P2P 네트워크에서 멀티소스 스트리밍을 이용한 컨텐츠 분배

이 성 용[†] · 소 양 선[††] · 이 재 길[†††] · 최 창 열[††††]

요 약

CDN(Contents Delivery Network)과 달리 P2P 네트워크에서는 컨텐츠를 제공하는 주체가 고정되어있지 않으므로 컨텐츠 분배는 매우 중요한 서비스이다. 그리고 P2P 미디어스트리밍에서 수신피어는 미디어 데이터를 재생함과 동시에 저장하고, 스트리밍이 끝나면 다른 피어에게 미디어 파일을 제공하는 새로운 소스피어로 동작한다. 따라서 P2P 네트워크를 통해 컨텐츠를 분배하기 위해서는 미디어 스트리밍과 파일의 저장을 동시에 수행할 수 있어야한다. 본 논문은 기존의 컨텐츠 공유 모델이 미디어 데이터를 모두 다운로드한 후에 재생을 하기 때문에 응답시간이 길고 P2P 멀티소스 미디어스트리밍에서 컨텐츠 분배가 어려운 점을 해결하여, 컨텐츠를 다운로드하면서 재생이 가능한 P2P 멀티소스 미디어스트리밍 시스템을 제안, 구현한다. 제안된 시스템에서는 수신피어가 데이터 스트리밍과 동시에 소스피어의 역할을 하므로 사용자 응답시간이 단축되고, 미디어스트리밍에 참여하는 소스피어의 수가 늘어나면 비트전송률이 증가되어 동시수요자 수도 증가된다. 또한, 피어가 가진 미디어파일의 일부분을 전송하는 메커니즘을 제공하여 컨텐츠 분배가 빨라진다.

Multi-Source Media Streaming based Contents Distribution in P2P Network Environment

Sung Yong Lee[†] · Yang Seon So[††] · Jae Gil Lee[†††] · Chang Yeol Choi[††††]

ABSTRACT

For a P2P network, the contents distribution is a very important service because the contents provider is not fixed. And in the P2P media streaming, a request peer replays and saves media data simultaneously, and after streaming it acts as a new source peer providing media files to other peers. Therefore streaming and file saving operations should be simultaneously carried out in order to distribute contents through the P2P network. In this paper, a P2P multi-source media streaming system which can replay the contents data during downloading is proposed and implemented. The system reduces the user response time and the number of simultaneous user increases more than two times. Moreover, transmitting a part of media file makes fast distribution and diffusion of contents possible.

키워드 : P2P, 컨텐츠 분배(Contents Distribution), 미디어 스트리밍(Media Streaming)

1. 서 론

최근 들어 인터넷상에서 P2P 컨텐츠 공유 프로그램이 활발하게 사용되면서 P2P 기반의 서비스들에 대한 관심이 더욱 커지고 있다. P2P 네트워크는 일반 PC에서 고성능 서버에 이르기까지 다양한 사양의 컴퓨터들로 구성되며, 각 피어(peer)는 서버와 클라이언트 역할을 모두 수행할 의무를 가져 다른 피어로부터 컨텐츠를 공급받을 뿐 아니라 제공하는 역할도 한다. 이때 컨텐츠를 구성하는 미디어 데이터를 P2P 네트워크에 연결된 하나의 특정 피어로부터만 수신하면 사용자 QoS를 만족시키지 못하게 된다. 그리고 피어의 네트워크 이탈이 잦아 하나의 소스피어로부터 데이터를 수신하는 일대일 P2P 패러다임은 사용자가 요구하는 전송 속도와 데이터 안정성을 보장하지 못하게 된다[1]. 이를 해결하기 위해 여러 개의 소스피어로부터 데이터를 수신하는 멀티소스 다운로드 또는 멀티소스 스트리밍이 제안되고 있다[1, 3, 4].

멀티소스 다운로드 방식으로 P2P 컨텐츠를 공유하는 e-Donkey[5]는 여러 소스피어로부터 데이터를 다운로드하여 각 소스피어별로 임시파일을 만들고, 수신이 모두 끝난 후에 임시파일들을 하나의 파일로 조합하여 속도를 빠르게 한다. On-Demand 방식의 P2P 미디어 스트리밍 시스템인 GNUS-

※ 본 연구는 정보통신부 대학 IT연구센터 육성·지원사업과 강원대학교 정보통신연구소의 지원을 받았습니다.
† 준 회 원 : 강원대학교 대학원 컴퓨터정보통신공학과
†† 정 회 원 : LG전자 정보통신사업본부 단말연구소 연구원
††† 정 회 원 : 원주대학 컴퓨터정보관리과 교수
†††† 정 회 원 : 강원대학교 전기전자정보통신공학부 교수
논문접수 : 2004년 6월 19일, 심사완료 : 2004년 8월 26일

TREAM은 Gnutella 컨텐츠 공유 응용프로그램인 Gnucleus를 기반으로 하며, 여러 소스피어로부터 서로 다른 데이터 세그먼트를 받아 조합, 재생하며 피어 상태를 감지하고, 버퍼를 조절하는 기능을 갖는다. 이와 같이 eDonkey와 GNUSTREAM은 멀티소스로부터 데이터를 수신하여 전송률을 높이고, 소스피어의 이탈에 대한 피해를 감소시키고자 하였다. 그러나 eDonkey의 경우, 파일을 모두 다운로드 한 후에 재생을 하기 때문에 파일을 수신하는 동안에는 미리 열어 볼 수 없어 응답시간이 길어진다. GNUSTREAM은 서버-클라이언트 방식의 스트리밍처럼 미디어 데이터를 재생한 후에 소비하기 때문에 미디어 컨텐츠를 다른 피어와 공유하지 못해 P2P 네트워크에서의 컨텐츠 분배가 어렵다.

