모바일 에드 혹 네트워크 상의 신뢰성 있는 P2P 파일 공유를 위한 BU-Chord 메커니즘

준회원 정 홍 종*, 송 점 기**, 정회원 김 동 균***

BU-Chord Mechanism for Reliable P2P File Sharing over MANET

Hong-Jong Jeong*, Jeomki Song** *Associate Members*,
Dongkyun Kim*** *Regular Member*

요 약

MANET과 P2P응용 모두 고정된 인프라에 의존하지 않고, 각 노드들의 동작만으로 전체 네트워크를 유지한다. 이러한 공통적인 특징을 바탕으로, P2P 응용을 MANET상의 대표적인 응용의 사례로 볼 수 있다. 여러 P2P 구조들 중에서 중앙의 인덱스 서버를 사용하는 방식의 경우 신뢰성 있는 노드가 필요한데 MANET의 특성상 신뢰성 있는 노드를 찾을 수 없다. 따라서 본 논문에서는 완전 분산형의 신뢰성 높은 P2P 검색을 위해 DHT기반의 P2P 색인 프로토콜인 Chord를 사용한 P2P 파일 공유 응용을 설계하였다. Chord는 공유할 파일의 key들을 네트워크 전체에 분산하여 저장하고 이를 검색하는 방법을 제안하고 있다. 하지만 Chord 색인 프로토콜을 노드들의 가입과 탈퇴가 빈번히 발생하는 MANET환경에 적용할 경우 노드가 탈퇴할 때 그 노드가 저장하고 있던 key들이 손실되는 문제가 발생한다. 따라서 본 논문에서는 임의의 네트워크를 떠나는 노드가 네트워크가 떠나는 상황을 인식하고, 그 노드가 저장하고 있던 key들을 복구하여 지속적인 P2P 색인 서비스가 가능하도록 하는 BU-Chord 메커니즘을 제안한다.

Key Words : MANET, P2P, DHT, Chord

ABSTRACT

MANET and P2P applications have a common nature that they don't have any fixed infrastructures that might maintain network topologies. With such common characteristics, a P2P application can be a killer application over MANET. Due to absence of reliable node which serves indexing services in MANET, fully distributed P2P applications are more suitable for MANET. By using DHT like Chord, we can save network bandwidth and avoid a point of failure of a directory server. However, since MANET allows nodes to depart from network freely, P2P file sharing applications using Chord lookup protocol should address how to recover the keys stored at the departed node. In this paper, we propose BU-Chord in order to detect and recover the departure of nodes by creating and storing backup file information in distributed manner. Our BU-Chord shows off better performance than existing Chord especially in case of high departure rate of nodes.

* 경북대학교 컴퓨터공학과 이동인터넷망 연구실(hjjeong@monet.knu.ac.kr),
** 경북대학교 컴퓨터공학과 이동인터넷망 연구실(jksong@monet.knu.ac.kr),
*** 경북대학교 컴퓨터공학과 이동인터넷망 연구실(dongkyun@knu.ac.kr)
논문번호 : KICS2005-05-186, 접수일자 : 2005년 5월 3일, 최종논문접수일자 : 2006년 3월 20일

Ⅰ. 서 론

MANET[1]은 제한된 무선 전송 범위를 가진 이동 노드들 간의 상호 작용을 통해 멀티 홉 노드와 통신이 가능한 네트워크이다. 현재까지 MANET에서의 연구는 MAC 프로토콜과 라우팅 프로토콜에 관련된 연구 분야에 집중되어 왔다. 응용 분야 역시 전장에서 병사들 간의 통신 혹은 응급현장에서 구조대원들 간의 통신 같은 분야로 한정되었다. 하지만 최근 들어 기존의 연구들을 바탕으로 MANET의 특징을 잘 활용할 수 있는 응용에 대한 연구의 중요성이 부각되고 있다.

기존의 인터넷 상의 응용들은 크게 서비스 제공자와 수요자의 구분 여부에 따라 서버-클라언트 모델과 P2P(Peer-to-Peer)모델로 구분할 수 있다. 전자는 서비스를 제공하는 주체인 서버와 서비스를 제공받는 클라이언트의 역할이 구분되어 동작하는 형태를 말한다. 후자는 네트워크에 참여한 노드들이 서비스를 제공하는 서버인 동시에 서비스를 제공받는 클라이언트의 역할을 동시에 수행한다. 특히 이러한 P2P 네트워크에서 서버와 클라이언트의 역할을 수행하는 노드를 피어라 부른다. MANET의 경우 아직까지 응용의 형태들에 대한 분류가 마련되지는 않았으나, 현재 MANET의 대표적인 응용 사례를 연구하기 위한 노력들이 활발히 진행되고 있다.

기존의 유선상에서의 응용들을 MANET에 적용할 때 MANET의 여러 가지 제약 사항들로 인한 문제점들이 발생한다. 잘 알려진 MANET의 제약사항들로는 동적인 네트워크 특성, 배터리의 제약, 낮은 무선 전송의 대역폭 등이 있다. 노드들이 네트워크 참여와 탈퇴가 빈번한 동적인 네트워크 특성과 배터리 제약으로 인해 고정된 신뢰성 있는 노드로부터의 서비스를 기대 하기가 어렵고, 낮은 무선 전송의 대역폭과 에러가 잦은 네트워크의 특성으로 신뢰성 있는 전송을 보장하기 힘들다. 이러한 MANET의 여러 제약사항들을 극복할 수 있는 응용을 개발하는 것이 중요한 이슈가 되고 있다.

MANET과 P2P 모델의 응용에서 많은 공통점을 발견할 수 있다. P2P 응용은 모든 피어들이 자유롭게 네트워크 참여와 탈퇴가 가능하다. 이는 모든 피어들이 서버의 기능을 분산하여 동작하고 있어, 어느 피어가 네트워크를 탈퇴한 후에도 다른 피어가 탈퇴한 피어의 역할을 대신해 줄 수 있기 때문이다. 어느 임의의 노드들이 네트워크를 떠난 후에도 다른 피어들이 계속해서 서비스를 제공할 수 있기 때문에 네트워크 전체의 기능이 유지될 수 있다. 모든 피어들이 기능을 분산하여 네트워크가 유지되는 장점을 바탕으로 P2P 모델이 MANET의 대표적인 형태의 응용이 될 수 있다.

낮은 네트워크 대역폭을 가진 MANET의 특징을 고려하여 DHT(Distributed Hashing Table)을 사용하는 P2P 모델들을 적용함으로써 리소스 검색과정에 필요한 메시지 수를 줄여 네트워크 대역폭을 절약 할 수 있다. P2P 응용 중 DHT를 사용하는 대표적인 예로 Chord[2]가 있다. Chord는 네트워크상에서 리소스 검색을 위한 색인 기능을 네트워크 전체가 분산하여 수행하도록 정의하고, 이러한 분산된 색인 기능을 통해 리소스를 빠르고 정확하게 검색하는 방법을 정의 하고 있다. 대표적인 P2P응용의 형태인 파일 공유 응용을 Chord를 사용하여 설계할 경우 공유할 파일의 정보들을 네트워크 전체에 분산하여 저장하게 된다. 하지만 MANET에 이러한 응용을 적용할 경우 잦은 노드들의 네트워크 탈퇴로 인해 분산하여 저장하고 있는 파일들의 정보가 손실되는 일이 발생할 수 있다. 따라서 잦은 공유파일 정보가 손실되는 상황을 대처하기 위한 과정들이 추가 되어야 한다.

