Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.033081
Article

Tech Science Press

check for updates

Hopping-Aware Cluster Header Capability for Sensor Relocation in Mobile IoT Networks

Moonseong Kim[1], Jaeyoung Park[2], Young-Joon Kim[3] and Woochan Lee[4,*]

[1]Department of IT Convergence Software, Seoul Theological University, Bucheon, 14754, Korea
[2]Department of Computer Engineering, Hongik University, Seoul, 04066, Korea
[3]Department of Electronic Engineering, Gachon University, Seongnam, 13120, Korea
[4]Department of Electrical Engineering, Incheon National University, Incheon, 22012, Korea
*Corresponding Author: Woochan Lee. Email: wlee@inu.ac.kr
Received: 07 June 2022; Accepted: 04 August 2022

Abstract: Mobile sensor nodes such as hopping sensors are of critical importance in data collection. However, the occurrence of sensing holes is unavoidable due to the energy limitation of the nodes. Thus, it is evident that the relocation of mobile sensors is the most desirable method to recover the sensing holes. The previous research conducted by the authors so far demonstrated the most realistic hopping sensor relocation scheme, which is suitable for the distributed environment. In previous studies, the cluster header plays an essential role in detecting the sensing hole and requesting the neighboring cluster to recover the sensing hole that occurred in the sensor node. However, the limitations of the cluster header in the previously proposed relocation protocol are not fully considered. Because the cluster header jumps more frequently than non-header nodes, its energy consumption is relatively high compared to other nodes. Therefore, it is most likely to lead to header node failure and can lead to data loss on the network. In this paper, the jumping ability and energy consumption of the cluster header are seriously considered. Additional ability to replace cluster headers in case of failure is also implemented. Simulation results show that the data collection time can be further increased, which demonstrates the validity of the proposed algorithms.

Keywords: Hopping sensor; mobile internet of things; relocation protocol; header node; numerical simulation

1 Introduction

Solid data collection is fundamental to several up-to-date technologies such as big data processing or data mining. However, previous attention may tend to focus on only analyzing a large amount of data neglecting the sustainability of collecting the data [1]. Usually, it is hard to find a region where the collected data is completely intact. Thus, in recent years, technologies that can collect data without defects or contamination have drawn more attention. For example, for such an area as a battlefield or contaminated area, several Internet of Things (IoT) devices can be employed to collect informative data [2–4]. Unmanned Aerial Vehicles (UAVs), like drones, are useful tools for deploying small sensors.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

However, it is tough to dispatch small sensors to completely cover the region of interest to collect the information stably. That means that it is likely to miss the representative data of the region of interest. Thus, small devices are likely to consume energy to collect unnecessary data leading to device failure. Specific areas where data collection is no longer available are defined as sensing holes. If the sensing holes cover the region of interest entirely, the whole network can be down.

Relocating the IoT sensors is an efficient way to remove the above sensing holes, which can resume data collection. Early research regarding wheel-based mobile sensors is limited to the obstacle-free area. Therefore, jumping-enabled (hopping) sensors were proposed to overcome the obstacle. Since the hopping sensors move by consecutive jumps, they can avoid rocks that hinder the desired path or escape from the sands [5]. In addition, the data transmission range can be controlled by setting jump height [6].

In recent years, several hopping sensor relocation algorithms have been studied. For example, recovering the sensing hole by borrowing the hopping sensor nodes from the shortest-path cluster zone was proposed [7]. Relocating the hopping sensor reflecting the degree of the obstacle was also proposed in [8]. However, some studies tend to establish routes for relocation, assuming that all cluster headers or nodes are possible to know all the information of the whole network [9,10]. It is apparent that the location and degree of obstacles of nodes in each area are not easily identified. Regardless of the size of the network, it is practically impossible for all cluster headers to exchange state information and establish routes for the nodes in real-time.

Based on our results of several studies related to the centralized hopping sensor relocations [8–10], for the first time, we solved the mentioned problem in a distributed manner [11]. Unlike the centralized manner, all sensor nodes do not need information about all sensor nodes and the overall network structure. It is a very realistic approach. When a cluster header requests some necessary hopping sensors to overcome its sensing hole status, the neighboring cluster header could help recover the sensing hole by supplying the needed sensors. However, the initial relocation protocol [11] had various limitations. First, a specific node that provides the fastest response among neighboring clusters is repeatedly used, so a state in which a specific node is requested only for a particular neighboring cluster may occur. It can cause ping-pong problems. The ping-pong problem is a status in which two specific cluster headers are repeatedly and continuously requesting the needed sensors from each other. Also, we could check that the hopping sensor node moves to a range near the cluster header even when the sensing hole is restored. The paper [12] solves this ping-pong problem and distributes the sensors evenly across the cluster area. Also, in the papers [13,14], the ping-pong problem is solved. In addition, the paper [15] extended the role of relay nodes. The relay node only performs the role of forwarding the message of the cluster header, solving the problem that there are many areas where data collection is not possible. Furthermore, we have conducted various researches in recent years [16,17].

This paper studies the limitations of a cluster header overlooked in previously proposed relocation protocols. The header node has to check the status of its cluster zone by periodically jumping and frequently transmitting messages to the neighboring headers to recover its sensing hole. The frequency of jumps is high, and the header node failure frequency is relatively high. Therefore, we propose a relocation protocol that lightly changes the role of the cluster header without entailing heavy load new clustering of the whole network area. In the performance evaluation section, the performance improvement of the proposed protocol is confirmed by analyzing the effects of existing problems.

This paper is organized as follows. Section 2 describes the previous work, and Section 3 describes the proposed relocation protocol. Section 4 analyzes the simulation and performance evaluation, and Section 5 concludes the paper.

