Adaptive Success Rate-based Sensor Relocation for IoT Applications

Moonseong Kim[1] and Woochan Lee[2*]
[1] Department of IT Convergence Software, Seoul Theological University
Bucheon 14754, Republic of Korea
[e-mail: moonseong@stu.ac.kr]
[2] Department of Electrical Engineering, Incheon National University
Incheon 22012, Republic of Korea
[e-mail: wlee@inu.ac.kr]
[*]Corresponding author: Woochan Lee

Received January 7, 2021; revised March 13, 2021; accepted March 28, 2021; published September 30, 2021

Abstract

Small-sized IoT wireless sensing devices can be deployed with small aircraft such as drones, and the deployment of mobile IoT devices can be relocated to suit data collection with efficient relocation algorithms. However, the terrain may not be able to predict its shape. Mobile IoT devices suitable for these terrains are hopping devices that can move with jumps. So far, most hopping sensor relocation studies have made the unrealistic assumption that all hopping devices know the overall state of the entire network and each device's current state. Recent work has proposed the most realistic distributed network environment-based relocation algorithms that do not require sharing all information simultaneously. However, since the shortest path-based algorithm performs communication and movement requests with terminals, it is not suitable for an area where the distribution of obstacles is uneven. The proposed scheme applies a simple Monte Carlo method based on relay nodes selection random variables that reflect the obstacle distribution's characteristics to choose the best relay node as reinforcement learning, not specific relay nodes. Using the relay node selection random variable could significantly reduce the generation of additional messages that occur to select the shortest path. This paper's additional contribution is that the world's first distributed environment-based relocation protocol is proposed reflecting real-world physical devices' characteristics through the OMNeT++ simulator. We also reconstruct the three days-long disaster environment, and performance evaluation has been performed by applying the proposed protocol to the simulated real-world environment.

Keywords: Hopping Sensor, Mobile IoT, Reinforcement Learning-based Protocol, Relocation Protocol, Sensory Data Networking, Simulation

This work was supported by Incheon National University Research Grant in 2020(2020-0136).

1. Introduction

Artificial Intelligence (AI) technology is generally focusing on collecting and analyzing vast amounts of data [1]. However, if a problem occurs in collecting some data, it is complicated to find an abnormal point because the data volume is infinitely large [2]. Therefore, in recent years, the technology that can continuously collect data in the observation area attracts attention as a significant issue.

With the development of Internet of Things (IoT) devices, it has become easier to collect various data. For example, to collect data in an area where human access is not possible, sensing devices could be scattered by Unmanned Aerial Vehicles (UAV) such as drones (see **Fig. 1**). However, it is not easy to evenly deploy small IoT devices through scattering in the drones. Accordingly, it is difficult to collect accurate data, and a small device's energy may be exhausted due to the continuous collection of inaccurate data, and unexpected device defects may occur. In the worst case, the whole network communication could be disconnected, and data collection may no longer be possible [3]. An area where it is difficult to collect data anymore is called a sensing hole.

The ideal way to recover the sensing hole is to move the mobile IoT sensing device directly to the sensing hole to enable data collection. In general, early research on the movement of mobile sensors was a method using wheels. However, the wheel has a limitation, challenging to move in a rough area with many obstacles. To overcome the limitations of wheel-based movement, an IoT hopping sensor device in which the sensor jumps and moves in the desired direction has been proposed [4, 5]. Since the hopping sensor node moves in a jump, it is straightforward to migrate in areas such as rocks or sand. In addition, the hopping sensor node is able to adjust the data transmission radius because data can be transmitted while jumping. For example, the authors of the paper [6] studied that the data transmission radius can increase about six times compared to the ground communication radius when a hopping node jumps $1m$ from the ground. The author of the paper [7] implemented a projectile to implement a hopping sensor.

In recent decades, various hopping sensor relocation algorithms have been researched. In the representative paper [8] based on Dijkstra's algorithm, recovering the sensing hole by relocating hopping sensor nodes in the cluster zone on the shortest path to the sensing hole was first proposed. Also, in the study of [9], relocating the hopping sensor according to the level of rugged terrains was first studied. However, various studies [10-12] set up paths using all the current network information, including all hopping sensor nodes' statuses. For instance, an unrealistic assumption is that the location and the level of rugged terrains in each area are figured out. Even if the network area is minimal, it is practically impossible for all nodes to exchange information and establish routes.

Recently, our research group solved this problem in the paper [13]. Every sensor node does not need to know the surrounding sensor nodes' information and the entire network environment. It is a relocation protocol based on the distributed networking that recovers the sensing hole by requesting sensor nodes for relocation from nearby areas. Using this relocation protocol [13], the paper [14] proposed a protocol to recover the sensing hole by predicting rugged terrains' level based on the relocation's success rate. However, the shortest path-based relocation protocols repeatedly use specific neighbor areas while requesting nodes needed. In particular, if the distribution of rugged terrains in the relocation paths is not uniform, a request in a direction having a high movement success rate will be appropriate. It may be necessary to avoid the method based on the shortest path and, it could be useful to manage the distribution of rugged terrains in each direction.

Reinforcement learning is a very well-known machine learning method [15]. For the best policy decision, the policy is decided by applying the maximum reward. The reward is repeatedly calculated by considering samples in a given environment, and the samples are also continuously updated. In this paper, in order to request hopping member nodes to move, the selection of relay nodes that will transmit the request message of the sensing hole cluster header is solved with reinforcement learning. A relay node selection random variable is considered to select the best relay node among the relay node candidates. A simple Monte Carlo approach is applied to choose the most accomplished relay node based on the selected random variable, and the random variable is reinforced continuously. This method could reduce a large number of network traffic generated by existing relocation methods. Also, by overcoming the limitation of repeatedly selecting specific relay nodes based on the shortest path, it makes sure to increase the relocation success rate of the hopping sensors remarkably. In addition, (in the world, no research team has been able to proceed with) we first simulate the proposed hopping sensor relocation protocol based on a distributed networking using OMNeT++ similar to the real environment [16].

This paper is organized as follows. Section 2 explains previously proposed relocation protocols based on the distributed environment, and Section 3 describes the proposed hopping sensor relocation protocol. Section 4 describes the simulation and performance evaluation and finally concludes in Section 5.

2. Previous Work

In this Section, a survey of the relocation protocols that have been studied so far is provided, and primary considerations are described to explain the new relocation protocol proposed in the next Section. In addition, relocation protocols to be compared together are also briefly described to help to understand this study.

