CORBA A/V 스트리밍 서비스

경희대학교 홍영래·이승룡*

1. 서 론

네트워크 대역폭과 CPU의 처리 능력이 향상되고, 인터넷 사용이 보편화됨에 따라 멀티미디어 데이터를 실시간으로 전송하고 재생하는 NetShow[1], RealSystems[2]과 같은 멀티미디어 스트리밍 프레임워크가 출현하게 되었다. 스트리밍 기술을 사용하면 인터넷 방송, VOD/AOD(Video on Demand/Audio on Demand) 서비스, 원격 교육, 화상회의 등이 가능하게 된다. 스트리밍이란 재생하기 전에 데이터를 완전히 다운로드 받는 방식과는 달리, 데이터를 전송받으면서 거의 동시에 재생이 이루어져 사용자에게 동시성(isochronous)을 제공하는 실시간 전송기술이다. 스트리밍이 제대로 이루어지기 위해서는 데이터 전송 중에도 소스(source)와 싱크(sink)가 상호 작용을 하며 네트워크의 상태에 따라 스트림 전송 속도를 조절할 수 있어야 하고, 전송 시 손상되거나 손실된 데이터가 있어도 원하는 수준의 서비스 품질(QoS: Quality of Service)을 유지할 수 있어야 한다.

현재 개발되고 있는 많은 스트리밍 프레임워크들은 멀티미디어 응용을 위해 각 개발회사 자체의 스트림 설정/제어 메커니즘을 사용하고 있기 때문에 서로 다른 하드웨어 플랫폼, 운영체제, 네트워크 환경에서 상호운용성(interoperability)이 결여되어 있다. 이러한 문제점을 해결하기 위해서 분산객체 관리 구조에 대한 표준을 정하고 있는 OMG(Object Management Group)에서는 상호 운용성을 제공하는 CORBA 환경에서 A/V(Audio/Video) 스트리밍을 제어하고 관리하는데 필요한 주요 인터페이스와 시맨틱(semantic)을 규정하는 OMG A/V 스트리밍 서비스 표준을 제정하였다[4].

CORBA[3], DCOM(Distributed Component Object Model)[8], Java RMI(Remote Method Invocation)[9]와 같은 전통적인 분산 객체 컴퓨팅(DOC : Distributed Object Computing) 미들웨어는 기본적으로 분산 어플리케이션에 대한 요구/응답 시맨틱을 지원한다. 최근, 멀티미디어 서비스의 요구가 증대됨에 따라 이러한 미들웨어를 사용하는 많은 어플리케이션들은 멀티미디어 데이터의 스트리밍을 요구한다. 그러나, DOC 미들웨어 자체는 시간 제약의 특성을 지닌 스트림 데이터를 지원하기 위한 전송 메커니즘으로서는 부적합하다. 예를 들어, CORBA IIOP(Internet Inter-ORB Protocol)는 너무 많은 데이터 복사와 메모리 할당을 수행하여 패킷의 지연을 증가시키며, 마샬링/디마샬링(marshaling/demarshaling)도 스트림 데이터의 전송율을 감소시킨다[6]. 따라서, DOC 미들웨어에서 제공되는 이식성과 유연성을 잘 유지하는 범위에서 OMG는 기존의 CORBA 구조를 변경하지 않고 A/V 스트림을 제어하기 위한 표준 인터페이스를 제정하였다.

OMG A/V 스트리밍 서비스 사양은 스트림 제어와 관리 객체에 대한 인터페이스, 각 스트림 단말의 스트림 인터페이스 제어 객체에 대한 인터페이스를 정의하고 있으며, 멀티미디어

*종신회원

데이터의 전송은 CORBA GIOP(Generic Inter-ORB Protocol)/IIOP를 사용하지 않고 멀티미디어 데이터 전송에 적합한 전송 프로토콜을 사용할 수 있도록 설계되어 있다.

