

논문 2007-44TC-1-7

디지털 케이블 방송 환경에서 개인 미디어를 위한 스트리밍 프로토콜 연구

(A Research on Streaming Protocol for User-Created Contents in
Digital Cable Broadcasting environment)

김 성 원*, 김 정 환*, 시 장 현*, 정 문 열**

(Seong Won Kim, Jung Hwan Kim, Jang Hyun Si, and Moon Ryul Jung)

요 약

최근 인터넷에서 많은 관심을 보이고 있는 동영상 개인 미디어(UCC:User-Created Contents)를 영상 콘텐츠의 기본적인 매체라 할 수 있는 TV상에서 직접 즐길 수 있다면 관련 미디어의 대중적인 효과는 배가 될 것이다. 디지털 케이블 방송에서 UCC와 같은 다양하고 많은 미디어 콘텐츠를 서비스하고자 한다면 현재의 OCAP 표준 환경에서는 RVOD(Real Video on Demand)와 동일한 서비스 모델을 가져야 하는 부담과 UCC 제공을 위한 방송 플랫폼별 인터페이스가 필요할 것이다. 이러한 문제점을 보완하기 위해 Return Path기반으로 RTP(Real-time Transport Protocol)을 이용해 기존의 VOD방식의 스트림 대역에서 분리하고 별도의 네트워크 VOD 프로토콜을 구현하고자 한다. 이는 라이브 방송환경과 동일한 인프라의 지속적인 확장보다 기존의 네트워크 환경을 이용하여 VOD 서비스를 위한 인프라에 많은 영향을 받지 않는 서비스가 가능할 수 있도록 하는 것이다. 따라서 본 논문에서는 RTP 사용을 위해 셋톱박스상의 다운로드 가능한 구조의 클래스 분석과 디코딩을 위한 전송 스트리밍 서버 및 트랜스코딩 과정의 설계에 중점을 두고 서술한다. 경량화 된 VOD 서비스 환경을 설계함으로써 방송환경 하에서 주문형 서비스의 효율적인 확장성을 보장할 수 있는 기회를 가져올 수 있을 것으로 기대한다.

Abstract

In this paper, on-demand service streaming protocol for user-created contents in digital cable broadcasting environment is considered. In order to provide variety media contents service like UCC in digital cable broadcasting environment, the same service model as RVOD(Real Video on Demand) is required and the different interface for each broadcasting platform is needed in a current OCAP environment. Using return path based RTP, we separate existing VOD stream band into broadcasting and VOD stream band. In order to broaden On-demand service, consistent expansion of the infrastructure same as live broadcasting system is inefficient in the digital cable broadcasting environment. Using existing network protocol, the service which is insensitive to the infrastructure for VOD service becomes possible. Therefore we considered the analysis of the class of download available structure in the Set-Top-Box for RTP(Real-time Transport Protocol) and designing the decoding available streaming server for UCC transcoding and transmission in the receiver. Designing a efficient VOD service and system under the broadcasting environment gives a expansion of On-Demand service and more chance to upload and utilization of contents.

Keywords : UCC, RTP, RVOD, OCAP, 케이블방송

I. 서 론

개인 미디어의 발전은 인터넷이라는 매체를 통해 시간이 지날수록 다양한 콘텐츠와 네트워크 기술로 거듭 발전하고 있다. 특히 동영상 UCC(User-Created Contents) 전문 사이트가 폭발적인 인기를 누리고 있으며 콘텐츠의 주인공이 사회적으로 이슈가 되는 현상들

* 학생회원, ** 정회원, 서강대학교 영상대학원
미디어공학과

(The Department of Media Technology,
Sogang University Graduate School of Media
Communications

※ 본 논문은 “2006년 한국방송공학회 학술대회”에서
일차 발표된 논문에 기초한 연구결과입니다.

접수일자: 2006년12월10일, 수정완료일: 2007년1월15일

을 미루어 볼 때 사용자 제작 동영상 콘텐츠의 상업적, 문화적 가치는 미디어 분야에서 재평가 될 것으로 예상된다. 이러한 콘텐츠가 디지털 TV 화면을 이용해서 보다 대중적이고 보편적으로 접근 가능하다면 그 효과는 배가 될 수 있을 것이다.

하지만 이러한 개인의 콘텐츠를 방송으로 전송할 수 있는 시스템은 MPEG-2 TS(Transport Stream) 기반의 전송 시스템을 갖춘 방송 플랫폼에 추가적으로 콘텐츠를 위한 시스템을 구축하거나 해당 서비스를 연계하는 망을 개설하는 등의 접근은 인터넷 시스템과는 다른 방송 시스템으로서는 쉽게 풀 수 있는 문제가 아니다. UCC 활용 서비스를 위한 방송 플랫폼의 인프라적인 제한 사항을 극복하기 위해서는 기존의 네트워크 시스템을 이용해 응용 가능한 프로토콜을 OCAP(OpenCable Application Platform) 환경에서 구현이 이루어져야 되며 현재 구축되어 있는 인프라를 변경하거나 새로운 시스템을 증축하는 일은 사실상 불가능하다.

