Enhanced 6P Transaction Methods for Industrial 6TiSCH Wireless Networks

YUVIN HA\textsuperscript{*}, (Student Member, IEEE), AND SANG-HWA CHUNG\textsuperscript{\textregistered}, (Member, IEEE)
Department of Computer Engineering, Pusan National University, Busan 46241, South Korea
Corresponding author: Sang-Hwa Chung (shchung@pusan.ac.kr)
This work was supported in part by the Korea Institute of Energy Technology Evaluation and Planning (KETEP), in part by the Ministry of Trade, Industry, and Energy (MOTIE) of the South Korea under Grant 20182020109660, and in part by the Ministry of Science and ICT (MSIT), South Korea, under the ICT Consilience Creative Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2020-2016-0-00318.

ABSTRACT IEEE802.15.4 time-slotted channel hopping (TSCH) is designed to provide reliability even in harsh wireless networks compromised by metal structures and external noise. TSCH provides reliability through coordinated communication based on link schedule however, no policies regarding this are defined by the IEEE802.15.4 standard. 6TiSCH (IPv6 over the TSCH mode of IEEE802.15.4e) provides management policies and link schedule negotiation mechanism that are not defined in the standard, and many studies are based on it. However, in most link-scheduling studies, 6top protocol (6P) for link schedule negotiation remains a premise, and the effect of harsh networks on 6P is rarely considered. Experimental results demonstrate that the performance of 6P is degraded due to the loss of 6P packets and transaction errors. These problems can threaten the fundamentals of existing link-scheduling studies. The main disadvantage of 6P in heavy traffic low-power and lossy networks (LLNs) is that nodes indiscriminately generate 6P requests according to their own demand. In this study, we analyze the performance and problems of 6P in harsh networks and propose parent-initiated 6P transaction and transaction revert methods to overcome such problems. The proposed methods enable stable cell negotiation even in LLNs under heavy traffic load through coordinated transactions of a parent based on the cell requirement of a child. The proposed methods are implemented using OpenWSN, an open-source project that implements the 6TiSCH IoT network stack, and its performance is evaluated through extensive experimentation. The results reveal that the proposed methods improve the success rate of 6P transactions and PDR of data packets by over 100% and 22%, respectively.

INDEX TERMS Cell allocation, harsh wireless network, IEEE802.15.4 TSCH, industrial wireless network, Internet of Things, low-power and lossy network (LLN), 6TiSCH.

I. INTRODUCTION
Internet of Things (IoT) envisions a paradigm of providing intelligent services through connectivity between numerous objects around people [1]. IoT services are spreading widely from smart homes and smart cities for basic living environments around people to industrial IoT (IIoT) for improving the safety and productivity of factory facilities. In this IoT vision, IEEE802.15.4-based low-power wireless network technology for communication by constrained devices is one of the core technologies of IoT to provide connectivity between objects. Industrial environments, such as smart factories, often have harsh wireless networks due to complex metal structures, vibrations, high temperatures, and external noise. Additionally, heavy traffic generated by numerous sensors and actuators adds to the burden of low-power wireless networks. Nevertheless, low-power wireless networks for IIoT should be able to provide robustness and reliability even in such environments.

Time-slotted channel hopping (TSCH) has been adopted as a MAC protocol for industrial wireless networks, such as WirelessHART [2] and ISA100.11a [3], owing to its high reliability, robustness against external interference [4], and deterministic characteristics [5]. Since then, the TSCH has been officially ratified as one of the MAC modes of the IEEE802.15.4-2015 standard through the IEEE 802.15.4e MAC amendment [6]. TSCH has attracted attention as a core network technology for low-power embedded systems.
in IIoT through its combination with IETF 6LoWPAN [7] for compatibility with IEE802.15.4 and IPv6 networks.

6LoWPAN provides IPv6-based communication in constrained nodes and enables the direct connection between embedded devices and the Internet. Moreover, the recently developed Ipv6PLC [8] of IETF 6lo WG can provide IPv6 network connectivity in industrial power line communication networks and accelerate the realization of IIoT.

IETF ROLL WG was established for 6LoWPAN-based routing in the IEE802.15.4 TSCH network. ROLL WG defined a low power and lossy network (LLN) as an environment where many constrained nodes are connected through a network with frequent packet loss. In this environment, the routing protocol for low-power and lossy networks (RPL) is designed to provide adaptive performance for the features of various IoT applications with low overhead [9].

TSCH provides reliability and deterministic features through coordinated communication based on link-scheduling. However, the IEE802.15.4 standard specifies only the operating mechanism of the TSCH and does not specify the operating policy and the creation and exchange method of link-scheduling. IETF 6TiSCH WG was established to reduce this gap and improve the usability of TSCH. The 6TiSCH defines the 6TiSCH operational sublayer (6top) as a management plane and 6top Protocol (6P) for link-scheduling negotiation between devices [10]. Additionally, 6TiSCH proposes a minimal configuration [11], which is a basic guideline for operating a TSCH network, and a scheduling function (SF) [12], which is a link-scheduling method. The IEEE 802.15.4 TSCH network operates as a complete IoT network suite through 6LoWPAN, RPL, 6P, and SF. The IoT network suite can establish connectivity for IoT services.

IoT services have various purposes, and the TSCH network must provide a link schedule suitable for the services. To this end, a variety of recent studies, [13]–[16], proposed 6TiSCH network techniques that can provide reliability even in harsh industrial environments. However, most link-scheduling studies have focused on the generation of efficient link schedules between devices; however, the exchange of created schedules still remains a premise.

Harsh industrial LLNs result in a poor link-state, and packets for 6P transactions to exchange link schedules can be lost in such an environment [17]. Moreover, unstable 6P transactions delay and break the link-scheduling, which threatens the fundamentals of link-scheduling studies.

The 6P protocol negotiates the link schedule using commands, such as ADD, DEL, and RELOCATE, and exchanges sequence numbers for the consistency of both schedules. Loss of 6P transaction packets can interfere with cell allocation and degrades schedule consistency between devices. Network statistics (such as queue utilization) that change due to these problems can affect the performance of RPL and link-scheduling methods [17].

In this study, we analyzed the problems of the conventional 6P transactions that might occur in LLNs under heavy traffic loads and evaluated its performance extensively through experiments on a testbed. Additionally, we proposed parent-initiated 6P transactions, which can solve such problems, and transaction revert, which can improve network stability. The proposed methods were implemented on OpenWSN [18], an open-source project that implements the 6TiSCH protocols, and its performance was evaluated through comparison with standard methods. The main contributions of this study are as follows.

1) We present various error scenarios of 6P transactions, which are the basis of the existing 6TiSCH link-scheduling study.

2) We analyze and experimentally demonstrate the effects of frequent packet loss and heavy traffic load on 6P transactions in harsh IIoT wireless networks.

3) The proposed parent-initiated 6P transaction and Transaction Revert methods enable stable cell negotiation even in LLNs under heavy traffic loads.

