Computers, Materials & Continua CMC, vol.65, no.2, pp.1181-1201, 2020

Success Rate Queue-Based Relocation Algorithm of Sensory Network to Overcome Non-Uniformly Distributed Obstacles

Sooyeon Park[1], Moonseong Kim[2] and Woochan Lee[1, *]

Abstract: With the recent development of big data technology that collects and analyzes various data, the technology that continuously collects and analyzes the observed data is also drawing attention. Moreover, its importance is growing in data collection in areas where people cannot access. In general, it is not easy to properly deploy IoT wireless devices for data collection in these areas, and it is also inappropriate to use general wheel-based mobile devices for relocation. Recently, researches have been actively carried out on hopping moving models in place of wheel-based movement for the inaccessible regions. The majority of studies, however, so far have unrealistic assumptions that all IoT devices know the overall state of the network and the current state of each device. Moreover, various physical terrain environments, such as coarse gravel and sand, can change from time to time, and it is impossible for all devices to recognize these changes in real-time. In this paper, with the migration success rate of IoT hopping devices being relocated, the method of estimating the varying environment is proposed. This method can actively reflect the changing environment in real-time and is a realistic distributed environment-based relocation protocol on behalf of non-realistic, theory-based relocation protocols. Also, one of the significant contributions of this paper is to evaluate its performance using the OMNeT++ simulation tool for the first time in the world to reflect actual physical environmental conditions. Compared to previous studies, the proposed protocol was able to actively reflect the state of the surrounding environment, which resulted in improved migration success rates and higher energy efficiency.

Keywords: Mobile IoT, hopping sensor, sensory data networking, relocation protocol, simulation.

1 Introduction

Big data and AI technologies are usually focused on data analysis. If there is a problem with data collection, it is complicated to find an abnormal point due to an infinitely large volume of data. Therefore, the technology to continuously collect data from observation areas has recently become a crucial issue. The rapid development of IoT devices enabled

[1] Department of Electrical Engineering, Incheon National University, Incheon, 22012, Korea.

[2] Department of Liberal Arts, Seoul Theological University, Bucheon, 14754, Korea.

[*] Corresponding Author: Woochan Lee. Email: wlee@inu.ac.kr.

Received: 28 April 2020; Accepted: 01 June 2020.

data collection in various fields [Alioto and Shahghasemi (2018)]. To be specific, continuous data collection by small IoT devices also enabled big data analysis in areas that are inaccessible to humans [Chen, Mao and Liu (2014); Yin and Wei (2018)]. Deployment of small IoT devices via unmanned aerial vehicles such as drones can collect data from inaccessible areas. However, it is challenging to deploy IoT sensor nodes appropriately in the desired areas, which can lead to distortions in the analysis of collected data. Moreover, small IoT devices have inherent energy depletion problems [Zhang, Fang, Zhao et al. (2020); Yaqoob, Ahmed, Hashem et al. (2017)].

There are many applications where wireless data network technology using small IoT devices (called sensor nodes) is utilized. In general, dozens or hundreds of sensor nodes can be deployed in earthquake, radiation spills, or enemy lines where military activity is underway to collect various information. The characteristics of these areas are those that cannot be accessed by civilians due to harmful substances, rough territory with many obstacles, or unintended exposure to the enemy. Therefore, it would be essential to scatter small devices in a vast, rough area to obtain valuable information. However, there is still a limitation that they can be improperly deployed. Furthermore, after deploying small sensor nodes, assume that data collection is excessively frequent in specific areas. Some sensors may run out of energy earlier than others due to the ongoing collection and transmission of data in such areas. This situation is called a sensor node failure, and in the worst-case scenario, communication on the entire network will be lost, and the desired data cannot be collected. An area where a certain amount of devices fail or fail to deploy correctly, and thus fail to obtain the desired data, is called sensing holes [Kim, Park and Lee (2019)].

In the past, various studies have been carried out to prevent sensing holes occurrence, including how to minimize energy consumption by adjusting the sensor's active and idle states, or how to set energy-efficient paths. Still, they have not thoroughly addressed the faults of sensor nodes caused by energy problems [Kim, Park and Lee (2018); Wang, Gao, Yin et al. (2018); Wang, Gao, Liu et al. (2019)]. Therefore, the most realistic solution is to have a relocation protocol to restore the sensing hole by moving other devices to the area where the sensing hole occurred. For this reason, research on mobile sensors has begun to draw attention. Mobile sensors can move to dark areas, where there is no information, to collect data, and can also continue tasks by replacing energy-starved sensor nodes. Initial studies of the mobile sensor were wheel-based moving models, but there are many physical constraints to these movements. In other words, energy consumption should be further considered for wheel drive, and in reality, wheel-based action is not appropriate in very rough or obstructed areas.

In order to overcome the limitations of these movements, hopping-based moving models were introduced. Mobile nodes are biomedically designed so that they can jump on their own like locusts and move to the desired direction. The research article from Zhang et al. [Zhang, Chen and Yang (2016)] provides detailed performance comparisons for the maximum jump height, travel distance, etc., for various jump-capable robots. Also, recent studies have been actively carried out on the relocation of hopping sensors to restore sensing holes [Cintron (2013); Snyder (2014); Senouci and Mellouk (2016)]. However, most of the studies so far have made a non-realistic assumption that the cluster (the area of interest properly divided, the cluster's set is the entire area) because the header sensor

nodes understand all the information in the entire network area. No matter how small the entire network area is, setting up information exchanges and paths between the entire cluster headers is not only very difficult in reality, but also involves the transaction of numerous control messages.

Recently, our research team has solved these problems [Kim, Park and Lee (2019); Park, Kim and Lee (2020)]. First of all, the cluster header does not need to know information from other cluster headers or all networks. It is a distributed networking-based relocation protocol of restoring sensing holes by requesting the sensor nodes to nearby cluster headers. Of all the sensor nodes in a nearby cluster, appropriate nodes can be selected based on intercommunicated information and moved to nearby sensing holes. However, so far, various relocation algorithms have assumed an ideal environment for all areas where sensors are placed. Previous article from Kim et al. [Kim, Mutka and Choo (2010)] has first studied how to relocate hopping sensors depending on the degree of obstacles, but there is an unrealistic assumption that deployed sensors are identifying the degree of obstacles in all regions. Another study from Park et al. [Park, Kim and Lee (2020)] tried to determine the degree of obstacles through moving sensor nodes, but it is still difficult to predict a realistic environment. In this paper, we propose an active estimation method of the obstacle level of the surrounding environment by grasping the condition of the moving sensors moving by the cluster header and also propose a relocation scheme considering the active obstacle estimation method. Besides, it can be significant that the first realistic experiment was conducted using OMNeT++ [OMNeT (2020)].

The rest of this paper is organized as follows. Chapter 2 describes our prior studies of relocation protocols, and Chapter 3 provides detailed descriptions of proposed protocols, along with scenarios. In Chapter 4, performance analysis is done through simulation, and in Chapter 5, this paper is concluded.

2 Previous work

This chapter describes a preliminary study of our proposed protocols in Chapter 3. Our research group has conducted many preliminary studies over a long period to propose a realistic relocation protocol. Among them, a simple summary of the studies was made on underlying assumptions, protocols for the redeployment of distributed environments, and methods to consider the surrounding environment.