OMG A/V 스트리밍 서비스의 설계 목표는 표준화된 스트림 설정/제어 인터페이스를 제정하고, 다양한 전송 프로토콜과 VOD 서버, 비디오 카메라, 마이크, 스피커 등 다양한 소스/싱크 유형을 지원하는 것이다. 그리고, OMG A/V 스트리밍 서비스에서 다루는 내용은 유니캐스트, 멀티캐스트, 브로드캐스트 등 다양한 연결 유형과 한 스트림이 여러 개의 플로우를 포함하고 있을 경우의 스트림 제어, 스트림의 표현방법, 설정, 전송, 변경, 종료, 다중 프로토콜 지원방법, 플로우 동기화, 상호운용성, 보안 등 이다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 OMG A/V 스트리밍 서비스 구성요소에 대해서 설명을 하고, 3장에서 이 사양에 따라 A/V 스트리밍 어플리케이션을 구현한 워싱턴 대학의 TAO A/V 스트리밍 서비스[6]와 IONA 사의 Orbix MX[7]에 대해서 살펴본 후, 4장에서 결론을 맺는다.

2. OMG A/V 스트리밍 서비스

스트림은 객체들 사이의 데이터 플로우 집합이며, 플로우는 단방향에서 일련의 프레임들이다. 스트림 인터페이스는 객체와 관련된 하나 이상의 소스/싱크 플로우 종점의 집합이다.

어떠한 유형의 데이터도 객체 사이를 흐를 수 있지만 OMG A/V 스트리밍 서비스 사양은 오디오와 비디오를 다루는 어플리케이션에 초점을 두고 있다. 이 사양은 스트림의 설정과 제어를 위해 잘 정의된 모듈, 인터페이스, 시맨틱을 효율적인 멀티미디어 데이터 전송에 적합한 전송 계층의 메커니즘과 통합하였다. 뿐만 아니라, DOC 미들웨어에 의해서 제공되는 이식성과 유연성도 포함하고 있다.

그림 1은 두개의 스트림 종점 간에 하나의 플로우를 갖는 스트림 구조를 나타낸다. 하나의 스트림 종점은 데이터 소스로 동작하고, 다른 하나는 싱크로 동작한다. 그림에서 가는 화살표는 제어 메시지의 흐름을 나타내며, 굵은 화살표는 스트림 데이터의 흐름을 나타낸다. 점선으로 표시되어 있는 스트림 종점은 IDL (Interface Definition Language)을 제공하는 스트림 인터페이스 제어 객체(Stream Interface Control Object)와 데이터 플로우의 플로우 데이터 소스/싱크, 네트워크를 통해서 프레임을 전송하고 수신하는 스트림 어댑터(Stream Adaptor)로 구성된다.

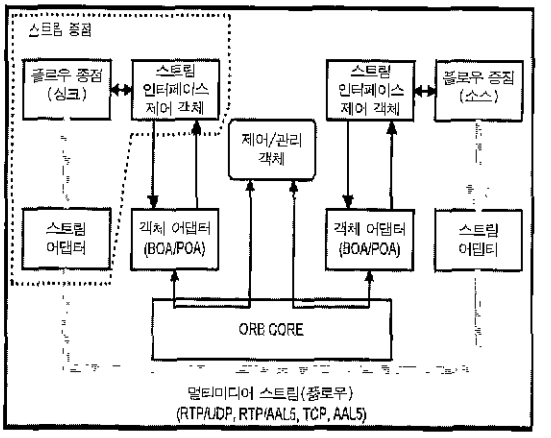


그림 1 OMG 스트림 구조

스트림 인터페이스 제어 객체는 CORBA의 규약에 따라 제어 메시지를 송수신하는 BOA/POA(Basic Object Adapter/Portable Object Adapter)를 사용하여 ORB와 통신하며, 스트림을 제어하기 위해 BOA/POA를 변경하지는 않는다. 그림 1에서 소스/싱크 객체는 스트림 어댑터를 통해서 데이터를 송수신하는데 OMG A/V 스트리밍 사양에서는 이들이 어떻게 통신하는지에 대해서는 정의하고 있지 않다. 그러나, 제어와 시그널은 ORB의 GIOP/IIOP를 이용하여 전송되는 반면 스트림 데이터는 IIOP보다 멀티미디어 스트리밍에 더 적합한 RTP(Real-time Transport Protocol)와 같은 프로토콜을 사용하여 전송할 수 있도록 정의하고 있다.

이 장에서는 스트림을 구성하는 OMG A/V 스트리밍 서비스의 구성요소에 대해서 살펴보고, 스트림의 QoS를 정의하고 변경하는 방법, 그리고 다양한 전송 프로토콜을 관리하는 방법들을 살펴본다.

