韓國IT서비스學會誌
第8卷 第3號
2009年 9月

질의의 지역성을 이용한 효율적인 하이브리드 검색 서비스*

이상환** · 한재일*** · 김철수**** · 황재각****

An Efficient Hybrid Lookup Service Exploiting Localized Query Traffic*

Sanghwan Lee** · Jae-Il Han*** · Chulsu Kim**** · Jaegak Hwang****

■ Abstract ■

Since the development of the Distributed Hash Tables (DHTs), the distributed lookup services are one of the hot topics in the networking area. The main reason of this popularity is the simplicity of the lookup structure. However, the simple key based search mechanism makes the so called "keyword" based search difficult, if not impossible. Thus, the applicability of the DHTs is limited to certain areas. In this paper, we find that DHTs can be used as the ubiquitous sensor network (USN) metadata lookup service across a large number of sensor networks. The popularity of the Ubiquitous Sensor Network has motivated the development of the USN middleware services for the sensor networks. One of the key functionalities of the USN middleware service is the lookup of the USN metadata, by which users get various information about the sensor network such as the type of the sensor networks and/or nodes, the residual of the batteries, the type of the sensor nodes. Traditional distributed hash table based lookup systems are good for one sensor network. However, as the number of sensor network increases, the need to integrate the lookup services of many autonomous sensor networks so that they can provide the users an integrated view of the entire sensor network. In this paper, we provide a hybrid lookup model, in which the autonomous lookup services are combined together and provide seamless services across the boundary of a single lookup services. We show that the hybrid model can provide far better lookup performance than a single lookup system.

Keyword : Lookup, USN, Hybrid, DHT, DChord

논문투고일 : 2009년 07월 17일 논문수정완료일 : 2009년 09월 09일 논문게재확정일 : 2009년 09월 12일

* 이 논문은 MIC/IITA의 IT R&D 프로그램(2006-S022-02, USN 미들웨어 플랫폼 기술 개발)과 2009년도 국민대학교 교내연구비의 지원으로 이루어졌음.

** 국민대학교 전자정보통신대학 컴퓨터공학부

*** 국민대학교 전자정보통신대학 컴퓨터공학부, 교신저자

**** 한국전자통신연구원 RFID/USN 서비스 연구팀

1. 서 론

분산 해쉬 테이블(Distributed Hash Table, DHT) 기법은 최근 몇 년 동안 네트워크 연구자들 사이에서 많은 연구가 진행되어온 분야이다[1-4]. DHT는 간단히 정의하면 여러 개의 분산된 노드 사이에 해쉬 테이블을 구성하면서, 노드 간의 메시지양의 측면에서 부하를 균등하게 분할하고, 검색 속도를 빠르게 하는 시스템이다. 대표적인 예로는 Chord [1], Pastry[2], CAN[3], Leopard[4] 등을 들 수 있다. 이러한 DHT는 기본적으로 key-value의 쌍을 저장하는데 어느 노드에 저장하는지에 관한 규칙을 정하여 특정한 노드에 저장하고, 검색을 하여 이 key-value 쌍을 찾고자 할 때도 같은 규칙을 적용하여 원하는 쌍을 찾게 된다. 즉 DHT는 해쉬 테이블이 복수의 노드로 분산되어 구현된 형태라고 할 수 있다. DHT는 P2P 파일 검색 시스템 등에서 사용되는 경우가 많은데 key에 기반한 검색은 키워드 검색을 어렵게 하기 때문에 일반적인 방식은 아니다.

이 논문에서는 DHT가 이용될 수 있는 한 분야로 유비쿼터스 센서 네트워크(Ubiquotous Sensor Network[5])을 제시하고, 이 USN에 적합하도록 DHT를 변형하여 검색하는 기법을 제시하고자 한다.

센서 네트워크를 이용한 서비스는 일반적으로 물리적인 센서 네트워크와 사용자 응용 프로그램간의 인터페이스를 담당하는 미들웨어가 존재하여 다양한 응용 프로그램이 센서 네트워크의 종류에 상관없이 센서 네트워크를 이용할 수 있도록 한다. 이러한 USN 미들웨어에서 중요한 기능 중 하나가 센서 네트워크, 센서 노드 등에 대한 기본적인 정보-메타데이터-를 유지하고 관리하는 것이다. 이러한 기능을 USN 디렉토리 서비스(USN Directory Service)라 하는데, 이 디렉토리 서비스는 메타데이터에 대한 식별자(Identifier)를 정의하고, 이 식별자를 기반으로 데이터베이스를 구성하여 해당 메타데이터의 내용을 데이터베이스에 저장하면 어렵지 않게 구현할 수 있다.

하지만 센서 네트워크가 실제 실용화 될 경우 다양한 기관에서 여러 가지 센서 네트워크를 설치하고 응용 서비스를 제공하게 된다. 이러한 상태에서 어떤 기관에서 타 기관이 제공하는 센서 네트워크의 메타데이터를 이용하고자 하는 경우 타 기관의 메타데이터를 효율적으로 검색할 수 있는 방법이 필수적인데, 이를 위해 DHT가 사용될 수 있다.

즉 여러 기관이 독립적으로 자신의 메타데이터 관리 서버들을 운영하고 서비스를 제공하고 있을 때, 특정 센서의 메타데이터를 관리하는 메타데이터 관리 서버의 위치를 검색하는데, 그 메타데이터를 관리하는 기관 내의 요청뿐만 아니라, 외부의 요청에 대해서도 그 메타데이터관리 서버의 위치를 알려주도록 할 수 있다.

하지만 매우 많은 수의 관리 서버들 사이에 단일한 구조의 DHT를 사용하는 것은 관리 서버들 사이의 질의 양을 고려하지 않기 때문에 비효율적이 된다. 예를 들면, 어떤 관리 서버들 사이에서는 매우 많은 질의가 일어나는데, 단일한 DHT를 구성할 경우 이 관리 서버들 사이의 경로가 길어지면, 많은 메시지가 발생하여, 네트워크 자원을 불필요하게 많이 소모하게 된다.

따라서 이 논문에서는 USN의 특성에 적합한 Hybrid DHT 모델을 제시하고, 이를 USN의 구조하에서 단일한 DHT 모델에 비해 나은 성능을 보임을 제시한다.

구체적으로 Hybrid DHT 모델은 시스템에 참여하는 모든 노드를 질의량이 많은 노드들 끼리 또는 같은 서비스 사업자에 속하는 노드끼리 등 특정한 기준에 의해 그룹을 지어 이들 간에 하나의 DHT 시스템을 구성하게 한다. 또한 검색의 완결성을 위해 모든 노드를 또 하나의 그룹인 전체 그룹에 속하게 한다. 이를 일반화 하면 한 노드는 자신의 필요에 의해 복수 개의 그룹에 속하고, 각 그룹 내에서는 하나의 DHT를 사용하게 된다. 이 때 사용하는 DHT는 각 그룹에서 독립적으로 선택하면 된다. 즉 서로 다른 종류의 DHT 기법이 사용

될 수 있다는 의미에서 Hybrid DHT 모델이라 이름 붙였다. 이 그룹들은 Disjoint하기 보다는 중첩이 가능하고, 각 그룹에서 사용하는 DHT 시스템은 어떤 종류라도 상관없다. 이러한 구조하에서 질의 패턴이 그룹 내부에서 많은 질의가 발생하고, 그룹 외부로는 적은 질의가 발생하는 질의 지역성(Query Locality)를 가지는 경우, Hybrid DHT 모델이 매우 좋은 성능을 가짐을 보인다.