2 Related Work

Recently we have researched various studies while proposing hopping sensor relocation protocols based on a decentralized manner [11–17]. In particular, the survey of various literature related to the hopping sensor is described in detail in [11]. This literature is also the first research in a decentralized manner. This section describes the well-known previous relocation protocols shortly. First, representative terms for a typical relocation method may be described as follows. As shown in Fig. 1, the hopping sensor is assumed to be distributed in an area where data collection is required but is difficult to access. All initially deployed sensors are appropriately clustered through various clustering algorithms and divided into multiple regions [18]. The sensor at the center of each divided area is called the "cluster header" (H_A, H_B, H_C), and every sensor in the same clustered area as the cluster header is called the "member node" (M1, M2, …). The cluster header tried to communicate periodically to manage information about member nodes. Here, it is assumed that the clustering method uses various well-known schemes, and since this is not the subject of this paper, further discussion is omitted.

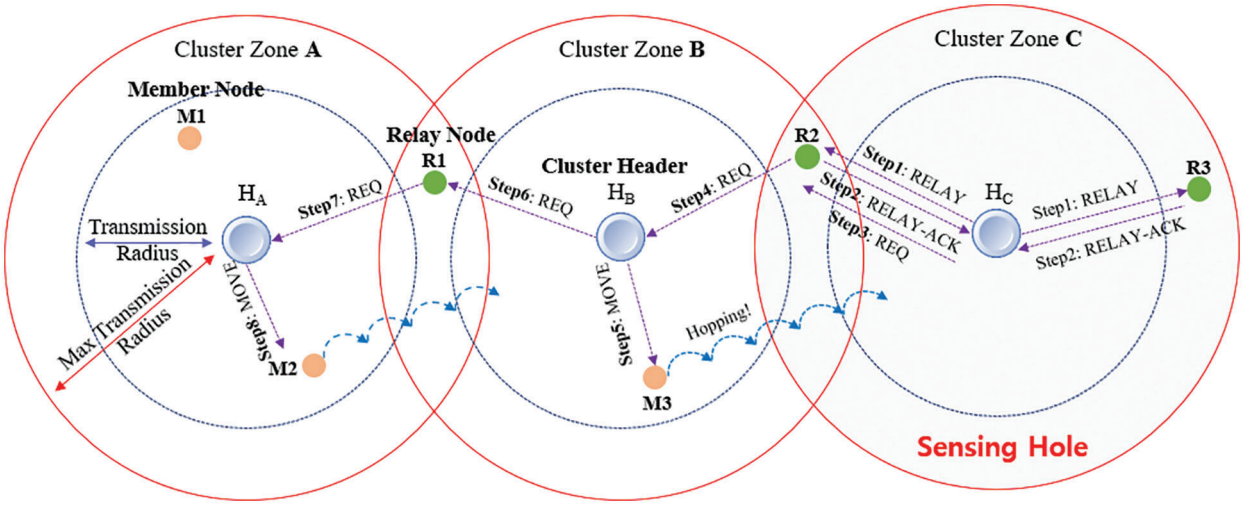


Figure 1: Representative relocation algorithm [11]

A hopping sensor can jump and communicate with nearby sensors within its transmission radius, and each sensor can use a Global Positioning System (GPS) unit to determine its location. As shown in Fig. 1, the blue dot-line indicates a hopping sensor communication radius on the ground. The red line also indicates the maximum transmission radius when data is sent with the highest jump, and here the maximum radius area can be defined as a "cluster zone." Therefore, direct communication between cluster headers could be highly likely to be impossible. However, it is possible for some member nodes to communicate with more than one cluster header in an area where multiple cluster regions overlap, i.e., near the maximum transmission radius of a particular cluster header. These member nodes are called "relay nodes." That is, the role of the relay node is to transmit data in the middle for communication between cluster headers. When the entire network area is clustered in general, each cluster area is determined based on the transmission radius of a sensor node. Since the hopping sensor can widen the transmission radius while jumping, the number of clusters can be reduced compared to the number of clusters in the typical sensor network. Therefore, it is inevitably essential to manage the jumping ability of the hopping sensor serving as a cluster header. Moreover, the jumping ability of the cluster header is critical because it has to manage many member nodes within a wide cluster zone and transmit/receive data with them.

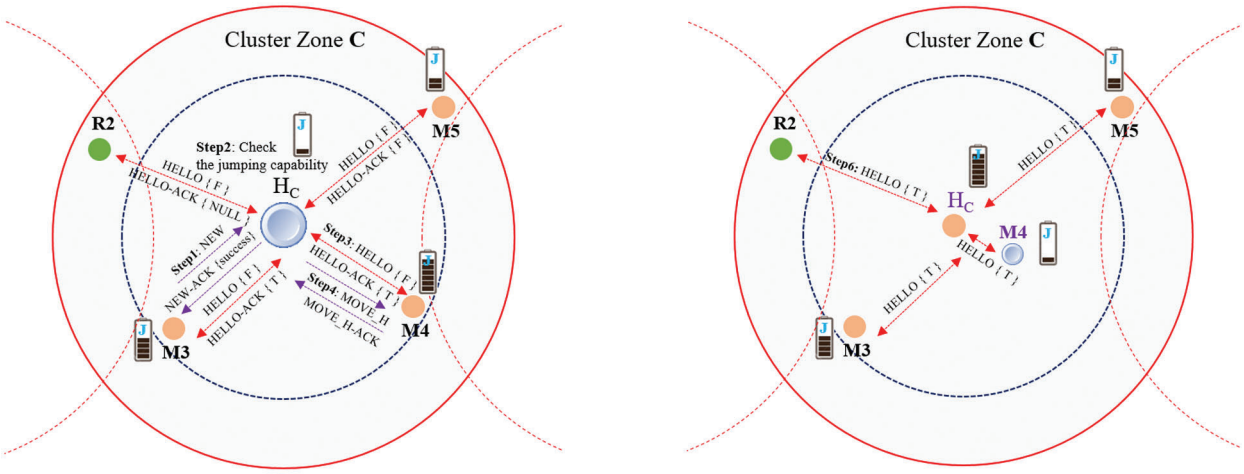
Fig. 1 illustrates a simple example of the most usual relocation protocol, and here message types of the protocol should follow the paper [11] as much as possible. Each cluster header periodically broadcasts Hello messages to its respective regions and continuously checks the status of its member nodes. The member node responds to a cluster header and informs each header that it is a relay node when more Hello-ACK messages than one are received. If a cluster detects fewer than the specified number of member nodes required by a cluster to collect data, the cluster header determines that its region has become a sensing hole. In Fig. 1, the cluster header H_C determines that a sensing hole has occurred and performs a relocation strategy to recover it.

