RTP 기반의 H.263 스트리밍 시스템 구현

박 일환○ 차 호정 김혁만†
광운대학교 전자계산학과
†한국통신 멀티미디어연구소

Implementation of a RTP-based H.263 Streaming System

Ilhwan Park, Hojung Cha and Hyeokman Kim†
Dept. of Computer Science, Kwangwoon University
†Multimedia Tech. Research Lab., KTR&DG, Korea Telecom

요 약

본 논문에서는 H.263 비디오 코덱을 사용한 인터넷 스트리밍 서버와 클라이언트의 구현을 기술한다. 서버와 클라이언트간의 데이타 전송은 RTP 실시간 프로토콜을 사용하여 이루어진다. 스트리밍 서버는 저장된 H.263화일 또는 입력받은 비디오를 실시간으로 H.263으로 엔코딩하여 멀티캐스팅하고, 클라이언트는 멀티캐스트 세션에 접속하여 서버가 전송한 H.263 데이타를 받아 실시간으로 디코딩한다.

1 서론

멀티미디어에 대한 관심이 컴퓨터, 통신, 방송, 오락등의 영역으로 급속히 확산됨에 따라 제반 기술이 급속도로 발전되고 있다. 멀티미디어를 위한 국제 표준중, 비디오 압축을 위한 코덱으로 MPEG과 H.261/H.263이 중요한 위치를 차지하고 있다. MPEG의 경우, CD-ROM등의 저장 매체를 통한 동영상의 저장 및 재생을 위한 비디오 코덱으로서 높은 대역폭을 요구하여 인트라넷을 대상으로한 VOD 시스템의 구현에 사용되나, 낮은 대역폭을 요구하는 인터넷 환경에서는 사용하기 어렵다. H.261/H.263[1]은 초기에 ISDN을 위한 화상회의용 비디오 코덱으로서 개발되었으나, 화질 개선 및 효율적인 압축 알고리즘의 개발로 인해 도난 방지용 감시 시스템이나 인터넷 방송등과 같은 다양한 응용 분야에 널리 사용되고 있다. 한편, 멀티미디어 압축 기술의 발전과 더불어 실시간 미디어의 전송에 대한 관심이 커지고 있다. 비디오나 오디오와 같은 실시간 미디어는 데이타 손실을 있더라도 정해진 시간내에 전송이 이루어져야하는 특성을 가지는데, TCP 등과 같은 기존의 통신 프로토콜은 이에 적합하지 않다. 최근 멀티미디어 데이타의 실시간 전송 및 보장을 위한 다양한 통신 프로토콜들이 개발되고 있는데, RTP(Realtime Transport Protocol)[2] 는 대표적인 어플리케이션 레벨의 실시간 통신 프로토콜이다.

본 논문에서는 저대역폭의 인터넷 환경에서 RTP 통신 프로토콜에 기반한 H.263 스트리밍 서버 및 클라이언트의 구현을 기술한다. 서버와 클라이언트는 윈도우즈95상에서 비주얼 C++를 사용하여 구현되었다. 서버 및 클라이언트의 실시간 H.263 엔코딩 및 디코딩을 위해 TMN 소스 코드를 수정하였다. 본 논문의 구성은 다음과 같다. 2장에서는 전체 시스템의 구성과, 서버 및 클라이언트의 구현에 사용된 RTP와 H.263에 대해 간단히 기술한다. 3장에서는 서버와 클라이언트의 구조 및 구현 방법을 기술하며, 4장에서 결론을 맺는다[1]

2 RTP 기반의 H.263 스트리밍 시스템

H.263 스트리밍 시스템은 저장되어 있는 H.263 화일 또는, 입력받은 비디오를 실시간으로 엔코딩해 멀티캐스팅하는 서버와, 서버쪽에서 전송한 데이타를 받아 사용자에게 보여주는 클라이언트로 나누어진다([그림 1]) 서버와 클라이언트간의 데이타 전송