이 논문의 구성은 다음과 같다. 제 2장에서 관련 연구로 DHT와 USN 미들웨어에 대한 간략한 설명을 하고, 제 3장에서 Hybrid DHT 모델의 구조와 검색 방법에 대해서 설명한다. 제 4장에서는 다양한 질의 지역성을 반영한 성능분석 결과를 제시하고, 제 5장에서 논문의 결론을 제시한다.

2. 관련연구

이 절에서는 DHT 시스템에 대해 간략하게 설명하고, 이 DHT 시스템이 USN 디렉토리 서비스에 어떻게 사용될지 설명한다.

DHT 시스템은 Chord, CAN 등의 시스템이 제안되면서부터 활발하게 연구되기 시작했다. DHT 시스템 중 가장 대표적인 시스템인 Chord 시스템을 간략하게 설명한다. Chord 시스템에 참여하는 모든 노드는 128비트로 이루어진 ID를 하나씩 가지게 된다. 이 ID가 부여되는 방식은 Chord와 상관없이 임의의 방법을 사용하면 되는데, 단 한 가지 조건은 ID가 노드들 간에 고유하다는 점이다. 이렇게 고유한 ID를 가지는 노드들은 몇 개의 다른 노드에 대한 정보를 저장하게 되는데 가장 기본적으로는 ID로 봤을 때 자신보다 다음에 나오는 노드 (successor)와 자신의 바로 전에 나오는 노드(predecessor)의 정보 (IP 주소와 ID 등 메시지를 보내는데 필요한 정보)를 저장한다. ID를 이용하여 다음 노드와 바로 전 노드를 찾을 때는 모듈로 연산을 이용한다. ID를 놓고 봤을 때 ID 2^{128-1} 다음에 ID 0이 나오게 된다. 간단히 말하면, 각 노드가 ID를 이용하여 시계방향으로 원 상에 배치되고, 자신의 앞과 뒤의 노드에 대한 정보를 저장하고 있다고 할 수 있다.

이렇게 구조를 형성한 상태에서, Chord는 key-value 쌍을 저장하는데 key도 ID와 마찬가지로 128비트로 구성된다. 이 때 key-value 쌍이 어느 노드에 저장될지를 결정해야 하는데, Chord에서는 key를 노드가 구성하는 원에 배치시켰을 때 시계방향으로 바로 다음에 나오는 노드에 이 key-value 쌍을 저장하는 것이다. 이렇게 저장이 되면 찾을 때도 마찬가지로, 주어진 key 바로 다음에 나오는 노드를 찾아가서 그 key-value 쌍을 찾아오면 된다. 여기에서 각 노드들 간에 key-value 쌍을 저장하는 양을 균등하게 분배하기 위해 응용에서 제공하는 key를 그대로 사용하는 것이 아니라 key에 Uniform Hash Function을 적용하여 128비트로 변형한 후 사용한다.

문제는 이러한 상황에서의 검색 효율인데, 어떤 응용이 한 노드에 특정한 key의 검색을 요청했을 경우, 그 노드가 이 key를 가지고 있지 않은 경우에는 이 key를 가지고 있는 노드로 이 검색 요청 메시지를 전달해야 하는데, 이 때 노드는 자신의 successor에게 메시지를 전달하게 된다. 이 과정이 모든 노드에서 반복되면 최종적으로 그 key를 보관하는 노드에 메시지가 전달되어, 응용은 원하는 key-value 쌍을 얻을 수 있다. 이러한 메시지 전달 과정은 재귀적(recursive) 방식으로 제공될 수도 있고, 반복적(iterative) 방식으로 제공될 수 도 있다. 하지만 이 방식은 최악의 경우 모든 노드를 탐색해야만 원하는 key-value 쌍을 얻을 수 있게 된다.

이러한 상황에서 검색속도를 높이기 위해 각 노드는 전체 노드 ID 영역을 분할하는데, 자신을 기준으로 차례로 길이가 1인 영역, 2인 영역, 4인 영역, 8인 영역으로 분할한다. 이렇게 하면 전체 영역이 한 노드를 기준으로 128개의 영역으로 분할된다. 그 다음 각 영역의 시작점을 기준으로 맨 처음 나오는 노드의 정보를 저장하게 된다. 따라서 이론적으로 128개의 노드에 대한 정보를 저장하게 되는데 어떤 특정한 영역에는 노드가 하나도 존재

하지 않을 수도 있기 때문에 몇 개의 인접한 영역에 대해서는 같은 노드의 정보를 저장하게 된다. Chord의 저자들은 O(log(N))개의 서로 다른 노드에 대한 정보를 저장하게 됨을 보였다. 여기에서 N은 Chord의 존재하는 노드의 수이다. 이러한 추가적인 정보-분할된 영역과 그 영역을 담당하는 노드의 정보-를 저장하는 자료구조를 Finger Table이라 하였다. 이러한 상황에서 특정 key에 대한 검색 요청이 오면 자신의 Finger Table 상에서 그 노드가 속하는 영역을 찾아서 그 영역을 담당하는 노드에게 메시지를 전달하게 된다. 이 방식을 사용할 경우 O(log(N)) 만큼의 메시지 전달이 있으면 원하는 key-value 쌍을 찾을 수 있음을 보였다.

다른 DHT들도 Chord와 비슷하게 Uniform Hash Function을 사용하여 key를 특정한 공간으로 매핑시키고, 수학적인 규칙을 이용하여 key-value 쌍을 저장하고, 검색하게 된다.

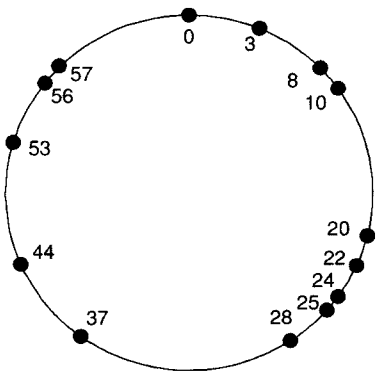
DHT가 많이 사용될 수 있지만 단점은 Uniform Hash Function을 사용하기 때문에 범위 질의를 할 수 없는 등의 다양한 문제점이 있다. 하지만 Uniform Hash Function을 사용하지 않을 경우에는 부하 분산이 되지 않는 문제도 발생하게 된다.

이러한 문제점을 해결하기 위해 DChord[6]에서는 Finger Table을 구성할 때 ID로 분할한 영역을 기준으로 Finger 노드를 선택하는 대신에 노드의 배치 순서를 기준으로 Finger 노드를 선택하였다. 구체적으로, 한 노드의 원 상에 배치된 노드를 기준으로 시계방향으로 나오는 노드 중 첫 번째 노드, 2번째 노드, 4번째 노드, 8번째 노드, 좀 더 일반적으로, 2^i번째 노드를 선택하여 이 노드들의 정보를 저장하였다. 이렇게 하면 모든 노드가 동일한 수의 Finger 노드를 가지게 되어 안정적인 구조를 형성하게 된다. 이러한 구조 하에서 DChord에서는 Chord와 다르게 한 노드가 저장하는 key는 자신의 ID부터 시작하여 자신의 바로 다음 노드의 ID 바로 전까지에 해당한다.