2.1 Relocations for Hopping Sensors

Various hopping sensor relocation algorithms have been proposed in recent decades. A hopping sensor's characteristic is that it is possible to perform a movement in rough areas such as disaster areas by jumping rather than by general means of movement, such as wheels. The authors of [9, 14] studied that the relocation performance could be improved using a hopping sensor in an obstacle-distributed environment. Furthermore, it is also possible to extend the data transfer radius of the sensor nodes through jumps. The authors of [6] confirmed in an experiment that a sensor node could adjust its communication radius while jumping to an appropriate height. For example, they showed that if a sensor node jumps 1m from the ground, it can be increased by about six times than a typical radius of communication on the ground. The authors of [7] directly measured the transmission radius according to the height change from the ground using a jumping launcher. Extensions of the communication radius over jumps can improve the connectivity of sensor nodes across the network, and in this paper, we also leverage these mechanisms in setting the management area of the cluster header.

The study of hopping sensors-based relocation algorithms began with the authors of [8] proposing a scheme to recover a sensing hole by relocating some hopping nodes on the shortest paths between the sensing hole and the cluster zones. However, another sensing hole occurs quickly as hopping sensors move through specific clusters on the shortest path and repeatedly send request messages for relocation to headers in neighboring cluster regions. Furthermore, some hopping sensors in these cluster regions had limitations that repeated movements could lead to losing their movement capabilities. To address this problem, the authors of [10] set up the most disjoint path to avoid duplicating the relocation path as much as possible when multiple sensing holes occur. Furthermore, the authors of [11] utilized the relocation policy of multi-path sensor nodes instead of the shortest paths. These studies have improved the migration success rate of sensors and hopping sensors' capability across the network over shortest path-based relocation schemes.

However, the studies mentioned above establish paths for real-time relocation using all the information in the current entire network by all the cluster headers and nodes. Even though the network area is minimal, it is practically impossible for all cluster headers to exchange information and set paths, and so many control messages are sent and received. To overcome these limitations, the authors of [13] proposed a distributed manner-based relocation protocol. Every sensor node needs not know any information about the surrounding sensor nodes and the entire network. It is only that the header of the sensing hole requests the neighboring cluster header the required number of sensor nodes for sensing hole recovery. Using this mechanism, the authors of [14] predicted obstacle levels based on relocation success rates. Besides, the authors of [17] further improved the distributed-based relocation protocol by addressing the limitations (such as the well-known ping-pong problem) arising from the paper [13].

2.2 Basic Terms and Relocation Strategy

As shown in **Fig. 1**, after scattering IoT hopping sensors using UAV in disasters or military areas that are not accessible to humans, data of interest can be gathered for big data analysis. All the sensors initially deployed can be divided into several appropriate cluster zones with various clustering algorithms [18]. Sensors in each cluster zone center are elected as cluster headers (H_A, H_B, ...), and sensors in the same zone as cluster headers are called member nodes (M1, M2, ...). The cluster header communicates to figure out information of its member nodes periodically. This paper supposes that network clustering and header selection are possible using various well-known algorithms, and further discussion is omitted.

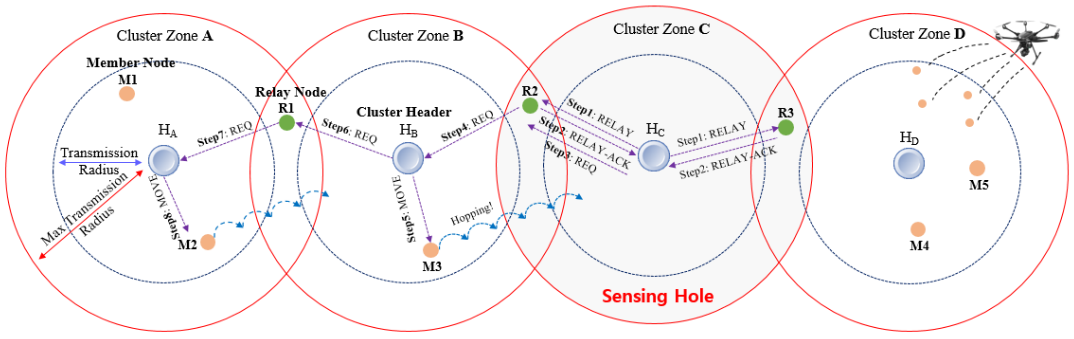


Fig. 1. Relocation algorithm in [13]

The hopping sensor is able to jump to have communication with adjacent sensors, and each sensor can know its location using the GPS unit. As shown in **Fig. 1**, a blue line indicates the hopping sensor's transmission radius on the ground. A red line indicates the maximum transmission radius that can communicate by jumping as high as possible. The cluster header's maximum transmission radius area is defined as a cluster zone; thus, direct communication between cluster headers is likely impossible. There is a possibility that some member nodes in the area intersect the cluster zones, and they can communicate with more than one cluster header. These member nodes are called relay nodes, and the role of relay nodes serves to transmit data in the middle for communication between cluster headers [19].

When a sensing hole occurs because the number of sensor nodes required for data collection in a cluster zone is insufficient, the sensing hole's cluster header requests relocation of member nodes necessary for sensing hole recovery in the adjacent cluster zone. **Fig. 1** describes a simple example of the most representative relocation protocol [13], and the message types are shown in **Table 1**. In the next section, the proposed scheme also uses some of the message types described below. (In advance, RELAY and RELAY-ACK messages are not used in the proposed protocol.)

Table 1. Message type and description for the relocation protocol in **Fig. 1**

Type	Description/Format
HELLO	· The header checks the state of its zone with periodic broadcasting. · { type, source address, destination address (broadcasting) }
HELLO-ACK	· Member node sends relay node field (T/F) in response to receiving HELLO message. · { type, src address, dst address (header), relay node field (T/F) }
RELAY	· The header performs multicasting on the relay nodes to select the appropriate relay nodes. · { type, src address, dst addresses (relay nodes) }
RELAY-ACK	· Relay nodes receive RELAY and then answer. · { type, src address, dst address (header) }
REQ	· Header requests a relay node to move hopping sensors for sensing hole recovery · { type, src address, dst address (relay), # of req. members, sensing hole header address, header GPS information }
MOVE	· Header sends moving commands to the selected members by multicasting. · { type, src address, dst address (hopping members), sensing hole header address, sensing hole header GPS information }

Each cluster header (H_A, H_B) periodically broadcasts a HELLO message in each cluster zone, and it can continuously check the states of the hopping sensor member nodes in its area. Member nodes should respond to their states to their headers and inform that they are relay nodes through HELLO-ACK messages if they have received HELLO messages more than one. If the initial network policy identifies lower than a certain number of member nodes, the cluster header can determine that its cluster zone has become a sensing hole. In **Fig. 1**, the cluster header H_C determines that a sensing hole occurs, and a brief look at the relocation strategy performed to recover it is as follows:

Step 1. The cluster header H_C multicasts RELAY messages to all its relay nodes (R2, R3) to request one member node required from any neighboring cluster zones (Clusters B and D).

