

Received November 26, 2018, accepted December 9, 2018, date of publication December 14, 2018, date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886375

# Frame-Type-Aware Static Time Slotted Channel Hopping Scheduling Scheme for Large-Scale Smart Metering Networks

HUIUNG PARK<sup>1</sup>, HAEYONG KIM<sup>2</sup>, KYEONG TAE KIM<sup>2</sup>, SEON-TAE KIM<sup>2</sup>, AND PYEONGSOO MAH<sup>1,2</sup>

<sup>1</sup>Korea University of Science and Technology, Daejeon, South Korea

<sup>2</sup>Electronics and Telecommunication Research Institute, Daejeon, South Korea

Corresponding author: Pyeongsoo Mah (pmah@etri.re.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) Grant funded by the Korean government (MSIP) (Development of Integrated Development Solution Supporting Low-Power OS for Lightweight Embedded Devices) under Grant 1711042605, and in part by the Korea Evaluation Institute of Industrial Technology (KEIT) Grant funded by the Korea government (MOTIE) (Development of Wearable Development Toolkit for Various Wearable Application Services) under Grant 10065738.

**ABSTRACT** Time-slotted channel hopping (TSCH) medium access control is a promising technology for the construction of reliable large-scale smart metering networks. However, the existing TSCH scheduling methods do not meet the requirements of large-scale smart metering applications. In particular, link throughput limits exist, which yield packet latency and buffer overflows. In this paper, we propose a static TSCH scheduling scheme that permits all nodes in the TSCH network to transmit or receive frames in any slot. To reduce network control message collisions, we define the broadcast slots and unicast slots individually. To assess the performance of the proposed TSCH scheduling scheme, an evaluation is performed in a real-world testbed. The proposed scheduling scheme achieves a high packet delivery ratio (PDR), even in large-scale and densely deployed networks. In most scenarios, the reliability required by smart metering services is achieved. In a 100-node network, in particular, the proposed scheduling method achieves a PDR exceeding 99%, even when 350-byte packets are collected every 60 s. The scheme and results reported in this study have potential application as guidelines for implementation of large-scale TSCH-based smart metering networks.

**INDEX TERMS** Internet of things (IoT), smart metering networks, time slotted channel hopping (TSCH) scheduling, wireless sensor networks.

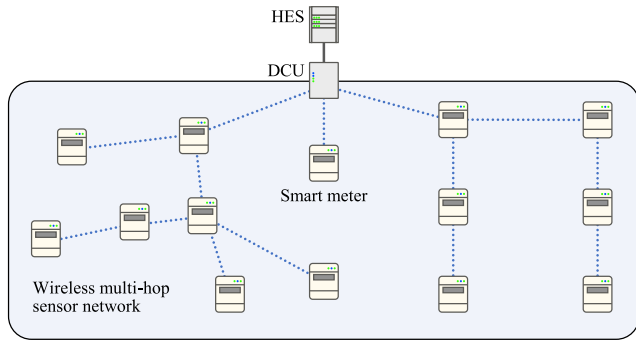
## I. INTRODUCTION

Implementing an advanced metering infrastructure (AMI) consisting of thousands of smart meters is very challenging. For example, in a previous case in Norway, distribution system operators required an AMI solution that could manage 780,000 smart meters [1]. Such smart meters are connected to smart metering networks [2] and wirelessly transmit meter data, as in the example shown in Fig. 1. The success of a massive deployment project such as that implemented in Norway depends on the scalability and reliability of the smart metering network.

The Time Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4 [3], which is inspired by technology used in industrial wireless networks [4], [5], is a promising Medium Access Control (MAC) scheme for construction of

reliable large-scale smart metering networks [6]. TSCH minimizes interference from coexisting networks and communication failures caused by multi-path fading [7]. Thus, TSCH increases node connectivity, network efficiency, and stability [8]. However, the MAC duty cycle is often limited by government radio regulations, which stipulate that wireless devices should not occupy a given channel for longer than a certain time period. TSCH readily meets this duty cycle requirement through use of channel hopping. In a TSCH network, each node's transmission/reception time and channel are managed by the TSCH schedule. However, the TSCH mode of the IEEE 802.15.4 specification does not specify the TSCH schedule construction.

In previous works, various TSCH scheduling schemes have been proposed to reduce collisions and maximize channel



**FIGURE 1.** Example of smart metering network consisting of data concentrator unit (DCU) and smart meters. The DCU, which is the gateway of a smart metering network, collects the sensed data from the surrounding smart meters and sends it to the head end system (HES).

utilization [9]–[14]. In those studies, however, the performance impact of large-size packet transmission was not considered. Thus, the packets used in the corresponding experiments were only a few tens of bytes in size. In smart metering networks, the metered data that must be delivered by a smart meter to a DCU can exceed hundreds of bytes [15]. Packets containing such a large amount of data cannot be encapsulated in a single data frame. Therefore, such packets are split into multiple fragments and then transmitted [16]. However, this fragmentation is known to cause high packet delay [17] and a low packet delivery ratio (PDR) [18] in low-power wireless communication networks and, as a result, this problem has been confronted in previous research. In TSCH-based networks, packet fragmentation may have a more severe impact on the network performance than in other networks, because of the buffer delay. To the best of our knowledge, existing TSCH scheduling schemes have not addressed the problems induced by such large data transfers and fragmentation.

Another problem that has been insufficiently addressed in the literature is collision of broadcast frames in dense networks. Enhanced beacons (EBs) [3] or network control messages (i.e., regarding routing information [19], router advertisement, and router solicitation [20]) are transmitted

in broadcast frames. When a sender transmits a broadcast frame, every neighbor should be able to receive the frame. In the past, common shared slots have been used for this purpose [14], [21]. In a common shared slot, any node can send/receive a frame on the same channel and in the same slot. In congested networks, however, the probability of packet collision in a common shared slot is high; thus, network control messages may not be correctly exchanged, which may make network operations difficult.

