QU-RPL: Queue Utilization based RPL for Load Balancing in Large Scale Industrial Applications

Hyung-Sin Kim
Department of Electrical Engineering,
Seoul National University and INMC,
Seoul, Republic of Korea
E-mail: hskim@netlab.snu.ac.kr

Jeongyeup Paek
Dept. of Computer Information
Communication Engineering,
Hongik University,
Sejong, Republic of Korea
E-mail: jpaek@hongik.ac.kr

Saewoong Bahk
Department of Electrical Engineering,
Seoul National University and INMC,
Seoul, Republic of Korea
E-mail: sbahk@snu.ac.kr

***Abstract*—RPL is an IPv6 routing protocol for low-power and lossy networks (LLNs) designed to meet the requirements of a wide range of LLN applications including smart grid AMIs, industrial and environmental monitoring, and wireless sensor networks. RPL allows bi-directional end-to-end IPv6 communication on resource constrained LLN devices, leading to the concept of the Internet of Things (IoT) with thousands and millions of devices interconnected through multihop mesh networks. In this paper, we investigate the load balancing and congestion problem of RPL. Specifically, we show that most of packet losses under heavy traffic are due to congestion, and a serious load balancing problem exists in RPL in terms of routing parent selection. To overcome this problem, this paper proposes a simple yet effective queue utilization based RPL (*QU-RPL*) that significantly improves end-to-end packet delivery performance compared to the standard RPL. *QU-RPL* is designed for each node to select its parent node considering the queue utilization of its neighbor nodes as well as their hop distances to an LLN border router (LBR). Owing to its load balancing capability, *QU-RPL* is very effective in lowering the queue losses and increasing the packet delivery ratio. We verify all our findings through experimental measurements on a real testbed of a multihop LLN over IEEE 802.15.4.**

***Index Terms*—Low-power Lossy Network (LLN), RPL, IPv6, 6LoWPAN, IEEE 802.15.4 Load Balancing, Congestion Control, Routing, Wireless Sensor Network**

I. Introduction

Low-power and lossy networks (LLNs) comprised of thousands of embedded networking devices can be used in a variety of applications including smart grid automated metering infrastructures (AMIs) [1][2], industrial monitoring [3][4], and wireless sensor networks [5][6]. Recently, most LLN deployments employ the open and standardized IP/IPv6-based architecture to connect with the larger Internet. This approach makes LLNs more interoperable, flexible, and versatile, leading to the emerging concept of the *Internet of Things (IoT)*. With the support of various standardization efforts from the IEEE, IETF, and Zigbee, IoT systems are now equipped with protocols and application profiles that are ready for large-scale deployments [7][8][9]. Among these, this paper focuses on the load-balancing problem of the recently standardized IPv6 routing protocol for LLN, termed RPL [7][10].

RPL is designed for resource constrained embedded devices to support upcoming smart grids and many other LLN applications [7]. It is a distance vector type routing protocol that builds directed acyclic graphs (DAGs) based on routing metrics and constraints. In most deployment scenarios, RPL constructs tree-like routing topology called destination-oriented directed acyclic graph (DODAG) rooted at an LLN border router (LBR), and supports bi-directional IPv6 communication between network devices. Each node in RPL advertises routing metrics and constraints through DODAG information object (DIO) messages, and builds a DAG according to its objective function (OF, rules governing how to build a DAG) and the information in DIO messages. Upon receiving DIO messages from its neighbors, a node chooses a routing parent according to its OF and local policy, and then constructs a routing topology (i.e., DODAG).

Cisco's field area network (FAN) solution for smart grids [1] (CG-Mesh) is a good commercial example that uses RPL in LLNs. It is based on the IPv6 architecture, and uses IEEE 802.15.4g/e at the PHY and MAC layer to form LLNs. On top of that, it uses 6LoWPAN, RPL, and IPv6 to provide end-to-end two-way communication to each smart metering endpoint. It supports up to 5,000 nodes per LBR (DODAG root), and envisions millions of nodes within a FAN. Cisco's CG-Mesh system provides an initial evidence that use of IPv6 and RPL over IEEE 802.15.4 is feasible towards large scale LLNs. It is also part of growing industry efforts to invest in LLN solutions to facilitate IoT. Furthermore, it shows that although RPL has been mainly designed and used for low rate traffic scenarios, it needs to be capable of handling high rate traffic. This is because, even though each node generates low rate data, nodes near a sink (i.e., LBR) have to relay very high rate traffic. For this reason, congestion and load-balancing issues need to be investigated for RPL under high traffic scenarios.

For Hyung-Sin Kim and Saewoong Bahk, this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012R1A2A2A01046220).

For Jeongyeup Paek, this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2014R1A1A2056626).

In this paper, we tackle the load balancing and congestion problem of RPL. To do so, we first provide an experimental measurement study of RPL in high traffic scenario on a real multihop LLN with low-power embedded devices. As a result, we identify that most of packet losses under heavy traffic are due to congestion and that there exists a serious load balancing problem in RPL in terms of routing parent selection. To solve this problem, we propose a simple yet effective enhancement to RPL, termed "*Queue Utilization based RPL*" (*QU-RPL*), that significantly improves the end-to-end packet delivery performance by balancing the traffic load within a routing tree. We show performance enhancements of *QU-RPL* through experimental measurements on a real multihop LLN testbed running RPL over IEEE 802.15.4.

We evaluate our proposal against a prototype implementation of RPL in TinyOS, called TinyRPL, which uses the default objective function *OF0* [11] and hop count based routing metric. We call this *default RPL* or simply *RPL* to distinguish it from our proposed *QU-RPL*. Our implementation of *QU-RPL* is a modification on this implementation.

It is worth noting that the RPL standard [7] decouples the definition of OFs and routing metrics from the main standard to provide a high degree of flexibility. It allows implementations to freely choose OFs, local policies, routing metrics, and constraints to be used for parent selection. Thus, our proposal can be regarded as *optional* from the viewpoint of an RPL implementation, and it is still standard compliant. Similarly, the TinyRPL implementation which combines the hop count for rank calculation (OF) and the ETX for parent selection is also standard compliant.

