韓國 컴퓨터情報學會 論文誌
第11卷 第2號, 2006. 5.

2006-11-2-2-6

다중 피어 결합을 이용한 P2P 멀티미디어 스트리밍 프로토콜

정 의 현*

A P2P Multimedia Streaming Protocol Using Multiple-Peer Binding

Eui-Hyun Jung*

요 약

P2P 기술의 각광에도 불구하고 P2P 상에서 멀티미디어 스트리밍에 대한 연구는 상대적으로 주목을 받지 못하고 있다. 이것은 피어간의 낮은 대역폭과 신뢰도가 낮은 접속 등의 몇 가지 이유 때문에 P2P 멀티미디어 스트리밍이 어렵기 때문이다. 본 논문에서는 이 문제를 해결하기 위하여 여러 피어와의 연결을 한 개의 가상적인 채널로 제공하는 프로토콜인 다중피어 바인딩 프로토콜(MPBP: Multi-Peer Binding Protocol)을 제안한다. 이 프로토콜은 응용들이 여러 피어로부터 동시에 데이터를 다운로드 받아서 성능 개선과 안정된 스트리밍을 할 수 있게 해준다. 이를 위해서 MPBP는 미디어 파일을 작은 청크로 분리하고, 각 청크를 구분하고 전송할 수 있는 기법을 제공한다. 구현된 MPBP 엔진은 데이터 송신 서버로부터의 갑작스런 접속 단절을 효과적으로 다루는 것에 중점을 두어 개발되었으며, 실험 결과는 MPBP 프로토콜이 이를 효율적으로 다루는 것을 보여주었다. 또한 MPBP는 여러 미디어 타입에 대해서 지원이 가능하도록 설계되었으며, 이를 검증하기 위하여 본 논문에서는 비디오와 오디오 응용을 MPBP 엔진을 이용하여 개발하였다.

Abstract

In spite of the popularity of P2P technology, a multimedia streaming using the P2P technology has been neglected. The reason for this is that the P2P multimedia streaming has suffered from several inherent problems especially poor bandwidth and unreliable connection among peers. We suggest a Multi-Peer Binding Protocol (MPBP) in this paper that provides a virtual single channel composed of multiple connections to several peers to ease these problems. The protocol enables applications to download data from multiple peers simultaneously, so they can achieve throughput improvement and reliable streaming. For this, the MPBP splits media files into small chunks and provides a mechanism for identifying and transmitting each chunk. Implemented MPBP engine focuses on handling an abrupt disconnection from data sending peers and the evaluation result shows the MPBP is able to handle it gracefully. The MPBP is also designed to support various media types. To verify this, video and audio applications are implemented using the MPBP engine in this paper.

▸ Keyword : P2P(Peer-to-Peer), 멀티미디어 스트리밍(Multimedia Streaming)

• 제1저자 : 정의현
• 접수일 : 2006.03.28, 심사완료일 : 2006.05.22
* 안양대학교 디지털미디어공학과 교수

Ⅰ. 서 론

인터넷이 멀티미디어 데이터의 주요한 매체로 각광받으면서, 인터넷상에서 효율적으로 멀티미디어를 재생 및 분배하고자 하는 요구가 증대하고 있다. 이러한 요구를 만족시키기 위하여 산업계와 학계는 미디어 프로토콜, 서버 아키텍처, 브라우저 분야에서 다양한 연구와 제품을 제시하였다[1][2][3]. 일반적으로 네트워크를 통한 멀티미디어의 분배(Multimedia Retrieval)는 크게 다운로드 모드(Download)와 스트리밍(Streaming) 모드로 나눌 수 있다. 다운로드 모드는 원격(remote) 머신에 위치한 데이터 전체를 모두 전송받은 후에 로컬(local) 머신에서 재생하는 방식이다. 이 방식은 대역폭의 제한 없이 안전한 데이터 전송을 보장하는데 비해, 사용자가 원하는 시점에 해당 멀티미디어 데이터를 사용할 수 없는 단점을 갖고 있다. 이에 비해 스트리밍 모드는 원격 머신의 일부분의 데이터를 전송받은 후에 로컬 머신에서 바로 재생을 하고, 필요한 데이터를 원격 머신에서 계속 전송받는 방식이다.

스트리밍 모드는 데이터의 즉각적인 사용이 가능하다는 장점 때문에 다운로드 모드에 비해서 사용자들이 높은 선호도를 보이며, 이러한 이유 때문에 미디어 스트리밍에 대한 많은 연구가 진행되었다 [2][3]. 그러나 지금까지 대부분의 연구는 전용의 미디어 타입과 프로토콜에 근거한 클라이언트/서버 방식을 이용하고 있으며, 클라이언트/서버 방식에 근거한 미디어 스트리밍 구조는 기존의 P2P 기술 분야에서 지적된 문제를 그대로 안고 있다 [4][5]. 가장 먼저 지적되는 문제는 미디어 데이터의 확장성(scalability)에 관한 문제이다. 미디어 서버의 저장 용량에는 한계가 있기 때문에, 어떤 미디어 서버도 사용자가 원하는 데이터를 모두 제공할 수 없다. 따라서 사용자가 요구하는 미디어 데이터를 사용자가 원하는 시점에 제공할 수 없다는 문제를 갖게 된다. 두 번째로 부하 분산(load balancing)의 문제이다. 여러 클라이언트들이 동시에 서버에 접속하는 경우에는 반드시 부하 집중의 문제가 발생하게 된다. 이를 해결하기 위하여 클러스터링(clustering) 방식 등이 제안되고 있으나 대량의 트래픽(traffic)이 발생하는 경우에는 네트워크의 부하 집중 현상은 역시 피할 수 없게 된다.

