사설망을 지원하는 확장된 DONet 프로토콜

(An Extension of DONet Protocol to Support Private Networks)

이 문 수 † 한 성 민 †† 박 성 용 †††
(Moonsoo Lee) (Sungmin Han) (Sungyong Park)

요 약 오버레이 네트워크 기반 스트리밍 서비스는 확장성을 잃지 않으면서 효율적인 네트워크를 구성하기 어렵다. 이에 대해 제안된 여러 기법들 중 DONet은 대표적인 비구조적 스트리밍 오버레이 네트워크이다. 하지만 DONet은 NAT를 고려하지 않았기 때문에, 노드의 수가 많을수록 더 큰 성능을 기대할 수 있는 P2P의 특성을 100% 활용할 수 없다. Hole Punching은 NAT 내부의 노드들을 스트리밍 오버레이 네트워크에 참여시키는 대표적인 기법이다. 하지만, 한 대의 랑데부 서버를 사용하는 Hole Punching은 P2P 환경에서 확장성에 문제를 초래할 수 있다. 본 논문에서는 DONet에 다수의 노드들을 랑데부 서버로 이용할 수 있도록 설계한 Distributed Hole Punching 기법을 적용하여 확장성을 잃지 않고 높은 성능을 제공하는 DONet-P를 제안한다. 제안한 DONet-P의 성능을 측정한 결과 추가적으로 발생한 오버헤드는 크지 않고 높은 확장성을 가지며 기존의 DONet보다 더 좋은 데이타 연속성을 갖는다.

키워드 : 피어-투-피어, 스트리밍, 오버레이, 사설망

Abstract *It is difficult to construct streaming services based on the overlay networks without any loss of scalability. DONet is one of the most representative streaming overlay network protocols without managing any specific structure. Since DONet does not support the nodes on private networks, it can be considered that the performance of the overlay is not the best. Hole Punching is one of the famous techniques participating the nodes on private networks to streaming overlay networks by using a rendezvous server. However, using only a single rendezvous server cannot be suggested in P2P environment, because it can cause problems in terms of scalability and so on. In this paper, we propose DONet-P, an extension of DONet with Distributed Hole Punching techniques. It supports the nodes on private networks without loss of scalability. The experimental results show the better performance and scalability than DONet with a minimum overhead for additional control messages.*

Key words : Peer-to-Peer, streaming, overlay, private network

· 이 논문은 2007 한국컴퓨터종합학술대회에서 'DONet-P : 사설망을 지원하는 확장된 DONet 프로토콜'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : (주)매크로임팩트 부설연구소 HA개발팀 연구원
leemoonsoo@gmail.com
†† 정 회 원 : 서강대학교 컴퓨터학과
cuspy@sogang.ac.kr
††† 종신회원 : 서강대학교 컴퓨터학과 교수
parksy@sogang.ac.kr

논문접수 : 2007년 10월 2일
심사완료 : 2008년 5월 28일

정보과학회논문지 : 정보통신 제35권 제4호(2008.8)

1. 서 론

최근 스트리밍 서비스는 인터넷 트래픽의 많은 부분을 차지하는 인기 있는 서비스가 되었다. 인터넷 TV와 같이 하나의 데이타 소스를 다수의 사용자들에게 실시간으로 전송해야 하는 경우, 가장 효율적인 방법으로 IP 멀티캐스트를 생각할 수 있다. 하지만 IP 멀티캐스트는 이를 지원하지 않는 네트워크 장비나 ISP(Internet Service Provider)의 정책과 보안 문제로 인해 널리 이용되기는 힘들다. 이 때문에 이를 해결하기 위해 오버레이 네트워크를 만드는 어플리케이션 레벨의 멀티캐스트 방법이 제안되었다.

효율적인 오버레이 네트워크는 다음 사항들을 고려해

야 한다. 높은 확장성을 가져야 함은 물론이고, 노드들이 스트리밍 데이타를 끊이지 않고 받을 수 있도록 데이타 연속성을 보장해야한다.

초기 많은 오버레이 네트워크는 IP 멀티캐스트로부터 직접적으로 연상할 수 있는 트리구조를 형성해 데이타를 전송했다. ESM, NICE, ZigZag와 같은 시스템이 이에 해당한다. 하지만 노드의 출입이 빈번한 상황에서는 노드를 트리구조로 재구성하는데 많은 비용이 들게 된다. 이 후, DONet[1], PRO, 최신버전의 ESM과 같이 트리구조 같은 특별한 구조를 유지 하지 않는 스트리밍 오버레이 네트워크가 제안되었다. 이러한 네트워크는 노드의 출입 시 네트워크에서 트리와 같은 특별한 구조를 유지할 필요가 없기 때문에 노드의 잦은 출입에도 좀 더 유연하게 대처할 수 있다.

NAT(Network Address Translation)는 IP 주소 부족을 해결하기 위해 현재 인터넷에서 널리 사용되고 있다. 인터넷을 구성하는 노드들의 네트워크 환경과 지역 정보를 수집하는 illuminati[2]프로젝트에 따르면 전체 인터넷 노드의 73.81%에 해당하는 노드가 NAT를 이용하고 있다. NAT는 NAT안에 있는 노드로의 연결을 제한하기 때문에 더 좋은 스트리밍 오버레이 네트워크 구성을 위한 특정 노드로의 연결 시도를 실패하도록 만든다.

기존의 스트리밍 오버레이 네트워크는 NAT에 대한 고려가 없이 설계되어, NAT로 인해 실제로 연결할 수 있는 노드가 제한되어 원하는 형태의 네트워크를 구성할 수 없는 문제점이 있다. 따라서 NAT안에 있는 노드로 연결을 할 수 있는 방법을 스트리밍 오버레이 네트워크에 적용하여, 많은 NAT가 사용되고 있는 인터넷 환경에서도 높은 성능을 낼 수 있도록 하는 연구가 필요하다.

스트리밍 오버레이 네트워크에 NAT안에 있는 노드로의 연결을 가능하게 하는 기법[3]을 적용할 경우 NAT안의 노드로의 연결이 제한되는 문제를 해결할 수 있지만, 스트리밍 오버레이 네트워크에 기능을 추가함으로서 확장성에 문제가 생길 수 있다. 예를 들어 NAT안에 있는 노드로의 연결을 가능하게 하는 기법 중 하나인 Hole Punching[4]을 이용할 경우 모든 노드가 연결을 유지하고 있는 단 하나의 서버를 필요로 하기 때문에, 서버에 부하가 집중되어 확장성이 떨어지게 된다.

본 논문에서는 높은 확장성을 잃지 않으면서 NAT안에 있는 노드로의 연결을 가능하게 하는 기법을 기존의 스트리밍 오버레이 네트워크에 적용시키는 것에 초점을 맞춘다. NAT안에 있는 노드로의 연결을 가능하게 하는 기법을 적용한 후 스트리밍 오버레이 네트워크는 노드의 더 높은 데이타 연속성을 제공함은 물론, 노드의 출입이 빈번한 상황에서도 여전히 높은 확장성을 가지고 있어야 한다.

제안한 스트리밍 오버레이 네트워크의 검증을 위해서 기존의 대표적인 스트리밍 오버레이 네트워크인 DONet, DONet에 단순히 NAT안에 있는 노드로의 연결을 가능하게 하는 기법을 적용한 DONet-C, DONet에 NAT안에 있는 노드로의 연결을 가능하게 하는 분산화 된 기법을 적용한 DONet-P를 확장성과 데이타 연속성 등의 측면에서 비교했다.

본 논문의 구성은 다음과 같다. 1장에서는 본 연구의 배경이 되는 스트리밍 오버레이 네트워크와, 현재 NAT가 사용되는 인터넷환경에서 스트리밍 오버레이 네트워크의 문제점을 살펴본다. 2장에서는 스트리밍 오버레이 네트워크의 하나인 DONet을 살펴보고, NAT안에 있는 노드로의 연결을 가능하게 하는 기법과 기존 연구의 문제점에 대해 알아본다. 3장에서는 기존의 문제점을 해결하기 위해 제안된 DONet-P에 대해 알아본다. 4장에서는 기존의 DONet과, DONet-C, DONet-P를 확장성과 데이타 연속성 등의 측면에서 비교하여 성능 향상을 검증한다. 마지막으로 5장에서 논문의 결론과 향후 과제에 대해 논의한다.

2. 관련연구

본 장에서는 대표적인 스트리밍 네트워크 중 하나인 DONet과, NAT안에 있는 노드로의 연결을 만들 수 있는 기법을 살펴보고 기존 연구의 한계점을 분석한다.

