SDS : Sequential Download System을 이용한 P2P 미디어 스트리밍

박병철* · 노선식** · 이동은*

* 청운대학교 인터넷학과 · ** 광주대학교 정보통신학과

A P2P Media Streaming by SDS : Sequential Download System

Byung-chul Park* · Sun-Sik Roh* · Dong-eun Lee**

* Dept. of Internet, Chungwoon University · ** Dept. of info & com engineering, Gwangju Univ.

E-mail : dejavu-kr@nate.com, ssroh@gwangju.ac.kr, delee@chungwoon.ac.kr

요 약

본 논문에서는 P2P 네트워크 상에 분산되어 있는 파일들을 수집하고 피어의 신뢰도가 낮은 접속에도 끊김 없는 스트리밍 서비스를 제공하기위해 Sequential Download System(SDS)을 제안하였다. SDS는 Download Manager를 이용하여 여러 피어로부터 파일을 수신하고, 파일 수신 시에 피어의 갑작스런 이탈에 대처하며 스트리밍 시 발생하는 버퍼링 시간을 최소화하여 안정된 스트리밍을 제공하게 된다. 또한 Memory map을 이용하여 다른 피어의 요청에도 대처할 수 있다.

키워드

Peer-to-Peer(P2P), Media Streaming

Ⅰ. 서 론

Peer-to-Peer(P2P) 기술은 파일공유시스템으로써 인트라넷 및 홈 네트워크 등 많은 분야에서 활용되고 있다. 그중에서도 P2P 미디어 스트리밍 서비스는 미디어 파일을 다운로드하지 않고도 재생할 수 있다는 이점으로 인하여 실시간 재생 서비스 등에서 많이 활용되고 있다. 하지만 피어간의 서로 다른 대역폭과 전송속도 그리고 신뢰도가 낮은 접속 등으로 인하여 P2P 네트워크 상에서 안정된 스트리밍 서비스를 제공하는데 문제가 발생한다. 또한 미디어 파일들이 P2P 네트워크 상에 분산되어있기 때문에 이런 미디어 파일들을 찾아 스트리밍에 활용해야 하는 문제점이 있다.

기존의 클라이언트/서버 모델에서 대량의 정보를 다운로드하기 위해 많은 사용자가 접속하면 서버는 병목현상을 일으킨다. P2P 모델과는 대조적으로 사용자들이 네트워크 상에서 서로 과다한 요청을 주고받으면 관리자의 업무가 늘어나고 네트워크 자체에 부하를 준다. 클라이언트/서버 모델과는 달리, P2P 모델의 상당한 강점은 서버에 대한 의존도가 낮다는 것과 서버로부터 중앙 집중제어를 받지 않는다는 것이다.

P2P 모델은 연결방식에 따라 크게 두 가지로 분류될 수 있다. 하나는 순수 P2P(Pure P2P) 모델로서 어떠한 중앙 서버에도 의존하지 않고 전적으로 개개의 컴퓨터들에 의존하여 작동한다. Gnutella 계열이 순수 P2P 모델에 속한다[1]. 다른 하나는 혼합형 P2P(Hybrid P2P) 모델이다. 혼합형 P2P 모델은 중앙 서버가 존재하지만 최소한의 검색 정보(피어들의 IP 정보 및 공유목록)만을 제공한다. Napster 계열이 혼합형 P2P 모델에 속한다[2]. 이러한 P2P는 파일 공유, 컨텐츠 분배, 그리고 분산 컴퓨팅 등 다양한 분야에서 활용되고 있다. 최근 들어 인터넷상에서 사용자들이 기존의 텍스트 기반의 컨텐츠가 아닌 UCC나 실시간 방송 등 고대역의 미디어 컨텐츠를 선호하면서 미디어 스트리밍에 대한 관심이 증가하고 있다. 그러나 이러한 관심에도 불구하고 P2P 네트워크 상에서 스트리밍 서비스를 제공하고 있던 피어의 갑작스런 이탈로 인하여 스트리밍 서비스가 중단되는 등의 문제가 발생되고 있다. 이에 중단 없는 스트리밍 서비스에 대한 연구가 필수적이다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 살펴보고, 3절에서는 제안한 시스템의 구조에 대해 기술한다. 그리고 이어 4절에서는 제안한 시스템을 이용한 P2P시스템의 구현, 5절에서는 결론을 기술한다.

