IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-9427-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOMWKSHPS57453.2023.10225947

Efficient Schemes for Improved Performance in 6TiSCH Networks

Alakesh Kalita
ECE department
National University of Singapore
Singapore
alakesh@nus.edu.sg

Abhishek Hazra
ECE department
National University of Singapore
Singapore
hazra@nus.edu.sg

Mohan Gurusamy
ECE department
National University of Singapore
Singapore
gmohan@nus.edu.sg

***Abstract*—The 6TiSCH (IPv6 over Time Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4e) network is standardized to provide high reliability, higher throughput, delay-bounded, and energy-efficient communication in various resource-constrained Internet of Things (IoT) networks. However, it is observed that 6TiSCH network suffers from load balancing issue due to the default parent selection mechanism of its *de-facto* routing protocol, RPL (Routing Protocol for Low power and Lossy network). Furthermore, the performance of a 6TiSCH network degrades due to improper management of nodes' queue/buffer. None of the existing works studied these problems in RPL-based 6TiSCH networks. Therefore, in this work, we first validate these two problems using testbed experiments. Then, we propose an *Early Parent Switching* (EPS) scheme to deal with the load balancing problem of RPL by which nodes are allowed to change their parents depending on the remaining queue capacity of the parents. We propose another scheme named *Parrondo' Paradox based Queue Management* (PPQM) for efficiently managing the nodes' queue so that buffer overflow can be reduced. We implement EPS and PPQM on Contiki-NG OS and perform testbed experiments on FIT IoT-LAB. Testbed experiment results show that both the proposed schemes can significantly improve the performance of 6TiSCH networks.**

***Index Terms*—IoT, RPL, 6TiSCH, load balancing, Parrondo's Paradox, Queue Loss.**

I. Introduction

INTERNET OF THINGS (IoT) is the concept of creating communication networks among the non-living physical objects (also called "*things*") that are found in homes, healthcare systems, industries, and to name a few. IoT helps the computational system interact with the physical world by sensing, communicating, and actuating. The IPv6 over Time Slotted Channel Hopping mode of IEEE 802.15.4e (6TiSCH) network is standardized to meet the requirements of various IoT applications such as high reliability, bounded end-to-end latency, and longer network lifetime. Basically, 6TiSCH leverages the traditional IPv6-based network protocols to build upon IEEE 802.15.4e TSCH Medium Access Control (MAC) protocol [1]. Including 6TiSCH networks, most of the Low power and Lossy Networks (LLNs) use the RPL (Routing Protocol for Low power and Lossy network) routing protocol [2] to provide bi-directional IPv6 enabling routing facilities to low power and resource-constrained IoT devices. RPL organizes the nodes in a tree called DODAG (Destination Oriented Directed Acyclic Graph), which is rooted towards the root node. The nodes forward their sensory data packet towards the root node via their *preferred parents* (*i.e.,* best parent from the *parent set*). The root node is also called *Border Router* (BR). Then, the BR processes the received information by either itself or forwards it to the *cloud*. Fig. 1 shows a 6TiSCH network connected to the Internet. For forwarding the data from the leaf nodes to the BR, the nodes need to schedule *cell*[1] with their preferred parents. The 6TiSCH networks either use distributed scheduling algorithms such as MSF [3] or autonomous scheduling algorithms such as Orchestra [4] to schedule the data transmission cells between the parent-child pairs. Note that Orchestra is the *de-facto* cell scheduling scheme implemented on Contiki OS [5].

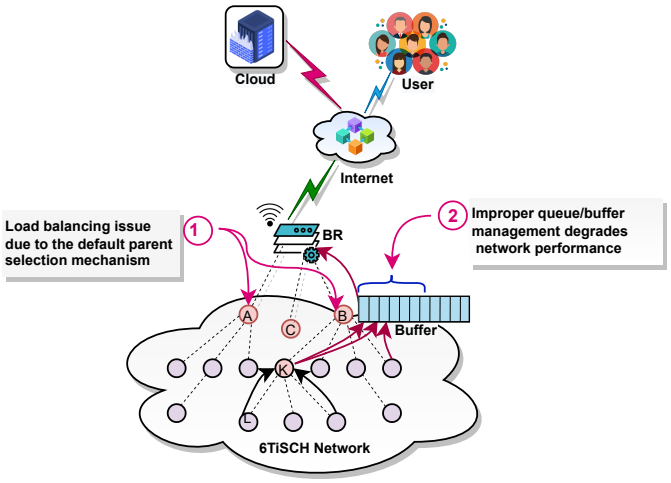


Fig. 1: An RPL-based 6TiSCH Network.