2.1 DONet

DONet은 실시간 스트리밍을 위한 비구조적인 오버레이 네트워크이다. DONet을 구성하는 각각의 노드는 멤버쉽 매니저, 파트너쉽 매니저, 버퍼, 버퍼맵, 스케줄러로 구성되어있고 네트워크 인터페이스를 통해 다른 DONet의 모든 노드는 파트너의 요청에 따라 데이타조각을 제공하기도 하고, 파트너노드들과 메시지를 주고받는다. 멤버쉽 매니저는 오버레이 네트워크에 참여하고 있는 노드 일부에 대한 정보를 가지고 있고 이를 유지한다. 파트너쉽 매니저는 데이타를 주고받기위해 다른 노드와 연결하고 이를 관리하는 기능을 한다. 이때 파트너쉽 매니저에 의해 스트림 데이타를 주고받을 목적으로 연결된 노드를 파트너라고 한다. 데이타는 일정한 크기의 조각으로 나뉘어져 버퍼에 저장되고 전송된다. 또한 버퍼맵은 버퍼의 상태를 비트열로 표시한다. 스케줄러는 데이타 스트림 조각들을 각각 어느 노드로부터 전송 받을 것인지 결정한다. 소스노드를 제외한 에게 데이타 조각의 전송 요청을 보내기도 한다.

2.1.1 멤버쉽 프로토콜

• 노드의 참여

DONet에 참여하는 모든 노드는 유일한 식별자를 갖

는다. 모든 노드의 멤버쉽 매니저는 현재 네트워크에 참여하고 있는 일부 노드들의 정보를 유지하고 있다. 새로 네트워크에 참여하는 노드는 우선 소스노드에 접근한다. 소스노드는 자신의 역할을 대신할 노드를 멤버쉽 매니저로부터 선택해 새로 접근한 노드에게 알린다. 새로 참여한 노드는 네트워크에 대한 아무런 정보도 가지고 있지 않으므로, 소스노드의 대리노드에 연결하여 대리노드의 멤버쉽 매니저가 유지하고 있는 네트워크 일부에 대한 정보를 전송받는다. 새로 참여한 노드의 멤버쉽 매니저는 이제 네트워크 일부에 대한 정보를 유지하게 된다. 또 새로 참여한 노드는 자신의 멤버쉽 매니저로부터 파트너 후보를 골라 연결을 시도하게 된다.

• 노드의 탈퇴

DONet의 노드는 언제라도 네트워크에서 빠져나갈 수 있다. 노드는 자신이 종료된다는 메시지를 생성해 네트워크에 명시적으로 자신이 빠져나감을 알릴 수 있다. 종료 메시지를 보내지 않고 네트워크에서 빠져나간 경우에는 파트너들이 주기적으로 버퍼맵에 관한 정보를 주고받는 메시지가 일정시간동안 특정 파트너로부터 오지 않을 때, 해당노드가 네트워크에서 빠져나갔음을 알 수 있다.

• 멤버쉽 정보의 추가와 삭제

멤버쉽 매니저에 정보가 갱신되는 경우는 네트워크에 새로운 노드가 참여하거나 탈퇴했을 경우이다. 새로운 노드가 네트워크에 참여하면 새로운 노드는 이를 알리는 메시지를 보내게 된다. 이 메시지는 SCAM(Scalable Gossip Membership protocol)[5]에 의해 네트워크에 퍼지고 이를 수신한 노드는 필요에 따라 자신의 멤버쉽 매니저에 새로운 노드를 추가한다. 노드가 탈퇴한 경우, 탈퇴한 노드 자신 또는 탈퇴한 노드의 파트너는 해당 노드가 네트워크에서 빠져나감을 SCAM을 통해 네트워크에 알린다. 노드가 탈퇴했음을 알리는 SCAM메시지를 수신한 노드는 자신의 멤버쉽 매니저에서 해당 노드를 삭제한다.

2.1.2 파트너 선택

DONet에 참여하는 모든 노드는 데이타를 주고받기 위해 M개의 파트너를 가진다. 만약 현재 연결된 파트너의 개수가 M보다 작다면, 필요한 만큼의 노드를 멤버쉽 매니저로부터 무작위로 선택하여 연결을 시도한다. 또한 점차적으로 더 좋은 파트너를 가지기 위해서 일정 시간마다 새로운 파트너를 멤버쉽 매니저로부터 선택하여 연결한다. 새로운 파트너 연결이 성공하면 현재 가지고 있는 파트너 중 가장 대역폭이 작은 것을 골라 파트너에서 제외하여 시간이 지남에 따라 성능이 좋은 노드들이 모인 파트너 집합을 기대할 수 있다.

2.1.3 버퍼

DONet에서 전송되는 데이타 스트림은 같은 크기의 조각으로 나누어진다. 이 조각들을 파트너들끼리 주고받음으로서 네트워크 전체로 데이타를 전파시킨다. 각각의 노드는 해당 노드가 어떤 조각을 버퍼에 가지고 있는지를 표현하기 위해 버퍼맵을 사용한다. 버퍼맵에는 현재 자신의 버퍼에 준비가 된 조각과 준비되지 않은 조각이 무엇인지에 관한 정보가 있다. 버퍼맵 정보는 파트너들끼리 일정 시간마다 주고받는다. 모든 노드는 자신의 모든 파트너들의 버퍼맵 정보를 알고 있고, 스케줄러는 이를 바탕으로 어느 노드로부터 어느 데이타 조각을 받을지 결정한다.

2.1.4 스케줄러

스케줄러는 자신의 버퍼맵의 정보와 현재 연결된 파트너들의 버퍼맵 정보를 이용하여 어떤 파트너로부터 어떤 데이타 조각을 받아야 할지를 결정한다. DONet의 스케줄러는 데이타 조각의 재생을 위한 마감시간(Deadline)과 파트너의 대역폭(Bandwidth)을 고려하여 스케줄한다. 스케줄러는 데이타 조각을 데이타가 재생되어야하는 시점인 마감시간까지 전송받지 못할 경우를 최소화해야 한다. 각기 다른 대역폭을 갖는 파트너들로부터 마감시간이 있는 데이타 조각을 받아야 하는 문제는 NP-hard문제로 알려져 있다[6]. 따라서 DONet에서는 휴리스틱 알고리즘을 이용해 스케줄러를 작성하였다.

2.1.5 컨트롤 메시지 오버헤드

평균적으로 노드 하나가 받는 컨트롤 메시지 오버헤드는 멤버쉽 정보와 파트너의 유지를 위해 주고받는 메시지, Hole Punching을 위한 메시지, 데이타 전송을 위한 메시지의 합으로 나타낼 수 있다.

O_k를 어떤 노드가 k작업을 수행하는데 발생하는 메시지 오버헤드라고 할 때, O_j는 네트워크에 참여, O_l은 네트워크에서 탈퇴하는 노드가 발생시키는 메시지 오버헤드라고 정의한다. D_j를 단위시간동안에 네트워크에 참여하는 노드의 비율, D_l을 단위 시간동안에 네트워크에서 탈퇴하는 노드의 비율이라고 하면, 네트워크가 N개의 노드로 구성되어 있을 때 단위시간동안 멤버쉽 정보의 유지를 위해 주고받는 메시지 CM_{donet}은 식 (1)과 같다.

$$CM_{donet} = N \times (D_j \times O_j + D_l \times O_l) \quad (1)$$

DONet에서 네트워크에 새로 참여하는 노드는 M개의 파트너를 생성해야하고, 파트너 하나의 연결을 위해서는 O_p의 메시지 오버헤드가 발생한다. 또한 이미 연결된 파트너가 네트워크에서 떠날 때, 새로운 파트너를 찾아 연결하는 오버헤드가 추가된다. 따라서 식 (2)와 같이 파트너의 유지를 위한 메시지 오버헤드 CP_{donet}을 구할 수 있다.

$$CP_{donet} = N \times (D_j \times O_p \times M + D_l \times O_p) \quad (2)$$

데이타 전송을 위해서 각각의 파트너와 버퍼맵 정보를 주기적으로 주고받을 때 단위시간동안 O_b의 메시지 오버헤드가 발생한다. 모든 노드는 M개의 파트너를 유지하므로 식 (3)과 같이 데이타 전송을 위한 컨트롤 메시지 오버헤드 CD_{donet}을 구할 수 있다.

$$CD_{donet} = N \times M \times O_b \quad (3)$$

DONet의 노드 하나가 단위 시간동안 받는 평균 메시지 오버헤드 C_{donet}은 식 (1), 식 (2), 식 (3)의 합을 전체 네트워크에 참여하는 노드의 개수 N으로 나눈 식 (4)로 나타낼 수 있다.

$$C_{donet} = \frac{CM_{donet} + CP_{donet} + CD_{donet}}{N} \quad (4)$$

2.2 NAT안의 노드로 연결을 생성하는 기법

NAT안의 노드로 연결을 생성하는 방법으로는 수동으로 NAT의 매핑 테이블을 수정하여 어플리케이션이 외부로부터의 연결을 받아들일 수 있게 하는 방법, UPnP포럼[7]에서 제안한 프로토콜로서 UPnP 프로토콜을 사용하는 방법, 어플리케이션 레이어 게이트웨이, Hole Punching 등이 있다.

