A New Load-Balancing Aware Objective Function for RPL's IoT Networks

Baraq Ghaleb, Ahmed Al-Dubai, Elias Ekonomou, *Wajeb Gharibi, **Lewis Mackenzie and ***Mustafa Bani Khalaf
School of Computing, Edinburgh Napier University,10 Colinton Road, EH10 5D, Edinburgh, UK, Emai:{B.Ghaleb, A.Al-Dubai, E. Ekonomou}@napier.ac.uk *College of Computer Science & Information Systems, Jazan University, Saudi Arabia, Email: gharibi@jazanu.edu.sa, **Dept. Computing Science, University of Glasgow, Email: Lewis.Mackenzie@glasgow.ac.uk mbanikhalaf@yu.edu.jo ***Network and Information Security Department, Yarmouk University, Irbid, Jordan.

***Abstract*— The IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) has been recently standardized as the de facto solution for routing in the context of the emerging Internet of Things (IoT) paradigm. RPL, along with other standards, has provided a baseline framework for IoT that has helped advance communications in the world of embedded resource-constrained networks. However, RPL still suffers from issues that may limit its efficiency such as the absence of an efficient load-balancing primitive. In this study, we show how RPL suffers from a load-balancing problem that may harm both the reliability of the protocol and its network lifetime. To address this problem, a novel load-balancing scheme is introduced that significantly enhances the reliability of RPL and fosters the protocol's efficiency in terms of power consumption.**

***Keywords*— Internet of Things, Low-power and Lossy Networks, RPL, Trickle Algorithm, Routing Maintenance.**

I. INTRODUCTION

Recently, the tight integration of the physical world and computing has given birth to a new communication paradigm referred to as the Internet of Things (IoT). One of the building blocks of IoT is the Low-power and Lossy Network (LLN), a collection of interconnected embedded devices, such as sensor nodes, typically characterized by constraints on both node resources and underlying communication technologies [1][2]. Node constraints may include restrictions on available power, processing and storage, while the communication system is often subject to high packet loss, frame size limitations, low data rates, short communication ranges and dynamically changing network topologies [3][4]. Such limitations make it difficult to find efficient routing solutions and primitives for LLNs, a task made still more arduous by the potential size of some of these networks which can comprise up to thousands of nodes [4].

As a major enabling component of IoT systems, LLNs have attracted much attention from industry, academia and standards bodies, with the goal of developing routing solutions that guarantee efficient use of limited network resources. In 2009, as a part of the Internet Engineering Task Force (IETF) efforts to design routing solutions for LLNs, it was suggested that the conventional ad hoc routing protocols, such as AODV [5], are inefficient to satisfy the LLNs unique requirements [6]. Consequently, a multitude of routing solutions and primitives targeting LLNs specifically have come into existence including e.g. CTP [7], and Hydro [8]. These attempts then led to the standardization by IETF of the IPv6 Routing Protocol for Low power and Lossy Networks (RPL). However, many studies have reported that RPL still suffers from issues that may harm the reliability and energy efficiency of the network [4][9][10][11] . One of these is the absence of an efficient load-balancing mechanism that can ensure a fair distribution of network traffic among network nodes [12][13]. Once a preferred parent towards the network root has been selected, all traffic is forwarded through this parent, as long as it is reachable, without any attempt to perform load-balancing [3][14]. This behavior can have a detrimental effect in two ways: firstly, it may drain the power of overloaded nodes leading to network disconnections [12]; and, secondly, it may lead to higher packet losses due to congestion around overloaded nodes [13][15]. It is also evident as reported in [16] that the majority of the proposed load-balancing mechanisms suffer from instability problem due to the continuous switching of the nodes' preferred parent to achieve the load balancing. An issue that may jeopardize the conceived benefits of introducing load-balancing mechanisms. Thus, an efficient load-balancing mechanism does not just mean providing the protocol with the capacity to distribute the traffic among respective nodes, but also ensures network stability.

In this article, we aim at developing a more efficient objective function that alleviates those problems, featuring the following primary characteristics: i) capacity to load-balance the traffic in a way that fosters network reliability and lifetime, ii) capacity to rule out all unnecessary preferred parent switches as a way to preserve network stability.