It is observed that RPL suffers from *load balancing* problem *i.e.,* some of the intermediate nodes need to handle more traffic compared to other intermediate nodes [6] and [7]. For example, in Fig. 1, nodes A and B need to handle more traffic than node C. Because of this load balancing problem, the performance of the RPL-based LLNs drops significantly as the overloaded nodes start discarding the new incoming packets from their child node(s) when their buffers are full.

[1]Cell is a combination of *timeslot* and *physical channel*. A parent-child pair needs to use a unique cell for exchanging information.

978-1-6654-9427-4/23/$31.00 ©2023 IEEE

In the literature, several works have attempted to handle the load balancing problem in RPL by altering the routing parent either periodically or depending on various network conditions. However, existing works induce extra control packet overhead in the network, which increases the energy consumption of the nodes. Furthermore, none of the existing works have considered the load balancing problem in RPL-based 6TiSCH networks. Additionally, the existing works have not studied the queue[2] overflow problem in 6TiSCH networks. Due to the limited buffer space of the LLNs' nodes and the convergence time of the cell scheduling algorithms, the nodes' buffers become full, and new incoming packets start getting discarded. It is noteworthy that a node deletes a frame from its buffer only after the maximum number of re-transmissions of the frame, and so, the frame occupies a buffer space for a longer time. However, it is generally not a good idea to randomly and unnecessarily free the nodes' queues in a computer network which can severely affect the basic network operation and decrease performance. Therefore, it is essential to carefully manage the nodes' queues to ensure reliable and efficient data transmission in 6TiSCH networks.

In this work, we propose two schemes to deal with the load balancing and queuing problems in 6TiSCH networks. In our first scheme, the nodes transmit their *buffer occupancy count* (*i.e.,* the number of outstanding packets in buffer) in the *Information Element* (IE) of the *Enhanced Beacon*[3] (EB) frame. When a (child) node finds that its parent's buffer is about to full and there is some other node(s) in its parent set with more buffer spaces, then the node probabilistically attempts to change its parent to maintain equal load in the network. We named this scheme as *Early Parent Switching* (EPS). As the information about buffer occupancy is transmitted over EB, EPS does not create any extra overhead in the network.

In our second scheme, we use the ***Parrondo's Paradox*** model [8] to efficiently manage the nodes' queues in 6TiSCH networks. Parrondo's paradox model says that two losing strategies can be combined in a certain manner to get a winning outcome. Parrondo's paradox model is used in different domains such as thermodynamics, quantum theory, biology, and medicine [9]. In 6TiSCH networks, the buffers of the intermediate nodes quickly become full due to limited buffer space and inefficient queue management, which results in network performance degradation. On the other hand, deleting packets from the nodes' queues also has a significant adverse effect on the performance of the networks. However, through our testbed experiments, we find that the performance of 6TiSCH networks can be improved by intelligently dropping packets from the queue. Therefore, we attempt to alternatively use both the *losing strategies* (*i.e.,* queue overflowing and packet dequeuing) in a certain manner to get a *winning outcome i.e.,* performance improvement of 6TiSCH networks.

[2]We use the terms "*buffer*" and "*queue*" interchangeably in this work.

[3]EB frame is periodically broadcasted by the already joined nodes, which contains basic network information to help the new nodes (aka *pledge*) to join the network and to maintain network synchronization.

The main contributions of this work can be summarized as follows,

- We perform testbed experiments to show how the load balancing problem of RPL and inefficient queue management can affect the performance of 6TiSCH networks.
- We propose an *Early Parent Switching* scheme to deal with the load balancing problem in 6TiSCH networks.
- We designed a novel *Parrondo' Paradox based Queue Management* (PPQM) scheme to efficiently manage the nodes' queue in 6TiSCH network.
- We implement both EPS and PPQM on Contiki-NG [5] and perform testbed experiments on FIT IoT-LAB [10] to validate the effectiveness of the proposed schemes.

