디지털산업정보학회 논문지 제16권 제4호-2020년 12월
http://dx.doi.org/10.17662/ksdim.2020.16.4.009

IoT 네트워크의 센싱홀 복구를 위한 센서 이동 균등 요청 방법

김 문 성*·박 수 연**·이 우 찬***

Uniform Sensor-node Request Scheme for the Recovery of Sensing Holes on IoT Network

Kim Moonseong·Park Sooyeon·Lee Woochan

〈Abstract〉

When IoT sensor nodes are deployed in areas where data collection is challenging, sensors must be relocated if sensing holes occur due to improper placement of sensors or energy depletion, and data collection is impossible. The sensing hole's cluster header transmits a request message for sensor relocation to an adjacent cluster header through a specific relay node. However, since a specific relay node is frequently used, a member sensor located in a specific cluster area adjacent to the sensing hole can continuously receive the movement message. In this paper, we propose a method that avoids the situation in which the sensing hole cluster header monopolizes a specific relay node and allows the cluster header to use multiple relay nodes fairly. Unlike the existing method in which the relay node immediately responds to the request of the header, the method proposed in this paper solves a ping-pong problem and a problem that the request message is concentrated on a specific relay node by applying a method of responding to the request of the header using a timer. OMNeT++ simulator was used to analyze the performance of the proposed method.

Key Words : Mobile IoT, Hopping Sensor, Sensing Hole Recovery, Simulation

Ⅰ. 서론

IoT 기기의 발달로 다양한 데이터의 수집이 용이해졌다. 예를 들면, 사람의 접근이 불가능한 지역에서 데이터 수집을 위해서는, 드론과 같은 무인 비행 장치에 소형 IoT 기기를 장착(<그림 1> 참고)하여 IoT 센싱 기기들을 배포할 수 있다. 그러나 드론에서의 살포를 통하여 소형 IoT 기기들을 고르게 배치하는 것은 쉽지 않다. 이로 인하여 정확한 데이터 수집이 어렵고, 또한 부정확한 데이터의 지속적인 수집으로 소형 기기의 에너지가 고갈되어 예기치 않는 기기 결함이 발생할 수 있다[1]. 최악의 경우, 전체 네트워크의 통신이 단절 되어 데이터 수집이 불가능해 질 수 있다. 이렇듯 더 이상 데이터 수집이 어려운 일부 지역을 '센싱홀(sensing hole)'이라고 한다[2].

* 서울신학대학교 교양학부 조교수(제1저자)
** 인천대학교 전기공학과 박사후 연구원(참여저자)
*** 인천대학교 전기공학과 조교수(교신저자)

최근 IoT 센싱 기기의 에너지 소비를 고려한 다양한 방법이 제안되고 있다. 예를 들어, 수많은 센서가 배치된 지역에서 일부 영역의 센서 밀도를 고려한 데이터들의 수신, 센서의 Sleep 간격 예측을 위해 배터리 잔량 및 배터리의 이전 사용 이력, 그리고 애플리케이션에 필요한 정보 유형을 이용하는 방법 등이 제안되었다[3, 4]. 그러나 이것 역시 센서의 근본적인 에너지 고갈 문제를 극복할 수는 없다. 가장 이상적인 방법은 센싱홀이 발생하면 기기를 센싱홀로 직접 이동시켜 데이터 수집을 가능하게 복구하는 재배치 방법이다. 일반적으로 모바일 센서의 이동에 관한 초기 연구는 바퀴를 이용한 방식이었다. 그러나, 바퀴는 장애물이 많은 거친 지역에서의 이동이 자유롭지 못하다는 한계점이 있다.

바퀴 기반 이동의 단점을 극복하기 위해 센서가 점프하여 원하는 방향으로 이동하는 홉핑 센서가 제안되었다. 초기 호핑 센서 재배치 방법은, 이동 경로의 설정을 위해 모든 센서가 네트워크 전체 영역에 대한 정보(모든 센서의 현재 위치 등)를 알고 있다고 가정했으나, 이것은 현실적이지 않다[5, 6]. 이러한 문제를 현실적으로 해결하기 위해 우리 연구팀은 분산 환경을 기반으로 한 홉핑 센서 재배치 프로토콜을 제안했다[7]. 그러나 모바일 센서의 이동 패턴(특정 경로로만 이동하는 패턴)에 대한 한계점 및 이로 인한 메시지 핑퐁 현상(데이터를 지속적으로 서로 주고 받는 현상)이 발견되었다. 따라서, 본 논문에서는 이러한 한계점을 극복한 IoT 기기의 재배치 프로토콜을 제안한다. 또한 OMNeT++를 사용하여 제안 된 재배치 프로토콜을 시뮬레이션한다[8].