The contribution of this article is four-fold:

1) a new routing primitive, based on data-plane messages, to calculate the number of children;
2) a new load-balancing mechanism, based on the lexical combination of the primary routing metric and the number of children, to distribute energy expenditure fairly among nodes;
3) a new mechanism for path selection (i.e. switching the preferred parent for load-balancing purposes) with the goal of eliminating the instability problem that may undermine load balancing;
4) a new efficient mechanism for propagating routing information, to provide up-to-date routing information at minimum overhead.

The remainder of this paper is organized as follows. Section II introduces a brief overview of the RPL routing protocol. In section III, we discuss the problem of load balancing in RPL, highlighting the consequences that may arise from the absence of such a mechanism. A literature review of related and recent work addressing the issue of load balancing in RPL is presented in section IV. The proposed load-balancing mechanism is introduced in section V. Detail of protocol evaluation and discussions is in section VI. Finally, Section VII concludes the paper and discusses future work.

II. RPL ROUTING PROTOCOL OVERVIEW

RPL [3] is basically an IPv6 proactive distance-vector routing protocol designed by the IETF community specifically

to fulfill the unique requirements of a wide gamut of applications within the context of IoT. In particular, RPL targets MultiPoint-to-Point (MP2P) applications where data is gathered by a group of sensors and then transmitted to a central location for further processing. RPL also provides support for the Point-to-Multipoint (P2MP) traffic pattern or the *Downward* traffic while the Point-to-Point (P2P) pattern is indirectly supported.

A. RPL Topology

RPL organizes its physical network into a form of *Directed Acyclic Graphs* (DAGs) where each DAG is rooted at a single destination. In RPL's terminologies, such a rooted DAG is referred to as *Destination-Oriented DAG* or simply DODAG. The DODAG root represents the final destination for the traffic within the network domain and its main role is to bridge the RPL domain with other IPv6 domains such as the Internet. In the context of LLNs, a DODAG root is referred to as LLN Border Router (LBR).

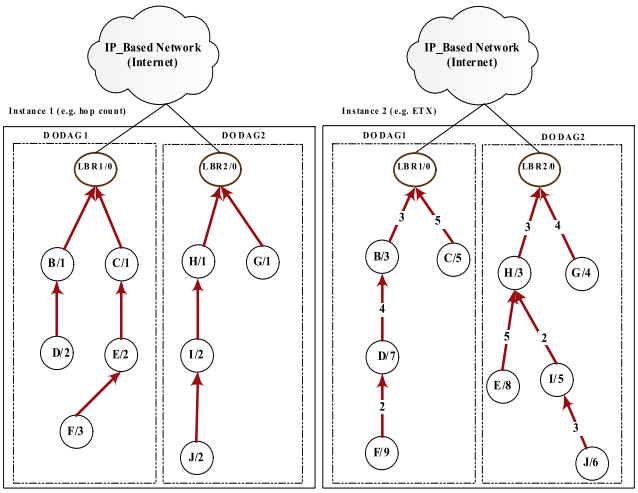

Fig. 1 RPL instances and DODAGs

The DODAG in RPL is constructed as a tree-like structure, with the exception that it permits a node to have more than one parent, enabling the existence of redundant paths. The protocol uses the term *RPL instance* to refer to multiple DODAGs sharing the same routing policies and mechanisms. Multiple RPL instances may coexist concurrently under the same physical topology and a node may join more than one instance at time. However, within each instance, a node is allowed to associate with only one root (DODAG). An example of an RPL topology comprising two instances, with two DODAGs, is illustrated in Fig. 1.

