분산 네트워크 환경에서 Super Peer를 이용한 Mobile Peer-to-Peer system

한정석*, 송진우*, 이광조*, 양성봉*
*연세대학교 컴퓨터과학과
e-mail: {leohan, fantaros}@cs.yonsei.ac.kr, kjl5435@gmail.com,
yang@cs.yonsei.ac.kr

Mobile Peer-to-Peer system using Super peers for Distributed Network Environment

Jung-Suk Han*, Jin-Woo Song*, Kwang-Jo Lee*, Sung-Bong Yang*
*Dept of Computer Science, Yonsei University

요 약

모바일 기기 사용이 급증함에 따라 모바일환경에서 이루어지는 P2P방식 연구가 활발히 진행되고 있다. 본 논문에서는 기존의 모바일환경에서의 P2P방식이 지닌 peer들 사이의 multi-broadcasting 방식의 문제점을 보완하고, 새로운 routing table을 구축하기 위해 peer들을 2개의 계층으로 구분하였다. 즉, peer들을 super peer들과 각 super peer에 의해 관리되는 sub peer들로 구분하였다. 파일의 탐색과 전송은 super peer들이 관리하므로, 기존의 불필요한 multi-broadcasting 방식을 피할 수 있다. 본 논문에서는 분산 네트워크를 위한 peer의 계층화 작업을 설계하였다. MIS(Maximal Independent Set)알고리즘을 이용하여 peer들의 계층화 작업을 외부의 도움 없이 peer들간의 통신으로 할 수 있게 만들었다. 이처럼 peer들을 super peer와 sub peer로 구분하면 불필요한 broadcasting을 피할 수 있어 시스템 성능이 향상되며, 이를 실험을 통하여 증명하였다.

1. 서론

최근 많은 사람들이 모바일 기기인 핸드폰, PDA를 사용하게 되면서, 모바일환경에서의 P2P가 많은 관심을 받고 있다. 모바일은 이동성을 가지는 객체이고, 각각의 모바일은 통신범위라는 제약조건이 있기 때문에 기존의 유선환경의 P2P방식을 사용할 수 없다. 때문에 모바일환경의 P2P는 기존의 유선 P2P와는 다른 알고리즘으로 개발이 되어왔다. 대표적인 모바일환경의 P2P로는 ORION(Optimized Routing Independent Overlay Network)[4]이 있고, 이에 대해서 간략하게 알아보도록 하자.

2. 관련연구

2.1 Mobile P2P

MANET(Mobile Adhoc Network)은 전송범위가 정해진 노드들 사이의 네트워크를 구축하기 위한 방법으로 고정된 라우터나 호스트 등을 필요로 하지 않는 네트워크 구축 방법이다. MANET은 군사적 목적으로 초기에 연구되어 왔다. MANET은 네트워크를 구축하는 각각의 구성요소들이 라우터로 동작하므로 MANET을 구축하기 위해서는 이동성에 적합한 라우팅 프로토콜이 필요하다. IETF의 MANET 워킹그룹(working group)의 가장 큰 초점은 정적이고 동적인 위상을 가진 노드의 무선 라우팅 어플리케이션을 표준화 하는 것이다. 무선 인터페이스는 몇 개의 독특한 라우팅 인터페이스 특색을 가지며 무선 라우팅 영역에 있는 노드의 위상은 움직이나 그 외의 요소에 의해서 확장될 수 있다는 것이 기본적인 디자인이다.

