순수 P2P 네트워크 환경에서 프락시-서버 할당을 위한 동적 피어 선정 기법 설계 및 구현

김 영 진[†] · 김 문 정[†] · 김 응 모[††] · 엄 영 익[††]

요 약

현재 네트워크 보안을 위한 방화벽과 공용 IP 주소 부족 문제를 해결하기 위한 NAT의 사용이 일반화되고 있다. 그러나 이러한 환경에서는 기본적으로 서로 다른 방화벽/NAT 내에 있는 사용자들 간의 상호 연결이 불가능하기 때문에 제한된 서비스들만이 제공되어질 가능성이 있다. 이러한 문제점은 서로 다른 피어들간에 직접 연결을 하여 서비스를 주고받는 순수 P2P 환경에서는 각종 서비스의 제공에 큰 방해 요소로 작용할 수 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 P2P 환경에서 방화벽/NAT 내에 있지 않은 피어들 중 네트워크 트래픽이 적은 피어를 동적으로 선정하여 프락시-서버를 할당한 후 이를 피어들의 통신 매개체로 사용하는 기법을 제안한다. 제안 기법은 운영체제 독립적이고, 양방향 통신을 가능하게 해주며, 기존의 클라이언트-서버 환경에서 프락시-서버로 집중될 수 있는 네트워크 트래픽을 여러 피어들에게 분산시킬 수 있는 장점을 가지고 있다.

Design and Implementation of Dynamic Peer Selection Scheme for Allocating Proxy-Server on Pure P2P Network Environments

Young-jin Kim[†] · Moon Jeong Kim[†] · Ung Mo Kim[††] · Young Ik Eom[††]

ABSTRACT

Recently, deployments of firewalls and NATs are increasing to provide network security features or to solve the problem of public IP shortage. But, in these environments, peers in different firewall or NAT environments may get limited services because they cannot open direct communicate channels. This can be a significant problem in pure P2P environments where the peers should get or provide services by opening direct channels among themselves. In this paper, we propose a scheme for dynamically selecting a peer that can be used as a proxy server. The proxy server supports the communication between the peers in different firewall or NAT environments. The proposed scheme is operating system independent and supports bidirectional communication among the peers in P2P environments. Additionally, the proposed scheme can distribute network traffic by dynamically allocating proxy servers to the peers that is not located in the firewall or NAT environments.

키워드 : P2P 네트워크(P2P Network), 방화벽(Firewall), NAT, 프락시-서버 할당(Proxy-server allocation)

1. 서 론

기존의 클라이언트-서버(client-server) 환경이나 웹 환경에서는 각 사용자들의 자료를 서버에 저장하여 관리하고, 다른 사용자들이 해당 서버로부터 자료를 다운로드하는 형태의 중앙 집중식 시스템이 구성되어 왔다. 이러한 환경에서는 많은 사용자들의 네트워크 트래픽(network traffic)이 서버로 집중되면서 전체적인 네트워크 속도 저하가 발생하게 되었으며, 서비스 제공자가 클라이언트들이 접속할 서버를 제공하지 않으면 사용자들은 더 이상의 서비스를 제공받을 수 없게되는 문제점을 낳게 되었다. 그러나 최근 들어 개인 컴퓨터의 성능이 향상되고 네트워크의 속도가 증가되면서 이러한 문제점을 해결하기 위한 방법으로 P2P(Peer-to-Peer) 네트워크 모델이 제시되었다[1].

한편, 많은 기업에서는 내부 네트워크의 보호를 위한 보안상의 이유로 방화벽(firewall)을 사용하고 있으며, 부족한 공용 IP 주소 문제를 해결하기 위해 NAT(Network Address Translator)를 사용하고 있다. 이러한 이유로 방화벽과 NAT의 사용은 점차 일반화되고 있지만 방화벽에서는 외부 피어(peer)로부터 내부 피어로의 접속이 불가능하기 때문에 각 피어간 상호 연결이 실패하는 경우가 발생할 수 있다. 또한, NAT 환경에서는 NAT 환경 내의 각 피어들이 NAT 환경 외의 피어와 통신할 때에는 NAT를 운영하는 시스템 주소를 사용하게 되므로, NAT 환경 외에 있는 피어는 NAT 시스템과 통신을 하는 것으로 인지하여 NAT 환경 내에 존재

※ 본 논문은 한국과학재단이 지정한 지역협력연구센터(RRC)인 충남대학교 소프트웨어연구센터의 지원으로 수행된 2002년도 과제의 결과입니다.
† 준 회 원 : 성균관대학교 대학원 정보통신공학부
†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2002년 12월 3일, 심사완료 : 2002년 12월 27일

하는 특정 피어까지 연결을 시도할 수 없게 된다[2-4].

이러한 문제점들을 해결하기 위해 서로 다른 방화벽/NAT 환경 내에 있는 각 피어들간의 연결을 성공적으로 수립하려는 연구가 활발히 진행되고 있다. 그러나 이러한 기법들 중에서, HTTP(Hyper Text Transfer Protocol) 포트 연결 기법은 방화벽 관리자가 개방한 포트를 다른 응용 프로그램이 사용하지 말아야 한다는 단점을 가지며, 3-way-handshake 기법은 특정 운영체제에서만 실행이 가능하며 연결에 필요한 IP와 포트를 관리해 주는 특정 서버가 존재해야 한다는 단점을 갖고 있다. 또한, 프락시-서버(proxy-server)를 이용한 기법은 특정 서버에 부하를 준다는 단점을 갖는다[5-7].