The rest of the paper is organized as follows. Section II discusses the RPL and Parrando's Paradox model, and the existing works on RPL's load balancing problem and Parrando's Paradox. In Section III, we investigate the load balancing and queue management problems using testbed experiments. We briefly discuss our proposed schemes in Section IV and provide comparison-based testbed experimental results in Section V. Finally, we conclude our work in Section VI.

II. Background and Related Work

A. Overview of RPL and its load balancing Issue

RPL [2] uses *distance vector routing protocol* to build DODAG routing tree. A DODAG is rooted toward its root node, and it can have only one root node. Basically, the root node/BR provides interoperability between the Internet and IEEE 802.15.4e based LLN. The construction of the routing path from any leaf node or intermediate node to the BR is decided by the *Objective Functions* (OFs) and different *routing metrics* used by RPL. The DIO (DODAG Information Object) control packet of RPL carries the required routing information for the construction of DODAG. In brief, the construction of DODAG is governed by the OFs and routing metrics. However, RPL leaves the designing of OFs (or, in other words, parent selection schemes) open to everyone, which gives great flexibility in designing efficient parent selection schemes to improve the overall performance of the LLNs. It is noteworthy that none of the existing OFs, such as OF0 [11] and MRHOF [12], consider the buffer occupancy of the intermediate nodes for parent selection. Even though the ETX (*Expected Number of Transmission*) metric in MRHOF penalizes bad links, it does not consider any penalty when a packet is loss due to buffer overflow. Therefore, using the OFs such as OF0 and MHROF, RPL faces load balancing issue as shown in Fig. 1.

Several existing works [6], [7] have shown this load balancing problem and its affect in RPL based LLNs. Accettura *et al.* in [13] studied the performance of RPL in terms of throughput, RPL overhead, and packet delay using the COOJA simulations. However, the impact of queue size is not studied in [13]. Authors in [14] first studied the RPL load balancing problem and used multiple gateways to reduce the load in a single gateway. However, while this work did not consider the load balancing problems within LLNs, it tried to balance the

load at the BRs. Liu *et al.* addressed load balancing problem in a single gateway (*i.e.,* BR) network and introduced LB-RPL, a solution that enhances the performance of RPL as reported in [6]. However, this work has control packet overhead and is validated using simulation only. The work in [7] extensively studied the load balancing problem using testbed experiments and proposed QU-RPL to maintain balance load across the LLNs. However, it has signaling overhead and increases the nodes' energy consumption. Therefore, it can be seen that none of the existing works studied load balancing problems in RPL-based 6TiSCH networks. Even though the load balancing problem in the RPL-based 6TiSCH network is similar to the other RPL-based LLNs, the existing solutions might not work effectively due to limited resources allocated for control packet transmission in 6TiSCH network *i.e.,* one shared cell per slot frame [15], [16], [17]. Therefore, load balancing in an RPL-based 6TiSCH network is still an active research problem.

B. Parrando's Paradox Model

Parrando's Paradox is a paradox in game theory that says "*there exist pairs of games, each with a higher probability of losing than winning, for which it is possible to construct a winning strategy by playing the games alternately*" [18], [19]. Mathematician Juan Parrondo showed his analysis using the Brownian ratchet experiment, where a machine can purportedly extract energy from random heat motions [8], [18], [19]. Parrando's Paradox can be easily understood using the below coin-tossing example with two games where capital C_t is changed depending on the outcome of the game played at time t *i.e.,* for winning $C_{t+1} = C_t + 1$ and for losing $C_{t+1} = C_t - 1$ [19]. The games are as follows,

- **Game A**: A biased coin is used with winning probability $P_1 = \frac{1}{2} - \epsilon$, where $\epsilon > 0$. So, this game is a losing game.
- **Game B**: If the current capital C_t is a multiple of M, another biased coin is used with winning probability $P_2 = \frac{1}{10} - \epsilon$, otherwise, another biased coin is used with winning probability $P_3 = \frac{3}{4} - \epsilon$. For any integer value such as $M = 3$ and $\epsilon = 0.003$, it can be seen that Game B also turns out to be a losing game if we model the game as a Markov Chain.

