비디오 스트리밍 데이타 전송시 RTCP를 이용한 효율적인 네트워크 트래픽 제어

(An Effective Control of Network Traffic using RTCP for Transmitting Video Streaming Data)

박 대 훈 † 허 혜 선 †† 홍 윤 식 †††

(Dae-Hoon Park) (Hye-Sun Hur) (Youn-Sik Hong)

요 약 네트워크 상에서 비디오 스트리밍 데이타를 전송할 때에는 다른 어플리케이션에 비해 훨씬 큰 대역폭을 차지하게 되며, 이에 따라 같은 네트워크를 사용하는 다른 어플리케이션과의 충돌로 과부하가 발생하게 된다. 본 논문에서는 이러한 문제점을 해결하기 위해 RTP와 RTCP를 이용한 스트리밍 데이타 전송 방식을 채택하였다. 즉, RTCP의 RR(Receiver Report) 패킷을 수신하여 네트워크 트래픽 발생 여부를 실시간으로 조사한다. 트래픽 발생 여부에 따라 JMF에서 사용하는 RTP 인코딩 방식 중 하나인 Motion JPEG의 양자화 계수를 동적으로 조절함으로써 전체 네트워크 트래픽을 줄이고자 시도하였다. 전체 전송량 평균과 세션별 전송량 평균 비율이 5% 범위를 넘어설 때, 각 세션별 전송량을 세션 평균값에 가깝게 동적으로 조절한 결과 과부하가 줄어들 뿐만 아니라 전체적인 전송 효율도 개선됨을 확인할 수 있었다.

키워드 : 비디오 스트리밍, 네트워크 트래픽, 세션별 전송량, 모션 JPEG 양자화 계수, 미디어 서버

Abstract When we want to transfer video streaming data through computer networks, we will have to be allocated a larger bandwidth compared to a general application. In general, it causes a serious network overload inevitably due to the limited bandwidth. In this paper, in order to resolve the problem, we have taken a method for transmitting video streaming data using RTP and RTCP. With RR(Receiver Report) packet in RTCP we will test it to check whether the traffic in a network has occurred or not. If it happened, we have tried to reduce the overall network traffic by dynamically changing the quantization factor of the Motion JPEG that is one of the encoding styles in JMF. When the ratio of the average of transmission for each session to the average of overall transmission is greater than 5%, we should adjust the amount of data to be transmitted for each session to reach the session mean values. The experimental results show that the proposed method taken here reduces the overload effectively and therefore improves the efficiency for transmitting video streaming data.

Key words : RTSP, RTP, RTCP, JMF, Motion JPEG, video streaming

1. 서 론

최근 들어 인터넷의 통신속도가 비약적으로 증가하고 인트라넷에서도 클라이언트/서버 프로그래밍을 통한 다양한 용도의 프로그램이 분산환경에서 실행되고 있다. 또한 사용자의 다양한 요구에 따라 멀티미디어 컨텐츠가 각종 프로그램에서 활용되고 있고 실시간 영상회의와 멀티미디어 컨텐츠가 네트워크를 이용하여 공유되고 있다. 하지만 이런 멀티미디어 컨텐츠는 시간적, 공간적 압축이 되어 있어도 멀티미디어 컨텐츠를 사용하지 않는 프로그램의 네트워크 사용량에 비하여 대단히 큰 대역폭을 차지하게 되고 이에 따라 기존 분산환경의 프로그램은 실행속도와 성능면에서 많은 제약을 받게 된다.

이에 따른 문제점을 해결하기 위하여 시간, 공간적으

· 본 논문은 2001년도 한국과학재단지정 인천대학교 멀티미디어 연구센터 및 인천대학교 교내연구비 일부를 지원받아 연구되었음.

† 학생회원 : 인천대학교 컴퓨터공학과
pparkdh@human.incheon.ac.kr
†† 학생회원 : 인천대학교 정보통신공학과
mshush@human.incheon.ac.kr
††† 종신회원 : 인천대학교 컴퓨터공학과 교수
yshong@incheon.ac.kr

논문접수 : 2001년 9월 3일
심사완료 : 2002년 4월 9일

로 영상을 압축하는 기술과 멀티캐스팅으로 전송하는 방법 등 다양한 형식의 연구가 활발히 진행되고 있다. 대표적인 연구결과로는 멀티미디어 컨텐츠의 특성 중 하나인 실시간으로 재생하거나 일정수준의 프레임을 전송할 수 있도록 설계된 RTP(Real-Time Transport Protocol)와 RTCP(Real-Time Control Protocol)가 있다. 또한 재생 지연시간을 줄이기 위해 전체파일을 다운로드하지 않고 조금씩 전송하는 기법인 스트리밍 기법을 사용할 수 있도록 제안된 RTSP(Real-Time Streaming Protocol)가 있다[1][2][3].

