□특집□

차세대 웹 상에서의 멀티미디어

김 두 현† 김 지 용†† 황 승 구†††

◈ 목 차 ◈

1. 서 론

지난 몇 년 동안 웹과 관련한 기술의 성장과 활용의 증대는 가히 폭발적으로 이루어져 왔고 이는 인터넷의 보급을 촉발하여 전세계 인터넷인구가 2002년에는 2억명, 2005년에는 3억 5천만명에 이를것으로 추정되고 있다. 이러한 현상의 배경에는 정보를 소유하려는 인간의 원초적 욕구와 합치되는 것과 무관하지 않지만 기술적으로 볼 때 웹 하부 구조가 갖는 단순성과 개방성을 들 수 있다. 인터넷의 어느 곳의 어느 정보든 접근할 수 있는 표준화된 URL과 사용자가 쉽게 숙달하고 이용할 수 있는 HTML 표현 그리고 이들 내용을 전송할 수 있는 HTTP 표준 프로토콜의 3가지 요소의 개방형 특성이 사용자 및 개발자의 확대를 유도하였다.

이러한 구조는 기본적으로 단순한 텍스트 또는 그래픽을 포함한 텍스트 만을 다루도록 설계되어 왔다. 그러나 오늘날 많은 일반 사용자나 응용 서비스 개발자 및 콘텐CM 제공자들은 텍스트 뿐만 아니라 오디오 및 동영상 데이터를 전송하거나 받을 수 있는 서비스를 요구한다. 이러한 요구는 인터넷에서 멀티미디어를 전송하는데 필요한 여러 통신 프로토콜의 개발을 촉진하였고, 대표적인 표준으로 IETF에서 제공한 RTP(Realtime Transport Protocol, RTP Control Protocol(RTCP)[2, 3], 및 RTSP[4]와 World Wide Web Consortium (W3C)의 SMIL (Synchronized Multimedia Integration Language)[1]을 들 수 있다.

한편 이러한 개방형 추세에 맞추어 시스템 하부구조의 다양성을 감추면서 상위 구조로 하여금 시스템 구조에 가능한 무관한 API를 제공함으로서 상위 프로그램의 호환성과 이식성을 최대화할 수 있도록 하는 멀티미디어 미들웨어의 필요성이 갈수록 증대되어 가고 있다. 멀티미디어의 경우에는 특히 수 없이 많은 종류의 입출력 장치이나 코덱 포멧이 존재하고 있는 현실에서 멀티미디어용 미들웨어야 말로 다양한 응용 프로그램의 활성화에 핵심이라 할 수 있다. 특히 컨텐츠의 호환성이 생명이라할 수 있는 웹 환경에서 멀티미디어 컨텐츠를 제공하기 위해서는 이러한 미들웨어 개념은 반드시 필요한 것이다.

본 논문에서는 웹을 위한 멀티미디어 핵심 기술로서 SMIL 및 RTSP를 소개하고, 미들웨어로서 실제로 웹을 통하여 플랫폼에 무관한 멀티미디어

† 정회원 : 한국전자통신연구원 멀티미디어연구부 멀티미디어그룹웨어팀 팀장
†† 정회원 : 한국전자통신연구원 멀티미디어연구부 멀티미디어그룹웨어팀 연구원
††† 정회원 : 한국전자통신연구원 멀티미디어연구부 부장

컨텐츠 서비스를 실현해 나가고 있으며 기술적으로나 상업적으로나 경쟁적인 관계에 놓여있는 마이크로 소프트사의 DirectShow[6, 7]와 썬 마이크로시스템사의 JMF(Java Media Framework)[8, 9]의 개념을 소개한다.

2. SMIL[1]

SMIL[1]은 W3C가 인터넷에서 TV와 같은 품질의 컨텐트를 전송할 수 있도록 하기 위한 목적으로 승인한 일종의 마크업 언어로서 지난 1998년 6월에 버전 1.0이 제정되었다. SMIL의 특징은 기존의 HTML 등 하이퍼텍스트를 위한 마크업 언어에서 도외시 되었던 멀티미디어 데이터 간의 시간적인 동기 관계를 명시할 수 있도록 함과 아울러 시간 축에서 하이퍼링크도 지정할 수 있도록 한다는 점이다. SMIL에서는 시간적인 동기 관계의 명시함에 있어서 두가지의 주요 태그인 <seq>와 <par>을 이용하여 모든 동기 관계를 명시하도록 함으로서 배우기에 쉽고 사용도 편리하도록 하고 있다. 또한 CWI, RealNetwork, NIST 등에서 SMIL 편집기와 플래이어 등을 참조용으로 또는 베타버전으로 제공하고 있다. SMIL의 개념을 소개하는 것은 비교적 간단한 일이지만 문법 차원까지 자세히 설명하는 것은 본 논문의 범주를 벗어나므로 약하기로 하고, 본 장에서는 SMIL 규격서에 나와있는 예제를 인용하여 SMIL의 동기관계 명시 기능을 <seq>와 <par>를 중심으로 소개하도록 한다.

