무선센서네트워크 통신장애에서 원형 큐를 이용한 센싱 데이터 저장 및 복구 방법†

신동현, 김창화*
강릉원주대학교 컴퓨터공학과
e-mail : dhshin@cs.gwnu.ac.kr, kch@gwnu.ac.kr

A Method for Storing and Recovering Sensing Data using Circular Queue in Wireless Sensor Network Communication Failures†

DongHyun Shin, Changhwa Kim*
Dept of Computer Science&Engineering, Gangneung-Wonju National University

요 약

무선센서네트워크 (Wireless Sensor Network, WSN)는 최근 각광 받고 있는 스마트 서비스 (Smart Service)등에서 많이 응용되고 있다. 하지만 무선센서네트워크의 구성요소인 노드들이 각자 무선 전원을 사용하고 있어 전력 공급이 안정적이지 못하다는 점과 유선 통신에 비해 상대적으로 불안한 무선 통신을 사용한다는 점을 볼 때, 데이터를 송, 수신하는 과정에서 데이터가 손실될 확률이 굉장히 높다. 이러한 점을 고려해볼 때 중요한 데이터의 경우 통신 장애 발생 시 센싱 데이터 손실을 줄이고 신뢰성 높은 데이터 송, 수신을 위한 방안이 필요하다. 따라서 본 논문에서는 노드에서 데이터가 송, 수신될 때, 서버와의 접속 혹은 상위 노드와의 접속이 원활하지 않을 경우 생길 수 있는 데이터 손실을 줄이기 위해 원형 큐를 이용하는 방법을 제안한다.

1. 서론

무선센서네트워크는 저용량의 센싱 데이터를 가공하여 목적에 맞는 정보를 제공하며, 필요에 따라 각 노드 또는 사물들이 상황에 맞는 판단을 내리는 IoT (Internet of Things)를 위한 기술이기도 하다. 현재 버스 시스템, 공항, 해양 환경 등에서 활용되고 있으며, 특히 최근 각광받고 있는 홈 네트워킹 시스템 (Home Networking System)과 해양 환경에서의 참치 가두리 양식을 위한 정보를 얻기 위해 센서에서 수집된 데이터에 대해 데이터 마이닝 (Data mining)기법을 사용하는 것처럼 인간이 직접 취하기 어려운 데이터를 대신 수집하여 편의성을 높여준다[1].

이러한 무선센서네트워크의 구성요소는 센서, 센서노드, 싱크노드/게이트웨이, 서버, 사용자 등으로 이루어진다. 센서노드의 센서에서 수집된 데이터는 Zigbee와 같은 무선 통신으로 싱크노드 혹은 게이트웨이로 전달되고, 게이트웨이는 CDMA (Code Division Multiple Access), LTE (Long Term Evolution)등을 통해 서버로 전달되어 최종적으로 사용자에게 전해지게 된다.

일반적으로 노드간 무선 통신을 이용하여 데이터를 전달할 때 활용되는 IEEE802.11 DCF는 송신측에서 RTS (Request to Send)를 통해 수신측에 전송할 데이터가 있음을 알리고, 수신측은 이에 대한 응답으로 CTS (Clear to Send)를 통해 데이터를 전송해도 좋다고 알리면 비로소 데이터를 전송하게 된다[2].

이러한 무선 통신은 데이터 손실에 굉장히 취약한 요소들이 존재한다. 데이터 손실에 취약한 요소로 다음의 몇 가지 예를 들 수 있다. 첫째, 통신을 위한 모듈이 달려있는 노드는 무선 전원을 사용하고 있기 때문에 센서네트워크의 해결과제인 에너지 효율 (Energy Efficiency)이 굉장히 중요하다. 이는 곧 배터리의 전력이 모두 소모 되었을 경우 다른 노드로의 데이터 전송 시 문제가 생길 수 있음을 말한다. 둘째, 데이터를 전달하는 중간 과정에서의 통신이 불안정하거나 접속이 끊길 수 있다. 센서노드에서 게이트웨이로, 혹은 게이트웨이에서 서버로 데이터 전송 중에 통신이 불안정하거나 접속이 끊겨 데이터를 전송하지 못하는 경우에는 그동안 수집된 데이터들이 손실된다.