2.1 Basic assumptions and network components

In order to collect and analyze data of a region of interest, first, small mobile IoT devices are randomly distributed. They can be distributed randomly using drones, etc., as shown in Fig. 1, in areas that are difficult to access. Thereafter, the entire area is divided into cluster zones of appropriate size with an appropriate clustering algorithm, and a cluster header is selected for each cluster zone. The cluster header periodically communicates with mobile IoT devices (hereinafter referred to as hopping sensor node members) in its cluster zone and can manage representative information of each node. The primary purpose of this study is to study the technique of relocating the hopping sensor nodes, so it is assumed that the clustering and the selection of headers are possible in various ways [Rostami, Badkoobe, Mohanna et al. (2018)]. Again, the clustering and the header selection is not our concern in this paper.

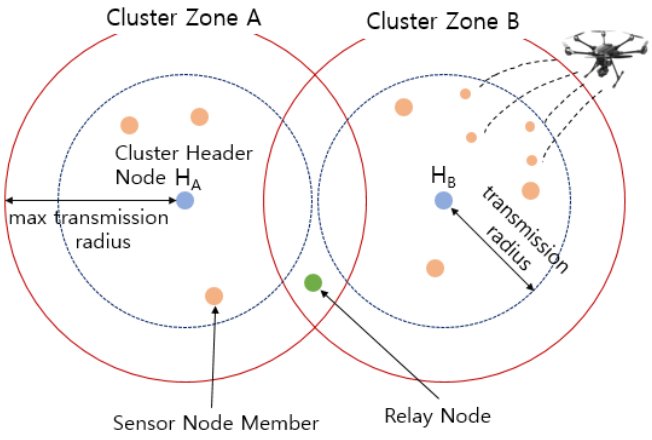


Figure 1: The terms in hopping sensor networks

Considering the connectivity problem of the wireless data network, communication between all hopping sensor node members and the cluster header can send and receive messages by jumping to an appropriate height [Cintron, Pongaliur, Mutka et al. (2012)]. Moreover, the sensor nodes include a GPS unit capable of determining their current location [Sabor, Sasaki, Abo-Zahhad et al. (2017)]. Fig. 1 briefly explains the terminology used in this paper. All hopping sensor nodes can be 'cluster header nodes' or 'sensor node members.' Since the maximum transmission area when the cluster header performs the maximum jump is defined as a 'cluster zone,' there is a high possibility that direct communication between cluster headers is impossible. Some sensor node members near an area intersecting the cluster zones, i.e., the maximum transmission radius of the cluster header, are likely to communicate with more than one cluster header. This sensor node is called a 'relay node,' and the role of the relay node will serve to relay communication between cluster headers [Kim, Park and Lee (2019)].

2.2 Relocation protocol in a distributed environment

Unlike the past various hopping sensor relocation protocols, which are difficult to apply in reality, research paper from Kim et al. [Kim, Park and Lee (2019)] proposed a relocation protocol suitable for a distributed environment so that it can be applied to a real environment. In other words, the protocol includes the most practical methods, such as a mechanism in which the cluster header recognizes the sensing hole and a request to surrounding clusters for immediate sensing hole recovery. Fig. 2 summarizes the underlying distributed environment relocation protocol. In each cluster zone, the cluster headers (H_A, H_B, H_C, H_D) can periodically check the status of sensor node members in their zone using broadcasting of the HELLO message. If less than a certain number of sensors are detected (as determined in the initial network policy), the cluster header can determine for itself that its zone has become a sensing hole. Here, the cluster header H_C can detect the number of node members less than three, and it can be recognized that its zone has become a sensing hole. The strategy of relocation protocol to recover this is as follows.

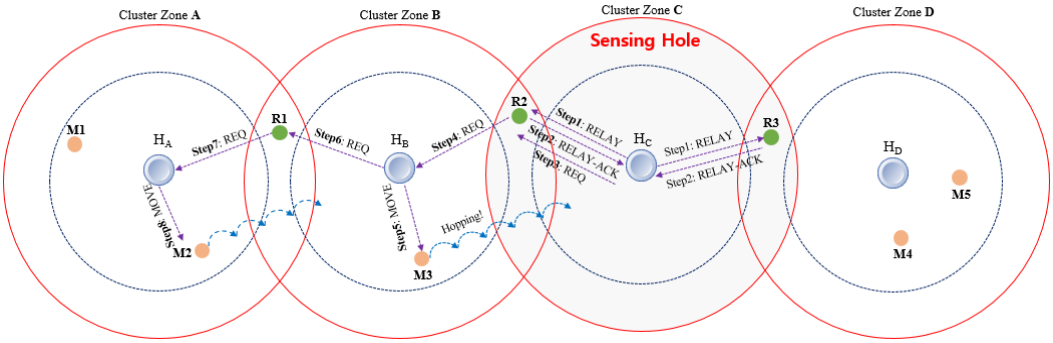


Figure 2: Relocation algorithm to recover a sensing hole

Step 1. The cluster header H_C broadcasts a RELAY message to all its relay nodes (R2, R3) to request one sensor node member from the neighboring cluster zones (Cluster Zones B, D).

Step 2. Each relay node immediately sends a RELAY-ACK message in response to the RELAY message, where the response from R2 arrives at H_C the fastest. The response of R3 to be received later is ignored.

Step 3. The cluster header H_C sends a REQ message to the selected relay node R2 to request one sensor.

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

Step 5. The cluster header H_B selects M3 as a sensor node member that can move from its zone to the neighboring zone and sends a MOVE message to move M3 to Cluster Zone C. The member M3 receiving the message moves to the neighboring cluster zone.

Step 6. At the same time, the cluster header H_B predicts that its zone will also be a sensing hole and sends a REQ message requesting one sensor to the relay node R1.

Step 7. Relay node R1 delivers the REQ message to the cluster header H_A.

Step 8. The cluster header H_A selects M2 from among sensor node members in its zone and commands M2 to move to cluster zone B.

As a result, one sensor is properly added within each cluster zone, and the sensors are relocated so that all sensing holes of the entire network can be recovered.

2.3 Relocation protocol considering terrain conditions

The environment in which the hopping sensor must be distributed through an unmanned aerial vehicle is very different from the typical terrain. Therefore, by combining the terrain information for obstacles around the cluster zone in addition to the study from Kim et al. [Kim, Park and Lee (2019)] in Section 2.2, it is possible to increase the relocation rate of the hopping sensor [Park, Kim and Lee (2020)]. If the cluster header of the sensing hole does not consider the surrounding environment, the satisfaction of the sensor node movement request for its neighbor zones is inevitably insufficient. Since there is still a sensing hole that cannot be overcome and persists, the number of messages

requesting neighboring zones is bound to increase.

This situation places a heavy load on the entire network and must be very negative in terms of energy efficiency. Indirectly predicting the distribution of obstacles (stones, puddles, mud, etc.) around the sensing hole is very important in reality, and we will look at the basic mechanism through the example below.