본 논문에서는 인터넷에서 실시간 미디어 전송을 위해 쓰이는 RTP(Real-time Transport Protocol: RFC1889)를 이용해 OCAP 환경에서 Return Path를 이용한 실시간 스트리밍 프로토콜과 서비스를 위한 시스템의 설계에 대해 기술한다. 응용하고자 하는 프로토콜에 대한 정의와 서비스 시스템은 웹 환경에서 보다 많은 사용자 접근을 통한 콘텐츠 제공과 이에 따라 생성된 콘텐츠를 디지털 케이블 방송 환경에서 송출이 가능한 형태의 시스템 설계를 목적으로 한다. 따라서 셋톱박스상의 스트림 접근을 위한 미디어 컨트롤 API 구조를 분석하고 스트리밍을 받는 클라이언트의 설계와 웹을 통한 콘텐츠 등록이 가능한 클라이언트 대응의 스트리밍 서버가 고려되어야 한다.

이에 본 논문에서는 RTP를 통해 전송된 미디어 스트림의 접근을 위한 OCAP 표준 API에 대해 분석하고 현재의 구조상의 문제점과 미디어 컨트롤이 가능한 애플리케이션의 구조를 도출하고 콘텐츠 제공을 위한 사용자 인터페이스의 구현 및 스트리밍 서버의 구현 방법에 대해 설명한다. 본 논문의 II장에서는 UCC의 개념과 방송환경에서 서비스 가능한 모델에 대해서 설명하고 III장에서는 콘텐츠의 전송을 위한 RTP에 대해 기술한다. 그리고 IV장에서는 프로토콜의 접근 제어 및 적용을 위한 JMF API에 대해 분석하고 UCC를 위한 클라이언트 애플리케이션의 구조를 정의한다. V장에서는 제안된 시스템의 구현과 실험의 결과를 기술한다. 마지막으로 VI장에서는 본 논문의 결론을 맺고 향후 보완해야 할

사항과 연구방향에 대해 서술한다.

II. UCC 개념 및 방송 서비스 모델

UCC는 사용자(시청자) 제작 콘텐츠로 참여, 공유, 개방의 개념이 인터넷 사회에 본격화되면서 소비자들은 적극적으로 전문지식을 갖춘 프로슈머(Prosumer:앨빈 토플러 “제3의 물결”에서 등장한 개념, 생산자=소비자)로 변화하고 있다. 위키피디아(Wikipedia), 네이버 지식검색, 싸이월드 미니홈피 등 모두 네트워크 강조형의 콘텐츠들이 대표적인 UCC의 사례로 들 수 있다. 또한 디지털 콘텐츠 제작 및 전달 기술이 발전함과 동시에 UCC 서비스는 일반인들이 사용하는데 지장이 없도록 변화되고 있으며, 기존의 텍스트나 이미지 위주의 콘텐츠에서 동영상과 같은 다양한 미디어 형태의 콘텐츠가 생산, 유통, 판매에 이르기까지 보편화되고 있다. UCC의 미디어 영향력이 현재는 인터넷에서 확장되어 이동통신에도 접목되는 등 다양한 디지털 플랫폼으로 확대되고 있는 실정이다.^[2]

1. 서비스 가능 모델

이러한 UCC를 디지털TV의 화면에서 경험할 수 있다면 보다 일반화된 매체 활용에 따른 대중적 인기를 증대시킬 수 있으며 사용자의 콘텐츠 접근 측면에서도 높은 편의성을 제공할 수 있을 것이다.

본 논문에서는 방송 환경 하에 UCC 서비스 제공을 위한 방법으로 아래와 같이 시청자(콘텐츠 제작자)를 위한 인터넷상의 콘텐츠 등록 인터페이스와 등록된 콘텐츠의 송출과 셋톱박스에서 디코딩 가능한 형태의 스트림 구조로 변환하는 트랜스코딩 시스템, 그리고 셋톱

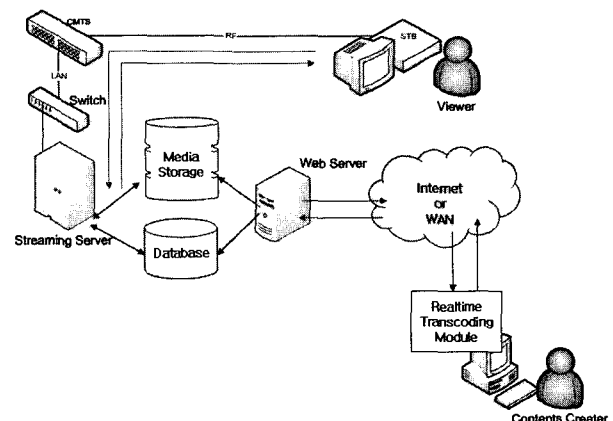


그림 1. UCC 서비스를 위한 시스템 구조
Fig. 1. Proposed system architecture for UCC service.

박스에서 수신/제어하는 구조로 설계하였다.

