P2P 네트워크에서 익명성 보장을 위한 패킷 대치 기법

김 병 룡[†] · 김 기 창[††]

요 약

플러딩 기반 P2P 시스템은 기본적으로는 익명성을 제공하고 있다. 그리고 익명의 상태에서 사용자와 제공자가 정보를 교환하고 있다. 노드에서 노드로 전달되는 대부분의 패킷들은 패킷을 보낸 노드의 신원정보를 포함하고 있지 않다. 그리고 이러한 패킷들은 중간 노드들이 동적으로 구성하고 있는 라우팅 시스템에 의해서 목적지로 전송되어진다. 따라서 누가 최초로 전송했는지, 누가 지정된 수신지인지 아는 것은 불가능하다. 그러나 다운로드와 업로드 하는 호스트의 IP주소가 외부로 드러나기 때문에 익명성을 제공하지 못하고 있다. 본 논문에서는 익명성이 문제가 될 수 있는 시스템에서 검색질의 응답 패킷의 교체를 통해서 사용자와 리소스 제공자의 신원 보호를 위해 익명성을 제공할 수 있는 기법을 제안 한다.

키워드 : Peer-to-Peer, P2P, 익명성보장

Packet Replacement Technique for Securing Anonymity in P2P Network

Kim Byung Ryong[†] · Kim Ki Chang[††]

ABSTRACT

Flooding based p2p system basically provides anonymity and under the anonymity circumstances user and provider exchange information. Most of packets transferred from node to node do not contain identity information on node that sent packet. And these packets are transmitted to the destination through the routing systems dynamically composed of intermediate nodes. Therefore it is impossible to know who transmitted it for the first and who the designated recipient is. But since downloading and uploading host's IP address is exposed it does not provide anonymity. This study introduces techniques to provide anonymity for protecting identification of users and resource providers by replacing QueryHit packets in systems where anonymity can cause trouble.

Key Words : Peer-to-Peer, P2P, Securing Anonymity

1. 서 론

Peer-to-Peer(P2P)는 직접, 혹은 간접적으로 연결된 컴퓨터들 간의 리소스의 공유이다. 서버/클라이언트 환경에서 사용자는 원하는 데이터를 검색하기 위해 서버로 검색요청을 하게 된다. 이때에 모든 사용자의 검색 요청은 중앙서버로 전송되고 많은 양의 검색 요구를 동시에 처리해야 하기 때문에 부하가 발생하고, 이로 인해서 서버가 클라이언트에게 제공하는 서비스의 질이 떨어지게 되거나, 서버의 처리 능력을 향상시키기 위해 많은 비용을 지불 해야만 한다. 그러나 순수 P2P 네트워크에서는 검색이 어느 한곳의 PC나 중앙서버에서 이루어지는 것이 아니고, 분산된 검색을 하기 때문에 검색처리 비용이 덜 들고, 업데이트 비용 또한 적게 된다.

현재 이러한 P2P 검색 시스템은 크게 플러딩 기반 모델과 분산 해시 테이블 기반 모델로 나눌 수 있다. 플러딩 모델은 FreeNet[1]과 Gnutella[2, 17]와 같은 형태이고, 분산 해시 테이블 모델은 Tapestry[3, 4], CAN[5], Chord[6]와 같은 모델을 예로 들 수 있다.

플러딩 모델은 많은 질의를 브로드캐스팅 하기 때문에 여러 가지 문제가 발생되고 있고 이를 보완한 모델이 바로 분산 해시 테이블 기반 모델이다. 플러딩 모델은 연결 가능한 노드들의 연결정보를 수집한 뒤에, 연결된 모든 노드로 검색 질의를 브로드캐스팅 한다. 그러나 분산 해시 테이블 기반 모델은 노드의 위치정보(IP)나 콘텐츠가 검색을 위한 해시 함수의 키로 이용되기 때문에 플러딩 모델의 검색 트래픽이 $O(N)$인 반면에 분산 해시 테이블 기반 모델은 $O(log(N))$로 줄게 되고, 검색 보장에 대한 측면에 있어서도 훨씬 성능이 높다.

P2P시스템이 서버/클라이언트 구조를 벗어나서 많은 잇

※ 이 논문은 인하대학교의 지원에 의하여 연구되었음.
† 준 회 원 : 인하대학교 전자계산공학과
†† 정 회 원 : 인하대학교 정보통신공학부
논문접수 : 2005년 1월 7일, 심사완료 : 2005년 4월 11일

점을 얻을 수 있지만, 몇 가지 문제가 있다. 다른 여러 프로토콜과의 확장성 문제 및 리소스 공유에 대한 책임을 지지 않는다는 것, 얼마만큼 책임지고 있는지 관리하고 측정하기 힘들다는 것, 그리고 각 노드들이 관리적 측면에서 독립되어 있기 때문에 어떤 노드와 통신하고 있다는 사실은 알 수 있지만 그 노드가 누구인지 정확한 정보를 알기 힘들다는 것이다.

보안적 측면에 있어서도 서버/클라이언트의 환경에서 보여주는 견고함 만큼의 신뢰도를 유지하기에는 P2P시스템이 너무 자유롭고 무책임하다. 하지만 P2P의 성격상 완벽하게 관리할 수도 없고 각 노드가 독립적인 권한을 가지고 스스로 관리해야 하므로 각 노드들은 익명성을 가지고 있어야 한다. 플러딩 모델인 Gnutella 프로토콜인 경우에 브로드캐스팅 되는 검색질의와 PING 패킷은 기본적으로 익명성을 제공하고 있으나, 최종 목표인 다운로드와 업로드 시에는 라우팅을 하지 않기 때문에 이때에는 익명성을 제공할 수가 없다. 따라서 불특정 다수가 참여하고 있는 P2P 네트워크 안에서 특정한 노드의 정보가 노출되게 되는 경우가 발생된다. 따라서 악의적이지 않을지라도 노출된 노드가 서비스 거부 공격 및 저장 공간 범람과 같은 공격에 노출될 수 있다[7, 8].