B. Objective Function

In RPL, the route selection and optimization have been decoupled from the core protocol operations to satisfy the conflicting requirements of different LLN applications. Hence, the core protocol is centered on the intersection of these requirements, whereas additional modules are designed to address application-specific objectives [18]. The term Objective Function (OF) is used to describe the set of rules and policies that governs the process of route selection and optimization, in a way that meets the different requirements of the various applications [3]. In technical terms, the objective function is used for two primary goals: first, it specifies how one or more routing metrics, such as delay, can generate a Rank value that reflects the node's relative position in the network; second, it defines how the Rank should be used for selecting the preferred next-hop (parent) to the DODAG root. Currently, two objective functions have been standardized for RPL namely, the Objective Function Zero (OF0) [14] and the Minimum Rank with Hysteresis Objective Function (MRHOF) [18]. The OF0 is designed to select the nearest nexthop to the DODAG root with no attempt to perform any load balancing. The Rank of a node R_N is calculated by adding a strictly positive scalar value (rank_increase) to the Rank of a selected preferred parent R_P according to Eq. 1 and Eq. 2 as follows.

$$R_N = R_P + \text{rank\_increase} \quad (1)$$

$$\text{rank\_increase} = (Rf * S_p + S_r) * \text{MinHopRankIncrease} \quad (2)$$

where the *step-of-rank*, S_p, represents a value related to the parent link metric and properties such as the hop-count or the expected transmission cost (ETX), while the rank factor (Rf) and *stretch_of_rank* (S*r*) are normalization factors. The OF0 does not specify which metric/metrics should be included for the calculation of rank_increase but, for example, the ContikiRPL implementation uses the hop count. Unlike OF0, the MRHOF is designed to prevent excessive churn in the network topology and will not always replace a minimum path unless a significant change in the cost has been discovered (i.e. the Rank has changed by more than a pre-defined threshold called the *Hysteresis* value).

C. RPL Operations

To facilitate the Multipoint-to-Point (MP2P) traffic pattern, a DODAG topology centered at the network root should be constructed. In such a topology, each non-root node willing to participate in the MP2P communication must select one of its neighbors to act as that node's default route (DODAG parent) towards the root of that DODAG. The term *Upward Routes* is used to describe the routes that carry the MP2P traffic in the direction of the DODAG root. The construction of the DODAG starts by having the DODAG root multicast control messages called Data Information Objects (DIOs) to its RPL's neighbors. Those messages carry the information and configuration parameters required to build the DODAG and an indication of the Objective Function that should be used [19]. Once a multicast DIO is received by a neighboring node. the node: (1) adds the sender address to its candidate parent set; (2) calculates its own rank with respect to the DODAG root based on the routing metric, advertised OF and rank of candidate parent; (3) selects its preferred parent from the candidate set; and (4) updates the received DIO with its own rank and multicasts it to other neighboring nodes. This process will continue until all nodes have setup their default routes in the upward direction towards the DODAG root.

III. Related Work

The load-distribution problem of the RPL standard in LLNs has been the subject of several studies. For instance, in [13] proposes a probability-based load-balancing solution for RPL referred to as LB-RPL. LB-RPL achieves load balancing by having each node distributes the traffic among its top *k* parents

(in terms of Rank) based on their traffic load. A parent experiencing heavy load may signal its status by delaying the broadcasting of its scheduled DIO message. This enables the child nodes to remove that parent from their top *k* and hence, exclude it from data forwarding. The implicit signaling through delayed DIO has no extra overhead, but a lost DIO might easily be misinterpreted as delayed, giving a false alarm of higher workload at some nodes [20].

The load-distribution in LLNs was also addressed in [12] which proposes a QoS-aware fuzzy logic OF referred to as *Fuzzy-Logic Objective Function* (OF-FL). OF-FL combines hop count, node energy, link quality and end-to-end delay in what was called *holistic* routing. The drawback of this OF is that several parameters must be transmitted to calculate the fuzzy values, thus requiring larger DIO messages, at increased risk of fragmentation and consequent additional overhead. In addition, fuzzy-based approaches are known to incur greater complexity than others, especially when multiple instances exist under the same RPL topology.

The authors in [20] propose a multi-path routing mechanism based on RPL allowing the protocol to forward traffic to multiple preferred parents. In this proposal, a new metric, the *Expected Lifetime metric* (ELT) is introduced, which aims to balance energy consumption among network nodes and maxmise lifetime for the bottlenecks. However, because several parameters must be exchanged (i.e. data rate, retransmission count, throughput, transmission power and residual energy) to calculate the rank, this approach, like OF-FL, requires higher overhead and bigger DIOs. Apart from these shortcomings, all the aforementioned mechanisms lack an efficient routing primitive that handles the "herding-effect" problem. They also either overreact to changes in load-balancing routing information or respond too slowly.

