RPL Load Balancing via Minimum Degree Spanning Tree

Marwa Mamdouh, Khaled Elsayed, Ahmed Khattab
Department of Electronics and Communications Engineering
Faculty of Engineering, Cairo University
Giza, Egypt 12613
marwa_mamdouh90@cu.edu.eg, khaled.elsayed@cu.edu.eg, akhattab@ieee.org

***Abstract*— RPL is the routing protocol for low-power and lossy networks and has recently been considered as the standard routing protocol for IPv6 based wireless sensor networks using the IEEE 802.15.4 protocol. RPL is oriented towards supporting multi-point to point communications in which multiple points typically communicate with one sink node that considered the root of the RPL tree. RPL aims to minimize the cost to reach the sink node from any node in the network using an objective function. However, RPL lacks a load balancing mechanism to maximize the lifetime of the battery-powered wireless sensor networks by preventing overloaded nodes from being drained quickly. In this paper, we propose the Minimum Degree RPL (MD-RPL) which builds a minimum degree spanning tree to enable load balancing in RPL. MD-RPL modifies the original tree formed by RPL to decrease its degree. Extensive simulations were carried out to evaluate the proposed scheme using the COOJA simulator. Our MD-RPL results in up to 15.6% improvement in the maximum power consumed which implies an improvement in the lifetime of the network.**

***Index Terms*— Load Balancing, Minimum Degree Spanning Tree, Wireless Sensor Networks, Low-power Lossy Network (LLN)**

I. INTRODUCTION

The Routing Protocol for Low-Power and Lossy networks (RPL) has recently been considered as the standard routing protocol for IPv6 based low power personal area networks using the IEEE 802.15.4 protocol [1][2]. IPv6 over Low Power Wireless Personal Area Network (6LoWPAN) are widely used in many applications nowadays such as environmental, medical and military applications [3]. RPL is designed to meet the different requirements of 6LoWPANs. It guarantees a fast network setup that allows the efficient monitoring of critical applications [4]. PRL is optimized to support Multipoint-to-Point (MP2P) traffic. It builds a loop free tree using an objective function to minimize the cost to reach the sink node from any node in the network. However, RPL needs a load balancing mechanism to maximize the lifetime of 6LoWPANs to distribute the load on the different 6LoWPAN nodes and prevent overloaded nodes from being battery-drained. Wireless Sensor Networks (WSNs) (as an example of 6LoWPANs) consist of hundreds or thousands of small and cheap nodes called sensor nodes which are powered by batteries causing the sensor nodes to be prone to failures. Thus, the lifetime of these nodes is a main concern for WSNs.

Replacing the battery of sensor nodes is not practical because of the large number of nodes, besides, it is not appropriate solution economically [5]. Therefore, it is crucial to minimize the power consumption of sensor nodes to save their batteries and extend the lifetime of the network using load balancing schemes [6].

Several existing works tackled the load-balancing problem using clustering techniques [7][8][9][10]. Hierarchical algorithms such as clustering have limitations on the distance between the nodes in the same cluster. Moreover, some of the existing clustering algorithms are complex and need many iterations for electing the cluster heads. Therefore, there is a need for more suitable solution than clustering for non uniform nodes' deployment and large scale LLNs. Other Algorithms use new routing metrics [11][12] to extend the lifetime of the network. Other approaches that use RPL perform load balancing by using modified routing metrics [13][14][15][16] or using mobile sink nodes [17][18] which is not suitable for all applications. In contrast, to the best of our knowledge, we handle this problem with a different approach that is not presented in previous related work. We integrate a minimum degree spanning tree algorithm with RPL to solve such a problem.

A minimum degree minimum weight spanning tree algorithm is presented in [19]. However, it is not compatible with RPL and it uses a large number of control messages that will increase the power consumption which we need to improve. Therefore, the algorithm presented in [19] needs to be re-designed to be compatible with RPL and use less number of control message.

In this paper, we propose the Minimum Degree RPL (MD-RPL) algorithm which forms a load balanced minimum degree spanning tree.MD-RPL modifies the original tree formed by RPL to decrease its degree and enhance RPL load balancing capabilities.

The MD-RPL control messages are incorporated inside the original messages defined in the legacy RPL. Moreover, MD-RPL uses less number of control messages compared to the large number of messages used in prior minimum degree spanning tree algorithms [19]. MD-RPL is suitable for different applications such as rescue application where sensor nodes are

978-1-5090-0724-0/16/$31.00 ©2016 IEEE

distributed randomly. It takes into consideration that the position of nodes relative to the sink node is not highly affected. Moreover, it achieves more fairness between users in a load balanced network. It combines the advantages of both legacy RPL and the minimum degree spanning tree algorithm.

The MD-RPL scheme is evaluated in random topologies with different number of nodes to show its performance advantages. We test our results using COOJA simulator. Our results show a significant improvement of up to 15.6% in the maximum power consumption in the network which implies an improvement in the lifetime of the network. Moreover, MD-RPL achieves reduction in the variance in the average consumed power up to 65%.

The rest of this paper is organized as follows: Section 2 discusses the system model, the RPL preliminaries, and an overview of the distributed approximation algorithm for the Minimum Degree Minimum Weight Spanning Trees [19]. In section 3, we present the Minimum Degree RPL algorithm (MD-RPL). The performance evaluation of the MD-RPL is presented Section 4. Section 5 discusses the existing literature that handled the problem of load balancing in 6LoWPAN. We conclude the paper in section 6.