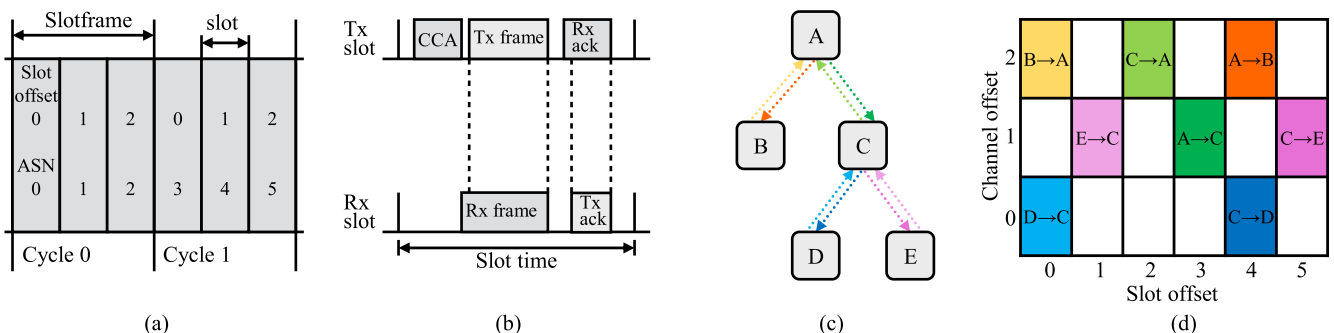
In this paper, we propose a frame-type-aware static TSCH scheduling scheme that maximizes the link throughput of unicast frames and minimizes packet collision in broadcasting. We introduce a fragmentation-aware TSCH frame buffer and simple transmission decision mechanism for effective buffer management, even in environments where large-size unicast packets are generated in large quantities. To verify the performance of our proposed scheme, evaluation and comparison with a state-of-the-art TSCH scheduling method [14] are conducted in a real-world testbed.

The remainder of this paper is organized as follows. In Section II, basic information on the TSCH MAC and TSCH scheduling techniques is presented. The proposed TSCH scheduling scheme is described in Section III. The experiment setup for evaluating the proposed TSCH scheduling scheme is detailed in Section IV, and the results are given in Section V. In Section VI, the suitability of the proposed scheme for smart metering is described. Conclusions are provided in Section VII.

## II. BACKGROUND

### A. TSCH MAC

TSCH is a MAC layer technology that uses time-synchronized slotted access and channel hopping. For slotted access, TSCH MAC employs a periodic slotframe structure consisting of a series of slots, as shown in Fig. 2(a). A node in a TSCH network communicates with others in its scheduled slots; i.e., the Tx and Rx slots. In detail, a node transmits a data frame during a Tx slot, as shown in Fig. 2(b). Immediately before data transmission, the sender performs a clear



**FIGURE 2.** Time Slotted Channel Hopping (TSCH) Medium Access Control (MAC) and scheduling. (a) Example of three-slot slotframe. (b) Data transmission sequence between sender and receiver, whose slots are scheduled to Tx and Rx slots, respectively (CCA: clear channel assessment). (c) Illustration of multi-hop network; nodes and links are indicated by rounded boxes and arrows, respectively. (d) Table describing all TSCH schedules in network. A TSCH schedule contains information on the link, slot offset, and channel offset. Each TSCH schedule is visualized as a cell in the table, and the color of each cell corresponds to the link with the same color in (c). Node D can transmit a frame to node C simultaneously when node B transmits a frame to node A, even if they are in interference range, because these node pairs are communicating on different channels.

channel assessment (CCA) to avoid contention from other networks. After transmission, the sender awaits acknowledgment from the receiver to confirm successful transmission. If the transmission fails, the sender retransmits the frame in the next available slot via the TSCH carrier-sense multiple access with collision avoidance (CSMA/CA) mechanism.

For channel hopping, TSCH employs a hopping sequence that is a pseudo-random sequence of channels. At the beginning of a scheduled slot, a node sets its channel as follows:

$$C = HS[(ASN + \text{channeloffset}) \bmod N_{\text{channels}}], \quad (1)$$

where  $C$  is the frequency channel,  $HS$  is the hopping sequence, the absolute slot number ( $ASN$ ) is the accumulated number of slots from the beginning of the network,  $\text{channeloffset}$  is the value obtained from the TSCH schedule, and  $N_{\text{channels}}$  is the number of channels used in the network. As  $ASN$  increases in each slot, the channel of the node changes continuously, as apparent from (1).

## B. TSCH SCHEDULING METHODS

A TSCH schedule determines which node communicates at which slot and the channel offset. Previous studies on TSCH scheduling have targeted collision minimization and channel utilization maximization, as shown in Figs. 2(c) and 2(d). These studies can be categorized as considering either centralized or decentralized methods [7].

In centralized methods, a coordinator node manages the TSCH schedules for all nodes in the network. The coordinator collects network information (e.g., topology and traffic information [9], [10]). With this information, the coordinator creates an individual TSCH schedule for each node; then, the schedules are distributed to each designated node. This process is repeated every time the topology or traffic change, making application of this technique to a dynamic environment difficult. Moreover, it is difficult to calculate optimal TSCH schedules in large-scale and dense networks.

For large-scale networks, decentralized methods are more appropriate. In general distributed techniques, each node manages its TSCH schedule through negotiation with its neighbors.

The decentralized traffic-aware scheduling scheme (DeTAS) is one such approach employing a decentralized method [11]. DeTAS includes a traffic information collection mechanism in which nodes estimate the number of upward packets according to the number of their descendant nodes, which transmit information to their parents. This scheme also includes a mechanism for allocating uplink slots between the parent and child nodes based on the collected traffic information. DeTAS is a collision-free scheduling algorithm that maximizes channel utilization. However, DeTAS schedules uplink slots for static upward traffic only. Moreover, all nodes in the network must reschedule the uplink slots whenever the network topology or traffic changes.

The On-the-Fly (OTF) scheduling scheme has also been proposed [12]. OTF adjusts the number of Tx slots allocated to each neighbor based on the estimated volume of

outgoing traffic. The format and slot negotiation procedures defined by Internet Protocol version 6 (IPv6) over the TSCH mode of the IEEE 802.15.4e (6TiSCH) working group [22] are implemented in OTF, which can be applied for both uplink and downlink scheduling. However, the OTF performance can be problematic in a dense network, as the nodes may try to provision their slots in a “greedy” manner. Thus, some nodes may fail to allocate a sufficient number of slots, causing them to continue to request additional slots. This scenario reduces the network packet delivery ratio (PDR) and increases the OTF negotiation traffic.