RTP와 RTCP는 실시간 화상회의 시스템에서 많이 이용되는 프로토콜이며 사용되는 코덱으로는 H.263 과 G.723이 있다. 특징을 살펴 보면 적은 네트워크 대역폭으로도 화상회의가 가능하다는 장점이 있지만 화면의 해상도가 좋지 않고 화질이 떨어지며 화면이 열화되는 등 단점을 가지고 있다. 또한 RTCP 프로토콜에 정의되어 있는 SR(Sender Report)과 RR은 멀티미디어 컨텐츠 패킷의 전송량과 지연 속도 등 패킷 전송 상태의 통계치를 제공하지만 이런 통계치에 따른 RTP 패킷의 제어는 프로그램을 통해 직접 구현해야만 한다.

이런 문제점을 해결하기 위해 RTCP 패킷의 전달시간을 동적으로 조절하여 네트워크의 부하량을 조절하는 연구와 TCP 패킷과의 경쟁을 유도하여 RTP 패킷의 대역폭을 확보하는 연구가 있었다[4][5]. 본 논문은 미디어 서버측 화상데이타의 품질이 다소 저하되어도 클라이언트 측 사용자의 시각이 이를 감지하지 못하는 것[6]에 착안하여 네트워크 트래픽에 따른 전송량을 제어하도록 구현하였다.

본 논문에서는 멀티미디어 컨텐츠 전송 프로토콜인 RTP및 RTCP 기반의 미디어 서버를 구현하였다. 세션 상태와 다중접속상태를 유지하기 위하여 RTSP를 사용하였다. 전송량을 가변적으로 제어하기 위하여 일정구간마다의 평균 전송량과 평균지연시간을 추출하고 이를 전체평균시간의 평균값과 비교하여 네트워크 대역폭의 상태를 예측할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 미디어 서버에서 사용하는 RTSP서버와 클라이언트간 교환되는 메시지 형식과 기타 전송방식에 대해 설명한다. 3장에서는 JMF에서 미디어 데이타를 제어하는 방법과 RTSP의 상관관계에 대하여 기술한다. 4장에서는 네트워크 전송량을 제어하기 위한 방법과 지연시간에 따른 상관 관계에 대해 설명하고, 5장에서는 실험결과를 보여주며, 6장에서 결론을 맺고자 한다.

2. RTSP

RTSP(Real-Time Streaming Protocol)는 멀티미디어 전용 프로토콜이다. 즉, 유니캐스트 및 멀티캐스트를 모두 지원하는 어플리케이션 계층의 프로토콜이다. RTSP는 멀티미디어 서버를 위한 원격제어와 오디오나 비디오 같은 연속적인 미디어 스트림을 동기화 한다[7].

RTSP는 제어와 실시간 전송을 위해서 RTP의 상위 계층에서 설계되었다. 따라서 RTP의 규격이 바뀌거나 추가되어도 RTSP에서 계속적으로 사용할 수 있는 장점이 있다.

RTSP 프로토콜 형식은 구조적으로 HTTP와 매우 유사하다. 구문과 작동 방법이 유사하며 대부분의 경우 HTTP 메시지를 기본으로 RTSP가 이를 확장하였다. 그러나 RTSP와 HTTP의 중요한 차이점은 HTTP는 상태를 유지하지 않는 프로토콜인 반면에, RTSP는 전송하는 멀티미디어 컨텐츠의 상태를 가지고 있는 프로토콜이기 때문에 전송 제어를 쉽게 할 수 있는 장점이 있다.

RTSP 프로토콜은 여러 가지 메소드를 가지고 있어 다양한 상태에서 부가적인 정보를 획득할 수 있도록 정의되어 있다. 표 1은 RTSP에서 사용하는 대표적인 메소드들이다.

표 1 RTSP에서 사용하는 메소드

메소드	용도
Options	클라이언트가 명령을 전송하기 전 서버가 응답할 수 있는 메소드들의 리스트를 획득하기 위해 사용된다
Describe	클라이언트가 서버에 저장되어 있는 미디어나 실시간 컨텐츠를 요청하기 전에 서버에 있는 멀티미디어 컨텐츠 형식과 정보를 확인하고 싶을 때 사용된다
Setup	클라이언트와 서버간 RTP와 RTCP 패킷을 주고 받을 수 있는 포트번호와 유니캐스트 등과 같은 전송방식을 지정하기 위해서 사용된다
Play	서버에서 Setup 메소드 전송시에 지정한 방식을 사용해서 멀티미디어 컨텐츠를 전송하고 전송시 시작과 종료점을 지정한다
Pause	서버에게 멀티미디어 컨텐츠의 전송을 일시적으로 중단할 것을 요청한다
Teardown	서버에게 멀티미디어 컨텐츠의 전송을 중단하고 세션을 종료할 것을 요청한다
Redirect	현재 서버에 많은 부하가 걸려있고 요청한 서버보다 근거리에 있는 다른 서버에게 멀티미디어 컨텐츠를 전송하도록 알려줄 때 사용한다
Record	서버가 클라이언트에게 보내는 멀티미디어 컨텐츠를 특정 URL에 저장하도록 지정한다

