센서 네트워크 환경에서 이웃 노드들의 상황 인지를 이용한 라우트 선택 기법

김미정 O* 이동현**, 인호**
*고려대학교 컴퓨터정보통신대학원, **고려대학교 정보통신대학
{mary914O, tellmehenry, hoh_in}@korea.ac.kr

SARS : Route Selection using Situation Aware of Neighbors in Sensor Networks

Mi-jeong Kim O* Dong-hyun Lee**, Hoh Peter In**
*Graduate School of Computer and Information Technology, Korea University
**College of Information and Communications, Korea University

요 약

센서 네트워크는 배터리 교체가 어려운 저가의 소형, 저전력 센서 노드들로 구성되며 멀티 홉 방식으로 통신하기 때문에 센서 네트워크 수명을 연장하기 위한 라우트 선정 방법은 중요하다. 그러나 다수의 라우팅 기법은 노드의 정적인 상태 정보만 이용하여 라우트를 선택하기 때문에 링크 상태에 따른 데이터 손실이나 특정 노드의 에너지 부족에 의해 발생하는 패스 손실과 같은 동적인 상황은 반영하지 못하는 문제점이 있다. 본 논문에서는 이웃 노드들의 상황을 인지하여 가장 데이터를 전달 받기 좋은 상황에 있는 노드를 라우트로 선택하는 기법을 제안하며, 제안하는 기법을 SARS(Situation Aware Route Selection)라 부르겠다. 제안하는 SARS는 이웃 노드들의 상황을 고려하여 라우트를 선택하기 때문에 데이터 손실이나 패스 손실에 의한 재전송 노드의 수를 최소화 할 수 있고, 이를 통해 센서 네트워크 수명이 연장되는 효과를 얻을 수 있을 것으로 기대된다.

1. 서론

최근 유비쿼터스 컴퓨팅 비전을 실현하기 위한 기반 기술 중 하나로 센서 네트워크 기술이 대두되고 있다[1][2]. 이유는 센서 및 RFID 태그가 부착된 노드로부터 주변 환경 정보뿐만 아니라 사물에 대한 인식정보를 수집할 수 있고, 수집된 데이터는 노드간 통신을 통해 정보 처리 능력이 있는 시스템에 실시간으로 전달되어 군사지역 모니터링, 물류관리, 헬스케어, 홈네트워크, e-러닝 등 다양한 분야에서 응용가능하기 때문이다. 그러나 유비쿼터스 컴퓨팅 환경이 도래했음에도 불구하고 센서 네트워크는 저가의 소형, 저전력 센서 노드들로 구성되기 때문에 제한된 에너지를 효율적으로 관리할 수 있는 기법은 여전히 중요한 기술적 이슈 중 하나이다[3].

센서 노드의 전송 반경은 에너지를 절약하기 위해 근거리로 제한되기 때문에 데이터는 멀티 홉을 거쳐서 싱크 노드로 전달된다. 그리고 센서 네트워크 환경은 데이터 손실이 발생할 수 있으며 센서 필드 내에 배포되는 노드의 밀도가 높기 때문에 어떠한 노드를 선택하여 라우팅 패스를 구성하느냐에 따라 에너지 효율성이 달라질 수 있다. 기존 라우팅 기법들을 살펴보면, 새로운 라우트가 요구될 때, 소스 노드는 이웃 노드들과 통신을 통해 이웃 노드들을 탐색하고, 이웃 노드로부터 정보를 수집하여 하나 또는 일부 노드를 라우트로 선택한다[4][5][6]. 이웃 노드들은 라우팅 정책에 따라 수집된 정보를 이용하여 좋고 나쁨을 구분할 수 있다. 예를 들어, 최단 거리 패스를 구성할 경우, 이웃 노드들 중 최소 홉 카운트를 갖는 노드는 좋은 노드로, 그 외는 나쁜 노드로 분류 가능하다. 최소 에너지를 소모하는 라우팅 패스를 구성한다면, 이웃 노드들 중 최소 에너지를 소모하는 노드는 좋은 노드로, 나머지는 나쁜 노드로 구분할 수 있다. 그러나 홉 카운트나 에너지 소모량은 노드의 정적이고 단편적인 상태만 나타내기 때문에 링크 상태에 따라 발생할 수 있는 데이터 손실이나 임의의 노드만 에너지를 빨리 소모하여 발생할 수 있는 패스 손실과 같은 동적인 상황을 반영하지 못하는 문제점이 있다.

