P2P 환경을 위한 확장성있는 멀티소스 미디어스트리밍

소양선° 최창열
강원대학교 컴퓨터·정보통신공학과
so@mmslab.kangwon.ac.kr°, cychoi@kangwon.ac.kr

Design and Implementation of Scalable Multi-source Media Streaming for P2P Environments

YangSeon So° ChangYeol Choi
Department of Computer, Information and Telecomm. Engineering, Kangwon National University

요 약

Peer-to-peer(P2P) 네트워크를 구성하는 각 피어는 서로 다른 네트워크대역폭, 업로드속도, 컴퓨터성능을 가지므로 미디어 요청피어가 특정 피어로부터만 데이터를 받는 경우엔 사용자 QoS를 만족시키지 못할 가능성이 높다. 또한 미디어데이터를 제한된 시간 내에 수신해야하는 스트리밍시스템에서는 여러 피어로부터 데이터를 수신하는 멀티소스스트리밍이 더욱 요구된다. 본 논문에서는 P2P 네트워크 환경에서 여러 피어로부터 미디어를 송수신하고 파일로 저장하는 멀티소스스트리밍시스템을 설계, 구현하였다. 그리고 스트리밍 중인 피어가 가진 미디어파일의 일부분을 전송하는 기법을 적용하여 비트 전송률과 동시 수요자 수를 증가시켰다.

1. 서 론

최근들어 인터넷 상에서 P2P파일공유응용프로그램이 활발하게 사용되면서 P2P에 대한 관심이 더욱 커지고 있다. 기존 P2P 시스템과 P2P미디어스트리밍은 데이터공유모드에 따라 구별되며, 데이터공유모드에는 'Open-After-Downloading' 과 'Play-While-Downloading' 이 있다[1]. 미디어파일을 모두 다운받은 후에 재생하는 P2P응용프로그램은 전자에, 다운로드와 동시에 재생을 하는 P2P미디어스트리밍은 후자에 속한다. P2P미디어스트리밍에서 미디어요청피어는 재생과 동시에 미디어데이터를 저장하고 스트리밍이 끝나면 다른 피어에게 미디어 파일을 제공하는 새로운 소스피어가 된다.

P2P미디어스트리밍은 P2P파일공유에 비해 도전해야할 과제가 많다. 그 이유는 P2P네트워크를 구성하는 각 피어는 서로 다른 네트워크대역폭, 업로드속도, 컴퓨터성능을 가지므로, 임의의 하나의 피어로부터 데이터를 수신하면, 사용자의 QoS를 만족시키지 못할 가능성이 높기 때문이다. 뿐만아니라 P2P네트워크의 특성상, 피어는 언제든 온라인과 오프라인을 반복할 수 있어 피어의 네트워크 이탈이 잦다. 따라서 하나의 소스로부터 데이터를 받는 기존의 P2P패러다임을 P2P미디어스트리밍에 효과적으로 적용하기는 어렵다[3].

현재, 미디어스트리밍의 주체인 송신 피어(이하 소스피어) 여러 개로부터 미디어를 수신하는 멀티소스스트리밍의 연구가 진행되고 있다. P2P 멀티소스스트리밍에 관한 연구로는 PALS[2], GNUSTREAM[3],[4] 등이 있다. PALS는 가용한 대역폭의 변화에 따른 품질 저하를 최소화하기 위한 적응(adaptation) 이론을 제시하였고, [4]는 인터넷 상에서의 멀티소스스트리밍 알고리즘을 제시하였지만 순수 P2P환경에는 적합하지 않다. GNUSTREAM에서는 그누텔라와 연동하여 계층적 버퍼를 통해 멀티소스스트리밍 시스템을 구현하였지만, 스트리밍 이후에 파일로 저장하지 않고 소비하며, 스트리밍 중인 피어가 가진 미디어파일의 일부분을 전송하는 메커니즘이 없다.