본 논문에서는 MANET 환경에서 잦은 노드들의 탈퇴로 인해 손실된 파일 정보들을 복구하기 위한 BU-Chord 프로토콜을 제안한다. 그리고 BU-Chord를 적용하여 MANET상에서 파일을 공유하고 검색할 수 있는 P2P응용인 DiPP(fully Distributed Peer-to-Peer)를 구현하였다. 본 논문의 구성은 다음과 같다. 2장에서 MANET상에서 개발된 P2P응용들과 Chord 프로토콜에 대한 간단히 소개한다. 3장에서는 BU-Chord를 제안하고, 4장에서는 BU-Chord를 사용한 응용인 DiPP을 소개한다. 5장에서는 성능평가와 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 관련 연구

초기의 MANET 응용들은 군사적인 목적으로 개발되고 활용되었다. 근래에 들어 개인 휴대용 무선 기기들, 무선랜과 같은 무선 네트워크 장비들의 발달로 실생활에서의 MANET 응용에 대한 연구들이 활발히 진행되고 있다. MANET의 대표적인 응용 모델로써 P2P 모델의 응용이 있다. P2P 응용의 동작 형태는 MANET과 유사하게 어떠한 인프라에 의존하지 않고, 네트워크를 구성하는 각 노드들의 동작만으로 이루어진다. MANET은 노드들의 잦은

이동으로 인해 네트워크가 동적으로 변하는 특성이 있다. 이러한 특성을 고려하여 P2P 네트워크 구성이 용이하고 안정적으로 지속할 수 있도록 하는 연구가 계속 되고 있다.

2.1 MANET 상의 P2P 응용

ORION(Optimized Routing Independent Overlay Network)[3]은 MANET에서의 빠르고 효율적인 검색을 위해 MANET 라우팅과 P2P 검색이 결합된 형태의 P2P 검색 알고리즘이다. ORION은 MANET 상에서, 효과적인 파일 검색을 위하여 응용 계층의 질의 메시지와 네트워크 계층의 질의 메시지를 통합하여 두 개의 라우팅 테이블인 경로 라우팅 테이블과 파일 라우팅 테이블을 관리한다. 라우팅 테이블은 AODV(Ad-hoc On-demand Distance Vector)[4]의 라우팅 테이블과 유사한 구조이며, 파일 라우팅 테이블은 원하는 파일에 대한 경로 정보를 유지한다. 이 방식은 파일을 검색하기 위한 P2P 검색을 MANET 라우팅과 동시에 수행함으로써 네트워크 트래픽의 부하를 줄이고 빠르게 검색할 수 있는 장점이 있다. 아래 그림 1은 ORION의 동작과정을 나타낸 것이다. 노드 A의 파일 라우팅에 1~4 파일을 획득할 수 있는 파일 라우팅 테이블 정보를 유지하고 있다.

MPP(Mobile Peer-to-Peer protocol)[5]는 MANET 상에서 P2P 네트워크 구축을 위해 정의된 프로토콜이다. 응용 계층에서 P2P 오버레이(Overlay) 네트워크를 구성하는 응용을 위해 하부 계층의 프로토콜들을 새롭게 정의 하였다. 아래 그림 2와 같이 MPP 프로토콜들은 응용계층에서 동작하는 MPP, 네트워크 계층에서 동작하는 EDSR(Enhanced Dynamic Source Routing), 그리고 이 두 프로토콜간

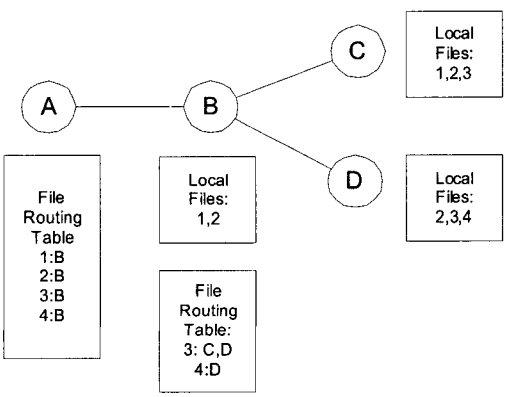


그림 1. ORION의 검색과정

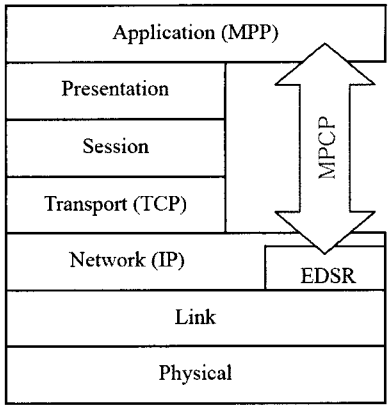


그림 2. MPP 프로토콜 구조

의 통신을 위한 MPCP(Mobile Peer Control Protocol)로 구성된다. MPP는 Ad-hoc 네트워크상에서 노드의 이동성으로 인해 발생하는 링크 실패에 대한 빠른 제어를 통해 안정적인 오버레이 네트워크를 유지한다. 가능하다. 하지만 이러한 프로토콜들은 특정 어플리케이션을 위해서 설계되어 범용으로 사용되기에는 한계가 있다.

MANET에서 P2P 응용을 개발하기 위한 플랫폼으로 PROEM[6]이 제안 되었다. PROEM은 크게 PROEM Runtime System과 Peerlet Development Kit으로 두 개의 컴포넌트로 구성된다. PROEM은 각 계층별로 새로운 프로토콜을 정의한 MPP와 다르게, 오버레이 네트워크 상에서 각 피어들의 기능에 따라 프로토콜을 정의한다. Presence, Data, Community 프로토콜들이 오버레이 네트워크를 구성하는 P2P 응용의 기능에 따라 설계되었으며 응용 개발에 필요한 API와 Runtime Engine을 제공한다. 그림 3은 전체적인 PROEM의 구조를 나타낸 그림으로 Protocol Stack, Peerlet Engine, Service의 세 부분으로 구성된다. 예를 들어 PDK (Peerlet Development Kit)는 Naming, Communication, Data management, 이벤트 처리를 위한 자바 인터페이스와 클래스들의 집합으로 자바 런타임 환경을 요구한다.