이러한 문제를 해결하기 위해서 미디어 스트리밍에 P2P 구조를 적용하고자 하는 연구 들이 시도되었다[5][6]. PeerCast[7]나 Streamer[8] 등은 P2P 네트워크에서 사용자에게 오디오 스트리밍을 제공하는 연구로 주목받았으며, 앞서 지적한 클라이언트/서버 구조의 문제점에 대한 대안으로 간주되었다. 또한 검색을 위한 P2P 알고리즘에 대한 연구도 활발하게 일어났다[9].

그러나 P2P 네트워크의 특성 때문에 P2P 네트워크에서 미디어 스트리밍을 제공하는 것은 몇 가지 문제점을 안고 있다. 첫째, P2P 네트워크에서는 피어(peer)들이 자유롭게 세션(session)에 참여 및 탈퇴가 가능하기 때문에 안정적인 스트리밍을 보장할 수 없다는 점이다. 예를 들어 스트리밍이 한창 진행 중인 상황에서 미디어 송신 피어 (media sending peer)가 급작스럽게 접속을 단절하면, 미디어 수신 피어 (media receiving peer)는 스트리밍의 중단을 피할 수 있는 방법이 없다. 둘째, 일반적인 피어들은 전용의 미디어 서버와 같은 충분한 정도의 대역폭과 성능을 갖고 있지 않다는 점이다. 따라서 여러 피어로부터 스트리밍 요청을 받게 되는 경우 제대로 이를 감당할 수 없게 된다. 그러나 P2P 네트워크의 피어들은 다른 피어들에게 서비스를 제공하기 위해 인터넷을 사용하는 것이 아니기 때문에 이에 대한 근본적인 해결 방안을 찾기는 어려운 문제이다. 이러한 이유들 때문에 P2P를 스트리밍에 단순히 적용하는 것은 다른 문제의 시발점이 되고, P2P의 다른 분야와 달리 스트리밍에 관한 연구는 활발하지 못한 상황이다 [6].

본 논문에서는 이를 해결하기 위하여 다수의 피어들이 협동하여 안정적인 세션을 제공하는 다중피어 결합 프로토콜인 MPBP(Multi-Peer Binding Protocol)를 제안한다. 피어간의 단일 접속(single connection)을 이용한 기존 P2P 스트리밍 방식의 경우는 미디어 송신 피어가 갑자기 접속을 끊거나 피어간의 접속의 대역폭이 낮아지게 되면 제대로 스트리밍을 제공할 수 없다. 그러나 다중 피어 결합은 미디어 수신 피어가 스트리밍에 필요한 데이터를 여러 피어로부터 동시에 전송받아 하나의 가상적인 세션을 갖도록 구성해준다. 이 방식을 이용하면 여러 피어간의 접속을 결합하여 가상적인 대역폭의 확장이 가능하며, 세션 내의 일부 피어가 접속을 단절해도, 단절된 피어가 소유한 데이터를 다른 피어로부터 수신하여 미디어 수신 피어에게 안정적인 스트리밍을 제공할 수 있게 된다.

제안된 프로토콜은 다중 피어 결합을 제공하기 위해, 미디어 데이터를 64K 바이트 단위의 청크(chunk)로 분할하고, 청크 단위로 피어간의 통신 세션의 전송을 처리하게 된

다. 청크의 유일성을 보장하기 위해 MD5 [10] 알고리즘으로 미디어 데이터에서 얻은 128비트의 해쉬(hash)와 미디어 데이터 내의 위치를 이용하여 각 청크들에 대한 구분을 하는 구조를 갖도록 하였다.

MPBP의 효용성을 검증하기 위하여 JXTA[11] 위에 운용 가능한 MPBP 프로토콜 엔진이 설계되었다. 이 엔진은 여러 서버에 흩어져있는 미디어 데이터에 대한 구분(identification) 기능과 대상(target) 미디어 데이터에 대한 검색 기능을 제공한다. 구현된 엔진의 기능과 성능을 테스트하기 위해, 전송성능(throughput)과 접속 단절에 대한 면을 평가하였다. 평가 결과는 기존 방식에 비해 MPBP의 전송성능이 향상되었고, 접속 단절시 안정적인 스트리밍을 제공할 수 있다는 것을 확인하였다. 또한, MPBP는 다양한 미디어 렌더링 모듈과 연동되도록 설계되어, 다양한 미디어 스트리밍 응용의 하부 구조로 사용될 수 있다. 이러한 유연성을 보여주기 위해 오디오 스트리밍 응용인 AoN (Audion on Net)과 비디오 스트리밍 응용인 VoN (Video on Net)을 구현하여 MPBP와 결합하였다. 두 개의 응용은 하부 네트워크는 MPBP를 이용하고, 단지 사용자 인터페이스와 미디어 렌더링 모듈만을 구현하였음에도 안정적인 미디어 스트리밍을 제공함을 보여주었다.