본 논문에서는 순수 P2P 네트워크 환경을 기반으로 하여 상호간 통신을 하려는 두 피어 중 어느 하나라도 방화벽/NAT 환경 내에 존재하지 않는 경우에는 연결 방향을 조절하여 두 피어간 직접 연결을 시도하게 하며, 양쪽 피어 모두 서로 다른 방화벽/NAT 환경 내에 존재하는 경우에는 방화벽/NAT 환경 외의 피어들 중 네트워크 트래픽이 적은 피어를 선정하여 프락시-서버를 할당하고 이를 이용하여 두 피어간 간접 연결을 시도하게 한다. 따라서 본 논문에서는 서로 다른 방화벽/NAT 환경 내에 있는 피어들간에도 원활한 통신이 가능하도록 하는 기법을 제안한다. 또한 본 논문에서 제안하는 기법은 운영체제에 독립적이며, 상호 연결을 관리해 주는 별도의 서버를 필요로하지 않고, 동적으로 프락시-서버를 할당하게 되므로 특정 서버의 부하를 방지해 줌으로써 파일 전송, 화상 채팅, 멀티미디어 서비스 공유 등의 여러 서비스에 응용될 수 있다.

본 논문의 2장에서는 P2P 네트워크 환경에서 서로 다른 방화벽/NAT 내부 피어들간의 접속을 위한 기법들에 대해 소개한다. 3장에서는 P2P 네트워크 환경에서 프락시-서버 할당을 위한 피어를 동적으로 선정하는 기법에 대해 이의 동작 과정과 시나리오를 제시한다. 4장에서는 제안한 시스템의 운영결과를 보이고, 5장에서는 기존의 기법들과 성능 비교를 보인다. 마지막으로, 6장에서 결론 및 향후 과제를 살펴본다.

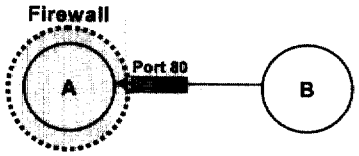
2. 관련 연구

본 절에서는 현재 P2P 네트워크 환경에서 방화벽/NAT 환경 외에 있는 피어로부터 내에 있는 피어로 접속을 시도하기 위한 기법들과 이 기법들이 갖는 제한점에 대해 소개한다.

2.1 HTTP 포트 연결 기법

방화벽을 사용하는 환경에서는 외부에게 특정 서비스를 제공하기 위하여 일정 포트를 개방하는 것이 일반적이다. HTTP 포트 연결 기법은 이러한 특징을 이용하여 가장 많이 개방하고 있는 웹 포트를 사용하는 기법이다.

방화벽 환경이 80번 포트로 들어오는 패킷을 막지 않을 경우, 방화벽 환경 외에 있는 피어는 포트 80번으로 연결 수락 대기상태로 있는 방화벽 환경 내의 피어에게 네트워크 소켓 연결을 할 수 있다. (그림 1)은 80번 포트를 이용하여 방화벽 환경 외에 있는 피어가 방화벽 환경 내에 있는 피어에게 네트워크 소켓 연결을 하는 모습을 보여준다.



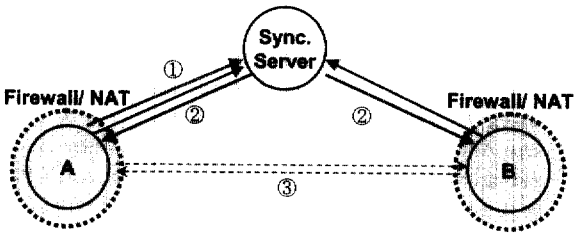
(그림 1) HTTP 포트를 이용한 연결 상태

그러나 이 기법은 방화벽을 사용하는 모든 시스템이 HTTP 포트를 개방하고 있다는 보장이 없으며, 80번 포트를 웹 서버 등의 다른 응용 프로그램에서 사용하고 있는 경우에는 적용시킬 수 없다. 또한 NAT의 경우에는 사설 IP를 사용하고 있는 여러 피어중 어느 피어에서 80번 포트를 개방하고 있는지 알 수 없으며[8], 또한 알고 있다 하여도 NAT 환경 내에 있는 모든 피어들이 80번 포트를 개방하면 80번 포트로 접속 요청을 받은 NAT 시스템은 어느 피어에게 해당 패킷을 전달할지 결정할 수 없기 때문에 NAT 환경에서는 사용될 수 없다[9, 10].

2.2 3-way-handshake 기법

서로 다른 방화벽/NAT 환경 내에 존재하는 두 피어가 서로에게 연결하려고 할 때 사용할 수 있는 기법으로, 방화벽/NAT 환경 내에 있는 피어가 외에 있는 피어로 연결이 가능한 것을 이용한 기법이다.

(그림 2)에서는 이 기법의 동작과정을 보인다.



(그림 2) 3-way-handshake 기법의 동작과정

이 기법은 피어 A가 피어 B에게 네트워크 소켓 연결을 위해 보내는 SYN 패킷이 피어 A가 속해 있는 방화벽 테이블에 통신하는 양쪽 피어의 IP와 포트 정보가 등록되는 것을 이용하는 기법으로, 이후 피어 B로부터 들어오는 SYN 패킷은 피어 A가 피어 B에게 접속 요청을 했던 패킷에 의해 테이블에 이미 등록이 되었기 때문에 방화벽에 차단되지 않고 피어 A까지 도달할 수 있게 된다.

그러나, 이 기법은 서로 다른 방화벽/NAT 환경 내에 존재하는 두 피어가 동시에 상대방에게 연결을 요청해야 하기 때문에 각 피어들이 동시에 접속할 수 있도록 관리해줄 수 있는 특정 서버에 항상 연결되어 있어야 하며, 이 서버는 접속된 각 피어가 다른 피어로부터 접속을 수락할 수 있는 IP와 포트 정보를 테이블로 유지하고 있어야한다. 또한 이 기법은 소스가 공개되어 있는 리눅스 등의 운영체제에서는 구현 가능하지만, 많은 사용자 층을 가지고 있고 P2P 기반 프

로그램이 실질적으로 동작할 마이크로소프트의 윈도우 등과 같은 운영체제에서는 실행이 불가능하며 통신하고자 하는 각 피어들은 상대방 피어에게 동시에 연결을 시도해야 한다는 어려움을 갖고 있다[5, 7].