II. 관련 연구

기존의 P2P 컨텐츠 공유 시스템과 P2P 미디어 스트리밍 시스템은 데이터 공유 모드에 따라 미디어 파일을 모두 다운받은 후에 재생을 하는 'Open-After-Downloading' 방식과 미디어 파일을 다운로드하는 동시에 플레이어에서 재생하는 'Play-While-Downloading' 방식으로 구분된다[3]. 'Play-While-Downloading' 방식은 다운로드와 재생을 동시에 할 수 있는 강점이 있으나 미디어 파일을 전송하던 피어의 이탈 등으로 인하여 재생이 끊기는 문제가 발생한다. 이들 문제점에 대한 해결책 중의 하나는 하나의 피어로부터 미디어 파일을 전송받는 것이 아닌 여러 피어로부터 미디어 파일을 전송받는 멀티소스 미디어 스트리밍 기법이다. 이에 관한 연구로는 GNUSTREAM[4], MPBP[5]등이 있다.

GNUSTREAM[4]은 기존 방식의 P2P 네트워크에서 스트리밍 시 발생하는 문제점을 해결하고자 Gnutella 기반으로 구현된 멀티소스 미디어 스트리밍 기법이다. GNUSTREAM은 다중 송신기 대역폭 집약, 적응 버퍼 제어, 피어 장애 또는 이탈 감지, 스트리밍 품질 유지의 특징을 가지고 있다. 하지만 전송된 미디어 컨텐츠를 스트리밍 이후에 파일로 저장하지 않고 소비해버리며 스트리밍 중인 피어가 가진 미디어파일의 일부분을 전송하는 메커니즘을 가지고 있지 않다.

MPBP(Multi-Peer Binding Protocol)[5]는 동일한 미디어 파일의 여러 부분을 다중의 피어들로부터 동시에 전송받을 수 있도록 하는 기법이다. 특징으로는 미디어파일을 청크의 단위로 나누어 전송하도록 되어 있어 미디어 파일의 일부분만 보유하고 있는 피어도 전송에 참여할 수 있다. 또한 DST(Download Schedule Table)를 두어 전송 피어들의 목록을 유지/관리한다. 하지만 전송피어의 시스템 및 네트워크의 환경에 따라 미디어 파일을 전송하는 속도에 차이가 있을 수 있다. 만약 스트리밍 시 미디어 파일을 전송하던 일부 피어의 전송속도가 네트워크나 시스템 장애로 인해 저하되어 재생에 필요한 부분을 전송받지 못하게 된다면 수신피어는 전송받을 때까지 버퍼링 상태로 남아 있어야하는 문제가 발생하게 된다.

III. Sequential Download System의 설계

본 논문에서는 빠르고 안정적인 스트리밍 서비스를 제공하고자 Sequential Download System (SDS)을 제안하였다. SDS는 다중의 피어들로부터 미디어 파일을 동시에 전송받음으로써 스트리밍 초기에 발생하는 버퍼링 시간을 최소화하여 빠르고 안정적인 스트리밍 서비스가 가능하게 하고 또한 데이터를 순차적으로 다운로드함으로써 스트리밍 시 발생되는 버퍼링을 최소화하여 스트리밍 QoS를 높일 수 있다. 3장에서는 제안한 시스템의 특징과 기본 동작에 대해 기술한다.

3.1 Sequential Download System의 특징

① 피어들 간의 XML을 이용한 통신

본 애플리케이션에서는 각 피어들 간의 통신에 있어서 XML을 사용하였다. XML은 요소 변경이 용이하기 때문에 P2P 네트워크에서 피어들의 메시지 교환 시 속성 값만 쉽게 변경하여 전달할 수 있다. 또한 XML로 메시지를 교환하게 되면 각자 가지고 있는 P2P 애플리케이션이 다른 프로그램 언어로 구성되더라도 서로 연동이 가능하기 때문에 다양한 프로그램언어로 구성된 P2P 애플리케이션을 P2P 네트워크 상에 배치할 수 있게 된다.

