계층적 무선 센서 네트워크에서의 연속 범위 질의들의 점진적 처리

이정훈, 황규영, 임효상

CS/TR-2008-297

December 2008

K A I S T

Department of Computer Science

최종 수정일: 2008년 11월 22일

계층적 무선 센서 네트워크에서의 연속 범위 질의들의 점진적 처리

Progressive Processing Continuous Range Queries in Hierarchical Wireless Sensor Networks

이정훈, 황규영, 임효상

Jeong-Hoon Lee, Kyu-Young Whang, Hyo-Sang Lim

한국과학기술원 전산학과

Department of Computer Science and

Korea Advanced Institute of Science and Technology (KAIST)

연구(투고) 분야: 데이타베이스

키워드: 센서 네트워크, 범위 질의 처리

e-mail: {jhlee_s, kywhang}@mozart.kaist.ac.kr, hslim@cs.purdue.edu

계층적 무선 센서 네트워크에서의 연속 범위 질의들의 점진적 처리

Progressive Processing Continuous Range Queries in Hierarchical Wireless Sensor Networks

요 약

본 논문은 계층적 무선 센서 네트워크에서의 연속 범위 질의 처리 문제를 다룬다. 최근 유비쿼터스 환경의 발전과 무선 네트웍의 발전에 힘입어 센서 네트워크의 규모가 커짐에 따라, 이를 계층적으로 구성하는 방법이 실용적인 대안으로 대두되고 있다. 이 방법은 플랫한(flat) 구성 방법이 동일한 능력을 가진 센서 장치들로 네트워크를 구성하는 것과는 대비되는 방법으로, 서버에 더 가까운 층(즉, 상위층)일수록 더 큰 능력을 가진 센서 장치를 배치하는 방법이다. 현재까지 플랫한 센서 네트워크에서의 질의 처리 문제는 많이 다루어지고 있는 반면, 계층적 센서 네트워크에 대해서는 연구가 미흡한 상황이다. 무선 센서 네트워크의 구성에 있어서 주요하게 고려해야 할 비용은 데이터를 전송하기 위해 필요한 에너지와 질의를 저장하기 위한 스토리지다. 이 비용들은 서로 trade-off 관계에 있다. 본 논문에서는 먼저 계층적 센서 네트워크에서 다수의 연속 범위 질의들을 효율적으로 처리하는 점진적 질의 처리 방법을 제안한다. 이 방법은 기본적으로는 Xiang et al.의 query merging 기술을 사용하되, 에너지와 스토리지의 trade-off도 고려한다. 즉, 하위층에서는 더 많은 질의들을 병합하여 스토리지 비용을 줄이고, 상위층에서는 더 적은 질의들을 병합하여 false alarm을 줄여서 에너지 비용을 줄인다. 다음으로, 스토리지와 에너지의 비용에 주어진 weight에 따라, 센서 네트워크를 두 측면에서 모두 최적화하여 구성하는 방법을 제시한다. 이 방법은 네트워크 환경과 응용에 따라 다른 중요도를 가지는 메모리 필요량과 데이터 전송량의 trade-off를 systematic하게 조정하여 최소 비용의 네트워크를 구성할 수 있도록 한다. 마지막으로, 실험을 통해 본 논문에서 제시하는 방법이 실제로 스토리지와 에너지를 near-optimal하게 조정할 수 있음을 보이고, 분석적으로 제시한 최적의 수치와 실험적으로 제시한 수치간에 큰 차이가 없음을 보임으로써 이론적 모델의 정확도를 보인다. 그리고, 기존의 방법과 비교를 통해서 제안하는 방법이 약 1.019 $\sim$ 2.666배의 절감된 비용으로 계층적 센서 네트워크를 구성함을 보인다.

1 Introduction

컴퓨팅 환경의 유비퀴터스화에 따라 센서 네트워크에 대한 관심과 연구가 활발히 일어나고 있다. 센서 네트워크 환경에서 센서 노드들은 센싱된 데이터를 수집하는 서버(혹은 base station)와 네트워크를 통해서 연결된다[1].

이러한 센서 네트워크를 활용한 응용의 대표적인 예는 온도/습도등의 환경 모니터링, 공장에서의 생산 공정 추적, 교통 상황 모니터링, 경비 시스템에서의 침입자 탐지 등을 들 수 있다. 특히 최근에는 무선 네트워크가 일상화 됨에 따라, 배치(deploy) 비용을 줄이기 위하여 센서 노드들을 ad-hoc 네트워크 형태로 무선으로 연결하여 서버까지 데이터를 전송하는 무선 센서 네트워크에 대한 연구가 많이 이루어지고 있다. 무선 센서 네트워크에서는 일반적으로 제한된 자원(e.g., battery 용량,스토리지 용량)을 가진 값 싼 센서들을 배치하고 이를 사용하여 최대한 오랜 기간동안 데이터를 수집하는 것을 목표로 하기 때문에 제한된 자원을 최대한 효율적으로 활용하여 질의 처리를 수행할 수 있도록 하는 것이 중요한 연구 이슈이다.

최근 센서 네트워크의 응용 범위가 도시 관리, 전 지구적 환경 모니터링 등으로 광범위하게 확장됨에 따라 무선 센서 네트워크의 규모가 증가하는 경향을 보이고 있다. 다시 말해, 다수의 사용자에 의해 사용될 목적으로 광범위한 영역에 다수의 노드들을 배치하는 센서 네트워크들이 나타나고 있다[14, 3]. 예를 들어, NOVAC(Network for Observation of Volcanic and Atmospheric Change) Project[14]의 경우, 5개 대륙에 있는 15 화산에 배치된 무선 센서 네트워크들이 다층(multi-tier) 구조로 서로 연결되어서, global한 화산 모니터링 연구를 지원하고 있다. 또 다른 예로, EarthNet Online[3] 경우, 세계의 날씨와 조류의 이동등과 같은 전 지구적인 관찰 정보를 무선 센서 네트워크를 통해서 모으고, 수천여 명의 개인과 기관에 유용한 정보로써 제공하고 있다. 이와 같이 규모가 증대됨에 따라, 시스템에서 동시(concurrent) 수행되는 질의의 개수와 센서 데이터의 양 역시 비례하여 증가하고 있다. 따라서, 이러한 large-sacle의 질의와 데이터를 효율적으로 처리할 수 있는 무선 센서 네트워크를 어떻게 하면 저비용으로 구축할 수 있을까 하는 연구가 그 중요성을 더해 갈 것으로 예상된다.

본 논문에서는 센서 네트워크를 구성하는 비용으로서 각 센서 노드에서 질의를 저장하기 위해 필요한 스토리지와 수집된 데이터를 서버까지 보내기 위해 필요한 에너지(즉, battery 용량)를 중요하게 다룬다. 이 두가지 요소는 서로 trade-off 관계에 있다. 센서 네트워크를 구성하는 가장 naive한 두 가지 방법[17]인 1) centralized approach와 2) distributed approach를 통해서 이러한 trade-off를 설명한다.먼저,centralized approach는 센서 노드들은 수집한 모든 데이터를 서버로 전송하는 역할만 수행하고 서버가 질의처리를 수행하는 방법이다. 이 방법은 센서 노드에 질의를 저장할 필요가 없어 스토리지 비용을 줄일 수 있다는 장점이 있지만 모든 데이터를 전송해야 하기 때문에 에너지 소비가 크다는 단점이 있다. 다음으로, distributed approach는 센서 노드에 모든 질의들을 저장하여 질의 처리를 수행하고 서버로는 결과만을 전송하는 방법이다(이 방법은 *in-network 질의 처리*라 불리기도 한다). 이 방법은 수집된 데이터 중 일부만 서버로 보내기 때문에 에너지 소비는 줄일 수 있으나 질의를 저장하기 위해서 스토리지가 많이 필요하다는 단점이 있다.

이 두가지의 naive한 방법은 스토리지와 에너지 비용 측면에서 고려했을 때, 대규모의 무선 센서 네트워크를 구성하기 위해 사용되기에는 현실적인 어려움이 있다. Centralized approach를 사용했을 때, 서버에 가까이 위치한 센서 노드들은 서버로부터 멀러 떨어진 노드들이 전송한 데이터들까지 누적하여 전송하므로, 상대적으로 더 많은 데이터를 전송한다. 이러한 현상은 특히 센서 노드의 개수가 증가가 증가할수록 그 경향이 급격히 커진다. 즉, 서버에 가까운 센서 노드들은 서버로부터 멀리 떨어진 노드들로부터 전송되어 온 데이터까지 전송하기 위해 더 많은 에너지를 소모하게 되어 급격히 burnout 되기 때문에 대규모의 센서 네트워크를 구성하는데 적합하지 않다.반면에 distributed approach는 질의의 개수가 많아지면 현실적으로 사용할 수 없다.센서노드는 자신이 가진 제한된 자원과 컴퓨팅 파워로 인해, 다수의 질의를 처리할 수 없다. 8∼128 Kbyte의 플래시 메모리와 0.5∼8 Kbyte의 RAM을 가진 값싼 센서 노드인 Micamote[15]를 예로 들어 보자. 만약 64Kbyte의 플래시 메모리에서 10%를 이차원 범위 질의들을 저장 하기 위해 사용한다고 하자. 그리고, 질의를 구성하는 각 attribute의 value가 8byte의 실수이고, 질의의 selection 조건은 $constant_1$ op_1 *attribute* op_2 $constant_2$로 표현된다고 가정 하자: *attribute*는 어떤 attribute의 이름, $constant_1$과 $constant_2$는 실수 값들, 그리고 op_1과 op_2는 binary comparison 연산자이다(따라서, 질의 1개의 크기는 적어도 32 byte 이상임). 그러면, 약 200개의 질의만이 Micamote 1개에 저장될 수 있다. 즉, 모든 센서 노드가 모든 질의를 저장할 수 있을 만큼 충분한 메모리를 가져야 하는데 이는 고비용을 초래하므로 대규모 센서 네트워크를 구성하는데 적합하지 않다.

최근, 이러한 large-scale 문제를 해결하기 위해 무선 센서 네트워크를 계층적으로 구성하는 방법이 실용적인 대안으로 대두되고 있다. 계층적 무선 센서 네트워크는 서로 다른 수준의 자원과 컴퓨팅 파워를 가진 센서 노드들의 다층(multi-tier) 구조로 구성된다[21]. 서버에 가깝게 위치한 노드일수록 더 멀리 떨어진 노드보다 더 많은 자원과 컴퓨팅 파워를 가지며, 이러한 특성이 저성능의 노드들만으로는 할 수 없는 질의 처리를 가능하게 한다. 계층적 무선 센서 네트워크에서는 더 작은 자원과 컴퓨팅 파워를 가진 노드들이 더 많은 자원과 컴퓨팅 파워를 가진 노드에 재귀적으로(recursively) 연결된다[18, 16, 21].따라서,서버에 인접한 노드들은 하위층으로부터 누적되어 전송된 대용량의 데이터를 처리할 수 있도록 설계되어 있다. 우리는 이러한 특성이 위에서 언급한 대규모 센서 네트워크에서 질의 처리 문제를 해결하는데 적합한 특성이라고 생각한다. 그러나, 현재까지는 플랫한 센서 네트워크(즉, 동일한 능력을 가진 센서 노드들로 이루어진 센서 네트워크)에서의 질의 처리 연구가 주류를 이루고 있을 뿐, 계층적 센서 네트워크에서의 연구는 충분하지 않다.

본 논문에서는 에너지와 스토리지의 trade-off에 주목해서 효율적으로 질의 처리를 수행할 수 있는 대규모 계층적 센서 네트워크를 구성하는 방법을 제시한다. 특히, 본 논문에서는 연속 범위 질의를 주 대상으로 한다. 많은 센서 네트워크 응용에서, 특히 모니터링에서, 범위 질의는 중요한 질의 형태로 사용되고 있으며[10], 질의 처리 성능을 향상시키기 위해 인덱스를 사용하여 결과 전송을 위해 거치는 hop수를 줄이기 위한 방법등의 연구가 활발히 수행되어 왔다[8]. 본 논문에서 제안하는 방법은 유사한 범위를 가지는 질의들을 병합하여 개수를 줄임으로써 에너지 소비량과 스토리지 사용량의 trade-off를 systematic하게 조정하는 기법에 기반을 둔다. 질의 결과의 중복 전송으로 인한 에너지 소비를 방지하기 위해 질의들을 병합하여 처리함으로써 에너지 소비량을 줄이는 방법은 이미 기존연구[12, 22, 23]에서 제시된 바 있으나, 이러한 연구들은 플랫한 센서 네트워크에 초점을 두고 있기 때문에 층마다 센서 노드의 능력에 차등을 두는 계층적 센서 네트워크의 특성을 활용하지 못한다. 또한 storage 사용량을 고려하지 않아, trade-off를 똑바로 반영하지 못하였다. 반면, 본 논문에서는 서버로부터 하위층으로 갈수록 더 많은 질의들을 병합하여 저장하고 처리하는 점진적인 방법을 사용하여 계층적 센서 네트워크의 특성을 최대한 활용한다. 그리하여, 에너지 소비량과 storage 사용량의 trade-off를 optimize한다.즉, 개수가 많은 하위층의 센서 노드에서는 질의들을 더 많이 병합하여 질의 개수를 줄임으로써 스토리지의 필요량을 줄이고, 개수가 작은 상위 노드에서는 질의들을 더 적게 병합하여 많은 질의들을 저장하는 대신 정확도를 높여서 누적된 데이터를 더 많이 필터링함으로써 에너지 소비량을 줄인다.

먼저, 본 논문에서는 점진적 질의 처리 방법의 모델과 알고리즘을 제시한다. 제안하는 점진적 질의 처리 방법은 크게 1) 질의 병합(query merging)과 2) 질의 결과의 필터링(result filtering)으로 이루어진다. 질의 병합의 주요한 아이디어는 서버로부터 최하위 층으로 가면서 점진적으로 병합을 수행한다는 것이다. 다시 말해, 입력된 질의들을 병합하여 2번째 층에 위치한 센서 노드에 저장될 질의들을 구성하고, 다시 이들을 병합하여 3번째 층에 위치한 센서 노드에 저장될 질의들을 구성하는 식으로 층별로 재귀적으로 질의 병합이 수행된다. 층별로 저장될 병합된 질의들의 집합은 센서 네트워크의 각 층에 위치한 노드들에게 전송되어, 전체적으로는 역 계층 질의 구조*(inverted hierarchical query structure)*를 생성하게 된다.