2.3 프락시-서버 기법

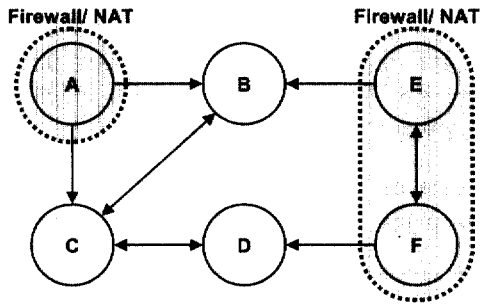
프락시-서버 기법은 하이브리드 P2P(hybrid P2P)에서 사용되는 기법으로, 방화벽/NAT 환경 외의 피어가 방화벽/NAT 환경 내에 있는 피어로 연결을 시도할 경우 프락시-서버를 사용하여 통신할 수 있도록 제공하는 기법이다.

그러나 이 기법은 서로 다른 방화벽/NAT 환경 내에 있는 피어와 통신할 수 있도록 하기 위한 임시방편으로, 방화벽/NAT 환경이 일반화 될수록 프락시-서버가 할당되는 호스트의 네트워크 트래픽을 증가시키게 되어 병목현상을 초래하게 된다는 단점을 갖고 있다.

3. 제안 기법

3.1 시스템 환경

본 논문에서 제안하는 기법은 순수 P2P 네트워크 환경을 기반으로 하며 각 피어는 자신이 방화벽/NAT 환경 내에 존재하는지의 여부를 알고 있다고 가정한다. 또한 각 피어들은 순수 P2P로 연결된 다른 피어들의 존재를 알지 못한다고 가정한다. 본 논문에서 제안하는 기법에 적용될 네트워크 구성은 (그림 3)과 같다.

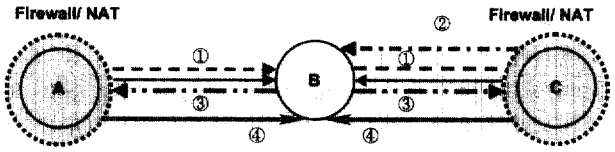


(그림 3) 네트워크 구성

(그림 3)에서 화살표 방향은 연결 시도 가능한 네트워크 소켓 연결 방향을 나타낸다. 즉, 방화벽/NAT 환경 외의 피어는 방화벽/NAT 환경 내의 피어로 연결을 시도할 수 없음을 보인다. 그리고 피어 A와 피어 E, F는 각각 서로 다른 방화벽/NAT 환경 내에 존재하는 피어들이며, 피어 B, C, D는 방화벽/NAT 환경이 아닌 일반적인 네트워크 환경에 존재하는 피어들이다.

3.2 기본 동작

본 논문에서의 제안 기법은 연결 요청 단계, 프락시-서버 요청 단계, 연결 응답 단계, 그리고 연결 확인 단계로 구성되며, (그림 4)는 이 제안 기법의 4단계 동작과정을 단계별로 보인다.



(그림 4) 동작과정

연결 요청 단계에서는, 연결 요청 피어(A)가 목적지 피어를 찾기 위해 이웃하는 피어(B)에게 연결 요청 메시지(CQP)를 전송하고, CQP를 받는 피어 B는 목적지 피어가 아니므로 CQP에 피어 B의 이름을 추가하여 이웃하는 다른 피어(C)에게 전달한다. 프락시-서버 요청 단계에서는, CQP를 받은 목적지 피어(C)는 연결 요청 피어와 목적지 피어 모두 서로 다른 방화벽/NAT 환경 내에 있는 것을 확인하고, 방화벽/NAT 환경 내에 있지 않은 피어 중 목적지 피어에 가장 가까운 피어를 선택하게 되며, CQP를 보낸 피어 B에게 선택된 피어를 목적지로 하는 프락시-서버 요청 메시지(PQP)를 보내게 된다. 연결 응답 단계에서는, PQP를 받은 피어 B가 프락시-서버를 할당한 후, 피어 A와 C에게 프락시-서버로 연결을 할 수 있도록 IP와 포트를 담은 연결 응답 메시지(CPP)를 보낸다. 연결 확인 단계에서는, 피어 A와 C가 프락시-서버에게 연결을 한 후 연결 확인 메시지(CFP)를 보내게 되며 통신 연결이 완료된다.

본 논문의 제안 기법에서 프락시-서버는 서로 다른 방화벽/NAT 환경 내의 피어들 간에 통신을 할 수 있도록 지원하기 위해 방화벽/NAT 환경 외의 피어들에게 할당되는 것으로, 목적지 피어는 연결 요청 피어까지 연결되어 있는 피어들중 네트워크 트래픽이 적은 피어를 선택하여 프락시-서버 할당 요청을 하게 된다. 본 논문의 제안기법에서 사용하고 있는 프락시-서버는 연결을 원하는 두 피어 사이에서 데이터를 전달해주는 작업을 수행하게 된다.

3.3 패킷 형식(packet format)

본 논문에서 제안하는 기법은 각 단계별로 연결 요청 패킷(CQP), 프락시-서버 요청 패킷(PQP), 연결 응답 패킷(CPP), 그리고 연결 확인 패킷(CFP)을 사용하며, 그 형식은 (그림 5)와 같다.

