무선 센서 네트워크에서 연결성 정보만을 이용하여 노드 위치를 추정하는 분산 알고리즘

권 오 흠[†] · 송 하 주[††] · 김 숙 연[†††]

요 약

본 논문에서는 무선 센서 네트워크 상에서 노드들 간의 연결성만을 이용하여 노드의 위치를 추정하는 분산 알고리즘을 제안한다. 본 논문에서 제안하는 알고리즘들은 기본적으로 공통 인접 노드의 개수를 이용하여 인접한 두 노드간의 거리를 추정하는 기법에 기반하고 있다. 제안된 알고리즘들의 위치 추정 성능을 모의실험을 통하여 분석 비교하였다.

키워드 : 무선 센서 네트워크, 연결성 기반 위치추적

Connectivity-Based Distributed Localization in Wireless Sensor Network

Oh-Heum Kwon[†] · Ha-Joo Song[††] · Sook-Yeon Kim[†††]

ABSTRACT

We present several distributed algorithms for localizing nodes of a wireless sensor network. Our algorithms determine locations of nodes based on the connectivity between nodes. The basic idea behind our algorithms is to estimate distances between nearby nodes by counting their common neighbors. We analyze the performance of our algorithms experimentally. The results of experiments show that our algorithms achieve performance improvements upon the existing algorithms

Key Words : Wireless Sensor Network, Connectivity-based Localization, Unit Disk Graph

1. 서 론

무선 센서 네트워크(wireless sensor network)는 군사적 혹은 상업적인 분야에서 그 응용 범위가 빠른 속도로 확대되고 있는 중요한 연구 분야의 하나이다. 무선 센서 네트워크의 기본 개념은 수백 개에 이르는 작고 값이 싼 센서 노드들을 일정한 지역에 배치한 후, 각 센서가 자신의 주변 환경을 감지하고, 감지된 데이터를 처리하며, 그 데이터를 중앙 시스템에 전송하는 것이다. 센서 노드들은 미리 정해진 위치에 계획적으로 배치되기 보다는 비행기를 통한 살포 등의 방법으로 배치되는 경우가 많기 때문에 그 위치가 무작위로 정해지는 경우가 많다. 반면 센서 노드에 의해서 감지된 데이터는 그 데이터가 감지된 위치가 어디인지를 알아야만 의미가 있다. 따라서 무작위로 살포된 센서 노드들의 위치를 알아내는 것은 무선 센서 네트워크에서 가장 기본적으로 해결되어야 하는 문제이다.

† 정 회 원 : 부경대학교 전자컴퓨터정보통신공학부 부교수
†† 정 회 원 : 부경대학교 전자컴퓨터정보통신공학부 전임강사
††† 정 회 원 : 한경대학교 컴퓨터공학과 조교수

논문접수 : 2005년 3월 18일, 심사완료 : 2005년 7월 4일

가장 간단한 해결책은 각 센서노드에게 GPS 수신기를 부착하는 것이다. 그러나 일반적으로 GPS 수신기는 부피가 크고 비싸며 또한 많은 전력을 소비하기 때문에 값이 싼 많은 수의 센서 노드들을 이용하는 무선 센서 네트워크에는 부적절한 경우가 많다. 그래서 GPS와 같은 비싼 추가 장치 없이 노드들의 위치를 알아내는 방법에 대한 연구가 꾸준하게 이루어져 왔다.

GPS 없이 센서 노드들의 위치를 알아내는 문제에 대해서 현재까지 알려져 있는 방법들은 대체로 다음과 같은 두 가지 유형으로 분류할 수 있다. 첫째는 물리적인 측정에 의존하는 방법이다 [2, 6-9]. 이 방법에서는 RSS(Received Signal Strength), TOA(Time of Arrival) 등과 같은 방법으로 근접해 있는 두 노드들 간의 거리를 측정하거나, 혹은 AOA(Angle of Arrival)과 같은 방법을 이용하여 인접한 노드로부터 시그널이 들어오는 각도를 측정한다. 이렇게 물리적인 방법으로 측정된 거리 혹은 각도에 대한 정보를 이용하여 각 노드의 위치를 추정하는 방법이다.

둘째는 노드 쌍들 간에 통신이 가능한가 가능하지 않은가에 대한 정보만을 이용하여 위치를 추정하는 방법이다[1, 4, 5]. 두 노드 간에 통신이 가능하다는 것은 두 노드의 거리가

각 노드의 최대 통신 거리 범위 이내라는 의미가 된다. 노드 쌍들 간의 통신 가능성에 대한 이러한 정보는 하나의 그래프로 표현된다. 일반적으로 이 그래프를 단위 디스크 그래프(unit disk graph)라고 부른다. 이 그래프의 구조를 분석하면 대략적인 노드의 위치를 추정하는 것이 가능하다. 이러한 기법들을 일반적으로 "연결성을 이용한 위치 추정기법"이라고 부른다.

연결성을 이용한 위치 추정은 물리적 측정에 기반 한 방법에 비하여 훨씬 적은 정보를 이용하므로 당연히 위치 추정의 정확도 측면에서 불리할 수밖에 없다. 그러나 위치 추정을 위하여 어떤 추가적인 하드웨어 장치도 필요하지 않으므로 비용의 측면에서 장점을 갖는다. 무선 센서 네트워크에서 위치 추정의 정확도에 대한 요구 사항은 응용분야에 따라서 다양하다고 할 수 있다. 정확한 위치 추정이 매우 중요한 응용 분야에서는 물리적 측정에 기반한 방법을 적용하는 것이 바람직하겠지만, 연결성에 기반한 추정 방법만으로도 원하는 정도의 정확성을 얻을 수 있는 경우 또한 많다. 따라서 연결성에 기반한 위치 추적 방법에 대한 연구는 그 활용도가 높다고 할 수 있다.

본 논문에서는 연결성을 이용하여 노드들의 위치를 추정하는 분산 알고리즘에 대해서 다룬다. 현재까지 알려져 있는 연결성을 이용한 위치 추정 분산 알고리즘들은 대부분 기본적으로 비슷한 구조를 가지고 있다. 우선 만약 어떤 노드도 자신의 위치를 모르는 상황이라면 노드들 간의 통신만으로 자신의 절대적 위치를 알아낸다는 것은 논리적으로 불가능하다. 따라서 위치를 결정할 기준점의 역할을 하는 존재가 필요하다. 그래서 보통의 노드들 외에 적어도 3개 이상의 앵커(anchor)라고 부르는 특수한 노드들이 있다고 가정한다. 앵커 노드들이 다른 노드들과 다른 점은 자신의 위치를 미리 알고 있다는 것이다. 앵커 노드들이 어떻게 자신의 위치를 미리 알 수 있는가 하는 문제는 별개의 문제이다. 수작업에 의해서 위치를 입력할 수도 있을 것이며, 혹은 앵커 노드에 한해서 GPS 수신기를 부착할 수도 있을 것이다.

위치를 추정하는 알고리즘은 대체적으로 다음과 같은 3단계를 거친다. 첫째, 각 노드들은 각각의 앵커 노드로부터 자신까지의 거리를 추정한다. 둘째, 각 노드들은 앵커 노드로부터 자신까지의 거리에 대한 추정치를 이용하여 자신의 위치를 추정한다. 셋째, 이렇게 추정된 위치를 자신의 주변에 있는 노드들의 추정 위치를 고려하여 반복적으로 수정해 나간다.

일반적으로 연결성을 이용한 위치 추정 알고리즘들 간에 가장 뚜렷하게 차이 나는 부분은 첫 번째 단계, 즉 각 노드가 앵커로부터 자신까지의 거리를 추정하는 부분이다. 대표적인 위치 추정 알고리즘의 하나인 DV-hop 알고리즘[4]이나 Amorphous 알고리즘[5]에서는 앵커로부터 자신까지의 홉 수(hop count)를 이용하여 거리를 추정한다. 먼저 각 노드들은 거리-벡터 라우팅(distance vector routing) 알고리즘과 유사한 분산 알고리즘을 이용하여 각각의 앵커로부터 자신까지의 홉 수를 알아낸다. 그런 다음 이 홉 카운트를 거리로 변환하기 위해서 "평균 홉 거리(average hop distance)"라는 개념을 사용한다. 평균 홉 거리는 말 그대로 한 홉 떨어진 노드들 간의 평균적인 거리를 의미한다. DV-hop 알고리즘에서는 모든 앵커 쌍들 간의 실제거리의 합을 모든 앵커 노드들 간의 홉 카운트로 나눈 값을 평균 홉 거리로 정의한다. 반면 Amorphous 알고리즘에서는 Kleinrock과 Silverster의 공식을 이용하여 평균 홉 거리를 평균 노드 조밀도(node density)의 함수로 표현한다 [3].