In Step 1, the cluster header H_C broadcasts a RELAY message to all relay nodes R2 and R3 to try to request one member node to overcome the occurred sensing hole. In Step 2, each relay node immediately sends a RELAY-ACK message in response to the received RELAY message, where the response from R2 arrives at H_C the fastest. The R3 response to be received later is ignored. Here, the paper [15] deletes the use of RELAY and RELAY-ACK messages. Using such the message will likely cause well-known ping-pong problems because a specific relay node can be used continuously. In the paper [13], each relay node uses a timer set to a random time after receiving the RELAY message from the cluster header. When each timer is expired, it responds to the header with RELAY-ACK. It could solve the ping-pong problem of concentrating the request message on a specific relay node. In addition, the paper [12] uses a queue for managing priorities to select relay nodes. In the paper [13], each relay node manages its timer, but in the paper [12], the cluster header manages the priority for its relay nodes to forward REQ message to request some hopping sensors from the neighbor cluster header. The paper [14] used a random variable (a simple Monte Carlo method) to select each relay node in the cluster header. As mentioned above, the RELAY and RELAY-ACK messages are not also considered in the proposed protocol. As in the paper [12], it is assumed that a relay node could be selected through an appropriate policy in the cluster header.

In Step 3, the cluster header H_C transmits a REQ message to the selected relay node R2 to request one necessary sensor member. In Step 4, the relay node R2 immediately transmits the REQ message from the cluster header H_C to the other cluster header H_B. In Step 5, the cluster header H_B transmits a MOVE message to move the M3 to cluster zone C. In paper [15], the defect of the relay node is determined, and the recovery is performed. In Step 6, at the same time, the cluster header H_B predicts that its area will be a sensing hole and sends a REQ message to the relay node R1 to request one of the necessary sensors, and in Step 7, the relay node R1 also forwards its REQ message to the cluster header HA. In Step 8, the cluster header HA selects M2 from among the sensor node members within zone A and commands it to move to cluster zone B. The continuous movement of M3 and M2 is called "cascaded movement."

3 The Proposed Relocation Protocol

In Fig. 1, member node M3 moved to a neighboring cluster zone C. In Figs. 2 and 3, the proposed protocol is described while describing the following successive scenarios after the example of Fig. 1. In Step 1, the moved M3 sends a NEW message to the new cluster header H_C. H_C registers M3 as its member node and transmits a NEW-ACK message. On the other hand, it is assumed that nodes M4 and M5 were also newly migrated to cluster zone C. Fig. 4 shows the pseudo-code for the proposed algorithm that exchanges the roles between the cluster header and the member node in this paper. Here, the role according to the state of each node is described, and the parts (such as lines 38, 42) unrelated to the main contribution were omitted as much as possible.



(a) Header replacement pre-procedure **(b)** Header replacement

Figure 2: Header replacement procedure for the proposed protocol

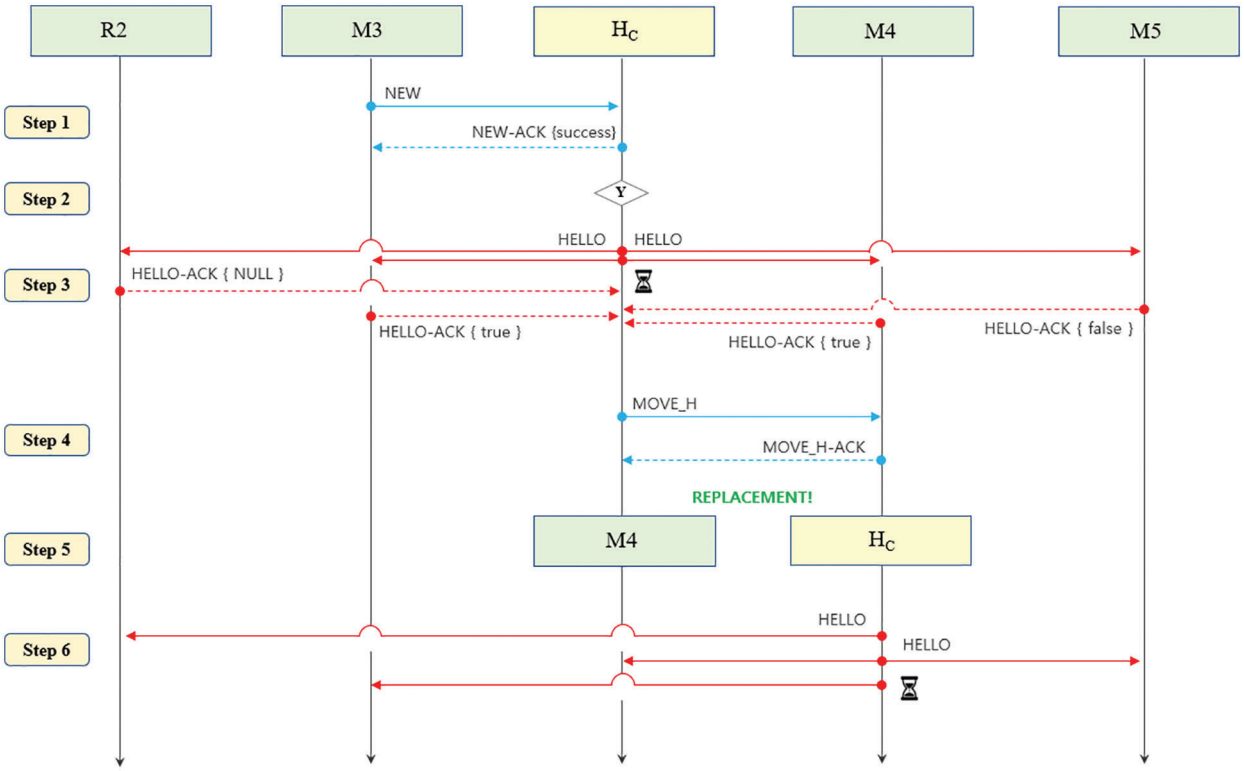


Figure 3: Message flows: header replacement procedure of the proposed relocation protocol