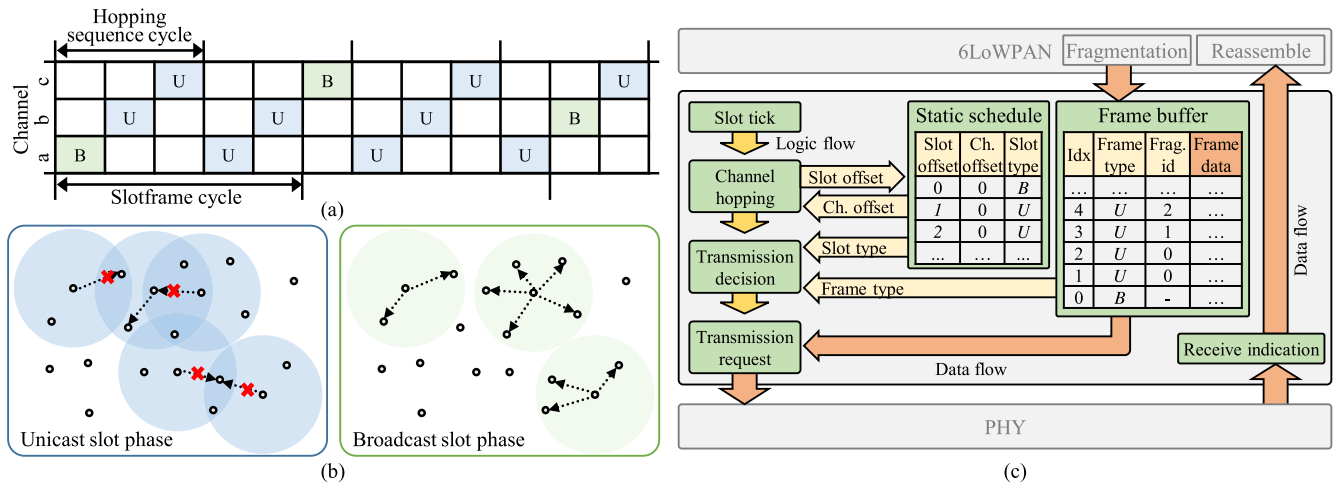
Decentralized adaptive multi-hop scheduling for 6TiSCH networks (DeAMON) has also been proposed [13]. DeAMON sequentially schedules uplinks so that upward traffic can reach the root node as quickly as possible. This scheme employs a routing-assisted rescheduling mechanism to adapt to topology changes. However, DeAMON does not consider downward traffic.

Finally, a unique autonomous scheduling scheme called Orchestra has been developed [14]. Orchestra is a decentralized scheduling scheme without negotiation that autonomously assigns slots to neighbor nodes based on the MAC address and routing protocol for low-power and lossy networks (RPL) [19]. Therefore, there is no network overhead for negotiation packets, which is an essential condition for the methods implemented in [11]–[13]. The assigned slots are automatically rescheduled whenever a change occurs in the routing topology. Orchestra also has the advantage of handling both upward and downward traffic. This is accomplished through use of two scheduling types: sender- and receiver-based. In sender-based scheduling, a node creates a unique Tx slot based on its MAC address, while each neighbor creates an Rx slot based on the node’s MAC address. Thus, each node has a unique Tx slot and multiple Rx slots. In receiver-based scheduling, a node creates a Tx slot based on a neighbor’s MAC address. In that case, each node has a unique Rx slot and multiple Tx slots. In the evaluations performed in [14], the sender-based scheduling scheme of Orchestra achieved a superior PDR to the receiver-based approach, because collision-free scheduling was implemented. Furthermore, the sender-based scheduling scheme yielded a maximum PDR of 0.99997. However, Orchestra does not consider the impact of fragmentation, similar to all other scheduling schemes mentioned above.

## C. PACKET LATENCY

In the smart metering industry, there is relatively minimal interest in latency because a smart meter senses data 4 to 24 times per day. Therefore, the packet latency required by applications is from several seconds to several hours [15]. Nevertheless, the causes of packet latency may relate to the TSCH network reliability. To illustrate this effect, we must first analyze the causes of packet latency in a TSCH network based on the link throughput and buffer latency.

The link throughput is the maximum number of frames per unit time that one node can send to a neighbor. Slots in a



**FIGURE 3.** Overview of static TSCH scheduling scheme. (a) Static TSCH schedule comprised of one shared broadcast slot (B) and four shared unicast slots (U). The channel of the same slot offset differs with each cycle according to the hopping sequence. (b) Difference between unicast and broadcast slot phases. In a denser network or when the nodes transmit large packets, collisions occur more frequently. In this scheme, in the unicast slot phase, the nodes transmit unicast frames only, which are mostly packet fragments containing sensor data. In the broadcast slot phase, the nodes transmit broadcast frames only, which are mostly network control messages. (c) Architecture of static TSCH scheduling MAC.

slotframe can be scheduled to different destinations; therefore, the link throughput may vary for each destination. The link throughput from node  $i$  to neighbor node  $j$  can be expressed as

$$Q_t(i, j) = \frac{s(i, j)}{S \cdot t} \tag{2}$$

where  $s(i, j)$  is the number of Tx slots in which node  $i$  can transmit to destination neighbor  $j$ ,  $S$  is the slotframe length, and  $t$  is the slot time. According to (2), the uplink throughput of each node in the Orchestra-based TSCH network proposed in [14] is only 1 frame/s, if  $S = 100$  and  $t = 10$  ms. The reason for this low throughput is that Orchestra assigns only one Tx slot to each node. The low throughput of Orchestra can cause buffer latency and very high packet latency in an unbalanced traffic environment, as explained in [23].

The buffer latency is the delay time from when a frame is placed in the frame buffer to when the frame is successfully transmitted. TSCH MAC transmits the frames stored in the frame buffer in a first-in-first-out (FIFO) manner. A frame newly inserted in the buffer awaits its turn until all frames in the buffer have been transmitted. If the traffic on a given node is higher than the node throughput, the buffer latency becomes longer and, eventually, buffer overflow occurs. Therefore, buffer overflow is more likely to occur at relay nodes.

**D. FRAGMENTATION AND TRAFFIC**

In IPv6-enabled networks, a node may fragment an IPv6 packet into multiple frames for transmission when the packet is too large to be transmitted in one frame [16]. All these frames must be forwarded to the next hop for successful packet forwarding. Therefore, this fragmentation increases the traffic and packet latency. In particular, if the link throughput from the sender to receiver is low, the packet latency of a fragmented packet may be significantly longer.

**III. PROPOSED TSCH SCHEDULING SCHEME**

**A. OVERVIEW**

We propose a static TSCH scheduling scheme that minimizes network performance degradation in a large-scale smart metering environment. This scheduling scheme is based on two key concepts:

- 1) As frame-type-aware static scheduling is implemented, the proposed scheme minimizes broadcast collisions and maximizes link throughput for unicast frames;
- 2) The proposed scheme employs a fragmentation-aware TSCH frame buffer and a simple transmission decision mechanism.

**B. FRAME-TYPE-AWARE STATIC SCHEDULE**

Broadcast frame transmission can be problematic because collisions are inevitable in a common shared slot. In environments where collisions are frequent, broadcast-based network control messages are exchanged incorrectly; thus, network configuration or operation is difficult.