본 논문에서는 플러딩 모델에서 노드들 간에 업로드와 다운로드 시에도 익명성을 제공할 수 있도록, 선정된 중계노드에서의 검색질의 응답 패킷 대치 기법을 제안한다. 검색질의와 검색질의 응답 패킷의 구조적 특성을 이용하여 원하는 콘텐츠의 위치를 이웃한 다른 노드의 위치로 가장 시킨다. 가장되어 선정된 중계노드는 서버와 클라이언트 사이에서 콘텐츠 서비스를 시도한다.

제안된 기법은 MUTE[9]와 같이 익명성을 제공하는 파일 공유 기법에서 사용했던 서버와 클라이언트의 연결 경로에 위치한 모든 노드를 통해서 파일을 전송할 때 발생했던 대용량 콘텐츠 전송의 문제점도 해결 할 수 있다. 또한 Mantis[10]나 Onion Routing[11, 12]에서 사용하는 노드간의 암호화된 통신이나, 콘텐츠를 서비스할 때 서비스를 제어하기 위해서 서버와 클라이언트사이의 경로에 놓여진 모든 노드를 통해 서비스 제어 데이터를 송수신 하지 않아도 익명성을 제공할 수 있는 장점이 있다.

본 논문의 2장에서는 플러딩 기반 네트워킹의 개념과 기존의 익명성을 위한 기법 및 제안된 기법에서 이용한 브로드캐스팅 되는 패킷의 확률적 특성에 대해서 알아본다. 3장에서는 제안한 알고리즘을 자세하게 살펴보고 이에 따른 문제점의 해결 방안에 대해서 이야기한다. 4장에서는 제안한 알고리즘을 구현한 운영 결과를 보이고, 5장에서는 기존의 연구들과의 비교 평가를 하고, 6장에서 결론을 맺는다.

2 관련 연구

2.1 동적 라우팅

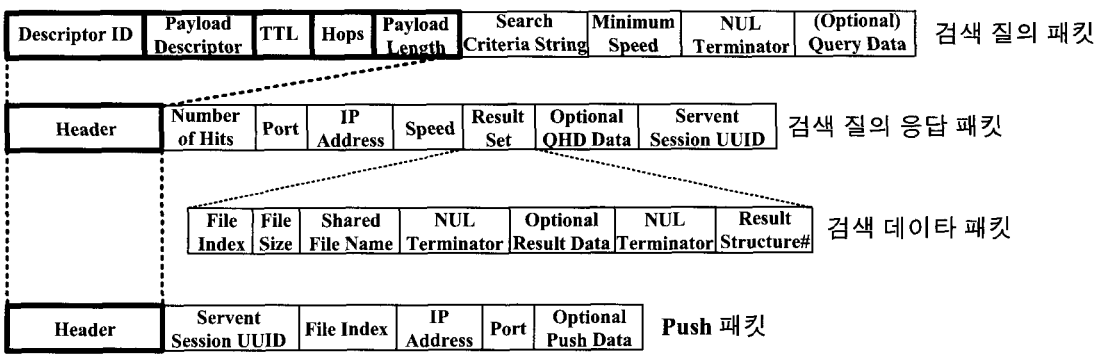
플러딩 기반 네트워크에 참여하는 각 노드는 지속적인 연결을 요구하지 않는다. 분명히 각 노드는 네트워크에 참여하고 있으나 반드시 직접적으로 연결되어 있지는 않다. 각 노드는 메시지를 중계하면서 자신도 모르게 서로가 연결되는 것이다. 질의와 PING패킷과 같이 서로가 서로에게 메시지를 브로드캐스트하고 중계하게 된다.

Gnutella 네트워크에서는 노드마다 UUID(GUID)또는 128비트의 고유 식별자를 노드마다 부여하게 되는데, 이 UUID 값이 각 노드를 거칠 때마다 해당 노드에 기억되고, 나중에 자신과 연결된 노드 중에서 이 패킷을 어느 연결로 되돌려 보낼지를 결정할 때 사용되게 된다[2, 17, 18]. 이렇게 브로드캐스팅 되는 패킷들에 의해서 동적으로 라우팅이 만들어지게 되는 것이다. 따라서 패킷들은 노드들의 IP주소와 포트번호에 의해서 구별되는 것이 아니고, UUID에 의해서 식별되게 된다. 그러므로 UUID없이는 라우팅을 통해서 패킷을 되돌릴 방법이 없다.

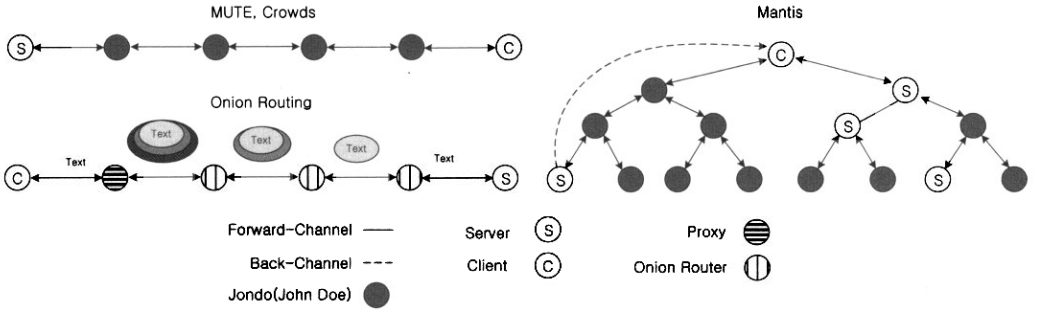
2.2 기존의 익명성 보장을 위한 방법들

서버와 클라이언트의 익명성 보장을 위한 방법에는 (그림 2)와 같이 MUTE[9]와 Onion Routing[11, 12], Crowds[13], 그리고 Mantis[10]등을 예로 들 수 있다.