Step 2. Each relay node immediately sends a RELAY-ACK message in response to the RELAY message received, and here, the reply from R2 arrives at the H_C the fastest. R3 responses to be received later are ignored. (In other words, we can know that it is the shortest path-based relocation method by selecting the relay node that responded fastest.)

Step 3. Cluster header H_C transmits a REQ message to the chosen relay node R2 to request one sensor required.

Step 4. The relay node R2 immediately delivers the REQ message received from the cluster header H_C to another its cluster header H_B.

Step 5. The cluster Header H_B chooses M3 as a movable hopping sensor member node and sends a MOVE message to move to cluster zone C. Upon receiving the message, and the member node M3 moves to the neighboring sensing hole cluster zone.

Step 6. Simultaneously, the cluster header H_B predicts that its zone will also be a sensing hole and sends a REQ message to the relay node R1 to request one sensor needed. (Of course, there is a relay node selection process for sending REQ messages here, but it has been omitted.)

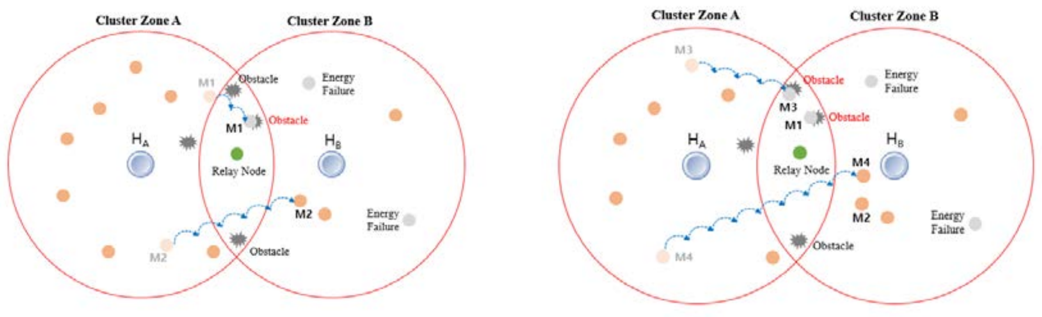
Step 7. The relay node R1 also delivers its REQ message to cluster header H_A.

Step 8. Cluster Header H_A selects M2 from sensor node members inside its zone and orders the move to cluster zone B.

As a result, one sensor is appropriately relocated inside each cluster zone, so that all sensing holes can be recovered.

2.3 Improved Relocation Mechanism to Overcome Obstacle Environment

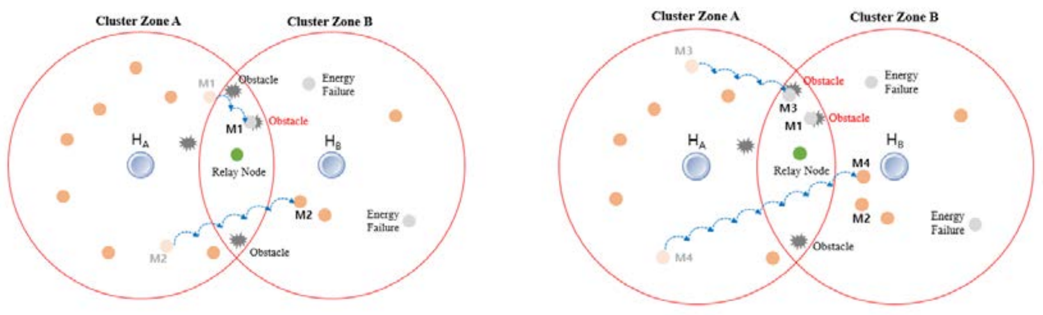
In general, the environment in which the hopping sensor is considered is not a flat terrain. When considering the topographic information about obstacles around the cluster zone, it is necessary to consider defects that may occur during movement. In the paper [14], it was possible to increase the sensing hole recovery rate by indirectly predicting the distribution state of obstacles (stone, mud, sinkhole, *etc*.) around the sensing hole.



(a) (b)

Fig. 2. Scenarios to describe existing relocation methods in [13, 14]

First, let us look at a typical member node relocation scenario that does not consider obstacles. In cluster zone B of **Fig. 2(a)**, there are node failures because two members ran out of energy. After a while, the header H_B determines that its zone is a sensing hole and requests two member nodes necessary for cluster zone A. Two member nodes (M1, M2) ordered move from zone A move to zone B. One member (M1) is caught in an obstacle and failed to move to zone B, but the other (M2) is succeeded in moving to zone B. Zone B's header determines a sensing hole again and requests additional member movement to zone A. In **Fig. 2(b)**, member M3 of zone A fails to move to zone B because of an obstacle again during movement. In order to overcome the sensing hole, the header of zone B requests the movement of one member from zone A again so that the member M4 succeeds in the movement and recovers the sensing hole. We have looked at the relocation through three REQ messages to recover the first sensing hole.



(a) (b)

Fig. 2. Scenarios to describe existing relocation methods in [13, 14]

Second, let us look at the relocation of node members when considering the existence of obstacles. In cluster zone B of **Fig. 2(a)**, two members' energies are exhausted, resulting in node failure. After a while, the header H_B determines a sensing hole and tries to request two member nodes from zone A. In the relocation protocol of [14], the number of members required is calculated by considering the movement success rate. The movement success rate p and the number of requested members cnt are calculated as follows:

$$p = (\text{\# of members successfully moved}) \Big/ (\text{\# of members requested}) \tag{1}$$

$$cnt = \text{Ceiling}\ [\ (\text{\# of members requested}) \cdot \left(1 + \left(1 - p\right)\right)\ ] \tag{2}$$

After setting the initial value of p to 1, the number of member requests is $\left\lceil 2 \cdot \left(1 + \left(1 - 1\right)\right) \right\rceil$, that is, two members are requested from zone A. In **Fig. 2(a)**, two members are ordered to move from zone A, and as in the first scenario, one member (M1) fails to move due to an obstacle, but the other (M2) successfully moves to zone B. In **Fig. 2(b)**, zone B's header calculates the value of p as 1/2 according to Equation (1), and it determines that the sensing hole could still not be recovered. The number of members requested from zone A is $\left\lceil 1 \cdot \left(1 + \left(1 - 1/2\right)\right) \right\rceil$, and two members are requested to move again. For the two members of zone A, one member (M3) fails to move to zone B as in the previous scenario. The other member (M4) can successfully move to zone B, so the sensing hole is recovered. We have looked that two REQ messages used are lower than those of the first scenario's three REQ messages to recover the sensing hole.