본 논문에서 제안하는 SARS(Situation Aware Route Selection)는 이웃 노드에 대한 상황 인지를 기반으로 라우트를 선택한다. 이웃 노드들의 상황을 인지하기 위한 파라미터로 노드가 배포된 위치나 네트워크 토폴로지 및 에너지 효율성을 높일 수 있는 다양한 노드의 상태 등이

이용될 수 있다. 제안하는 SARS는 이웃 노드들이 처한 상황을 고려하여 이웃 노드들 중 가장 데이터를 전달 받기 좋은 상황에 있는 노드를 라우트로 선택하기 때문에 데이터 손실이나 패스 손실에 의한 재전송 횟수가 줄어들게 되어 센서 네트워크 수명이 연장될 것으로 기대된다.

본 논문의 나머지는 다음과 같이 구성된다. 2장은 관련 연구로 센서 네트워크 환경에서 에너지 효율성을 향상시키기 위한 라우팅 기법에 대해 설명하겠다. 3장에서는 제안하는 SARS에 대한 개요와 센서 노드의 상황을 인지할 수 있는 파라미터를 정의하고 SARS 동작 방식 및 알고리즘 에 대해 설명하겠다. 그리고 4장에서 실험 환경 및 결과에 대해 기술하고, 5장에서 결론 및 향후 연구 과제에 대해 기술하겠다.

2. 관련 연구

센서 네트워크 환경에서 다수의 노드들이 싱크 노드로 데이터를 전달하는 가장 간단한 방법은 Flooding이다. 그러나 센서 필드 내에 위치한 모든 노드로 데이터를 전달하기 때문에 Broadcast Storm이 발생될 수 있고, 중복된 데이터 수신에 의해 불필요한 에너지가 낭비되는 문제점이 발생된다. 이 문제를 해결하기 위해 연구된 라우팅 프로토콜은 다음과 같다.

SPIN(Sensor Protocols for Information via Negotiation)[5]은 광고 메커니즘(ADV-REQ-DATA 메시지 이용)을 이용하여 데이터에 관심이 있는 노드를 라우트로 선택하여 데이터를 전달하는 방식으로 중복 수신에 의한 문제점을 해결하였다. 그러나 데이터에 관심을 보이기만 하면 데이터를 전달하기 때문에 불필요한 노드들을 거쳐 싱크 노드로 데이터가 전달되거나 전송 지연이 발생할 수 있다. 그리고 모든 노들이 데이터에 관심을 보일 경우, Flooding에서 나타난 문제와 유사한 메타 데이터 Broadcast Storm이 발생하는 문제점이 나타난다.

GBR(Gradient Based Routing)[6]은 센싱된 모든 데이터가 싱크 노드로 전달되는 현상을 이용하여 라우팅 패스를 구성하는 방식이다. 각 노드는 싱크 노드로부터 생성된 쿼리를 받을 때마다 자신이 싱크 노드로부터 몇 홉 거리에 있는지 판단하고, 쿼리의 홉을 하나씩 증가시켜 다른 노드로 전달한다. 이렇게 형성된 패스를 이용하여 데이터를 전달하면 루프 없이 전달할 수 있기 때문에 에너지 효율성이 향상된다. GBR을 이용하여 최단거리 라우팅 패스를 구성한다고 가정할 경우, 센서 네트워크 환경이 정적이고 데이터 손실이 발생하지 않는다면, 최단거리 패스는 최소 전송 횟수를 의미하기 때문에 가장 에너지 효율적인 데이터 전달 방법이 될 수 있다. 하지만 실제 센서 네트워크 환경은 동적이며 데이터 손실이 발생할 수 있기 때문에 재전송에 의한 추가 에너지 소모도 고려해야 한다. 그리고 센서 노드는 데이터 전송뿐만 아니라 라우트 역할도 수행하기 때문에, 최적화된 패스만 이용할 경우, 라우팅 패스에 속한 노드만 일찍 에너지가 소모되어 센서 네트워크가 동작하지 않는 문제점이 발생할 수 있다.

