DOI: 10.3745/KIPSTA.2010.17A.2.103

근사 최소 스타이너 트리를 이용한 효율적인 클러스터 센서 네트워크의 구성

김 인 범†

요 약

클러스터 센서 네트워크는 여러 개의 중심 노드 주위에 다른 입력 노드들이 밀집된 분포를 보이는 센서 네트워크이다. 최소 스타이너 트리는 스타이너 포인트들을 도입하여 모든 입력 노드들을 최소 비용으로 연결하는 트리이다. 본 논문에서는 센서 노드와 베이스 스테이션의 연결인 간선들을, 클러스터 내에서와 클러스터 사이에서 각각 생성하고, 이를 이용하여 근사 최소 스타이너 트리를 반복적으로 생성하여, 단축된 길이의 클러스터 센서 네트워크를 구성하는 방법을 제안한다. 실행 시간 복잡도가 $O(N^2)$인 제안된 방법으로 생성된 클러스터 센서 네트워크들은, 본 논문의 실험에서 유클리드 최소 신장 트리 방법의 네트워크들과 비교하여 생성 시간이 1170.5% 증가하였으나 최소치보다 0.1% 증가된 길이의 네트워크는 20.3%의 증가된 시간에 생성이 가능했다. 이 클러스터 센서 네트워크의 평균 길이는 유클리드 최소 신장 트리 방법과 비교하여 최대 3.7%, 평균 1.9% 감소되었다.

키워드 : 클러스터 센서 네트워크, 근사 최소 스타이너 트리, 유클리드 최소 신장 트리, 스타이너 포인트, 스타이너 그래프

A Design of Efficient Cluster Sensor Network Using Approximate Steiner Minimum Tree

Inbum Kim†

ABSTRACT

Cluster sensor network is a sensor network where input nodes crowd densely around some nuclei. Steiner minimum tree is a tree connecting all input nodes with introducing some additional nodes called Steiner points. This paper proposes a mechanism for efficient construction of a cluster sensor network connecting all sensor nodes and base stations using connections between nodes in each belonged cluster and between every cluster, and using repetitive constructions of approximate Steiner minimum trees. In experiments, while taking 1170.5% percentages more time to build cluster sensor network than the method of Euclidian minimum spanning tree, the proposed mechanism whose time complexity is $O(N^2)$ could spend only 20.3 percentages more time for building 0.1% added length network in comparison with the method of Euclidian minimum spanning tree. The mechanism could curtail the built trees' average length by maximum 3.7 percentages and by average 1.9 percentages, compared with the average length of trees built by Euclidian minimum spanning tree method.

Keywords : Cluster Sensor Network, Approximate Steiner Minimum Tree, Euclidean Minimum Spanning Tree, Steiner point, Steiner Graph

1. 서 론

센서 네트워크는 다양한 형태의 센서 노드가 수집한 자료와 정보를 무선 통신 등을 이용해서 다른 센서 노드나 베이스 스테이션등을 매개로 하여 전송하는 네트워크로, 최근 정보통신 기술의 급속한 발전함에 따라 이러한 센서 네트워크의 응용분야는 확대되며 활성화되고 있다. 예를 들면 인간의 접근이 용이하지 않은 위치에 이러한 센서 네트워크를 구축하여, 관측 및 감시 활동에 유용하게 활용할 수 있다. 클러스터 센서 네트워크는 각 노드들이 여러 개의 특정 노드를 중심으로 클러스터를 형성하여 같은 클러스터내의 다른 노드들과 빈번하게 통신하고 필요에 따라 다른 클러스터의 노드들과 가끔 통신하는 센서 네트워크이다. 이러한 경우 통신 라우팅을 위해 전체 노드가 아닌 해당 클러스터내

※ 이 논문은 2010학년도 김포대학의 연구비 지원에 의하여 연구되었음.
† 정 회 원 : 김포대학 IT학부 부교수
논문접수 : 2009년 8월 10일
수 정 일 : 1차 2009년 12월 28일
심사완료 : 2009년 12월 29일

의 노드들을 우선한 효율적인 트리의 구성은 네트워크 트래픽 감소와 더불어 전송 거리의 단축이 가능하므로, 신뢰할 수 있는 정보의 신속한 전달이 가능하다.

본 논문에서는 클러스터 센서 네트워크를 구성하는 많은 센서 노드와 베이스 스테이션을 연결하여 트리 또는 네트워크를 구축함에 있어, 최소 신장 트리를 적절히 활용한 근사 최소 스타이너 트리의 생성 및 적용 방법을 제안한다. 이 방법에서 클러스터 센서 네트워크의 각 클러스터에 소속된 입력 노드들을 완전 연결하는 간선(edge)들과, 각 클러스터 사이를 최소 거리로 연결하는 간선들을 생성한 후, 입력 노드들과 이러한 간선들을 이용하여 최소 신장 트리를 생성하고, 이를 기반으로 근사 최소 스타이너 트리를 미리 정의해 둔 종료조건이 만족될 까지 반복적으로 생성한다. 종료 조건을 만족하는 어떤 근사 최소 스타이너 트리의 간선 중에서 제한 전송 거리를 초과하는 간선들을 찾고, 이들을 분해하여 제한 전송 길이 이내의 간선들로 대체하는 연결들을 생성한다. 이 제한 전송 거리는 무선통신에서의 최대 전송 거리와 동일하다. 이 방법으로 생성된 최종 트리의 노드들과 간선들이 클러스터 센서 네트워크의 각 요소들인 센서 노드, 베이스스테이션과 연결이다.

본 논문의 구성은 다음과 같다. 2장은 센서 네트워크와 스타이너 트리에 대한 주요 내용이고, 3장은 본 논문에서 제안하는 근사 최소 스타이너 트리를 이용한 클러스터 센서 네트워크 구축에 관한 내용이다. 4장에서는 제안된 방법을 적용한 클러스터 센서 네트워크의 생성 실험 및 다른 방법으로 생성된 네트워크와의 비교이고, 5장은 결론이다.

2. 센서 네트워크와 스타이너 트리

센서 네트워크는 인간의 생활을 편리하게 하는 여러 다양한 기구나 장치, 장비에 송수신장비를 부착하여 이를 연결하고 통신할 수 있게 하거나, 사람의 접근이 힘든 공간에 많은 센서 노드를 설치하여, 각 센서 노드에서 수집한 정보를 유무선으로 연결하고 이를 이용하여 감시활동 등에 이용할 수 있다[1, 2]. 최근의 무선 개인 영역 네트워크 기술 및 초소형 네트워크 장비 기술 등 관련 기술의 급속한 발전에 힘입어 센서 네트워크를 활용한 응용이 활성화되고 있으며 향 후에 다양한 분야에 유용하게 널리 활용될 전망이다.