시청자가 제작한 동영상은 웹을 통해 동영상 콘텐츠를 등록과정에서 실시간 트랜스코딩을 통해 MPEG-2 TS로 전송 가능한 타입의 자료로 변환하여 Media Storage에 적재됨과 동시에 해당 콘텐츠의 메타정보는 데이터베이스에 저장된다. 스트리밍 서버는 RTP서버로 구현되며, Session Management 및 Command/Control의 Event에 반응한다. 클라이언트, 즉 셋톱박스와 스트리밍 서버의 프로토콜 연결은 Return Path를 통해 연결되며 케이블 종단 라우터(CMTS : Cable Modem Termination System)와 스트리밍 서버는 Fast-Ethernet상에 존재하는 것으로(전용선을 통한 Return Path일 경우 일관된 서비스 속도의 보장이 어려움) 가정한다. 클라이언트는 셋톱박스 Host-Mac Address에 할당된 IP를 통해 스트리밍 서버로 TCP/IP로 접근하게 되고 스트리밍 서버는 RTP를 통해 콘텐츠를 전송하게 된다. 이때 클라이언트와의 Video Control 통신은 RTSP(Real-time Streaming Protocol)를 사용하게 된다. 관련 프로토콜 스택에 대한 정의는 다음 장에서 상세히 다루도록 하겠다.

이러한 구조의 Thin-VOD 시스템의 구조는 고품질의 유료 서비스가 아닌 정보형의 주문형 동영상 시스템에 적합할 것이며, UCC 와 같은 형태의 방송 서비스는 개인의 콘텐츠 제작의 포맷 및 품질(bitrate, 해상도 등)의 일관성이 낮을 수 밖에 없기 때문에 이와 같은 서비스 구조에 해당된다고 할 수 있다.

III. 스트리밍 프로토콜

현재 OCAP 환경에서 인터넷상의 VOD시스템과 유사한 RVOD(Real Video-on-Demand : 실제 시청자의 버튼조작을 통한 조작이 가능한 주문형 Video 서비스)는 대규모 인프라 구축이 필요하며 단순한 네트워크를 통한 서비스가 아니라 방송 스트림을 통해 시청자에게 제공하는 시스템이다. RVOD 시스템은 VOD Pump, Edge QAM, SRM(Session Resource Manager) 등의 라이브 방송환경과 거의 유사한 형태의 시스템으로 구성되어 있다. 따라서 주문형 서비스의 다양화 또는 수요의 증가에 대한 인프라 증축의 부담이 생길 수밖에 없다. 이러한 문제점을 개선하기 위해 RVOD 시스템과 인터넷 프로토콜을 이용한 시스템을 비교 분석하고 UCC 서비스에 적합한 프로토콜을 설계하고자 한다.

1. 현재 OCAP 환경하의 RVOD 시스템

RVOD시스템은 인터넷상의 VOD 시스템과 동일한 서비스로 Transport Stream을 생성하는 VOD 서버 (Video Pumping 서버) Control을 통한 Start, Stop, Pause, Rewind, Fast-Forward 등의 동영상의 제어가 가능하다.

RVOD 시스템의 구성은 크게 4가지로 구분할 수 있으며 VOD서비스를 요청하는 클라이언트와 콘텐츠를 제공하는 서버, 그리고 클라이언트의 요청에 따른 Session을 관리하는 SRM과 서버로부터 콘텐츠를 서비스 위한 각 섹터별 셋톱박스로 전송을 위한 변조를 담당하는 Edge QAM으로 크게 분류 할 수 있다. CAS 및 SMS/Billing시스템 및 다양한 비즈니스 모델에 따른 시스템에 대한 언급은 본 논문에서 제외한다.

VOD 시청의 일반적인 단계는 먼저 시청자가 시청하고자 하는 콘텐츠를 선택하고 결제 및 승인이 되면, 시청하고자 하는 Video 영상이 보여지게 된다. 이때부터 시청자의 요청 정보 즉, 정지, 앞으로, 뒤로, 일시정지 등의 제어가 가능하다. 이러한 과정은 다음의 그림 2와 같은 일련의 과정을 거쳐서 처리된다.

클라이언트의 Session 요청에 따라 클라이언트 측이 선택한 Asset(Content관련 각종 정보 : 줄거리, 감독, 배우, 고객정보 등)을 SRM으로 보내면 SRM과 연결된 부가 시스템으로 인증 및 결제정보가 정상적으로 처리되게 되며, VOD 서버로 해당 콘텐츠 데이터의 Stream에 대한 Push가 이루어진다. 이에 따라서 해당 시청자의 Session의 가용여부가 판단되어 서버의 Session에 연결된다.

그리고 지속적인 클라이언트 측의 Media Control은 VOD 서버 쪽으로 지속적인 통신이 이루어지면서 스트리밍을 제어하게 된다. Edge QAM 및 Session관리의 Load Balancing을 위해 일반적으로 VOD 서버와 Edge

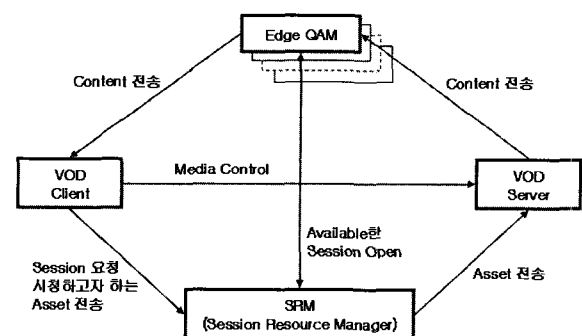


그림 2. RVOD 시스템 프로세스
Fig. 2. System process of RVOD.