② 해쉬를 이용한 미디어파일의 구분

P2P 네트워크가 아닌 클라이언트/서버 구조에서는 사용자가 원하는 파일이 서버에 하나의 이름으로 존재하기 때문에 파일을 여러 부분으로 나누어 동시에 다운로드하여도 하나의 완전한 파일을 생성하는 것은 아무런 문제가 되지 않는다. 그러나 P2P 네트워크의 경우에는 하나의 파일이 여러 피어 상에 다른 이름으로 존재할 수 있고, 반대로 같은 이름을 가진 파일이라 할지라도 서로 다른 내용을 가질 수 있다. 이런 환경에서 파일의 이름만 가지고 여러 피어로부터 동일한 파일을 다운로드한다는 것은 문제가 될 수 있다. 따라서 P2P 네트워크에서 피어가 찾는 정확한 파일을 다운로드하기 위해서는 파일을 구분할 수 있는 기능이 필요하다.

이러한 문제를 해결하기 위하여 본 논문에서는 MD(Message Digest)5[6] 해쉬 알고리즘을 사용하였다. MD5는 메시지 요약 알고리즘으로, 128비트의 메시지 요약을 생성해내는 알고리즘이며 다른 해쉬 알고리즘보다 수행속도가 빠르고, 쉬우며, 간결하다는 특징을 가지고 있다. P2P 애플리케이션은 공유폴더에서 새로운 파일을 찾을 때마다 MD5 해쉬 알고리즘을 이용하여 파일에 대한 128비트 값의 File_Hash를 생성하게 된다. 이렇게 생성된 File_Hash를 이용하여 여러 피어들에게 존재하는 파일들을 구분하게 된다. File_Hash를 이용하기 때문에 여러 피어에 흩어져있는 동일한 파일들을 다운로드하거나, 파일의 이름이 다른 동일한 내용의 파일을 다운로드 하는 것에는 아무런 문제가 되지 않는다.

③ 파일의 전송

파일 전송 시 하나의 송신피어로부터 파일 전체를 수신하게 된다면 다운로드 시 많은 시간을 소비하게 되며, 송신피어가 P2P 네트워크에서 이탈 시 전송 중이었던 송신피어가 재접속하기를 기다리거나 해당 파일을 보유한 다른 피어에게서 파일을 처음부터 다시 전송받아야하는 문제가 발생하게 된다. 이러한 문제를 해결하기 위해 전체 파일을 하나의 피어에게서 전송받는 방법이 아닌 하나의 파일을 일정한 크기로 나눠서 다중 피어로부터 전송받는 방법인 다중 분할 전송방법을

사용하였다. 파일을 나눠서 전송받게 되면, 하나의 피어가 아닌 여러 피어로부터 동시에 전송받을 수 있기 때문에 빠른 다운로드가 가능하며, 또한 전송도중 송신피어가 P2P 네트워크에서 이탈하게 된다 하더라도 이탈한 송신피어가 전송하던 부분부터 재전송 받으면 되기 때문에 파일을 처음부터 다시 받아야하는 문제를 해결할 수 있게 된다. 파일을 분할하는 단위로는 논문 [5]에서 제시한 64Kbyte의 청크를 사용하였다. 청크는 File_Hash, Chunk_ID, File_Size, File_Name 그리고 Data로 구성되어있다. Chunk_ID는 자신이 속한 파일에서 자신이 위치한 블록의 위치를 32bit의 인덱스 값으로 나타낸다.

④ Seamless한 스트리밍

다수의 피어로부터 전송되는 청크들은 피어의 네트워크 환경에 따라 전송속도가 다르기 때문에 다운로드가 완료되는 속도가 서로 상이하다. 이로 인해 버퍼에 저장되는 청크들의 순서가 섞이게 되어 스트리밍 시 다음 재생에 필요한 청크들을 순서에 맞게 정렬해야 하기 때문에 버퍼링이 발생하게 된다. 버퍼링의 횟수가 많아지게 되면 Seamless한 재생이 불가능하게 되어 스트리밍 QoS가 떨어지게 되는 문제가 발생하게 된다. 이를 해결하기 위해 다운로드 속도가 서로 다른 청크들을 재생순서에 맞게 정렬하여 재생 시 다음 재생에 필요한 청크를 찾는 시간을 줄임으로서 버퍼링을 최소화할 수 있게 해주는 기능이 필요하다. 본 논문에서는 청크들을 버퍼에 저장하기 전에 청크들의 Chunk_ID를 비교하여 청크들을 순서에 맞게 정렬함으로써 Seamless한 재생이 가능하도록 하는 Sequence controller를 제안하였다.

Sequence controller는 다운로드 매니저의 Download scheduler와 함께 기능을 수행하여 빠르고 안정적인 스트리밍을 제공하도록 하여 스트리밍 QoS를 높이는 역할을 담당한다.