센서 네트워크의 보편적 활용을 위한 필수적인 기술은 저전력 및 저가 무선통신 기술, 초소형 마이크로프로세서 기술, Ad-hoc 네트워크 기술, 다양한 형태의 센싱 기술, 임베디드 시스템 기술 등이다. 센서 네트워크의 주요 구성요소인 센서 노드는 감지 기능, 계산 처리 능력, 무선통신 기능 등을 보유하고 있으며, 또 다른 구성요소인 베이스 스테이션은 일반적인 네트워크에서의 게이트웨이 또는 데이터 집중국의 역할을 한다. 센서 노드는 감지된 정보나 자료를 베이스 스테이션으로 전달하고, 베이스 스테이션은 기존에 설치된 다양한 형태의 네트워크를 통하여 특정 사용자나 데이터 센터에 해당 정보 제공한다. 센서 노드들은 기존에 설치된 네트워크의 이용 없는 Ad-Hoc 네트워크를 구성할 수도 있다. 센서 네트워크가 활용되는 대부분 환경에서는 에너지원의 공급이 원활하지 않기 때문에 물리적으로 최대한 전력 소모가 적게 설계하여 센서 노드의 수명을 최대화하여야 한다. 또한 센서 노드는 대량 생산되는 제품에 내장될 수 있으므로 가격이 비교적 저렴해야 하며, 네트워크 토폴로지 변화에 잘 적응할 수 있어야 한다. 센서 네트워크를 구성하는 하드웨어에는 전원공급 장치, 무선통신을 위한 무선 송수신기, 다양한 센서와 아날로그/디지털 변환기로 구성된 센싱 장치, 실행 소프트웨어가 설치된 프로세싱 장치, 센서 노드 자신의 정확한 위치 정보를 위한 위치 추정 시스템, 물리적 이동성을 제공하는 이동장치 등이 있다. 이와 함께 필요한 기술들은, 소형 운영체제 기술, 저 전력 네트워크 프로토콜 기술, 동기 및 위치인식 기술, 미들웨어 기술, 보안기술 등이다. 이 중에서 저 전력 네트워크 프로토콜 기술은 저 전력 라우팅, 데이터 기반 라우팅과 함께, 다른 네트워크와의 상호 운용성이 보장되어야 한다. 또한 센서 네트워크는 일반적으로 멀티-홉 기반의 Ad-hoc 네트워크이므로 정확한 동기화 및 위치인식 기술이 필요한데, 위치인식은 센서 노드의 위치를 집중적, 분산적인 방법을 이용하여 인지한다. 또한 센서 네트워크를 안전하게 활용하기 위해서 센서 네트워크와 교환되는 정보에 대한 보안기술이 필요하다.

센서 네트워크에서 네트워크 토폴로지 구성과 라우팅에 최소 신장 트리가 사용될 수 있다. 제한된 에너지원을 갖는 센서 노드들의 특성 상, 센서 네트워크의 토폴로지 구성 시, 전체 네트워크 수명의 극대화 및 라우팅 에너지의 효율성을 심각히 고려해야 한다. 이를 위해 M. Sanchez 등은 최적의 전송 범위를 결정하는 방법 등을 제안하였고[3], J. Li 등은 라우팅을 위해 최소 신장 트리 기반의 토폴로지 구성 방법을 연구하였다[4]. 모든 센서 노드가 직접 베이스 스테이션과 통신하는 라우팅 방식 중에서 라우팅 트리의 총 길이와 소모 에너지가 가장 작은 PEDAP는 라우팅 트리로 최소 비용 신장 트리를 사용하고, 이 PEDAP를 개선하여 각 노드의 에너지의 잔량분포를 균등하게 함으로 전체 네트워크의 수명을 연장시키기 위해, 사용되는 링크 값으로 에너지 값을 에너지 잔량으로 나눈 결과를 사용하는 PEDAP-PA가 연구되었다[5].

최소 신장 트리는 노드과 간선이 주어져 있을 때, 주어진 일부 입력 간선을 이용해서 주어진 노드들을 최소 비용으로 모두 연결하는 트리이다. 일반적으로 비용은, 트리의 길이를 의미한다. 이 최소 신장 트리는 네트워크 설계, 통신 라우팅, 회로 설계, 도로 건설 등에 활용될 수 있다. 이 문제를 해결하는 대표적인 알고리즘은 Kruskal과 Prim의 알고리즘이다[6]. 유클리드 최소 신장 트리 문제는 간선이 입력으로 주어지지 않는 것이 일반적인 최소 신장 트리 문제와의 차이이다. 이 문제의 해를 위한 한 방법은, 우선 주어진 모든 입력 노드 N 개에 대해서, (N-1)개의 다른 노드들을 모두 연결하여 1/2×N×(N-1) 개의 간선들을 생성하고, 입력 노드들과 생성된 간선들을 Kruskal 혹은 Prim의 최소 신장 트리 알

고리즘 등에 적용하는 것이다. 이 방법의 결과가 유클리드 최소 신장 트리의 한 모습이 될 것이다.

전통적인 컴퓨터 이론 분야에서 네트워크 길이를 단축하려는 최적화 문제의 하나로 최소 스타이너 트리(Steiner Minimum Tree) 문제가 있다 [7, 8]. Jakob Steiner는 평면상에 존재하는 특정한 점을 찾아, 몇 개의 입력 단말 노드들을 최소 거리의 합으로 연결하는 방법을 연구하였다. 이것은 매우 많은 노드들의 집합을 입력으로 하여 별 모양 형태의 *Steiner Star*를 형성하고, 그것의 중심이 되는 노드를 찾는 문제로 확대되었다. Courant와 Robbins은 여러 개의 중심, 즉 스타이너 포인트들의 집합을 이용하여 최소 비용으로 모든 입력 단말 노드들을 연결하는 최단 길이의 네트워크를 찾는 방법을 연구하였다. 이러한 네트워크가 최소 스타이너 트리이다. 즉, 스타이너 포인트라는 입력 단말 노드가 아닌 부가적인 노드들을 도입하고, 기존 입력 노드들과 더불어 이 노드들을 활용하여 입력 단말 노드들을 모두 연결하는 트리가 일반적인 스타이너 트리이고, 이들 스타이너 트리들 중에서 가장 적은 비용의 트리가 최소 스타이너 트리이다. 이 문제는 비다항식적 시간(Non-Polynomial Time) 문제로 알려져 있다[7-9]. 이는 현실적인환경에서, 다항적 시간 내에 이 문제에 대한 해를 구할 수 없다는 것을 의미하는 것으로 현실 문제에 응용하기 위해서는 적절한 휴리스틱을 이용한 근사 최소 스타이너 트리가 필요하다. 이러한 근사 최소 스타이너 트리는 동적 라우팅, ad-hoc 네트워크, 회로 설계, 항로 결정, 도로 연결 등 여러 분야에 활용될 수 있다. 현재까지 최소 스타이너 트리 문제 영역에서는 근사 트리를 구하여 근사 비율을 높이는 연구와 근사 트리를 구하는 휴리스틱에 관한 연구들이 주로 수행되어왔다. 최소 스타이너 트리 문제의 변형인 GOSST(Grade of Services Steiner Minimum Tree) 문제에 대해, J.Kim 등은 GOSST 문제에 PTAS(Polynomial Time Approximation Scheme) 기법을 적용하여, $(1+\varepsilon)$ 근사 트리를 다항적 시간 내에 얻을 수 있음을 증명하였다[9]. 또한 근사 스타이너 트리를 이용하여 원격 무선 검침 시스템의 네트워크를 효과적으로 구성한 연구가 발표되었다. 이 연구는 원격 검침 시스템을 구성하는 검침기, 중계기, 집중기의 배치 및 연결을 근사 스타이너 트리를 이용한 방안을 제안하였고, 이를 최소 신장 트리를 이용한 방법과 비교하였다[10]. 비 방향 스타이너 트리 문제를 위한 연구로, 그래프 축소 규칙을 이용하여, 먼저 불필요한 노드들과 간선들을 제거한 후, Prim의 알고리즘을 적용한 Max-Min Ant Colony 최적화 방법에 관한 연구가 발표되었다[11]. 하나의 송신자에서 다수의 수신자로 전송하는, 멀티캐스팅 문제에서 스타이너 트리를 활용한 연구로는, 멀티캐스팅 문제와 순회 판매원 문제의 근본적인 차이를 분석하여, 기존의 개미 시스템(Ant System)을 변경한 엘리트 에이전트에 의한 개미 멀티캐스트 라우팅 모델의 제안과 멀티 캐스트 통신에서의 QoS(Quality of Service)를 처리할 수 있는 다중 제약 멀티 캐스트 처리 알고리즘에 관한 연구가 대표적이다[12, 13]. 또한 센서 네트워크의 효율적인 라우팅을 위하여 센서 노드들을 최적으로 상호 연결하는 배치 문제를 스타이너 트리를 활용하여 해결할 수 있다는 전제에서, 센서 네트워크에서의 물리적인 특성이, 일반적인 그래프 노드 연결 문제의 범위를 축소할 수 있다는 점을 주장하며 이를 기반으로 실행시간과 최적 치에 대한 근사비율이 연구되어 발표되었다[14]. 스타이너 트리를 VLSI 라우팅 및 설계에 응용하기 위하여, 이를 변형시킨 직선 Steiner 최소 트리(Rectilinear Steiner Minimal Tree, RSMT)의 생성에 관한 여러 연구도 수행되었는데, Polygonal Contraction 기법을 이용하여 RSMT를 직선 최소 신장 트리(rectilinear minimum spanning tree, RMST)로 변환하여 원하는 목표 RSMT을 생성하는 연구와 IC 설계 시 고려해야하는 Large-Scale Power Networks와 Antenna Jumpers 등과 같은 장애 요소들을 위한 Obstacle-Avoiding Rectilinear Steiner Minimal Tree (OARSMT) 문제를 해결하기 위해, Obstacle-Avoiding Routing Graph (OARG)라는 새로운 라우팅 그래프를 제안하고, 3단계로 구성된 효과적인 알고리즘에 관한 연구가 발표되었다[15, 16].