본 논문은 다음과 같이 구성된다. 2장은 관련 연구, 3장은 새로운 이동 센서 재배치 방법을 기술한다. 4장은 시뮬레이션 및 성능 평가를 기술하고, 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

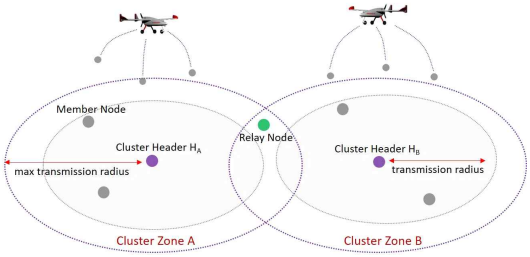
IoT 홉핑 센서 기기(노드)는 점프 수단으로 이동하므로 바위나 모래와 같은 장애물이 있는 지역에서의 이동이 매우 용이하다. 또한, 점프하는 동시에 데이터를 전송할 수 있으므로 데이터 전송 반경의 조정이 가능하다. 예를 들어, 노드가 지상에서 1m 점프하면 지상 통신 반경과 비교해 약 6배 증가 시킬 수 있다고 연구되었으며[9], 논문 [10]에서는 실제 발사체를 만들어 홉핑센서의 점프 높이에 따라 변하는 통신 반경에 대한 직접적인 측정을 수행하였다.

최근 십여 년간 다양한 홉핑 센서 재배치 알고리즘이 제안되고 있다. 논문 [5]에서는 센싱홀과 최단 경로상의 클러스터 존에 있는 홉핑 센서 노드를 재배치하여 센싱홀을 복구하는 방법이 최초로 제안되었다. 그러나 최단 경로상의 특정 노드들을 거쳐 이웃한 클러스터 영역의 헤더에 재배치를 위한 요청 메시지를 반복적으로 전송하므로 또 다른 센싱홀이 쉽게 발생할 수 있다. 또한 이들 클러스터 영역에 있는 일부 센서는 반복적인 이동으로 인해 이동 기능을 상실하는 문제가 발생할 수 있다.

그러나 최근의 다양한 연구들은 모든 클러스터 헤더 및 노드들이 현재 전체 네트워크의 모든 정보를 이용하여 실시간으로 재배치를 위한 경로들(최단경로[5], 다중경로[11], 균등이동[12], 재배치후 ID배정[13] 등)을 설정한다. 네트워크 영역이 매우 작다 할지라도 모든 클러스터 헤더가 정보를 교환하고 경로를 설정하는 것은 실제로 불가능하며 매우 많은 제어 메시지가 송수신된다. 최근 우리는 논문 [7]에서 이러한 문제를 해결했다. 모든 센서 노드는 주변 센서 노드들 및 전체 네트워크의 정보를 전혀 알 필요가 없다. 센싱홀의 헤더가 재배치를 위한 센서 노드를 근처 클러스터 헤더에 요청하여 센싱홀을 복구하는 분산 네트워킹 기반 재배치 프로토콜이다. 본 재배치 프로토콜을 활용하여 논문 [14]에서는 재배치 성공률 기반으

로 장애물 수준을 예측하였으며, 논문 [15]에서는 실시간으로 변할 수 있는 장애물 환경에 적용 가능한 재배치 프로토콜을 제안하였다.

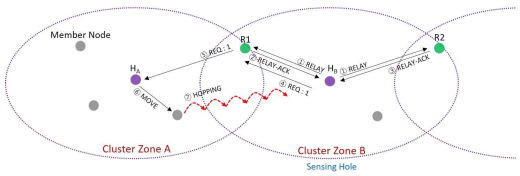
IoT 홉핑 센서는 특정 영역에 무작위로 배치되어 관심 있는 데이터를 수집한다. <그림 1>과 같이 사람이 접근할 수 없는 지역은 무인 비행체, 드론 등을 이용해 IoT 기기를 배치할 수 있다. 각 클러스터 영역의 중앙에 있는 센서를 클러스터 헤더라 가정하자. 각 헤더의 통신 반경 안에 있는 센서들은 클러스터 헤더의 멤버 노드가 된다.