3.2 Sequential Download System

SDS는 P2P애플리케이션을 실행 시 자신의 공유폴더에 저장되어있는 모든 파일을 MD5 해쉬 알고리즘을 이용하여 File_Hash를 생성하고 이를 Share list에 저장해 놓는다. 검색 요청이 들어오게 되면 검색 조건과 일치하는 파일을 Share list에서 찾아 File_Name, File_Size, File_Type, File_Hash를 응답 메시지와 함께 전송한다. 응답 메시지를 받은 피어는 Download manager의 Download Scheduler에 의해서 응답 메시지를 전송한 각 피어들의 응답 속도를 측정하게 된다. 피어는 검색 리스트 중에서 자신이 원하는 파일을 찾아 다운로드 및 스트리밍 서비스를 요청하게 된다. Download Scheduler를 요청 메시지를 보내기 전에 해당 파일에 대해서 Download schedule table을 생성한다. Download Schedule Table을 이용하여 해당파일을 보유한 피어에게 File_Hash와 Chunk_ID, 그리고 전송채널이 포함된 요청 메시지를 전송한다. 전송 피어는 동일한 File_Hash를 가진 파일을 청크 단위로 분리하여 해당 Chunk_ID에 속하는 청크를 전송하게 된다. 요청 피어는 스트리밍 요청 시 수신한 청크들을 버퍼에 전송하게 되고 버퍼는 미디어 플레이어를 이용하여 재생을 시작하고 재생이 완료된 부분은 Memory map에 저장한다. map offset이 다 체크가 되면 분리되어있는 청크들을 완전한 미디어 파일로 변환하고 스트리밍 서비스를 완료한다.

File_Name	File_Size	File_Type	File_Hash

그림 5 Share list의 구조

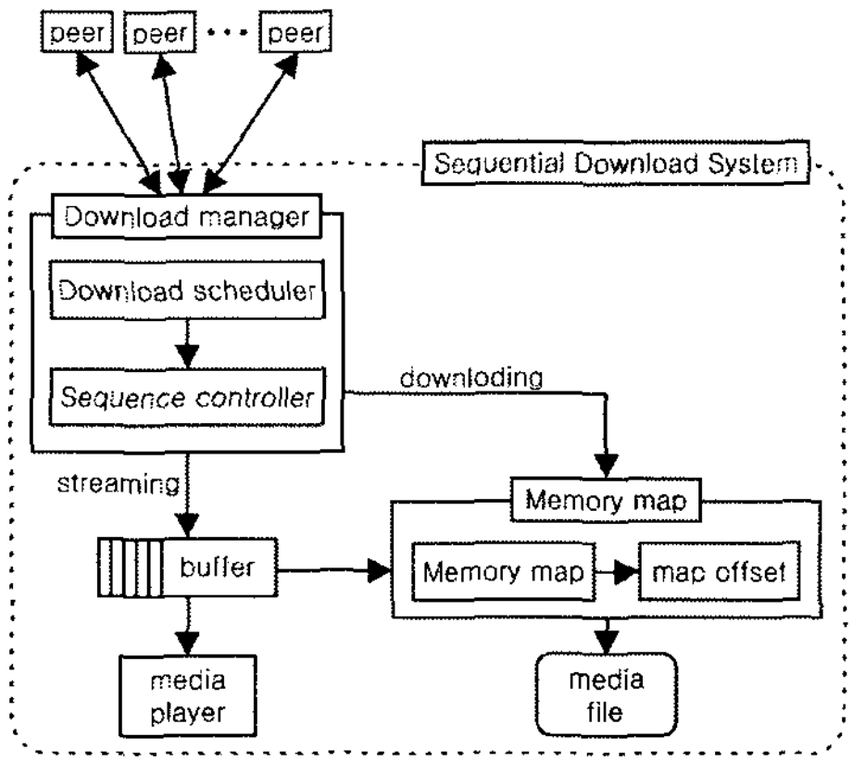

그림 6 Sequential Download System

SDS는 다운로드 매니저(Download manager), 버퍼(buffer), 미디어 플레이어(Media player), 메모리 맵(Memory map)으로 구성되어있다.

① 다운로드 매니저(Download manager)

다운로드 매니저는 SDS에서 파일의 다운로드 및 전송 피어의 선정, 순차적인 다운로드를 위한 청크들의 정렬 등을 담당하는 핵심부분이다. 다운로드 매니저는 Download scheduler와 Sequence controller로 구성되어있다.