MANET 어플리케이션 개발을 위한 연구과정에서 Microsoft .NET Compact Framework와 DSR (Dynamic Source Routing) 프로토콜을 이용하여 MANET상에서 인스턴트 메신저를 설계하고 구현했다[7]. 이 프로젝트에서는 DSR 프로토콜의 Route Discovery(Route Request, Route Reply)기능만을 구현했으며 Route Maintenance는 향후 개발로 남겼

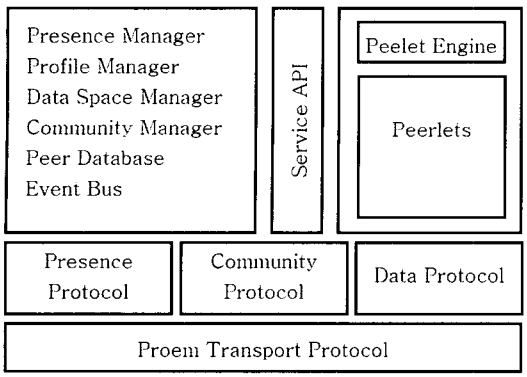


그림 3. PROEM 프로토콜 구조

고 데이터를 기술하는데 XML을 사용했다. 프로젝트에서는 802.11을 기반으로 하는 MANET상에서 DSR 프로토콜이 적당한 라우팅 프로토콜이라고 말하고 있다. "Pocket Chat" 어플리케이션은 브로드캐스팅을 위해 UDP를 사용하며 Microsoft .NET Compact Framework을 개발 플랫폼으로 Microsoft Visual Studio를 이용하여 C# 언어로 개발하였다.

2.2 Chord

Chord[2]는 데이터를 빠르고 정확하게 검색하기 위해 각 노드들끼리 데이터의 key를 분산저장하고, 분산 저장된 key들을 검색하기 위한 P2P 색인(lookup) 프로토콜이다. 냅스터[8]와 같이 검색할 데이터의 key들이 중앙의 서버에 저장되어 검색을 수행하는 형태와는 달리 Chord는 데이터의 key들을 네트워크에 참여한 노드들에 골고루 분산하여 저장하는 구조를 취하고 있다. 이렇게 n개의 노드들 사이에 분산 저장된 key들을 검색하기 위해 각 노드들은 다른 O(log n)개의 노드들에 대한 정보만 저장하면 되고, 또한 key의 검색 시에도 O(log n)개의 메시지만 사용하여 원하는 key를 검색할 수 있다. 따라서 네트워크의 규모가 커지더라도 유효한 검색을 위해 유지해야 할 다른 노드의 정보와 검색에 수행되는 시간은 큰 영향을 받지 않는다.

Chord는 각 노드와 key들에게 SHA-1[9] 해쉬 함수를 사용하여 m 비트의 식별자를 할당한다. 노드의 식별자는 노드의 IP주소를 해쉬한 결과이고, key의 식별자는 key를 해쉬한 결과이다. 따라서 Chord는 0에서 2^{m-1}까지의 식별자 범위를 가지게 된다. key들을 네트워크 전체의 노드들에게 골고루 분산하여 저장하기 위해 consistent hashing을 사용한다. Key k는 Chord 식별자 범위 상에서 k의 식별자보다 같거나 바로 큰 식별자를 가진 노드에게 저장된다. 이 노드를key k의 successor라 부르고, successor (k)로 표현한다. Chord에서 색인과정을 수행하기 위해서 각 Chord 노드들은 각각의 successor, predecessor, 그리고 finger table의 정보를 유지해야 한다. 노드 N의 successor와 predecessor는 각각 Chord 식별자 범위 상에서 노드 n의 앞과 뒤에 위치한 노드이다. 즉, n 바로 다음의 식별자를 가진 노드가 노드 n의 successor이고, n 바로 직전의 식별자를 가진 노드가 노드 n의 predecessor이다. Finger table은 key의 색인 과정을 좀 더 빠르게 수행하기 위한 자료구조로써 m개의 엔트리로 구성되었다. Finger table의 i 번째 엔트리는 successor($n+2^{i-1}$)를 가리킨다. ($1 \le i \le m$) 아래 그림 4는 6 비트 크기의 식별자를 사용하는 Chord ring을 나타낸 그림이다. 이 네트워크는 10개의 노드에 4개의 key가 할당되어 있다. K24, K30은 그의 successor인 노드 N32에 저장되고, K38과 K54또한 각각 그의 successor인 노드 N38과 N56에 저장되어 있다.

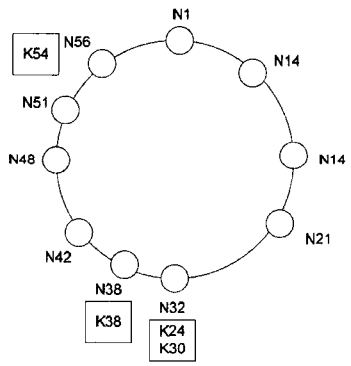


그림 4. Chord Ring

Ⅲ. BU-Chord 메커니즘

본 논문에서 제안하는 BU-Chord(Back-Up Chord) 메커니즘은 Chord 프로토콜을 사용한 검색에서 노드가 네트워크를 갑작스럽게 떠날 때 발생하는 Key 손실을 복구하는 방법을 제안한다. P2P응용에서 Chord 색인 프로토콜을 사용하여 데이터를 저장 할 경우 데이터의 key들을 각 노드들이 나누어 저장하도록 되어 있다. 하지만 MANET의 특성상 네트워크에 참여한 노드들이 네트워크를 자유롭게 드나들 수 있다. 이로 인해 떠나는 노드에 저장된 key들이 손실되는 문제에 대한 해결이 필요하다. 이 문제를 해결하기 위해 네트워크를 떠나는 노드가 저장하고 있는 key들을 다른 이웃 노드에게 복사해 둠으로써

이 노드가 네트워크를 떠나더라도 저장되었던 key들의 손실을 막을 수 있다. BU- Chord 메커니즘에서는 key의 successor에게 저장된 key들을 다른 노드에 복사하는 방법과 successor가 네트워크를 떠난 후 이 key들이 계속해서 검색 될 수 있도록 복구하는 방법을 정의한다.

3.1 개발배경

기존의 유선상의 P2P응용에서의 검색 방법을 크게 두 가지로 구분하면 데이터를 검색하기 위한 색인을 중앙의 서버에 의존하는 하이브리드 P2P (Hybrid P2P)와 이러한 서버 역할 없이 오직 노드들의 동작만으로 검색을 수행하는 순수 P2P(Pure P2P) 형태로 나눌 수 있다. 전자의 경우 대표적인 예로써 넵스터를 들 수 있다. 이러한 모델의 검색방법을 MANET에 적용하고자 했을 경우 어느 한 MANET 노드가 P2P검색을 위한 색인 서버의 기능을 수행해야 한다. 하지만 MANET의 구조적 특성상 어느 한 노드에게 고정적인 서버의 역할을 기대하기 힘들다. 또한 색인 서버에게 트래픽이 집중되는 일이 발생하는 문제 또한 발생할 수 있다. 후자의 순수 P2P 형태의 경우 그 대표적인 예가 그누텔라[10]이다. 그누텔라는 P2P검색을 위해 네트워크 전체에 검색 메시지를 브로드캐스트한다. 따라서 앞서 언급한 넵스터 방식에서의 문제점은 해결이 되지만 브로드캐스트를 수행하는 과정에서 발생하는 전송 대역폭이 낭비되는 문제점이 발생한다. 특히나 데이터 전송시 전송 대역폭이 작은 무선 매체를 사용하는 MANET에서는 전송 대역폭의 낭비문제 또한 해결되어야 한다.