II. PERLIMINIARIES AND MOTIVATION

A. Overview of RPL

RPL has been introduced by the Internet Engineering Task Force (IETF) as the standard Routing protocol for Low Power and Lossy Networks (LLN). It is a distance vector routing protocol which builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted towards the DODAG root. RPL aims to minimize the cost of reaching the sink from any node in the network using an objective function (OF). RPL is optimized to support the many-to-one traffic. However, it provides a set of mechanisms to support Point-to-Point (P2P) traffic and Point-to-Multipoint (P2MP) traffic. It uses the Internet Control Message Protocol version 6 (ICMPV6) control messages to detect the neighboring routers and to advertise RPL information within the network. The most important RPL messages are:

1. DIO (DODAG Information Object): A broadcast message sent by the root node to construct the DODAG. DIO contains general information such as the RPLInstanceID, rank, DODAGID, …, etc.
2. DIS (DODAG Information Solicitation): This message is sent by new nodes to join the DODAG and to discover neighboring routers.
3. DAO (Destination Advertisement Object): It is a message used to build downward routes from the DODAG root to the other nodes.

1) DODAG Construction Overview

First, the DODAG root initializes the DODAG construction by sending the first DIO broadcast message (with trickle timer) and advertises itself as a parent node for other nodes in its neighborhood. When a node receives this DIO message, it uses the information in it to choose its parent set and compute its rank (its relative position with respect to the root). Choosing the preferred parent (for upward traffic towards the sink) depends on the used OF (e.g. Expected Transmission Count (ETX)). After that, the node generates a new DIO message and broadcasts it until the tree is formed and each node knows its parent through which it reaches the sink node. When a node chooses its preferred parent, it sends a DAO message to its parent to be used for downward routes. When the parent node receives this DAO message, it sends a DAO message to its parent until it reaches the root node as shown in Fig. 1. The nodes receiving the DAO will record the address of the sender in their routing table as the next hop to reach the destination address.

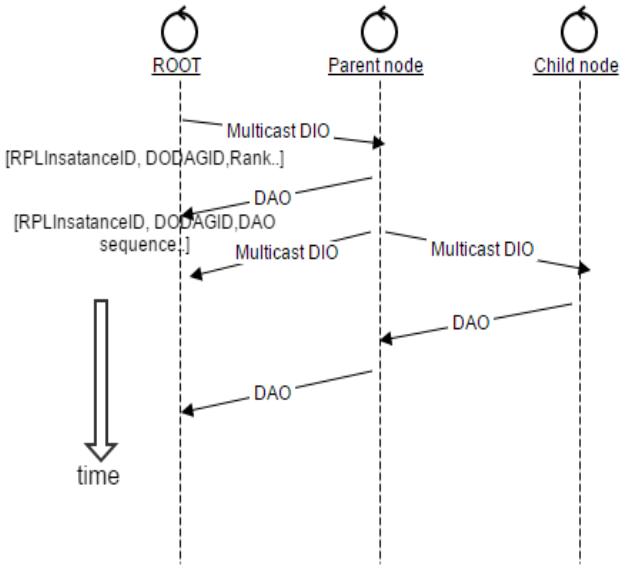

Fig. 1. A Timeline that shows RPL messages

A node can join the DODAG using DIO or DIS messages. If a node does not receive DIO message from its neighbors until the timer of the DIS message expires, the node sends a multicast DIS message to its neighbors to solicit joining the DODAG. When a neighbor node receives the DIS message, it resets the DIO timer to send the DIO message faster. RPL updates the tree according to an objective function by sending DIO messages periodically.

The trickle time *r* (used for the transmission of the DIO) slows down the transmission of the DIO every time it receives a "consistent" message (e.g. implying no change in the topology), and increases the rate when there is a change (e.g. due to a loss of parent or a change in the rank) by resetting the timer.

B. Motivation

RPL in its original setup builds the routing tree according to a routing metric which does not take into consideration the number of connected nodes per each tree node. If a battery-operated node in the 6LoWPAN is heavily loaded, this will cause the node to fail fast. To solve such a problem, we

consider an approach that uses a minimum degree spanning tree algorithm integrated with RPL. A minimum degree minimum weight spanning tree (MDMWST) algorithm is presented in [19]. However, it is not suitable for 6LoWPANs as it uses a large number of control messages which will affect the average power consumed by the sensor nodes. Moreover, the algorithm presented in [19] is not compatible with RPL, and hence, it needs to be redesigned in order to use common RPL functions and messages to form the minimum degree spanning tree. For instance, we need to integrate the degree of the node with the routing metric used by RPL. Thus motivated, we aim at developing our own Minimum Degree RPL (MD-RPL) algorithm that only uses the basic minimum degree spanning tree idea presented in [19]. The following subsection provides an overview of the MDMWST algorithm presented in [19].

C. Distributed Approximation Algorithm for the Minimum Degree Minimum Weight Spanning Trees

The algorithm of [19] aims at finding a MDMWST. Assume that the algorithm starts with any minimum weight spanning tree (MWST) *T* of a given graph *G*. To perform load balancing, the degree of the tree *T* is to be minimized. The degree of a tree is defined as the maximum of degrees of the different nodes in the tree where the degree of a node is the number of edges connected to this node. To minimize the degree of *T*, the degrees of the heavily-loaded nodes should be minimized by replacing some of their edges by alternative edges. The algorithm is divided into four phases.