To overcome this problem, we introduce broadcast slots and unicast slots. A broadcast slot is a common shared slot that transmits broadcast frames only, while a unicast slot is a common shared slot for unicast frame transmission that employs a back-off retransmission mechanism [24] to increase the frame reception ratio (FRR). By separating the Tx slots of these two frame types, broadcast frame collisions are reduced, as shown in Fig. 3(b). In the proposed scheme, to maximize unicast throughput and minimize buffer latency, a slotframe consisting of one slot designated as the broadcast slot is employed, with the remaining slots assigned as unicast slots. That is, the slotframe has a common schedule consisting of common shared slots only. As a result, as soon as a node connects to a network under this scheme, it can communicate with all nodes in the vicinity. Fig. 3(a) shows an example of a 5-slot frame-type-aware static schedule.



Propagation of a static schedule is performed using EBs. A node that is already joined to a TSCH network can transmit an EB that contains slotframe information and a static TSCH schedule. A node joining the TSCH network utilizes an EB received from a neighboring node. The joining node generates the same slotframe and TSCH schedule using the slotframe information and TSCH schedule obtained from the EB.

### C. FRAGMENTATION-AWARE TSCH FRAME BUFFER AND SIMPLE TRANSMISSION DECISION MECHANISM

The proposed TSCH scheduling scheme has a MAC structure, as shown in Fig. 3(c). All frame data are stored in a frame buffer before transmission and are transmitted in a FIFO manner. After a packet is fragmented at the IPv6 low-power wireless personal area networks (6LoWPAN) layer, the fragments are stored in the frame buffer along with the same fragment group identification (ID). If transmission of one of the fragments fails despite the retransmission mechanism, the packet delivery fails. In this scenario, the remaining fragments having the same group ID are removed from the buffer to reduce unnecessary transmissions. Note that broadcast frames are stored in the frame buffer without a group ID, because the 6LoWPAN layer does not define a broadcast frame fragmentation/reassembly mechanism.

---

#### Algorithm 1 Transmission Procedure Pseudocode

---

```

Function slot_tick()
  update_asn();
  slot_offset ← get_slot_offset();
  channel_offset ← get_channel_offset(slot_offset);
  channel_hopping(channel_offset);
  slot_type ← get_slot_type(slot_offset);
  frame_type ← get_buffer_next_frame_type();
  if slot_type = broadcast then
    if frame_type = broadcast then
      | send_buffer_next_frame();
    end
  else
    if frame_type = unicast then
      | send_buffer_next_frame();
    end
  end
end

```

---

The transmission procedure employed in our scheduling scheme is outlined in Algorithm 1. Each time a slot is initiated, the channel is hopped according to the static schedule. Then, the communication type of the current slot and the frame type of the oldest data inserted in the frame buffer are identified. If the communication and frame types are equivalent (e.g., a unicast slot and unicast frame), and the back-off process is not for retransmission, a frame is generated and transmitted.

### D. SLOTFRAME LENGTH DECISION

The broadcast and unicast slot ratio in the static schedule determines the broadcast frame and unicast frame link throughput. It is important to determine an appropriate ratio because the latency and collision probability of each frame type depend on the link throughput. In the proposed approach, we set all slots as unicast slots, except for one broadcast slot. Therefore, the link throughputs of the unicast frames,  $Q_u$ , and broadcast frames,  $Q_b$ , can be expressed as

$$Q_u = (S - 1) \cdot S^{-1}, \quad (3)$$

$$Q_b = S^{-1}, \quad (4)$$

respectively, where  $S$  is the slotframe length.

The unicast packet latency is affected by the link throughput. Here, we introduce a packet latency model considering IPv6 fragmentation and multi-hops. If no collisions occur and there are no packets waiting in the buffer for all nodes in the network, the ideal packet latency can be obtained. For a given routing path  $P = \langle v_1, \dots, v_H \rangle$ , the minimum packet latency  $T$  required to transmit one IP packet from  $v_1$  to  $v_n$  is

$$T_d(P) = \sum_{i=1}^{H-1} \frac{F \cdot t}{T(v_i, v_{i+1})}, \quad (5)$$

where  $F$  is the number of packet fragments. Based on this formula, the minimum packet latency in the network with the proposed scheduling scheme can be described according to the hop distance  $H$ . Thus,

$$T_d = H \cdot F \cdot \frac{S}{S-1} \cdot t. \quad (6)$$

The throughput should have the ability to handle a predicted traffic load. The broadcast traffic load on a broadcast slot of a node  $n_j$  can be expressed as

$$B_{load}(n_j) = S \cdot B_{traffic}(n_j), \quad (7)$$

where  $B_{traffic}(n_j)$  is the average number of broadcast frames generated in the slot time. If  $B_{load}(n_j) \leq 1$ ,  $B_{load}(n_j)$  can be regarded as the transmission probability in a given broadcast slot,  $P_{broadcastTx}(n_j)$ . If more than two nodes in the network transmit broadcast frames in the same broadcast slot, a collision may occur. Therefore, the probability of broadcast frame collision,  $P_{broadcastCol}$ , can be expressed as

$$1 - \frac{1}{|N|} \sum_{n_i \in N} \prod_{n_j \in N, n_j \neq n_i} \{1 - P_{broadcastTx}(n_j)\}, \quad (8)$$

where  $N$  is the set of nodes in the network. Likewise, the unicast traffic load on a unicast slot of node  $n_j$  can be expressed as

$$U_{load}(n_j) = \frac{S}{S-1} \cdot U_{traffic}(n_j), \quad (9)$$

where  $U_{traffic}(n_j)$  is the average number of unicast frames generated in the slot time. The probability of unicast frame collision,  $P_{unicastCol}$ , can be expressed as

$$1 - \frac{1}{|N|} \sum_{n_i \in N} \prod_{n_j \in N, n_j \neq n_i} \{1 - P_{unicastTx}(n_j)\}, \quad (10)$$

where  $P_{unicastTx}(n_j)$  is the probability of transmission in a unicast slot of  $n_j$ .