표 1에 있는 메소드를 사용할 때 다양한 옵션을 헤더필드에 포함시켜 서버에 요청하고 결과 메시지를 받을 수 있는데 서버는 헤더필드에 삽입된 옵션을 분석하고 이 정보를 바탕으로 적절한 메시지를 보낼 수 있게 된다. 표 2에 있는 다양한 헤더필드를 사용하여 서버와 클라이언트간 전송하고자 하는 미디어의 포트번호와 전송방식 등을 확인하게 된다

표 2 RTSP에서 사용하는 헤더정보

헤더필드	용도
Accept	자신이 어떠한 설명 형식을 지원하는지를 서버에게 알려주기 위해 사용된다.
Allow	지원하는 메소드들의 리스트를 알리기 위한 용도로 사용된다.
CSeq	모든 요청과 응답 메시지에 포함되어 있어야 하는 필드로, sequence number값을 지정한다. 특정 요청에 대한 응답 메시지는 요청 메시지와 동일한 값을 가져야 한다.
Range	비디오의 임의 위치 재생기능의 구현을 위해 사용된다.
Scale	1의 값을 가지면 정상재생속도를 의미한다. 1이 아니면 그 값은 정상적인 속도와의 비율을 의미한다. 서버가 그 값을 맞출 수 없을 경우 근사한 재생속도를 선택한 뒤 응답코드에 자신이 선택한 재생속도를 표시해야 한다.
Transport	전송방식을 지정하는 일을 수행한다. RTP와 RTCP의 전송을 위하여 서버와 클라이언트는 각각의 포트번호를 지정해주어야 한다.

표 1, 2를 사용하여 클라이언트가 서버에게 원하는 작업과 세부적인 정보를 서버에게 전달하게 되면 정상적인 요청일 경우와 서버에서 해결할 수 없는 에러, 메시지의 오류 등 여러 가지 상황에 맞는 메시지를 보내주어야 한다. RTSP의 응답메시지는 HTTP에서 정의하고 있는 응답 메시지와 의미가 거의 똑같다. 표 3은 HTTP의 응답메시지와 동일한 의미를 가진 메시지를 제외한 응답 코드이다.

표 3 RTSP에서 사용하는 응답코드

응답코드	내 용
250	서버가 Record기능을 수행할 때
405	요청한 메소드를 실행하지 못할 때
451	수신자가 지원하지 못하는 파라미터가 있을 때
453	요청된 대역폭의 할당에 실패했을 때
454	RTSP Session ID값이 없을 때
455	현재 세션상태에서 처리할 수 없는 메소드 일 때
457	Range헤더정보에서 지정한 구간이 실제 재생시간을 벗어나는 경우
461	Transport필드에 지원되지 않는 전송방식이 포함되어 있는 경우

표 4는 클라이언트가 서버에 접속하여 RTP 패킷이 전송되는 과정을 표시한 것이다. 1번 과정은 클라이언트가 서버에게 twister라는 멀티미디어 컨텐츠에 대한 정보를 요구하고 있는 것이다. 헤더필드에 들어가는 CSeq는 클라이언트가 서버에게 2개 이상의 자료를 요구할 경우가 있기 때문에 Sequence Number로 이를 구분하기 위해 보낸다. 2번 과정은 1번 과정의 메시지가 정상적인 요청이라는 응답을 보내고 헤더 정보 안의 데이타의 전송방식을 지정하기 위하여 Content-type 헤더필드에 데이타의 전송방식을 SDP 방식으로 보내겠다고 지정하고, 클라이언트가 요구한 미디어의 세부정보를 전송하게 된다. 이곳에서 서버에서 제공되는 미디어 서버의 비디오/오디오 트랙에 대한 정보를 전달하게 된다. 정보의 세부적인 내용으로는 세션 안에 포함되어 있는 비디오/오디오 트랙의 일련번호, 인코딩 방식, 각각의 트랙을 개별적으로 호출할 때 사용하는 주소 등을 포함하고