또한 노드가 새로 추가되고 삭제됨에 따라 노드의 Finger Table의 내용이 변화해야 하는데, 이를 위해 DChord에서는 Chord와 마찬가지로 주기적으로 Finger Table의 내용을 최신의 내용으로 변경한다. 이를 위해 다음과 같은 간단한 규칙을 사용하여 Finger Table 수정에 필요한 메시지를 줄이게 된다.

한 노드의 i번째 Finger 노드는 이 노드의 i-1번째 Finger 노드의 i-1번째 Finger 노드이다.

예를 들면, [그림 1]에서 노드 0은 3, 8, 20, 28번을 Finger 노드로 저장하고 있는데, 네 번째 Finger 노드인 28인 세 번째 Finger 노드인 노드 20의 세 번째 Finger 노드임을 알 수 있다.



[그림 1] DChord 상에서의 노드의 배치 예

이제 USN 디렉토리 서비스에 대해 간략하게 설명한다.

USN은 Ubiquitous Sensor Network의 약자로 인간의 생활공간, 생활기기, 기계 등 모든 사물에 센싱 기능, 컴퓨팅 기능, 네트워크 기능을 부여하여, 환경과 상황의 자동인지를 통해 인간에게 최적의 기능을 제공함으로써 인간 생활의 편리성과 안전성을 고도화한다. 이러한 USN은 다양한 분야에 대해 자동화가 가능하게 하며, 이러한 기술을 통해 텔레매틱스, 홈네트워크, 재고관리, 환자관리, 동물관리, 자연재해관리, ITS 시스템 등 인간에게 편리한 다양한 서비스를 제공할 수 있다[5].

이러한 다양한 응용을 가능하게 하는 센서 네트워크는 사용자에 대한 서비스를 제공하는 사업화를 가능하게 하여 다양한 사업자들이 센서 네트워

크 응용 서비스에 참여할 수 있다. 이들 사업자들은 자신의 자본과 인력을 이용하여 넓은 지역에 센서 네트워크를 설치하게 되는데, 자본과 인력의 한계로 인하여 모든 지역을 담당하지 못하게 된다. 따라서 서로의 공동 이익을 위해 다른 사업자들이 사용하는 센서 네트워크를 공유할 필요성이 생기게 된다. 이는 마치 서로 다른 무선 전화 사업자의 가입자들간에 통화가 이루어지는 경우와 비슷하다.

하지만 기존에 제안된 시스템은 서로 다른 사업자간의 센서 네트워크 메타데이터를 공유하도록 하는 방식은 존재하지 않고, 한 사업자 내부에서 자신이 관리하는 메타데이터 관리 서버들 간의 검색기법만 존재한다.

구체적으로는 센서 네트워크와 응용 서비스 간의 인터페이스를 위해 USN 미들웨어가 개발되었다. 이러한 USN 미들웨어에는 다양한 종류가 있는데 COSMOS가 그 중 하나이다. USN 미들웨어에서 중요한 부분 중 하나가 센서 네트워크, 센서 노드 등의 메타데이터를 저장하고, 관리하는 기능이다. 이 기능을 USN 디렉토리 서비스라고 한다. USN 미들웨어는 일반적으로 센서 네트워크와 물리적으로 가까운 거리에 독립적인 컴퓨터상에서 존재하게 되는데, 하나의 미들웨어가 여러 개의 센서 네트워크를 관리할 수 있다. 이러한 USN 미들웨어는 지역적으로 분산된 여러 곳에 위치하게 되는데, 하나의 미들웨어에 속하는 센서 네트워크나 센서 노드들이 다른 미들웨어가 관리하는 센서 네트워크나 센서 노드로 이동하게 되는 경우, 이 새로운 미들웨어는 이동해온 USN 자원(센서 네트워크, 센서 노드 등)가 원래 소속되어 있던 미들웨어의 디렉토리 서비스에 요청하여 이 자원의 메타데이터를 가져와야 한다. 즉 이 과정은 복수의 디렉토리 서비스가 연동하여 메타데이터를 관리해야 하는 것이다. 이러한 목표를 효율적으로 달성하기 위해 DHT를 이용하여 Location Service를 구현할 수 있다. 즉 메타데이터의 식별자(ID)와 그 메타데이터가 어느 디렉토리 서버에 존재하는지에 관한 정보를 DHT로 구성하여 저장한다면 쉽게 특정 식별자의 메타데이터가 존재하는 서버의 정보를 찾아낼 수 있게 된다. 이렇게 하여 복수의 디렉토리 서버들 사이의 연동은 처리될 수 있다.

하지만 하나의 DHT를 이용하는 방식은 이 복수의 디렉토리 서버를 관리하는 기관이 하나일 경우에는 별 문제 없지만 여러 개의 기관이 여러 개의 디렉토리 서버들을 관리하고 이들 기관들이 관리하는 모든 디렉토리 서버를 연동하기 위해서는 하나의 DHT만으로는 효과적이지 못하다. 그 이유는 디렉토리 서버가 많아지는 경우 필연적으로 메시지 전달 경로가 길어지는데, 디렉토리 서버 사이에는 서로 주고받는 질의의 양이 균등하지가 않을 수 있기 때문이다.

이러한 문제점을 해결하기 위해 이 논문에서는 Hybrid DHT 모델을 제시한다.

3. 하이브리드 검색 시스템

이 절에서는 하이브리드 검색 시스템의 구조에 대해 구체적으로 설명한다. 우선 Hybrid DHT의 일반적인 모델에 대해 설명하고, 이 모델을 USN 미들웨어에 적용하는 방법을 설명한다.

Hybrid DHT 모델은 하나의 검색 서비스를 제공하기 위해 방식이 다른 여러 가지 DHT를 섞어서 사용하는 모델이다. 단일 DHT 구조상의 일정한 노드를 그룹으로 묶은 후 각 그룹별로 지역적인 P2P 검색 구조를 형성하고, 이 그룹들 간에 다시 P2P 검색 구조를 형성하는 방식은 [8]에서도 논의 되었다. 하지만 [8]에서는 전체 ID 영역을 연속적인 지역 그룹으로 구분하고, 이 지역 그룹을 하나의 노드로 보고 P2P 구조를 생성한 다음, 각 지역 그룹을 다시 다른 P2P 구조로 생성하기 때문에 각 지역 그룹 내의 ID는 서로 연속적이어야 한다.

이 방식의 장점은 각 그룹의 대표 노드인 Super peer들을 선택할 때 다른 노드들에 비해서 안정적인 노드를 선택하여, 검색 실패 확률을 줄이는데 있다. 각 그룹 내의 노드의 아이디가 연속적-즉 전체 구조에서 다른 그룹에 속한 노드의 아이디가

이 그룹 내의 아이디 중간에 끼어들지 않는다.-이기 때문에 해당 그룹의 모든 노드가 담당하는 아이디의 범위를 super peer의 아이디를 통해 지정할 수 있다. 즉, super peer의 아이디를 그 그룹 내의 가장 큰 id로 하면 그룹간의 P2P 구조가 Chord일 경우 그룹 내의 아이디 정보를 쉽게 요약할 수 있다.