2.1 OMG A/V 스트리밍 서비스 구성요소

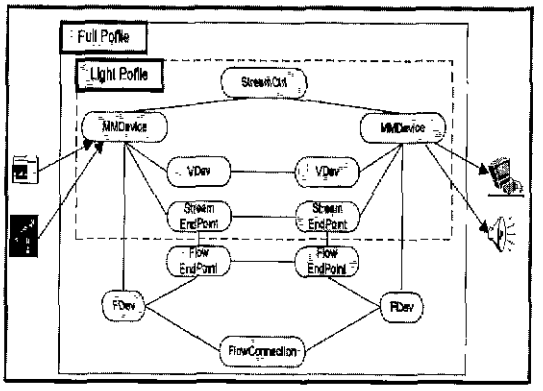


그림 2 OMG A/V 스트리밍 서비스 구성요소

그림 1에 나타난 스트림 인터페이스 제어 객체와 스트림 제어/관리 객체는 그림 2에서 보는 바와 같이 멀티미디어 디바이스(MMDevice)/가상멀티미디어 디바이스(VDev), 스트림 제어(StreamCtrl), 스트림 종점(StreamEndPoint), 플로우연결(FlowConnection)/플로우종점(FlowEndPoint), 플로우 디바이스(FDev)로 구성된다.

OMG A/V 스트리밍 서비스는 두 가지 유형의 IDL을 제공한다. Full Profile은 한 스트림 내에 여러 개의 플로우가 존재할 때 그 안에 있는 각 플로우에 대한 전반적인 제어 인터페이스를 제공하는 반면, Light Profile은 Full Profile의 부분집합으로서 여러 개의 플로우를 독립적인 스트림으로 관리하기 위한 인터페이스를 제공한다. 즉, Light Profile을 사용하여 두 개의 서로 다른 미디어 포맷을 서비스할 때는 두 개의 MMDevice, VDev, StreamEndPoint 객체를 생성하여 관리해야 한다. 이들 각 구성요소들의 기능에 대한 설명은 다음과 같다.

2.1.1 멀티미디어 디바이스(MMDevice/VDev)

멀티미디어 디바이스는 하나 또는 그 이상의 멀티미디어 하드웨어를 나타내고 IDL 인터페이스에서 MMDevice로 표현된다. 멀티미디어 디바이스는 스트림을 사용하는 하나 이상의 멀티미디어 디바이스와 연결될 수 있고, 여러 개의 스트림을 다른 멀티미디어 디바이스에 전송할 수 있다. 이 때 각 스트림의 연결은 VDev와 StreamEndPoint를 생성함으로써 이루어진다.

예를 들어 스트림을 이용하여 두개의 멀티미디어 디바이스를 연결하기 위해 프로그래머는 다음과 같이 bind() 오퍼레이션을 호출할 수 있다.

```
StreamCtrl bind(in MMDevice peer_device,
                inout streamQoS the_qos,
                out boolean is_met,
                flowSpec the_spec);
```

위에서 매개 변수 peer_device는 스트림을 사용하여 이 멀티미디어 디바이스에 연결되어야 할 디바이스를 정의한다. 매개 변수 the_qos는 스트림의 QoS 유형과 파라미터를 정의하는 리스트이며, is_met은 요구한 QoS 파라미터를 peer_device가 만족시킬 수 있는지를 가리킨다. 스트림의 QoS를 정의하는 방법은 2.2절에서 설명되어 있다. 마지막으로, 매개변수 the_spec은 스트림 내의 플로우들을 정의하는 리스트이다. 만일 리스트가 비어 있다면 모든 플로우들을 연결한다. 이 오퍼레이션은 프로그래머가 스트림을 시작하고 중지하며 처리하는데 사용할 수 있도록 StreamCtrl에 대한 참조점을 반환한다.