4) We evaluate the performance of the conventional and proposed methods in LLNs under heavy traffic loads through extensive experiments on a testbed.

The remainder of this article is structured as follows: Section II describes the background and related works to assist in understanding the proposed methods. Section III analyzes the problems of a conventional 6P transaction to explain the motivation for our research. Section IV describes the current implementation of OpenWSN and demonstrates the impact of the analyzed problem through simple experiments. Section V introduces parent-initiated 6P transaction and Transaction Revert to solve such problems, and Section VI evaluates the performance of the proposed methods through extensive experimentation. Finally, Section VII concludes our research.

II. BACKGROUND AND RELATED WORK

A. IEEE 802.15.4 TSCH MAC

TSCH is one of the MAC modes defined in the IEEE 802.15.4-2015 standard. TSCH prevents collisions in the network through link-scheduling, based on time division media access, and avoids external interference of the network through channel hopping. These features can provide reliability and deterministic delay suitable for IIoT services.

Fig. 1 illustrates an example of a typical TSCH network topology and link schedule. All nodes in the TSCH network operate in time synchronization, such that all nodes share the same absolute slot number (ASN). ASN is the number of slots counted from the start of the network, and each node

![TSCH network topology and link schedule.](image-url)
determines the channel for communication using (1) in every timeslot.

\[ CH = macHopSeq(ASN + CH_{offset} + %macHopSeqLen) \]  

(1)

\[ macHopSeq \] is an array in which the channel hopping order is specified, and \( CH_{offset}(channelOffset) \) is a logical channel specified by the link schedule. Because ASN is a continuously changing value, the actual frequency used for communication is changed at every slot even if a node uses the same channelOffset. This channel-hopping feature provides robustness against multipath fading and interference by heterogeneous communications using the same frequency band [4].

A slotframe is composed of a predefined number of timeslots. Link-scheduling is performed for one slotframe, allocating a link, which is a pair of time (i.e., slotOffset) and frequency (i.e., channelOffset), for communication between two nodes. The allocated link schedule is continuously repeated. As illustrated in Fig. 1, the communication of each node is coordinated by the link schedule, preventing internal interference by other nodes. Each node performs Tx and Rx according to the link schedule specified in the timeslot, and switches to the sleep state to reduce energy consumption when the schedule is not specified. However, the IEEE802.15.4 standard does not define a method for determining the link schedule and does not deal with path information as the basis of the link schedule. Therefore, it must construct a complete IoT network stack through collaboration with RPL and 6TiSCH.

B. IETF RPL

Embedded devices used in IoT services operate at low power and are vulnerable to packet loss. IETF ROLL WG defines this network as an LLN. RPL operates in a distributed manner such that each node locally determines its own network parent. The objective function (OF), which is the basis of parental decisions, can be freely set up to create a network suitable for various IoT services.

The RPL network aims to create a DODAG (Destination Oriented Directed Acyclic Graph) that connects to the root, and the logical distance to the root is called RANK. Each node periodically transmits a DIO (DODAG Information Object) including RANK and metric information necessary for parent decisions. The node receiving the DIO selects the node with the closest logical distance to the root as the parent using various metric information [19]. The OF for calculating the logical distance is a formula for optimizing network performance according to the purpose of the network. The IETF ROLL WG defines OF0, which is a guideline for the development of the network [20].

The node whose parent selection is completed sends the information of its parent and child through the transmission of the DAO (Destination Advertisement Object), and the RPL creates a path for downward traffic through the reception of the DAO. The downward routing is classified into storing mode or non-storing mode according to the method of processing downlink traffic. The default option is a non-storing mode, which is responsible for processing downward traffic on the root node.

In this study, we focus on the performance of 6TiSCH 6P in harsh LLNs; thus, we use the RPL OF0 proposed in RFC6550 and rank computation proposed in 6TiSCH minimal configuration (RFC8180) [11].

\[ R(N) = R(P) + RankIncrement \]  

(2)

\[ RankIncrement = ((3 \times ETX) - 2) \times MinHopRankIncrease \]  

(3)

The rank \( R(N) \) of the node \( N \) is calculated using (2) based on the information of the parent node \( P \). RankIncrement is calculated using (3) based on the ETX (Expected Transmission count) between the parent and child. RPL using this configuration generates the DODAG by selecting the parent with the lowest ETX, that is, the highest link quality.

C. 6TiSCH OPERATIONAL SUBLAYER (6TOP)

6TiSCH defines 6top, a sublayer for managing TSCH on the MAC layer [10], and provides a minimal 6TiSCH profile for basic operation [11]. 6top allows nodes participating in the network to periodically transmit EB (Enhanced Beacon), and generates KA (Keep Alive) packets to maintain time synchronization with the neighbors.

An IE (Information Element) is contained in the EB, and Table 1 presents the IEs used by 6TiSCH. The node receiving the EB is synchronized through the information contained in the synchronization IE and sets the TSCH network parameters through Slotframe & Timeslot IE.

The synchronized node generates minimal cells for EB and DIO reception using information contained in Slotframe and Link IE according to the minimal configuration of RFC8180 [11]. Thereafter, the node transmits and receives EBS according to the policy of 6top, and a minimal cell is used. Each node can maintain basic network settings by receiving control packets (i.e., EB and DIO) through a minimal cell.

The node that sets the minimal cell receives the RPL DIO from neighbors and selects the parent node. When the parent node is selected, link allocation for data communication is started using SF or link-scheduling. RFC8480 [10] of 6TiSCH defines 6P and 6P transaction to aid in this link allocation procedure. Table 2 lists the commands of 6P.

Fig. 2 depicts an example of a 2-step 6P transaction. Node A sends the ADD command and its available cell list to Node B. And Node B excludes the cells already allocated from the list by referring to its link schedule. Then, Node B selects the number of cells specified in the request (i.e., NumCells) from the list and sends a response. The return code is specified in the response packet, and Table 3 presents the return codes. A 3-step 6P transaction also exists, and in contrast to the 2-step transaction, a cell list is created at the

\(^\text{1}\)In 6TiSCH, a scheduled link or timeslot is called a cell.
TABLE 1. IEEE 802.15.4 Information Elements (IEs) Used by 6TiSCH.

| Sub-ID value | NAME                  | Type            | Description                                                                 |
|--------------|-----------------------|-----------------|-----------------------------------------------------------------------------|
| 0x1e         | Time Correction IE    | Header IE       | Used by ACK to indicate time correction                                     |
| 0x1a         | TSCH Synchronization IE | Payload IE Short | Used for time synchronization. Contains ASN and join metric               |
| 0x1b         | TSCH Slotframe and Link IE | Payload IE Short | Indicates the size and number of slotframes and announces the number of active cells in the advertised schedule |
| 0x1c         | TSCH Timeslot IE      | Payload IE Short | Timing template for a timeslot                                              |
| 0x9          | Channel hopping IE    | Payload IE Long | Channel-hopping sequence description                                        |