The battery's shape in Fig. 2a describes the level of jumping ability remaining of a hopping sensor node. In Step 2, H_C detects that 10% of its jumping capability remains with continuous jumps. (Line 08 in Fig. 4) In Step 3, H_C finds a member capable of sufficiently serving as a new header among the members in its cluster zone. The node capable of performing the role of the header could be found by checking the jumping capability of the node through the HELLO message. (Lines 14–15) Here, H_C tries to find nodes among

his members, with about 20% of his jumping ability remaining. (Line 31) According to the network policy, the 10% and 20% may change depending on the situation. A field to confirm jump capability is newly added in the HELLO message format. The formats of messages are shown in Tab. 1.

```
01  Node_State( Act )
02    ∀Act ∈ { cluster header, member node, relay node } and candidates_set ← ∅
03    while( 1 )
04    {
05      if ( Act == cluster header )
06      {
07              ◦ Check its jumping capability level
08              ◦ if ( the checked level < the predetermined threshold )
09                      ◦ Set the jumping capability field as 'F' in HELLO message
10              ◦ Broadcast HELLO and Set a hello message timer
11              ◦ do
12                {
13                      ◦ Receive HELLO-ACK or the other types of messages
14                      ◦ if ( the jumping capability field of HELLO-ACK == 'T' )
15                              ◦ candidates_set ← the member for the HELLO-ACK
16                } while( the timer is not expired )
17              ◦ if ( candidates set ≠ ∅ )
18                {
19                      ◦ Choose one member of candidates_set as the next cluster header
20                      ◦ Send MOVE_H to the member selected
21                      ◦ Act ← member node and candidates_set ← ∅
22                }
23      }
24      if ( Act == member node )
25      {
26              ◦ do  // wait for receiving messages
27                {
28                      ◦ if ( the received message == HELLO )
29                              ◦ if ( the jumping capability field == 'F' )
30                                      ◦ Check its jumping capability level
31                                      ◦ Set the field as 'T'/'F' in HELLO-ACK according to the policy
32                              ◦ Transmit HELLO-ACK to the cluster header
33                      ◦ if ( the received message == MOVE_H )
34                              ◦ Send MOVE_H-ACK to the cluster header
35                              ◦ Migrate to the close location of the cluster header
36                              ◦ Act ← cluster header
37                      ◦ if ( the received message == MOVE )
38                              ... ...
39                } while( Act ≠ cluster header )
40      }
41      if ( Act == relay node )
42              ... ...
43    } // End of while loop
```

Figure 4: The pseudo-code for header replacement

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

Type	Description
HELLO	• A cluster header checks the state of its zone by broadcasting periodically.
	• { type, source address, destination address (broadcasting), jumping capability (T/F) }
HELLO-ACK	• Member node sends whether it is a relay node or not (T/F) in response to the HELLO message.
	• { type, src address, dst address (header), relay node field (T/F), jumping capability (T/F/NULL) }
REQ	• A header requests a selected relay node for supply hopping sensors for sensing hole recovery.
	• { type, src address, dst address (relay), # of req. members, sensing hole header address, header GPS information }
MOVE	• A neighbor header of requesting header sends migration commands to the members chosen by multicasting.
	• { type, src address, dst address (hopping members), sensing hole header address, sensing hole header GPS information }
NEW	• A member moved to a neighboring cluster zone by the MOVE command sends a message to the new cluster header for new registration after the migration is completed.
	• { type, src address, dst address (sensing hole's cluster header) }
NEW-ACK	• The header recognizes the migration of the new node and responds that it has registered as a member of its cluster zone after receiving the NEW message.
	• { type, src address, dst address (hopping members) }
MOVE_H	• The header requests the selected member to come to its close location after choosing an appropriate member capable of performing a cluster header.
	• { type, src address, dst address }
MOVE_H-ACK	• While moving to the header, it is requested to perform the role of the cluster header. Upon receiving this message, the cluster header serves as a member node.
	• { type, src address, dst address }

Since relay node R2 has to perform the role of a relay node, the added field is checked with 'NULL.' Member node M5 checks with 'F' because its jumping ability is low. Assume that members M3 and M4 have the high jumping ability, so they check with 'T.' (Lines 29–31).

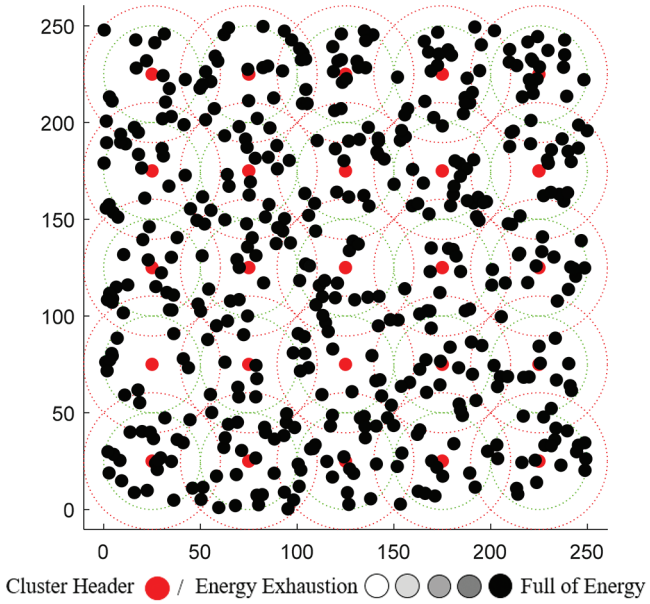
In Step 4, considering the location of M3 and M4, H_C transmits a MOVE_H message to the nearest M4 to move. (Lines 19–20) After that, M4 sends a MOVE_H-ACK message to the header H_C while moving (Lines 34–35) and exchanges roles with each other in Step 5. (Lines 21, 36) In this case, the old header also transmits basic information about the cluster to the new header. In Step 6, the new cluster header H_C sends a HELLO message to acquire new information. Replacing a member with such a high jump capability with a cluster header intuitively shows that it could be very efficient because it need not require re-electing the cluster header and re-performing clustering across the network area.