Type	Seq.	SouPeer	DesPeer	ChkBits	PktSize	RefPeer	IP	ServPort

(a) CQP(Connect reQuest Packet)

Type	Seq.	SouPeer	DesPeer	FinalPeer	PktSize	RefPeer

(b) PQP(Proxy-server reQuest Packet)

Type	Seq.	SouPeer	DesPeer	FinalPeer	PktSize	RefPeer	IP	ServPort

(c) CPP(Connect rePly Packet)

Type	Seq.	PeerName

(d) CFP(Connect conFirm Packet)

(그림 5) 패킷 형식

(그림 5)에서 보이는 각 패킷들은 필드 Type으로 메시지의 형식이 구분되며, 필드 Seq는 패킷의 중복 처리를 방지하기위해 사용된다. 필드 SouPeer와 DesPeer에는 연결 요청 피어와 목적지 피어의 이름이 저장된다. 패킷 CQP의 필드 RefPeer에는 CQP가 지나가는 중간 피어들의 이름이 추가되며, 필드 PktSize는 필드 RefPeer에 추가된 피어들의 개수가 기록된다. 필드 ChkBits에는 SouPeer와 필드 RefPeer의 각 피어가 방화벽/NAT 환경 내에 존재하는지의 여부를 나타내는 값이 저장된다. 필드 FinalPeer는 해당 패킷의 최종 도착 피어의 이름이 들어간다. 즉, 패킷 PQP에서 필드 FinalPeer는 프락시-서버 할당을 요청받는 피어의 이름이 들어가게 되고, 패킷 CPP의 필드 FinalPeer에는 연결 요청 피어 또는 목적지 피어의 이름이 들어가개 된다.

3.4 알고리즘(algorithm)

본 논문에서 제안하는 기법은 임의의 피어가 연결을 시도하는 연결 요청 단계, 연결 요청피어와 목적지 피어가 서로 다른 방화벽/NAT 환경 내에 존재할 경우 방화벽/NAT 환경 외에 존재하는 인접된 피어에게 프락시-서버를 할당하기 위해 요청하는 프락시-서버 요청 단계, 목적지 피어나 프락시-서버에 의해 연결 응답을 받는 연결 응답 단계, 그리고 연결 요청 피어와 목적지 피어가 프락시-서버로 연결되었음을 확인하기 위한 연결 확인 단계로 구성되며, 본 절에서는 각 단계별로 세부 알고리즘을 설명한다.

3.4.1 연결 요청 단계

순수 P2P 네트워크 환경에서 연결 요청 피어는 목적지 피어와 연결하기 위하여 CQP를 생성하여 요청하게 된다. (알고리즘 1)은 CQP를 생성하는 알고리즘이다.

```
input : other peer
output : none
{
    static int seq = 0 ;
    message = type (CQP) + Seq (seq) + SouPeer (myPeerName)
              + DesPeer (other peer name) ;
    if (this.FirewallCheckBit == FALSE) {
      _port = create port ;
      message += IP (this.IP) + ServPort (_port) ;
    }
    send message to all peer that is connected ;
}
```

(알고리즘 1) CQP 생성 절차

CQP를 만들어 보내는 연결 요청 피어가 방화벽/NAT 환경 외에 존재한다면 목적지 피어가 연결을 할 수 있는 IP와 포트를 메시지에 추가한 후 전송한다. 그러나 방화벽/NAT 환경 내에 존재한다면 IP와 포트는 추가하지 않는다.

CQP를 수신한 피어의 작업 절차는 (알고리즘 2)와 같다.

```
input : Seq, SouPeer, DesPeer, FinalPeer, RefPeer (, IP, ServPort)
output : socket handle
{
    if (history.CQP.SouPeer.Seq == Seq)   return ;
    history.add(CQP, Seq, SouPeer) ;
    if (DesPeer == myPeerName) {
      if ( SouPeer.FirewallCheckBit == FALSE) {
      connect IP, ServPort ;
      return socket handle ;
    }
    if (this.FirewallCheckBit == TRUE) {
      while (_peer = (get last peer of RefPeer)) {
        if (_peer.FirewallCheckBit == FALSE) {
          make PQP message ;
          send message to next peer within RefPeer field ;
          return ;
        }
        remove last peer of RefPeer ;
      } // end while
      } // else
      create service port ;
      make CPP message and send it to previous peer ;
      return ;
    }
    message.RefPeer += this.PeerName ;
    send message to all peer except previous peer ;
}
```

(알고리즘 2) CQP를 수신한 피어의 작업 절차

CQP를 수신한 피어는 다른 피어에서 들어올 수 있는 동일한 메시지를 중복 처리하지 않기 위해 캐쉬를 사용하여 확인 작업을 거친다.

CQP를 처리하는 피어가 최종 목적지이며 연결 요청 피어와 목적지 피어 모두 서로 다른 방화벽/NAT 환경 내에 있을 경우 목적지 피어는 프락시-서버 할당을 요청하기 위해 RefPeer에 방화벽/NAT 환경 내에 있지 않은 피어들 중 목적지 피어와 가장 가까운 피어를 선택하여 PQP를 전송한다. 그러나 연결 요청 피어가 방화벽/NAT 환경 내에 있지 않는 경우에는 직접 연결을 시도하며 목적지 피어만이 방화벽/NAT 환경 외에 있을 경우에는 연결 요청 피어가 접속할 수 있도록 IP와 포트를 담은 CPP를 전송한다.

3.4.2 프락시-서버 요청 단계

PQP를 수신한 피어의 작업절차는 (알고리즘 3)과 같다.

```
input : Seq, SouPeer, DesPeer, FinalPeer, RefPeer
output : none
{
    if (FinalPeer == myPeerName) {
      create Proxy-Server ;
      make CPP for SouPeer and send it to SouPeer ;
      make CPP for DesPeer and send it to DesPeer ;
    } else send CPP to next peer within RefPeer field ;
}
```

(알고리즘 3) PQP를 수신한 피어의 작업절차

PQP를 받은 피어가 메시지 최종 도착 피어일 경우, 새로운 프락시-서버 할당 요청을 받은 피어는 네트워크 소켓 연결을 위한 IP와 포트를 포함한 CPP를 연결 요청 피어와 목적지 피어에게 전달한다.