2.1.2 스트림 제어(StreamCtrl)

스트림 제어는 시작, 중지 등 가상 디바이스(VDev) 사이의 연속 미디어 전송을 제어하고, 스트림을 사용하는 멀티미디어 디바이스들을 연결하는 오퍼레이션을 지원한다. 그리고, IDL 인터페이스에서 스트림의 시작과 중지, QoS 변경 등에 대한 오퍼레이션을 제공하는 Basic_StreamCtrl과 이를 상속하여 bind() 오퍼레이션을 제공하는 StreamCtrl로 표현된다. 만일 프로그래머가 되감기와 같은 더 복잡한 기능을 필요로 한다면 Basic_StreamCtrl을 확장하여 지원할 수 있다.

프로그래머는 멀티미디어 디바이스 간에 스트림을 요구할 때 스트림의 QoS를 명시할 수 있다. QoS는 각 레벨에서 서로 다른 의미를 갖는다. 예를 들어, 비디오를 지원하는 멀티미디어 디바이스는 프레임율, 색상 등과 같은 QoS 파라미터들을 명시할 수 있다. 이러한 유형의 QoS는 어플리케이션 레벨의 QoS로 다루어진다. 그러나, 스트리밍은 기반 네트워크 프로토콜에 의해서 지원되기 때문에, 네트워크 프로토콜을 위한 QoS는 최소 대역폭, 지터 등과 같은 파라미터들을 명시해야 한다. 이러한 유형의 QoS는 네트워크 레벨의 QoS로 취급된다.

프로그래머는 다음과 같이 bind_devs() 오퍼레이션을 사용하여 두 디바이스 간의 스트림을 설정할 수 있다.

```
Boolean bind_devs(in MMDevice a_party,
                  in MMDevice b_party,
                  inout streamQoS the_qos,
                  flowSpec the_spec);
```

매개 변수 the_qos는 어플리케이션 레벨의 QoS를 정의한다. 적절한 프로토콜이 StreamCtrl에 의해서 선택되어지면, 이 프로토콜을 위해 어플리케이션 레벨의 QoS는 네트워크 레벨의 QoS로 변환된다. 만일 MMDevice가 연결을 승인할 수 있으면 스트리밍을 지원하기 위해 StreamEndPoint와 VDev를 생성한다.

미디어 스트리밍 프레임워크는 멀티포인트 스트림을 지원한다. 이것은 하나의 소스 디바이스가 멀티포인트 스트림을 통해서 여러 개의 싱크 디바이스와 연결되는 것을 말한다. 예를 들어, 프로그래머는 비디오 방송을 위해 다음과 같이 할 수 있다.

```
MyStream->bind_devs(cameraDev, nilObject,
                    someQoS, nilFlowSpec);
```

즉, b_party 매개변수로 nilObject에 대한 참조를 사용하는 것은 멀티캐스트 a_party로 카메라 디바이스를 추가하는 것과 같은 효과를 갖는다. 또한, 프로그래머는 아래와 같이 a_party 매개변수로 nilObject에 대한 참조를 사용함으로써 브로드캐스팅을 위한 TV들을 추가할 수 있다.

```
MyStream->bind_devs(nilObject, TVDev,
                    someQoS, nilFlowSpec);
```

2.1.3 스트림 종점(StreamEndPoint)

스트림 종점은 논리적으로 스트림 내에서 각 플로우에 대한 플로우 종점을 포함하고, 이들을 제어하며, IDL 인터페이스에서 StreamEndPoint로 표현된다. 여기에는 두개의 스트림 종점(A-party와 B-party)이 존재한다. A-party는 B-party와 마찬가지로 소스 또는 싱크 스트림 종점이 될 수 있다. 따라서, 종점이 A-party인지 B-party인지를 결정하는 것은 중요하지 않다. 스트림을 설정하거나 연결하는 *connect*()의 호출은 StreamEndPoint_A 또는 StreamEndPoint_B에서 이루어질 수 있다.

그림 3은 StreamCtrl 객체에서 bind_devs()오퍼레이션이 호출될 때의 과정을 나타내고 있으며 그림 내부의 번호는 동작순서를 나타낸다.