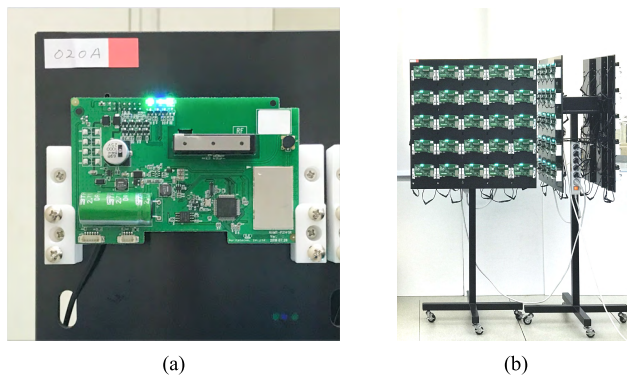
Note that the retransmission traffic is not considered in (9) and (10). Furthermore, (7) and (9) indicate that a trade-off exists between the broadcast frame load and unicast frame load, while (8) and (10) show the trade-off relationship between  $P_{broadcastCol}$  and  $P_{unicastcastCol}$ . Longer  $S$  corresponds to higher  $P_{broadcastCol}$  and lower  $P_{unicastcastCol}$ . Conversely, shorter  $S$  corresponds to lower  $P_{broadcastCol}$  and higher  $P_{unicastcastCol}$ . Thus, selection of an appropriate value of  $S$  is important. The value of  $S$  should be largest under the condition that  $P_{broadcastCol}$  is below the implementation specific threshold. The procedure for selecting the value of  $S$  is described in Section IV-C. This method not only maintains network stability through smooth delivery of network control packets, but also maximizes the data packet throughput.

## IV. EXPERIMENTAL SETUP

### A. PLATFORM AND OPERATING SYSTEM

An experiment was performed to evaluate the performance of the proposed scheme. In the experiment, we employed a radio frequency (RF) modem produced for real smart meters. The modem was powered by an STM32F412RG microcontroller (MCU), featuring a 32-bit ARM Cortex M4 core and having 256 KB of static random access memory (SRAM) and 1 MB of internal flash memory. For wireless communication, a CC1200 [25] sub-Giga RF chip was contained in the modem.

We implemented TSCH MAC on a NanoQplus operating system [26], [27]. NanoQplus is a lightweight operating system for embedded devices and supports the full IP stack, including 802.15.4 physical layer (PHY)/MAC, 6LoWPAN [16], [20], [28], IPv6, the user datagram protocol (UDP), RPL [19], and the constrained application protocol (CoAP) [29]. Thus, we could compare it to the classic CSMA/CA MAC [30].



**FIGURE 4.** Platform and testbed setup. (a) Radio frequency (RF) modem. (b) Two jigs with 100 modems attached. The modems were attached to both the front and back sides of the jig with a  $5 \times 5$  grid (each jig held a maximum of 50 modems).

### B. TESTBEDS

We prepared testbeds at a research center and employed 101 modems. As shown in Fig. 4, 100 modems were attached

**TABLE 1.** Testbed summary.

Testbed	50-node	100-node
Network size	51	101
Avg. hop/Max hop	3/5	3/5
Avg. number of children	2	2
Avg. number of neighbor	25	51

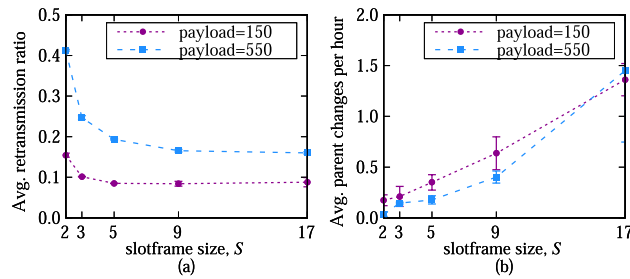
to two jigs placed close together. The remaining modem was the root node and was placed on a table beside the jigs. This setup created a dense and congested communication environment. We implemented 50- and 100-node testbeds based on this setup, so as to measure the scalability. The 50-node case was implemented easily by powering one of the two jigs only. The details of the testbeds are summarized in Table 1.

### C. PHY AND MAC CONFIGURATIONS

We applied the following configurations to the TSCH MAC. The slot time was set to 10 ms and the PHY layer gross bitrate was set to 500 kbps so that the data transmission sequence could be implemented within the given slot time. Four channels were used for channel hopping and the TSCH frame buffer was set to store up to 127 frames. We added a timeout to the TSCH frame buffer, so that each frame did not remain for longer than a certain time; i.e., 32 s in our experiment. Based on this basic TSCH MAC configuration, we implemented two scheduling schemes: the proposed static scheduling scheme and Orchestra [14].

We utilized an EB for time synchronization, which was identical to that used in Orchestra. Thus, it was necessary to derive the slotframe size from the EB traffic. We set the nodes to send EBs every 24 s on average, with a deviation of 8 s in order to distribute the EB transmission time. As a result of the nature of the platform board used in this study, the nodes began unsynchronized by the time drift, if they did not receive EBs from their parent for longer than 1 min. Therefore, it was necessary to set the probability of more than three consecutive EB reception failures to less than 1%. Based on (7), (8), and the given implementation-specific threshold, the value of  $S$  was required satisfy  $(1 - P_{broadcastCol}) \cdot \sum_{i=0}^3 P_{broadcastCol}^i > 0.99$ . The result was  $S < 9.197$ ; thus, we set the slotframe size to 9. We performed a preliminary experiment that proved the slotframe size was appropriate, as shown in Fig. 5. The maximum number of retransmission attempts was seven.

Orchestra was selected as a comparative scheduling scheme as it is autonomous, has no negotiation process, and can handle both upward and downward traffic. Specifically, the sender-based scheduling scheme was selected, as it was reported to have the highest PDR in the literature [14]. To realize collision-free scheduling, the slotframe length was set larger than the network size (i.e., 64 and 127 for the 50- and 100-node testbeds, respectively). The maximum number of retransmissions was set to 1 as no collisions occurred. The CSMA/CA MAC parameters were set to the default values provided in the standards.



**FIGURE 5.** Comparison of (a) average retransmission ratio and (b) parent change rate for different slotframe sizes. A high parent change rate implies that the nodes could not receive the network control message normally.

### D. TOPOLOGY

We used whitelist filtering and the RPL to create a multi-hop wireless sensor network. Through the whitelist filtering, we established 20 nodes per hop distance. Thus, a five-hop network was constructed for each experiment. The routing topology was responsible for the RPL. The parent selection process was based on an objective function, which defined the manner in which the RPL nodes selected and optimized routes. In addition, we set an objective function that selected the candidate node with the shortest hop distance from the root node. If any node found a node at a shorter hop distance than its current parent node, that node was forced to change its parent. As a result, the number of nodes with the same hop distance remained constant in our testbeds. However, this approach did not fix the routing topology. In the topology formation process, it was probable that a node would select a parent from among multiple parent candidates having the same conditions.