DHT(Distributed Hash Table)기반의 P2P 검색 알고리즘을 사용하여 위의 문제들을 해결할 수 있다. DHT 알고리즘의 경우 데이터의 색인을 네트워크에 참여한 노드들이 분산하여 저장하기 때문에 네트워크의 병목현상 문제나 검색 메시지 브로드캐스트의 문제가 발생하지 않는다. 이러한 DHT의 장점을 활용한 대표적인 검색 방법 중 하나가 Chord이다. Chord를 기반으로 한 P2P 파일공유 응용에서 각 노드들은 파일을 공유하기 위해 Chord lookup 프로토콜을 사용하여 공유할 파일의 key k를 successor(k)에 등록해 두어야 한다. 공유한 파일을 검색하기 위해서는 Chord lookup 프로토콜을 사용하여 파일을 등록할 때와 마찬가지로 key k의 successor인 successor(k)를 검색함으로써 파일을 검색 할 수 있다. 하지만 MANET의 동적인 네트워크 특성상 노드들이 어떠한 예고도 없이 네트워크를 빈번하게 떠나게 된다. 따라서 어떠한 노드가 네트워크를 떠한 후, 그 노드에 등록된 key의 경우 검색 될 수 없다. 위의 문제를 해결하기 위해 손실된 key들을 복구해 주는 방법이 필요하다. 본 논문에서는 successor에 등록된 key들을 다른 이웃 노드에게 복사해 두고, successor가 네트워크를 떠난 후 successor에 등록되었던 key들을 복구하는 방법인 BU-Chord 메커니즘을 제안한다.

3.2 모델

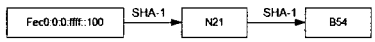
BU-Chord의 동작을 위해 기존 Chord에서의 successor/predecessor 이외에 backup successor/backup predecessor의 기능을 추가적으로 사용한다. Backup successor(BS)는 backup predecessor(BP)에 key들을 복사해 두고, BP의 존재 유무를 지속적으로 확인한다. 그리고 만약 BP가 네트워크를 탈퇴할 경우, BP에 저장되어 있던 key들의 새로운 successor를 찾아 복구한다.

BU-Chord의 동작에서도 기존의 Chord와 마찬가지로 각 노드와 key들에게 m 비트 크기의 식별자를 할당한다. 노드의 식별자는 각 노드의 IP 주소를 해쉬함수의 입력 값으로 연산한 결과 값이고, key의 식별자는 key를 해쉬한 결과 값이다. 노드가 할당받은 key들의 복구를 위해 BS를 설정하게 되는데 이를 위해 각 노드마다 m 비트의 backup 식별자를 추가적으로 할당한다. 추가적으로 생성된 m 비트 backup 식별자 B_i는 노드의 식별자를 해쉬하여 생성한다. BS의 선정은 Chord 노드들 중에서 B_i의 successor를 노드의 BS, 즉 successor(B_i)로 선정한다. BS를 선정하는데 있어 consistent hashing을 사용함으로 인해 각 노드들의 BS가 네트워크 전체에 골고루 퍼지게 되어 복구를 수행하는 과정의 부하를 분산시키는 효과가 있다. 또한 복구의 수행이 네트워크 전체에 영향을 미치지 않고 일부 몇몇 노드의 동작으로 수행되는 장점이 있다.

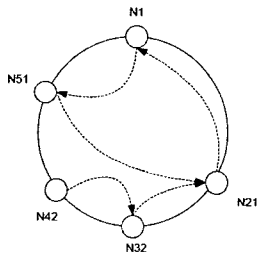
3.3 초기화

BU-Chord의 동작을 위해 각 노드들은 자신이 저장하고 있던 key들을 복사해 둘 backup successor (BS)를 선정하여 key들을 복사해둔다. 노드 m이 저장하고 있는 key들을 노드 m의 BS에게 저장하기 위해서 우선노드 m의 식별자를 해쉬하여 m의 backup 식별자 B_m을 얻어낸다. BS는 backup 식별자 B_m의 successor인 Successor(B_m)를 선정한다. 노드 m은 자신의 BS에게 자신이 backup predecessor

(BP)임을 알리고, 자신의 predecessor와 successor 정보를 BS에 저장한다. 그리고 노드 m이 저장하고 있던 key들을 모두 m의 BS에게 복사한다. 이렇게 BS에 저장된 정보들은 BP인 노드 m이 네트워크를 떠난 후, key들을 새로운 successor를 찾아 복구하는데 사용된다. BS 등록과정과 key들의 복사 과정이 끝나면, BP와 BS는 주기적으로 BEACON/BEACON-ACK를 주고받으며 서로가 네트워크상에서 잘 동작하고 있는지를 확인한다. 아래 그림 5는 5개의 노드가 BS를 선정한 예를 나타내고 있다. 각 노드들은 8 비트 크기의 식별자를 사용한다. 그림 5(a)와 같이 노드의 IP 주소가 fec0:0:0:ffff::100인 노드가 자신의 IP 주소를 해쉬한 결과인 N21을 노드 식별자로 선정한다. 그리고 N21을 해쉬하여 backup 식별자 B54를 생성한 후, B54의 successor인 N32를 BS로 선정하게 된다.



(a) Backup 식별자 선정 예



노드 식별자	Backup 식별자	Backup successor
N1	B46	N51
N21	B54	N1
N32	B17	N21
N42	B54	B32
N51	B9	B21

(b) BU-Chord Ring

그림 5. BU-Chord의 백업 노드 선정

BS를 선정하기 위해 Chord에서의 key를 할당하는 과정에서와 같이 consistent hashing을 사용함으로써, 각 노드들의 key 들을 네트워크에 분산하여 저장하게 된다. 따라서 네트워크의 규모가 커지더라도 각 노드들이 유지해야 하는 정보의 량은 일정하게 유지되므로, 네트워크의 확장성이 높아지게 된다. 그리고 노드들의 백업과 복구 과정이 네트워크 전체에 영향을 미치지 않고 일부 노드들끼리만의 동작을 통해 지역적으로 수행되는 분산 구조를 유지하게 되어 네트워크의 견고함을 높이게 된다.

3.4 가입

Chord 네트워크에 새로운 노드가 참여하게 되더라도 지속적으로 색인 서비스가 지속하기 위해, 새롭게 참여한 노드를 고려해 기존 노드들이 저장하고 있는 key들을 재배치해야 한다. 새롭게 가입한 노드 n이 노드 n의 successor m에게 네트워크에 참여했음을 알리게 된다. 노드 n의 successor는 Chord ring에서 n의 식별자 바로 다음의 노드 식별자를 가진 노드이다. 노드 m은 노드 n이 새롭게 네트워크에 참여함을 인지 한 후, n의 식별자보다 작은 key들을 저장할 새로운 successor n에게 전달함으로써 기존의 Chord 색인이 정상적으로 동작하도록 한다. Chord 네트워크에 참여하여 key들의 재배치와 노드의 successor와 predecessor 정보들이 반영된 후, 노드 n이 저장한 key정보들을 저장해 둘 BS를 찾아 백업 과정을 수행한다. 노드 n의 BS B_n에게 n이 할당 받은 key들과 n의 successor, predecessor 정보들을 B_n에게 복사해 둔다. 복사 과정이 끝난 후 n과 B_n은 BEACON/BEACON-ACK를 주고받으면서 서로의 네트워크 탈퇴를 주기적으로 확인하게 된다.

