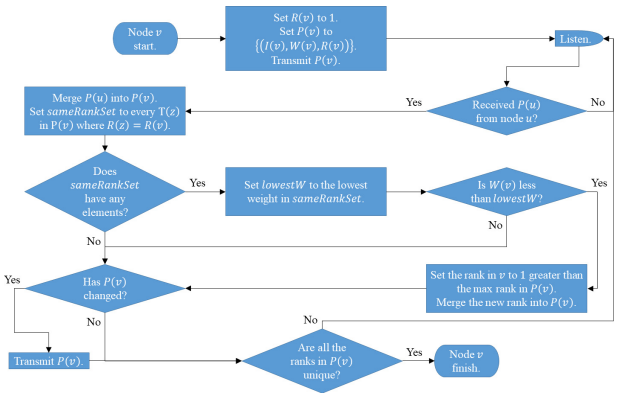

FIGURE 1
WiLE execution flowchart for each node v.

the modified progress view to its neighbors. Finally, if node v progress view has the same size of the network, then node v has received information about every node in the network. If all nodes in its progress view have unique ranks, its progress view accurately represents the entire network IDs, weights, and ranks. The nodes that have ranks 1 to K are the leaders of the network, and node v exits the execution of the protocol.

To prevent the transmission of redundant information by repeated broadcasting of the entire progress view, changes to the progress view during the *ReceiveProgress* algorithm are tracked. Tuples that were either added or modified are moved into the reduced progress view $P_{opt}(v)$, which is then broadcast using **transmit**$\langle P_{opt}(v)\rangle$ in lieu of **transmit**$\langle P(v)\rangle$. If the progress view changes before the reduced progress view is broadcast, those changes are incorporated into the reduced progress view, overwriting previous changes if required.

Figure 1 depicts the execution of *WiLE* at any node v. As soon as *WiLE* starts for node v, *InitLE* (Algorithm 1) is run. Then, the node enters its listening loop. When the node receives the progress view $P(u)$ from any node u, it exits the listening loop and runs *ReceiveProgress* (Algorithm 2). If, by the end of *ReceiveProgress*, all of the ranks in the progress view $P(v)$ are unique, then the protocol terminates for node v. Otherwise, the protocol returns to the listening loop, and the process continues.

Example. Figures 2 to 6 illustrate the operations of *WiLE* in a sample network with five nodes. The weights of the nodes are shown in the top-left corner. Figure 2 depicts the initial state of the network, after the nodes have initialized the protocol. Figures 3 to 5, focus on the interactions of just two nodes, namely, nodes B and D. Figure 3 depicts the state of these two nodes after they have received and processed each others progress views. In

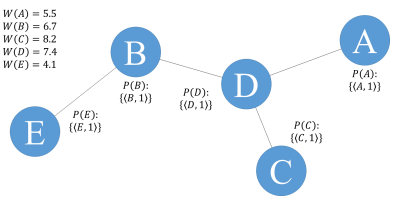

FIGURE 2
Initial state of the network, after *InitLE*.

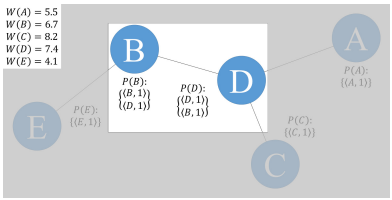

FIGURE 3
Observing the interactions of two nodes.

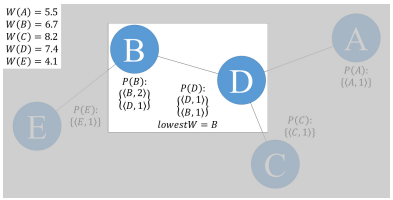

FIGURE 4
Determination of changes based on weight.

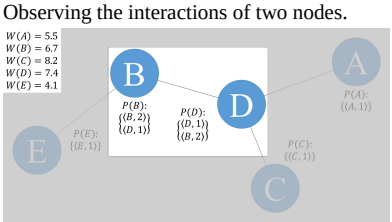

FIGURE 5
After the broadcast from node B.

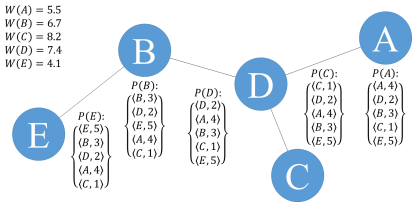

FIGURE 6
Final progress views, with the determination of the leader (node C).

Figure 4, the lowest weight is determined. Since $W(B) < W(D)$, node B modifies its progress view. Between figures 4 and 5, node B broadcasts its updated progress view. Finally, Figure 5 shows the state of the nodes after node D has received the updated progress view from node B, thereby updating its own progress view. Eventually, the network will reach the state depicted in Figure 6, where the network is fully ranked. At this point, the nodes with rank 1 to K are elected the leader of the network. For the case of $K = 1$, this would be node C, which has the largest weight. For larger values of K, this would be followed by nodes D, then B, then A, and finally E, which is in accordance with their weights. In the event of a tie between weights, the the protocol uses the nodes' unique IDs to break the tie.

2.3 Correctness

We show that the execution of *WiLE* on a network with node set V results in every node $v \in V$ attaining a unique rank.