그중 Hole Punching은 NAT의 매핑테이블을 직접 수정하지 않고 패킷을 보내 NAT의 매핑 테이블이 원하는 대로 구성되도록 한 후, NAT에 내부 노드의 매핑된 포트 번호를 알아내어 내부 노드와 연결을 하는 방법이다.

Hole Punching을 위해서는 모든 노드들이 접근할 수 있는 하나의 랑데부 서버가 필요하고, NAT에 연결하기 원하는 노드와 NAT안에 있는 노드 모두 랑데부 서버와 연결을 유지 하고 있어야 한다. NAT안에 있는 노드는 랑데부 서버에 접속할 때 자기 자신의 내부 주소와 포트번호를 접속 요청 메시지에 실어 전송한다. 이를 받은 랑데부 서버는 패킷의 출발지 주소로부터, 해당 노드가 매핑된 NAT의 주소와 포트번호를, 패킷에 실려 있는 메시지로부터 해당노드의 내부 주소와 포트번호를 알아내 이를 저장한다. 또한 많은 경우 NAT는 일정 시간(약 20초) 동안 패킷의 전송이 없으면 매핑 테이블에서 해당 엔트리를 제거하기 때문에[4], 모든 노드는 NAT의 매핑 테이블을 유지하기 위해 랑데부 서버와 주기적으로 메시지를 주고받는다.

그림 1의 (a)는 서로 다른 NAT안에 있는 두 노드 A와 B가 랑데부 서버 S에 각각 연결을 가지고 있는 상태를 나타낸다. 따라서 랑데부 서버 S는 A와 B의 내부주소(A의 내부주소 192.168.1.1:2222, B의 내부주소 192.168.2.1:2222)와 외부주소(A의 외부주소 217.39.24.13:65000, B의 외부주소 65.12.98.23:65010)를 모두 알고 있는 상태이다. 그림 1의 (b)와 같이 A가 S에게 B로의 연결 요청을 한 경우 S는 A와 B에게 각각 B와 A의 내부 주소와 외부 주소를 전달한다. A와 B는 각각 B와 A의 내부주소와 외부주소로 패킷을 보내 연결을 시도한다. 이 경우 A에서 B의 내부주소로 보낸 패킷(192.168.1.1:2222에서 192.168.2.1:2222로)과 B가 A의 내부주소로 보낸 패킷(192.168.2.1:2222에서 192.168.2.1:2222로)은 도착하지 않는다. 하지만 A가 B의 외부주소로 보낸 패킷(192.168.1.1:2222에서 65.12.98.23:650101로)은 A의 NAT에 의해, A의 외부주소에서 B의 외부주소로 보내는 패킷(217.39.24.13:65000에서 65.12.98.23:65010로)으로 변환된다. B의 NAT가 Symmetric NAT가 아닌 경우, 이미 S로의 연결로 인해 매핑 테이블이 생성된 상태이므로 패킷을 A의 외부주소에서 B의 내부주소로 보내는 패킷(217.39.24.13:65000에서 192.168.2.1:2222)으로 변환시켜 B에 전달한다. B가 A의 외부 주소로 보내는 패킷도 마찬가지의 과정을 거쳐 A에 전달된다. 결과적으로 A와 B가 각각 서로의 외부 주소로 보낸 패킷이 도착해 A-B연결이 생성된다.

2.3 기존 연구의 문제점

DONet은 인터넷에서 널리 사용되고 있는 NAT를 고려하지 않아 네트워크 구성을 위해 연결해야 하는 노드가 NAT로 인해 제한됨으로서 이로 인해 스트리밍 네트워크의 성능 저하가 발생한다.

NAT안에 있는 노드로의 연결을 하는 기법 중 수동 설정은 내부노드의 주소가 바뀌거나, 어플리케이션이 이용하는 포트번호가 일정하지 않을 때, 내부노드의 개수가 많을 때, 사람이 일일이 손으로 매핑 테이블을 수정하기가 어렵다. 이를 해결할 수 있는 UPnP의 경우 모든 NAT가 아직 UPnP프로토콜을 지원하지 않는다는 문제점이 있다. Hole Punching은 하나의 랑데부 서버에 네트워크에 참여하고자 하는 모든 노드가 연결을 유지하고, 접속 요청을 하므로, 랑데부 서버 하나로 부하가 집중된다는 단점이 있다. 이는 스트리밍 오버레이 네트워크같이 대규모의 노드가 참여하는 경우 랑데부 서버가 오버레이 네트워크의 확장성을 저해하는 요인이 된다.

본 논문에서는 DONet을 확장하여 NAT장비의 교체나 새로운 NAT장비의 보급 없이 NAT안에 있는 노드와의 통신을 가능하게 하는 Hole Punching기능을 가지면서도 하나의 랑데부 서버에 부하가 집중되어 확장성이 떨어지는 문제점이 발생하지 않는 스트리밍 오버레이 네트워크 프로토콜인 DONet-P를 설계한다.

3. DONet-P

본 장에서는 분산 Hole Punching 기능을 가지는 스트리밍 오버레이 네트워크 프로토콜인 DONet-P에 대

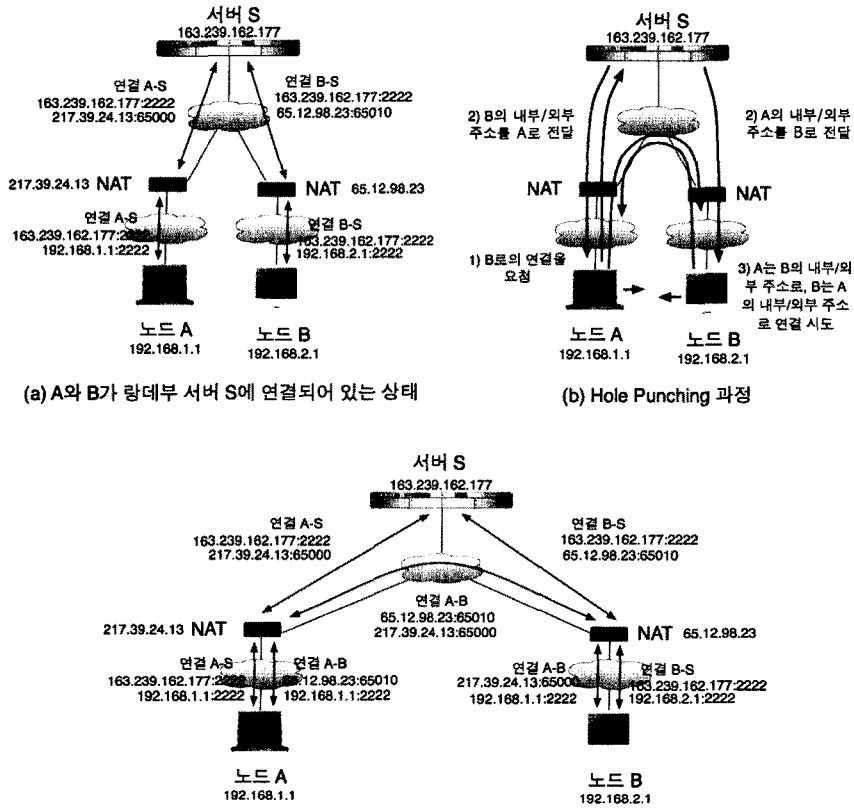


그림 1 서로 다른 NAT안에 있는 노드 A와 B의 Hole Punching

해 설명하고자 한다. 먼저 DONet-P의 설계 목표를 살펴보고, DONet과의 차이점을 살펴봄으로서 DONet-P의 생성 기법에 대해 설명하도록 하겠다.

3.1 설계 목표

본 논문에서 제시하는 DONet-P는 기존의 DONet에 NAT안에 있는 노드로의 연결 기법인 Hole Punching 기법을 적용한 스트리밍 오버레이 네트워크로, 크게 세 가지의 설계 목표를 가지고 있다.

첫째, 오버레이 네트워크를 생성하고 유지하는데 소요되는 비용을 최소화 하여야 한다. DONet-P는 기존의 DONet에 NAT안에 있는 노드로의 연결을 위한 Hole Punching기법을 추가하여 적용하였으므로 추가된 컨트롤 메시지의 오버헤드를 최소로 유지할 필요가 있다.

둘째, 데이타 연속성의 향상이다. DONet-P는 NAT안에 있는 노드와의 연결이 가능해지므로 더 좋은 오버레이 네트워크가 생성됨을 기대할 수 있다. 더 좋은 오버레이 네트워크는 노드 각각이 더 좋은 파트너를 가지고 있다고 말할 수 있다. 더 좋은 파트너는 빠르게 데이타를 전송할 수 있음을 뜻하므로 원하는 시간에 원하는 데이타가 준비되어있을 확률이 높아진다. 따라서 DONet-P는 기존의 DONet보다 더 높은 데이타 연속성을 가져야 한다.