*역 계층 질의 구조(inverted hierarchical query structure)*는 질의들의 범위를 저장한 다차원 색인을 레벨별로 분할(partition)하여 센서 네트워크의 최하위층 노드들은 색인의 루트 노드를, 서버는 색인의 최하위 레벨을 저장하는 구조로 본 논문에서 새로이 제안하는 구조이다.이는 계층적 센서 네트워크가 일반적인 트리 형태의 색인 구조에서와는 반대로 상위 층의 센서 노드가 더 자세한 정보를 가지고 있고, 하위층으로 갈수록 abstract된 정보를 가지고 있다는 특징에 착안한 데이터 구조이다. 질의 결과의 필터링에서는 하위층으로부터 상위층으로 데이터를 전송하면서, 더 정확한 결과를 얻는 방법으로 질의 처리를 점진적으로 수행한다. 이때, 각 층에서의 질의 결과를 찾아내기 위해서 역 계층 질의 구조를 사용함으로써, 마치 다차원 색인을 루트로 부터 하위 레벨까지 한번 검색하는 것과 동일한 효과를 얻음으로써, 보다 효율적으로 질의 처리를 수행할 수 있도록 한다.

다음으로, 스토리지와 에너지에 각각에 주어진 weight에 따라 두 비용 간의 trade-off를 systematic하게 조정하여 계층적 센서 네트워크를 optimal하게 구성하는 방법을 제시한다. 스토리지와 에너지 비용의 중요성은 응용과 환경에 따라 달라질 수 있으므로, 본 논문에서는 네트워크의 구성 비용을 스토리지 비용과 에너지 비용의 weighted sum 형태로 표현하고 이를 최적화한다.최적화 parameter로서 계층적 센서 네트워크의 각 층에서 질의들을 어느 정도 병합해야 하는지를 결정한다.

마지막으로, 실험을 통해서 본 논문에서 제시하는 방법이 비용면에서 효율적인 계층적 센서 네트워크를 구성하는데 유용함을 보인다. 먼저, 본 논문에서 분석적으로 제시한 최적의 수치가 실험적으로 구한 수치와 큰 차이가 없음을 보인다. 그리고, 기존 플랫한 센서 네트워크에서의 질의 처리 방법과의 비교를 통해 본 논문에서 제시하는 방법이 총 비용(weighted sum)면에서 우수함을 보인다.

본 논문의 구성은 다음과 같다. 먼저, 제 2장에서는 관련연구들을 간략히 제시한다. 제 3장에서는 계층적 무선 센서 네트워크에서의 점진적 질의 처리 방법의 모델과 알고리즘을 제시한다. 제 4장에서는 스토리지와 에너지 비용의 trade-off를 고려하여 계층적 센서 네트워크를 효율적으로 구성하는 방법을 분석적으로 제시한다. 제 5장에서는 실험을 통하여 본 논문에서 제시하는 방법의 우수성을 보인다. 마지막으로, 제 6장에서 결론을 제시한다.

2 Related Work

본 장에서는 관련연구로서 센서 네트워크에서의 연속 범위 질의 처리에 관한 기존 연구와 계층적 무선 센서 네트워크의 현황에 대해서 설명한다.

2.1 센서 네트워크에서의 연속 범위 질의 처리

센서 네트워크에서의 질의 처리 연구는 크게 시스템에서 한번에 단 하나의 질의만을 처리하는 단일 질의 처리와 여러개의 질의들을 동시에 처리하는 멀티플(multiple) 질의 처리로 나눌 수 있다.

단일 연속 범위 질의 처리

Li et al. [8]의 연구에서는 data-centric storage[17]를 센서 네트워크에 적용하여 단일의 연속 범위 질의를 처리하는 방법을 제안하였다. Data-centric storage를 이용한 질의 처리 방법은 수집된 데이터를 값에 따라서 정해진 센서 노드로 보내고, 질의 처리시 그 결과를 가진 센서 노드들로만 질의를 전달하여 처리하는 방법이다. 제안하는 방법에서는 저장된 데이터 값들의 범위가 인접하는 노드들은 물리적으로도 인접하도록 order-preserving hash를 사용하여 데이터를 분배함으로써, 질의와 결과를 전달하기 위해 거쳐야 하는 노드의 hop수를 최소화하여 질의 처리 비용을 줄였다. Madden et el. [10]은 data-centric approach와는 다르게, 각각의 노드가 자신이 센싱한 데이터를 저장하는 환경을 고려하였다. 이 방법은 데이터 값들의 범위를 기반으로 R-tree와 유사한 색인인 *SRTree*를 생성하는 방법을 제안하였다. 또한 센서 노드의 에너지 소비량을 줄이기 위해, predicate을 reordering하는 방법을 제안하였다. 하지만, 이러한 연구들은 하나의 센서 네트워크에서 하나의 질의만을 처리하는 것을 가정함으로써 오늘날과 같이 많은 사용자들이 동시에 센서 네트워크에 질의를 등록하고 그 결과를 얻는 환경에는 부적합하다는 단점이 있다.

멀티플 연속 범위 질의 처리

Ratnasamy et al. [17]의 연구에서는 센서 네트워크에서의 멀티플 질의 처리를 위한 가장 기본이 되는 두가지 방법을 제안하였다. 두가지 방법은 각각 서버에서 질의를 처리하는 방법(즉, centralized approach)과 센서 노드에서 질의를 처리하는 방법(즉, distributed approach)이다. 전자의 방법에서는, 질의들이 서버에 저장되어 있고, 센서 노드에서 센싱된 모든 데이터들이 질의 처리를 위해 서버까지 전송된다. 이 방법은 질의의 영역 전체를 union한 영역이 전체 data space에 근사할 때만 효과적이며, 그렇지 않을 때는 서버로 불필요한 데이터를 보내는 overhead가 발생한다. 즉, 이 방법은 센서 노드에서는 질의를 저장하지 않기 때문에 각 노드의 메모리 필요량은 줄일 수 있으나 모든 데이터를 서버로 전송해야 하기 때문에 에너지 소비량이 크다는 단점이 있다. 후자의 방법에서는, 모든 질의들이 센서노드로 전송되어 저장되고, 센서 데이터는 노드별로 처리되어 결과만이 서버로 전송된다. 이 방법은 전자의 방법에서와 같은 문제는 피할 수 있지만, 저장해야 할 질의들의 개수가 많은 경우 메모리 부족으로 인하여 이를 다 저장할 수 없는 경우가 발생할 수 있다는 단점이 있다. 즉, 이 방법은 질의의 조건에 맞는 데이터만을 서버로 보냄으로써 에너지 소비량을 줄일 수 있으나 다수의 질의를 처리하기 위해서 스토리지 필요량이 커진다는 단점이 있다. 우리는 이러한 두 가지 기본적인 질의 처리 방법으로 부터 센서 노드의 주요한 두 자원인 메모리와 에너지 사이에 trade-off가 있음을 관찰 할 수 있다.

보다 최근에는 이러한 기본적인 방법인 centralized approach와 distributed approach를 보완하고자 하는 연구가 있었다. 특히, 이 방법들은 여러 개의 질의들 간에 질의 조건이 겹치는 영역들이 있을 경우 겹침 영역에 대한 질의 처리를 공유하는 방법이다. 이 방법들은 먼저 사용자에 의해 주어진 질의들 중에 질의 영역이 겹치는 것들을 식별하고, 이러한 질의들을 재작성(rewriting)함으로써, 중복 처리와 중복 데이터 전송을 제거하였다.이러한 방법들은 크게 *분할(partitioning)* 방법과 *병합(merging)* 방법으로 구분할 수 있다.

분할 방법에서는 서버가 각각의 질의 영역들을 겹치는 부분과 그렇지 않는 부분으로 분할한다. 그리고, 분할된 영역들과 원 질의를 함께 센서 노드로 전송하여 저장한다. 질의 처리는 각 분할된 영역에 대해 수행되고, 질의 결과는 서버나 센서 노드에서 병합된다. Trigoni et al. [20] amd Yu et al. [24]은 센서 노드의 *위치정보*에 대한 범위 질의를 처리하기 위해 이 방법을 사용하였다. 이 방법은 각 파티션별로 처리된 결과들을 병합하면 원 질의의 처리 결과와 동일하여, false positive가 발생하지 않는다는 장점이 있다. 그러나, 겹치는 조건이 많은 경우 특정 센서 노드에 저장해야 하는 파티션의 수가 증가하게

되어 필요한 스토리지가 커질 수 있다는 단점이 있다.

병합 방법에서는 서버가 겹치는 영역이 있는 질의들을 하나의 질의 영역으로 병합한다. 그리고, 병합된 질의를 센서 노드로 전송하여 저장한다. 질의 처리는 먼저 병합된 질의들에 대해서 수행되고 다시 이로부터 원 질의의 결과로 "재구성"되는 과정을 거친다. 이 방법은 센서 노드에 저장될 질의 개수를 줄임으로써, 다수의 질의를 동시에 처리할 수 있다는 장점을 가진다. 그러나, 질의들을 병합함으로써, false alarm이 발생하는 문제가 있다. Miuller and Alonso[12]는 범위 질의의 predicate들을 비교하여, 모든 질의들에서 공통으로 가지는 predicate들을 추출하고, 공통의 predicate들만을 질의 조건으로 가지는 단 하나의 질의를 구성하는 방법을 제안하였다. 이 방법에서 모든 질의들이 공통으로 가지는 predicate이 없는 경우, 질의 조건이 없는 단일 질의를 생성한다. 따라서, 이런 경우에는 false alarm이 많이 발생하는 문제가 있다. Xiang et al. [22, 23]은 겹치는 질의 영역들을 점증적(incremental)으로 병합하고, 병합 결과로 생성된 질의들을 원 질의들 대신 처리하는 방법을 제안하였다. 여기서, 점증적 병합은 질의들을 병합하였을 때 발생하는 false alarm들을 전송하기 위한 비용이 병합하지 않았을 때의 각각의 질의의 겹치는 부분의 결과를 중복 전송하기 위한 비용보다 크지 않을 때까지 수행된다. Xiang et al.의 질의 처리 방법은 centralized 방법와 distributed 방법의 장점을 동시에 취한 hybrid 방법으로서 (즉, 메모리 필요량과 데이터 전송량을 줄이는 방법) 의미가 있지만, 모든 센서 노드들의 능력이 동일한 "플랫한" 센서 네트워크를 대상으로 하고, 모든 센서 노드들이 동일하게 병합된 질의 집합을 저장함으로써 계층적 센서 네트워크의 특징을 활용하지 못하는 단점이 있다. 본 논문은 Xiang et al.의 방법과 마찬가지로 병합 방법을 사용하지만, 이를 보다 발전시켜서 계층적 센서 네트워크에서 각 센서 노드의 능력에 따라서 질의 병합의 정도를 조절함으로써 메모리 필요량과 데이터 전송량의 요구에 따라 센서 네트워크를 구성할 수 있다는 장점이 있다. 즉, 이 방법은 메모리 필요량과 데이터 전송량의 trade-off를 systematic하게 조정하는 방법을 제공할 수 있다는 장점을 가진다.

2.2 계층적 무선 센서 네트워크

센서 네트워크의 규모가 커짐에 따라 모든 센서 노드가 동일한 능력을 가지는 플랫한 구조에서 필요에 따라 각 계층에 서로 다른 능력을 가진 센서 노드들을 배치하는 계층적 구조가 실제 응용에서 많이 사용되고 있다[21].

이러한 계층적 무선 센서 네트워크의 대표적인 예로는 COSMOS[18]에서 언급한 센서 네트워크인 PASTA(Power Aware Sensing, Tracking and Analysis)와 SOHAN[4]을 들 수 있다. PASTA는 적의 움직임을 감시하기 위한 군사 응용에서 사용되는 센서 네트워크로 서버와 20여개의 단말 센서 노드들을 묶어서 관리하는 400여개의 중간 층 노드들로 구성된다. SOHAN은 도로 가장자리에 배치된 센서 노드들을 사용하여 교통량을 측정하는 교통 체증 모니터링 응용에 사용되는 센서 네트워크로, 서버와 200여개의 단말 센서 노드들을 묶어서 관리하는 50여개의 중간 층 노드들로 구성된다.

이러한 계층적 센서 네트워크 구조는 응용의 규모와 요구가 커짐에 따라 앞으로 점점 더 활발히 활용될 것으로 보인다. 그러나, 아직까지 각 층 별로 센서 노드가 서로 다른 능력을 가진다는 특성을 활용하여 멀티플 질의 처리를 수행하는 것에 대한 본격적인 연구가 이루어지지 않고 있다. Srivastava et al. [19]은 계층적 센서 네트워크에서 질의 처리를 수행할 때 각 질의 연산들을 어떻게, 어떤 노드에서 수행할지에 대한 연구를 하였다. 그러나, 이 연구는 주로 단일 질의 처리에 대해서 다루고 있기 때문에 멀티플 질의 처리에 적용하기 어려운 점이 있다. 본 연구에서는 계층적 센서 네트워크의 특징(즉, 서로 다른 자원과 컴퓨팅 파워를 가진 센서 노드들로 구성된 다층 구조[21])을 활용하여 멀티플 질의 처리를 효율적으로 수행하는 방법을 제안한다.

기타 계층적 무선 센서 네트워크에서의 연구로는 다음과 같은 것들이 있다. 무선 센서 네트워크 응용들이 점차적으로 다양해 짐에 따라, 복잡한 연산과 대량의 데이터를 처리하고자 하는 요구가 커지고 있다. 보안 감시나 교통 체증 모니터링을 위해 대량의 데이터를 처리하는 센서 네트워크들이 그 대표적인 예이다[18]. 이런 센서 네트워크에서는 카메라가 설치된 센서 노드들이 이미지 데이터를 전송한다. 이런 이미지 데이터는 일반적으로 크기가 커서 모든 데이터를 서버로 전송하는 것은 적합하지 않기 때문에, 네트워크상에서 데이터를 전송하는 중에 이미지 처리를 수행하는 연구가 진행되고 있다. 또 복잡한 연산을 수행하는 센서 네트워크의 예로는 센서 노드들에 대한 서로 다른 사용자의 접근을 사용자의 권한에 대한 정보를 가지고 조정하는 센서 네트워크[16]를 들 수 있다. 이런 센서 네트워크에서는 개별 사용자를 식별하기 위해 키(key)정보를 생성하는 과정에서 복잡한 연산을 필요로 한다. 따라서, 값싸고 낮은 능력의 센서 노드들만으로는 이러한 처리를 수행하기 어려워, 소수의 고성능 센서 노드들에서 키 정보를 생성하는 방식으로 계층적 센서 네트워크를 활용하는 연구가 진행되고 있다.