WiLE can be considered to occur in N discrete operational steps. Each step corresponds to the number of connected sub-graphs in V, such that, for each sub-graph L:

- For every node $u \in L$, the progress view $P(u)$ contains the tuples $T(v)$ for every node $v \in L$. More formally:

$$[\forall u, v \in L] : \big(T(v) \in P(u)\big)$$

- For every node $u \in L$, its rank $R(u)$ is unique. More formally:

$$[\forall u, v \in L]\big(u \neq v \rightarrow R(u) \neq R(v)\big)$$

- There exists only one node $u \in L$ such that $R(u) = 1$.

The number of sub-graphs in the network at any given moment is m. When a sub-graph exchanges its progress view with that of another sub-graph, the progress views will be merged. This will result in the the nodes from both sub-graphs getting re-ranked as per the weights of those nodes. When this happens, the two sub-graphs can be considered to have merged into one, and the protocol can be considered to have entered the next step. Thus, the current step at any given point can be expressed as $s = N - m + 1$.

The proof of correctness by induction is as follows. At any given step $s \in [1, N]$, it can be said that the predicate $C(s)$ holds true if the network V can be divided into $m = N - s + 1$ of the aforementioned sub-graphs. The base case is simply that $C(1)$ holds true; this is proven in Lemma 1. The successor case is that if $C(s)$ holds true for some $s \in [1, N)$, then $C(s + 1)$ also holds true; this is proven in Lemma 2. Finally, the conclusion is that $C(s)$ holds true for any $s \in [1, N]$, as proven in Theorem 1.

Lemma 1. *$C(1)$ holds true.*

Proof. At the start of the protocol, every node has a rank of 1. Thus, every node is a sub-graph of its own. This means that there are $m = N$ sub-graphs in V. For every sub-graph L:

- The node $v = L$ holds the progress view $P(v)$, such that $T(v) \in P(v)$.
- Since there is only one node, there is only one rank $R(v)$, which is automatically unique.
- It has already been established that the rank of the node is 1.

Therefore, $C(1)$ holds true. □

Lemma 2. *If $C(s)$ holds true for some $s \in [1, N)$, then $C(s+1)$ also holds true.*

Proof. Since $C(s)$ holds true, thus V can be divided into $m = N - s + 1$ connected sub-graphs. Let L_1 and L_2 be two sub-graphs, such that $u \in L_1$ and $v \in L_2$ are adjacent nodes. Following Lemma 1, $P(u) \supseteq \{[\forall l \in L_1] : T(l)\}$ and $P(v) \supseteq \{[\forall l \in L_2] : T(l)\}$.

At some point, u broadcasts $P(u)$ to all of its neighbors, including v. Shortly thereafter, v receives the broadcast progress view. The node v merges $P(u)$ with $P(v)$ to create the new progress view $P(uv) \supseteq \{[\forall l \in L_1] : T(l)\} \cup \{[\forall l \in L_2] : T(l)\}$.

Eventually, the node v broadcasts the new progress view $P(uv)$. Subsequently, the node u receives $P(uv)$, and updates its own progress view. Following this exchange, the progress view $P(uv)$ is propagated to all of the other nodes in both L_1 and L_2. When every node in $L_1 \cup L_2$ share the same merged progress view, they are all guaranteed to have unique ranks.

Initially, both L_1 and L_2 each had a node with a rank of 1. However, after the two progress views were merged, one of those two nodes would have had to increment its rank. The other would be the only node left in $L_1 \cup L_2$ with a rank of 1.

By this point, L_1 and L_2 can be considered to have merged. Now, the total number of connected sub-graphs (bearing the aforementioned properties) remaining is $m = N - s$. Therefore, $C(s+1)$ holds true. □

Theorem 1. *$C(s)$ holds true for any $s \in [1, N]$.*

Proof. Following Lemma 1, $C(1)$ holds true. Following Lemma 2, if $C(s)$ holds true for some $s \in [1, N)$, then $C(s+1)$ also holds true. Therefore, by induction, $C(s)$ holds true for any $s \in [1, N]$. □

Since $C(N)$ holds true, that means that every node eventually attains a unique rank. When every node in the network has a unique rank, the top K leaders naturally arise from the ordering.

3 PERFORMANCE EVALUATION

We evaluate the performance of *WiLE* via simulations on networks of variable size and density, and running it on a small testbed. We start by defining the node characteristics and the settings used in our experiments (Section 3.1). This is followed by definitions of evaluated metrics that are investigated for both simulation-based experiments and on the testbed (Section 3.2). We then

provide a simulation-based comparative performance evaluation of *WiLE* against two leader election protocols for wireless networks (Section 3.3). After this, we show results from a physical testbed, set up indoors in the Northeastern University main campus (Section 3.4). Testbed-based results are also used as a means to validate simulations.

3.1 Node characteristics and settings

The wireless networks considered in our experiments are comprised of *MagoNode++* motes. The MagoNode++ is a low-power wireless mote featuring an integrated transceiver that uses an IEEE 802.15.4 compliant 2.4 GHz radio module [24]. Its on-board microcontroller uses an 8-bit, 16MHz CPU with 256 KB of ROM and 32 KB of RAM. The MagoNode++ uses a simple CSMA at the MAC layer. The channel data rate is 250 Kbps per second. The transmission power is -2 dBm.