또 다른 연결성을 이용한 위치 추정 알고리즘인 GhoST 알고리즘에서는 단위 디스크 그래프의 구조를 고려하여 거리를 추정한다 [1]. 이 알고리즘에서는 trimmer와 stretch 라고 부르는 특별한 형태의 부 그래프(subgraph)가 존재하는지를 검사하고, 그 결과를 이용하여 노드들 간의 거리에 대한 상한 값 혹은 하한 값을 알아낸다. 이 상한 값 혹은 하한 값을 이용하면 알고리즘이 거리 추정에 있어서 최악의 경우에 빠지는 일을 어느 정도 방지할 수 있다고 알려져 있다.

일단 추정된 위치를 다시 반복적으로 수정함으로써 보다 정확한 위치를 추정해 나가는 몇몇 기법들이 알려져 있다 [10, 11]. 일반적으로 자신의 주변 노드들의 추정된 위치를 근거로 자신의 위치를 수정하고, 다시 그 수정된 위치가 주변 노드들의 위치 수정에 반복적으로 이용되는 구조를 가지고 있다. 이러한 방법의 문제점은 일반적으로 시간이 많이 걸리며, 언제 이 과정이 끝나게 될지 예측하기 어렵고, 심지어는 반드시 종료될지조차 불확실한 경우도 있다는 점이다. 또한 반복적 수정 기법들은 일반적으로 위치 추정 알고리즘 자체와는 독립적이다. 실제로 기존에 제안된 모든 반복적 위치 수정 기법이 본 논문에서 제시하는 위치 추정 알고리즘에 적용될 수 있다. 그런 의미에서 위치 추정 알고리즘의 성능에 대한 분석은 반복적 위치 수정 단계를 거치지 않은 상태로 이루어지는 것이 보다 공정하다고 할 수 있을 것이다. 따라서 본 논문에서는 더 이상 반복적 위치 수정 단계에 대해서는 언급하지 않을 것이며, 또한 모든 성능 분석은 반복적 위치 수정을 거치지 않은 상태로 이루어진다.

본 논문에서는 연결성을 이용해서 노드 위치를 추정하는 몇 가지 분산 알고리즘을 제안한다. 본 논문이 제안하는 알고리즘들의 기본 아이디어는 어떤 노드 쌍에 대해서 두 노드에 공통으로 인접한 노드의 개수를 셈으로써 두 노드간의 거리를 대략적으로 추정할 수 있다는 것이다. 일반적으로 두 노드의 거리가 가까울수록 두 노드에 공통으로 인접한 노드의 수는 많아질 것이다. 각 노드들이 통신을 할 수 있는 거리의 범위가 r로 일정하다고 한다면, 각 노드와 그 노드에 인접한 노드들은 그 노드를 중심으로 하는 반지름이 r인 원의 내부에 있다. 어떤 두 노드와 공통으로 인접한 노드들은 두 노드를 각각의 중심으로 하는 반지름이 r인 두 원의 교차 영역 내에 위치할 것이다. 노드들이 평면상에 균일하게 분포하고 있다고 가정한다면 두 노드의 인접한 노드의 개수와 공통 인접 노드의 개수를 이용하여 교차 영역의 면적을 추정하는 것이 가능하고, 교차 영역의 면적은 두 노드 사이의 거리에 의해서 정해지므로 결과적으로 두 노드 사이의 거리에 대한 추정이 가능하다.

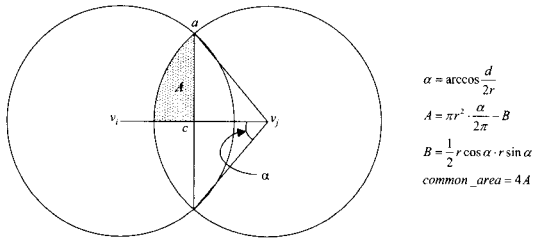
이러한 노드간 거리 추정 기법을 이용하여 본 논문에서는 먼저 세 가지 위치 추정 알고리즘을 제안한다. 제안된 세 가지 알고리즘의 이름은 각각 Trigonometric, Linear_Progress, 그리고 Nonlinear_Progress이며 기본적으로 동일한 구조를 가지고 있다. 모의실험의 결과에 따르면 세 알고리즘은 모두 기존의 알고리즘들 보다 뛰어난 위치 추정 성능을 보여주었으며, 그 중에서도 Nonlinear-Progress 알고리즘이 가장 우수하였다.

세 알고리즘은 우수한 위치 추정 성능을 보여주지만 기존의 알고리즘들에 비해서 많은 메시지 교환이 필요하다는 단점이 있다. 이 알고리즘들이 보여주는 위치 추정의 정확성을 유지하면서 필요한 메시지의 수를 줄이기 위해서 네 번째 알고리즘을 고안하였다. 네 번째 알고리즘은 Simplified 알고리즘이라고 부르며, 노드 조밀도가 적정한 선을 유지할 경우 Nonlinear-Progress 알고리즘에 비견될만한 위치 추정 정확성을 보여주면서 메시지의 개수는 DV-hop 등의 기존의 알고리즘과 동일하다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 본 논문에서 제시한 알고리즘들이 공통적으로 사용하는 인접한 노드들 간의 거리를 공통 인접 노드의 개수를 이용하여 추정하는 방법에 대해서 설명한다. 3장에서는 2장의 방법을 사용하는 세 가지 위치 추정 알고리즘을 기술하고, 4장에서는 Simplified 알고리즘에 대해서 기술한다. 5장에서는 우리가 실시한 모의실험의 결과를 기술하고, 6장에서는 결론을 맺는다.

2. 공통 인접 노드의 개수를 이용한 거리 추정

우선 본 논문에서 사용될 몇 가지 표기법들을 소개하겠다. 무선 센서 네트워크는 n개의 센서 노드로 구성되며 각각의 센서의 통신 가능한 거리 범위를 r이라고 한다. 두 노드 v_i와 v_j 간의 실제 거리를 d_{ij}라고 표시한다. 단위 디스크 그래프 $G=(V,E)$는 각각의 센서 노드들을 정점(vertex)으로 하고, 두 노드간의 실제 거리가 r 이내이면 에지(edge)로 연결되는 그래프이다. 그래프 G에서 두 노드 간에 에지가 있다는 것은 해당하는 센서들 간에 직접적인 양방향(bidirectional) 통신이 가능하다는 것을 의미한다. 두 노드간의 홉 수(hop count)는 두 노드를 연결하는 경로(path) 상의 에지 개수의 최소값을 의미한다. 두 노드 v_i와 v_j간의 홉 수를 h_{ij}라고 표시한다. 홉 카운트가 1인 노드들을 서로 간에 이웃(neighbor) 노드라고 부른다. 임의의 노드 v_i에 대해서 이웃 노드들의 집합을 N_i라고 표시한다.



(그림 1) 교차영역의 면적

서로 인접하거나 혹은 두 홉 떨어진 임의의 노드 쌍 v_i와 v_j를 생각해 보자. 즉 $h_{ij} \le 2$이다. 이 경우 두 노드는 일반적으로 공통 이웃 노드를 가지며, 두 노드 간의 실제 거리 d_{ij}는 최대 $2r$이다. 두 노드의 공통 이웃 노드들은 두 노드를 각각의 중심으로 하는 반지름 r인 두 원의 교차 영역 내에 위치하게 된다. 평면상에서 노드들의 분포가 균일하다는 가정 하에서 노드 v_i의 이웃 노드의 개수와 공통 이웃 노드의 개수의 비율 ρ_1을 노드 v_i를 중심으로 하는 반지름 r인 원의 면적과 두 원의 교차 영역의 면적의 비 ρ_2와 등치함으로써 두 노드간의 거리를 추정하려 한다.

$$\rho_1 = \frac{|N_i \cap N_j|}{|N_i|} \quad \text{그리고} \quad \rho_2 = \frac{common\_area}{\pi r^2}$$

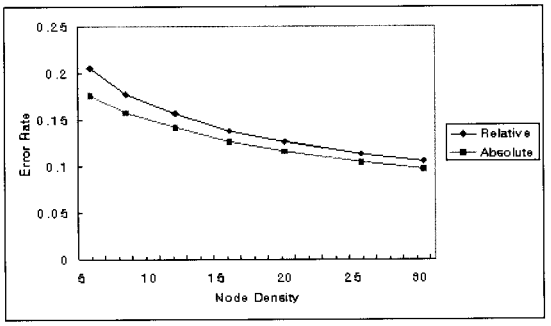
두 원의 교차영역의 면적은 (그림 1)에서 기술한 바와 같이 두 노드간의 거리 d_{ij}의 함수로 표현할 수 있다.