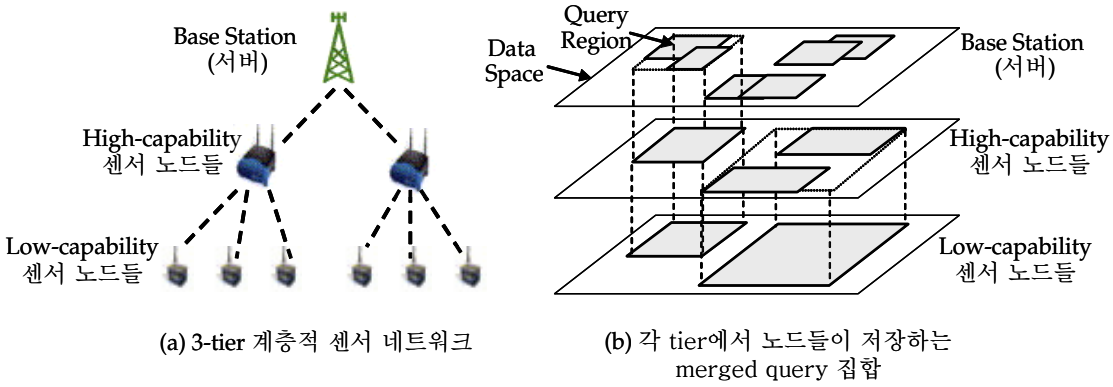
3 계층적 센서 네트워크에서 점진적 처리

본 장에서는 본 논문에서 제시하는 계층적(즉, 다층 구조의) 센서 네트워크에서의 질의 처리 방법인 점진적 처리 방법의 모델과 알고리즘을 제시한다.

3.1 Overview

점진적 질의 처리에서는 개수는 많지만 더 작은 능력을 가진 하위층의 노드가 질의 처리의 일부만을 수행하고, 개수는 적지만 더 큰 능력을 가진 상위 층의 노드에서 하위 층에서 하지 못한 처리를 수행함으로써 전체적인 처리 비용을 systematic하게 조정한다.

Example 1 (계층적 무선 센서 네트워크에서 점진적 처리의 예) 그림 1(a)는 계층적 무선 센서 네트워크의 예이다. 최하위 층에 숫자는 가장 많지만 성능은 가장 낮은 low-capability 센서 노드들이 있고 이것들은 보다 성능이 높은 high-capability 센서 노드와 연결되어 있다. 이때, 서버를 제외한 모든 센서 노드들에서 주기적으로 데이터를 센싱하며, 센싱 데이터는 상위 층의 노드로 relay를 통해서 서버(base station)까지 전송된다. 서버는 최종적으로 질의 처리 결과를 사용자에게 전달하는 역할을 한다. 그림 1(b)는 각 층의 센서 노드들이 저장하는 질의 집합을 보여준다. 이 그림에서 사각형의 영역들은 범위 질의들을 표현한 것이고, boundary 사각형은 질의에서 사용되는 애트리뷰트들로 정의된 domain space(즉, data space)를 나타낸다. 그림에서 서버는 사용자가 등록한 원 질의 6개를 모두 그대로 저장하고 있고, 두번째 층의 센서 노드들(즉, high-capability 센서 노드들)은 원 질의를 2개씩 병합하여 3개의 질의만을 저장한다. 최하위 층의 센서 노드들(즉, low-capability 센서 노드들)은 이를 다시 병합하여 2개의 질의만을 저장한다. 질의 처리시에는 최하위 층의 센서 노드에서 센싱된 데이터는 저장된 2개의 질의들을 처리하여 조건에 맞는 데이터 만을 상위 층으로 전송한다. 상위 층의 센서 노드에서는 자신이 센싱한 데이터와 하위 층의 센서 노드들에서 전송되어 온 데이터에 대해서 자신이 저장하고 있는 질의만을 처리하여 조건에 맞는 데이터만을 상위 층(즉, 서버)으로 전송한다. 이때, 상위 층의 센서 노드는 하위 층의 센서 노드보다 fine granulality로 질의를 가지고 있기 때문에 하위 층에서보다 전송에 있어서 false alarm을 줄여 에너지 소비를 줄일 수 있다. 서버는 이렇게 최종적으로 전송되어 온 센서 데이터에 대해 원 질의를 처리하고 최종 결과를 사용자들에게 돌려준다. □



(a) 3-tier 계층적 센서 네트워크 (b) 각 tier에서 노드들이 저장하는 merged query 집합

그림 1. 계층적 무선 센서 네트워크와 점진적 처리의 예.

그림 1(b)로부터 본 논문에서 사용하는 질의 저장 방법이 일반적인 트리 구조인 다차원 색인의 역(inverted) 구조를 취하고 있음을 알 수 있다. 즉, 다차원 색인에서는 트리의 단말 노드에 object를 저장하고 상위 노드로 갈수록 이를 묶어서 abstract하는 것과는 반대로 제안하는 질의 저장 구조는 루트(즉, 서버)가 전체 object를 저장하고 하위 노드로 갈수록 이를 묶어서 abstract하고 있다. 본 논문에서는 이러한 upside-down 특성에 기반하여, 제안하는 구조를 역 계층 질의 구조(inverted hierarchical query structure)라고 부른다.

점진적 처리는 역 계층 질의 구조를 생성하기 위해서 계층적 센서 네트워크의 각 층별로 저장할 질의 집합을 생성하는 *질의 병합* 과정과 이렇게 생성된 구조를 사용하여 센싱 결과를 처리하여 서버까지 전송하는 *질의 처리*의 두 과정으로 이루어진다. 질의 병합은 배치(batch)작업으로 off-line으로 수행되며, 질의 처리는 데이터가 생성될 때마다 on-line으로 수행 된다. 또한, 질의들은 "점진적"으로 병합되어 최하위층의 노드까지 전달되고, 센싱된 데이터는 그 역방향으로 "점진적"으로 필터링되어 서버까지 전달된다. 질의 병합 과정에서는 먼저 질의들의 minimum bounding rectangle(MBR)을 구하고 이를 병합된 질의로 표현한다.이때, 중요한 것은 질의들을 최종적으로 몇개의 MBR로 병합하여 표현할 지를 정하는 것으로, 이때 정해진 개수가 에너지 소비량와 스토리지 사용량간의 trade-off에 영향을 미친다. 즉, 많은 질의들을 병합할수록, 센서 노드가 질의를 저장하기 위해 사용하는 스토리지는 줄일 수 있으나 false alarm이 발생하여 에너지 소비량은 커진다. 본 장에서는 우선 질의들을 몇개로 병합할지 미리 정해져 있다고 가정하고 모델과 알고리즘을 제시한다. 그리고 제 4장에서 주어진 비용 모델에 따라서 병합 정도를 어떻게 정해야 optimal이 되는지 분석적으로 제시한다.

질의 처리 과정에서는 서버를 제외한 모든 센서 노드들은 자신이 센싱한 데이터와 하위 층의 노드들로부터 받은 데이터가 저장하고 있는 병합된 질의의 조건에 만족되면 이를 상위층의 노드로 전송한다. 본 논문에서 제안하는 방법에서는 상위층의 노드로 갈수록 보다 많은 수의 (즉, fine granularity) 질의를 저장하기 때문에, 질의 결과의 정확도가 점점 더 높아져서 질의 결과가 점진적으로 얻어진다는 특징을 가진다.

3.2 질의 처리 모델

먼저, 본 논문에서 다루는 계층적 센서 네트워크에 대해서 자세히 설명하고 이를 정형적으로 정의한다. 우리가 가정한 센서 네트워크의 구조는 다음과 같다. 모든 센서 노드들은 서버를 루트로 하는 트리형태로 연결되어 있으며, 같은 depth에 위치한 센서 노드들끼리 하나의 층을 이룬다. 트리의 최하위층에 위치한 센서 노드들뿐만 아니라 중간 층의 센서 노드들도 모두 데이터를 센싱하고, 각각 센싱된 데이타는 무선 네트워크를 통해서 상위 층의 센서 노드로 전송되는 방식으로 서버까지 전달된다. 동일층의 센서 노드들은 동일한 능력을 가진다. 즉, 동일한 메모리 용량과 battery 용량을 가진다.그리고, 서버에 가까운 층의 노드일수록 더 큰 능력, 즉 더 큰 battery 용량과 메모리 용량을 가진다. 덧붙여서, 동일한 층에 위치한 모든 노드들은 모두 같은 질의 집합을 저장한다. 본 논문에서는 계층적 센서 네트워크를 정의 1과 같이 정형적으로 정의한다.

Definition 1 (계층적 센서 네트워크) 계층적 센서 네트워크는 높이가 h인 트리 $T = (V, E)$로 표현된다. 여기에서, V는 루트가 서버인 네트워크상에 위치한 센서 노드들과 서버의 집합이며, E는 센서 노드와 그 센서 노드의 부모 센서 노드(혹은 서버)간의 무선 연결을 표현하는 에지(edge)들의 집합이다. 트리 T의 각 i^{th} 층 $(1 \leq i \leq \mathrm{h})$의 노드를 $node_i$라고 하고, $node_i$의 스토리지 용량과 에너지 용량을 각각 s_i과 e_i이라고 했을 때, 계층적 센서 네트워크는 $s_j < s_k$와 $e_j < e_k$인 관계를 만족한다$(1 \leq j < k \leq h)$. □

본 논문에서는 이러한 계층적 센서 네트워크에서의 질의 형태로 범위 질의를 다룬다. 질의의 애트리뷰트들(즉, 질의에 명시된 애트리뷰트들)을 축으로 하는 다차원 data space를 가정했을 때, 질의는 이 다차원 data space상의 hyper rectangle 영역으로 표현된다. 범위 질의의 정형적인 정의는 Lim et al.[9]의 범위 질의 정의를 따른다.

점진적 질의 처리의 첫번째 단계인 **질의 병합**은 병합할 질의들을 모두 둘러싸는 MBR을 찾는 방법으로 수행된다. 이때, 질의 병합을 점진적으로 수행한다는 의미는 하위 층으로 갈수록 점점 더 많은 질의들을 병합한다는 것을 의미한다. 따라서, 하위층일수록 질의 영역의 크기는 커지는 반면, 저장되는 질의의 개수는 작아진다. 이때, 상위층 노드의 질의를 병합하여 하나의 MBR로 표현한 질의를 merged query라 하고, i^{th} 층의 각 노드에 저장될 merged query들의 집합을 Q_i라 한다. 그러면, 층별 merged query들의 집합은 그림 2에서와 같이 역 계층 질의 구조로 표현된다. 이 그림에서 화살표는 질의들이 병합되는 순서를 나타것으로, 화살표의 꼬리에 위치한 질의들이 다음 층의 화살표 머리에 위치한 질의를 생성하기 위해 병합된다는 것을 뜻한다. 그림의 예에서, 1층의 질의 $q_{1,1}$은 $q_{1,2}$, $q_{1,3}$와 함께 병합되어 2층의 질의 $q_{2,1}$이 된다.

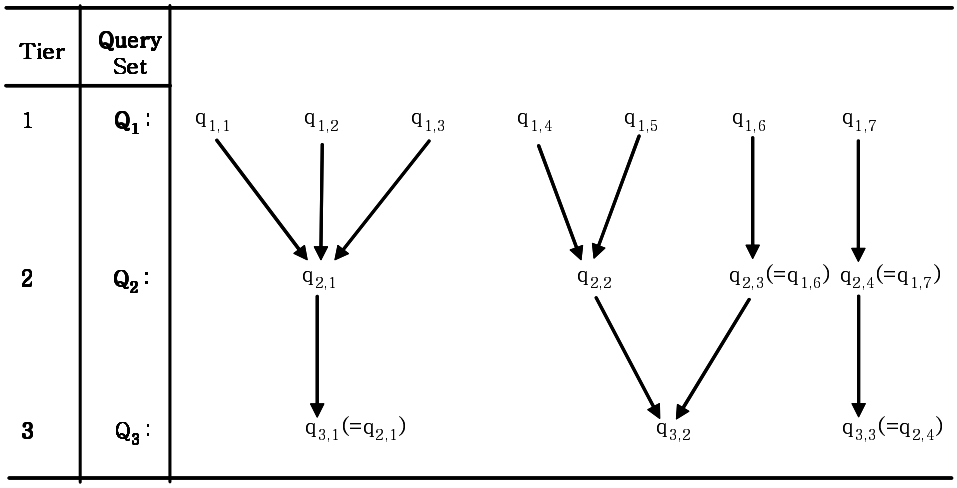


그림 2. 점진적 질의 병합의 예.

질의 병합은 질의 집합 전체를 서로 disjoint한 질의 집합들로 파티션했을 때, 파티션된 질의 집합 각각을 병합하는 것으로도 표현될 수 있다. 그림 3은 그림 2에 있는 7개의 질의를 표현하고 있다. 예를 들어, 그림 2의 질의 $q_{2,1}$는 그림 3의 Q_2에서 파티션된 질의 집합 $\{q_{1,1}, q_{1,2}, q_{1,3}\}$ 에 대응된다. 이러한 파티션은 하위층으로 갈수록 더 coarser해진다.

데이터 element(즉, 노드가 센싱한 값)는 질의의 애트리뷰트들에 의해 정의되는 다차원 data space상에서 점(point)으로 표현된다.

Tier	Query Set	
1	Q_1 :	$\{\{q_{1,1}\},\{q_{1,2}\},\{q_{1,3}\},\{q_{1,4}\},\{q_{1,5}\},\{q_{1,6}\},\{q_{1,7}\}\}$
2	Q_2 :	$\{\{q_{1,1},q_{1,2},q_{1,3}\},\{q_{1,4},q_{1,5}\},\{q_{1,6}\},\{q_{1,7}\}\}$
3	Q_3 :	$\{\{q_{1,1},q_{1,2},q_{1,3}\},\{q_{1,4},q_{1,5},q_{1,6}\},\{q_{1,7}\}\}$

그림 3. 점진적 파티션 병합의 예.

Definition 2 (데이터 element) 센서 노드가 질의에 기술된 d개의 애트리뷰트들 a_1, a_2, ..., a_d의 값들로부터 데이터 element를 생성한다고 가정하자. 그러면, 각각의 데이터 element는 애트리뷰트들 a_1, a_2, ..., a_d에 대응하는 d개 차원 축들의 Cartesian product로 정의되는 d-차원 공간에서 점으로 표현된다. □

질의 처리는 주어진 질의에 대해서 데이터 point가 질의영역내에 포함되는지를 판단한다. 즉, point로 표현된 데이터 값이 영역으로 표현된 질의 영역에 속하는지를 판단한다. 점진적 데이터 필터링은 역 계층 질의 구조에 따라 최하위 층의 노드로부터 최상위 층의 노드(서버)로 데이터를 전송하면서, 범위 질의의 predicate을 평가(evaluation)한 결과에 따라 데이터 element를 필터링하는 처리 방법이다.