Each node has a unique ID. The length of a node unique ID (UID) is 16 bits, which is one of the options provided by the IEEE 802.15.4 specifications [15]. In our experiments unique IDs were assigned to the nodes randomly, making sure that no two nodes receive the same ID. Each node has a weight. The weight is a real value, represented by a double precision floating point variable that is 8 bytes in size. The node weights are randomly generated and assigned prior to each experiment.

3.2 Performance metrics

Performance is evaluated with respect to the following metrics:

1. *Number of packets*. This is the total number of packets transmitted by the nodes in the network.
2. *Completion time*. This is the time it takes to complete leader election, in seconds. It is measured as the time from when the last node to start commences the execution of the protocol to the time when every node has finished it, i.e., it knows the K global leaders in the network.
3. *Bytes transmitted*. This is the total number of bytes sent by all nodes over the duration of the protocol. It is determined by counting the bytes of each broadcast packet, the MAC layer overhead and the overhead from the physical layer.
4. *Energy consumed per node*. This is the average energy consumed by each node during the execution of the protocol. The energy needed for transmission and reception is determined using the bytes of each broadcast packet, the MAC layer overhead and the overhead from the physical layer. The byte count is multiplied by the average power needed for transmission and reception and the time it take to transmit or receive those bytes. The energy consumption for the entire network is calculated, and that value is divided by the number of nodes.

3.3 Simulation-based comparative performance evaluation

WiLE and the top-K leader election protocols proposed by Vasudevan et al. [31] and Raychoudhury et al. [28] are implemented in the open-source simulator GreenCastalia [4].[3] GreenCastalia is an extension of the Castalia simulator that takes into consideration energy-related aspects of WSNs in details [8]. Communication and other parameters of the MagoNode++ are modeled accurately [30]. Packet collisions are determined based on the additive interference model, according to which simultaneous transmissions from multiple nodes are calculated as interference at the receiver by linearly adding their effect. When not transmitting or receiving packets, it is assumed that the node radios are in sleep mode, i.e., consume negligible energy. This can be obtained by endowing each node with an on-board wake-up low-power radio receiver [30].

In our experiments, topologies of different sizes are generated by randomly scattering nodes over a square area of the size 500 m $\times$ 500 m. We only considered fully connected topologies.[4] The simulator uses the generated topologies to evaluate *WiLE*, *Vasudevan*, and *Raychoudhury*. The network size N is varied in $\{25, 50, 75, \ldots, 200\}$. The topologies so obtained have nodes whose average number of neighbors (density) varied from 3.6 ($N = 25$) to 27.5 ($N = 200$), thus providing insights on the protocol operations on sparse to very dense WSNs. The number of leaders K to elect is set to the value of N. The packet size P is varied in $\{31, 63, 95, 127\}$ B. Subsequently, the corresponding MAC overhead sizes (i.e., the sum of header size and trailer size) are set to $\{18, 20, 23, 25\}$ B. These overheads sizes reflect MAC addressing schemes for proportionally sized packets, based of the 802.15.4 specification [15]. Results are averages over 1000 simulation runs for each value of N and P. This achieves a 3.1% statistical precision for a 95% confidence level. Table 1 summarizes all of the experiment parameters.

The results of the experiments on the simulated randomized topologies are as follows:

1. *Number of packets*. Figure 7 shows the total number of packets transmitted during the execution of each protocol, for all packet sizes. It can be seen that the number of packets increases for larger networks, as expected: Larger networks must exchange more messages in order to complete leader election, resulting in more packets being transmitted. However, the number of packets decreases for larger packet sizes. This happens because for larger packets more data can be sent in each packet,

[3] We describe the two protocols used to benchmark the performance of *WiLE* in Section 4.1.

[4] Our protocol would correctly work also in disconnected networks. In that case it would find the top K leaders in each of the network connected components.

Parameter	Value
Radio hardware	MagoNode++
MAC	CSMA
Data rate	250 Kbps
Transmission power	−2 dBm
Length of node UID	16 bits (IEEE 802.15.4)
Number of nodes N	25, 50, 75, . . . , 200
Number of leaders K	N
Packet size P	31 B, 63 B, 95 B, 127 B
MAC overhead size	18 B, 20 B, 23 B, 25 B
PHY overhead size	6 B
Runs	1000

TABLE 1
Simulation parameters.

which allows protocol messages to be transmitted in fewer overall packets. In any case, *WiLE* requires fewer packets transmitted to complete leader election. *WiLE* transmits an average of 17% of the number of packets transmitted by *Vasudevan*. Compared to *Raychoudhury*, *WiLE* transmits an average of 34% of the number of packets. This is because both *Vasudevan* and *Raychoudhury* build support structures in order to complete the leader election process. *Vasudevan* builds a *spanning tree* rooted at the node that initiates the election algorithm. *Raychoudhury* builds *diffusion trees* that are rooted at particular nodes in the network. These support structures require the use of a multiple packet types while *WiLE* uses just one packet type. Since *WiLE* does not use such support structures, fewer packets are generated and transmitted.