아래 그림 6은 Chord 네트워크에서 노드의 식별자가 N27인 노드가 네트워크에 참여하는 예제이다. 노드 N27이 기존의 네트워크에 참여하고자 하면, 노드 N27은 식별자의 successor인 N32에게 가입을 요청하게 되고, 요청 받은 노드 N32는 자신이 관리하던 key들 중 (K24, K30) 사이에 존재하는 key들을 N27에게 옮긴다. 그리고 N27은 자신의 노드 식별자를 사용하여 backup 식별자 B41를 생성하고, BS N42에게 자신이 저장하고 있던 key인 K24를 저장한다.

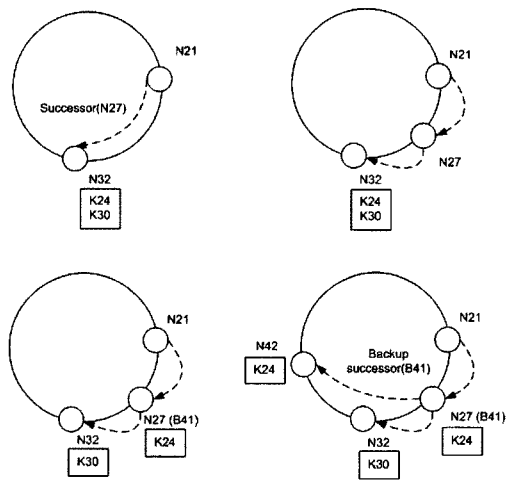


그림 6. 신규노드 가입 과정

3.5 손실 복구

BU-Chord의 복구는 backup predecessor(BP)와 backup successor(BS)간에 주기적으로 주고받는 BEACON/BEACON-ACK 메시지가 일정시간 동안

오지 않을 때 수행하는 과정이다. BU-Chord의 각 노드들은 자신의 BS나 혹은 BP로부터 일정시간 이상 동안 BEACON/BEACON-ACK의 교환이 없으면 그 노드가 네트워크를 떠난 상황으로 간주한다. 만약 어느 노드가 자신의 BS의 손실을 감지하였다면 새로운 BS를 찾아 앞서 설명한 BS 초기화 과정을 수행하고, BP의 손실을 감지하게 되면 자신이 복사 해 둔 BP의 정보들을 바탕으로 하여 key들을 새로운 successor에게 새롭게 할당하는 것으로 복구 과정을 수행한다.

BS가 탈퇴한 경우의 복구는 새로운 BS를 찾아 초기화를 다시 해주는 것만으로 복구를 마칠 수 있다. BS가 할당 받아 Chord 색인을 위해 관리 하던 key들은 BS의 BS가 복구 과정을 수행하게 된다. BP가 네트워크에서 탈퇴퇴한 경우 BP의 successor와 predecessor에게 BP의 손실을 알린다. successor와 predecessor는 각각 서로를 새로운 predecessor와 successor로 인식하게 되어 끊어진 Chord ring을 복구하게 된다. 그리고 BP가 저장하고 있던 손실된 key들을 복구하기 위해, BS는 key들의 새로운 successor인 탈퇴 노드의 successor에 복사한 후 BS의 역할을 종료하게 된다.

아래의 그림 7은 BU-Chord에서 BP가 네트워크를 탈퇴했을 경우 BS의 동작으로 key들이 복구되는 예제이다. Backup 식별자가 B41인 노드 N27이 네트워크를 탈퇴한 경우를 가정하면 그의 BS N42는 N27의 탈퇴를 인식하게 된다. 그 후, N42는 노드 N27의 successor N32에게 predecessor가 N21로 변경되었음을 알리고, predecessor N21에게 successor가 N32로 변경되었음을 알린다. Chord ring이 복구

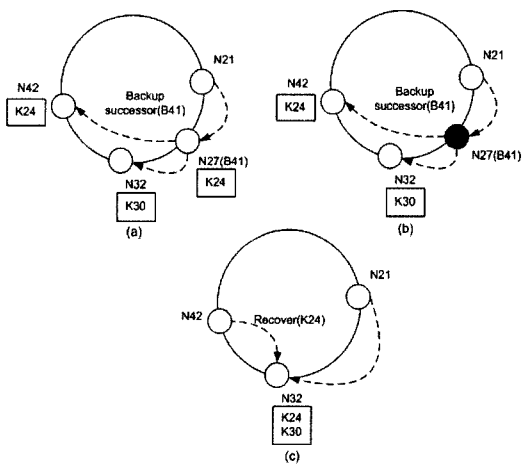


그림 7. 탈퇴노드 복구 과정

된 후 N27이 저장하고 있던 키인 K24를 새로운 successor인 N32에게 저장함으로써 손실된 key의 복구를 마치게 된다.

이처럼 BS와 BP의 개념을 사용하여 손실된 key의 복구를 수행함으로써 노드가 네트워크를 탈퇴한 상황을 즉각 인지할 수 있고, 노드의 탈퇴로 인한 손실된 key의 복구도 빠르게 수행 할 수 있다. 그리고 노드의 탈퇴를 인지하고 복구하는 형태 또한 완전 분산형의 P2P 구조로 수행함으로써 일부 노드의 기능에 이상이 생기더라도 전체 네트워크가 동작하는 데는 크게 영향을 미치지 않는다. 또한 복구를 위한 동작들을 전체 노드들이 골고루 분산하여 수행함으로써 복구과정의 부하가 분산되는 효과를 가진다.

3.6 다중 노드 손실 복구

앞서 설명한 BU-Chord의 복구는 네트워크에서 탈퇴한 노드의 backup successor(BS)와 successor의 동작만으로 수행된다. 네트워크의 변화가 극심한 토폴로지 상에서는 복구를 수행하는 노드들이 복구가 완료되기 전에 동시에 사라지게 되는 상황이 발생할 수 있다. 어느 임의의 네트워크를 탈퇴하는 노드와 그의 BS, successor 중 어느 노드가 동시적으로 탈퇴하는 상황을 다중 노드 손실 상황으로 정의하고, 이를 인지하고 복구하기 위한 방법들을 기술한다. 다중 노드 손실 상황은 다음의 세 가지 경우로 정리할 수 있다; 첫째로 탈퇴 노드와 그의 successor의 동시적인 탈퇴, 둘째로 탈퇴 노드와 그의 BS 의 동시적인 탈퇴, 셋째로 탈퇴 노드와 그의 successor, BS가 모두 탈퇴하는 경우로 정리 할 수 있다. 이러한 다중 노드 상황을 해결하기 위해서는 BS에 몇 가지 추가적인 동작과정이 수행되어야 한다.