센서 네트워크 수명은 센서 노드들의 균등한 에너지 소모를 통해 연장될 수 있기 때문에 EAR(Energy Aware Routing)[7]는 최적화된 라우팅 패스 외에 대체 경로를 두어 복수개의 패스 중 하나를 확률에 의해 선택하는 방식이다. 그러나 이 기법 역시 전송할 데이터가 발생할 때마다, 미리 정해진 라우팅 패스들 중 하나를 선택하기 때문에 센서 필드 내에 배포된 모든 노드를 균등하게 사용한다고 말할 수 없다. 그리고 임의의 노드가 복수개의 패스에 포함될 경우, 특정 노드의 에너지만 소모되기 때문에 라우팅 패스 단절에 의해 센서 네트워크 수명이 단축되는 한계가 있다.

3. 제안 내용 : 상황인지 기반 라우트 선택

본 논문에서 제안하는 SARS(Situation Aware Route Selection)는 다수의 노드가 하나의 싱크로 데이터를 전달하는 센서 네트워크를 위한 프로토콜이다. 그림 1은 SARS의 구조를 나타낸다. 임의의 노드가 라우트를 선택해야 할 때, 통신을 통해 이웃 노드의 상태를 나타내는 컨텍스트를 수집하고, 수집된 컨텍스트로부터 노드의 상황을 인지할 수 있는 파라미터를 유추하여 가장 데이터를 수신하기 좋은 상황에 있는 이웃 노드를 라우트로 선택하는 구조이다. 3.1에서는 이웃 노드로부터 수집할 컨텍스트와 이웃 노드의 상황을 인지할 수 있는 파라미터를 정의하고 3.2에서는 SARS의 동작 방식 및 알고리즘에 대해 설명하겠다.

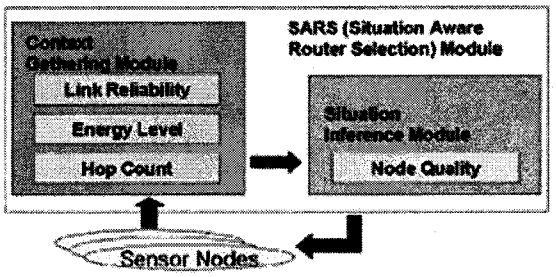

그림 1 SARS 구조

3.1. 센서 노드를 위한 컨텍스트와 상황인지

상황인지에서 일반적으로 사용되는 용어인 컨텍스트는 개체가 처한 상황을 특징지을 수 있는 상태를 의미하며, 센서 네트워크를 통해 수집된 컨텍스트를 분석하여 사용

자뿐만 아니라 시스템의 상황을 인지하기 위해 사용된다[8]. 상황인지 네트워크[9][10]에서 소개된 상황인지는 물리 계층 기능으로 셀룰러 네트워크를 구성하는 기지국 주변 환경을 관찰하고, 기지국의 현재 상태를 나타내는 컨텍스트를 수집하여 최적의 네트워크 환경을 유지하기 위한 서비스를 제공한다.

센서 노드를 위한 컨텍스트는 센서 네트워크 자체의 동적인 특성과 어플리케이션 종류에 따라 다양한 상태를 고려할 수 있다. 그러나 제안하는 SARS에서는 그림 2에서 볼 수 있듯이 라우트 선정 기준으로 이용할 수 있는 노드의 링크 신뢰도, 잔여 에너지량, 홉 카운트만 고려한다.