이러한 문제로 발생되는 데이터 손실을 방지하여 신뢰성 높은 데이터 수집이 가능할 수 있도록 하기 위해 본 논문에서는 원형 큐 (Circular Queue)를 이용한 알고리즘을 제안한다. 원형 큐를 이용한 알고리즘은 전원이 꺼져도 저장된 데이터가 보존되고, 512KB 이상의 플래시 메모리를 가진 MCU (Micro Controller Unit)가 사용되는 센서네트워크에 많은 양의 센싱 데이터를 저장할 수 있다.

본 논문은 2장에서 관련연구, 3장에서 본 논문에서 제안하는 알고리즘을 기술한다. 4장에서는 보간법을 이용하여 센싱 데이터들 사이의 값을 유추하는 방법을 다룬다.

† 본 연구는 국토해양부의 지원으로 수행하고 있는 “수중 광역 이동통신 시스템 개발” 사업 결과의 일부임을 밝히며 지원에 감사드립니다.
* 교신저자

2. 관련연구

2.1 센서네트워크에서의 데이터 저장 기법

센서네트워크는 무선 전원을 사용하기 때문에 에너지 효율이 굉장히 중요하다. 이를 고려한 센싱 데이터 저장 기법 중 트라젝토리 기반 데이터 저장 기법은 질의를 처리할 때 한 노드에 부하가 걸리지 않고, 주위의 여러 노드들로 분산시켜 네트워크의 수명을 늘리는데 도움을 준다[3, 4]. 이는 질의처리를 할 때, 부하를 여러 노드로 분산시키기 때문에 노드의 에너지 소모도 함께 분산되어 에너지를 더 효율적으로 관리할 수 있기 때문이다[4].

이 외에도 많이 사용되는 기법 중 데이터 중심 저장 기법이 있다. 이 기법은 데이터 전송 시 소모되는 에너지를 줄이기 위해 데이터 측정값의 안전영역 (Safe Region)을 설정 후 이 영역을 벗어나는 경우에만 데이터를 전송하게 하는 기법으로, 기존 기법에 비해 에너지 소모량이 60%가량 감소하는 것을 보였다[5].

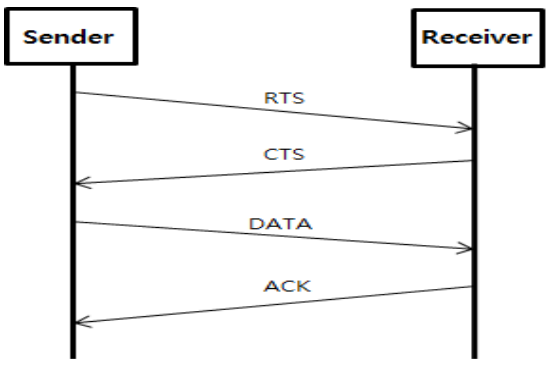
2.2 기존 데이터 저장 기법의 문제점

기존의 데이터 저장 기법은 센서네트워크에서의 에너지를 효율적으로 관리하기 위하여 이루어져왔다[3, 4, 5]. 따라서 최소한의 에너지로 많은 양의 데이터를 송, 수신할 수 있지만, 데이터 송, 수신 과정 중 통신에 문제가 생기면 데이터는 손실된다. 기존의 연구방법에서는 에너지 효율에 중점이 맞추어져있고, 통신 장애에 대한 요소는 고려되어있지 않았기 때문이다. 따라서 신뢰성 높은 데이터를 위하여 통신 장애로 인하여 중요한 데이터 손실을 줄일 필요가 있다.