IV. The Problem Statement

Considering load balancing, RPL lacks an efficient load-balancing routing primitive that ensures a fair distribution of data traffic among respective nodes while minimizing overhead. This prevents the distribution of traffic over multiple paths where these exist potentially increasing data loss caused by node packet buffer overflow [13] or leading to the faster depletion of the energy of overloaded nodes [20]. This may lead to formation of holes in the network as a result of the early death of overloaded bottleneck nodes, resulting in service disruption.

However poorly implemented load balancing causes problems too. An example is the herding-effect, in which the network suffers topological instability [15] caused by sets of nodes repeatedly switching preferred parents in a futile attempt to achieve load balancing. In Fig 2a, three nodes have selected the lightly loaded preferred parent (a), upon receiving a DIO from it. However, (Fig. 2b) a new DIO from (b) then causes all three nodes simultaneously to change preferred parent to (b) which now has fewer children than (a). Upon receiving a new DIO from (a), however, the migration reverses, resulting in load oscillation with no balancing.

V. Proposed Solution

In order to address the load-balancing problem of RPL, a new load balancing OF is proposed in this study and discussed in the following subsections. The first step (section A) is to determine the most suitable metric for load-balancing information and how to measure it. The second step (section B) is to ascertain how best to propagate load balancing information in a timely efficient manner before it becomes obsolete. The third step (Section C) to combine such a load-balancing metric with other metrics to preserve other performance aspects such as reliability. Finally (Section D), a mechanism is proposed to mitigate the instability problems that may arise from load-balancing.

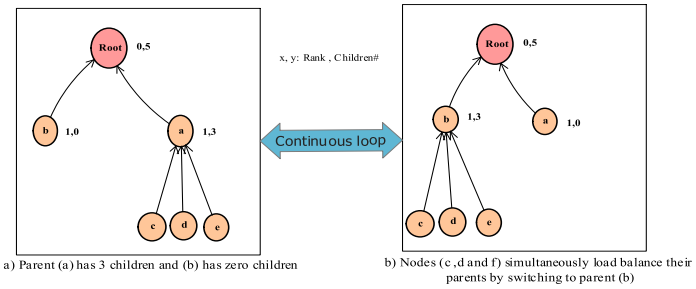

Fig. 2. Topology instability due to herding-effect problem

A. The Load-balancing Metric

Several routing metrics for achieving load balancing in RPL networks have been proposed in the literature including number of children, throughput and queue utilization factor. Each has its own pros and cons. For instance, [15] points out that the number of children metric is only updated by reception of DAO messages and timeouts of routing table entries (deletion) and thus may not reflect the actual load of the network promptly. Furthermore, DAO messages are not used in the No-Downward-Route nor the non-storing MOPs of RPL. To address this issue, our previous study in [21] proposed adding the parent address of each node to the DIO messages as an option, so as to enable the calculation of child numbers in the absence of DAO messages. A minor concern with adopting this mechanism is the extra 16-byte overhead required to carry the parent address option. In addition, as DIOs are regulated by means of a Trickle algorithm, it is not guaranteed that routing information for load-balancing purposes would be timely updated and there might be a problematic gap between the time the node has changed its preferred parent and the scheduled time for the DIO to be sent at. However, the number of children can be easily calculated based on data-plane traffic eliminating the need for DIO-based approach, especially in periodic applications (a combination of both approaches may be used in non-periodic applications or when the maximum Trickle interval for propagating DIOs is lower than the data plane traffic rate). In this study, we introduce for the first time the notion of calculating the number of children based on the data-plane messages rather than relying on DAO messages in RPL. We exploit the fact that IPv6 data packets carry the source address of the sender in the header, in addition to an RPL *hop-by-hop option*. When an IPv6 packet is received, we first determine if it is a control or data packet by looking for the RPL hop-by-hop option. If the received packet is data, we inspect its direction to decide if the sender is a child or not (direction is specified by the Down flag in the RPL hop-by-hop option). If the packet is heading upward, the sender is a child so we add the sender IP Address (*CHA*) to the list of children (*CHList*) of the receiver. If no

data packets are received from an existing child during a pre-specified interval, it is removed. We set this interval proportional to the traffic rate. The pseudocode for this process is illustrated in Algorithm2.