1) Initialization Phase

In this phase, the maximum degree (the maximum number of nodes connected to one node) in the tree *T* is determined. To perform this, each node broadcast its degree then the node wins the election if its degree is the maximum one. For each `edge (p,r)` connected to the node of maximum degree, the node of maximum degree sends a message called "`Side`" (corresponding to the `edge (p,r)` to each node *x* in the tree *T*. The objective of this message is to let each node *x* knows if it belongs to either the subtree of the node p or the subtree of node r in the tree *T* excluding the `edge (p,r)`.

2) Search-Edge Phase

When an `edge (p,r)` connected to the node of maximum degree ends its initialization phase, it begins a search for an alternative edge (that does not already exist in T) to be exchanged with itself to decrease the degree of the node of maximum degree by one. This alternative edge should satisfy a set of conditions. First, both the degrees of the nodes of the alternative edge are less than the maximum degree minus 2. Second, the weight of the alternative edge is equal to that of the to-be-removed edge. Third, the alternative edge exist in the graph *G* but does not exist in *T*. Fourth, one of the nodes of the alternative edge belongs to the subtree of the node *p* and while the other node belongs to the subtree of node *r*. The fourth condition guarantees that the tree remains connected after removing the `edge (p,r)`. If such an appropriate alternative edge is found, the node of maximum degree is informed then it begins the following phase. On the other hand if it is not found, the algorithm terminates.

3) Edge-Election Phase

When a node of maximum degree reaches this phase, it means that there is an appropriate swap for one of its edges with an alternative edge. However, there might be another node of maximum degree that reached the Edge-Election phase. However, only one node should be elected to perform the swap. Therefore, each node that reached this phase forwards its identity in *T* then an election process will start. The node with minimal identity will win the election.

4) Edge-Exchange Phase

The winner node in the previous phase will execute the edge swap. The tree T will be updated when the old edge (e.g. `edge (p,r)`) is removed and the alternative one is connected. When this phase ends, a new round will initiate.

III. MINIMUM DEGREE RPL (MD-RPL) ALGORITHM

Minimum Degree RPL (MD-RPL) algorithm aims at maximizing the lifetime of the 6loWPAN based networks. MD-RPL algorithm integrates RPL and a minimum degree spanning tree algorithm to perform load balancing. The MD-RPL algorithm attempts to minimize the degree of the initial tree formed by RPL taking into consideration the routing metric used and the rank of its nodes. This is achieved by first identifying the most heavily loaded nodes in the tree formed by RPL, and then performing a swap of some of their edges with new suitable alternative edges.

Integrating RPL and the minimum degree spanning tree algorithm results in many challenges such as minimizing the number of control messages to form the minimum degree tree to save the power and the required memory as the memory of sensor nodes is a challenge as well. Another challenge in our design is to avoid collisions between control messages. Moreover, the compatibility with RPL is a challenge. Our algorithm achieves an improvement in the maximum power consumed. Subsequently, it will significantly increase the lifetime of the wireless sensor network. The design goals of the proposed MD-RPL algorithm are as follow:

1. MD-RPL works on minimizing the degree of the tree as much as possible to perform load balancing, and hence, maximize the lifetime of the network.
2. MD-RPL algorithm is compatible with RPL. Consequently, the degree adjustment is applied through RPL control messages. Also, the degree adjustment takes into account the routing metric already used to build the initial RPL tree.
3. Unlike [19], the MD-RPL algorithm takes into consideration the number of hops towards the sink in the network in forming the minimum degree spanning tree.
4. Further minimization of the number of control messages used for degree adjustment is a target to save energy.

A. System Model

We consider a wireless sensor network represented by a graph *G* with the number of sensor nodes *N* rooted towards the sink node. All sensor nodes are of the same type. All nodes in the network are static. ETX is the objective function that legacy RPL uses to build its tree. The sink node (the root node) is the

only receiver node while other nodes are acting as forwarders and traffic sources. All nodes (except the root node) are powered by batteries while the root node is mains-operated.

B. Minimum Degree RPL Overview

MD-RPL is applied after the initial tree formed by the original RPL is stabilized. To ensure the stabilization of the tree, we wait until the period of sending DIO increases to a relatively large value.

MD-RPL is composed of three phases: 1) the maximum degree node identification phase, 2) the initialization phase, and 3) the search phase. Messages in these phases are sent through the modified RPL control messages. They are sent through the field options of DIO message. MD-RPL schemes uses the variables already defined in DIO message (e.g. IP addresses, ETX, rank, ...). Moreover, It adds new variables used for the degree adjustment phases (e.g. the maximum degree, depth of the nodes, etc). DAO messages are also used to identify the children of each node. In what follows, we present the technical rationale behind the phases of the MD-RPL algorithm and how they manage to solve the aforementioned problems we identified in [19] that make it inapplicable to RPL. Algorithm 1 summarizes the proposed MD-RPL algorithm.

1) Maximum Degree Node Identification Phase

In this phase, the root node searches for node of maximum degree and identifies its path. First, the root node solicits the degree of each node. After that, each node sends a unicast message to its parent (containing the maximum of its degree and its children degrees) and so on until it reaches the root node. Note that in [19] each node broadcasts its degree to all its neighbors to know the node of maximum degree. Therefore, our algorithm uses significantly less number of control messages and consumes less power to determine the node of maximum degree compared to the algorithm presented in [19].

