제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

순수 P2P 네트워크 환경 유지를 위한 피어 목록 관리 기법

김영진*, 엄영익*
*성균관대학교 정보통신공학부
e-mail:door21c@dslab.skku.ac.kr, yieom@ece.skku.ac.krr

Peer List Management Scheme for Pure P2P Network Environments

Young-jin Kim*, Young Ik Eom*
*School of Information and Communication Engineering,
Sungkyunkwan University

요 약

기존의 네트워크 환경에서 서비스 제공자들은 사용자들이 사용하고 있는 네트워크의 느린 속도와 가상 IP 사용으로 인해 빠른 대역폭과 대형 저장 공간, 그리고 고정 IP를 가진 서버를 운영함으로써, 사용자들에게 일방적으로 서비스를 제공하는 방식으로 발전해왔다. 그러나 인터넷 사용자의 폭발적인 증가와 함께 네트워크의 속도가 향상되어 각 사용자들은 서버를 경유하지 않고 사용자간 직접적인 파일 송수신을 할 수 있게 되었다. 각 단체에서는 이러한 P2P 환경의 장점을 이용하여 여러 P2P 응용 프로그램들을 개발하였지만, 네트워크 그룹에 참여하기 위하여 프로그램 내부에 서버의 IP가 지정되거나 사용자가 직접 IP를 입력해야 하기 때문에 서버가 더 이상의 서비스를 제공하지 않는다면 일반 클라이언트를 사용하는 사용자들은 서비스를 제공받지 못하는 문제점이 발생하게 되었다. 본 논문에서는 이러한 문제점을 해결하기 위해 순수 P2P 환경에서 각 피어들은 인접한 피어로부터 다른 피어들의 IP를 얻어 목록으로 관리함으로써, 소켓으로 연결된 피어가 종료하여도 목록에 유지된 다른 피어에게 연결하여 네트워크 그룹에 지속적으로 연결 유지를 할 수 있는 기법을 제안하고자 한다. 이 제안기법은 최초 실행시 기존의 방식과 같이 특정 피어의 IP를 지정해주지만, 이후의 동작과정에서는 어떠한 피어 IP도 지정하여 주지 않기 때문에 더욱 더 안정된 서비스를 제공받을 수 있다.

1. 서론

기존의 네트워크 환경에서 특정 서비스 제공자는 공용 IP를 사용하지 않거나 느린 네트워크 속도를 보유한 사용자들로 인해 빠른 대역폭과 대형 저장 공간을 가진 서버를 운영하여 서비스를 제공하는 클라이언트/서버 방식으로 발전해왔다. 그러나 인터넷 사용자의 폭발적인 증가와 함께 네트워크의 속도 또한 향상하였으며 고정 IP를 사용하는 호스트가 많아지면서 각 사용자들은 서버를 경유하지 않고 사용자간 직접적인 소켓 연결을 통해 파일의 송수신을 할 수 있게 되었다.

이에 따라 여러 단체에서는 서버에 네트워크 트래픽을 증가시키는 기존 클라이언트/서버 방식에서, 특정 서버에 자원이 있지 않고 각 사용자들이 공유한 파일 목록만을 관리하는 방식인 하이브리드 P2P(hybrid P2P) 방식으로 발전하게 되었다[1]. 그러나 이러한 방식은 여러 가지 사회적 문제로 인해[2] 순수 P2P(pure P2P) 방식으로 발전하게 되었다. 이러한 방식을 이용한 프로그램들은 네트워크 그룹에 참여하기 위하여 특정 프로그램 내부에 다른 피어의 IP가 지정되어 있거나 사용자가 직접 네트워크 그룹에 참여하고 있는 피어의 IP를 입력해야 한다. 이후 지정된 피어들이 종료된다면 네트워크 그룹에 참여할 수 없다는 문제가 발생한다[3].

이러한 문제점을 해결하기 위해 순수 P2P 환경에서 각 피어들은 인접한 피어로부터 피어들의 IP를 확보하여 목록으로 관리함으로써, 소켓으로 연결된 피어가 종료하여도 목록에 유지된 다른 피어에게 연결하여 네트워크 그룹

에 지속적으로 연결 유지를 할 수 있는 기법을 제안하고자 한다. 이 제안기법은 기존의 방식과 같이 최초 실행시 특정 피어의 IP를 지정해주지만, 이후의 동작과정에서는 어떠한 피어 IP를 지정하여 주지 않기 때문에 네트워크 그룹으로부터 지속적인 서비스를 제공받을 수 있다.