<그림 1> IoT 홉핑 센서 네트워크

본 논문에서는 기존의 다양한 알고리즘들을 사용하여 네트워크 영역의 클러스터링(클러스터 존 분할) 및 헤더 선출이 적절하게 수행된다 가정한다[16, 17]. 홉핑 센서는 이웃 센서와 통신하기 위해 점프할 수 있고, 이들 센서는 GPS를 사용하여 자신의 위치를 알 수 있다. 클러스터 헤더가 점프하여 통신 가능한 영역이 클러스터 영역이므로, 일반적으로 헤더 간에는 통신이 불가능 하다. 따라서, 데이터를 전달해 줄 수 있는 릴레이 노드로 이웃한 클러스터 헤더들 간의 통신을 가능하게 할 수 있다.

어떤 클러스터 존에서 데이터 수집을 위해 필요한 센서 노드 수가 부족하여 센싱홀이 발생하면, 그 헤더는 인접 클러스터 존에게 센싱홀 복구를 위해 필요한 멤버 노드들의 재배치를 요청한다. <그림 2>는 가장 대표적인 재배치 프로토콜[7]의 간단한 예제이며,



<그림 2> 홉핑센서 재배치 [7]

메시지 타입은 <표 1>과 같다. 향후 본 논문에서 제안하는 방법 또한 하기 기술하는 메시지 타입을 사용할 것이다.

<표 1> 메시지 타입 및 설명

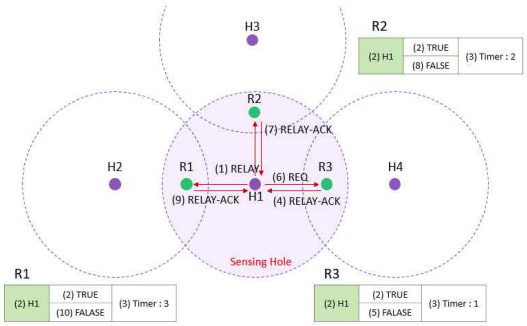
타입	포맷 및 설명
HELLO	• 헤더가 주기적인 브로드캐스팅으로 자신의 존의 (센싱홀)상태를 확인 • { 타입, 소스주소, 목적주소(브로드캐스팅) }
RELAY	• 헤더가 적절한 릴레이 노드의 선택을 위해 릴레이 노드들에게 멀티캐스팅을 수행 • { 타입, 소스주소, 목적주소(릴레이 노드들) }
RELAY-ACK	• 릴레이 노드가 RELAY 수신후 답변 • { 타입, 소스주소, 목적주소(헤더) }
REQ	• 헤더가 릴레이 노드에게 센싱홀 복구를 위해 홉핑센서 이동을 요청 • { 타입, 소스주소, 목적주소(릴레이), 요청멤버수, 센싱홀 헤더주소, 헤더GPS정보 }
MOVE	• 헤더가 선택된 멤버들에게 멀티캐스팅으로 이동 명령을 전달 • { 타입, 소스주소, 목적주소(홉핑멤버들), 센싱홀 헤더주소, 센싱홀헤더GPS정보 }

클러스터 존 B의 헤더 H_B가 센싱홀을 감지하면 H_B는 릴레이 노드 R1과 R2에 RELAY 메시지를 브로드캐스트한다(①). R1과 R2는 이에 대한 답신으로 각각 RELAY-ACK 메시지를 H_B에게 전송한다(②,③). H_B가 R1으로부터의 RELAY- ACK 메시지를 제일 먼저 수신하면, R1에게 필요한 센서 멤버 노드 수를 포함하는 REQ 메시지를 전송한다(④). R1은 수신된 REQ 메시지를 클러스터 존 A의 헤더 H_A로 즉시 전달한다(⑤). H_A는 REQ 메시지에 기록된 요청 노드 수만큼

자신의 멤버 노드들을 선택하고 센싱홀로 이동시키기 위한 MOVE 메시지를 전송한다(⑥). MOVE 메시지를 수신한 센서 멤버 노드는 클러스터 존 B로 이동하고(⑦), 센싱홀은 복구 된다.