4 Performance Evaluation

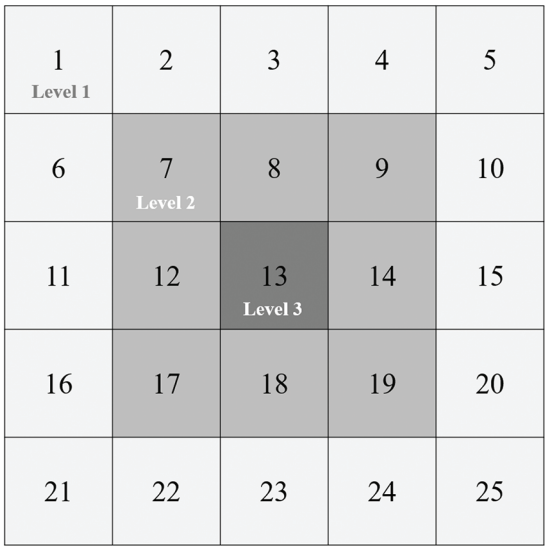
This section performs performance evaluations of the proposed relocation protocol. Tab. 2 describes the typical configuration settings used in the simulation. 500 hopping sensors are initially randomly sprayed in a given 250 m × 250 m area, as shown in Fig. 5a. The red, and black nodes are represented a cluster header, a relay node or node member, respectively. Fig. 5b shows the hole area as a grid. The level of data occurrence frequency is defined in levels 1 to 3. The cluster ID 13 with the darkest color is level 3, and data generation events occur the most. The outermost brightly colored grid is level 1, with minor data events occurring. We consider the exponential distribution of the mean $H_t/2^{Level}$; levels 1, 2, and 3 are averaged 15, 7.5, and 3.75 min, respectively. The total simulation period is considered 4 days.

Table 2: Simulation environments

Parameters	Values
Network area	250 m × 250 m
Number of total member sensor nodes distributed	500
Rate for number of minimum members for each cluster zone to properly gather data	θ: 0.3, 0.5, 0.6, 0.7
Maximum distance that a member moves forward with one hopping	1.5 m
Transmission radius for each member on the ground (a small circle of green dotted lines in Fig. 5)	25 m
Maximum transmission radius when highly jumping (a big circle of red dotted lines in Fig. 5)	35 m
HELLO message interval time (H_t)	30 min
Simulation time	4 days