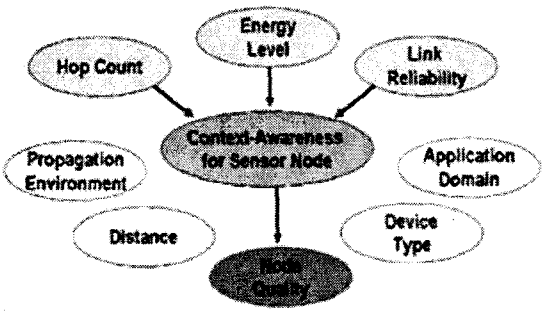

그림 2 센서 노드를 위한 컨텍스트

컨텍스트 1. 링크 신뢰도 (Link Reliability)

링크 신뢰도는 소스에서 전송한 데이터가 목적지에 오류 없이 전달될 확률을 의미하며 주로 데이터 링크 계층에서 전송 에러 발견 및 교정에 이용된다[3]. 센서 네트워크는 무선 통신을 근간으로 하기 때문에 본래 동적이고 데이터 손실이 발생할 수 있으며, 데이터 전송 주기에 따라 다양한 연결성이 나타나기 때문에[3][11] 노드마다 다른 링크 신뢰도가 나타날 수 있다. 만약 라우트로 링크 신뢰도가 낮은 노드를 선택하여 데이터 손실이 발생할 경우, 데이터 전송에 소모된 에너지 자체가 무의미할 뿐만 아니라 어플리케이션에 따라 재전송이 요구되기 때문에 추가 비용이 발생할 수 있다. 그리고 센서 노드에서 수행되는 태스크 중 전송이 가장 큰 에너지 소모 비율을 차지하기 때문에[12] 재전송에 의한 추가 비용은 민감한 요소로 작용한다. 따라서 이웃 노드들 중 링크 신뢰도가 가장 높은 노드는 링크 신뢰도가 낮은 노드보다 데이터 손실 확률이 적기 때문에 좋은 상황에 있는 노드로 추정할 수 있다. 본 논문에서는 RSSI(Received Signal Strength Indicator)를 이용하여 링크 신뢰도를 측정한다.

컨텍스트 2. 잔여 에너지량 (Energy Level)

실제 센서 네트워크에서 데이터를 센싱하고, 싱크 노드로 데이터를 전달하는 이벤트는 모든 노드에서 균등하게 발생하지 않을 수 있다. 그리고 노드에서 수행되는 태스크 마다 다른 에너지를 소모하기 때문에[12] 노드마다 시간이 경과됨에 따라 잔여 에너지량은 다를 수 있다. 만약 임의의 노드만 일찍 에너지를 소모할 경우, 데이터 전송 단절 및 패스 손실에 의해 센서 네트워크 수명이 단축될 수 있다. 따라서 센서 필드 내에 배포된 노드들의 균등한 에너지 소모를 유도하기 위해 이웃 노드들 중 잔여 에너지량이 많을수록 좋은 상황에 있는 노드로 추정할 수 있다.

컨텍스트 3. 홉 카운트 (Hop Count)

홉 카운트는 라우팅 비용을 측정하기 위한 가장 간단한 방법이다. 홉 카운트는 임의의 노드로부터 떨어진 거리를 추정할 수 있는 기준이 될 수 있고, 소스에서 목적지까지 데이터를 전달하기 위한 전송 방향성을 나타내는 기준으로 이용될 수 있다. 따라서 싱크 노드를 기준으로 홉 카운트를 계산할 경우, 이웃 노드들 중 홉 카운트가 작을 수록 싱크 노드에 보다 가까이 위치해 있기 때문에 좋은 상황에 있는 노드로 추정할 수 있다.