3.4.3 연결 응답 단계

네트워크 소켓으로 연결을 할 IP와 포트가 포함되어 있는 CPP를 수신한 피어의 작업절차는 (알고리즘 4)와 같다.

CPP는 프락시-서버나 목적지 피어에 의해 생성되는 메시지로서, CPP를 수신한 피어가 메시지 최종 도착 피어일 경우 CPP에 있는 IP와 포트로 네트워크 소켓 연결을 수행한다.

```
input : Seq, SouPeer, DesPeer, FinalPeer, RefPeer, IP, ServPort
output : socket handle
{
   if (FinalPeer == myPeerName) {
      connect IP, Port ;
      make CFP and send it to IP ;
      return socket handle ;
   } else send CPP to next peer within RefPeer field ;
}
```

(알고리즘 4) CPP를 수신한 피어의 작업절차

3.4.4 연결 확인 단계

CPP의 메시지 최종 도착 피어는 CPP에 있는 IP와 포트로 네트워크 소켓 연결을 마친 후 소켓을 이용하여 CFP를 보냄으로써 연결 확인 단계를 거치게 된다. 이 단계를 거친 후부터는 연결이 완료된 상태이므로, 연결 요청피어와 목적지 피어는 통신을 할 수 있게 된다.

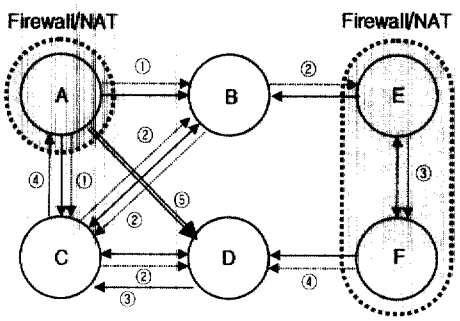
3.5 시나리오

본 논문에서 제안하는 기법의 시나리오는 연결을 요청하는 피어와 연결의 목적지 피어의 환경에 다음과 같이 분류될 수 있다.

3.5.1 연결 요청 피어만이 방화벽/NAT 환경 내에 존재하는 경우

CQP의 최종 목적지에서는 CQP에 연결 요청 피어의 IP와 포트가 포함되어 있지 않으므로 목적지 피어에서는 자신의 IP와 포트를 포함한 CPP를 연결 요청 피어에게 보내게 된다. CPP를 수신한 연결 요청 피어는 목적지 피어가 제공한 IP와 포트로 연결을 시도하여 서비스를 제공받게 된다.

(그림 6)에서는 피어 A가 피어 D에게 연결 요청을 하는 과정을 보인다.



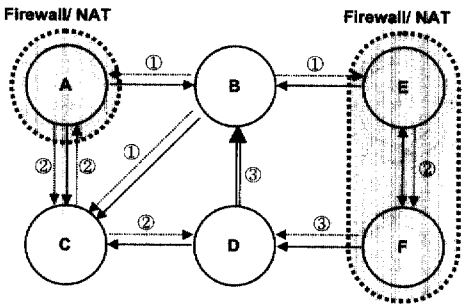
(그림 6) 연결 요청 피어만이 방화벽/NAT 환경일 경우

피어 A가 보낸 CQP는 각각의 피어를 거치게 된 후 피어 D에게 도달하게 되며, 피어 D는 연결 요청 피어만이 방화벽/NAT 환경 내에 있기 때문에 IP와 포트를 할당하여 다시 피어 A에게 CPP를 보낸다. 피어 A는 CPP를 받은 후 피어 D에게 네트워크 소켓 연결을 하여 통신할 수 있게 된다.

3.5.2 연결 요청 피어와 목적지 피어 모두 방화벽/NAT 환경 외에 있을 경우

연결 요청 피어는 방화벽/NAT 환경 내에 있지 않기 때문에 자신의 IP와 포트를 포함한 CQP를 전달하게 된다. 따라서 CQP를 수신한 목적지 피어는 CQP에 포함된 IP와 포트로 네트워크 소켓 연결을 하여 통신할 수 있다.

(그림 7)에서는 피어 B가 피어 D에게 연결 요청을 하는 과정을 보인다.



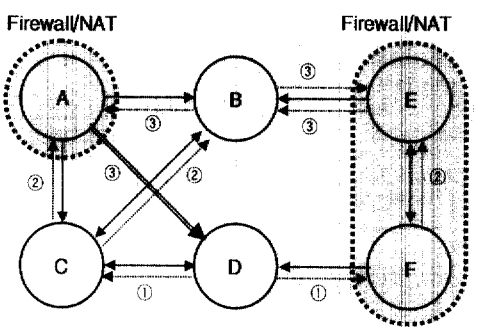
(그림 7) 연결 요청 피어와 목적지 피어 모두 방화벽/NAT 환경 내에 있지 않을 경우

피어 B는 방화벽/NAT 환경 내에 있지 않기 때문에 목적지 피어가 접속할 수 있는 연결 요청 피어의 IP와 포트를 담은 CQP를 전송한다. CQP를 받은 피어 D는 곧바로 피어 B에게 네트워크 소켓 연결을 하여 통신을 할 수 있다.

3.5.3 목적지 피어만이 방화벽/NAT 환경일 경우

목적지 피어만이 방화벽/NAT 환경 내에 있는 이 경우에도 시나리오 2와 마찬가지로, 목적지 피어는 연결 요청 피어가 방화벽/NAT 환경 내에 있지 않기 때문에 목적지 피어의 방화벽/NAT 환경 유무에 상관없이 연결 요청 피어에게 네트워크 소켓 연결을 하여 통신하게 된다. 따라서 목적지 피어는 연결 요청 피어가 방화벽/NAT 환경 내에 존재하지 않을 경우에는 목적지 피어의 방화벽/NAT 환경 유무에 상관없이 연결 요청 피어가 보낸 CQP의 IP와 포트로 네트워크 소켓 연결을 하게 된다.