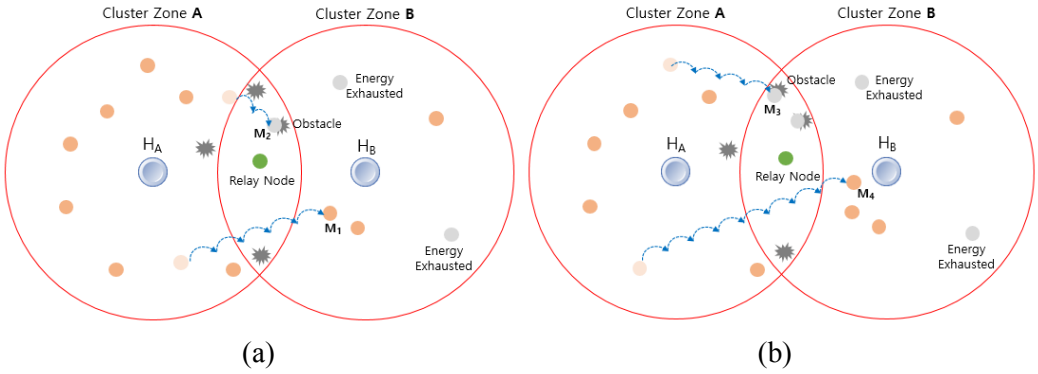


Figure 3: An example of predicting obstacle level nearby sensing hole

First, let us look at the relocation scenario of ordinary node members who do not consider the existence of obstacles. In cluster zone B of Fig. 3(a), two members exhausted energy, resulting in a node failure. After a while, header H_B determines that his zone is in the sensing hole state, and requests two node-members to cluster zone A (first REQ). The two members (M1, M2) who are ordered to move in Zone A move to Zone B. One member (M1) made a successful move, but the other (M2) was caught by an obstacle and failed to move to Zone B. The header of zone B determines that it is a sensing hole again, and requests another member movement to zone A (second REQ). In Fig. 3(b), the member M3 of Zone A fails to move to Zone B because of an obstacle again during the movement. In order to overcome the sensing hole, the header of zone B again requests another member movement to zone A (third REQ), and the member M4 succeeds in the movement and can recover the sensing hole. We have seen that three relocations have been performed to restore the first sensing hole.

Second, we will consider relocating node members when considering the existence of obstacles. In cluster zone B of Fig. 3(a), two members exhausted energy, resulting in a node failure. After a while, Header H_B decides that it is a sensing hole and tries to request two node members to Zone A. First, in the relocation protocol of Park et al. [Park, Kim and Lee (2020)], the previous work of our research group, the number of members to be requested is calculated in consideration of the 'transfer success rate.' While continuously relocating, the 'success rate of migration' p is continuously updated, and the number of requested members (cnt) considering each p-value is calculated as follows in consideration of the surrounding environment.

p=(number of members successfully migrated)/(number of members requested) (1)

cnt=Ceiling [(number of members needed)*(1+(1-p))] (2)

After setting the initial value of p to 1, the number of member requests is calculated as Ceiling [2×(1+(1-1))], and two member movements are requested to zone A (first REQ). In Fig. 3(b), two members who are commanded to move in Zone A move, and as in the first scenario, one member (M1) makes a successful migration, but the other (M2) is caught in an obstacle and moves to Zone B. The move failed. The header of zone B updates the value of p to 1/2 and determines that it is a sensing hole. The number of members to request to Zone A is Ceiling [1×(1+(1-1/2))], and the header of Zone B requests the movement of two members (second REQ). In Fig. 3 (b), of the two members of Zone A, one member (M3) fails to move to Zone B, as in the previous scenario. However, the other member M4 can successfully move to zone B and overcome the sensing hole state of zone B.

Looking at the previous two scenarios, it can be considered that the distribution of obstacles between clusters can be measured by considering the success rate of migration. Accordingly, it can be seen that considering the success rate of migration for the relocation of the sensor node member, it is possible not only to enable rapid sensing hole recovery but also to reduce control messages such as the number of request messages across the network.

3 Proposal of relocation protocol based on the prediction of the surrounding environment

In this chapter, we look at what problems the method of considering the obstacle state in the previous study has and propose an appropriate method for predicting the surrounding environment.

3.1 A Study on the prediction of the surrounding environment

In the paper from Park et al. [Park, Kim and Lee (2020)], Eq. (1), which is the successful movement rate of the hopping sensor node members, is used to consider the surrounding obstacles. However, it can be confirmed that there is a difficulty in node members moving between the cluster zone and the cluster zone, and predicting the large area with just one movement. Fig. 4 illustrates this example as follows.

In the example of Fig. 4, suppose that at least 5 sensor node members must be retained for sensing hole recovery. As shown in Fig. 4(a), the header HB of cluster zone B detects the sensing hole and requests 5 members to neighbor cluster zone A (first REQ message). In Fig. 4(b), five members selected by the header H_A are moving to the sensing hole. One of the members became a fault due to obstacles. As shown in the first row of the table in Fig. 5, the header H_B rewrites the result of the first REQ message; the migration success rate is updated to 4 move successes out of 5 requests, that is, 4/5=0.8.

In Fig. 4(c), the header H_B that failed to recover the sensing hole uses Eq. (2) (Ceiling [1×(1+(1-0.8))]=2) and transmits the two members to the neighboring cluster zone A through the second REQ. The two members selected in Fig. 4(d) fail to move due to obstacles. As shown in the second row of the table in Fig. 5, the header HB updates the migration success rate to 0/2=0.0 as a result of the second REQ message.