한편, P2P네트워크는 일반PC에서 서버에 이르기까지 다양한 사양의 컴퓨터로 구성되며, 각 피어는 서버와 클라이언트 역할을 모두 수행할 의무를 가진다. 즉, P2P네트워크 상의 피어는 컨텐츠를 제공함과 동시에 원하는 컨텐츠를 다른 피어로부터 공급받는다. 나아가 순수한 P2P네트워크에서는, CDN (Contents Delivery Network)과 달리 컨텐츠를 제공하는 주체가 고정되어 있지 않으므로 컨텐츠 분배는 매우 중요한 서비스가 된다. 따라서 P2P미디어스트리밍에서 네트워크를 통해 컨텐츠를 분배하기 위해서는 스트리밍과 파일저장을 동시에 실행할 수 있어야 한다. 그리고 특정 미디어파일을 소유한 피어의 수가 적으면 멀티소스스트리밍의 소스피어의 수도 적어지므로 특정미디어에 대한 요청 수를 충족시키지 못할 수도 있다.

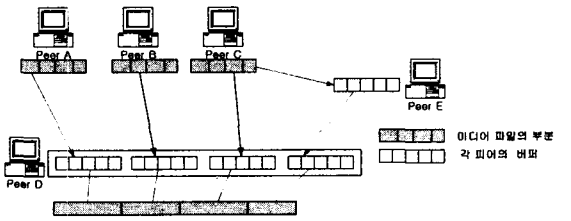
본 논문에서는 미디어송신, 미디어재생, 파일저장을 동시에 할 수 있는 P2P멀티소스미디어스트리밍 시스템아키텍처를 제시하여, 요청피어가 데이터를 수신, 재생, 저장함으로써 스트리밍과 동시에 소스피어의 역할을 수행하게 한다. 이로써 사용자 응답시간이 단축되고 네트워크를 통한 컨텐츠 분배도 가능하다. 그리고 피어가 가진 미디어파일의 일부분을 전송하는 메커니즘을 적용하여 멀티소스스트리밍 시 소스피어의 수가 늘어남에 따라 비트전송률을 높이고 동시수요자 수가 증가한다.

2. 확장성 있는 미디어 스트리밍

본 논문에서 제안하는 확장성이 있는 멀티소스미디어스트리밍 기법의 기본 동작, 피어간 통신단계, 그리고 단계별 알고리즘을 살펴본다.

2.1 기본 동작

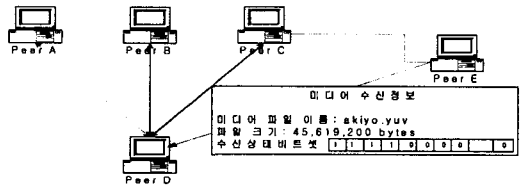
멀티소스스트리밍시스템에서 미디어스트리밍 예는 <그림 1>과 같다.



< 그림 1 > 피어간 미디어스트림 흐름의 예

<그림 1>에서 PeerD는 수신피어로서 클라이언트 역할을, PeerA, B, C는 소스피어로서 서버 역할을 하고 PeerE는 PeerC로부터 미디어스트림을 수신한다. 이때 PeerA, B, C는 각각에

게 배정된 미디어파일의 일정부분을 PeerD에 송신하며, PeerE는 자신이 수신, 저장하고 있는 미디어데이터를 PeerD로 전송한다. PeerD는 PeerA,B,C,E로부터 받은 데이터를 버퍼의 피어별 해당 구간에 쓰고, 미디어재생용 버퍼로 보내 재생한다. PeerE는 송수신을 동시에 수행하며, PeerD는 PeerE로부터 미디어수신상태비트셋을 <그림 2>와 같이 전송받아 PeerE의 미디어 저장상태를 확인하여 자신의 미디어수신상태비트셋과 비교함으로써 다음 미디어 요청을 결정하게 된다.

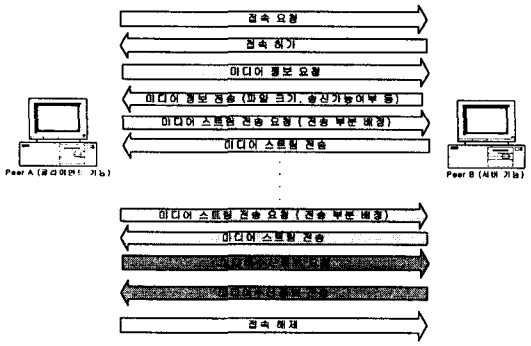


< 그림 2 > 미디어수신상태정보 교환

2.2 피어간 통신 및 알고리즘

2.2.1 피어간 통신과정

피어간 통신은 미디어파일이 있는 피어의 IP를 안다는 가정에서 이루어지며, 크게 미디어정보 요청-응답, 미디어스트림 요청-전송, 미디어수신상태정보 갱신요청-전송 단계로 구분되며, 통신 과정은 <그림 3>과 같다.



