MPEG-4 Layered Video Streaming서비스 제공을 위한 시스템 설계 및 구현

신정아[0] 김현정 김상형 이홍기 이성인 유원경* 유관종
충남대학교 컴퓨터 과학과, 성신여자대학교*
(jasinn[0], hjkim, kimsh, helius, silee, kjyoo)@cs.cnu.ac.kr, wyoo@cs.sungshin.ac.kr*

Design and implementation of a system for providing MPEG-4 layered video streaming service

Jeongah Sinn[0] Hyunjung Kim Sanghyoung Kim Hungki Lee Sungin Lee Wonkyoung Yoo* Kwanjong Yoo
Dept. of Computer Science, Chungnam National University, Dept. of Computer Science, Sungshin Women's University*

요 약

본 논문은 MPEG-4기반으로 안정적인 QoS를 제공할 수 있는 스트리밍 데이터 서비스 시스템의 설계 및 구현 방법을 제한다. 시스템은 서버와 클라이언트로 이루어져 있으며, 서버와 클라이언트에 모두 QoS 관리자가 존재한다. 기존의 기술에서는 네트웍 상황에 따라 미리 나누어진 스트리밍 데이터를 Layer별로 제공하여 QoS를 만족시켰다. 본 논문에서는 QoS 관리자를 통해 전송 중 실시간으로 QoS Level을 측정하여 적용시켰다.

1. 서 론

광대역 네트웍 시대로 발전하면서 인터넷을 이용하는 많은 사용자들의 다양한 멀티미디어 데이터에 대한 수요가 늘어나고 있다. 그러나 광대역 네트웍이라는 이름이 무색하게 멀티미디어 데이터에 대한 제공은 여전히 많은 불만을 불러 일으키고 있다. 이는 제공되는 서비스 컨텐츠(Contents) 자체에 대한 불만이 아니라, 이용하는데 있어서 잦은 버퍼링(Buffering)과 패킷(Packet) 손실이나 네트웍 대역폭에 따른 끊김 현상과 같은 서비스 질에 관련한 것이다[1][2].

이러한 상황으로 인해 멀티미디어 데이터 서비스 시스템에 대한 질적 향상을 위한 QoS(Quality Of Service)에 대한 연구가 지속적으로 이루어지고 있다. 대표적인 연구로 다음 두 가지를 들 수 있다. 하나는 QoS 요구 사항을 채널의 주어진 용량에 맞추기 위하여 대역폭 할당이나 허용 제어, 자원 예약 등의 기법에 대한 연구이고, 다른 하나는 미디어 스케일링 기법을 통해 멀티미디어 데이터의 데이터량을 조절함으로써 채널에 실리는 부하를 경감시키려는 연구이다[3][4].

본 논문에서는 후자인 미디어 스케일링 기법인 Layering 기술을 시스템에 적용하였다. 기존의 Layering 기술은 하나의 스트리밍 데이터에 대해 대역폭에 따른 비트율이 다른 여러 개의 스트리밍 데이터를 중복 저장하여 서비스를 제공했다[3]. 기존 미디어 스케일링 기법을 그대로 이용하지 않고 중복 저장에 따른 저장 공간 비효율성을 해소하려 했다. 하나의 Layered 스트리밍 데이터를 다양한 대역폭으로의 서비스를 가능하게 하고자 멀티미디어 서비스 제공 중에 네트웍 대역폭의 변화를 측정할 수 있도록 했다.

2장에서는 멀티미디어 스트리밍 시스템 구현과 실시간으로 QoS를 제공하기 위한 방법에 대해 알아보고, 3장과 4장에서는 본 논문에서 실시간으로 QoS를 제공하기 위한 구체적인 방법과 구현된 전체 시스템에 대해 서술한다. 마지막으로 5장에서는 결론과 향후 연구 과제에 대해 제시하고 논문을 마치도록 한다.