3. 통신 프로토콜과 센싱 데이터 관리 알고리즘

3.1 통신 프로토콜과 통신 장애 식별

일반적으로 무선 통신에 사용되는 프로토콜은 IEEE 802.11 DCF다. 이 프로토콜은 (그림 1)과 같이 송신측에서 RTS를 보내면 수신측은 RTS에 대한 응답으로CTS를 전송한다. 송신측은 CTS를 수신하면 데이터를 전송하게 되고 수신측은 모든 데이터를 받은 후 ACK를 송신측으로 전송해주어야 한다.



(그림 1) IEEE802.11 DCF

본 논문에서 제안하는 알고리즘은 IEEE802.11 DCF 바탕이기 때문에 (그림 1)프로토콜에서 통신장애에 대한 식별이 필요하다. 통신장애는 <표 1>과 같이 식별한다.

<표 1> IEEE802.11 DCF에서의 통신장애 식별

주체	통신장애 요인
Sender	RTS에 대한 CTS 미 수신
	Data전송에 대한 ACK 미 수신
Receiver	CTS전송 후 DATA 미 수신

3.2 원형 큐를 이용한 센싱 데이터 관리

본 논문에서 제안한 알고리즘은 센싱 데이터를 전송하기 전에 원형 큐에 저장한다. 비정상적인 통신으로 인해 데이터를 전송할 수 없는 경우에는 본 논문에서 제안한 알고리즘에 의해 처리된다.

먼저, 알고리즘에서 사용되는 함수는 <표 2>에 정의되어있다.

<표 2> 사용 함수

Function	Explain
Sensing_Data _Queue(T, D);	·센싱 데이터를 원형 큐에 삽입 및 압축하는 함수((그림 3), (그림 5) 참조) ·T: Time, D: Data, P: Period ·(그림 3)알고리즘은 파라미터로 T, D만 사용하며, (그림 5)알고리즘은 파라미터로 T, D, P사용
Sensing_Data _Queue(T, D, P);	
Sensing_Data _Queue_Delete (CQ->CQueue[]);	·센싱 데이터 전송 후 원형 큐에 저장되었던 센싱 데이터 삭제
TIME _OUT();	·메시지 전송 후 RTS혹은 ACK에 대한 Time Out체크
Receive_ACK();	·ACK 수신
SEND_Data(CQ-> CQueue[]);	·데이터 전송
SEND_RTS();	·RTS 전송

전체적인 알고리즘의 순서는 (그림 2)과 같이 진행된다. 첫째, 전송할 센싱 데이터를 가지고 있다면 원형 큐 구조의 메모리에 저장한다. 둘째, IEEE802.11 DCF에 따라 통신을 시작 한다. 셋째, IEEE802.11 DCF에 따라 기존 원형 큐에 저장되어 있던 모든 데이터를 포함하여 데이터 전송 후 ACK를 수신하면 원형 큐 메모리 구조에서 전송한 데이터를 삭제한다. 여기에서 기존 원형 큐에 저장되어 있던 모든 데이터란, 통신 장애로 인하여 전송되지 못하고 원형 큐에 남아 있던 데이터들을 말한다. 이 때, 통신이 다시 연결되었다는 것은, 송신측이 수신측에 RTS를 전송하여 CTS수신하면 통신이 다시 연결된 것으로 판단하고, 원형 큐에 쌓여있던 데이터와 새로운 센싱 데이터를 순차적으로 재전송한다.