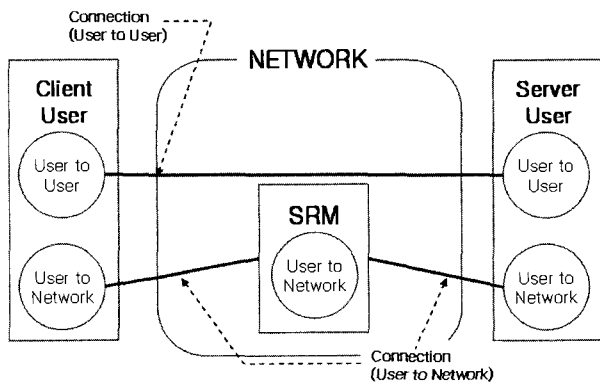


그림 3. DSM-CC Message(U-U, U-N)
Fig. 3. DSM-CC Message(U-U, U-N).

QAM에 대해 시청자들을 Cell 단위로 분리하여 각 Cell 당 Edge QAM, VOD 서버를 할당하게 된다. RVOD 시스템은 Transport Stream 환경으로 일반적인 인터넷과 같은 네트워크 프로토콜을 사용하는 방법을 가지지 않는다. 따라서 Media Control, Session 관리 등은 MPEG-2 표준의 Part 6(ISO/IEC 13818-6)에 정의한 DSM-CC (Digital Storage Media, Command and Control) 표준을 사용한다. DSM-CC는 현재 Broadband Connection을 통한 멀티미디어 서비스 및 어플리케이션에 대한 프로토콜을 제공하는 부분까지 확장되었지만 최초 제안된 목적은 VOD 및 네트워크 VTR과 같은 네트워크 Video 자원을 제어하기 위한 표준이다. 때문에 그 근본 목적에 가장 부합되는 프로토콜이 되겠다.

DSM-CC Message에는 DSM-CC U-U Message, DSM-CC U-N Message, DSM-CC Download Message로 크게 3가지의 분류로 나눌 수 있다. RVOD 시스템은 여기에서 U-U, U-N Message를 사용하여 Media Control 및 Session을 관리하게 된다. SRM을 중심으로 송수신되는 메시지는 DSM-CC U-N Message에 해당된다.

U-N Message 중에서 세션 관리 기능 블록에서 제공해야 하는 제어 절차는 Client-initiated Session Set-Up, Client-initiated Session Release, Server-initiated Resource Addition, Server-initiated Deletion과 같은 제어 절차에 따라 이루어진다.

이와 같은 방식으로 현재의 케이블 방송 환경에서는 Stream을 생성하고 제어하는 방법으로 VOD 서비스를 하고 있다.

만약 이러한 서비스에 있어서 시스템 확장이 필요하게 된다면 인터넷과 같이 서버중설 및 부하분산 정도의 단계로 완성될 수 없다. 또한 서비스 제공에 있어서도

실제 과금이 이루어 지지 않은 서비스 모델의 무료 VOD 서비스의 경우에도 이렇게 방송 스트림의 대역을 사용한다면 서비스 목적과 효율성에 대비해서 비경제적일 수밖에 없다. 이러한 한계점은 서비스 목적에 따라서 RTP/RTSP를 사용가능한 서비스로 구분한다면 서비스의 효과적인 확장과 타 매체와 연계성이 보다 원활해질 것으로 생각된다.

1. RTP(Real-Time Transport Protocol)

케이블 방송 환경에서 RTP/RTSP를 도입하게 되면 현재 VOD 서비스의 영역을 확장할 수 있게 될 것이다.

실시간 전송환경에서는 높은 대역의 Bandwidth를 필요로 하며 데이터를 수신 할 때는 전송중의 전송 지연에 대한 보상을 위한 처리보다는 손실 데이터에 대한 보상이 구현상 더 쉬운 편이기 때문에 멀티미디어 데이터의 전송에 요구되는 환경은 파일과 같은 정적인 데이터를 보내는 프로토콜에 적합하지 않다. 때문에 RTP라는 프로토콜을 사용할 수 있다. RTP는 Unicast와 Multicast 모두 사용가능하며 전송된 데이터에 대한 식별과 패킷의 순서 결정, 다중 미디어에 대한 효율적인 동기화 기능을 수행하지만, 송신단에서 보내지는 패킷의 순서대로 수신단에서 패킷을 수신한다는 보장은 없다. 또한 송신한 모든 데이터 패킷이 모두 수신단에 도달한다는 보장도 없으므로 수신단의 패킷 시퀀스를 다시 만들고, 패킷 헤더에서 제공되는 정보를 가지고 손실된 데이터 패킷을 검출하는 모든 부분이 바로 수신단의 책임이 된다. 따라서 RTP는 주기적인 전송의 보장이나, 서비스 Quality에 대한 보장이 없기 때문에 데이터 전송의 품질 보증을 위해서는 RTCP(Real-time Control Protocol : 데이터 품질과 함께 RTP session에 대한 모니터링 기능 제공)를 사용하여 보완하고 있다. RTSP는 RTP보다 상위 단계의 프로토콜로서 멀티미디어 스트림에 대한 Command 및 Control 기능을 제공하며, UDP와 같이 RTSP도 비연결지향 프로토콜이며, 각각의 스트림은