첫 번째 경우로 탈퇴 노드(DN)와 그의 successor (SC)의 동시적인 탈퇴가 발생한 경우이다. 앞서 설명한 복구 방법에서는 DN의 BS가 DN의 key를 그의 SC에게 전달함으로써 복구를 수행한다. 하지만 DN와 SC가 동시에 떠나게 된다면 위의 복구방법으로 복구 할 수 없다. 따라서 이러한 상황을 복구하기 위해서 다음의 동작들을 추가적으로 적용하여 위의 문제를 해결할 수 있다. BS가 DN의 탈퇴를 인지하고, 복구를 수행하기 전에 SC와 BEACON/BEACON-ACK 교환하여 SC의 동작유무를 확인한다. 만약 SC와 BEACON/BEACON-ACK를 주고 받을 수 없는 상황이라면 다중 노드 손실 상황으로 고려하고 그에 대한 복구를 수행을 시작한다. DN의 BS는 SC의 노드 식별자를 해쉬하여 SC의 backup

식별자를 생성해낸다. 그리고 이를 사용하여 SC의 BS를 색인하여 SC의 탈퇴 사실을 통보하고 DN의 key를 전달한다. DN의 key를 전달받은 SC의 BS는 SC의 기존 단일 노드의 탈퇴 상황을 복구하는 방법을 사용하여 SC의 key와 DN의 key를 복구한다; 즉, SC와 DN의 key를 SC의 SC에게 전달한다.

그림 8은 DN과 SC가 동시에 네트워크를 떠난 다중 노드 손실 상황의 복구과정을 나타낸다. 노드 N25가 네트워크를 탈퇴한 후 N25의 key들이 복구되기 전에 그의 SC인 N36이 네트워크를 탈퇴하여 다중 노드 손실상황이 발생했다고 가정한다. N25가 네트워크를 탈퇴한 후, 그의 BS인 N57은 N25의 복구를 수행하기 위해 N25의 SC(즉, N36)의 동작 유무를 확인한다. 하지만 N36역시 네트워크를 탈퇴한 상황이므로 BEACON-ACK를 주고받을 수 없다. N56은 N25와 그의 SC인 N36이 동시에 네트워크를 탈퇴한 다중 손실 상황으로 인식하고 다음의 과정들을 수행한다. N36의 BS를 찾기 위해 N36의 backup 식별자(B47)를 생성한 후 backup 식별자의 SC를 색인 하는 과정을 수행하여 N36의 BS인 N49를 찾아낸다. N57은 SC의 BS N49에게 N36의 탈퇴를 알리고 N25의 key들(K22, K23)의 복구를 요청한다. 복구 요청을 받은 N49는 N36의 key들의 복구뿐만 아니라 N57로부터 요청 받은 N25의 key들을 N36의 SC인 N39에 전달함으로써 복구 과정도 동시에 수행한다.

두 번째 경우의 DN과 DN의 BS가 동시에 네트워크를 떠난 다중 노드 손실 상황이다. 이러한 상황은 BS의 BS에 추가적인 복구 과정을 추가함으로써 해결할 수 있다. BS의 BS가 BS의 탈퇴를 인식한 후, BS의 복구를 수행하기 위해 BS의 SC에게 BS

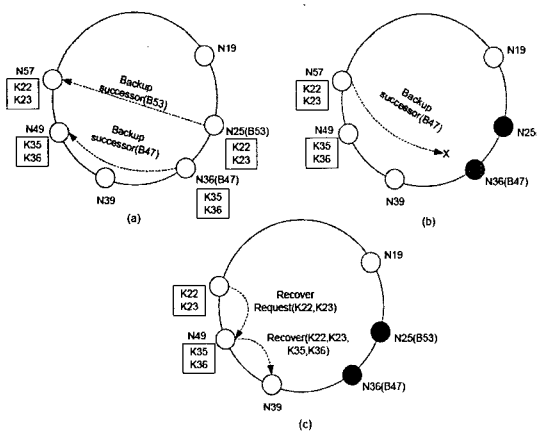


그림 8. 다중노드 손실 복구 과정

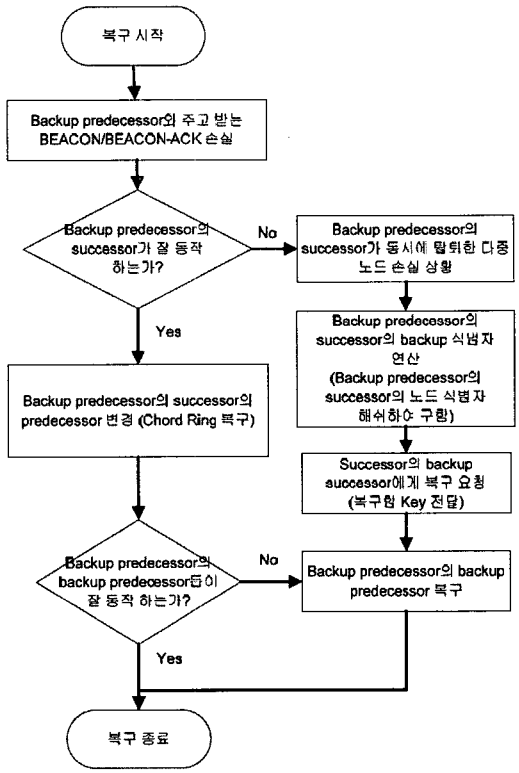


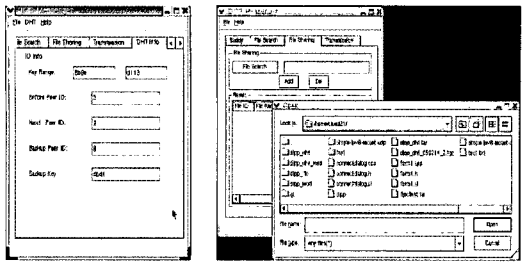
그림 9. 다중 노드 손실 복구를 위한 backup successor 동작 과정

의 key들을 전달할 것이다. 그 후, BS의 BS는 BS의 BP들의 동작 유무를 확인하는 과정을 수행함으로써 다중 노드 손실을 방지할 수 있다. 만약 다중 손실이 발생한 경우라면 BS의 BS가 저장하고 있던 BS의 key들 중 DN의 영역에 해당하는 key들을 DN의 SC에게 전달하는 과정을 수행하게 된다.

마지막의 경우, 즉 DN, BS, SC 세 노드의 동시적인 네트워크의 탈퇴가 발생한 경우는 앞서 설명한 BS과 SC을 복구하기 위한 BS와 SC의 BS가 수행하는 복구를 통해 해결될 수 있다.

Ⅳ. DiPP 구현

Chord의 색인 알고리즘과 BU-Chord의 key복구 알고리즘을 사용하여 모바일 에드 혹 네트워크 환경에서의 P2P응용인 DiPP(fully Distributed Peer-to-Peer)을 구현하였다. DiPP은 네트워크에 참여한 모든 노드들이 서비스를 제공하는 서버의 역할과 서비스를 제공받는 클라이언트의 역할을 동시에 수행하는 P2P구조로 설계되었다. DiPP을 설계 시 고려사항은 다음과 같다. 네트워크의 변화가 잦은 동



(a) (b)

그림 10. DiPP의 실행 화면

적인 네트워크 상황 고려, 각 노드간의 데이터 전송은 유선에 비해 전송 대역폭의 낮은 무선랜을 사용한다.