2. 관련 연구

2.1 멀티미디어 스트리밍 시스템

멀티미디어 스트리밍 시스템을 이루기 위해서는 서버와 클라이언트가 있고, 세부적으로 서비스에 이용하는 데이터를 저장하기 위한 저장 모듈, 서버와 클라이언트 간에 연결을 위한 전송 모듈, 안정적 서비스를 위한 QoS 모듈, 오디오와 비디오 데이터 간 동기화 모듈 및 디코더 모듈, 사용자 인터페이스 모듈이 있다[5].

미디어 서버에 있는 스트리밍 데이터는 대개 스케일러블을 제공하는 방법과 제공하지 않는 방법을 이용한다. 스케일러블을 제공하는 기법에는 SNR, Spatial, Temporal 의 방법이 있다[3][4].

서버와 클라이언트 간에 서비스에 이용되는 통신 프로토콜로는 TCP, UDP, RTP가 있다. TCP를 이용하는 경우 UDP를 병행하고, 멀티미디어 데이터 전송을 위한 프로토콜인 RTP는 컨트롤을 위한 프로토콜인 RTCP, RTSP등과 함께 이용하여 구현한다[2].

2.2 QoS 방법

QoS 방법은 크게 Congestion control과 Error Control로 나뉜다[5]. 전자의 방법은 전송하면서 QoS를 보장해주기 위한 방법이고 후자는 전송이 된 후 나타나는 Error를 처리하기 위한 방법이다.

Congestion Control과 관련된 메커니즘은 Rate Control, Rate Shaping이 있다[5]. Rate Control은 서버에서 클라이언트로의 전송 시 네트웍 대역폭 측정을 통해 전송량을 조절하는 메커니즘의 하나이다. 모듈이 동작하는 위치가 서버측인지 클라이언트측인지에 따라 Source-based rate control, Receiver-based rate control, hybrid rate control 이렇게 3가지 방법을 갖는다.

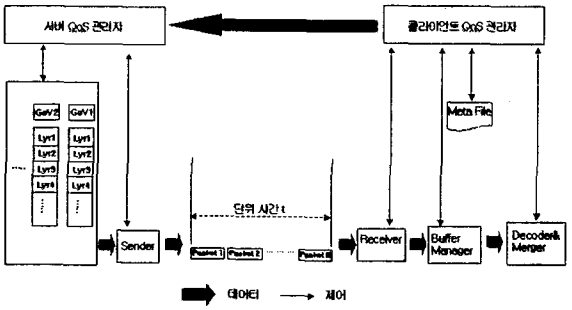
Error Control 메커니즘은 FEC, retransmission, error-resilient encoding, error concealment가 있다[5].

3. 실시간 QoS 제공을 위한 알고리즘

미디어 스케일링 기법에서는 이미 코딩 되어 있는 스트림을 서로 비트율이 다른 여러 버전으로 생성하여 미디어 서버에 유지한다. 이 방식을 따르면 각 세션마다 그의 대역폭을 만족시킬 수 있는 버전의 스트림을 전송함으로써 모든 클라이언트의 QoS 요구 사항을 만족시킬 수 있다[3][4].

기존 기법과는 다르게 본 논문에서는 하나의 Layered 스트리밍 데이터을 이용한다. 즉, 하나의 버전을 생성해 두고 서비스를 요청하는 클라이언트와의 대역폭을 측정하여 제공한다.

실시간으로 대역폭을 측정하기 위해 서버와 클라이언트에 QoS 관리자를 두었다. [그림1]은 본 논문에 적용된 QoS 관리자 모듈을 보여준다.



[그림1] QoS 관리자 모듈 구조

서버의 QoS 관리자와 클라이언트의 QoS 관리자는 각각 다른 일을 수행한다. 전자는 클라이언트로부터 QoS Level 값을 받아 다음 번 GoV 전송에 이용하게 되는데, 서비스가 시작된 처음에는 GoV 전체를 보내는 것으로 초기화되어 있다. 후자는 서버로부터 서비스를 받으면서 QoS Level을 계산하고 다시 서버로 보내준다.