(그림 2)에서 MUTE는 파일 공유에서의 익명성 보장을 위해서 서버와 클라이언트가 직접 연결되어 파일 공유 서비스를 하는 것이 아니라, 파일 송수신을 Jondo와 같은 중간 노드들을 모두 거치면서 동작 하게 된다. 따라서 대용량의



(그림 1) 검색 질의 패킷과 검색질의 응답 패킷 및 PUSH패킷 형식



(그림 2) 익명성 보장을 위한 기존의 기법들

멀티미디어 파일을 송수신 하고자 한다면 모든 노드들을 거치기 때문에 상당한 대역폭의 낭비를 초래할 수밖에 없다.

Onion Routing은 익명의 연결을 보장하기 위해서 라우팅 헤더를 숨기고 라우팅 경로의 추적을 위한 통계학적 추정을 어렵게 하기 위해서 데이터 암호화를 사용하고 있다. 클라이언트는 최초에 Onion Routing에 접속하기 위해서 임의의 라우팅 경로를 만들고 데이터를 암호화 하는 프록시에 연결해야 한다. 그리고 만들어진 라우팅 경로를 따라 Onion 라우터들을 통해서 목적지까지 도달하게 된다. 각 라우터들은 암호화된 데이터 층을 하나씩 해제해 나가면서 그 다음 라우터를 알게 되고, 결국 최종 목적지까지 도달하게 된다. 그리고 다시 응답이 송신 경로를 따라 역으로 돌아갈 때는 다시 암호화된 데이터 층을 하나씩 덧붙여 나가면서 최초 송신자에게 전송됨으로써 서버와 클라이언트 상호간의 익명성을 보장한다. 하지만 암호화라는 추가적인 작업이 필요하다.

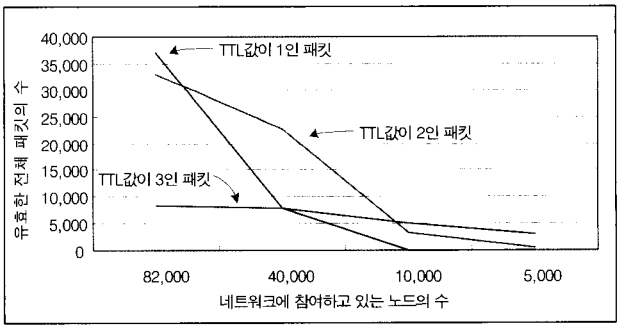
Crowds는 사용자가 웹 브라우징을 하는 동안에 사용자의 프라이버시를 보호하기 위해서 개발 되었다. Crowds에 참여하고 있는 클라이언트는 자신이 원하는 콘텐츠를 얻기 위해서 미리 서버의 주소를 알고 있는 상태에서 서버에게 요청하는 것이 아니고, 현재 Crowds에 참여하고 있는 다른 클라이언트에게 콘텐츠 요청을 하는 것이다. 따라서 서버는 네트워크에 참여하고 있는 많은 사용자들 중에서 누가 제일 처음 요청을 하였는지 알 수가 없다.

Mantis는 Crowds와 비슷하지만, 요청이 Jondo들을 거치면서 익명의 서버들에게 전파되고, 일치하는 경우에는 원래 서비스 요청을 한 노드로 응답을 보내게 된다. 그리고 서버는 자신의 IP주소를 숨기고 UDP통신을 이용하여 클라이언트와 콘텐츠 서비스를 수행한다. 따라서 이 경우에도 검색요청을 보낸 클라이언트가 누구인지, 그리고 응답을 보낸 서버가 누구인지 알 수 있는 노드가 어디에도 없게 된다. 그러나 (그림 2)에서처럼 서버와 클라이언트가 UDP통신을 하는 도중에 서버와 클라이언트 사이의 트리에 연결된 모든 노드를 통해서 패킷 손실이나 재전송 제어를 위한 제어 데이터 교신이 필요하다. 그리고 각 노드들 간에 연결은 Onion Routing처럼 암호화되어 있다.

2.3 브로드캐스팅 되는 패킷의 분포

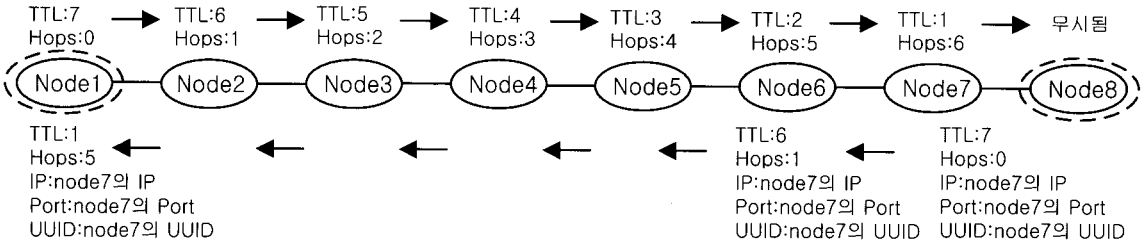
PING과 검색질의 패킷은 주기적, 또는 요청이 있을 경우에 브로드캐스팅 된다. (그림 3)은 모든 패킷중에서 UUID값에 의해서 중복되어 수신된 패킷을 제외한 패킷, 즉 노드에 전송된 다음에 리턴값(검색결과 또는 응답값)을 만들어 내게 하는 패킷의 분포도를 나타낸 것이다. 네트워크에 참여하고 있는 한 개의 노드가 검색질의 패킷을 한번 브로드캐스팅 하였을 때의 네트워크 전체 패킷의 분포도이다. Minism시뮬레이션 프로그램[15]을 수정 보완하여 네트워크에 참여하고 있는 노드의 수에 따라서 시뮬레이션 하였고, 노드 당 최대 연결수와 TTL값은 Newtella 데몬[16]에서 사용 하고있는 7로 설정하여 시뮬레이션 하였다.