CDN(Contents Delivery Network)과 달리 P2P 네트워크에서는 컨텐츠를 제공하는 주체가 고정되어있지 않으므로 컨텐츠 분배는 매우 중요한 서비스이다. 그리고 P2P 미디어 스트리밍에서 수신피어는 미디어 데이터를 재생함과 동시에 저장하고 스트리밍이 끝나면 다른 피어에게 미디어 파일을 제공하는 새로운 소스피어로 동작한다. 따라서 P2P 네트워크를 통해 컨텐츠를 분배하기 위해서는 미디어 스트리밍과 파일의 저장을 동시에 수행할 수 있어야 한다. 그리고 특정한 미디어 파일을 소유한 피어의 수가 적으면 멀티소스 미디어스트리밍에 참여할 수 있는 소스피어의 수도 적어지므로 특정미디어 서비스를 요청하여 제공받는 수요자의 수가 제한될 수 있다.

본 논문은 기존의 컨텐츠 공유 모델이 미디어 데이터를 모두 다운로드한 후에 재생을 하기 때문에 응답시간이 길고 P2P 멀티소스 미디어스트리밍에서 컨텐츠 분배가 어려운 점을 해결하여, 컨텐츠를 다운로드하면서 재생이 가능한 P2P 멀티소스 미디어스트리밍 시스템을 제안, 구현한다. 제안된 시스템에서는 수신피어가 데이터 스트리밍과 동시에 소스피어의 역할을 하므로 사용자 응답시간이 단축되고 P2P 네트워크를 통한 컨텐츠 분배도 가능하다. 또한, 피어가 가진 미디어파일의 일부분만을 전송하는 메커니즘을 지원하므로써 컨텐츠 분배가 빨라지고, 멀티소스 미디어스트리밍에서는 소스피어의 수가 늘어나면 비트전송률이 증가되어 동시수요자 수도 증가된다.

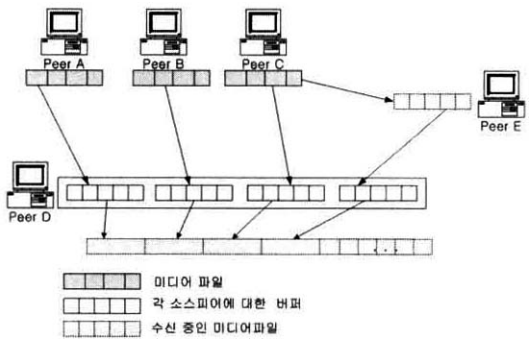
본 논문의 구성은 다음과 같다. 2장에서는 기존 연구의 단점을 보완하는 P2P 멀티소스 미디어스트리밍 시스템의 설계와 구현 내용을 구체적으로 기술한다. 그리고 3장에서는 제안된 시스템의 시험과 검증에 대해 언급하고 마지막 4장에서 결론을 보인다.

2. 스트리밍 시스템의 설계 및 구현

본 논문에서 제안하는 멀티소스 미디어스트리밍 시스템의 특징과 기본 동작, 피어간 통신과 통신단계별 알고리즘 그리고 클라이언트 모듈과 서버 모듈에 대해 기술한다. 제안하는 시스템은 P2P 컨텐츠 공유 모델을 바탕으로 스트리밍 기능을 추가, 확장한 형태로서 미디어데이터의 인증, 검색과 같은 기본 기능은 지원된다고 가정한다.

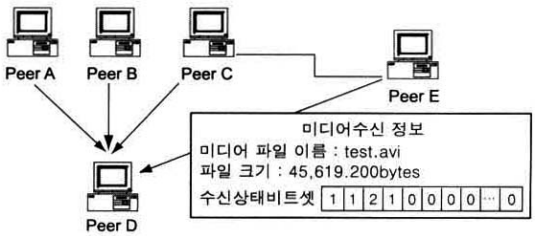
2.1 기본 동작

P2P 네트워크에 연결된 모든 피어는 데이터의 송신과 수신 기능을 동시에 수행할 수 있다. 수신피어는 미디어 파일을 작게 나눈 구간을 소스피어들에 요청하여 미디어 파일의 서로 다른 부분을 병렬로 받음과 동시에 미디어를 재생한다. 송신 주체인 소스피어들은 동일한 인코딩형식과 재생 비트전송률, 같은 크기의 미디어 파일을 소유하며, 미디어 파일 전체를 갖고 있을 때뿐만 아니라 일부분을 갖고 있을 때에도 미디어의 송신이 가능하다.



(그림 1) 피어간 미디어데이터 흐름

미디어스트리밍의 전체적인 흐름은 (그림 1)과 같다. (그림 1)에서 PeerD는 수신피어로서 클라이언트 역할을, PeerA, B, C는 소스피어로서 서버 역할을 하고 PeerE는 PeerC로부터 미디어데이터를 수신한다. 이때 PeerA, B, C는 각각에게 배정된 미디어파일의 일정부분을 PeerD에 송신하며, PeerE는 자신이 현재 수신, 저장하고 있는 미디어데이터를 PeerD로 전송한다. PeerD는 PeerA, B, C, E로부터 받은 데이터를 버퍼의 피어별 해당 구간에 쓰고, 파일로 저장하여 수신 중인 파일을 재생한다. PeerE는 송수신을 동시에 수행하며, PeerD는 PeerE로부터 미디어수신정보를 (그림 2)와 같이 전송받아 PeerE의 미디어 저장상태를 확인하여 자신의 미디어수신상태정보비트셋과 비교함으로써 다음 미디어의 수신 여부를 결정한다.