Although there have been a lot of performance evaluation studies on RPL in LLNs [2][12][13][14][15][16][17][18], there is no experimental study of load balancing in RPL over a real multihop multinode LLN testbed. Our results and findings will provide an understanding of the load balancing problem in RPL and suggest its enhancements to build up large scale LLNs.

The remainder of this paper is structured as follows. Section II presents the background and related work to clarify the motivation of our work, and Section III describes the considered scenario and experimental environments. Then, Section IV discusses the load balancing problem of RPL and its implementation based on TinyRPL. Next, Section V proposes *QU-RPL* as a simple but efficient way to alleviate the load balancing problem, and Section VI compares the performance of *QU-RPL* and RPL using testbed experiments. Finally Section VII concludes the paper.

II. Background and Related Work

Path selection and topology construction in RPL are governed by OFs and routing metrics used by RPL. The OF defines how to use link metrics and constraints for the rank computation, and how to select and optimize routes in a DODAG. However, the RPL standard [7] does not mandate any particular OF nor routing metric to be used, and leaves this open to implementations. Thus, it offers a great flexibility in meeting different optimization criteria required by a wide range of deployments, applications, and network design. IETF has defined some recommendations on how to implement OFs [11][19][20], but without specifying routing metrics to be used, and still leaves the exact selection of a parent set as an implementation choice. RFC 6551 [21] proposed some routing metrics and constraints to be used for path calculation in RPL, but also leaves the specific selection to implementations.

For this reason, most prototype implementations of RPL use OF0 [11] as their default OF[1] while combining ideas from the 'minimum rank with hysteresis objective function' (MRHOF) [19]. They also use simple routing metrics such as hop count or ETX or their combination for path calculation and parent selection. For example, TinyRPL implementation in TinyOS uses OF0 with hop count for path calculation, and also uses ETX with hysteresis at the link level for parent selection. Furthermore, Cisco's CG-Mesh system [1] is known to use the ETX objective function (ETXOF) [20] which can be regarded as MRHOF combined with ETX metric. All of these implementations are standard compliant thanks to its great flexibility.

Thus design and selection of OFs and routing metrics that meet requirements of applications and network topology are still an open research issue. Recently, Goddour et al. proposed QoS aware fuzzy logic OF (OF-FL) that combines a set of metrics to provide a configurable routing decision based on fuzzy parameters with an aim of supporting various application requirements [22]. In [23], the authors proposed a combination of two routing metrics among hop count, ETX, remaining energy, and RSSI. They also proposed two ways of combining these metrics, simple combination and lexical combination, and compared their performance and tradeoffs.

The work in [24] analyzed the impact of OFs on network topology using OF0 and link quality OF (LLQ OF). The authors in [25] proposed a delay efficient RPL routing metric. Macro et al. proposed two MAC-based routing metrics that consider not only ETX but also packet losses due to MAC contention. They also considered traffic load in the viewpoint of power consumption and reliability required at application level [26]. However, none of these works investigate the load-balancing problem. The RFC6552 for OF0, which is most widely used, explicitly states that there is no attempt to perform any load balancing. Our work is to fill this gap and to provide a mechanism that achieves load balancing while conforming to the RPL standard.

Aside from investigating OFs and routing metrics, several prior pieces of work have investigated the performance of RPL under various scenarios and configurations. Ko et al. experimentally evaluated the performance of RPL using TinyRPL implementation in TinyOS [13], and have shown that its performance is similar to that of collection tree protocol, the *de facto* data collection protocol in TinyOS, while having the benefit of an IPv6-based architecture. In [27], they also

[1]OF0 is designed as a default OF of RPL that will allow interoperation between implementations in a wide spectrum of use cases. However, it still does not define which link properties to be used as routing metrics.

evaluated and compared the performances of ContikiRPL and TinyRPL using the two most widely used operating systems in wireless sensor networks, i.e., Contiki and TinyOS. Furthermore, in [15], Herberg et al. compared RPL with LOAD using NS-2 simulations and found that LOAD incurs less overhead if the traffic pattern is bi-directional. The work in [17] presented simulation results on the network convergence process of RPL over an IEEE 802.15.4 multihop network, and investigated improvements and trade-offs. In [16], Accettura et al. analyzed performance of RPL using COOJA simulations, and Clausen et al. provided a critical evaluation of RPL regarding limitations, trade-offs, and suggestions for improvements [14]. The work in [18] investigates the interoperability problem of two downward routing modes of operations (MOPs) defined in the RPL standard, and show that there exist a serious connectivity problem in RPL when two MOPs are mixed within a single network, even for standard-compliant implementations. To address this problem, they propose DualMOP-RPL, an enhanced version of RPL, that supports nodes with different MOPs for downward routing to communicate gracefully in a single RPL network while preserving the high bi-directional data delivery performance. However, none of these works have investigated nor tackled the load balancing problem of RPL over a real multihop LLN testbed.

Ha et al. have investigated the load balancing problem when using multiple gateways [28]. They proposed MLEq and compared its performance to that of RPL. However, they reduce traffic congestion only by using additional gateways and does not address load balancing problem in an LLN with a single gateway. Liu et al. tackled the load balancing problem in a single gateway network and proposed LB-RPL which improves load balancing performance of RPL [29]. It is similar to our work in that LB-RPL allows a node to prioritize its parent candidates considering their queue utilization. However, a node detects the queue utilization information of its neighbors from how long a neighbor delays its DIO transmission; if a node is congested, it *delays* the dissemination of routing information. This is problematic because DIO packet losses and use of *TrickleTimer* do not allow DIO reception time to exactly reflect the queue utilization. Furthermore, it causes slow recovery due to long DIO transmission interval and *herding effect* by always removing the congested parent from the parent candidate set. More importantly, both of these work evaluated their proposed schemes using NS-2 simulations, and neither conduct experiments in a real LLN nor implement their schemes on real embedded devices.