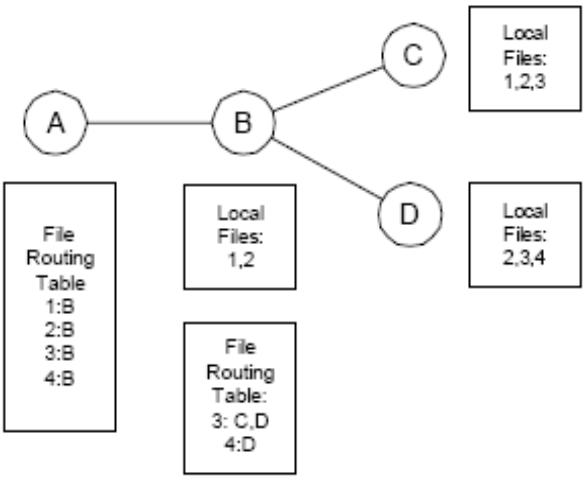
2.2 기존의 Mobile P2P인 ORION

기존의 유선 환경에서의 P2P시스템은 냅스터(Napster)[1], 오픈냅(OpenNAP)[2], 아이알씨(IRC @find) 와 같은 서버-클라이언트 방식과, 뉴텔라(Gnutella)[3]나 프리넷(Freenet)와 같이 모든 peer들이 네트워크에서 동등하게 참여하는 방식이 있다. P2P 네트워크 구조는 1969년 4월 7일에 제정된 RFC(Request for Comments)란 인터넷 규약의 초기 버전에 등록되어 있는 표준이다. 모바일환경에서의 P2P로는 ORION이 대표적이다. ORION은 라우팅을 위한 테이블과, 파일관리를 위한 파일 테이블을 따로 관리하여 효율적인 파일 공유를 꾀하였다. 라우팅 방법은 AODV (Ad-hoc On-demand Distance Vector)[5] 라우팅 테이블과 유사한 구조이다.

ORION은 네트워크 링크가 구축이 되면, flooding 방식을 사용하여 이용자가 원하는 파일을 찾는다. ORION이 (그림 1)처럼 라우팅 테이블이 구성했다고 가정하자. A 사용자가 파일 1,2,3,4를 찾기를 원한다면, 자신한테 연결된 모든 peer들에게 multi-broadcasting을 한다. 이 경우, B 사용자가 A 사용자가 보낸 query를 받고, B는 같은 query

를 자신에게 연결된 사용자 C, D한테 보낸다. 사용자 B는 A가 보낸 query를 보고 자신의 local file table을 살펴본다. 그리고 B는 A가 원하는 파일 1,2,3,4중 1,2를 가지고 있다고 A에게 *Response* 메시지를 보낸다. C와 D의 경우도 자신의 local file table을 참고하여 B로부터 온 A가 보낸 메시지를 본다. 자신의 file table에 있는 file 1,2,3(C 경우), file 2,3,4(D 경우)를 *Response* 메시지로 보낸다. 이와 같은 경우가 Flooding방식이다. Flooding 방식은 자신에게 연결된 모든 peer들에게 같은 query를 보내주는 방식으로 peer들의 연결이 많이 있을수록 query의 전송 횟수가 기하급수적으로 늘어나게 된다. MANET의 규모가 커지면 peer들 간의 routing table이 복잡해지게 되는데, 그 상태에서 flooding을 하게 되면 query의 수가 많아질 수밖에 없다.

본 논문은 라우팅 테이블을 효과적으로 구성하고, broadcasting 방식을 flooding 방식이 아닌 특정 peer에게 보내는 방식을 택하여 파일 탐색 시 query의 수를 줄이도록 하였다.



(그림 1) ORION의 flooding 방식

3. Super Peer를 적용한 방식

기존의 모바일 P2P방식은 모든 peer들이 동등한 계층에서 자신 주위에 연결된 모든 이웃 peer들에게 multi-broadcasting 방식을 사용하였다. 본 논문에서는 peer들을 단일 계층이 아닌 두 개의 계층으로 나눴다. 즉, 주위 peer들을 관리하는 *super peer*들과 super peer에 의해 관리되는 *sub peer*들로 나누었다. super peer는 sub peer를 관리하는 entry를 가지게 되고, sub peer의 모든 정보(*id*, 주소, 파일목록)를 가지고 있다. 한 peer가 파일을 찾고자 한다면, 먼저 자신을 관리하는 super peer로 가서 원하는 파일이 있는지 없는지를 살펴본다. 만약, 해당 super peer가 관리하는 entry안에 찾고자 하는 파일이 있다면, super peer는 sub peer에게 찾고자 하는 파일이 있는 peer의 주소를 알려준다. 만약, super peer의 entry안에 찾고자 하는 파일이 없다면, super peer는 다른 super peer에게 해당 파일을 찾아달라고 요청한다. 요청을 받은 다른 super peer들은 자신의 entry에서 찾아보고, 있으면 *Response* 메시지를 보낸다.

이러한 방법을 이용하면 ORION처럼 자신에게 연결된 모든 peer들에게 multi-broadcasting하는 방식을 피할 수 있어 탐색하는데 필요한 query의 수와 시간을 줄일 수 있다.