클라이언트의 QoS 관리자는 서비스 초기에 받은 메타 파일, 자신이 수신한 패킷의 양, 그리고 클라이언트의 버퍼 크기를 바탕으로 하여 QoS Level을 설정한다.

메타 파일에는 전송될 Layered 스트리밍 데이터의 GoV Number, Sequence Number, Start Time, Size 등이 저장되어 있다. 수신되는 패킷의 양과 가용 버퍼의 크기는 버퍼 관리 모듈에서 체크되어 QoS 관리자 모듈로 보내져 QoS Level 계산된다.

이렇게 만들어진 QoS Level 값은 다시 서버의 QoS 관리자로 전송되어진다. 서버의 QoS 관리자에서는 다음 GoV 전송에 클라이언트로부터 받은 QoS Level을 이용한다.

[표1]은 본 논문에서 구현한 클라이언트의 QoS 관리자에서 QoS Level을 계산하는 알고리즘이다.

```
LastLayer = 메타 화일로부터 현위치
mGoV = 현재 GoV 개수를 하나 증가.
for(Layer의 Temporal의 개수만큼 loop)
{
  // 현재 위치에서의 Spatial Number 만큼 Loop
  // 실제 받은 패킷 사이즈와 Info 화일에서의
  // 사이즈를 비교해서 현재까지 받아들인 Spatail
  // Number로 QoS Level을 셋팅한다.
  for(Layer의 Spatial의 개수만큼 loop)
  {
LyrSize = 메타화일을 통해 알게된 현재 Layer의 본래 크기
LyrSqnNo=LyrSize로 현재 받은만큼의 패킷 사이즈를 나눔
RecvSqnNo=현GoV,layer 스트림에 따른 시퀀스 number
  if( LyrSqnNo와 RecvSqnNo이 같지 않을 경우 )
    Break ;
  }
  if(패킷이 도착한 경우)
      계산된 QoS Level로 셋팅한다.
  else
    for (Layer의 Temporal의 개수만큼 loop)
      하위 레이어의 QoS Level을 -1로 셋팅한다.
}
```

[표 1] Qos level 계산 알고리즘

4. 시스템 설계 및 구현

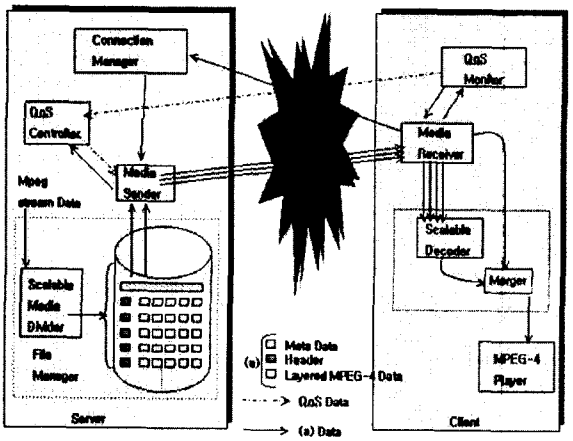
시스템의 구현 환경은 다음과 같다. OS는 Windows 2000 Professional을, 사용 언어는 C++로 MS의 Visual Studio 6.0을 기반으로 하였으며, Console 과 window API 두 가지를 사용했다. Console에 보여지는 부분은 서버와 클라이언트의 송수신 상황 체크 및 클라이언트 측의 디코딩과 병합 체크를 하도록 구성되어 있고, Console 동작 중 플레이 가능한 양의 버퍼링 완료 후 윈도우 새 창을 띄워 수신된 데이터를 플레이하도록 하였다.

3.1 전체 시스템의 구조

다음 [그림2]는 본 논문에서 구현한 시스템의 전체 구조이다.