2) Initialization Phase

After the maximum degree is identified, the root node informs the node of maximum degree (only one node that perform the operation unlike [19]) to begin searching for an alternative edge instead of one of its edges to decrease its degree. The node of maximum degree chooses one of its edges to be removed and exchanged with another edge. The algorithm chooses the worst edge from the RPL OF (ETX) point of view. While in [19] more messages are sent because all edges of the node of the maximum degree perform the search for alternative edges then elect the appropriate swap. Moreover, Many messages are sent in [19] to all nodes in the tree before searching for an alternative edge. To exchange an `edge (p,r)` in [19], many messages are sent to divide the nodes in the network into two groups. One group is for the nodes that belong to the subtree of node `p`. While the other gorup is for the nodes that belong to the subtree of node `r`. Choosing a new edge whose nodes belong to different subtrees of the nodes of the to-be-removed edge ensures that the tree remains connected after the swap occurs. In our algorithm, these messages are removed and altered by checking this condition locally using significantly less number of messages while searching for an alternative edge in the following phase. To check if a node

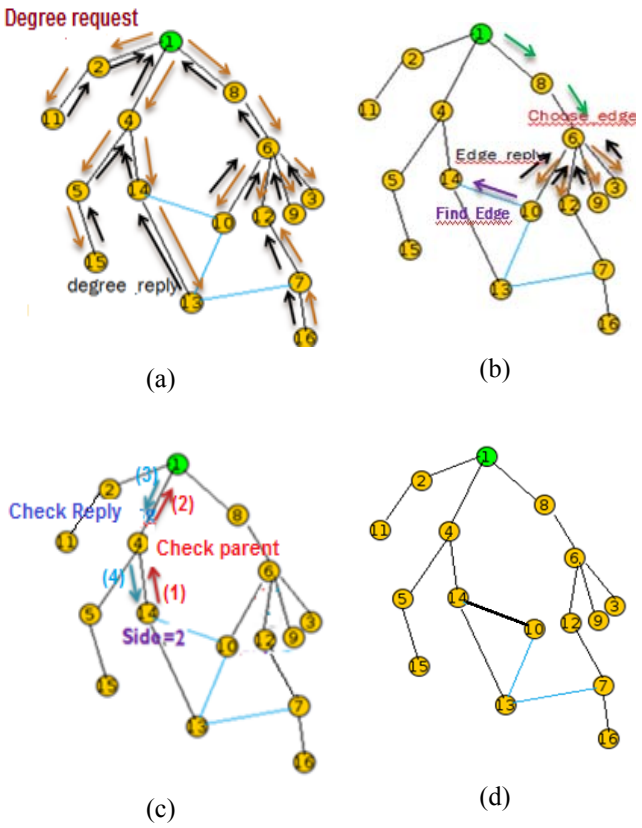

Fig. 2. Messages sent in different phases

belongs to the subtree of the node of maximum degree or not during the search phase, this node sends a message to its parent. After that, the parent node forwards this message to its parent and so on until it reaches either the node of maximum degree (the node belongs to its subtree) or the root node (it does not belong to the subtree of the node of maximum degree). Subsequently, the tree will remain connected without need to define this information for all nodes during initialization phase.

3) Search Phase

In this phase, the MD-RPL algorithm searches for an alternative edge to be exchanged with the chosen edge in the previous phase. This alternative edge should satisfy set of condition. First, both the degrees of the nodes of the alternative edge should be less than the maximum degree minus 2. Second, the ETX of the alternative edge is less than the ETX of the to-be-removed edge plus a threshold. Unlike [19], the weight of the alternative edge is not necessarily equal to the weight of the node of maximum degree that allows flexibility in choosing the alternative edge. Third, the new rank of the removed node is less than or equal to its old rank plus 3. The third condition ensures the efficiency of the network as we do not allow to increase the difference between number of hops in the initial tree (formed by legacy RPL) and the new tree (formed by MD-RPL) more than three hops. Fourth, one of the

nodes of the alternative edge belongs to the subtree of the node of maximum degree while the other node does not to keep the tree connected. Fifth, we put limitations on the node that makes the search not to be apart from the node of maximum degree by more than three levels. Sixth, this edge is already not connected in the present tree. If such an appropriate edge is found, the tree will be updated. Some nodes will need to update their parents and others will update their ranks according to this edge swap in the tree. After that, the algorithm goes again to the maximum degree node identification phase as long as number of iteration does not exceed 7 iterations. On the other hand, if there is no an appropriate edge to be exchanged with any of the edges of the node of maximum degree, then the algorithm terminates. Loops are avoided in the MD-RPL algorithm because all control messages are exchanged from a child to its parent or from a parent to its children and neighbours. We send messages to neighbours only while searching for a new edge. The neighbour that belongs to a different subtree is the only allowed neighbour for the edge swap. Moreover, limiting the number of iterations in MD-RPL ensures the termination of the process, and hence ensures the stability of the protocol.