B. The Propagation Mechanism

In RPL, the propagation of routing information carried in DIO messages is regulated by means of the Trickle algorithm [22] in which the DIO transmission rate is increased (i.e. reset to the minimum interval) upon detecting inconsistencies in the network. The current implementation of the Trickle algorithm in Contiki OS has restricted the number of times the network is determined inconsistent to a few cases to minimize the control-plane overhead. Those cases include global repair, local repair, and parent change. Here, it is clear that a change in a node's rank does not trigger a resetting of the Trickle timer, so some routing decisions might be taken, based on obsolete routing information due to the long DIO transmission period in the consistent state [23]. To overcome this concern, we have opted to permit the node to declare an inconsistency upon detecting that its number of children has been increased or decreased by a pre-specified threshold. This will allow neighboring nodes to receive the load information in a timely manner, at the cost of some increased overhead. To reduce the overhead resulting from resetting the Trickle timer, we do not do this every time the node's balancing information changes (in terms of number of children). Instead, we opt to use a *FastPropagation Timer*: when this expires a node checks whether it should reset its Trickle timer or not, as shown in Algorithm 2. The propagation of rank information is still governed by the Trickle algorithm itself.

C. The Proposed Combined Metric and Parent Selection

The proposed OF lexically combines a primary metric (e.g. hop count or ETX) with a secondary (e.g. number of children) with the goal of building a balanced topology. The primary metric is used by a node to calculate its rank and select a set of candidate parents toward the DODAG root. Once the candidate parent set has been selected, the secondary metric is used to break ties among selected parents if there is more than one in the parent set. The details of the parent selection process is illustrated in Algorithm 1.

D. Avoiding the herding problem

One of the obstacles toward achieving load balancing in RPL is the way the protocol switches preferred parents. The common strategy adopted by RPL is to allow a specific node to switch immediately to a better parent upon detecting such a parent by means of DIOs. This may have the advantage of timely switching; however, in the context of load balancing, changing preferred parent immediately on discovery may give rise to the herding effect explained earlier.

To address this issue, we have introduced the idea of the Balancing_Timer. Here, each node performs the process of parent selection according to a regular pre-specified scheduling interval rather than immediately upon receiving a new DIO. However, we exclude the first received DIO from this policy to allow faster convergence time at the stage of DODAG construction. Otherwise parent selection is not performed until the expiration of the BalancingTimer. The details of this process is also illustrated in Algorithm 2.

Algorithm 1 : Parent Selection Algorithm

```
Function getPreferredParent(P1,P2)
Input : P1 , P2 from the parent set
Output : Preferred Parent (PP)
    if P1= PP or P2 = PP Then
        if P1. Rank = P2.Rank Then
            if P1. ChildN < P2.ChildN - α Then
                return P1
            else if P2. ChildN < P1.ChildN - α Then
                return P1
            else
                return PP
            end if
        else if P1. Rank < P2.Rank – β Then
            return P1
        else if P2. Rank < P1.Rank – β Then
            return P2
        else
            return PP
        end if
    else
        if P1. Rank = P2.Rank Then
            if P1. ChildN < P2.ChildN  Then
                return P1
            else
                return P2
            end if
        else if P1. Rank < P2.Rank Then
            return P1
        else
            return P2
        end if
    end if
end function
```

Algorithm 2 : Load-Balancing Objective Function

```
procedure Initialization
    Set FastPropagationTimer
    Set BalancingTimer
    Init CHList // Children list
end procedure

procedure Data Packet Received
    if RPL HbH Option set to 1 Then
        if CHA is not in CHList Then
            add CHA to CHList
            Set CHAliftimeTimer
        else
            Reset CHAliftimeTimer
        end if
    end if
end procedure

procedure CHAliftimeTimer Expired
    Remove CHA from CHList
end procedure

procedure  FastPropagationTimer Expired
    if CHList has changed by a specific threshold
    Then
        Reset Trickle Timer
    end if
end procedure

procedure  BalancingTimer Expired
    Execute Parent Selection Algorithm
end procedure
```