(a) Hopping sensor nodes scattered initially

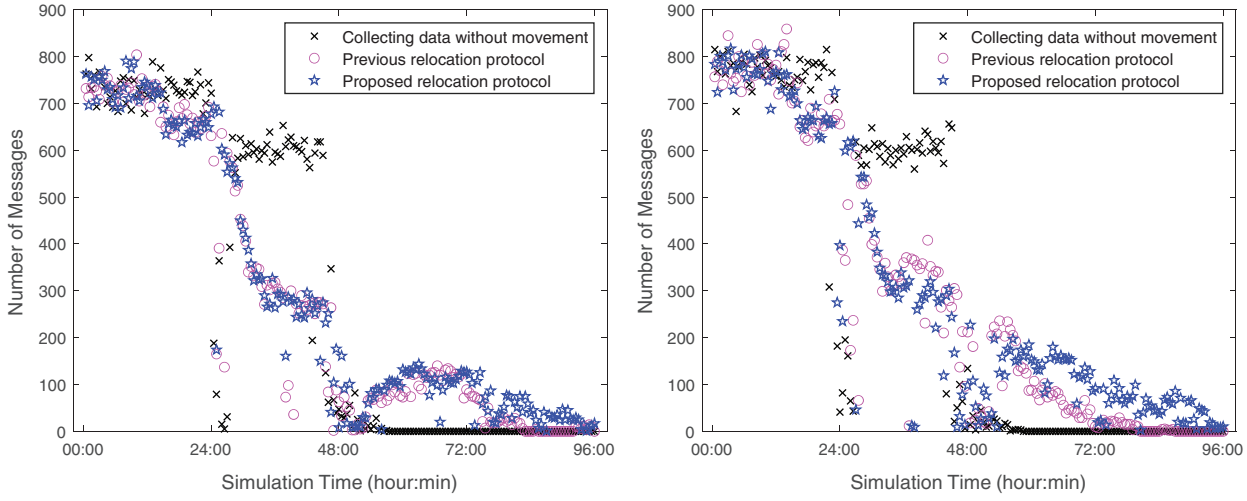


(b) Level of events generated

Figure 5: Network area

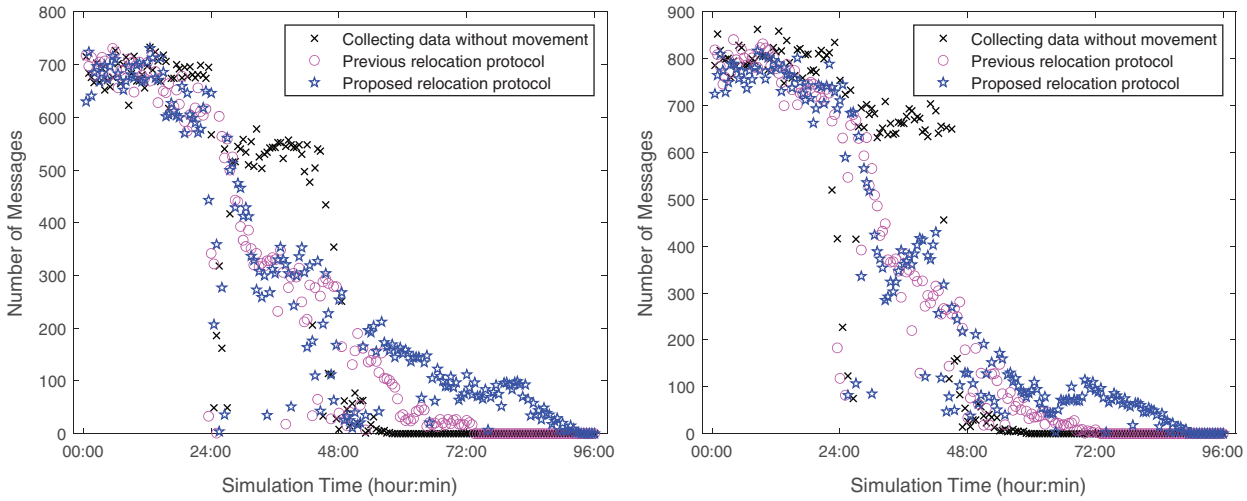
We assume that the transmission radius of every hopping sensor on the ground without jumping is 25 m, and the max-transmission radius of the hopping sensor, which is jumping to the maximum height, is about 35 m. It can move 1.5 m forward in one jump and can move up to 350 m, which means the ability to move a diagonal of the given area. The energy of each node is initially set to 100 units, and it consumes 0.25 units per event to obtain data generated. Simulation studies on energy models consumed in transmitting and receiving data are diverse [19]. However, in this simulation, relative comparisons are performed in the same environment, so it is assumed to be a "unit" without providing a particular unit. If the node's energy is 0, it cannot collect data for the event, and the hopping sensor cannot move if there is no jumping ability, so we consider node fault. Every cluster header broadcasts a HELLO message periodically every 30 min to predict a sensing hole. When 500 nodes are distributed initially over the whole area, about 20 nodes may be scattered on average in one grid.

The criterion of the sensing hole depends on the ratio of the variable θ. For instance, we assume that a case of fewer than 14 (i.e., $\lceil 20 \times 0.7 \rceil$) nodes is determined as a sensing hole when θ is 0.7. Fig. 6 shows that the simulation is performed for 4 days in the environment of Fig. 5a, and the number of occurrences of messages is counted at levels 2 and 3 of Fig. 5b, which has a high probability of occurrence of a sensing hole due to a high event of data generation rate. The x-marks are the number of messages occurring in the primary data gathering scheme in which every sensor cannot move and jump. The circles indicate the number of messages generated in the previous relocation protocol [15]. However, in the previous research, the condition and role of the cluster header were overlooked. In this simulation, thus, the initial values of the jumping ability and energy of each cluster header are considered. Both jumping ability and energy consumption of the cluster header are also considered in the proposed protocol of the star-mark. In addition, when the jumping ability of the header is exhausted, the closest member node is considered for header replacement. The criterion for determining that the header does not have the jumping ability is less than 10% of its jumping ability, and it is considered when the jumping ability of the candidate member node to be replaced exceeds 20%. The criteria values could be appropriately modified in some cases.



(a) $\theta = 0.3$ **(b) $\theta = 0.5$**

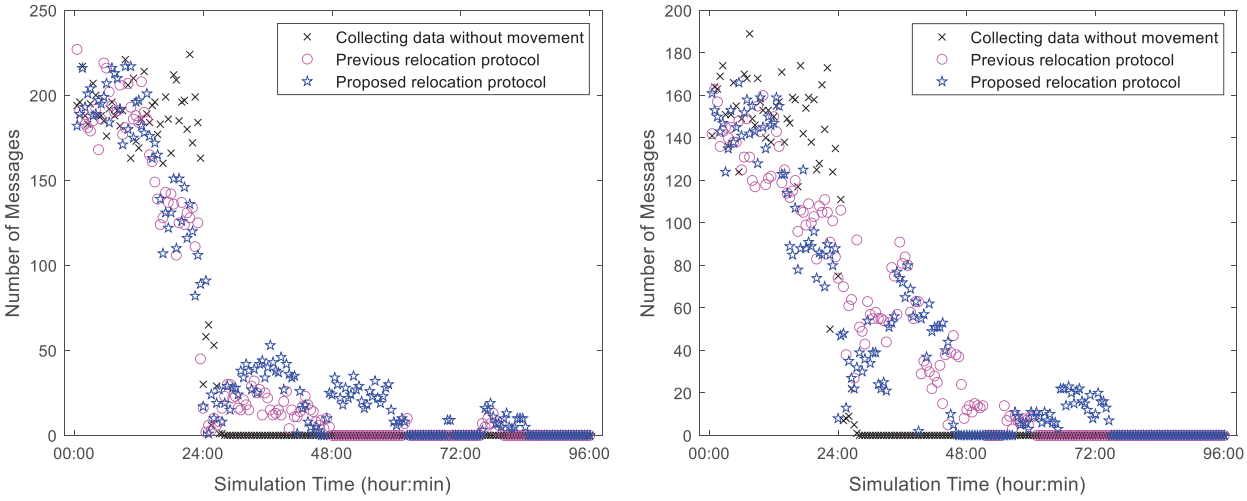
Figure 6: (Continued)