본 논문의 구성은 다음과 같다. 2장에서는 다중 피어 결합의 개념과 MPBP 프로토콜의 구조에 대해서 설명한다. 3장에서는 MPBP 프로토콜 엔진과 렌더링 모듈의 구현에 대해서 논한다. 4장에서 구현된 엔진에 대한 실험 및 평가 결과를 논하고, 5장에서 결론을 맺는다.

II. MPBP 프로토콜 구조

2.1 해쉬를 이용한 분산 컨텐츠 구분

MPBP의 기본적인 개념은 동일한 미디어 파일의 여러 부분을 다중의 피어로부터 동시에 전송받아서 전송 성능과 안정성을 확보하는 것이다. 이와 비슷한 개념으로는 HTTP 프로토콜 1.1 버전의 Content-Range [12] 기능이다. 이 기능은 웹 서버에서 파일을 다운로드 받을 때, 클라이언트가 파일을 여러 부분으로 나누어 동시에 다운로드 하여 속도를 높이는 기능이다. 이러한 기능이 가능한 것은 대상이 되는 파일이 하나의 서버에 존재하기 때문에, URL을 이용한 유일한 파일의 구분자(identifier)를 얻을 수 있기 때문이다. 그러나 클라이언트/서버 아키텍처와 달리 P2P 네트워크에서는 같은 파일도 여러 개의 피어에 다른 이름으로 존재하는 것이 가능하며, 반대로 같은 이름을 가진 파일도 다른 내용을 가질 수 있다. 따라서 피어간의 단일 접속을 통하여 파일을 전송하는 것은 큰 문제가 없지만, 여러 피어에 흩어져있는 동일한 파일을 구분하여 파일의 부분들을 전송받는 것은 어려운 문제가 된다. 따라서 MPBP의 개념을 적용하기 위해서는 P2P의 특성을 고려한 미디어 파일의 구분(identification) 기능이 필요하게 된다.

이러한 문제를 해결하기 위하여 본 논문에서는 MD5 [10] 해쉬 알고리즘이 사용되었다. MD5는 일방향 해쉬 함수(one way hash function)로서 입력된 데이터에 대해서 128비트의 고유값을 생성하는 알고리즘이다. MPBP 엔진은 자신의 파일 저장소에서 새로운 파일을 찾을 때마다 해당 파일에 대해서 MD5를 이용해서 128비트 값의 File-Hash를 생성하게 된다. 이 File-Hash는 오로지 입력된 데이터에 대해서만 고유값이 나오게 되며, 파일이름 등의 다른 속성에는 영향을 받지 않기 때문에, 파일의 구분자로 사용이 가능하다. 앞에서 지적한 것처럼 동일한 파일들이 여러 피어에 분산될 수 있기 때문에 각 피어에서 해당 파일의 이름이나 속성이 변경될 수 있지만, 해당 파일의 내용은 변경되지 않기 때문에 파일의 이름보다는 File-Hash로 해당 파일을 정확히 구분할 수 있게 된다.

2.2 가상 세션의 구조

MPBP는 여러 피어로부터의 접속을 하나의 가상 세션으로 결합시키는 구조로 되어 있다. 그림 1에서 볼 수 있는 것처럼 이 가상 세션은 여러 피어로부터의 동시 접속으로 구성되어 있다. 가상 세션 내의 각 접속은 스트리밍을 위해서 데이터의 순서대로 다른 피어로 연결되어 있다. 이러한 구조를 이용하면, 여러 피어에서 동시에 데이터가 전달되기 때문에 전송성능이 향상되며, 만일 미디어 송신 피어가 중간에 접속을 단절하여도 해당 데이터를 가진 다른 피어와 접속을 하여 데이터를 복구할 수 있게 된다.

가상 세션 내에서의 전송 단위는 64Kbyte의 청크(chunk)이다. 미디어 파일의 일부분을 지정하지 않고, 파일을 부분별로 나눈 청크를 사용한 이유는 전체 P2P 네트워크에서 리소스(resource)의 가용성(availability)을 높일 수 있기 때문이다. 예를 들어 기존 P2P 스트리밍에서는 완

전한 파일을 가지고 있어야만 스트리밍의 송신자 역할로 참여할 수 있지만, 청크를 사용하게 되면 미디어 파일의 일부 청크만 소유한 피어도 송신자 역할을 할 수 있게 되어 전체 P2P 네트워크 차원에서는 가용성이 높아지게 된다. 그림 1에서 볼 수 있는 것처럼, 미디어 송신 피어 #1이 청크 1개만을 갖고 있지만, 송신 피어로서 가상 세션에 참여하여 전체 네트워크의 전송 성능을 향상시킴을 알 수 있다. 마찬가지로 미디어 수신 피어에게 전송된 청크는 청크 파일로 저장되어, 다른 수신 피어의 요청이 있는 경우에 미디어 수신 피어가 송신 피어로서의 역할로 해당 수신 피어에게 청크 파일을 전송하게 된다.