두 비율을 등치함으로써 얻어지는 식 $\rho_1 = \rho_2$의 정확한 해를 수학적으로 구하는 것은 어렵지만, ρ_2가 거리 d_{ij}에 관한 단조 감소(decreasing) 함수이므로 거리 d_{ij}에 대하여 이진검색(binary search)을 수행함으로써 우리가 원하는 정도의 정확도를 가진 근사해를 구하는 것은 쉽다. 이진 검색의 구간은 $h_{ij}=1$인 경우에는 $[0,r]$이고 $h_{ij}=2$인 경우에는 $[r,2r]$이다. 이진 검색에서 대략 5~6번 정도의 반복이면 이 경우에 필요한 정확성을 얻을 수 있다.

이 방법으로 거리를 제대로 추정할 수 없는 경우가 있다. 예를 들어서 노드 v_i가 노드들이 배치된 영역의 가장자리에 있고, 영역의 내부에 있는 노드 v_j까지의 거리를 추정하려 한다고 해보자. 이 경우 노드 v_i의 이웃 노드들 중의 대부분이 노드 v_j와의 공통 이웃일수가 있다. 따라서 만약 $\rho_1 = \frac{|N_i \cap N_j|}{|N_i|}$이라고 한다면 두 노드간의 거리는 실제보다 매우 작게 추정될 것이다. 이런 경우를 방지하기 위해서 본 논문에서는 실제로는 $\rho_1 = \frac{|N_i \cap N_j|}{\max\{|N_i|,|N_j|\}}$으로 정의한다. 이것은 거리를 추정하려는 노드가 자신의 이웃 노드의 개수만이 아니라 상대편의 이웃 노드의 개수도 알아야만 한다는 것을 의미한다. 어떤 노드가 자신과 상대편의 이웃 노드의 개수, 그리고 두 노드간의 공통 노드의 개수를 알아내는 방법은 제3장에서 설명한다.

이 거리 추정 방법의 정확성은 일반적으로 노드의 분포가 균일하면서 밀도가 높을수록 향상될 것이다. (그림 2)에서는 모의실험을 통해서 이 거리 추정 방법의 정확도를 분석한 결과를 보여준다. (그림 2)에서 상대적 오차는 거리 추정 오차를 실제 거리로 나눈 값이며, 정규화된 오차는 거리 추정 오차를 통신 범위 r로 나눈 값이다. (그림 2)에 제시된 실험 결과는 정사각형 모양의 영역 내에 임의로 분포한 250개의 노드를 샘플로 하여 분석된 결과이다. 노드의 밀도는 반지름이 r인 원 내에 존재하는 평균 노드수를 의미하며, 실험에서는 통신 범위 r을 변경함으로써 조절되었다. 그림에도 보듯이 상대적 오차는 노드 밀도가 7이상일 때 대략 10%에서 20%정도 사이에 분포한다. 거리 추정의 오차는 샘플에서의 노드의 개수가 증가되면 향상되지만 큰 차이는 나지 않는다. 노드 밀도

가 6 혹은 그 이하가 되면 많은 수의 고립된 노드들이 존재하게 되고, 네트워크가 여러 개의 연결 성분(connected component)로 쪼개지게 될 확률이 높으므로, 무작위로 노드들을 배치하는 경우에는 일반적으로 노드 밀도가 적어도 6 혹은 그 이상이 되어야 한다.



(그림 2) 거리 추정의 정확도

이 방법의 거리 추정 정확도는 TOA와 같은 물리적인 측정 방법과 비교하면 나쁜 편이다. 이러한 거리 추정에 있어서의 부정확함은 거리 추정 알고리즘을 설계할 때 충분히 고려되어야만 하는 중요한 요소이다. 거리 추정의 부정확함을 고려하지 않은 위치 추정 알고리즘은 성공적이기 어렵다. 따라서 물리적 거리 측정 방법에서 사용되는 위치추정 알고리즘을 직접적으로 적용하는 데는 어려움이 있다. 이 거리 추정 방법이 가지는 한 가지 장점은 서로 인접하지 않고 두 홉 떨어진 노드들 간의 거리를 추정할 수 있다는 점이다. 이 점은 위치 추정 알고리즘을 설계하는데 있어서 기존의 물리적인 거리 측정 방법과는 다른 기법을 적용할 여지를 제공해준다.

3. 세 가지 위치 추정 알고리즘

본 장에서는 2장에서 기술한 노드들 간의 거리 추정방법을 이용하여 노드들의 위치를 추정하는 세 가지 알고리즘을 기술한다. 세 알고리즘들은 기본적으로 동일한 구조를 가지고 있다. 먼저 적어도 3개 이상의 앵커 노드가 존재하며, 앵커 노드들은 자신의 위치를 미리 알고 있다고 가정한다. 자신의 이치를 미리 알고 있다는 점을 제외하고는 앵커 노드는 모든 면에서 보통의 노드와 동일하다. 알고리즘은 세 단계로 이루어진다. 첫째, 각 노드는 먼저 자신의 이웃 노드들, 두 홉 떨어진 노드들, 그리고 그들과의 공통 노드의 개수를 알아낸다. 둘째, 각 노드는 각각의 앵커 노드로부터 자신까지의 거리를 추정한다. 셋째, 각각의 노드는 앵커노드로부터의 추정된 거리를 이용하여 자신의 위치를 추정한다. 알고리즘에서 이루어지는 모든 통신은 한 노드가 자신의 이웃노드들에게 메시지를 방송(broadcast)하는 것으로 이루어진다.

3.1 이웃 노드, 두 홉 떨어진 노드, 그리고 그들과의 공통 이웃 찾기

각각의 노드는 먼저 자신의 존재를 주변 노드들에게 알리는 메시지를 방송한다. 이 메시지에는 자신의 id가 포함된다. 이 메시지를 수신한 노드는 그 메시지에 자신의 id를 추가한 후 그 메시지를 자신의 이웃들에게 다시 방송한다. 한 노드가 자신의 존재를 알리기 위해서 방송하는 메시지는 이렇게 자신의 이웃 노드 각각에 의해서 한번씩 재방송됨으로써, 결국 모든 노드들이 자신의 이웃들과 또한 자신과 두 홉 떨어진 노드의 존재를 알게 된다.

어떤 노드의 존재를 알리는 메시지를 중간에 다른 노드를 경유하지 않고 직접 받았다면 그 노드는 이웃 노드이고, 그렇지 않다면 두 홉 떨어진 노드이다. 또한 어떤 노드의 존재를 알리는 메시지를 중간 노드를 경유해서 받았다면 그 중간 노드는 바로 자신과 메시지를 보낸 노드의 공통 이웃이다. 따라서 경유된 메시지의 개수를 셈으로써 공통 이웃 노드의 개수를 알 수 있다.

이제 이 과정에 방송된 메시지의 개수를 고려해 보자. 각각의 노드는 자신의 존재를 알리는 메시지를 한번 방송하고, 그 메시지는 자신의 모든 이웃 노드들에 의해서 한번씩 재방송 된다. 따라서 이 노드의 존재를 알리는 메시지의 총 개수는 이 노드의 이웃 노드의 개수 더하기 1이다. 단위 디스크 그래프의 관점에서 보면 해당하는 노드의 분지수(degree) 더하기 1이다. 이 값을 모든 노드들에 대해서 더하면, 그래프에서 모든 노드의 분지수의 합은 에지 개수의 2배가 되므로

$$\sum_{v_i \in V}(|N_i|+1) = n + \sum_{v_i \in V}|N_i| = n+2m$$

이다. 즉, 단위 디스크 그래프의 각 정점 당 하나의 메시지, 그리고 각 에지 당 두개의 메시지가 방송되는 셈이므로 메시지의 총 개수는 $n+2m$개가 된다. 여기서 n은 노드의 개수이고 m은 단위 디스크 그래프의 에지의 개수이다.