그림 4는 처리 방법의 예를 보여 준다. 이 그림에서 화살표는 데이터의 흐름을 나타낸다. 화살표의 꼬리에 위치한 질의의 predicate을 데이터 element(v)가 만족할 때, 그 데이터 element는 화살표의 머리가 가리키는 상위층의 질의에게 전송된다. 그림의 예에서는 서버에 저장된 질의 $q_{1,1}$이 데이터 element v를 질의 결과로 하게 된다.

다음 장에서는 이러한 모델을 사용하여 점진적으로 질의를 처리하는 알고리즘들에 대해서 설명한다.

3.3 질의 병합 및 질의 처리 알고리즘

점진적 질의 병합 알고리즘

점진적 질의 병합 알고리즘은 각 i^{th} 층에 저장할 크기 C_i의 merged query 집합 Q_i를 생성한다. 이때, 본 알고리즘의 목표는 센서 노드의 제한된 메모리 용량을 고려하여, 질의 처리 비용이 최소가 되도록

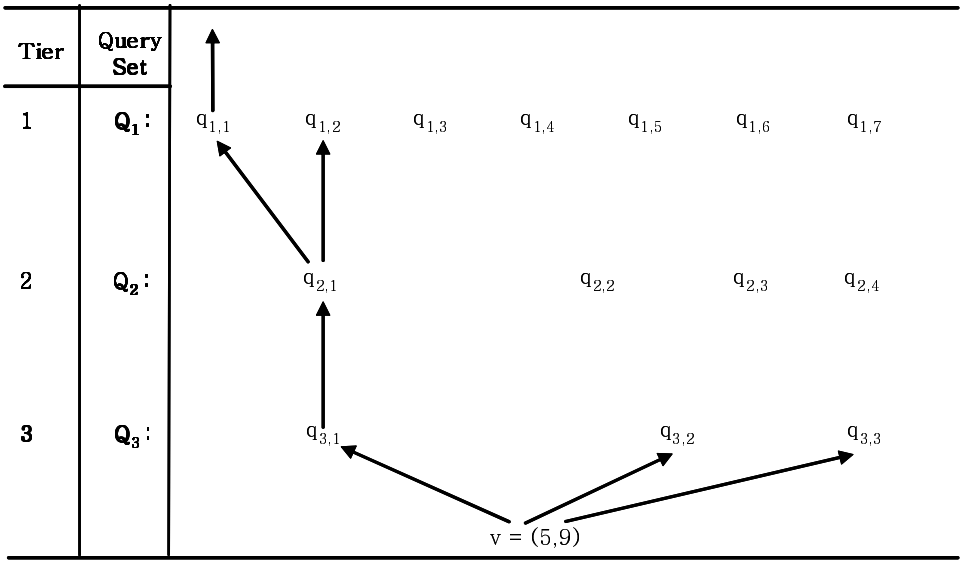


그림 4. 점진적 질의 처리의 예.

하는 C_i개의 merged query를 생성하는 것이다. 특정 merged query의 집합 Q_i에 대해서 질의 처리 비용이 얼마나 될지를 예측하기는 매우 어렵다. 이는 질의 처리 비용에 라우팅 등의 네트워크에 의존적인 요소들 뿐만 아니라, 질의의 분포, 데이터의 분포 등의 알려지지 않은 요소들이 관여하기 때문이다. 본 논문에서는 데이터 필터링 과정에서 필요한 데이터 전송양을 척도로 하여 이를 단순화한 Xiang et al.[22]에서의 모델을 기본으로 하여 메모리 사용량을 고려하도록 이를 확장한다. Xiang et al.이 제안한 모델에서는 병합되는 두개의 질의 간에 서로 겹치는 영역인 겹침 영역(overlap region)의 크기와 MBR로 묶었을 때 false alarm을 발생시키는 영역인 죽은 영역(dead region)의 크기를 계산으로 구하고, 겹침영역(O)과 죽은영역(D)의 차이, 즉 O-D를 비용으로 하여, 이를 최대가 되도록 하는 방향으로 병합을 수행한다. 이는 결국 공통되는 영역이 많은 질의를 질의들끼리 병합하는 것으로서 데이터 전송양을 줄이는데 있어서 합리적인 척도이다.

제안하는 알고리즘은 이러한 척도를 기반으로 greedy하게 병합을 수행한다. 배치 처리시, 병합된 질의들과 병합되지 않는 질의들 모두에 대해서, 서로 다른 2개의 질의 q_i 와 q_j 중에, $O(q_i, q_j) - D(q_i, q_j)$의 값이 가장 큰 질의들을 먼저 병합한다. Xiang et al.[22]이 제안한 정책에서 $O(q_i, q_j) - D(q_i, q_j) \geq 0$를 만족하는 질의의 쌍들만 고려한 점만 제외하고, 제안한 정책은 Xiang et al.[22]의 정책과 동일하다. 제안하는 정책에서 threshold 값인 0를 사용하지 않는 이유는 질의를 저장하기 위해 사용되는 센서 노드의 메모리 용량을 고려하기 위함이다. 즉, merged query들의 양이 커서 센서 노드의 메

모리에 모두 저장할 수 없는 경우와 충분한 메모리가 있는대도 질의를 병합하여 저장함으로써 불필요한 데이터를 전송하는 경우가 발생할 수 있기 때문이다. 전자의 경우, 제안하는 방법에서는 질의 처리 비용을 손해보면서, 남아 있는 질의들 중에 몇몇을 강제로 병합한다. 그리고, 후자의 경우, 메모리 사용량 비용을 손해보면서, 더 많은 질의를 노드에 저장토록 하여 데이터 전송양을 줄인다. 그러나, Xiang et al.[22]이 제안한 방법에서는 이러한 메모리 제약조건을 고려하지 않는다.

Algorithm **Progressive Query Merging**
Input: (1) $Q=\{q_1, q_2, \cdots, q_n\}$: 병합할 질의들의 집합
(2) h: 센서 네트워크의 height
(3) $K=\{k_1, k_2, \cdots, k_h\}$: 센서 노드에 저장될 층 별 merged query 개수의 집합 /* $k_1 \geq k_2 \geq \cdots \geq k_h$ 이고, $k_1=n$ */
Output: $\{Q'_2, \cdots, Q'_h\}$: 2층부터 h층까지의 merged query들의 집합
Algorithm:
Begin
1. For tier t = 2 to h begin
2. $Q_t' = Q_{t-1}'$ /* $Q'_1 = Q$ */
3. repeat
4. i=0,1,…,$|Q_t'|$ and j=0,1,…,$|Q_t'|$ ($j < i$)에 대해 $O(q_i,q_j) - D(q_i,q_j)$가 최대인 질의의 쌍 q_i와 q_j를 찾는다; q_i와 q_j를 Q_t'에서 제거한 뒤에 병합한다;
5. Merged query를 Q'_t에 삽입한다.
6. until ($|Q'_t| \leq k_t$)
7. end
8. Return $\{Q'_2, \cdots, Q'_h\}$
End

그림 5. 점진적 질의 병합 알고리즘.

그림 5는 progressive query merging 알고리즘을 보여 준다. 이 알고리즘은 원 질의 집합 Q, 구성할 계층적 센서 네트워크의 height h, 각 층 별로 노드에 저장할 merged query의 개수의 집합 K를 입력으로 받아서 각 층별로 노드에 저장할 merged query의 집합들을 생성한다.질의 병합 과정(line 3~6)에서는 모든 질의 쌍에 대해, overlap region(O)과 dead region(D)의 차이, 즉 $O-D$를 계산하고, 이 값이 최대가 되는 질의 쌍을 찾아 병합한다. 이 과정은 각 t^{th} 층에 대해서 merged query 집합의 크기가 k_t가 될 때까지 반복 수행된다. 최종적으로 결과를 리턴한다(line 8).

점진적 질의 처리 알고리즘

질의 처리 과정은 노드에서 센싱한 데이터와 하위 층의 노드들로부터 전송되어 온 데이터로부터 (merge된) query들을 만족하는 것들만을 필터링하여 상위층의 노드로 다시 전송하는 역할을 수행한다. 그림 6은 점진적 질의 처리 알고리즘을 나타낸다.제안하는 점진적 질의 처리 알고리즘은 각 데이터 및 질의 별로 수행된다. 따라서, 질의 처리 시간의 관점에서 본다면 다소 비효율적일 수 있다. 그러나 질의 처리 시간은 본 논문에서 다루는 주요한 비용인 energy 및 storage 비용과는 independent한 비용이므로, 시간적인 측면에서 질의를 효율적으로 처리하는 알고리즘은 여기서 다루지 않는다.

Algorithm **Progressive Query Processing**

Input: (1) $Q_t = \{q_1, q_2, \cdots, q_m\}$: t번째 tier에 위치한 임의의 sensor node s에 저장된 set of merged queries

(2) $D_t = \{d_1, d_2, \cdots, d_n\}$: sensor node s가 생성한 data

(3) $R_{t+1} = \{r_1, r_2, \cdots, r_o\}$: sensor node s가 (t+1)번째 tier의 child sensor node들로부터 질의 처리 결과로서 수신한 data

Output: $R_t = \{r'_1, \cdots, r'_k\}$: sensor node s가 질의 처리한 결과로, (t-1)번째 tier에 위치한 parent sensor node로 전송할 data ($R_t \subseteq (D_t \cup R_{t+1})$)

Algorithm:

```
Begin
1.  D = D_t ∪ R_{t+1}
2.  For each data element e_i in D begin
3.    For each query q_j in Q_t begin
4.      if e_i 의 value가 q_j의 query region에 속하면 begin
5.        Insert e_i into R_t.
6.        break              /* line 3의 loop를 탈출함 */
7.      end
8.    end
9.  end
10. Return R_t
End
```

그림 6. 점진적 질의 처리 알고리즘.

점진적 질의 처리 알고리즘에서 t^{th} 층의 센서 노드는 자신이 센싱한 데이터인 D_t 뿐만 아니라 자신의 바로 하위층, 즉 $(t+1)^{th}$ 층의 질의 처리 결과인 R_{t+1}을 입력으로 받아 질의 처리의 대상이 되는 데이터로 삼는다(line 1). 그리고, 질의 병합 알고리즘의 결과로 생성된 merged query 집합 Q_t와 비교하여 결과로 나올 수 있는 입력만을 선택하고(line 2~9), 그 결과를 R_t로 하여 상위 층, 즉 $(t-1)^{th}$ 층에 있는 자신의 부모 노드에게 전달한다. 이러한 과정은 최하위 층의 센서 노드로 부터 최상위의 서버

까지 재귀적으로 수행되고 이를 통해서 최종적으로 질의 결과를 얻을 수 있다.

본 장에서는 각각의 층의 센서 노드들이 몇개의 merged query를 저장해야 하는지 이미 알고 있다고 가정하고 알고리즘을 제시하였다. 다음 장에서는 응용과 사용자에 의해 주어진 여러 parameter들에 따라서 최선의 merged query 개수를 구하는 최적화 방법을 제시한다.

4 Analysis

본 장에서는 계층적 센서 네트워크를 설계하기 위해 각 층에서 저장할 merged query의 optimal 개수를 분석적으로 구하는 방법을 제시한다. 먼저 4.1절에서는 비용 모델을 제시하고 4.2절에서는 이를 최적화하는 방법을 제시한다.

4.1 Cost model

본 논문에서는 질의를 저장하기 위해서 필요한 스토리지 비용과 결과를 전송하기 위해서 필요한 에너지 비용의 weighted sum을 비용 모델로 사용한다. 이때, 스토리지 비용으로는 모든 센서 노드들의 메모리 사용량의 총합을, 그리고 에너지 비용으로는 질의 처리시 발생하는 데이터 전송량의 총합을 사용한다.[1]

식 1는 비용 모델을 표현한 함수인 $weightd_sum$을 나타낸다.

$$\begin{aligned} weighted\_sum &= \alpha \cdot \textit{총 데이터 전송량} + \textit{총 메모리 사용량} \\ where\ \alpha &= \textit{사용자에 의해 주어진 scale factor} \end{aligned} \tag{1}$$

이 수식에서 α는 에너지 소비량과 스토리지 사용량의 중요도를 정하는 weight 값으로 사용자가 어느 비용을 더 중요하게 보는 가에 따라 다른 값이 주어진다. 즉, 에너지의 소비가 더 중요한 환경에서는 α 값으로 큰 값이 주어지고, 스토리지의 사용이 더 중요한 환경에서는 상대적으로 작은 값이 주어진다. 본 논문에서는 에너지 소비량과 스토리지 사용량의 trade-off를 위해, 두 비용들의 중요도를 동등하게 하는 α 값을 정하여 이를 기준 α라 부르고, α_0로 표시한다. α_0는 서로 다른 scale을 사용하는 두 비용 간에 균형을 맞추는 값으로서 이를 기준으로 하여 응용에 맞는 적합한 α 값을 정하는데 guideline으로

[1]본 논문에서는 데이터 전송량의 총합과 메모리 사용량의 총합으로 byte 단위로 환산된 값을 사용한다.

사용한다. 식 2는 α_0의 정의를 나타낸다.

$$\alpha_0 = \frac{\textit{총 메모리 사용량의 최대값}}{\textit{총 데이터 전송량의 최대값}} \tag{2}$$

식 2에서, 총 데이터 전송량의 최대값은 원 질의를 모두 병합하여 하나의 질의만을 모든 센서 노드들에 저장한 경우, 질의 처리시 발생하는 데이터 전송량의 총 합을 나타낸다. 총 메모리 사용량의 최대값은 주어진 질의를 그대로 모든 센서 노드들에 저장하기 위해서 필요한 메모리 사용량의 총 합을 나타낸다. 즉, α_0는 worst case의 메모리 사용량을 worst case의 데이터 전송량으로 나눈 값이 된다.