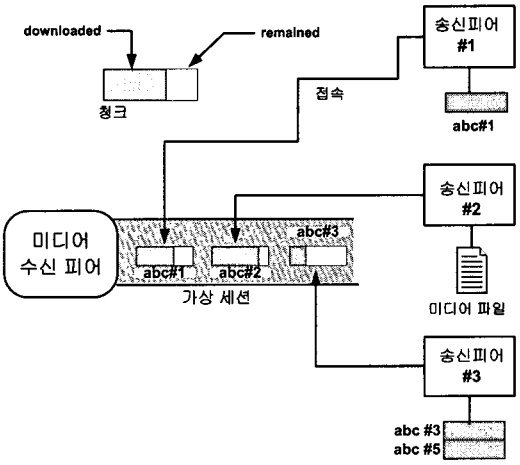

그림 1. 다중 피어의 접속으로 구성된 가상세션
Fig 1. Virtual session consisted of multiple connections to other peers

하나의 미디어 데이터를 여러 개의 조각인 청크로 분할하였을 때, 청크 자체도 여러 피어에 흩어져 있기 때문에 각 청크를 구분할 수 있는 방안이 필요하며, 본 논문에서는 구분자로 Chunk-ID라는 구조를 설계하였다. Chunk-ID는 자신이 속한 파일을 구분하는 File-Hash와 해당 미디어 파일에서 자신이 위치한 블록의 위치를 표시하는 32비트 값의 인덱스 값으로 구성된다. Chunk-ID는 그 자체로 미디어 파일과 그 부분을 명확하게 지정할 수 있기 때문에, 전체 P2P 네트워크에서 다른 청크들과 고유하게 구분되는데 사용된다. 청크 파일의 구조는 그림 2와 같다. 파일 헤더에는 청크 헤더가 들어있고, 원 파일의 크기와 이름이 포함되어 있다. 헤더의 이러한 정보는 쿼리 프로토콜과 전송 프로토콜에 중요하게 사용된다.

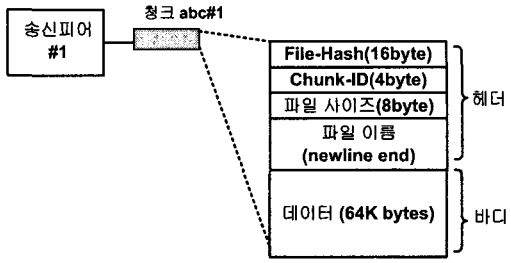

그림 2. 청크의 구조
Fig 2. Structure of chunk

2.3 쿼리 프로토콜의 구조

MPBP를 이용하여 미디어 스트리밍을 하기 위해서는 먼저 대상 미디어 파일을 P2P 네트워크에서 검색해서 미디어 파일을 소유하고 있는 피어에 대한 정보를 얻어야 한다. MPBP에서 이러한 검색은 2 단계로 구성된 쿼리 프로토콜을 이용한다. 쿼리 프로토콜의 첫 번째 단계는 "파일 검색"이고, 두 번째 단계는 "청크 정보 수집"이다.

2.3.1 파일 검색

파일을 검색하기 위해서는 파일 이름이 포함된 키워드(keyword) 검색이 사용된다. 스트리밍을 시작하고자 하는 미디어 수신 피어는 그림 3처럼 파일 이름의 일부가 포함된 키워드와 미디어 송신 피어로부터 응답을 받기 위한 데이터 수신 채널의 정보를 브로드캐스트(broadcast)하게 된다. 이때 데이터 수신 채널은 다른 미디어 송신 피어의 응답을 수집하기 위한 용도로 사용된다. 이 브로드캐스트 메시지를 받은 다른 피어들은 해당 파일이 자신이 소유한 파일이나 청크에 해당하는지를 확인하고, 적합한 파일이나 청크가 발견되면 대상 파일의 이름, 크기, 파일 해쉬로 구성된 응답을 보내게 된다. 주어진 키워드에 적합한 여러 개의 파일이 존재할 수 있기 때문에, 응답은 여러 개의 결과가 하나의 패킷으로 묶여진 형태로 전송되게 된다. 이렇게 여러 파일들을 포함하는 응답이 필요한 이유는 키워드를 포함하는 여러 개의 파일이 존재할 수 있으므로, 해당 키워드가 포함된 파일들을 사용자들에게 보여주고, 사용자들이 선택할 수 있도록 정보를 제공해야 하기 때문이다. 이 단계의 프로토콜 흐름은 그림 3과 같다.