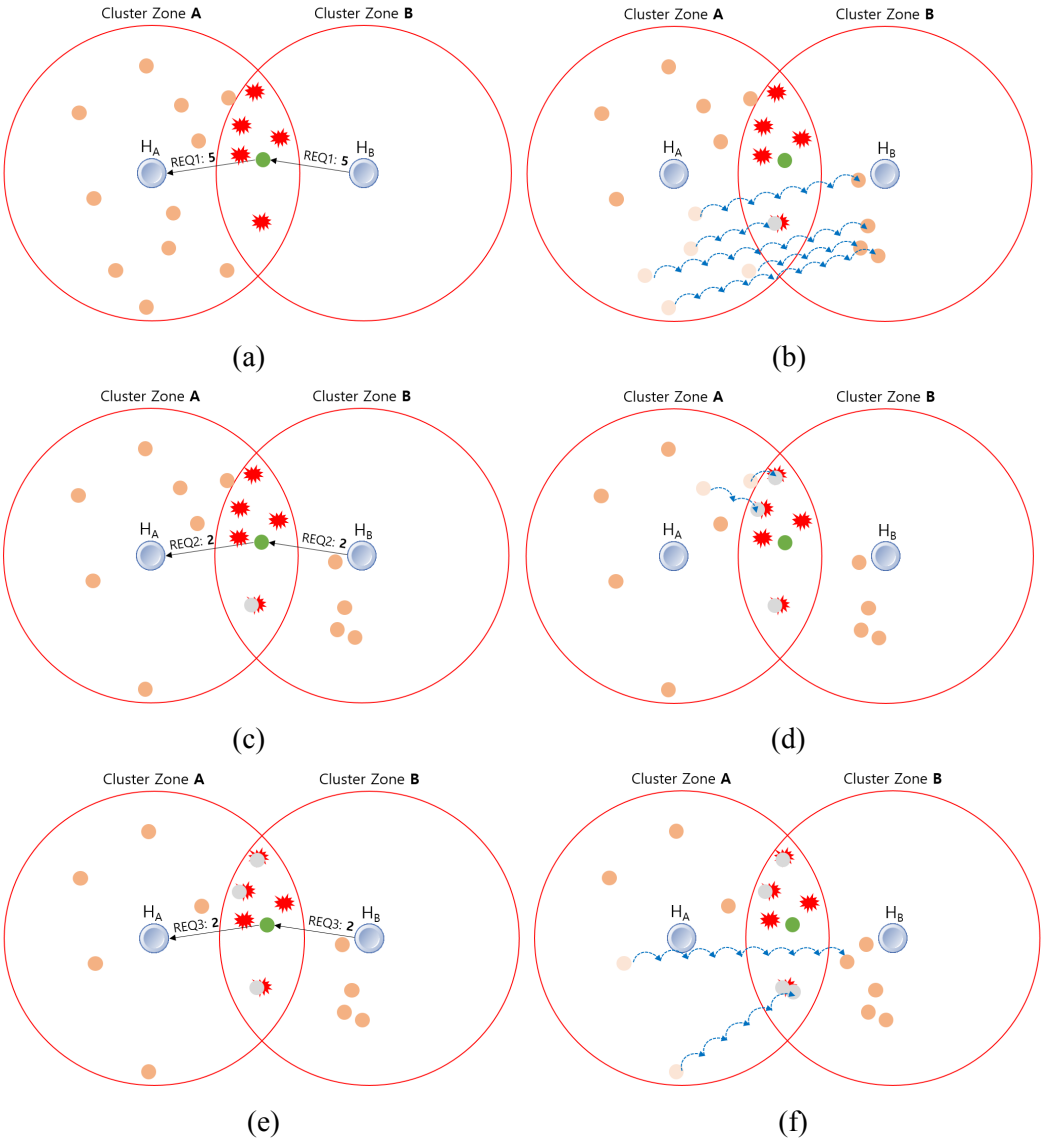


Figure 4: Obstacle prediction mechanism [Park, Kim and Lee (2020)]

Again, as shown in Fig. 4(e), the header H_B requests two members to the cluster zone A through the third REQ. Finally, as shown in Fig. 4(f), one member among the moving members can arrive at the cluster zone B and recover the sensing hole. As shown in the third row of the table in Fig. 5, the migration success rate of the move is updated to 1/2=0.5 as a result of the third REQ message.

Through the three REQ messages, it was possible to overcome the status of the sensing hole in cluster zone B, but let us look at the change in the migration success rate. That is, in this method, it is assumed that the obstacle state between the cluster and the cluster can be predicted through the success rate of migration.

However, while the state of the obstacle is fixed, the state of the obstacle (1-*p*), which is considered as a change in the success rate of migration, is dramatically changing to 20%, 100%, and 50%. For this reason, it is judged that an unreasonable assumption is made that the distribution of obstacles in a large area between clusters can be estimated as a result of only one member request. If this assumption is appropriate, the distribution of obstacles will be evenly distributed, unlike the example in Fig. 4. The example in Fig. 4 is an example where the obstacle terrain is biased to one side.

# of REQ	current # of members	# of members needed	# of members requested	# of members migrated	current success rate
1	0	5	5	4	0.8
2	4	1	2	0	0.0
3	4	1	2	1	0.5

Figure 5: A table of H_B to update the success rate of migration in Fig. 4

3.2 Proposal for accurate prediction of the surrounding environment

In order to more accurately predict the environment, it is most appropriate to consider the distribution of success rates for request events. This is because the nature of the distribution will converge to a specific model as the size of the sample set for request events is increased to some extent. Fig. 6 demonstrates a simple example of two sample sets, as shown in Fig. 4.

In the example of Fig. 6, suppose that it is necessary to have at least 5 sensor node members for sensing hole recovery. As shown in Fig. 6(a), the header H_B of cluster zone B detects the sensing hole and requests 5 members to neighbor cluster zone A (first REQ message). The calculation of the requested number '5' uses the average value of the migration success rates stored in the queue holding the success rate. Initialize all the internal values of the queue that will initially store the success rates, and assume the length of the queue is 2 (i.e., write P2 [1, 1]). Then, the average P^* of the values stored in P2 is the average of the success rates, and it can be assumed to be 1 as the initial value. If Eq. (2) is modified as follows, the number of requests (*cnt*) to be inserted into REQ is calculated by Eq. (3).

$$cnt = \text{Ceiling}\,[(\text{number of members needed}) \times (1+(1-P^*))] \quad (3)$$

In Fig. 6(b), five members selected by the header H_A are moving to the sensing hole. Suppose that only one member of the group moved to the sensing hole. In other words, four members became 'fault' due to the influence of obstacles. As shown in the first row of the table in Fig. 7, the header H_B updates the current migration success rate to 1/5=0.20, which is one move out of five requests, as a result of the first REQ message. Also, the status of the queue is updated to P2 [1.00, 0.20] so that P^* is 0.60.

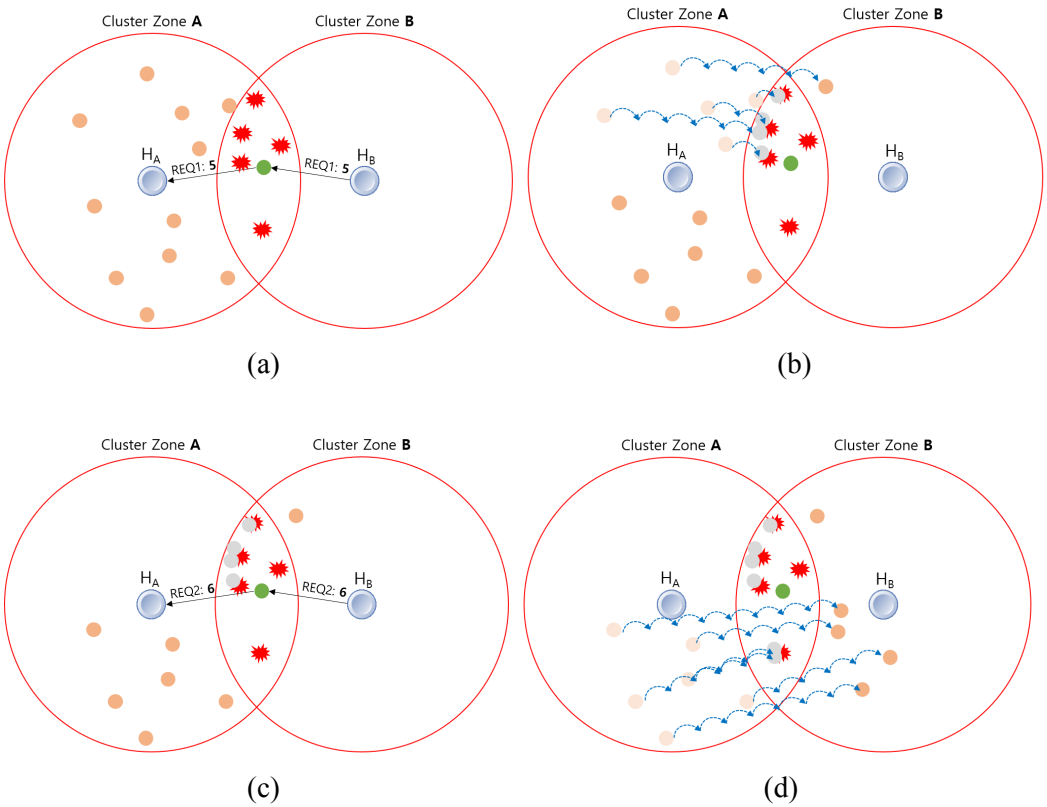


Figure 6: An example of the proposed algorithm

Cluster Header H_B								
# of REQ	current # of members	# of members needed	previous success rate	# of members requested	# of members migrated	current success rate	average success rate (queue length: 2)	terrain condition expected
1	0	5	1.0	5	1	0.20	0.67	0.33
2	1	4	0.2	6	4	0.67	0.44	0.56