<그림 2>에서 클러스터 존 B에 센싱홀이 발생할 때마다 헤더 H_B는 RELAY 메시지를 브로드캐스트한다. 그러나, 릴레이 노드들의 응답 메시지 중 릴레이 노드 R1의 RELAY-ACK 메시지가 항상 먼저 도착한다 가정하자. 본 재배치 프로토콜은 최단 거리 기반 방식(즉, 가장 빠른 응답 메시지를 고려함)이므로, 클러스터 헤더 H_B는 클러스터 존 A에 필요한 센서를 지속적으로 요청하게 되며, 멤버 센서의 이동은 특정한 존(여기서는 클러스터 존 A)에서만 발생하게 된다. 또한 이로 인하여 클러스터 존 A에서도 센싱홀이 발생하게 된다면, 이웃 존 또한 센싱홀임을 인지 못하는 헤더 H_B의 모든 요청이 계속 실패하게 되고, 헤더 H_A 또한 H_B에게 이동 멤서 센서들을 요청하면 서로 요청 메시지를 연속적으로 주고 받는 핑퐁 현상이 발생할 가능성이 존재하게 된다.

Ⅲ. 제안하는 방법



<그림 3> 릴레이 노드 선정 과정 (센싱홀이 1개일 경우)

<그림 3>은 4개의 클러스터 존이 있고, 각 존의 헤더를 H1, H2, H3, H4이라 하고, 헤더 H1의 클러스터 존에서 센싱홀이 발생하였다 가정하자. 센싱홀 복구를 위하여 헤더 H1은 자신의 릴레이 노드들에게 RELAY 메시지를 전송한다. 그리고 모든 릴레이 노드는 RELAY를 전송한 헤더 정보들을 관리하기 위하여 테이블을 활용한다고 가정하자. 상기 테이블은 헤더 주소, bool 형식의 값을 저장하기 위한 필드를 가지고 있다. 헤더 주소 필드는 RELAY 메시지를 전송한 헤더의 주소를 저장한다. RELAY 메시지를 전송한 헤더의 bool 형식의 값을 TRUE로 설정하고, 난수값으로 타이머를 실행한다. 상기 타이머가 종료하면, 헤더에게 RELAY-ACK 메시지를 전송하고, bool 형식의 값은 FALSE가 된다.

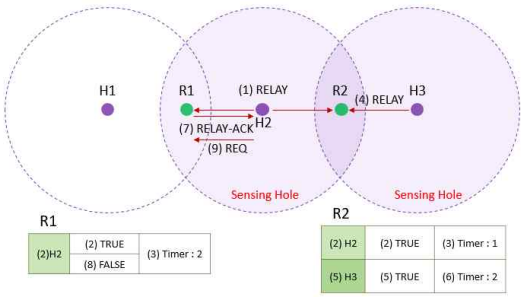
다음은 H1이 헤더인 클러스터 존에서 센싱홀이 발생했을 때 각 노드들의 동작을 단계별로 기술하였다.

1. 헤더 H1은 자신의 릴레이 노드인 R1, R2, R3에게 RELAY 메시지를 브로드캐스팅한다(1).
2. RELAY 메시지를 수신한 각 릴레이 노드 R1, R2, R3는 자신의 테이블에 RELAY 메시지를 전송한 헤더의 주소를 저장하고, bool 형식의 값을 TRUE로 설정한다(2). 이는 RELAY 메시지에 대한 RELAY-ACK 메시지를 답신하는 단계를 표현하기 위해서이다.
3. 각 릴레이 노드 R1, R2, R3는 랜덤수를 이용하여 타이머를 실행한다. 여기서, R1은 3초, R2는 2초, R3은 1초로 타이머가 각각 실행되었다고 가정하자(3).
4. 릴레이 노드 R3의 타이머가 가장 먼저 종료되면, 릴레이 노드 R3는 헤더 H1에게 RELAY-ACK 메시지를 즉시 전송하고(4), 테이블에서 헤더 H1의 bool 형식의 값을 FALSE로 변경한다(5). (여기에서, 테이블의 bool 값 필드가 두 행으로 표현되었으나, 사실 T/F를 표현하는 하나의 필드이다. 예제를 설명하며 독자의 이해를 돕기 위해, 변화하

는 모습을 단계별로 기재할 목적으로 두 개의 행으로 괄호안의 번호와 함께 표현하였다.) 릴레이 노드 R3로부터 RELAY-ACK 메시지를 수신한 헤더 H1은 릴레이 노드 R3에게 필요한 센서 멤버 노드들을 요청하는 REQ 메시지를 전송한다(6).