3.1 MIS 방식

MIS 방식은 MIS 알고리즘을 이용하여 super peer를 정하는 방식이다. 본 논문은 MIS 알고리즘으로 Luby's randomized MIS 알고리즘을[6] 사용하였다. Luby's randomized MIS 알고리즘에 따르면, 모든 peer들은 처음에 생성이 되면 자신만의 임의의 숫자를 가진다. n이 네트워크에 있는 전체 peer수라고 한다면, 임의의 수의 범위는 1부터 n^4사이가 된다. Luby는 임의의 수의 범위를 위와 같이 정의 한다면, 모든 peer들은 고유한 임의의 수를 가지게 된다고 증명하였다. 이렇게 모든 peer들이 고유의 임의의 숫자를 가지고 있다면 super peer와 sub peer의 구성 방법은 다음과 같다.

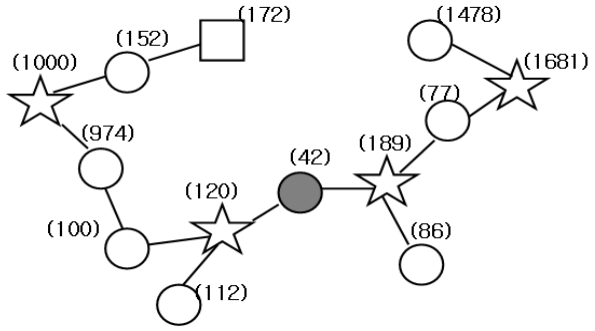
과정 1)Peer들을 계층화하기: 하나의 peer가 있다면 그 peer를 중심으로 통신범위 안에 있는 여러 다른 peer들이 있을 것이다. 네트워크에 있는 모든 peer들은 자신의 이웃 peer들의 임의의 수와 자신의 임의의 비교하여, 모든 이웃들보다 자신의 임의의 수가 크다면 그 peer는 super peer가 되고, 그렇지 않다면 sub peer가 된다. 만약, 어떤 한 peer가 super peer가 되었다면 그 주위에 있는 이웃 peer들은 모두 sub peer가 된다. 모든 peer들이 super peer이거나 sub peer가 되기 전까지 과정1을 반복한다. 반복하는 과정에서 한번 super peer나 sub peer가 된 peer는 다음 계층화 작업 과정에서 고려하지 않는다.

과정 2)다른 super peer를 가진 sub peer들끼리 만나는 경우: 서로 다른 super peer를 가진 sub peer들이 만나는 경우, 이 sub peer들은 super peer들 사이의 다리역할을 한다. 그러므로 sub peer들은 위와 같은 경우가 생기면 자신의 super peer에게 알려준다. 즉, 자신을 통하면 다른 super peer를 통해서 갈 수 있다는 정보를 알려주는 것이다. Super peer의 entry에는 sub peer를 관리 하는 공간 이외에 다른 super peer를 관리하는 공간이 필요하다.

과정 3)하나의 sub peer에 2개 이상의 super peer가 충돌할 경우: 위의 과정1을 하다보면 하나의 sub peer에 2개 이상의 super peer가 연결 될 경우가 생긴다. 이 경우, 해당 sub peer는 나중에 발견 된 super peer의 sub peer로 속하게 된다. 하지만 이전의 super peer의 정보(id, 주소, 파일목록)를 저장하면 파일 찾기를 하는 과정에서 2개 이상의 super peer를 사용할 수 있기 때문에 보다 효과적으로 파일을 찾을 수 있다.

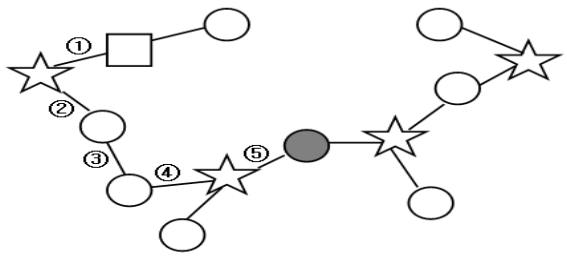
과정 4)이웃 sub peer수가 1개인 경우: 과정 1, 과정 2, 과정 3이 끝난 후, sub peer가 한 개만 있는 super peer가 있을 수 있다. 즉, 일정 시간동안 sub peer의 수가 하나인

경우, 해당 super peer는 자신의 sub peer와 연결된 sub peer의 수가 두 개 이상인 다른 super peer의 sub peer로 속하게 만들어준다. 그 이유는 super peer의 수를 줄이기 위해서이다.