(그림 8)에서는 피어 D가 피어 A에게 연결 요청을 하는 과정을 보인다.



(그림 8) 목적지만이 방화벽/NAT 환경일 경우

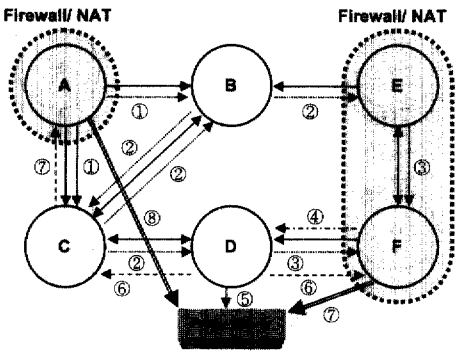
피어 D는 시나리오 2와 같이 방화벽/NAT 환경 내에 있지 않기 때문에 CQP에 IP와 포트를 담아 보낸다. 피어 A는

CQP의 IP와 포트를 이용해 피어 D에게 네트워크 소켓 연결을 할 수 있다.

3.5.4 연결 요청 피어와 목적지 피어가 서로 다른 방화벽/NAT 환경 내에 존재하는 경우

본 시나리오는 프락시-서버가 사용되어야 하는 경우로, 본 논문에서 제안하는 4단계 메시지를 모두 거쳐야 한다.

(그림 9)에서는 피어 A가 피어 F에게 연결 요청을 하는 과정을 보인다.



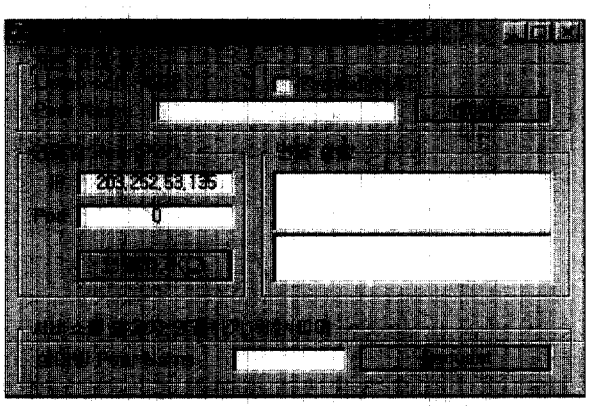
(그림 9) 연결 요청 피어와 목적지 피어 모두 방화벽/NAT 환경일 경우

통신하고자 하는 피어 A와 피어 F 모두 서로 다른 방화벽/NAT 환경 내에 존재하므로 어느 한쪽으로도 직접 연결을 할 수 없다. 따라서 피어 F는 피어 A와 통신을 위해 적당한 피어를 선정하고, 프락시-서버 할당을 요청하게 된다. 프락시-서버로 할당된 피어 D는 프락시-서버 모듈을 생성한 후 자신의 IP와 포트를 포함한 CPP를 연결 요청 피어와 목적지 피어에게 전달하게 된다. CPP를 받은 연결 요청 피어와 목적지 피어는 프락시-서버로 연결하여 패킷을 송수신할 수 있게 된다.

4. 구 현

4.1 구현 개요

본 논문에서 제안하는 기법을 테스트하기 위해 Windows 2000 플랫폼을 사용하였으며, 툴은 MS사의 Visual C++ 6.0을 사용하였다. 구현 프로그램의 화면 구성은 (그림 10)에서 보인다.



(그림 10) 테스트 프로그램의 화면

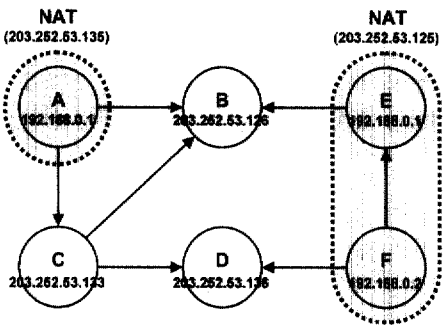
프로그램 화면을 구성하는 각 컨트롤들의 역할은 <표 1>에서 설명한다.

〈표 1〉 프로그램 화면 구성 요소

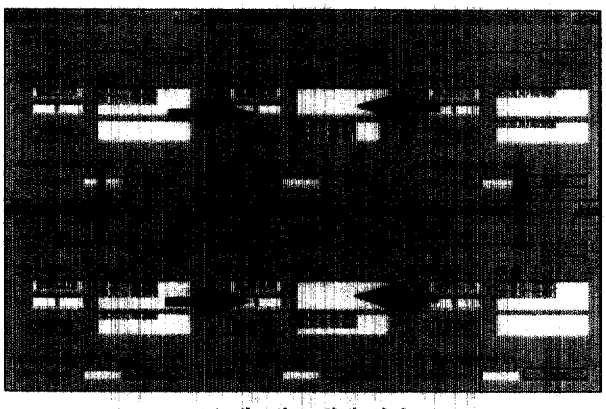
명 칭	내 용
Listen port Firewall/NAT Peer Name Initialize	다른 피어의 연결을 수락할 수 있도록 열어놓은 포트 해당 피어의 네트워크 환경 설정 해당 피어의 이름 설정 "개인 피어 정보"를 바탕으로 초기화하는 버튼
IP/Port 연결하기	연결하고자 하는 상대 피어의 IP/포트 연결하기 위한 버튼
상대방 Peer Name Request	서비스를 위해 연결하고자 하는 목적지 피어 이름 CQP를 생성하여 연결 시도를 하기 위한 버튼

테스트를 위한 네트워크 구성은 (그림 11)에서 보인다.