C. Illustrative Example

Fig. 2 describes the messages sent in different phases for the initial tree shown in the Fig. 2. Fig. 2a shows the messages sent in the maximum node identification phase where the root node initiates this phase by sending a message called "`Degree Request`" then each node replies by the maximum of its degree and its children degrees in a message called "`Degree Reply`". Finally, the root node knows the maximum degree and the path to the node of maximum degree (node 6). After that, the root node sends a message to the node of maximum degree (node 6) to initiate the search process as show in Fig. 2b. The node of maximum degree (node 6) chooses one of its edges to be exchanged by sending a message called "`Choose Edge`" to its children then each child replies by the message "`Edge Reply`" containing the ETX value of its edge.

After node 6 chooses the to-be-removed edge (the edge with the worst ETX let it be `(6,10)`), node 10 starts a search for an appropriate neighbor to be connected to it instead of node 6 by sending a message called "`Find Edge`" to its neighbors (e.g. it sends the message first to node 14). All conditions mentioned in the search phase are satisfied for the `edge (10,14)`. First: both degrees of node 10 (degree=1) and node 14 (degree=2) is less than the maximum degree (maximum degree=5) minus 2. Second: the ETX of the `edge (10,14)` is the best ETX among the other possible alternative edges which can be connected to node 10. Third: when the `edge (10,14)` is connected, the rank of node 10 will be 3 instead of 4 (better rank). Fourth: to check if node 14 does not belong to the subtree of node 6, it sends a message called "`Check Parent`" to its parent as shown in Fig. 2c. The parent node forwards it to its parent and so on until it reaches the root node without passing by node 6. Therefore, node 14 does not belong to the subtree of node 6 while node 10 does. Fifth, node 10 is apart from node 6 (the node of maximum degree) by only one level (direct child). Sixth, `edge (10,14)` is not connected in the initial tree. If node 10 finds among its neighbors more than one appropriate edge, it chooses the one with the best ETX. Assuming `edge (10,14)` is chosen, node 10 then updates its parent as shown in Fig. 2d. After the tree is updated, go again to the maximum node identification phase and repeat the operation unless number of iterations exceeds 7 iterations.

Algorithm 1. MD-RPL Pseudo Code

```
if (Legacy RPL Tree is STABLE){
  Begin MD-RPL Algorithm
  switch (MD-RPL phase){
    case (Maximum Degree Node Identification Phase):
      if (MD-RPL message == Degree_Request){
        Maximum_Degree = Node_Degree;
        if(!Leaf_node)
          Send_DIO (Degree_Request ,All children)
          //Forward the message to the node's children;
        else{
          MD-RPL message = Degree_Reply;
          Send_DIO (Degree_Reply ,Parent node)//containing the
Node_Degree } }
      else if (MD-RPL message == Degree_Reply){
        Maximum_Degree = max (Node_Degree , Children degrees)
        Set the variable "VIA" with the IP of the node of max degree;
        Forward to the parents until reach the root
        if (ROOT Node)
          MD-RPL phase = Initialization phase;
          MD-RPL message= Start_search;
          Send_DIO (Start_search, VIA);//contain the final maximum
degree calculated } }
    break;
    case (Initialization Phase):
      if (MD-RPL message == Start){
        if (Node_Degrree != Maximum_Degree)
          Send_DIO (Start, VIA)
        else{    //reach the node of max degree
          Chosen Edge to be removed = The edge of the worst ETX value;
          MD-RPL phase = Search phase;
          Send_DIO (intiate_search, the other node of the chosen edge } }
    break;
    case (Search Phase)
      if (MD-RPL message = initiate_search){
        if (relative depth of nodes<=3) //relative depth is the position of nodes
relative to the node of maximum degree
          Send_DIO (Find_Alternative_Edge, all neighbours);
        else if (MD-RPL message == Find_Alternative_Edge)
          if (Node_Degree < Maximum_Degree-2 &&
  ETX of this edge is the best among the other possible alternative edge &&
```

```
 New rank of the chosen node <= Old rank +3 && This edge does not exist
in the old tree && The node is not in the subtree of the node of maximum
degree)
            Send_DIO (Found_Edge, Parent Node)
          else
            Send_DIO (Not_Found_Edge, Parent Node)}
         else if (MD-RPL message == Found_Edge){
          Update the tree;
          Repeat for the next node of maximum degree; }
        else if(MD-RPL message == Not_Found_Edge)// from all neighbours
          Send_DIO (initiate_search,one of the children)
       break;}}
```

IV. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed MD-RPL in different settings/topologies using Conitiki 2.6 operating system [20] and the Cooja simulator [21]. We use the same configuration and type for all nodes. All nodes are selected to be of type "sky motes". The used sky mote nodes have CPU frequency equal to 2.688 MHz, transmission range equal to 50 m and mote operational voltage equal to 3V. The generated traffic rate is 1/60 packets/sec while simulation time is equal to 2 hours. We evaluate the performance of the MD-RPL scheme in different random topologies considering different number of nodes (20, 25, 30, 35, 40 and 45 nodes). For a given number of nodes, we generate 10 different random topologies and the reported results are the average of these topologies. For each random topology, the percentage of time the radio is on (as indication of consumed power) is calculated for all nodes in two cases: legacy RPL and MD-RPL. Moreover, the maximum value of the consumed power (maximum percentage of time the radio is on) and the average power of all nodes are calculated for each random topology then we take the average of maximum power and the average power of the ten random topologies in case of legacy RPL and the MD-RPL and compare them. Power consumption results are calculated based on the Cooja PowerTracker tool [22]. The PowerTracker tool represents the percentage of time spent by each node being on, in transmission state and in reception state.