실제로는 각 노드가 이웃 노드들로부터 받는 메시지들을 한번에 하나씩 재방송하지 않고, 여러 개를 모아서 하나의 메시지로 합친 후 한번에 재방송함으로써 메시지의 개수를 줄일 수 있다. 물론 이 경우 각각의 메시지의 길이는 증가하겠지만 메시지 방송에 들어가는 다른 오버헤드들을 감안한다면 이렇게 하는 편이 더 효율적일 것이다. 이 경우 메시지의 개수는 하나의 메시지의 길이를 얼마로 제한하느냐에 따라 달라질 것이다.

3.2 추정된 거리로부터 노드 위치의 추정

본 절에서는 알고리즘의 단계 3인 앵커로부터의 거리를 이용하여 자신의 위치를 추정하는 방법을 먼저 설명한다. 본 논문의 알고리즘은 이 단계에서 대부분의 위치 추정 알고리즘들이 공통적으로 사용하는 방법인 멀티레터레이션(multilateration)을 이용한다. 예를 들어서 노드 v가 k개의 앵커 노드의 위치 (x_i,y_i), $1 \le i \le k$, 를 알고 있고, 각각의 앵커로부터의 거리 d_i, $1 \le i \le k$,를 추정했다고 하자. 노드 v가 자신의 위치를 (x,y)라고 가정한다면 자신과 각 앵커와의 거리는 $\sqrt{(x_i-x)^2+(y_i-y)^2}$가 된다. 이렇게 계산된 거리와 추정된

거리 d_i와의 차이의 제곱을 모든 앵커에 대해서 합한 값, 즉

$$\sum_{i=1}^{k}(\sqrt{(x_i-x)^2+(y_i-y)^2}-d_i)^2$$

를 최소로 만드는 위치 (x,y)를 찾음으로써 자신의 위치를 결정하는 방법을 멀티레터레이션이라고 부른다. 보다 상세한 계산 과정은 문헌 [5]를 참조하라.

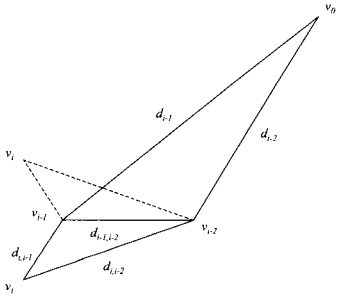
3.3 앵커로부터의 거리 추정

본 절에서는 각각의 노드가 특정한 앵커 v_0로부터의 거리를 어떻게 추정하는지 설명한다. 이하에서는 앵커 v_0로부터의 거리와 홉 수 d_{0i}와 h_{0i}를 줄여서 그냥 d_i와 h_i로 표시할 것이다. 앵커로부터의 거리를 추정하는 알고리즘의 기본 골격은 DV-hop알고리즘이나 Amorphous 알고리즘의 경우와 마찬가지로 잘 알려진 거리-벡터 라우팅 알고리즘과 유사하다.

알고리즘은 앵커 노드 v_0가 자신의 id와 위치를 담은 메시지를 이웃 노드들에게 방송함으로써 시작된다. 각각의 노드는 알고리즘이 진행되는 과정을 통해서 앵커 v_0로부터 자신까지의 홉 수와 거리를 알아내게 된다. 모든 메시지는 메시지를 보내는 노드의 홉 수와 거리를 포함하고 있다.

임의의 노드 v_i가 이웃 노드 v_j로부터 메시지를 받았다고 하자. 그러면 먼저 노드 v_i는 메시지에 포함된 노드 v_j의 홉 수 h_j를 검사한다. 만약 노드 v_i의 홉 수 h_i가 아직 미정이거나 혹은 $h_i > h_j+1$이면 노드 v_i는 이 메시지를 받아들이고, 자신의 홉 수를 h_j+1으로 설정 혹은 수정한다. 노드 v_i는 받아들인 메시지의 내용을 이용하여 자신의 거리 d_i를 추정하고, 이렇게 추정된 거리를 새로운 메시지에 담아서 자신의 이웃들에게 방송한다. 이때 노드 v_i를 이 새로운 메시지의 "발송 노드"라고 부르고, 노드 v_j를 이 메시지의 "선행 노드"라고 부른다. 노드 v_i가 방송하는 새로운 메시지에는 다음과 같은 정보가 포함된다.

(1) 앵커 노드 v_0의 id와 위치
(2) 발송 노드 v_i의 id, 이웃 노드의 개수 $|N_i|$, 홉 카운트 h_i, 그리고 추정된 거리 d_i
(3) 선행 노드 v_j의 id, 이웃 노드의 개수 $|N_j|$, 홉 카운트 h_j, 그리고 추정된 거리 d_j
(4) 노드 v_i와 노드 v_j간의 추정된 거리 d_{ij}



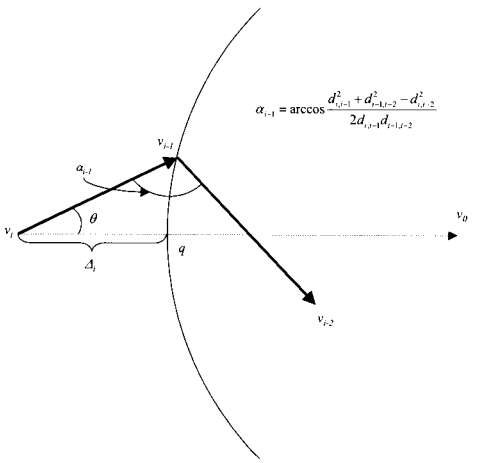
(그림 3) 노드 v_i의 두 가지 위치

앵커 v_0에 의해서 방송된 최초의 메시지는 물론 선행 노드가 존재하지 않으며, 따라서 항목 (3)과 (4)는 비워두면 된다. 일반적으로 항목 (1)과 (3)은 선행 노드가 발송 노드에게 보낸 이전 메시지에 담겨 있는 내용이므로 그 메시지로부터 복사하면 된다. 항목 (4)는 발송 노드 v_i에 의해서 계산될 수 있는 값이다. (2)번 항목인 노드 v_i의 거리 d_i를 어떻게 추정하는가가 이 절의 나머지 부분의 주제이다.

노드 v_i가 자신의 이웃으로부터 메시지를 받았다고 가정해 보자. 먼저 $h_i \le 2$인 경우라면, 노드 v_i는 2장에서 기술한 방법으로 앵커 v_0로부터 자신까지의 거리를 추정할 수 있다. 이제 $h_i > 2$인 일반적인 경우를 고려해 보자. 노드 v_{i-1}과 v_{i-2}를 각각 노드 v_i가 방금 받은 메시지의 발송 노드와 선행 노드라고 가정해보자. 노드 v_i는 다음과 같은 다섯 개의 거리에 대한 추정치를 가지고 있는 상황이다.

$$d_{i-1},\ d_{i-2},\ d_{i,i-1},\ d_{i-1,i-2},\ d_{i,i-2}$$

이 중 d_{i-1}, d_{i-2}, 그리고 $d_{i-1,i-2}$는 받은 메시지에 포함된 정보이고, $d_{i,i-1}$과 $d_{i,i-2}$는 노드 v_i에 의해서 추정 가능한 거리이다. 이하에서는 이 다섯 개의 거리 추정치를 이용하여 거리 d_i를 추정하는 세 가지 서로 다른 방법을 기술한다.



(그림 4) 세 노드의 위치와 꺾인 정도

3.3 Trigonometric 알고리즘

(그림 3)에서 보이듯이 노드 v_i, v_{i-1}, 그리고 v_{i-2}는 일반적으로 하나의 삼각형을 구성하며, 노드 v_{i-1}, v_{i-2}, 그리고 v_0 역시 하나의 삼각형을 구성한다. 다섯 개의 거리는 각각 두 삼각형의 변의 길이가 된다. 이때 이 다섯 개의 길이에 관한 요구 사항을 만족하는 노드 v_i의 위치는 일반적으로 (그림 3)에서와 같이 두개이다. 노드 v_{i-1}과 v_{i-2}를 연결하는 직선을 기준으로 하나가 다른 하나의 대칭이 되는 위치이다.