(그림 11)에서와 같이, 본 논문의 알고리즘이 적용될 네트워크 환경은 NAT 운영 호스트 2대, NAT를 이용하여 가상 IP를 사용하는 호스트 3대, 그리고 NAT 외부에 존재하는 호스트 3대로 구성된다. 이와 같이 네트워크 환경을 구성하기 위해 실행된 프로그램들은 (그림 12)에서 보이는 바와 같다.



(그림 11) 테스트를 위한 네트워크 구성도



(a) 네트워크 환경 설정

GEV	DUB	AJZ
DFG	JHX	WHF

(b) 프로그램별 피어 이름

(그림 12) 초기 P2P 네트워크 환경 설정

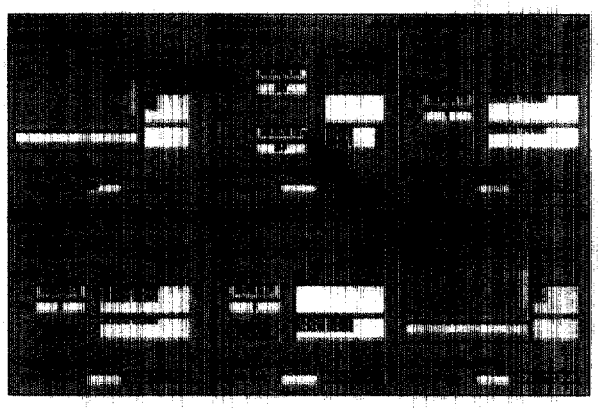
(그림 12)에서 화살표 방향은 네트워크 소켓 연결 방향을 나타내며, 방화벽/NAT 환경 내에 있는 피어는 방화벽/NAT

환경 외에 있는 피어이거나 같은 방화벽/NAT 환경 내에 있는 피어로 연결이 되어 있는 것을 볼 수 있다.

4.2 시나리오 실행 결과

본 절에서는 4가지의 시나리오를 테스트한 결과 중 연결 요청 피어와 목적지 피어 모두 서로 다른 방화벽/NAT 환경 내에 있을 경우에 목적지 피어가 방화벽/NAT 환경 내에 있지 않은 임의의 피어를 동적으로 선택하여 프락시-서버를 할당하게 되는 상황을 보인다.

(그림 13)은 피어 "GEV"가 피어 "WHF"에게 연결 요청을 하는 과정을 보인다.

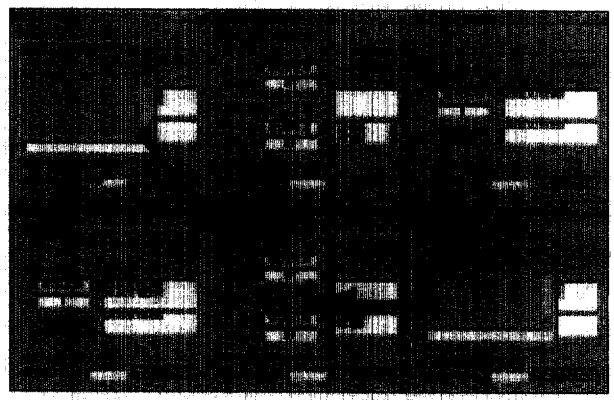


(그림 13) 프락시-서버를 통해 연결된 경우

(그림 13)에서, 피어 "WHF"는 피어 "GEV"가 보낸 CQP를 수신하였을 때 양쪽 피어 모두 방화벽/NAT 환경 내에 있다는 사실을 확인한 후 본 논문의 알고리즘에 의하여 피어 "DUB"가 프락시-서버 모듈을 실행하는 피어로 선택된다. 이후 피어 "GEV"와 피어 "WHF"는 피어 "DUB"가 생성한 프락시-서버 모듈을 이용하여 통신하게 된다.

(그림 14)에서는 이후 피어 "GEV"가 피어 "WHF"에게 다시 연결을 시도했을 때 네트워크 트래픽이 적은 또다른 피어에게 프락시-서버가 할당된 상황을 보인다.

(그림 13)에서, 프락시-서버가 할당된 피어 "DUB"는 이미 네트워크 트래픽이 다른 피어보다 증가하였기 때문에, 그림 14에서처럼 피어 "DUB"를 지나가는 CQP보다 피어 "JHX"를 지나가는 CQP가 최종 목적지에 더 빨리 도착하게 되는 결과가 나타나게 된다. 따라서 피어 "JHX"에 새로운 프락시-서버가 할당된 후 통신하게 된다.



(그림 14) 네트워크 트래픽에 의해 다른 피어가 프락시-서버로 선택된 경우

5. 성능 비교

기존에 제안된 기법들과 본 논문에서 제안하는 기법을 상대적으로 비교한 결과를 <표 2>에서 보인다.

<표 2>에서 HTTP 포트를 이용한 접속 기법은 다른 프로그램이 웹 포트를 사용할시 사용이 불가능하고, 3-way-handshake 기법은 많은 OS에서 사용이 불가능하다. 또한 서버 제공 프락시-서버는 네트워크 트래픽이 중앙에 집중되는 현상이 발생하며, 순수 P2P 프락시-서버 기법은 운영체제에 상관없이 실행 가능하고 특정 포트를 사용하지 않으며 순수 P2P 환경에서 빠른 피어에 프락시-서버가 할당되기 때문에 하나의 호스트에 프락시-서버를 할당하는 서버 제공 프락시-서버 기법보다 네트워크 트래픽을 분산 시킬 수 있음을 설명한다. 또한 방화벽/NAT 환경 외에 있는 피어들은 프락시-서버로 할당될 가능성이 있어 다른 피어를 위해 네트워크 트래픽이 증가될 수 있지만, 이러한 경우는 공유한 자료를 다른 피어에게 제공하여 생기는 네트워크 트래픽과 비교하였을 때, 두 경우 모두 다른 피어를 위해 네트워크 트래픽이 증가되며 제공하는 피어의 역할에 차이가 없기 때문에 동일하다고 본다.