(c) $\theta = 0.6$ **(d) $\theta = 0.7$**

Figure 6: Number of messages generated for four days on levels 2 and 3

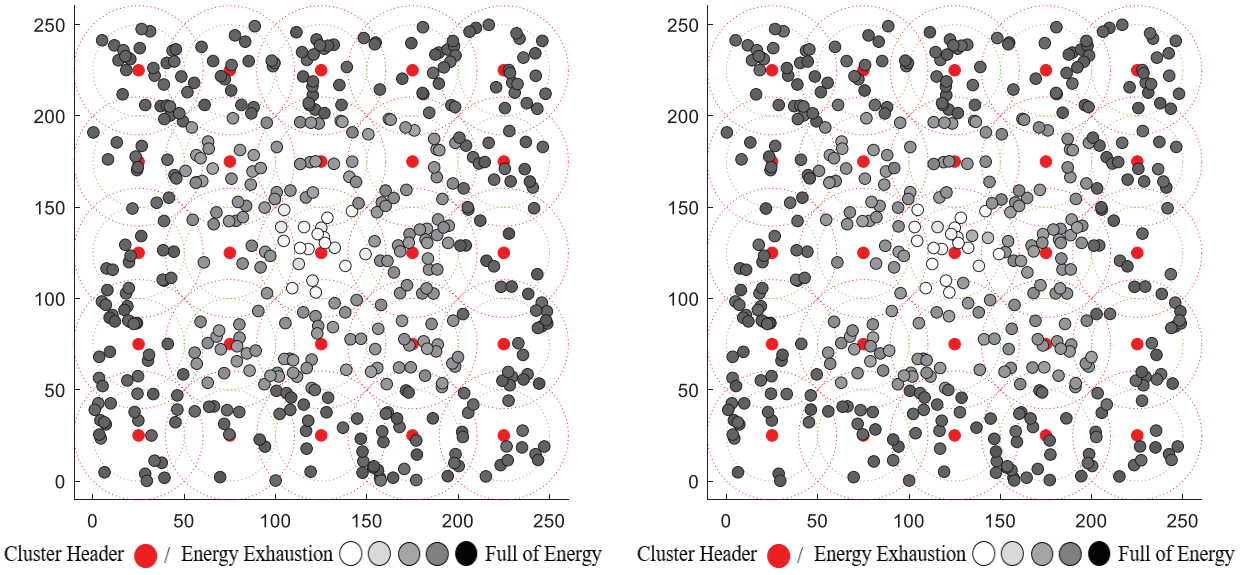
The candidate member node could be replaced by the role of the exhausted header after the member moves to the coordinates of the header. As shown in Fig. 6, as the θ value increases, the frequency of occurrence of the sensing hole could occur more frequently, and it can be seen that the header jump occurs more frequently. Therefore, it can be seen that data collection is not being performed because the sensing hole cannot be recovered due to a header fault. In the proposed protocol, it can be seen that the header is replaced frequently so that continuous data collection can be performed. Roughly, when θ value is 0.7, it can be seen through simulation that about 20 replacements of the head have been completed in 4 days. In addition, Fig. 7 shows the number of data collected at level 3 in Fig. 5b. It can be seen that the proposed method can collect data for a more extended period than the previous schemes, even though it is the cluster with the most sensing holes.



(a) $\theta = 0.3$ **(b) $\theta = 0.7$**

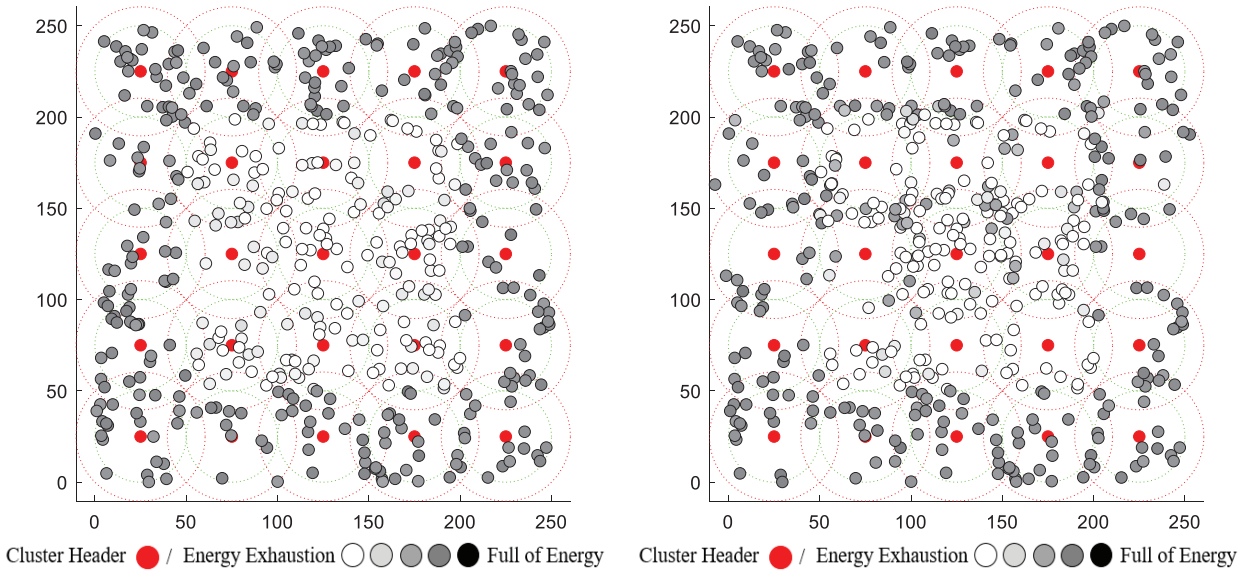
Figure 7: Number of messages generated for four days on level 3

Figs. 8 and 9 show the whole network states after 2 and 3 days in Fig. 6d, θ is 0.7, respectively. The black circles are nodes that are fully energy-charged. As the node consumes energy, it gradually turns black, gray, and white. In other words, white nodes indicate nodes where energy defects occur. In Fig. 7b, it was observed that the number of messages abruptly decreased in the scheme without mobility after one day; thus, it can be seen that most nodes of the center cluster (ID 13 of Fig. 5b) in Fig. 8a are defective. In order to overcome this, it can be seen that numerous hopping nodes have moved to the center of the area, as shown in Fig. 9b. Fig. 9b shows the network state that the number of hopping sensor nodes moved to the center area, where data generation events are significant for 3 days, is large. The sensing hole state occurred a lot in the middle, and many defects in the relay node also occurred, so it could be confirmed that the relay node has been replaced with a relay node with sufficient energy. When the jumping ability of the cluster header was exhausted and the role of the header was switched with a member node, the moved member node is set to be positioned one the old header in the computer simulation. The color of the old member node (a new cluster header) is also red.