본 논문의 구성은 2장에서 기존의 P2P 네트워크 환경에 대한 관련연구를 소개한다. 3장에서는 순수 P2P 네트워크 환경 유지를 위한 피어 목록 관리 기법에 대한 개념을 기술하고, 4장은 제안 기법에 필요한 여러 가지 수치의 성능평가를 실시하며, 마지막으로 5장에서는 이에 대한 결론 및 향후 연구로 결론을 맺는다.

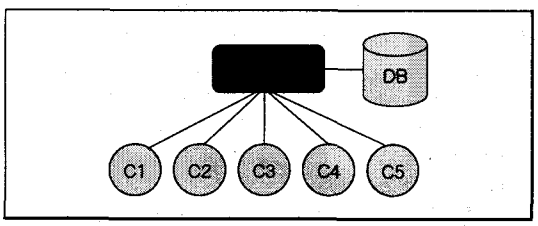
2. 관련연구

본 절에서는 현재 P2P 상에서 피어간 데이터를 송수신하기 위한 기존 방식들을 알아보도록 한다.

2.1 Napster 방식

이 방식은 하이브리드 P2P 방식으로, 각 클라이언트들은 서비스를 제공하는 특정 서버에 접속한 후 사용자가 지정한 공유 파일의 정보를 서버에 전송하게 되며, 이를 받은 서버는 각 클라이언트들의 파일 목록을 관리하게 된다. 이후 다른 사용자가 특정 파일을 찾기 위해 검색 메시지를 서버에 전송하면 서버는 클라이언트들이 보낸 파일 목록 중 사용자가 원하는 파일을 가지고 있는 위치와 파일명 등의 정보를 자료 요청한 피어에게 전달하게 되며, 이 메시지를 받은 클라이언트는 파일을 보유하고 있는 클라이언트에게 직접 연결을 함으로써 파일을 수신할 수 있게 된다.

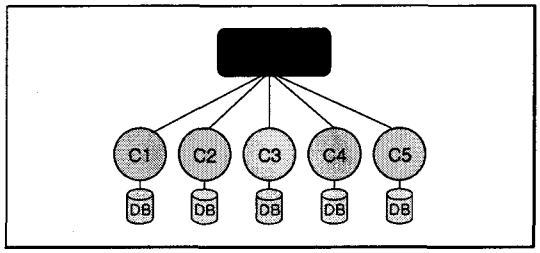
이러한 방법은 서버가 파일들의 목록만을 가지고 있기 때문에 기존 클라이언트/서버 구조에서 사용자들이 특정 파일을 서버에서 받아감으로써 발생하는 네트워크 트래픽을 일반 클라이언트로 분산시켜 자원을 빠르게 공유할 수 있다는 장점을 가지고 있다. 그러나 각 클라이언트는 서버와의 연결을 유지해야 하고 클라이언트가 공유하고 있는 파일 목록을 서버로 전송해야하기 때문에 서버는 클라이언트가 공유하고 있는 파일들의 목록 유지를 위한 파일 정보 메시지로 인해 네트워크 트래픽이 여전히 발생하게 된다. 또한 클라이언트들은 이러한 목록을 관리하는 서버가 종료되면 서비스를 제공받을 수 없다는 단점을 가지고 있다. 최근, Napster 방식은 불법적 파일을 중재하는 역할을 한다는 법적 책임을 안고 있다.



