응용 독립적인 RTP 통신 모듈의 설계 및 구현

박 상 현† · 박 상 윤† · 김 명 준†† · 엄 영 익†††

요 약

본 논문에서는 멀티미디어 데이타와 같은 실시간 특성을 지닌 데이타의 종단간 전달 기능과 서비스의 품질(QoS) 감시 기능 등을 지원하기 위해 개발된 실시간 전송 프로토콜인 RTP를 분석하고 설계, 구현하였다. 기존의 RTP 구현 모듈들은 응용에 의존적인 RTP 자체의 특성에 의해 특정 응용에 종속적인 형태로 설계, 구현되어 재사용이 힘들었으며, 동작 과정 중에 발생할 수 있는 비효율적인 멀티미디어 데이타 처리 등의 문제점들을 지니고 있었다. 본 논문에서 제시한 RTP 통신 모듈은 이와 같은 문제점들을 극복하기 위해 특정 응용의 구조 및 동작 과정에 관계없이 모든 종류의 상위 응용들에서 사용할 수 있도록 응용 독립적인 형태로 설계, 구현되었다.

Design and Implementation of the Application-independent RTP Communication Module

Sang-Hyun Park† · Sang Yun Park† · Myung Joon Kim†† · Young Ik Eom†††

ABSTRACT

In this paper, we present the design and implementation results of RTP(Real-time Transport Protocol) which is introduced to provide end-to-end delivery service for multimedia data with real-time characteristics and to provide QoS(quality of service) monitoring services. Conventional RTP communication modules have some problems such as poor reusability and inefficiency. Reusability problem stems from the inherent characteristics of RTP that its framework is deliberately not complete, and so, RTP should have been designed and implemented in application-imbedded form. The RTP communication module, proposed in this paper, is designed and implemented in application-independent form in order to be used in all kinds of high level applications independent of their architectures and functions.

1. 서 론

최근 컴퓨팅 기술의 발전 및 응용 분야의 다양화는 기존의 텍스트 데이타와 같은 이산 미디어 데이터(discrete media data) 중심의 통신 방식뿐만 아니라 멀티미디어 데이타와 같은 연속성, 실시간성 및 오류 허용성 등의 특성을 가진 연속 미디어 데이타(continuous media data)의 네트워크 송·수신에 대한 적절한 처리를 요구하고 있다. 이러한 연속 미디어 데이타를 송·수신하는 실시간 응용의 예로는 인터넷 전화, 멀티미디어 회의, 교안 전송과 미팅, 분산 시뮬레이션, 네트워크 게임 등이 있으며 앞으로 원격 진료, 분산 작업그룹, 원격 교육 및 원격 컴퓨팅 등에 본격적으로 사용될 것으로 보인다.

그러나 많은 데이타를 짧은 시간에 처리해야 하는

※ 이 논문은 한국전자통신연구원에서 수행중인 "인터넷 분산 시스템 소프트웨어 기술 개발 사업"의 결과물 중 일부임.
† 준 회 원 : 성균관대학교 대학원 전기전자 및 컴퓨터공학부
†† 종신회원 : 한국전자통신연구원 인터넷서비스연구부 부장
††† 종신회원 : 성균관대학교 전기전자 및 컴퓨터공학부 교수
논문접수 : 1999년 1월 29일, 심사완료 : 1999년 8월 5일

실시간 응용의 특성상 기존의 네트워크 환경을 그대로 사용하기에는 어려운 점이 많이 있다. 이는 기존의 네트워크 환경이나 프로토콜들이 이산 미디어 데이타인 텍스트 데이타 처리를 중심으로 설계되어졌으므로 멀티미디어 데이타의 실시간성과 같은 데이타 각각의 특성을 충분히 지원해주지 못하기 때문이다[1].

그럼에도 불구하고 기존의 네트워크 환경을 기반으로 실시간 멀티미디어 응용을 구현할 수 있다면 몇 가지 상당한 장점을 얻을 수 있다. 우선 기존의 네트워크 환경은 인프라구조가 이미 성립되어 있어 새로운 네트워크 환경 구축을 위한 비용에 대해 경제적 부담이 덜할 뿐만 아니라 다양한 통신 방식의 수용이 가능하며 그리고 해당 네트워크가 허용하는 범위 안에서 어느 정도 실시간 응용의 적용이 가능하다는 것이다.

이러한 배경을 바탕으로 기존 네트워크 환경을 통한 연속 미디어 데이타 전송 프로토콜에 대한 연구가 수행되어 왔다. 이러한 연구들 중 하나가 UDP와 같은 하부 전송 프로토콜과 상위 응용 사이에 RTP(Real-time Transport Protocol)와 같은 새로운 프로토콜을 삽입하는 것이다. RTP는 네트워크를 통한 미디어 데이타의 전송, 탑재 미디어의 다양한 유형 지원, 전송되는 데이타에 대한 동기화 정보 제공, 네트워크 흐름 상태 및 혼잡에 대한 제어 정보 등을 지원한다. RTP와 그에 관련된 표준안들은 현재 IETF(Internet Engineering Task Force)의 Audio-Video Transport Working Group에서 RFC 1889, 1890 등의 표준안들로 제시되고 있다[2, 3].

본 논문에서는 멀티미디어 데이타의 실시간 전송 기능을 제공하며 전송간 발생하는 네트워크 상황에 대한 QoS(Quality of Service) 정보를 생성, 관리하는 RTP 기반 데이타 통신 모듈을 RFC 1889, 1890 스펙에 맞춰 설계하고 구현하였다. 본 RTP 기반 데이타 통신 모듈은 응용과 네트워크 사이에 존재하면서 양단간 인터페이스를 제공하고 멀티캐스팅을 지원하며, 실시간 응용이 QoS를 만족시킬 수 있도록 다양한 정보를 제공할 수 있게 하였다. 그리고 기존의 RTP 구현 모듈들이 특정 응용에 종속되게 설계되어 다른 응용의 구현에 재사용하기가 힘들었던 점이나 실시간 데이타의 전달시 응용의 동작과 RTP 구현 모듈의 동작이 분리되지 않아 발생할 수 있는 비효율적인 처리 과정과 같은 문제점들을 극복하기 위해 재사용이 가능하도록 설계하였고, 구동시에도 응용에 독립적으로 동작할 수 있도록 구현하였다.

본 논문의 2장에서는 RTP에 대한 개요 및 RTP를 기반한 실시간 멀티미디어 데이타 전송에 관하여 기술하고, 3장에서는 RTP 기반 멀티미디어 데이타 통신 모듈의 설계 사항과 RTP 동작 시나리오를 소개하며, 4장에서는 통신 모듈의 구현 결과를 설명하고, 5장에서는 본 논문에서 구현한 통신 모듈에 대한 평가를, 6장에서는 멀티미디어 데이타 통신 모듈의 구현으로 인한 기대 효과 및 향후 연구 과제를 제시한다.