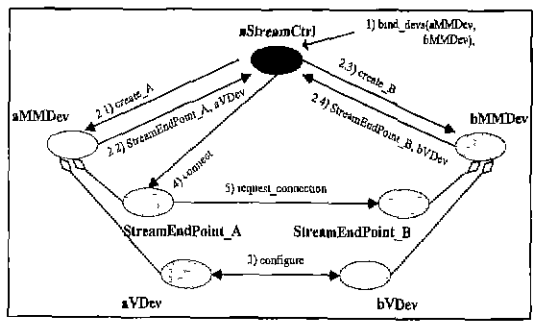


그림 3 스트림 설정 과정(StreamCtrl::bind_devs())

1) 두 멀티미디어 디바이스 간의 스트림을 초기화하기 위해 프로그래머는 aStreamCtrl 객체를 생성한다. 프로그래머는 aMMDev와 bMMDev 간의 스트림 설정을 위해 aStreamCtrl 인터페이스 내의 bind_devs()를 호출한다.

2) bind_devs() 오퍼레이션은 aMMDev와 bMMDev 내의 create_A()와 create_B()를 호출하여 각 멀티미디어 디바이스로 하여금 StreamEndPoint와 VDev를 생성하게 한다. 그러면, create_A()는 aVDev와 관련된 StreamEndPoint_A 객체의 참조점을 반환하며, create_B()도 같은 작업을 수행한다.
3) 2)에서 생성된 aVDev는 bVDev 인터페이스 내의 환경 설정 오퍼레이션 configure()를 호출하고, bVDev도 aVDev 내의 환경 설정 오퍼레이션을 호출한다. 이 오퍼레이션은 두 디바이스가 해당 미디어 포맷과 전송 프로토콜에 대해서 처리 여부를 결정하는 기능을 제공한다.
4) 3)에서 환경 설정이 이루어지면 aStreamCtrl는 StreamEndPoint_B를 매개 변수로 이용하여 StreamEndPoint_A의 connect()를 호출하여 스트림 연결을 요청하고, 스트림 데이터 수신을 기다린다.
5) 스트림 연결 요청을 받은 StreamEndPoint_A는 StreamEndPoint_B의 request_connection()을 호출한다. 그러면, StreamEndPoint_B는 스트림 데이터의 송신을 위한 준비를 하게 된다.

지금까지의 설명은 두 멀티미디어 디바이스 사이의 스트림에 대한 것이었다. 그러나, 프로그래머는 분산 어플리케이션에서 멀티미디어 디바이스와 관련이 없는 객체 사이의 스트림을 원할 수도 있다. 예를 들어, 분산 멀티플레이어 게임에서 플레이어의 위치를 변경하고자 할 때, 이와 관련된 멀티미디어 디바이스는 없다. 그런데, 만일 이를 관리하기 위해 프로그래머에게 더미(dummy) 멀티미디어 디바이스를 정의하게 하는 것은 부적절하며 불필요한 오버헤드를 가져온다. 따라서, OMG A/V 스트리밍 프레임워크는 멀티미디어 디바이스에 독립적으로 존재하는 스트림을 가질 수 있도록 한다. 이때, 스트림은 프로그래머에 의해서 생성된 두개의 StreamEndPoint 사이에 직접 설정되며, 다음과 같이 StreamEndPoint를 매개변수로 하여 StreamCtrl의 bind() 오퍼레이션을 호출함으로써 구현된다.

```
boolean bind(in StreamEndPoint_A A_party,
             in StreamEndPoint_B B_party,
             inout streamQoS theQoS,
             in flowSpec theFlows);
```

2.1.4 플로우 종점/플로우 연결/플로우디바이스(FlowEndPoint/FlowConnection/Fdev)

스트림 설정을 위한 지금까지의 예제에서 스트림은 플로우 종점과 스트림 종점이 공존하는 Light Profile이었다. 그러나, Full Profile을 지원하는 시스템에서 플로우는 개별적으로 접근될 수 있다. 이것은 좀더 정교한 스트림 설정과 제어를 가능하게 한다. 실제로, 스트림은 FlowConnection을 셋업하고 이들을 StreamCtrl로 묶음으로써 설정될 수 있다.

플로우 종점은 is_fep_compatible() 오퍼레이션을 지원한다. 이것은 제 3자가 두 플로우 종점간의 연결이 가능한지를 검사할 수 있다. 플로우 종점은 프로토콜의 선택과 인코딩/디코딩 포맷의 선택을 지원한다. 동일한 전송 프로토콜과 미디어 인코딩/디코딩 포맷을 공유한다면 두 플로우 종점은 호환이 가능하다.