(그림 2) 미디어수신상태정보 교환

2.2 피어간 통신

피어간 통신은 미디어파일이 있는 피어의 IP를 안다는 가정에서 이루어지며, 크게 미디어정보 요청-응답, 미디어데이터 요청-전송, 미디어수신상태정보 갱신요청-전송 단계로 구분된다. 각 단계별 피어간 세부 통신과정은 다음과 같다.

• 미디어정보 요청

미디어정보요청단계에서는 선정된 소스피어들에게 수신파일명을 전송한다.

```
for j = 1 to N(Selected Source Peers's Num)
    if (Connection State == TRUE) then
    Message = PacketSize + Type(File_Info_Req) + FileName
Send message to selected peers that are connected
```

• 미디어정보 응답

미디어정보응답단계에서는 온전한 파일이 존재하면 파일의 송신가능여부와 파일정보를 전송하며, 수신 중인 파일일 때는 파일정보와 현재 수신상태비트셋(BitStream_Index)을 함께 전송한다.

```
if (State == SEND_OPEN)
  if (State of Media file == Entire File) then
    Message = PacketSize + Type(File_Info_Ack)
              + State of File(Entire File)
              + File Name + FileSize
  else if (State of Media file != Entire File) then
    Message = PacketSize + Type(File_Info_Ack)
              + State of File(Not entire File)
              + File Name + FileSize + BitStreamIndex
Send message to request peer
```

• 미디어데이터 요청

미디어데이터요청단계에서는 각 소스피어에게 송신을 원하는 미디어 파일의 일정부분을 배정(Start Index + End Index)하여 파일명과 함께 전송하며, 미디어수신 중에 자신의 미디어수신상태비트셋을 체크하여 비트가 '0'인 부분 즉 아직 요청하지 않은 미디어파일부분의 인덱스를 배정하여 소스피어에게 전송한다.

```
case state bit is 0 : not yet requested
case state bit is 1 : corresponding part has been completely
                      received
case state bit is 2 : corresponding part is being now received

for j = 1 to N(Selected Source Peers's Num)
  Message = PacketSize + Type(File_Stream_Req) + FileName +
            (Request FileName) + Start Pointer + End Pointer
Request again after checking my Bit_Stream_Index
```

• 미디어데이터 전송

미디어데이터전송단계는 미디어스트림요청 패킷에서 Start Index와 End Index를 분리한 후, 해당 부분을 파일 또는 메모리맵파일에서 복사하여 패킷단위로 보낸다. 이때 Offset은 배정된 부분에서 패킷단위로 나눈 구분자이며, Content Size는 하나의 패킷으로 보내는 미디어데이터의 크기이다. Start Index와 End Index는 배정된 부분의 전송이 끝날 때까지 소스피어에서 계속 유지한다.

```
Message = PacketSize + Type(FILE_STREAM_ACK) + Send Point +
          Offset + Content Size + Media Data
Send message to request peer
```

• 미디어수신상태정보 갱신요청

수신피어는 자신의 미디어수신상태비트셋과 소스피어들의 미디어수신상태비트셋을 비교하여 소스피어에게 요청할 구간이 있는지를 검사하여 요청할 구간이 더 이상 없을 때, 소스피어에게 다시 미디어수신상태정보 갱신요청을 한다.

```
Message = PacketSize + Type(Update_bitStream_index_req) + FileName
Send message to request peer
```

• 미디어수신상태정보 전송

미디어수신상태정보갱신요청이 오면, 미디어수신상태정보 전송단계에서는 현재 미디어수신상태 비트셋을 '0'과 '1'의 비트열로 변환하여 수신피어에게 전송한다.

```
Message = PacketSize + Type(Update_bitStream_index_Ack) + FileName
          + Bit_Stream_index
Send message to request peer
```

2.3 클라이언트 기능 모듈

본 P2P 멀티소스 스트리밍 시스템은 WindowsXP 상에서 Visual C++와 C로 구현하였다. 모든 피어는 서버 모듈과 클라이언트 모듈을 가지며, 서버 모듈은 다른 피어가 요청한 명령을 수행하는 루틴이고, 클라이언트 모듈은 소스피어의 명령을 수행한다.

클라이언트 모듈은 소스피어 선별, 전송구간 배정, 데이터 조합, 파일저장, 실시간 재생 단계로 구성되며, 각각은 다음과 같다.

2.3.1 소스피어선별

P2P 네트워크 상의 소스피어와 수신피어는 서로 다른 네트워크 환경, 컴퓨터 성능을 가질 뿐 아니라, 소스피어와 수신피어의 거리도 매우 다양하다. 따라서 여러 소스피어 중에서 전송이 유리한 피어를 선별한다. 소스피어를 선별하기 위해 우선 파일이 현재 전송 가능한지, 소스피어가 최대로 연결할 수 있는 수신피어의 수를 초과하지 않는지를 검사한다. 그리고 소스피어의 네트워크 대역폭, 수신피어와의 근접성에 따라 판단한다.