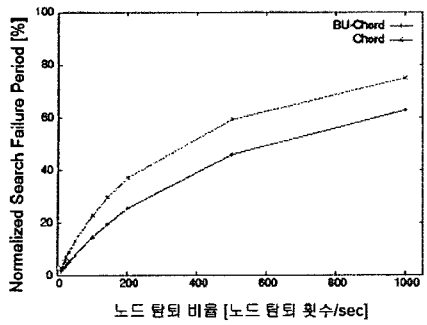
DiPP 노드들은 IP 주소를 SHA-1 해쉬함수의 입력 값으로 넣어 생성한 식별자를 노드의 식별자로 사용한다. 각 노드들의 식별자에 따라 P2P Chord 오버레이 네트워크를 구성하고, BU-Chord의 백업 알고리즘에 따라 백업 노드를 설정한다. 각 노드가 공유하고자 하는 파일의 이름을 key로 사용하여 key값을 해쉬 한 결과를 key의 식별자로 사용한다. 공유할 key의 식별자를 할당할 노드를 선택하는 과정과 key를 검색하는 과정에는 앞서 설명한 Chord의 색인 알고리즘을 사용한다. Chord 색인 알고리즘을 사용하여 key를 검색함으로써, 기존의 대표적인 P2P 검색 방법인 그누텔라 검색에 비해 검색 메시지로 인한 대역폭의 낭비를 막을 수 있다. 그누텔라 검색을 사용하는 P2P응용의 경우 네트워크 전체에 있는 피어들에게 검색메시지가 전달되므로 그 과정에서 불필요한 검색메시지와 중복된 검색 메시지의 전달로 대역폭이 낭비된다. 하지만 Chord 색인 알고리즘의 경우 노드의 개수가 n개인 네트워크에서는 O(log n)개의 검색 메시지만 사용하면 key를 검색 할 수 있어 검색에 사용되는 대역폭을 줄일 수 있고, 검색에 소요되는 시간 또한 절약 할 수 있다. 또한 BU-Chord 복구 알고리즘을 적용하여 어떠한 노드가 네트워크를 떠나거나 꺼지더라도 그 노드가 저장하고 있던 key들이 손실되지 않고 복구되어 계속해서 검색이 가능하도록 유지된다. DiPP의 기능은 네트워크상의 각 노드들과 텍스트 메시지를 주고받을 수 있는 인스턴트 메신저 서비스와 각 노드들이 공유한 파일들을 검색하고 전송 받을 수 있는 파일공유 기능을 수행한다. 아래 그림 10은 DiPP의 실행화면이다. 그림 10 (a)에서는 노드의 식별자와 backup 식별자의 정보들을 보여주는 화면이고, 그림 10 (b)는 공유할 파일을 선택하는 화면이다.

V. 성능평가

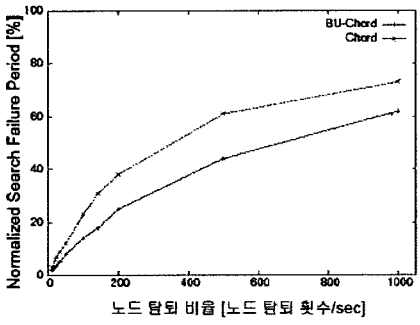
기존의 Chord와 BU-Chord의 key 복구 알고리즘을 적용했을 때의 성능을 시뮬레이션을 통해 비교하였다. Chord는 Chord ring에 참여한 노드가 네트워크를 떠났을 경우 그 노드가 저장하고 있던 key들을 복구하는 방법을 정의하고 있지 않다. 그리하여 본 실험에서는 Chord에 다음과 같은 가정을 적용하였다. Chord 노드의 탈퇴로 그 노드가 저장하고 있던 각 공유할 파일에 해당하는 key들이 손실되면, 초기에 key들을 등록했던 노드들이 다시 key를 새로운 successor에게 할당하는 것으로 가정한다. 우리는 상대적으로 다수의 노드로 구성된 Ad-hoc 네트워크와 적은 숫자의 노드들로 구성된 Ad-hoc 네트워크 두 가지 경우에서의 성능을 비교하였다. 각각 100개와 16개의 노드들이 격자 형태의 네트워크 토폴로지를 구성하도록 설계하였다. 노드들 사이의 BEACON/BEACON-ACK 메시지를 교환하는 오버헤드를 줄이기 위해 프로엑티브(Proactive) 기반의 Ad-hoc 네트워크 라우팅 프로토콜이 사용되는 상황으로 가정하였다. BU-Chord와 Chord의 성능비교를 위해 노드가 네트워크를 떠나는 departure rate를 달리하였다. 각 노드는 평균시간 1초인 정규분포를 따라 검색 메시지를 생성한다. 그 때 검색하고자 하는 키는 랜덤하게 생성되고, 키를 담은 질의 메시지가 키의 successor에 까지 전달되면 검색이 성공한 것으로 가정하고, 그 이외의 경우들을 실패로 간주한다. 만약 손실된 키들이 복구되지 않았다면, 키의 검색은 실패할 것이다.

우리는 노드의 네트워크 탈퇴 비율에 따른 Normalized Search Failure period(NSFP)를 측정하였다. NSFP는 전체 시뮬레이션 시간 중 전체의 key들 중 검색될 수 없는 key의 비율을 나타낸다. 그림 11에서 볼 수 있듯이 노드의 탈퇴 비율이 높아질수록 NSFP의 비율 또한 증가함을 알 수 있다. 노드의 탈퇴 비율과 노드의 밀집도에 상관없이 BU-Chord의 NSFP의 비율이 낮음을 알 수 있다. 이는 BU-Chord의 경우 backup successor의 동작을 통해 각 노드들의 네트워크 탈퇴를 인식하고, 또 그 노드가 저장하고 있던 key들을 즉시 복구 하도록 되어있기 때문이다.