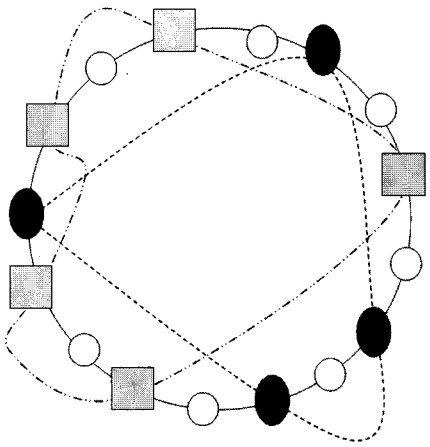
하지만 이 방식은 일반적이지 못한데, 우선 그룹을 형성하는 방식에 있어서 연속된 노드들만으로 그룹을 형성해야 하는데, 일반적인 상황에서 비연속적인 노드를 하나의 그룹으로 형성하는 경우에는 사용하지 못한다. 특히 USN 디렉토리 서비스의 경우 그룹을 형성하는 방법이 각 사업자가 관리하는 위치 서버들을 하나의 그룹으로 형성할 것이기 때문에 연속성을 가정할 수 없다. USN 미들웨어에서는 그 메타데이터의 식별자의 구조에 따라 미들웨어가 존재하는 위치에 의존하는 식별자를 가지게 된다. 각 기관 별로 복수개의 위치 서버를 가지는 경우 지역적으로 분산된 곳에 위치 서버를 배치하게 되고, 이는 ID의 분포가 매우 다양해진다. 다른 기관이 담당하는 위치 서버도 이 기관이 담당하는 지역과 비슷하게 분포하게 되면, 이러한 모든 노드를 하나의 Chord또는 DChord로 포함시켰을때 서로 다른 그룹의 노드들이 섞여 있는 구조를 가지게 된다. 이러한 상황에서는 [8]의 방법을 사용할 수 없게 된다. 또한 하나의 노드가 여러 개의 그룹에 속하는 경우도 생각해볼 수 있는데 [8]의 방식으로는 해결할 수 없다.

이 논문에서 제시하는 모델은 지역 그룹과 상위 그룹의 이분화된 계층적 구조가 아니고, 동등한 레벨의 여러 개의 그룹이 어떤 식으로 서로 연결되는지를 제시한다. 자세한 내용은 아래와 같다.

Hybrid DHT 모델에서의 노드는 여러 개의 그룹에 속할 수 있다. 예를 들면 전체 100개의 노드가 존재하는데 이 중 10개는 Group A에 소속되어 하나의 DChord 구조를 형성하고, 다음 10개는 Group B에 소속되어 또 하나의 DChord 구조를 형성한다. 만약 Group A와 Group B의 모든 노드가 서로의 정보를 공유하고자 할 경우에는 이들 20개 노드가 모여 또 다른 Group C를 형성하게 된다. 각 그룹은 고유한 아이디를 갖는다.

각 노드는 부팅이 될 때 자신이 속할 그룹에 대한 Join 정보를 설정 파일 등을 통해 입력 받는다. Join 정보는 그 그룹의 아이디와 그 그룹의 구조를 포함하고 있고, 그 구조에 Join하기 위한 정보를 포함한다. 예를 들면, DChord의 경우 그 그룹에 속한 한 노드의 정보가 되고, 트리형일 경우에는 루트 노드의 정보가 될 수 있다.

각 노드는 자신의 아이디를 생성하고, 이 아이디를 이용하여 설정 파일에 정의된 각 그룹별로 Join과정을 수행한다. 각 그룹별 Join 과정은 기존의 단일 구조일 때의 Join 과정과 동일한데, 차이점은 그룹의 아이디가 필요하다는 점이다. 또한 각 그룹별로 고유한 Finger Table 등을 유지한다.



[그림 2] Hybrid DHT의 예

Query는 Iterative 방식을 이용한다. 즉 응용으로부터 맨 처음 Query를 받은 노드는 자신이 속한 그룹에서 미리 관리자가 설정한 그룹 탐색 순서에 의해 그룹별로 탐색을 시작한다. 이 때 현재 그룹에서 정보를 찾을 수 없음이 밝혀지면 그 다음 그룹을 탐색하게 된다. 이렇게 모든 그룹을 탐색했을 때도 정보가 없으면 탐색은 실패한다. [그림 2]는 이러한 하이브리드 구조의 한 예를 보여준다.

네모로 표시된 노드들이 하나의 그룹을 형성하여 P2P 구조를 형성하고, 타원으로 표시된 노드들이 또 하나의 자체적인 그룹을 형성하고, 다시 전체 노드들이 하나의 그룹을 형성하여 P2P 구조를 형성하였다. 이러한 구조의 장점은 다음과 같다.

- 매우 유연한 구조 형성 : 각 그룹별로 원하는 구조를 형성할 수 있고, 이 그룹 내의 노드들을 원하는 만큼만 공유할 수 있다. 이 때 공유하고자 하는 노드들만을 선별하여 새로운 그룹을 형성하게 된다.
- 유연한 정보 공유여부 선택 : 한 그룹에서 정책적으로 정보를 공유하지 않고자 할 경우에는 그 노드를 공유 그룹에 포함시키지 않을 수도 있다.
- 검색 분포에 따른 검색 성능 향상 : 일반적으로 노드 상에서 검색이 일어날 경우 각 노드에 가장 밀접한 그룹 내의 정보에 대한 검색이 가장 많이 일어나기 때문에 다양하고 많은 그룹의 노드들이 공유 그룹을 형성하더라도, 일단 밀접한 연관을 갖는 그룹 내에서 검색을 먼저 하게 됨으로 단일한 그룹을 사용할 경우에 비해 검색 속도가 향상된다.

이러한 다중 그룹 하이브리드 구조에서 검색을 하기 위한 알고리즘은 다음과 같다.

SearchLocation()은 노드 n이 주어진 id에 해당하는 IP 주소를 검색하는 과정이다. 이 때 노드 n은 자신이 속한 그룹에서 순서대로 검색을 수행하여 어느 한 그룹에서 IP 주소를 찾으면 더 이상의 검색을 중단하고 그 IP를 return 하고, 그렇지 않으면 다음 그룹을 검색한다. 자신이 속한 모든 그룹을 검색한 후에서 찾을 수 없으면 NULL을 리턴한다. SearchLocation() 알고리즘의 Line 3에서는 한 그룹 내에서의 검색이 진행되는데 이는 그룹 g의 구조에 따라 해당하는 검색을 수행하게 된다. 검색할 그룹의 순서는 트래픽 모델을 기반으로 하여 정하게 된다. 이에 대한 내용은 다음 절에서 자세히 설명한다.

```
Algorithm : n.SearchLocation(id)
1 : G <= Groups that node n belongs to
2 : for all g in G
3 :     ip <= search in g with id
4 :     if (ip != NULL)
5 :         return ip ;
6 :     end
7 : end
8 : return NULL ;
```

4. 성능분석

이 절에서는 Hybrid DHT 모델의 성능을 검증한다. 이를 위해 우선 성능 측정 도구(Performance Metric)을 다음과 같이 정의한다.