본 논문에서 원형 큐에 데이터 삽입 전 front와 rear를 확인하여, CQ->front==CQ->rear+1 mod CQ->Capacity 인 경우 Full로 식별하고, 데이터 삭제 후의 front와 rear가 CQ->front==CQ->rear인 경우 Empty로 식별한다.

```
/* Main Algorithm */
struct Queue{
  char front=rear=0;
  char Capacity=MAX_SIZE; //원형 큐 사이즈(홀수)
  char *CQueue=malloc(Capacity*sizeof(Data));
};
struct Queue *CQ=malloc(sizeof(struct Queue));
while(1){
  Sensing_Data_Queue(T, D, P);
  SEND_RTS(); //RTS전송
  while((!(TimeOut=TIME_OUT()))
        &&((ReceiveCTS=Receive_CTS())==FALSE);
  if(ReceiveCTS){
    SEND_Data(CQ->CQueue[]); //최대 데이터 전송
    while((!(TimeOut=TIME_OUT()))
          &&((ReceiveCTS=Receive_CTS())==FALSE);
    if(ReceiveACK){
      Sensing_Data_Queue_Delete(CQ->CQueu[]);
      CQ->front=(CQ->front+1) mod CQ->Capacity;
    }
  }
}
```

(그림 2) 주 알고리즘

3.3 FIFO방식의 원형 큐 압축 알고리즘

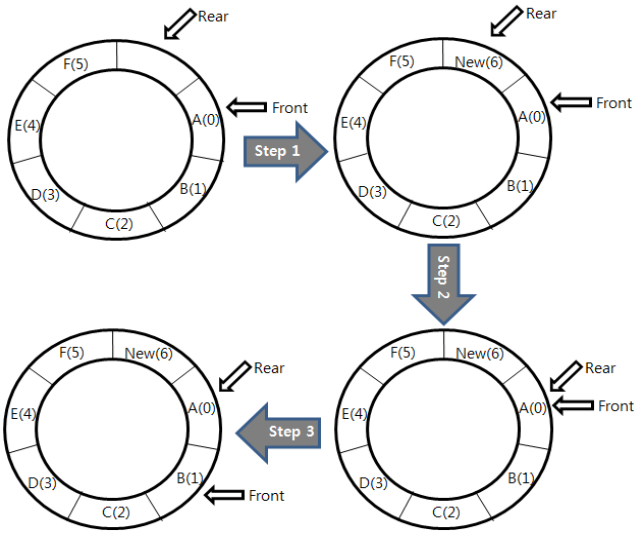
(그림 2)에서 Sensing_Data_Queue();는 정상적인 데이터의 송, 수신이 이루어지지 않은 경우와 정상적으로 데이터의 송, 수신이 이루어진 경우에는 원형 큐에 데이터를 저장하고, 원형 큐가 가득 차있다면 원형 큐를 압축 하는 알고리즘이다. 큐에 데이터가 저장된 후 정상적으로 데이터가 송신되면 큐에 저장 된 데이터는 삭제되지만 그렇지 않은 경우에는 원형 큐 압축 알고리즘에 의해 원형 큐에 저장된 데이터가 계속 유지된다. 이 경우 원형 큐의 최대 크기보다 데이터의 크기가 초과될 경우 더 이상 데이터를 저장할 수 없게 된다. 이러한 경우 원형 큐 압축을 통해 계속해서 데이터를 저장하기 위한 2가지 알고리즘을 제안한다.

첫 번째 알고리즘인 (그림 3)은 다음과 같은 순서로 진행된다. 첫째, 원형 큐가 가득 차있는지 체크한다. 원형 큐가 가득 찬 경우 원형 큐 압축과정인 (그림 4)을 진행한다. 원형 큐가 가득 찬 경우의 압축과정은 큐에 저장된 가장 오래된 데이터를 삭제하고, 원형 큐에 새로운 센싱 데이터를 삽입하는 과정으로 이루어진다. 이때, 가장 오래된 데이터를 판단하기 위해서는 시간 값과 데이터를 파라미터로 넘겨받아야 한다. 이와 반대로 큐가 가득 차있지 않다면 센싱 데이터를 삽입한다.