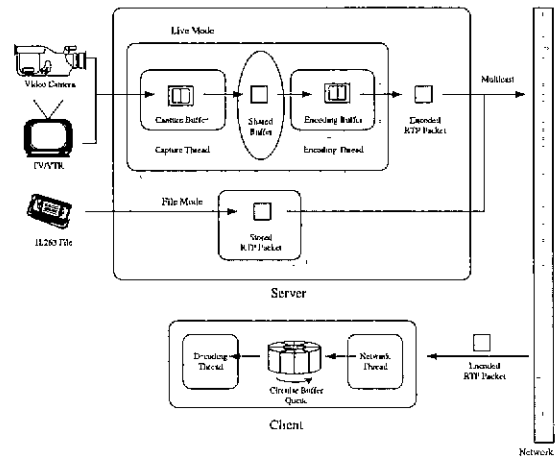

그림 1: 전체 시스템 구조

에는 RTP가 사용된다. RTP는 오디오, 비디오와 같은 실시간 미디어를 전송하기 위해 개발된 프로토콜로서, TCP/UDP와는 달리 사용자 어플리케이션 레벨에서 구현된다. RTP를 사용하는 대표적인 어플리케이션으로 화상회의를 위한 vic/vat와 MPEG을 사용하는 인트라넷 방송을 위한 IP/TV가 있다. RTP는 두개의 부분으로 나누어지는데 데이타 전송에는 RTP가 사용되고, 데이타 전송 제어에는 RTCP(Realtime Control Protocol)가 사용된다.

[그림 2]는 RTP의 구조를 나타내고 있다. M은 마커비트를 의미하며 오디오 데이타와 비디오 데이타에 따라 의미가 달라지는데, 비디오 데이타의 경우 한 프레임의 구분에 사용되므로 한프레임의 데이타를 전달하는 패킷들중 마지막 패킷의 M 비트를 1로 셋팅함으로써 그 프레임의 마지막 패킷임을 나타낸다. PT는 RTP 패킷이 전달하는 미디어의 종류를 나타내는 페이로드타입을 뜻하며, 서버에서는 H.263을 나타내는 페이로드타입인 34로 셋팅을 한다. sequence number는 각각의 패킷의 손실유무를 체크하고 순서화 기능을 제공하기 위해, 전송되는 패킷마다

[1]본 연구는 한국통신의 '98년도 정보통신기초 연구사업(관리번호 98-19)에 의해 지원받았음

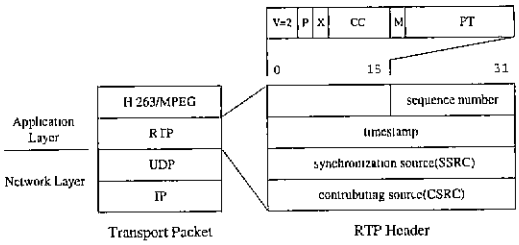

그림 2· RTP 헤더 구조

연속적으로 1씩 증가한다 timestamp는 프레임이 엔코딩된 시간을 반영하며, 동기화 기능과 지터측정에 사용되고, 서버에서는 하드디스크에 저장된 데이타를 정확히 엔코딩된 프레임 레이트대로 전송하기 위해 사용한다 SSRC는 여러세션을 통해 패킷이 전달될때 각각의 패킷을 구분하기 위해 중복되지 않는 32비트 값을 나타내는데, 서버에서는 MD5 알고리즘[3]을 사용하여 생성된 랜덤한 32비트 값을 SSRC에 할당한다.

MPEG은 CD-ROM등의 저장 미디어를 대상으로 하고, H.261/H.263은 화상회의나 영상전화와 같은 통신 미디어를 대상으로 한다. 또한 MPEG의 경우에는 디코딩시에는 실시간성이 요구되지만 엔코딩시에는 반드시 실시간성을 요구하지는 않는 반면에 H.261/H.263의 경우에는 양방향 통신의 특성상 엔코딩과 디코딩시에 모두 실시간성을 요구한다 화상회의나 영상전화와 같은 프로그램에서는 엔코딩과 디코딩시의 지연시간을 약 150ms이하가 되도록 권하고 있다 H.261/H.263의 경우는 다음과 같은 방식으로 압축이 행해진다 하나의 프레임에 대한 정지화상의 압축은 DCT에 의해 이루어지는데 그 단위는 매크로블록이 되며, 프레임간의 압축은 움직임 보상에 의해 이루어진다. 움직임 보상이나 DCT에 의해 생성된 비트열을 호프만코딩에 기초한 엔트로피 코딩을 통해 비트열을 줄이고 양자화에 의해 전체 비트열을 줄임으로써 압축을 끝마치게 된다. 서버와 클라이언트에서의 비디오 엔코딩과 디코딩에는 H.263 코덱[1]이 사용되는데, H.261 비디오 코덱을 확장하여 개발된 코덱으로 H.261에 비해 동일한 화질에서 상대적으로 적은 비트를 발생하기 때문에 대역폭을 절약할 수 있다. 또한 H.261에 비해 더욱 다양한 픽쳐 포맷을 지원하며, 움직임 보상 알고리즘이 향상되었고, 새로운 프레임 개념이 소개되였다.