The first scenario is the relocation method first proposed in the paper [13], and it aims to recover the sensing hole quickly based on the shortest path. The second scenario is a relocation method that indirectly estimates the state of an obstacle considering the success rate of movement in the paper [14]. There is a profit in that the number of requesting member nodes could be increased according to the movement success rate so that the sensing hole can be recovered quickly.

However, many things have been realistically overlooked in these relocation schemes. First, the assumption that obstacles are evenly distributed is somewhat unreasonable. For example, the distribution of obstacles in a disaster area is likely not uniform depending on the direction. Second, both relocation algorithms [13, 14] are the shortest path based schemes. A method of multicasting a RELAY message to relay nodes and selecting the relay node that sent the RELAY-ACK message first is used. However, considering the distribution of obstacles, there is no reason to stick to only the shortest path. In other words, it could be most advantageous for sensing hole recovery to request necessary member nodes to the cluster zone with the lowest distribution of obstacles. Therefore, in the next section, we propose how the cluster header of the sensing hole reflects uneven obstacles to select the appropriate relay node to transmit the REQ message and how to determine the best policy through reinforcement learning of the cluster header.

3. Proposed Relocation Protocol

In this section, we explore the problems of how to consider obstacle conditions in existing studies and propose real-world surrounding obstacle environment predictions through reinforcement learning.

3.1 Reviewing Problems in Previous Studies

The background picture of **Fig. 3** depicted a natural disaster in Kenya in 2018 and is intended to be used as an example of a non-uniform distribution of obstacles. Researchers initially deploy sensors evenly as much as possible to analyze the characteristics of the terrain of interest. However, suppose that the continuing operation of the sensors in cluster zone B caused an energy defect, making cluster zone B a sensing hole.

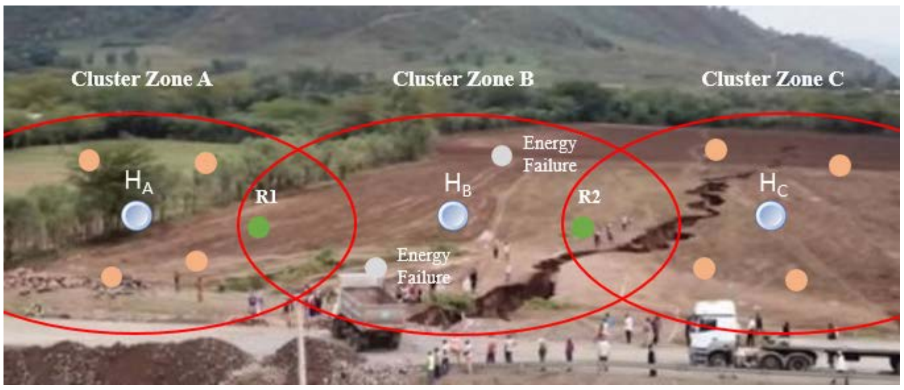


Fig. 3. An example of non-uniformly distributed obstacles: Kenya's Big Crack [20]

Since the two previously introduced relocation algorithms [13, 14] are shortest-path based relocation algorithms, **Fig. 3**'s sensing hole cluster header H_B will probably choose relay node

R2 to request member nodes needed for sensing hole recovery. However, no matter how much the relocation algorithm in [14] predicts the level of obstacles and requests additional hopping sensor member nodes beyond the required number, an obstacle status of **Fig. 3** will most likely result in a continuous movement failure. After all, it is clear that continuous selection of relay node R2 on the shortest path basis will be fundamentally wrong, with additional costs such as multiple member-movement defects and message transmission.

To overcome the selection of only specific relay nodes, which is the shortest path-based limit, a recent paper [17] proposes a method of selecting relay nodes to some extent equally using queues. The cluster headers queue the information of relay nodes in the order in which they received the RELAY-ACK messages and then sequentially pull them out of the queue to perform the selection of relay nodes so that they are not skewed. However, in obstacle environments such as **Fig. 3**, there is still a possibility that about half of the requests for movement (*i.e.*, zone C of clusters zone A and C) will fail.

3.2 The Proposed Relocation Algorithm

Fig. 4 shows the mechanism of the cluster header unit that executes the proposed relocation algorithm. In **Fig. 4(a)**, the header broadcasts the HELLO message from **Table 1** to his zone. In **Fig. 4(b)**, when the header receives HELLO-ACK messages, it identifies the relay nodes among each member node. In **Fig. 4(c)**, the header identifies and counts the member nodes except the relay nodes. In **Fig. 4(d)**, the header determines if his zone has become a sensing hole. If identified as a sensing hole, the header requests the neighboring zone for the member nodes he needs through the algorithm of **Fig. 5**.

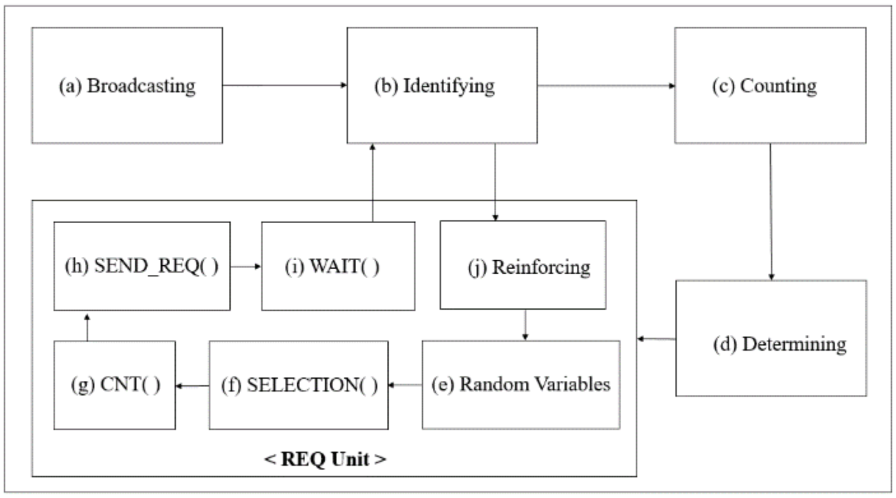


Fig. 4. Header unit of hopping sensor member node

The algorithm for the REQ Unit of **Fig 4** is presented in **Fig 5**. In **Fig. 5**, lines 01-05 describe the input and output values. Line 02 is a set ($Relay$) of the relay nodes, and Line 03 is a set ($\mathbf{P}_t$) of the time instance t in which the sensing hole occurred and the success rates for each relay node that reflects the success rate of the requested moving nodes through the selected relay node. Line 04 is a set ($\mathbf{R}_t$) of relay node selection random variables used at the $t+1$ point in time. Line 05 defines REQ messages as output values sent during the time (T) the cluster header is operational.