TABLE 2. 6P Command identifiers.

| Command     | Code | Description                  |
|-------------|------|------------------------------|
| ADD         | 1    | Add one or more cells        |
| DELETE      | 2    | Delete one or more cells     |
| RELOCATE    | 3    | Relocate cell(s) in the schedule |
| COUNT       | 4    | Count the scheduled cells    |
| LIST        | 5    | List the scheduled cells     |
| SIGNAL      | 6    | Placeholder for SF-specific commands |
| CLEAR       | 7    | Clears all cells between two neighbors |

TABLE 3. 6P return codes.

| Return code     | Description                                |
|-----------------|--------------------------------------------|
| RC_SUCCESS      | Operation success                         |
| RC_EOL          | Reached end of list when listing cells    |
| RC_ERR          | General error. Incorrect format or invalid parameters |
| RC_RESET        | Critical error. Reset transaction          |
| RC_ERR_VERSION  | Unsupported 6P version                     |
| RC_ERR_SFID     | Unsupported SFID                           |
| RC_ERR_SEQNUM   | Schedule inconsistency                     |
| RC_ERR_CELLLIST | Cell list error in a transaction           |
| RC_ERR_BUSY     | Busy in a concurrent transaction           |
| RC_ERR_LOCKED   | Requested cells are locked by another transaction |

FIGURE 2. 2-step 6P transaction.

Node B side through the request of Node A. Node A sends a confirmation packet containing the selected cell to complete the transaction. [21] and [12] detail the bootstrap process of the 6TiSCH network.

All 6P transactions use sequence numbers to maintain schedule consistency. If the sequence numbers do not match due to 6P packet loss, it is recommended to initialize the schedules of both sides through a CLEAR request or to resolve the inconsistencies through a LIST request. Retrieving the cell through the LIST command incurs a large overhead, due to receiving of all cell information, and clearing the schedule using the CLEAR command also incurs a large overhead, due to rebuilding of the schedule. In IIoT services operating on a lossy network, 6P packet loss is not a negligible problem.

D. SCHEDULING FUNCTIONS

6P transactions provide cell negotiation functions but do not decide when to add or delete cells and which cells to use. This behavior is covered in SF. 6TiSCH WG uses a minimal scheduling function (MSF) as the default SF [12]. MSF maintains two autonomous cells for 6P transaction: an autonomous Rx cell calculated using its own EUI64 address and an autonomous Tx cell calculated using the EUI64 address of a frame to be transmitted. Autonomous Rx cells must always be maintained after synchronization, whereas autonomous Tx Cells are installed on demand. The formula for autonomous cell calculation is as follows:

\[
\text{slotOffset} = 1 + \text{hash}(\text{EUI}64, (\text{slotframeLen} - 1))
\]

\[
\text{channelOffset} = \text{hash}(\text{EUI}64, 16)
\]  

The MSF calls the cell generated through the 6P transaction as a negotiated cell and records the statistics of cell usage. If the cell usage during 16 slotframes is greater than the upper threshold (i.e., 75%), a new cell is added, and if it is lower than the lower threshold (i.e., 25%), one cell is deleted. Additionally, when a specific negotiated cell has a significantly lower PDR (Packet Delivery Ratio) than other
cells targeting the same neighbor, the cell is reallocated using the 6P RELOCATE command.

E. RELATED WORKS

The IEEE802.15.4 TSCH network can provide reliability through proper link-scheduling. Recently, many studies have been conducted to improve the performance of link-scheduling in the 6TiSCH network.

On-the-fly (OTF) [22] is one of the representative scheduling methods using 6TiSCH 6P. OTF periodically measures the traffic load to be handled by performing housekeeping and allocates additional cells when more cells than the scheduled cells are needed. Conversely, when more cells than the threshold are allocated, cells are deleted to reduce energy consumption. OTF can provide high performance in a simple method by using 6P efficiently.

Reference [23] proposes a distributed scheduling method based on the well-known industrial control paradigm referred to as proportional, integral, and derivative (PID). PID scheduling considers the difference between the number of negotiated cells and the required cells for a target as an error, and calculates the required cells based on current errors, past records, and trends for the future. PID can calculate more precise cell requirements than OTF, but it generates more 6P transactions.

Reference [24] proposes an efficient link-scheduling method in a dense 6TiSCH network. This study assumes features similar to the IIoT network. Each node announces its link schedule to neighbors through selective broadcasting, and the neighbor can select cells that can avoid interference based on this information. However, the cell allocation procedure is not considered, and selective broadcasting can also be easily lost in LLNs.

Reference [25] is a link-scheduling study for providing reliability in a TSCH network with a high data rate and unpredictable traffic pattern. This study proposes a hybrid approach where dedicated cells and shared cells coexist in the same link schedule. Reference [25] assumes an environment similar to the traffic patterns generated by numerous sensors and actuators in IIoT, and the proposed method may improve the reliability of industrial wireless networks. However, in a harsh industrial wireless network, the allocation of dedicated cells remains a premise.

Most studies on the 6TiSCH network, including the above, focus on cell selection and allocation time, and assume that the 6P transaction is stable.

Reference [26] proposes ESF0 as a cell-selection method in a 6TiSCH network. It uses a 3-step 6P transaction, and the sender divides the slotframe and sends a request packet with the density of each part. The receiver creates a cell list using the densities contained in the request, and the sender sends a confirmation of the cell selection to complete the negotiation. This study proposes a new 6P transaction method to improve the efficiency of 6P. However, because the cell selection is dependent on the proposed 6P method, it is difficult to use in a general link-scheduling method in which 6P is only responsible for cell allocation. Additionally, because ESF0 requires a 3-step transaction, additional overhead is incurred. Moreover, the effects on the loss of 6P packets is not considered.

Reference [17] analyzes the performance of 6P through an experiment on a testbed. Through experiments using OTF and PID, this study demonstrates that 6P transaction error and delay are significantly increased according to the traffic rate and the link-scheduling method can be affected by the performance of 6P.

III. PROBLEM STATEMENT

A 6P transaction is a fundamental element for 6TiSCH networks to provide reliability. In a harsh industrial wireless network, 6P packets for negotiating a cell may be lost, and in a case in which 6P does not operate properly, the link-scheduling technique cannot easily provide intact performance. Therefore, it is necessary to determine the problems that may occur in 6P transactions on a lossy wireless link and analyze their effect on the overall performance of the network.

A. 6P TIMEOUT

Because the 6P packet is a unicast packet, when the acknowledgement (ACK) reception fails, retransmission is attempted a predefined number of times. The 6P packets are generally transmitted using shared cell types; therefore, a back-off mechanism is required.