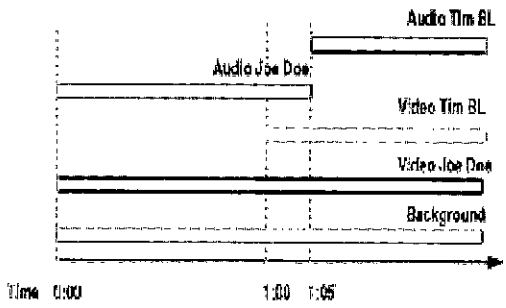
<예제 1>에서 보는 바와 같이 SMIL 문서의 기본 골격은 HTML과 매우 유사하다. 즉 모든 SMIL 문서는 <smil> 태그로 시작하여 </smil>로 끝나고 그 내부는 <head>부분과 <body>부분으로 구성된다. 우선 헤더부분에서는 SMIL 문서의 메타 정보를 제공하는데 여기에는 저자이름, 제목, 저작권 등을 명시할 수 있을 뿐아니라 이후 바디에서 레이아웃으로 사용될 채널을 정의할 수 있다. <예제 1>은 웹의 성장에 관한 뉴스를 SMIL로 마크업한 것으로 네개의 채널이 정의되어있다.

```
<smil>
  <head>
    <layout type="text/smil-basic">
      <channel id="left-video" left="20" top="50/>
      <channel id="left-text" left="20" top="120" />
      <channel id="right-video" left="150" top="50" />
      <channel id="right-text" left="150" top="120" />
    </layout>
  </head>
  <body>
    <par>
      <img    src="bg"/>
      <seq>
        <par>
          <img  src="graph" channel="left-video" dur="60s"/>
          <text src="graph-text" channel="left-text"/>
        </par>
        <par>
          <a href="http://www.w3.org/People/Berners-Lee">
            <video src="tim-video" channel="left-video"/>
            <text src="tim-text" channel="left-text"/>
          </a>
        </par>
      </seq>
      <seq>
        <audio src="joe-audio"/>
        <audio src="tim-audio"/>
      </seq>
      <video id="jv"
                src="rtsp://www.w3.org/video/joe-video.mpg"
                channel="right-video"/>
      <text  src=" http://www.w3.org/video/joe-text.txt"
                channel="right-text"/>
    </par>
  </body>
</smil>
```

(예제 1) SMIL 문서 예

바디 부분은 각종의 미디어 리소스와 이들의 동기관계를 명시한 것으로 <그림 1>과 같이 도식화 할 수 있다. 제일 먼저 병렬 태그인 <par>가 있고 </par>와의 사이에 5개의 요소, 즉 <img>, <seq>, <seq>, <video>, <text>가 있다. 이들 5개 요소는 <par>의 특성에 의하여 동시에 시작된다. 즉, 플래이가 시작되면 백그라운드 이미지가 출력되면서(src="bg"), 화면의 왼쪽에 웹의 성장에

관한 이미지(src="graph")가 이미 헤더에서 정의된 left-video 채널에 60초간 출력됨과 아울러 그바로 밑에 graph-text에 저장된 텍스트가 left-text 채널에 출력된다. 한편 이와 동시에 //www.w3.org/video/joe-video.mpg에 있는 Joe의 비디오가 RTSP 프로토콜을 통하여 스트리밍되면서 right-video 채널에 출력되면서 바로 밑에 //www.w3.org/video/joe-text.txt에 있는 텍스트가 http 프로토콜을 통하여 전달되어 right-text 채널에 출력된다. 또한 동시에 Joe의 음성이 시작된다(src="joe-audio"). 여기서 joe-audio의 러닝타임이 65초라 가정한다면 65초 후엔 순차 태그인 <seq>문의 특성에 의하여 tim-audio가 시작된다(src="tim-audio").



(그림 1) 예제 1의 동기 관계 도식

또한 두번째의 <seq> </seq> 쌍 내에는 다시 두개의 <par> </par> 쌍이 있는데 이들은 순차 태그인 <seq>의 특성에 따라 순차적으로 수행된다. 즉, 첫번째 <par> </par> 쌍 내의 <img>의 출력 기간이 dur="60s"로서 60초로 지정되었으므로 두번째 <par> </par> 쌍은 시작후 60초가 지난 후에 작동하게되어 left-video 채널에는 tim-video가 그리고 left-text 채널에는 tim-text가 출력되기 시작한다. 한편 두번째 <par> </par> 쌍 내의 <a>와 태그는 HTML의 하이퍼링크와 유사한 개념으로 60초 후에 tim-video와 tim-text가 출력되기 시간한 후에는 사용자가 언제든지 left-video 채널이나 left-text 채널에 해당하는 영역을 클릭하면 HTML 브라우저가 팝업되면서 Tim의 홈페이지인 //www.w3.org/People/Berners-Lee가 나타나게 된다.