However, if we play these two losing games alternatively, such as $AABBAABB\ldots$ or $ABBABB\ldots$, these combinations of the two games are, paradoxically, a winning game. This coin-tossing example is a popular canonical illustration of Parrondo's Paradox.

Recently, the work [9] used Parrando's Paradox model to restrict Covid-19 spreading and lower its impact on countries' economies. The authors proposed a Parrando's Paradox based model to alternate between imposing a lockdown strategy and keeping the community open in a specific manner, which results in a favorable outcome. The Parrando's Paradox concept is used by different researchers in different application domains such as social dynamics, life science, ecology, evolutionary biology, and other interdisciplinary work [20], [21]. However, none of the existing works have studied Parrando's Paradox model in either computer networks or LLNs.

TABLE I: Experimental Settings

Parameter	Value
Operating System	Contiki-NG
Mote	Cortex M3 IoT nodes
Testbed	FIT IoT-LAB, Grenoble
RPL version	RPL Classic
Timeslot length	10 ms
Unicast SF size	7, 19, 67, 101 slots
Common SF size	31 slots
EB SF size	397 slots
EB rate	16 SFs
Application traffic rate	2, 3 per SF
Queue size	64 pkts
Number of channels	4
Application warm-up time	20 mins
Data collection time	40 mins

III. Preliminary Study: RPL + Orchestra

In this section, we present a preliminary case study on RPL [2] using the autonomous cell scheduling scheme, Orchestra [4]. We perform testbed experiments on FIT IoT-LAB using the topology shown in Fig. 2, and other experimental settings are as mentioned in Table I. Note that the topology shown in Fig. 2 is constructed by the RPL routing protocol during our experiments. In our experiments, we use end-to-end periodic upward traffic to the root, where every node generates a data message (i.e., application traffic rate) after a certain configured slotframes (SF) and forwards them to the root node. We use the following three metrics for evaluation,

- **Packet Delivery Ratio (PDR)**: Ratio between the number of packets received by the root node and number of packet generated by a node. PDR of a node is calculated at the root node.
- **Packet Acknowledgement Ratio (PAR)**: Ratio between the number of received MAC-layer acknowledgment (ACK) and number of transmitted packets in link layer. Each node calculates the average PAR by itself.
- **Queue Loss**: Number of packets loss due to buffer overflow, which is calculated at the parent nodes.

We consider the **Sender Based (SB)** and **Receiver Based (non-storing) (RB)** operation modes of Orchestra. RB mode usages fewer `Rx` slots/SF compared to SB mode. In brief, in RB mode, each parent maintains a single `Rx` slot/SF for all of its child nodes, while in SB mode, each parent has a `Rx` slot/SF for each of its child nodes. For evaluation, we consider a high application traffic rate of $3/6$ data message per slotframe to study the load balancing and queue overflow problems. This high traffic rate is possible in large IoT networks consisting of $5,000$ nodes (e.g., in Cisco's CG-Mesh deployment). Note that we keep 20 minutes warm-up time before collecting the experimental data so that all the nodes join the 6TiSCH network.

The average values of the results obtained from running each experiment 10 times on the testbed are shown in Fig. 3. It can be seen from the obtained results that when the SF size increases (*i.e.,* from 7 to 101 slots), the overall average PDR of the network has drastically decreased (*i.e.,* from 82.54% to 16.16% in RB with data rate $3pkts/SF$). There could be two possible reasons for this decreasing PDR which are increasing