현재까지 발표된 근사 최소 스타이너 트리 구성을 위한 휴리스틱들 중에서, 외접 삼각형을 이용한 방법이 유명하다[17]. 이 방법은 트리에 존재하는 인접한 두 간선에 존재하는 세 정점으로 한 개의 삼각형을 생성하고 이 삼각형의 가장 큰 변을 L, L의 대각을 R로 정의한다. L을 한 변으로 하여 정삼각형을 생성하고, 이 정삼각형의 구성을 위해 새로 도입된 정점을 Q라 정의할 때, 이 정삼각형의 외접원과 정점 Q와 R을 연결한 직선의 교점을 스타이너 포인트로 정의한다. 모든 간선의 쌍에 대해 이와 같은 방법으로 스타이너 포인트들을 생성하고 이를 활용하여 스타이너 트리를 생성한다. 즉, 고려 대상인 트리에서, 서로 인접한 모든 두 간선을 구성하는 세 개의 노드들의 집합에 대해 삼각형들을 형성하고, 위의 방법을 이용하여 스타이너 포인트를 생성하여 최소 스타이너 트리를 생성한다.

3. 제안 방법

본 논문에서 제안하는 방법을 검증하기 위한 비교대상은 naïve spanning 센서 네트워크 이다. naïve spanning 센서 네트워크의 구성방법은 센서 노드들과 베이스 스테이션들을 모델링하는 각 노드들에 대하여, 다른 노드들을 모두 완전 연결하는 간선들을 생성하고 이들 노드들과 간선을 이용하여 최소 신장 트리를 생성하는 것이다. 생성된 최소 신장 트리에 대하여 각 간선의 길이를 점검하여 노드의 최대 전송 거리를 초과하는 것에 대해서는 최대 전송 거리 단위로 두 노드 사이에 중계 노드를 위치시켜 네트워크를 완성한다.

본 논문에서 제안하는 방법은 각 클러스터내의 모든 입력 노드들을 완전 연결하고 또 각 클러스터를 최소의 길이로 연결할 수 있는 노드들을 각 클러스터에서 찾아 서로 연결하는 간선들을 생성한다. 입력 노드들과 생성된 간선들을 이용하여 최소 신장 트리를 생성한 후, 적절한 방법을 이용

하여 스타이너 포인트들을 생성한다[17]. 생성된 스타이너 포인트들을 이와 관련된 노드들과 연결하여 스타이너 그래프를 생성한 후, 이 그래프의 노드들과, 스타이너 포인트, 간선을 이용하여 최소 신장 트리를 생성한다. 생성된 최소 신장 트리에서 필요 없는 스타이너 포인트나 연결을 삭제하여 중간 단계의 근사 최소 스타이너 트리를 생성한다. 이 중간 단계의 트리를 좀 더 최적화하기 위해, 스타이너 포인트들을 생성하여 새로운 스타이너 그래프와 최소 신장 트리를 생성하고 필요 없는 스타이너 포인트와 연결을 삭제하는 작업을 다시 차례로 반복하여 새로운 근사 최소 스타이너 트리를 생성한다. 미리 정해진 종료 조건을 만족할 때까지 이러한 과정으로 반복적으로 근사 최소 스타이너 트리를 생성하여 최종 트리를 생성하고, 이 트리의 간선 중에서 최대 전송거리를 초과하는 간선에 대하여 최대 전송 거리 단위로 두 노드 사이에 중계 노드를 추가한다. 종료 조건의 예는 미리 정해진 반복 회수이거나 각 반복되어 생성되는 트리의 전체 길이 변화의 범위가 될 수 있다.

이렇게 구성된 센서 네트워크는 필요한 정보를 이 네트워크의 경로에 따라 유무선 통신을 통해 내부적으로 정보를 전송할 수 있거나 다른 외부 네트워크와 연결하여 송수신할 수 있다.

센서 노드와 베이스 스테이션, 중계기 등으로 구성된 센서네트워크를 신속하고 효율적으로 구성하기 위해 본 논문에서 제안하는 단계별 과정은 다음과 같다.

단계 1: 각 클러스터 C_i에 속한 입력 노드 집합 N_i들에 대해 완전 연결하여 생성된 간선들의 집합 E_1을 생성한다.

단계 2: 각 클러스터들을 최소 거리로 연결하는 노드들을 찾아 서로 연결한 간선의 집합 E_2를 생성한다.

단계 3: 입력 노드 집합 N과 단계1, 2에서 생성된 간선 E_1, E_2를 이용하여 최소 신장 트리 SP_1을 생성한다.

단계 4: SP_1의 서로 인접한 두 간선을 구성하는 세 노드들의 집합을 이용하여 스타이너 포인트 집합 S를 생성한다.

단계 5: S의 원소인 각 스타이너 포인트 s_j와 s_j 생성 시 이용된 노드들을 연결한 간선들의 집합 E_3를 생성한다. SP_1에 S와 E_3를 추가하여, 스타이너 그래프를 SG을 생성한다.

단계 6: SG의 입력 노드 집합 N, 스타이너 포인트 집합 S, 연결 간선의 집합 E_1, E_2, E_3들을 이용하여 최소 신장 트리 SP_2를 생성한다.

단계 7: SP_2에서 불 필요한 스타이너 포인트 집합 S_{-1}과 연결 간선의 집합 E_{-1}을 삭제하여 근사 스타이너 트리 ST을 생성한다.

단계 8: 종료조건을 조사하여 만족하지 않으면 ST을 SP_1으로 변환한 후, 단계 4로 이동하고, 만족하면 단계 9로 이동한다.

단계 9: ST의 각 간선의 길이를 조사하여 최대 전송거리를 초과하는 간선에 대해 적절한 길이의 위치에 중계 노드 집합 M을 추가하면서 이를 이용해 해당 간선을 분할 한, 최종 근사 최소 스타이너 트리 ST'을 출력한다.

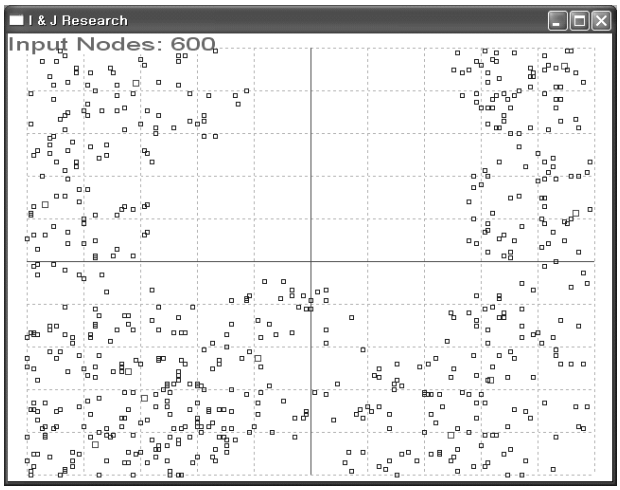
위 과정의 실행시간을 분석하면 다음과 같다. 단계 1에서 전체 입력 단말 노드의 수 N은 각 클러스터에 속한 노드의 수를 합한 것과 동일하다. 클러스터의 수를 C라 할 때, 각 클러스터의 속한 노드의 수가 거의 비슷하다면 각 클러스터의 소속 노드의 수는 $n_a \fallingdotseq N/C$ 이다. 이 경우 완전 연결을 위한 실행 총 시간은 $\sum_{i=1}^{C} n_i \times (n_i - 1) = O(C \times n_a^2) = O(n_a^2) = O(N^2)$이다. 단계 2에서의 각 클러스터간의 최소거리를 구하기 위해서 모든 클러스터 쌍에 속한 노드들 사이의 거리를 구해야 하므로 이를 위한 실행 시간은 $\sum_{j>i}^{C} \sum_{i=1}^{C} n_i \times n_j = O(C^2 \times n_a^2) = O(n_a^2) = O(N^2)$이다. 단계 3에서 Prim의 알고리즘을 이용한 최소 신장 트리를 생성하기 위한 시간은 O(N×logN)이다[6].