두 개의 후보 위치 중에서 하나를 선택하기 위해서는 관련된 네 노드 이외의 다른 노드에 관한 정보를 참조할 수밖에 없다. 만약 매우 정확한 거리 측정이 요구되고, 또 그에 걸맞는 정확한 물리적 거리 측정법이 사용되는 경우라면 이 문제

를 올바로 해결하는 것이 매우 중요하다고 할 수 있을 것이다. 그러나 본 논문에서와 같이 비교적 부정확한 거리 추정법을 사용하는 경우에 이 문제가 복잡한 방법을 동원하여 해결할 만한 가치가 있는지는 다소 의문스럽다.

이런 점을 분명히 하기 위해서 우리는 두 가지 방법을 모의실험을 통하여 비교해 보았다. 첫 번째 방법은 두 후보 위치 중에서 무조건 앵커 v_0로부터 멀리 떨어져 있는 점을 선택하는 방법이다. 두 번째 방법은 항상 올바른 선택을 하는 방법이다. 여기서 올바른 선택을 한다는 의미는 두 노드의 실제 위치를 이용하여 결정한다는 의미이다. 구체적으로 말하면 네 노드의 실제 좌표를 이용하여 노드 v_i가 대칭선을 기준으로 노드 v_0와 실제로 같은 편에 있는지 아니면 다른 편에 있는지를 결정하였다.

두 가지 방법을 모의실험을 통해서 비교한 결과 놀랍게도 항상 올바른 선택을 하는 방법이 오히려 평균적으로 나쁜 결과를 보여주었다. 이 실험 결과가 의미하는 바는 대체로 두 가지로 요약될 수 있다. 우선 노드 v_0를 출발하여 노드 v_{i-2}와 v_{i-1}을 지나서 v_i에 도달하는 경로는 노드 v_0로부터 v_i로의 최소 홉 경로이므로, 전반적으로 노드 v_i가 대칭선의 반대편에 존재할 확률이 높다는 의미이며, 둘째로는 노드들 간의 거리 추정의 오차가 잘못된 후보를 선택함으로써 발생하는 오차를 압도한다는 의미이다.

이런 실험 결과에 근거하여 본 논문에서는 단순히 앵커 v_0로부터 더 멀리 떨어져 있는 후보를 선택하는 방법을 채택하였으며, 그 방법을 Trigonometric 알고리즘이라고 부른다. 대각선의 길이 $\overline{v_iv_0}$는 노드 v_{i-1}의 좌표를 $(0,0)$라고 가정하고, (그림 3)에서와 같이 예각 $\alpha = \angle v_0v_{i-1}v_{i-2}$와 $\beta = \angle v_iv_{i-1}v_{i-2}$라고 정의하였을 때, 두 노드 v_0의 좌표와 v_i의 좌표가 다음과 같이 표현되므로 쉽게 계산할 수 있다.

$$\alpha = arccos\left(\frac{d_{i-1}^2 + d_{i-1,i-2}^2 - d_{i-2}^2}{2d_{i-1}d_{i-1,i-2}}\right)$$

$$\beta = arccos\left(\frac{d_{i,i-1}^2 + d_{i-1,i-2}^2 - d_{i,i-2}^2}{2d_{i,i-1}d_{i-1,i-2}}\right)$$

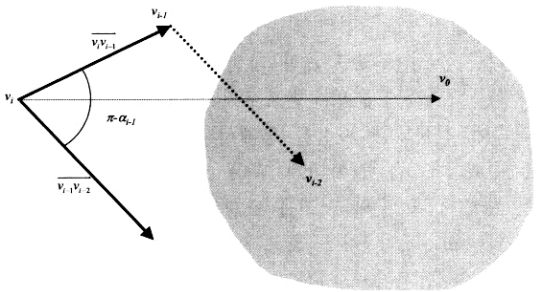
$$v_0 = (d_{i-1}\cos\alpha, d_{i-1}\sin\alpha), \quad v_i = (d_{i,i-1}\cos\beta, -d_{i,i-1}\sin\beta)$$

3.4 Linear_Progess 알고리즘

이 방법에서는 노드 v_i가 직접적으로 거리 d_i를 추정하는 대신에 두 거리의 차이 $\Delta_i = d_i - d_{i-1}$을 먼저 추정한다. 일단 Δ_i가 추정되면 거리 d_i는 $d_{i-1} + \Delta_i$로 추정된다. 이 방법에서 사용되는 기본적인 아이디어는 다음과 같은 것이다. 가령 아주 구불구불한 길이 있다고 가정해보자. 그런 길을 걸어간다면 그 길의 한 구간을 걸어감으로써 실제로 목표점에 가까워지는 정도는 걸어간 거리에 비해서 상대적으로 아주 작을 것이다. 반면 비교적 똑 바로 나있는 길을 걸어간다면 거의 우리가 걸어간 거리만큼 실제로 목표점과의 거리가 가까워질 것이다.

노드 v_i에서 노드 v_{i-1}을 거쳐서 앵커 v_0를 향해 걸어간다고 가정해 보면, 노드 v_i에서 노드 v_{i-1}까지의 거리가 우리가 걸어간 거리에 해당하고, Δ_i는 그렇게 걸어감으로써 실제로 목표점인 v_0에 얼마나 가까워 졌는가를 표현한다. 여기서 우리는 길이 구불구불한 정도를 추정하여 그로부터 Δ_i를 추정하려 하는 것이다. (그림 4)에서 보는 바와 같이 추정된 거리 $d_{i,i-1}$, $d_{i-1,i-2}$, 그리고 $d_{i,i-2}$를 이용하면 예각 $\alpha_{i-1} = \angle v_iv_{i-1}v_{i-2}$를 계산할 수 있다. 각 α_{i-1}은 노드 v_i에서 v_0까지의 경로가 노드 v_{i-1}이 있는 지점에서 얼마나 휘었는지를 표현한다. 우리는 여기서 Δ_i를 각 α_{i-1}와 추정된 거리 $d_{i,i-1}$을 이용하여 다음과 같이 근사하려고 한다.

$$\Delta_i = f(\alpha_{i-1}) \cdot d_{i,i-1} \qquad \text{(식 3.1)}$$



(그림 5) 앵커의 위치에 대한 가정

여기서 f는 $f(0) = 0$과 $f(\pi) = 1$을 만족하는 단조 증가 함수이다.

Δ_i를 이렇게 근사하는 것에는 몇 가지 단순화를 위한 가정이 깔려있다. 첫째는 여기서는 Δ_i가 연속된 세 노드 v_i, v_{i-1}, 그리고 v_{i-2} 간의 국부적인 관계의 함수로 표현되었으나 실제로는 그렇지 않다. 부분적으로 떼어서 보면 휜 정도가 아주 미미하나 전체적으로 보면 아주 심하게 휘는 경로도 있을 수 있다. 가령 경로상의 모든 노드에서 항상 동일한 방향으로 조금씩 휘다보면 전체적으로는 소용돌이 모양의 심하게 휜 경로가 만들어질 수도 있다. 둘째로는 $f(0) = 0$이라고 정의하고 있는 점이다. 이것은 노드 v_i에서 노드 v_{i-1}으로 움직임으로써 실제로는 v_0로부터 오히려 멀어지게 되는 경우, 즉 노드 v_0가 노드 v_i의 뒤쪽에 있는 경우를 고려하지 않는다는 의미이다.

이런 가정들에도 불구하고 (식 3.1)은 실제로는 의미가 있다. 그 이유는 위의 식을 적용하려는 경로가 최소 홉 경로이기 때문이다. 노드의 밀도가 일정 수준 이상이 되면 위 단락에서 언급한 그런 극단적인 경우가 최소 홉 경로에서 발생할 확률을 매우 낮아지기 때문이다. 또한 (식 3.1)보다 더 정확한 식을 수립한다고 하더라도 인접 노드들 간의 거리 추정에 포함되어 있는 오차가 그 의미를 반감시키는 측면이 있다고 할 수 있다.

함수 f를 어떻게 정의하느냐가 남은 문제이다. 여러 가지 방법이 있을 수 있으나 그중 가장 간단한 방법은 함수 f를 선형 함수라고 가정하는 것이다. 여기서는 간단하게 $f(\theta) = \frac{\theta}{\pi}$라고 가정하고, 그 알고리즘을 Linear_Progress라고

부른다.