III. System Model

Consider an IoT LLN system as depicted in Fig. 1. There are thousands of LLN endpoints that form a low-power lossy mesh network rooted at an LBR. The LBR connects the LLN to a wide area network (WAN) which can be either the public Internet or a private IP-based network [30]. Multiple servers for various purposes such as applications, network management, DHCP, security, etc. reside behind the WAN. In this scenario, LLN endpoints utilize IEEE 802.15.4 links

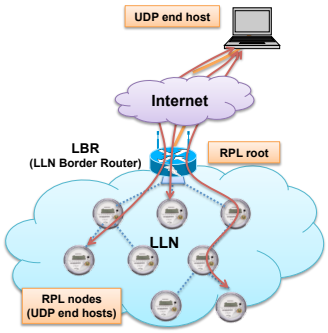

Fig. 1. Illustration of a scenario where an IoT LLN is connected to the Internet via an LBR.

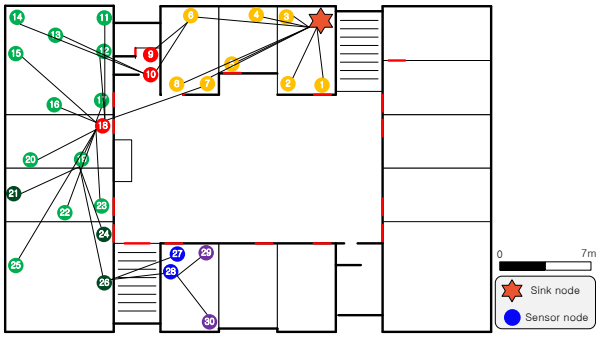

Fig. 2. Testbed topology map with a snapshot of routing paths given by RPL.

to communicate with each other, and use RPL to construct routes towards the LBR. On top of that, endpoints use IPv6 to communicate with servers. In this work, we focus on uplink traffic from LLN endpoints to servers.

To study the data delivery performance of RPL in multihop LLNs, we have configured a testbed environment as depicted in Fig. 2. There are 30 LLN endpoints and one LBR (marked with the star) in an office environment. The LBR is composed of a Linux desktop PC and an LLN interface which uses *ppprouter* stack in TinyOS and forwards IPv6 packets to the PC through UART. Each LLN node is a TelosB clone device [31] with an MSP430 microcontroller and a CC2420 radio, and uses a transmission power of -17dBm with an antenna gain of 5dB which forms a 6-hop network in our testbed.

TinyOS was used as an embedded software in our experiments. The IPv6 stack and the RPL implementation in TinyOS are called *BLIP* and *TinyRPL*, respectively. We have not used a duty cycling mechanism such as low power listening [32][33], and each node employs the default CSMA of TinyOS and a FIFO transmit queue size of 10 packets.

Using the above hardware and software setup for the experiments, we consider relatively high data rates where each node sends a data packet every 2 seconds or faster. Even though each node generates low rate data in typical LLN applications, traffic environments given by large scale applications such as smart grids can lead to a congested scenario. For example, in a network consisting of 5,000 nodes as in Cisco's CG-Mesh

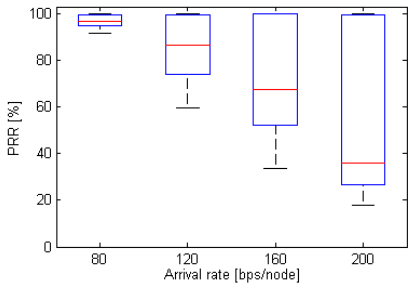

Fig. 3. Packet reception ratio (PRR) vs. uplink traffic load from all 30 nodes.

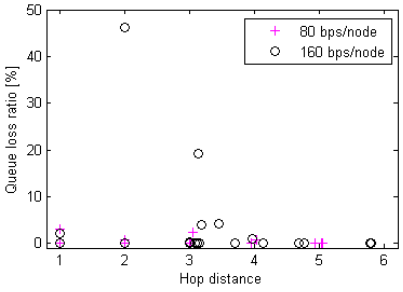

Fig. 4. Queue loss ratio of each node vs. hop distance from the LBR.

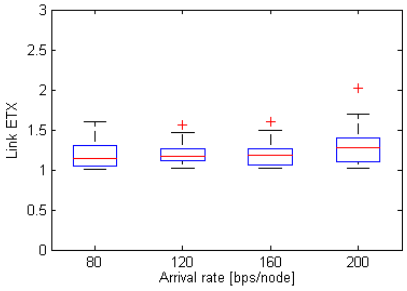

Fig. 5. Link layer ETX vs. uplink traffic load from all 30 nodes.

deployment, one packet generation every ~ 5.5 minutes at each node corresponds to the same traffic load at the bottlenecked LBR in our test scenario.

IV. Problem : RPL with OF0

In this section, we first provide an experimental measurement study of RPL with OF0 in high traffic scenarios on a real multihop LLN testbed. We show that most of packet losses in high traffic scenarios are due to congestion, and that there exists a serious load balancing problem in RPL in terms of routing parent selection. We then describe the TinyRPL implementation as a basis for describing our *QU-RPL* in Section V.

A. Load balancing problem of RPL

Fig. 3 plots the end-to-end packet reception ratio (PRR) of RPL (default TinyRPL) with varying traffic load for uplink UDP from all 30 nodes. The packet interval of each node varies from 0.8 to 2 seconds. Considering a network size of 5,000 nodes, this corresponds to the packet interval of 2.2 to 5.5 minutes per node. First of all, we observe that our testbed provides near perfect PRR for all nodes when the traffic load is light, meaning that we have reliable wireless environments.[2] The PRR gradually degrades with the traffic load, and there are some nodes with PRR as low as $\sim$20% at heavy load.

Then our first question is: *"what are the reasons for packet loss?"* To investigate this, we have collected data on how many packets are lost at the link layer and also at the packet queue within a node. We found that the link loss is negligible ($< 2\%$) even in a heavy traffic scenario (figure omitted for brevity), and most of packet losses occur at the packet queue within a node (Section VI). In other words, the queue loss is a main reason for PRR degradation in high traffic scenarios. Furthermore, queue loss occurs very severely only at a few nodes, which means that only a few nodes are suffering extremely high relay burden.