최소 신장 트리를 구성하는 간선의 수는 (노드의 수)-1 이므로, 단계 4에서 서로 인접한 두 간선의 전체 수는 O(N)이고, 따라서 새로 생성되는 스타이너 포인트들의 집합 S의 크기 $N_S \le O(N)$이다. 단계 5에서 새로 생성된 스타이너 포인트의 연결 간선의 생성 시간은 생성된 스타이너 포인트의 수에 비례하므로 O(N) 이다. 단계 6에서 최소 신장 트리 SP_2를 생성하기 위해 고려해야 할 노드의 수 N_1은 기존의 입력 단말 노드의 수에 생성된 스타이너 포인트의 수를 추가하여, $O(N_1)=O(N+N_S) \le 2O(N)$이고, 이들을 대상으로 최소 신장 트리를 생성하기 위한 시간은 $O(N_1 \times \log N_1) \le O(2 \times N \times \log(2N)) = 2 \times O(N \times \log N)$이다. 단계 7에서 불필요한 스타이너 포인트를 검사하고 처리하기 위해 고려해야 할 대상은 단계 4에서 생성된 스타이너 포인트의 집합 S이고, 그것의 크기는 $N_S \le O(N)$ 이므로 O(N)이다. 이렇게 생성된 트리가 종료조건을 만족하지 못했을 경우 두 번째 반복의 단계 4에서 생성되는 스타이너 포인트 수 $N_S \le O(N_1) = 2 \times O(N)$이다. 단계 5에서 새로 생성된 스타이너 포인트의 연결 간선을 위한 시간은 $O(N_1) = 2 \times O(N)$이고, 단계 6에서 최소 신장 트리를 생성하기 위해 고려해야 할 노드의 수는 $N_2 = O(N_1 + N_S) \le 2O(N_1) = 2^2 \times O(N)$이고 실행시간은 $O(N_2 \times \log N_2) \le 2^2 \times O(N \times \log N)$이다. 단계 7에서 삭제를 고려해야 할 스타이너 포인트의 수는 $N_S \le O(N_1)$ 이므로 $O(N_1) = 2 \times O(N)$이다. k번째 반복에서 종료조건을 만족했을 때, 단계 9에서 고려해야 할 것은 현재 트리의 간선의 수가 되는 데, 이 트리의 간선의 집합 크기가 최대인 경우는 단계 6에서 생성된 최소 신장 트리 SP_2의 간선의 수가 될 것이다. k번째 반복의 단계 6의 SP_2의 노드의 수 $N_k = 2^k \times O(N)$이고 간선의 최대 수는 $N_k - 1 = 2^k \times O(N) - 1 \fallingdotseq 2^k \times O(N)$이고, 따라서 단계 9의 실행 시간은 $2^k \times O(N)$이다. 참고로 k번째 반복의 최소 신장 트리 SP_2를 생성하는 시간은 $2^k \times O(N \times \log N)$이다.

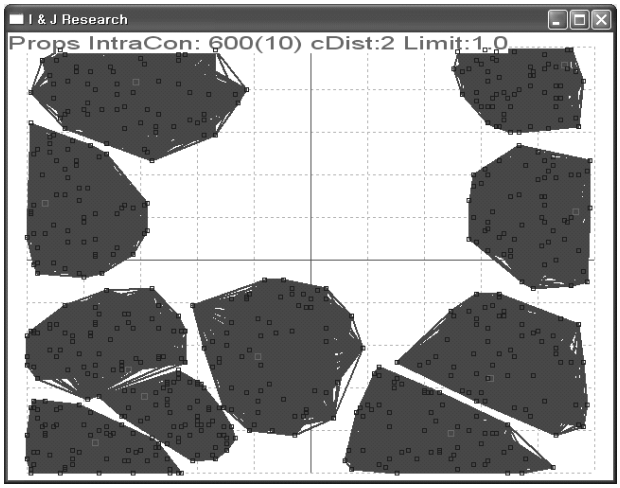
단계 4부터 단계 7까지의 과정에서, k번째 반복의 경우 가장 큰 시간을 요구하는 것이 단계 6의 $2^k \times O(N \times \log N)$이

고, k번째가 종료조건을 만족할 경우 소모되는 시간은 $(2^{k+1}-2)\times O(N\times logN)$이다. 따라서 위에서 기술한 과정을 수행하기 위한 총 수행시간은 단계 1과 단계 2의 완전 연결 간선을 생성하는 과정을 포함하여 $O(N^2)+(2^{k+1}-2)\times O(N\times logN)$인데, 노드의 수 N에 비해 반복 회수 k를 작게 할 경우, 이 값은 $O(N^2)$이 된다. 따라서 정해진 반복회수 k를 외부 인자로 입력하여 상수화 했을 경우, 본 논문에서 제안하는 방법은 $O(N^2)$ 시간 내에 클러스터 센서 네트워크를 형성할 수 있다. 이는 본 연구에서 제안하는 방법과 비교대상인 naïve spanning 방법과 동일하다.

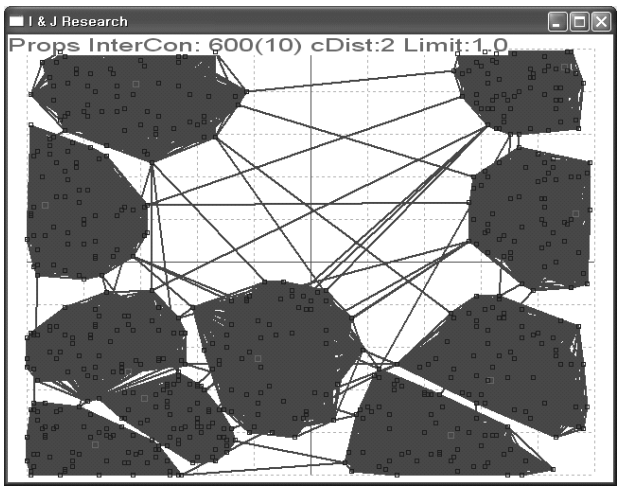
(그림 1)은 10개의 클러스터로 형성된 600개의 입력 단말 노드를 보이고 있다. 각 클러스터의 중심 노드와 일반 노드들은 센서 네트워크의 베이스 스테이션과 센서 노드로 모델링할 수 있다. (그림 2)는 각 클러스터에 속한 입력 노드들을 모두 완전 연결한 간선들이 보인다. (그림 3)은 각 클러스터를 최소 거리로 연결하는 노드들을 각각 찾아, 이들을 서로 연결한 결과를 보인다. C개의 클러스터가 있다면 한 클러스터 당 C-1개의 이러한 클러스터간 연결이 생성된다.