(그림 2) MIS를 이용한 peer들의 계층화

(그림 2)는 위에서 언급한 4단계의 MIS 과정을 거친 후의 peer들의 모습이다. 별 모양의 peer가 super peer이고, 동그라미 모양의 peer가 sub peer이다. 각각 peer들 위에 적힌 숫자는 Luby의 알고리즘에 의해 생성된 임의의 수이다. (그림 2)에서 보는 것과 같이 별모양의 peer들이 주변의 동그라미 peer보다 임의의 수가 크다. 임의의 수 974를 가진 sub peer와 임의의 수 100을 가진 sub peer가 서로 만나는 경우, 자신의 super peer가 누군지 살펴보게 된다. 위의 그림에서 두 sub peer의 super peer는 서로 다르기 때문에 이들은 자신의 super peer에게 새로운 super peer의 존재를 알려주고 자신을 통하면 갈 수 있다는 정보를 알려준다. 검은색으로 칠해진 동그라미는 두 개의 super peer를 가진 sub peer이다. 예를 들어 처음에는 임의의 수 120을 가진 super peer(super peer(120))와 연결되어 sub peer가 되었지만 시간이 흘러 임의의 수189를 가진 super peer(super peer(189))를 만나게 되면서 super peer(189)의 sub peer로 들어가게 된다. 이 경우, 과정 3에 의해 super peer(120)와 연결을 끊는 것이 아니라 cache에 super peer(120) 정보를 저장하여 파일을 찾을 때 유용하게 쓸 수 있도록 한다. 네모 모양을 가진 peer는 원래는 super peer이지만 자신이 가진 sub peer가 한 개 이기 때문에 과정 4에 의해 자신의 임의의 수 152를 가진 sub peer의 또 다른 super peer인 임의의 수 1000을 가진 super peer의 sub peer로 등록이 된다.



(그림 3) 파일을 찾는 과정

(그림 3)은 파일을 찾는 과정을 보여준다. 별 모양의 peer는 super peer이고, 동그라미 모양의 peer는 sub peer이다. 네모 모양의 peer가 '파일 1'을 찾고 싶어하는 sub peer이고, 색칠된 동그라미 모양의 peer가 '파일 1'을 가진 peer라고 가정하자. 네모 모양의 peer는 sub peer이기 때문에 자신의 super peer에게 '파일 1'을 찾아 달라고 요청을 한다.(과정 ①) Super peer는 자신의 entry를 살펴보지만 '파일 1'을 찾을 수 없어, 다른 super peer에게 찾아 달라고 query를 보낸다.(과정 ②,③,④) 앞에서 설명한 것처럼 super peer들끼리의 경로는 sub peer들에 의해서 자동적으로 만들어진다. 주위의 super peer들이 search query를 받으면 자신의 entry를 찾아보게 되고, '파일 1'이 있는 것을 발견한다.(과정 ⑤) 그리고 '파일 1'을 가진 super peer는 *Response* 메시지를 보내준다.

● **Peer들의 삽입과 삭제에 의한 super peer의 재구성**

모바일환경에서는 사용자의 이동이 자유롭기 때문에 자신에게 이웃한 peer들이 수시로 바뀌게 된다. 때문에 super peer와 sub peer의 재구성이 필요하다. 모든 peer들은 일정한 주기마다 같은 시각에 자신의 주변에 있는 peer들에게 자신의 임의의 수를 전송하여 peer들의 계층화 작업을 갱신해야 한다. 이 과정은 시간의 동기화가 필요한데 이는 GPS를 활용하면 가능하다.

4. 실험

본 논문은 기존의 ORION 방식과는 달리 peer들을 두 개의 계층으로 나누었다. 그 결과 peer 간의 구성과 broadcast하는 방식이 달라졌다. 우리는 그 차이를 실험을 통해서 알아보도록 하자. 실험의 환경은 [표 1]과 같다.

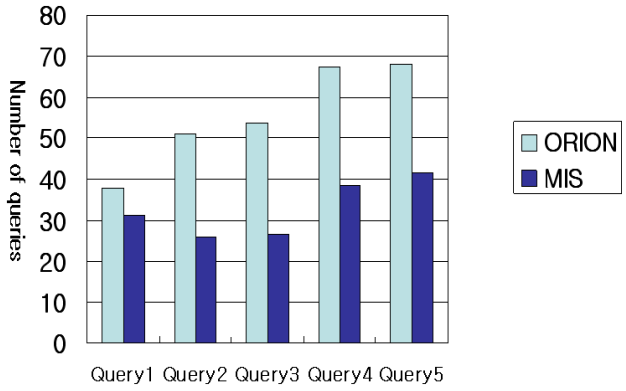
변수	값
Peer의 개수	40
통신범위(m)	115
지역의 크기(m^2)	1000*1000