Fig. 3 illustrates the 6P transaction state machine used in the implementation of OpenWSN. After the 6P request packet is generated, a node transitions to the SIX_STATE_WAIT_6P_REQUEST_SENDDONE state and waits for a minimal or autonomous cell. When the 6P request is successfully transmitted, the node transitions to the SIX_STATE_WAIT_6P_RESPONSE state and waits for a 6P response. If the node fails to receive the 6P response within a predefined timeout period, it is considered as a failure and returns to the IDLE state. The timeout period should be defined considering the time taken to transmit the response in the worst case, and is calculated as follows:

\[
\text{Timeout}(\text{ms}) = (2^{\text{MAXBE}} - 1) \times \text{MaxRetries} \times (\text{SfLen}/\text{NumCellsPerSf}) \times \text{TsLen}
\]

\[
\text{MAXBE} = \text{MaxRetries} = \text{NumCellsPerSf} = \text{TSChLength} / \text{SfLen} / \text{NumCellsPerSf} = 1, \text{timeout} = 45,450 \text{ ms, which is not considered as negligible overhead in a lossy environment.}
\]

B. 6P PACKET LOSS AND SEQUENCE NUMBER ERROR

First, we assume the case of using 2-step transactions. The example in Fig. 2 describes the update point of a sequence
number in a successful 6P transaction. When Node A receives a response and Node B receives an ACK for the response, the sequence number is updated.

1) 6P RESPONSE LOSS SCENARIO
Fig. 4 illustrates a scenario for a sequence number error caused by the loss of 6P response packets. Node B, which successfully receives the 6P request, creates and transmits a 6P response; however, the response is dropped due to an unstable link. By receiving the retransmitted response, Node A updates the sequence number; however, if Node B fails to receive the ACK from Node A, Node B does not update the sequence number, and the sequence numbers on both sides do not match. Errors in this scenario can be mitigated by increasing the maximum number of retransmissions; however, the overall burden on the network is increased.

2) 6P TIMEOUT SCENARIO
Fig. 5 depicts a scenario for a sequence number error caused by a 6P timeout. This may occur when using a timeout period shorter than the value calculated by (5). Additionally, if a dedicated cell for a transaction target cannot be provided according to the network policy, timeout may occur because the transmission opportunity is shared with other packets. In (5), the number of cells for 6P packets per slotframe is used. The 6top transitions to the IDLE state due to the 6P timeout; however, because layer-2 normally returns an ACK to the received packet, Node B recognizes that the response has been successfully transmitted and updates the sequence number.

Because other transactions are blocked while waiting for a response, an excessively long timeout period may degrade the performance of 6P transactions. Thus, it may be more efficient to set an appropriate timeout period and recover the sequence number by handling errors with a small overhead.

3) STATE INCONSISTENCY SCENARIO
Fig. 6 illustrates a scenario for sequence number error that may occur in the REL1.24 and FW-850 versions of OpenWSN. When an ACK for a request is lost, Node A maintains the SIX_STATE_WAIT_6P_REQUEST_SENDDONE state to attempt retransmission; however, Node B assumes that it has successfully received the request and creates a response.
packet. The 6top sublayer of Node A ignores the response because it does not transit to a state for receiving the response; however, layer-2 of Node A normally returns an ACK for the response. Node B assumes that the 6P transaction has completed successfully and updates the sequence number. This scenario can be prevented by modifying the design of the state machine in OpenWSN. However, any implementation of 6top based on the state machine should consider this problem.

In the 3-step transaction, the sequence number is also updated through the reception of the final step (i.e., confirmation packet) and an ACK for the confirmation. Therefore, the above scenarios may occur due to the loss of the confirmation packet and its ACK. The 3-step transaction has the advantage of being able to propose a cell list on the Node B side. However, a larger number of packet transmissions and a greater transaction time is required. In a harsh LLN, there is the possibility of timeout in response and confirmation, thereby increasing the transaction block period. For this reason, only 2-step transactions are considered in this article.

RFC8480 [10] discusses methods to detect and resolve schedule inconsistencies that may occur through the above scenarios, and recommends the following countermeasures:

1) Issue a 6P CLEAR request to clear the schedule, and then rebuild
2) Issue a 6P LIST request to retrieve the schedule
3) Internally roll back the schedule

OpenWSN adopts the first method; however, 6P CLEAR for a node with many cells (e.g. bottleneck node) drops numerous packets generated in the subtree of the node and impairs network stability. The second method is to selectively retrieve the inconsistent schedule using the 6P LIST command. This requires additional 6P transactions and results in a greater overhead in heavy traffic loads and lossy links. Moreover, because the sequence numbers of each node are already inconsistent, additional mechanisms and sequence number management policies for recovery are required.

In addition to the above scenarios, an error may occur when the sequence number is initialized by a power cycle or an old neighbor management policy. Apart from these exceptional cases, the error of the sequence number usually has a difference of 1; thus, it may be more efficient to solve the problem through the last method, i.e., rollback. In this study, we implement the last method called transaction revert to deal with sequence number errors effectively in Section V-B. As far as we know, this is the first implementation and evaluation of the 6P transaction rollback method. We demonstrate its performance through experiments.

C. 6P TRANSACTIONS UNDER HEAVY TRAFFIC LOAD

6P transactions are vulnerable to collision because they are transmitted through shared type cells. Moreover, because the convergecast traffic pattern occupies a large portion of the IIoT service, 6P transactions also converge to the parent node. A node using MSF can allocate autonomous cells for 6P transactions without exchanging information using the hash value of the EUI64 address.

Fig. 7 depicts the 6P transaction interference relations that may occur when using MSF autonomous cells. Node A is a bottleneck node where traffic converges. All the child nodes of Node A send 6P requests using slotOffset 8, which is the autonomous Rx cell of Node A, and the collisions of 6P requests can occur. It is assumed that Node A sends an ADD request to the parent Node R to process the converged traffic. Unfortunately, the response of Node R interferes with requests generated by the children of Node A. If the response packet is lost, Node A waits for a response for approximately 40–60 s, and the 6P transactions of the children are blocked during the timeout period. Thus, the unresolved 6P packets of the children accumulate, increasing the probability of collision. Dynamic events, such as the initial formation process and topology changes, and heavy traffic load generate many 6P packets. Because conventional 6P transactions must converge to the single autonomous Rx cell of the parent node, collisions of 6P packets are inevitable.

In an RPL network, a parent node always comprises fewer available cells than a child node. This gap becomes larger under heavy traffic loads, and 2-step 6P transactions face difficulties in negotiation when most of the cells of the parent node are allocated. In Node A of Fig. 7, most of the cells are allocated to child nodes. Node D selects a portion of its available cells and sends an ADD request to Node A; however, the rest of the cells, except 23, are already allocated to other nodes. Because the child node is not aware of the cell allocation status of the parent node, further allocation becomes more difficult as available resources decrease. When using 3-step transactions, the cell list is provided by the parent node, and thus, this problem can be resolved; However, since the 3-step transaction is also initiated with the request of a child, it is still difficult to avoid the collision.