(그림 1) 600개의 입력 단말노드

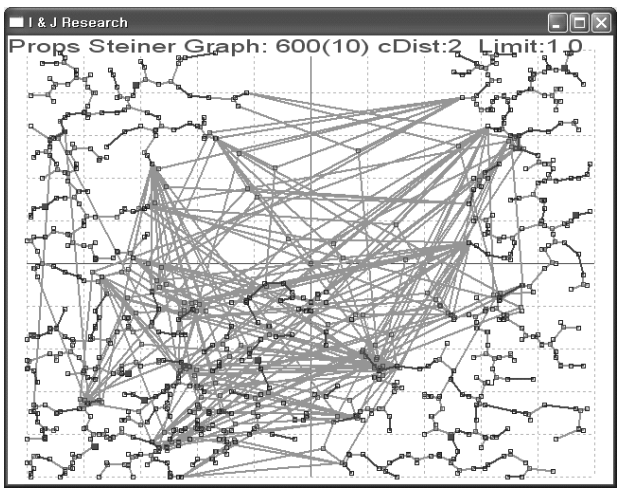


(그림 2) 클러스터 내부 노드들의 완전연결



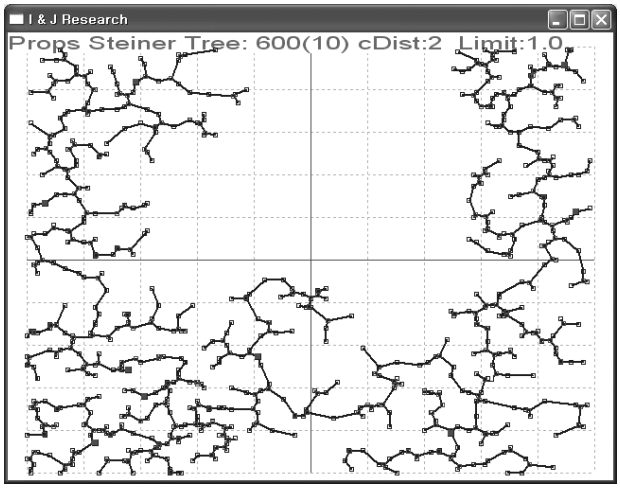
(그림 3) 클러스터 간의 최소 거리 연결

(그림 4)는 (그림 3)의 노드들과 간선들을 이용하여 최소 신장 트리를 생성하고, 이 트리내의 서로 인접한 두 간선의 노드들을 이용하여 스타이너 포인트를 생성한 후 이들을 연결한 스타이너 그래프의 모습이다. 이 때, 생성되는 스타이너 포인트들은 각 클러스터 내부의 연결과 함께 스타이너 간 연결에 대해서도 생성이 된다. (그림 5)는 스타이너 그래프의 노드들과 스타이너 포인트, 그리고 간선을 이용하여 최소 신장 트리를 생성하고 여기서 불필요한 스타이너 포인트와 이것과 관련된 연결과 함께 해당 스타이너 포인트를 제거한 근사 최소 스타이너 트리의 모습이다. 최종적으로 각 연결의 길이가 최대 전송 거리를 초과했을 경우, 최대 전송 거리 이내의 적절한 길이의 위치에 중계 노드를 생성하는데, 이것들은 센서 네트워크에서의 중계기에 대응된다. (그림 6)은 동일한 입력에 대해 naïve spanning 방법으로 생성된 naïve 최소 신장 트리의 모습이고, (그림 7)은 cluster spanning 방법의 클러스터 최소 신장 트리이다. 클러스터 최소 신장 트리는 각 클러스터에 속한 입력 단말 노드들을 완전 연결하고, 또 각 클러스터 사이의 최소 길이의 연결을

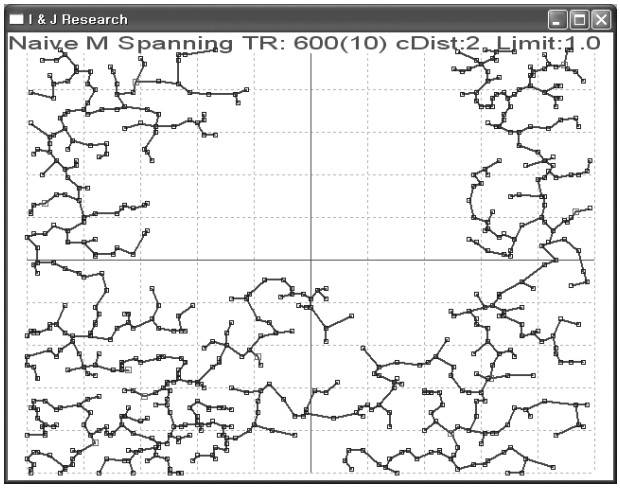


(그림 4) 스타이너 그래프

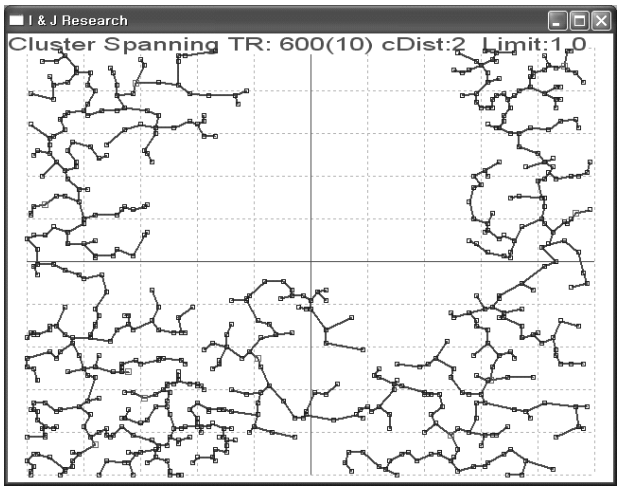
생성 한 후에 이것에 대해 최소 신장 트리를 생성한다. 이 방법은 전체 입력 노드와 이들의 완전 연결로 생성하는



(그림 5) 제안된 방법의 스타이너 트리(길이=131.56)



(그림 6) Naive 최소 신장 트리(길이=134.52)



(그림 7) 클러스터 최소 신장 트리(길이=135.77)

naïve 최소 신장 트리에 비해 전체적인 트리의 길이는 증가할 수 있으나, 신속하게 생성할 수 있는 장점이 있다. 본 연구의 실험에서 cluster spanning 방법이 naïve spanning 방법에 비해 평균 51.7%의 시간 절감을 보였다. 본 논문에서 제안한 방법은 10개의 클러스터로 구성된 600개의 입력 단말 노드와 최대 전송 거리가 1인 환경에서, naïve 최소 신장 트리에 비해 2.2%, cluster spanning 방법에 비해 3.1%의 네트워크 길이의 절감을 보인다.

4. 실험 및 결과

본 논문의 제안 방법을 검증하기 위한 실험의 인자로 입력 노드의 수 N, 클러스터의 수 C, 그리고 각 노드들의 최대 연결 길이 L이 이용된다. 본 연구에서는 각 노드의 최대 전송 거리를 1로 정의한다. 관찰 대상은 생성된 센서 네트워크 트리의 길이 및 생성 시간, 최종 네트워크를 생성하기 위한 근사 스타이너 트리의 최소 반복 생성 회수이다. 또한 3장에서 기술한 naïve spanning 방법, cluster spanning 방법, 그리고 본 논문에서 제안한 방법의 센서 네트워크들의 길이와 생성 시간을 비교한다. 실험을 위해 무작위로 생성된 노드의 수 N은 600, 1200, 1800, 2400, 3000개이다. 각 노드의 생성 방법은 무작위로 C개의 각 클러스터를 대표하는 중심 노드를 생성하고, 가장 가까운 중심 노드와의 거리가 최대 L만큼 떨어진 입력 단말 노드들을 생성한다. 클러스터의 수 C는 2, 4, 6, 8, 10등 5가지 이고, 최대 연결거리 L은 2, 3, 4, 5 등 4가지로 선택하여 실험하였다. 생성된 입력 노드들은 서로 중복되지 않는, -5.0과 5.0 사이의 x, y 좌표에 위치한다. 실험 환경은 Intel 1.83 GHz (T5600) 프로세서와 4기가 메모리의 랩탑 컴퓨터이고, 본 논문에서 제안하는 메카니즘은 C++로 구현하였다.