표 4 실제로 교환되는 RTSP 메시지

```
1) C→S: DESCRIBE rtsp://foo/twister RTSP/1.0
CSeq: 1
2) S→C: RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 164
v=0
o=-2890844256 2890842807 IN IP4
172.16.2.93
s=RTSP Session
i=An Example of RTSP Session
Usage
a=control:rtsp://foo/twister
t=0 0
m=audio 0 RTP/AVP 0
a=control:rtsp://foo/twister/audio
m=video 0 RTP/AVP 26
a=control:rtsp://foo/twister/video
3) C→S: SETUP rtsp://foo/twister/audio RTSP/1.0
CSeq: 2
Transport: RTP/AVP;unicast;client_port=8000-8001
4) S→C: RTSP/1.0 200 OK
CSeq: 2
Transport: RTP/AVP;unicast;client_port=8000-8001;
server_port=9000-9001
Session: 12345678
5) C→S: PLAY rtsp://foo/twister RTSP/1.0
CSeq: 4
Range: npt=0-
Session: 12345678
6) C→S: TEARDOWN rtsp://foo/twister RTSP/1.0
CSeq: 892
Session: 12345678
7) S→C: RTSP/1.0 200 OK
CSeq: 892
```

있어 클라이언트가 미디어의 정보를 분석한 다음 원하는 비디오/오디오 트랙을 선택하고 다른 미디어의 트랙과 Mixing이 가능하도록 하고 있다.

3번과 4번 과정은 서버가 필요한 미디어의 트랙을 선택하면서 전송 받을 포트번호를 서버에게 전달하게 되고 서버는 클라이언트에게 전송시킬 포트번호를 전달하면서 세션번호를 부여하게 된다. 여기서 발생되는 세션번호를 기준으로 클라이언트는 Play와 Pause등 미디어의 제어를 요청할 수 있고 서버간 redirection과 Record등을 요청하고 전달받을 수 있다. 5번 과정은 서버와 클라이언트간 미디어 패킷을 전송할 포트를 교환한 후 실제로 패킷 전송을 개시하는 메시지다. 이때 사용되는 Range 헤더필드는 미디어의 전체가 아닌 미디어의 부분 부분을 전송할 수 있도록 시작위치와 종료위치를 지정할 때 사용된다. 또한 미디어의 지정을 3번 과정에서 얻은 트랙의 이름으로 지정하면 미디어의 전체트랙이 아닌 선택된 오디오와 비디오 트랙을 전송할 수 있게 된다.

이러한 전송 방식 중 Ingo Busse[8]는 패킷 손실율에 의한 동적 흐름제어 기법을 제안하였다. 패킷 손실율은 수신자가 RTP 패킷의 순차 번호를 이용하여 측정하고, 이 측정값을 RTCP의 패킷손실 필드에 기록하여 송신자에게 전송한다.

송신자는 수신자가 측정한 패킷 손실률에 따라 네트워크를 세가지 상태, 혼잡(congested), 정상(loaded), 무부하(unloaded) 상태로 나누어 혼잡상태에서는 지수적으로 전송률을 감소하고 무부하 상태에서는 선형적으로 전송률을 증가시켜 전송한다.

하지만 네트워크 상태에 따라 전송률을 동적으로 변화시켜 네트워크 상태가 정상상태로 접근해야 하는데, 선형적인 증가율의 크기에 네트워크 상태가 혼잡과 무부하 상태를 반복하는 진동 현상이 관찰되었다. 이런 진동현상을 감소시키기 위하여 모수정[5]은 표 5와 같은 알고리즘을 제시하였다.

표 5 진동현상을 줄이기 위한 알고리즘

```
#define delta 10,000 // 10kbps
if (decision == UNLOADED) { // 무부하 상태
if (flow_bw < bthresh)
① ------- flow_bw += delta; // 지수적 증가
else if (flow_bw > bthresh)
② -------- flow_bw += (1 / flow_bw) // 선형적 증가
* delta * 10000;
if (maxbw_ < flow_bw) // 최고 대역폭 찾기
maxbw_ = flow_bw;
congest_flag = 1;
}
else if (decision == CONGESTED) { // 혼잡상태
if (congest_flag) // 대역폭 상승에서 감소로 전환
③ ------- bthresh = (1 - BETA) * bthresh
+ BETA * maxbps / 2;
flow_bw = max(flow_bw // 전송량 결정
* DEC_AMOUNT, MIN_RATE);
congest_flag = 0;
}
```

위 알고리즘을 사용하면 TCP통신을 하는 여러 프로그램과 경쟁력 있는 대역폭 경쟁을 하게 되지만 동일한 멀티미디어 스트림을 제공하는 멀티미디어 서버에서는 동일한 서버에서 전송하는 다중 스트림간 불필요한 대역폭 경쟁을 하는 단점이 있다.

그래서 본 논문에서는 이러한 단점을 개선하기 위하여 한 서버에서 여러 스트림을 제공할 때 스트림 전송간의 평균값을 이용한 제어방법을 추가하였다.

3. JMF와 RTSP 서버

Sun 사에서 개발한 Java는 플랫폼에 관계없이 프로그램을 실행시킬 수 있는 프로그래밍 언어이다. 초기의 Java는 클라이언트측에 동적인 컨텐츠를 제공할 목적으로 주로 사용되었으나, 최근에는 Servlet 등과 같이 서버측에서 다양한 어플리케이션을 개발하기 위한 웹 서버 프로그래밍 언어로 사용되는 등 점차 그 활용 범위가 확대되고 있다[9].