Lines 06-07 are initializations for success rate and random variables. The initial relay node selection probability variables ($\mathbf{R}_0$) were all given equal probability distributions (**Fig. 4(e)**). Lines 08-18 describe a mechanism for transmitting REQ messages at the time of sensing hole occurrence t. Line 09 selects the relay node at t-time using $\mathbf{R}_{t-1}$. The **SELECTION**() function can select the best relay node using random variables, such as the Monte Carlo method (**Fig. 4(f)**). The number of member nodes to be requested to the selected relay node r is calculated using the **CNT**() function (**Fig. 4(g)**). The **CNT**() function uses Equation (2) and success rate variables $\mathbf{P}_{t-1}$ to calculate the cnt of Equation (2). In line 11, the cluster header sends REQ messages (**Fig. 4(h)**), waits the next HELLO broadcasting at line 12 (**Fig. 4(i)**), and calculates the success rate (p of Equation (1)) at lines 13-14 (**Fig. 4(j)**).

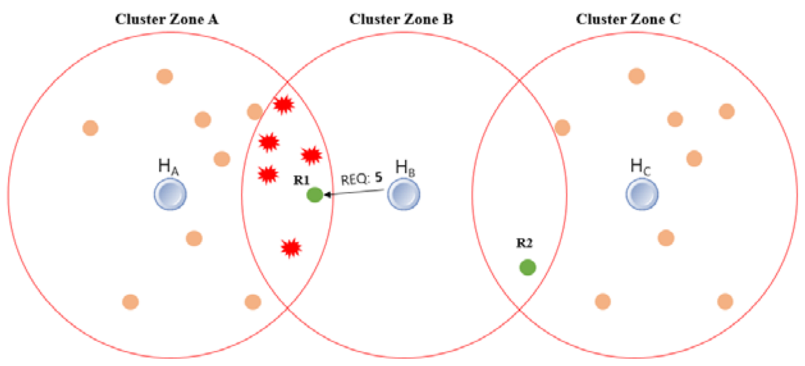
```
01.  **Input**:
02.          $Relay \leftarrow \{ R1, R2, ..., Rn \}$
03.          $\mathbf{P}_t \leftarrow \{ p_t^{R1}, p_t^{R2}, ..., p_t^{Rn} \}$
04.          $\mathbf{R}_t \leftarrow \{ \Re_t^{R1}, \Re_t^{R2}, ..., \Re_t^{Rn} \}$
05.  **Output**: REQ message including # of members requested and
             the selected relay node
06.  $\mathbf{P}_0 \leftarrow \{ 1, 1, ..., 1 \}$
07.  $\mathbf{R}_0 \leftarrow \{ \frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n} \}$
08.  **FOR** $( t \in T )$ **DO**
09.          $r \leftarrow \mathbf{SELECTION}(\mathbf{R}_{t-1})$
10.          $cnt \leftarrow \mathbf{CNT}(p_{t-1}^r)$
11.          $\mathbf{SEND\_REQ}(r, cnt)$
12.          $\mathbf{WAIT}(\text{Hello Message Interval Time})$
13.          $s \leftarrow$ # of members successfully moved
14.          $p \leftarrow \frac{s}{cnt}$
15.          $\mathbf{P}_t \leftarrow \mathbf{P}_{t-1}$
16.          $p_t^r \leftarrow \frac{\left( \sum_{k=t-m+1}^{t-1} p_k^r + p \right)}{m}$
17.          $\mathbf{R}_t \leftarrow \frac{\mathbf{P}_t}{\sum_{i=1}^{n} p_t^{Ri}}$
18.  **END FOR**
```

Fig. 5. The proposed algorithm for REQ unit of **Fig. 4**.

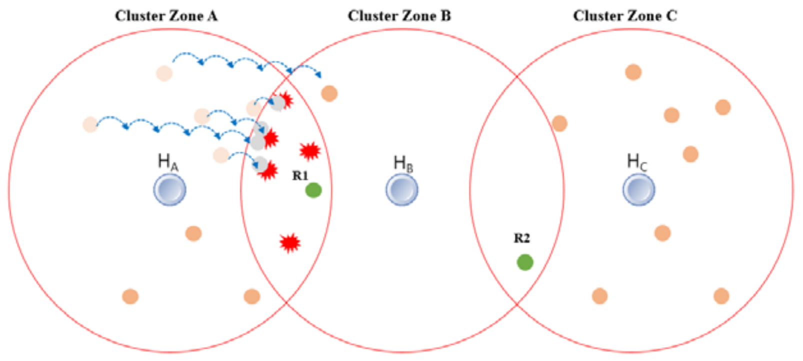
Lines 15-17 reinforce the set of success rates $\mathbf{P}_{t-1}$ and the set of selection random variables $\mathbf{R}_{t-1}$ to $\mathbf{P}_t$ and $\mathbf{R}_t$. We calculate $\mathbf{P}_t$ using the most recent $m-1$ set from a set of success rates $\{\mathbf{P}_0, \mathbf{P}_1, \cdots \mathbf{P}_{t-2}, \mathbf{P}_{t-1}\}$ and the success rate p just calculated. The selection random variable $\mathbf{R}_t$ is then calculated by normalizing $\mathbf{P}_t$ on line 17 (**Fig. 4(j)**).

3.3 A Case Study for Relocation Algorithm

In the example of **Fig. 6**, assume that at least five sensor node members are required to recover the sensing hole. In the paper [14], considering the obstacle environment, the header H_B of cluster zone B detects the sensing hole and delivers RELAY messages to the relay nodes, as shown in **Fig. 6(a)**. Select relay node R1 closest to it and respond first, and request five members from neighboring cluster zone A. In **Fig. 6(b)**, five members selected by the header H_A are moving to the sensing hole. Of these, four members were faulted by obstacles, and only one member was able to move to restore the sensing hole. The header H_B, which has yet to recover the sensing hole, needs four additional members to recover the sensing hole, but using Equation (2), $\lceil 4\times(1+(1-1/5)) \rceil$, 8 members are requested. It is likely that R1 will still be selected as the shortest path base for the selection of relay nodes to be requested, and cluster zone A will request eight members. However, in the direction of zone A, there are many obstacles as previously experienced, and there are currently only three member-nodes, so even if they all succeed in moving, they cannot recover the sensing hole.