Then our next question is: *"is unbalanced queue loss is natural and unavoidable in a multihop network?"* It may be natural that nodes closer to the sink experience more relay burden and inevitably experience high queue losses in high traffic scenarios. However, our investigation claims otherwise as shown in Fig. 4, which depicts the queue loss ratio of each node according to its hop distance from the LBR, for both light and heavy loads. At heavy load, queue loss is significantly unbalanced even among nodes with the same hop distance from the LBR. Furthermore, the most congested node has the hop distance of two rather than one. We can infer that the queue loss is neither balanced nor a non-increasing function of hop distance, but comes from the inefficient parent selection mechanism of RPL. Thus, our intuition is that the queue loss can be reduced significantly by designing an enhanced RPL under heavy traffic scenarios.

Our last question is: *"does the link ETX used for parent selection reflect traffic congestion?"* Our observation indicates otherwise as shown in Fig. 5, which depicts the link ETX of each node with varying uplink traffic load. Interestingly, the ETX does not increase notably even in heavy traffic because the link capacity is higher than the queue capacity. It is due to the fact that RPL is implemented on low cost and resource constrained devices which can provide queue sizes much smaller than those of devices used for high rate communication such as LTE or WiFi (i.e., 1,000 packets at IP layer by default and more than 100 packets at link layer). Thus traffic congestion is not well recognized using ETX, and as a result, a node does not change its parent node even when the parent continuously suffers queue loss. For this reason, it is desirable to use a new routing metric and a parent selection strategy to alleviate the load balancing problem of RPL.

B. TinyRPL - RPL implementation with OF0

In this subsection we describe TinyRPL, i.e., the default RPL implementation in TinyOS 2.1.2, which implements the RPL standard [7] with OF0 along with the hop count metric for rank calculation and the ETX for parent selection.

RPL broadcasts the routing information using DIO messages which are transmitted through the *TrickleTimer* [34] to achieve a balance between control overhead and fast recovery. Furthermore, *RANK* is defined and used by the OF to represent the routing distance from a node to the LBR, and link and node metrics are used for *RANK* calculation and parent selection.

TinyRPL with OF0 uses *hop count* for *RANK* calculation, and together with *ETX* for parent selection. Specifically, *RANK*

[2]We have conducted our experiments during a night to avoid interference from daily activities of office occupants.

of node k is defined as

$$RANK(k) = h(k) + 1 \tag{1}$$

where $h(k)$ is the hop count between node k and the LBR. That is, $RANK(LBR) = 1$, and $RANK(k) = \infty$ before node k joins the network. Node k broadcasts DIO messages containing $RANK(k)$. $ETX(k, p_k)$ measured by node k is a link quality indicator between node k and its parent candidate p_k, defined as

$$ETX(k, p_k) = \frac{\text{Num. of total transmissions from } k \text{ to } p_k}{\text{Num. of successful transmissions from } k \text{ to } p_k}. \tag{2}$$

RPL smoothes the ETX using an exponentially weighted moving average (EWMA) filter, making it robust to sudden changes in link condition.

Each node recognizes its neighbor nodes by DIO messages received from them. Node k generates its parent candidate set $\mathbf{P}_k$ from its neighbor set $\mathbf{N}_k$ as

$$\mathbf{P}_k = \{ n_k \in \mathbf{N}_k \,|\, h(n_k) < h(k),\ ETX(k, n_k) < \delta \} \tag{3}$$

where δ is a threshold to remove neighbors which are connected through unreliable links.

Each node performs parent selection process when its information on parent candidates has been changed. Node k selects its best alternative parent $\hat{P}_k$ as

$$\hat{P}_k = \arg\min_{p_k \in \mathbf{P}_k} \{ R(p_k) \} \tag{4}$$

where $R(p_k)$ is a routing metric given as

$$R(p_k) = RANK(p_k) + ETX(k, p_k). \tag{5}$$

Then, it changes its parent node from the current parent P_k to the best alternative $\hat{P}_k$ if

$$R(\hat{P}_k) < R(P_k) - \sigma \tag{6}$$

where σ is a stability bound to mitigate unnecessary and inefficient parent changes, which is set to 0.5 by default. This is a hysteresis component (similar to MRHOF) of TinyRPL, and we refer to it as the stability condition. Thus, RPL allows each node to select a parent node which has a reliable link and minimum hop distance to the LBR, regardless of traffic load.

V. *QU-RPL*: Queue Utilization based RPL

In this section, we propose *QU-RPL* that is simple, but has the capability of load balancing which is lacking in RPL.

A. QU-RPL *Design*

To consider traffic congestion for parent selection, *QU-RPL* uses the *queue utilization (QU) factor* of each node k, defined as

$$Q(k) = \frac{\text{Number of packets in queue}}{\text{Total queue size}}. \tag{7}$$

QU-RPL applies the same EWMA filter for statistical calculation of $Q(k)$ as for ETX calculation.

Each node generates a parent candidate set from its neighbor nodes according to Eq. 3, and selects the best alternative (new) parent node from Eq. 4 (same as in RPL). However, for *QU-RPL*, $R(p_k)$ is replaced with a new routing metric $R_{QU}(p_k)$, defined as

$$R_{QU}(p_k) = h(p_k) + 1 + ETX(k, p_k) + \alpha Q(p_k), \tag{8}$$

where α is a coefficient which controls the weight given to the QU with respect to the hop count and ETX metric. α should be greater than one for QU to have a notable effect on the parent selection. Otherwise, $Q(p_k)$ has smaller effect than $h(p_k)$ since $0 \le Q(p_k) \le 1$.

When determining whether to change the current parent to the best alternative or not, *QU-RPL* considers both the stability condition given by Eq. 6 and the traffic congestion condition given by

$$\mu_k > \gamma \tag{9}$$

where μ_k indicates the traffic congestion around node k (which will be explained later) and γ is a threshold value for deciding when to perform load balancing. We empirically chose γ as 0.5 to reflect the idea that our load balancing scheme should come into action when the QU of a congested node is above 50%. If the condition (9) is satisfied, node k needs to change its parent node considering load balancing. Otherwise if only the stability condition is satisfied, node k in *QU-RPL* changes its parent node from P_k to $\hat{P}_k$ in the same way as in the default RPL. In other words, *QU-RPL* has the same parent change mechanism as RPL when a node does not experience congestion.