(그림 1) Napster 방식 네트워크 구조

2.2 소리바다 방식

이 방식 또한 하이브리드 P2P 방식이지만 기존의 Napster 방식과는 다르게 클라이언트들은 파일에 대한 어떠한 정보도 서버에게 보내지 않는다. 단지 일반 사용자들은 서버에 접속한 상태에서 파일 검색 메시지를 전송하면 서버는 현재 연결된 모든 사용자들에게 검색 요청 메시지를 전송하게 된다. 이 메시지를 받은 클라이언트 중 해당 파일을 가지고 있는 클라이언트는 파일 정보를 서버에게 전송하게 되며, 검색 요청에 대한 클라이언트의 응답을 받은 서버는 최초 자료 검색 요청을 한 클라이언트에게 응답 메시지를 전송하는 방식으로 진행된다[4]. 그러나 이러한 방식 또한 서버가 모든 요청과 답변을 중재함으로써 네트워크 트래픽이 여전히 발생하게 되며 기존의 방식처럼 서버 의존적인 연결 방식이기 때문에 서버가 종료되면 서비스를 제공받을 수 없게 된다. 또한, 서버가 파일 검색 요청과 응답을 중재하지 않으면 불법적인 파일 송수신을 할 수 없다는 이유 때문에 법적인 문제를 안고 있다.

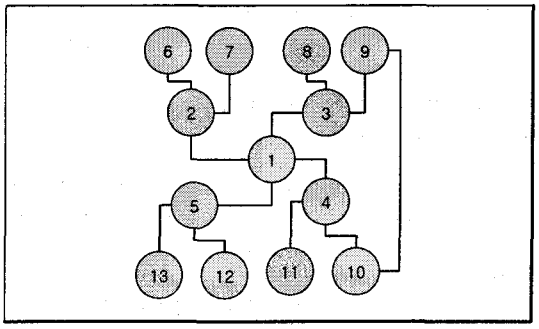


(그림 2) 소리바다 방식 네트워크 구조

2.3 Gnutella 방식

이 방식은 순수 P2P 방식으로 특정한 서버가 없으며, 모든 사용자들이 서버와 클라이언트의 기능을 동시에 가지는 방식이다[5].

Gnutella 방식을 이용하는 프로그램이 처음 실행되면 Gnutella 네트워크에 참여하고 있는 한 개 이상의 피어 IP를 입력해야 하며, 이러한 방식으로 모든 피어들이 피라미드식으로 연결되어 무제한의 자료를 공유하게 된다. 그러나 수많은 피어들이 Gnutella 네트워크에 참여하고 있기 때문에 자원 검색 속도의 저하를 일으키는 단점이 있다.



(그림 3) Gnutella 방식 네트워크 구조

3. 제안기법

본 절에서는 기존의 Gnutella 방식과 같은 순수 P2P 네트워크 환경에서 사용자가 입력한 IP 이외의 피어 IP들을 관리하여 주는 기법을 소개한다.

순수 P2P 방식을 응용한 프로그램이 처음 실행될 경우 사용자가 지정한 피어들의 IP에 접속을 시도하지만, 모두 실패로 돌아갈 경우 본 논문에서 제안하는 기법에 의해 유지된 IP로의 접속을 시도하도록 한다.

3.1 시스템 환경

본 기법은 모든 피어가 동일한 포트를 개방하고 있으며, 최초 실행시 사용자가 네트워크 그룹에 참여하고 있는 하나 이상의 피어 IP가 이미 입력되어 있다고 가정한다.

3.2 기본 동작

본 논문에서는 다른 피어에게 네트워크 소켓 연결을 시도하여 피어 IP들을 받는 IP 목록 수신 단계와 다른 피어로부터 수신된 IP를 이용하여 피어에 프로그램이 실행중인지 확인하는 피어 확인 단계로 나뉜다.

(1) IP 목록 수신 단계

다른 피어들의 IP에 대한 정보는 네트워크 그룹에 참여하고 있는 피어로 소켓 연결을 한 후 상대방 피어로부터 얻게 된다.

상대방으로부터 소켓 연결을 받은 피어가 처리해야 하는 과정은 알고리즘 1에서 보인다.