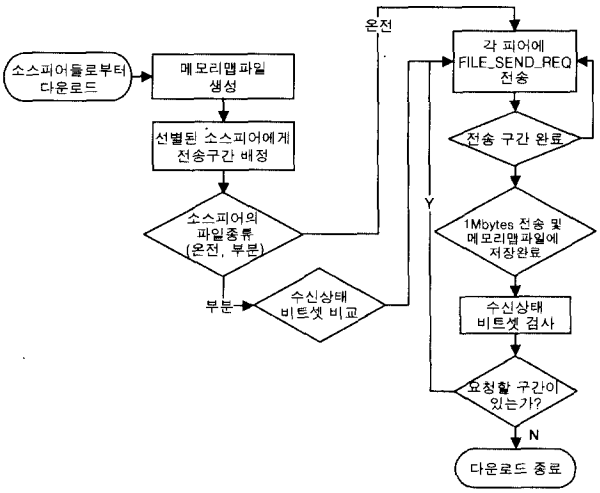
이용 가능한 네트워크 대역폭은 TCP-friendly rate control algorithm(TFRC)[4]을 이용하여 다음과 같이 계산한다. B는 송신자와 수신자 사이의 현재 이용 가능한 대역폭을 나타내며, Trto는 TCP time out, R은 round-trip 시간, p는 loss rate이며, s는 TCP 패킷세그먼트를 나타낸다. 이때 네트워크 대역폭은 R에 반비례하므로, 실제 구현에서는 ICMP 패킷을 여러 번 전송하여 평균 round-trip 시간을 계산하여 피어선별을 위한 하나의 인자로 넣는다.

$$B = \frac{s}{R\sqrt{\frac{2p}{3}} + Trto\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

미디어파일을 갖고 있는 소스피어 중 가장 근접한 피어를 선택하는 기준인 피어간 근접성은 두 피어의 IP주소가 동일한 ISP 또는 WAN일수록, 나아가 같은 LAN 상에 있을수록 유사하다는 점을 이용하여 계산한다. 계산된 결과 값은 두 피어 사이의 물리적인 거리를 의미하지는 않지만 소스피어의 근접성을 판단하기위한 값으로서 수신피어와 근접 값이 작은 소스피어가 선택된다[6]. 전송에 유리한 n개의 소스피어가 선택되면 각 소스피어에게 전송구간을 배정한다.

2.3.2 전송구간 배정

(그림 3)은 소스피어 각각에 전송구간을 배정하는 과정을 보인다. 소스피어 선별과정에서 최대 소스피어 수 이하의 소스피어가 선택되면 소스피어 각각에 최대 전송률을 배정한다.



(그림 3) 전송구간 배정 및 다운로드 과정

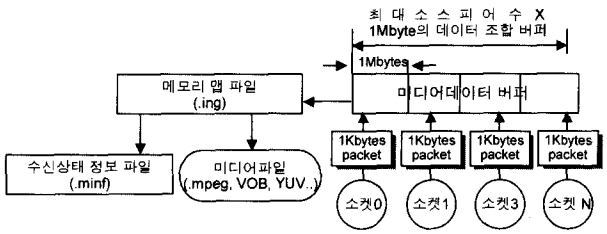
전송구간배정 단계에서는 각 소스피어 당 일정구간(기본: 1Mbytes)을 배정하되 전송이 먼저 끝난 소스피어에게는 미디어수신상태 비트셋을 검사하여 '0'인 부분 중 가장 앞에 있는 구간을 배정한다. 이로써, 전송속도가 가장 빠른 소스피어에게 다음 전송구간을 배정하고, 전송속도가 빠른 소스피어가 더 많은 구간을 전송하게 된다.

전송구간배정은 파일형식에 따라 두 가지로 나뉜다. mpg 파일은 파일의 처음부터 순차적으로 1Mbytes씩 배정하고, RIFF 파일 규격을 따르는 avi의 경우는 header와 avi인덱스를 먼저 수신한 후 비디오/오디오 부분을 1Mbytes씩 순차적으로 배정한다. 여기서 AVI Index chunk에 파일 안의 데이터 chunk들의 위치정보가 포함되므로[7] 다운로드와 함께 파일을 재생하기 위해서는 header chunk와 AVI Index chunk를 먼저 필요로 한다. 기존의 파일공유 응용프로그램은 파일형식에 관계없이 순차적으로 다운로드 받기 때문에 다운로드와 동시에 재생이 불가능하다. 따라서 재생에 필요한 header chunk와 AVI Index chunk를 먼저 수신하여 메모리맵의 해당위치에 쓰도록 함으로써 재생할 때 파일 렌더링이 가능하도록 하였다.

2.3.3 데이터조합 및 파일저장

각 소스피어에 전송구간을 배정하여 데이터를 수신하면, 수신피어는 데이터를 조합하여 하나의 파일로 만드는 과정인 데이터 조합과 파일저장 단계를 거치게 된다.

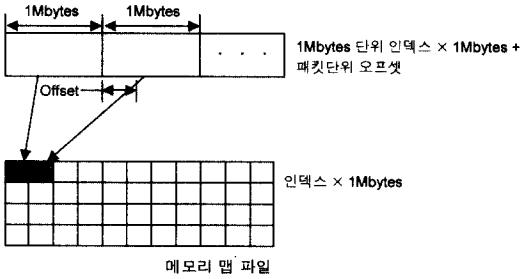
모든 피어는 (그림 4)와 같은 미디어데이터버퍼와 메모리맵 파일을 갖고 있으며 이를 이용하여 미디어데이터송수신, 미디어파일저장, 미디어재생을 동시에 수행한다.



(그림 4) 미디어 데이터 조합

데이터조합 및 파일저장 단계는 소스피어로부터 받은 미디어데이터를 순서에 맞게 조합하는 과정으로, 클라이언트 소켓을 통해 전달된 미디어데이터를 소스피어별로 배정된 미디어데이터버퍼구간에 쓴다. 여기서 미디어데이터버퍼의 크기는 소스피어의 최대 수에 따라 결정되며, 소스피어별로 1Mbytes의 버퍼구간이 배정된다.