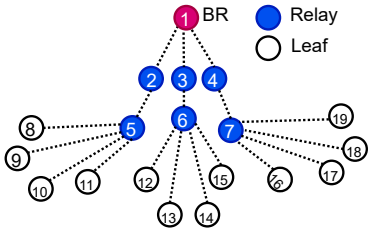
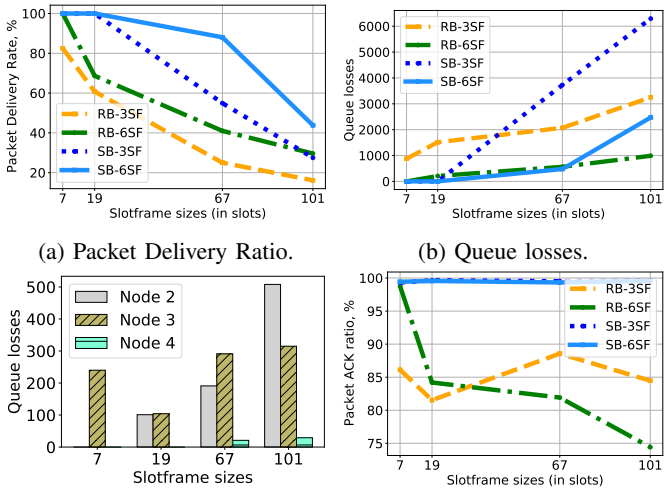


Fig. 2: Routing topology constructed by RPL.

collisions in parent's Rx slot and buffer overflow. Nodes get less opportunity to transmit their data packets when SF (*i.e.,* 101 *slots*) size is large compared to small SF (*i.e.,* 7 *slots*) size. It is because, in large SF, Rx slot appears after a longer interval. Because of this, the queues of the nodes become full when the incoming packet rate is more than the outgoing packet rate and causes buffer overflows. Due to this buffer overflow, the intermediate nodes need to drop new incoming packets, which results in decreasing PDR. The results on PDR are shown in Fig. 3a, and in Fig. 3b, we provide the results on the number of queue losses by varying the SF size. Results show that queue loss increases with the increasing size of SF.

We observe an interesting point that the differences in the queue losses of the nodes near BR *i.e.,* node 2, 3, and 4 are very high for all the experimental settings *i.e.,* in both RB and SB and for all the SF sizes. The results on queue losses by the node 2, 3, and 4 are shown in Fig. 3c for RB mode of Orchestra schedule with $3pkts/SF$. This difference in queue losses signifies that the default RPL could not efficiently distribute the traffic from node 5, 6, and 7 among node 2, 3, and 4. Therefore, nodes 2 and 3 suffer from high queue loss compared to node 4 in all the experimental settings. The queue losses could be decreased if the traffic from node 5, 6, and 7 is evenly distributed among the node 2, 3, and 4. Note that all the nodes from node 2 to node 7 are in the same communication range. *This difference in queue losses signifies the load balancing problem of default RPL.* We also observe that sometimes the other nodes, such as node 5, 12, and 16 have higher queue losses than node 2, 3, and 4. It happens because MRHOF favors good links over short paths, and so, sometimes, nodes such as 2, 3, and 4 also transmit packet to 5, 6, and 7, when they do not get response from the root node. However, we do not solve this problem in this work to make our work concise to load balancing and queue overflow problems. In Fig. 3d, we show the average PAR of the network with varied SF sizes. It is noteworthy that PAR does not change significantly with varied SF sizes, unlike PDR. The PAR is calculated hop-to-hop *i.e.,* the receiver node sends link layer ACK to the sender when it successfully receives a packet irrespective of whether the packet is queued or not in current Contiki-NG implementation.

As the in-built channel hopping feature of TSCH reduces the interference effect, it results in fewer link losses, so higher PAR can be observed for all the experimental settings. However, it can be seen that even though nodes have often successfully transmitted packets to their one-hop distance parents (as PAR is high), the packets failed to reach the BR (as PDR is low). These lower PDR and higher PAR values signify that the network is severely suffered from buffer overflows. Note that, in Contiki NG, a node transmits link layer ACK whether the received packet is stored or discarded from the queue. Hence, both load balancing and buffer overflow problems raise the question *"Can we improve the performance of 6TiSCH networks by effectively distributing the traffic load among the nodes and managing the queue?"*



(a) Packet Delivery Ratio. (b) Queue losses.

(c) Node-wise Queue losses. (d) Packet ACK Ratio.

Fig. 3: Testbed experimental results with varied SF sizes. (x) SF denotes 'x' packets are generated per SF.

IV. Proposed Approaches

We propose two schemes to deal with the problems of RPL load balancing and buffer overflow in 6TiSCH networks in the following subsections.