이 중 JMF(Java Media Framework)[10] 는 Java 개발 환경에서 다양한 형식의 멀티미디어 데이타를 처리할 수 있도록 Sun 사와 IBM이 공동으로 개발한 API 이다. JMF 1.0은 단순히 저장된 미디어 파일을 디스플레이 하는 기능만을 제공했으나, JMF 2.0으로 업그레이드 되면서 미디어 트랙의 분리, RTSP Client의 지원, RTP 데이타 전송 기능 등이 추가되었다. 최근 발표된 JMF 2.1.1은 저장된 미디어를 RTP 데이타 형식으로 클라이언트에 전송할 수 있을 뿐만 아니라 음성 데이타와 영상 데이타를 캡쳐하여 이를 전송할 수 있도록 개선되었다.

종전에는 멀티미디어 데이타를 읽어들이기 위해 하드웨어 자원을 사용했다. 그러나, 이러한 방법은 특정 하드웨어에 종속될 뿐만 아니라, 하드웨어 종류에 따라 각기 다른 어플리케이션을 작성해야 된다는 단점이 있었다. JMF에서는 하드웨어에 종속된 부분은 모두 추상화시키는 대신, 실제로 사용자에게 필요한 클래스와 메소드만을 제공함으로써 보다 쉬운 프로그래밍이 가능하다. 그러나, JMF는 RTSP Client 기능만 클래스로 제공하고 있고 RTSP Server 기능을 제공하고 있지 않기 때문에 RTSP Parsing 엔진은 따로 만들어야만 한다(그림 1).

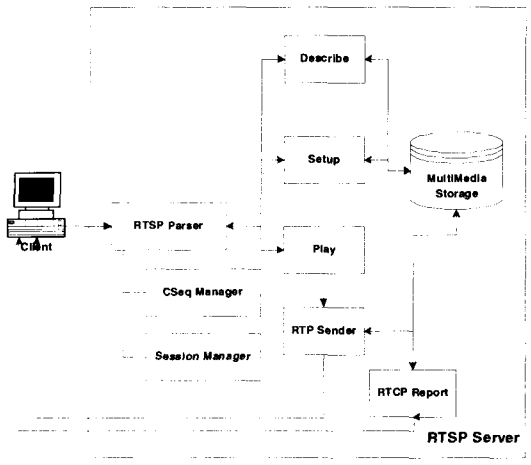

그림 1 RTSP 서버

JMF에서 사용하는 클래스는 그 기능에 따라 크게 2가지로 나눌 수 있다. 즉, 캡쳐 디바이스를 검색하고 사용자가 원하는 비디오/오디오 형식으로 지정하는 CaptureDeviceManager 클래스와 미디어를 제어하기 위해 미디어 파일의 내용을 추상화, 은닉화 시켜놓은 DataSource 클래스가 있다.

DataSource 클래스의 기능은 미디어 종류(캡쳐 디바이스, 미디어 파일)에 상관없이 동일한 클래스로 제어할 수 있기 때문에 사용자 인터페이스를 동일하게 만들 수 있는 장점이 있다. DataSource 클래스로 미디어 파일을 설정한 다음 Processor 클래스를 사용하여 미디어 트랙의 분리 및 합성, 인코딩 방식을 변경할 수 있다. VideoFormat 및 AudioFormat 클래스는 트랙 정보를 Processor 클래스에 저장하고 있으며, 트랙의 형식과 인코딩 방식을 제어할 수 있다. 트랙을 RTP 방식으로 변경하면 RTSP 서버에서 RTP 미디어의 전송을 제어할 수 있다. 또한 Processor 클래스는 미디어의 처리 기준을 Configuring, Configured, Prefetching, Prefetched, Playing 등 각 상태 별로 이벤트를 구성하고 이를 제어할 수 있는 Control 클래스를 제공한다. 이는 RTSP 메시지 처리 방식과 상당히 유사하다.

SessionManager 클래스는 Processor 클래스로부터 변경된 트랙의 정보를 받아 RTP 형식으로 클라이언트에게 전송할 때 사용된다. 또한, SessionManager 클래스는 RTCP의 기능 중 SR과 RR등을 전송 받아 전송 품질을 판단할 수 있는 기능을 제공한다.

JMF는 멀티미디어 컨텐츠를 제어할 수 있는 기능을 제공하지만 RTSP서버에서 지원하는 기능은 제공하지 않기 때문에 RTSP 메시지를 클라이언트에게 전송하기 위한 Parsing 클래스를 따로 만들어야 한다. Parsing은 표 4에서 발생했던 메시지를 바탕으로 미디어 정보를 클라이언트에게 전달할 수 있도록 만들면 된다. RTSP는 상태 정보를 가진 프로토콜이기 때문에 현재 접속되어 있는 클라이언트에게 요청할 정보를 저장해 놓고 클라이언트의 요구에 따라 적당한 응답 메시지를 전달해야 한다.