FDev는 StreamCtrl에서의 MMDevice와 유사한 기능을 하며, FlowConnection은 StreamCtrl이 MMDevice를 연결하는 것과 동일한 방법으로 FDev를 연결하는데 사용될 수 있다. 그러나, MMDevice가 StreamEndPoint와 VDev를 생성하는 반면에 FDev는 FlowEndPoint만을 생성한다.

Full profile에서 플로우와 스트림의 두 가지 중요한 차이점은 다음과 같다.

- 스트림 종점은 서로 다른 노드에 위치한 플로우 종점을 묶는 역할을 한다. FlowEndPoint는 StreamEndPoint에서 add_fep()를 호출함으로써 StreamEndPoint에 추가될 수 있다.
- FlowConnection은 단지 하나의 플로우만을 제어한다는 것을 제외하곤 StreamCtrl과 똑같다. 여러 개의 FlowConnection이 설정될 수 있고, 각 FlowConnection에 대해서 StreamCtrl의 set_flow_connection

()을 호출함으로써 StreamCtrl 하에서 그룹화 될 수 있다. StreamCtrl에 적용되는 시작과 중지 등의 오퍼레이션은 FlowConnection에 적용된다.

2.2 QoS 요구사항의 정의와 변경

어플리케이션 프로그래머는 스트림 연결 설정을 위해 어플리케이션 레벨 혹은 네트워크 레벨의 QoS 파라미터를 사용할 수 있다. 어플리케이션 레벨의 QoS는 내부적으로 네트워크 레벨의 QoS로 변환된다. QoS 정의는 속성들의 이름 리스트와 그 값들이다. QoS 구조체를 위한 IDL은 다음과 같다.

QoS 구조체는 QoS의 유형과 파라미터로 구성되어 있으며, QoS 파라미터는 CORBA의 속성 서비스를 이용하여 정의한다.

```
// From Property Service
typedef string PropertyName;

struct Property {
  PropertyName property_name;
  any property_value;
};
typedef sequence<Property> Properties;

struct QoS {
  string QoSType;
  Properties QoSParams;
};
typedef sequence<QoS> streamQoS;
```

QoS구조체는 QoS의 유형과 파라미터로 구성되며, QoS 파라미터는 CORBA의 속성 서비스(Property Service)를 이용하여 정의한다.

스트림을 연결하고 QoS를 변경하는 오퍼레이션은 streamQoS를 매개변수로 사용하며, 어플리케이션 프로그래머로 하여금 플로우 단위로 QoS를 명시할 수 있도록 한다. 예를 들어, "video", "audio"로 불리는 두 개의 플로우를 갖는 스트림이 있다고 하자. 이런 유형의 스트림을 설정할 때 어플리케이션 프로그래머는 video의 QoS와 audio의 QoS를 각각 명시할 수 있는 streamQoS 파라미터와 함께 bind_devs()를 호출한다.

OMG A/V 스트리밍 서비스는 스트림에 관련된 QoS를 서비스 도중에 변경할 수 있다. 이것은 다음과 같이 StreamCtrl의 modify_QoS()를 호출하여 직접적으로 QoS의 변경을 요구함으로써 가능하다.

```
// QoS 변경
try {
  myStream->modify_QoS(newQoS);
} catch(QoSRequestFailed f) {
  ...
}
```

어플리케이션 레벨의 QoS 변경은 다른 객체들에게 영향을 주기 때문에 VDev와 StreamEndPoint의 modify_QoS()를 호출하게 된다. VDev 객체는 QoS 변경 사항에 대하여 만족 여부를 평가하고, 만족할 수 없으면 스트림을 종료하던가 best-effort 서비스를 한다. 만일 QoS를 만족시킬 수 있다면 VDev와 관련된 StreamEndPoint 객체의 modify_QoS()를 호출한다. 이때의 변경은 네트워크 레벨의 QoS 변경이다.

2.3 플로우 프로토콜(Flow Protocol)

일반적인 ORB의 전송 요구사항과 멀티미디어 스트림을 위한 전송 요구사항 사이에는 차이가 있다. ORB는 신뢰성 있는 전송을 요구한다. 이것은 프레임이 폐기되거나 오류가 있을 때 재전송 함을 의미한다. 그러나, 재전송은 지터를 발생시킴으로써 등시성을 가지는 스트림에 치명적인 영향을 미친다. 스트림 데이터는 정확성보다는 시간 제약사항을 만족시키는 것이 더 중요하다. 프레임워크에 의해서 지원될 수 있는 전송에는 다음의 세 가지 유형이 있다.