Real-Time Media Frameworks and Applications	
Real-Time Control Protocol (RTCP) Real-Time Transport Protocol (RTP)	
Other Network and Transport Protocols (TCP, ATM, ST-II, etc.)	UDP
	IP

그림 4. RTP 구조^[6]
Fig. 4. Architecture of RTP.^[6]

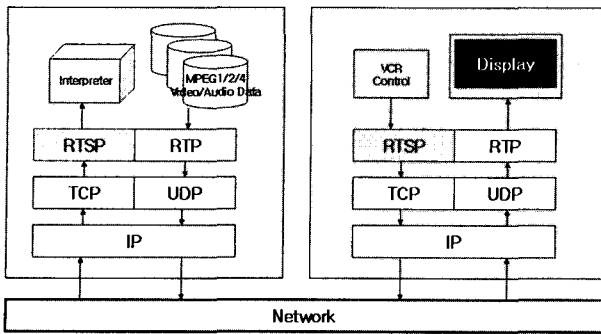


그림 5. RTP/RTSP 프로세스 스택^[11]
Fig. 5. The Stack of RTP/RTSP Process.^[11]

Session ID에 의해 서로 구별가능하다. 단, RTSP의 Control 요청은 연결이 보장된 TCP를 통해 전송된다.

RTP를 이용한 애플리케이션은 일반적인 Socket 통신과 같은 형태의 서버와 클라이언트로 구성되어 있으며 아래 그림 5와 같은 형태의 프로토콜 스택으로 구성되어 Request 및 Response의 형태가 이루어진다.

IV. 프로토콜 접근 제어 및 적용

RTP 접근을 통한 미디어 재생은 JMF (Java Media Framework) version 2.0에서는 기본적으로 javax.media.rtp를 제공하고 있지만 Java TV API는 Broadcast Media Pipeline을 관리하기 위해 JMF 1.0 API를 이용한다. 때문에 현재 RTP 접근을 위한 방법에는 새로운 인터페이스를 구현하여 접근 해야만 한다.

하지만 RTP를 통한 스트리밍 데이터의 접근방법은 JavaTV API에서 접근에 대해 기술하고 있는 방법으로 구현이 가능하다.

1. JMF 인터페이스

대부분 JMF상의 구현은 새로운 미디어 스트림이 Render 될 때마다 완전한 Decoding Pipeline이 생성된다고 간주하고 있다. 데스크탑 환경에서는 가능한 표현법이다. 일반적으로 개별적인 Network Connection은 각각의 미디어 스트림을 요구하며, 각각의 Connection은 완전히 구분된 Pipeline의 요구를 가지고 잠재적으로 Connection Source와 혼합적인 협상을 요구하게 된다.

하지만, 방송환경에서는 그렇지 않다. Broadcast Network를 위한 인터페이스는 Multiplex의 Multiplex를 모델로 할 수 있다. 이와 같은 모델은 실제 Tuner(첫번째 Multiplex)와 Demux(두번째 Multiplex)를 연결한다.

따라서 Broadcast Network로의 인터페이스의 제어는

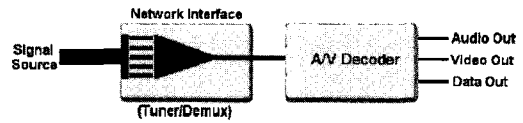


그림 6. Broadcast Network Interface^[5]
Fig. 6. Broadcast Network Interface.^[5]

튜닝(Primary Multiplex Selection)과 Stream Selection (Secondary Multiplex Section)에 의해 구성된다.^{[4][5]}

2. 프로토콜의 접근 제어

JavaTV API에서는 Television Broadcast Signal에서의 전송된 데이터의 접근을 허용한다. 이들 API는 Broadcasting File Systems, IP Datagrams, 스트리밍 데이터의 세가지 형식으로 전송된 데이터의 접근을 지원한다. 본 논문에서 설계하고자 하는 RTP 응용 클래스 구조를 구현하기 위해서는 스트리밍 데이터를 접근할 수 있어야 한다. JavaTV API에서는 JMF Package의 javax.media.protocol을 이용하여 스트리밍 데이터의 접근을 제공하며, 비동기적인 스트리밍 데이터는 javax.tv.media.protocol.PushSourceStream2 인터페이스를 이용하여 포함될 수 있다. JavaTV API 애플리케이션은 일반적으로 javax.tv.locator.Locator Object를 사용하여 개별적인 Data Service Component를 참조한다. Locator.toExternalForm() Method를 사용하여 Locator Object를 javax.media.MediaLocator Object가 생성되어지는 String으로 전환시킨다.

MediaLocator는 결과적으로 javax.media.Manager로부터 javax.media.DataSource Object를 포함하는데 사용되며 Object 생성후 DataSource object는 하나 이상의 PushSourceStream2 object를 포함하는데 사용된다.