스트리밍 데이터는 오프라인상에서 Layering 기술이 적용되어 서버 쪽에 저장되어 있다.

클라이언트로부터 서비스 요청이 들어오면 서버에서는 요청 스트리밍 데이터의 메타 파일을 전송한다. 이후 Layered 스트리밍 데이터를 GoV 단위로 전송한다. 메타 파일과 QoS Level 정보 전송에는 TCP를 이용하였고, 스트리밍 데이터 전송에는 UDP를 이용했다.



[그림2] 전체 시스템 구조

수신한 스트리밍 데이터는 파일 버퍼에 저장이 되고, 병합기로 넘겨진 데이터와 유실되거나 뒤늦게 도착하여 재사용이 불가능한 파일을 관리하기 위해 파일 버퍼 내 저장된 위치를 관리하는 메모리 버퍼를 두었다. 파일 버퍼와 메모리 버퍼 두 가지를 이용하는 경우, 클라이언트의 버퍼 상황에 따라 유용하다. 버퍼 크기가 작은 경우, 메모리 버퍼를 통해 파일 버퍼를 재사용함으로써 버퍼 크기의 한계를 어느 정도 극복 가능하다.

병합기에서는 버퍼에서 수신된 Layered 스트리밍 데이터를

읽어 디코딩과 병합 작업을 한다. 플레이어에서는 병합기에서 작업이 끝나 버퍼에 저장되어 있는 데이터를 보여주는 역할을 수행한다.

3.2 각 모듈 구조

● MPEG-4 Divider

프레임마다 존재하는 프레임 간의 의존성을 바탕으로 I, P, B Picture로 나누는 시간 스케일러빌리티와, Block의 정보에서 저주파 영역과 고주파 영역을 나누어 화면의 변화를 나타내는 공간 스케일러빌리티를 이용하여 15개의 Layer를 생성할 수 있다[1]. 하나의 스트림을 여러 개의 Layer로 나누는 것은 네트웍 상황에 맞춰 이질적 환경을 가지는 클라이언트에 안정적으로 제공함에 목적이 있다.

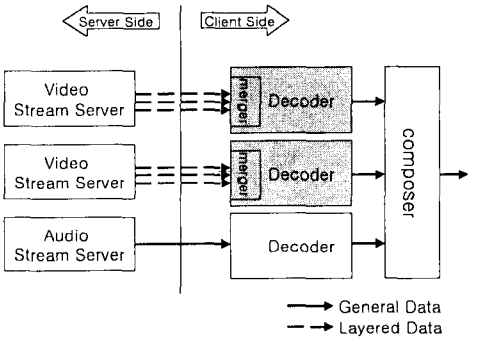
이러한 개념을 바탕으로 하여 본 논문에서 적용된 Layering 방식은 기본적으로 15개의 Layer 내에서 다양한 개수(N)의 Layer를 생성하여 서비스하는 MPEG-4 N-Layer Divider를 적용하였다. 본 논문에서는 12개의 Layer로 생성하여 테스트하였다.

● 병합기와 디코더 모듈

일반적으로 Layered 스트리밍 데이터를 다루는 시스템에서는 클라이언트 측에 Layered 데이터를 재조합 하는 병합기 모듈과 조합된 데이터를 디코딩하는 디코더 모듈이 필요하다.

기존에 수행하던 MPEG-2의 연구에 이용된 모듈은 병합이 이루어진 뒤 디코딩이 시작되는 형태로 두 개의 모듈은 따로 동작을 하였다. 따로 동작하다 보니, 병합기에서 Layered data 병합을 위해 하나의 데이터를 최하위 Layer까지 파싱(Parsing)하고 다시 디코더 모듈에서 같은 작업을 진행한다. MPEG-4 스트림의 크기가 큰 경우, 클라이언트의 CPU 자원을 낭비하게 되고 이로 인해 병합기 모듈에 할당된 자원이 디코더 모듈의 성능 저하를 가져오는 단점이 된다[1].