### E. SCENARIO

We configured the following types of communication to occur within the network:

- 1) Periodic unicast: sensor data, network control packet;
- 2) Aperiodic unicast: network control packet;
- 3) Periodic broadcast: EB, network control packet;
- 4) Aperiodic broadcast: network control packet.

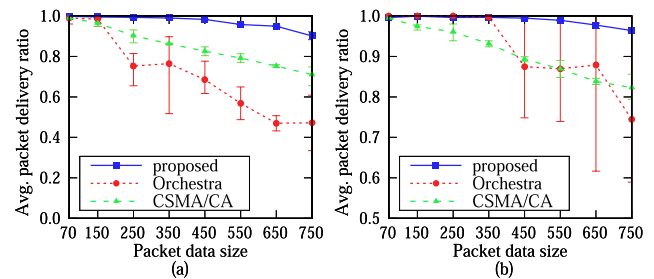
First, every node periodically sent a UDP packet to the root node in 60-s intervals. We measured the performance in scenarios where the UDP packet size payloads were 70, 150, 250, 350, 450, 550, 650, and 750 bytes, respectively. The UDP packet payloads contained information on the network status and performance, e.g., the number of successful and failed transmissions. The root node logged all the information from the packet. The payloads also contained the ASN at the time of packet creation. When the root node received a UDP packet from the node, the root node measured the packet delivery time.

To obtain averaged experimental results, we repeated the experiment at least three times for each scenario. Hence, we could analyze the average performance for the various topologies. In total, the experiments were conducted 308 times. Each experiment required at least 100 min and

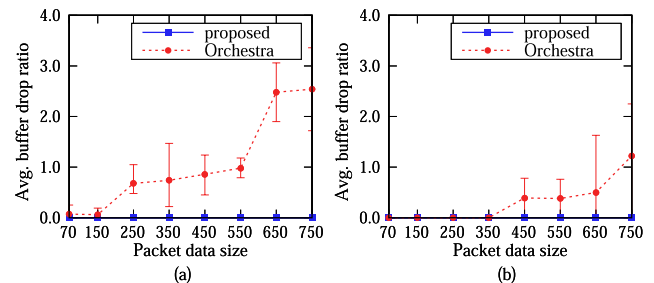
up to 37 days. A total of 147,463,902 unicast frame transmissions were successfully conducted; more than 8,970,905 broadcast frames were sent; and a total of 13,890,699 UDP packets were sent to the root node.

## V. RESULTS

In this section, we present the experimental results for the evaluated methods in terms of reliability, scalability, and latency. First, we compared the reliability of the proposed scheduling scheme with that of Orchestra. Fig. 6 shows a comparison of the PDRs. In the experiments, the proposed scheduling scheme generally exhibited a higher PDR than Orchestra. In the 100-node testbed, the proposed scheduling scheme achieved a PDR exceeding 99% when the packet size was 350 bytes or fewer. In contrast, Orchestra achieved a PDR of less than 90% when the size of the periodic packet exceeded 150 bytes. In the 50-node testbed, the proposed scheduling scheme achieved a PDR exceeding 99% when the packet size corresponded to fewer than 550 bytes. However, Orchestra achieved a PDR of over 90% only when the packet had 350 bytes or fewer. This lower PDR is related to the buffer drop ratio (BDR).



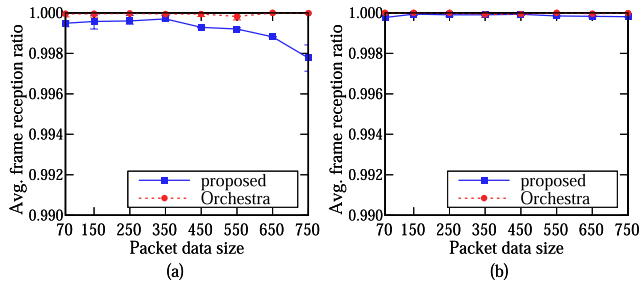
**FIGURE 6.** Comparison of packet delivery ratio (PDR). (a) 100-node testbed. (b) 50-node testbed.



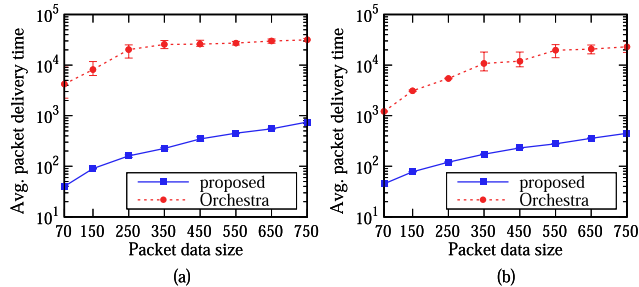
**FIGURE 7.** Comparison of buffer drop ratio (BDR). (a) 100-node testbed. (b) 50-node testbed.

Fig. 7 shows the results for the BDR, which represents the number of frames that dropped from the TSCH buffer per minute. Owing to the limited link throughput of Orchestra, the BDR increases in larger networks and heavier traffic. This is the cause of the low PDR of Orchestra. In the proposed static scheduling scheme, however, a buffer drop barely occurs. Thus, the performance degradation of the proposed scheduling scheme is more closely related to collisions and retransmissions than a buffer drop.

Fig. 8 shows the frame reception ratio (FRR), which represents the link layer reliability and, specifically, the ratio



**FIGURE 8.** Comparison of frame reception ratio (FRR). (a) 100-node testbed. (b) 50-node testbed.



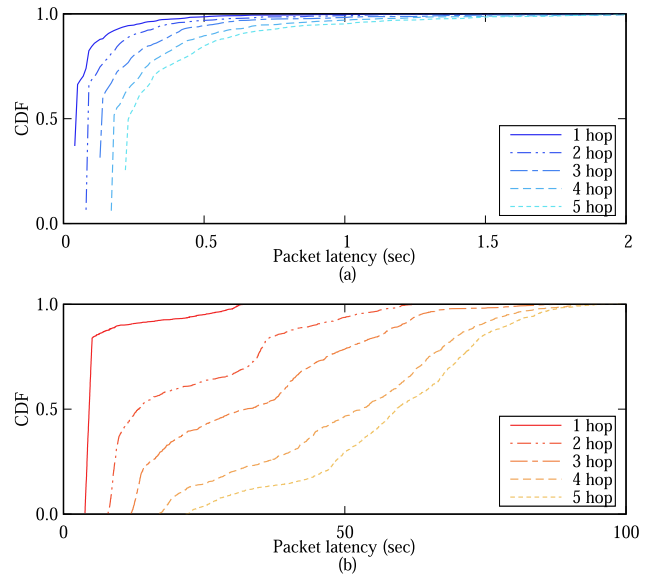
**FIGURE 9.** Comparison of packet latency. (a) 100-node testbed. (b) 50-node testbed.

between the number of frames and the number of successfully transmitted frames. In general, a network with a high FRR exhibits a high PDR. Interestingly, Orchestra exhibited a very high FRR but a very low PDR. In every experiment, Orchestra achieved an FRR of more than 99.99%. Nevertheless, in heavy-traffic scenarios in particular, Orchestra exhibited poor PDR due to the high BDR.