해당 미디어데이터버퍼구간이 다 차면 버퍼의 미디어데이터는 메모리맵 파일의 해당 위치에 쓰여지며 일정 시간이 지난 후, 메모리맵 파일을 열어 미디어를 재생한다. 미디어데이터버퍼와 메모리맵 파일 사이의 데이터는 (그림 5)와 같이 이동된다. 미디어데이터버퍼의 해당구간이 차면, Start Index와 End Index 값에 따라 메모리맵 파일의 정해진 위치에 데이터가 쓰여진다. 이때 미디어수신상태정보비트셋에 해당구간의 수신상태를 표시하여 미디어데이터를 요청하고 다른 피어에게 미디어 각 부분의 수신상태를 알려주는데 사용한다. 미디어수신정보비트셋은 '0', '1', '2'로 상태를 표시하며, '0'은 아직 요청하지 않은 상태, '1'은 수신완료 상태, '2'는 현재 수신 중인 상태를 나타낸다.



(그림 5) 버퍼와 메모리맵 파일 간 데이터 이동

2.3.4 실시간 재생

실시간 재생 단계에서는 수신중인 미디어 파일을 열고, 얼마간의 시간이 지난 후에 재생을 한다. 여기서 얼마동안의 시간은 끊김 없는 재생을 위한 initial delay 또는 rolling time으로서 보통 1초에서 15초 사이의 값으로 설정한다. 데이터의 실시간 재생을 위한 미디어플레이어는 DirectShow를 기반으로 구현하였다. 구현한 미디어플레이어 이외에 윈도우미디어플레이어, 상용 미디어 플레이어로도 재생이 가능하며, 미디어플레이어의 파일열기를 통해 ing파일을 직접 열어 볼 수 있다.

2.4 서버 기능 모듈

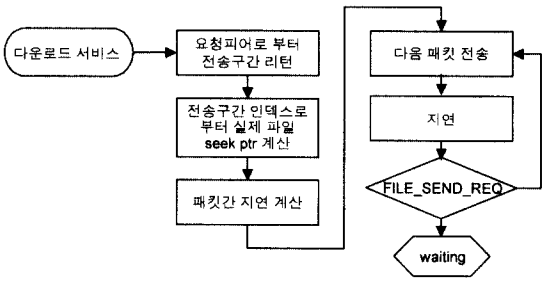
서버 모듈은 파일정보 전송, 피어연결, 전송구간설정, 데이터전송으로 구성된다. 피어연결 단계에서는 설정된 최대 전송피어 수를 넘지 않도록 하며, 전송구간설정 단계에서는 수신피어에 따른 인덱스 설정 및 패킷 생성을 한다. 데이터전송 단계에서는 소스피어의 자원을 보호하기 위해 패킷 간 지연을 두어 송신전송률을 제한한다.

2.4.1 파일정보 전송

파일정보 전송은 수신피어가 미디어를 검색한 후에 파일의 동일성, 전송가능 유무, 피어상태 등을 알기위한 과정으로서 크게 다운을 완료한 파일 즉 온전한 미디어파일의 정보와 다운로드 중인 미디어파일의 정보 전송으로 나뉜다. 다운로드 중인 미디어파일의 파일확장자는 .ing로 설정한다. 파일정보 전송 요청이 오면 공유디렉토리에서 해당파일의 정보를 검사하는데, ing파일인 경우는 파일의 일반 정보인 파일 크기, 피어상태, 파일명 이외에 현재 다운로드 중인 미디어파일의 수신정도를 나타내는 수신상태비트셋을 함께 리턴한다.

2.4.2 전송구간 설정 및 미디어데이터 전송

요청된 전송구간을 설정하여 미디어 데이터를 전송하는 과정은 (그림 6)과 같다. 수신피어가 보낸 1Mbytes단위 전송구간 인덱스에서 실제 파일 seek 포인터를 계산한 후에 해당되는 1Mbytes 파일의 부분을 읽어 설정된 패킷간 지연 이후에 1Kbytes의 패킷으로 나누어 전송한다.



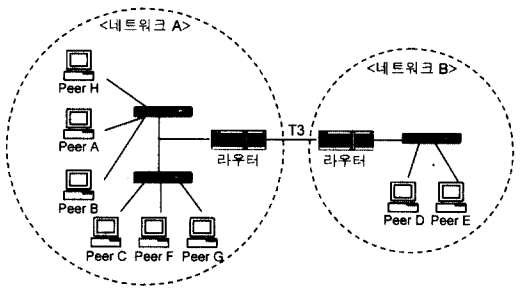
(그림 6) 전송구간 설정 및 미디어데이터 전송 과정

3. 시험 및 결과

간단한 P2P 네트워크를 구성하여 소스피어의 수에 따른 데이터전송률, 서로 다른 형식으로 된 데이터의 스트리밍, 다운로드 및 동시업로드 시험을 통해 제안된 시스템의 기능과 성능을 확인하고 검증하였다.

3.1 테스트베드

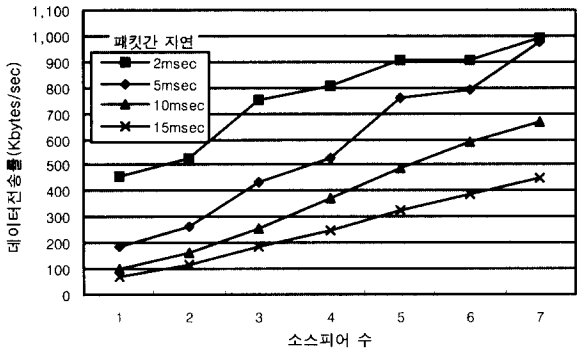
멀티소스 미디어 스트리밍을 시험하기 위해 (그림 7)과 같이 간단한 P2P 네트워크를 구성하였다. 네트워크 A와 B는 T3로 연결하고, 네트워크 A와 B의 내부는 100Mbps 이더넷으로 연결하여 가정의 인터넷 환경보다는 네트워크 전송률이 월등히 높다. 그리고 각 네트워크의 피어들은 Pentium Ⅳ, 256MBytes 이상의 메모리, Windows XP를 장착하여 일반 사무실에서 쉽게 접할 수 있는 시스템들과 유사하게 구성하였다.