(a)



(b)

Fig. 6. Example of relocation algorithms

However, in the proposed method, the header H_B of the cluster zone B detects a sensing hole and manages a random variable for selecting relay nodes. In all the relocation algorithms presented so far, the relay node was selected as a method of transmitting RELAY and RELAY-ACK messages, but this process was excluded. Moreover, the relay node selection random variable is managed by the REQ unit of the cluster header. Initially, for each of the relay nodes R1 and R2, the success rate variable and the selection random variable are initialized to $\mathbf{P}_0$ = {1, 1} and $\mathbf{R}_0$ = {1/2, 1/2}, respectively. As **Fig. 6(a)**, assume that the header H_B detects the sensing hole and selects R1 according to the random variable $\mathbf{R}_0$. In the same way as the previous example, as **Fig. 6(b)**, suppose that only one member moved to the sensing hole while five members selected by the header H_A were moving to the sensing hole.

The header H_B checks that the success rate of relay node R1 changes from 1 to 1/5, calculates as (1+1/5)/2 = 3/5, and updates as $\mathbf{P}_1$ ={3/5, 1} (For simplicity of the example, the value m in **Fig. 5** is set to 2). Therefore, in order to normalize $\mathbf{P}_1$, since 3/5 + 1 = 8/5, the selection random variable can be reinforced to $\mathbf{R}_1$ = {3/8, 5/8}. In **Fig. 6(b)**, according to the selection random variable $\mathbf{R}_1$, the proposed scheme selects the relay node R2 and requests four members using the current success rate p_1^{R2} =1. The probability of recovering the sensing hole could increase. The proposed algorithm does not insist on relay node R1 by considering the shortest path as a method of transmitting RELAY and RELAY-ACK messages, and reflects the failure of R1 using the Monte Carlo method. Thus, relay node R2 can be selected by the result of the best decision making.

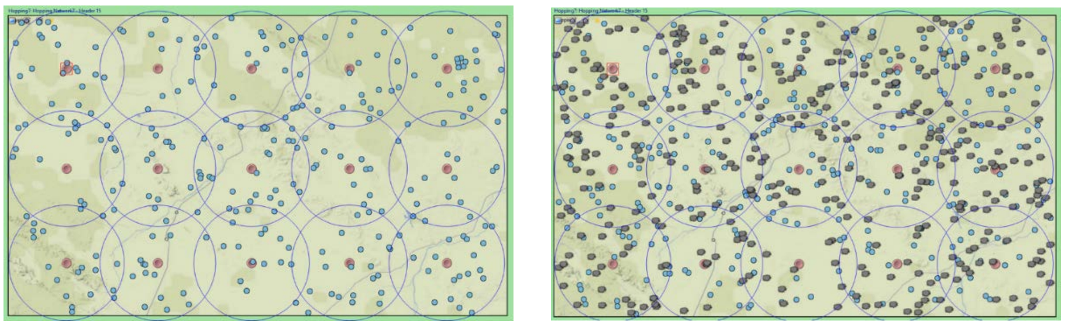
4. Performance Evaluation

A performance evaluation of the proposed relocation protocol is performed in this section. For realistic performance evaluation, the authors made the world's first attempt to completely implement the simulation using OMNeT++ [16, 21]. As it is well known, implementing the simulation by OMNeT++ requires considering all layers of the network, and it must have been a very arduous task as it took years to perfectly implement the movement and communication of hundreds of hop sensor member nodes. Table 2 provides an overview of the OMNeT++ implementation environment.

Table 2. Simulation environments

Parameters	**Values**
Network area	250m×150m
Number of cluster headers	15
Minimum number of members for each cluster to properly gather data	10
Maximum communication radius for each sensor node	20m
Maximum communication radius when highly jumping	29m
Maximum distance that a sensor node moves forward with one jump	2m
HELLO message interval	15, 30min

Three hundred hopping sensors are randomly distributed in an area of $250m \times 150m$ to collect data, as shown in **Fig. 7**. Among them, 15 cluster headers are marked in red, and the remaining is 285 sensor member nodes. When there are less than ten sensor nodes in a cluster zone, a sensing hole could occur. The transmission radius of each hopping sensor on the ground is supposed to be 20m. The maximum transmission radius is 29m when the sensor jumps to the full height. The hopping sensor could move about 2m forward with one jump and hop up to 130 times. As shown in **Fig. 7(b)**, obstacles are uniform-based randomly generated with a value of 1% of the total area. The obstacle's size appears more extensive than the actual size; however, the obstacle is assumed to be a square with a width and length of 1m. Just for the visual effect, the obstacles are made to appear larger. A cluster header broadcasts a HELLO message every 15 minutes to detect that its zone is a sensing hole. For the relocation protocol's performance analysis, a sensing hole is created in the central cluster zone, as shown in **Fig. 7** and **Fig. 10**. To generate the sensing hole, we assume a scenario that the member nodes continuously collect data, so the nodes' energy is rapidly consumed. Every sensor member node in the central cluster generates an event of data collection under an exponential distribution with an average of 5 minutes. For convenience, the initial energy value for sensing is set to 100, and 1 (energy consumption) is reduced for each event. The pseudocode's value m in Fig. 5 for the relay node selection random variable of the proposed scheme is set to 2 for simplicity. The simulation time is set to three days.



(a) (b)

Fig. 7. Snapshot of the simulation topologies

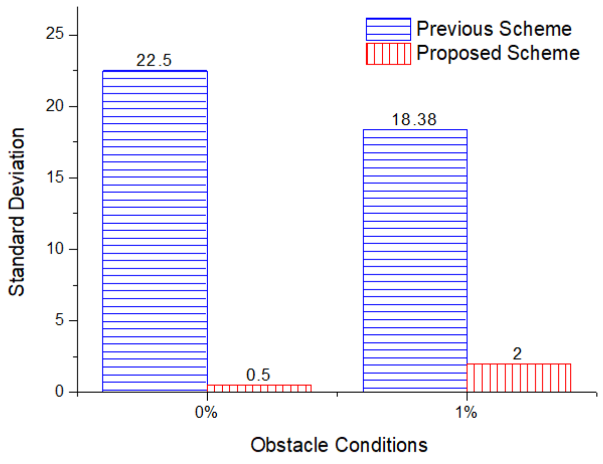


Fig. 8. Standard deviation for relay node selection

Fig. 8 shows the standard deviation values of the frequency of relay nodes selected to request necessary sensors by the central cluster header when the zone becomes a sensing hole. In the existing method [14], since the relay node that responds fastest is intensively selected (i.e., the shortest path-based mechanism), we can check that the standard deviation value is very high. In other words, there is a wide variation for relay node selection. However, it can be seen that the relay node selection of the proposed method is uniformly selected overall with a low standard deviation value.