이 알고리즘은 최근 출시되는 MCU의 플래시 메모리의 용량이 512KB이상인 것을 감안하면 충분히 활용이 가능하다고 볼 수 있다. 한 예로, 센싱 데이터의 크기가 약 1byte, 저장가능 메모리가 300KB라면, 센싱 데이터를 약 300,000개 이상 저장 가능하며, 이는 센싱 데이터의 주기가 1시간이라고 할 때 12,500일의 센싱 데이터를 저장할 수 있고, 1분마다 저장한다면 208일, 10초마다 저장해도 약 34.7일간 저장할 수 있는 용량이다.

```
/* Circular Queue Algorithm Based on FIFO */
Algorithm Sensing_Data_Queue(char Time, string Data){
  if(CQ->front==(CQ->rear+1 mod CQ->Capacity)){
    CQ->CQueue[CQ->rear]=Data;
    CQ->rear=(CQ->rear+1) mod CQ->Capacity;
    CQ->front=(CQ->front+1) mod CQ->Capacity
  }
  else{ //큐에 데이터 삽입
    CQ->CQueue[CQ->rear]=Data;
    CQ->rear=(CQ->rear+1) mod CQ->Capacity;
  }
  return;
}
```

(그림 3) FIFO방식의 원형 큐 압축 알고리즘



(그림 4) FIFO방식의 원형 큐 압축과정

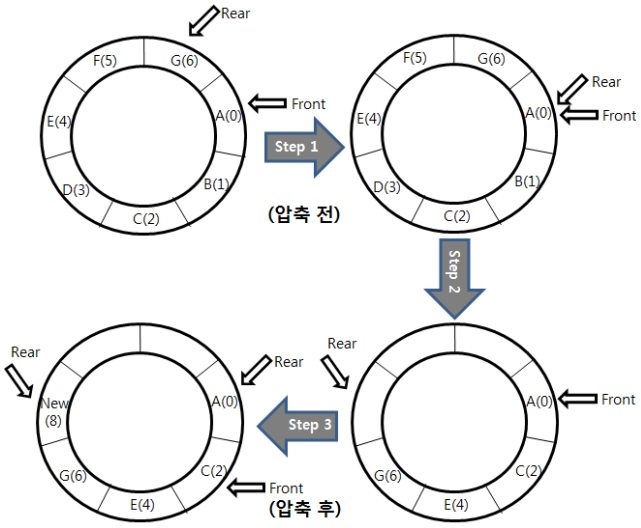
3.4 주기변경을 이용한 원형 큐 압축 알고리즘

두 번째 알고리즘 (그림 5)는 다음 순서를 따른다. 첫째, 원형 큐가 가득 차있는지 체크한다. 만약, 원형 큐가 가득 차있다면 (그림 6)의 절차에 따라 원형 큐에 저장되어있는 데이터들을 재구성 한다. 이때, 홀수 번째 위치에 저장되어 있는 데이터들만 순차적으로 재구성하여, 2배의 주기에 해당하는 데이터들만 남겨둔다. 이 경우 큐의 데이터 저장 공간이 짝수 개라면, 다음에 저장되는 데이터의 주기가 맞지 않기 때문에 큐는 홀수개로 구성한다. 큐를

재구성 하고 나면 센싱 데이터 주기를 2배로 늘리고, 데이터를 수신한다. 만약, 원형 큐가 가득 차있지 않다면 원형 큐에 센싱 데이터를 삽입한다. 이 방법은 추후 보간법을 이용하여 센싱 데이터를 유추할 때 주기에 대한 정보가 필요하므로, 주기 정보를 따로 관리한다. 이 알고리즘은 원형 큐가 가득 차더라도 기존의 센싱 데이터의 근사치 정보를 가지고 있으므로 알고리즘1과 비교하였을 때, 더욱 높은 신뢰성의 센싱 데이터를 보존할 수 있다.

```
/* Circular Queue Algorithm Based on Period Change */
Algorithm Sensing_Data_Queue(char Time, string Data, int Period){
  if(front==(rear+1 mod CQ->Capacity){
    CQ->Queue[CQ->rear]=Data;
    for(int i=0; i<=(CQ->rear/2); i++){
      CQ->Queue[i]=CQ->Queue[2*i]
    }
    Period=2*Period;
    CQ->rear=(CQ->rear)/2+1;
  }
  else{
    CQ->Queue[CQ->rear]=Data;
    CQ->rear++;
  }
  return;
}
```