Figure 7: A table of H_B to update the level of the terrain condition expected in Fig. 6

In Fig. 6(c), the header H_B that did not remove the sensing hole chose the 6 members calculated using the formula (3) (Ceiling [4×(1+(1-0.60))]=6) and transmitted them to the neighbor cluster zone A as a result of the second REQ. The six-member group selected in Fig. 6(d) can recover the sensing hole when four members of the moving members arrive at cluster zone B. Then, as shown in the second row of the table in Fig. 7, the success rate of the migration is updated to 4/6=0.67 as a result of the second REQ message. Also, the queue is updated with P2 [0.20, 0.67], so P^* is 0.44.

Unlike Fig. 4, through the two REQ messages, it was possible to overcome the sensing hole state of cluster zone B. Here, the state of obstacles ($1-P^*$) as seen by the change in the average of the migration success rates (by the queue length) is 33% to 56%. The examples in Figs. 4 and 6 assume that the obstacle between each cluster zone is 50% in

the area, and the 50% assumes an obstacle environment in which 40% is distributed in the upper half and 10% in the lower half.

In this environment in which the distribution of obstacles is not uniform, the number of REQ messages can be reduced by one even if only the length of the queue in which migration success rates are to be stored is two. Also, it was possible to intuitively observe how the distribution status of all obstacles approached 50%. In this paper, we propose a method to stably calculate the number of requests from movable hopping sensor node members by properly guessing the distribution rate even for obstacles that are not uniformly distributed.

3.3 Descriptions of the proposed protocol

In this section, for the operation of the protocol in which the newly proposed method operates, the message type and the protocol implemented in each cluster header and hopping sensor node member are defined. First, all message types and their detailed definitions are shown in Tab. 1.

Table 1: Types of messages and descriptions

Type	Descriptions
HELLO	· A cluster header broadcasts this message periodically. · {message type, source address, destination address (broadcasting)}
HELLO-ACK	· Replied message for HELLO message · {message type, source address, destination address, relay node(T/F)}
RELAY	· A header of sensing hole multicasts it to all of its relay nodes to choose a relay node to forward a message to neighbor headers. · {message type, source address, destination address (multicasting)}
RELAY-ACK	· Replied message for RELAY message · {message type, source address, destination address}
REQ	· A header of sensing hole requests members needed. · {message type, source address, destination address, number of requested (cnt), sensing hole address, sensing hole GPS info., etc.}
ADV	· The cluster header that receives REQ broadcasts this message to choose members to move to the sensing hole. · {message type, source address, destination address (broadcasting)}
ADV-ACK	· Replied message for ADV message · {message type, source address, destination address, some info.}
MOVE	· The cluster header multicasts this message to members, which sent ADV-ACK message, to move to the sensing hole. · {message type, source address, destination address (multicasting), sensing hole address, sensing hole GPS info.}

As shown in Fig. 8, the cluster header periodically broadcasts a HELLO message to identify its hopping sensor node members. The broadcasting period means an interval at

which a timer (helloMsgTimer) is executed and ends. After receiving the HELLO message, the hopping sensor node determines the level of its own jump by grasping the received signal strength and then transmits the HELLO-ACK message to the cluster header. The timer can be set to a time when all sensor node members initially deployed can send HELLO-ACK messages to the cluster header (e.g., set in our simulation to 30 minutes).

However, as the number of active sensors decreases due to energy defects, the fixed timer, which was initially set long enough, can leave the sensing hole longer. For example, even though a sensing hole has occurred, checking at fixed interval will make the cluster header unable to recognize the sensing hole immediately. Thus, adjusting the timer after the cluster header has determined the number of active sensors can efficiently reduce the time spent recognizing sensing hole occurrence. These discussions could be another valuable research topic to go on in the future.

Fig. 9 shows the procedure for determining whether a hopping sensor node member is a relay node after receiving a HELLO message. If a sensor node member receives HELLO messages from multiple cluster headers, it remembers each header as its own cluster header and recognizes that it has become a relay node.

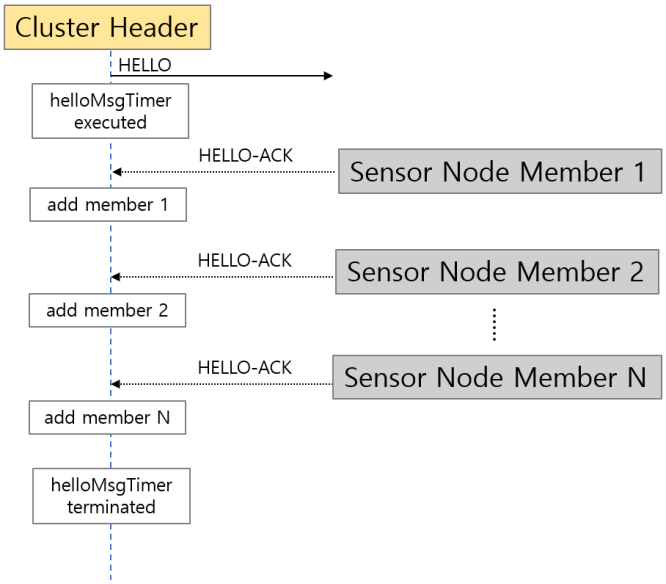


Figure 8: A procedure for verifying sensor node members in the cluster header

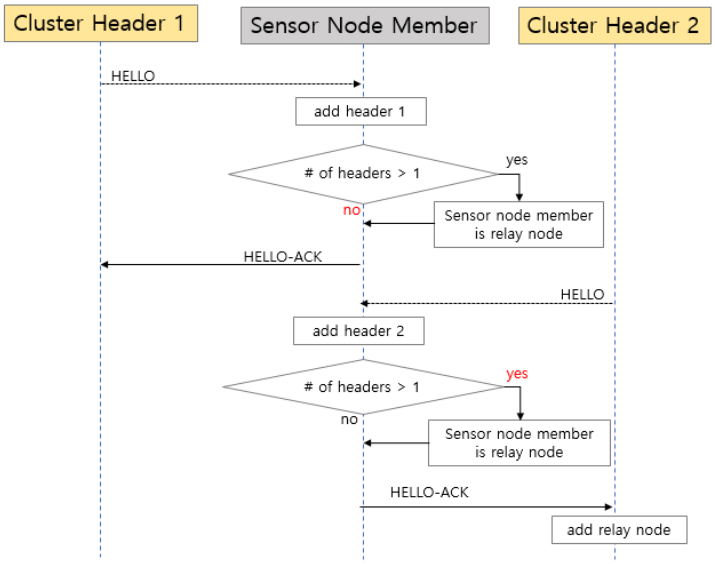


Figure 9: A procedure for determining relay node of hopping sensor node member

In Fig. 10, when the timer of the cluster header ends, it is determined whether or not its cluster zone is a sensing hole by checking the number of registered members. If it is not a sensing hole, the cluster header periodically sends a HELLO message again and restarts the timer. If it is judged as a sensing hole, it sends a RELAY message to its relay nodes. As soon as the relay nodes receive the RELAY message, they immediately send the RELAY-ACK message back to the cluster header. The cluster header can sequentially receive multiple RELAY-ACK messages. The relay node of the first received RELAY-ACK message is selected as a relay node capable of delivering a message to the header of the neighboring cluster, and other incoming RELAY-ACK messages are ignored.