- **Download scheduler**

Download scheduler는 검색 요청 메시지를 전송할 때 메시지 전송시간을 기록한 $time_{send}$을 요청 메시지의 헤더부분에 포함하여 피어들에게 전송한다. $time_{send}$은 응답 메시지에도 포함되어 검색 요청 피어에게 되돌아오게 된다. 응답 메시지를 수신 후 메시지에 포함된 $time_{send}$을 이용하여 각 피어들에 대한 응답속도($time_{resp}$)를 식(1)과 같이 계산하고 함께 저장해 놓는다.

$$time_{resp} = time_{now} - time_{send} \quad (1)$$

피어가 다운로드 요청을 하게 되면 해당 파일의 File_Hash와 동일한 File_Hash를 보유한 피어들을 그룹화 한다. 그룹화 된 피어들의 응답속도

($time_{resp}$)를 비교하여 응답속도가 빠른($time_{resp}$이 작은) 것부터 나열한다. 그룹화 된 전체 피어(P_t)는 식(2)에 의해 2/3는 전송에 참여하는 송신피어(P_s)가 되고 1/3은 송신피어 이탈시 해당 송신피어와 대체될 대기피어(P_w)가 된다.

$$P_s = \frac{2P_t}{3} \quad , \quad P_w = \frac{P_t}{3} \qquad (2)$$

각 송신피어 마다 전송될 청크의 수(γ)는 식(3)을 이용하여 계산하고 Download schedule table을 구성한다.

$$\gamma = \left\lceil \frac{File_{size}}{64_{Kbyte}} \right\rceil \times \frac{1}{P_s} \qquad (3)$$

Download schedule table을 이용하여 해당 피어로부터 해당 청크들을 다운로드 하게 된다. 다운로드 도중 송신피어가 P2P 네트워크에서 이탈하게 되면 대기피어들 중 한 피어를 이탈한 송신피어와 대체하여 파일을 계속 다운로드 할 수 있도록 한다.

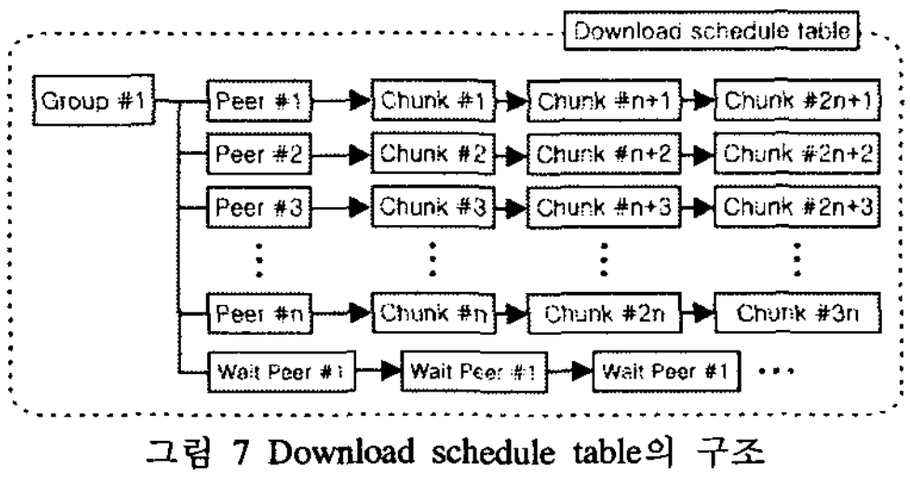

그림 7 Download schedule table의 구조

- Sequence controller

미디어 파일 수신 시 Download schedule table을 이용하여 다중의 피어들로부터 파일의 부분인 청크들을 순차적으로 다운로드할 수 있지만 송신피어의 갑작스런 네트워크 문제 등으로 인하여 전송속도가 떨어져 버퍼에 기록되는 순서가 바뀔 수 있게 된다. 이렇게 버퍼에 저장된다면 재생 도중 다음 재생할 부분의 순서가 맞지 않아 재생 도중에 다음 순서에 해당하는 청크를 찾고 배치하게 된다면 재생 도중 버퍼링이 잦게 되고 이에 따른 스트리밍 QoS도 떨어지게 된다. Sequence controller는 버퍼에 저장된 후가 아닌 버퍼에 저장되기 전에 다중의 피어들로부터 수신하는 청크들을 Chunk_ID의 순서에 맞게 정렬하여 버퍼에 순차적으로 저장될 수 있게 하는 역할을 한다. 이는 스트리밍 시 발생되는 버퍼링을 최소화 하여 스트리밍 QoS를 높일 수 있는 방법이 된다.