셋째, 확장성의 유지이다. 기존의 DONet은 높은 확장성을 가지고 있지만 DONet-P는 Hole Punching 기법을 추가하였기 때문에 추가적인 오버헤드가 발생한다. 따라서 추가적인 오버헤드가 크다면 확장성이 떨어질 수 있다. 또한 Hole Punching 기법은 하나의 랑데부 서버에 모든 노드가 연결되어 있어야 하므로, 하나의 랑데부 서버에 부하가 집중되어 확장성이 떨어지게 된다. 그러므로 랑데부 서버가 받는 부하를 분산시켜 높은 확장성을 유지하도록 해야 한다.

3.2 개요

앞서 제시한 설계 목표를 달성하기 위해서는 최소한의 메시지만을 DONet에 추가하여 Hole Punching 기능을 가지도록 하고, Hole Punching을 위한 랑데부 서버의 역할을 스트리밍 오버레이 네트워크에 참여하는 노드에 골고루 분산시켜야 한다.

네트워크에 참여하는 모든 노드 집합을 N_{all}이라 할

때 N_{all}중에서 NAT안에 있는 노드의 집합을 N_{nat}, N_{all} 중에서 NAT를 사용하지 않는 노드의 집합을 N_{public}이라고 정의하면, 네트워크에 참여하는 노드의 집합은 $N_{all} = N_{public} \bigcup N_{nat}$의 관계가 성립한다. 이때 랑데부 서버의 역할은 NAT안에 있는 노드가 할 수 없으므로, N_{public}에 속한 노드들로 제한된다. 그림 1은 랑데부 서버와의 연결을 나타낸 그림이다. 그림 2에서와 같이 N_{public}에 속한 노드들이 랑데부 서버의 역할을 하고, N_{nat}에 속한 노드는 N_{public}에 속한 노드 중 하나의 노드를 선택해 자신의 랑데부 서버로 이용을 한다.

네트워크에 있는 모든 노드는 M개의 파트너 연결을 유지한다. 파트너는 스트리밍 데이타를 주고받을 수 있는 노드이다. 처음 노드가 네트워크에 참여하면 노드는 네트워크에 참여하고 있는 노드 중 M개를 무작위로 선택해 파트너 연결을 시도한다. 노드는 또한 M개의 파트너를 가지고 있다 하더라도, 일정 시간마다 새로운 파트너 연결을 생성한다. 대신 자신의 파트너 중 가장 성능이 좋지 않은 것을 파트너에서 제외시킨다. 이렇게 함으로서 시간이 지남에 따라 노드는 더 좋은 성능을 갖는 파트너 연결을 가지고 있을 수 있다.

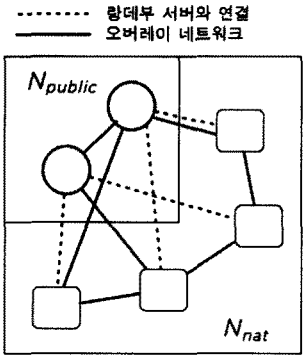


그림 2 랑데부 서버와의 연결

그림 3은 노드 A가 노드 B로의 파트너 연결을 맺는 과정이다. 노드 A는 랑데부 서버로 노드 RA를 이용하고 있으며, 노드 B는 랑데부 서버로 노드 RB를 이용하고 있다. 그림 3의 1)과 같이 노드 B로의 연결을 시도하는 노드 A는 노드 B의 랑데부 서버 RB에게 노드 B의 식별자와 자신의 내부 주소를 전송한다. 그림 3의 2)와 같이 RB는 B에게 A의 외부 주소와 내부 주소를 알려주고, 노드 A에게는 노드 B의 내부주소와 외부 주소를 보낸다. A와 B는 서로의 외부 주소와 내부주소로 패킷을 보내 접속을 시도한다. 연결이 성공하면 파트너들 사이에서는 서로의 버퍼 상태를 주기적으로 교환하고, 파트너의 대역폭을 측정한다. 스케줄러는 파트너의 버퍼 상태와 대역폭을 고려하여 자신이 필요한 데이타를 어떤 파트너로 부터 받을 것인지를 결정해 전송 요청을 하게 된다. 전송된 데이타는 버퍼에 저장되어 재생되기를 기다리며, 데이타를 받은 노드는 새로운 데이타가 버퍼에 도착했음을 주변 파트너에게 알리게 된다.

그림 3에서 설명한 분산화 된 랑데부서버를 사용한 연결 생성 과정 자체는 간단하지만, 이를 위해 노드의 참여 탈퇴 시 추가되어야 하는 몇 가지 과정이 있다. 노드의 참여시 다른 노드들이 새로 참여한 노드가 N_{public}에 속하는 노드인지, N_{nat}에 속하는 노드인지를 알 수 있어야 한다. 그리고 N_{nat}에 속하는 노드의 경우는 노드가 사용하는 랑데부 서버의 주소 또한 알 수 있어야 한다. 특히 랑데부서버로 사용되고 있는 노드가 네트워크에서 탈퇴할 때, 해당 노드를 랑데부 서버로 사용하는 노드는 모두 새로운 랑데부 서버를 찾아야 한다. 게다가 랑데부 서버가 바뀌었음을 다른 노드가 알 수 있도록 해야 한다. 이런 추가적인 오버헤드를 최소화해 노드의 잦은 출입에도 여전히 높은 확장성을 가지고 있도록 해야 한다. 이를 위해 DONet을 기반으로 하여 최소한의 메시지를 추가하여 분산화 된 Hole Punching기능을 가지는 DONet-P를 생성하였다.

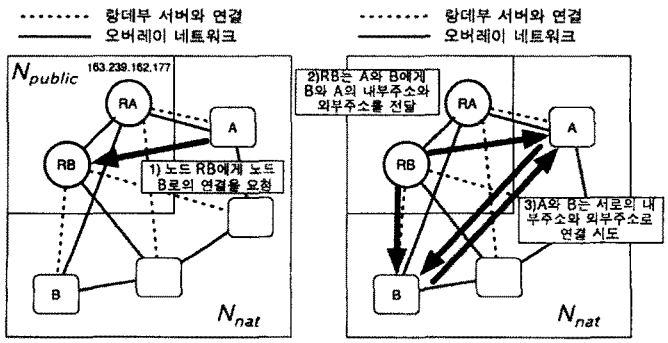


그림 3 분산화 된 랑데부 서버를 이용한 Hole Punching

3.3 노드의 구조

DONet-P는 참여하는 노드 각각이 랑데부 서버모듈을 가지고 있다는 것 이외에는 DONet과 노드 구조가 동일하다. DONet-P의 멤버쉽 매니저는 {식별자, 랑데부 서버 주소}의 엔트리를 갖는다. N_{nat}에 속한 노드인 경우, 노드를 유일하게 구분해 낼 수 있는 식별자와 노드의 랑데부서버 주소가, N_{public}에 속한 노드인 경우 노드의 식별자에는 *NULL*이 입력되고 랑데부 서버의 주소에 해당 노드의 주소가 들어간다.

3.4 멤버쉽 프로토콜

3.4.1 노드의 참여

새로 참여하는 노드를 N, 소스 노드를 S라고 한다. 새로 참여하는 노드 N은 우선 소스노드 S에 접근해서 소스노드의 대리노드 D를 요청한다. 소스노드는 자신의 멤버쉽 매니저에서 식별자 필드가 *NULL*인 노드들 중 하나를 라운드 로빈 방식으로 선택해 새로 참여하는 노드에게 전달한다. 식별자 필드가 *NULL*인 노드는 N_{public}에 속한 노드로서 랑데부 서버로 사용될 수 있기 때문이다. 만약 멤버쉽 매니저가 식별자 필드가 *NULL*인 노드를 찾을 수 없다면 자기 자신의 주소를 보낸다. 대리 노드 D의 주소를 받은 노드 N은 D에게 자신의 내부 주소를 실어 네트워크에 참여 한다는 메시지를 보낸다. 노드 D는 노드 N으로부터 받은 참여 메시지 속의 노드 N의 내부 주소와 참여 메시지 패킷 헤더의 주소를 비교한다. 만약 메시지 속의 주소와 패킷 헤더의 주소가 같다면, 새로 참여하는 노드 N은 N_{public}에 속하

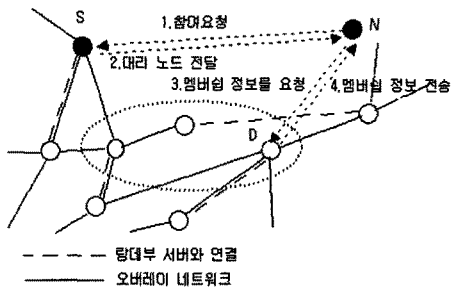


그림 4 DONet-P 노드의 참여

```
boolean JoinNetwork(Address inN, MEntry S)
// called when a node(N) joins the network
// Address inN: the internal address of itself
// MEntry S: the membership entry of the source node
{
  Get the address of D by requesting it to S
  Make a join request message // the message includes inN
  Send the join message to D
  IF N joins the network successfully THEN
    RETURN true
  ENDIF
  RETURN false
}
```

```
void SendAddrOfD(MEntry *list, MEntry dest)
//called when a source node receives a message requesting a deputy node
//MEntry *list: a list of membership entries managed by membership manager
//MEntry dest: the membership entry for the newly joining node
//count(global): the total number of rendezvous server address retrievals
//size(global): the number of total membership entries in list
{
  Retrieve the rendezvous server address of (count%size)th entry in list
  Send the address to dest and increment the count value
}
```

```
void processJoinMsg(MEntry *list, Message joinMsg)
//called when a deputy node receives a join message(joinMsg)
{
  Extract the membership entry of the joining node from joinMsg
  Add the entry of the new node to list
  IF the identifier field of the new node is NULL THEN
    Add the entry in its rendezvous server module
  ENDIF
  Send success message to the joining node
  Broadcast the join of N to every node managed by its membership manager through SCAM
}
```

그림 5 DONet-P 노드의 참여

는 노드임을 알 수 있다. 메시지 속의 주소와 패킷 헤더의 주소가 다르다면 노드 N이 N_{nat}에 속하는 경우이므로, 노드 D는 자신의 랑데부 서버 모듈에 노드 N을 등록한다.