5. 릴레이 노드 R2의 타이머가 종료하면, 릴레이 노드 R2는 헤더 H1에게 RELAY-ACK 메시지를 전송하고(7), 테이블에서 헤더 H1의 bool 값을 FALSE로 갱신한다(8). 그러나, 헤더 H1은 릴레이 노드 R2가 전송한 RELAY-ACK 메시지는 무시한다.
6. 릴레이 노드 R1 또한, R2와 유사한 과정이 수행되며(9, 10), 헤더 H1은 R1이 송신한 RELAY-ACK 메시지를 무시한다.

센싱홀이 발생할 때마다 단계1에서 단계6의 과정이 실행 된다. 반복 실행될 때마다, 단계3에서 타이머로 설정되는 값이 랜덤수로 설정된다. 이로 인해 릴레이 노드 R1, R2, R3가 비교적 균등하게 센싱홀 헤더(H1)의 REQ 메시지를 수신하게 된다.



<그림 4> 릴레이 노드 선정 과정 (센싱홀이 복수일 경우)

<그림4>는 3개의 클러스터 존에서 헤더 H2와 H3의 클러스터 존에서 센싱홀이 발생할 경우, 릴레이 노드 선정 과정을 단계별로 기재하였다.

1. 가장 먼저, 센싱홀 헤더 H2가 릴레이 노드 R1과 R2에게 RELAY 메시지를 브로드캐스팅한다(1).
2. 릴레이 노드 R1과 R2는 테이블에서 헤더 H2의 주소를 등록하고, bool 값을 TRUE로 설정한다(2).
3. 릴레이 노드 R1과 R2는 난수를 이용하여 타이머를 설정하며, 릴레이 노드 R1은 2로, R2는 1로 타이머를 실행한다(3).
4. 잠시 후(타이머 종료 전), 헤더 H3에서 센싱홀이 발생하여 헤더 H3가 릴레이 노드 R2에게 RELAY 메시지를 전송한다(4).
5. RELAY 메시지를 수신한 릴레이 노드 R2는 테이블에 헤더 H3의 주소를 등록하고, bool값을 TRUE로 설정한다(5). 그리고 난수를 이용하여 타이머 2를 실행한다(6).
6. 단계3에서 실행한 릴레이 R2의 타이머가 가장 먼저 종료되고, 릴레이 노드 R2는 자신의 테이블에 있는 두 개의 헤더 H2와 H3의 bool 값이 모두 TRUE이므로 RELAY-ACK를 전송하지 않는다. TRUE라는 것은 헤더 H2와 H3의 클러스터 영역이 센싱홀이라는 것을 의미하기 때문이다.
7. 단계3에서 실행한 릴레이 노드 R1의 타이머가 종료된다면, 릴레이 노드 R1은 헤더 H2에게 RELAY- ACK를 전송하고(7), 테이블에서 H2의 bool 값을 FALSE로 갱신한다(8).
8. 릴레이 노드 R1으로부터 RELAY-ACK를 수신한 헤더 H2는 R1에게 REQ 메시지를 전송한다(9).

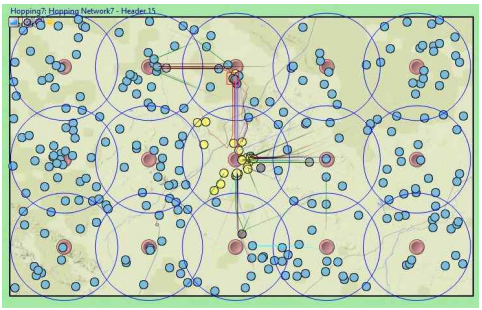
릴레이 노드는 자신의 테이블에서 각 헤더의 bool 값이 TRUE일 경우, 그 헤더가 있는 클러스터 영역에 센싱홀이 발생했음을 알 수 있다. 그러므로, RELAY-ACK를 전송하기 전에 테이블에 있는 모든 헤더의 bool 값이 TRUE이면, RELAY-ACK를 전송하지 않는다. 이처럼 RELAY-ACK 전송을 보류하면서 다른 릴레이 노드가 RELAY-ACK를 전송할 기회를 주어 발생 가능한 핑퐁 현상을 방지할 수 있다.