3.5 Nonlinear_Progress 알고리즘

본 절에서는 기술하는 알고리즘은 3.4절에서 기술한 Linear_Progress 알고리즘과 동일하며 다만 함수 $f(\theta)$만을 다르게 정의한다. 다시 (그림 4)를 보자. 노드 v_i와 앵커 v_0을 연결하는 직선과 앵커 v_0을 중심으로 하고 .

반지름이 d_{i-1}인 원의 교차점을 q라고 부르자. 그러면 실제로 $\Delta_i = |\overline{v_i q}|$이다. 만약 앵커 v_0가 노드 v_i로부터 충분히 멀리 떨어져 있다면 $\Delta_i = |\overline{v_i q}| \cong d_{i,i-1}\cos\theta_i$이다. 여기서 각 θ_i는 예각 $\angle v_{i-1}v_iv_0$이다. 만약 우리가 각 θ_i의 값에 관하여 적절한 확률적인 분포를 정의할 수 있다면 Δ_i의 기대값을 다음과 같은 식으로 표현할 수 있을 것이다.

$$E[\Delta_i] = \int_0^{\pi} prob(\theta_i = \theta) \cdot d_{i,i-1}\cos\theta d\theta \qquad \text{(식 3.2)}$$

여기서 $prob(\theta_i = \theta)$는 θ_i가 θ, $0 \le \theta \le \pi$,가 될 확률이다. 확률 $prob(\theta_i = \theta)$를 실제로 분석하는 것은 매우 흥미로우나 다소 어려운 문제로 보인다. 본 논문에서는 $prob(\theta_i = \theta)$를 분석하는 대신 다음과 같이 간단하게 가정하고자 한다. (그림 5)를 보자. 우리가 알고 있는 것은 노드 v_i에서 v_0까지의 경로가 먼저 v_i에서 v_{i-1}방향으로 꺾이고 그 다음 v_{i-1}에서 v_{i-2}방향으로 꺾인다는 것이다. 이 두 번의 꺾임으로부터 우리는 노드 v_0가 이 두 번 꺾이는 방향에 의해서 정의되는 각도 범위 이내에 있을 확률이 높다고 가정한다. 즉, (그림 5)에서와 같이 앵커 v_0는 점 v_i를 기점으로 하는 두 벡터 $\overrightarrow{v_iv_{i-1}}$과 $\overrightarrow{v_{i-1}v_{i-2}}$에 의해서 정의되는, 그림에서 회색으로 표현되어진 영역 내의 어딘가에 있다고 가정한다. 다르게 말하면 θ_i가 0과 $\pi - \alpha_{i-1}$의 범위 내에 있을 확률을 1이라고 가정한다는 의미이다. 또한 그 범위 내에서는 모두 동일한 확률로 분포되어 있다고 가정한다.

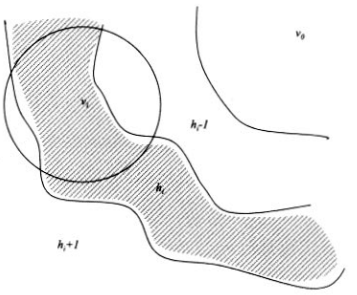
$$prob(\theta_i = \theta) = \begin{cases} 0 & \text{if } \theta > \pi - \alpha_{i-1}, \\ \dfrac{1}{\pi - \alpha_{i-1}} & \text{if } 0 \le \theta \le \pi - \alpha_{i-1} \end{cases} \qquad \text{(식 3.3)}$$

이제 (식 3.3)을 (식 3.2)에 대입함으로써 다음과 같이 Δ_i에 대한 기대값을 얻을 수 있다.

$$E[\Delta_i] \cong \frac{1}{\pi - \alpha_{i-1}} \int_0^{\pi - \alpha_{i-1}} d_{i,i-1}\cos\theta d\theta = \frac{\sin\alpha_{i-1}}{\pi - \alpha_{i-1}} \cdot d_{i,i-1}$$

이제 $f(\alpha_{i-1}) = \dfrac{\sin\alpha_{i-1}}{\pi - \alpha_{i-1}}$이라고 정의하고, 이 알고리즘을 Nonlinear_Progress라고 부른다.

지금까지 설명한 세 알고리즘에서 거리 d_i를 추정하는 방법을 요약하면 다음과 같다.

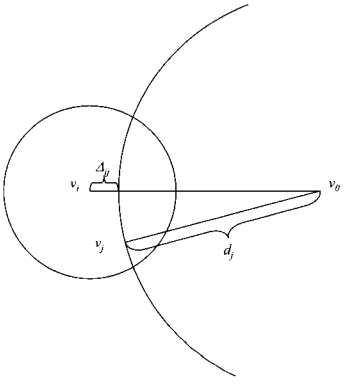


(그림 6) 동일 홉 수를 가진 노드들의 위치

$$d_i = \begin{cases} \textit{the method in Section2} & \text{if } h_i \le 2, \\ \textit{the length of diagonal} & \text{if } h_i > 2 \textit{ in Trigonometric} \\ d_{i-1} + \dfrac{\alpha_{i-1}}{\pi} d_{i,i-1} & \text{if } h_i > 2 \textit{ in Linear\_Progress} \\ d_{i-1} + \dfrac{\sin\alpha_{i-1}}{\pi - \alpha_{i-1}} d_{i,i-1} & \text{if } h_i > 2 \textit{ in Nonlinear\_Progress} \end{cases}$$

본 논문에서는 모의실험을 통해서 세 알고리즘의 위치 추정 성능을 비교 분석하였다. 그 결과 세 알고리즘 모두 기존의 알고리즘보다 뛰어난 위치 추적 성능을 나타내었으며, 그 중에서도 Nonlinear_Progress 알고리즘이 가장 우수하였다. Nonlinear_Progress 알고리즘은 Trigonometric 알고리즘과 비교해서 그 차이가 크진 않으나 매우 일관되게 더 나은 결과를 보여주었다. 그 이유는 오류가 포함된 거리 정보를 이용하여 직접적으로 거리 d_i를 계산하는 대신 일종의 평균을 구함으로써 포함된 오류를 다소 완화시키는 효과를 얻은 것으로 판단된다.

이 알고리즘들의 단점은 단계 1에서 이웃 노드들과의 공통 이웃 노드의 개수를 구하기 위해서 $O(m)$개의 메시지가 사용된다는 점이다. 만약 단계 2에서 앵커로부터의 거리를 구하는 과정에서 전송되는 메시지들이 각 메시지의 전송 노드와 선행 노드의 이웃 노드의 집합을 포함하도록 한다면, 단계 1에서 각 노드가 자신의 존재를 알리기 위해서 보내는 메시지들을 이웃 노드들이 다시 재방송할 필요가 없어진다. 그렇게 되면 필요한 메시지의 개수는 $O(n)$개로 줄어들 수 있다. 물론 이 경우 단계 2에서 전달되는 각각의 메시지의 길이는 길어지게 될 것이다.



(그림 7) Δ_{ij}의 계산

4. 알고리즘 Simplified

이 장에서는 제3장에서 제안한 알고리즘의 단점을 보완하고자 위치 추정 성능에 있어서는 Nonlinear-Progress 알고리즘에 근접하면서 전송되는 메시지의 개수를 기존의 DV-hop 알고리즘 등과 같은 수준으로 줄인 새로운 알고리즘을 제안한다. 이 새로운 알고리즘은 크게 세 단계로 구성된다. 첫째 단계에서 각 노드는 먼저 앵커 v_0로부터의 홉 수를 알아낸다. 두 번째 단계에서 각 노드는 앵커 v_0로부터의 거리를 추정한다. 세 번째 단계에서의 다른 알고리즘들과 마찬가지로 멀티레터레이션을 이용하여 각 노드의 위치를 추정한다.

이 알고리즘의 기본 아이디어는 다음과 같다. 일반적으로 앵커 v_0로부터 동일한 홉 수만큼 떨어져 있는 노드들은 (그림 6)에서처럼 하나의 띠 형태의 영역 내에 위치해 있을 것이다. 그 영역 내에 있는 한 노드 v_i의 입장에서 보면 노드 v_i의 이웃 노드들은 연속된 세 개의 띠 영역에 걸쳐서 흩어져 있을 것이다. 이때 노드 v_i의 이웃 노드들 중에서 홉 수가 h_i-1인, 즉 노드 v_i보다 홉 수가 1 작은 노드들의 개수는 노드 v_i의 거리 d_i에 대해서 많은 정보를 내포하고 있다. 본 장에서 기술할 알고리즘은 그 정보를 이용하여 노드 v_i의 거리를 추정하고자 하는 방법이다.