A. Early Parent Switching

We propose the *Early Parent Switching* (EPS) scheme to address the load balancing problem in RPL-based 6TiSCH networks. To achieve this, we enable the intermediate/relay nodes to transmit their buffer occupancy count in the IE of their EB frames. Since every joined node periodically broadcasts EB, nodes can exchange their buffer occupancy counts without incurring any signaling overhead in the network. The child nodes of an intermediate node use the buffer occupancy counts of their parents, along with other metrics defined in the RPL standard, to select their preferred parent. This parent selection (or parent switching) using EPS follows this process: when a node does not receive link-layer ACKs for two consecutive frames from its parent and the buffer occupancy count of its current parent has already crossed the *minimum threshold*, the node immediately tries to change its preferred parent by selecting the next-best parent from its parent set. If the second-best parent also does not have sufficient buffer space, the next parent from the parent set is chosen as the preferred parent, and so on. Both the *minimum threshold* and two missing

ACKs indicate that many nodes are transmitting packets to the same parent, leading to collisions in the parent's Rx slot. On the other hand, when the buffer occupancy count of the current parent surpasses the *maximum threshold* value, the node *probabilistically* attempts to change its preferred parent by selecting the second-best parent from its parent set. As EPS selects the parent from the parent set of RPL, it is compliant with default RPL. Algorithm 1 illustrates the steps of EPS.

Algorithm 1 Early Parent Switching

INPUT: Parents' buffer occupancy count
OUTPUT: Parent switching decision
if a node does not receive two consecutive ACKs **then**
    if parent's buffer occupancy count is more than *minimum threshold* **then**
        Choose next best parent from the parent set
    end if
end if
if parent's buffer occupancy count is more than *maximum threshold* **then**
    Choose next best parent with probability *P*
end if

B. Parrondo's Paradox based Queue Management

Distributed scheduling schemes like MSF take some time (known as *convergence time*) to schedule the required cells with their parents. Hence, frames remain in the queue until their successful transmission, including the cell allocation time (if not allocated before). Furthermore, when a frame is transmitted in any wireless network, there is always a chance that it may be lost due to various factors such as interference, noise, or other issues with the transmission channel. Therefore, a frame needs to be re-transmitted several times before successful transmission. As a result, the more the re-transmissions (Contiki-NG OS allows up to 8 re-transmissions), the longer a frame stays in the queue. For these reasons, the limited buffer of a node quickly becomes full and the node starts discarding incoming packets, reducing network performance significantly. On the other hand, dropping packets too frequently or uncontrolled from the queue can also lead to poor network performance. Therefore, we propose Parrondo's Paradox based Queue Management (PPQM) scheme to address these problems (*i.e.,* losing strategies). PPQM alternatively allows nodes to accept new incoming packets as well as delete packets from their queues to maximize the overall performance of the network. Thus, PPQM behaves like Parrondo's Paradox model. PPQM works as follows: when a node finds that its buffer occupancy count is more than the predefined $threshold$, it starts *probabilistically* deleting a certain number of packets from its buffer to create room for new incoming packets. However, when the queue is full, the node starts deleting more packets from its queue with a higher probability. When the queue occupancy count of the node is less than the *threshold*, the node does not delete any packet from its queue. Thus, PPQM can prevent buffer overflow by dropping packets when the queue is about to be filled, resulting in reliable and efficient packet delivery in the 6TiSCH network. Algorithm 2 shows the steps of PPQM.

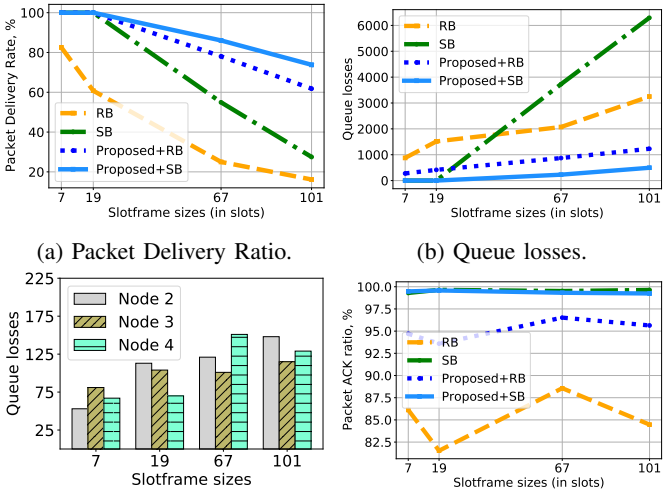
Algorithm 2 Parrondo's Paradox based Queue Management