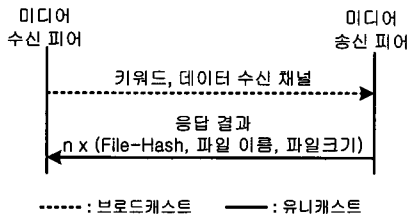

그림 3. 파일 검색의 프로토콜 흐름
Fig 3. Protocol flow for file searching

2.3.2 청크 정보 수집

실제 스트리밍을 위해서는 파일 자체에 대한 정보보다는 청크를 소유한 피어에 대한 정보가 필요하게 된다. 따라서 어떤 피어들이 필요한 파일에 대한 청크를 소유하고 있는지에 대한 "청크 정보 수집"단계가 반드시 필요하다. 이를 위해 미디어 수신 피어는 검색 단계에서 얻은 File-Hash를 브로드캐스트하게 된다. 파일 이름 대신에 파일 해쉬를 이용하는 이유는 앞에서도 지적한 것처럼, 파일 이름이 각 피어에서 자주 변경될 수 있기 때문에 보다 정확한 구분이 가능한 파일 해쉬를 이용하는 것이다. 한편, 브로드캐스트된 File-Hash에 대한 정보를 받은 미디어 송신 피어들은 File-Hash에 해당하는 파일이나 청크를 갖고 있는 경우에, 청크들의 리스트와 파일 해쉬, 그리고 해당 청크를 받을 수 있는 송신 피어의 데이터 송신 채널에 대한 정보를 응답해 주게 된다. 데이터 송신 채널은 다른 미디어 수신 피어들이 해당 청크를 다운로드 하는데 사용되는 통신 채널이다. 이 단계의 프로토콜 흐름은 그림 4와 같다.

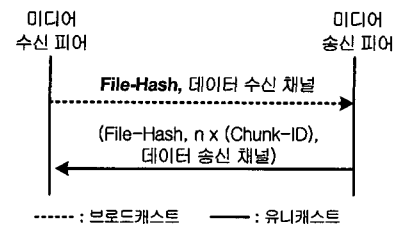

그림 4. 청크 정보 수집의 프로토콜 흐름
Fig 4. Protocol flow for collecting chunk information

2.4 청크의 전송과 전송 에러 복원

청크 정보수집 단계에서 모든 정보를 얻은 후, 미디어 수신 피어는 내부적으로 다운로드 스케줄 테이블(DST: Downloading Schedule Table)을 만든다. 그림 5에서 볼 수 있는 것처럼 DST는 미디어 송신 피어의 데이터 송신 채널에 대한 정보들을 연결 리스트로 유지하는 테이블이다. 미디어 수신 피어는 각 청크를 소유하고 있는 미디어 송신 피어의 정보를 이용해서 여러 개의 커넥션을 하나의 세션으로 묶을 수 있게 된다. 그림 5의 경우를 보면, 1번째 청크의 경우에는 3개의 데이터 송신 채널에서 얻을 수 있으며, 2번째 청크의 경우에는 2개의 데이터 송신 채널에서 얻을 수 있음을 알 수 있다.

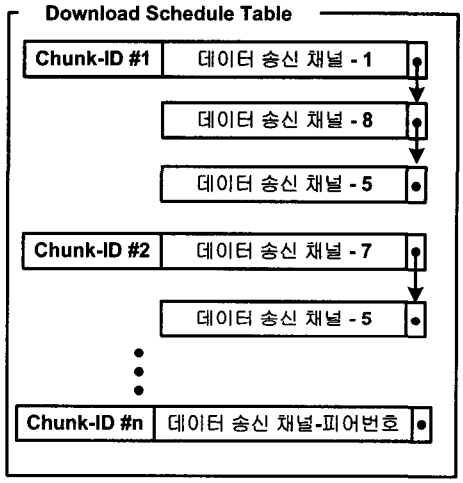

그림 5: DST의 구조
Fig 5. Structure of DST

기본적으로 미디어 수신 피어는 가상 세션 내의 다운로드되는 각 청크를 DST의 정보를 이용하여 각각 다른 피어들의 데이터 송신 채널을 이용하도록 구성하여 가상적으로 전송성능을 높이게 된다. 또한 전송 중의 송신 피어가 접속을 단절하게 되면, DST의 정보를 이용하여 해당 청크를 가진 다른 송신 피어에 접속을 하여 데이터를 복원하게 된다. 예를 들어 그림 5에서 만일 청크 2번을 전송하던 7번 송신 피어가 접속을 단절한 경우에, 미디어 수신 피어는 해당 연결을 5번 피어의 데이터 송신 채널을 이용하여 복원하게 된다. 즉 가상 세션 내의 접속이 끊어지게 되면 미디어 수신 피어는 DST 내의 다른 피어들의 접속 정보를 이용해서 자동으로 미디어 스트리밍을 유지하게 된다.