<표 1> 실험 환경

Text file에는 노드의 *id*, 주소, 위치, 파일 목록이 기록되어 있어, text file을 한 줄씩 읽어 한 peer에 저장하고 $1000*1000m^2$ 지역에 추가했다. 이러한 text file을 20개를 만들었다. 또한 실험에는 query file을 사용하였는데 한 파일에 256개의 임의의 query가 있다. 이러한 query file이 총 5개가 있다. 위와 같은 조건으로 본 논문에서 제시한 MIS방식과 ORION을 비교해보았다.

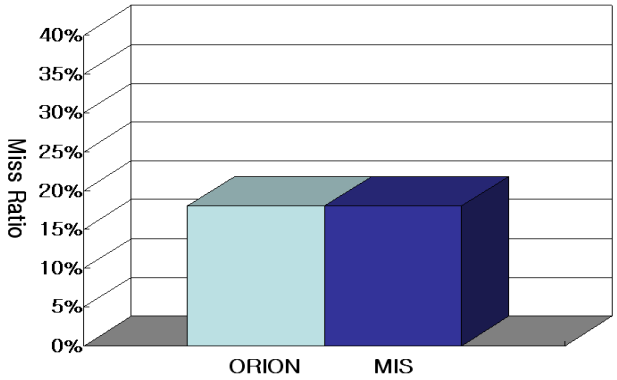
(그림 4)는 peer가 필요한 파일을 찾기 위해 사용한 query의 평균이다. ORION의 경우, 평균 55.66개의 query를 사용하였지만, MIS의 경우 평균 32.72개의 query를 사용하였다. MIS방식은 기존의 multi-broadcasting 방식을 피하였기 때문에 좋은 성능을 보여주었다.

다음으로 살펴 볼 실험은 파일을 찾는데 실패한 확률이



(그림 4) 필요한 파일을 찾기 위해 사용한 query의 수

다. 모바일의 가장 큰 특성은 이동성이다. 이러한 특성과 각 peer들간의 통신범위가 제한적이기 때문에 기존에 구축한 super peer, sub peer의 모양이 수시로 바뀌게 된다. 이 때문에 기존의 연결된 peer들의 연결이 끊어져 파일 찾기에 실패하는 경우가 발생 할 수 있다. 본 논문은 이를 염두에 두어 실패한 정도를 측정하였다.



(그림 5) 필요한 파일을 찾는데 실패한 정도

(그림 5)는 파일을 찾는데 실패한 확률이다. 두 방법 모두 18%정도 실패하였다. 두 방법 모두 자신의 통신 범위 안에 들어온 peer들끼리 연결하기 때문에 실패할 확률이 같게 나왔다.

5.결론

본 논문은 MANET을 구성하는 peer들을 두 개의 계층으로 나눠 기존의 routing table을 바꾸고 flooding 방식을 피하려고 하였다. 그 결과, 파일 탐색에 요구되는 query의 수는 이전 방식(ORION)에 비해 적게 소요되었다. 하지만 본 논문은 peer들을 계층화 하는데 소요된 시간과 overhead에 대해서 고려하지 않았다. 또한 super peer로 선택된 peer는 주위의 sub peer를 관리해야 하기 때문에 상당한 overhead가 있을 거라 예상한다.

앞으로 우리는 peer를 계층화 할 때 소요되는 시간과 overhead의 감소에 초점을 둘 것이다. 또한 super peer의 수를 잘 조절하여 super peer로 선정된 peer가 너무 많은 일을 하지 않게 하는 방법에 대해서 연구할 계획이다.

6. 참고문헌

[1]. The Napster home page, http://www.napster.com/

[2]. The openNap home page, http://opennap-ng.sourceforge.net/

[3]. Gnutella, http://www.gnutelliums.com/

[4]. A. Klemm, C. Lindemann, and O. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad hoc Networks," *Proceedings on the Vehicular Technology Conference (VTC)* 2003, vol.4, pp.2758-2763, October 6-9, 2003.

[5]. C. Perkins, E. Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, http://www.ietf.org/internet-drafts/draft-ietf-manetaodv-11.txt, IETF Internet Draft (work in progress), June 2002.

[6]. Michael Luby. "A simple parallel algorithm for the maximal independent set problem," SIAM Journal of Computing, Vol.15, No.4, pp.1036-1053, November 1986.