SMIL에는 이상에서 설명한 <par>, <seq>, <a> 태그와 아울러 통신환경이나 사용자 컴퓨터의 QoS에 따라 선택이 가능한 <switch> 태그와 시간축 상에서 비디오 상으로부터 하이퍼링크가 가능한 <anchor> 태그가 제공된다. 특히 <a> 태그가 해당 미디어 리소스 전체에 대하여 시작시점부터 끝시점까지 링크가 가능하도록 고정된 것에 반하여 <anchor> 태그는 정의된 기간동안 화면 어느 곳, 특히 플래이되고 있는 비디오 위에도 하이퍼링크 영역을 지정할 수 있다.

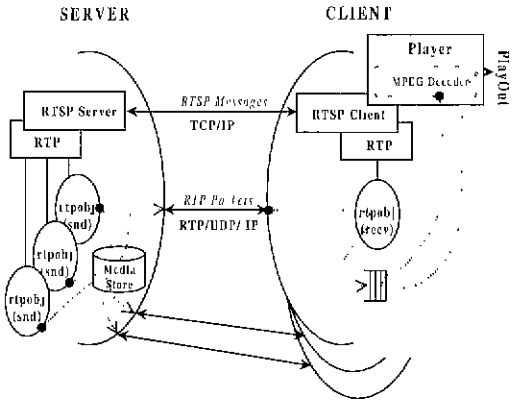
3. 스트리밍 프로토콜

RTSP[4]는 HTTP의 개발과 같은 맥락을 갖으며, 따라서 표준의 많은 부분이 동일하다. 그러나 HTTP는 TCP를 이용하여 시간적 제약 없이 완전한 신뢰성 전송을 요구하므로 시간 도메인에서 수행되어야 하는 멀티미디어 프리젠테이션에는 적합하지 않다. 따라서 HTTP는 웹 페이지를 전송하는데 적합하며, 프리젠테이션을 위한 실시간 멀티미디어 데이터의 전송 제어를 위해서 RTSP가 권고되고 있다. 한편, 실시간 응용 서비스를 위한 시간 재조정, 패킷 손실 검증, 보안 및 전송 데이터 유형 식별 등 실시간 특성을 갖도록 제공하는 RTP 프로토콜은 RTSP와 연동하여 서버의 프리젠테이션용 멀티미디어 파일을 클라이언트에 단 방향 전송을 담당한다. 따라서 서버의 멀티미디어 데이터를 클라이언트 사이트에 나타내기 위해 스트림 엔진으로 RTSP와 RTP 프로토콜의 구현이 요구된다.

3.1 RTSP[4, 5]

RTSP는 클라이언트-서버형 멀티미디어 프리젠테

이션 제어 프로토콜이며, IP 통신 위에서 스트림 형태의 미디어 데이터를 효과적으로 전송하기 위한 제어 기능을 정의한다. 1998 IETF MMUSIC working group에 의해 RFC2326으로 표준화되었고 현재에도 새로운 연구와 표준화 작업이 계속되고 있다.



(그림 2) RTSP 서버 클라이언트 구조

RTSP 처리는 기능적으로 클라이언트와 서버의 구조가 다르다. 멀티미디어 데이터를 저장하고 있는 서버는 여러 클라이언트로부터 데이터 전송을 요청 받을 수 있으며, 동일한 미디어 파일을 비동기적으로 전송할 수 있어야 한다. 여기서 저장되어 있는 데이터는 콘텐츠 제공자로부터 받은 MPEG과 같은 프리젠테이션 파일을 의미한다.

<그림 2>는 클라이언트-서버의 구조와 그들 간의 상호 작동 예를 나타낸다[5]. 전송 제어를 담당하는 양측의 RTSP 프로토콜은 표준에서 정의된 RTSP 메시지들을 이용하여 여러 제어 정보를 송수신한다. 이때 이용되는 인터넷 프로토콜은 TCP 또는 UDP 모두를 이용할 수 있으나, 주로 제어 정보의 신뢰성을 고려하여 TCP를 이용한다. 따라서 응용 서비스(웹)로부터 멀티미디어 전송을 요청 받고 RTSP 메시지들이 전송되기 전에 이미 연결 설정이 이루어진다. 이를 위해 서버측 RTSP는 listen 모드에서 연결을 기다린다. RTSP와 연동하는 RTP프로토콜은 연결되는 클라이언트 마다 각 RTP 오브젝트가 할당되고, 오브젝트 내의 쓰레드는 독립적으로 각 클라이언트와 UDP를 이용하여 Media Store의 요청된 파일을 RTP 패킷으로 전송한다.

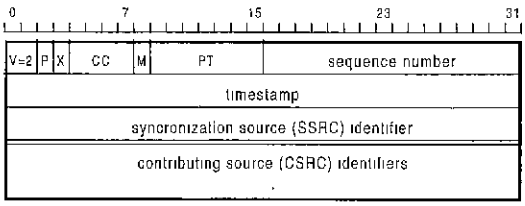
클라이언트 측의 RTP 오브젝트는 지정된 포트로부터 전송된 RTP 패킷을 수신하여 Player의 디코더를 통해서 재생된다. 이때, RTP 오브젝트에 수신된 패킷은 12 바이트의 헤더와 미디어 데이터로 구성되고, 이들 패킷은 RTP 고유의 기능인 패킷손실 검증, 지연 변이 및 패킷 순서 재조정 등의 QoS에 관련 매커니즘을 처리한 후 Player가 인식할 수 있는 버퍼에 저장한다.