If both the stability condition and the congestion condition are satisfied, node k changes its parent node from P_k to $\hat{P}_k$ with the probability

$$\max\left\{ \kappa \left(Q(P_k) - Q(\hat{P}_k) \right), 0 \right\} \tag{10}$$

where κ is a non-negative coefficient which controls what percentage of children nodes change their parents. That is, *QU-RPL* allows a node to probabilistically change its parent considering its QU differences. This probabilistic parent change condition is designed to avoid the *herding effect*: a large number of nodes simultaneously select their parents, each of which has the smallest QU, resulting in severe traffic load for that parent, and then they will change their parents again to another one which has the smallest QU. In this way, each node may change its parent continuously without achieving load balancing, and the number of children nodes under bottlenecked parent nodes will oscillate indefinitely.

Finally, node k adjusts $Q(k)$ after the parent selection using $Q(k)$ and $Q(P_k)$, according to

$$Q(k) = \max\{ Q(P_k) - \lambda, Q(k) \}. \tag{11}$$

The intuition behind this adjustment comes from our observation that when congestion occurs at the parent of node k, $Q(P_k)$ is usually significantly greater that $Q(k)$. That is, $Q(k)$ may be small even when P_k is severely congested. However,

although a node k has low QU, it is better not to be selected as a parent by other neighbors if its parent node suffers from severe congestion. It is because node k forwards all the packets received from its children to P_k after all. Thus, *QU-RPL* performs the above QU adjustment to lower the probability of a node being selected when its parent node is congested, where λ is a small positive QU reduction factor which makes $Q(k) < Q(P_k)$. We empirically set λ as 0.25, which means that a congested node can trigger QU adjustment for up to three-hop children nodes.

B. QU-RPL *Implementation*

In *QU-RPL*, each node distributes its QU information to its neighbor nodes. There are several ways of implementing this within the RPL standard. One way is to use an optional *Metric Container* within the DIO message. For doing so, *QU-RPL* needs to newly define the *QU Metric Container* and modify the OF to use it while processing the DIO. Another way is to modify only the OF and redefine *RANK* to contain the QU value together with the previously defined *RANK* (i.e., hop count). Both approaches are within the scope of RPL standard, and thus standard compliant thanks to the flexibility and openness of RPL.

To this end, in our *QU-RPL* implementation, we take the latter approach and newly define $RANK_{QU}(k)$ for the DIO message to contain both $h(k)$ and $Q(k)$ as follows:

$$RANK_{QU}(k) = \beta\,(h(k)+1) + (\beta-1)\,Q(k) \tag{12}$$

where β is a cipher parameter used to embed and decode two values from a single numeric field. Each node broadcasts the DIO message containing $RANK_{QU}$, and extracts the hop count and the QU of a neighbor node n from its received $RANK_{QU}(n)$ as

$$h(n) = \left\lfloor \frac{RANK_{QU}(n)}{\beta} \right\rfloor - 1, \tag{13}$$

$$Q(n) = \frac{\mathrm{mod}\,(RANK_{QU}(n), \beta)}{\beta - 1}, \tag{14}$$

respectively, where $\lfloor x \rfloor$ is the largest integer which is smaller than or equal to x and $mod()$ is modulo operation. Our *QU-RPL* implementation provides each node with the QU information of neighbor nodes without changing the message format nor adding any optional field to the DIO message.

However, the current DIO transmission strategy in RPL does not provide QU information in a timely manner due to its long *TrickleTimer* period which is reset to a minimum only when changes in routing topology occur. To alleviate the problem, we also reset the *TrickleTimer* period when a node experiences a certain number of consecutive queue losses. This reset strategy allows the node to fast escape from its congested parent node.

Regarding the congestion indicator μ_k, there are some candidates such as $Q(k)$ and $Q(P_k)$. We empirically observed that $Q(k)$ is much smaller than $Q(P_k)$ even when P_k experiences a lot of queue losses, and thus it cannot reflect the traffic congestion properly. Moreover, when $Q(k)$ is large, the parent selection of node k cannot help reducing the load. Rather, it is more important to let its children nodes migrate under another parent.

We also observed that $Q(P_k)$ is not a proper congestion indicator. Once the traffic load is well balanced, each node has low $Q(P_k)$ and the congestion condition (9) is not satisfied. Thus each node changes its parent node in the same way as in RPL, causing the traffic load unbalanced again. Therefore, we consider exploiting $Q_{k,max}$ which is the maximum QU among all the parent candidates that node k had in the recent past, represented as

$$Q_{k,\max} = \max\left\{Q_{k,\max}, \max_{p_k \in \mathbf{P}_k}\{Q(p_k)\}\right\}. \tag{15}$$

With the use of $Q_{k,max}$, each node memorizes the previous congestion event, mitigating hasty parent changes that may occur when the traffic load is balanced by exploiting QU under heavy traffic. Overall, our *QU-RPL* implementation requires only 3,480 bytes of ROM and 22 bytes of RAM extra compared to the current TinyRPL implementation, and operates well on low cost embedded devices.

VI. Performance Evaluation

In this section, we analyze measurement results for *QU-RPL* on the testbed setup, and evaluate it against TinyRPL.

A. Packet delivery performance

Fig. 6 depicts the queue loss ratio (i.e., dropped packets divided by injected packets into the IP layer packet queue) with varying uplink traffic load. We observed that *QU-RPL* reduces the average queue loss ratio significantly, especially at bottlenecked nodes. This reveals that considering the QU for parent selection has a critical impact on the load balancing, and thus enables *QU-RPL* to achieve lower and fairer queue loss compared to RPL.