- 경로 길이(Path Length) : 질의를 처리하기 위해 방문한 노드의 수
- 검색 시간(Latency) : 사용자의 질의가 첫 번째 노드에 도달한 시점에서 이 노드로 검색 결과가 도착할 때까지의 시간. 단위 초

일반적으로 DHT에서 가장 중요시 하는 성능 측정 도구는 Path Length 이다. 즉 하나의 질의를 처리하기 위해 얼마나 많은 노드를 방문했는지 인데, 이 값이 작을수록 더 좋은 시스템이라 할 수 있다. 앞에서 언급한대로 Chord는 O(log(N))의 성능을 보인다. 하지만 하나의 노드를 방문하는 1 hop의 메시지 전송 과정에서 실제 거리에 따라 메시지 전달 시간이 달라진다. 즉 Path Length는 작아도, 실제 걸리는 시간은 길수가 있다. 이러한 상황을 복합적으로 평가하기 위해 이 논문에서는 위 두 가지 성능 측정 도구를 사용하여 성능을 분석한다.

성능 분석은 시뮬레이션(Simulation)과 Planetlab [9]의 노드를 이용한 실험으로 구성되는데 우선 시뮬레이션을 위한 환경을 설명한다.

우선 시뮬레이션에 사용된 노드의 수는 2^k개 이고, k는 10에서 16까지 사용하였다. 즉 총 노드의 수는 1,024에서 65,536개이다. 이 노드를 대한민국

을 둘러싼 직사각형의 위치 내에 랜덤하게 생성하였다. 실제적으로 노드가 존재할 수 있는 곳은 육지가 대부분이지만, 이 실험에서는 실험의 단순화를 위해 직사각형으로 가정하였다. 노드의 ID는 USN 메타데이터 식별자의 형식을 가지게 되는데 위와 같은 좁은 지역 내의 직사각형에서 노드를 생성함으로써 전체 ID 공간에서 매우 좁은 범위에서만 ID가 선택된다고 할 수 있다.

생성된 노드를 바탕으로 그룹을 구성하였는데, 그룹의 수는 총 G개이고, 각 그룹의 크기는 S이다. 즉 한 그룹에 S개의 노드가 존재한다.

시뮬레이션을 위해 그룹 생성 모델과 Query Traffic Model을 정의하였다.

우선 그룹 생성 모델은 2가지를 정의했는데 아래와 같다.

- Uniform Random 그룹 : 전체 노드에서 G개의 지역 그룹을 생성하는데, 각 그룹의 크기는 S이다. 그룹 멤버는 Random 하게 선택된다.
- Localized 그룹 : 전체 노드에서 G개의 노드를 Random 하게 선택하고, 이 노드를 중심으로 가장 가까운 노드를 S개 선택하여 그 그룹의 멤버 노드로 한다.

그 다음으로 Query Traffic Model을 3가지 정의하였다. 이는 기본적으로 실제 상황에서 어떠한 분포로 Query가 발생되는지에 대한 믿을 만한 정보가 없기 때문에 모델링을 통해 이를 해결하는데 이 때 기존에 다른 연구자들이 많이 사용한 Power law[7]모델링을 도입하고자 한다. 이 실험에서 사용할 Query Traffic Model은 다음 3가지이다.

- Uniform Traffic Model : 이 모델은 Source 노드는 Random 하게 선택하고, Destination 노드를 선택할 때 전체에서 Random하게 선택하는 모델이다.
- Localized Traffic Model 1 : 이 모델은 각 노드들이 그룹을 구성하고, 하나의 노드가 query를 발생시킬 때 그 Destination이 되는 노드가 어떤 특정한 그룹에 속할 확률이 그룹의 크기에 반비례한다고 가정하는 모델이다. 즉 여러 개의 그룹에 속해 있을 때 가장 작은 그룹이 가장 많은 정보 교환이 일어나는 그룹이 되겠다. 좀 더 구체적으로 Destination이 어떤 특정 그룹 g에 속할 확률 P(g)는 다음과 같다.

$$P(g) = \frac{1/|g|^{\alpha}}{W}$$

여기서 $W = \sum_{g \in G} \frac{1}{|g|^{\alpha}}$ 이다. 하나의 그룹이 선택되면 그 내부에서 Random 하게 하나의 노드를 선택하여 이 노드가 Destination이 되도록 한다.

- Localized Traffic Model 2 : Localized Traffic Model 1과 비슷한데 하나의 그룹이 선택될 확률을 계산할 때 그룹의 크기를 바탕으로 계산하는 것이 아니라 그룹 크기의 순서로 계산한다. 우선 그룹 순서 i를 다음과 같이 정의한다.

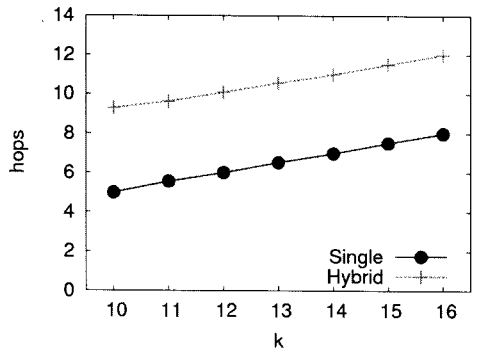
i : 그룹 번호, 그룹을 그룹의 크기에 따라 오름차순으로 정렬하여 나온 순서. i = 1, 2, …

이 때 i번째 그룹이 선택될 확률 P(i)는

$$P(i) = \frac{1/i^{\alpha}}{V}$$

여기서 $V = \sum_{i} \frac{1}{i^{\alpha}}$ 이다. 하나의 그룹이 선택되면 그 내부에서 Random하게 하나의 노드를 선택하여 이 노드가 Destination이 되도록 한다.

이러한 모델 하에서 앞에서 언급한 2가지 성능 측정 도구인 경로 길이와 검색 시간을 측정하였다. 시뮬레이션에서는 실제적인 메시지 전송 시간을 측정할 수 없기 때문에 검색 시간 대신에 Query 메시지의 이동 거리를 계산하였다. 이 실험에서는 그룹의 수 G = 10으로 하고, 그룹의 크기 S = 64로 하였다.