② 메모리 맵(Memory map)

메모리 맵은 재생이 완료된 파일의 일부분을 저장하고 저장된 부분의 map offset을 체크하여 다른 피어가 이 미디어 파일을 다운로드 요청하면 map offset을 확인하여 현재 보유하고 있는 부분일 경우 요청피어로 전송을 하게 된다. 이 메모리 맵을 이용하게 되면 스트리밍 중이어도 다른 피어에게 청크들을 전송할 수 있고, 재생이 완료된 미디어 컨텐츠를 파일로 저장하여 가지고 있을 수 있게 된다.

IV. Sequential Download System의 구현

본 논문에서 제시한 SDS는 Windows XP에서 C#과 C++로 구현하였다. P2P 시스템은 Browser와 Listener로 구성되어 있으며, Listener는 P2P 네트워크로의 접속 및 공유 자원을 관리하고 Browser는 파일을 검색하여 다운로드 및 스트리밍을 제공한다.

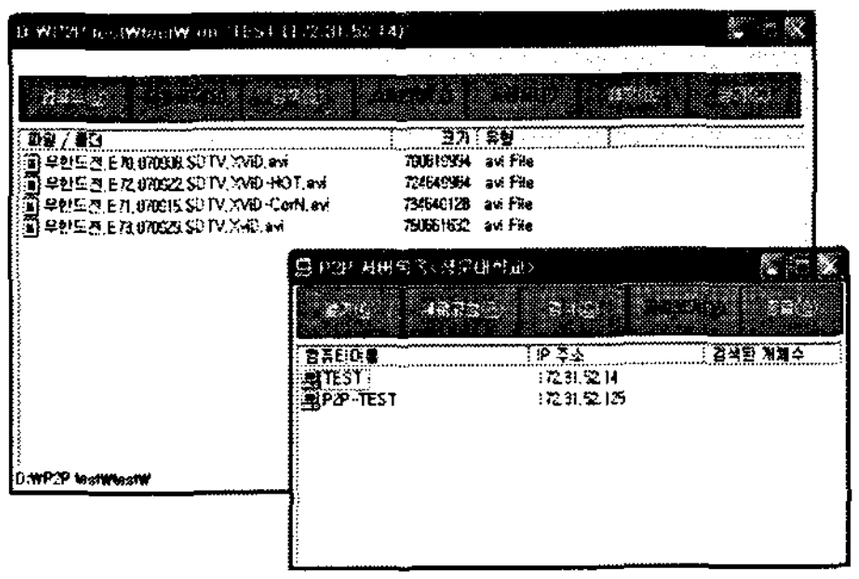

그림 8 SDS를 이용한 P2P 시스템

V. 결 론

본 논문에서는 송신피어들의 갑작스런 이탈에 대처하고 스트리밍 시 발생되는 버퍼링을 최소화 하여 스트리밍 QoS를 높일 수 있는 시스템인 SDS를 제안하였다. 또한 다운로드 또는 스트리밍 중인 파일을 다른 요청피어에게 전송할 수 있도록 시스템에 Memory map을 두도록 제안하였다.

향후에는 최근 사용되고 있는 P2P 애플리케이션에 스트리밍 기능을 적용하여 성능을 평가하고 다양한 방식의 P2P 시스템과의 호환성을 평가하고자 한다.

참고문헌

[1] Gnutella. http://www.gnutella.com
[2] Napster. http://www.napster.com
[3] D. Xu, M. Hefeeda, S. Hambrusch and B. Bhargava, "On Peer-to-Peer Media Streaming", IEEE ICDCS 2002, July. 2002.
[4] X. Jiang, Y. Dong, D. Xu, B. Bhargava, "GnuStream: a P2P Media streaming system prototype", In Proc of IEEE Intern. Conf. on Multimedia and Expo(ICME 2003), Baltimore, MD, June. 2003.
[5] 정의현, "다중 피어 결합을 이용한 P2P 멀티미디어 스트리밍 프로토콜", 컴퓨터정보학회논문지, 제11권 2호, pp. 253~261, 2006. 05.
[6] Rivest, R.L., "The MD5 message-digest algorithm," RFC-1321, 1992