VI. PERFORMANCE EVALUATION

In this subsection, the proposed scheme is compared to the RPL with OF0 in terms of power consumption and Packet Delivery Ratio (PDR). In particular, we have compared three versions of our proposed schemes with RPL. The compared versions are: LBPLAIN, in which we implement the load

balancing part of our proposed scheme without employing Balancing or FastPropagation timers; LBS in which we employ the Balancing timer, but not the FastPropagation timer; and, finally, LBSR which uses both timers. The popular Cooja simulator [24] [25] of the Contiki operating system is used to carry out the simulation experiments. Contiki is selected because it is particularly designed for IoT LLN devices and includes a standard implementation of RPL in ContikiRPL library, used as a ground for our implementation. For each scenario, five simulation experiments with different seeds are run to get statistically valid results. The simulation time is 60 virtual minutes for each experiment.

In the first set of experiments, we have evaluated the performance of RPL, LBPLAIN, LBS and LBSR in a network of 50 nodes spread randomly over an area of 50 x 50 meters. Fig. 3 shows the Packet Delivery Ratio (PDR) with various traffic rates. As can be observed the RPL protocol has the lowest PDR especially when the network is characterized by heavy or moderate traffic loads (30 and 12 packet per minute) while the LBSR has the highest PDR (33% improvement over RPL and 10% over both LBS and LBPLAIN in the case of heavy traffic load). The superiority of LBSR over RPL is attributed to the load balancing primitive that strives to distribute load among respective parents by constructing a balanced tree in terms of number of children. The absence of load-balancing in RPL leads to some nodes being highly overloaded, risking buffer overflows. While both LBS and LBPLAIN also try to build a balanced topology, they fail to achieve the delivery rates of LBSR. In the case of LBPLAIN, this can be attributed to the herding-effect which prevents it from achieving a balanced topology giving rise to several overloaded node, though fewer than in RPL. LBS is somewhat different and the main reason for low PDRs compared to LBSR is the inefficient propagation of the load-balancing routing information, with outdated information leaving the topology partially balanced. This is addressed in the LBSR protocol by resetting Trickle Timer periodically upon detecting that the load-balancing information (number of children) has changed significantly.

The protocols were also evaluated in terms of average power consumption as shown in Fig. 4. It is also clear from the figure that the LBSR protocol is the most efficient in terms of average network power consumption. For instance, LBSR registers average power consumption slightly less than that of RPL under heavy and moderate traffic loads even though it has higher data delivery rates (under the low traffic rate of 6 packet per minute, they register comparable PDR and power consumption rates). In RPL, when a node becomes overloaded with children, there is a higher probability that packets and acknowledgments will be lost, more retransmissions at the MAC layer and increased power consumption compared to LBSR, despite lower PDRs. LBPALIN registers the worst power consumption rates among all compared protocols. Apart from being incapable of fully balancing the topology, resulting in higher energy consumption rates than LBSR, it suffers from the herding problem causing churn in the network, as depicted in Fig.5, leading to higher energy consumption rates. This is addressed in LBS and LBSR by limiting the number of times a node is allowed to change preferred parent through introduction of the Balancing timer.

We have also investigated how well the network is load-balanced in terms of energy expenditure under the different schemes. One way assess this is via the Coefficient of Variance (CV), the ratio of the standard deviation to the mean. The lower the value of the CV, the more well balanced the network is and vice versa. Fig.5 depicts the CV for varying traffic rate for the protocols being compared. It is clear from the figure that the RPL protocol has the highest CV values with approximately 90%, 40% and 27% under the different traffic rates while the LBSR has the lowest CV values (i.e. 10%, 15% and 18%), indicating a significant enhancement over RPL.

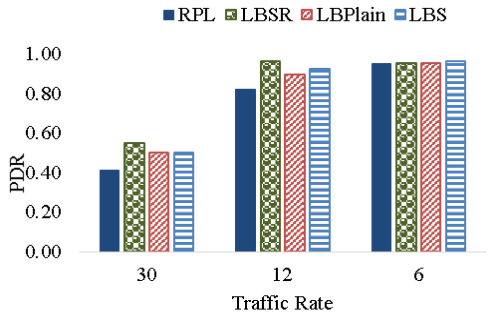

Fig. 3. Packet Delivery Ratio under various traffic rates

This illustrates that the proposed LBSR protocol succeeds in load balancing the traffic and consequently the energy consumption, which translates into improvements in both average energy consumption as well as the PDR.