III. 프로토콜 엔진과 렌더링 모듈의 구현

3.1 프로토콜 엔진의 구현

MPBP 프로토콜의 효용성을 검증하기 위하여, JXTA [11] 네트워크 위에 MPBP 프로토콜 엔진을 구현하였다. MPBP 프로토콜에서 검색 기능을 제공하기 위해서는 P2P 네트워크에 검색 요청을 하는 브로드캐스트 채널과 검색에 대한 응답을 수집하는 유니캐스트(unicast) 채널이 준비되어야 한다. JXTA 네트워크에서는 브로드캐스트 통신을 위하여 Propagate Pipe [11]라는 기능이 제공된다. Propagate Pipe는 JXTA 피어가 등록을 해두면 해당 채널로 전달된 정보를 피어에게 전달해주게 된다. 유니캐스트 기능을 제공하기 위해서는 JXTA Unicast Pipe [11]를 사용하다. Unicast Pipe는 자신의 채널 정보를 Pipe Advertisement [11]로 구성하여 검색 요청을 브로드캐스트할 때 같이 전달하게 된다. Pipe Advertisement를 얻은 피어들은 해당 Unicat Pipe를 통하여 유니캐스트 통신을 할 수 있게 된다.

그림 6은 MPBP 프로토콜 엔진의 구조를 나타내고 있다. MPBP 엔진 내의 쿼리 처리기(Query Processor) 부분은 JXTA의 통신 기능을 이용하여 유니캐스트와 브로드캐스트 통신을 처리할 수 있는 구조로 구성되어 있다. 쿼리 처리기는 쿼리 응답기(Query Responder)와 응답 수집기(Response Collector)로 이루어져 있는데, 쿼리 응답기는 외부의 검색 요청에 대해서 응답을 해주는 모듈이고, 응답 수집기는 외부의 검색 요청에 대한 응답을 수집하여 상위 응용에 알려주는 모듈이다. 그림 6에서 볼 수 있는 것처럼 두 모듈의 Unicast Pipe와 Propagate Pipe의 전송 방향이 목적에 따라 반대 방향인 것을 알 수 있다.

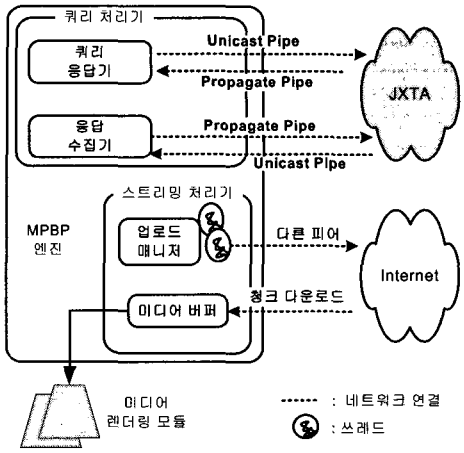

그림 6. 구현된 MPBP 프로토콜 엔진의 구조
Fig 6. Structure of implemented MPBP protocol engine

검색 기능 외에 실제로 스트리밍을 위해서는 미디어 파일을 수신하거나 송신할 수 있어야 한다. 이 기능을 위해서 MPBP 프로토콜 엔진에서는 스트리밍 처리기(Streaming Processor)가 제공된다. 스트리밍 처리기는 업로드 매니저(Upload Manager)와 미디어 버퍼(Media Buffer)의 두 모듈로 구성되어 있다. 업로드 매니저는 다른 미디어 수신 피어에서 요청된 청크의 전송을 처리해주는 모듈이며, 다른 피어에게는 스트리밍 서버의 역할을 하게 된다. 이를 위해서 TCP/IP의 서버 소켓의 형태를 가지며, 청크 정보 수집 단계에서 응답을 보낼 때 전송되는 데이터 전송 채널 정보에 피어의 IP 주소와 포트 번호를 담아서 넘겨주게 된다.

미디어 버퍼는 외부의 피어에서 청크를 받아와서 미디어를 렌더링(rendering)하는 모듈에게 데이터를 제공해주는 역할을 한다. 미디어 버퍼는 네트워크 쪽에서는 여러 피어로부터 전송된 청크들을 조립하여 가상 세션을 만드는 기능을 제공한다. 그리고 동시에 미디어 렌더링 모듈 쪽에는 입출력 버퍼링 기능을 제공해주도록 구성하였다. 따라서 미디어 렌더링 모듈 입장에서는 하드 디스크나 CD-ROM과 같은 입출력 디바이스와 동일한 접근이 가능하다. 이를 위해서 미디어 버퍼는 Java Input Stream [13] 클래스에서 상속받은 하위 클래스로 구현되었다.