2. 관련 연구

2.1 RTP/RTCP 개요

RTP(Real-time Transport Protocol)는 네트워크 종단간에 오디오, 비디오, 시뮬레이션 데이타와 같이 실시간 특성을 갖는 데이타를 전달하는 역할을 수행한다. RTP는 기본적으로 다자간 멀티미디어 회의의 필요에 의해 설계되기는 하였지만 특정 응용에 제한을 두기보다는 연속 데이타의 저장, 대화식 분산 멀티미디어 응용, 그리고 제어 및 측정 응용 등 다양한 응용에 적용될 수 있다. RTP의 특징은 선택 메커니즘을 첨가함으로써 부가적으로 기능을 확장할 수 있게 하는 전통적인 프로토콜과는 달리 필요한 만큼 RTP 패킷의 헤더를 수정하거나 헤더에 첨가함으로써 특정 응용에 유연하게 적응할 수 있게 한다. 그리고 전형적으로 RTP는 멀티플렉싱, 체크섬, 그리고 멀티캐스팅과 같은 서비스를 제공하기 위해 UDP 위에서 동작한다. RTP는 탑재 유형 식별, 패킷의 순서 번호화, 타임스탬핑 등의 기능을 제공한다.

그러나 RTP 그 자체로는 시간에 따르는 순서화 배달, 패킷의 실시간 서비스, 기타 QoS 보장을 위한 어떠한 매커니즘도 제공하지 않는다. 따라서 세션의 QoS를 모니터 하거나 세션에 참가 중인 참가자들에 대한 식별 정보와 같은 여러 가지 정보들을 전달하는, 그리고 세션의 최소 제어 등을 제공하기 위한 실시간 트랜스포트 제어 프로토콜인 RTCP(RTP Control Protocol)가 부가적 확장 프로토콜로 사용된다. RTCP는 분실된 패킷 수, 지터(jitter) 간격, 직전 패킷과의 지연 시간 등의 QoS 정보를 교환하도록 하여 응용이 적절한 QoS를 설정할 수 있게 해 준다. RTP 패킷과 RTCP 패킷은 하위 망의 지원 여부에 따라 유니캐스트 또는 멀티캐스트 환경에서 모두 사용될 수 있다. RTP는 RTP

패킷의 탑재 유형을 변환시켜 전달하는 중계방식인 변환기 사용과 다중 RTP 패킷들을 탑재 유형들을 유지하면서 하나의 패킷으로 조합하는 혼합기 사용을 지원한다[2].

RTP의 발전 및 표준화 과정은 인터넷 기반 오디오 데이타 전송 분야에서 비롯되었다고 볼 수 있는데 1991년 DARTnet(ARPA 네트워크)에서 내부적으로 시행된 오디오 대상 연구는 RTP 기반의 MBone 전송 기술의 토대를 수립하였고 오디오 코덱을 포함하는 Sun Microsystems의 SPARCstation의 LBL 오디오 회의 도구인 vat 초기 버전에서 차기에 RTP 버전-0으로 인정받는 프로토콜이 사용되었다. 1992년 12월에는 여러 차례의 인터넷 draft 등록과정을 거쳐 RTP 버전-1이 Berlin에서 Henning Schulzrinne(GMD)에 의해 발표되었다. 마침내 1995년 11월 22일, RTP는 IESG에 의해 인터넷 표준안으로 인정되었고 몇 번의 추가적인 갱신 후에 RTP 버전-2가 탄생되었다.

네트스케이프는 산업계 협의안인 RTSP(Real Time Streaming Protocol)[4]의 기초 프로토콜로써 RTP를 사용하였는데, 1996년 1월 31일, Netscape사는 RFC 1889의 RTP와 기타 표준들을 기반으로 하는 "Netscape LiveMedia"를 발표하였다.

인텔, 마이크로소프트와 100개 이상의 기술 기관 공동체는 간단한 전화 연결로써 인터넷을 통한 음성, 영상 및 데이타의 전송을 가능케 하기 위한 관련 표준안들을 기초로 개방형 플랫폼을 구성하기로 협약하였다. 개방형 인터넷 통신 플랫폼을 제시하는 IMTC(International Multimedia Teleconferencing Consortium)이라고 하는 이 공동체는 데이타 통신을 위한 T.120과 오디오 및 비디오 회의를 위한 H.323을 포함하는 ITU(International Telecommunication Union)과 IETF(International Engineering Task Force)의 규약들과 RTP/RTCP와 RSVP의 규약들을 구현을 위한 기초로 한다고 언급하고 있다. 마이크로소프트는 NetMeeting 회의 소프트웨어가 RTP를 지원한다고 언급하였고 ITU의 연구 그룹은 H.320에 호환 가능한 LAN 기반 회의 응용을 위하여 RTP를 사용하는데 동의하고 있다.

2.2 RTP 데이타 전송 프로토콜

RTP는 실시간 성질을 갖는 데이타를 전달하기 위한 실시간 전송 프로토콜이다. RTP 패킷은 12 바이트의 고정 헤더를 가지며, 특정 프로파일에 의해 고정 헤더 뒤에 헤더 확장이 올 수 있다. 그리고 혼합기의 사용에 따라 최대 15개까지의 CSRC(Contributing source identifier) 리스트의 크기가 추가될 수 있으며 헤더 필드 다음에는 멀티미디어 데이타와 같은 전송할 데이타가 온다.

RTP 고정 헤더의 각 필드에는 RTP의 버전이나 패딩 옥텟의 존재 여부, 특정 헤더 확장, CSRC의 수와 전송 데이타의 탑재 유형, 순서 번호와 RTP 패킷에 있는 첫 번째 옥텟의 샘플링 인스턴트를 반영하는 타임스탬프, 동기화 소스를 식별하기 위한 SSRC(Synchronization Source Identifier), RTP 패킷에 포함된 탑재물에 대한 기여 소스를 식별하기 위한 CSRC 필드 등이 있다. 이러한 RTP 패킷의 형태와 각 필드들에 대한 자세한 내용은 IETF RFC 1889에 자세히 기술되어 있다[2].

RTP의 완전한 사양은 하나 이상의 동반 문서를 필요로 하는데 이러한 동반 문서의 예로는 1996년 1월에 제시된 RFC 1890이 있다. 이러한 동반 문서에는 패킷의 탑재 유형 코드를 탑재 형식으로 대응시키는 기본 대응이 제시되어 있으며, 부가적인 기능 확장을 위해 RTP 헤더의 수정 및 확장을 정의하고 있다[3, 5, 6, 7, 8].

2.3 RTCP

실시간 전송 제어 프로토콜인 RTCP는 RTP 패킷과 동일한 배포 메커니즘을 사용하여 주기적으로 제어 패킷을 전송한다. RTCP는 데이타 전송 상태의 질에 대한 피드백을 제공하며, RTCP 패킷에는 트랜스포트 수준의 식별자나 최소 세션 제어 정보 등이 포함된다.