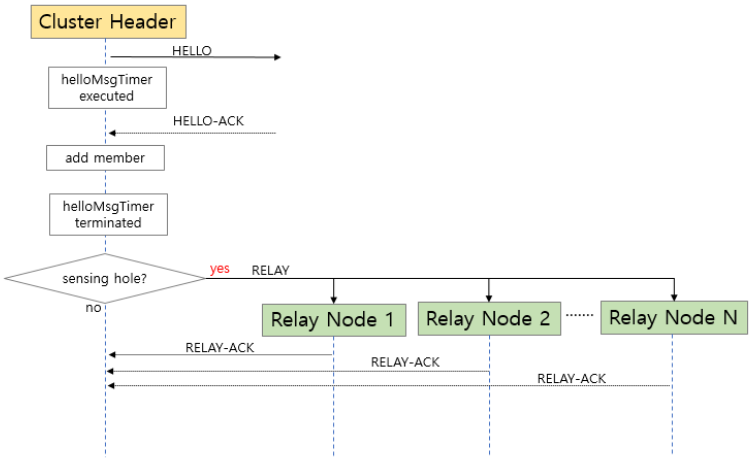


Figure 10: A procedure for determining the sensing hole of the cluster header

The cluster header of the sensing hole delivers node member requests for sensing hole recovery to the header of the neighbor cluster through a REQ message. The number of node members required for recovery may be calculated using a difference (T-C) between a threshold value for determining a sensing hole (T) and a current number of sensors (C). However, in reality, since the environment around the cluster header is rough terrain, there is a high possibility that the sensing hole will continue when the sensors moved less than the requested number of (T-C). Therefore, the cluster header can calculate the number of member sensor requests (cnt) that can overcome the sensing hole using Eq. (4) below with Eq. (3).

$$cnt=\text{Ceiling}\ [(T\text{-}C)\ (1+(1-P^{*}))] \tag{4}$$

where P^{*} is the average of the migration success rates of the queue P_L (queue length L) storing the migration success rates.

In Fig. 11, if the relay node receives the REQ message from the cluster header of the sensing hole, it transmits the REQ message to another cluster header of the relay node. Here, the source address of the REQ message is its own address, and the destination address is modified to the address of the cluster header to which the message is to be delivered.

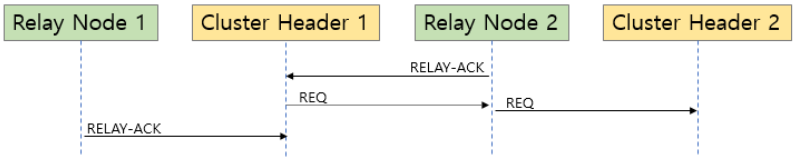


Figure 11: A procedure for handling REQ messages on the relay node

In Fig. 12, when the cluster header of the zone located in the neighborhood of the sensing hall receives the REQ message, the ADV message is broadcasted to determine the movable member among the member sensor nodes in its zone. Upon receiving the message, the member nodes transmit an ADV-ACK message containing their current physical information to the cluster header. Here, 'physical information' is a variety of information for a relocation strategy [Virdis and Kirsche (2019)] with current energy status, hopping mobility level, GPS coordinate information, and the like.

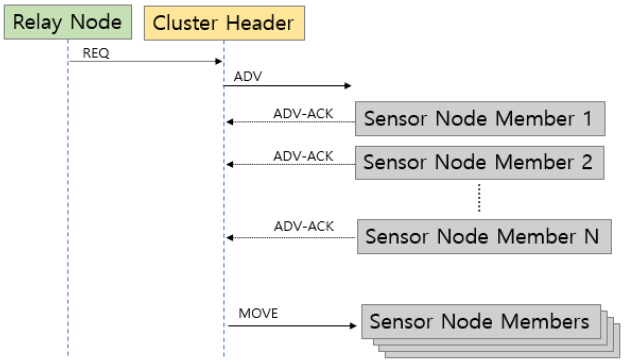


Figure 12: A procedure of handling REQ messages in the cluster header

The cluster header receives ACK messages and selects movable member sensor nodes. Then, the MOVE command message including the location information of the cluster header of the sensing hole is transmitted to the selected members. The member nodes receiving the MOVE message hop to the neighboring zone by using the GPS information of the cluster header of the zone to be moved.

4 Simulation result and analysis

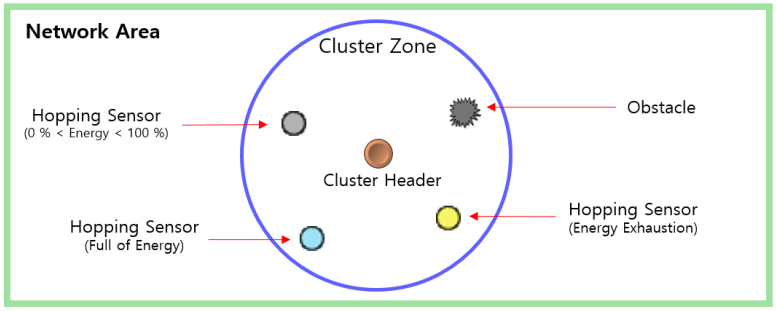
The performance evaluation so far from most of the hopping sensor relocation protocol studies has been unreliable because the protocol has been an unrealistic theory-based relocation protocol. However, one of the significant contributions of this paper is that OMNeT++ [Zarrad and Alsmadi (2017); Virdis and Kirsche (2019)] simulation is used to reflect the actual physical environment.