이 기법을 사용함으로써 P2P 서비스를 받지 못하였던 방

<표 2> 방화벽/NAT 해결방법들의 성능 비교

	HTTP Protocol	3-way handshake	Server 제공 프락시-서버	순수 P2P 프락시-서버
동작 OS	모든 OS	(Redhat) Linux	모든 OS	모든 OS
상호 연결시도	양방향 연결 (설정 따라 단방향)	양방향 연결	양방향 연결	양방향 연결
네트워크 트래픽	낮 음	낮 음	높 음	중 간
장 점	방화벽 환경인 피어에 연결 가능	직접 연결이 가능	서비스 제공용이	모든 OS에서 가능 순수 P2P지원 가능 네트워크트래픽 분산
단 점	포트 충돌 가능 NAT 환경지원 불가능 방화벽 환경설정에 종속적임	동시 연결 곤란	프락시 서버의 부하가 큼	직접적으로 참여하지 않는 피어에 네트워크 트래픽 부여

화벽/NAT 환경 내에 있는 사용자들에게 파일송수신, 멀티미디어 서비스 등을 제공함으로써 보다 많은 사용자들에게 다양한 서비스를 제공할 수 있게 된다.

6. 결론 및 향후 연구과제

현재, 서로 다른 방화벽/NAT 환경 내에 있는 각 피어들간의 성공적인 연결을 위한 여러 기법들이 제안되고 있지만 순수 P2P 환경에서 각 피어들 간에 서비스를 주고받기 위해 완벽한 연결을 지원하지 못한다.

본 논문에서는 순수 P2P 환경에서 방화벽/NAT 환경 내에 있는 피어들간의 연결을 해결하기 위한 방법으로 이미 연결되어 있는 네트워크 연결망을 이용하여 프락시-서버를 사용하며, 이때 프락시-서버를 할당하기 위한 동적 피어 선정 기법을 제안하였다. 따라서 본 알고리즘은 연결 요청 피어와 목적지 피어 중 방화벽/NAT 환경 내에 있지 않은 피어가 있으면 그 피어에게 직접 연결을 하지만, 두 피어 모두 방화벽/NAT 환경 내에 있을 경우에는 프락시-서버를 이용하여 통신을 수행하게 된다.

기존에 제안된 기법들과 비교를 하였을 때 이 기법은 운영체제에 독립적이고, 방화벽/NAT 환경 유무에 상관없이 통신을 할 수 있으며, 어느 한 피어에게 네트워크 트래픽을 집중시키지 않는다는 장점을 가지고 있다.

향후 이 기법을 이동 에이전트의 자원 검색 알고리즘에 직접 적용하여, 직접 연결을 할 수 없어 서비스를 주고받지 못하였던 방화벽/NAT 환경 내의 사용자들에게 보다 나은 서비스를 제공할 수 있을 것으로 기대된다.

참 고 문 헌

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

[2] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," IETF, RFC 2663, Aug., 1999.

[3] B. Carpenter, "Internet Transparency," IETF, RFC 2775, Feb., 2000.

[4] N. Freed, "Behavior of and Requirements for Internet Firewalls," IETF, RFC 2979, Oct., 2000.

[5] P2PWG, "Bidirectional Peer-to-Peer Communication with Interposing Firewalls and NATs," NAT/Firewall Traversal White Paper, P2PWG, Aug., 2001.

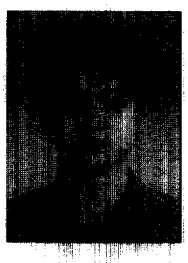
[6] D. C. Hyde, "How New Peer to Peer Developments May Effect Collaborative Systems," Third International Symposium on Collaborative Technologies and Systems, Jan., 2002.

[7] J. Postel, "Transmission Control Protocol," IETF, RFC 793, Sep., 1981.

[8] R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," IETF, RFC 1933, Apr., 1996.

[9] A. V. Anderson and S. H. Robinson, "Peer-to-Peer Network Communications in a World of NATs and Firewalls," An Intel Corporation White Paper, 2001.

[10] JXTA, "Project JXTA Virtual Network," Sun Microsystems, Inc., Feb., 2002.



김 영 진

e-mail : door21c@dslab.skku.ac.kr

2002년 대전대학교 컴퓨터공학과(학사)

2002년~현재 성균관대학교 대학원 정보통신공학부 석사과정

관심분야 : P2P, 이동 에이전트, 분산 시스템 등

김 문 정

e-mail : tops@ece.skku.ac.kr

1998년 성균관대학교 정보공학과(학사)

2000년 성균관대학교 대학원 전기전자 및 컴퓨터공학부(석사)

2002년 성균관대학교 대학원 정보통신공학부(박사)

관심분야 : 분산시스템, 이동 컴퓨팅, 애드-혹 네트워크, P2P, 이동 에이전트 등

김 응 모

e-mail : umkim@yurim.skku.ac.kr

1981년 성균관대학교 수학과(석사)

1986년 Old Dominion University 전산학과(석사)

1990년 Northwestern University 전산학과(박사)

현재 성균관대학교 정보통신공학부 교수

관심분야 : 웹 데이터베이스, 데이터베이스 보안, 데이터 마이닝, 데이터웨어하우징, 지능정보시스템, 이동 에이전트 등

엄 영 익

e-mail : yieom@ece.skku.ac.kr

1983년 서울대학교 계산통계학과(학사)

1985년 서울대학교 대학원 전산과학과(석사)

1991년 서울대학교 대학원 전산과학과(박사)

2000년~2001년 Dept. of Info. and Comm. Science at UCI 방문교수

현재 성균관대학교 정보통신공학부 교수

관심분야 : 분산시스템, 이동 컴퓨팅 시스템, 이동 에이전트, 시스템 소프트웨어 등