무선 센서네트워크에서 효율적인 양방향 라우팅 프로토콜

안태원 ° 조인휘
한양대학교 정보통신 대학원
fryingpan@ihanyang.ac.kr, iwjoe@hangyang.ac.kr

An Efficient Bi-directional Routing Protocol for Wireless Sensor Networks

TaeWon Ahn ° InWhee Joe
The Graduate School of Information and Communication, Hanyang University

요 약

현재 컴퓨터에서 사용되는 수의 체계는 컴퓨터의 메모리구조에서 사용하기 가장 알맞은 2진법과16진법이며 여러 프로그램들에서 효율적으로 사용되어 왔다. 그러나 어떠한 수의 체계든지 프로그램의 성질에 따라 예외적으로 드러나는 비효율적인 특성을 가지고 있다.

본 논문에서, 우리는 라우팅 관점에서의 2진법과 16진법의 예외적인 부분을 이용한 새로운 라우팅 어드레스의 표기법을 제안하며, 제안된 어드레스의 표기법을 이용한 라우팅 테이블과 라우팅 어드레스맵을 구성, 최소한의 메모리를 사용하여 센서네트워크 상에서 효율적인 Ad-Hoc 양방향 라우팅 방식을 구현하였다.

1. 서 론

기존의 라우팅 알고리즘들은 주변 노드의 주기적인 hello message에 포함된 기본적인 정보, 노드의 주소, 상태, 연결성을 이용한 라우팅 정보의 가공과 저장에만 관심이 있었다. 그 결과 정보의 삽입, 삭제, 유지를 위한 라우팅 테이블 관리에 많은 메모리와 리소스가 필요하다.

센서 네트워크 노드들은 매우 제한적인 리소스를 부여받고 있으므로 모든 주변 노드 정보의 저장이 불가능하다. 이러한 이유로 현재 나와있는 알고리즘들은 단 방향으로 구현되어 있는 경우가 많거나 브로드 케스팅의 방식을 이용하고 있다.

양방향 라우팅 프로토콜을 구현하기 위해 여러 메커니즘들이 제안 되었으나 실제적인 구현이 힘들거나 성능을 저하 시키는 원인이 되기도 한다. 성능저하의 원인은 라우팅 알고리즘은 간단하면서도 작은 리소스의 사용 그리고 신뢰성이 높아야 한다는 원칙에 반하기 때문이다.

본 논문에서는 요약문에서의 설명과 같이 라우팅의 관점에서 보았을 경우의 효과적인 2진법과 16진법의 어드레스 표기법, 이 표기법을 이용한 어드레스 맵의 구성, 그것을 이용하여 적은 메모리 리소스의 양방향 라우팅 알고리즘을 구현하였다.

2. 어드레스 표기법

현재 라우팅에 사용되는 데이터의 표기법은 컴퓨터의 메모리의 구조, 논리와 유사한 2진법과 그 이진법을 이용한 16진법이다. 이 방식은 매우 간단히 표현 가능하며 메모리의 사용량을 줄여준다.

그러나 본 논문의 관점 즉 센서네트워크 라우팅에서 보았을 경우 매우 비효율적인 방법이 될 수 있다. 그 이유는 서론에서 설명한 경우와 같이 센서 노드의 메모리는 매우 작으므로 주변 노드가 증가할수록 라우팅 테이블에 저장해야 할 정보가 증가 하기 때문이다.

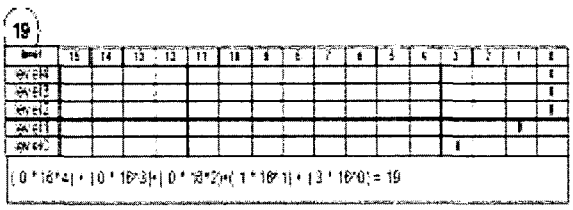


그림 [1] 어드레스 표기법

$$address = \sum_{level=0}^{number\ of\ level} (bitnumber * 16^{level}) \quad \text{....식(1)}$$

본 논문의 라우팅 알고리즘에서 어드레스는 그림[1]과 같이 표현 하며 식(1)에 의해 10진수로 변환 한다. 어드레스를 그림[1]과 같이 표현하는 것은 어드레스 맵을 구성하기 위해서 이며, 어드레스 맵의 업데이트가 필요할 경우에만 한시적으로 라우팅 테이블의 노드 어드레스를 이용하여 생성한다. 본 논문에서는 라우팅 테이블의 개수는 16개로 설정하였다.

3. 어드레스 맵

어드레스 맵은 다운링크를 위해서 만들며 위에서 만들어진 어드레스 표기법을 이용한다.