4.1 제 1 단계

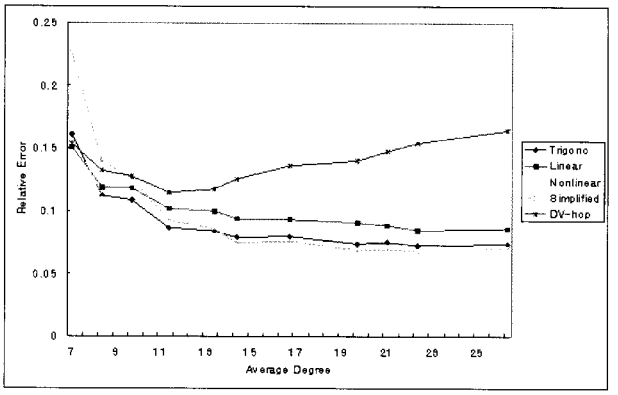
알고리즘의 제 1 단계는 각 노드의 홉 수 h_i를 계산하기 위한 것이다. 알고리즘은 기본적으로 3장에서 설명한 거리-벡터 라우팅 알고리즘과 동일하다. 다만 여기서는 모든 노드는 자신이 듣는 모든 메시지를 저장해 놓는다. 즉 메시지를 보내온 노드의 홉 수가 자신의 홉 수보다 큰 경우에도 그 메시지를 무시하지 않고 받아서 저장해 둔다. 실제로는 메시지 전송자의 홉 수를 검사하기 위해서는 어짜피 그 메시지를 받아서 저장해야 하므로 따라서 사실상 추가로 해야 할 일은 아무것도 없다. 모든 노드들은 자신의 홉 수가 갱신될 때 마다 메시지를 방송하므로 단계 1인 종료되면 모든 노드는 자신의 이웃 노드의 존재와 또한 이웃 노드의 홉 수를 알게 된다.

4.2 제 2 단계

알고리즘의 제 2단계에서 각 노드는 앵커 v_0로부터의 거리를 계산한다. 계산을 완료하면 그 계산된 거리를 메시지에 담아서 자신의 이웃들에게 방송한다. 알고리즘은 앵커 노드 v_0가 최초의 메시지를 방송함으로써 시작된다. 먼저 노드 v_i에 대해서 $h_i \le 2$라고 가정해 보자. 단계 1로부터 노드 v_i는 자신의 이웃 노드의 개수를 알고 있다. 또한 이웃 노드들 중에 홉 수가 1인 노드들이 바로 자신과 앵커 v_0간의 공통 이웃 노드이다. 따라서 노드 v_i는 2장에서 기술한 방법을 이용하여 자신과 앵커 v_0간의 거리를 추정할 수 있다.

이제 $h_i > 2$인 일반적인 경우에 대해서 고려해 보자. 노드 v_i의 이웃 노드들 중에서 그 홉 수가 h_i-1인 노드들의 집합을 N_i^{-1}이라고 표기하자. 먼저 노드 v_i는 N_i^{-1}에 속한 모든 노드들이 자신의 거리를 추정한 후 메시지를 방송할 때까지 기다린다. N_i^{-1}에 속한 모든 노드들의 거리를 전달받아서 알게 되면, 노드 v_i는 N_i^{-1}에 속한 각각의 노드 v_j에 대해서 거리의 차이 $\Delta_{ij} = d_i - d_j$를 다음과 같은 방식으로 추정한다. 집합 N_i^{-1}에 속한 노드들 중에서 v_0로부터의 추정된 거리가 d_j보다 작거나 같은 노드들은 (그림 7)과 같이 노드 v_i를 중심으로 하는 반지름 r인 원과 앵커 v_0를 중심으로 하는 반지름 d_j인 원의 교차 영역에 존재한다고 할 수 있다. 따라서 2장에서와 마찬가지로 집합 N_i^{-1}에 속한 노드들 중에서 v_0로부터의 추정된 거리가 d_j보다 작거나 같은 노드들의 개수와 노드 v_i의 이웃 노드들의 개수의 비를 이용하여 두 원의 교차 영역의 면적을 추정할 수 있고, 두 원의 교차 영역의 면적은 근본적으로 (그림 1)과 동일한 방법으로 Δ_{ij}의 함수로 표현이 가능하다. 이렇게 해서 Δ_{ij}를 계산하고 나면 거리 d_i는 다음과 같이 N_i^{-1}에 속한 각 노드 v_j에 대해서 추정된 거리 $d_j + \Delta_{ij}$의 평균값으로 추정된다.

$$d_i = \frac{1}{|N_i^{-1}|} \sum_{v_j \in N_i^{-1}} (d_j + \Delta_{ij})$$



(그림 8) 앵커와 일반 노드 간의 거리 추정의 정확도

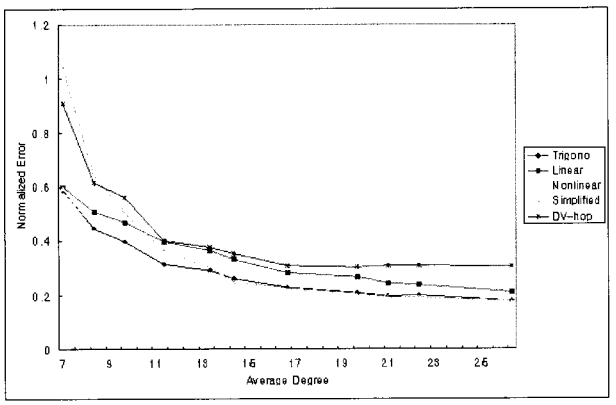
실제로는 이 알고리즘의 단계 1과 단계 2는 파이프라인 방식으로 동시에 실행될 수 있다. 그렇게 되면 알고리즘의 총 실행 시간은 감소될 수 있을 것이다. 이 알고리즘의 수행을 위해서 필요한 메시지의 개수는 거리-벡터 라우팅 알고리즘을 두 번 연속해서 실행하는 경우와 동일하며 이는 DV-hop 알고리즘의 경우와 동일하다.

5. 모의실험을 통한 알고리즘의 성능 분석

본 장에서는 본 논문에서 제안한 알고리즘의 위치 추정 성능을 모의실험을 통해 분석한 결과를 기술한다. 본 논문에서는 기존의 알고리즘들 중에서 DV-hop 알고리즘을 선택하여 본 논문에서 제안한 알고리즘과 성능을 비교하였다. 또 다른 알고리즘인 Amorphous 알고리즘의 경우에는 실험해 본 거의 대부분의 경우에 DV-hop 알고리즘보다 조금씩 나쁜 결과를

보여주었다. 또한 GHoST 알고리즘의 경우에는 단독으로 사용되는 하나의 알고리즘이라기보다는 다른 어떤 알고리즘에든 적용되어 그 알고리즘이 최악의 경우에 빠지는 것은 일정 정도 방지하는 역할을 하는 하나의 기법이라고 보는 것이 보다 적절하다고 판단된다. 그런 이유로 본 장에서는 Amorphous 알고리즘 및 GHoST 알고리즘과의 성능 비교는 기술하지 않았다.

본 장에서 기술될 모의실험은 다음과 같은 환경에서 실시되었다. 먼저 노드들은 크기가 500×500인 정사각형 형태의 영역에 균일한 확률로 무작위로 분포되어 있다. 배포된 노드의 개수는 250개이며, 각 노드의 좌표는 정수이다. 단위면적당 노드의 평균 개수, 즉 노드 밀도는 통신 범위 r을 변경함으로써 조절되었다. 앵커 노드는 정해진 개수만큼 배포되어 있는 노드들로부터 무작위로 추출하였다.