결과적으로 새로 참여하는 노드는 네트워크를 구성하는 노드 중, NAT내부에 있지 않아 랑데부 서버 역할을 할 수 있는 노드들 중 하나를 선택해 랑데부 서버로 이용하게 된다. 그림 4는 노드의 참여과정을 나타내고 그림 5는 실질적인 프로토콜의 동작을 묘사한다.

3.4.2 노드의 탈퇴

네트워크에서 노드의 탈퇴는 노드의 탈퇴를 알리는 메시지를 SCAM을 통해 네트워크에 전파시킴으로서 처리된다. SCAM을 통해 전파되는 노드 탈퇴 메시지를 수신한 경우, 수신한 노드와 탈퇴한 노드가 N_{public}인지 N_{nat}인지에 따라서 처리 과정이 다르다.

노드 탈퇴 메시지를 수신한 노드가 N_{public}나 N_{nat}에 속하는 노드고 탈퇴한 노드는 N_{public}에 속하는 노드라면, N_{public}에 속하는 노드는 랑데부 서버로 이용되므로, 자신의 멤버쉽 매니저에서 랑데부 서버로 탈퇴한 노드

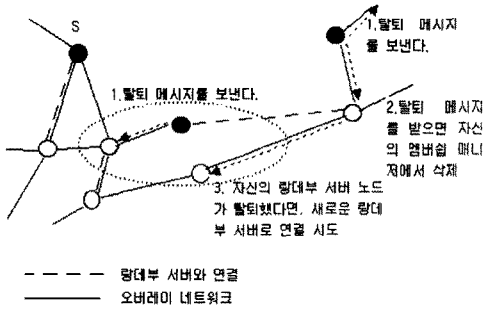


그림 6 DONet-P 노드의 탈퇴

를 이용하는 모든 노드를 찾아 삭제한다. 이때 탈퇴 메시지를 수신한 노드가 N_{nat}에 속하는 노드고 탈퇴한 노드가 자신이 이용하는 랑데부 서버라면, 네트워크에서 더 이상 새로운 연결을 받아드릴 수 없는 상태이므로, 자신의 멤버쉽 매니저에 있는 N_{public}에 속하는 노드에 자신의 내부 주소를 포함한 네트워크 참여 요청 메시지를 보내 새로운 랑데부 서버를 찾는다. 멤버쉽 매니저에서 N_{public}에 속하는 노드를 찾을 수 없는 경우, 처음 새로운 노드가 네트워크에 참여하는 방법대로 소스노드에 랑데부 서버로 사용할 대리 노드를 요청한다.

탈퇴한 노드가 N_{nat}에 속하는 노드라면 탈퇴 메시지를 수신한 노드의 멤버쉽 매니저는 해당노드의 정보를 찾아 삭제한다. 탈퇴 메시지를 수신한 노드가 N_{public}에 속하는 노드라면 탈퇴한 노드가 자신을 랑데부 서버로 사용한 노드인지를 살펴본 후, 랑데부 서버 모듈에서 해당 노드에 관한 정보를 삭제한다.

이와 같은 과정으로 모든 노드의 멤버쉽 매니저는 N_{nat}에 속하는 노드가 사용하는 랑데부 서버의 주소를 항상 최신으로 유지하고, N_{nat}에 속하는 노드는 언제나 랑데부 서버가 네트워크에서 탈퇴했을 때 새로운 랑데부 서버와 연결을 하게 된다. 그림 6, 그림 7은 노드의 탈퇴과정을 나타낸다.

3.4.3 멤버쉽 정보의 추가와 갱신

N_{public}에 속하는 노드는 새로운 노드 N의 참여 요청 메시지를 받은 후 N이 네트워크에 새로 참여했음을 SCAM을 통해 네트워크에 전파시킨다. 이때 SCAM을 통해 네트워크에 전파되는 메시지는 멤버쉽 매니저가 유지하는 엔트리인 {식별자, 랑데부 서버 주소}와 동일한 필드 구조를 갖는다. 즉 식별자와 랑데부 서버 주소를 필드로 갖고, N_{public}에 속하는 노드인 경우 식별자

```
void handleLeave(MEntry cNode, MEntry dNode, MEntry *list)
//called when a node get a message about the leave of a node through SCAM
//MEntry cNode: the membership entry of itself
//MEntry dNode: the membership entry of the leaving node
{
  IF the identifier of cNode is NULL THEN // cNode uses NAT
    IF the dNode is a rendezvous server of cNode THEN
      WHILE the return value of JoinNetwork(cNode identifier, S) is true
      ENDWHILE
    ENDIF
  ELSE
    IF the dNode is a node using cNode as its rendezvous server THEN
      Retrieve and remove the entry of dNode from the rendezvous server module of cNode
    ENDIF
  ENDIF
  Retrieve and remove cNode from list
}
```

그림 7 DONet-P 노드의 탈퇴

필드에는 NULL을, 랑데부 서버 주소 필드에는 노드의 주소를 싣는다.

새로운 노드 N이 참여했음을 알리는 메시지를 SCAM에 의해 수신한 노드는, 자신의 멤버쉽 매니저에 해당 노드에 관한 항목이 있다면, 새로 수신한 정보로 해당 항목을 갱신한다. 만약 멤버쉽 매니저에 해당항목이 없다면, 해당 항목을 추가한다. 이때 멤버쉽 매니저가 유지할 수 있는 항목의 개수는 상한이 있으므로 상한을 넘어 항목 추가가 불가능한 경우가 발생한다. 이 경우 무작위로 하나의 항목을 삭제한 후 새로운 항목을 추가한다. 이와 같은 과정을 통해 모든 노드가 새로운 노드 N의 참여를 인지함과 동시에, 노드 N이 N_{nat}에 속하는 경우 노드 N의 랑데부 서버 주소를 알 수 있다.

3.5 파트너 연결

파트너 연결 시 Hole Punching알고리즘이 사용된다. M개 이하의 파트너를 가지고 있는 노드 A는 멤버쉽 매니저로부터 무작위로 선택된 노드 B로 파트너 연결 요청을 하게 된다. 선택한 항목의 식별자가 NULL인, N_{public}에 속하는 노드라면, 노드의 주소로 곧바로 파트너 연결 요청 메시지를 보낸다. 선택한 항목의 식별자가 NULL이 아닌, N_{nat}에 속하는 노드라면, 선택한 노드의 랑데부 서버 RB로 연결을 원하는 노드의 식별자와 자신의 내부 주소를 실은 Hole Punching 요청 메시지를 보낸다.

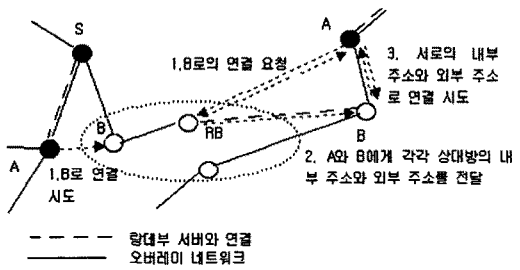


그림 8 DONet-P 파트너 연결

랑데부 서버가 Hole Punching 요청 메시지를 받으면 메시지에 담긴 식별자로 연결을 원하는 노드가 노드 B임을 파악한다. 그리고 노드 A와 노드 B로 각각 노드 B와 노드 A의 내부주소와 외부주소를 보낸다. 랑데부 서버로부터 응답을 받은 노드 A와 노드 B는 서로의 내부주소와 외부주소로 파트너 연결 요청 패킷을 보내 연결을 시도한다. 파트너 연결 요청을 받은 노드는 M개 이하의 파트너를 가지고 있다면 파트너 연결 요청을 수락하고, 그렇지 않다면 거절한다. 그림 8, 그림 9는 파트너 연결과정을 나타낸다.