전체 패킷들을 TTL값을 기준으로 분류 하였다. 한 개의 노드가 패킷 한 개를 한번 브로드캐스팅한 결과이고, Gnutella 프로토콜 v0.4 기반인 Newtella 데몬을 기준으로 한 것이다. 이론적으로 TTL값이 7이고 최대 연결수가 7일 경우 7^7=823,543개의 노드의 콘텐츠를 검색할 수 있다. 이중에 약 10분의 1정도가 파일을 공유한다고 가정했을 때, 약 82,000개의 노드에 대해서 검색이 가능할 것이다. 우연히도 Edonkey2000의 서버 리스트를 관리하는 http://ed2k.2x4u.de/list.html 사이트에서 상위20개 서버의 평균접속자수도 이와 비슷한 수치를 나타내었다. TTL값이 4,5,6,7인 경우는 비교적 매우 작은 수이기 때문에 생략하였다.

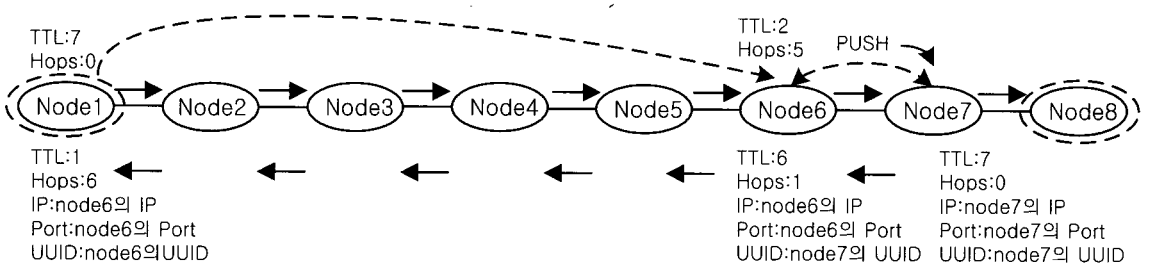


(그림 3) 전체 네트워크에 퍼져있는 유효한 패킷의 분포도

TTL값이 1인 유효한 패킷의 수는 네트워크에 참여하고 있는 노드들의 수가 감소할수록 급격하게 감소하고 유효한 패킷의 양도 노드 수에 비해 매우적다. TTL값이 2와 3인 유효한 패킷인 경우에는 노드수가 많지 않다고 가정했을 때 급격한 변화를 보이지는 않고 있다.



(그림 4) Gnutella 네트워크의 검색 질의 흐름도



(그림 5) 교체된 검색 응답 패킷에 의한 콘텐츠 다운로드

(그림 3)에서 한 개의 노드에서 한번 브로드캐스팅한 한 개의 검색질의 패킷 중 유효한 패킷은 TTL값이 2,3일 때가 가장 많고 그중에서도 2일 때라고 볼 수 있다. 따라서 TTL 값이 2인 검색 패킷을 받은 노드가 검색 결과를 가장 많이 전송하게 될 것이다. 그러므로 이런 노드의 콘텐츠를 다운로드하게 될 가능성이 가장 높다고 할 수 있다. 본 논문에서 제안한 검색질의 응답 패킷 대치 기법은 이러한 통계적 특성을 이용하고 있으며, 3절에서 자세히 설명 한다.

3 검색질의 응답 패킷 대치 기법

본 절에서는 플러딩 기반 P2P네트워크에서의 익명성 제공을 위한 알고리즘을 서술하고, 기존의 방법과 어떻게 다른지 설명한다. 제안된 기법에서 발생할 수 있는 문제점을 살펴보고 이를 해결할 수 있는 대안을 제시한다.

3.1 패킷 대치 기법

Gnutella 네트워크에서 기본적으로 제공하고 있는 익명성이 다운로드와 업로드에서 사용자의 신원이 노출됨으로써 한계에 부딪치게 된다. 이로 인해서 예기치 못한 상황이 발생될 수도 있을 것이다. 예를 들어 리모트 컨트롤러와 같은 악의적인 파일을 공유한 다음에 이 파일을 다운로드한 노드의 정보(IP주소)를 이용하여 다양한 공격을 시도할 수 도 있으며, 노출된 정보를 악용하여 서비스 거부 공격을 할 수도 있을 것이다[7].

본 논문에서 제안한 익명성 제공 기법은 검색질의 응답 패킷 중 IP주소와 포트번호를 조작하고 대치한 다음에 이런 정보를 세션UUID값을 키로 하는 테이블에 저장하고, 대치된 패킷을 동적 라우팅을 통하여 최초로 검색질의 패킷을 보낸 노드로 전송 한다. 그리고 다운로드 요청이 오면 프록시처럼 동작하게 된다.

(그림 4)는 일반적인 Gnutella 프로토콜을 예로 들어 검색질의/응답 패킷이 전송되는 모습을 나타낸 것이다. Node1이 검색질의 패킷을 브로드캐스팅하여 결국 Node7까지 도달하게 되고, Node7에서 검색질의에 일치하는 콘텐츠가 검색되어 응답 값을 다시 역방향으로 되돌려주는 전체적인 흐름도이다.