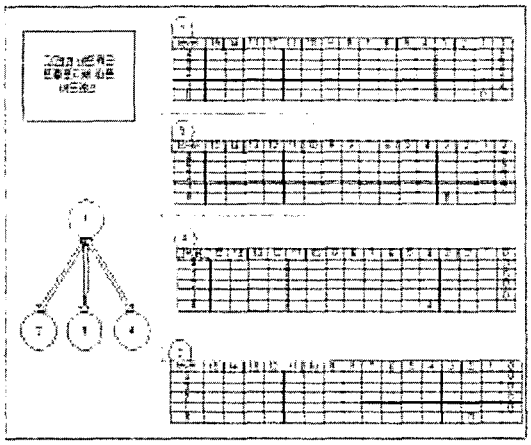


그림 [2] 각각의 노드의 어드레스 표기

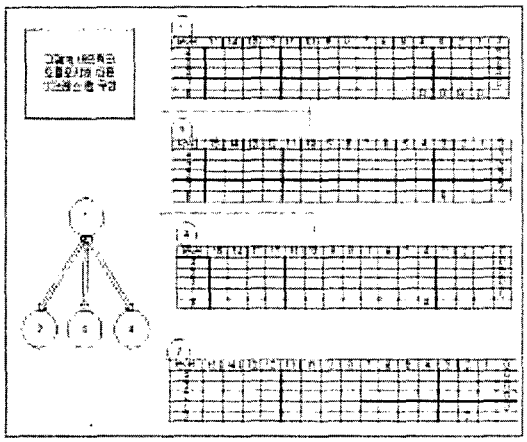


그림 [3] 네트워크 토플로지에 따른 어드레스 맵

어드레스 맵을 만들기 위해서는 3단계의 과정이 필요하다.

1. 그림[2]와[3]과 같이 각각의 노드는 토플로지를 구성해야 하며 페어런트를 가지고 있어야 한다.
2. 페어런트는 자신의 어드레스를 포함한 자식 노드의 어드레스 주소들을 OR 연산을 하여 새로운 맵을 만든다.
3. 페어런트는 다시 그 맵을 자신의 페어런트에게 전송하며 그 맵을 유지한다.

결과적으로 최상위의 노드는 자신의 모든 하위 노드의 어드레스를 중첩적으로 가질 수 있다.

어드레스 맵이 필요한 이유는 아래의 5. 다운링크에서 자세히 설명되며 최종적으로 자신의 페어런트에게 전송한 어드레스 맵을 제외한 나머지 들은 일시적으로 생성되는 것이다.

4. 어드레스 맵 관리

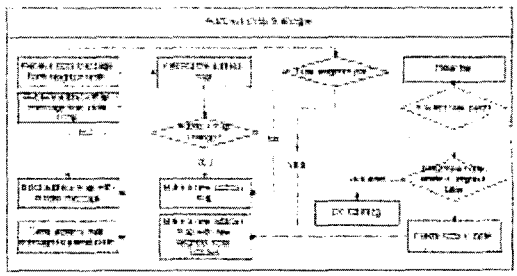


그림 [4] 어드레스 맵 관리 알고리즘

위의 그림[4]는 어드레스 맵 관리 알고리즘을 나타내며 가장 기본이 되는 동작의 파라미터는 어드레스 맵의 변화이다.

어드레스 맵 변화의 원인은 외부에서 들어온 데이터와 라우팅 테이블 안의 각각의 노드들에게 적용되는 타임 윈도우에 따른 라우팅 테이블의 업드이트 인터럽트가 발생하는 상황이다.

4-1. hello 메시지, 어드레스 맵 메시지

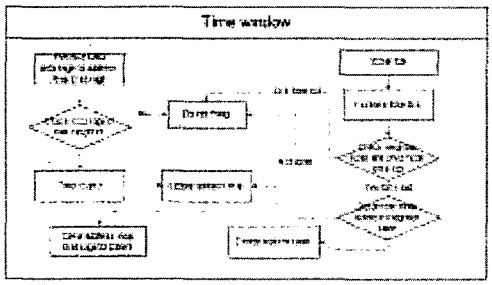
외부에서 들어오는 데이터 종류는 두가지로 나누어진다. 이 두 가지 메시지는 주기적으로 주변 노드로부터 들어오는 hello 메시지와 네트워크상의 하위 노드, 즉 자식 노드에서 전송된 어드레스 맵 메시지 이다. 주변 노드로부터 들어오는 hello 메시지 안에는 주변 노드의 어드레스 맵과 주변 노드와 관련된 여러 상태 정보들이 들어 있다. 이 상태 정보에 따른 라우팅 테이블 업데이트 정책(삽입, 제거) 의해 어드레스 맵이 새로 만들어지며 수신된 메시지로부터 추출된 어드레스 맵과 현재의 어드레스 맵이 다른 경우 수신된 어드레스 맵과 현재의 어드레스 맵을 OR연산 하여 새로운 맵을 구성, 페어런트로 어드레스 맵 메시지를 전송하거나 다음 hello 메시지에 포함하여 전송할 수 있다.

노드 자신의 주소로 어드레스 맵 메시지가 수신되면 노드는 자신이 가지고 있는 어드레스 맵과 비교한다. 변화가 없으면 아무런 동작을 하지 않으며 만약 다른 부분이 발견될 시에는 현재의 어드레스 맵과 도착한 어드레스 맵을 OR연산하여 자신의 맵을 업데이트하며 이 결과를 다시 페어런트에 전송한다.