3.2 미디어 렌더링 모듈

MPBP의 또 하나의 특징은 렌더링을 하고자 하는 미디어의 종류에 상관없이 스트리밍 기능을 제공해준다는 점이다. 일반적으로 하나의 프로토콜이나 응용이 모든 종류의

파일 형식에 효율적인 스트리밍을 제공하는 것은 거의 불가능하다. 따라서 미디어 렌더링 구조와 스트리밍 기능을 분리하는 것이 P2P 스트리밍에서는 바람직하다. 이러한 분리는 하나의 프로토콜을 이용해서 여러 종류의 미디어 렌더링 응용들이 동시에 운용될 수 있는 구조를 제공한다. 이것은 또한 하나의 P2P 네트워크에서 여러 개의 다른 미디어 형태를 충돌 없이 운용할 수 있게 해준다. 미디어 렌더링 모듈을 쉽게 결합할 수 있도록 하기 위해, MPBP는 미디어 버퍼를 제공한다. 미디어 버퍼는 입출력 디바이스와 유사한 기능을 제공하므로, P2P 미디어 스트리밍을 지원하고자 하는 응용들은 단지 미디어 버퍼를 입력 소스(source)로 연결하기만 하면 스트리밍 기능을 이용할 수 있다. 본 논문에서는 오디오 스트리밍 응용인 AoN (Audio on Net)과 비디오 스트리밍 응용인 VoN (Video on Net)을 만들었다. AoN은 Java Sound API [14]를 이용하여 오디오 렌더링 모듈을 구현하였으며, VoN은 Java Media Framework [15]을 이용하여 비디오 데이터를 렌더링 하였다. 두 응용의 실행 모습은 그림 7과 같다.

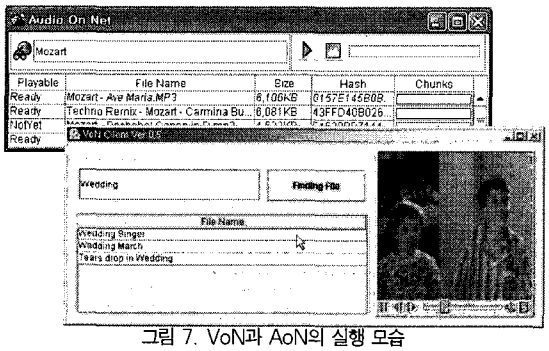

그림 7. VoN과 AoN의 실행 모습
Fig 7. Snapshots of VoN and AoN

IV. 실험 및 평가

4.1 전송 성능 평가

구현된 시스템의 전송 성능(throughput)을 평가하기 위하여, 100 Mbps의 LAN 환경에서 2대의 컴퓨터에 AoN 응용을 운용하였다. 응용이 설치된 한 지점에서 4.2M 바이트 크기의 오디오 파일을 위치시켰다. 기존 단일 피어를 이용한 스트리밍과 MPBP를 이용한 스트리밍의 전송 성능(throughput)을 평가하기 위하여, 미디어 버퍼의 연결 개수를 1개에서 3개까지 변화시키면서 실험을 진행하였다. MPBP에서 연결 개수가 1개인 경우는 일반적인 단일 피어 간의 스트리밍 응용과 같은 성능을 보여주게 된다. 평가는 총 5회에 걸쳐서 반복되었으며, 그 결과는 그림 8과 같다. 결과에서 볼 수 있듯이, 연결 개수가 늘어나게 되면 예상된 바와 같이 전송 성능이 향상됨을 볼 수 있다.

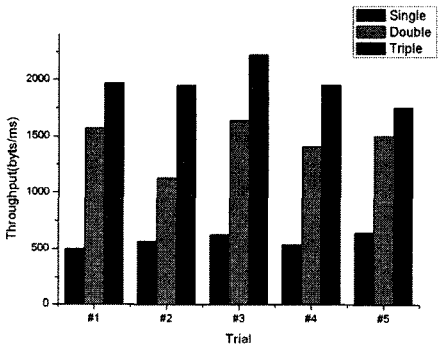

그림 8. 전송 성능 평가 결과
Fig 8. Results of throughput evaluation

4.2 접속 단절에 대한 안정성

구현된 시스템을 평가하기 위하여, 100 Mbps의 LAN 환경에서 5대의 호스트에 AoN 응용을 운용하였다. 응용이 설치된 호스트 A,B,C,D에 4,901,117 바이트 크기의 동일한 오디오 파일을 위치시켰고, 파일이름을 달리하였다. 나머지 호스트 E에서 AoN 응용에서 대상 오디오 파일을 스트리밍으로 재생하였다. 미디어 데이터를 갖고 있는 A,B,C 호스트의 AoN을 1분 단위로 순서대로 중지시켰다. 호스트 E에서는 대상 오디오 파일을 갖고 있는 호스트가 D만 남은 상황에서도 오디오 스트리밍이 정상적으로 진행되었다.

그림 9는 미디어 스트리밍을 진행하는 호스트 E 입장에서 전송된 패킷의 송신지를 나타낸다. 초기 10개의 청크는 4개의 피어에서 골고루 전송되지만, 시간이 지나서 피어의 접속이 단절되면 남아있는 피어들에게서 전송되는 청크들이 많아지게 된다. 그림 9의 그래프에서 보면 60개 청크의 전송이 일어나는 시점에서는 단 두 개의 피어에서만 청크가 전송되는 것을 볼 수 있다. 이 결과는 세션에서 남아있는 피어들이 접속이 끊어진 피어들을 대신하여 스트리밍의 소스 역할을 하는 것을 알 수 있다. 이 결과는 MPBP를 이용

함으로써 여러 피어가 결합하여 미디어 스트리밍을 진행하는 피어에게 데이터를 제공할 수 있음을 나타내며, 피어간의 갑작스런 단절을 유연하게 해결할 수 있음을 보여준다.