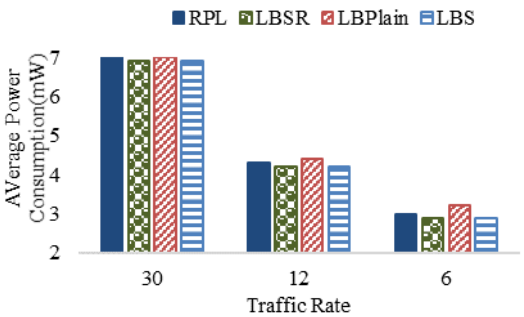

Fig. 4. Average Power Consumption under various traffic loads

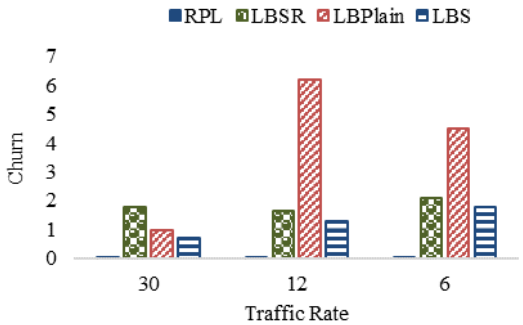

Fig. 5. Churn under various traffic loads

To get more insight into these facts, we have plotted a 3D mesh of power consumption for both RPL and LBSR in Fig. 7. The figures indicate that there are a few nodes that significantly consume power under standard RPL.

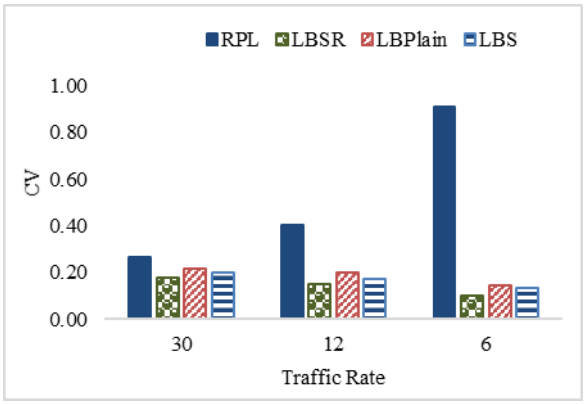

Fig. 6. The CV under different traffic loads

It also illustrates how LBSR manages to load balance energy consumption with all nodes having power consumption rates between 4 and 8mW. In the case of RPL, some nodes register average power consumption of 14mW, which is double that of the most overloaded node in LSBR.

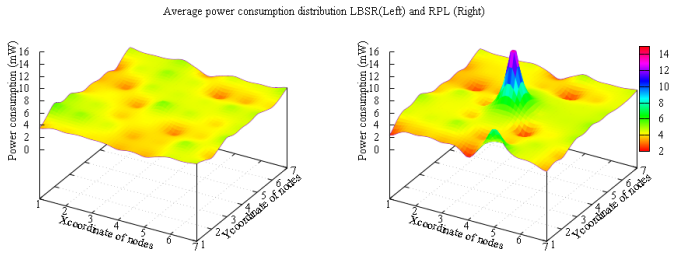

Fig. 7. Power Consumption Distribution, LBSR (left), RPL (Right)

VII. Conclusion and Future Work

In this study, a new load balancing Objective Function called LBSR is proposed for the RPL protocol. In particular, a new routing primitive is devised to calculate number of children. In addition, a parent selection and optimization primitive is introduced, based on a lexical combination of number of children and a primary metric such as hop count. A new primitive for scheduling the parent selection mitigates the herding-effect and the notion of a FastPropagation Timer regulates efficient propagation of routing information. The performance evaluation of the proposed protocol in comparison with the RPL standard has been carried out and highlights the efficiency of our proposed protocol in terms of PDR, energy consumption and load distribution. As future work, we aim to validate the efficiency of the proposed approach on real testbeds.