이 과정을 연속적으로 상위의 노드로 이어지면 SINK 노드까지 이어지며 SINK노드는 자신의 하위 노드의 모든 어드레스 맵을 OR연산한 어드레스 맵을 가지게 된다.

4-2 Time Window

두 번째 어드레스 맵의 변화의 원인은 각각의 노드에 대한 타임 윈도우에 따른 어드레스 맵 업데이트의 유무와 새로운 페어런트의 선택 이다.



그림[5] Time Window 알고리즘

타임 윈도우의 구성은 우선 기준이 되는 타이머 인터럽트 타임 틱과 각각의 노드에 대한 타임 스탬프, 그리고 어드레스 맵의 업데이트 알고리즘이 있다.

1. 타임 틱은 일정한 시간마다 발생하며 타이머 인터럽트가 발생할 때 마다 1씩 증가한다.
2. 새로운 주변 노드가 합류 하거나 어드레스 맵에 관한 새로운 정보가 들어오면 그 노드에 관한 들어온 타임 틱을 저장한다.
3. 어드레스 맵의 업데이트는 타이머 인터럽트 때마다 모든 라우팅 테이블의 노드의 타임 스템프와 비교하여 현재의 타임 틱과 일정한 차이(if (current time_tick-stamped time tick>10)가 발생하면 어드레스 맵의 업데이트 유무를 판단
4. 어드레스 맵의 업데이트의 유무는 아래 식(2)와 같은 경우에 하며 어드레스 맵이 업데이트 되면 4-1의 동작을 한다.

$$\left(\frac{total\,send\,formchild - successrecive}{total\,sendmessage}\right)*100 < 50 \quad \text{.식(2)}$$

5. Down link

다운링크는 위에서 만들어진 어드레스 맵을 이용한다. 그림[6]을 보면 각각의 노드는 자신의 하위 노드들의 어드레스 맵과 자신의 어드레스 맵이 OR연산된 결과를 가진다.

1. 네이버 테이블이나 라우팅 테이블에 존재하는 노드 어드레스들과 목적노드의 어드레스를 비교하여 존재하면 그 노드로 데이터를 전송한다.
2. 어드레스 비교를 실패한·경우 목적 노드에 대한 어드래스 맵을 구성하여 아래의 그림[6]에 각각의 노드가 가진 어드레스맵과 AND연산하여 결과 값이 목적 어드레스 맵과 일치하게 나오면 그 노드로 전송한다.

결과적으로 그림[6]과 같이 6번 노드에게 데이터를 전송 할려면 1번 노드는 일치하는 어드레스가 없고 2번의 어드레스 멥과 비교하여 참이 나오므로 2번으로 전송하며 2번 노드의 라우팅 테이블에 6번 노드가 존재하므로 6번 노드에게 데이터를 전송한다)

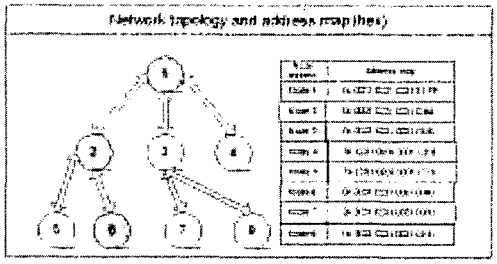


그림 [6] 다운링크

6. 결과

본 알고리즘의 테스트 플렛폼은 mica2의 콜론인 MAXFOR사의 TIP30을 30개 사용하였으면 기본적인 업링크 알고리즘은 TinyOS에서 제공하는 Surge를 사용하였다. 다운링크 Packet를 10개 전송하면 8번의 성공률을 보였으며 2번의 실패 원인은 TinyOS에서 제공하는 B-MAC의 재전송기능을 Disable하였기 때문이다.

$$h = log_n(\sum_{i=0}^{h} n^i) - 1 \quad \text{.....(식3)}$$

식(3))에서의 h는 총 노드 수에 대한 완전한 n진수의 트리 구조를 가질 때의 평균적인 hop count(depth)를 나타낸다

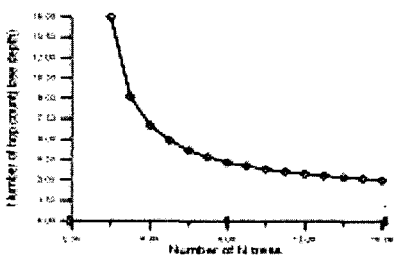


그림 [7] 토플로지 트리 구성에 따른 평균 hop count

그림[7]은 노드의 수가 65536개일 경우의 트리 구성에 따른 평균적인 depth와 hop count를 나타내며 트리 구성에 따른 브로드 케스트 패킷의 총수는 tree^(depth-1)이다. 본 논문의 알고리즘은 위의 그래프처럼 브로드 케스트와 비교하여 작은 수의 패킷과 메모리로 데이터 전송이 가능하다.

참고문헌

[1]B.Bellur and R.G ogier," Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks",Proceeding IEEE INFOCOM' March 1999.