Node6은 Node1이 보낸 검색질의 패킷을 받아서 Node7로 전송하고, Node1의 UUID를 Key로 하는 SearchRoutes[16]라고 불리는 동적 라우팅 테이블을 업데이트 한다. Node7로의 검색질의 응답 메시지가 도착하고 검색된 콘텐츠 명 및 인덱스 등을 Node6으로 전송하게 된다. Node6은 이 패킷이 자신의 검색질의에 대응되는 결과값이 아니기 때문에 Node7의 세션UUID값을 키로 하는 라우팅 테이블인 Push Routes[16]에 Node7의 연결 정보를 저장한다. 그런 다음에 다시 동적라우팅을 통해 Node5로 전송된다. 이런 흐름을 반복적으로 수행하여 최종적으로 Node1으로 전송되고, Node1은 Node7의 정보를 이용하여 콘텐츠 다운로드를 요청하게 된다. 그러나 이렇게 되면 Node1과 Node7은 서로의 정보가 노출되기 때문에 익명성을 상실하게 된다.

본 논문에서는 이러한 문제를 해결하기 위해서 검색 요청을 보낼 때는 일반적인 방법을 통해서 브로드캐스팅하고 검색요구에 일치하는 경우 응답 값을 전송하는데, 그 응답 값을 최초로 받은 노드는 동적 라우팅 방법에 의해서 그대로 되돌려지는 것이 아니라, 응답 값인 검색질의 응답 패킷을 대치한 뒤에 전송하는 것이다.

(그림 6)에서 Node7은 Node1이 전송한 검색 패킷에 대한 응답 값을 되돌리게 된다. 이 응답 값을 제일 처음 수신하

는 Node6은 검색질의 응답 패킷의 TTL과 Hops의 값이 각각7과 0임을 확인할 수 있기 때문에 Node6은 이 검색질의 응답 패킷을 제일 먼저 수신한 노드가 자신임을 확인하게 된다. 이럴 경우, Node6은 검색질의 응답 패킷을 그대로 Node5로 전송하는 것이 아니라 Node7이 보낸 검색질의 응답 패킷의 IP주소, 포트번호를 자신(Node6)의 것으로 대치하고 이러한 정보와 원래의 검색질의 응답 패킷의 관계 정보를 테이블(PushRoutes)에 저장한 뒤에 대치된 검색질의 응답 패킷을 Node5로 보내게 된다. 결국에는 이 패킷이 동적 라우팅을 통해 중간 노드들을 거쳐 Node1까지 전송된다.

Node1은 대치된 검색질의 응답 패킷을 받았기 때문에 Node6에게 연결한 다음에 HTTP헤더를 전송함으로써 콘텐츠 다운로드를 요청하게 된다. 그러나 헤더에 포함된 UUID값은 Node6의 세션UUID값이 아니라 Node7의 세션UUID값이다. 이때 Node6은 PushRoutes에서 헤더의 세션UUID값을 검색한 결과 Node7의 세션UUID값임을 알게 된다. HTTP Get 요청은 아래와 비슷하게 전송된다.

```
GET /get/<File Index>/<File Name>/<Server Session UUID>/HTTP/1.0\r\n
User-Agent: Gnutella/0.4\r\n
Range: bytes=<Start Offset>-\r\n
Connection: Keep-Alive\r\n
\r\n
```

Node6은 Node7로 PUSH패킷을 전송한다. 그러면 Node7은 Node6으로 새로운 연결을 만들고 콘텐츠를 서비스한다. 이제 Node6은 Node1과 Node7사이에서 HTTP헤더를 주고받으면서 중계자 역할을 하게 된다. 즉, Node6은 서버(Node7)와 클라이언트(Node1)의 익명성을 보장하기 위한 중계노드로 선정된 것이다.

제안된 기법은 중간 단계까지는 Crowds와 비슷하지만 실제로 콘텐츠를 서비스할 때는 모든 노드들을 거치면서 서비스를 하는 MUTE와는 다르고, Mantis와는 유사하다. 하지만 콘텐츠를 서비스하기 위해서 Node1과 Node6사이에 있는 모든 노드들을 거치면서 서비스 제어 데이터를 주고받지 않는다는 점에서 다르다. 또한 Gnutella 프로토콜과 호환되기 때문에 검색질의/응답 패킷 송수신시에 각 노드들과 암호화된 통신을 수행하지 않는다.

3.2 중계 노드가 방화벽/NAT 환경 내에 있는 경우

검색질의 응답 값을 넘겨받은 중계노드가 방화벽/NAT 환경 내에 있는 경우에는 (그림 6)의 절차를 따른다. 중계노드가 방화벽/NAT환경 내에 있는 경우 클라이언트는 직접 중계노드로 서비스 요청을 할 수 없다. 중계노드는 서버로부터 받은 검색 응답 패킷을 테이블에 저장하지 않고, 동적 라우팅에 의해서 검색 질의 패킷을 보낸 이웃한 피어에게 패킷을 대치하지 않고 그대로 전송하여 중계역할을 이웃한 피어에게 위임한다. 위임된 중계노드는 제안된 패킷 대치 알고리즘에 따라서 동작한다. (그림 6)과 같이 위임된 중계노드와 서버 간에 직접 연결이 없다면 PUSH패킷을 어떻게 전송할 것인가? 검색질의 패킷을 최초로 수신한 중계노드는 위임된 중계노드와 서버 간에 연결이 없을 가능성이 있기 때문에 아래와 같은 절차를 따른다.