3.6 컨트롤 메시지 오버헤드

DONet-P에 참여하는 노드가 받는 컨트롤 메시지 오버헤드는 멤버쉽 정보와 파트너의 유지를 위해 주고받는 메시지, 랑데부 서버와의 연결 유지를 위한 메시지, 데이타 전송을 위한 메시지의 합으로 나타낼 수 있다. 멤버쉽 정보의 유지를 위한 컨트롤 메시지는 노드가 네트워크에 새로 참여하거나 탈퇴하는 경우 발생한다. DONet-P에 노드가 네트워크에 참여하거나 탈퇴하는 경우 DONet과 동일한 컨트롤 메시지 오버헤드가 발생하지만, 랑데부 서버로 사용되는 노드가 네트워크에서 탈퇴하는 경우에 해당 노드를 랑데부 서버로 사용하는 노드들이 새로 랑데부 서버를 선택해 네트워크에 다시 참여해야 한다. R_{nat}는 NAT를 사용하는 노드의 비율, NR은 랑데부 서버 역할을 하는 노드가 평균적으로 Hole Punching 서비스를 제공하는 노드의 개수라고 할 때, 멤버쉽 정보의 유지를 위한 컨트롤 메시지 오버헤드 $CM_{distributed}$는 식 (5)와 같이 나타낼 수 있다.

$$NR = \frac{R_{nat}}{1-R_{nat}}$$

$$CM_{distributed} = CM_{donet} + N \times D_l \times (1-R_{nat}) \times NR \qquad (5)$$

```
void managePartnership(MEntry *list)
// called when the number of partners is less than M
{
  WHILE the number of partners is smaller than M
    Retrieve a target node different from current partners in the list randomly
    IF the identifier is not NULL THEN // the target node uses NAT
      Get internal address of the target node from the node's rendezvous server
    ENDIF
    Send a partnership request message to the target node
    IF the partnership be made successfully THEN
      Increment the number of partners
    ENDIF
  ENDWHILE
}
```

그림 9 DONet-P 파트너 연결

DONet-P에서 파트너의 유지를 위해 주고받는 메시지의 경우, DONet에서 파트너 유지를 위해 주고받는 컨트롤 메시지 오버헤드 CP_{donet}에 Hole Punching시 발생하는 컨트롤 메시지 오버헤드가 추가된다. O_h가 Hole Punching시 발생하는 컨트롤 메시지 오버헤드라고 한다면 식 (6)과 같이 파트너 연결에 필요한 컨트롤 메시지 오버헤드 $CP_{distributed}$를 구할 수 있다.

$$CP_{distributed} = N \times \{(M \times D_j + D_l) \times (O_p + O_h)\} \quad (6)$$

NAT안에 있는 모든 노드는 랑데부 서버와의 연결 유지를 위해 주기적으로 주고받아야 하므로 랑데부 서버로의 연결 유지를 위해 단위시간당 발생하는 컨트롤 메시지 오버헤드를 O_k라고 하면, 랑데부 서버와의 연결 유지를 위한 컨트롤 메시지 오버헤드는 식 (7)과 같이 구할 수 있다.

$$CK_{distributed} = N \times R_{nat} \times O_k \quad (7)$$

데이타 전송을 위해 버퍼맵 정보를 교환할 때 발생하는 컨트롤 메시지 오버헤드는 DONet의 그것과 같다. 결과적으로 하나의 노드가 단위시간동안 평균적으로 받는 컨트롤 오버헤드 $C_{distributed}$는 식 (8)과 같이 구할 수 있다.

$$C_{distributed} = \frac{CM_{distributed} + CP_{distributed} + CD_{donet} + CK_{distributed}}{N} \quad (8)$$

식 (8)에 따라 DONet-P에 참여하는 노드가 받는 오버헤드는 단위시간동안 네트워크에 참여하거나 탈퇴하는 노드의 비율, NAT를 사용하는 노드의 비율에 비례하여 증가 할 것이다. 한편 O_r이 Hole Punching 요청을 처리하기 위해 랑데부 서버가 받는 메시지 오버헤드라고 하면, 식 (9)는 NAT를 사용하지 않는 노드가 Hole Punching역할을 하기 위해 단위시간당 받는 컨트롤 메시지 오버헤드를 나타낸다.

$$CR_{distributed} = NR \times (M \times D_j + D_l) \times N \times O_r + \frac{CK_{distributed}}{N \times (1 - R_{nat})} \quad (9)$$

식 (9)에서 보는 것과 같이 DONet-P에서 Hole Punching을 위해 네트워크에서 생성된 모든 컨트롤 메시지 오버헤드는 NAT를 사용하지 않는 노드가 나누어 가진다. 따라서 NAT를 사용하지 않는 노드 하나가 Hole Punching역할을 하기 위해 받는 컨트롤 메시지 오버헤드는 단위시간동안 네트워크에 참여하거나 탈퇴하는 노드의 비율과, NAT를 사용하는 노드의 비율에 비례하여 증가할 것이다.

한편 기존의 DONet에 하나의 랑데부 서버를 사용하여 Hole Punching 기법을 추가한 DONet-C의 경우 하나의 노드가 단위시간동안 받는 컨트롤 메시지 오버헤드 $C_{centralized}$는 랑데부 서버로의 연결과 연결 유지에 필요한 컨트롤 메시지 오버헤드 $CK_{centralized}$, Hole Punching에 필요한 컨트롤 메시지 오버헤드 $CH_{centralized}$가 DONet에서의 컨트롤 메시지 오버헤드에 추가된다. 따라서 $C_{centralized}$는 식 (10)과 같이 구할 수 있다.

$$CH_{centralized} = N \times R_{nat} \times (D_j + D_l) \times O_h \quad (10)$$

$$C_{centrulized} = C_{donet} + \frac{CK_{centralized} + CH_{centralized}}{N}$$

DONet-C의 랑데부 서버가 Hole Punching기능을 수행하기 위해 받는 컨트롤 메시지 오버헤드 $CR_{centralized}$만을 따로 구해보면 식 (11)과 같이 랑데부 서버로의 연결 요청, Hole Punching메시지, 연결 유지 메시지의 합으로 나타낼 수 있다.

$$CR_{centralized} = CK_{centralized} + CH_{centralized} \quad (11)$$

랑데부 서버가 Hole Punching기능을 수행하기 위해 받는 컨트롤 메시지 오버헤드는 단위시간동안 네트워크에 참여하거나 탈퇴하는 노드의 비율, NAT를 사용하는 노드의 비율, 총 노드의 개수에 비례하여 늘어날 것이다. 특히 총 노드의 개수에 비례하여 랑데부 서버가 받는 컨트롤 메시지 오버헤드가 증가하기 때문에 DONet-P와는 다르게 스트리밍 오버레이 네트워크의 확장성이 제한됨을 알 수 있다.

4. 성능 평가

본 장에서는 앞서 제안한 분산 Hole Punching기능을 가지는 스트리밍 오버레이 네트워크의 성능을 측정했다. 이를 위해 네트워크 시뮬레이터 위에서 DONet, DONet에 하나의 랑데부 서버를 사용하는 스트리밍 오버레이 네트워크, 분산 Hole Punching기능을 가지는 스트리밍 오버레이 네트워크를 구현하여, 세 가지 네트워크의 컨트롤 오버헤드와 데이타 연속성, 확장성을 측정해 성능을 비교했다.

4.1 시뮬레이션 모델

네트워크 시뮬레이션 환경을 제공하는 OMNeT++ 3.3[8]을 사용해 세 가지 네트워크를 구현하였다. 또한 IP네트워크 환경을 만들기 위해 [8]에서 제공하는 INET Framework release 2006-10-12를 사용했다. 실험을 위해서는 물리적인 토폴로지를 실제 인터넷과 유사하게 생성해야 할 필요가 있다. 실제 인터넷과 같은 큰 규모의 물리적인 네트워크 토폴로지는 small world와 power law속성을 갖는다고 알려져 있다[9,10]. BRITE[11]는 토폴로지 생성 툴로 실제 인터넷과 같이 small world와 power law속성을 갖는 토폴로지를 생성할 수 있다[9]. BRITE를 이용해 다양한 크기의 토폴로지를 생성해 시뮬레이션에 사용했다. Symmetric NAT는 존재하지 않

는다고 가정했다.

실험에 앞서 제안한 오버레이 네트워크의 여러 파라메타를 설정해야 한다. 최대 파트너의 개수인 M은 4로 고정했다. M값의 크기에 대한 성능 변화는 [12]에 분석되어있으며, 실험하는 네트워크에서 M이 4인 경우 95%의 노드가 6홉 안에 서로 연결이 가능할 정도로 충분한 값이다[1]. 실험의 스트림 데이타는 200Kbps의 전송률을 가지며, 각각의 노드는 120초 분량의 스트림 데이타를 저장할 수 있는 크기의 버퍼를 가지고 있다. 버퍼는 120조각으로 나누어져 하나의 데이타 조각은 1초 분량의 데이타를 가진다. DONet은 네트워크의 어느 두 노드를 선택해도 해당 노드가 재생하고 있는 스트림 데이타의 시간 차이가 최대 1분이 넘지 않는다[1]. 본 논문에서 제시한 오버레이 네트워크는 Hole Punching기능을 제외하고는 DONet과 같기 때문에 120초는 충분한 분량의 버퍼이다.