위에서 정의한 센서 노드를 위한 세가지 컨텍스트는 임의의 노드 n과 통신 과정을 통해 수집할 수 있다. 이웃 노드로부터 수집된 컨텍스트를 이웃 노드의 상황을 인지하기 위한 파라미터는 그림 3과 같이 정의한다.

$$NQ(n,t) = LR(n,t) + EL(n,t) / (HC(n) * maxEL(n))$$
$$LR(n,t) = SSI(n,t) \mid DSSI(n,t)$$
$$SSI(n,t) = sign(SS(n,t))$$
$$SS(n,t) = RSSI(n,t) + 86$$
$$DSSI(n,t) = sign(DSS(n,t))$$
$$DSS(n,t) = SS(n,t+1) - SS(n,t)$$

그림 3 상황인지를 위한 파라미터

표 1 파라미터 표현 기호

n	센서 필드 내에 배포된 임의의 노드
t	측정 시간
NQ	노드의 품질
LR	노드의 링크 신뢰도
EL	노드의 잔여 에너지량
HC	노드의 홉 카운트
maxEL	노드의 최대 에너지량
SSI	신호 세기 인자
DSSI	신호 세기 차이 인자
sign	0 보다 크거나 같으면 1, 작으면 0 반환

RSSI값이 -86dBm 이상이면 링크 상태가 좋다고 추정할 수 있기 때문에[13] 그림 3에서 신호 세기 인자 값인 SSI가 1이면 링크 신뢰도가 높고, 0이면 링크 신뢰도가 낮다고 추정한다. 그리고 신호 세기 차이를 나타내는 DSS는 링크 안정성을 추정하기 위한 파라미터로 ASBM 방식을 [14] 이용하여 측정한다. 만약 신호 세기 차이 인자 값을 나타내는 파라미터 DSSI가 1이면 신호 세기가 강해지기 때문에 링크 신뢰도가 높고, 0이면 신호 세기가 약해지기 때문에 링크 신뢰도가 낮다고 추정한다. 그림 3에서 정의한 식에 의해 NQ 값이 클수록 링크 신뢰도가 높아 데이터 손실 확률이 적고 노드의 잔여 에너지량이 많아 패스 손실 확률이 적으며 싱크 가까이 위치한 노드로 추정할 수 있다. 만약 두 개 이상의 노드로부터 동일한 NQ값을 얻을 경우, 잔여 에너지량이 큰 노드를 선택한다.

3.2. 이웃 노드의 상황인지를 이용한 라우트 선택

제안하는 SARS는 다수의 노드가 하나의 싱크 노드로 데이터를 전달하기 위한 기법이기 때문에 홉 카운트는 싱크 노드로부터 떨어진 거리를 의미한다. 홉 카운트는 거리 백터 라우팅 기법을 이용하여 구할 수 있기 때문에 그 과정은 생략한다. 그리고 모든 노드는 고정형 노드이며 자신의 홉 카운트를 알고 있다고 가정한다.