INPUT: Current buffer occupancy, $threshold$
OUTPUT: Packet deletion decision
if current buffer occupancy is more than $threshold$ **then**
    Delete n packets from the queue with probability P_2
else if the buffer is completely full **then**
    Delete $n + k$ packets from the queue with probability P_3; $k > 0$ and $P_3 > P_2$
else
    Do not delete packet from the queue
end if

V. Experimental Results

We implement both EPS and PPQM schemes on Contiki-NG and perform testbed experiments on FIT IoT-LAB combining both schemes together. In our testbed experiments, we use the same topology and similar experimental settings as shown in Fig. 2 and Table I, respectively. To verify the effectiveness of our proposed schemes, we run both EPS and PPQM along with RB and SB modes of the Orchestra schedule. The comparison of testbed results is shown in Fig. 4. It can be seen in Fig. 4a that EPS and PPQM together improve the overall PDR of the network for RB and SB modes. As RB usages less number of Rx slots/SF compared to SB mode, and so enables higher collision, therefore, using the proposed schemes along with SB gives better performance than with the RB mode of Orchestra. The reasons for the performance improvement can be described by the results shown in Figs. 4b and 4c. In Fig 4b, it can be seen that the number of queue losses has decreased using the proposed schemes. It is because of the traffic distribution among the nodes using the proposed EPS scheme. When a node finds that the current parent's buffer occupancy is high, it selects the second best parent as its preferred parent. Note that we consider the values of *minimum threshold* and *maximum threshold* as 90% and 95% total queue capacity, respectively, and P as 50% for the EPS scheme. The network's overall performance varied with values of *minimum threshold*, *maximum threshold*, and P, and we selected these values based on several experiments. Fig. 4c clearly shows how EPS can balance the load among the nodes compared to the default RPL. The almost equal queue losses by the node 2, 3, and 4 signify that these nodes need to handle almost same amounts of traffic. Thus, using EPS, some nodes do not need to stay idle, whereas others need to handle large amounts of traffic. Because of this load balancing by EPS, the total number of queue losses decreased as shown in Fig. 4b.

Apart from the EPS, the other proposed scheme *i.e.,* PPQM also improves the network performance related to PDR and queue losses by deleting the packets from nodes' queues when their queue is about to be or is already full. For PPQM, we consider the values of *threshold* as 95% of the total queue



(a) Packet Delivery Ratio. (b) Queue losses.

(c) Node wise Queue losses. (d) Packet ACK Ratio.

Fig. 4: Testbed experimental results of the proposed schemes using application data rate $3pkts/SF$. Fig. 4c shows the results for (Proposed+ RB) scheme.

capacity, P_2 as 25%, P_3 as 85%, $n = 2$ and $k = 3$ based on our experiments. Basically, PPQM improves the network performance by deleting the packets from the queue which are there for a long time and get attempted several times for re-transmission. Note that nodes follow *backoff algorithm* for the re-transmission of the packets. Hence, nodes stay longer time in the queue with the increasing number of re-transmission due to longer *backoff* time. Thus, the new incoming packets keep getting discarded for a longer time when the queue is full, which ultimately decreases the PDR and increases queue losses. By deleting such packets from the queue, PPQM improves the performance of 6TiSCH networks in terms PDR and queue loss. In Fig. 4d, we show the PAR by varying SF sizes, where we can see that both ESP and PPQM improve the PAR by reducing the collision on `Rx slot` of the parents. As the nodes distribute their traffic to different parents, the contention on the `Rx slots` also gets reduced, which increases overall PAR of the network.