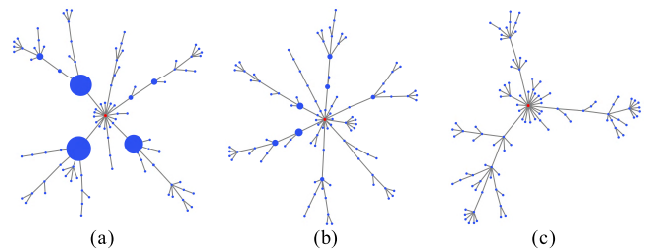
The packet latency is an indicator of the quality of service (QoS). Fig. 9 shows a comparison of the packet latency performance of the two schemes. In all cases, the proposed scheduling scheme delivered packets much more rapidly than Orchestra. In particular, for Orchestra, larger packet size yielded greater latency.

Fig. 10 shows the cumulative distribution functions (CDFs) of the packet latency from the individual experiments. In the case of the proposed scheduling scheme, the theoretical minimum of the packet latency was small. Thus, most of the packets were delivered to the root node within 1 s. On the other hand, Orchestra exhibited limited link throughput. Therefore, the theoretical packet latency was exceptionally high. This increased the packet waiting time in the buffer. In Fig. 10(b), it is clearly apparent that the packet latency of the five-hop nodes was distributed in time.

In addition, we found that Orchestra exhibits a performance drift. In the experiments, the PDR of the Orchestra-based network varied depending on the topology, which was randomly formed for each run, as shown in Fig. 11. In the case shown in Fig. 11(a), seven single-hop nodes had descendant nodes, three of which had a remarkably large number of descendants. The BDR of the three nodes was high because the uplink traffic was much higher than the link throughput. As the majority of the uplink traffic was lost at



**FIGURE 10.** Cumulative distribution functions (CDFs) of 350-byte packet latency in 100-node testbed. (a) Result given by proposed scheme. (b) Orchestra result.



**FIGURE 11.** Visualization of topology and performance of three instances of 350-byte scenario in 100-node testbed. (a) Orchestra with PDR of 51.74%. (b) Orchestra with PDR of 89.71%. (c) Proposed scheduling scheme with PDR of 99.83%. Each dot represents a node. The dot size is proportional to the node BDR. The red dot represents the root node.

the nodes, the PDR of the network was low. In the case of Fig. 11(b), ten one-hop nodes had descendant nodes and the number of their descendants was roughly uniform. Therefore, there were fewer buffer drops than in the previous case. The PDR was better, but still under 90%. Low performance was obtained because buffer drops still occurred in some nodes having a large number of descendants. However, no buffer drop occurred for the proposed scheduling scheme, shown in Fig. 11(c), even in a node that had 34 descendants; this was because the uplink throughput was high owing to the unicast slot. Consequently, the network PDR was high.

## VI. DISCUSSION

From the experimental results, we observed that the reliability and low latency of the proposed static scheduling scheme allowed realization of a scalable smart metering network. The proposed method satisfied the 99% PDR requirement of typical smart metering applications. In addition, the proposed method enabled stable IPv6 packet communication. Note that collision-free TSCH scheduling schemes such as



Orchestra [14] are more suitable for application to networks in which reliability is critical, but traffic is relatively light.

Furthermore, the proposed scheme exhibited low packet latency. This has the advantage of ensuring the QoS of smart metering services. However, some traffic-based scheduling schemes [10], [12], [13] have mechanisms to adjust the slot quota to match the traffic to the node. The current throughput is adjusted based on the past traffic volume in those schemes. However, that method exhibits difficulties in cases involving instantaneous traffic due to user requests at arbitrary times.

Note that the proposed scheduling scheme may have relatively high power consumption in addition to packet retransmission owing to collisions. However, low power consumption is less important in smart metering networks, as smart meters are connected to the power grid, which provides them with a constant power supply. As a consequence, our scheme is more suited to line-powered networks than battery-powered networks, in which the node lifetime will be reduced.

In the experiments performed in this study, we found that finding an appropriate parent node in an environment with more than 100 neighboring nodes is important and generates a new challenge. This issue must be addressed in future work.

## VII. CONCLUSIONS

In this paper, we proposed a frame-type-aware static TSCH scheduling scheme suitable for operating large-scale smart metering networks. Our proposed TSCH scheduling scheme can collect large volumes of data while maintaining the stability of a large network. In our scheduling scheme, all nodes in the TSCH network can transmit or receive frames in any slot. To reduce network control message collisions and maximize the link throughput, the broadcast slots and unicast slots are defined separately. In addition, to address the fragmentation problems, we have developed a fragmentation-aware TSCH MAC structure.

In this study, the performance of the proposed TSCH scheduling scheme was analyzed in a real-world testbed and compared with a high-performance scheduling scheme, i.e., Orchestra. Orchestra exhibited a lower packet delivery ratio due to low link throughput and the resulting buffer overflow. In contrast, our scheduling scheme exhibited higher packet delivery ratios in large-scale and high-traffic networks as it minimized buffer overflow. We expect that our scheme and results can be used as guidelines for implementing large-scale TSCH-based smart metering networks.