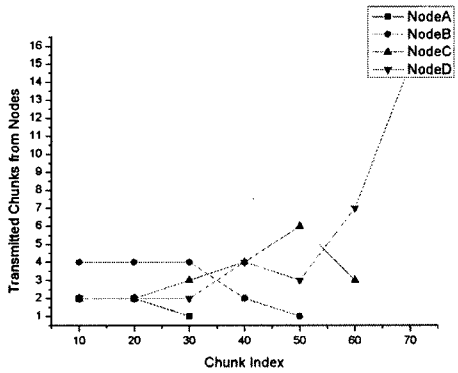

그림 9. 전송된 청크의 소스
Fig 9. Sources of transmitted chunks

V. 결론

P2P 미디어 스트리밍은 P2P 분야에서 큰 발전이 예상되는 분야이나, 기존의 P2P 환경에서는 피어간의 낮은 접속 신뢰와 피어간의 대역폭의 한계 등으로 인하여 미디어 스트리밍이 실현되기 어려웠다. 본 논문에서는 이를 극복하기 위하여 다중 피어 결합 프로토콜인 MPBP (Multi-Peer Binding Protocol)를 제안하였다. MPBP는 미디어 데이터를 가진 다중의 피어들을 결합하여 스트리밍의 전송 효율을 높이고, 피어간의 접속 단절 문제를 해결하기 위해 설계된 프로토콜이다. 설계된 MPBP의 성능을 평가하기 위하여, JXTA 네트워크 상에서 MPBP 엔진을 구현하였으며, 구현된 MPBP 엔진은 다양한 미디어 스트리밍 응용과 결합될 수 있도록 설계되었다. 본 논문에서는 오디오 응용을 위하여 AoN, 비디오 응용을 위하여 VoN을 구현하여 결합하였다. 구현된 응용 들을 이용하여 전송 성능과 안전성을 평가하였으며, 평가 결과는 MPBP가 기존 P2P 스트리밍 방식의 문제점인 급작스런 접속 단절과 대역폭 한계에 대해 적절하게 대처할 수 있음을 보여주었다.

향후에는 다양한 미디어의 특성에 적합한 스트리밍을 지원하기 위해 전송 패킷의 크기를 사전에 조율할 수 있는 통신 구조에 관해서 연구가 진행되어야 할 것이다.

참고문헌

[1] Abdel-Baki.N, Aumann.B, Grossman.H.P. "Analyzing Multimedia Streaming in a Distributed Environment," ECUMN 2002, pp.56-63, Apr.2002

[2] Gebhard.H, Lindner.L, "Virtual Internet Broadcasting," IEEE Communication Magazine, vol.39, no.6, pp.182-188, Jun.2001

[3] Yuqing.S, Mielke.M, Aidong.Z, "NetMedia: synchronized streaming of multimedia presentations in distributed environments," Multimedia Computing and Systems, vol.2, pp.585-590, Jun.1999

[4] 홍성준, 이용수, "자치적응성 콘텐츠 서비스 네트워크," 한국컴퓨터정보학회논문지 제 9권 3호. pp.149-155, 2004

[5] Andy.O (Ed.), "Peer-to-Peer: Harnessing the Power of Disruptive Technologies," O'reilly, Mar.2001

[6] Detlef.S, Kai.F, "Peer-to-Peer Prospects," CACM, vol.46, no.2, pp.27-29, Feb.2003

[7] PeerCast: p2p broadcasting for everyone, web site, "http://www.peercast.org"

[8] Streamer: p2p Internet relay radio, web site, "http://www.chaotica.u-net.com/streamer.htm"

[9] 김분희, 이준연, "개선된 노드 분산율을 위한 적응적 P2P 검색 알고리즘," 한국컴퓨터정보학회논문지, 제 10권 4호. pp.93-102, 2005

[10] Rivest, R.L., "The MD5 message-digest algorithm," RFC-1321, 1992

[11] Joseph.D.G, Joe.G, "Mastering JXTA: Building Java Peer-to-Peer Applications," Aug., 2002, SUN official document, "Java Sound Programmer Guide," http://java.sun.com/j2se/1.4.2/docs/guide/sound/programmer_guide/contents.html

[12] Fielding.R., Gettys.J., Frystyk.H., Masinter.L.,

Leach.P., Berners-Lee.T, "Hypertext Transfer Protocol - HTTP/1.1," RFC-2616, 1999
[13] Harold.E.R.,"Java I/O," O'reilly, 1999
[14] Craig. A. L.,"Digital Audio with Java, "Prentice Hall, 1999
[15] Rob,G., Stephen,T., "Essential JMF - Java Media Framework," Prentice Hall, 1998

저 자 소 개

정 의 현

1999년 2월 : 한양대학교 전자공학박사
2004년 ~ 현재 : 안양대학교 디지털미디어 공학과 전임강사
관심분야 : 시맨틱 웹, 디지털 컨버전스