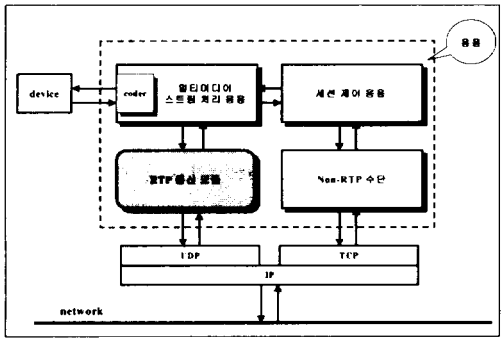
RTCP 패킷은 전달하는 제어 정보에 따라 5가지 종류로 나누어지는데 각 RTCP 패킷의 기능은 다음과 같다. SR(Sender Report RTCP packet)은 RTP 패킷 송신 측이 전송 및 수신 통계에 대한 정보를 전달하기 위한 패킷이며, RR(Receiver Report RTCP packet)은 RTP 패킷 수신 측이 수신 통계에 대한 정보를 전달하기 위한 패킷이다. SDES(Source DEScription RTCP packet)는 CNAME(canonical end-point identifier)이나 EMAIL과 같은 RTCP 패킷 송신 측의 부가적인 정보를 전달하기 위한 패킷이고, BYE(Goodbye RTCP packet)는 세션 참가자가 세션을 떠남을 알리기 위한 패킷이며, APP(Application-defined RTCP packet)는 특정 응용의 실험적 사용을 위해 사용된다. 각 패킷에 대한 구조와 패킷 필드들에 대한 설명은 IETF RFC

1889에 자세히 기술되어 있다[2].

모든 RTCP 패킷들은 여러 RTCP 패킷 유형들이 하나의 패킷으로 구성되는 복합 패킷의 형식으로 전달되어야 한다. 그리고 복합 패킷에는 항상 SR 또는 RR이 포함되어야 하며 SDES의 CNAME(canonical end-point identifier)도 항상 포함되어야 한다.

2.4 RTP 통신 모듈을 이용한 응용의 예

RTP는 응용 수준의 프레임화와 통합계층 처리의 원리에 따르는 새로운 형태의 프로토콜로서 별도의 층으로 구현되기보다는 주로 응용 처리에 통합되어 사용될 것이다. RTP를 사용하는 응용들은 무척 다양할 수 있겠지만 그 중 대표적인 예인 멀티미디어 데이타를 사용하는 응용의 일반적인 구성을 (그림 1)에서 보인다.



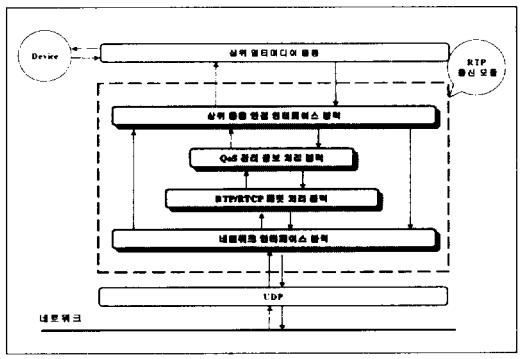
(그림 1) RTP를 이용한 응용의 구성 예

RTP를 사용하는 멀티미디어 응용 안에는 세션 제어 등을 포함한 응용 자체의 조작을 위한 부분이 있으며 코덱[9]을 포함한 멀티미디어 데이타의 생성과 소비를 위한 부분이 있게 된다. 그리고 멀티미디어 자료가 아닌 암호화 키와 같은 정보를 배포하기 위해 E-mail과 같은 non-RTP 수단이 사용되며 멀티미디어 데이타의 송·수신을 위한 RTP 통신 모듈이 있게 된다. 이와 같이 RTP를 이용한 응용에서는 멀티미디어 데이타와 세션 제어 정보와 같은 실시간성을 갖지 않는 데이타는 서로 다른 경로를 통해 세션의 다른 참가자에게 전달된다[10].

3. RTP 통신 모듈의 설계

3.1 RTP 통신 모듈의 전체 구조 설계

(그림 2)는 RTP 통신 모듈과 상위 응용, 그리고 하위 전송 프로토콜로 구성되어 있는 세션 참가자 응용을 예시하고 있다. 본 논문에서는 RTP 통신 모듈을 이용하는 상위 응용으로 멀티미디어 응용을 고려하였으며 하위 전송 프로토콜로는 비신뢰적이나 많은 데이타를 빠르게 보낼 수 있고 또한 멀티플렉싱 및 체크섬 서비스도 사용할 수 있는 UDP를 사용하였다. 그리고 RTP 통신 모듈은 실시간 성질을 가지는 데이타와 QoS 정보를 송·수신할 수 있도록 설계하였다.



(그림 2) RTP 전체 블록 구성도

본 논문에서 설계한 RTP 통신 모듈의 특징으로는 여러 응용에서 다시 사용할 수 있도록 응용에 독립적으로 설계하였다는 점이다. 별도의 계층이 아닌 특정 응용 처리에 통합되어 구현되는 RTP 통신 모듈의 근본적인 특징상 특정 응용에 결합되어 구현된 RTP 통신 모듈을 다른 응용의 구현에서 사용하기에는 어려운 점들이 있다. 이를 해결하기 위해 상위 응용은 RTP 통신 모듈에서 제시하는 인터페이스를 통해서 송·수신 부분을 구현할 수 있도록 하였고 응용의 구동 시에도 RTP 통신 모듈은 상위 응용의 동작에 독립적으로 수행할 수 있도록 쓰레드로 동작하게 하였다.

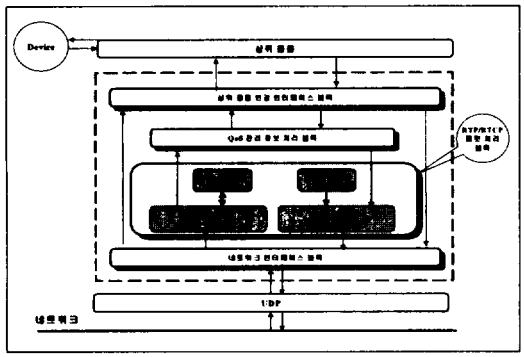
본 논문의 RTP 통신 모듈은 기능상 모두 4개의 블록으로 구성되어 있다. 상위 응용 연결 인터페이스 블록은 상위 응용과 RTP 통신 모듈간에 멀티미디어 데이타와 QoS 정보를 주고받을 수 있도록 중개 역할을 담당한다. 이 블록에 의해 RTP 통신 모듈은 상위 응용에 독립적으로 구현될 수 있다. QoS 관리 정보 처리 블록은 패킷 송·수신 중에 발생한 QoS 정보와 세션 참가자들의 정보를 생성, 유지하는 역할을 수행한다. RTP/RTCP 패킷 처리 블록은 RTP/RTCP 패킷을 생성하여 네트워크 인터페이스 블록에 전달하거나 네트

워크 인터페이스 블록으로부터 RTP/RTCP 패킷을 전달받아 필요한 정보들을 추출하는 기능을 담당한다. 네트워크 인터페이스 블록은 RTP/RTCP 패킷의 네트워크 송·수신을 담당한다.

3.2 RTP 통신 모듈의 세부 블록 설계

3.2.1 RTP/RTCP 패킷 처리 블록의 설계

RTP/RTCP 패킷 처리 블록은 RTP/RTCP 패킷을 생성하거나 RTP/RTCP 패킷에서 필요한 정보를 추출하는 기능을 담당한다. RTP/RTCP 패킷 처리 블록은 (그림 3)에서 보이는 바와 같이 RTP/RTCP 패킷 생성기와 RTP/RTCP 패킷 해독기, 순서 번호 발생기, 패킷 유효성 검사기로 구성된다.