Table 2: Simulation environments

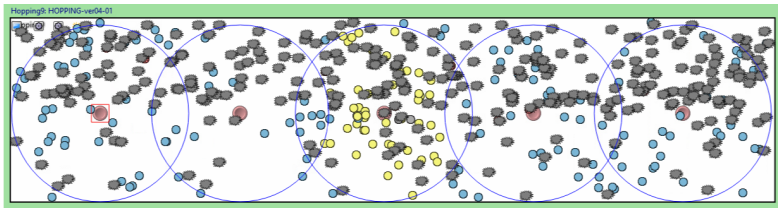
Parameters	Values
Network area	250 m×60 m
Number of total member sensor nodes distributed	200
Number of min members for each cluster zone to properly gather data	15
The probability that an obstacle exists over the entire network area	2%
Maximum distance that a member moves forward with one hopping	2 m
Maximum communication radius for each member	20 m
Maximum communication radius when highly jumping	29 m
Length of queue for the proposed algorithm	20

Tab. 2 describes the environment settings used in OMNeT++ for performance evaluation. As shown in Fig. 13(a), 200 hopping sensors were randomly sprayed in an area of 250 m×60 m to collect data. If there are less than 15 sensor nodes in the cluster zone, it is assumed to be a sensing hole. The communication of each sensor uses the IEEE 802.11 model, and it is assumed that the standard transmission radius is 20 m, and the maximum transmission radius when the hopping sensor jumps to the maximum height is 29 m. It was assumed that it is possible to move about 2 m forward in a single jump and that it is possible to hop up to 130 times. Hopping of 130 means the amount of movement that can move a diagonal line, which is the longest moving distance in the network area.

As shown in Fig. 13, obstacles are randomly generated with a value of 2% of the total area. Although the size of the obstacle seems very large, it was assumed that the obstacles were square with horizontal and vertical dimensions of 1 m, respectively. For visual effect, the obstacle display was made large in the picture. The distribution of the obstacles is not evenly distributed, and the obstacles are distributed with the weight of the obstacles in the upper and lower half being 80%:20%. That is, this is for reproducing a state in which the distribution of obstacles between clusters is not uniform and biased toward one side.



(a) legend



(b) Simulation finished

Figure 13: Simulation snapshot (finished state)

For the performance analysis of the relocation protocol, a sensing hole was generated in the middle of a given area, as shown in Fig. 13. Assuming that continuous data collection occurs in the middle cluster zone, a scenario is set in which the energy of sensor nodes in the middle zone is rapidly consumed. Each sensor in the middle cluster generates a data collection event with exponential distribution (5 minutes on average). For convenience, the initial energy value for sensing is set to 100, and energy is consumed by 1 for each event. Then, the cluster header determines the occurrence of the sensing hole (less than 15 sensors) with a 30-minute interval HELLO message. In order to perform a simulation of the relocation of sensor node members, it is assumed that sensors in different zones have no data collection. In Fig. 13, after the continuous data collection of the sensor, it is indicated in yellow that the energy is exhausted and the fault has occurred.

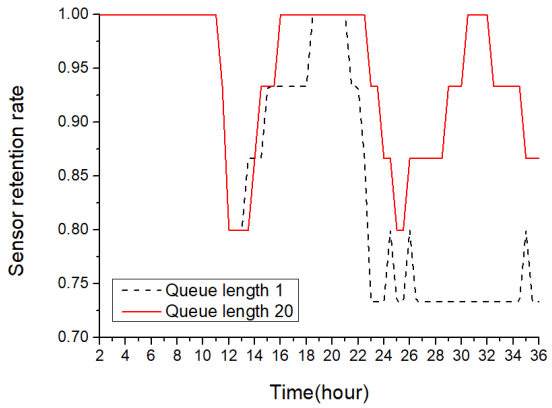


Figure 14: Sensor retention rate at the middle cluster

Fig. 14 shows the ratio of the minimum number of sensors (15) for the central cluster zone not to be a sensing hole in a time series of 36 hours. The average satisfaction rate for the entire simulation time can be seen that the proposed method improved by 7.9%. These results show that the proposed method is more active in overcoming the sensing hole. Moreover, even if a sensing hole still exists for each method, there will be an effect that the remaining sensors in the sensing hole continuously collect/deliver data to reduce the frequency of outliers at the observation point.

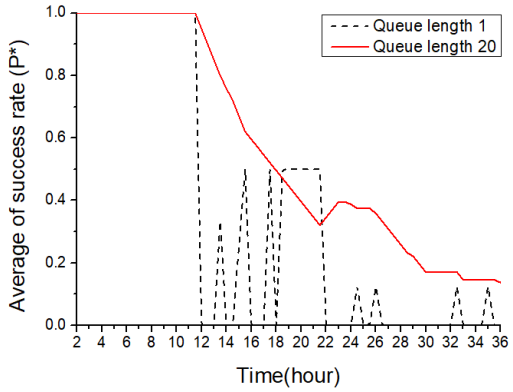


Figure 15: Average of migration success rate at the middle cluster

Fig. 15 shows the change in the average migration success rate. In the given environment, the distribution of obstacles is not evenly distributed, especially in the upper part. In this environment, if a different previous migration success rate is used each time, as in the case of one queue length, a subsequent migration success rate may change dramatically. On the other hand, it can be seen that the change in the average migration success rate considering 20 queue lengths is more stable than when only the previous migration success rate is considered (1 queue length case). Therefore, when requesting members from clusters in a nearby area, a more stable average migration success rate (in the case of 20 queue lengths) was used, so the degree of obstacles fixed around the sensing hole can be better predicted compared to the previous method (that is, 1 queue length case).

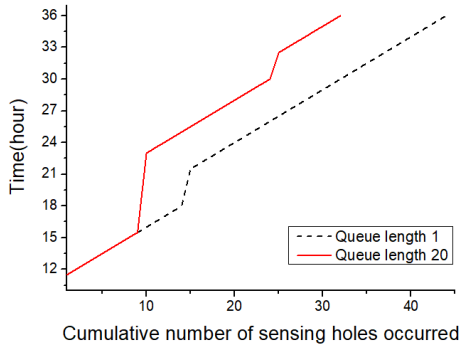
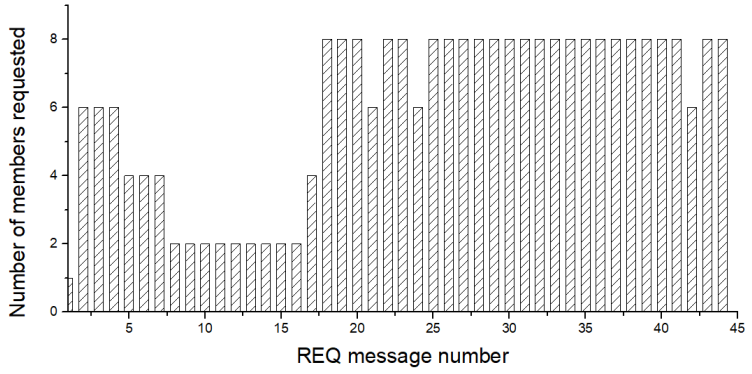


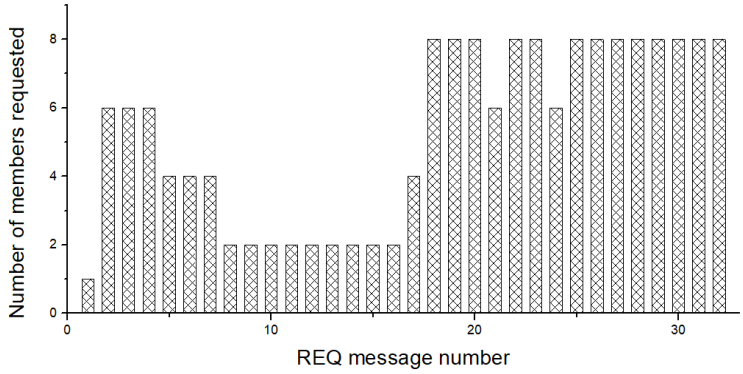
Figure 16: The moment for sensing hole appearance at the middle cluster

Fig. 16 indicates the frequency and time of occurrence in which the central cluster becomes a sensing hole. For 36 hours, 32 sensing holes occurred in the proposed method, but 44 sensing holes occurred in the previous method. Whenever a sensing hole occurs, a cluster header sends massive messages to members of its own zone, and also sends massive request messages to a neighboring zone. Therefore, the proposed method can reduce energy consumption for message transmission compared to the previous method.