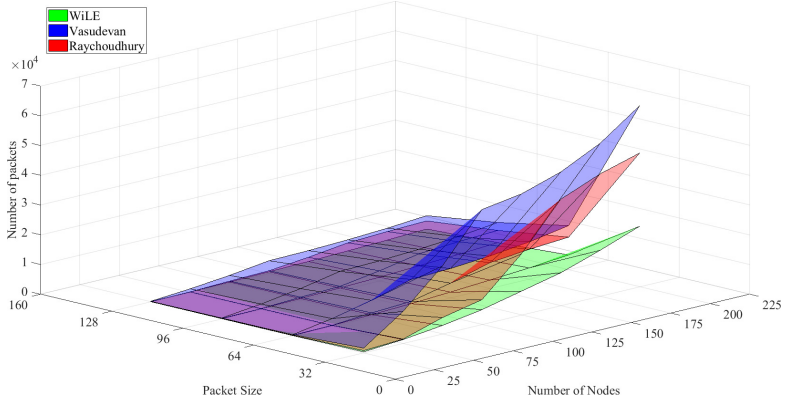

FIGURE 7
Number of packets.

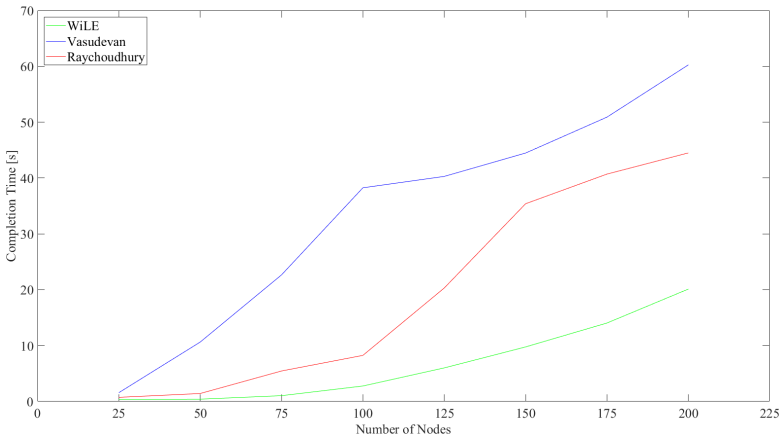

FIGURE 8
Completion time.

2. *Completion time*. Figure 8 shows the time taken to complete the election process for each protocol for $P = 128$ B. In general, it is observed that larger networks require significantly more time for the leader election protocols to complete. Since larger networks result in a greater number of packets transmitted, there are more collisions and re-transmissions, thus imposing longer runtime. Irrespective of network size, *WiLE* is significantly faster than all of the compared protocols because of the localized nature of packet exchange. In the case of *Vasudevan* and *Raychoudhury*, the construction of the support structures using multiple types of packets and end-to-end messaging results in slower protocol execution. Completion time is proportional to the number of packets transmitted. Thus, on the average, *WiLE* completes in 17% of the time taken by *Vasudevan*, and 34% of the time taken by *Raychoudhury*. Results for other packet sizes show similar trends. We observe that increased packet size results in shorter protocol runtime. This is because larger packet size results in fewer total packets needing to be transmitted, resulting in fewer collisions and re-transmissions.
3. *Bytes transmitted*. Figure 9 shows the numbers of bytes transmitted during the execution of each protocol for $P = 128$ B. For all protocols, there is a direct relationship between the size of the network and the number of bytes transmitted: The larger the network, the more the bytes. This corresponds to the increased number of packets exchanged throughout the network. However, the number of bytes transmitted is not strictly proportional to the number of packets transmitted. Regardless, *WiLE* resulted in the lowest number of bytes transmitted of all the protocols. *WiLE*

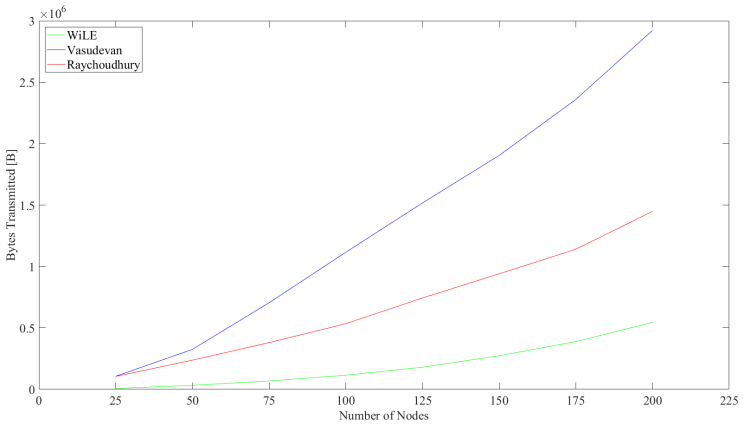

FIGURE 9
Bytes transmitted.

transmits between 6% and 19% of the number of bytes transmitted by *Vasudevan*, averaging around 12%. Compared to *Raychoudhury*, *WiLE* transmits between 7% and 38% of the number of transmitted bytes, averaging around 23%. For different packet size we notice similar trends. In general, we observe that, contrary to intuition, increased packet size correlates with increased bytes transmitted. This is because, even though the overall number of packets is fewer, the increased overhead of the larger packets results in more bytes being transmitted.