To show more details, we use Fig. 7 which depicts the data transmission burden of each node with varying uplink load where outliers are represented by plus markers. From this figure, we directly observe that *QU-RPL* distributes the traffic load more evenly than RPL. *QU-RPL* significantly reduces the data transmission burden of the most congested node, while only slightly increasing those of other nodes. This is the main cause of the queue loss reduction shown in Fig. 6.

The reduction in queue losses are directly translated into PRR improvements, as shown in Fig. 8 which depicts the PRR performance of each node for RPL and *QU-RPL* with varying uplink load. It shows that *QU-RPL* enhances the PRR performance significantly for most of the nodes in the network. For RPL, its PRR degradation mainly comes from its use of limited sized queues and inadequate parent selection.

Fig. 9 depicts the link ETX of each node to its parent node with varying uplink traffic load. We observe that both RPL and *QU-RPL* allow a node to select its parent with good link quality since the measured ETX is at most 2 in all cases. Moreover, the link ETX results for both protocols do not vary significantly with different traffic loads, revealing that

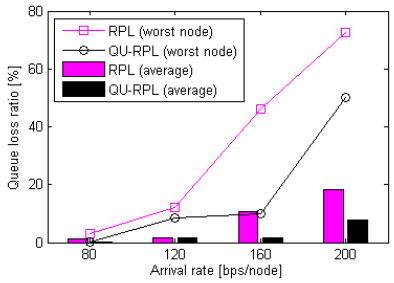

Fig. 6. Loss ratio at packet queue vs. uplink traffic load.

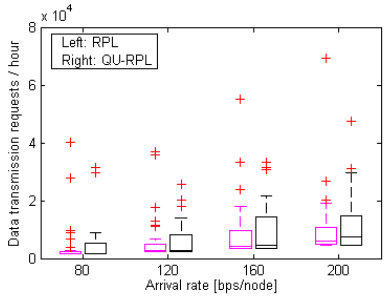

Fig. 7. Data transmission requests of each node vs. uplink traffic load.

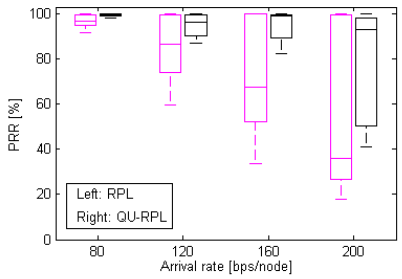

Fig. 8. PRR of each node vs. uplink traffic load.

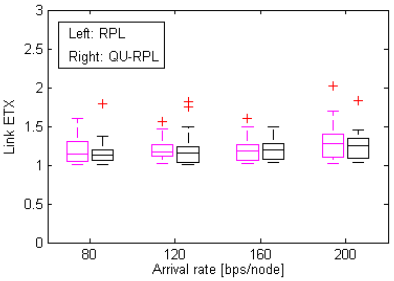

Fig. 9. Link ETX of each node to its parent node vs. uplink traffic load.

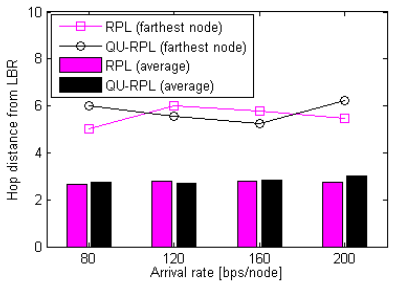

Fig. 10. Hop distance from the LBR vs. uplink traffic load.

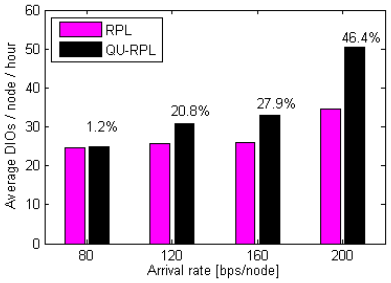

Fig. 11. Average DIO overhead of each node vs. uplink traffic load.

the wireless link capacity is not the main cause of packet loss even under heavy traffic.

Fig. 10 depicts the hop distance of a node from the LBR with varying uplink traffic load. From comparing the average hop distance and the hop distance of the farthest node, we can see that *QU-RPL* shows similar hop distance results to RPL while achieving load balancing. It is mainly because *QU-RPL* exploits the same criterion of hop distance as RPL when generating a parent candidate set. Since a node's neighbor nodes with large hop distances are already removed from its parent candidate set, its parent selection considering QU does not increase its hop distance from the LBR. Moreover, the QU weighting factor α of $R_{QU}(k)$ can be tuned considering the trade-off between hop distance and load distribution. For instance, for $\alpha = 2$, when a node experiences severe congestion, its parent node has the distance of at most one hop greater than that given by RPL .

From the results of Fig. 6 through 10, we find that simply providing a multihop route with good link quality for each node is insufficient for large scale applications with resource constrained (i.e., small queue sized) devices. *QU-RPL* requires low implementation cost and low operation overhead, but achieves significant improvement in packet delivery performance by using hop distance and QU information together.

B. Routing Overhead

Fig. 11 plots the average DIO overhead of each node under varying uplink traffic load. Each node in *QU-RPL* generates extra DIO overhead since it resets its *TrickleTimer* more frequently than in RPL to fast distribute the QU information when it suffers from consecutive queue losses. However, the increase in DIO overhead of *QU-RPL* is not significant compared to the great reduction in relay burden. Even in the lightest traffic scenario where a node generates 1,800 data packets per hour, the amount of data traffic is 36 times more than that of DIO traffic. Furthermore, RPL requires the most bottlenecked node to transmit more than 40,000 data packets per hour in the lightest traffic scenario as shown in Fig. 7. Since *QU-RPL* significantly reduces the data transmission burden of bottlenecked nodes and packet losses at those queues, their overall transmission cost will be reduced greatly.

In RPL, there is another routing overhead of transmitting Destination Advertisement Object (DAO) messages which are used for downlink route setup between a node and the LBR. Each node sends DAO messages toward the LBR periodically[3] when its route is consistent, and also when its upstream route has changed. Fig. 12 plots the average DAO overhead of each node under varying uplink traffic load. It shows that *QU-RPL* requires a similar amount of DAO overhead compared to RPL because they have similar parent changes in number.