To overcome such problems, we propose parent-initiated 6P transactions. Child nodes do not send 6P requests but wait for OFFER of the parent node. Since the parent node selects the child node with the highest cell requirement and initiates a 6P transaction, the coordinated transaction of the proposed method can prevent collisions. Moreover, since the OFFER of parent node always contains the available cell list, allocation failure due to lack of resources can be prevented.

IV. OPENWSN AND PERFORMANCE EVALUATION

A. OPENWSN IMPLEMENTATION

OpenWSN is an open-source implementation of the 6TiSCH and IEEE802.15.4 TSCH standard protocol stack. OpenWSN provides an IoT network full-stack composed of TSCH, 6top sublayer, 6LoWPAN, RPL, and CoAP. OpenWSN is implemented in C and is ported to various hardware platforms: TelosB, OpenMote, and the IoT-Lab motes [18].

First, we analyzed the implementation of OpenWSN. Table 4 lists the major specifications of the most recent release version (i.e., REL-1.24) of OpenWSN [27]. RPL of
OpenWSN implements DIO, DAO, and DIS features, and operates in non-storing mode, such that downward traffic is transmitted from the root using source routing. As the objective function, OF0 and rank computation of [11] are used. However, because the poisoning feature is not implemented, it takes some time to recover the DODAG loop in heavy traffic. The method of solving the DODAG loop is determined by network policy; however, in this study, we implement and use the poisoning feature to actively respond to the loop.

MSF [12] uses receiver-based (RB) autonomous cells based on the receiver’s EUI64. All nodes must maintain an autonomous Tx cell corresponding to the autonomous Rx cell of the parent. Since MSF uses RB autonomous cells, a child sends 6P requests using the autonomous Rx cell of the parent (Same as the autonomous Tx cell of the child). Conversely, a 6P response from the parent is returned using the autonomous Rx cell of the child. However, MSF of OpenWSN REL-1.24 only uses parent-based (PB) autonomous cells. Fig. 8 depicts an example of a 6P transaction in OpenWSN REL-1.24. All 6P packets are transmitted using the autonomous Rx cell of the parent. The PB method causes more collisions. The RB autonomous cell has been applied since the FW-848 developer version.

The TSCH configuration follows RFC8180 [11], and EB and DIO are transmitted with a 10% probability in every slotframe. The length of the packet queue is generally set to 10; however, considering heavy traffic load, we set it to 20.

In this study, we used OpenMote-B [28], [29] based replicas. OpenMote-B uses TI-CC2538 SoC [30] equipped with Cortex-M3-based MCU and a 2.4 GHz IEEE802.15.4-compatible RF transceiver. Additionally, it can use Atmel AT86RF215 [31], which is a sub-1 GHz IEEE 802.15.4-compatible RF transceiver. The testbed system uses the 2.4 GHz band frequency.

**B. OPENWSN PERFORMANCE EVALUATION**

In this section, we evaluate the performance of 6P transactions of OpenWSN REL-1.24 according to traffic load
changes. After the FW-848 developer version, OpenWSN provides 6P transactions based on RB autonomous cells; however, we first evaluated its performance based on the latest released version. Detailed performance analysis with RB-based 6P transactions is described in Section V.

In a heavy traffic network, the traffic load of the parent node changes rapidly according to the parent selection of the child node. MSF adds or deletes only a single cell based on cell usage in a certain period, making it difficult to respond quickly. Therefore, we use an OTF that can immediately allocate the number of required cells corresponding to the traffic. The number of required cells for OTF was calculated considering ETX, and the threshold $T$ was set to 3.

Experiments for evaluating the performance of OpenWSN were conducted with 12 nodes except the root, and all nodes were placed at a distance of 20 cm such that they could be connected to the root by a single hop. Performance evaluations compare cases with and without antennas to simulate LLN environments. The Tx power of the node was set to 3 dBm, and we experimentally confirmed that communication is feasible up to a distance of 30 cm without an antenna. In the absence of an antenna, the radio wave radiation pattern is irregular, and the link quality of nodes at a distance of 20 cm was variously measured from 60% to 95%. Therefore, a network composed of nodes without antennas would simulate a lossy IIoT network in which complex structures or other wireless protocols coexist.

Fig. 9 depicts the transaction success ratio (TSR) of 6P packets according to traffic load. Traffic load is adjusted to packets per second (pps) for each node and increases from 0.33 to 1. As the traffic load increases, more transactions occur, and the TSR decreases due to collisions. Packet drop rarely occurs until 0.5 pps; however, the TSR decreases rapidly at 1 pps.

Fig. 10 depicts the 6P transaction delay according to the traffic load. In cases without an antenna at 0.33 and 0.5 pps, no significant difference is observed in TSR, because packet loss is handled by retransmission; however, the number of transaction delays is higher. Because the 6P transaction delay is measured only when the transaction is successful, the delay does not increase noticeably even at 1 pps, although the loss increases significantly.

Fig. 11 depicts the PDR of the data packets according to the traffic load. In the absence of an antenna, the PDR of the data packet tends to decrease slightly as the traffic load increases. However, it is difficult to demonstrate the effect of 6P transaction failure on the PDR, based on this result alone. Section V evaluates the impact in more detail through extensive experiments.

Fig. 12 illustrates the history of the 6P CLEAR command and data packet loss that occurs for 20 min. The 6P CLEAR command is issued when the parent is changed, or a schedule inconsistency is detected due to the loss of 6P packets. Not all CLEAR commands have a significant effect; however, the CLEAR command of the bottleneck node instantaneously causes numerous packet losses. The CLEAR command issued at 820 s, depicted in Fig. 12, instantaneously drops approximately 150 packets. A proper handling method of schedule inconsistency can prevent instantaneous burst loss and improve network stability.

V. ENHANCED 6P TRANSACTION

A. PARENT-INITIATED 6P TRANSACTION

In this section, we introduce the parent-initiated 6P transaction method for stable cell negotiation in heavy traffic LLNs. As traffic increases, 6P requests from child nodes are concentrated with a single autonomous Rx cell of the parent node. As shown in the experiment in IV-B, the TSR decreases due to the convergence of 6P packets. In general, the autonomous Rx cell of the parent nodes tend to be more congested than the autonomous Rx cell of the child nodes, so the parent node initiates a transaction to prevent congestion.
Y. Ha, S.-H. Chung: Enhanced 6P Transaction Methods for Industrial 6TiSCH Wireless Networks

1) CELL SOLICITATION
SFs periodically measure traffic load to calculate cell requirements. Unlike initiating a 6P transaction based on the cell requirement, in the parent-initiated 6P transaction, it is necessary to send its own cell requirement to the parent node. For this purpose, an EB that is periodically transmitted is used. Each node can transmit information regarding itself to its neighbors using various IEs, as presented in Table 1. In this study, we defined Cell Solicitation IE, so that each node transmits the short address of the parent (2 bytes) and cell requirement (1 byte). After synchronization, to participate in the network, the node determines the parent node by receiving the DIO. Then, EB, including the Cell Solicitation IE, should be transmitted to inform the parent node of the cell requirement. However, because the node has not completed the joining process, TSCH Synchronization IE and TSCH Slotframe and Link IE should be excluded from the EB so that other nodes do not participate in the network through this node.