4. *Energy consumed per node* Figure 10 shows the amount of energy consumed by each protocol for $P = 128$ B. Since energy consumption

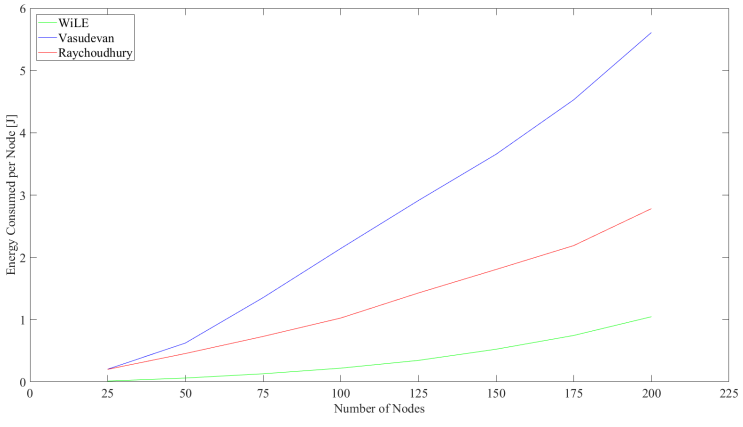

FIGURE 10
Energy consumed.

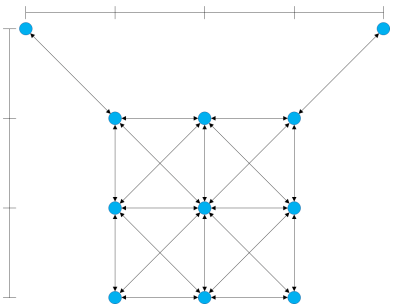

FIGURE 11
Testbed topology.

FIGURE 12
Neighboring nodes in the testbed.

correlates with the bytes transmitted, this too grows with the size of the network. Again, *WiLE* consumed the least energy of all the protocols, with an average of 12% of the energy consumed by *Vasudevan* and an average of 23% of the energy consumed by *Raychoudhury*. As before, this is mainly due to the localized message exchange, as opposed to the network wide tree construction and end-to-end messaging scheme of the other two protocols.

3.4 Testbed-based evaluation

WiLE is also evaluated on a testbed of 11 MagoNode++. The topology of our network is depicted in Figure 11. The testbed is deployed on level 4 of Snell Library, in the Northeastern University's Boston campus. Figure 12 shows neighboring nodes in the testbed.

Two packet sizes are considered: 47 bytes (the MagoNode++ default), and 128 bytes (the maximum packet size, as per the IEEE 802.15.4 specifications [15]). The corresponding MAC overhead sizes are set to 19 bytes and 25 bytes, as per the 802.15.4 specification [15]. As with the simulations, the number of leaders K to elect is set to the value of N. For each packet size, *WiLE* is run 1000 times on the testbed. Table 2 summarizes the experiment parameters.

The results of the experiments on the physical testbed are listed in Table 3.

We observe that the number of packets decreases as packet size increases. This is expected, since larger packets can transmit more data per packet, allowing the protocol messages to be transmitted in fewer packets. Similarly, completion time decreases slightly with the larger packet size. This is because the larger packet size results in fewer packets needing to be transmitted, which reduces the number of collisions. However, the bytes transmitted and energy consumed per node increases with increased packet size. Despite the reduced number of larger packets transmitted, larger packets have larger

Parameter	Value
Radio hardware	MagoNode++
MAC	CSMA
Data rate	250 Kbps
Transmission power	−2 dBm
Length of node UID	16 bits (IEEE 802.15.4)
Number of nodes N	11
Number of leaders K	N
Packet size P	47 B, 127 B
MAC overhead size	19 B, 25 B
PHY overhead size	6 B
Runs	1000

TABLE 2
Testbed parameters.

Packet Size [B]	Number of packets	Completion Time [s]	Bytes Transmitted [B]	Energy consumed [J]
47	103.7	0.170501339	3317.83	0.006370234
128	93.59	0.165067623	3728.075	0.00715776

TABLE 3
Performance results for the physical testbed.

Packet Size [B]	Number of packets	Completion Time [s]	Bytes Transmitted [B]	Energy consumed [J]
47	102.41	0.167360649	3290.47	0.006317702
128	95.75	0.161772405	3736.75	0.00717456

TABLE 4
Performance results for the simulated testbed.

headers than smaller packets (25 B vs. 19 B, respectively), which contributes to more bytes being transmitted overall. Since the energy consumption is proportional to the bytes transmitted, this also increases.

In order to verify the accuracy of our simulation results, we replicate the physical testbed in the simulator, using the same topology (Figure 11). All of the experiment parameters are the same as the parameters of the physical testbed evaluation, as listed in Table 2. We run *WiLE* 1000 times on the simulated testbed. The results of the experiments on the simulated testbed are listed in Table 4.

We observe that the testbed results agree with those of the simulations. This provides validation of our simulation-based investigation.

4 RELATED WORKS

Several works describe leader election protocols that optimize various criteria, and which operate on different kinds of wireless networks. For example, Bansal et al. propose three protocols—one deterministic and two randomized—that strive to minimize the number of time slots required to execute on a multi-channel wireless network [2]. Olabiyi et al. propose a clustering hierarchy based leader election protocol for cognitive radio networks [23]. This protocol uses the quality of service (QoS) of a cognitive radio node as part of the leader election criteria, as well as energy and location. Finally, Cahng et al. [9] propose a consensus based leader election protocol that is an improvement on the Bully Election protocol, originally proposed for general distributed systems [14]. This protocol uses residual battery power and node degree to determine the most eligible leaders, with the general aim to optimize energy efficiency as well as network consistency.