We compare between Legacy RPL and MD-RPL using three metrics: the maximum consumed power, the average consumed power, and the variance in the consumed power.

A. Maximum Consumed Power

As shown in Figure 3, the maximum power in case of MD-RPL is consistently better than that of legacy RPL. This implies an improvement in the network lifetime in all cases. In addition, the gain in the average of maximum power increases as number of nodes increases in average as shown in Figure 4. It increases up to 15.6%. MD-RPL minimizes the degrees of the heavily loaded nodes. The heavily loaded nodes consume more power than the other nodes as it serves a large number of children. Therefore, they consume more power in transmission and reception. This causes them to fail fast. Decreasing the degree of these nodes by removing some of their children decrease the maximum power consumed and subsequently improves the lifetime of the network. Therefore, MD-RPL tends to perform better as the number of nodes increases.

B. Average Consumed Power

As shown in Figure 5, the average power of MD-RPL is better than legacy RPL. For instance, at 40 nodes, the average power of nodes running our MD-RPL is 3% less than that of legacy RPL. When MD-RPL decreases the degrees of the heavily loaded nodes, it will load balance the network and decrease the average power consumed by nodes. Thus, MD-RPL gives better performance the legacy RPL from the power consumption and network lifetime point of view.

C. Variance in the Consumed Power

The variance of the power of all nodes became closer to the mean as number of nodes increases as shown in Figure 6. The reduction in variance reaches the value 65% as shown in Figure 7. The smaller variance is considered a good indication of the ability to load balance the network as nodes tend to be utilized on the same level. Moreover, it indicates more fairness between users in traffic distribution.

V. RELATED WORK

Due to limitation of sensor node's battery, there is a challenge in 6LoWPAN to save power of nodes. Therefore, many researches aims at load balancing the 6LoWPAN to minimize power consumption and extend of 6LoWPAN. The algorithm used in [23] aims at balancing the nodal lifetime to prolong the network lifetime. It is a lifetime balanced data aggregation algorithm that adjusts the aggregation holding time between two neighboring nodes dynamically. A mobile data collector is used in [18] to perform load balancing. The algorithm divides the network into four logical partitions. Scheduling the movement of the mobile data collector depends on the residual energy and the density of each partition. A load balancing algorithm is presented in [15]. The routing protocol used is RPL. It defines a new OF that takes into consideration the number of packets sent by a node either generated of forwarded. A node chooses its preferred parent based on two metrics: the number of transmitted packets and ETX metric. If the difference between the ETX values of two parents is relatively small, a node chooses the one with the least traffic. In LB_RPL [14], the traffic is distributed among different parent nodes (not only one parent node as Legacy RPL). The parent set is arranged according to the buffer utilization. This is achieved by setting the timer of DIO proportional to the traffic sent in the last period. Subsequently, the traffic is distributed among the k parents (with lower traffic in the parent set) relative to the link quality. Therefore, the parent node with high buffer utilization in one period will be avoided by all or many of its children in the following period. An Energy balancing algorithm is presented in [16]. It aims at maximizing the lifetime of the bottleneck nodes. Expected life time is the metric that RPL uses to choose the preferred parent. The calculation of the expected lifetime of a node depends on its

generated traffic, the forwarding traffic from its children, the ETX metric of its link to the preferred parent node, the data rate, its occupation ratio and the residual energy. When a new node joins the DODAG, it compares its lifetime with the updated lifetime of the bottleneck node. The bottleneck node of a certain path is calculated as the node minimum expected lifetime. The preferred parent is the parent with largest bottleneck lifetime in its path.

Many researches develop load balancing schemes using clustering [7][8][9][10] . In [7], the network is divided into a number of clusters with a high-energy gateway in each cluster. The gateway first broadcasts a message to know the sensor nodes in its communication range then it adds the nodes to its cluster based on the load on this gateway and the communication cost. The communication cost depends on the energy dissipated to transfer data from a node to the gateway. In [9], a secure load balancing protocol is used with data aggregation. It aims at balancing clusters from a security point of view. Security is achieved by using shared keys between sensor nodes and cluster heads. Load balancing is achieved by using pseudo sinks with high storage capability then has the sensor nodes in congested clusters to send their data to pseudo sinks or neighboring cluster heads in the lightly loaded clusters. Therefore, data aggregation is performed at both cluster heads and pseudo sinks. An algorithm of load balancing with multi-hop clustering is used in [10]. This algorithm uses two layers, one for inner cluster communication and another for intra-cluster communication.

Some approaches use clustering techniques to solve the problem of load balancing [7][8][9][10]. Other approaches change the routing metrics to achieve load balancing [11][13][14][15][16]. However, none of the prior works targets load balancing using the minimum degree spanning tree algorithms in the networks that use RPL routing. In this paper we propose to incorporate load balancing with RPL in 6loWPAN based networks through applying a minimum degree spanning tree algorithm.

VI. Conclusion

In this paper, we have proposed the Minimum Degree RPL scheme which forms a load balanced minimum degree spanning tree to enhance RPL load balancing capabilities. Our simulation results indicates a significant improvement in the maximum power consumption up to 15.6%. Thus, MD-RPL increases the lifetime of the wireless sensor networks. Moreover, MD-RPL reduces the average power consumed by all nodes and minimizes the variance in the average consumed power up to 65%. The reduction in the variance indicates more fairness between nodes. Therefore the MD-RPL gives better performance the Legacy RPL from the power consumption and network lifetime point of view. In future, we plan to study the local repair of the network in case of the MD-RPL.