The parent node that receives the EB that includes the Cell Solicitation IE can proceed to the cell offering for the child node. Each node does not issue an additional EB for cell solicitation, and there is little additional overhead because only a small Cell Solicitation IE is added.

However, EB transmitted through a minimal cell is vulnerable to collisions and cannot guarantee stable transmission. Therefore, if the cell offering is not provided until the expiration of the three OTF housekeeping periods although the cell solicitation is announced, more aggressive cell solicitation is performed. For this purpose, we define the 6P SOLICIT request format depicted in Fig. 13.

The 6P SOLICIT request specifies its own cell requirement in the NumCells field, and the NumCells value is the sum of the number of cells to be added and the number of cells to be deleted. Because 6P SOLICIT is intended to announce a cell requirement of a child node, the parent node does not increase its sequence number and does not return a response to it. Therefore, 6P SOLICIT does not block the transactions of other nodes, and the parent node can receive it even in the cells lock state. Because 6P SOLICIT is transmitted to the designated parent node of a child node, even if the parent node does not receive Cell solicitation IE, the parent node can recognize the parent selection of the child node and prepare an OFFER for the child node.

Algorithm 1 (A1) describes reverse OTF for parent-initiated 6P transactions. Each node periodically measures traffic load through OTF housekeeping and calculates the cell requirement (A1 line 3). The calculated cell requirement is referred to when an EB packet is generated (A1 line 17–18). The node compares the cell requirement

| FIGURE 12. History of data packet loss due to 6P CLEAR for 20 min. |
| FIGURE 13. 6P SOLICIT Request format. |
Algorithm 2: Cell Offering

Initialization:

Initialization:

\(\text{const defaultOfferPeriod list addList} = \{\}.\)
\(\text{delList} = [\].\) \(\text{var childldx} = 0\)

Procedure: CellOffering()

1. \(\text{if IsCellLocked()} \) then
2. \(\text{return}\)
3. \(\text{var childAddr} \rightarrow \text{GetHighestCellReqChild()}\)
4. \(\text{if childAddr} = \text{NULL} \) then
5. \(\text{(childAddr, childldx)} = \text{GetNextChild(childldx)}\)
6. \(\text{end}\)
7. \(\text{if childAddr} == \text{NULL} \) then
8. \(\text{return}\)
9. \(\text{addList} = \text{GetCandidateAddList()}\)
10. \(\text{delList} = \text{GetCandidateDelList(childAddr)}\)
11. \(\text{SendOffeif(childAddr, addList, delList)}\)
12. \(\text{ClearSolicit(childAddr)}\)
13. \(\text{SetTimer(defaultOfferPeriod / GetNumChild(), CellOffering)}\)

2) CELL OFFERING

The parent node can recognize the cell requirements of the child nodes by receiving the EB or 6P SOLICIT of the child node. The cell offering is performed when 6P SOLICIT is received or when the cell requirement specified in the Cell Solicitation IE of the received EB is not 0. Algorithm 2 (A2) describes the procedure of the cell offering.

When the cell solicitation is received, the cell offering is stopped if the negotiation for the other child nodes is in progress (A2 line 2–4). However, because the cell requirement is recorded, the next cell offering can be provided to the child node. The parent node prioritizes the child with the highest cell requirement (A2 line 5). Experientially, a node with high cell requirements suffers serious packet queue overflow, and the preferential response to the node can improve network stability. In an LNN network, EB and cell solicitation may not be successfully transmitted due to collision and loss. Certain nodes may be alienated due to continuous collisions and loss of cell solicitation. Thus, if there is no cell requirement in all the child nodes, the next child is selected in a round robin manner (A2 line 6–8).

Fig. 14 defines the 6P OFFER request format for cell offering. The packet format is the same as the conventional 6P request format. Because the child nodes have a smaller opportunity than in conventional 6P transactions, the 6P OFFER processes the functions of ADD, DEL, and RELOCATE simultaneously. A parent node selects its available cells and loads them into the CellList to Add field (A2 line 12). The number of available cells is specified in the NumCells field in Fig. 14. In the CellList to Delete field, cells are loaded in the order of lowest usage among negotiated cells for the target child node (A2 line 13). In the same manner as in the 6P RELOCATE request, the length of the CellList to Delete field is calculated by the length of the payload IE header. The generated cell lists are transmitted through the 6P OFFER request, and the cell requirement of the child node is initialized (A2 line 14–15).

\[
\text{OfferPeriod} = \text{DEFAULT}_-\text{PERIOD}/\text{numChild} \quad (6)
\]

The period of the cell offering is calculated by (6), and this ensures DEFAULT_PERIOD for each child (A2 line 16). Periodic cell offering can provide a balanced opportunity (A2 line 6–8), even if the cell solicitation of the child node is lost.

When the child node receives the 6P OFFER request, the OFFER callback defined in reverse OTF (or SF) is called (A1 line 13). The received cell lists are compared with its available cells, and cells to be added (or deleted) are selected as calculated in OTF housekeeping (A1 line 14–15). The result is returned to the 6top layer, and a 6P OFFER response is generated. Fig. 15 defines the format of the 6P OFFER response packet.

6P negotiation is completed by the parent node receiving the 6P OFFER response. These procedures are coordinated through the parent node, enabling stable 6P transactions even under extremely heavy traffic LLNs.

The key point of the proposed method is that the subject of cell allocation initiates a 6P transaction to prevent collisions caused by converged cell requests. Since RPL is used as the default L3 protocol of 6TiSCH, the execution subject of the proposed method becomes the parent in RPL structure. However, the subject of the proposed method does not have to be a parent. Performance improvement can be expected in...
all routing protocols having a converged traffic pattern due to a single sink.

### B. TRANSACTION REVERT

As depicted in Fig. 12, when a schedule inconsistency is handled by the CLEAR command, severe packet loss occurs instantaneously. In this study, we propose `transaction revert` as a local rollback method to effectively handle this issue.

It is difficult to recognize the schedule inconsistency until the next 6P packet is received. Therefore, to implement `transaction revert`, it is necessary to record the transactions performed with the neighbors. Because sequence number errors can occur on both sides, a node must record transactions for each neighbor. The best method is to maintain a transaction history for all neighbors; however, this requires extensive memory. Therefore, we recorded the history of recent $N$ transactions. If $N$ is set too small, recovery performance decreases due to insufficient recording. Conversely, as $N$ increases, more memory is required. We empirically observed that if the value of $N$ is 10, it is a sufficient record for recovery. The node records the address, sequence number, 6P command, and cell list for a successful 6P transaction. When a sequence number mismatch is detected, `transaction revert` can be performed by searching the history and executing the matching entry in reverse.