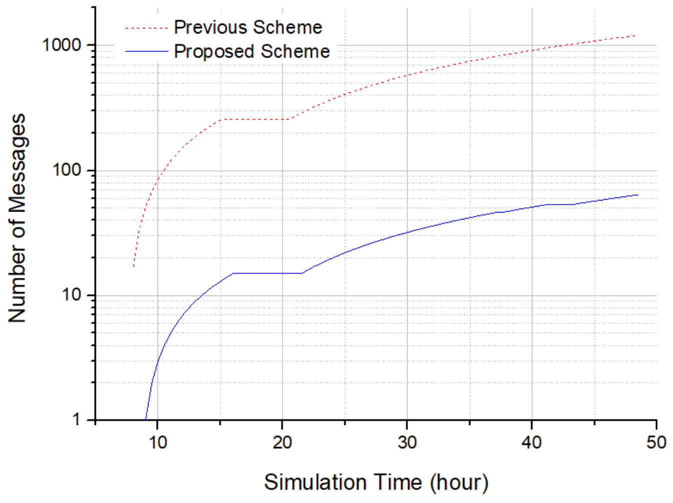


Fig. 9. Number of primary messages generated (with obstacle 1%)

Fig. 9 shows the number of primary (RELAY, RELAY-ACK, and REQ) messages occurring in the central cluster. The previous scheme establishes the shortest path using these three types of messages. However, since only the REQ message transmission is generated using the REQ unit of **Fig. 4** in the proposed scheme, the number of messages than the previous one could be reduced in an enormous amount. Therefore, we can predict that the amount of primary message generation across the whole network has to be significantly reduced.

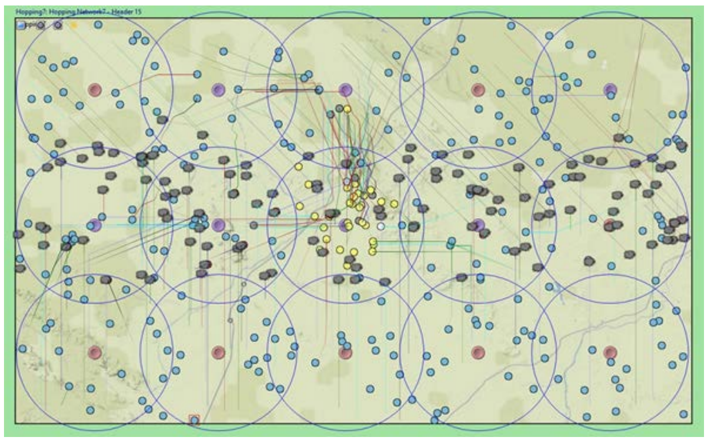


Fig. 10. Simulation snapshot for non-uniform obstacle distribution (0.33%) topology

To consider the non-uniform distribution of obstacles, we set to establish a topology environment as **Fig. 10**. Here, obstacles are non-uniformly generated with a value of 0.33% of the total area. The HELLO message interval time is 30 minutes. In particular, only the middle area is considered an environment with many obstacles. Yellow nodes are defective nodes, and hopping sensor nodes can be seen from solid lines that many movements have occurred in the north direction to recover the sensing hole.

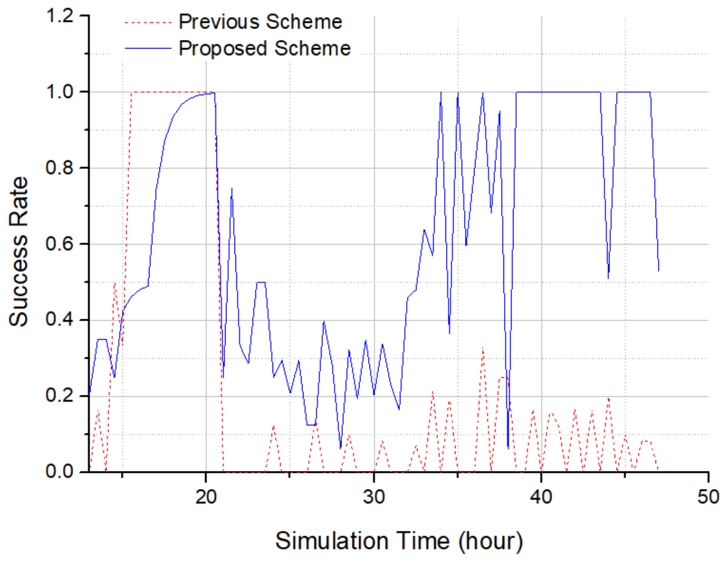


Fig. 11. Relocation success rate

Fig. 11 explains the success rate for nodes relocation according to the REQ message transmitted whenever a sensing hole occurs. In the environment shown in **Fig. 10**, it could be that the success rate is meager since the previous scheme only requests relocation in a specific direction. However, we are sure that the proposed method through reinforcement learning rapidly increases the relocation success rate after 30 hours.

5. Conclusion

It is most desirable to relocate the mobile sensor when a sensing hole condition in which data collection is impossible due to improper placement of IoT sensor devices or energy depletion failure. In the most realistic distributed networking-based relocation protocol so far, the sensing hole's cluster header transmits a request message for sensor relocation to the adjacent cluster header via a specific relay node at the nearest distance for rapid recovery of physical faults. However, sticking to specific nodes at the closest distance from such a rough terrain where the disaster occurred is not an appropriate way to successfully recover sensing holes. If a disaster occurred in an area where the periphery of that nearby node is challenging to move around, it would harm the hopping sensor's relocation. Furthermore, frequent use of specific relay nodes can increase the likelihood of unequal energy use from distributed sensor nodes, resulting in another node fault.

In this paper, we select the best relay node as reinforcement learning by applying a simple Monte Carlo method based on the relay nodes selection random variables that reflect the characteristics of the obstacle distribution, rather than the shortest path-based selection policy when the cluster header of the sensing hole selects relay nodes. The proposed scheme selects relay nodes with a high success rate and relocates the hopping sensors to recover a sensing hole. In an environment with no obstacles or uniformly distributed, relay nodes could be selected evenly. In order to evaluate the proposed scheme's performance and reflect the actual physical communication equipment, the well-known OMNeT++ simulator is used. The proposed method can also reduce the vast number of communication messages compared to the previous one. In this work, however, there is a limitation that relay nodes managed by the cluster header are fixed. In real-world disaster areas, a sensor node failure could occur frequently, so we would need to study how to exclude faulty relay nodes and add new nominated relay nodes. Also, we plan to research relocation protocols that can appropriately respond to frequently changing surrounding obstacle environments in the future.