References

[1] IETF WG, "Routing Over Low Power and Lossy Networks." Available online: http://tools.ietf.org/wg/roll.

[2] T. Winter and P. Thubert, “RPL: IPv6 Routing Protocol for Low powerand Lossy Networks, RFC 6550,” IETF ROLL working group, Mar.2012.

[3] D. Culler, D. Estrin, and M. Srivastava, “Guest Editors’ Introduction: Overview of sensor network,” Computer, vol. 37 Issue: 8, pp. 41-49, Aug. 2004

[4] N. Accettura, L. A. Grieco, G. Boggia, and P. Camarda, "Performance analysis of the RPL routing protocol," pp. 767–772, Apr. 2015.

[5] C. Guo, L. C. Zhong, and J. M. Rabaey, "Low power distributed MAC for ad hoc sensor radio networks," vol. 5, pp. 2944–5, 2001

[6] S. Mor and M. V. Saroha, " Load balancing in wireless sensor networks," pp. 116-119, 2013.

[7] G. Gupta and M. Younis, "Performance evaluation of load-balanced clustering of wireless sensor networks," In Telecommunications, 2003. ICT 2003. 10th international conference on IEEE , Vol. 2, pp. 1577-1583.

[8] N. Israr and I. U. Awan , "Multihop clustering algorithm for load balancing in wireless sensor networks", 2007.

[9] S. Ozdemir, "Secure load balancing via hierarchical data aggregation in heterogeneous sensor networks," J. Inf. Sci. Eng., vol. 25, no. 6, pp. 1691-1705, 2009.

[10] H. Zhang, L. Li, X. Yan, and X. Li, "A load-balancing clustering algorithm of WSN for data gathering," pp. 915–918, Aug. 2010.

[11] A. Dehghani, K.Jamshidi, S. Mirshams and K. RahimiZadeh , "A routing metric for load balancing considering energy and delay constraints in wireless sensor network," In Telecommunications, 2008. IST 2008. International Symposium on IEEE, pp. 745-750, Aug. 2008.

[12] J. Zhu et al. "An energy balanced reliable routing metric in wsns," Wireless Sensor Network, 2009.

[13] H.-S. Kim, J. Paek, and S. Bahk, “QU-RPL: Queue Utilization based RPL for load balancing in large scale industrial applications,” in Proc. IEEE SECON’15, Jun. 2015.

[14] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, “Load balanced routing for low power and lossy networks,” in WCNC’ 13, Shangai, China, April 2013.

[15] T. B. Oliveira, P. H. Gomes, D. G. Gomes and B. Krishnamachari, "ALABAMO: A LoAd BAlancing MOdel for RPL".

[16] O. Iova, F. Theoleyre and T. Noel, "Improving the network lifetime with energy-balancing routing: Application to RPL," In Wireless and Mobile Networking Conference (WMNC), 2014, pp. 1-8, May 2014.

[17] O.Gaddour,A.Koubâa, R. Rangarajan, O.Cheikhrouhou, E.Tovar, M. Abid, "Co-RPL: RPL routing for mobile low power wireless sensor networks using Corona mechanism." Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on. IEEE, 2014.

[18] A. Patooghy, M. Kamarei, A. Farajzadeh, F. Tavakoli, and M. Saeidmanesh , "Load-Balancing enhancement by a mobile data collector in wireless sensor networks," in International Conference on Sensing Technology (ICST), Liverpool, UK, pp. 634-638, 2014.

[19] C. Lavault and M. Valencio-Pabon, "A distributed approximation algorithm for the minimum degree minimum weight spanning trees," Laboratoire d’Informatique de Paris-Nord, 2007.

[20] N. Tsiftes, J. Eriksson, and A. Dunkels, “Low-Power Wireless IPv6 Routing with ContikiRPL,” April 2010.

[21] J. Eriksson, F. Osterlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, ¨ R. Sauter, and P. Marron, “COOJA/MSPSim: interoperability testing ´ for wireless sensor networks,” in Proceedings of the 2nd International Conference on Simulation Tools and Techniques. ICST, 2009, p. 27.

[22] L. GuangDi and T. Ling, "The Test of Contiki-based Protocol Stack of 6LoWPAN,"2015.

[23] Z. Li, Y. Peng, D. Qiao, and W. Zhang, “LBA: lifetime balanced data aggregation in low duty cycle sensor networks,” in Proc. IEEE ,2012.