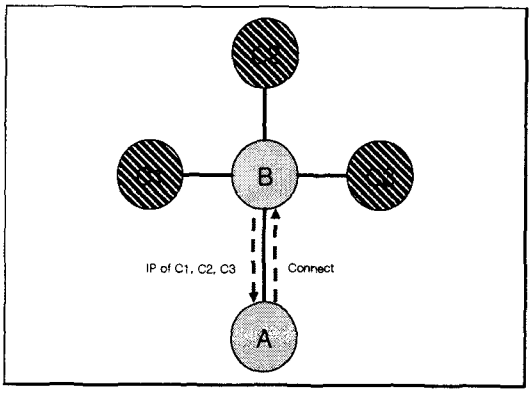
```
OnAccept()
{
    socket *as;
    as = new socket;
    Accept(*as);
    // 연결되어 있는 소켓들의 IP를 보내어준다.
    for ( int a=0; a<MasterSocket.GetConnectedSocketNumber(); a++) {
        socket h=MasterSocket.GetConnectedSocket(a);
        String message = h.GetIP();
        as.SendPacket(message);
    }
    if ( !ip_list.IsIP(as.GetIP()) )        ip_list.AddIP(as.GetIP());
    if ( MasterSocket.GetConnectedSocketNumber() >= MAX_SOCKET ) {
        disconnect as;
        delete as;
        return;
    }
    MasterSocket.AddSocket(as);
}
```

(알고리즘 1) Accept Event시의 알고리즘

각 피어들은 다른 피어에 의해 연결 수락이 발생되면, 이미 연결된 다른 피어들의 IP 목록을 방금 접속한 피어에게 전달하여 준다. 또한 많은 피어들이 한 피어로의 집중적인 연결을 막기 위해 네트워크 소켓 연결의 최대 수치를 넘어가지 못하도록 제한한다.

MAX_SOCKET은 네트워크 소켓의 연결이 이루어질 수 있는 최대 제한값이며, MIN_SOCKET는 기본적으로 연결이 되어 있어야 하는 최소 권장값이다.

그림 4는 알고리즘 1의 대략적 흐름을 보인다.



(그림 4) 피어의 최초 접속 시도 모습

그림 4에서와 같이, 피어 A가 피어 B에게 연결 시도하면 피어 B는 자신에게 연결되어 있는 피어 C1, C2, C3의 IP 정보를 피어 A에게 전송하여 준다.

(2) 피어 확인 단계

본 단계는 IP 목록 수신단계에서 구축한 IP 목록들 중 일정 시간동안 오프라인 상에 있는 피어를 구별하는 단계이다.

저장되어 있는 IP들은 표 1과 같은 여러 가지 필드와 함께 저장된다.

<표 1> IP 테이블 정보

명 칭	설 명
IP	피어 IP
NumSuccess	연결 성공한 수
NumFail	연속 접속 실패한 수
NumAttempt	연결 시도한 수
BitFW	BitFW
BitVal	Valid

각 피어들은 IP 테이블 내에 있는 IP에게 주기적으로 소켓 연결 시도를 하게 된다. 연결 시도를 할 때마다 NumAttempt를 1씩 증가시키며, 연결이 성공하면 NumFail은 0으로 세팅하고 NumSuccess를 1 증가시킨다. 그러나 연결에 실패하면 NumFail을 1만큼 증가시키고 연결 시도하였던 IP가 이미 연결되어 있다면 BitFW 필드를 TRUE로 체크하여 상대방이 방화벽/NAT 환경 내에 있다는 표시를 한다.

본 기법에서는 이러한 연결 시도 후 특정 IP 필드의 연속 접속 실패한 수가 특정 수치에 도달하게 되면, IP 테이블 내의 해당 IP BitVal 필드를 TRUE로 체크하여 해당 IP를 사용하는 피어는 더 이상 실행되지 않는 것으로 간주하게 되며, 차후 피어가 네트워크 그룹에 참여하기 위해 접속을 시도하는 IP에서 제외한다.

4. 비교 분석

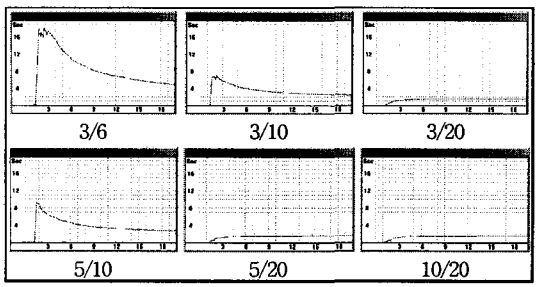
본 시뮬레이션은 24시간동안 다른 피어로 접속하여 네트워크 그룹에 참여하는데 걸리는 시간을 측정하였으며,

표 2의 수치를 기반으로 한 시뮬레이션 결과물은 그림 5에서 보이며, 표 3의 결과물은 그림 6에서 보인다.