Fig. 13 depicts the average number of parent changes of a node with varying uplink traffic load. First of all, both RPL and *QU-RPL* exhibit increasing parent changes in number with the traffic load. It is also observed that the frequency of parent change in *QU-RPL* is similar to that in RPL under light traffic,

[3]Depending on the implementation, it can be pseudo-periodic. The RPL standard RFC6550 does not mandate the transmission timing of DAO messages.

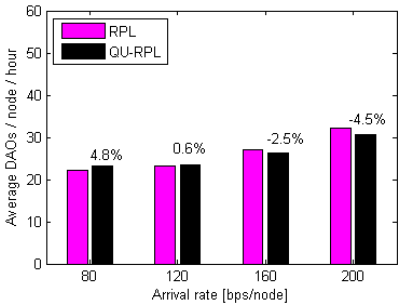

Fig. 12. Average DAO overhead of each node vs. uplink traffic load.

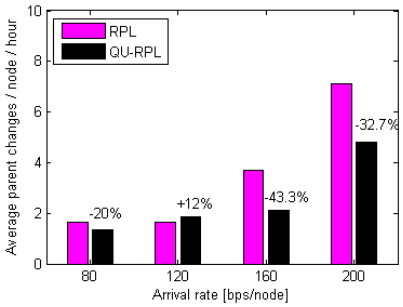

Fig. 13. Number of parent changes of each node vs. uplink traffic load.

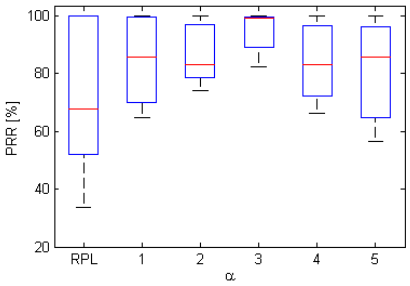

Fig. 14. PRR of each node with use of *QU-RPL* vs. weighting factor α.

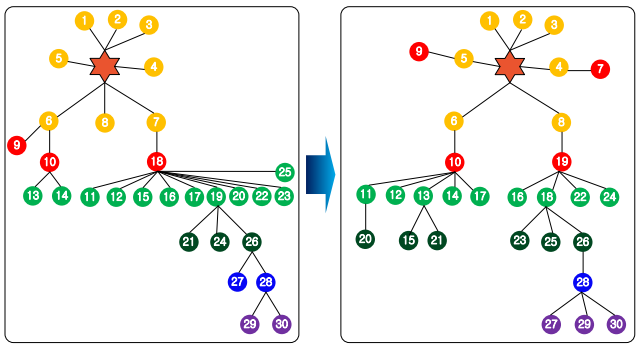

Fig. 15. Routing topology change from the default RPL to *QU-RPL*.

but significantly lower under heavy traffic. This is interesting because, at a glance, a node in *QU-RPL* seems to change its parent more frequently than in RPL since it triggers a parent change in response to congestion even when its link quality to its current parent is good.

However, the results claim otherwise. In fact, when a node in *QU-RPL* detects traffic congestion, it changes its parent node only if its alternative parent has smaller QU than its current parent. That is, each node does not change its parent even when it finds a neighbor node with better link quality and hop distance if the neighbor node has higher QU. Even when the above condition is satisfied, *QU-RPL* has another condition for parent change which happens probabilistically. This means that, compared to RPL where a node changes its parent by considering only its link quality and hop distance, *QU-RPL* has more restrictions on accepting parent change in a congested scenario. Considering the results of Fig. 8 together, we can say that *QU-RPL* improves packet delivery performance with smaller number of parent changes, resulting in stable load balancing.

C. Routing topology

Fig. 15 depicts a snapshot of the routing topology at the end of experiments for RPL and *QU-RPL*. From this, we graphically describe the effect of *QU-RPL* on topology construction.

Let us consider the physical topology in Fig. 2 again for analysis. In RPL, many nodes have selected node 18 as their parent and the number of nodes under the subtree of node 18 is more than half of all the nodes in the network. As a result, node 18 experienced a significant relay burden and dropped a lot of packets at its queue, which causes significant PRR degradation. Its children nodes are unaware of the severe congestion since they are able to successfully transmit almost all the packets to the parent, which is expected by their low ETXs.

In contrast, in *QU-RPL*, although node 19 has the most number of nodes in its subtree, node 10 and 19 have better balanced numbers of nodes in their subtrees, achieving balanced load distribution. Specifically, nodes 11, 12 and 17 select node 10 as their parent node even though node 19 is with better link quality, resulting in successful avoidance of traffic congestion. Furthermore, nodes 15, 20, and 21 select children nodes of node 10 as their parents, neither node 19 nor its children nodes. The smart use of QU leads those nodes to avoid traffic congestion by rejecting node 19 even though it has smaller hop distance. Also, *QU-RPL* prevents them from selecting a child node of node 19 as their parent node. This is because, through QU adjustment, *QU-RPL* lowers the probability that a node having a congested parent is selected as a parent node.

Quantitatively, the standard deviation of the number of children nodes per node has decreased from 1.91 to 1.30, and that of the number of nodes in subtree per node has decreased from 4.48 to 3.53, which shows that *QU-RPL* indeed achieves load balancing of children nodes.

D. Effect of parameter α

Lastly, we analyze the effect of the design parameter α on the performance of *QU-RPL*. Fig. 14 depicts the PRR given by *QU-RPL* with varying α. We added the PRR of RPL for comparison. First of all, *QU-RPL* improves the PRR performance of RPL even when $\alpha = 0$ since it permits parent change only when the best alternative parent has smaller QU than the current parent. We also observe that the PRR first increases and then decreases with α. This is due to the trade-off between congestion avoidance and routing direction. For large α, a node mainly considers QU when selecting its best alternative parent, and easily avoids traffic congestion. However, it may take a path that is significantly longer than the

shortest path by ignoring hop distance information of parent candidates. The figure shows that *QU-RPL* provides the best PRR performance when $\alpha = 2$. Therefore we exploited this value throughout our testbed experiments, which allows a *QU-RPL* node to increase the hop distance by one at most for congestion avoidance.