As in the example in Fig. 16, when a mismatch occurs at the Node A side, the error is recognized through the response of the next 6P transaction ($SeqNum = 78$), and it can return to the previous state ($SeqNum = 77$) through `Transaction Revert`. The sequence number should not be increased when RC_ERR_SEQNUM is returned on the Node B side. If the response returning the error is lost and the sequence number of Node B is updated, Node A cannot resolve the error because it cannot trace the lost sequence number. If Node A cannot find a matching entry, the Node A issues a 6P CLEAR request to clear the schedule.

Fig. 17 depicts the inconsistency at the Node B side. Node A can immediately return to the previous state through `Transaction Revert`. Therefore, Node A does not recognize any inconsistency and completes the transaction smoothly. If Node B cannot find a matching entry, the Node B returns an RC_ERR_SEQNUM so that Node A can attempt to recover.

### VI. EXPERIMENT AND PERFORMANCE EVALUATION

Fig. 18 illustrates the topology of the experiment, comprising one root node and 30 nodes that generate data. The nodes are divided into three tiers, and each tier is composed of 1, 12, and 18 nodes, respectively. As described in Section IV-B, the distance between each tier is 20 cm, and in the absence of an antenna, the root node located at tier 0 and the nodes belonging to tier 2 cannot communicate directly. Communication between adjacent tiers has a 60–95% link quality, which is not proportional to the distance between them. This environment can simulate a random LLN with poor link quality.

To evaluate the `parent-initiated 6P transaction` under various traffic loads, the performance was measured by varying...
the number of nodes and the number of packets per second (pps) for each node. When each node is synchronized and at least one negotiated cell is allocated, packets are generated at a predetermined cycle. If a problem occurs in the routing or link schedule, the data packet may not be transmitted, and the packet queue may overflow. To prevent this problem, each node was limited to generating up to three data packets in the packet queue. However, because this case also corresponds to a transmission failure, the performance statistics considered that the packet was generated but failed transmission.

OTF housekeeping and DEFAULT_PERIOD were set to 15 s, and the size of the 6P transaction history buffer was set to 10. For cell allocation, 6P uses PB autonomous OpenWSN REL-1.24, RB autonomous MSF, and parent-initiated 6P transaction (PI) using autonomous RB.

A. CHANGING THE NUMBER OF NODES

First, we explain the performance change according to the number of nodes. Each node generates traffic of 0.5 pps/node, and the performance is measured for the cases of 10, 20, and 30 nodes. The number of nodes per tier in each experiment is (1, 4, 6), (1, 8, 12), and (1, 12, 18).

Fig. 19 depicts the TSR of 6P packets. In the case of 10 nodes with considerably light traffic load, all methods result in an average performance of 97%. However, the performance of PB and RB gradually decreases as the traffic load increases. In the case of PB and RB, as TSR decreases, the probability of schedule inconsistency increases; thus demonstrating a slightly better performance in the case where Transaction Revert (RVT) is used. In contrast, in the case of PI, the result reveals a constantly high TSR regardless of the increase in traffic load because it involves coordinated transactions under parent nodes. Additionally, because the probability of schedule inconsistency is significantly reduced, no difference in performance is observed due to RVT.

Fig. 20 depicts the 6P transaction delay. The decrease in TSR due to the traffic load causes a slight increment in the delay in PB and RB. This may be attributable to the 6P packet loss due to collisions and lossy links. In contrast, PI maintains a delay of 2 s on average. Moreover, RVT does not appear to have any effect in terms of latency.

Fig. 21 depicts the number of 6P packets issued for cell negotiation. PI performs cell offering periodically even if there is no cell solicitation by the child; therefore, a larger number of 6P packets are issued in the case where the traffic load is light. In all the methods, as the traffic load increases, a larger number of 6P packets are issued. The empty area in the graph of PI implies 6P SOLICIT, and the red line indicates the middle value of it. 6P SOLICIT does not require a response and does not cause block the other transactions, and thus has a much lower overhead than the other types of packets. Therefore, in the case of 30 nodes, PI exhibits a slightly lower overhead than other techniques.
Fig. 22 depicts the number of failed 6P transactions. The failed 6P transactions can be categorized into loss and 6P error, and the loss can be further classified into timeout and drop. If the request is lost, the packet is dropped, and if the response is lost, a timeout occurs. The colored area of the graph indicates packet loss, and the empty area indicates 6P error. In the case where the traffic load is light, transaction failure mainly occurs due to packet loss; however, as the load increases, the weight of the 6P error increases. Loss of response keeps the child node waiting for a response and increases the duration of cells lock. Because RB and PI use different autonomous cells for the request and response, response packets can be transmitted more reliably. This reduces the cells lock period and significantly reduces the occurrence of RC_ERR_BUSY.

Fig. 23 depicts the PDR of the data packet of each method. PI exhibits a slightly higher PDR in comparison with other methods and demonstrates almost uniform performance in all cases. However, in the case of 30 nodes, the other methods demonstrate that the PDR decreases slightly with TSR.

In the experiment, each node generates packets at regular intervals, it is possible to calculate the operating ratio through the total number of packets generated during the operation time of the experiment. This operating ratio increases as the network is built faster and the operation becomes stable. Fig. 25 depicts the operating ratio of each method. PI exhibits a 10% lower performance than PB and RB under light traffic loads. This is because nodes using PI must wait for the cell offering of the parent node, which is more disadvantageous with a small number of neighbors. In contrast, as the traffic load and the number of neighbors increase, the operating ratio of PI does not differ significantly from that in other techniques.

The traffic load that occurs in the above experiment is somewhat insufficient to be referred to as heavy traffic. In fact, the traffic load does not significantly affect the performance of each node. We created a larger number of packets in a network of 30 nodes and demonstrated the performance of each technique.

**B. CHANGING PACKET PERIOD**

Following the previous experiment, to evaluate the performance under higher traffic loads, the performance of each method was evaluated by changing the packet generation period to 0.5, 1, and 2 pps per node in the topology consisting of 30 nodes, as depicted in Fig. 18.

Fig. 26 depicts the TSR of a 6P packet. In the 6TiSCH network, a larger number of 6P packets are issued than in...
the previous experiment to handle the heavier traffic load. The PB and RB exhibit a significantly lower TSR as the traffic load increases, and finally achieve a success rate of less than 50%. In contrast, PI ensures stable transactions even under a heavy traffic load through the coordination of the parent node, and no difference in performance is observed from that in the light traffic load environment in the previous experiment.

Fig. 27 depicts the 6P transaction delay. Numerically, there is no noticeable change in comparison with the performance in the light traffic load environment in the previous experiment. However, in all cases, PI exhibits a lower latency than those of PB and RB, and guarantees a predictable range of latency regardless of the traffic load.

Fig. 28 depicts the number of 6P packets issued for 6P negotiation. PB and RB generate more than twice the number of 6P packets as the traffic load increases. In the case of PI, the PDR of the cell solicitation packet decreases as the traffic load increases, indicating that many 6P SOLICITs have occurred. However, 6P SOLICIT has a relatively small overhead, and the number of actual 6P transactions of PI is maintained at a similar level despite an increase in traffic load.