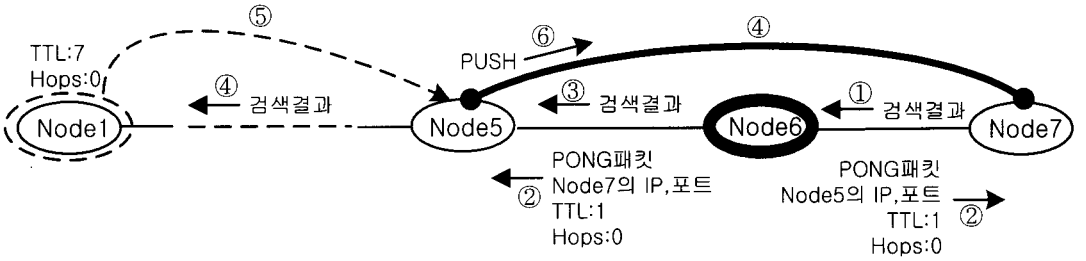
Node6은 마치 Node5와 Node7이 PING패킷을 보내서 그 응답 값으로 PONG패킷을 보내는 것처럼 가장하여 서버와 위임된 중계노드에게 서로의 IP주소와 포트번호를 넣어 PONG패킷을 전송한다. 이렇게 되면 서버와 위임된 중계노드는 연결이 안 되어 있다면 연결을 시도할 것이다.

응답 패킷을 받은 이웃노드인 위임된 중계노드는 3.1절에서 설명한 절차대로 패킷을 재구성하여 대치한 뒤에 백 워드 경로로 전송 한다. 이제 콘텐츠 요청은 방화벽/NAT환경 내에 있는 중계노드가 아니라 서버와 직접 연결된 다른 위임된 중계 노드에게 전송 된다.

3.3 네트워크 트래픽 제어

중계 노드가 아주 낮은 대역에 연결되어 있고, 서버 노드가 높은 대역에 연결되어 있을 경우에, 중계 노드가 서비스를 중계하기 때문에 큰 양의 콘텐츠를 전송 받고자 하는 클라이언트에게는 아주 낮은 서비스를 제공할 수밖에 없다. 예를 들어 MUTE의 경우에는 많은 중계노드를 거치게 되기 때문에 더 큰 문제가 발생 된다. 그리고 서버에 상당히 인기 있는 콘텐츠가 있을 경우에, 고속으로 연결된 중계 노드일지라도 서비스 요청이 많아지면 감당하기 어렵게 될 수도 있다. 본 논문에서는 이 문제를 해결하기 위해서 2.3절에서 설명하였던 패킷 분포 및 패킷 수신 확률에 기반 한 캐싱 기법을 적용 한다.

2.3절의 (그림 3)에서, 한 개의 노드가 한 개의 검색질의 패킷을 브로드캐스팅 하였을 때 생성되는 질의 패킷 중에서,



(그림 6) 중계노드가 방화벽/NAT 환경 내에 있는 경우의 처리 흐름도

TTL값이 2인 질의 패킷이 가장 많음을 알 수 있었다. 따라서 나머지 패킷들은 중복 수신과 이동 경로 루프를 막기 위해서 거의 파기 되었다는 것이다.

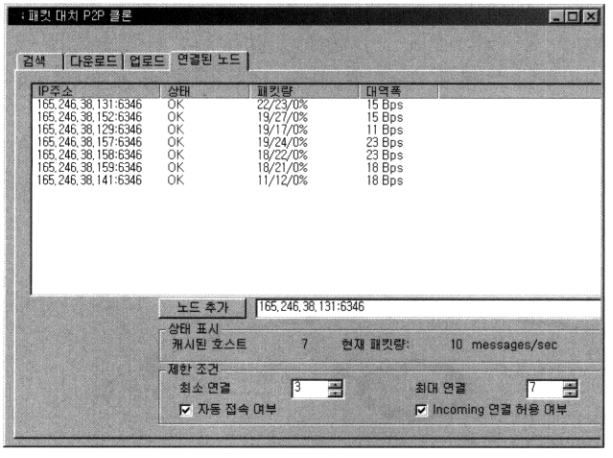
검색질의 패킷을 수신하였을 때, 그 패킷의 TTL값이 2인 확률이 가장 높기 때문에, (그림 3)에서 보는 것처럼 검색질의 응답 패킷 역시 TTL값이 2인 검색 질의 패킷에 대한 응답일 확률이 가장 높다. 따라서 중계노드도 클라이언트로부터 TTL값이 2인 검색질의의 응답 패킷에 대한 콘텐츠 서비스 요청을 받을 확률이 가장 높다.

순수 P2P 환경에서 중계노드는 현재 네트워크에서 가장 인기 있는 콘텐츠가 무엇인지 알기는 쉽지 않다. 그리고 P2P 환경에서 어떤 콘텐츠가 인기 있다면, P2P 네트워크의 참여하고 있는 사용자의 특징에 비추어 볼 때, 그 콘텐츠는 시간이 지나면서 이미 수많은 노드로 다운로드 되었을 것이다. 따라서 어느 한 노드에 집중적으로 콘텐츠 서비스 요청이 지속적으로 오랫동안 요구되지는 않는다[21].

위에서 설명한 것처럼 전반적으로 순수 P2P 네트워크에서 인기 있는 콘텐츠가 무엇인지 알아내서 캐싱 하기가 어렵기 때문에, 만약 서비스 요청 빈도가 증가한다면 중계노드는 캐싱 능력 범위에 맞추어 TTL값이 2인 검색 질의와 이에 해당하는 결과 값인 검색질의 응답 패킷에 대한 콘텐츠 및 서비스 정보를 캐싱 한다. 이렇게 하면 전체적인 중계노드들과 서버 노드들 간에 전송양이 줄어든다.