Ⅳ. 성능평가 및 분석

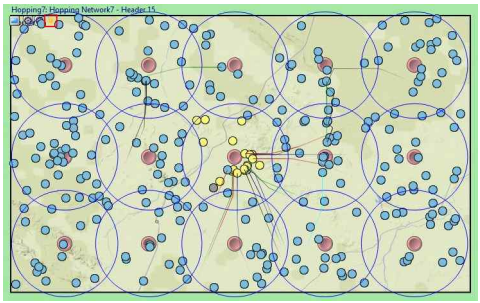
본 논문에서 제안한 제배치 프로토콜을 OMNeT++ [8, 18]를 사용하여 시뮬레이션하였으며, <표 2>는 시뮬레이션 환경이다.

<표 2> 시뮬레이션 환경

변수	값
네트워크 크기	250m × 250m
클러스터 헤더 수	15개
홉핑 센서 멤버 노드수	285개
클러스터 존 최소 멤버수	10개
최대 통신 범위	20m
점프시 최대 통신 범위	29m
점프시 이동 거리	2m
이동모델	MovingMobilitiyBase
시뮬레이션 시간	3일
타이머 설정	최대 3초



(a) 이전 재배치 프로토콜[7]에서 홉핑 센서들의 이동 모습

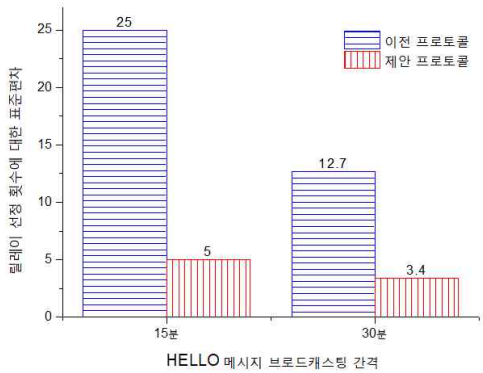


(b) 제안하는 재배치 프로토콜에서 홉핑 센서들의 이동 모습

<그림 5> 3일간의 시뮬레이션 결과 스냅샷

<그림 5>는 네트워크 영역에 15개의 클러스터를 기준으로 홉핑 센서를 임의로 배치 후 시뮬레이션을 실행한 모습이다. 네트워크의 가운데 있는 클러스터 존을 센싱홀로 가정을 하였다. 지속적인 센싱홀의 발생을 위하여 가운데 클러스터 존의 센서 멤버 노드들에게 지수분포(평균 5분)를 따르며 데이터를 수집하는 이벤트를 발생시켜 에너지를 소비(파란색→회색→노란색)하게 하였다. 에너지 부족으로 노드들이 결함(노란색)을 일으키며, 헤더(빨간색)는 주기적인 HELLO 메시지로 센서 멤버 노드들의 개수를 파악하며 센싱홀 임을 인지할 수 있다.

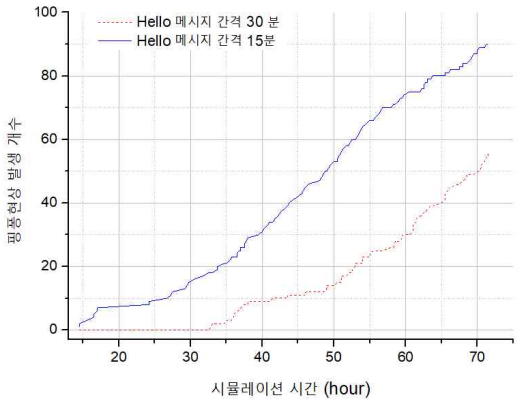
<그림 5(a)>를 살펴보면, 이전 재배치 프로토콜에서 특정 릴레이 노드를 이용하여 부족한 멤버 노드를 요청한다는 것을 알 수 있다. 홉핑 센서의 이동 경로는 실선으로 표현되었다. 반면에 <그림 5(b)>에서와 같이 제안하는 방법은 공평하게 릴레이 노드를 이용하여, 센서 멤버 노드들이 고르게 이동한 것을 살펴볼 수 있다.



<그림 6> 센싱홀 헤더가 릴레이 노드들을 선정한 횟수들의 표준 편차

<그림 6>은 센싱홀 클러스터 헤더의 릴레이 노드들이 수신한 REQ 메시지 개수에 대한 표준 편차이다. 여기에서, 클러스터 헤더의 센싱홀 감지를 위한 주기적인 HELLO 메시지 전송 간격은 15분과 30분으

로 가정하였다. 이전 재배치 방법에서의 표준 편차는 HELLO 메시지가 15분 간격에서 약 25개이고, 30분 간격에서 약 12개이다. 즉, 클러스터 헤더가 릴레이 노드들을 선정할 경우 특정한 릴레이 노드들에게 편중하였다는 것을 알 수 있다. 그러나, 제안한 방법에서는 REQ 개수의 표준 편차가 5.0, 3.4로 비슷하였으며. 이는, 센싱홀의 클러스터 헤더가 자신의 릴레이 노드들에게 공평하게 REQ 메시지를 전송하여 필요한 홉핑 센서 멤버 노드들을 요청한다는 것을 알 수 있다. 즉 <그림 5(b)>와 같이, 센서 노드들의 이동이 모든 방향에서 공평하게 수행된 것을 알 수 있다.