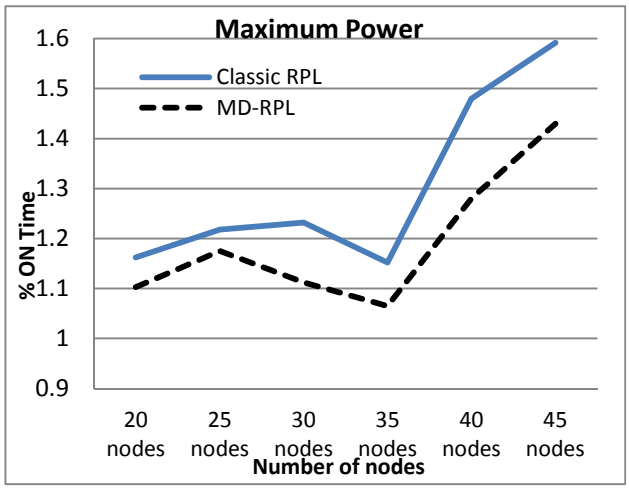

Fig 3. The average of maximum power of all nodes in different cases

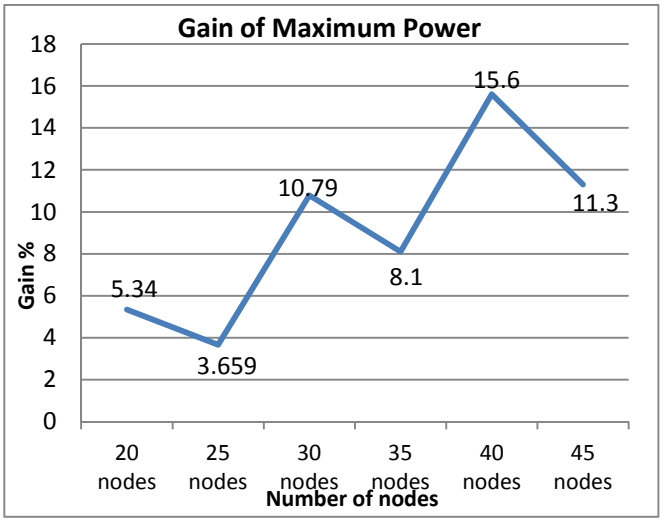

Fig 4. The gain in the average of maximum power in different cases

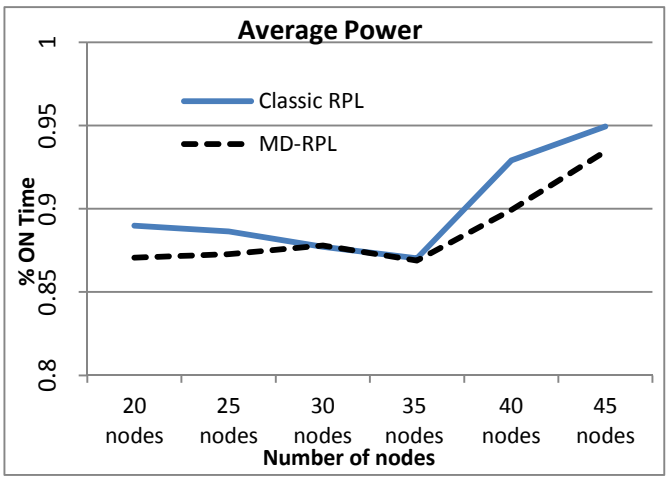

Fig 5. The average of average power of all nodes in different cases

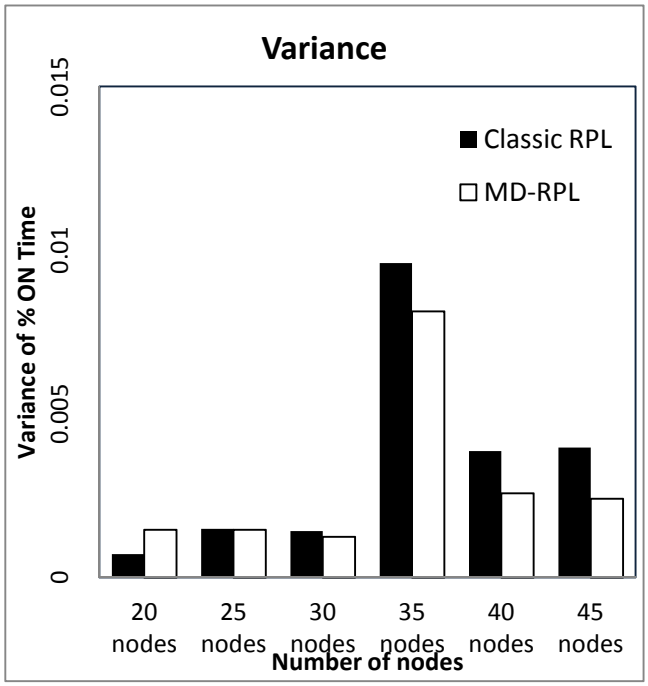

Figure 6. Variance in power for different cases

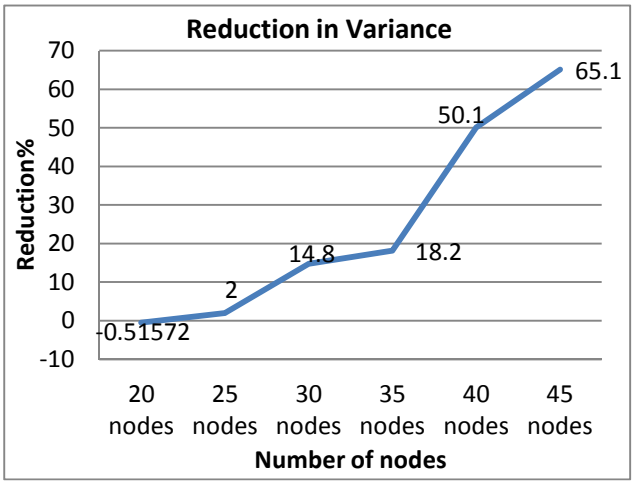

Figure 7. The reduction in variance for different cases