식 1에서, 총 메모리 사용량은 층별로 센서 노드들에 저장되는 질의들의 용량에 의해 결정되며, 총 데이터 전송량은 이에 따라 층별로 전송되는 데이터의 용량에 의해 결정된다. 본 논문에서는 먼저 층별로 센서 노드들에 저장되는 질의들의 용량을 formulate하기 위해 merging_rate를 정의하고 이를 weighted_sum을 최적화할 target 파라미터로 사용한다. Merging_rate는 계층적 센서 네트워크에서 서로 인접한 층의 노드간의 메모리 사용량의 비를 나타내며, 수식 3과 같이 정의된다.

$$\begin{aligned} merging\_rate = & \frac{i^{th}\ \textit{층에 위치한 센서 노드에 저장된 질의의 개수}}{(i-1)^{th}\ \textit{층에 위치한 센서 노드에 저장된 질의의 개수}} \\ & where\ 2 \leq i \leq h\ (h = \textit{센서 네트워크의 height}) \\ & \textit{서버는 1층에 위치하며, 모든 질의들을 저장하고 있음} \end{aligned} \tag{3}$$

정의에 따르면 merging_rate는 서버에 저장된 질의들이 층별로 병합되는 정도를 나타내며, 0에서 1사이의 값을 가진다. 이때, 값이 0에 가까우면 하위 층으로 가면서 더 많은 개수의 질의들을 병합함을 의미한다. 즉, merging_rate 값에 따라서 각 층의 센서 노드에서 저장해야 할 질의의 개수가 정해진다. 예를 들어, merging_rate가 0 이면 제안하는 방법은 naive 방법중 centralized apporach에 해당하게 되고 1이면 distributed approach에 해당하게 된다.

다음으로, 층별로 전송되는 데이터의 양을 formulate하기 위해, cover라는 개념을 정의한다. Cover는 질의 영역들을 union한 영역이 data space상에서 차지하는 크기의 비율로 정의한다. 데이터 전송량의 정확한 값을 구하기 위해서는, 각 계층별로 병합된 질의들의 selectivity와 병합으로 인해서 생성되는 죽은 영역의 크기 등의 정보가 필요하다. 그러나, 이러한 정보는 데이터와 질의의 분포 등 응용 환경에 많은 영향을 받기 때문에 네트워크 설계 시점에 이에 대한 정확한 값들을 구하기는 어렵다. 따라서, 본 논문에서는 cover를 approximate하게 모델링하여 사용한다. 정의 3은 질의 집합 Q의 cover를 나타낸다.

Definition 3 (질의 집합 Q의 cover)

$$
\begin{aligned}
Query\ set\ Q &= \{\ q_1,\ q_2,\ \cdots,\ q_n\ \}\ 에\ 대해 \\
cover(Q) &= \frac{|\bigcup_{q_i\ \in\ Q}(\Phi(q_i))|}{|\Phi(D)|} \\
where\ D &= data\ space,\ \Phi(q_i)\ =\ 질의\ q_i\ 의\ 영역, \\
|\Phi(q_i)| &= 질의\ q_i\ 의\ 영역크기
\end{aligned}
$$

□

질의들이 data space상에 uniform하게 분포되어 있다고 가정하고, 여기서 n은 merged query의 개수, s는 질의의 평균 selectivity, c는 질의의 cover라 하였을 때, 수식 4의 cover(n)은 cover의 approximation을 나타낸다.

$$
\begin{aligned}
cover(n) &= \begin{cases} -a\ \cdot\ n\ +\ b & (n\ <\ \frac{c}{s}) \\ c & (n\ \geq\ \frac{c}{s}) \end{cases} \\
where\ a &= s\ \cdot\ \frac{1\ -\ c}{c\ -\ s},\ b\ =\ 1\ +\ a
\end{aligned} \tag{4}
$$

$cover(n)$은 다음과 같은 특징을 가진다. (1) n=1일 때, $cover(n)$의 값은 1이다.(2) n의 값이 증가함에 $cover(n)$의 값은 linear하게 감소하다가, $n\ \geq\ \frac{Q의\ cover}{Q의\ 평균\ selectivity}$이면, $cover(n)$은 원 질의 집합 Q의 cover와 동일한 값을 가진다.

이러한 특징들은 query들이 실제로 병합되어 가면서 나타내는 cover의 변화를 관찰한 것에 기반한 것이다. 특징(1)은 MBR을 기반으로한 질의 병합 방법으로부터 기인한다. Merged query의 질의 영역은 merge 될 질의 영역들을 포함하는 MBR로 표현되므로, merged query의 질의 영역 크기는 merge 될 질의 영역들을 union한 영역의 크기보다 항상 크거나 같다. 본 논문에서는 질의 공간에 uniform하게 분포하는 다수의 질의를 처리하는 환경을 가정하므로, 원 질의들 모두를 1개의 merged query로 병합하였을 때, 그 merged query의 cover는 1이 되는 것으로 가정하였다. 특징 (2)는 질의 병합이 진행됨에 따라, 질의들간의 겹침 영역의 크기는 감소하는 현상으로부터 기인한다. 질의 병합에서는 겹침 영역의 크기가 큰 질의들을 먼저 병합하므로, 질의 병합이 진행됨에 따라 질의들간에 겹침 영역의 크기는 감소하게 된다. 겹침 영역의 크기가 작은 질의를 병합하는 것이 큰 질의를 병합할 때보다, 결과로 생성되는 merged query의 질의 영역 크기가 더 커지므로, 결과로 생성되는 merged query들의 개수가 감

소하면 merged query들의 cover는 증가하게 된다. cover 모델에서는 질의가 어느 정도 병합되면, 겹침 영역이 존재하지 않는 merged query들만 남게 되는 것으로 가정하였다. 그리고, 이러한 경우의 merged query 개수를 $\frac{query의\ cover}{query의\ 평균\ selectivity}$ (서로 영역이 겹치지 않는 질의들만으로 data space의 영역을 완전하게 채울 수 있는 질의들의 이론적인 개수)로 정하였다.

다음 절에서는 cover 모델과 merging_rate를 사용해서 모델링된 weighted_sum의 값이 최소가 되도록 하는 merging_rate값인 optimal merging_rate를 구하는 방법을 제시한다.

4.2 최적화 방법

본 절에서는 먼저 4.1절에서 설명한 merging_rate 및 cover 모델을 이용하여 weighted_sum 수식을 표현한다(상세한 전개 과정은 appendix A를 참고). 그리고, weighted_sum을 최소로 하는 optimal merging_rate를 분석적으로 구한다. 표 1은 본 절에서 사용하는 주요 notation을 정리한 것이다. 본 절에서는 수식을 단순화하기 위해 각 node가 질의 처리시 1개의 sensor data만을 생성한다고 가정한다.

표 1. 파라미터에 대응하는 notation

파라미터	**Notation**
원 질의의 개수	N_Q
원 질의의 cover	c
원 질의의 평균 selectivity	s
Data element의 크기	d
Merging_rate	x
Sensor network의 height	h
Sensor network의 fanout	f

수식 5는 수식 4의 cover 모델과 merging_rate를 이용하여 총 데이터 전송량을 formulate한 것이다.

$$
\begin{aligned}
total\_transmission &= \sum_{i=2}^{h}(d \cdot f^{i-1} \cdot \sum_{j=2}^{i}(-a \cdot x^{j-1} \cdot N_Q + b)) \\
where\ a &= \frac{s \cdot (1 - c)}{c - s},\ b = 1 + a
\end{aligned} \tag{5}
$$

수식 6은 총 메모리 사용량의 수식을 formulate한 것이다.

$$total\_storage \;=\; \sum_{i=2}^{h}(\; 2 \;\cdot\; d \;\cdot\; f^{i-1} \;\cdot\; N_Q \;\cdot\; x^{i-1}) \tag{6}$$

수식 5와 수식 6에 의해, weighted_sum은 다음과 같다.

$$\begin{aligned} weighted\_sum &= \alpha \cdot total\_transimission \cdot total\_storage \\ &= \alpha \cdot d \cdot \sum_{i=2}^{h}(f^{i-1} \cdot [\; \sum_{j=2}^{i}(-a \cdot x^{j-1} \cdot N_Q + b)) \\ &+ \quad 2 \cdot N_Q \cdot x^{i-1} \;]) \\ where\ a &= \frac{s \cdot (1 - c)}{c - s},\ b = 1 + a \end{aligned} \tag{7}$$

본 논문에서는 optimal merging_rate를 구하기 위해서 먼저 weighted_sum을 미분한 뒤에, 미분 결과 수식으로 부터 근들을 구한다. 그리고, 구해진 근들 각각을 weighted_sum 수식에 대입하여 수식을 evaluation한 결과가 최소가 되도록 만드는 근, 즉 optimal merging_rate를 구한다. 이를 위해 본 논문에서는 수학 tool인 maple[11]을 사용하였다.

5 성능 평가

본 절에서는 실험을 통해서 제안하는 방법의 우수성을 보인다. 제 5.1 절에서는 성능 평가를 수행하는데 사용한 실험 데이타 및 질의, 그리고 실험 환경을 소개한다. 다음으로, 제 5.2 절에서는 실험 결과를 제시한다.

5.1 실험 데이타 및 실험 환경

본 논문에서는 2가지 종류의 실험을 수행한다. 실험 1에서는 파라미터들의 값을 달리하며 제안한 비용 모델의 정확도를 보인다. 그리고, 실험 2에서는 제안하는 질의 처리 방법이 기존 방법에 비해 질의 처리를 위한 총 cost, 즉 weighted_sum 측면에서 더 우수한 성능을 가짐을 보인다. 실험 1과 2에서 공통으로 사용한 파라미터들은 데이터 전송량과 메모리 사용량간의 weight값인 α, 원 질의의 cover, 원 질의의 selectivity, 원 질의의 차원, 센서 네트워크의 height 및 fanout으로 총 6개이다.

실험 결과를 위해, 본 논문에서는 optimal merging_rate와 weighted_sum을 measure로 사용한다. 실험 1에서는 실험 데이터 및 분석을 통해 구한 optimal merging_rate와 weighted_sum을 비교하여 분석적 방법으로 구한 optimal 값이 실험 데이터로부터 구한 optimal 값과 큰 차이가 없음을 보인다. 그리고, 실험 2에서는 본 논문에서 제안하는 질의 방법인 progressive_approach와 Xiang et al.[22]이 제안한 질의 처리 방법인 iterative_approach의 weighted_sum을 비교하여, 대부분의 parameter 값들에 대해, 제안하는 방법이 기존 방법보다 더 작은 weighted_sum을 가짐을 보인다.

실험 1과 실험 2는 동일한 데이터와 질의들을 사용하여 수행된다. 실험에서 사용한 데이터와 질의는 uniform 분포를 따르도록 random하게 생성한 syntactic 데이터와 질의이다. 여기서 "unform"하다는 것은 질의(혹은 데이터 element)가 질의 space(혹은 data space) 상에서 어디에 위치할 지가 random하게 정해진다는 것을 의미한다. 실험에서 사용한 질의들은 2가지 서로 다른 방법으로 생성하였다. 즉, 원 질의의 개수를 조정하는 방법을 기본으로 하되, 원 질의의 cover를 달리한 실험에서 사용할 질의들을 생성하기 위해, 원 질의 cover를 조정하는 방법도 사용하였다.여기서 원 질의의 개수를 조정하는 방법은 생성된 질의 개수가 입력 파라미터인 질의 개수와 일치할 때까지 질의들을 생성하는 방법이다. 그리고, 원 질의의 cover를 조정하는 방법은 생성된 질의들의 cover가 입력 파라미터인 cover와 일치할 때까지 질의들을 생성하는 방법이다. 이와 같이 2가지 서로 다른 방법을 사용하여 질의들을 생성한 이유는, 원 질의 개수와 원 질의의 cover가 서로 의존성 — uniform 분포를 따르는 질의들의 집합에서 원 질의의 selectivity가 고정된 경우, 질의 개수가 증가하면 cover도 증가함 — 을 가지기 때문에, 질의 개수와 cover를 동시에 조정해서는 uniform한 분포의 질의 집합을 생성할 수 없기 때문이다. 본 논문에서는 원 질의의 cover를 달리한 실험에서만 질의의 cover를 조정하는 방법으로 생성된 질의들을 사용하였으며, 나머지 파라미터들의 값을 달리한 실험에서는 질의 개수를 조정하는 방법으로 생성된 질의들

을 사용하였다. 질의(MBR)생성의 경우, 데이터 공간상에서 random하게 선택한 점을 질의 영역의 좌상점으로 하여, 그 점을 기준으로 각 차원축에 대한 width가 같은 질의들을 생성하였다. 데이터의 경우, 데이터를 구성하는 각각의 애트리뷰트가 그 애트리뷰트의 domain에 속하는 값을 random하게 가지도록 생성하였다. 그리고, 생성된 데이터 element의 개수가 입력 파라미터인 데이터의 개수에 도달하면 데이터 생성을 완료하였다.

실험은 Pentium 4 2GHz, 1Giga byte의 RAM을 가진 Linux-Redhat 시스템에서 수행하였다. 본 논문에서는 센서 네트워크와 관련된 데이터베이스 분야의 연구들에서와 같이 시뮬레이터 프로그램을 사용하여 실험을 수행한다. 이는 실험을 위해서 대규모의 센서 네트웍을 실제로 구축하고 필요에 따라 configuration을 수정하기가 매우 어렵기 때문이다. 실험에서 많이 사용되는 대표적인 센서 네트웍 시뮬레이터로 TOSSIM(TinyOS SIMulator)[7]과 ns-2[13]등을 들 수 있다. 그러나, 이러한 기존 시뮬레이터들은 다수의 연속 범위 질의들(e.g., 15개 이상)을 동시 처리하는 환경을 제공하지 않기 때문에[22] 본 논문에서와 같이 대량의 질의를 대상으로 하는 실험에서 사용하기에는 부적합하다. 따라서, 본 논문에서는 주요한 파라미터들을 중심으로 계층적 센서 네트워크 환경을 단순화하고, 이러한 환경에서 질의 처리 비용을 계산할 수 있는 시뮬레이터를 직접 구현하여 실험을 수행하였다.

실험에 사용한 파라미터들은 표 2와 같다. 본 논문에서는 실험 결과들간의 비교를 위해, 표 3의 용어들을 정의하여 사용한다.

5.2 실험 결과

실험 1:

본 절에서는 먼저, 분석적 방법과 실험으로 구한 optimal merging_rate를 비교함으로써 제 4.2절에서의 제안한 비용 모델의 정확도를 보인다.