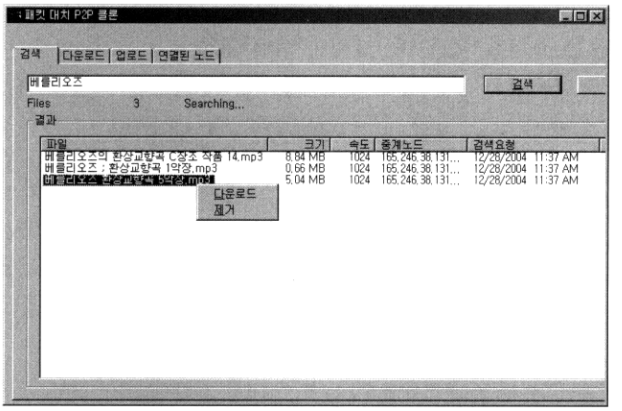
4. 구 현

본 논문에서 제안한 패킷 대치 알고리즘을 테스트하기 위해서 Windows 2000 Advanced Server 플랫폼을 이용하여 Gnutella 프로토콜 v0.4(Document revision 1.2)기반으로 개발하였고, 개발 툴은 Microsoft 사의 Visual C++ 6.0을 이용하였다. 구현한 프로그램의 인터페이스 구성은 (그림 7)에서 보인다.

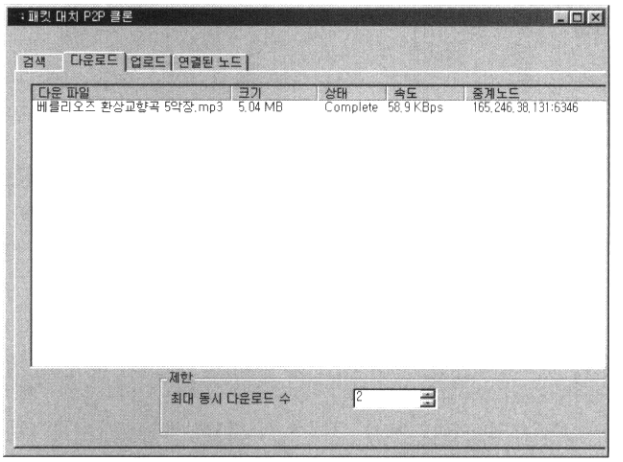


(그림 7) 구현된 테스트 프로그램의 인터페이스(노드 연결 화면)

(그림 7)은 현재 P2P 네트워크에 연결된 노드들을 나타낸 것이다. 최대 연결은 7, 최소연결은 3, 그리고 TTL은 7로 설정하였다. (그림 8)는 검색질의 패킷을 보내고 응답 값인 검색질의 응답 패킷의 내용을 표현한 화면이다. 현재 콘텐츠 다운로드 요청을 보내려고 하고 있다. 현재 검색된 콘텐츠를 가지고 있는 노드들의 IP주소들은 대치된 주소가 표시되고 있다.



(그림 8) 구현된 테스트 프로그램의 인터페이스
(검색 및 다운로드 요청 화면)



(그림 9) 구현된 테스트 프로그램의 인터페이스
(다운로드 완료 화면)

(그림 9)는 다운로드가 완료된 콘텐츠의 상태를 표시하고 있다. 중계노드에서 중계되어 전달된 콘텐츠가 가장된 서버 노드로부터 58.9KBps의 속도로 다운로드가 완료되었음을 알 수 있다. 그림에서 중계노드는 (그림 8)에서의 중계노드로부터 다운로드 되었음을 나타내고 있다.

5. 비교 평가

본 논문에서 제안한 익명성 보장 기법은 플러딩 기반 모델인 Gnutella 프로토콜 0.4를 기반으로 하였다. Gnutella 프로토콜을 따르기 때문에 여타의 암호화가 필요 없고, 그리고 익명성 제공을 위해서 모든 노드들을 거치면서 데이터

전송을 하지 않기 때문에 MUTE에 비해 대용량 파일 전송에도 유리하다. 그리고 추가적으로 중계노드를 사용하고 있지만 Mantis처럼 중계노드와 클라이언트 간에 직접 연결하여 파일을 전송한다. Mantis는 파일 전송 제어를 위해서 서버와 클라이언트 간에 연결된 모든 노드를 통한 제어 데이터를 전송해야 하며, 게다가 모든 노드들 간에는 통신은 Onion Routing처럼 암호화 되어 있기 때문에 암호화 비용 및 네트워크 부하를 초래한다. 하지만 제안한 기법은 검색 질의응답 패킷에 대해서 최초로 수신한 노드가 중계노드가 되어 패킷을 대치한 뒤에는 HTTP프로토콜 기반 하에 직접 통신한다. 따라서 서버-클라이언트 사이에 놓인 모든 노드를 통한 제어 데이터 통신이 필요 없다. 그리고 선정된 중계노드가 방화벽 환경내에 있는 경우에는 중계 임무를 방화벽에 놓여있지 않는 이웃노드로 그 역할을 위임하기 대문에 방화벽 환경내에 있는경우에는 서버와 클라이언트의 익명성을 보장할 수 있다. 정리하면 아래와 같다.