클라이언트-서버 간의 프로시저[5]는 이벤트 처리기의 상태 관리 정보를 이용하여 수행되며 각 세션 마다 독립적으로 처리된다. 아래에서 양 시스템 간의 RTSP 메시지 전송 절차를 제시한다. 메소드 처리 순서는 4단계로 구분할 수 있다. 먼저 RTSP 메시지 전송을 위해 클라이언트-서버 사이를 연결한다. 클라이언트-서버 간의 연결 설정이 이루어 지면, OPTIONS 메소드가 양측에서 초기화되고, 미디어 디스크립터를 전송하기 위해서 DESCRIBE 메소드를 이용한다. 디스크립터 메시지는 요청된 URL에 정의된 프리젠테이션 오브젝트의 속성을 클라이언트에 전송한다. 이때, 클라이언트는 서버가 인지할 수 있는 디스크립터를 제시하는데 서버는 미디어의 속성을 나타내는 포맷을 제시하고 제시된 포맷에 따라 속성 들의 요소를 기술 한다. 세 번째 단계로 시스템은 READY_STATE로 천이되어 SEPUP 메소드를 처리한다. SETUP 메시지를 받은 서버는 요청에 따라 전송 모드(unicast, multicast) 및 전송 포트를 설정한다. 따라서 양 시스템은 READY_STATE에서 RTP 오브젝트를 생성한다, 마지막으로 서버가 PLAY 메소드를 받으면, RTP 오브젝트가 구동 되어 DESCRIBTE 메소드에서 지정한 파일을 패킷화

하여 클라이언트에 전송한다. 프리젠테이션 미디어의 전송이 완료되면 READY_STATE로 천이 되고 PAUSE, PLAY 또는 STOP 등의 명령을 기다리고 있다가 사용자에 의하여 요구가 발생하면 처리한다.

3.2 RTP[2,3]

RTSP에 의하여 스트림 전송을 위한 세션이 설정되면 멀티미디어 데이터는 RTP를 통하여 패킷화되어 클라이언트 측으로 전송된다. RTP는 오디오와 비디오 간의 동기를 맞추고 실시간성을 유지하는 데에 사용되는 정보들을 포함하는 패킷 헤더라 할 수 있다. RTP 헤더 형식은 <그림 3>와 같으며, 각 필드의 의미는 다음과 같다.



(그림 3) RTP 헤더 형식

① 버전(V) : RTP 버전을 나타내는 필드로 현재 버전 2이다.

② 패딩(P) : 부가적인 패딩 옥텟을 나타내는 필드로, 주로 암호화를 위해 사용하는 것으로, 1-비트 0으로 정의한다.

③ 확장(X) : RTP 헤더 확장을 나타내는 필드로, 헤더 확장을 지원하지 않을 경우, 1-비트 0으로 정의한다.

④ CSRC 수(CC) : CSRC의 수를 기술하는 필드로, RTP Mixer를 지원하지 않을 경우 4-비트 0으로 정의한다.

⑤ 마커(M) : 주로 비디오 프레임 간의 경계를 나타내기 위한 것으로, H.261을 사용할 경우 코덱 보드에서 시스템으로 전달되는 데이터 스트림은 프레임 간의 경계를 나타내지 않기 때문에 마커 필드를 나타낼 방법이 없다. 따라서 1-비트 0으로 정의한다.

⑥ 탑재 유형(PT) : 다음 표와 같이 오디오/비디오 각각의 탑재 유형을 7-비트로 정의한다. 동일 RTP 세션에서 여러 탑재 유형을 허용하지 않는다.

⑦ 순서 번호(sequence number) : 보내는 패킷마다 1씩 증가하여 순서 번호를 지정함으로써 수신 측에서 패킷의 순서를 확인하여 패킷 손실을 알 수 있다. 16-비트 필드로 범위는 [1.. 65536]의 순환 형태로 기술한다.

⑧ 타임스템프(timestamp) : 수신 측에서 매체 자체 또는 매체 간의 동기화를 위해 또는 지터 계산을 위해 필요한 시간 필드로 msec 단위로 지원되며, 따라서 msec의 동기화 정확도를 갖는다. 여기에서 타임스템프의 시점은 샘플링 인스턴스를 의미하나, 코덱에서 샘플링 시간을 시스템으로 알려주지 못하는 경우가 많으므로, 단지 오디오/비디오 버퍼에 전달된 시간을 타임스템프로 이용한다.

⑨ SSRC 식별자 : 같은 세션 내의 동기화 소스를 유일하게 식별하기 위한 구분자로, RFC1321의 MD5 루틴을 이용하여 32-비트의 난수 식별자를 생성한다. 또한 식별자 충돌의 발생을 해결할 수 있는 메커니즘을 포함한다.