그림 7은 weighted_sum에서 α의 값을 변화시킨 경우에 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 $0.905 \sim 2.619$ 이다. 예외적으로 α 값이 $\frac{\alpha_0}{10}$ 일 때, Opt_Merging_Rate_Ratio는 2.619로, 실험을 통해 구한 optimal 값이 분석을 통해 구한 값보다 약 2배 크다. 그러나, 다른 α값들에 대해서는 Opt_Merging_Rate_Ratio가 $0.905 \sim 1.000$로, Opt_Merging_Rate_E와 Opt_Merging_Rate_S간의 비가 1에 근사한다. 그리고, Opt_Weighted_Sum_Ratio의 값은 $0.929 \sim 1$로

표 2. 실험을 위한 파라미터의 요약

질의생성방법 / 파라미터	질의 개수를 control한 방법		질의 cover를 control한 방법
센서 네트워크의 height	3, 4, 5		4
센서 네트워크의 fanout	2, 4, 8, 16		8
데이터 전송량과 메모리 사용량 간의 weight value α	$\frac{\alpha_0}{100}$, $\frac{\alpha_0}{10}$, α_0, $\alpha_0 \cdot 10$, $\alpha_0 \cdot 100$		α_0
원 질의의 평균 selectivity	0.00001, 0.0001, 0.001		0.0001
원 질의의 차원	1, 2, 3		2
원 질의 개수	1046*		101, 1046, 52685
원 질의의 cover	Selectivity의 변화	0.01, 0.10, 0.64	0.01, 0.10, 0.99
	차원의 변화	0.1, 0.1, 0.1	
* 질의의 개수는 "질의 cover를 control하는 방법"으로 생성된 질의들의 개수들 중에 중간값(즉,1046)으로 정함			

표 3. 실험을 위한 용어의 요약

용어	정의
$Opt\_Merging\_Rate\_E$	실험 데이터로부터 구한 optimal merging_rate
$Opt\_Merging\_Rate\_S$	분석적으로 구한 optimal merging_rate
$Opt\_Merging\_Rate\_Ratio$	$\frac{Opt\_Merging\_Rate\_E}{Opt\_Merging\_Rate\_S}$
$Opt\_Weighted\_Sum\_Ratio$	$\frac{\textit{실험 데이터에서 Opt_Merging_Rate_E 점에서의 weighted_sum}}{\textit{실험 데이터에서 Opt_Merging_Rate_S 점에서의 weighted_sum}}$
$Weighted\_Sum\_Gain$	$\frac{\textit{Xiang 방법의 weighted_sum}}{\textit{실험 데이터에서 Opt_Merging_Rate_S 점에서의 weighted_sum}}$

optimal에서 실험 및 분석을 통해 구한 weighted_sum이 유사한 값들을 가진다. 실험 결과에서 보는 바와 같이, α 값의 증가는 총 데이터 전송량의 cost에 대한 weight가 총 메모리 사용량의 cost에 대한 weight보다 상대적으로 커지도록 하는 효과를 일으키므로, weight가 큰 총 데이터 전송량의 cost를 감소시키는 방향으로 — optimal merging_rate가 1에 근사하도록 — optimal이 결정된다.

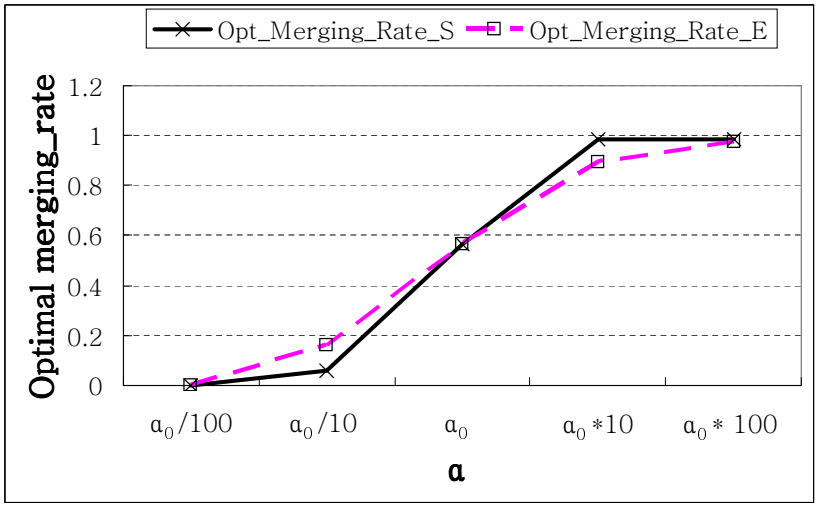


그림 7. α 값을 달리하였을 때, optimal merging_rate

그림 8은 weighted_sum에서 원 질의의 cover를 변화시킨 경우에 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 0.00092 ~ 1.008 이다. 예외적으로 원 질의의 cover가 0.99인 경우 Opt_Merging_Rate_Ratio는 0.00092로, 분석을 통해 구한 optimal 값이 실험을 통해 구한 값보다 매우 크다. 그러나 다른 cover값들에 대해서는 0.985 ~ 1.008로 Opt_Merging_Rate_E와 Opt_Merging_Rate_S간의 비가 1에 근사한다. 그리고, Opt_Weighted_Sum_Ratio의 값은 0.995 ~ 1로 optimal에서 실험 및 분석을 통해 구한 weighted_sum이 유사한 값들을 가진다. 원 질의의 cover가 증가하게 되면, 총 데이터 전송량의 최대 및 최소값간의 차이가 줄어, 데이터 전송량이 메모리 사용량에 비해 전체 비용이 미치는 영향이 줄어들게 된다. 따라서, optimal은 총 메모리 사용량의 cost를 줄이는 방향으로 — optimal merging_rate가 0에 근사하도록 — optimal이 결정된다.

그림 9는 원 질의의 평균 selectivity값을 변화시킨 경우에 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 0.958 ~ 1.183 이고, Opt_Weighted_Sum_Ratio의 값은 0.983 ~ 1이다. Selectivity의 증가는 cover의 증가와 밀접한 관계를 가진다. 즉, 실험에서와 같이 질의 개수가 고정된 상태에서 selectivity가 증가하면, 원 질의의 cover 역시 증가한다. 따라서, cover가 증가하는 실험의 결과에서와 같이 selectivity가 증가하면, 총 메모리 사용량의 cost를 줄이는 방향 — optimal merging_rate가 0에 근사하는 방향 — 으로 optimal이 결정된다.

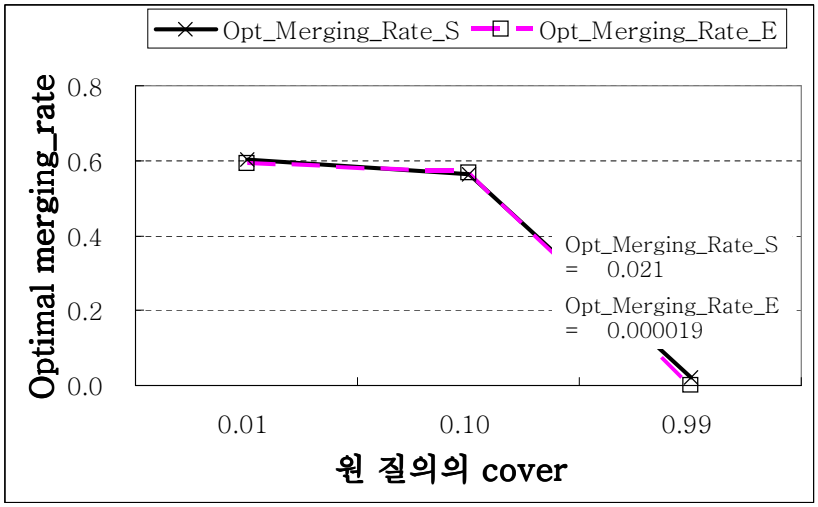


그림 8. 원 질의의 cover를 달리하였을 때, optimal merging_rate

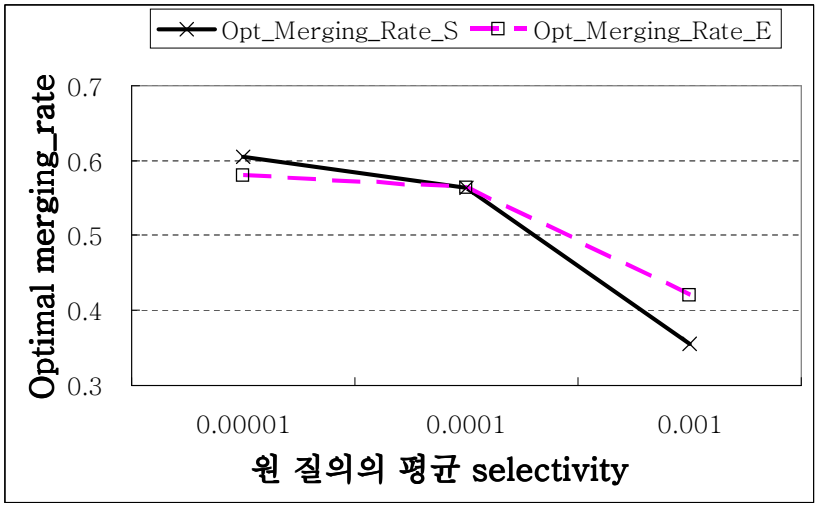


그림 9. 원 질의의 평균 selectivity를 달리하였을 때, optimal merging_rate

그림 10은 센서 네트워크의 height를 변화시킨 경우에 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 0.993 ~ 1.088 이고, Opt_Weighted_Sum_Ratio의 값은 0.993 ~ 1 이다. 네트워크의 height의 증가하게 되면 데이터 전송량의 cost가 메모리 사용량의 cost에 비해 더 빨리 증가하게 되므로, 총 데이터 전송량의 cost를 감소시키는 방향 — optimal merging_rate가 1에 근사하는 방향 — 으로 optimal이 결정된다. 이는 센서 네트워크에서 데이터가 층별로 누적되어 전송되는 특성으로부터 기인한다. 센서 네트워크의 fanout과 원 질의의 차원을 달리한 실험 결과는 appendix B를 참고하기 바란다.

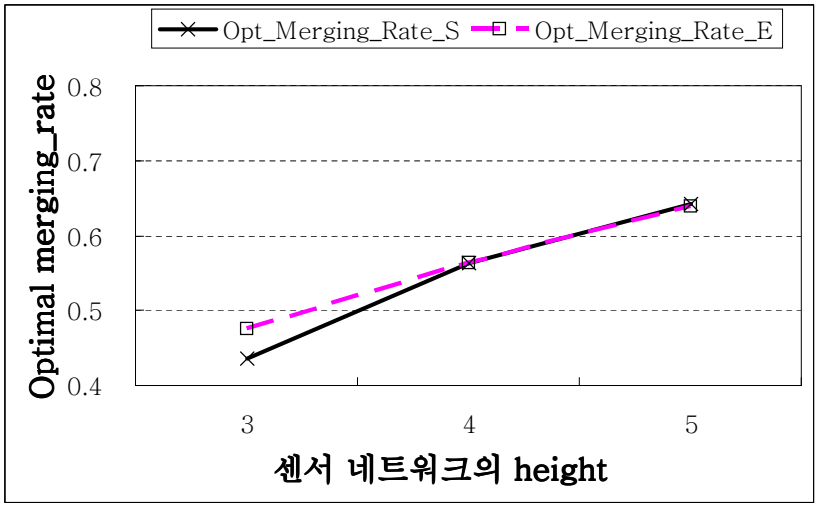


그림 10. 센서 네트워크의 height 달리하였을 때, optimal merging_rate

위의 실험 결과에서 볼 수 있듯이 Opt_Merging_Rate_E와 Opt_Merging_Rate_S간의 비를 나타내는 Opt_Merging_Rate_Ratio는 0.00092 ∼ 2.619로 다소 차이를 보였다. 그러나, 값이 큰 차이를 보이는 경우는 α의 값이 $\frac{\alpha_0}{10}$과 원 질의의 cover값이 0.99인 경우로 국한된다. 더우기 Opt_Weighted_Sum_Ratio의 값은 모든 parameter들의 전 범위에 대해 0.929 ∼ 1 로, 1에 근사하는 값을 가졌다. 결과적으로 본 논문의 분석적 모델이 정확성을 가짐을 알 수 있다. 따라서, 본 논문의 나머지 실험에서는 분석적 방법으로 구한 optimal merging_rate를 사용하여 실험을 수행한다.

실험 2:

본 절에서는 total cost 측면에서 제안한 방법의 우수성을 보여 주기 위해, progressive approach와 iterative approach[22]의 weighted_sum을 비교한다. Iterative approach에서의 weighted_sum은 progressive approach에서 α_0를 구한 뒤, 이를 기준 α로 하여 [α · Iterative approach에서의 총 데이터 전송량 + Iterative approach에서의 총 메모리 사용량]으로부터 구하였다.

그림 11은 α값을 변화시킨 경우에 두 방법의 weighted_sum을 비교한 결과이다. Weighted_Sum_Gain은 0.989 ∼ 84.995로, 예외적으로 α의 값이 α_0 · 10인 경우에만, Weighted_Sum_Gain의 값이 0.989로 optimal에서 progressive approach의 weighted_sum이 iterative approach의 weighted_sum보다 근소하게 큰 값을 가진다. 이유는, 논문에서 제안한 cover 모델이 실제 환경의 approximation이라서 실제 cost와 평가된 cost간에 오차가 있을 수 있기 때문이다. 그러나, 다른 α값들에 대해서는 Weighted_Sum_Gain이

1.004 ~ 84.995으로, progressive approach의 weighted_sum이 iterative approach의 weighted_sum보다 작은 값을 가진다. 실험 결과로부터, memory 사용량을 significant한 비용으로 고려하는 경우, Xiang et al.[22] 방법에 대비해 제안한 방법의 성능 향상이 크며, 데이터 전송량을 significant한 cost로 고려한 경우에도, 제안한 방법이 Xiang et al. 방법과 competitive한 결과를 보임을 알 수 있다.

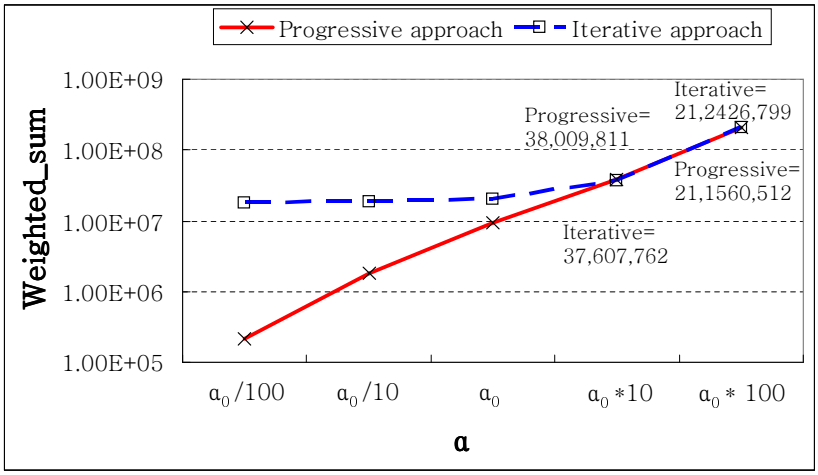


그림 11. α 값을 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

그림 12는 원 질의의 cover를 변화시킨 경우에 두 방법의 weighted_sum을 비교한 결과이다. Weighted_Sum_Gain은 1.019 ~ 2.498의 범위를 가진다. 실험 결과로부터 Xiang et al.이 제안한 방법에 비해 본 논문에서 제안하는 방법이 cover의 전 범위에 대해 더 나은 성능을 나타냄을 알수 있다. 또한 cover가 증가하면, Xiang et al.이 제안한 방법에 비해 제안하는 방법이 가지는 성능 향상 정도가 감소하는 것을 알수 있다. 이는 cover가 커서 1에 근사하는 값을 가지는 경우, 질의 병합을 통해 얻을 수 있는 데이터 전송량 관점에서의 이득이 최대가 되기 때문이다. 이러한 경우, 본 논문에서 제안한 방법이나 Xiang et al.이 제안한 방법 모두 원 질의들을 1개의 merged query로 병합하게 되므로, 두 방법의 총 데이터 전송양과 총 memory 사용양이 모두 유사한 값을 가지게 되어, 두 방법의 weighted_sum들 역시 유사한 값들을 가지게 된다. 따라서, 본 논문에서 제안하는 방법은 원 질의의 cover가 더 작은 값을 가질수록 우수한 성능을 나타낸다.