(그림 7) 시험네트워크의 구성

3.2 데이터전송률 시험

멀티소스로부터의 데이터전송률은 소스피어를 1대에서 7대까지 증가시키면서, 소스피어의 데이터 패킷간 지연이 2, 5, 10, 15msec일 때, 수신피어가 30Mbytes의 미디어데이터를 파일에 쓰는 시간을 측정하였다. 데이터 전송률은 (그림 8)에서 보는 것처럼 소스피어의 수가 늘어나고 패킷간 지연이 작을수록 증가하는 특성을 보였다. 특히 패킷간 지연이 2msec일때 소스피어를 1개에서 7개로 늘리면 데이터전송률은 약 2배 증가하고, 패킷간 지연이 15msec이고 소스피어 1대일 때의 최저 데이터전송률을 나타냈고, 패킷간 지연이 2msec이고 소스피어 7대인 경우의 최대 데이터전송률을 보였으며 그 차이는 1,476%였다.



(그림 8) 패킷간 지연과 소스피어 수에 대한 데이터전송률

3.3 스트리밍 시험

스트리밍 시험에서는 MPEG-1, MPEG-2, DivX로 인코딩된 미디어 파일을 다운로드와 동시에 재생 가능하게 하는 최저 소스피어의 수와 소스피어의 패킷간 지연을 찾고, 소스피어의 수가 증가함에 따라 미디어 파일 전체를 다운로드하는데 소요되는 시간의 변화와 평균 데이터전송률도 측정하였다.

3.3.1 MPEG-1, 2 스트리밍

MPEG-1 스트리밍은 크기가 44.6MBytes이고 비트율이 172KBytes/s인 저화질 파일(Amuro.mpg)과 크기가 65.2MBytes이고 비트율이 259KBytes/s인 고화질 파일(Boa.mpg)을 해상도 352×240과 프레임 전송률 30fps로 설정하고, MPEG-2 스트리밍은 크기가 287MBytes이고 722KBytes/s의 비트율을 갖는 파일(Sheri.VOB)을 해상도 720×480과 프레임 전송율 30fps로 설정하여 시험하였다. 결과는 <표 1>과 같다. 예를 들어, MPEG-2 Sheri.VOB 파일은 2ms의 패킷간 지연을 갖는 소스피어 3개를 이용하면 392초에 다운로드되어 파일재생시간 407초보다 적어 끊김이 없이 미디어 재생이 가능함을 알 수 있다.

〈표 1〉 MPEG-1, 2 다운로드 및 동시재생 시험결과

파일명	소스피어 수	패킷간 지연	데이터 전송률 (Kbytes/s)	다운로드 시간(sec)	파일재생 시간(sec)
Amuro.mpg (저화질) MPEG-1	1	5ms	182.85	250	268
	2	5ms	262.56	174	
	3	15ms	187.88	243	
Boa.mpg (고화질) MPEG-1	1	2ms	455.11	147	259
	2	5ms	262.56	255	
	3	10ms	259.24	258	
Sheri.VOB MPEG-2	3	2ms	749.26	392	407
	4	2ms	808.42	364	

3.3.2 DivX Avi 스트리밍

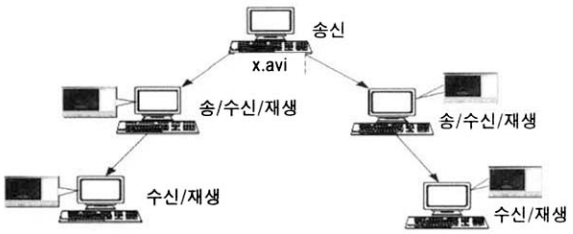
DivX Avi에는 여러 인코딩 형식이 존재하므로, DivX3, DivX4, mp3 형식의 파일, Xvid, mp3 형식의 파일, DivX3, AC3 형식의 파일을 이용하여 스트리밍을 시험하였으며, 그 결과는 <표 2>와 같다.

〈표 2〉 DivX Avi 다운로드 및 동시재생 시험결과

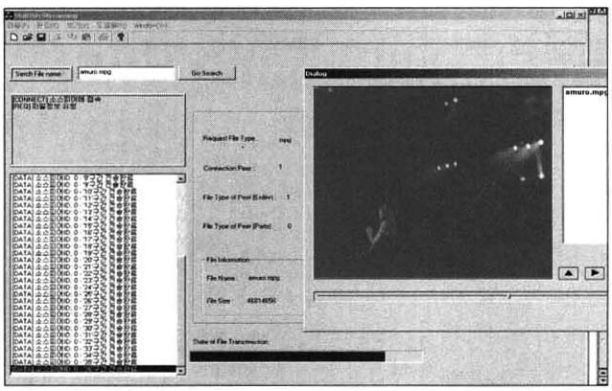
파일명 (인코딩형식)	소스피어 수	패킷간 지연	데이터 전송률 (Kbytes/s)	다운로드 시간(sec)	파일재생 시간(sec)
Ani.avi (Divx4, mp3)	1	15ms	66.06	387	1203
X-file.avi (Divx3, mp3)	1	5ms	182	184	314
	2	15ms	117.70	284	
badboy.avi (Xvid, mp3)	1	5ms	192	520	622
	2	10ms	190	526	
	3	15ms	208	480	
Firstlove.avi (Divx3, AC3)	1	2ms	455.11	340	629
	2	5ms	262.56	589	
	3	10ms	259.24	597	