(그림 3) RTP/RTCP 패킷 처리 블록

RTP/RTCP 패킷 생성기는 멀티미디어 데이타를 전송하기 위한 RTP 패킷을 생성하는 기능과 QoS 정보를 전송하기 위한 RTCP 패킷을 생성하는 기능을 제공한다. RTP 패킷을 송신하는 경우에는 QoS 관리 정보 처리 블록에서 전달받은 값들을 참조하여 RTP 헤더 필드 값들을 설정하고 이러한 값들을 통신상에 적합한 자료형으로 변환하는 marshalling 과정을 거쳐 RTP 헤더를 생성한다. 이러한 인코딩 과정을 통해 만들어진 RTP 헤더와 상위 응용에서 전달받은 멀티미디어 데이타가 서로 결합되어 RTP 패킷이 생성된다. RTCP 패킷을 송신하는 경우에는 QoS 관리 정보 처리 블록에서 전달받은 값들을 참조하여 RTCP 헤더를 포함한 복합 RTCP 패킷을 생성한다.

RTP/RTCP 패킷 해독기는 하위 전송 프로토콜을 통해 전달받은 RTP/RTCP 패킷으로부터 필요한 정보를 추출한다. RTP 패킷의 수신시에는 우선 RTP 헤더와 멀티미디어 데이타를 분리한다. RTP 헤더는 디코딩 과정을 통해 각 필드의 값들이 추출되며 이렇게 추출된 정보들은 패킷 유효성 검사 과정을 통해 패킷의 유효성을 검증 받게 된다. 유효성 검사 과정에서 패킷의 유효성이 입증된 경우에는 RTP 헤더의 각 필드 값들은 QoS 관리 정보 처리 블록에 전달되며 멀티미디어 데이타는 상위 응용에 전달된다. 유효성 검사 과정에서 패킷이 유효하지 않다고 판정되면 해당 패킷에 대한 처리는 더 이상 진행되지 않고 무시된다. 수신한 RTCP 패킷의 처리 과정도 RTP 패킷의 수신시와 근본적으로 동일하다.

패킷 유효성 검사기는 RTP/RTCP 패킷 헤더에서 추출한 각 필드 값과 패킷의 길이에 대한 정보를 사용하여 RTP/RTCP 패킷이 유효한지 검사하며 순서 번호 발생기는 RTP 패킷 생성시 RTP 패킷에 순차적인 번호를 할당한다.

3.2.2 네트워크 인터페이스 블록의 설계

네트워크 인터페이스 블록은 RTP 패킷과 RTCP 패킷의 송·수신을 위해 UDP와 RTP 통신 모듈간의 인터페이스를 제공하는 블록이다. 네트워크 인터페이스 블록은 패킷의 송·수신을 위해 시스템에서 제공하는 소켓 인터페이스를 사용한다. 소켓을 구성하기 위해 필요한 포트 번호와 주소는 상위 응용에서 전달한다.

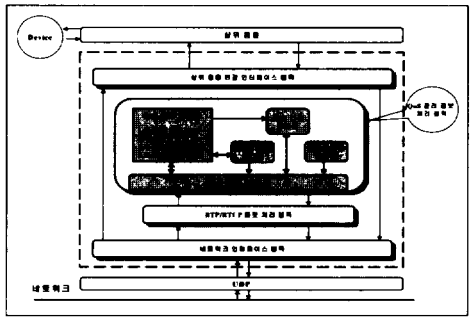
네트워크 인터페이스 블록은 RTP 패킷과 RTCP 패킷의 송·수신을 위해 네 개의 쓰레드들로 구성되며 각각의 쓰레드들은 유니캐스트와 멀티캐스트 기능을 지원한다.

3.2.3 QoS 관리 정보 처리 블록의 설계

QoS 관리 정보 처리 블록은 RTP/RTCP 패킷의 송·수신 과정을 통해 획득한 QoS 정보들을 관리하는 역할을 수행한다. 이렇게 관리되는 QoS 정보들을 통해 상위 응용은 서비스의 질을 만족시키기 위한 QoS 정책을 수행할 수 있다. 아래 (그림 4)에서는 설계한 QoS 관리 정보 처리 블록의 구조를 보인다.

본 QoS 관리 정보 처리 블록에서 취급하는 QoS 정보들로는 Round-Trip-Time과 패킷 분실율, 그리고 짧은 기간의 패킷 전달 상황을 지시하는 지터 값 등이 있다. 이러한 정보 외에도 이 블록에서 취급하는 정보들로는 세션의 대역폭이나 세션 참가자에 대한 정보, 각 참가자들의 데이타 전달 상황과 같은 세션에 관련

된 정보들도 유지하고 있다.



(그림 4) QoS 관리 정보 처리 블록

QoS 관리 정보 처리 블록은 기능에 따라 크게 세 가지로 구분될 수 있다. 첫 번째는 다른 참가자들 및 자신의 QoS 정보를 저장하는 QoS 정보 테이블이다. 두 번째는 QoS 정보를 자체적으로 생성하여 생성된 정보들을 요구하는 객체들에게 전달하는 QoS 정보 생성기들이다. 마지막으로는 QoS 정보 전반을 다루며 QoS 관리 및 패킷 생성에 필요한 계산과 처리를 수행하는 QoS 정보 관리기가 있다.

QoS 정보 테이블에는 자신의 QoS 정보를 관리하는 자료 구조를 가진 테이블과 다른 참가자와 자신의 패킷 송·수신 상태에 관한 정보를 유지하는 테이블, RTCP 패킷 전송 간격 사이에 세션의 다른 참가자로부터 수신된 RTP 패킷에 관한 정보를 유지하는 큐 형태의 자료구조를 가진 테이블, 세션 전반에 관한 정보를 유지하고 있는 테이블이 있다.

QoS 정보를 자체적으로 생성하는 기능을 가진 객체들로는 RTCP 전송 간격을 계산하는 모니터, NTP 타임스탬프[11]와 RTP 타임스탬프와 같은 타임스탬프를 발생하는 발생기, 그리고 도착간 지터나 Round-Trip-Time과 같은 패킷 송·수신 상태에 관한 정보를 생성하는 생성기 등이 있다.

QoS 전반에 관한 처리를 담당하는 QoS 정보 관리기는 QoS 관리 정보 처리 블록의 다른 모듈들로부터 정보를 받아 RTP/RTCP 패킷 처리 블록이나 상위 응용의 요구에 대한 QoS 정보를 전달하는 역할을 담당한다. 그리고 세션의 다른 참가자들과 RTP/RTCP 패킷을 주고받을 때 생성되는 QoS 정보들을 QoS 관리 정보 처리 블록의 다른 모듈들에게 전달하여 필요한 QoS 정보를 생성할 수 있도록 지원한다.

3.2.4 상위 응용 연결 인터페이스 블록의 설계

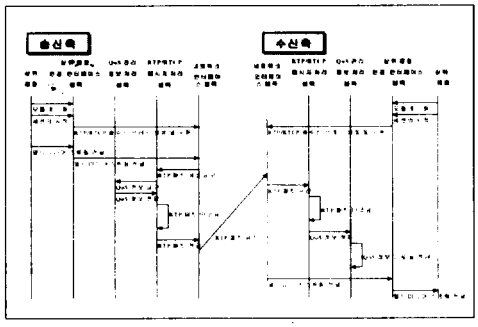
상위 응용 연결 인터페이스 블록은 상위 응용과 RTP 통신 모듈간의 인터페이스를 제공한다. 즉, 상위 응용의 RTP 통신 모듈에 대한 접근은 상위 응용 연결 인터페이스 블록을 통해서만 가능하며, 상위 응용은 상위 응용 연결 인터페이스 블록에서 제시하는 인터페이스들을 조작함으로써 동적으로 구성될 수 있다. 상위 응용 연결 인터페이스 블록의 기능으로는 멀티미디어 데이타나 QoS 정보 등을 상위 응용과 RTP 통신 모듈 사이에서 서로 교환할 수 있게 하는 것이다.