Acknowledgement

This work was supported by Incheon National University Research Grant in 2020(2020-0136).

References

[1] D. Sun, H. Yan, S. Gao and Z. Zhou, "Performance Evaluation and Analysis of Multiple Scenarios of Big Data Stream Computing on Storm Platform," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 7, pp. 2977-2997, July 2018. Article (CrossRef Link)

[2] J. Camacho, G. Maciá-Fernández, N. M. Fuentes-García, and E. Saccenti, "Semi-Supervised Multivariate Statistical Network Monitoring for Learning Security Threats," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2179-2189, August 2019. Article (CrossRef Link)

[3] M. Kim, S. Park, and W. Lee, "A Robust Energy Saving Data Dissemination Protocol for IoT-WSNs," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 12, pp. 5744-5764, December 2018. Article (CrossRef Link)

[4] M. E. Snyder, "Foundations of Coverage Algorithms in Autonomic Mobile Sensor Networks," Ph.D. dissertation, Dept. Computer Science, Missouri Univ. of Science and Technology, MO, USA, 2014.

[5] J. Zhao, J. Xu, B. Gao, N. Xi, F. J. Cintrón, M. W. Mutka, and L. Xiao, "MSU Jumper: A Single-Motor-Actuated Miniature Steerable Jumping Robot," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 602-614, June 2013. Article (CrossRef Link)

[6] F. Cintr'on, "Network Issues for 3D Wireless Sensors Networks," Ph.D. dissertation, Dept. Computer Science, Michigan State Univ., MI, USA, 2013.

[7] M. S. Kim, "Design of a Transmission Process for Hopping Sensors to Enhance Coverage," M.S. thesis, Dept. Mobile Communication Engineering, Sungkyunkwan Univ., Suwon, Republic of Korea, 2012.

[8] Z. Cen and M. W. Mutka, "Relocation of Hopping Sensors," in *Proc. of 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, pp. 569-574, May 19-23, 2008. Article (CrossRef Link)

[9] M. Kim, M. W. Mutka, and H. Choo, "On Relocation of Hopping Sensors for Rugged Terrains," in *Proc. of 2010 International Conference on Computational Science and Its Applications*, Fukuoka, Japan, pp. 203-210, March 23-26, 2010. Article (CrossRef Link)

[10] M. Kim and M. W. Mutka, "On Relocation of Hopping Sensors for Balanced Migration Distribution of Sensors," in *Proc. of Computational Science and Its Applications – ICCSA 2009*, pp. 361-371, June 2009. Article (CrossRef Link)

[11] M. Kim and M. W. Mutka, "Multipath-based Relocation Schemes Considering Balanced Assignment for Hopping Sensors," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, pp. 5095-5100, October 10-15, 2009. Article (CrossRef Link)

[12] M. Kim and M. W. Mutka, "Recycled ID assignment for relocation of hopping sensors," in *Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Lucca, Italy, pp. 1-3, June 20-24, 2011. Article (CrossRef Link)

[13] M. Kim, S. Park, and W. Lee, "Energy and Distance-Aware Hopping Sensor Relocation for Wireless Sensor Networks," *Sensors*, vol. 19, no. 7, p. 1567, 2019. Article (CrossRef Link)

[14] S. Park, M. Kim, and W. Lee, "Energy-Efficient Wireless Hopping Sensor Relocation Based on Prediction of Terrain Conditions," *Electronics*, vol. 9, no. 1, p. 49, 2020. Article (CrossRef Link)

[15] L. Nie, Z. Ning, M. S. Obaidat, B. Sadoun, H. Wang, S. Li, L. Guo, and G. Wang, "A Reinforcement Learning-Based Network Traffic Prediction Mechanism in Intelligent Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2169-2180, March 2021. Article (CrossRef Link)

[16] OMNeT [Online] https://www.omnetpp.org, Accessed on: March 30, 2021.

[17] M. Kim, S. Park, and W. Lee, "Ping-Pong Free Advanced and Energy Efficient Sensor Relocation for IoT-Sensory Network," *Sensors*, vol. 20, no. 19, p. 5654, 2020. Article (CrossRef Link)

[18] A. S. Rostami, M. Badkoobe, F. Mohanna, H. Keshavarz, A. A. R. Hosseinabadi, and A. K. Sangaiah, "Survey on clustering in heterogeneous and homogeneous wireless sensor networks," *Springer, The Journal of Supercomputing*, vol. 74, pp. 277-323, 2018. Article (CrossRef Link)

[19] J. Seo, M. Kim, I. Hur, W. Choi, and H. Choo, "DRDT: Distributed and Reliable Data Transmission with Cooperative Nodes for Lossy Wireless Sensor Networks," *Sensors*, vol. 10, no. 4, pp. 2793-2811, 2010. Article (CrossRef Link)

[20] CBSN, "Massive Crack in Earth Opens up in Kenya," April 2018. [Online] https://youtu.be/RG-wx-KYnTk, Accessed on: March 30, 2021.

[21] Virdis and M. Kirsche, *Recent Advances in Network Simulation: The OMNeT++ Environment and Its Ecosystem*, Switzerland: Springer, 2019.

Moonseong Kim received the M.S. degree in Mathematics, August 2002 and the Ph.D. degree in Electrical and Computer Engineering, February 2007 both from Sungkyunkwan University, Korea. He was a Research Professor at Sungkyunkwan University in 2007. From December 2007 to October 2009, he was a Research Associate in ECE and CSE, Michigan State University, USA. He was a Deputy Director and a Patent Examiner with Korean Intellectual Property Office, Daejeon, Korea, from October 2009 to August 2018. In September 2018, he joined the Seoul Theological University, Bucheon, Korea, where he is currently working as an Assistant Professor and the Head of the Department of IT Convergence Software. His research interests include wired/wireless networking, sensor networking, mobile computing, network security protocols, and simulations/numerical analysis. Since March 2009, he has been an editor of KSII Transactions on Internet and Information Systems (TIIS).

Woochan Lee received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, Korea, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2016. He was commissioned as a Full-time Lecturer and a First Lieutenant with the Korea Military Academy, Seoul, Korea, from 2005 to 2008. He was a Deputy Director and a Patent Examiner with Korean Intellectual Property Office, Daejeon, Korea, from 2004 to 2017. In 2017, he joined the Department of Electrical Engineering, Incheon National University, Incheon, Korea, where he is currently working as an Associate Professor. His current research interests include computational electromagnetics, numerical analysis, and IoT applications with machine learning.