Leader election in wireless networks with mobile nodes is more complex. Protocols for such networks have to be able to account for changes in the topology. Malpani et al. propose two leader election protocols that are based on the Temporally-Ordered Routing Algorithm (TORA) [20]. These protocols manipulate the network topology in such a way that it converges to a directed acyclic graph (DAG), whose sink is the leader. The first protocol can tolerate a single topology change, and is guaranteed to be correct. The second protocol can tolerate multiple topology changes. Parvathipuram et al. propose a message-efficient leader election protocol for single-hop MANETs, which is proven to guarantee leader election [25]. Combining multiple aforementioned concepts, Masum et al. propose a consensus based leader election protocol [21]. The proposed protocol uses extrema finding and proves correctness by evaluating the fairness of the algorithm. Boukerche et al. [6, 7] propose an improvement to the algorithm described by Vasudevan et al. [31] to operate on a mobile network. The proposed protocol constructs a spanning tree across the network and elects the node with the highest remaining battery life. It is shown that the proposed protocol executes in less time and with fewer messages than its predecessor. Lee et al. propose a diffusing-computation, candidate-based leader election algorithm for improved message efficiency [19]. Rahman et al. [26] also propose a candidate-based leader election protocol, which extends the algorithm described in Vasudevan et al. [31]. Ingram et al. [16] build on ideas first introduced by Malpani et al. [20], such as using TORA and building a DAG. The proposed protocol is

proved to guarantee the determination of unique leaders for every connected component of the topology. Chung et al. describe the Regional Consecutive Leader Election (RCLE) problem—a stricter definition of the leader election problem for a dynamic network [10]. An algorithm is proposed, which solves RCLE in a fixed 2 or 3 dimensional space, without relying on the knowledge of the number of nodes or a common start-up time. Finally, Chung et al. improve on the RCLE protocol by bounding the runtime, reducing the message complexity, and accounting for the use of non-synchronized clocks [11].

WSNs tend to use a source-sink model for information consolidation. A typical WSN has a special node called a sink, which gathers all the sensor information relayed to it from source nodes across the network. The information flow can be further streamlined by partitioning the network into clusters, and having a cluster head act as a local sink for the cluster. Being a cluster head consumes significant amounts of energy, so cluster heads are intermittently changed to prevent individual nodes from being excessively drained. This process is known as load balancing. Thus, leader election is a little more complex for WSNs. Shirmohammadi et al. propose a leader election protocol that uses clustering hierarchy to elect leaders for clusters, which then connect to the network sink [29]. Other investigations build upon the cluster head concept, while focusing on more specific criteria. For example, Dong et al. propose a resilient cluster head protocol that safeguards against malicious attacks [12]. The proposed scheme focuses on node communication security. It also makes the network highly resistant to message loss. Going in a different direction, Akele et al. propose a highly efficient leader-group based leader election protocol that minimizes the message passing and energy consumption during election cycles [1]. The proposed protocol is shown to significantly outperform original and improved versions of the Bully Election algorithm. In another vein, Khedr et al. consider a stable election routing protocol that uses compressive sensing cluster head election to build the routing graph [17]. The described protocol uses probability-based hierarchical clustering approach that uses a heterogeneous 3-tier node setting. Similarly, Kim et al. propose a routing protocol that employs a novel cluster-head selection technique [18]. The selection of cluster heads is done using an adaptive, probabilistic scheme.

WBANs share the same complications as WSNs, but with more salient constraints, since WBAN nodes have to be small enough to be worn on or implanted in human bodies. Leader election protocols for WBANs have to be extremely energy efficient, in order to prolong the lifetime of the network. Zhang et al. propose a cluster leader protocol called Energy-Efficient Leader Election (EELE), which elects leaders based on their residual energy and locations [32]. The proposed scheme also implements a hybrid communication mode to reduce the burden of communication for nodes that are further

away from the leader. Zhang et al. extend the previous protocol as Reliable and Energy-Efficient Leader Election (REELE) [33]. This protocol focuses on node reliability, incorporating a node reliability model and total energy consumption model, to optimize the longevity of the network.

4.1 Benchmarking protocols

The two protocols we chose to benchmark the performance of *WiLE* have been found to be the best performing among those proposed for top K leader election protocols for multi-hop wireless networks. In this section we provide a brief description of each protocol. The reader is referred to the original papers for further details.

Vasudevan et al. propose a protocol for electing the top K leaders of a wireless network that uses diffusing computation methods [31]. The protocol operates by growing and shrinking a *spanning tree* that is centered around a "source" node, which initiates the algorithm. When the *spanning tree* construction has completed, the root of the tree has information on the entire network and the top K 'heaviest' nodes. These become the leaders of the network. The growth of the *spanning tree* is facilitated by the use of *Election* messages. When a node first receives an *Election* message, it designates the sending node as its parent. If a node receives a subsequent *Election* message that is not its parent, it responds with an *Ack* message. After a node has received *Ack* messages from all of its children, it sends an *Ack* message back to its parent. If a node does not have children, then it immediately sends its *Ack* to its parent. Thus, after the *spanning tree* has been constructed across the network, a wave of *Ack* messages are sent to "shrink" the tree back to the source node. As the tree shrinks, children send their parents the identifiers and weights of the 'heaviest' nodes in their respective sub-trees. Eventually, the source node learns the identifier and weight of the 'heaviest' node in the network. Finally, a *Leader* message is broadcast to all nodes in the network, informing them of their newly elected leader. For the election of K leaders, the shrinking of the tree must return the identifiers and weights of all of the nodes in the network. Then, the source node can sort this information and select the top K leaders.