- 연결지향형 전송 : TCP 같은 프로토콜에 의해서 제공되고, 데이터의 정확성과 신뢰성이 요구된다.
- 데이터그램 전송 : 어플리케이션이 일부 데이터를 손실하거나 어느 정도의 잘못된 시퀀스를 받아들일 수 있을 때 효과적이다.

UDP는 많은 인터넷 기반의 스트리밍 어플리케이션에서 사용한다. 프레임워크는 시퀀스 오류와 패킷 손실을 검사하기 위해 패킷에 일련번호를 삽입한다.

- 비신뢰성 연결지향형 전송 : ATM AAL5에 의해서 제공된다. 메시지는 순서대로 전송되지만 패킷 오류와 손실이 발생할 수 있다. 만일 오류가 발생하면 어플리케이션에 보고된다.

다양한 전송 유형에 따른 요구사항뿐만 아니라 플로우 구성에 영향을 미치는 어플리케이션 레벨의 요인들도 있다. 이것들은 일부 어플리케이션에서 타임스탬프를 요구한다. 또한, 미디어 패킷의 소스를 가리키는 필드의 추가도 요구할 수도 있다. 이 경우는 스트림 종점이 여러 개의 소스로부터 데이터를 받을 때이다. 추가되는 정보는 일련번호, 소스 구별자, 타임스탬프, 동기화 소스 등이다.

스트림 데이터를 위해 요구되는 모든 사항을 만족하는 단일 전송 프로토콜은 존재하지 않는다. ATM AAL5는 플로우 제어가 부족하기 때문에 전송측이 수신측의 전송속도를 넘어설 수 있다. RTP는 위에서 언급한 추가 정보들을 전송하는 메커니즘을 제공하지만 스트림 미디어를 전송하기 위해 플랫폼이 RTP를 지원해야 한다는 가정을 해서는 안 된다. 또한, 어떤 전송 프로토콜도 IDL 유형의 정보를 전송하는 표준은 제공하지 않는다.

멀티미디어 전송의 여러 요구사항을 다룰 수 있는방법은 다양한 전송 프로토콜에서 작동하고 플로우 컨텐트 전송에 독립적인 구조를 제공하는 단순화된 프로토콜을 정의하는 것이다. 이 프로토콜을 SFP(Simple Flow Protocol)라고 한다.

SFP는 전송 프로토콜이 아니라 GIOP와 같이 메시지 레벨의 프로토콜이다. 그리고, 기반 전송 프로토콜의 상위에 존재한다. 또한, 미디어 스트리밍 프레임워크가 SFP를 꼭 지원해야 하는 것은 아니다. SFP를 지원하는 플로우 종점은 이를 지원하지 않는 플로우 종점과 통신하기 위해 SFP의 사용을 중단할 수 있다. SFP의 사용여부는 스트림 설정 시 결정한다. 스트림 데이터가 IDL 유형이 아니라면 기본적으로 SFP는 사용되지 않는다. 이것은 옥테트(octet) 스트림 플로우가 하드웨어에서 직접 네트워크로 전송되게 한다.

3. OMG A/V스트리밍 서비스의 구현 사례

3.1 TAO A/V 스트리밍 서비스

미국 워싱턴 대학에서는 CORBA를 확장하여 실시간 어플리케이션 개발에 적합한 TAO (The ACE ORB)[5]를 개발하였다. TAO에서는 미국 오레곤 대학의 MPEG 플레이어를 OMG A/V 스트리밍 서비스 표준에 맞게 구현하였다[6]. TAO가 지원하는 미디어 포맷은