<그림 7> 네트워크에서 발생한 핑퐁 생성 시간(이전연구[7])

<그림 7>은 이전 재배치 방법에서 시뮬레이션 시간 동안 전체 네트워크에서 발생하는 핑퐁 현상의 개수를 보여주는 그래프이다. HELLO 메시지 간격이 짧으면 센싱홀을 감지할 가능성이 커지므로 REQ 메시지의 요청 횟수가 증가하고, 이로 인하여 핑퐁 현상이 나타날 가능성이 높다는 것은 자명하다. 본 그래프에서 알 수 있는 것은 이전 연구에서 <그림 5(a)>와 같이 특정한 방향에서의 센서 이동이 높았으며, 이로 인하여 핑퐁 현상 또한 꾸준하게 증가할 수 밖에 없다는 것을 알 수 있다. 그러나, 본 논문에서 제안하는 재배치 방법에서는 핑퐁 현상이 발생하지 않았으며 이로 인하여 이전 연구와는 다르게 매우 효율적인 홉핑 센서 재배치가 수행된다는 것을 알 수 있다.

Ⅴ. 결론

IoT 센서 기기의 부적절한 배치 또는 에너지 고갈로인해 데이터 수집이 불가능한 센싱홀 상태가 발생하면 이동 센서를 재배치하는 것이 가장 바람직하다. 지금까지 가장 현실적인 분산환경 기반의 재배치 프로토콜에서는, 센싱홀의 클러스터 헤더가 특정 릴레이 노드를 통해 인접한 클러스터 헤더에게 센서 재배치를 위한 요청 메시지를 전송한다. 그러나 특정 릴레이 노드를 빈번하게 이용하게 되어 센싱홀에 인접한 특정 클러스터 영역에 있는 홉핑 센서 멤버 노드들이 연속적으로 이동하는 한계점이 발생한다.

본 논문에서는 센싱홀의 클러스터 헤더가 특정 릴레이 노드를 독점하는 상황을 회피하고, 클러스터 헤더가 공평하게 여러 릴레이 노드를 이용할 수 있는 재배치 프로토콜을 제안하였다. 릴레이 노드가 헤더의 요청에 즉각적으로 응답하던 최단경로 기반의 기존 방법과는 다르게, 각각의 릴레이 노드들이 타이머를 사용하여 헤더의 요청에 응답하는 방식을 적용하여, 특정 릴레이 노드에게 요청 메시지가 집중되는 문제 및 핑퐁 현상을 해결하였다.

제안한 방법의 성능 분석을 위해 OMNeT++ 시뮬레이터를 사용하여 잘 알려진 최근 연구와 제안 방법에 대해 시뮬레이션을 진행하였다. 예상한 바대로, 제안한 프로토콜에서는 릴레이 노드들이 서로 공평하게 자신의 역할을 수행하여 전체적인 홉핑 센서 노드들의 이동 모습이 비교적 균등하게 이동한 것을 확인할 수 있었다. 또한 핑퐁 현상의 발생을 살펴볼 수 없었으며, 이와는 다르게 이전 재배치 방법에서는 한쪽으로 편중된 이동 모습과, 지속적으로 발생하는 메시지 핑퐁 현상을 살펴볼 수 있었다.

Acknowledgement

이 논문은 2020년도 서울신학대학교 교내연구비 지원에 의한 연구임. 또한, 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1G1A1007832).

참고문헌

[1] 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, 2018, pp.5744-5764.

[2] R. Kosar, E. Onur and C. Ersoy, “Redeployment Based Sensing Hole Mitigation in Wireless Sensor Networks,” IEEE Wireless Communications and Networking Conference, 2009, pp.1-6.

[3] N. Kaur and S. K. Sood, “An Energy-Efficient Architecture for the Internet of Things (IoT),” IEEE Systems Journal, Vol.11, No.2, 2017, pp.796-805.