⑩ CSRC 식별자 : RTP Mixer를 지원할 경우 최대 15개까지의 SSRC 식별값을 기록한다.

RTP를 이용한 멀티미디어 데이터 전송은 스트리밍 뿐만아니라 인터넷 영상회의의 ITU-T 국제 표준인 H.323[10]에서 오디오 및 비디오 정보의 멀티캐스팅을 위하여도 사용되고 있다. 따라서 RTP는 연속성이 생명인 멀티미디어 정보의 실시간 전송을 위한 표준 프로토콜로 자리잡아가고 있으며 이에

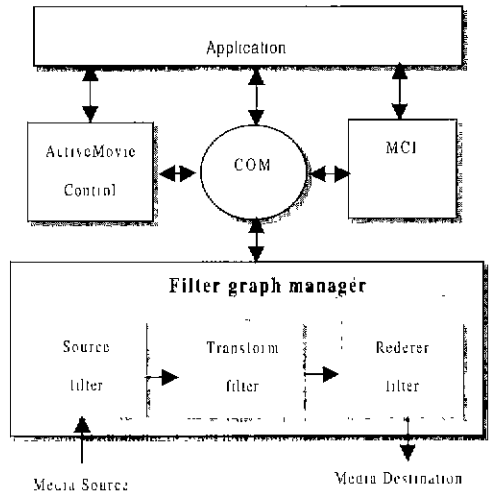
관련된 연구도 학계 및 업체에서 많은 연구가 되고 있다. 특히 오디오 비디오 스트림 간의 동기화, 패킷 손실의 보상, 지연 및 지터 예측 및 개선 등을 위한 연구가 응용 프로그램 개발자에서부터 인터넷 통신 프로토콜 및 장비 개발자에 이르기까지 다양한 분야에서 이루어지고 있다.

4. 미들웨어

4.1 Microsoft DirectShow[6]

DirectShow[6]는 DirectX Media SDK의 한 부분이다. DirectShow SDK는 몇 개의 다른 마이크로소프트 서비스를 기반으로 하고 있으며, 특히 DirectX 기본구성요소(Foundation)인 DirectDraw와 DirectSound API를 기반으로 하고 있다. DirectShow는 로컬 파일 또는 인터넷서비스를 통해 멀티미디어 스트림을 재생하거나 디바이스로부터 멀티미디어 스트림을 캡처하는 등의 기능을 제공하며, 특히, 여러 포맷으로 압축되어 있는 (예를 들자면, MPEG, Quicktime, AVI등을 포함한) 비디오나 오디오의 데이터를 일관성 있게 재생할 수 있도록 한다. 아울러 인터넷을 통하여 멀티미디어 데이터를 스트리밍하는 기능도 기본적으로 제공하고 있어, 현재 마이크로소프트사가 보급중에 있는 ActiveMovie의 핵심 플래이어인 Window Media Player도 이를 기반으로 만들어져 있다. DirectX 구성 요소들은 모두 COM[7] 인터페이스를 사용하고 있는데, 이 COM은 마이크로소프트에서 제시한 바이너리 표준 인터페이스 사양으로써 뛰어난 확장성을 제공하는 기반이 되고 있다. 따라서 웹을 이용한 멀티미디어 서비스의 호환성에 있어서도 이러한 COM의 강점으로인하여 미들웨어로서의 입지를 다지고 있다.

COM에 대한 자세한 설명은 본 논문의 범위를 넘어가므로 여기서 서술하지는 않지만, DirectShow를 포함한 DirectX를 심도있게 사용하려면 COM에 대한 지식이 필요하다. 그러나, 예를 들어, 개발자가 C나 C++를 사용하여 DirectShow의 필터를 생성하고 연결하여 응용 프로그램을 만들거나 자신만의 필터를 만들기 위해서는 COM 프로그래밍을 잘 이해하여야 하지만, Window Media Player의 컨트롤을 활용하는 정도의 응용 프로그램 개발을 위해서는 COM을 이해하지 않아도 될 정도로 간단하고 직관적인 인터페이스 함수들을 제공하고있어 웹과 멀티미디어 서비스를 연동하는데에 편리하다. 특히 마이크로소프트의 웹 브라우저인 Internet Explorer에서는 COM 객체를 임베드하여 사용할 수 있도록 하는 <object> 태그를 제공하고 있고 이를 이용하면 HTML에서 Window Media Player 컨트롤을 직접 임베드하여 사용할 수 있기 때문에 편리하게 웹 기반의 멀티미디어 서비스를 개발할 수 있다.