그림 1의 RTSP 서버 구조는 RTSP Parser가 미디어 서버의 내부를 총괄하고 있음을 보여주고 있다. 클라이언트에서 전달된 메시지는 CSeq Manager와 Session Manager를 통해 동일 클라이언트에서 전달되는 메시지인지 여부를 확인하고 요청한 메소드를 분석한 뒤 해당 클래스에게 전달해준다. Describe 및 Setup 클래스는 JMF의 DataSource, Processor클래스를 상속 받은 클래스로써 미디어를 분석하여 RTSP의 헤더 필드에 필요한 정보를 획득하고 미디어의 특정 트랙을 분리한다. Play 메소드가 호출되면 클라이언트가 요청하는 미디어의 특정 시간만 선택하여 RTP Sender를 호출하고 SessionManager 클래스를 상속 받은 RTP Sender는 클라이언트에게 RTP 패킷을 전달하게 된다.

일정 시간마다 서버와 클라이언트는 RTCP 패킷의 내용을 분석할 수 있도록 RTCP Report클래스는 SessionManager 클래스를 상속 받은 SR과 RR 을 호출하여 네트워크 지연 상황과 패킷의 전송상태를 확인하게 된다.

4. 세션별 전송량 제어 방식

기존 미디어 서버의 경우 클라이언트로부터 요구가 있을 때마다 멀티미디어 컨텐츠를 전송하는 방식을 사용하였다. 그러나, 이러한 방식에서는 클라이언트로부터의 요청이 많아질 경우 각각의 클라이언트에 대해 균등한 네트워크 대역폭을 할당해주지 못하고, 클라이언트가 요구하는 대로 대역폭을 할당해야만 했다. 따라서 클라이언트 수가 지나치게 많아 네트워크 대역폭을 넘어서는 요청이 들어오면 전송 불가 메시지를 보내거나 다른 미디어 서버로 우회 시키는 방식을 사용하였다.

먼저 본 논문에서 제안하는 방식이 적용될 네트워크 환경은 미디어 서버를 중심으로 인트라넷 상에 클라이언트가 연결되어 있으며, 각 클라이언트와 서버 간 전송 속도는 동일하다고 가정한다. 이러한 환경 하에서 클라이언트의 요청이 증가할 때마다 각 세션별 전송량 평균이 전체 전송량 평균에 근접하도록 네트워크 트래픽을 동적으로 조절함으로써 더 많은 수의 클라이언트 접속

이 이루어질 수 있도록 하기 위한 것이다. 즉, 전체 전송량의 평균과 세션별 전송량 평균 간 비율을 구해, 그 비율이 일정 범위 이상을 벗어날 경우 Motion JPEG의 양자화 계수를 조정한다(식 (1)). JMF의 인코딩 방식 중 하나인 Motion JPEG 방식의 JPEG 양자화 계수를 0~1 범위 내에서 변화시킴에 따라 전송량이 조절된다. 양자화 계수가 1에 가까울수록 Motion JPEG의 화질이 선명한 대신 전송량은 상대적으로 많아지기 때문이다.

$$T_{rate} = \frac{\sum_{n=\alpha}^{\beta} \frac{T_n - T_{n-1}}{n-1}}{T_{avg}} \quad (1)$$

먼저 T_n이 구간 n에서의 전송량(*bit rate*)이라고 할 때, $(T_n - T_{n-1})/(n-1)$은 소구간(n-1,n)에서의 전송량의 평균이다. 따라서, 전체 구간(α, β)에서 세션별 전송량 평균은 $\sum_{n=-}^{-} \frac{T_n - T_{n-1}}{n-1}$가 된다. 여기서 α와 β는 세션별 전송량 평균을 계산하는 구간의 시작과 끝이다. 식(1)은 세션별 평균 전송량과 전체 전송량 평균(T_{avg})과의 차를 계산하는 수식이다.

이 방식을 적용할 경우 멀티미디어 컨텐츠를 요구하는 클라이언트의 수가 늘어나게 되면 전체 전송량의 평균은 줄어들게 되고 세션별 평균과의 차는 커지게 되어(즉, T_{rate}<<1) Motion JPEG의 양자화 계수가 작아져서 더 많은 수의 클라이언트의 전송에 응답할 수 있게 된다.