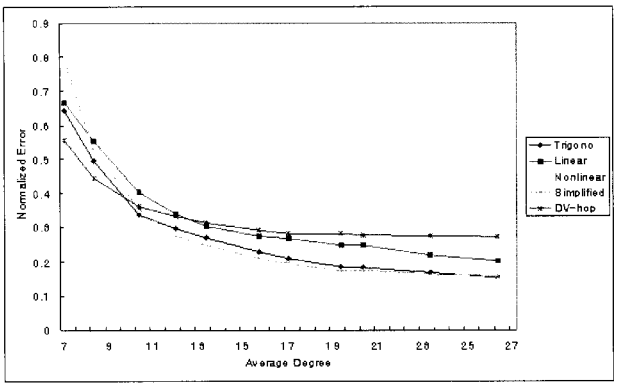
(그림 9) 위치추정의 정확도(앵커 비율 3%인 경우)

(그림 8)은 앵커 노드와 일반 노드들 간의 거리 추정의 오차율을 보여준다. 여기서 상대적 오차율은 거리 추정 오차를 실제 거리로 나눈 값이다. 앵커 노드의 개수는 전체 노드의 3%이다. 노드들 간의 거리 추정의 정확도는 본 논문에서 제시한 알고리즘의 경우에는 앵커의 개수와 무관하고, DV-hop 알고리즘의 경우에는 약간의 영향을 받는다. 물론 앵커의 개수가 증가할수록 약간 좋아지지만 큰 차이는 아니다.

(그림 8)에서 보여주듯이 Nonlinear_Progress 알고리즘이 거의 모든 영역에서 가장 정확하고, Trigonometric 알고리즘은 Nonlinear_Progress 알고리즘과 거의 유사하지만 일관되게 약간 나쁜 결과를 보여주었다. Simplified 알고리즘의 경우에는 노드 밀도가 12보다 작을 경우에는 비교적 좋지 않은 결과를 보여주나 노드 밀도가 그 이상일 경우에는 Nonlinear_Progress 알고리즘과 거의 구분되지 않는 정확도를 보여 주었다.

한 가지 특이한 사항은 DV-hop 알고리즘의 거리 추정 정확도가 노드 밀도가 일정 수준 이상이 되면 오히려 나빠진다는 사실이다. 실제로 이는 당연한 결과이다. 왜냐하면 노드 밀도가 올라가는 것은 통신 범위 r이 커진다는 것을 의미하고, 그것은 노드들 간의 평균 홉 수가 감소한다는 의미이다. 홉 수가 감소되면 거리 추정의 정밀성(granularity)은 당연히 떨어지게 된다. 이는 홉 수에 얼마를 곱해서 거리를 추정하는 모든 알고리즘의 공통된 특성이라고 할 수 있다.

(그림 9)와 (그림 10)은 각각 앵커 노드의 비율이 3%와 8%일 경우의 위치 추정의 정확도를 표현하는 그래프이다. 그래프에서 오차율은 거리 추정의 오차를 통신 범위 r로 나눈 값이다. 결과는 기본적으로 거리 추정의 정확도에 관한 데이터로부터 예상할 수 있는 그대로이다. 앵커의 비율이 3%인 겨우 노드 밀도가 대략 13정도가 될 때 Nonlinear_Progress 알고리즘과 Simplified 알고리즘의 위치 추정 오차율이 대략 통신 범위 r의 30%정도에 다다르게 되며, 앵커 비율이 8%인 경우에는 노드 밀도가 10~11 정도일 때 비슷한 오차율에 다다르게 된다. 두 알고리즘의 경우 DV-hop 알고리즘과의 오차율의 차이는 대략 10% 정도이다.



(그림 10) 위치추정의 정확도(앵커 비율 8%인 경우)

6. 결 론

본 논문에서는 공통 이웃 노드의 수를 셈으로써 두 노드 간의 위치를 추정하는 기법을 이용하여 무선 센서 네트워크에서 노드들의 위치를 추정하는 알고리즘을 제안하였다. 본 장에서는 지금까지 기술한 연구 결과를 확장해 나갈 몇 가지 차후 연구방향에 대해서 언급하고자 한다.

먼저 본 논문에서 제안한 알고리즘이 가지는 한 가지 한계점은 그것이 비교적 비현실적인 즉 이상적인 통신 범위에 관한 가정에 근거하고 있다는 점이다. 실제에 있어서는 비록 통신 범위 내에 있는 노드라고 하더라도 예측할 수 없는 다양한 이유들에 의해서 교신이 실패할 수 있다. 또한 각 노드의 통신 범위 자체도 지형이라든가 여러 가지 주변 여건에 따라서 일정하지 않을 수 있다. 이러한 보다 현실적인 요소들을 고려하여 본 논문에서 제시한 알고리즘들을 보완하는 것이 주요한 차후 연구 과제 중의 하나이다.

두 번째는 공통 이웃 노드의 수를 셈으로써 거리를 추정하는 방법을 위치 추정 알고리즘에 의해서 추정된 위치를 반복적으로 개선하는 용도에 적용해 보는 것이다. 이웃 노드들과의 공통 이웃의 개수는 실제로 노드들의 위치에 관한 아주 풍부한 정보를 내포하고 있는 것으로 생각된다. 이러한 정보들에 대한 보다 엄밀한 기하학적인 분석을 통해서 보다 정확

한 위치 추정이 가능할 것으로 생각된다.

참 고 문 헌

[1] R. Bischoff and R. Wattenhofer, "Analyzing Connectivity-Based MultiHop Ad-Hoc Positioning," Proc. of the Second Annual IEEE International Conference on Pervasive Computing and Communications, 2004.

[2] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," Proceedings of The Ninth International Conference on Mobile Computing and Networking(Mobicom), pp.81-95, San Diego, CA, Sep., 2003.

[3] L. Kleinrock and J. Silverster, "Optimum transmission radii for packet radio networks or why six is a magic number," Proc. Natnl. Telecomm. Conf., pp.4.3.1-4.3.5, 1978.

[4] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," Telecommunication Systems, Vol.22: 1-4, pp.267-280, 2003.

[5] R. Nagpal, H. Shrobe, and J. Bachrach. "Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network," 2nd International Workshop on Information Processing in Sensor Networks(IPSN 03), Palo Alto, CA, Apr. 22-23, 2003.

[6] N. B. Priynatha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," MIT Laboratory for Computer Science, Technical Report No. 892, April, 15, 2003.

[7] N. Patwari and A. O. Hero III, "Using proximity and quantized RSS for sensor localization in wireless networks,'" WSNA'03, September 19, 2003, San Diego, California, USA.

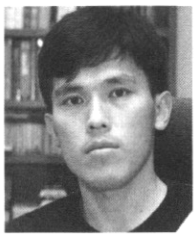
[8] N. Patwari, A. O. H. III, M. Perkins, N. S. Correal, and R. J. O'Dea, "Relative location estimation in wireless sensor networks," IEEE Trans. on Signal Processing, Vol.51, No.8, 2003, pp.2137-2148.

[9] A. Savvides, C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom 2001), Rome, Italy, July, 2001.

[10] C. Savarese, K. Langendoen, and J. Rabaey, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," USENIX Technical Annual Conference, Monterey, CA, 2002, pp.317-328.

[11] A. Savvides, H. Park, and M.B. Srivastava, "The Bits and Flops of the N-hop Multilateration Primitive For Node Localization Problems," WSNA'02, Atlanta, Georgia, USA. September, 28, 2002.

권 오 흠



e-mail : ohkwn@pknu.ac.kr
1988년 서울대학교 컴퓨터공학과(공학사)
1991년 KAIST 전산학과(공학석사)
1996년 KAIST 전산학과(공학박사)
1996년~1997년 전자통신연구소 박사후연수연구원
1997년~1999년 부경대학교 전자컴퓨터정보통신공학부 전임강사
1999년~2003년 부경대학교 전자컴퓨터정보통신공학부 조교수
2003년~현재 부경대학교 전자컴퓨터정보통신공학부 부교수
관심분야 : 알고리즘, 연결 네트워크, 센서 네트워크 등

송 하 주

e-mail : hajusong@pknu.ac.kr
1993년 서울대학교 컴퓨터공학과(공학사)
1995년 서울대학교 컴퓨터공학과(공학석사)
2001년 서울대학교 컴퓨터공학과(공학박사)
2003년 ㈜아이티포웹 부장
2003년~현재 부경대학교 전자컴퓨터정보통신공학부 전임강사
관심분야 : 데이터베이스, 웹서비스

김 숙 연

e-mail : entpentp@paran.com
1991년 연세대학교 전산과학과(이학사)
1993년 KAIST 전산학과(공학석사)
1998년 KAIST 전산학과(공학박사)
1998년~2004년 한국전자통신연구원(ETRI) 선임연구원
2004년~현재 한경대학교 컴퓨터공학과 조교수
관심분야 : Interconnection Network, 알고리즘, 그래프 이론