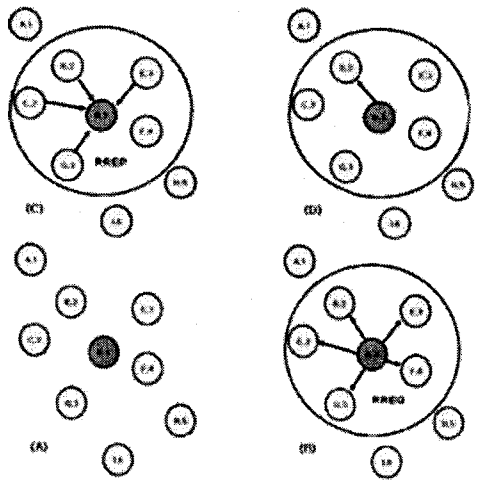

그림 4 SARS 동작 방식

이웃 노드의 상황을 인지하려면 우선 주변에 어떤 노드가 있는지 탐색하고, 이웃 노드로부터 3.1에서 정의한 컨텍스트를 수집해야 한다. SARS는 AODV(Ad hoc On Demand Distance Vector)[15] 라우팅 프로토콜을 변형하여 그 과정을 수행한다(RREQ-RREP 메시지 이용).
그림 4의 (A)는 센서 필드 내에 배포된 임의의 센서 노드들을 의미하며, 노드를 식별하기 위해 편의상 (노드 아이디, 홉 카운트)로 구분한다. 그림 4의 (B)에서 볼 수 있듯이 노드 D에서 싱크 노드로 전송할 데이터를 센싱하거나 이웃 노드로부터 데이터를 전달 받은 경우, 소스 노드 D는 자신의 홉 카운트 3을 RREQ 메시지에 첨부하여 브로드캐스팅한다. 홉 카운트를 함께 알리는 이유는 3.1에서 언급했듯이 홉 카운트가 싱크 노드로 데이터를 전달하기 위한 전송 방향성을 나타내기 때문이다. 노드 D의 전송 반경 내에 위치한 노드들은 RREQ 패킷을 수신하게 되는데, 라우팅 루프를 방지하기 위해 수신 노드의 홉 카운트가 소스 노드의 홉 카운트보다 큰 경우 RREQ 메시지를 무시한다. 그리고 홉 카운트가 작거나 같은 경우, 자신의 잔여 에너지량과 홉 카운트를 RREP 패킷에 첨부하여 소스 노드로 전송한다. 소스 노드는 이웃 노드로부터 RREP 패킷을 수신하면 이웃 테이블에 노드 아이디와, 수집한 컨텍스트를 이웃 테이블에 등록한다. 그림 4의 (C) 까지 수행되면 소스 노드를 중심으로 한 홉 이내의 거리에 있는 노드들과 이웃 그룹이 형성되고, 그림 4에서 노드 B, C, E, G가 이웃 테이블에 관리된다. 그리고 소스 노드는 이웃 노드의 상황을 추정하기 위해 이웃 노드로부터 수집한 컨텍스트를 이용하여 그림 3의 파라미터 값을 도출하고, 이웃 노드들 중 가장 노드 품질이 높은 노드를 선택하여 라우팅 테이블에 등록시키고 SARS_TTL을 세팅한 후 데이터를 전달한다. 라우트로 선택된 노드가 소스 노드가 되어 그림 4의 (A) ~ (D) 과정을 싱크 노드를 만날 때까지 반복한다. 매번 이 과정을 반복 할 경우, 이웃 노드의 상황을 인지하는데 많은 에너지가 소모될 수 있다. 따라서 이웃 노드들에 대한 상황인지 유효 시간인 SRAR_TTL이 유효한 경우, 노드의 라우팅 테이블을 참조하여 데이터를 전달하고, SARS_TTL이 만료된 경우 그림 4의 (A) ~ (D)과정을 수행한다.
그림 5, 그림6은 이웃 노드를 탐색하기 위한 알고리즘이고, 그림 7은 이웃 노드의 상황을 인지하여 라우트를 선택하기 위한 알고리즘이다.

알고리즘1. Neighbor Discovery (for source)

```
01. For Source Node
02.     route = 0
03.     rreq_ttl = 0
04.     if under SARS_TTL then
05.         route = neighbor_ID        //select route from routing table
06.         send(route, DATA)
07.     else if over SARS_TTL then
08.         rreq_ttl = MAX_RREQ_TTL
09.         RREQ.hopCnt = myHopCount
10.         send(broadcast, RREQ)
11.     end if
12. End For Source Node
```

그림 5 이웃 노드 탐색을 위한 알고리즘(소스용)

```
알고리즘2. Neighbor Discovery (for neighbor)
01. For Neighbor Node
02.     if (hop_count > RREQ.hopCnt) then
03.         stop
04.     end if
05.     RREP.hopCnt = myHopCount
06.     RREQ.energy = myEnergy
07.     send(source, RREP)
08. End For Neighbor Node
```

그림 6 이웃 노드 탐색을 위한 알고리즘(이웃용)

```
알고리즘3. Gather Context & Route Selection
01. For Source Node
02.     RL = 0
03.     // gather context
04.     while (req_ttl---- > 0)
05.         receive(RREP)
06.         get RSSI
07.         id = RREP.src
08.         hopCount = RREP.hopCnt
09.         energy = RREP.energy
10.         if (SSI >= 0) then
11.             RL = 1
12.         else if ((0)SSI >= 0) then
13.             RL = 1
14.         else
15.             RL = 0
16.         end if
17.         NQ = RL + energy / (hopCount * max_energy)
18.         insert context to neighbor table
19.     end while
20.
21.     // route selection
22.     while (node++ <= MAX_NODE_SIZE)
23.         max(RL, energy, hopCount, id)
24.     end while
25.     insert id to route table
26.     SARS_TTL = MAX_SARS_TTL
27.     send(id, DATA)
28. End For Source Node
```