H.261/H 263에서는 한프레임을 픽쳐층, GOB(Group of Block)층, 매크로블록층, 블록층으로 나눈다([그림 3]) 하나의 프레임을 픽쳐층이라 하며, 하나의 픽쳐층은 CIF의 경우에는 12개 QCIF의 경우에는 3개의 GOB층으로 나누어진다 GOB층은 매크로블록층의 모임으로 총33개의 매크로블록층을 포함하고있고, 매크로블록층은 휘도정보를 가지고있는 4개의 블록과 색상정보를 가지고있는 각각 1개씩의 블록을 가지고 있다 따라서 H 261/H.263에서의 가장 기본이 되는 단위는 블록이다 MPEG에서의 I,P,B 프레임구성과는 달리 H 261/H.263의 경우에는 I,P 프레임만으로 구성되어있다. I 프레임은 인트라 프레임이라고도 하며 하나의 완전한 프레임을 나타내고, P 프레임은 인터프레임이라고 하는데 I 프레임을 통해 얻어지는 변화된 부분이 반영된 프레임이다. H.261/H.263의 특성상 계속해서 P프레임이 전송될때, 중간에 프레임이 없어지는 경우에는 그다음부터의 전 프레임에 대한 변화가 제대로 반영이 안되기 때문에 화면이 깨지게 된다 따라서 최소한 132프레임에 한번은 인트라 프레임으로 엔코딩되어야 한다[5]. H 263에서는 PB-프레임[1, 4]이라는 새로운 개념의 프레임이 소개되었는데, 개념적으로 MPEG에서의 P,B 프레임을 묶어서 하나의 PB-프레임이라는 단위로 이루어진다 PB-프레임은 반드시 H.263 프레임에 포함되는 것이 아니고 H.263 비트열의 PTYPE에서 13번째 비트를 1로 셋팅함으로써 사용할 수 있다.

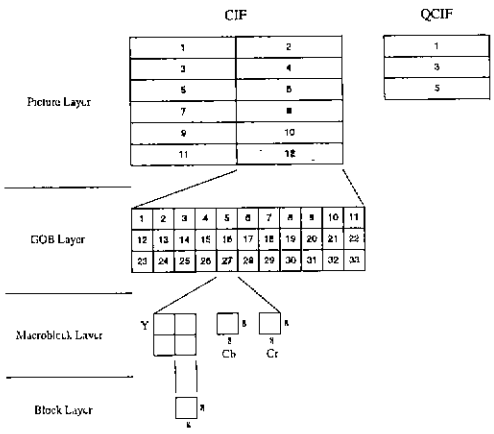

그림 3 H.263의 프레임 구조

3 설계 및 구현

다음에는 서버와 클라이언트의 구조 및 구현 방법을 기술한다.

3.1 서버 구성

서버는 윈도우즈 95상에서 비주얼 C++를 사용하여 작성하였다. 멀티캐스팅을 위한 입력으로서 비디오 카메라를 통한 입력, TV/VTR을 통한 입력, 그리고 H 263 화일을 통한 입력의 3가지의 입력모드를 가지고 있다. 비디오 카메라를 통한 입력과 TV/VTR을 통한 입력의 경우에는 입력받은 비디오를 실시간으로 H 263으로 엔코딩하는 기능을 가지고 있다. 서버는 비디오 캡쳐보드를 통해 비디오를 캡쳐하는 부분과, 입력받은 비디오를 엔코딩하여 멀티캐스팅하는 부분으로 나누어진다. 입력받은 프레임을 엔코딩하는 부분은 TMN 엔코더를 수정하여 작성하였다 단일 쓰레드를 사용하여 캡쳐와 엔코딩을 순차적으로 할경우 캡쳐와 엔코딩에 소요되는 시간의 차이로 인해 CPU 사용량이 남는 상황에서 프레임이 스킵되는 상황이 발생한다. 따라서 서버에서는 캡쳐와 엔코딩을 두개의 쓰레드로 나누어 각각 독립적으로 동작이 이루어지도록 구성되었다 [그림 4]는 이러한 두개의 분리된 쓰레드 구조를 나타내고 있다