그림 13은 원 질의의 평균 selectivity를 변화시킨 경우, progressive approach와 iterative approach의 weighted_sum을 비교한 결과이다. Weighted_Sum_Gain은 1.262 ~ 2.666의 범위를 가진다. 실험 결과로부터 Xiang et al.이 제안한 방법에 비해 본 논문에서 제안하는 방법이 selectivity의 전 범위에 대해 더

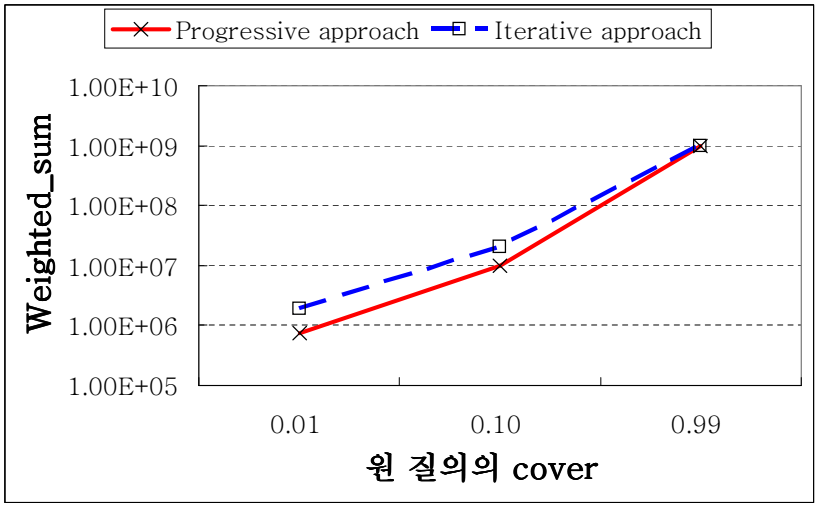


그림 12: 원 질의의 cover를 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

나은 성능을 나타냄을 알수 있다. 또한 selectivity가 증가하면, 제안하는 방법의 성능 향상 정도가 감소하는 것을 알수 있다. 실험 데이터와 분석을 통해 얻은 optimal merging_rate를 비교한 실험에서 이미 언급한 바와 같이, 원 질의의 selectivity가 증가하면, 원 질의의 cover 역시 증가하게 된다. 따라서, selectivity가 증가함에 따라, 제안하는 방법의 성능 향상 정도가 감소하는 이유는, 원 질의 cover의 증가에 기인한다. 따라서, cover의 변화에 따라 제안한 방법과 Xiang et al. 방법의 weighted_sum을 비교한 실험의 결과로부터 selectivity의 변화에 따른 결과를 예측할 수 있다. 결과적으로, selectivity가 증가하면, 제안하는 방법과 Xiang et al.방법의 총 데이터 전송량 및 총 memory 사용량이 각각 유사한 값을 가지게 되어 두 방법의 weighted_sum 역시 유사한 값을 가지게 된다. 따라서, 본 논문에서 제안하는 방법은 원 질의의 selectivity가 더 작은 값을 가질수록 우수한 성능을 나타낸다.

그림 14는 센서 네트워크의 height를 변화시킨 경우, progressive approach와 iterative approach의 weighted_sum을 비교한 결과이다. Weighted_Sum_Gain은 1.973 ∼ 2.220의 범위를 가진다. 실험 결과로부터 Xiang et al.이 제안한 방법에 비해 본 논문에서 제안하는 방법이 height의 전 범위에 대해 더 나은 성능을 나타냄을 알수 있다. 또한, height가 증가하면, 제안하는 방법의 성능 향상 정도가 증가하는 것을 알수 있다. 이유는 센서 네트워크의 height가 증가함에 따라, iterative approach의 총 메모리 사용량이 progressive approach의 총 메모리 사용량에 비해 큰 폭으로 증가하기 때문이다. 즉, iterative approach는 동일한 merged query set을 모든 센서 노드에 중복 저장하는 반면, progressive approach는 merging_rate에 따라 하위 tier로 갈수록 더 작은 개수의 merged query들을 저장한다. 따라서, 본 논문

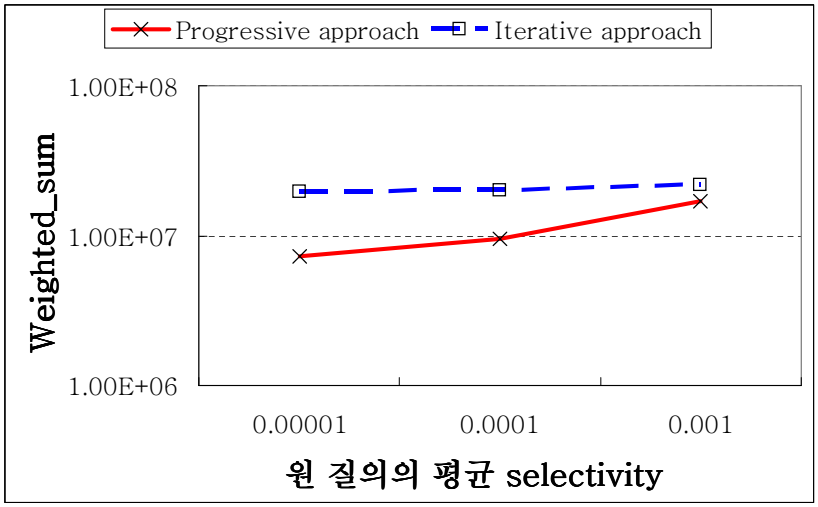


그림 13: 원 질의의 평균 selectivity를 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

에서 제안하는 방법은 sensor network의 height가 더 큰 값을 가질수록 우수한 성능을 나타낸다. 센서 네트워크의 fanout과 원 질의의 차원을 달리한 실험 결과는 appendix B를 참고하기 바란다.

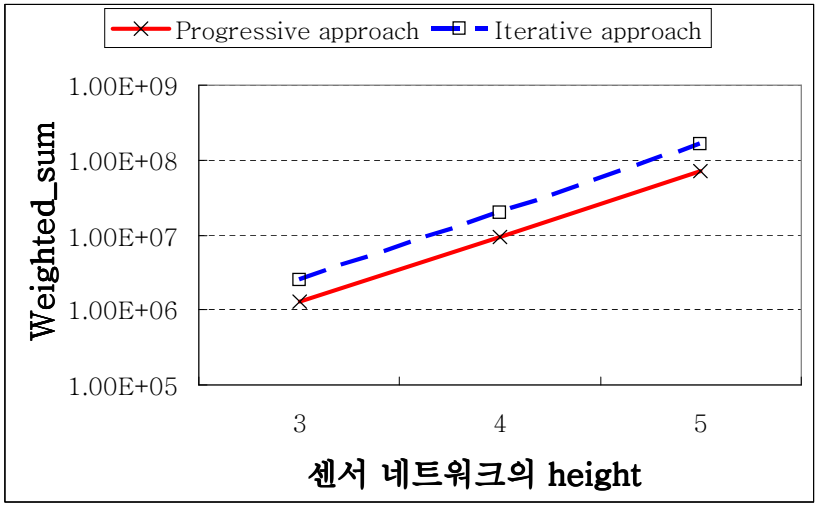


그림 14: 센서 네트워크의 height 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

요약하면, α값을 달리한 실험에서는 iterative approach의 weighted_sum값이 제안하는 방법인 progressive approach의 weighted_sum값이 0.989 $\sim$ 84.995배로 α값이 $\alpha_0 \cdot 10$인 경우를 제외한 파라미터 값들에 대해, 제안하는 방법의 weighted_sum값이 더 작다. 그리고, α가 아닌 다른 파라미터들, 즉 원 질

의의 cover, 원 질의의 평균 selectivity, 원 질의의 차원, sensor network의 height 및 fanout의 값을 달리한 실험들에서는 progressive approach의 weighted_sum값이 iterative approach의 weighted_sum값에 비해 1.019 ∼ 2.666배 작은 값을 가진다. 이를 통해서 본 논문에서 제안한 방법이 Xiang et al.[22]이 제안한 방법에 비해 비용 측면에서 더 나은 성능을 나타냄을 알 수 있다.

6 결론

본 논문에서는 계층적 센서 네트워크에서 연속 범위 질의들을 처리하는 새로운 질의 처리 방법인 점진적 처리를 제시하였다. 본 논문의 공헌을 요약하면 다음과 같다.

첫째, 본 논문에서는 energy와 storage의 trade-off를 고려하는 점진적 처리의 모델을 제시하였다. 점진적 처리 모델의 장점은 서버에 인접한 상위 계층으로 갈수록 high capability의 센서 노드들을 배치하는 계층적 센서 네트워크의 특성을 활용한 것이다. 즉, 상위 계층일수록 더 많은 개수의 질의를 저장함으로써, 노드의 memory를 최대한 활용할 수 있는 장점을 가진다. 또한 상위층에 비해 상대적으로 개수가 많은 하위층의 센서 노드들이 질의를 저장하기 위해 필요로 하는 스토리지 비용을 줄임으로써, 네트워크를 구성하는 전체 비용을 줄이는 장점을 가진다. 본 논문에서는 제안한 모델을 위한 질의 병합 및 질의 처리 알고리즘을 제시하였다.

둘째, 제안한 모델을 기반으로 사용자에 의해 주어진 weight에 따라 총 비용, 즉 energy와 storage의 weighted sum을 최적화 하는 방안을 제시하였다. 그리고, 총 비용이 최소가 되는 계층적 센서 네트워크를 systematic하게 구성하는 방법을 제시하였다.

셋째, extensive한 실험을 통해 제시한 방법의 우수성을 검증하였다. 제안한 비용 모델의 정확도를 검증한 실험의 결과는 비용 모델을 이용하여 분석적으로 구한 optimal 비용에 대해 실제 데이터 및 질의를 이용하여 구한 optimal 비용의 비가 0.929 ∼ 1임을 보였다. 결과로부터 제안한 모델을 이용하면 총 비용이 near-optimal인 네트워크를 구성할 수 있음을 알 수 있다. 또, 제안한 질의 처리 방법의 우수성을 검증한 실험의 결과는 제안한 방법이 Xiang et al.[22]에서 제시한 방법에 비해 최소 1.019배, 최대 2.666배의 성능 향상이 있음을 보였다. 더우기, 네트워크의 height나 fanout이 증가하면, 제안하는 방법이 Xiang et al.에서 제시한 방법에 비해 더 나은 성능을 보인다. 따라서, 제안하는 방법이 large-scale의 네트워크에 적합함을 알 수 있다.

이러한 결과들을 통해 제안한 방법이 energy와 storage의 trade-off를 고려하여 다수의 질의들을 효율적으로 처리하도록 large-scale의 계층적 센서 네트워크를 구성할 수 있는 새로운 framework를 제공함을 알 수 있다. 향후 연구로 데이터와 질의의 다양한 분포 및 질의의 다양한 타입(e.g., aggregate query)들을 고려하여, 모델과 질의 처리 방법을 개선하는 연구를 수행할 예정이다.

Acknowledgements

이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R0A-2007-000-20101-0).

참고 문헌

[1] Abiteboul, S. et al., "The Lowell database research self-assessment.," *Communications of the ACM(CACM)*, Vol. 48, No. 5, pp. 111- 118, May 2005.

[2] Akyildiz, I. F., Melodia, T., Chowdhury, K. R., " A survey on wireless multimedia sensor networks," *Computer Networks*, Vol. 51, no. 4, pp.921-960, March 2007

[3] EarthNet News (available at http://earth.esa.int/object/index.cfm?fobjectid=5106)

[4] Ganu, S., Zhao, S., Raju, L., Anepu, B., Seskar, I., and Raychaudhuri, D., "Architecture and Prototyping of an 802.11-based self-organizing hierarchical adhoc wireless network (SOHAN)," white paper, rutgers university, March 2004

[5] Garey, MR., and Johnson, D.S., "Computers and Interactability: A Guid to the theory of NP-Complementness," W.H.Freeman, New York, NY. 1979

[6] Gui, C., and Mohapatra, P., "Power Conservation and Quality of Surveilance in Target Tracking Sensor Networks," In *Proc. 10th Annual Int'l Conf. on Mobile Computing and Networking(MOBICOM)*, Philadelphia, PA, pp. 129-143, September 2004

[7] Levis, P., Lee, N., "TOSSIM:A Simulator for TinyOS Networks," *white paper*, September 2003 (available at http://www.eecs.berkeley.edu/ pal/pubs/nido.pdf)

[8] Li, X., Kim, Y.-J., Govindan, R., and Hong, W., "Multi-dimensional range queries in sensor networks," In *Proc. 1st Int'l Conf. on Embedded Networked Sensor Systems (ACM Sensys)*, Los Angeles, California, pp. 63-75, November 2003

[9] Lim, H., Lee, J., Lee, M., Whang, K., and Song, I., "Continuous Query Processing in Data Streams Using Duality of Data and Queries," In *Proc. 2006 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 313-324, Chicago, Illinois, June 26-29, 2006.