## REFERENCES

- [1] Spintelligent. *Tender News: Norwegian DSOs to Put out AMI Tender at EUW14*. [Online]. Available: <https://www.metering.com/norwegian-dsos-to-put-out-ami-tender-at-euw14/>
- [2] B. Lichtensteiger, B. Bjelajac, C. Mueller, and C. Wietfeld, "RF mesh systems for smart metering: System architecture and performance," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, Gaithersburg, MD, USA, Oct. 2010, pp. 379–384.
- [3] *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)—Amendment 1: MAC Sublayer*, IEEE Standard 802.15.4e, 2012.
- [4] *Wireless Systems for Industrial Automation: Process Control and Related Applications*, Standard ISA-100.11a, ISA, 2009.
- [5] J. Song et al., "WirelessHART: Applying wireless technology in real-time industrial process control," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, St. Louis, MO, USA, Apr. 2008, pp. 377–386.
- [6] O. Monnier, "A smarter grid with the Internet of Things," Texas Instrum., Dallas, TX, USA, Tech. Rep., 2013.
- [7] D. De Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Comput. Commun.*, vol. 88, pp. 1–24, Aug. 2016.
- [8] 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. (PE-WASUN)*, 2017, pp. 116–123.
- [9] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks," in *Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sydney, NSW, Australia, Sep. 2012, pp. 327–332.
- [10] M. R. Palattella, N. Accettura, L. A. Grieco, G. Boggia, M. Dohler, and T. Engel, "On optimal scheduling in duty-cycled industrial IoT applications using IEEE802.15.4e TSCH," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3655–3666, Oct. 2013.
- [11] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia, and M. Dohler, "Decentralized traffic aware scheduling in 6TiSCH networks: Design and experimental evaluation," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 455–470, Dec. 2015.
- [12] M. R. Palattella et al., "On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks," *IEEE Sensors J.*, vol. 16, no. 2, pp. 550–560, Jan. 2016.
- [13] A. Aijaz and U. Raza, "DeAMON: A decentralized adaptive multi-hop scheduling protocol for 6TiSCH wireless networks," *IEEE Sensors J.*, vol. 17, no. 20, pp. 6825–6836, Oct. 2017.
- [14] S. Duquenoey, B. A. Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust mesh networks through autonomously scheduled TSCH," in *Proc. SenSys*, Seoul, South Korea, Nov. 2015, pp. 337–350.
- [15] M. Kuzlu, M. Pipattanasomporn, and M. Rahman, "Communication network requirements for major smart grid applications in HAN, NAN and WAN," *Comput. Netw.*, vol. 67, pp. 74–88, Jul. 2014.
- [16] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks*, Standard RFC 4944, IETF, 2007.
- [17] B. Cody-Kenny, D. Guerin, D. Ennis, R. S. Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6LoWPAN protocol on MICAZ and TelosB motes," in *Proc. ACM Workshop Perform. Monit. Meas. Heterogeneous Wireless Wired Netw. (PM2HW2N)*, 2009, pp. 25–30.
- [18] J. Pope and R. Simon, "The impact of packet fragmentation and reassembly in resource constrained wireless networks," *J. Comput. Inf. Technol.*, vol. 21, no. 2, pp. 97–107, 2013.
- [19] T. Winter et al., *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, Standard RFC 6550, IETF, 2012.
- [20] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann, *Neighbor Discovery Optimization for IPv6 Over Low-Power Wireless Personal Area Networks (6LoWPANs)*, Standard RFC 6775, IETF, 2012.
- [21] K. Pister, T. Watteyne, and X. Vilajosana, *Minimal IPv6 Over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*, Standard RFC 8180, IETF, 2017.
- [22] Q. Wang, X. Vilajosana, and T. Watteyne, *6top Protocol (6P)*, document, Internet-Draft draft-ietf-6tisch-6top-protocol-07, IETF, 2017.
- [23] S. Rekik, N. Baccour, M. Jmaiel, and K. Drira, "A performance analysis of Orchestra scheduling for time-slotted channel hopping networks," *Internet Technol. Lett.*, vol. 1, no. 3, p. e4, 2017.
- [24] D. De Guglielmo, B. Al Nahas, S. Duquenoey, T. Voigt, and G. Anastasi, "Analysis and experimental evaluation of IEEE 802.15.4e TSCH CSMA-CA algorithm," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1573–1588, Feb. 2017.
- [25] Texas Instruments. (2013). *CC1200 Low-Power, High-Performance RF Transceiver*. [Online]. Available: <http://www.ti.com/lit/ds/symlink/cc1200.pdf>
- [26] S. Kim, H. Kim, J. Song, M. Yu, and P. Mah, "NanoQplus: A multi-threaded operating system with memory protection mechanism for WSNs," *Proc. 1st China-Korea WSN Workshop*, vol. 20, no. 8, 2008, pp. 1–8.

[27] J. Jeong, J. Kim, and P. Mah, "Design and implementation of low power wireless IPv6 routing for NanoQplus," in *Proc. 13th Int. Conf. Adv. Commun. Technol.*, Seoul, South Korea, Feb. 2011, pp. 966–971.

[28] P. Thubert and J. Hui, *Compression Format for IPv6 Datagrams Over IEEE 802.15.4-Based Networks*, Standard RFC 6282, IETF, Sep. 2011.

[29] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, Standard RFC 7252, IETF, 2014.

[30] C. Wang, T. Jiang, and Q. Zhang, Eds., *ZigBee Network Protocols and Applications*. Boca Raton, FL, USA: CRC Press, 2014.



**KYEONG TAE KIM** received the B.S. degree in computer engineering from Kangwon National University, South Korea, in 2004, and the M.S. degrees in information and communications from the Gwangju Institute of Science and Technology, South Korea, in 2006. Since 2006, he has been a Research Member with the Electronics and Telecommunications Research Institute. His research interests include network security and wireless communication.



**HUIUNG PARK** received the B.S in engineering from Hanyang University, in 2012. He is currently pursuing the Ph.D. degree with the Korea University of Science and Technology. He is one of the developers of the NanoQplus operating system (OS). His research interests are lightweight OS, wireless sensor networks, and the Internet of Things.



**SEON-TAE KIM** received the B.S. degree from the Department of Electrical and Electronics Engineering, KAIST, South Korea, in 1997, the M.S. degree from the Department of Electrical and Electronics Engineering, Seoul National University, South Korea, in 2000, and the Ph.D. degree from the Department of Electrical and Electronics Engineering, Korea University, South Korea, in 2012. In 2000, he joined the Real-Time Multimedia Team, Electronics and Telecommunications Research Institute, South Korea. Since 2011, he has been a Director with the Department of Embedded SW Research. His research interests include video compression, multimedia streaming, image processing, lightweight operating systems, and sensor networks.



**HAEYONG KIM** received the B.S. and M.S. degrees in computer science and engineering from Seoul National University, in 2004 and 2006, respectively. He is currently a researcher with the Electronics and Telecommunications Research Institute, South Korea. He is one of the developers of the NanoQplus operating system. His research interests include lightweight operating systems, embedded software, and sensor networks.



**PYEONGSOO MAH** received the M.S. degree in computer science and engineering from The City University of New York, USA, in 1992, and the Ph.D. degree in computer science and engineering from Wright State University, USA, in 1995. Since 1996, he has been a Principal Researcher with the Electronics and Telecommunications Research Institute. His research interests include lightweight operating systems, embedded software, and IoT systems.

...