(그림 4) COM, MCI, 필터 그래프 매니저

4.1.1 필터

DirectShow는 <그림 4>에 나타난 바와 같이 기존의 Video for Windows와 MCI 인터페이스를 대체하는 API이다. DirectShow의 핵심서비스는 필터라고 불리는 플러깅(plugging) 될 수 있는 컴포넌

트이다. 필터는 필터 그래프에 의하여 정렬되어 있으며, 필터그래프는 필터그래프 매니저라고 불리는 컴포넌트에 의하여 필터들의 연결이 관장되고 데이터 플로우 스트림이 제어된다. 따라서 응용 프로그램들은 필터그래프 매니저를 통해 필터 그래프의 동작을 제어하여야 하는데 이를 위하여 간접적으로는 마이크로소프트 윈도우의 미디어 플레이어 컨트롤을 사용하거나 직접적으로는 COM 인터페이스 메소드를 사용해서 할 수도 있을 것이다. 한편 DirectShow의 SDK는 DirectShow의 기본 클래스 라이브러리를 사용해서 개발자 자신에게 맞는 새로운 필터를 만들 수 있도록 하고 있으며 이를 위하여 기본 클래스 내에 필요한 COM 인터페이스를 기본적으로 제공함으로서 모든 필터의 플레임웍으로 사용되고 있다.

4.1.2 필터 그래프 매니저와 필터 그래프

<그림 4>와 같이 필터 그래프 매니저는 필터 그래프의 동작을 제어하는 역할을 한다. 필터 그래프는 연결된 필터들의 집합이며, 필터에는 소스 필터, 트랜스폼 필터, 랜더러 필터 등 세가지 타입이 있다. 이중 소스 필터는 파일이나 디스크 또는 인터넷 서버 등의 소스로부터 데이터를 읽어들이는 역할을 하고, 트랜스폼 필터는 소스 필터로부터 데이터를 받아 처리하고 랜더러 필터에게 넘겨주는 역할을 한다. 랜더러 필터는 하드웨어 디바이스에 맞게 오디오 또는 비디오 등을 랜더링하는 역할을 한다.

MPEG 비디오 파일을 예로 들면, <그림 5>와 같이

STEP 1. 소스 필터가 데이터를 디스크로부터 읽어들이고

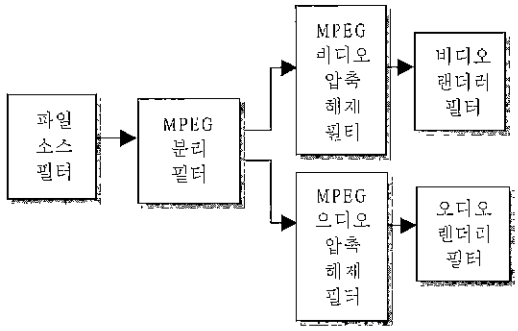
STEP 2. MPEG 필터가 스트림을 파싱하여 오디오와 비디오 스트림으로 분리하고,

STEP 3. 트랜스폼 필터가 비디오 데이터의 압축을 풀고,

STEP 3'. 트랜스폼 필터가 오디오 데이터의 압축을 풀고,

STEP 4. 비디오 랜더러 필터가 비디오 데이터를 스크린에 디스플레이하고,

STEP 4'. 오디오 랜더러 필터가 오디오 데이터를 사운드 카드에 보낸다.



(그림 5) MPEG 비디오 파일 예

4.2 Java Media Framework(JMF)[8, 9]

Java Media Framework (JMF)[8, 9]은 자바의 장점을 그대로 유지하면서 다양한 미디어 데이터 타입을 자바 어플리케이션이나 애플릿에서 일관되게 지원하기 위한 자바 클래스이다. 현 버전 1.0에서는 Java Media Player 및 이에 부속되는 클래스에 초점이 맞추어져 있어서 네트워크를 통하여 멀티미디어 데이터를 재생하는 애플릿 등, 웹 기반 어플리케이션을 지원하고 있다. 특히 인터넷 멀티미디어 스트리밍을 위한 표준 패킷 포멧인 RTP를 이용한 스트리밍 기능을 기본적으로 제공하고 있어서 JMF로 개발되지 않은 다양한 기기 또는 응용 프로그램과도 멀티미디어 정보를 교환할 수 있도록 하고 있다. 또한 현재는 JMF 자체의 포멧 만을 사용할 수 있도록 하고 있는데 앞으로 나올 버전에서는 3rd Party에서 특별한 오디오와 비디오 포맷을 캡춰하거나 랜더링할 수 있도록하고 다양한 코덱도 연동되도록 인터페이스를 제공할 예정이다.

4.2.1 Data Sources

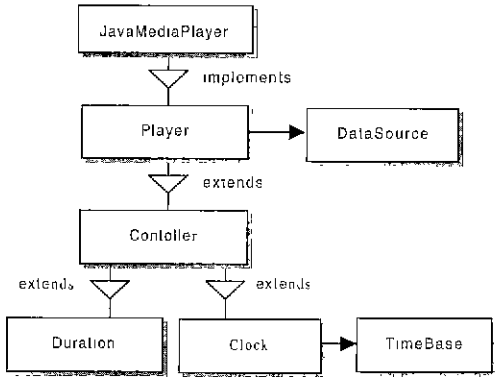
JMF의 DataSource는 미디어의 위치, 전송에 사용되는 프로토콜, 이에 필요한 소프트웨어를 합친 개념의 오브젝트이다. Java Media Player는 이 DataSource를 포함하게 되고, 이렇게 포함된 DataSource는 다른 미디어가 사용할 수 없다. DataSource는 다음과 같이 구분할 수 있다.