< 그림 3 > 피어간 통신과정

미디어정보 요청에 대한 응답을 보낼 때는 요청피어에게 전송가능여부, 파일정보를 보낸다. 해당 미디어파일이 스트리밍 중인 파일이면 미디어파일의 수신상태를 비트열로 표시하여 전송한다. 미디어정보 패킷을 수신하면 요청피어는 각 소스피어에게 전송부분인덱스를 보내 미디어스트림전송을 요청한다. 미디어스트림전송 요청을 받은 소스피어는 해당 미디어파일부분을 요청피어에게 송신한다. 미디어수신상태정보는 동적으로 소스피어에 요청하고 소스피어의 미디어수신상태정보를 갱신하여 미디어스트림전송 요청을 지속한다.

2.2.2 통신단계별 알고리즘

· 미디어정보 요청

```
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
```

미디어정보요청단계에서는 미디어 파일이 있는 소스피어에게 원하는 파일명을 전송한다. 소스피어의 수가 제한된 값을 초과할 때는 선택 정책에 의해 선별된다.

· 미디어정보 응답

```
if Media file is exist
  if State of Media file is an Entire File then
    Message = PacketSize + Type(File_Info_Ack) + State
              of File(Entire File) + File Name (Requested
              FileName) + FileSize
  else if State of Media file is not an Entire File then
    Message = PacketSize + Type(File_Info_Ack) + State
              of File(Not entire File) + File Name (Re
              -quested FileName) + FileSize
              + BitStreamIndex

    Send message to request peer
```

미디어정보응답단계에서는 완전한 파일이 존재하면 파일의 송신가능여부와 파일정보(파일크기)를 전송하며, 스트리밍중일 때는 전체 파일크기와 현재 수신상태비트셋(BitStream _Index)을 함께 전송한다.

· 미디어스트림 요청

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

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

  Request again after checking my Bit_Stream_Index
```

미디어스트림요청단계에서는 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.

· 미디어스트림 전송

```
if check Start Index and End Index then
  Message = PacketSize + Type(FILE_STREAM_ACK)
            + Send Point + Offset + Content Size
            + Media Data

  Send message to request peer
```

미디어스트림전송단계는 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.

· 미디어수신상태정보 갱신요청

```
if BitStream_Index have to update then
  Message = PacketSize + Type(Update_bitStream_index_req)
            + FileName

  Send message to request peer
```

소스피어의 미디어수신상태비트셋을 갱신해야할 시점이 되면 소스피어에게 미디어수신상태정보 갱신요청을 한다.

· 미디어수신상태정보 전송

```
Message = PacketSize + Type(Update_bitStream_index_Ack)
          + FileName + Bit_Stream_index

Send message to request peer
```

미디어수신상태정보갱신요청이 오면, 미디어수신상태정보전송단계에서는 현재 미디어수신상태 비트셋을 '0'과'1'의 비트열로 변환하여 요청피어에게 전송한다.

· 미디어 파일 저장

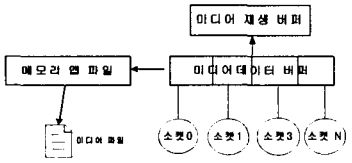
```
if All part of media file is not received then
  Request again