T_{rate}값의 범위에 따라 양자화 계수를 조절하기 위해서는 그 값의 범위를 구체적으로 지정하는 것이 필요하다. 이를 위해 RTCP 패킷에 있는 정보를 사용하였다. 즉, RTSP 서버에 접속 중인 클라이언트를 서버에서 모니터링하기 위해 RTSP 서버는 RTCP의 RR을 이용하였다. 대개 JMF에서는 사용자 임의대로 RTCP 패킷을 보낼 수 없으며, 네트워크 부하량이 적을 경우에만 자동으로 서버에 RR을 전송한다. JMF에서는 평균적으로 4초에 한 번씩 RTCP를 보낸다. RTCP에는 수신자의 상태를 모니터링 할 수 있는 여러 가지 내용들이 포함되어 있다. 이 중에서 RTSP 서버에서 사용하는 내용들은 Packet Lost(Cumulative number of packets lost)와 DLSR(Delay since last SR)의 항목들이다. 이들을 이용해서 네트워크의 전송속도를 제어하게 된다.

그림 2는 RTSP 서버 전체 패킷 전송량의 평균과 세션별 패킷 전송량의 비율(T_{rate})이 각각 1%, 5%, 10%일 때를 기준으로 세션별 전송량을 제어했을 경우 JPEG 양자화 계수의 변화를 나타낸 것이다. 앞서 설명한 바와 같이, 양자화 계수의 변화에 따라 클라이언트에 전송되는 전송량이 제어된다. 비율을 1% 범위 이내(0.99~1.01)로 설정하였을 경우 양자화 계수의 변화가 너무 커서 전체적인 평균값에 수렴하지 못할 뿐만 아니라 네트워크 사용량이 비효율적이었다. 또한 10%(0.9~1.1)로 설정했을 경우에는 전체 평균에 접근하는 시간이 너무 길며 평균값의 변화에 너무 늦게 반응하는 단점이 있었다.

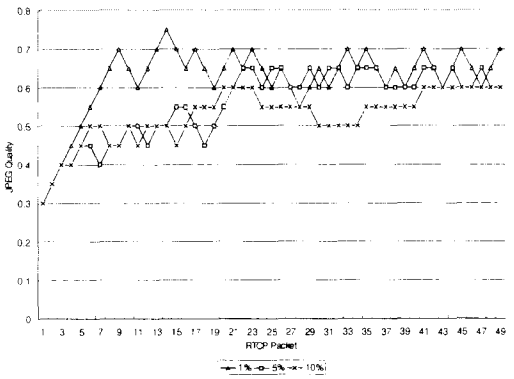

그림 2 JPEG 품질변화

따라서, 전체 전송량의 평균을 기준으로 각 세션별 평균 전송량과의 비율이 5%(0.95~1.05)이상 날 경우 전송량을 제어하도록 설계하였다. 따라서, 식(1)의 T_{rate}가 0.95이하가 되면 양자화 계수를 1에 가깝게 하고, T_{rate}가 1.05이상이 되면 양자화 계수를 0에 가깝게 변화시켰다.

5. 실험 결과

T_{rate}값 변동(식(1))에 따라 양자화 계수가 동적으로 조절되는 예를 표 6에 나타내었다. T_1에서 T_4에 이르기까지 T_{rate}은 모두 0.95보다 작기 때문에, 양자화 계수를 단계별로 0.5씩 계속 증가시켰음을 알 수 있다. 그

표 6 T_{rate}에 따른 양자화 계수 변화

T_n	$\sum_{n=-}^{-} \frac{T_n - T_{n-1}}{n-1}$	T_{avg}	T_{rate}	JPEG 양자화계수
1	419.4	510.3	0.88	0.3
2	479.8	524.3	0.91	0.35
3	426	525.1	0.81	0.40
4	348.6	510.1	0.68	0.45
5	543.6	385.3	1.41	0.40

러나, T_5에서 세션별 전송량 평균(543.6)이 전체 전송량 평균(385.3)보다 훨씬 커졌으므로(T_{rate}=1.41), 이 시점에서 양자화 계수를 반대로 0.5 감소시켰음을 알 수 있다. 즉, 세션별 전송량 평균을 줄이기 위해 JPEG 의 화질을 다소 줄인 것이다[11][12].

그림 3은 RTSP 서버가 다섯 개의 스트림을 전송할 때, 네트워크 전송량을 RTCP로 제어했을 경우와 제어하지 않았을 경우의 패킷 손실률(Packet Lost)을 나타낸 것이다. 네트워크 전송량을 제어하지 않았을 경우 각 클라이언트의 세션은 네트워크에 패킷을 내보내기 위해 서로간 경쟁을 하게 된다. 경쟁에서 뒤진 클라이언트는 패킷 전송에 실패하게 되고 JMF를 사용한 클라이언트 화면의 디스플레이 속도가 느려지거나 화면이 손상된다. 이러한 현상은 세션의 개수가 2~3개정도일 때는 발견되지 않았으나 그 이상일 경우에는 뚜렷이 나타남을 확인하였다. 즉, 네트워크 대역폭 이상의 전송량이 발생할 때 이러한 현상이 나타나게 되는 것이다. 본 논문에서 제안하는 전송방식을 사용하면 기존의 미디어서버의 전송방식을 사용할 경우보다 더 많은 수의 클라이언트 접속이 가능하며 각 세션이 네트워크 대역폭을 균등하게 사용하여 패킷 손실이 줄어듦을 알 수 있다