<표 1>에는 본 실험의 인자인 입력 단말 노드 수 5가지, 클러스터 수 5가지, 최대 연결 길이 4가지를 조합한 100가지 종류의 입력환경에서 주어진 방법에 따라 생성된 트리들의 평균 결과를 보인다. 본 논문에서 제안한 방법은 최종 클러스터 센서 네트워크 트리를 생성하기 위해 200번의 중간 단계의 근사 최소 스타이너 트리를 반복적으로 생성한다. 그 이유는 스타이너 트리의 생성은 비다항식적 시간(Non-Polynomial Time) 문제에 속하므로, 이것의 최적 해를 현실

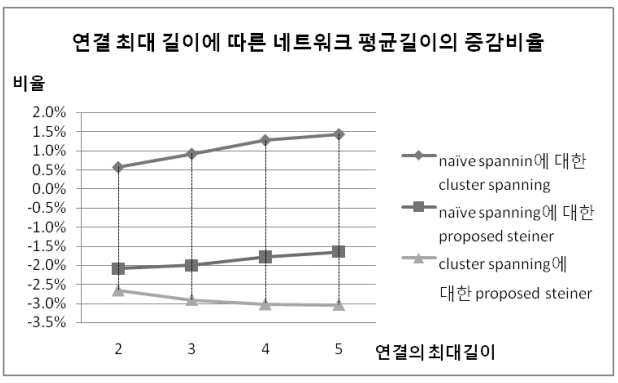
〈표 1〉 실험결과

생성트리 항목	트리 길이	평균길이 증감율	실행시간	시간 증감율	반복 회수
naïve spanning	235.64	0.0% (-1.1%)	46154.9	0.0%	1.0
cluster spanning	238.22	1.1% (0.0%)	22283.4	-51.7%	1.0
proposed Steiner(200)	231.25	-1.9% (-2.9%)	1526273.1	3206.9%	200.0
proposed Steiner(opt)	231.25	-1.9% (-2.9%)	863318.3	1770.5%	110.7
proposed Steiner (0.1%opt)	231.48	-1.8% (-2.8%)	55505.5	20.3%	2.8

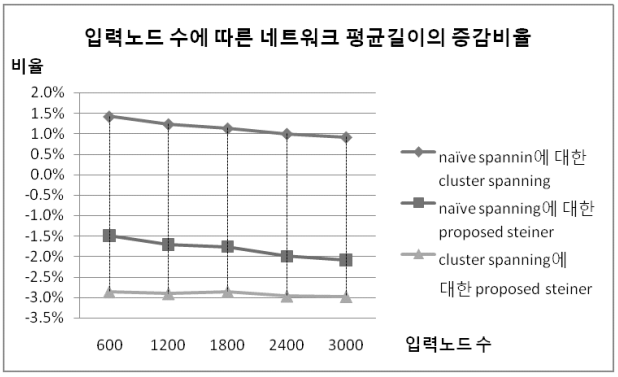
세계에서 다항식적 시간 내에 생성하는 것은 불가능하기 때문이다. 따라서 200번의 근사 스타이너 트리를 생성하는 가운데 최소 길이 D_{min}를 갖는 트리를 찾아 클러스터 센서 네트워크를 모델링하는 최종 근사 스타이너 트리로 출력하고 proposed Steiner(200)으로 표시한다. proposed Steiner(opt)는 200번 반복 생성되는 과정에서 첫 번째 생성되는 트리의 길이가 D_{min}인 트리이고, proposed Steiner(0.1%opt)는 트리 길이가 D_{min}보다 최대 0.1%길이가 추가된, 즉 $1.001D_{min}$ 이하인 트리들 중에서 첫 번째 생성되는 트리이다.

<표 1>에서 각 생성 방법은 naïve spanning과 비교되는데, 이것이 다항식적 시간 내에 최적의 해를 구할 수 있는 방법이기 때문이다. 본 논문에서 제안하는 방법인 proposed Steiner(opt), proposed Steiner(200)은 naïve spanning 방법보다 평균 1.9%의 트리 길이의 절감을 보였고, 노드 수가 3000, 클러스터의 수가 2, 최대 연결 길이가 2인 입력 환경에서, 제안된 방법으로 생성된 트리는 최대 3.7%의 트리 길이의 절감을 보였다. naïve spanning 방법은 입력 노드 전체에 대하여 다항적 시간 내에 최적화된 트리를 생성하기 때문에, 지역적 최적화(Local Minima) 전략을 사용하는 clustering spanning 방법보다 생성되는 네트워크의 길이가 감소된다. 본 논문에서 제안하는 방법인 proposed Steiner(opt), proposed Steiner(200), poposed Steiner(0.1%opt) 방법도 clustering을 통한 지역적 최적화를 일부 채용하지만, 그보다는 Steiner 트리라는 비다항식적 시간에서의 최적화를 목표로 휴리스틱이 구현되었기 때문에, 이 방법은 naïve spanning 방법보다는 네트워크 길이가 절감되는 것이다. proposed Steiner(0.1%opt) 방법은 실행 시간의 절감을 위해 proposed Steiner(opt) 방법의 네트워크 길이보다 최대 0.1% 증가된 길이의 네트워크로 생성된 것이므로 당연히 트리의 길이는 proposed Steiner (opt)보다는 증가된다. 실행시간 측면에서, cluster spanning 방법은 naïve spanning 방법에 비해 51.7%의 시간 절감율을 보인다. 전체 입력 노드를 N, 각 클러스터에 포함된 노드들의 평균 수를 n_a라고 할 때, Prim의 알고리즘을 적용하기 위한 노드들간의 완전 연결을 위한 시간이 $O(N) > O(n_a)$, 이므로 naïve spanning 방법이 clustering spanning 방법에 비해 실행 시간이 커지게 되는 것이다. 본 논문에서 제안하는 방법인 proposed Steiner(opt), proposed Steiner(200), proposed Steiner(0.1%opt)도 clustering을 통한 지역적 최적화를 일부 채용하지만, 그 실행 과정에서 근사 스타이너 트리를 위한 최소 신장 트리를 반복적으로 생성하고 이를 변형, 사용하므로, 단 한번의 신장 트리를 생성하는 naïve spanning이나 cluster spanning 방법보다 생성 시간이 많이 필요하게 된다. 200번의 반복 실행을 통해 네트워크를 생성한 후 최소 길이의 네트워크인 proposed Steiner(200)과 proposed Steiner(opt)는 동일한 네트워크로서, 모두 같은 최소의 네트워크 길이를 갖지만, 실행시간과 반복 회수는 다르다. proposed Steiner(0.1%)는 200번의 반복 실행 가운데, 최소 길이의 0.1%이하가 추가된 길이를 갖는 네트워크로, 최소 길이의 네트워크와 비교하여 길이의 차이는 미미하나, 실행시간은 naïve spanning tree에 비해 1770.5%에서 20.3%의 증가로 급속히 감소되었다. 이는 생성되는 트리들이 반복 생성되는 초기 단계에서 급속한 길이의 감소 변화를 보이고, 그 이후에 생성되는 트리들의 길이의 변화가 미세하기 때문이다. 본 논문의 실험에서 종료조건으로 설정한 근사 최소 스타이너 트리의 중복 생성 회수는 최대 200회이지만, 최소 길이의 트리들 중 최초로 생성하는 proposed Steiner(opt) 방법의 평균 반복 실행 회수는 110.7회이고, 최대 0.1%길이가 더해진 근사 스타이너 트리를 생성하는 proposed Steiner(opt) 방법의 평균 반복 회수는 평균 약 2.8회이다. 따라서 본 논문에서 제안하는 방법은 비교적 짧은 시간 내에 센서 네트워크의 길이의 절감이 필요한 응용에 잘 적용될 수 있음을 보인다.