if All part of media file is received(state bit is all 1) then
  Save memory-mapped-file as a file
```

미디어파일 저장단계는 요청피어가 모든 미디어파일을 수신하였을 때 수행된다. 자신의 미디어수신상태비트셋을 체크하여 모든 비트가 '1'일 때 실제 파일로 저장한다.

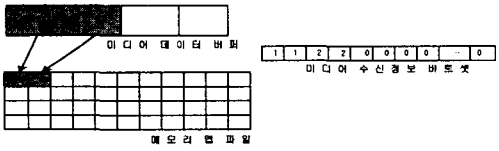
2.3 구현

본 P2P멀티소스스트리밍 시스템은 WindowsXP 상에서 Visual C++와 C로 구현하였다. 모든 피어는 서버스레드, 클라이언트스레드 그리고 미디어재생스레드를 갖는다. 서버스레드는 다른 피어의 요청 명령을 수행하는 루틴이며, 클라이언트스레드는 소스피어의 명령을 수행하는 루틴이다. 모든 피어는 이 스레드들과 <그림 4>와 같은 계층적 버퍼를 갖고 미디어스트리밍, 미디어데이터송신, 미디어파일저장, 미디어재생을 동시에 수행한다.



< 그림 4 > 버퍼 계층

각 소스피어로부터 온 미디어데이터는 클라이언트 소켓을 통해 소켓 별로 배정된 미디어데이터버퍼구간에 쓰여진다. 이때 미디어데이터버퍼의 크기는 소스피어의 수에 따라 동적으로 바뀌며, 해당 미디어데이터버퍼구간이 다 차면 미디어재생버퍼로 데이터가 복사되고, 메모리맵파일의 해당 위치에 쓰여진다.

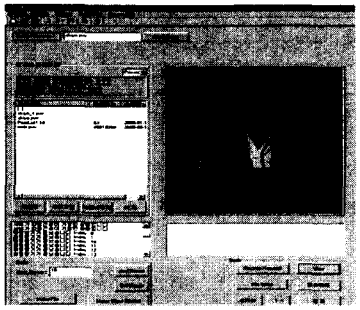


< 그림 5 > 버퍼와 메모리맵파일 간의 데이터이동

미디어데이터버퍼와 메모리맵파일 사이의 데이터는 <그림 5>과 같이 이동된다. 미디어데이터버퍼의 해당구간이 차면, Start Index와 End Index값에 따라 메모리맵파일의 정해진 위치에 쓰여진다. 이때 미디어수신상태정보 비트셋에 미디어파일 해당구간의 수신상태를 표시하여 미디어스트림요청하고 다른 피어에게 미디어 각 부분의 수신상태를 알려주는데 사용된다. 미디어수신정보비트셋은 '0', '1', '2'로 상태를 표시하며, '0'은 아직 요청하지 않은 상태, '1'은 수신완료 상태, '2'는 현재 수신중인 상태를 나타낸다. 요청피어에게 미디어수신정보비트셋을 전송할 때는 '0'과'1'의 비트열로 바꾸어 전송한다.

3. 결과 및 검토

<그림 6>은 소스피어로부터 미디어데이터를 수신하면서 동시에 파일로 저장, 재생하는 화면으로서 각 피어가 미디어의 송수신, 재생, 파일저장 기능을 동시에 수행할 수 있음을 보인다.



< 그림 6 > 미디어송수신 및 동시재생 화면

멀티소스스트리밍 시의 비트전송률을 보기 위해 소스피어의 수를 증가시키면서 100프레임의 YUV데이터(총 크기: 14.5M)를 버퍼링하는 시간을 측정하였다. 소스피어가 1개일 때는 32초로 비트전송률이 0.45Mbps이고, 2개일 때는 25초로 0.58Mbps, 3개일 때는 17초로 0.8Mbps 그리고 4개일 때는 14초로 1.04Mbps의 비트전송률을 보였다. 결과적으로 멀티소스스트리밍을 하는 경우에 비트전송률이 크게 증가함을 알 수 있다. 실험에 사용한 YUV미디어파일은 압축되지 않은 영상으로서 프레임당 152,064 바이트, 초당 24프레임을 기준으로 하면 3,649,536bps의 비트전송률이 요구된다. 따라서 압축영상을 이용하여 실제 환경에 응용하는 것이 필요하며, 이를 위해 MPEG을 지원하는 시스템을 구현한다.

4. 결론 및 향후 연구과제

본 논문에서는 P2P 네트워크 환경을 위한 멀티소스스트리밍 시스템을 설계하고 구현하였다. 제안된 시스템의 각 피어는 미디어송수신은 물론 미디어재생과 파일저장을 동시에 수행할 수 있어 사용자 응답시간이 단축되고 네트워크으로의 컨텐츠 분배가 용이하다. 또한 스트리밍 중인 소스피어의 미디어파일 일부분을 전송하는 기법을 적용하여 비트전송률을 높고 동시 수요자 수가 증가된다. 제안된 시스템에서 MPEG과 같은 압축영상을 지원하고 실제 인터넷 환경에서 실험하고 검증하는 것은 향후 과제로 남긴다.

참고문헌

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

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

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

[4] T. Nguyen and A. Zakhor, "Distributed Video Stream -ing Over Internet", SPIE/ACM MMCN 2002, Jan 2002.