3.3 RTP 통신 모듈의 동작 시나리오

RTP 패킷의 송·수신 과정은 상위 응용 연결 인터페이스 객체를 생성함으로써 시작한다. 상위 응용 연결 인터페이스 객체를 생성함으로써 세션에 필요한 기본적인 정보들을 설정하게 되고 모듈의 초기화가 이루어진다. 모듈의 초기화가 이루어지면 상위 응용은 상위 응용 연결 인터페이스 블록을 통해 네트워크 인터페이스 블록의 RTP/RTCP 송·수신 쓰레드를 생성하고 구동함으로써 RTP 통신 모듈을 이용한 세션이 시작된다.

RTP 패킷 송신 측의 경우, 상위 응용은 네트워크를 통해 전송할 멀티미디어 데이타를 상위 응용 인터페이스 블록의 송신용 버퍼에 넣는다. 네트워크 인터페이스 블록은 송신용 버퍼를 체크하여 버퍼가 비어 있지 않다면 RTP/RTCP 패킷 처리 블록에 RTP 패킷을 생성하기를 요구한다. 그러면 RTP/RTCP 패킷 처리 블록은 QoS 관리 정보 처리 블록에서 RTP 패킷 생성시 필요한 정보들을 획득하여 RTP 패킷을 생성한다. 생성된 RTP 패킷은 다시 네트워크 인터페이스 블록으로 보내져 하위 전송 프로토콜에 의해 전송된다.

RTP 패킷 수신 측의 경우, 네트워크 인터페이스 블록으로 RTP 패킷이 수신되면 수신된 RTP 패킷을 RTP/RTCP 패킷 처리 블록으로 보낸다. RTP/RTCP 패킷 처리 블록은 수신된 RTP 패킷을 디코딩해서 RTP 패킷 헤더에 적재되어 있던 여러 정보를 획득하여 패킷 유효성 검사를 수행한다. 패킷이 유효하다면 멀티미디어 데이타는 상위 응용 연결 인터페이스 블록의 수신용 버퍼에 적재되고 RTP 패킷 헤더에서 획득한 정보들은 QoS 관리 정보 처리 블록으로 보내져 적재되거나 지터 값과 같이 상위 응용에서 이용할 수 있는 형태로 가공되어 적재된다. 이러한 과정을 (그림 5)에서 보인다.

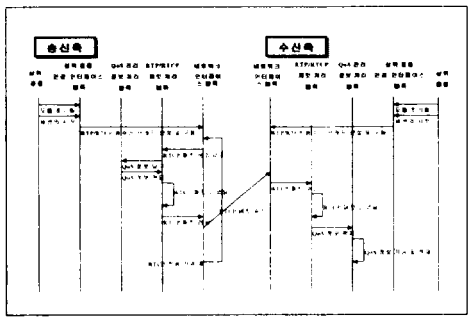


(그림 5) RTP 패킷 송·수신 시나리오

RTP의 부가적인 확장으로서 사용되는 RTCP 패킷의 송·수신 과정을 살펴보면 모듈의 초기화와 세션의 시작 과정까지는 RTP 패킷의 송·수신 과정과 동일하다.

RTCP 패킷 송신 측의 경우, QoS 관리 정보 처리 블록에서 계산된 RTCP 전송 간격에 따라 복합 RTCP 패킷을 주기적으로 전달한다. RTCP 패킷의 송신은 네트워크 인터페이스 블록이 RTP/RTCP 패킷 처리 블록에게 RTCP 패킷을 생성하도록 요구하는 것으로부터 시작된다. 이러한 요구를 받은 RTP/RTCP 패킷 처리 블록은 RTCP 패킷을 생성하기 위해 필요한 정보들을 QoS 관리 정보 처리 블록으로부터 전달받아 RTCP 패킷을 인코딩한다. 인코딩해서 생긴 복합 RTCP 패킷은 다시 네트워크 인터페이스 블록에 전달되어 네트워크로 송신된다.

RTCP 패킷의 수신 측의 경우, 네트워크 인터페이스 블록에서 수신한 복합 RTCP 패킷을 RTP/RTCP 패킷 처리 블록에 보내어 패킷 디코딩 과정을 수행한다. 패킷 디코딩 과정을 통해 획득한 여러 정보들은 QoS 관리 정보 처리 블록에 적재되거나 전송 지연 시간과 같이 상위 응용에서 필요로 하는 정보로 가공되어 적재된다. RTCP 패킷의 송·수신 과정을 (그림 6)에서 보인다.

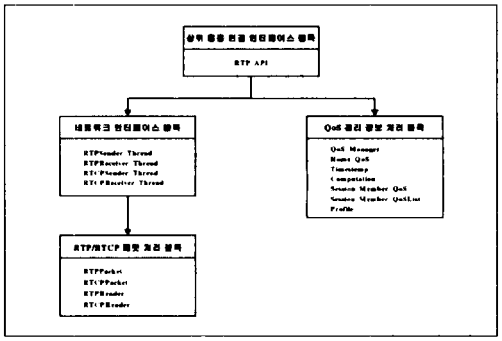


(그림 6) RTCP 패킷 송·수신 시나리오

4. RTP 통신 모듈의 구현

4.1 RTP 통신 모듈의 전체 구조 구현

(그림 7)에서는 RTP통신 모듈을 구현하기 위한 클래스들을 블록별로 나누어 예시하고 있으며, 각 블록들 간의 생성 관계를 설명하고 있다.



(그림 7) RTP 전체 블록별 클래스 구성

RTP 통신 모듈의 구동은 상위 응용 연결 인터페이스 블록을 구현한 RTP_API 객체를 생성함으로서 시작된다. RTP_API 객체는 RTP 통신 모듈의 다른 객체들이 QoS 정보들에 접근하기 위해 필요한 QoS 관리 정보 처리 블록의 QoS_Manager 객체를 생성한다. 생성된 QoS_Manager 객체는 QoS 정보를 생성, 유지하기 위해 QoS 관리 정보 처리 블록의 다른 객체들을 생성한다. 또한 RTP_API 객체는 RTP/RTCP 패킷을 송·수신하기 위해 필요한 네트워크 인터페이스 블록의 네 개의 송·수신 쓰레드들을 생성, 구동한다. 이렇게 구동된 네트워크 인터페이스 블록의 쓰레드들은 RTP/RTCP 패킷을 송·수신하는 과정에서 필요에 따라 RTP/RTCP 패킷 처리 블록의 객체들을 생성한다.