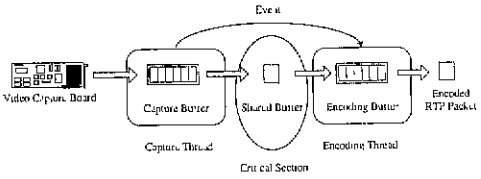

그림 4: 서버의 캡쳐/엔코딩 쓰레드 구조

비디오 캡쳐 쓰레드에서는 한프레임에 해당하는 비디오 데이타를 캡쳐하여 공유버퍼에 복사를 하고, 엔코딩 쓰레드에서는 공유버퍼에서 비디오 데이터를 가져와 자체내에 있는 엔코딩 버퍼에 복사를 하고 이 데이타를 이용하여 엔코딩을 시작한다

서버의 또다른 입력모드는 화일모드이다 서버에서는 H.263 화일과, RTP 패킷으로 라이브 엔코딩된 데이타를 저장하고, 이중 RTP 패킷으로 저장된 화일이 서버의 또다른 입력모드가 된다. 정확한 프레임 레이트대로 멀티캐스팅하기 위해 RTP 패킷의 타임스탬프가 사용된다. RTP 패킷의 타임스탬프는 해당 프레임이 엔코딩된 시간을 나타내기 때문에, 각 프레임간의 시간간격을

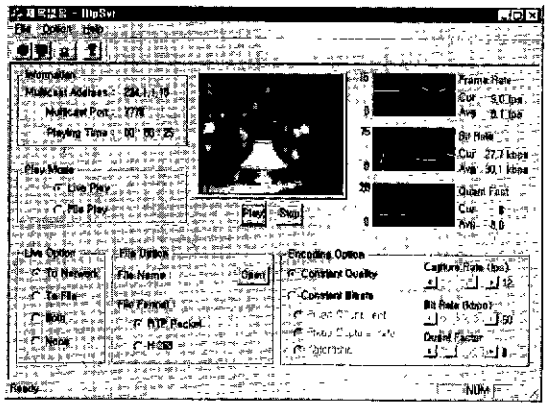

그림 5: 서버의 동작화면

정확히 알수있고, 따라서 정확한 프레임 레이트대로 데이타를 멀티캐스팅한다

[그림 5]는 서버의 동작화면이다 서버의 플레이모드는 비디오 캡쳐보드를 통한 라이브 플레이와 화일 플레이로 나누어진다. 라이브 플레이의 경우는 네트워크를 통한 멀티캐스팅, 화일로의 저장, 두가지 모드의 병행, 단순한 캡쳐화면제생의 네가지로 나누어진다 또한 화일저장에 관해 H.263 화일로의 저장과 RTP 페킷형태의 저장기능을 제공한다 플레이 모니터링 윈도우를 통해서 현재및 지난 10초간의 프레임 레이트, 비트율, 양자화 계수를 표현하고, 수치상으로 현재값과 10초간의 평균값을 출력한다 . 엔코딩 옵션을 통해서는 캡쳐 레이트, 비트율, 양자화 계수의 변경을 제공하고 이값들을 통해 엔코딩되는 상태를 조절할 수 있다 캡쳐 레이트를 조정함으로써 엔코딩되는 비디오의 프레임 레이트를 조절할 수 있고, 양자화 계수를 조정함으로써 비디오의 화질을 조절할 수 있다 H.263의 엔코딩 과정중 하나인 양자화에 사용되는 양자화 계수는 상대적으로 작은 값을 사용할 경우 화질이 좋아지는 반면에 압축률은 낮아져 비트율이 높아진다

3.2 전송률 조정

엔코딩 옵션의 조정은 화질을 고정시키는 방법과 비트율을 고정시키는 방식으로 나누어지는데 화질을 고정시키는 방법의 경우에는 화면의 변화량에 따라 비트율의 변화량이 심하다. 따라서 인트라넷 환경에는 적용가능하지만 인터넷 환경에는 다소 부적합하다. 비트율을 고정시키는 방법은 위에서 언급된 캡쳐 레이트와 양자화 계수를 데이타 전송중에 동적으로 변화하여 주어진 비트율에 맞게 데이타를 전송하는 방식이다. 비트율은 다음의 계산식을 통해 계산된다.