<표 2> 시뮬레이션 수치

피어 IP 상태	개 수
방화벽/ NAT 환경 내에 있는 피어 수	10
일반 고정 IP를 사용하고 있는 피어 수	30
유동 IP를 보유하고 있는 피어 수	10

피어 실행 시간	백분율
1시간~2시간	30%
30분~1시간	30%
10분~30분	40%



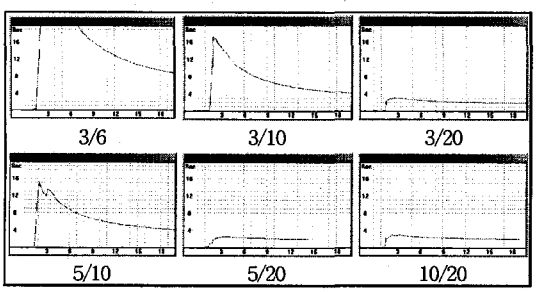
(그림 5) MIN_SOCKET/MAX_SOCKET의 변화에 따른 시간당 접속 연결 성공까지 걸리는 시간

위의 시뮬레이션 결과에서 보듯이 MIN_SOCKET과 MAX_SOCKET의 수치는 피어가 네트워크 그룹에 참여하는 시간에 영향을 준다.

<표 3> 시뮬레이션 수치

피어 IP 상태	개 수
방화벽/ NAT 환경 내에 있는 피어 수	20
일반 고정 IP를 사용하고 있는 피어 수	60
유동 IP를 보유하고 있는 피어 수	20

피어 실행 시간	백분율
1시간~2시간	30%
30분~1시간	30%
10분~30분	40%



(그림 6) MIN_SOCKET/MAX_SOCKET의 변화에 따른 시간당 접속 연결 성공까지 걸리는 시간

차후 프로그램이 실행되고 있는 피어가 유동 IP의 사용 및 방화벽/NAT 환경 내에 있는지 파악하여 순수 고정 IP인 사용자들의 목록만을 관리하게 한다면 더욱 더 빠른 접속 성공률을 보일 것으로 예상한다.

5. 결론 및 향후 연구과제

본 논문에서는 순수 P2P 환경에서 인접한 피어로부터 얻은 피어들의 IP를 확보한 후 목록으로 관리함으로써 연결되어 있는 피어가 종료하여 소켓 연결이 끊어져도 다른 피어에게 새로운 소켓 연결을 할 수 있는 방법을 제안하였다. 기존의 순수 P2P 프로그램들은 서버와 클라이언트의 기능을 모두 할 수 있다는 점에서는 순수 P2P 환경을 구현하였다고 할 수 있지만, 사용자로부터 입력된 IP를 가진 피어들이 종료될 경우 IP를 다시 입력하지 않는 이상 네트워크 그룹에 참여할 수 없는 문제를 앉고 있다. 따라서 이 기법은 다양한 피어들의 IP 목록을 구축하여 관리함으로써, 연결되어 있는 피어가 종료하여도 다른 피어에게 연결을 하여 네트워크 그룹에 지속적으로 참여할 수 있게 한다.

차후 에이전트 플랫폼(agent platform)에서 에이전트를 이주(migration)시키기 위한 기본적인 플랫폼 검색에 사용될 수 있으며, 여러 가지 순수 P2P 응용 프로그램에 적용하여 네트워크 그룹에 항상 연결할 수 있도록 유지하는 곳에 적용될 수 있다.

참고문헌

[1] A. Oram, "Peer-To-Peer", O'Reilly, Mar. 2001
[2] 휴맥스, "디지털기술의 발전과 지적재산권", 함께하는 시민행동, 2001
[3] JXTA, "Project JXTA Virtual Network," Sun Micro systems, Inc., Feb. 2002
[4] D. C. Hyde, "How New Peer to Peer Developments May Effect Collaborative Systems," Department of Computer Science Bucknell University, Jan. 2002
[5] CLIP2, "The Gnutella Protocol Specification v0.4", www.clip2.com