그림 7 컨텍스트 수집 및 라우트 선택 알고리즘

4. 실험 환경 및 결과

실험은 TinyOS에서 제공하는 event driven 방식의 시뮬레이션 툴인 TOSSIM[16]을 기반으로 진행하였다. 그리고 오픈 소스로 제공되는 TinyOS용 프로토콜인 TinyAODV[17]와 재전송 노드의 수, 잔여 에너지량 비율, 싱크 노드의 수신율에 대해 비교 평가 하였다.

4.1 실험 환경

실험을 위해 TOSSIM에서 제공하는 두 가지 Radio 모델 중 가우시안 패킷 손실 확률을 적용한 lossy 모델을 채택하고, 노드간 거리는 10 피트 간격을 유지하도록 하여 20x20 그리드로 구성하였다. 모트 0을 싱크 노드로 설정하여 그리드 가장자리에 배치하였다. 센서 필드내에 배치되는 모든 노드는 싱크 노드를 기준으로 홉 카운트를 계산하여 설정하였고, 주변 노드들 중 8개의 노드를 이웃으로 관리하도록 설정하였다. 그리고 시간이 경과됨에 따라 변경되는 잔여 에너지량을 측정하기 위해 [12]에서 측정한 전력 모델을 이용하였다[표2].

표 2 Mica2에서 측정한 전력 모델

Mode	Current	Mode	Current
CPU		Radio	
Active	8.0 mA	Rx	7.0 mA
Idle	3.2 mA	Tx (-20 dBm)	3.7 mA
Power-down	103 μA	Tx (-19 dBm)	5.2 mA
Power-save	110 μA	Tx (-15 dBm)	5.4 mA
Standby	216 μA	Tx (-8 dBm)	6.5 mA
LEDs	2.2 mA	Tx (-5 dBm)	7.1 mA
Sensor board	0.7 mA	Tx (0 dBm)	8.5 mA
EEPROM access		Tx (+4 dBm)	11.6 mA
Read	6.2 mA	Tx (+6 dBm)	13.8 mA
Read Time	565 μs	Tx (+8 dBm)	17.4 mA
Write	18.4 mA	Tx (+10 dBm)	21.5 mA
Write Time	12.9 ms		

4.2 실험 결과

다음 실험 결과들은 네트워크 크기 별로 1000sec 동안 시뮬레이션 한 결과이다. 네트워크 크기별로 40%의 노드만 패킷을 생성하여 싱크 노드로 전송하였고, 노드별 재전송 횟수는 최대 3회로 제한하였다.

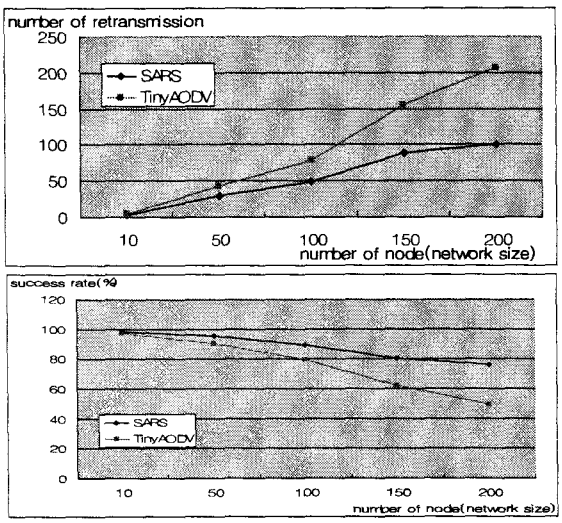

그림 8 센서 네트워크 크기에 따른 재전송 수(a)와 싱크 노드에서 메시지 수신 비율(b)