〈표 1〉 기존의 익명성 제공 기법과 패캣 대치 기법의 비교

	Onion Routing	MUTE	Mantis	패킷 대치 기법
Gnutella 호환성	없음	없음	없음	있음
암호화	사용	사용	사용	사용 안함
중계노드 수	다수	다수	중계노드 없음 전송 제어를 위한 다수의 중계 노드	단일 중계노드
방화벽 문제	고려 하지 않음	고려 하지 않음	고려 하지 않음	중계 역할을 위임 하여 문제 해결
익명성 제공	클라이언트 및 서버의 익명성 제공	클라이언트 및 서버의 익명성 제공	서버측 익명성 제공	클라이언트 및 서버의 익명성 제공

6. 결 론

현재, P2P환경에서 발생하는 익명성 보장에 대한 여러 기법들이 제안되고 있다. 하지만 완전한 클라이언트의 익명성과 완전한 서버의 익명성을 동시에 보장하는 것은 어렵다. 보장한다 하더라도 그에 따르는 많은 부가적인 비용을 초래한다.

본 논문에서는 순수 P2P환경에서 완전한 클라이언트와 서버의 익명성을 보장하기 위해서 플러딩 기반 모델인 Gnutella 프로토콜을 기반으로 하여 서버가 보낸 질의검색 응답 패킷을 최초로 수신한 노드에서 패킷을 재구성하여 전송하는 패킷 대치 기법을 제안하였다.

본 알고리즘은 클라이언트가 직접 서버에게 콘텐츠 전송 요구를 하지 않고, 서버가 전송한 검색질의 응답 패킷을 최초로 수신한 노드가 중계노드로 선점되어 패킷을 대치한 뒤에 전송하기 때문에 클라이언트는 중계 노드로 콘텐츠 서비스를 요청하게 된다. 그런 다음에 클라이언트와 중계노드, 그리고 서버 간에 콘텐츠 서비스를 위한 통신을 수행하기 때문에 클라이언트와 서버의 익명성이 유지된다.

기존에 제안된 기법들과 비교하였을 때, 이 기법은 서버와 클라이언트간의 통신에 연결된 모든 노드를 이용하여 전송하지 않고 중계노드를 선점하여 중계노드로만 통신하고 Gnutella 프로토콜을 따르기 때문에 노드들 간의 통신에 암호화가 필요 없다. 그리고 콘텐츠 서비스를 위한 제어 데이터를 다른 노드를 통해서 송수신 하지 않기 때문에 네트워크 트래픽 측면에서도 장점을 가지고 있다.

향후, 익명성이 요구되는 P2P시스템에 이 기법을 적용하여 보다 안전한 서비스를 제공할수 있을 것으로 기대된다.

참 고 문 헌

[1] "The Freenet Project.," http://freenet.sourceforge.net/

[2] The Gnutella Protocol Specification v0.41 Document Revision 1.2.

[3] Kirsten Hildrum, John Kubiatowicz, Satish Rao and Ben Y. Zhao, "Distributed Object Location in a Dynamic Network," Theory of Computing Systems, March., 2004.

[4] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," IEEE Journal on Selected Areas in Communications, January, 2004.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Schenker, "A scalable content-addressable network," Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications table of contents.

[6] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking, February, 2003.

[7] Neil Daswani, Hector Garcia-Molina, "Query-flood DoS attacks in gnutella," Proceedings of the 9th ACM conference on Computer and communications security table of contents, 2002.

[8] P. Krishna Gummadi, Stefan Saroiu, Steven D. Gribble, "A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems," ACM SIGCOMM Computer Communication Review, January, 2002.

[9] "MUTE: Simple, Anonymous File Sharing.," http://mute-net.sourceforge.net/

[10] Stephen C. Bono, Christopher A. Soghoian, Fabian

Monrose, "Mantis: A Lightweight, Server-Anonymity Preserving, Searchable P2P," Information Security Institute of The Johns Hopkins University, Technical Report TR-2004-01-B-ISI-JHU, June, 2004.

[11] Michael G. Reed and Paul F. Syverson, "Onion Routing," Proceeding of AIPA '99, March, 1999.

[12] Roger Dingledine, Nick Mathewson, Paul Syverson, "Tor: The Second-Generation Onion Router," Proceedings of the 13th USENIX Security Symposium, August, 2004.

[13] Michael K. Reiter, Aviel D. Rubin, "Crowds: anonymity for Web transactions," ACM Transactions on Information and System Security (TISSEC), November, 1998.

[14] "Gnutella Web Caching System.," http://www.gnucleus.com/gwebcache/

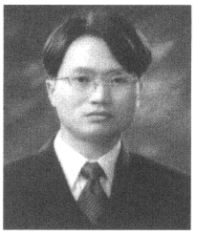
[15] "Gnutella Developer Forum.," http://groups.yahoo.com/group/the_gdf/

[16] "Open Source Newtella for Linux et al.," http://gnewtellium.sourceforge.net/

[17] "The Annotated Gnutella Protocol Specification v0.4 [(1)].," http://rfc-gnutella.source forge.net/developer/stable/index.html/

[18] A. Oram, "Peer-to-Peer," O'Reilly, Mar., 2001.

김 병 룡



e-mail : doolyn@super.inha.ac.kr

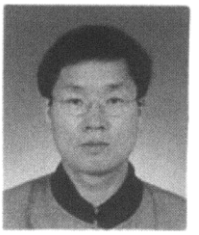
2000년 건양대학교 컴퓨터공학과(공학사)

2002년 인하대학교 전자계산공학과(공학석사)

2002년~현재 인하대학교 전자계산공학과 박사과정

관심분야 : 정보보호, 무선 프로토콜, P2P

김 기 창



e-mail : kchang@inha.ac.kr

1984년 California State Polytechnic University at Pomona, 전산학, 학사

1988년 University of California at Irvine, 전산학, 석사

1992년 University of California at Irvine, 전산학, 박사

1992년~1994년 IBM T.J. Watson 연구소, 연구원

1994년~현재 인하대학교, 정보통신공학부 부교수

관심분야 : 컴퓨터 시스템보안, 유무선 네트워크 보안, 실시간 운영체제, 병렬화 컴파일러