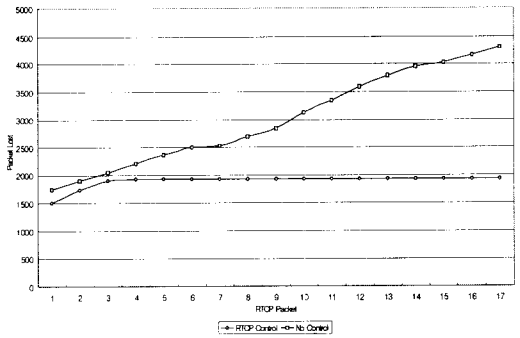

그림 3 전송량 비교

6. 결 론

본 논문에서는 네트워크 트래픽에 따라 비디오 스트리밍 데이타의 전송량을 가변적으로 제어하기 위한 방안을 제시하였다. 즉, 네트워크 전송량을 모니터링 하기 위해 RTCP의 RR에 포함되어 있는 분실된 패킷 수(Packet Lost) 등을 사용하였다. 또한 JMF에서 사용하는 RTP 인코딩 방식 중 Motion JPEG의 JPEG양자화 계수를 조절하여 JPEG품질을 변화시킴으로써 네트워크 전송량을 제어하였다. 실험 결과 네트워크 전체 전송량 평균과 세션별 전송량 평균 비율이 5% 를 넘어설 경우 각 세션별 전송량을 세션 평균값에 근접하게 조절함으로써 전체 네트워크 전송량이 줄어들며, 이에 따라 전송 효율이 개선됨을 확인할 수 있었다.

참 고 문 헌

[1] H.Schulzrinne, RTP profile for audio and video conferences with minimal control, RFC1890, 1998.
[2] H.Schulzrinne, S.Casner, R. Frederick,V.Jacobson, RTP: a transport protocol for real-time applications, RFC1889, 1998.
[3] H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol, RFC2326, 1998.
[4] 나승구, 윤성덕, 안종석, "TCP와 유사한 RTP/RTCP 흐름제어 알고리즘", 정보과학회 가을 학술발표논문집, 제25권, 제2호, p. 480-482, 1998.
[5] 모수정, 안종석, "RTP/RTCP를 위한 확장성 있는 피드백 제어기법", 정보과학회 가을 학술발표논문집, 제25권 제2호, p.477-479, 1998.
[6] S. McCanne, L. Berc, W. Fenner, R. Frederick, RTP Payload Format for JPEG-compressed Video, RFC 2035, October 1996.
[7] Mark A. Miller, "Voice Over IP," M&T, 2000.
[8] Ingo Busse, Bernd Deffner, Henning Schulzrinne, Dynamic QoS Control of Multimedia Applications based on RTP, http://www.fokus.gmd.de/step/acontrol/ac.html
[9] Steve Wilson, Jeff Kesselman, "Java Platform Performance," Sun Microsystems.
[10] Java Mediaframe Work, JavaOne Session, 2001.
[11] Craig Hunt, TCP/IP Network Administration, O'Reilly, 1998.
[12] Mike Loukides, "System Performance Tuning," O'Reilly, 1992.

박 대 훈
1997년 2월 인천대학교 전자공학과 졸업(공학사). 1997년 3월 ~ 2001년 2월 인천대학교 대학원 컴퓨터공학과(공학석사). 2001년 12월 ~ 현재 한국 컴퓨터 UNIX 지원팀. 관심분야는 무선 인터넷 프로그래밍, 이동 컴퓨팅, 사용자 인터페이스

허 혜 선

1994년 2월 인천대학교 전자계산학과 졸업(공학사). 1996년 3월 ~ 1998년 2월 인천대학교 대학원 전자계산학과(공학석사). 2000년 3월 ~ 현재 인천대학교 대학원 정보통신공학과 박사과정. 관심분야는 인터넷 텔레포니, 모바일 프로그래밍, Virtual Prototyping

홍 윤 식

1983년 2월 한양대학교 전자공학과 공학사. 1985년 2월 한국과학기술원 전기및 전자공학과 공학석사. 1989년 2월 한국과학기술원 전기 및 전자공학과 공학박사. 1989년 3월 ~ 1991년 8월 LG전자(주) 우면연구소 선임연구원. 1994년 7월 ~ 1995년 8월 Univ. of California, Irvine, 방문 연구원. 1991년 8월 ~ 현재 인천대학교 컴퓨터공학과 교수. 관심분야는 무선 인터넷 프로그래밍, 이동 컴퓨팅, 사용자 인터페이스