PushSourceStream2는 JMF version 1.0의 javax.media.protocol.PushSourceStream을 새로운 read 메커니즘과 함께 확장시킨 인터페이스이며 Java TV API는 PushSourceStream2를 통해서 포함된 데이터의 버퍼링 및 사용 가능성을 보증하지 않으며 PushSourceStream2.readStream() Method는 데이터의 payload의 접근과 데이터 손실을 알리는 exception을 제공하고 있다.^{[5][15]}

3. 프로토콜의 적용

RTP를 사용하여 미디어를 전송하고 수신하는 방법은 각각 2가지 방법이 있다. 서버 측은 MediaLocator와 DataSink를 이용한 방법과 SessionManager를 직접 사

```
String url = "rtp://10.10.10.5:42050/audio/1"; // RTP URI
MediaLocator mrl = new MediaLocator(url);

if(mrl == null) {
    System.out.println("Can't Found RTP Session");
    return;
}

try{
    player = Manager.createPlayer(mrl);
}catch (NoPlayerException e) {
    ...
    return;
}catch (MalformedURLException e) {
    ...
    return;
}catch (IOException e) {
    ...
    return;
}

player.realize();
```

그림 7. MediaLocator를 통한 Player 생성 부분
Fig. 7. Creating of the player using MediaLocator.

```
public void update( ReceiveStreamEvent event) {

    SessionManager source = (SessionManager)event.getSource();
    System.out.println(event.toString());

    if (event instanceof NewReceiveStreamEvent) {

        String cname = "Online Broadcast";
        ReceiveStream stream = null;

        try {
            stream = ((NewReceiveStreamEvent)event)
                .getReceiveStream();
            Participant part = stream.getParticipant();

            if(part != null)
                cname = part.getCNAME();

            // get a handle over the ReceiveStream datasource
            DataSource dsource = stream.getDataSource();
            this.startMedia(dsource);

        } catch (Exception e) {

            System.err.println("NewReceiveStreamEvent exception "
                + e.getMessage());
            return;

        }

    }
}
```

그림 8. SessionManager를 이용한 RTP 제어 부분
Fig. 8. Part of RTP control using SessionManager.

용해서 전송하는 방법이 있으며, 클라이언트 측은 서버와 비슷하게 MediaLocator를 이용하거나, Session-Manager를 직접 사용하는 방법으로 구분할 수 있다. 수신측에서 MediaLocator를 이용한 방법은 RTP세션을 생성하기 위해 MediaLocator를 사용하며 재생하고자 하는 스트림의 URI를 MediaLocator로 생성한 후 RTP 세션에서 미디어 스트림을 발견했을 때 Player Object를 넘겨주게 된다.

두 번째 방법에서 SessionManager는 세션에 연결된 클라이언트들의 정보와 RTCP 제어 및 전송 스트림을 관리하는 클래스이다.

클라이언트들의 정보를 Participant의 Object형태로 갖고 있으며, CNAME(canonical name) SDESS(source description)을 포함하는 새로운 RTCP Packet이 도착할 때 마다 새로 생성된다. 또한 전송되어지는 스트림을 ReceiveStream과 SendStream이라는 RTP Stream은 상속받는 두 가지의 스트림을 통해 관리된다. ReceiveStream은 SessionManager가 새로운 RTP 데이터를 찾아낼 때 마다 자동적으로 생성되며, 이와 반대로 SendStream은 직접 생성하도록 되어 있다.

이와 같은 방법으로 ReceiveStream의 변화를 알아내는 ReceiveStreamListener를 이용해 DataSource를 얻을 수 있게 된다. 두 가지 스트리밍 데이터 이용방법 중 본 논문에서는 JavaTV API에서 DataSource를 접근하는 MediaLocator 형태를 이용한 구조를 선택하였다.

4. 클라이언트 애플리케이션 구조

현재 상용화된 OCAP 표준의 셋톱박스에는 rtp Package가 구현되어 있지 않다. 때문에 UCC 서비스를 위한 애플리케이션은 rtp를 위한 클래스를 적재하여 Loading되어야 한다. 따라서 본 논문에서 제시하고 자 하는 UCC 서비스를 위한 클라이언트 애플리케이션의 구조는 다음과 같다.

현재 구현되어 있지 않은 rtp 클래스의 Session, Control, Management부분과 ReceiveStream에 대한 모니터링을 위한 Listener, 그리고 클라이언트에서 발생 가능한 Event로 구성된 구조를 가진다.

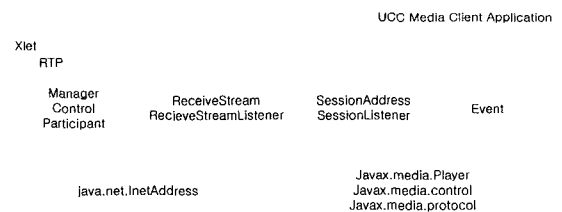


그림 9. 클라이언트 애플리케이션 구조
Fig. 9. Proposed the structure of Client application.