3.4 다운로드 및 동시업로드 시험

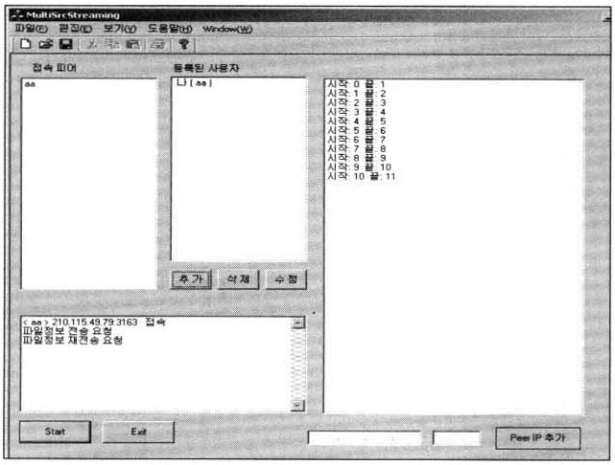
(그림 9)는 미디어 파일을 다운로드와 동시에 업로드가 가능하도록 하여 여러 수요자가 미디어파일을 동시에 재생할 수 있는 시험용 시스템의 구성을 보인다. 시험에 사용한 샘플 파일은 amuro.mpg로서 프레임 크기 350x240에 30fps, 172Kbytes/sec의 비트율을 가진다. (그림 9)에서 하나의 피어에 접속할 수 있는 피어는 최대 2대로, 그리고 소스피어의 전송률을 200Kbytes/sec이하로(패킷간 지연 : 4msec) 제한하였다. 수신피어가 소스피어에 접속하는 간격은 10~20초이며, 미디어파일은 초기에는 하나 밖에 없었으나 다운로드와 동시업로드를 통해, 4대의 피어가 미디어파일을 동시에 재생할 수 있게 된다. 이 시험을 통해 다운로드 및 동시업로드가 미디어파일을 빠르게 확산시키고 동시 수요자 수를 2배 이상 증가시킬 수 있음을 확인하였다. (그림 10)은 (그림 9)의 중간 노드의 클라이언트 뷰를, (그림 11)은 서버 뷰를 보이며, (그림 12)는 (그림 9)의 하단 노드의 클라이언트 뷰를 보인다.



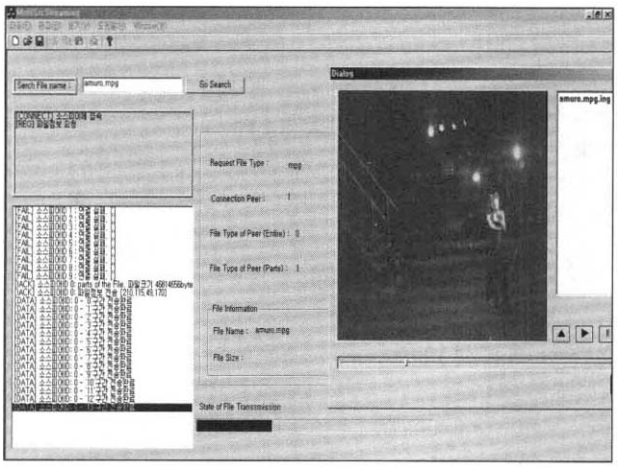
(그림 9) 다운로드 및 동시업로드 시험 구성도



(그림 10) 다운로드 및 동시업로드 피어의 클라이언트 뷰



(그림 11) 다운로드 및 동시업로드 피어의 서버 뷰



(그림 12) 다운로드 중인 소스피어로부터의 수신 및 재생

3.5 검토 및 분석

3.5.1 다중 소스피어로부터 데이터 수신 및 조합

P2P의 특성상, 각 피어의 네트워크 대역폭, 컴퓨터 성능 등이 서로 달라 일대일 수신으로는 사용자가 요구하는 데이터전송률을 충족시키지 못할 가능성이 높다. 또한 서버-클라이언트 모델과 달리 온라인과 오프라인을 사용자 임의로 정할 수가 있어 하나의 소스피어로부터만 데이터를 수신하면 소스피어의 상태에 따라 서비스가 중단될 위험성이 있다. 따라서 여러 개의 소스피어로부터 데이터를 수신하여 조합함으로써 데이터전송률을 향상시킬 수 있을 뿐 아니라 소스피어의 네트워크 이탈에 따른 피해도 줄일 수 있다.

3.5.2 파일의 저장과 동시에 재생

기존의 파일공유 시스템은 파일을 모두 다운 받은 후에 재생하는 형태로서 수 분에서 수 시간의 응답시간을 갖는다. 그리고 다운로드 중간에 파일을 재생할 수 없으므로, 다운로드가 끝난 후에야 자신이 다운로드받고 있는 파일이 원하는 파일인지를 알 수 있는 단점이 있다. 본 시스템은 파일의 다운로드와 동시에 재생이 가능하도록 하였으며, 이를 통해 사용자 응답시간이 단축된다.

3.5.3 동일 파일의 다운로드 및 동시 업로드

미디어파일을 다운로드와 동시에 업로드 할 수 있게 하여 P2P네트워크에서 컨텐츠의 분산이 빨라진다. 그리고 미디어파일을 제공하는 소스피어의 수가 증가함에 따라 동시 수요자 수도 증가한다.

3.5.4 모든 형식의 파일공유 및 다운로드, avi, mpg 데이터의 실시간 재생

대표적인 순수 P2P 파일공유 프로토콜인 Gnutella에서 보인 사용자 요청 빈도별 파일형식 분석[8]에 따르면 사용자의 65% 이상이 멀티미디어 컨텐츠를 요청하였으며, 그 중에 avi 파일의 요청이 18.72%로 가장 높게 나타났다. 본 시스템에서는 요청빈도가 가장 높은 avi, mp3, mpg의 데이터 형식을 지원하며, 그 외에 asf도 지원한다.