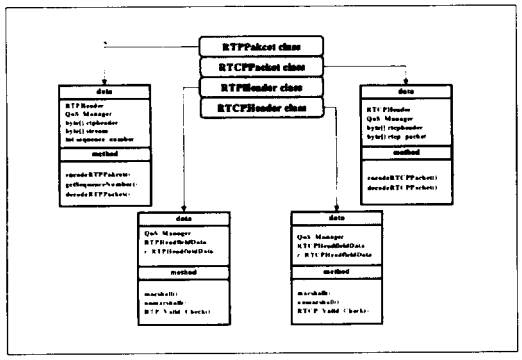
4.2 RTP 통신 모듈의 세부 블록 구현

4.2.1 RTP/RTCP 패킷 처리 블록의 구현

RTP/RTCP 패킷을 생성하거나 해독하는 기능을 담당하는 RTP/RTCP 패킷 처리 블록은 크게 패킷 생성 부분과 패킷 헤더 생성 부분으로 나뉘어 구현되었으며, 구체적인 클래스 구조는 (그림 8)에서 보이는 바와 같다.

RTPPacket 클래스와 RTCPPacket 클래스에는 encodeRTPPacket()과 decodeRTCPPacket()과 같이 해당 패

킷을 생성, 해독하기 위한 메소드들이 있다. RTCPPakcet 클래스에는 RTCP 패킷의 유형별로 marshalling, unmarshalling하는 메소드들이 있으며 특히 SDES 패킷을 생성할 때는 Profile에서 정의한 형식에 맞게 SDES 요소들을 구성한다. 그리고 세션 참가자가 세션을 떠나는 경우 BYE 패킷을 보내고 그 시스템을 종료할 수 있게 설정되어 있다. 이 두 클래스의 데이타 멤버로는 헤더의 생성과 해독을 위해 사용할 Header 객체와 QoS 정보를 조작하기 위한 QoS_Manager 객체나 해당 패킷을 구성하여 일시적으로 저장하기 위한 바이트 배열 등이 있다.

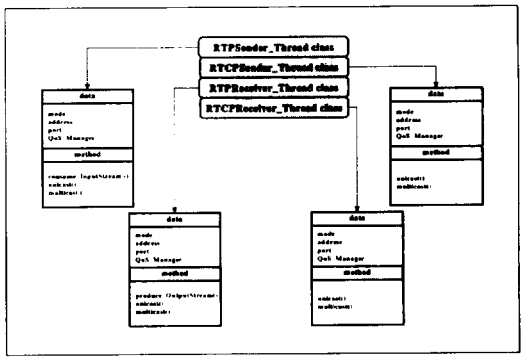


(그림 8) RTP/RTCP 패킷 처리 블록의 주요 클래스들

RTPHeader 클래스와 RTCPHeader 클래스에는 해당 패킷의 헤더를 통신에 적합한 자료형으로 바꾸기 위한 marshall()과 수신한 패킷의 헤더에서 필요한 정보들을 추출하기 위한 unmarshall(), 그리고 수신한 패킷의 유효성을 검사하기 위한 메소드들이 있으며 멤버 데이타로는 Packet 클래스와 마찬가지로 QoS 정보를 다루기 위해 QoS_Manager 객체 등이 있다.

4.2.2 네트워크 인터페이스 블록의 구현

(그림 9)의 클래스 구조를 갖는 네트워크 인터페이스 블록은 RTP/RTCP 패킷의 송·수신을 각각 담당하는 네 개의 쓰레드로 구현되며, 각각의 쓰레드들은 유니캐스트 방식과 멀티캐스트 방식을 모두 지원할 수 있도록 하였다. 각 쓰레드들은 패킷을 전송하기 위해 UDP 소켓 인터페이스를 사용하였으며 IP 주소와 포트 번호는 상위 응용에서 전달받도록 하였다. 그리고 패킷의 생성과 해독을 위해 해당 Packet클래스들을 데이타 멤버로 가지게 하였다.

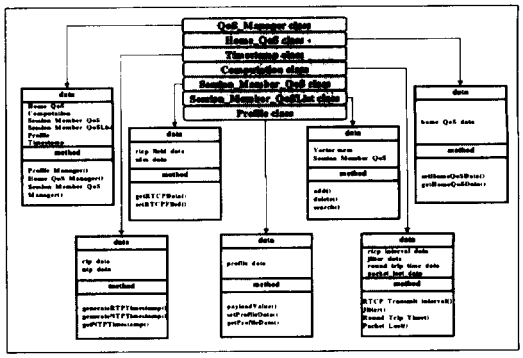


(그림 9) 네트워크 인터페이스 블록의 주요 클래스들

RTPSender_Thread 객체와 RTPReceiver_Thread 객체에는 앞서 말한 기능뿐만 아니라 멀티미디어 데이타를 상위 응용 연결 인터페이스 블록으로부터 받아오거나 저장할 수 있도록 하는 메소드들이 첨가되어 있다.

4.2.3 QoS 관리 정보 처리 블록의 구현

QoS 관리 정보 처리 블록은 QoS 정보들을 생성, 저장, 제공하기 위해 (그림 10)에서와 같은 7개의 클래스로 구현되었다. QoS 관리 정보 처리 블록의 모든 객체는 QoS_Manager 객체에 의해 관리되며 다른 객체와의 정보 전달도 이 객체에 의해 이루어진다.



(그림 10) QoS 관리 정보 처리 블록의 주요 클래스들

Home_QoS 클래스는 자신의 identifier인 SSRC나 CNAME과 같은 RTP 통신 모듈을 구동한 소스 측의 정보를 유지하는 자료 구조이며 Session_Member_QoS 클래스는 세션의 다른 참가자와 RTP 통신 모듈을 구동한 소스 사이에서 패킷을 주고받음으로써 획득한 도착간 지터 값이나 전송 지연 시간과 같은 QoS 정보를

유지하는 자료구조이다.

Session_Member_QoSList 클래스는 RTCP 패킷 전송 간격 사이에 RTP 패킷을 수신하여 얻은 타 세션 참가자의 정보를 저장하기 위한 클래스로서 SR 또는 RR의 Report Block을 생성하는데 사용된다. 이 클래스에 저장되는 타 세션 참가자 정보는 미리 그 크기를 알 수 없고 시시각각 그 크기가 변하므로 공간 최적화와 확장 가능성을 고려하여 자바에서 지원하는 Vector 클래스를 사용하여 해당 참가자 정보를 저장하였다. 그리고 이 클래스에 저장된 정보를 다루기 위해 정보 저장을 위한 add(), 리스트 내의 정보들을 전달한 뒤 리스트를 초기화하는 delete(), 그리고 리스트 내의 정보들을 검색하는 search()와 같은 메소드들을 설정하였다.

Profile 클래스는 세션 대역폭이나 헤더 확장과 같은 세션 전반적인 정보들을 설정하거나 유지하는 역할을 담당한다. 이 클래스의 멤버 데이타 값은 상위 응용에서 전달된 값으로 설정될 수도 있고 응용의 구현 시에 설정될 수도 있다.

Timestamp 클래스는 동기화와 지터 계산을 위해 필요한 시간 관련 정보들을 생성한다. 이 클래스의 멤버 데이타로는 chunk rate와 같이 RTP 타임스탬프를 생성하기 위한 것과 fraction과 같은 NTP 타임스탬프를 생성하기 위한 것들이 있다.