$$B = Q * F$$

B : 비트율
Q : 양자화 계수를 사용하여 엔코딩된 프레임의 크기
F 프레임 레이트

위의 식에 의해 현재 프레임의 엔코딩된 비트율 계산한다. 계산된 비트율을 셋팅된 비트율과 비교하여 더 작은 값이면 현재 설정된 캡쳐 레이트와 양자화 계수를 사용하여 엔코딩을 계속한다 만약 계산된 비트율이 셋팅된 비트율보다 크다면 캡쳐 레이트와 양자화 계수를 변경한다. 이 경우 현재 설정된 프레임 레이트를 유지하기 위해 화질을 변경시키는 방식과, 화질을 유지하기 위해 캡쳐 레이트를 변경하는 방식으로 나누어진다 두가지 방식 모두 하나의 인자변경으로도 비트율이 클 경우 다른 인자까지 변경한다.

3.3 클라이언트

클라이언트는 윈도우즈95상에서 비주얼C++를 사용하여 TMN 디코더를 수정하여 작성하였다. 클라이언트는 멀티캐스트 세션을 통해 데이타를 받아 버퍼에 넣는 네트워크 쓰레드와, 버퍼에서 데이타를 읽어와 디코딩하여 화면상에 출력하는 디코딩 쓰레드로 나누어진다 클라이언트의 구현방법에 있어서 한 프레임의 데이타를 받아서 버퍼에 넣는 과정없이 바로 디코더의 입력으로 넣는 방법을 생각해 볼 수 있는데 이와같은 방법의 경우에는 네트워크의 상태에 많은 영향을 받는 반면에 시간 간격을 최소화하여 서버쪽과 클라이언트쪽에서 실시간으로 화면을 볼수있다 현재 클라이언트는 네트워크 상태의 영향을 최소화하기 위해 내부 버퍼를 사용하는 방식을 사용하는데, 시간 간격을 줄이기위해 가능한 최소 크기의 버퍼를 유지한다 [그림 6]은 구현된 클라이언트의 실행모습이다

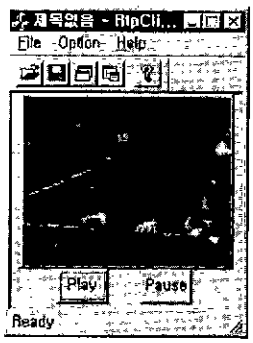

그림 6: 클라이언트의 동작화면

4 결론 및 향후과제

본 논문에서는 RTP에 기반한 H.263 스트리밍 서버와 클라이언트의 구현에 대해 기술하였다. 서버는 비디오 캡쳐보드를 통한 비디오 카메라, TV/VTR의 입력을 실시간으로 H.263으로 엔코딩하여 멀티캐스팅하는 기능을 제공하는데 현재 팬티엄 II 300의 시스템에서 최대 초당 15프레임 이상을 엔코딩할 수 있다 다양한 엔코딩 옵션을 통해 엔코딩 상태를 조절할 수 있고, 전송률 조정에 의해 멀티캐스팅하는 기법을 제시한다. 또한 엔코딩된 데이타를 저장하고, 저장된 데이타를 서버의 입력으로서 전송하는 기능을 제공한다. 현재 저 전송률 오디오 코덱인 G.723 1과의 통합을 위한 연구가 진행중이다.

참고 문헌

[1] ITU-T Recommendation H.263, *http://www.fou telenor.no /brukere/DVC/h263_wht/h263wht.html*

[2] H Schulzrinne, S Casner, R.Fredrick, V.Jacobson, 'RTP A Transport Protocol for Real-Time Application', RFC 1889, Feb. 1996

[3] R.Rivest, 'The MD5 message-digest algorithm', RFC 1321, Internet Engineering Task Force, Apr. 1992

[4] K.R.Rao, J J Hwang, *Techniques & Standards for Image, Video, Audio Coding*' Prentice Hall, 1996

[5] Raymond Westwater, Borko Furht, *'Realtie Video Compression - Techniques and Algorithms*, Kluwer Academic Publishers, 1997