Figs. 17 and 18 provide further details in terms of message transmission and energy consumption for the entire network area. Fig. 17 shows the number of members requesting for each REQ when a REQ message is generated for a member request that lacks the cluster header of the sensing hole. In addition, Fig. 18 accumulates the graph of Fig. 17 in chronological order, expressing the minimum number of members that have moved across the network since the start of observation. The difference between each graph over time can be regarded as an indicator for improving energy consumption across the network, and as a result, the proposed algorithm can save up to 36.9% of energy after 36 hours compared to the previous algorithm.



(a) Queue length 1



(b) Queue length 20

Figure 17: A bar graph of the number of members requested for each REQ message

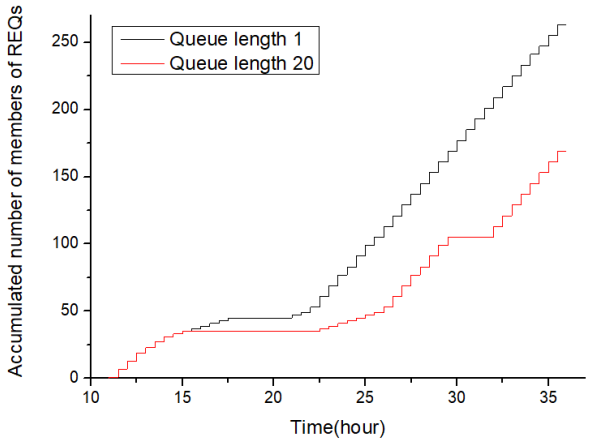


Figure 18: Accumulated number of members of REQs

5 Conclusion

The physical obstacles of the sensory network and the inherent energy deficiencies of micro IoT devices are not only adequate and hindering data collection, but can also lead to unintended consequences such as loss of reliability in big data analysis. Typically, the only way to quickly overcome the sensing hole is to relocate the movement sensors, but so far, most studies have been only theoretical studies that cannot be used in reality. Recently, our research team designed a relocation protocol based on a distributed environment that can dramatically solve these problems.

In this paper, a method for predicting the surrounding environment is proposed for the more successful relocation of hopping mobile IoT devices. In the existing method, the obstacle level between cluster zones was assumed to be the success rate for a single relocation request, but in this paper, the representative value of the distribution for several relocation request success rates was used to predict the environment of the obstacle. As a result, excellent sensor relocation results were obtained in the case where the obstacles were distributed non-uniformly and showed better sensing hole recovery ability and energy efficiency. Moreover, it is one of the most significant contributions to the design and performance analysis of protocols that can be used immediately in the real environment through simulation through OMNeT++.

Funding Statement: This work was supported by Research Assistance Program (2019) in the Incheon National University and the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science, ICT & Future Planning (No. NRF-2019R1G1A1007832).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Alioto, M.; Shahghasemi, M. (2018): The Internet of Things on its edge: trends toward its tipping point. *IEEE Consumer Electronics Magazine*, vol. 7, no. 1, pp. 77-87.

Chen, M.; Mao, S.; Liu, Y. (2014): Big data: a survey. *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171-209.

Cintr´on, F.; Pongaliur, K.; Mutka, M. W.; Xiao, L.; Zhao, J. et al. (2012): Leveraging height in a jumping sensor network to extend network coverage. *IEEE Transactions on Wireless Communications*, vol. 11, no. 5, pp. 1840-1849.

Cintron, F. (2013): *Network Issues for 3D Wireless Sensors Networks (Ph.D. Thesis)*. Michigan State University, USA.

Kim, M.; Park, S.; Lee, W. (2018): A robust energy saving data dissemination protocol for IoT-WSNs. *KSII Transactions on Internet and Information Systems*, vol. 12, no. 12, pp. 5744-5764.

Kim, M.; Park, S.; Lee, W. (2019): Energy and distance-aware hopping sensor relocation for wireless sensor networks. *Sensors*, vol. 19, no. 7, pp. 1567.

OMNeT (2020): *Website*, https://www.omnetpp.org.

Park, S.; Kim, M.; Lee, W. (2020): Energy-efficient wireless hopping sensor relocation based on prediction of terrain conditions. *Electronics*, vol. 9, no. 1, pp. 49.

Rostami, A. S.; Badkoobe, M.; Mohanna, F.; Keshavarz, H.; Hosseinabadi, A. A. R. et al. (2018): Survey on clustering in heterogeneous and homogeneous wireless sensor networks. *Journal of Supercomputing*, vol. 74, no. 1, pp. 277-323.

Sabor, N.; Sasaki, S.; Abo-Zahhad, M.; Ahmed, S. M. (2017): A comprehensive survey on hierarchical-based routing protocols for mobile wireless sensor networks: review, taxonomy, and future directions. *Wireless Communications and Mobile Computing*, vol. 2017.

Senouci, M. R.; Mellouk, A. (2016): *Deploying Wireless Sensor Networks: Theory and Practice*. Elsevier.

Snyder, M. E. (2014): *Foundations of Coverage Algorithms in Autonomic Mobile Sensor Networks (Ph.D. Thesis)*. Missouri University of Science and Technology, USA.

Virdis, A.; Kirsche, M. (2019): *Recent Advances in Network Simulation: the OMNeT++ Environment and Its Ecosystem.* Springer: Cham, Switzerland.

Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A. K.; Kim, H. J. (2019): An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *International Journal of Distributed Sensor Networks*, vol. 15, no. 3.

Wang, J.; Gao, Y.; Yin, X.; Li, F.; Kim, H. J. (2018): An enhanced PEGASIS algorithm with mobile sink support for wireless sensor networks. *Wireless Communications and Mobile Computing*, pp. 1-9.

Yaqoob, I.; Ahmed, E.; Hashem, I. A. T.; Ahmed, A. I. A.; Gani, A. et al. (2017): Internet of Things architecture: recent advances, taxonomy, requirements, and open challenges. *IEEE Wireless Communications*, vol. 24, no. 3, pp. 10-16.

Yin, B.; Wei, X. T. (2018): Communication-efficient data aggregation tree construction

for complex queries in IoT applications. *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3352-3363.

Zarrad, A.; Alsmadi, I. (2017): Evaluating network test scenarios for network simulators systems. *International Journal of Distributed Sensor Networks*, vol. 13, no. 10.

Zhang, L.; Chen, H.; Yang, Y. (2016): Review on mechanism and modeling of jumping robot. *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, pp. 255-260.

Zhang, W.; Fang, W.; Zhao, Q.; Ji, X.; Jia, G. (2020): Energy efficiency in Internet of Things: an overview. *Computers, Materials & Continua*, vol. 63, no. 2, pp. 787-811.