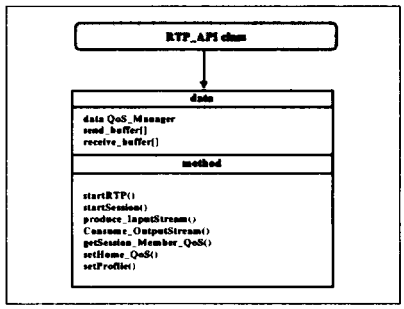
Computation 클래스는 RTP 통신 모듈에서 필요한 연산들을 수행한다. 이러한 연산으로는 RTCP 패킷의 전송 간격 계산, 도착간 지터 계산, 전송 지연 시간 계산, 패킷 손실율 계산 등이 있다.

QoS_Manager 클래스는 QoS 정보에 대한 전반적인 관리를 담당한다. 이 클래스의 멤버 데이타로는 Profile 객체와 같은 QoS 관리 정보 처리 블록의 객체들을 가지며 이러한 멤버 데이타들은 네 개의 RTP/RTCP 송·수신 쓰레드들에 의해 사용되기 때문에 쓰레드 간의 동기화가 유지되어야 한다. 그리고 QoS_Manager 클래스에는 QoS 정보들을 객체들 서로 간에 전달해 주거나 다른 객체의 기능을 사용할 수 있도록 지원하는 기능 등을 가지고 있다.

4.2.4 상위 응용 연결 인터페이스 블록의 구현

상위 응용 연결 인터페이스 블록은 RTP_API 클래스에 의해 구현되며 그 구조는 (그림 11)에서와 같다.

RTP_API 클래스의 멤버 데이타로는 멀티미디어 데이타를 상위 응용과 네트워크 인터페이스 블록으로 전달하기 위해 일시적으로 저장하는 버퍼들이 있으며 이러한 버퍼들은 원형 큐의 형태로 구현되었다. RTP_API 클래스의 메소드로는 SDES 정보나 프로파일 정보를 설정해 주는 것들과 지터 값과 같은 네트워크 상태에 관한 QoS 정보를 획득할 수 있도록 지원해주는 것들이 있다.



(그림 11) 상위 응용 연결 인터페이스 블록의 클래스들

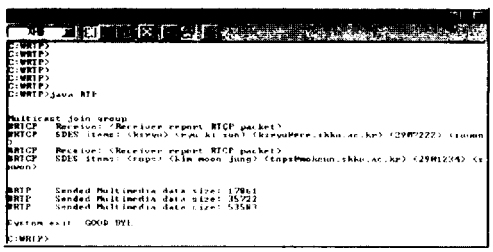
4.3 RTP 통신 모듈의 개발 환경

현실적인 RTP 통신 모듈은 특정 응용과 결합되어 사용되어지고 사용 환경도 다양할 수 있다. 그러므로 RTP 통신 모듈을 다양한 환경에서 사용할 수 있도록 개발하였다. RTP 통신 모듈의 개발 과정에서는 IBM PC(pentium 150)와 유닉스 머신(Sun sparc 20)들을 사용하였고, 프로그래밍 언어로는 플랫폼에 독립적이며 확장성도 높은 언어인 자바를 사용하였으며, 사용 툴로는 JDK 1.1을 기준으로 작성하였다.

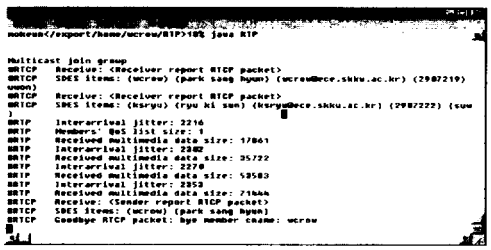
4.4 RTP 통신 모듈의 구동 결과

(그림 12)에서는 본 논문에서 구현한 RTP 통신 모듈을 이용하여 세 호스트 간의 RTP/RTCP 패킷의 송·수신 과정을 보여 주고 있다. 세 호스트는 미리 주어진 멀티캐스트 주소와 포트를 통해 호스트 간에 멀티캐스트 그룹핑이 이루어지게 되고 UDP를 이용하여 패킷들을 송·수신하게 된다. (그림 12)에서 #RTP는 RTP 패킷의 송·수신 과정 중에 발생하는 지터 계산과 송·수신한 패킷의 크기 등을 나타내고, #RTCP는 SDES 요소들과 같이 RTCP 패킷의 송·수신 과정 중에 획득한 정보나 BYE 패킷에 대한 처리를 나타내고 있다. (그림 12)에서는 RTCP 패킷을 RTCP 전송 간격에 따라 세 호스트 모두 주기적으로 송·수신하는 과정과 각 호스트에서 수신한 복합 RTCP 패킷 중

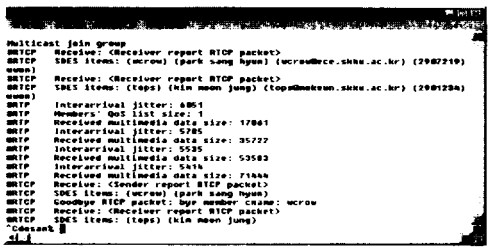
SDES의 내용을 보여 주고 있다. 그리고 (그림 12)의 (a)에서는 (a)호스트에서 송신한 멀티미디어 데이타의 크기들을 보여주고 있으며, (그림 (b), (c))에서는 (b), (c)호스트가 각각 수신한 멀티미디어 데이타의 크기를 보여주고 있다. 또한 (a)호스트의 참가자가 세션을 떠날 때 응용이 BYE 패킷을 보낸 뒤 자동 종료하는 모습을 보여주고 있다.



(a) IBM PC(pentium 150)



(b) SUN Sparc



(c) SUN Sparc

(그림 12) RTP/RTCP 통신 모듈의 구동 예

5. 평 가

본 논문에서는 멀티미디어 데이타의 효율적인 네트워크 전송을 위해 RTP 통신 모듈을 설계하였으며 관련 RFC들의 내용을 준수하여 구현하였다. RTP 통신 모듈의 각 기능들을 구현하면서 특징적인 측면들을 살펴본다면 다음과 같다.

RTP의 특성상 그 자체로는 완전하지 않은 프로토콜 골격이므로 결국 구현 과정에는 응용에 통합되어 특정 응용에 종속되어 사용되는 것이 보편적이나, RTP 자체가 모든 응용에 걸쳐 공통분모로 예상되는 기능에 대해 명시한 것이므로 또한 상위 응용과 구별되어 독자적으로 구현될 수도 있는 여지를 주고 있다. 본 논문에서는 이러한 상위 응용과 독립적인 부분들을 구분하고 상위 응용에게는 몇몇 인터페이스만을 주어 RTP 통신 모듈을 제어할 수 있게 함으로써 RTP 통신 모듈의 실제적인 동작을 상위 응용에게서 숨기도록 하였다. 이러한 설계는 RTP 통신 모듈이 특정 응용에만 국한되어 설계되는 일반적인 방법과 달리 보편적인 여러 응용에 재사용할 수 있게 한다는 장점을 갖는다. 또한 RTP_API 클래스만을 프로그래머가 접근하여 응용을 구현함으로써 다른 프로그래머들이 이 프로그램 소스를 재사용하기가 훨씬 쉬워진다는 장점을 가진다.