(그림 8)은 연결 최대 길이에 따라 생성되는 네트워크의 길이를 비교한 것이다. 연결 최대 길이가 증가 함에 따라 본 논문에서 제안하는 방법인 proposed Steiner 방법은 naïve spanning 방법에 비해 네트워크 길이의 절감율이 점차 감소하지만 clustering spanning 방법에 비해서는 점차 증가함을 알 수 있다. 이는 연결 최대 길이가 증가하면, 각 입력 노드가 소속된 클러스터의 중심 노드가 반드시 근접한 클러스터의 중심 노드일 보장이 없으므로, 각 입력 노드의 클러스터 중심 노드를 사전에 할당하는 방법에서는, 클러스터링을 고려치 않는 naïve spanning 방법에 비해 최적화에 근접하는 것이 어렵기 때문이다. 그러나 같은 클러스터링 기법을 사용하는 경우라도, Steiner 트리를 생성하는 과정에서 지역적 최적화의 문제점을 어느 정도 상쇄할 수 있는, 본 논문에서 제안하는 방법이 clustering spanning 방법보다 네트워크 길이 절감율이 높아지는 것이다. (그림 9)에서는 제안된 방법이 입력 단말 노드의 수가 증가함에 따라 naïve spanning 방법에 비해 네트워크 길이의 절감율이 높아짐을 확인할 수 있다. 그 이유는 입력 노드의 수가 증가하면 네트워크 상에서의 밀도가 높아지게 되고, 결과적으로 클러스터링에 의한 지역적 최적화의 문제점이 점차 감소되기 때문이다. 따라서 입력 노드가 증가할 때, 같은 클러스터링 기법을 사용하는 clustering spanning과 제안된 proposed Steiner 방법의 naïve spanning 방법에 대한 네트워크 길이에 대한



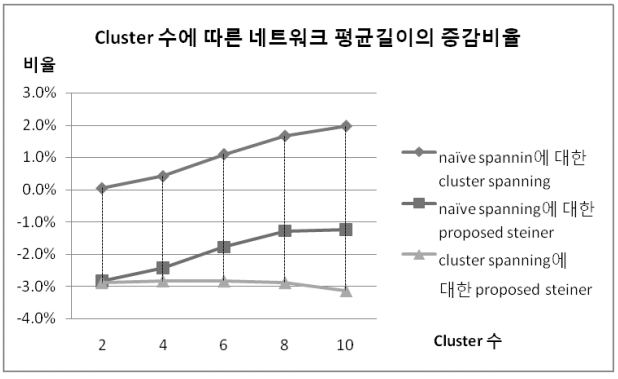
(그림 8) 연결 최대 길이와 네트워크 평균길이 증감비



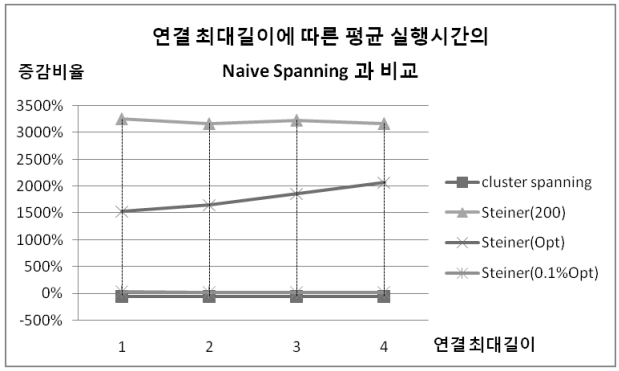
(그림 9) 입력 노드 수와 네트워크 평균길이 증감비

절감 추세는 유사함을 보일 것이다. cluster의 수가 증가할수록 클러스터링으로 인한 로컬 최적화의 문제점이 커질 것이다. 따라서 (그림 10)에서 나타난 것 같이, cluster 수가 증가할수록, 클러스터링 기법을 사용한 cluster spanning과 본 논문에서 제안된 방법은 naïve spanning 트리에 대한 네트워크 길이를 비교했을 때, 증가율이 점차 커짐을 확인할 수 있다.

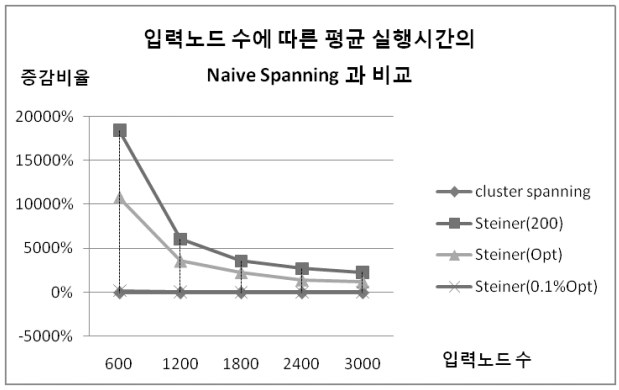
(그림 11)은 연결 최대 길이가 증가 함에 따라, naïve spanning 방법에 비해 제안된 Steiner(opt) 방법의 평균 실행 시간의 증가 비율이 높아지지만, cluster spanning는 평균 -51.7%, Steiner(0.1%opt)는 평균 20.3%로 거의 일정하며, Steiner(200) 경우도 평균 3206.9%로 거의 변화가 없다. 그 이유는 naïve spanning 방법과 cluster spanning 방법의 실행 시간은 연결 최대 길이와 무관하게 거의 일정한 실행 시간이 요구되기 때문이고, Steiner(0.1%opt)의 경우는 실행 회수가 작기 때문에, Steiner(200)의 경우는 미리 결정된 실행 회수인 200회를 반복하기 때문에 일정한 시간이 소모되는 것이다. Steiner(opt)의 실행 시간은 200회의 실행과정에서 최단 길이의 네트워크를 생성하는 시간인데, 이는 최대 연결 길이가 길어짐에 따라, 각각의 반복 실행되는 과정에서 생성되는 스타이너 트리의 변화가 상대적으로 심하기 때문에 최단 길이의 네트워크의 생성시간이 많아짐으로 분석될 수 있다. (그림 12)에서, 입력 노드의 수가 많을 수록 제안된 방법인 Steiner(200), Steiner(opt)의 실행시간이 naïve



(그림 10) 클러스터 수와 네트워크 평균길이 증감비



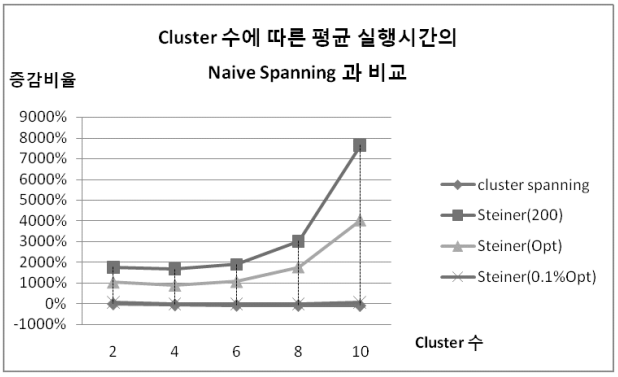
(그림 11) 연결최대 길이와 평균실행 시간 증감비



(그림 12) 입력 노드 수와 평균실행 시간 증감비

spanning 방법의 실행시간과 비교하여 점차 감소함을 알 수 있다. 이는 전체 입력 노드를 고려 대상으로 생성된 최소 신장 트리를 이용하는 naïve spanning 방법이 입력 노드의 수가 증가함에 따라 실행 시간이 매우 많이 필요하기 때문에, 클러스터링 기법으로 인하여, 노드 수의 증가가 상대적으로 영향을 덜 미치는 cluster spanning과 본 논문에서 제안하는 방법은, 실행시간이 naïve clustering 방법에 비해서 감소되는 것이다. 그리고 clustering spanning과 Steiner (0.1%opt) 방법의 경우는 절대적인 실행시간이 다른 방법에 비해 매우 작기 때문에 변화 량이 거의 없는 것으로 보인다. (그림 13)에서는 cluster 수가 많으면 지역적 최적화의 문제로 인하여 최적 치에 접근하는 시도를 여러 번 해야 하므로, cluster 수가 많을수록 제안된 방법 중 Steiner(200), Steiner (opt) 방법의 실행시간이 naïve spanning 방법에 비해 상대적으로 크게 증가함을 알 수 있다. 그러나 같은 clustering 기법을 도입하는 Steiner(0.1%opt)와 naïve spanning 방법도 이러한 영향을 받을 수 있으나, 실행시간이 다른 방법에 비해 절대적으로 작으므로 의미 있는 변화가 없음으로 분석된다.