References

[1] J. W. Hui and D. E. Culler, "Extending IP to Low-Power, Wireless Personal Area Networks," in IEEE Internet Computing, vol. 12, no. 4, pp. 37-45, July-Aug. 2008.

[2] J. Hui, P. Thubert, "RFC 6282 Internet Engineering Task Force RFC 6282", Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, September 2011.

[3] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, K. Pister, R. Struik, J. P. Vasseur, R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks", RFC6550, Mar. 2012.

[4] T. Clausen, U. Herberg and M. Philipp, "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Wuhan, pp. 365-372, 2011.

[5] C. E. Perkins, E. M. Royer, "Ad-hoc on-demand distance vector routing", Proc. 2nd IEEE Workshop Mobile Comput. Syst. Appl. (WMCSA), pp. 90-100, 1999.

[6] https://datatracker.ietf.org/wg/roll/charter/

[7] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, "Collection Tree Protocol", Proc. ACM Seventh Conf. Embedded Networked Sensor Systems (SENSYS), 2009.

[8] S. Dawson-Haggerty, A. Tavakoli and D. Culler, "Hydro: A Hybrid Routing Protocol for Low-Power and Lossy Networks," 2010 First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, pp. 268-273, 2010.

[9] Tripathi, J., Ed., de Oliveira, J., Ed., and JP. Vasseur, Ed., "Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6687, October 2012.

[10] E. Ancillotti, R. Bruno and M. Conti, "RPL routing protocol in advanced metering infrastructures: An analysis of the unreliability problems," 2012 Sustainable Internet and ICT for Sustainability (SustainIT), Pisa, pp. 1-10, 2012.

[11] J. Tripathi, J. C. de Oliveira and J. P. Vasseur, "A performance evaluation study of RPL: Routing Protocol for Low power and Lossy Networks," 2010 44th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, pp. 1-6., 2010.

[12] O. Gaddour, A. Koubaa, N. Baccour, and M. Abid, "OF-FL: QoSaware fuzzy logic objective function for the RPL routing protocol," in Proc. 12th Int. Symp. Modeling Optimization Mobile Ad Hoc Wireless Network, pp. 365–372, 2014.

[13] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, "Load balanced routing for low power and lossy networks," in Proc. IEEE Wireless Commun. Netw. Conf., pp. 2238–2243, 2013.

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

[15] H. S. Kim, J. Paek and S. Bahk, "QU-RPL: Queue utilization based RPL for load balancing in large scale industrial applications," 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Seattle, WA, pp. 265-273, 2015.

[16] H. S. Kim, H. Kim, J. Paek and S. Bahk, "Load Balancing Under Heavy Traffic in RPL Routing Protocol for Low Power and Lossy Networks," in IEEE Transactions on Mobile Computing, vol. 16, no. 4, pp. 964-979, 2017.

[17] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012.

[18] Gnawali, O. and P. Levis, "The Minimum Rank with Hysteresis Objective Function", RFC 6719, September 2012.

[19] B. Ghaleb, A. Al-Dubai, E. Ekonomou and I. Wadhaj, "A new enhanced RPL based routing for Internet of Things," 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, pp. 595-600, 2017.

[20] O. Iova, F. Theoleyre and T. Noel, "Improving the network lifetime with energy-balancing routing: Application to RPL," 2014 7th IFIP Wireless and Mobile Networking Conference (WMNC), Vilamoura, pp. 1-8. , 2014.

[21] M. Qasem, A. Al-Dubai, I. Romdhani, B. Ghaleb and W. Gharibi, "A new efficient objective function for routing in Internet of Things paradigm," 2016 IEEE Conference on Standards for Communications and Networking (CSCN), Berlin, pp. 1-6, 2016.

[22] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[23] B. Ghaleb, A. Al-Dubai, I. Romdha, Y. Nasser and A. Boukerche, "Drizzle: Adaptive and fair route maintenance algorithm for Low-power and Lossy Networks in IoT," 2017 IEEE International Conference on Communications (ICC), Paris, France, pp. 1-6. A, 2017.

[24] Dunkels, B. Gronvall and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," 29th Annual IEEE International Conference on Local Computer Networks, pp. 455-462., 2004.

[25] Contik O.S and cooja simulatoer" http://www.contiki-os.org.