(그림 5) 주기변경을 이용한 원형 큐 압축 알고리즘



(그림 6) 주기변경을 이용한 원형 큐 압축과정

4. 임의의 시점에서의 센싱 데이터 유추방법

큐를 재구성 하는 경우 원래의 주기대로 수집된 데이터는 알 수 없게 된다. 이때, 보간법을 이용하여 우리가 원하는 주기에서 혹은 임의의 시점에서의 센싱 데이터를 유추할 수 있다. 본 절에서는 보간법을 이용하여 센싱 데이터를 유추하는 방법을 서술한다. 보간법에는 여러 종류가 있는데, 센서네트워크에서 측정되는 값들은 보통 등간격이 아니기 때문에 이에 적합한 뉴튼 (Newton)의 분할 차분법을 이용하여 데이터를 유추하도록 한다. 이 보간법은 현재 사용되는 보간법 중 대표적인 보간법중 하나로 다음과 같은 수식에 의해 구할 수 있다[6, 7].

$$p_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots + a_n(x-x_0)(x-x_1)(\ldots)(x-x_{n-1})$$

예를 들어, 원래 1시간마다 데이터가 측정되어야 하지만 원형 큐 압축 알고리즘에 의해 14시 25.5도, 16시 28도, 18시 26.5도의 데이터가 저장되어 있다고 가정하다. 이 데이터를 뉴튼의 분할 차분법에 대입하여 15시의 데이터를 유추하기 위해 다음과 같은 식을 세울 수 있다.

$$25.5+1.25(2-1)-0.5(2-1)(2-3)=26.8$$

위의 식을 계산하면, 15시의 온도가 26.8임을 유추할 수 있다. 센싱 데이터의 주기가 바뀌더라도 위와 같은 보간법을 이용하면 우리가 알고자 하는 시점의 센싱 데이터를 유추해낼 수 있다.

5. 결론

무선센서네트워크는 인간이 직접 얻기 힘든 정보를 쉽게 얻을 수 있도록 도와주며, 인간에게 많은 유용성을 제공한다. 이러한 정보들은 센서에서 수집된 데이터를 바탕으로 제공되는 것이기 때문에 신뢰성 높은 데이터의 수집은 필수적이다. 따라서 본 논문에서는 무선 통신의 특성상 쉽게 손실될 수 있는 데이터를 신뢰성을 높이기 위한 알고리즘을 제안하였다. 이를 통해 손실되는 데이터는 줄어들 것이며, 보간법을 이용하여 원본 데이터에 가까운 데이터를 복구할 수 있을 것으로 기대된다.

향후 연구 과제로는 무선센서네트워크에서 본 알고리즘을 적용하고, 원본 데이터와 비교하여 센싱 데이터를 유추하였을 때의 데이터의 신뢰성과 성능을 평가하도록 한다.

참고문헌

[1] 남상엽, 정교일, 김성동, 유비쿼터스 센서 네트워크, 성학당, 2006
[2] FOROUZAN, Data Communiations and Networking 5/E, McGraw-Hill, 2013
[3] 임화정, 이헌길 “센서네트워크에서의 프록시 트라젝토리 기반 데이터 저장 기법”, 정보처리학회논문지, Vol.c15, No.6, pp.513-522, 2008
[4] 임화정 외 4인 “센서네트워크를 위한 적응적 지역 트라젝토리 기반의 데이터 저장소 기법”, 정보처리학회논문지, Vol.c15, No.1, pp.19-30, 2008
[5] 노규종 외 5인, “데이터 중심 저장 기법을 위한 효율적인 센서 데이터 압축 기법″, 한국콘텐츠학회논문지, 제 10권, 제 11호, pp.58-67, 2010
[6] 이관수, Numerical Methods for Engineers, 세화, 2014
[7] 최종근, Numerical analysis, Textbooks, 2010