Fig. 29 depicts the number of failed 6P transactions. As the traffic load increases, the total number of 6P transaction failures increases, and the ratio of 6P errors increases gradually. Additionally, both the number and loss rate of the 6P packets issued under heavy traffic load increase. Timeout caused by loss of response causes numerous 6P errors due to cells lock. In other words, 6P packet loss in heavy traffic load LLNs causes indirect 6P errors, further reducing network performance.

Fig. 30 depicts the PDR of a data packet. All methods suffer a decrease in performance as the traffic load increases. However, PI is relatively small in width and establishes a stable network even under heavy traffic loads. In a heavy traffic network, the RB exhibits a slightly more stable 6P transaction performance and a better data packet PDR than PB.

Fig. 31 depicts the radio duty cycle of each method. PI consumes a lower energy in comparison with the other methods, although it has an additional overhead for cell offering.
**C. INFLUENCE OF ANTENNA**

The previous experiments were performed by intentionally lowering the link quality by removing the antenna to simulate an LLN industrial network. In this study, to evaluate the side-effects of antenna removal and the effect of stable links on the performance of the 6P transaction and the proposed method, a separate experiment was conducted with 20 nodes equipped with antennas. Because the number of nodes was reduced, the packet generation period was further decreased for the traffic load (i.e., 2–3.33 pps/node).

Fig. 33, 34, and 35 depict the TSR, data packet PDR, and radio duty cycle of a network of 20 nodes equipped with antennas, respectively. The performance of each method demonstrates a similar trend as a result of simulating LLNs without antennas. As the traffic load increases, the performance degradation of RB increases. Additionally, it is necessary to consider the reduction in efficiency of PI due to the reduction in the number of nodes in comparison with the previous experiment and that stable link quality is provided. This result indicates that PI can exhibit improved performance not only in harsh network environments but also in stable environments.

**VII. CONCLUSION**

Considering the IIoT service, in which a large number of devices are connected generating a heavy traffic load, studies on the 6TiSCH network need to assume a harsher network environment. In this study, we focused on the performance of 6P transaction, a fundamental technique of 6TiSCH networks. The 6P transaction is the basis for providing reliability to the IEEE802.15.4 TSCH network, and therefore, it is necessary to perform a strict performance verification under the assumption of a harsh environment.

We analyzed the problems of 6P transactions that may occur in LLNs in heavy traffic environments using the OpenWSN platform, which is conducting leading research for 6TiSCH networks, and proposed enhanced 6P methods for improvement.
Parent-initiated 6P transaction, which can provide stable performance even in harsh networks, and transaction revert, which can locally recover the 6P errors, were implemented in the OpenWSN platform, and we evaluated the 6P method in various traffic load environments through extensive experiments. The experimental results demonstrated that the proposed methods could achieve a 6P transaction success rate of more than 100% and a data packet PDR of up to 22% in comparison with the previous methods.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Comput. Netw., vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] HART. Highway Addressable Remote Transducer, a Group of Specifications for Industrial Process and Control Devices Administered by the Hart Foundation. Accessed: Sep. 23, 2020. [Online]. Available: http://www.hartcomm.org

[3] Wireless Systems for Industrial Automation: Process Control and Related Applications, Standard ANSI/ISA-100.11 a-2011, 2011. [Online]. Available: https://www.isa.org/products/ansi-isa-100-11a-2011-systems-for-industry.

[4] T. Watteyne, A. Mehta, and K. Pister, “Reliability through frequency diversity: Why channel hopping makes sense,” in Proc. 6th ACM Symp. Perform. Eval. Wireless Ad Hoc, Sensor, Ubiquitous Netw., 2009, pp. 116–123.

[5] D. Dujovne, T. Watteyne, X. Vilajosana, and F. Thubert, “6TiSCH: Deterministic IP-enabled Industrial Internet (of things),” IEEE Commun. Mag., vol. 52, no. 5, pp. 36–41, Dec. 2014.

[6] IEEE Standard for Local and Metropolitan Area Networks-Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer, IEEE Standard 802.15.4-2015, Oct. 2015.

[7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, Internet Engineering Task Force Std., document RFC4944, Sep. 2007.

[8] J. Hou, Y. G. Hong, and X. Tang, Transmission of IPv6 Packets over PLC Networks, Internet Engineering Task Force Std. draft-hou-6lo-plc03 [Work-in-Progress], Dec. 2017.

[9] 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, Internet Engineering Task Force Std., document RFC6550, Mar. 2012.

[10] Q. Wang, X. Vilajosana, and T. Watteyne, 6top Protocol (6P), Internet Engineering Task Force Std. document RFC6480, Aug. 2018.

[11] X. Vilajosana, K. Pister, and T. Watteyne, Minimal IPv6 over the TSHC Mode of IEEE 802.15.4e (6TiSCH) Configuration, Internet Engineering Task Force Std. document RFC8180, May 2017.

[12] T. Chang, M. Vaccini, S. Duquennoy, D. Dujovne, and X. Vilajosana, 6TiSCH Minimal Scheduling Function (MSF), Internet Engineering Task Force Std. Standard draft-iert-6tisch-msf-06, Aug. 2019.

[13] X. Vilajosana, T. Watteyne, M. Vaccini, T. Chang, and K. J. Pister, “6TiSCH: Industrial performance for IPv6 Internet-of-Things Networks,” Proc. IEEE, vol. 107, no. 6, pp. 1153–1165, Jun. 2019.

[14] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, “LABEL: Link-based adaptive blacklisting technique for 6TiSCH wireless industrial networks,” in Proc. 26th ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst., pp. 25–33, 2017.

[15] G. Micoli, P. Boccardo, G. Valsecchi, A. Petitti, R. Colella, A. Millella, and L. A. Grieco, “ASAP: A decentralized slot reservation policy for dynamic 6TiSCH networks in industrial IoT,” in Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops), May 2019, pp. 1–6.

[16] Y. Jin, U. Raza, A. Alizai, M. Sooryabandara, and S. Gormus, “Content centric cross-layer scheduling for industrial IoT applications using 6TiSCH,” IEEE Access, vol. 6, pp. 234–244, 2018.

[17] F. Righetti, C. Vallati, S. K. Das, and G. Anastasi, “An experimental evaluation of the 6top protocol for industrial IoT applications,” in Proc. IEEE Symp. Comput. Commun. (ISCC), Jun. 2019, pp. 1–6.

[18] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, “OpenWSN: A standards-based low-power wireless development environment,” Trans. Emerg. Telecommun. Technol., vol. 23, no. 5, pp. 480–493, Aug. 2012.

[19] D. Barthel, J. Vasseur, K. Pister, M. Kim, and N. DeJean, Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks, Internet Engineering Task Force Std. document RFC6551, Mar. 2012.