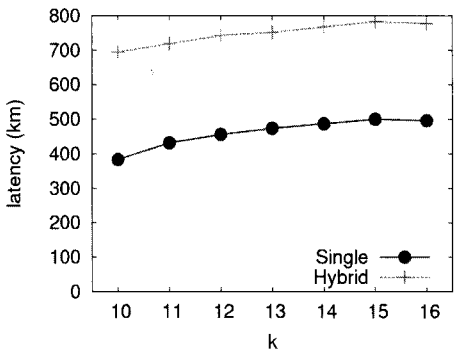


[그림 3] Uniform Traffic Model과 Uniform 그룹 생성 하에서의 경로 길이

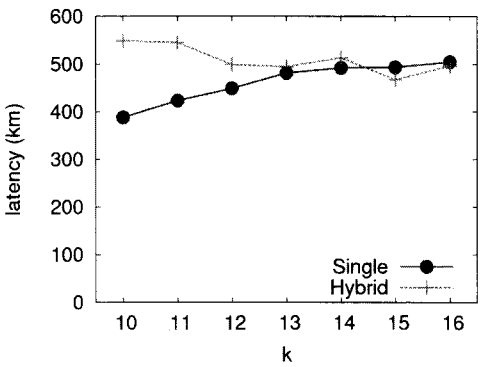
우선 Uniform Traffic Model 상에서의 단일(Single) DChord 시스템을 사용한 경우와 그룹별로 각각의 DChord 시스템을 사용하는 Hybrid 시스템에서의 경로 길이와 검색 시간을 분석한다. [그림 3]에서는 Uniform Traffic Model과 Uniform 그룹 생성 방식을 사용한 경우에 노드의 수에 따른 경로 길이를 보여주고 있다. Hybrid 방식이 Single 방식에 비해 2배 가까운 경로 길이를 가짐을 알 수 있다. 이는 매우 당연한 결과로 Hybrid의 경우 Uniform Traffic Model을 따를 경우 Destination이 Source 노드가 속한 그룹에서 생성될 확률이 매우 낮기 때문에 자신의 그룹 내부에서 검색을 시도하고, 다시 전체 그룹에서 검색을 해야 하기 때문에 더 많은 노드를 방문해야 한다.

검색 시간도 마찬가지로 그룹이 Random하게 선택된 경우에는 경로의 길이에 비례하여 증가한다. [그림 4]에서는 이러한 결과를 보여주고 있다. [그림 4]에서 km 단위로 검색 시간을 표시하였는데, 이는 검색을 위한 패킷이 전송되는 노드간의 거리를 합한 것으로, latency는 기본적으로 propagation delay에 의해 결정되기 때문에 이를 실제 시간으로 변환하지 않고, 거리 단위인 km로 표시하였다. 앞으로 나오는 모든 latency의 단위도 마찬가지로 km로 표시한다.

하지만 그룹 생성이 Localized 그룹 생성 방식을 따른 다면 지역 그룹 내의 검색은 Hop 수는 많을 지라도 전체적으로 이동 거리가 짧게 된다. [그림 5]에서는 Uniform Traffic Model과 Localized 그룹 생성 하에서의 검색 시간을 보여주고 있는데, 노드의 수가 증가할수록 검색 Single 시스템과 Hybrid 시스템과의 차이가 없음을 알 수 있다.



[그림 4] Uniform Traffic Model과 Uniform 그룹 생성 하에서의 검색 시간

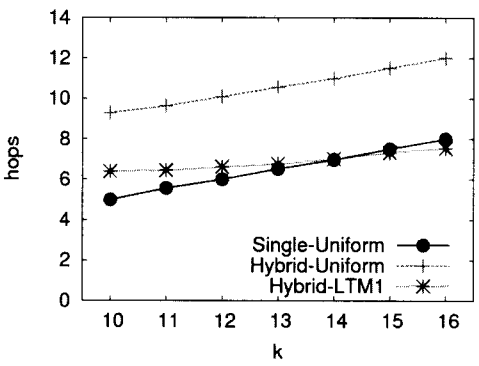


[그림 5] Uniform traffic Model과 Localized 그룹 생성 방식 하에서의 검색 시간

이들 실험의 결과로 보면 Query Traffic이 Uniform 일 경우는 Hybrid 모델이 큰 성능 향상을 보이지 않는다. 하지만, Query Traffic이 지역성을 띤다면, 지역 그룹 내의 검색을 먼저 하기 때문에 더 나은 성능을 보일 것이다. 이러한 예측은 다음 실험 결과로 알 수 있다.

[그림 6]에서는 Localized Traffic Model 1과 Uniform 그룹 생성 하에서의 경로 길이를 보여준

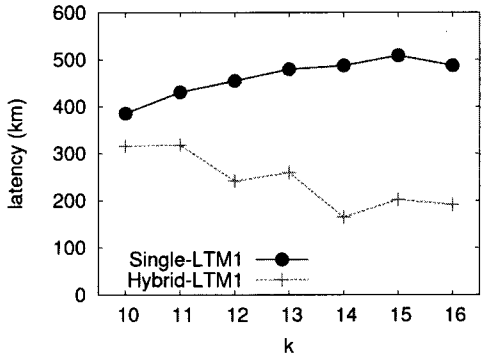
다. alpha 값은 0.0을 사용한 경우이다. alpha = 0.0 이면, 한 노드가 속한 그룹들 중 하나의 그룹이 Destination으로 선택될 확률이 동일함을 의미한다. 따라서 전체에서 Random 하게 Destination을 선택하는 것에 비해 지역 그룹의 노드가 Destination이 될 확률이 높아지게 된다. 따라서 전체적으로 Hybrid 시스템의 경로 길이가 짧아져서 Uniform Traffic 모델을 사용한 경우와 비슷한 성능을 보인다.



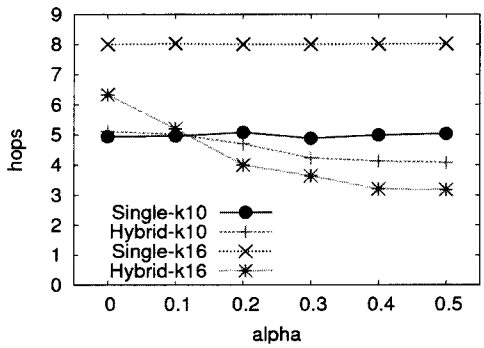
[그림 6] Traffic 모델에 따른 경로 길이, alpha = 0.0

하지만 그룹 생성 방식이 Localized일 경우에는 Hop수가 Single 시스템과 비슷하더라도 검색 시간은 큰 차이를 보인다. [그림 7]은 Localized Traffic Model 1과 Localized 그룹 생성 하에서의 검색 시간을 보여주고 있다. Hybrid 시스템이 Single 시스템에 비해 매우 낮음을 알 수 있다.

이러한 경향은 alpha 값이 커짐에 따라 더 심화된다. 이는 Destination이 지역 그룹에서 선택될 확률이 높아짐에 따라 Hybrid 시스템의 성능이 높아지기 때문이다. [그림 8]은 Localized Traffic Model 1과 Localized Group 생성 방식 하에서 alpha 값의 변화에 따른 경로길이를 보여주고 있다. alpha 값이 커질수록 Hybrid 시스템의 경우는 경로 길이가 짧아짐을 알 수 있다. 전체 노드의 수에 따른 결과를 살펴보기 위해 k = 10인 경우와 k = 16인 경우를 함께 비교하였다.