기술한 단점을 해결하기 위해 본 논문에서는 [그림2]과 같이 기존에 따로 동작하던 병합기와 디코더 모듈을 하나로 만들어 클라이언트에 적용하였다.



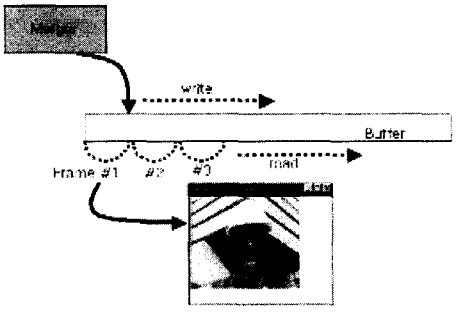
[그림3] 병합기/디코더 모듈

● 플레이어 모듈

병합기에서 디코딩과 머징을 마친 스트리밍 데이터가 플레이 버퍼에 일정량이 쌓이면 플레이어가 실행된다. YUV 형식의 데이터를 RGB 형식의 데이터로 변환해주고 한 프레임의 사이즈씩 읽어 들여 윈도우 창에 뿌려준다.

장면 플레이는 여러 장의 프레임을 시간에 맞춰 계속 바꿔주는 형태로 마치 노트에 사람의 동작을 시간 대별로 분리해서 그린 후, 노트를 빠르게 넘기면 하나 하나의 분리된 동작이 하나의 움직임이 있는 동작으로 보이는 것과 같다.

[그림3]은 버퍼링 대기와 플레이 동작을 보여준다. [그림3]에서 점선으로 된 화살표의 방향은 시간의 흐름을 의미한다. 병합기에서 Write를 시작한 후에 플레이어에서 Read를 한다.



[그림4] 디스플레이어 동작 구조

5. 결 론 및 향후 연구 과제

본 시스템은 기존의 기술과는 달리 실시간으로 대역폭을 체크하여 Layering 기술을 적용했다. QoS Level에 따라 대역폭에 맞도록 전송량을 변화시킴으로 클라이언트 환경에 따라 달라지는 환경에 적응한 서비스를 가능하게 QoS를 향상시켰다.

병합기와 디코더를 합하여 기존의 클라이언트 시스템에서 소비하는 프로세싱 크기가 적고, 파일 버퍼와 메모리 버퍼의 사용으로 버퍼 크기의 한계를 어느 정도 보완할 수 있다.

그러나, 클라이언트에서 QoS Level 측정 후 다시 서버에 측정치만큼의 데이터를 요청하도록 구현하였기 때문에 QoS 정보를 서버측으로 전송 시 네트웍 상태가 급변한 경우에 대한 대책이 미흡하다.

향후, QoS를 좀더 높은 질로 제공하기 위해서 다른 형태의 실시간 전송 상태 체크 방법에 대한 연구와 유선 환경에 적용한 시스템을 무선으로 확장하는 연구가 필요하다.

참고문헌

[1] 이홍기," 적응형 MPEG 시스템의 발전된 Layering Algorithm" ,한국정보과학회 충청지부 학술발표논문집 제13권 1호,2001년

[2] 박대훈," 비디오 스트리밍 데이터 전송시 RTCP를 이용한 효율적인 네트워크 트래픽 제어" ,정보과학회논문지 제 8권 3호,2002년

[3] 김형철," QoS 적응형 MPEG 비디오의 미디어 스케일링 방안 고찰" , 멀티미디어학회지 제3권 제1호, 1999년

[4] 이홍기, "이질적 환경에서의 Layering 되어진 멀티미디어 스트림 서비스, 정보과학회 춘계학술발표논문집 제 29권 제 1호, 2002년

[5] Dapeng Wu," streaming video over the Internet:Approaches and directions" ,IEEE transactions on circuits and systems for video tech,vol.11,no.3, 2001