(a) No movements (b) Proposed relocation protocol

Figure 8: Network states after two days in Fig. 4d



(a) No movements (b) Proposed relocation protocol

Figure 9: Network states after three days in Fig. 4d

5 Conclusion

Although sensor nodes are scattered for data collection, the occurrence of a sensing hole that cannot collect an appropriate amount of data becomes a significant barrier in researching the area. Therefore, it is self-evident that relocation using mobile sensors is the most desirable method. In particular, a hopping sensor is suitable for a rough or natural disaster area among mobile sensors. So far, we have proposed the most advanced relocation protocols based on a realistic distributed network environment. These protocols are a very realistic mechanism that eliminates the need to share all information between sprayed sensors.

In the previous work, the cluster header has a crucial role in detecting a sensing hole and requesting sensor nodes to recover the occurred sensing hole to neighboring clusters. However, the role and limitations of the cluster header in the relocation protocol were overlooked. The cluster header has a relatively high jumping ability and energy consumption compared to other nodes due to frequent jumps. Therefore, it is more likely to be a fault, leading to data loss in the network. In this paper, the jumping ability and energy consumption of the cluster header are considered in reality. A function to replace the cluster header is added to prepare for a fault situation. Simulation results show that the data collection time could be further increased. In the future, we will further study the relocation algorithm that can respond appropriately in regions such as deserts and polar regions where the environment is dynamically changing.

Funding Statement: This work was supported by Incheon National University Research Grant in 2020 (2020–0437).

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

References

[1] W. Lee, M. Kim and J. Park, "Speed-up of the matrix computation on the ridge regression," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 10, pp. 3482–3497, 2021.

[2] V. -V. Vo, D. -T. Le, M. Kim and H. Choo, "Data aggregation latency minimization in multichannel duty-cycled WSNs," in *Proc. IEEE ICOIN*, Jeju Island, Korea, pp. 345–348, 2022.

[3] T. -B. Dang, D. -T. Le, T. -D. Nguyen, M. Kim and H. Choo, "Monotone split and conquer for anomaly detection in IoT sensory data," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15468–15485, 2021.

[4] T. -D. Nguyen, D. -T. Le, V. V. Vo, M. Kim and H. Choo, "Fast sensory data aggregation in IoT networks: Collision-resistant dynamic approach," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15468–15485, 2021.

[5] F. Cintr'on, "Network issues for 3d wireless sensors networks," Ph.D. dissertation, Michigan State University, MI, USA, 2013.

[6] M. S. Kim, "Design of a transmission process for hopping sensors to enhance coverage," M.S. thesis, Sungkyunkwan University, Suwon, Republic of Korea, 2012.

[7] Z. Cen and M. W. Mutka, "Relocation of hopping sensors," in *Proc. IEEE ICRA*, Pasadena, CA, USA, pp. 569–574, 2008.

[8] M. Kim, M. W. Mutka and H. Choo, "On relocation of hopping sensors for rugged terrains," in *Proc. ICCSA*, Fukuoka, Japan, pp. 203–210, 2010.

[9] M. Kim and M. W. Mutka, "On relocation of hopping sensors for balanced migration distribution of sensors," in *Proc. ICCSA*, Seoul, Korea, pp. 361–371, 2009.

[10] M. Kim and M. W. Mutka, "Multipath-based relocation schemes considering balanced assignment for hopping sensors," in *Proc. IEEE/RSJ IROS*, St. Louis, MO, USA, pp. 5095–5100, 2009.

[11] M. Kim, S. Park and W. Lee, "Energy and distance-aware hopping sensor relocation for wireless sensor networks," *Sensors*, vol. 19, no. 7, pp. 1567, 2019.

[12] 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, pp. 5654, 2020.

[13] S. Park, M. Kim and W. Lee, "Uniform sensor-node request scheme for the recovery of sensing holes on IoT network," *Journal of Korea Society of Digital Industry and Information Management*, vol. 16, no. 4, pp. 9–17, 2020.

[14] M. Kim and W. Lee, "Adaptive success rate-based sensor relocation for IoT applications," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 9, pp. 3120–3137, 2021.

[15] M. Kim and W. Lee, "Novel sensing hole recovery with expanded relay node capability," *Computer Systems Science and Engineering*, vol. 44, no. 1, pp. 663–675, 2022.

[16] S. Park, M. Kim and W. Lee, "Energy-efficient wireless hopping sensor relocation based on prediction of terrain conditions," *Electronics*, vol. 9, no. 1, pp. 49, 2020.

[17] S. Park, M. Kim and W. Lee, "Success rate queue-based relocation algorithm of sensory network to overcome non-uniformly distributed obstacles," *Computers, Materials & Continua*, vol. 65, no. 2, pp. 1181–1201, 2020.

[18] A. S. Rostami, M. Badkoobe, F. Mohanna, H. Keshavarz, A. A. R. Hosseinabadi *et al.,* "Survey on clustering in heterogeneous and homogeneous wireless sensor networks," *The Journal of Supercomputing*, vol. 74, pp. 277–323, 2018.

[19] I. Shin, M. Kim, M. W. Mutka, H. Choo and T. -J. Lee, "MCBT: Multi-hop cluster based stable backbone trees for data collection and dissemination in WSNs," *Sensors*, vol. 9, no. 8, pp. 6028–6045, 2009.