[그림 7] Localized Traffic Model 1과 Localized 그룹 생성하에서의 검색 시간, alpha = 0.0



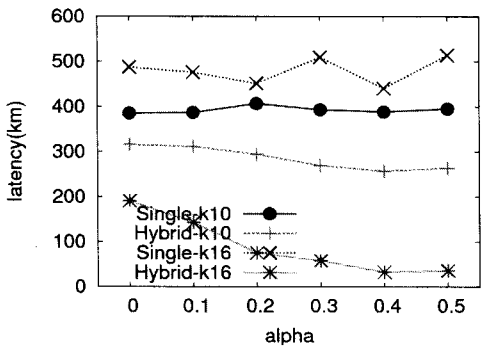
[그림 8] Localized Traffic Model 1에서 alpha 값에 따른 경로 길이

한 가지 유의할 점은 Hybrid 시스템에서 k = 10인 경우와 k = 16인 경우를 비교해 볼 때 k = 10일 경우의 감소폭이 k = 16인 경우보다 작은데, 그 이유는 k = 10일 경우, G = 10이고, S = 64이기 때문에 하나의 노드가 여러 개의 그룹에 속할 확률이 증가하여 여러 개의 지역 그룹을 검색해야 하는데 k = 16일 경우는 대부분의 노드가 하나의 지역 그룹만을 갖게 되고, 이 지역그룹이 선택될 확률이 1에 가까워지기 때문에 지역 그룹의 크기인 64개의 노드내에서의 평균 경로 길이인 1/2*log(64) = 3에 가까운 경로 길이를 보여주고 있다.

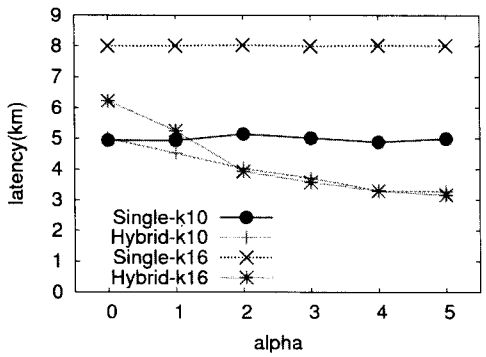
Hop 수의 감소에 따른 전체 검색 시간의 감소도 예상되는데, Localized 그룹 생성 방식을 사용하는 경우에는 그 감소폭이 매우 높다. [그림 9]에서는 Localized Traffic Model 1에서 alpha 값에

따른 검색 시간을 보여주고 있다. 기본적으로 hop 수가 줄어든 상황에서 그룹 내부의 메시지 전송은 짧은 거리만을 이동하게 되기 때문에 전체적인 검색 시간이 짧아지게 되었다. Hybrid-k 10의 검색 시간이 Hybird-k 16에 비해 더 큰 이유는 그룹을 구성할 때 가장 가까운 노드 S개를 선택한 것이기 때문인데 노드의 수가 많을수록 노드 간의 거리가 서로 가까워지기 때문이다.

이러한 결과는 Localized Traffic Model 2를 사용할 경우에도 관찰할 수 있다. 그 이유는 기본적으로 Localized Traffic Model을 사용할 경우 작은 그룹에서의 검색이 많아지기 때문이다. [그림 10]에서는 Localized Traffic Model 2와 Localized 그룹 생성 방식을 사용한 경우의 경로 길이를 alpha 값에 따라 보여주고 있다.



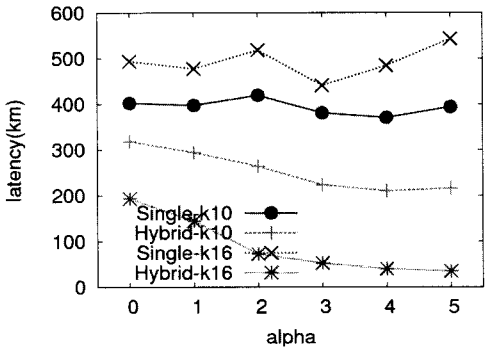
[그림 9] Localized Traffic Model 1에서 alpha 값에 따른 검색 시간



[그림 10] Localized Traffic Model 2에서 alpha값에 따른 경로 길이

Localized Traffic Model 1과 마찬가지로 alpha 값이 커질수록 Hybrid 모델의 경로 길이가 짧아짐을 알 수 있다. 한 가지 다른 점은 Hybrid-k10의 경우인데, Localized Traffic Model 1인 경우에서는 alpha 값이 큰 경우 Hybrid-k 16에 비해 큰 경로 길이를 보여주었는데, Localized Traffic Model 2에서는 Hybird-k16과 동일한 성능을 보여준다. 그 이유는 Localized Traffic Model 2의 경우 한 노드가 속한 그룹을 크기순으로 정렬을 하고 그 순서를 이용하기 때문에 크기가 동일함에도 불구하고, 첫 번째 그룹 내에서 Destination이 존재하는 경우가 많아지게 되어, 결과적으로 지역 그룹 하나에서만 검색을 하는 경우와 비슷한 경로 길이를 보여주게 되는 것이다.

마찬가지로 검색 시간에서도 Localized Traffic Model 1이나 Localized Traffic Model 2에서 비슷한 결과를 보여준다. [그림 11]은 Localized Traffic Model 2에서 alpha 값에 따른 검색 시간을 보여주는데, alpha 값이 커질수록 Hybrid 시스템의 검색시간이 짧아짐을 알 수 있다.

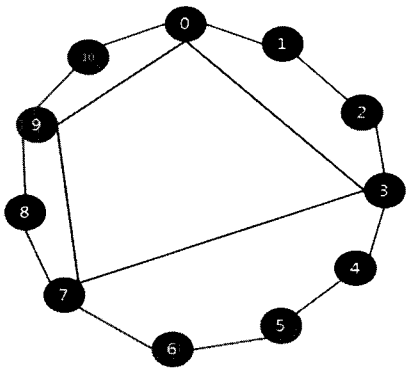


[그림 11] Localized Traffic Model 2에서 alpha값에 따른 검색 시간

이상과 같은 결과를 정리해보면, 서로 다른 그룹의 노드들이 서로 간의 연동을 위해 합쳐야 할 경우에, 전체를 하나로 묶는 단일 시스템으로 구성하기 보다는, 각 그룹 별로 독자적인 검색 시스템을 그대로 유지한 상태에서 다시 전체를 하나로 묶는

검색 시스템을 건설하는 Hybrid 방식이 경로 길이나 검색 시간의 측면에서 더 나음을 알 수 있다.

시뮬레이션의 경우 네트워크 구성에 따른 실제 메시지 전송 시간 등을 측정할 수 없기 때문에 물리적인 거리로 Latency를 측정하였는데, 실제 환경에서의 동작을 분석하기 위해 Planetlab [9]에서 실험을 수행하였다. [그림 12]는 11개의 Planetlab 노드를 이용한 Planetlab 실험 환경을 보여주고 있다. 총 11개의 노드로 이루어져 있고, 이중 4개의 노드가 지역 그룹을 형성하고 있다.



[그림 12] Planetlab 실험 환경

여기에 사용된 각 노드의 도메인 네임은 다음과 같다.

0 : pl2.csl.utoronto.ca
1 : planetlab4.wail.wisc.edu
2 : planetlab1.cs.wisc.edu
3 : planetlab01.cs.washington.edu
4 : planetlab3.wail.wisc.edu
5 : planetlab1.dtc.umn.edu
6 : planetlab1.eecs.umich.edu
7 : planetlab3.postel.org
8 : planetlab04.cs.washington.edu
9 : planetlab3.csres.utexas.edu
10 : planetlab2.csres.utexas.edu

도메인 네임에서 알 수 있다시피 대부분의 노드들이 미국과 캐나다에 존재한다. 이 경우라도 한반도에 비해서는 더 넓은 지역에 노드들이 분포되어 있다. 이 실험에서는 각 노드로부터 다른 모든 노드로의 Hop 수와 Latency를 측정하였다. 지역 그룹에 속한 4개의 노드가 아닌 경우에는 모두 전체 그룹에만 속해 있기 때문에 Single 시스템이나 Hybrid 시스템 모두 동일한 결과를 보인다.