4.2 성능 측정 및 분석

성능 비교를 위해 DONet, DONet에 하나의 랑데부 서버를 사용하여 Hole Punching 기능을 추가한 경우(이하 중앙 Hole Punching 오버레이 네트워크), 본 논문에서 제시한 분산 Hole Punching기능을 가지는 스트리밍 오버레이 네트워크(이하 분산 Hole Punching 오버레이 네트워크)를 비교했다. 세 가지 스트리밍 오버레이 네트워크에 대해 각각 표 1과 같은 실험을 하였다. 각각의 테스트케이스에 대해 네트워크에 참여하는 노드의 개수가 200, 400, 800개 일 때 실험을 수행했다. 표 1의 노드 참여/탈퇴는 1초에 네트워크를 구성하는 노드의 몇 퍼센트가 새로 참여하고 탈퇴하는가를 나타낸 것이다. 높을수록 빈번한 노드의 참여와 탈퇴가 일어남을 뜻한다. 일반적인 환경에서 노드의 참여와 탈퇴가 가장 빈번할 때, 초당 약 1%의 노드가 참여하거나 떠난다. 모든 실험은 10분간 네트워크를 유지하여 값을 측정했다.

4.2.1 데이타 연속성

데이타 연속성은 노드가 버퍼로부터 데이타 조각을 가져가 재생을 하려고 할 때, 필요한 데이타 조각이 준비되어 있을 확률로 정의된다. 그림 10은 각각 노드의 개수가 200일 때, N_{nat}안에 있는 노드의 비율과 노드 변화율을 달리하여 데이타 연속성을 측정한 그래프이다.

DONet, 중앙 Hole Punching 오버레이 네트워크, 분산 Hole Punching 오버레이 네트워크 모두 노드 변화율이 높아질수록 데이타 연속성이 떨어진다. Hole Punching기능이 없는 DONet의 경우는 N_{nat}에 포함되는 노드의 비율이 높을수록 노드 변화율 증가에 의해 급격하게 성능이 저하된다. 이는 노드의 참여와 탈퇴가 빈번하게 일어나 새로운 노드와의 파트너 연결을 해야 하는 경우가 많아지지만, N_{public}에 속하는 노드 비율이 작기 때문에 연결 가능한 노드의 후보가 제한됨으로서 효율적인 네트워크를 만드는데 어려움을 받기 때문이다. 이에 비해 중앙 Hole Punching 오버레이 네트워크와 분산 Hole Punching 오버레이 네트워크는 노드 변화율이 높아짐에 따라서 약간의 성능저하를 보여주지만, DONet에 비해서는 모든 경우에 월등이 좋은 데이타 연속성을 제공한다.

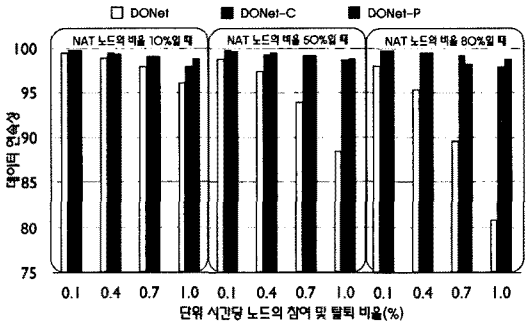


그림 10 데이타 연속성

표 1 테스트케이스

번호	노드의 개수	N_{pnblic}의 비율	N_{nat}의 비율	노드 참여/탈퇴
1	200/400/800	90%	10%	0.1% / s
2	200/400/800	90%	10%	0.4% / s
3	200/400/800	90%	10%	0.7% / s
4	200/400/800	90%	10%	1.0% / s
5	200/400/800	50%	50%	0.1% / s
6	200/400/800	50%	50%	0.4% / s
7	200/400/800	50%	50%	0.7% / s
8	200/400/800	50%	50%	1.0% / s
9	200/400/800	20%	80%	0.1% / s
10	200/400/800	20%	80%	0.4% / s
11	200/400/800	20%	80%	0.7% / s
12	200/400/800	20%	80%	1.0% / s

4.2.2 확장성

평균 메시지 오버헤드는 네트워크에 참여하고 있는 노드의 컨트롤 메시지 오버헤드의 평균값이고 랑데부 서버의 메시지 오버헤드는 랑데부 서버로 이용되고 있는 노드의 Hole Punching관련 컨트롤 메시지 오버헤드이다. DONet의 경우는 랑데부 서버가 존재 하지 않고, 중앙 Hole Punching 오버레이 네트워크의 경우는 하나의 랑데부 서버를 네트워크의 모든 노드가 사용한다. 분산 Hole Punching 오버레이 네트워크의 경우는 노드의 개수 $*N_{public}$ 수만큼의 랑데부 서버가 네트워크에 존재하게 된다.

그림 11은 테스트 8에서 800개의 노드를 가지고 시뮬레이션을 했을 때 얻은 결과를 표시한 것이다. 식 (11)과 같이 중앙 Hole Punching 오버레이 네트워크의 경우는 랑데부 서버에 메시지가 집중되기 때문에 랑데부 서버가 다른 모든 노드의 평균 메시지 오버헤드보다 상당히 큰 부하를 받는다. 이에 반해 분산 Hole Punching 오버레이 네트워크의 경우 식 (9)와 같이 오버헤드를 네트워크에 존재하는 모든 N_{public}에 포함되는 노드에 분산시켰기 때문에, 랑데부 서버가 받는 부하는 평균 컨트롤 메시지 전송량에 큰 영향을 미치지 않는 작은 값이다. 800개 노드 중 NAT를 사용하는 노드의 비율이 50%일 때, 중앙 Hole Punching오버레이 네트워크의 랑데부 서버가 분산 Hole Punching 오버레이 네트워크의 랑데부 서버보다 Hole Punching작업을 수행하기 위해 약 300배 정도 많은 컨트롤 메시지 오버헤드를 받음을 알 수 있다.

그림 12는 N_{public}에 속하는 노드의 비율이 가장 작고 노드 변화율이 가장 커 컨트롤 메시지의 오버헤드가 가장 높은 테스트 12에서, 분산 Hole Punching 오버레이

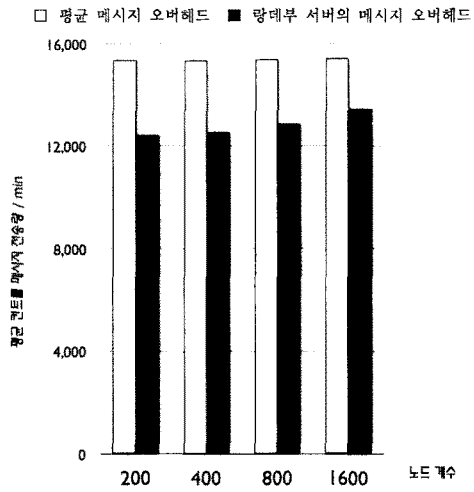


그림 12 DONet-P의 컨트롤 메시지 오버헤드

네트워크의 평균 컨트롤 메시지 오버헤드와 랑데부서버가 받는 메시지 오버헤드를 나타낸다. 그래프에서 분산 Hole Punching 오버레이 네트워크의 평균 컨트롤 메시지 오버헤드는 노드 수의 증가에 큰 영향을 받지 않고 거의 일정하게 유지됨을 알 수 있다. 랑데부 서버의 메시지 오버헤드는 노드의 개수가 200개에서 1600개로 늘어남에 따라 약 10%이내의 작은 증가를 보였다. 따라서 분산 Hole Punching 오버레이 네트워크는 높은 확장성을 가지고 있다고 말할 수 있다.

4.2.3 컨트롤 메시지 오버헤드

그림 13은 노드의 개수(800개)와 노드의 변화(1.0%/s)는 일정하고 N_{nat}에 속하는 노드의 비율이 달라질 때 전체 전송량에서 컨트롤 메시지가 차지하는 비율을 세 가지 네트워크에 대해 각각 나타낸 그래프이다.