VII. Conclusion

In this paper, we have discussed the congestion and load balancing problem of the RPL standard. We have identified the cases where routing concentrates on a small set of forwarding parents resulting in packet delivery failures due to queue overflows, and also verified our findings through proof-of-concept implementation and testbed experiments. To address this issue, we have proposed a simple light-weight solution, called *QU-RPL*, that aims to achieve load balancing by allowing each node to select its parent node according to the queue utilization of its neighbor nodes as well as their hop distances to the border router. We have also evaluated the performance of *QU-RPL* through extensive experiments on a real testbed in comparison to the TinyRPL, and proved that our proposal greatly alleviates the packet loss problem at queues, thereby achieving significant improvement in end-to-end packet delivery performance.

References

[1] Cisco, "Connected Grid Networks for Smart Grid - Field Area Network," http://www.cisco.com/web/strategy/energy/field_area_network.html.

[2] E. Ancillotti, R. Bruno, and M. Conti, "The role of the rpl routing protocol for smart grid communications," *Communications Magazine, IEEE*, vol. 51, no. 1, pp. 75–83, Jan. 2013.

[3] German Federal Ministry of Education and Research, "Project of the Future: Industry 4.0," http://www.bmbf.de/en/19955.php.

[4] V. Gungor and G. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 10, pp. 4258–4265, Oct. 2009.

[5] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The Tenet Architecture for Tiered Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, pp. 34:1–34:44, 2010.

[6] J. Paek, J. Hicks, S. Coe, and R. Govindan, "Image-based environmental monitoring sensor application using an embedded wireless sensor network," *Sensors*, vol. 14, no. 9, pp. 15 981–16 002, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/9/15981

[7] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC 6550*, Mar. 2012.

[8] "IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks. Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," Available at http://www.ieee802.org/15/pub/TG4.html, May 2003.

[9] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," *RFC 4944*, 2007.

[10] S. Bahk and M. E. Zarki, "Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area network," in *ACM SIGCOMM Conference'92*, Aug. 1992.

[11] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," *RFC 6552*, Mar. 2012.

[12] D. Wang, Z. Tao, J. Zhang, and A. Abouzeid, "RPL Based Routing for Advanced Metering Infrastructure in Smart Grid," in *IEEE International Conference on Communications Workshops (ICC)*, May 2010.

[13] J. Ko, S. Dawson-Haggerty, O. Gnawali, D. Culler, and A. Terzis, "Evaluating the Performance of RPL and 6LoWPAN in TinyOS," in *Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, Apr. 2011.

[14] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl)," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2011.

[15] U. Herberg and T. Clausen, "A comparative performance study of the routing protocols load and rpl with bi-directional traffic in low-power and lossy networks," in *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, 2011.

[16] N. Accettura, L. Grieco, G. Boggia, and P. Camarda, "Performance analysis of the rpl routing protocol," in *Mechatronics (ICM), 2011 IEEE International Conference on*, Apr. 2011.

[17] H. Kermajani and C. Gomez, "On the network convergence process in rpl over ieee 802.15.4 multihop networks: Improvement and trade-offs," *Sensors*, vol. 14, no. 7, pp. 11 993–12 022, 2014.

[18] J. Ko, J. Jeong, J. Park, J. A. Jun, O. Gnawali, and J. Paek, "DualMOP-RPL: Supporting Multiple Modes of Downward Routing in a Single RPL Network," *ACM Transactions on Sensor Networks*, vol. 11, no. 2, pp. 39:1–39:20, Mar. 2015.

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

[20] ——, "The ETX Objective Function for RPL," *draft-gnawali-roll-etxof-01*, May 2010.

[21] J. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," *RFC 6551*, Mar. 2012.

[22] O. Gaddour, A. Koubaa, N. Baccour, and M. Abid, "Of-fl: Qos-aware fuzzy logic objective function for the rpl routing protocol," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2014, pp. 365–372.

[23] P. Karkazis, H. Leligou, L. Sarakis, T. Zahariadis, P. Trakadas, T. Velivassaki, and C. Capsalis, "Design of primary and composite routing metrics for rpl-compliant wireless sensor networks," in *International Conference on Telecommunications and Multimedia (TEMU)*, Jul. 2012.

[24] A. Brachman, "Rpl objective function impact on llns topology and performance," in *Internet of Things, Smart Spaces, and Next Generation Networking*, ser. Lecture Notes in Computer Science, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds., 2013, vol. 8121, pp. 340–351.

[25] P. Gonizzi, R. Monica, and G. Ferrari, "Design and evaluation of a delay-efficient rpl routing metric," in *International Conference on Wireless Communications and Mobile Computing (IWCMC)*, Jul. 2013.

[26] P. Di Marco, C. Fischione, G. Athanasiou, and P.-V. Mekikis, "Mac-aware routing metrics for low power and lossy networks," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2013.

[27] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, J.-P. Vasseur, M. Durvy, A. Terzis, A. Dunkels, and D. Culler, "Beyond Interoperability: Pushing the Performance of Sensor Network IP Stacks," in *ACM SenSys*, 2011.

[28] M. Ha, K. Kwon, D. Kim, and P.-Y. Kong, "Dynamic and distributed load balancing scheme in multi-gateway based 6lowpan," in *IEEE/ACM iThings*, Oct. 2014.

[29] X. Liu, J. Guo, G. Bhatti, P. Orlik, and K. Parsons, "Load balanced routing for low power and lossy networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 2238–2243.

[30] Y. Z. et al, "On deploying relays for connectd indoor sensor networks," *Journal of Communications and Networks*, vol. 16, no. 3, pp. 335–343, Jun. 2014.

[31] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proceedings of IPSN/SPOTS*, 2005.

[32] D. Moss, J. Hui, and K. Klues, "Low power listening," *TinyOS TEP 105*.

[33] K. T. Cho and S. Bahk, "Duty cycle optimization for a multi hop transmission method in wireless sensor networks," *IEEE Communications Letters*, vol. 14, no. 3, Mar. 2010.

[34] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'04)*, 2004.