〈표 1〉 서버 0에서 다른 노드로의 경로 길이 및 검색 시간

dest	Hybrid		Single	
	hop	lat(ms)	hop	lat(ms)
0	0	1339	0	1562
1	1	2123	1	1826
2	1	2519	1	2107
3	1	3385	2	3747
4	2	3935	1	2061
5	3	4441	2	2607
6	3	5576	2	2717
7	1	4079	3	5400
8	2	6066	1	3403
9	2	4930	2	3576
10	4	6849	2	3669

<표 1>은 서버 0에서 다른 노드로의 경로 길이와 검색 시간을 Hybrid 시스템과 Single 시스템의 경우로 구분하여 보여준다. Destination 노드가 같은 지역 그룹에 속하는 경우는 배경색을 칠해 구분하였다. 간단하게 볼 수 있는 바와 같이 같은 그룹 내의 경우가 일반적으로 경로 길이도 짧고, 검색 시간도 짧다는 것을 볼 수 있다. 서버 0번 이외에 3, 7, 9번에서도 비슷한 결과를 보인다.

하지만 특수한 경우에는 같은 지역 그룹에 존재하는 Destination이라 할지라도 Single 시스템에서 더 작은 경로 길이를 가지는 경우가 있는데 예를 들면 노드 3에서 노드 0으로의 검색의 경우 지역 그룹 내에서는 2 Hop이 필요한데, Single 시스템에서는 1 Hop만 필요하다. 그 이유는 0번 노드가 3번 노드의 8번째 노드이기 때문에 3번 노드의 Finger Table에 0번이 포함되어 있기 때문이다. 하

지만 이런 경우는 전체 노드의 수가 증가할 경우 특수한 경우에 해당되며 일반적인 경우는 아니다.

5 결 론

이 논문에서는 서로 상이한 검색 시스템을 연동하고자 할 때 전체를 하나로 묶는 시스템으로 구성하기 보다는 각 그룹을 고유한 시스템으로 구성한 다음 다시 전체적인 시스템으로 구성하는 Hybrid 모델을 사용하는 것이 전체적인 성능을 향상시킴을 보여주었다.

이 논문에서는 다양한 Query Traffic Model과 그룹 생성 방식을 이용하여 성능을 분석하였는데, 실제 환경에서 Query Traffic Model을 미리 선험적으로 알 수 없기 때문에 각 노드들이 자신의 Query를 바탕으로 모델을 스스로 만들어나가는 방법도 하나의 미래의 연구 방향이다. 또한 현재는 여러 그룹이 존재할 경우 한 가지 검색 순서만을 사용하였는데 각 노드별로 휴리스틱을 사용하여 자신이 속한 여러 그룹 내에서 동적으로 검색 순서를 변경하여 성능을 향상시키는 방식에 대한 연구도 계획 중이다.

또한 이러한 연구 방향은 향후 센서 네트워크상에서 디렉토리 서비스를 제공하고자 할 때 필요한 요소 기술을 좀 더 발전시킬 수 있게 되며, 현재 많은 각광을 받고 있는 Service Computing의 활성화에도 기여할 것으로 기대된다.

참 고 문 헌

[1] Stoica, I., R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord : A scalable peer-to-peer lookup service for Internet applications", in *Proceedings of ACM SIGCOMM(2001)*, San Diego, CA.

[2] Rowstron, A. and P. Drushel, "Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems", in *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)(2001)*, Heidelberg, Germany.

[3] Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network", in *Proceedings of ACM SIGCOMM(2001)*, San Diego, CA.

[4] Yu, Y., S. Lee, and Z. L. Zhang, "Leopard : A locality-aware peer-topeer system with no hot spot", in *Proceedings of the 4th IFIP Networking Conference (2005)*, Waterloo, Canada.

[5] Weiser, M., "Some computer science issues in ubiquitous computing", *Communications of the ACM*, Vol.36, No.7(1993), pp.75-84.

[6] Lee, S. H., J. Han, and C. S. Kim, "DChord: An Efficient and Robust Peer-to-Peer Lookup System", submitted to a Journal.

[7] Adamic1, L. A. and A. H. Bernardo, "Zipf's law and the Internet", *Glottometrics*, Vol.3 (2002), pp.143-150.

[8] Garces-Erice, L., E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-Keller, "Hierarchical Peer-to-Peer Systems", *Euro-Par 2003, LNCS*, Vol.2790(2003), pp.1230-1239.

[9] Planetlab, https://www.planet-lab.org/.

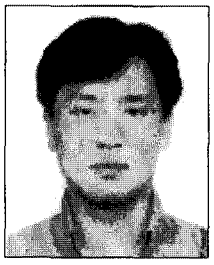
◈ 저 자 소 개 ◈

이 상 환 (sanghwan@kookmin.ac.kr)

1993년 서울대학교 계산통계학과에서 학사, 1995년 서울대학교 전산과학 전공에서 석사를 취득하고, 현대전자에서 2000년 5월까지 근무하였다. 이후 2005년 University of Minnesota, Twin Cities에서 Ph.D를 취득하고, 국민대학교 컴퓨터공학부 조교수로 재직 중이다. 현재 P2P Multimedia Streaming System, P2P 게임 등 P2P 관련 연구와 센서 네트워크 보안에 대한 연구를 수행하고 있다. 관심분야는 웹 프로그래밍, 네트워크 보안, P2P, Routing 등이다.

한 재 일 (jhan@kookmin.ac.kr)

연세대학교에서 이학사, 미국 Syracuse University에서 전산학 석사와 박사학위를 취득하고, 국민대학교 컴퓨터학부 교수로 재직 중이다. 현재 분산처리, 객체지향 시스템, RFID/USN 미들웨어와 SaaS 플랫폼 보안에 대한 연구를 수행하고 있다. 관심분야는 분산 시스템, 객체지향 시스템, 미들웨어, RFID/USN 기술, SaaS, 컴퓨터 및 네트워크 보안, 지능형 시스템, 공개 소프트웨어 등이다.



김 철 수 (chulsu1@etri.re.kr)

한밭대학교 컴퓨터과학과에서 학사, 충남대학교 컴퓨터과학과에서 전산학 석사학위를 취득하고, 한국전자통신연구원 RFID/USN 서비스연구팀 연구원으로 재직 중이다. 관심분야는 유비쿼터스 컴퓨팅 및 네트워킹, 텔레매틱스 서비스 시스템, 메타데이터 처리 및 관리, XML, XML 데이터베이스 등이다.

황 재 각 (jghwang@etri.re.kr)

서울산업대학교 전자계산학과에서 학사, 충북대학교 전산학과에서 공학석사학위를 취득하고, 한국전자통신연구원 RFID/USN 서비스연구팀 책임연구원으로 재직 중이다. 관심분야는 RFID/USN 기술, 데이터베이스 시스템, 분산컴퓨팅 등이다.