P2P 네트워크의 검색 성공률을 측정하기 위하여 노드의 탈퇴 비율에 따른 히트율(hit ratio)를 측정하였다. 히트율은 각 노드의 총 검색 시도 횟수에서 검색에 성공한 비율로 정의한다. 아래 그림 12에서

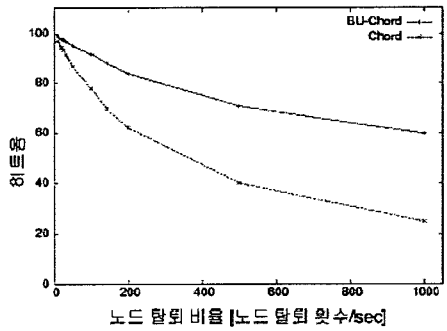


(a) 노드의 개수가 100개 네트워크

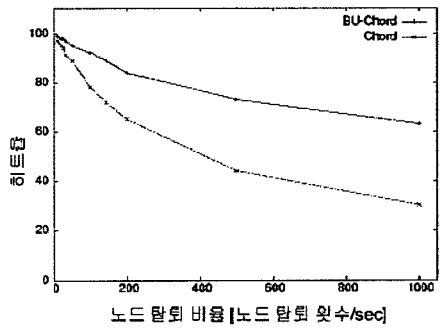


(b) 노드의 개수가 16개 네트워크

그림 11. 노드 탈퇴 비율에 따른 Normalized Search Failure Period

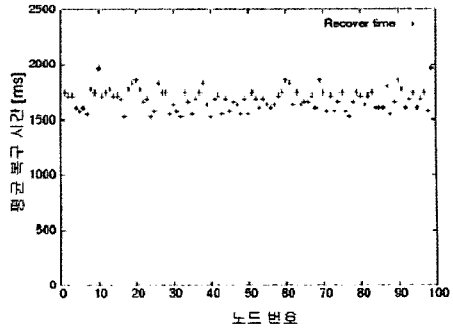


(a) 노드의 개수가 100개 네트워크

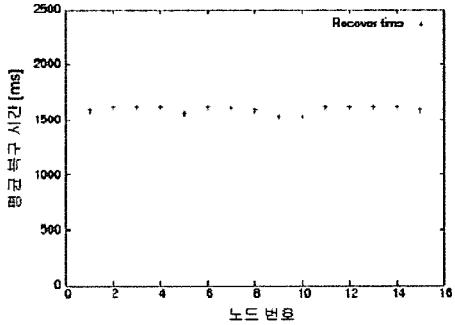


(b) 노드의 개수가 16개 네트워크

그림 12. 노드 탈퇴 비율에 따른 히트율



(a) 노드의 개수가 100개 네트워크



(b) 노드의 개수가 16개 네트워크

그림 13. 각 노드들의 평균 복구 시간

볼 수 있듯이 노드들의 수가 많은 큰 규모와 작은 규모의 Ad-hoc 네트워크 모두에서 탈퇴 비율이 높아질수록 히트율이 낮아짐을 알 수 있다. 그리고 두 경우 모두 BU-Chord의 성능이 우수함을 알 수 있다. 특히 탈퇴 비율이 높아질수록 BU-Chord와 Chord의 성능 차이가 더 커짐을 보여준다.

BU-Chord는 backup successor를 선정하여 key와 정보들을 복사하고, 노드의 탈퇴 후 복구를 수행하는 과정이 네트워크 전체에서 분산되어 수행된다. 따라서 아래의 그림 13과 같이 네트워크의 규모에 상관없이 각 노드들의 평균 복구 시간이 거의 균등하게 나타난다. 네트워크의 규모가 작은 상황에서 복구의 시간이 대체적으로 짧게 나타나는 것은 각 노드들 간의 거리가 상대적으로 짧기 때문이다.

V. 결 론

우리는 MANET 환경의 응용의 형태로 완전 분산형 구조의 파일 공유와 검색을 목적으로 하는 P2P 응용을 설계하였다. MANET의 특성상 모든 노드들은 자유롭게 네트워크를 드나들고, 각 노드의 무선 전송 대역폭이 작기 때문에 모든 노드들의 골

고루 분산하여 동작하는 분산형 구조의 P2P 응용의 형태가 적합하다. 그리고 P2P 형태의 검색에 있어서 좀 더 신뢰성 있고, 검색 메시지의 브로드캐스트로 인한 대역폭의 낭비를 막기 위해서 Chord 색인 프로토콜을 사용하여 문제를 해결 할 수 있다. 하지만 기존의 Chord색인 프로토콜만을 사용한 MANET 상의 P2P 응용에서는 노드의 네트워크 탈퇴로 인한 key손실이 발생할 수 있다. 따라서 본 논문에서는 노드가 네트워크를 탈퇴 할 때 발생하는 key 손실 문제를 해결하기 위해 BU-Chord 메커니즘을 제안하였다. BU-Chord에서는 Backup successor/Backup predecessor 개념을 도입하여 각 노드가 저장하고 있는 key와 이웃노드들의 정보들을 네트워크상에 분산하여 백업해 두고, 만약 어느 노드가 네트워크를 탈퇴하게 되면 그 노드가 관리하던 key들을 새로운 successor에게 복구하여 P2P응용의 계속적인 서비스가 가능하도록 설계하였다.

참 고 문 헌

[1] Internet Engineering Task Force, "Manet working group charter," http://www.ietf.org/html.charters/manet-charter.html

[2] I.Stoica, R.Morris, D.L.Nowell, D.R.Karger, M.F. Kaashoek, F.Dabek, and H.Balakrishnam, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networking, Vol.11, No.1, Feb.2003.

[3] Alexander Klemm, Christoph Lindemann, and Oliver P. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad-hoc Networks", IEEE VTC2003 Fall, October 2003.

[4] C. Perkins, Nokia Research Center, and E. Belding-Royer "Ad hoc On-Demand Distance Vector(AODV) Routing," RFC 3561, July 2003.

[5] Rudiger Schollmeier, Ingo Gruber, and Florian Niethammer, "Protocol for Peer-to-Peer Networking in Mobile Environments", IEEE ICCCN 2003, Dallas, USA, Octorber, 2003.

[6] Gerd Kortuem and Jay Schneider, "An Application Platform for Mobile Ad-hoc Networks", UBICOMP 2001, September-October, 2001.

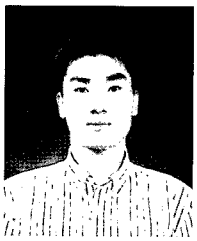
[7] http://people.cs.vt.edu/~irchen/Microsoft-grant/wireless-ad-hoc-messenger/

[8] Napster, http://www.napster.com

[9] "Secure Hash Standard," U.S Dept. commerce/ NIST, National Institute of Standards and Technology, Springfield, VA, FIPS 1801, Apr. 1995.

[10] The Gnutella Protocol Specification v0.4

정 홍 종 (Hong-Jong Jeong) 준회원



2004년 경북대학교 컴퓨터공학과(학사)

2006년 경북대학교 컴퓨터공학과(공학석사)

2006년 3월~현재 경북대학교 컴퓨터공학과 박사과정

<관심분야> Mobile Ad Hoc Network, P2P, 주소자동설정기법 등

송 점 기 (Jeomki Song) 준회원

2004년 경일대학교 컴퓨터공학과(학사)

2006년 경북대학교 컴퓨터공학과(공학석사)

<관심분야> Mobile Ad Hoc Network, 자동 네트 워킹 기술, IPv6

김 동 균 (Dongkyun Kim) 정회원

1994년 경북대학교 컴퓨터공학과(학사)

1996년 서울대학교 컴퓨터공학과(공학석사)

2001년 서울대학교 전기·컴퓨터공학부(공학박사)

1999년 미국 Georgia Institute of Technology, 방문 연구원

2002년 미국 University of California at Santa Cruz, Post-Doc. 연구원

2003년 3월~현재 경북대학교 컴퓨터공학과 조교수

<관심분야> 이동인터넷, 초고속 인터넷, Mobile Ad Hoc Network, 무선 LAN 등