[10] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W., "TinyDB: An Acqusitional Query Processing System for Sensor Networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, No. 1, pp. 122-173, March 2005

[11] Maple Software (available at http://www.maplesoft.com)

[12] Muller, R., and Alonso, G., "Efficient Sharing of Sensor Networks," *3rd IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems(MASS)*, Vancouver, Canada, pp.109-118, October 2006

[13] Network Simulator -ns-2 (available at http://www.isi.edu/nsnam/ns)

[14] NOVAC project (available at http://www.novac-project.eu)

[15] Park, S. M., "Technical Trend of Sensor Network Node Platform & OS," *전자통신동향분석*, ETRI, vol. 21, No. 1, pp. 14-24, February 2006

[16] Qingguang, Z., Yanling, C., and Juan,.L., "A Lightweight Key Management Protocol for Hierarchical Sensor Networks", In *Proc. 7th Int'l Conf. on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Taipei, Taiwan, pp. 379-382, December 2006

[17] Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., and Yu, F., "Data-Centric Storage in Sensornets with GHT, a Geographical Hash Table," *Mobile Networks and Applications*, vol. 8, No.4, pp. 427-442, August 2003

[18] Singh, M., and Prasanna, V. K., "A Hierarchical Model for Distributed Collaborative Computation in Wireless Sensor Networks ," In *Proc. the 17th Int'l Symposium on Parallel and Distributed Processing(IPDPS)* , Nice, France, pp. 166-258, April 2003

[19] Srivastava, U., Munagala, K., and Windom, J., "Operator placement for in-network stream query processing," In *Proc. the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems(PODS)* , Baltimore, Maryland, USA, pp. 250-258, June 2005

[20] Trigoni, N., Yao, Y., Demers, A. J., Gehrke, J., and Rajaraman, R., "Multi-query optimization for Sensor Networks," In *Proc. 1st Int'l Conf. on Distributed Computing in Sensor Systems(DCOSS)*, Marina del Rey, CA, pp. 307-321, July 2005

[21] Akyildiz, I. F., Melodia, T., and Chowdhury, K. R., "A Survey on Wireless Multimeida Sensor Networks," *Computer Networks*, vol.51, no. 4, pp. 921-960, March 2007

[22] Xiang, S., Lim, H. B., and Tan, K.-L., "Impact of Multi-query Optimization in Sensor Networks," *Workshop on Data Management in Sensor Networks(DMSN'06)*, Seoul, Korea, pp. 7-12, September 2006

[23] Xiang, S., Lim, H. B., and Tan, K.-L., "Multiple Query Optimization for Wireless Sensor Networks," *International Conference on Data Engineering(ICDE'07)*, Istanbul, Turjey, pp. 1339-1341, April 2007

[24] Yu, W., Le, T. N., Xuan, D., and Zhao, W.,"Query Aggregation for Providing Efficient Data Services in Sensor Networks," *1st IEEE Int'l Conf. on Mobile Ad-hoc and Sensor Systems(MASS)*, Florida, pp.31-40, October 2004

부록 A 총 데이터 전송량 및 총 메모리 사용량에 대한 수식의 전개

총 데이터 전송량 수식의 전개

총 데이터 전송량, total_transmission은 각 계층별로 노드들이 생성한 질의 결과가 상위 계층의 노드들을 통해 relay되어 서버까지 전송될 때 발생하는 데이터 전송량들을 합산한 값으로, 수식 8과 같이 표현된다.

$$total\_transmission \quad = \quad \sum_{i=2}^{h}(d_i \cdot \sum_{j=2}^{i}(c_j))$$

$$
\begin{aligned}
where\ c_j &= j^{th}\ 층에\ 위치한\ 센서\ 노드에\ 저장된\ merged\ query들의\ cover \\
d_i &= i^{th}\ 층에\ 위치한\ 모든\ 센서\ 노드들이\ 생성한\ 센서\ 데이터의\ 양
\end{aligned} \tag{8}
$$

수식 8에서 c_j는 수식 4의 cover 모델의 정의와 merging_rate의 정의를 이용하여 아래의 수식9와 같이 표현된다.

$$
c_j = cover(N_Q \cdot x^{j-1}) \tag{9}
$$

1^{st} 층인 server는 N_Q개의 질의를 저장하고, 임의어 sensor node는 그 node의 부모층에 위치한 node에 저장된 merged query 개수에 x를 곱한 개수만큼의 질의들이 저장한다. 따라서, j^{th}번째 층에 위치한 노드는 $N_Q \cdot x^{j-1}$개의 query들을 저장한다.

다음으로, 질의 처리때마다 각 node는 data element를 1개씩 만 생성하는 것으로 가정하였으므로, d_i는 아래의 수식 10과 같이 표현된다.

$$
\begin{aligned}
d_i &= (i^{th}\ 층에\ 위치한\ sensor\ node의\ 개수) \cdot (data\ element의\ 크기) \\
&= f^{i-1} \cdot d
\end{aligned} \tag{10}
$$

따라서, 총 데이터 전송량의 수식을 전개하면 아래와 같다.

$$
\begin{aligned}
total\_transmission &= \sum_{i=2}^{h}(d_i \cdot \sum_{j=2}^{i}(c_j)) \\
&= \sum_{i=2}^{h}(d \cdot f^{i-1} \cdot \sum_{j=2}^{i}(c_j)) \\
&= \sum_{i=2}^{h}(d \cdot f^{i-1} \cdot \sum_{j=2}^{i}(cover(N_Q \cdot x^{j-1}))) \\
&= \sum_{i=2}^{h}(d \cdot f^{i-1} \cdot \sum_{j=2}^{i}(-a \cdot x^{j-1} \cdot N_Q + b)) \\
where\ a &= \frac{s \cdot (1 - c)}{c - s},\ b = 1 + a
\end{aligned} \tag{11}
$$

총 메모리 사용량에 대한 수식의 전개

총 메모리 사용량, total_storage는 각 계층의 노드들에 merged query들을 저장하기 위해 필요로 하는 memory 양을 합산한 값으로 수식 12와 같이 표현될 수 있다.

$$\begin{aligned} total\_storage &= \sum_{i=2}^{h}(m_i) \\ where\ m_i &= i^{th}\ \textit{층에 위치한 모든 sensor node에 저장된 merged query들의 양} \end{aligned} \tag{12}$$

수식 12에서 수식 9에 의해 i^{th}번째 층에 위치한 임의의 노드에 저장되는 merged query 개수는 $N_Q \cdot x^{i-1}$로 표현될 수 있으므로, m_i는 수식 13으로 표현될 수 있다.

$$\begin{aligned} m_i &= (i^{th}\ \textit{층에 위치한 모든 sensor node들에 저장되는 merged query개수의 총합}) \\ &\cdot\ (\textit{질의 1개를 저장하기 위해 필요한 memory양}) \\ &= N_Q \cdot x^{i-1} \cdot 2 \cdot d \end{aligned} \tag{13}$$

결과적으로 총 메모리 사용량의 수식을 전개하면 아래와 같다.

$$\begin{aligned} total\_storage &= \sum_{i=2}^{h}(m_i) \\ &= \sum_{i=2}^{h}(f^{i-1} \cdot N_Q \cdot x^{i-1} \cdot 2 \cdot d) \\ &= \sum_{i=2}^{h}(\ 2 \cdot d \cdot f^{i-1} \cdot N_Q \cdot x^{i-1}) \end{aligned} \tag{14}$$

부록 B 센서 네트워크의 fanout과 원 질의의 차원을 변화시킨 실험의 결과

그림 15는 센서 네트워크의 fanout을 변화시킨 경우, 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 0.941 ∼ 1 이고, Opt_Weighted_Sum_Ratio의 값은 0.994 ∼ 1 이다.

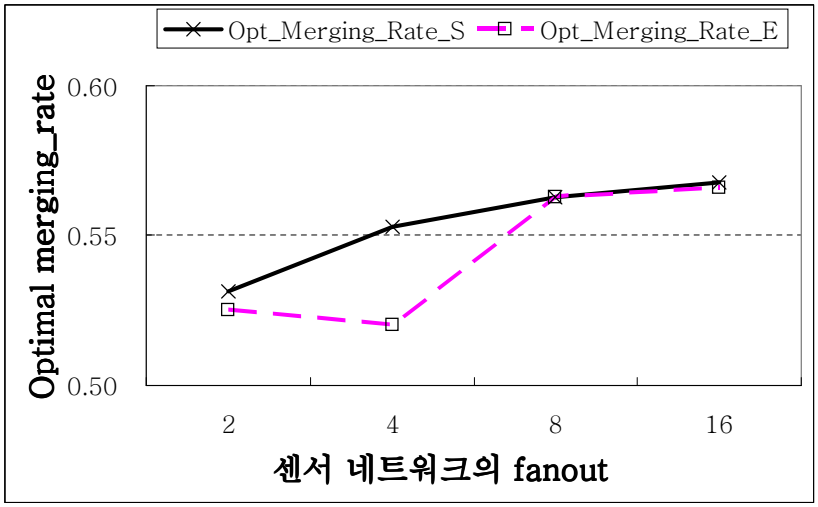


그림 15. 센서 네트워크의 fanout를 달리하였을 때, optimal merging_rate

그림 16은 질의의 차원을 변화시킨 경우, 실험과 분석을 통해 구한 optimal merging_rate 값을 비교한 것이다. Opt_Merging_Rate_Ratio는 0.971 ~ 1.044 이고, Opt_Weighted_Sum_Ratio의 값은 0.997 ~ 1 이다.

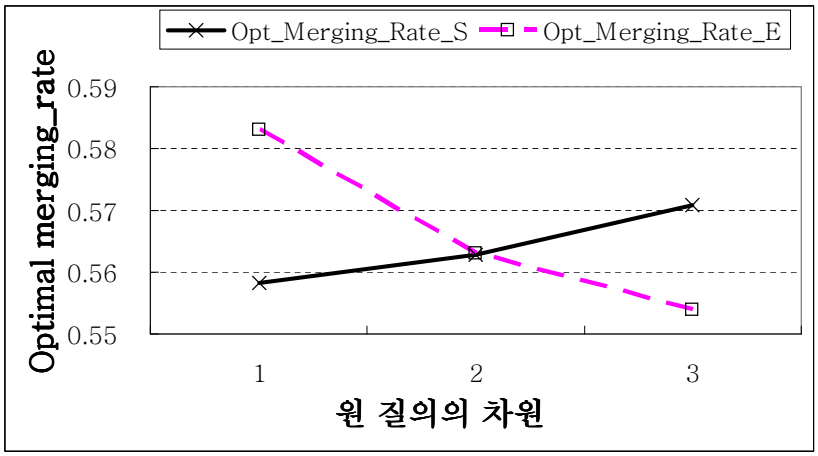


그림 16. 원 질의의 차원을 달리하였을 때, optimal merging_rate

그림 15의 실험 결과와 그림 16의 실험 결과는 실험과 분석을 통해 구한 optimal merging_rate간의 비와 optimal인 점에서의 weighted_sum간의 비가 모두 1에 근사함을 보여 주고 있다. 따라서, 센서 네트워크의 fanout을 변화시킨 경우와 원 질의의 차원을 변화시킨 경우, 제안한 비용 모델로 부터 얻은 비용이 높은 정확도로 실제 비용을 반영할 수 있음을 알 수 있다.

그림 17은 센서 네트워크의 fanout을 변화시킨 경우, 본 논문에서 제안하는 방법인 progressive approach와 Xiang et al.[22]이 제안한 방법인 iterative approach의 weighted_sum을 비교한 결과이다. Weighted_sum_gain은 2.103 ~ 2.159이다.

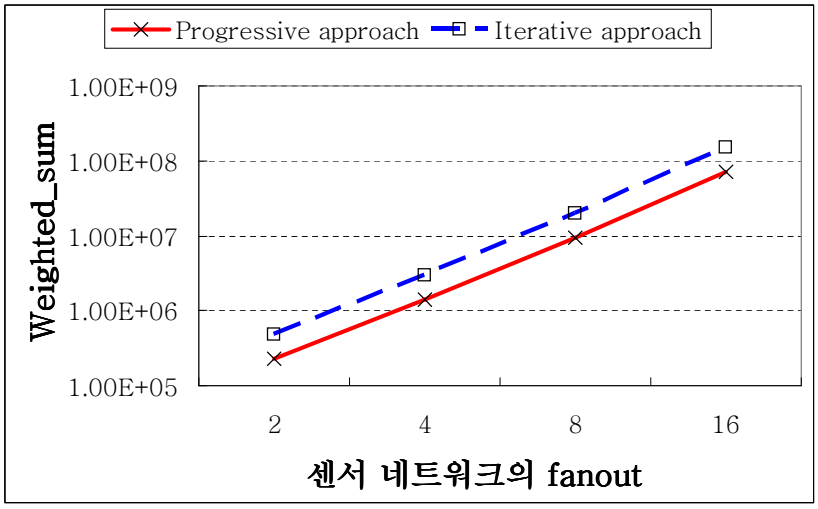


그림 17: 센서 네트워크의 fanout을 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

그림 18은 원 질의의 차원을 변화시킨 경우, 본 논문에서 제안하는 방법인 progressive approach와 Xiang et al.[22]이 제안한 방법인 iterative approach의 weighted_sum을 비교한 결과이다. Weighted_sum_gain은 1.842 ~ 2.157이다.

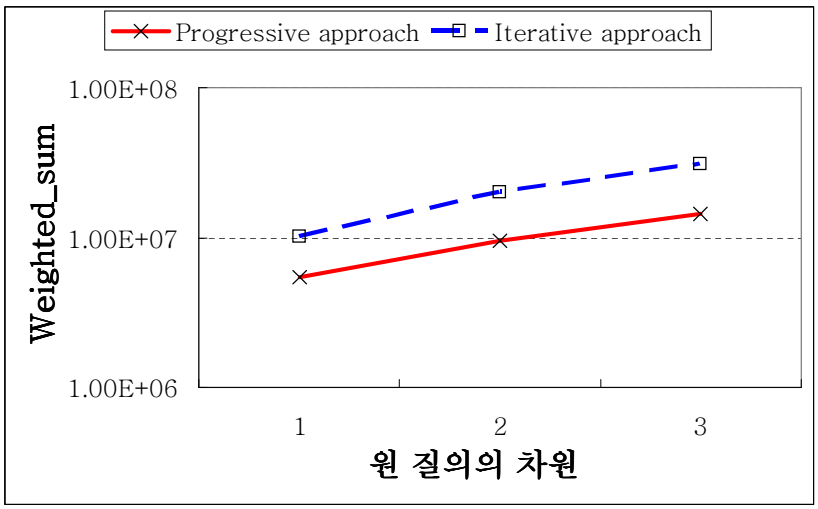


그림 18. 원 질의의 차원을 달리하였을 때, progressive approach와 iterative approach의 weighted_sum.

그림 17의 실험 결과와 그림 18의 실험 결과는 모두 progressive approach가 iterative approach보다 작은 weighted_sum값을 가짐을 보여 주고 있다. 따라서 센서 네트워크의 fanout을 변화시킨 경우와 원 질의의 차원을 변화시킨 경우, 본 논문에서 제안한 질의 처리 방법이 Xinag et al.이 제안한 질의 처리 방법에 비해 총 비용측면에서 우수함을 알수 있다.