이 실험 결과를 종합하면, 본 논문에서 제안하는 방법은 네트워크의 길이 측면과 네트워크 생성 시간 측면에서 모두 naïve spanning 방법에 비해, 최대 연결 길이가 작을수록, 입력 노드가 많을 수록, 클러스터의 수가 적을수록 좋은 성능을 보인다. 또한 clustering spanning 방법과 비교한다면,



(그림 13) Cluster 수와 평균 실행 시간 증감비

제안하는 방법은 최대 연결 길이가 길수록, 네트워크의 길이 측면에서 약간 좋은 성능을 보이며, 실행시간 측면에서는 최대 연결길이가 짧을 수록, 입력 노드가 많을수록, cluster 수가 적을수록 상대적으로 좋은 성능을 보인다.

5. 결 론

클러스터 센서 네트워크는 입력 노드들이 특정 노드들을 중심으로 클러스터를 형성하여 같은 클러스터내의 다른 노드들과 빈번하게 통신하고 필요에 따라 다른 클러스터의 노드들과 가끔 통신하는 센서 네트워크이다. 따라서 전체 노드가 아닌 해당 클러스터내의 노드들을 대상으로 한 트리를 효과적으로 구성한다면, 통신을 위한 라우팅 작업 시 네트워크 전체 트래픽의 감소, 전송 거리의 단축, 신속하고 정확한 정보의 전달 등이 가능할 수 있다. 최소 스타이너 트리는 특정 노드가 입력으로 주어졌을 때, 스타이너 포인트라는 새로운 노드들을 도입하여 모든 입력 단말 노드들을 최소 비용으로 연결하는 트리이다. 현실 세계에서 다항식적 시간 내에 최적의 해를 구할 수 있는 최소 신장 트리보다 이러한 최소 스타이너 트리의 비용은 적지만, 이를 구성하는 것은 비다항식적 시간(Non-Polynomial Time) 문제에 속하므로 이를 위한 알고리즘은 아직 발표되지 않았다.

본 논문에서는 클러스터 센서 네트워크의 신속한 구축을 위해, 각 클러스터 내에 존재하는 모든 노드들을 완전 연결하는 간선과, 각 클러스터들을 최소 거리로 연결하는 간선을 생성하여 이를 이용하여 최소 신장 트리를 구축하고 이 트리내의 노드와 간선를 이용하여 스타이너 포인트들을 적절히 생성하고, 이를 이용하여 근사 최소 스타이너 트리를 반복적으로 생성하는 방법을 제안한다. 이 방법은 모든 입력 단말 노드를 완전 연결하여 간선을 생성하는 유클리드 최소 신장 트리를 활용하여 구축한 naive spanning 방법의 센서 네트워크와 비교할 때, 생성 시간은 1770.5%의 증가했지만 네트워크 길이를 최대 3.7%, 평균 1.9% 감소시켰다. 주목할만한 점은 naïve spanning 방법의 평균 실행시간에 20.3%의 시간 추가를 했을 경우, 네트워크 길이를 naïve spanning 방법과 비교하여 평균 1.8% 감소시킨 점이다. 이는 본 논문에서 제안하는 방법의 문제점인 실행 시간이 큰 것에 대해서, 근사 최적화 길이보다 0.1% 증가된 네트워크의 길이를 허용한다면 naïve spanning 방식에 비해 비교적 크지 않은 추가 시간으로, 클러스터 센서 네트워크를 구축할 수 있음을 보이고, 결과적으로 클러스터 ad-hoc 센서 네트워크, 클러스터 이동 센서 네트워크 등과 같은 실시간 응용에 적용할 수 있는 가능성을 보인다. 본 논문에서 제안하는 방법은, 짧은 연결 최대 길이를 갖는 많은 수의 입력 단말 노드들이 적은 수의 클러스터들에 소속되어있는 환경에서, naïve spanning 방법으로 생성된 센서 네트워크 보다 높은 네트워크 길이의 절감율을 얻을 수 있고, 또한 실행 시간적인 측면에서 상대적으로 긍정적인 특성을 나타낸다.

향 후 연구는, 네트워크 길이 및 실행시간의 감소를 위한 제안된 방법의 개선이다. 또한 클러스터 센서 네트워크의 실 세계 응용에서 중요한, 매우 많은 센서 노드들을 효과적으로 처리하기 위해 다항적 시간 근사구조 (Polynomial Time Approximation Scheme)의 적용에 관하여 연구할 예정이다[9].

참 고 문 헌

[1] J. Al-Karaki and A. Kamal, “Routing Techniques in Wireless Sensor Networks: A Survey,”IEEE Wireless Communications, Vol.11, pp.6-28, 2004.

[2] L. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” IEEE Communications Magazine, pp.102-114, 2002.

[3] M. Sanchez, P. Manzoni and Z.J. Hass, “Determination of Critical Transmission Ranges in Ad Hoc Network,” Proceedings of MMT, 1999.

[4] J. Li, D. Cordes and J. Zhang, “Power aware Routing Protocols in Ad Hoc Wireless Sensor Networks,”Wireless Communications, IEEE Vol.12, No.6, pp.69-81, 2005.

[5] H.O. Tan and I. Korpeoglu, “Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks,”ACM SIGMOD Record Vol.32, No.4, pp.66-71, 2003.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, ‘Introduction to Algorithm’, 2nd Ed., MITPress, 2001.

[7] http://en.wikipedia.org/wiki/Steiner_tree, 2009.

[8] A. Hayrapetyan, C. Swamy and E. Tardos, “Network Design for Information Networks,”Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.933-942, 2005.

[9] J. Kim, M. Cardei, I. Cardei and X. Jia, “A Polynomial Time Approximation Scheme for the Grade of Services Steiner Minimum Tree Problem,” Journal of Global Optimization, Vol.24, pp.437-448, 2002.

[10] 김재각, 김인범, 김수인, “원격 검침 시스템에서 근사 최소 스타이너 트리를 이용한 집중기 및 중계기의 효율적인 배치와 연결”, 한국통신학회 논문지B, Vol.34 No.10, pp.994-1003, 2009.

[11] 서민석, 김대철, “스타이너 트리 문제를 위한 Mar-Min Ant Colony Optimization”, 경영과학, Vol.26, No.1, pp.65-76, 2009.
[12] 이승관, “멀티캐스트 라우팅 문제 해결을 위한 엘리트 개미 시스템”, 한국컴퓨터정보학회논문지, Vol.13, No.3, pp.147-152, 2008.
[13] 이성근, 한치근, “다중 제약이 있는 멀티캐스트 트리 문제에 관한 연구”, 한국인터넷정보학회논문지, Vol.5, No.5, pp.129-138, 2004.
[14] 김준모, “센서 네트워크 구축에서의 Combinatorial 기법 적용”, 대한 전자공학회 논문지TC, Vol.45, No.7, pp.9-16, 2008.
[15] C. Liu, S. Yuan, S. Kuo, S. Wang, “High-performance Obstacle-avoiding Rectilinear Steiner Tree Construction,” ACM Transactions on Design Automation of Electronic Systems, Article 45, Vol.14, No.3, 2009.
[16] Y. Wang, X. Hong, T. Jing, Y. Yang, X. Hu, G. Yan , “The Polygonal Contraction Heuristic for Rectilinear Steiner Tree Construction,” Proceedings of the 2005 Asia and South Pacific Design Automation Conference, Shanghai, pp.1-6, 2005.
[17] B. Bell, “Steiner Minimal Tree Problem,” http://www.css.taylor.edu/~bbell/steiner/, 1999.

김 인 범

e-mail : ibkim@kimpo.ac.kr
1989년 서울대학교 컴퓨터공학과(공학사)
1991년 서울대학교 공과대학원 컴퓨터공학과(공학석사)
2007년 University of Wisconsin-Milwaukee, Department of Computer Science (PhD)
1991년~1996년 대우통신 종합연구소, Oracle 코리아
1996년~현 재 김포대학 IT학부 부교수
관심분야 : 그래프 및 네트워크 알고리즘, 컴퓨터 이론, 데이터베이스 등