4. 결 론

초고속 인터넷 사용자의 확산과 개인 컴퓨터의 성능 향상으로 개인과 개인이 직접 통신하여 컴퓨터 자원과 자료, 정보 등을 교환하고 공유하는 P2P 기술이 많이 사용되면서 특정한 서버의 도움을 받지 않고도 다양한 자원과 컨텐츠를 공유할 수 있게 되었다. 그러나 기존의 컨텐츠 공유 응용프로그램은 파일의 다운로드가 끝난 후에야 재생을 하기 때문에 응답시간이 길고 원하는 컨텐츠인지 미리 확인할 수가 없었다. 한편 P2P 네트워크 상에서 실시간으로 미디어를 재생하는 P2P 멀티소스 미디어스트리밍에 대한 기존 연구들에서는 컨텐츠 분배 서비스를 고려하지 않았다.

본 논문에서는 P2P 네트워크 환경에서 컨텐츠를 공유하면서 스트리밍 기능을 갖는 시스템을 설계하고 구현하였다. 구현된 시스템은 미디어의 송수신, 파일저장 그리고 재생을 동시에 할 수 있어 P2P 피어들간에 컨텐츠 분배가 가능하고 사용자 응답시간이 단축되는 이점이 있으며 다음과 같은 특징을 갖는다. 첫째, 여러 피어로부터 데이터를 수신함으로써 재생에 요구되는 비트전송률보다 더 큰 미디어 전송률을 제공한다. 실험을 통해 1배 내외의 데이터전송률 증가를 확인하였다. 둘째, 미디어를 수신하여 파일로 저장함과 동시에 재생함으로써 미디어를 실시간으로 감상할 수 있으며, 네트워크를 통한 컨텐츠 분배도 가능하다. 셋째, 다운로드 중인 피어가 가진 미디어파일의 일부분을 전송할 수 있게 하여 네트워크 상에서 컨텐츠의 빠른 분배와 확산이 가능하고, 동시 수요자 수도 2배 이상으로 증가된다.

앞으로, 멀티소스로부터의 데이터를 수신할 때 파일의 동

일성을 단지 파일크기가 아닌 무결성으로 확인하고 네트워크의 상태에 따라 소스피어의 전송률을 동적으로 제어하는 기능에 대한 연구가 필요하다.

참 고 문 헌

[1] Xu-Xian JIANG and Yu Dong, "GnuStream : A MPEG Video Streaming System over Peer-to-Peer Network," Project Report for CS641 Multimedia Database, Fall 2002.

[2] D. Xu, M, Hfeeda, S. Hambrusch and B. Bhargava, "On Peer-to-peer Media Streaming," IEEE ICDCS 2002, pp.363-371, July 2002.

[3] Reza Rejaie, Antonio Ortega, "PALS : Peer-to-peer Adaptive Layered Streaming," Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp.153-161, June 2003.

[4] T. Nguyen and A. Zakhor, "Distributed Video Streaming Over Internet," SPIE/ACM MMCN 2002, January 2002.

[5] eDonkey, "http://www.edonkey2000.com."

[6] Avi RIFF file reference, "http://msdn.microsoft.com/library/en-us/directx9_c/htm/aviri- fffilereference.asp."

[7] CLIP2, "The Gnutella Protocol Specification v0.4," www.clip2.com.

[8] Doug Kaye, "Peer-to-Peer Content Delivery using Information Additive Codecs," RDS whitepaper, January 2002.

[9] Demertis Zeinalipour-Yazti, Theodoros Folias, "A Quantitative Analysis of the Gnutella Network Traffic," CS204 final project, July 2002.

[10] M. M. Hefeeda, B. K. Bhargava, D. K. Y. Yau, "A Hybrid architecture for cost-effective on-demand media streaming," Journal of Computer Networks, Vol.44, No.3, pp.353-382, 2004.

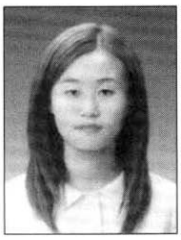
이 성 용

e-mail : moota4@mmslab.kangwon.ac.kr

2003년 강원대학교 정보통신공학과(학사)

2003년~현재 강원대학교 대학원 컴퓨터정보통신공학과 석사과정

관심분야 : 모바일스트리밍, 멀티미디어시스템



소 양 선

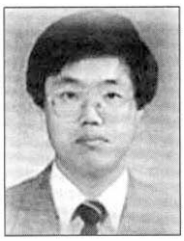
e-mail : sys2you@hanmail.net

2002년 강원대학교 정보통신공학과(학사)

2004년 강원대학교 컴퓨터정보통신공학과 (석사)

2004년~현재 LG전자 정보통신사업본부 단말연구소 연구원

관심분야 : 멀티미디어시스템, P2P통신, 모바일 통신프로토콜



이 재 길

e-mail : jglee@sky.wonju.ac.kr

1983년 경북대학교 전자공학과(학사)

1985년 경북대학교 전자공학과(석사)

1985년~1993년 ETRI 선임연구원

2000년~현재 강원대학교 컴퓨터정보통신공학과 박사과정

1993년~현재 원주대학 컴퓨터정보관리과 교수

관심분야 : 모바일컴퓨팅, 멀티미디어시스템

최 창 열

e-mail : cychoi@kangwon.ac.kr

1979년 경북대학교 전자공학과(학사)

1981년 경북대학교 전자공학과(석사)

1995년 서울대학교 컴퓨터공학과(박사)

1984년~1996년 ETRI 컴퓨터연구단 책임연구원/연구실장

1996년~현재 강원대학교 전기전자정보통신공학부 교수

관심분야 : 컴퓨터시스템, 모바일컴퓨팅, 멀티미디어시스템