5. 서버 시스템 구조

셋톱박스의 클라이언트 애플리케이션으로 동영상 스트리밍 서비스를 위한 서버 측의 시스템 구조는 서론에서 기술한 것과 같이 웹으로 접근 및 제어 가능한 확장

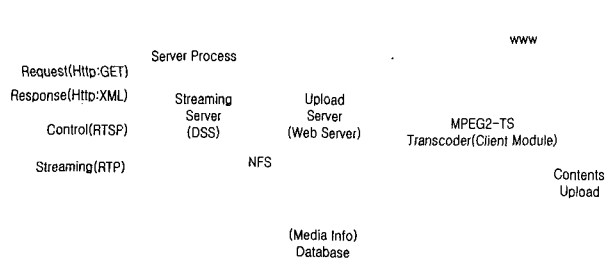


그림 10. 서버 시스템 프로세스
Fig. 10. Process of the server system.

시스템으로 해당 콘텐츠 관리를 통해 방송 송출이 가능한 시스템으로 구성되어 있다. 웹에서 콘텐츠 제작자가 Client 모듈을 통해 트랜스코딩으로 MPEG2-TS 파일로 변환되어 Upload Server(Web Server)로 데이터베이스 연동으로 등록되게 된다.

셋톱박스의 애플리케이션은 HTTP를 통해 Request 정보를 전송하고 콘텐츠 정보를 XML로 반환하게 된다. 클라이언트는 XML를 파싱하고 콘텐츠를 선택할 수 있는 인터페이스를 제공하게 되며, 선택 후 비디오 컨트롤 메시지(RTSP)를 전송하면 RTP를 통해 스트리밍이 전송되기 시작하는 구조를 가진다.

이후부터는 RTSP 메시지에 따라서 Tear-Down (Stop), Pause, Play등의 프로세스가 동작하게 된다.

V. 실험

본 논문에서는 rtp Package를 참조한 클래스의 Import와 RTP 데이터를 통한 전송 성공여부를 판단하는 시뮬레이션을 하였으며 스트리밍 서버는 FreeBSD기반으로 구성하였고 커널은 6.1기준으로 컴파일하였다. 스트리밍 서버 애플리케이션은 Darwin Streaming Server를 구현하여 전송하도록 하였으며 웹을 통한

```

This is free software, and you are welcome to redistribute
it under certain conditions;
see licence document for details.
*****
setting properties...
free/used/total: 983 K / 2800 K / 3704 K
running gc...
after gc...
free/used/total: 2189 K / 2418 K / 4600 K
[XletView]-INFO->loading Xlet... [MainXlet]
[XletView]-INFO->XLET started... [MainXlet]
Opening RTP session for: addr: 163.239.125.120 port: 42050 ttl: 1
...Waiting for RTP data to arrive...
ReceiveStream Listener - Received new RTP stream: JPEG/RTP
The sender of this stream had yet to be identified.
Creating player...
adding listener...
Starting player...
Session Listener - A new participant has just joined: nails99@dbc1.99
ReceiveStream Listener - The previously unidentified stream
JPEG/RTP
had now been identified as sent by: nails99@dbc1.99
Session Listener - A new participant has just joined: nails99@dbc1.99

```

그림 11. 클라이언트 실행 로그
Fig. 11. Execution log of Client.

```

own.Handler@1c78e57, previous=Realized, current=Prefetching, target=Prefetched]
ControllerEvent: javax.media.PrefetchCompleteEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Prefetching, current=Prefetched, target=Prefetched]
ControllerEvent: javax.media.StartEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Prefetched, current=Started, target=Started, mediaTime=java.x.media.Time@1c1ea29, timeBaseTime=java.x.media.Time@1f436f5]
ControllerEvent: javax.media.EndOfMediaEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Started, current=Prefetched, target=Prefetched, mediaTime=java.x.media.Time@9789144]
ControllerEvent: javax.media.MediaTimeSetEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, mediaTime=java.x.media.Time@1893afe]
ControllerEvent: javax.media.StartEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Prefetched, current=Started, target=Started, mediaTime=java.x.media.Time@186c6b2, timeBaseTime=java.x.media.Time@15ec671]
ControllerEvent: javax.media.EndOfMediaEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Started, current=Prefetched, target=Prefetched, mediaTime=java.x.media.Time@82701e]
ControllerEvent: javax.media.MediaTimeSetEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, mediaTime=java.x.media.Time@18ee9d6]
ControllerEvent: javax.media.StartEvent[source=com.sun.media.processor.unknown.Handler@1c78e57, previous=Prefetched, current=Started, target=Started, mediaTime=java.x.media.Time@19a9c7c, timeBaseTime=java.x.media.Time@a9ae05]

```

그림 12. 서버 실행 로그
Fig. 12. Execution log of Server.

UCC의 업로드 인터페이스는 httpd기반의 Tomcat 모듈을 이용하여 구성하였고, 실제 업로드 파일 변환을 위한 트랜스코딩 모듈은 이번 실험에서 사용하지 않고 트랜스포트 가능한 동영상 파일을 직접 업로드하고 원활한 스트리밍 전송환경을 위해 NFS(Network File System)를 통한 네트워크 접근 방식을 구성하였다.

실험은 스트리밍 전송 시뮬레이션으로 진행되었다.