* Pull Data Source - HTTP와 같이 클라이언트 쪽에서 데이터 전송을 초기화하고 데이터 플로우를 제어한다.
* Push Data Source - RTP와 같이 서버쪽에서 데이터 전송을 초기화하고 데이터 플로우를 제어한다.

4.2.2 Player

Java Media Player는 데이터 스트림을 진행시키거나 데이터 소스로부터 데이터를 읽어 들이고 랜더링하는 데에 주체가 되는 오브젝트이다. Player와 관련된 상속관계는 <그림 6> 나타나 있다.

* Clock - Player에서 기본적인 타이밍과 동기화 동작을 정의한다.
* Controller - 미디어 이벤트의 발생을 수신하기 위한 수신 메커니즘과 시스템 리소스를 관리하거나 데이터를 프리로딩하는 방법을 제공한다.
* Duration - Duration은 미디어가 플레이 되는 기간을 정한다.
* Player - Player는 표준 사용자 제어를 제공하고 Clock에 의한 제약을 조절한다.



(그림 6) JavaMediaPlayer 상속관계

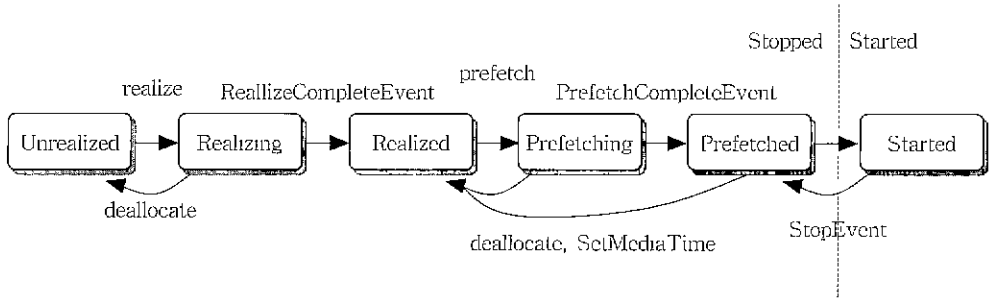
4.2.3 Media Events

JMF 이벤트 보고 메커니즘은 프로그램에서 미디어에서 발생한 에러 상황에 대처할 수 있도록 해준다. 예를 들면, 데이터 리소스의 부재, 리소스의 사용 불가 등이 있을 수 있다. JMF에서 이벤트를 발생시키는 오브젝트는 두 가지가 있는데 GainControl 오브젝트와 Controller 오브젝트가 그것이다. GainControl에서는 GainChangeEvent라는 이벤트 하나만을 발생 시키고, 그 외는 Controller가 발생 시키는 이벤트들이다.

4.2.4 Player States

Java Media Player는 <그림 7>에 나타낸 것과 같이 다음 여섯 상태중의 하나이다.

* Unrealized - 최초 인스턴스화 된 상태. 미디어에 대한 정보는 전혀 없다.
* Realizing - 리소스 요구조건 등을 결정하는 단계.



(그림 7) Player States

* Realized - 리소스 요구조건 등을 결정된 단계. 어떤 리소스가 필요한지 어떤 미디어를 지원할 것인지 알고 있다.
* Prefetching - 미디어를 지원하기 위해 준비하는 단계. 미디어 Player는 미디어 데이터를 프리로딩하고, 배타적인(exclusive-use) 리소스를 확보한다.
* Prefetched - Prefetching이 끝나고 Start단계로 진행될 준비가 되어 있는 상태.
* Started - 데이터의 스트리밍이 시작된 상태.

```
package ExampleMedia;
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.media.*;

public class Player Applet extends
Applet implements ControllerListener {
    Player player = null;
    public void init() {
        setLayout(new BorderLayout());
        String mediaFile= getParameter(FILE);
        try {
            URL mediaURL =
                    new URL(getDocumentBase(), mediaFile);
            Player = Manager.createPlayer(mediaURL);
            Player.addControllerListener(this);
        }
        catch (Exception e) {
            System.err.println(Got exception + e);
        }
    }
    public void start() {
        player.start();
    }
    public void stop() {
        player.stop();
        player.deallocate();
    }
    public void destroy() {
        player.close();
    }
    public synchronized
    void controllerUpdate(ControllerEvent event) {
        if (event instanceof RealizeCompleteEvent) {
            // codes for RealizeCompleteEvent
        }
    }
}
```

(예제 2) 간단한 JMP Player 애플릿

<예제 2>는 JMF를 이용하여 간단하게 작성된 자바 애플릿이다. 애플릿이므로 init, start, stop destroy의 단계를 거친다. 우선 최초 초기화 단계인 init()은 HTML 파일에서 FILE 파라미터의 값을 읽어 들여서 URL 오브젝트를 인스턴스화 하고, player 변수에 해당 URL의 player 객체를 만들어 할당한 다음, 애플릿을 ControllerListener로 등록한다. 실제 애플릿이 동작을 하게 되면 start()가 불려지는데, 여기서 player.start()로 미디어를 재생시키게 된다. stop()과 destroy()는 각각 player를 중지시키거나 완전히 종료하게 한다. 여기서 controller pdate()는 미디어 이벤트에 응답을 하기 위해 작성된다. 현재의 예에서는 RealizeComplete- Event만 처리해 주고 있다.