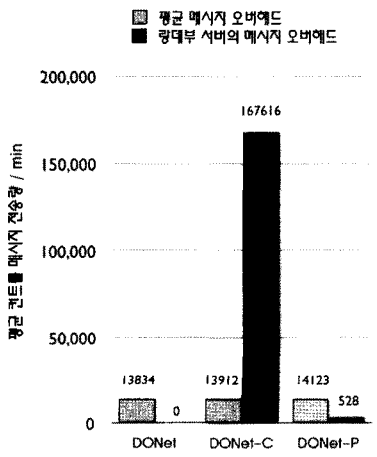


그림 11 평균 오버헤드와 랑데부 서버의 오버헤드

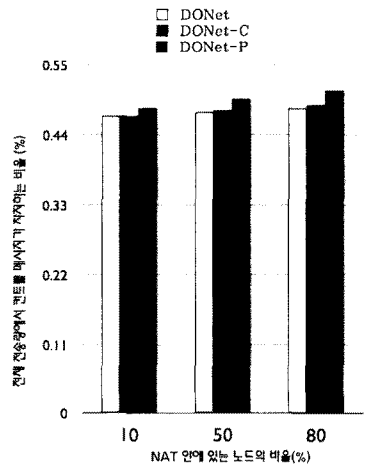


그림 13 컨트롤 메시지 오버헤드

그림에서 보는 것과 같이 N_{public}에 속하는 노드의 비율이 작아질수록 전체적으로 컨트롤 메시지의 양이 늘어나는 것을 볼 수 있다. 이는 N_{nat}에 속하는 노드가 늘어남으로서, Hole Punching에 필요한 메시지 전송에 의한 것임을 알 수 있다. 또한 DONet보다는 하나의 랑데부 서버를 사용한 중앙 Hole Punching 오버레이 네트워크가 컨트롤 메시지의 비율이 높은 이유는 식 10과 같이 중앙 Hole Punching 오버레이 네트워크는 DONet에 비해 Hole Punching을 위해 랑데부 서버와 주고받는 메시지와 NAT의 매핑 테이블 유지를 위해 주기적으로 랑데부 서버로 보내는 패킷이 있기 때문이다. 분산 Hole Punching 오버레이 네트워크가 중앙 Hole Punching 오버레이 네트워크보다 컨트롤 메시지의 비율이 높은 이유는 식 (8)과 같이 중앙 Hole Punching 오버레이 네트워크의 중앙 랑데부 서버를 분산화 하고 유지하는데 필요한 메시지 때문이다. 하지만 전체 전송량에서 컨트롤 메시지가 차지하는 비율은 모두 0.5%를 넘지 않는 크기이며 전송되는 데이타 크기에 비해 미미한 수준의 오버헤드이다.

그림 14는 노드의 개수(800)는 일정하고, N_{nat}안에 있는 노드의 비율과 노드 변화율을 다르게 하여 DONet, 중앙 Hole Punching 오버레이 네트워크, 분산 Hole Punching 오버레이 네트워크에 대해 컨트롤 메시지 오버헤드를 측정한 그래프이다. NAT안에 있는 노드의 비율이 같은 경우에는 단위시간당 노드의 참여 탈퇴 비율이 높아짐에 따라 컨트롤 메시지 오버헤드가 커짐을 볼 수 있다. 이는 노드의 네트워크에 참여, 탈퇴가 빈번하게 일어나기 때문에 이에 해당하는 메시지의 증가로 인한 오버헤드이다. 그리고 노드 참여시 중앙 Hole Punching 오버레이 네트워크의 경우는 Hole Punching을 위한 메시지만이 DONet에 비해 추가적인 오버헤드가 되므로 DONet과의 차이는 적은 편이다. 분산 Hole Punching 오버레이 네트워크의 경우는 Hole Punching을 위한 메시지 오버헤드뿐 아니라, Hole Punching기능을 네트워크 전체에 분산시키는데 필요한 메시지들이 추가되어 더 큰 메시지 오버헤드를 가진다. 하지만 여전히 전체적으로 전송되는 데이타 크기에 비해 약 0.5%를 넘지 않으므로, 스트림 데이타에 비교하면 작은 오버헤드를 가진다고 할 수 있다.

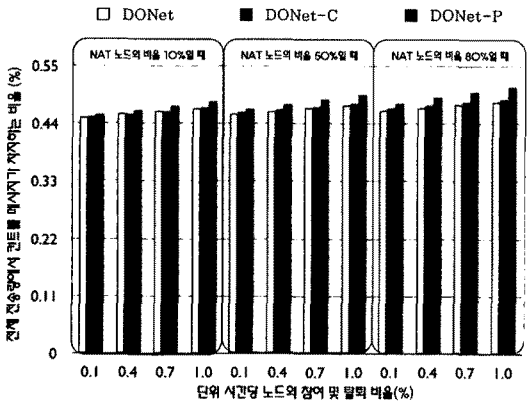


그림 14 컨트롤 메시지 오버헤드

5. 결론 및 향후 과제

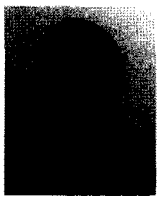
기존의 스트리밍 오버레이 네트워크는 인터넷에 존재하는 NAT에 대한 고려 없이 연구가 진행되어 왔다. 따라서 NAT가 존재하는 실제 인터넷에서는, 선택할 수 있는 파트너의 후보가 제한되어 오버레이 네트워크의 성능이 떨어지는 문제가 발생했다. 본 논문에서는 Hole Punching기능을 가지는 스트리밍 오버레이 네트워크를 제시하였다. Hole Punching기능을 제공하면서, 높은 확장성을 유지하기 위해 Hole Punching서비스를 수행하는 랑데부 서버의 작업을 네트워크 전체에 분산시키는 알고리즘을 제시하였다. 본 논문에서 제안한 알고리즘의 성능을 측정하기 위하여 컨트롤 메시지 오버헤드, 데이타 연속성을 측정했다. Hole Punching기능을 제공하지 않는 DONet과, 하나의 랑데부서버가 Hole Punching기능을 제공하는 네트워크, 분산된 랑데부 서버가 Hole Punching기능을 제공하는 네트워크에서 각각 컨트롤 메시지 오버헤드와 데이타 연속성을 비교 측정하였다. 본 논문에서 제시한 스트리밍 오버레이 네트워크는 NAT를 사용하는 노드가 크게 늘어나도(80%이상), 그렇지 않을 경우에 비해 뚜렷한 성능 저하가 없었다. 또 하나의 랑데부 서버에 부하가 집중되지 않고 여러 노드에 골고루 부하가 분산 되어 Hole Punching기능이 수행됨을 확인할 수 있었다.

하지만 본 논문에서는 파트너 연결을 위한 노드 선택시 멤버쉽 매니저에서 임의로 노드를 선택한다. [1]과 같이 지역적인 정보를 사용해 노드를 선택할 수 있는 네트워크는 오버레이 네트워크의 성능을 크게 향상시킬 수 있음이 알려져 있다. 추후 분산된 Hole Punching 기능을 이러한 오버레이 네트워크의 성능향상기법과 함께 적용할 수 있는 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] Zhang, X., Liu, J., Li, B. and Yum, T. P., "DONET: A Data-Driven Overlay Network for Efficient Live Media Streaming," in Proceedings of IEEE INFOCOM, pp.141-146, 2005.
[2] illuminati; http://illuminati.coralcdn.org.
[3] Rosenberg, J., Weinberger, J., Huitema, C., and

Mahy, R., "STUN-Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translaters(NATs)," RFC 3489, 2003.
[4] Ford, B., Srisuresh, P. and Kegel, D., "Peer-to-Peer(P2P) Communication across Network Address Translators(NATs)," USENIX Animal Technical Conference, 2005.
[5] Ganesh, A. J., Kermarrec, A.-M. and Massoulie, L., "Peer-to-peer membership management for gossip-based protocols," IEEE Transactions on Computers, Volume 52, pp.139-149, 2003.
[6] Cormen, T.H., Leiserson, C. E., Rivest, R. L. and Stein, C., Introduction to Algorithms, Second Edition, MIT Press, Cambridge, MA, 2001.
[7] UPnP Forum; http://upnp.org.
[8] OMNeT++, Discrete event simulation environment, http://www.omnetpp.org.
[9] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S. and Willinger, W., "Network Topology Generators: Degree-Based vs. Structural," in Proceedings of ACM SIGCOMM, Volume 32, pp.147-159, 2002.
[10] Saroiu, S., Gummadi, P. and Gribble, S., "A Measurement Study of Peer-to-Peer File Sharing Systems," in Proceedings of Multimedia Computing and Networking (MMCN), 2002.
[11] BRITE; http://www.cs.bu.edu/brite.
[12] Zhang, X., Liu, J., Li, B. and Yum, T.-S. P., "DONet/CoolStreaming: A data-driven overlay network for live media streaming," Technical Report, 2004.



이 문 수

2005년 서강대학교 컴퓨터공학과 졸업(공학사). 2007년 서강대학교 컴퓨터공학과 대학원 졸업(공학석사). 2007년~현재 매크로임팩스 시스템소프트웨어 연구소 연구원. 관심분야는 Distributed system, P2P Realtime streaming

한 성 민

2005년 서강대학교 컴퓨터공학과 졸업(공학사). 2007년 서강대학교 컴퓨터공학과 대학원 졸업(공학석사). 2007년~현재 LG전자 MC사업부. 관심분야는 임베디드 시스템, 네트워크

박 성 용

1987년 서강대학교 컴퓨터학과(공학사) 1994년 미국 Syracuse University 대학원(공학석사). 1998년 미국 Syracuse University(공학박사). 1998년~1999년 미국 Bell Communication Research 연구원 1999년~현재 서강대학교 컴퓨터학과 부교수. 관심분야는 Autonomic Computing, Peer to Peer Computing, High Performance Cluster Computing and System