셋톱박스 상에서 De-packetizing 및 Decoding Pipeline의 생성은 셋톱박스 상에서 다운로드 가능한 구조의 클래스 Loading을 위한 구조 설계가 완성되지 않아 이번 실험에서는 테스트 하지 못하였으며 데스크탑 상에서 Open-Source Emulator인 XletView를 이용하여 실험하였다.

IV. 결론

본 논문에서는 미디어 재생을 위한 Decoding Pipeline 구성을 위해 DataSource를 접근하는 방법으로 RTP의 MediaLocator 형태를 이용한 구조로 RTP 스트리밍 데이터 접근에 대한 애플리케이션에 대해 정의하고 시뮬레이션 하였다. 그리고 스트리밍 서버 및 사용자 콘텐츠 등록을 위한 서비스 시스템 구조에 대해서도 설계 하였다. 실험을 통해 데이터 전송 및 접근에 대해서는 결과를 확인 할 수 있었지만, 현재 OCAP 표준으로 구현된 미들웨어에서는 Decoding Pipeline의 생성과 Render에 대한 제어부분이 구현되지 않아 확인 할 수 없었다. 향후 애플리케이션 구조의 추가 구현 및 실험을 통해 RTP 스트림의 제어가 가능한 MediaLocator 어댑터의 구현과 DataSource 접근의 다른 방법 중에 하나인 SessionManager의 재정의를 통한 RTP Manager 클래스의 구현이 진행 중이다.

디지털 케이블 방송 환경에서 RTP를 응용하여 라이브 방송 시스템에 의존성이 적은 가벼운 VOD 서비스가 가능하다면 추가적인 인프라에 대한 고려 없이 UCC와 같은 콘텐츠뿐만 아니라 다양한 주문형 동영상 서비스의 도입이 가능할 것이다.

참 고 문 헌

- [1] 장동익, "케이블TV 기반 VOD서비스의 구현방안", 방송공학회지 제9권 제3호, p68-p77, 2004년
- [2] 홍효진, "컨버전스 시대의 디지털 콘텐츠 시장", NCA ISSUE REPORT 제 12호, 2006년
- [3] Colin Perkins, "RTP: Audio and Video for the Internet", Addison Wesley, June 12, 2003.
- [4] Steven Morris, "Interactive TV Standards", Focal Press, p257-p294, 2005.
- [5] Sun Microsystems, inc., "JavaTV API Technical Overview", Nov 2000.

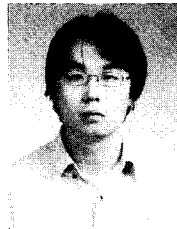
- [6] Sun Microsystems, "Java Media Framework API Guide" JMF FCS, Nov 19, 1999.
- [7] Sun Microsystems, "Personal Java API 1.2 Specification", 1999.
- [8] ISO/IEC 13818-6 Generic Coding of Moving Picture and Associated Audio : Digital Storage Media Command and Control, 1996.
- [9] Audio/Video Transport Working Group. Internet Engineering Task Force, "RTP : A Transport Protocol for Real-Time Applications", 2001.
- [10] ANSI/SCTE 90-2. "SCTE Application Platform Standard OCAP 2.0 Profile", 2005.
- [11] Richard Chernock, "Data Broadcasting", McGraw Hill, 2001.
- [12] Edward M. Schwalb, "iTV Handbook", Prentice Hall, 2004.
- [13] Interactive TV web, <http://interactivetvweb.org>
- [14] <http://eeca2.sogang.ac.kr/research/vod/vod.asp>
- [15] JAVA 멀티미디어 JMF, <http://www.jstorm.pe.kr/>
- [16] <http://www.cs.columbia.edu/~coms6181/>

저 자 소 개



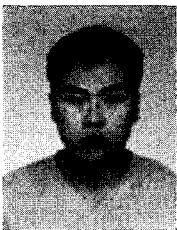
김 성 원(학생회원)
 2000년 대구대학교 경영학과 학사 졸업.
 2006년~현재 서강대학교 영상대학원 미디어공학과 석사 과정.

<주관심분야 : 미들웨어, IPTV, 양방향 방송 응용 프로토콜, Feature Detection>



시 장 현(학생회원)
 2006년 전북대학교 과학기술학부 학사 졸업.
 2006년~현재 서강대학교 영상대학원 미디어공학과 석사 과정.

<주관심분야 : 양방향 디지털 방송 콘텐츠, IPTV>



김 정 환(학생회원)
 2004년 아주대학교 미디어학부 학사 졸업.
 2006년 서강대학교 영상대학원 미디어공학과 석사 졸업.
 2006년~현재 동 대학교 영상대학원 미디어공학과 박사 과정.

<주관심분야 : iTV, IPTV, MPEG-4, 3DTV, 홀로그램>



정 문 열(정회원)
 1980년 서울대학교 계산통계학과 학사 졸업.
 1982년 한국과학기술원 전산학과 석사 졸업.
 1992년 Univ. of Pennsylvania 전산학과 박사 졸업

1992년~1994년 일본 구주공업대학 조교수
 1994년~1999년 숭실대학교 컴퓨터학부 부교수
 1999년~현재 서강대학교 영상대학원 미디어공학과 정교수

<주관심분야 : 인터랙티브 방송, 컴퓨터 그래픽스>