5. 결 론

본 논문에서는 웹을 기반으로 멀티미디어 서비스를 제공하도록 함으로서 차세대 웹을 위한 핵심 기술로 손꼽히고 있는 W3C의 SMIL과 IETF의 RTSP 및 RTP, 그리고 웹과의 멀티플랫폼 연동을 제공하기 위한 미들웨어로서 마이크로소프트사의 DirectShow와 썬마이크로시스템사의 JMF(Java Media Framework)에 대해 살펴보았다.

SMIL과 RTSP는 표준이 이미 제정되었지만 아직도 많은 보강과 연구가 필요하다. 특히 SMIL은 QoS를 반영하는 문제와 SMIL 문서를 WYSIWYG 방식으로 편집할 수 있도록 하기 위하여 멀티미디어 데이터 베이스나 멀티미디어 처리 기술과 연계된 많은 연구가 필요하다. RTSP도 통신망이나 단말기의 QoS에 적응하면서 세션을 구성하는 기술이나, RTSP 서버를 분산처리하는 기술 등에 대한 연구가 필요하겠고, RTSP가 사용하는 RTP의 신뢰성 및 실시간성 향상을 위한 많은 엔지니어링이 필요하다.

미들웨어에 있어서는 COM을 기반으로 한

DirectShow가 기존의 미들웨어 개념과 자연스럽게 연계되는 강점과 아울러 C++의 수행속도에 힘입어 많은 장점을 갖고 있지만, 자바기반의 JMF도 강력한 프로그램 이식 및 이동성과 간결성에 있어서는 많은 강점을 갖고 있는 것으로 보인다. 프로그램 개발자들은 이러한 장단점을 파악하여 자신이 개발코져하는 프로그램의 목표 환경을 철처히 분석하여 가장 적합한 기술을 선택하여야 할 것이다.

참고문헌

[1] W3C, "Synchronized Multimedia Integration Language(SMIL) 1.0 Specification," W3C, 1998, http://www.w3.org/TR/1998/REC-smil-19980615

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP:A transport protocol for real-time applications.", RFC-1889. Feb., 1996

[3] H. Schulzrinne, "RTP Profile for Audio and Video Conference with Minimal Control", RFC-1890. May, 1996

[4] H. Schulzrinne. A Rao, R. Lanphier,"Real Time Streaming Protocol(RTSP)", RFC-RFC2326, 1998

[5] 강민규 외3, "영상회의 시스템을 위한 RTP/RTCP 설계 및 구현", '97춘계학술발표논문집, 한국통신학회, 1997.

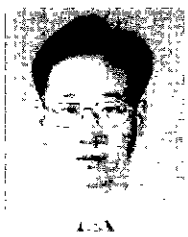
[6] Microsoft, DirectX Media SDK, 1998, http://www.microsoft.com/directx

[7] Dale Rogerson, Inside COM, Microsoft Press, 1997

[8] Sun Microsystems, Java Media Framework Programmer's Guide, V. 0.5, Dec. 21, 1998. http://www.javasoft.com/products/java-media/jmf

[9] Rob Gordon, Stephen Talley, and Robert Gordon, Essential JMF : Java Media Framework, Prentice Hall, 1999

[10] ITU-T, Recommendation H.323, Visual Telephone Systems and Terminal Equipment for Local Area Networks which Provide a Non-Guaranteed Quality of Service, 1995



김 두 현

1985년 서울대학교 컴퓨터공학과 (공학사)
1987년 한국과학기술원 전산학과 (이학석사)
1987년-현재 한국전자통신연구원 멀티미디어그룹웨어팀 팀장, 책임연구원

관심분야 : 인터넷 실시간 멀티미디어 서비스, 분산 멀티미디어 시스템, 멀티미디어 그룹웨어

김 지 용

1990-1995 : 서울대학교 컴퓨터공학과(공학사)
1995-1997 : 서울대학교 컴퓨터공학과(공학석사)
1998-1999 : (주)신테크
1999-현재 : 한국전자통신연구원 멀티미디어그룹웨어팀 연구원

관심분야 : 인터넷 실시간 멀티미디어 서비스, 멀티미디어 그룹웨어, 전자상거래, 인텔리전트 에어전트

황 승 구

1979년 서울대 공과대학 전기공학(학사)
1981년 서울대 광과대학 전기공학(석사)
1986년 미국 University of Florida 전기공학(박사)

현재 한국전자통신연구소 멀티미디어연구부 부장/책임연구원

관심분야 : Networking Computing, Intelligent Computing, Visual Computing, Mobile Computing