Raychoudhury et al. propose a protocol to determine the top K leaders of a network using diffusing computation methods [28]. Leaders are prioritized based on the nodes battery power, computational capabilities, etc. The protocol asynchronously builds diffusion trees, starting by electing the highest weighted nodes in 2-hop neighborhoods as local leaders. A nodes is initially designated as *White*, prompting it to broadcast an *Elect* message that advertises its weight to its neighbors. When a node has received *Elect* messages from all of its neighbors, it sends a *Vote* message to the neighbor with the highest weight. A *White* node that has received *Vote* messages from all of

its neighbors changes into a *Red* node. Eventually, there will be one or more isolated *Red* nodes in the network. Each of these local leaders begins constructing a *diffusion tree* by sending a *Search* message to all its neighbors. When a *White* node receives a *Search* message, it designates the sending node as its *predecessor* and the root of the local tree as the *visitor*. A node tracks the different *visitors* from which it receives *Search* messages, sending *Ack* or *Nack* messages in response, depending on whether the *visitor* of the received message is the same as the visitor of the node or not. Eventually, the *diffusion trees* reach their boundaries, and begin to shrink towards their root nodes. Nodes forward their lists of known node identifiers and weights to the root of their *diffusion tree* via *Signal* messages. *Red* nodes exchange their information using *Result* and *Direct* messages, until the highest weighted *Red* node has information on all of the nodes in the network. This node, now the global leader, chooses the top K highest weight nodes as leaders, and propagates this information to the rest of the network with *Leader* messages.

5 CONCLUSIONS

This paper introduces *WiLE*, a new top K leader election protocol for multihop wireless networks. Specifically, *WiLE* ranks each of the N nodes of the network in the range 1 through N, depending on their suitability for being leaders. *WiLE* is implemented on a testbed, to determine a baseline for performance. Then, the testbed is recreated in the GreenCastalia simulator. The physical testbed and simulated testbed are compared, and their results are found to be similar, thereby validating our simulation results. Subsequently, the performance of *WiLE* is evaluated via exhaustive simulations over randomly generated topologies of wireless networks scenarios, with varying network sizes and packet sizes. We investigated the time it takes to the protocols to complete the leader election process, the number of bytes transmitted to execute the protocol, the energy consumed by the execution of the protocol, and the average number of bytes per message. The performance of *WiLE* is compared to that of other top K leader election algorithms. Our results show that our protocol performs leader election significantly faster than the previous solutions, while requiring considerably lower amounts of energy.

Future directions of our research include the evaluation of the energy savings produced by using the assigned ranks as addresses for routing. We are also considering expanding our testbed experiments. Other directions concern ways of making *WiLE* more scalable, possibly in conjunction with a clustering scheme, and of further developing *WiLE* to more appropriately deal with network partitions and merges (e.g., due to mobility), as well as with node failures.

REFERENCES

[1] G. Akele, H. Redwan, and K. Kim. (July 8–11 2014). Virtual group leader election algorithm in distributed WSN. In *Proceedings of IEEE ICUFN 2014*, pages 143–148, Shanghai, China.

[2] T. Bansal, N. Mittal, and S. Venkatesan. (October 26–28 2008). Leader election algorithms for multi-channel wireless networks. In *Proceedings of WASA 2008*, volume LNCS 5258, pages 310–321, Dallas, TX.

[3] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors. (March 5 2013). *Mobile Ad Hoc Networking: Cutting Edge Directions*. IEEE Series on Digital & Mobile Communication. IEEE Press and John Wiley & Sons, Inc., Piscataway, NJ and Hoboken, NJ, second edition.

[4] D. Benedetti, C. Petrioli, and D. Spenza. (November, 11–14 2013). GreenCastalia: An energy-harvesting-enabled framework for the Castalia simulator. In *Proceedings of ACM ENSSys 2013*, Rome, Italy.

[5] S. K. Bhoi and P. M. Khilar. (2014). Vehicular communications: A survey. *IET Networks*, 3(3):204–217.

[6] A. Boukerche and K. Abrougui. (July 3–6 2006). An efficient leader election protocol for mobile networks. In *Proceedings of IWCMC 2006*, pages 1129–1134, Vancouver, BC, Canada.

[7] A. Boukerche and K. Abrougui. (June 24–28 2007). An efficient leader election protocol for wireless quasi-static mesh networks: Proof of correctness. In *Proceedings of IEEE ICC 2007*, pages 3491–3496, Glasgow, Scotland.

[8] A. Boulis. (2007). Castalia: Revealing pitfalls in designing distributed algorithms in WSN. In *Proceedings of ACM SenSys 2007*, pages 407–408, Sydney, Australia.

[9] H. C. Cahng and C. C. Lo. (June 4–6 2012). A consensus-based leader election algorithm for wireless ad hoc networks. In *Proceedings of IEEE IS3C 2012*, pages 232–235, Taiwan.