그리고 RTP 패킷 송신과정을 쓰레드로 활성화시켜 상위 응용의 멀티미디어 데이타 전달과 하위의 RTP 패킷 송신 과정을 분리하여 RTP 통신 모듈의 상위 응용 독립성과 효율성을 높였다. RTP 통신 모듈에서 멀티미디어 데이타의 송·수신만큼 중요한 문제가 RTP/RTCP 송·수신을 통해 생성된 QoS 정보들의 관리이다. QoS 정보들은 여러 활성 객체들이 비동기적으로 접근하므로 일관성 문제와 스케줄링 등의 문제가 발생할 수 있다. 이를 극복하기 위해서 본 구현에서는 자바에서 지원하는 모니터와 같은 쓰레드 간의 동기화 기법을 사용하였으며 스케줄링 부분은 자바 런타임 시스템에서 제공하는 기본적인 스케줄링 알고리즘에 의존하고 있다. 그리고 QoS 정보들을 각 멤버들마다 유지하여야 하므로 각 멤버마다 하나의 QoS 정보를 저장할 수 있는 테이블 형식의 객체를 활성화하였고 이러한 객체들을 구분하기 위해 SSRC를 식별자로 사용하였다.

본 논문의 RTP 통신 모듈에는 필수적인 기능들을 대부분 설계하고 구현되어 있다. 그러나 RTP 통신 모듈의 기능 확장을 위해 고려해 볼 측면은 아직 남아 있다. 특정 응용에서의 RTP 통신 모듈의 적응성, 예를 들자면 특정 응용과 프로파일에 의한 RTP 패킷 헤더의 수정과 확장들에 대한 고려가 필요하며, 변환기와 혼합기에 대한 고려라든지, 대역폭이 일정하지 않은 네트워크를 공유하는 세션에서 발생하는 문제를 어떻게

극복하는지 등 아직 고려해야할 상황이 남아 있다.

6. 결 론

네트워크를 통한 서비스가 다양해짐에 따라 멀티미디어 데이타를 송·수신하는 응용의 개발이 증가하고 있다. 따라서 멀티미디어 데이타의 연속성, 실시간성 및 오류 허용성과 같은 특성을 고려하여 만들어진 RTP는 매우 중요한 역할을 할 것으로 보인다.

본 논문에서는 실시간 전송 기능을 제공하고 전송간 발생하는 QoS 정보를 관리하며 이를 응용에 반영하여 멀티미디어 응용의 QoS 관리를 지원할 수 있는 프로토콜인 RTP를 분석하고 설계 및 구현하였으며, RTP/RTCP 패킷 송·수신 시나리오와 각 블록의 구성과 클래스들의 내용을 살펴봄으로써 RTP 통신 모듈의 동작 메커니즘을 소개하였다.

본 논문에서 설계 및 구현 결과를 소개한 RTP 통신 모듈은 서비스에 대한 질의 보장이 문제시되고 있는 현 네트워크의 실정에 QoS를 고려하려할 수 있는 프레임워크를 제공하고 있으며 또한 다양한 멀티미디어 데이타 형식을 지원하므로 활용 범위가 넓다고 할 수 있다. 향후에는 RTP 프로토콜의 저변 확대를 위해서 기존 멀티미디어 응용 및 세션 제어 프로토콜 등 관련 분야의 표준에 대한 연계가 요구되고 각 분야간의 표준 인터페이스의 확립에 대한 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] S. R. McCanne, "Scalable Compression and Transmission of Internet Multicast Video," Report No.UCB/CSD-96-928, University of California Berkeley, Dec. 1996.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP : A Transport Protocol for Real-Time Applications," IETF RFC 1889(ftp://ftp.isi.edu/in-notes/rfc1889.txt), Jan. 1996.

[3] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control," IETF RFC 1890(ftp://ftp.isi.edu/in-notes/rfc1890.txt), Jan. 1996.

[4] H. Schulzrinne, "Real Time Streaming Protocol," IETF RFC 2326(ftp://ftp.isi.edu/in-notes/rfc2326.txt), Apr. 1998.

[5] L. Berc, W. Fenner, R. Frederick, S. McCanne, and P. Stewart, "RTP Payload Format for JPEG-compressed Video," IETF RFC 2435(ftp://ftp.isi.edu/in-notes/rfc2435.txt), Oct. 1998.

[6] T. Turletti and C. Huitema, "RTP Payload Format for H.261 Video Streams," IETF RFC 2032(ftp://ftp.isi.edu/in-notes/rfc2032.txt), Oct. 1996.

[7] C. Zhu, "RTP Payload Format for H.263 Video Streams," IETF RFC 2190(ftp://ftp.isi.edu/in-notes/rfc2190.txt), Sep. 1997.

[8] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," IETF Internet-Draft (ftp://ftp.ietf.org/internet-drafts/draft-ietf-avt-crtp-05.txt), Jul. 1998.

[9] ITU-T, "Video Codec For Audiovisual Services at p*64 Kbit," ITU-T Recommendation H.261, Mar. 1993.

[10] ITU-T, "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Nonguaranteed Quality of Service," ITU-T Recommendation H.323, Nov. 1996.

[11] D. L. Mills, "Network Time Protocol(Version 3)," IETF RFC 1305(ftp://ftp.isi.edu/in-notes/rfc2326.txt), Mar. 1992.

박 상 현

e-mail : wcrow@mokeun.skku.ac.kr

1998년 성균관대학교 정보공학과 졸업(학사)

1998년~현재 성균관대학교 대학원 전기전자 및 컴퓨터공학과 석사과정

관심분야 : 분산 시스템, 분산 객체 시스템, 멀티미디어 통신

박 상 윤

e-mail : bronson@mokeun.skku.ac.kr

1997년 동국대학교 전자계산학과 졸업(학사)

1999년 성균관대학교 대학원 전기전자 및 컴퓨터공학과(석사)

1999년~현재 성균관대학교 대학원 전기전자 및 컴퓨터공학과 박사과정

관심분야 : 분산 시스템, 분산 객체 시스템, 멀티미디어 통신, 이동 컴퓨팅

김 명 준

e-mail : joonkim@etri.re.kr

1978년 서울대학교 자연과학대학 계산통계학과 졸업(학사)

1988년 한국과학기술원 전산학과(석사)

1986년 프랑스 Nancy 제1대학교 응용수학 및 전산학과(박사)

1980년~1981년 아주대학교 종합연구소 연구원

1981년~1986년 프랑스 Nancy 전산학 연구소(CRIN) 연구원

1986년~현재 한국전자통신연구원 컴퓨터·소프트웨어 기술연구소, 선임/책임연구원

1993년 프랑스 Univ. of Nice Sophia-Antipolis 방문 교수

관심분야 : 데이터베이스, 분산 시스템, 인터넷 서비스

엄 영 익

e-mail : yieom@ece.skku.ac.kr

1983년 서울대학교 계산통계학과 졸업(학사)

1985년 서울대학교 대학원 전산과학전공(석사)

1991년 서울대학교 대학원 전산과학전공(박사)

1983년~1986년 서울대학교 도서관 조교

1986년~1993년 단국대학교 전자계산학과 부교수

1993년~현재 성균관대학교 전기전자 및 컴퓨터공학부 교수

관심분야 : 분산시스템, 분산 객체 시스템, 이동 컴퓨팅, 멀티미디어 통신