VI. Conclusion

In this work, at the beginning, we showed the impact of load balancing and queue overflow problems in RPL-based 6TiSCH networks using testbed experiments. To address the load balancing issue, we proposed the *Early Parent Switching* (EPS) scheme, which ensures even distribution of traffic load among the RPL's parent nodes based on their buffer occupancy. Additionally, deleting or retaining data packets in node queues for extended periods significantly affects overall network performance. Therefore, we proposed the Parrondo's Paradox-based Queue Management (PPQM) scheme for efficient management of data packets in nodes' queues. We implemented EPS and PPQM on Contiki-NG and conducted testbed experiments on FIT IoT-LAB, which demonstrated that the combined use of EPS and PPQM can significantly improve the performance of 6TiSCH networks. In the future, we plan to dynamically adjust the threshold values used in EPS and PPQM schemes based on various network conditions.

Acknowledgments

This work is supported by the National University of Singapore under MoE AcRF Tier-2 Grant, NUS WBS No. A-0009452-01-00 (MoE T2EP50120-0021).

References

[1] X. Vilajosana *et al.*, "IETF 6TiSCH: A Tutorial," *IEEE Commun. Surv. Tuts.*, vol. 22, no. 1, pp. 1–21, First Quater 2020.
[2] T. Winter *et al.*, "RPL: IP 6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force, RFC 6550, March 2012.
[3] T. Chang, M. Vučinić, X. Vilajosana, S. Duquennoy, and D. R. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," RFC 9033, May 2021.
[4] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proc. of 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 337–350.
[5] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. of 29th Annual IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 455–462.
[6] 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*, 2013, pp. 2238–2243.
[7] 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," *IEEE Trans. Mobile Comp.*, vol. 16, no. 4, pp. 964–979, 2017.
[8] G. P. Harmer and D. Abbott, "A review of Parrondo's paradox," *Fluctuation and Noise Letters*, vol. 2, no. 02, pp. R71–R107, 2002.
[9] K. H. Cheong, T. Wen, and J. W. Lai, "Relieving cost of epidemic by parrondo's paradox: A covid-19 case study," *Advanced Science*, vol. 7, no. 24, p. 2002324, 2020.
[10] C. Adjih *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. of IEEE 2nd World Forum on Internet of Things*, December 2015, pp. 459–464.
[11] P. Thubert, "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)," RFC 6552, Mar. 2012.
[12] O. Gnawali and P. Levis, "The Minimum Rank with Hysteresis Objective Function," RFC 6719, Sep. 2012.
[13] N. Accettura, L. A. Grieco, G. Boggia, and P. Camarda, "Performance analysis of the RPL Routing Protocol," in *IEEE International Conference on Mechatronics*, 2011, pp. 767–772.
[14] M. Ha, K. Kwon, D. Kim, and P.-Y. Kong, "Dynamic and Distributed Load Balancing Scheme in Multi-gateway Based 6LoWPAN," in *IEEE International Conference on Internet of Things*, 2014, pp. 87–94.
[15] A. Kalita and M. Khatua, "6TiSCH – IPv6 Enabled Open Stack IoT Network Formation: A Review," *ACM Trans. Internet Things*, vol. 3, no. 3, jul 2022.
[16] A. Kalita and M. Khatua, "Adaptive Control Packet Broadcasting Scheme for Faster 6TiSCH Network Bootstrapping," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 395–17 402, 2021.
[17] A. Kalita and M. Khatua, "Opportunistic Priority Alternation Scheme for Faster Formation of 6TiSCH Network," in *Proc. 21st Int. Conf. Distrib. Comput. Netw*, 2020, pp. 1–5.
[18] G. P. Harmer and D. Abbott, "Parrondo's paradox," *Statistical Science*, pp. 206–213, 1999.
[19] G. Harmer and D. Abbott, "Losing strategies can win by Parrondo's paradox," *Nature*, vol. 402, no. 6764, pp. 864–864, 1999.
[20] K. H. Cheong, J. M. Koh, and M. C. Jones, "Multicellular survival as a consequence of Parrondo's paradox," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 23, p. E5258—E5259, June 2018.
[21] K. Cheong, J. M. Koh, and M. C. Jones, "Paradoxical Survival: Examining the Parrondo Effect across Biology," *BioEssays*, vol. 41, no. 6, p. 1900027, 2019.