[10] H. C. Chung, P. Robinson, and J. L. Welch. (September 16 2010). Regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of ACM DIALM-POMC Joint Workshop FOMC 2010*, pages 81–90, Cambridge, MA, USA.

[11] H. C. Chung, P. Robinson, and J. L. Welch. (June 9 2011). Optimal regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of ACM FOMC 2011*, pages 52–61, San Jose, California, USA.

[12] Q. Dong and D. Liu. (June 22–26 2009). Resilient cluster leader election for wireless sensor networks. In *Proceedings of IEEE SECON 2009*, pages 1–9, Rome, Italy.

[13] J. Evers, D. Kiss, W. Kowalczyk, T. Navilarekallu, M. Renger, L. Sella, V. Timperio, A. Viorel, S. van Wijk, and A. J. Yzelman. (January 24–28 2011). Node counting in wireless ad-hoc networks. *Proceedings of ESGI 2011*.

[14] H. Garcia-Molina. (January 1982). Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31(1):48–59.

[15] IEEE. (April 2016). IEEE standard for low-rate wireless networks. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY). specifications for low-rate wireless personal area networks (LR-WPANs).

[16] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. (May 23–29 2009). An asynchronous leader election algorithm for dynamic networks. In *Proceedings of the IEEE IPDPS 2009*, pages 1–12, Rome, Italy.

[17] A. M. Khedr and D. Omar. (January 2015). Sep-cs: Effective routing protocol for heterogeneous wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 26(1–4):211–232.

[18] K. T. Kim and H. Y. Youn. (2015). An energy-efficient and scalable routing protocol for distributed wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 29(1–4):195–212.

[19] S. Lee, R. M. Muhammad, and C. Kim. (May 14–16 2007). A leader election algorithm within candidates on ad hoc mobile networks. In *Proceedings of ICESS 2007*, pages 728–738, Daegu, Korea.

[20] N. Malpani, J. L. Welch, and N. Vaidya. (August 11 2000). Leader election algorithms for mobile ad hoc networks. In *Proceedings of ACM DIAL M 2000*, pages 96–103, Boston, MA.

[21] S. M. Masum, A. A. Ali, and M. T. Y. Bhuiyan. (April 18–20 2006). Asynchronous leader election in mobile ad hoc networks. In *Proceedings of IEEE AINA 2006*, volume 2, pages 827–831, Vienna, Austria.

[22] T. Melodia, H. Kulhandjian, L.C. Kuo, and E. Demirors. (March 5 2013). Advances in underwater acoustic networking. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, editors, *Mobile Ad Hoc Networking: Cutting Edge Directions*, chapter 23, pages 804–852. John Wiley & Sons, Inc., Hoboken, New Jersey, USA.

[23] O. Olabiyi, A. Annamalai, and L. Qian. (January 14–17 2012). Leader election algorithm for distributed ad-hoc cognitive radio networks. In *Proceedings of IEEE CCNC 2012*, pages 859–863, Las Vegas, NV.

[24] M. Paoli, Dora Spenza, Chiara Petrioli, M. Magno, and L. Benini. (11–14 April 2016). MagoNode++: A wake-up radio-enabled wireless sensor mote for energy-neutral applications. In *Proceedings of ACM/IEEE IPSN 2016*, pages 1–2.

[25] P. Parvathipuram, V. Kumar, and G. Yang. (December 22–24 2004). An efficient leader election algorithm for mobile ad hoc networks. In *Proceedings of ICDCIT 2004*, pages 32–41, Bhubaneswar, Odisha, India.

[26] M. M. Rahman, M. Abdullah-al wadud, and O. Chae. (February 2008). Performance analysis of leader election algorithms in mobile ad hoc networks. *International Journal of Computer Science and Network Security*, 8(2):257–263.

[27] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin. (April 2014). Wireless sensor networks: A survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48.

[28] V. Raychoudhury, J. Cao, R. Niyogi, W. Wu, and Y. Lai. (August 2014). Top K-leader election in mobile ad hoc networks. *Pervasive and Mobile Computing*, 13:181–202.

[29] M. M. Shirmohammadi, K. Faez, and M. Chhardoli. (January 6–8 2009). LELE: Leader election with load balancing energy in wireless sensor network. In *Proceedings of CMC 2009*, volume 2, pages 106–110, Kunming, Yunnan, China.

[30] D. Spenza, M. Magno, S. Basagni, L. Benini, M. Paoli, and C. Petrioli. (April 26–May 1 2015). Beyond duty cycling: Wake-up radio with selective awakenings for long-lived wireless sensing systems. In *Proceedings of IEEE INFOCOM 2015*, pages 522–530, Hong Kong.

[31] S. Vasudevan, J. Kurose, and D. Towsley. (October 5–8 2004). Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of IEEE ICNP 2004*, pages 350–360, Berlin, Germany.

[32] R. Zhang, H. Moungla, and A. Mehaoua. (December 8–12 2014). An energy-efficient leader election mechanism for wireless body area networks. In *Proceedings of IEEE Globecom 2014*, pages 2411–2416, Austin, TX USA.

[33] R. Zhang, H. Moungla, and A. Mehaoua. (June 8–12 2015). A reliable and energy-efficient leader election algorithm for wireless body area networks. In *Proceedings of IEEE ICC 2015*, pages 530–535, London, UK.