[4] 서두옥, 이동호, “사물인터넷 환경에서 CoAP 기반의 저전력, 신뢰성 향상을 위한 경량 프로토콜,” 디지털산업정보학회, 논문집, 제15권, 제1호, 2019, pp.21-28.

[5] Z. Cen and M. W. Mutka, “Relocation of Hopping Sensors,” IEEE International Conference on Robotics and Automation, 2008, pp.569-574.

[6] M. Kim, M. W. Mutka and H. Choo, “On Relocation of Hopping Sensors for Rugged Terrains,” International Conference on Computational Science and Its Applications, 2010, pp.203-210.

[7] M. Kim, S. Park and W. Lee, “Energy and Distance-aware Hopping Sensor Relocation for Wireless Sensor Networks,” Sensors, Vol.19, No.7, 2019, p.1567.

[8] OMNeT Web Site. : https://www.omnetpp.rg accessed on November 2020).

[9] F. Cintr'on, “Network Issues for 3D Wireless Sensors Networks,” Ph.D. Dissertation, Michigan State University, USA, 2013.

[10] 김민수, “Hopping sensors에서 커버리지를 증가시키기 위한 전송방법에 관한 연구,” 성균관대학교 일반대학원 석사학위논문, 2012.

[11] M. Kim and M. W. Mutka, “Multipath-based Relocation Schemes Considering Balanced Assignment for Hopping Sensors,” IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp.5095-5100.

[12] M Kim and M. W. Mutka, "On Relocation of Hopping Sensors for Balanced Migration Distribution of Sensors," Springer-Verlag, LNCS 5593, 2009, pp.361-371.

[13] M. Kim and M. W. Mutka, "Recycled ID Assignment for Relocation of Hopping Sensors," IEEE World of Wireless, Mobile and Multimedia Networks, 2011, pp.1-3.

[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, 2019, p.49.

[15] S. Park, M. Kim, and W. Lee, "Success Rate Queue-based Relocation Algorithm of Sensory Network to Overcome Non-uniformly Distributed Obstacles," CMC - Computers, Materials & Continua, Vol.65, No.2, 2020,

pp.1181-1201.

[16] A. S. Rostami, M. Badkoobe, F. Mohanna, et al., "Survey on Clustering in Heterogeneous and Homogeneous Wireless Sensor Networks," Journal of Supercomputing, Vol.74, 2018, pp.277-323.

[17] 정성민, "원전 무선 센서 네트워크에 적합한 클러스터 헤드 체인 라우팅 프로토콜," 디지털산업정보학회, 논문집, 제16권, 제2호, 2020, pp.61-68.

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

이 우 찬
Lee, Woochan

2017년 9월~현재
인천대학교 전기공학과 조교수
2004년 4월~2017년 8월
특허청 사무관 / 서기관 대우
2005년 7월~2008년 6월
육군사관학교 전자공학 교수사관
2016년 12월 미국 Purdue University
전기컴퓨터공학부(공학박사)
2005년 2월 서울대학교 전기컴퓨터공학부
(공학석사)
2002년 2월 서울대학교 전기공학부(공학사)

관심분야 : 전자기수치해석, 머신러닝,
네트워킹
E-mail : wlee@inu.ac.kr

논문접수일 : 2020년 12월 2일
수 정 일 : 2020년 12월 9일
게재확정일 : 2020년 12월 15일

■ 저자소개 ■

김 문 성
Kim, Moonseong

2018년 9월~현재
서울신학대학교 교양학부 조교수
2009년 10월~2018년 8월
특허청 사무관 / 서기관 대우
2007년 12월~2009년 10월
미국 미시간주립대학교 연구원
2007년 2월 성균관대학교 전기전자 및
컴퓨터공학부 (공학박사)
2002년 8월 성균관대학교 수학과(이학석사)

관심분야 : 네트워킹, 정보보호, 수치해석
E-mail : moonseong@stu.ac.kr

박 수 연
Park, Sooyeon

2019년 3월~현재
인천대학교 전기공학과
전자기수치해석연구실
박사후연구원
2011년 2월 한국산업기술대학교 컴퓨터공학과
(공학박사)
2007년 2월 한국산업기술대학교 컴퓨터공학과
(공학석사)
2004년 2월 한국산업기술대학교 컴퓨터공학과
(공학사)

관심분야 : 네트워킹, 수치해석, 시뮬레이션
E-mail : anisoo@inu.ac.kr