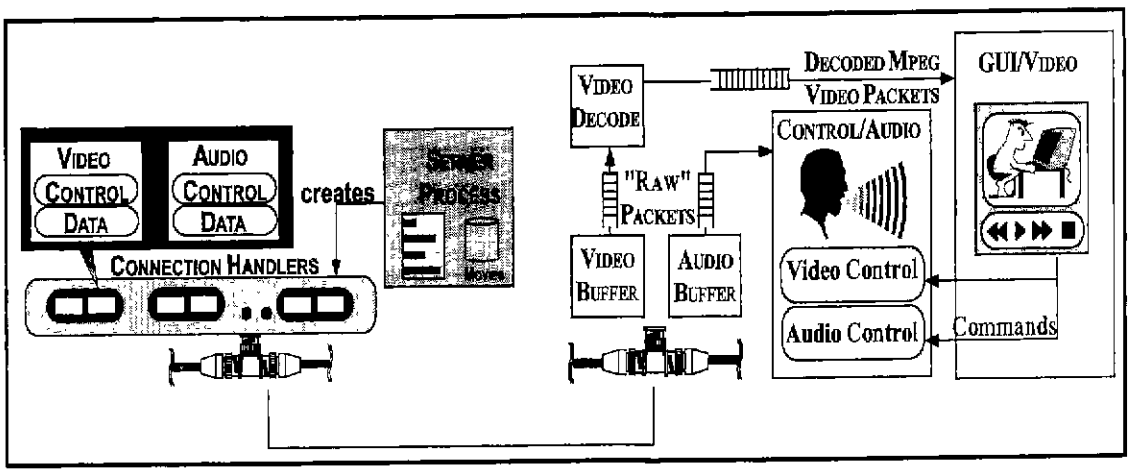


그림 4 TAO A/V 스트리밍 서비스 구조

MPEG-1 비디오와 SUN 사의 ULAW 오디오 포맷이며, TAO A/V 서버와 클라이언트의 구조는 그림 4와 같다.

TAO의 A/V 스트리밍 서비스는 OMG에서 정의한 컴포넌트를 구현하였으며, 다음의 기능들을 제공하고 있다.

- 다양한 스트림 종점 생성 전략 : 성능에 민감한 많은 멀티미디어 어플리케이션은 스트림 종점 생성에 있어서 정교한 제어를 요구한다. 예를 들어, MPEG 어플리케이션의 서버 측은 각 스트림을 병렬처리를 함으로써 스트림의 성능을 최대화하기 위해 프로세스 기반의 동시제어 전략을 사용한다. 반대로, MPEG 어플리케이션의 클라이언트 측은 하나의 프로세스에 관련된 모든 종점을 포함하는 reactive supplier를 생성한다(그림 5). 따라서, TAO 스트리밍 서비스는 스트림 컴포넌트의 생성과 동작을 분리하였고, 이것은 Factory Method와 Abstract Factory 패턴을 이용하였다.

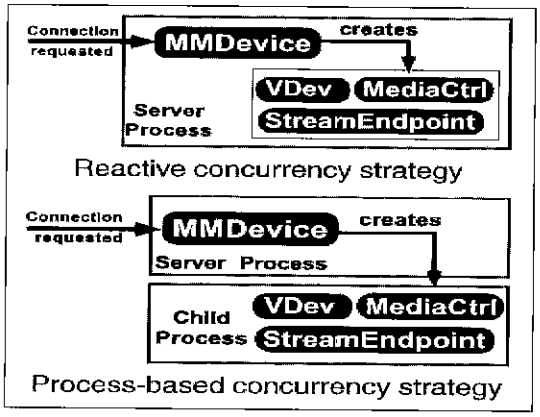


그림 5 스트림 종점 생성 전략

- 다양한 전송 프로토콜 지원 : 스트리밍 서비스는 다양한 전송 프로토콜을 선택하게 된다. 예를 들어, RealSystems과 같은 인터넷 기반의 스트리밍 어플리케이션은 UDP 프로토콜을 사용한다. 반면에, 로컬 인트라넷 기반의 영상회의 시스템은 native ATM 프로토콜에서 제공하는 QoS 특성을 선호한다. 이와 같이 스트리밍 서비스가 다양한 전송 프로토콜을 지원하는 것이 일반적이다.

  OMG A/V 스트리밍 서비스는 전송 프로토콜의 사용에 대해 어떠한 가정도 하지 않기때문에 TAO A/V 스트리밍 서비스는 TCP, UDP, ATM 같은 다양한 프로토콜을 지원하도록 설계되었다.

- 유연한 스트림 제어 인터페이스 : 스트리밍 프레임워크는 개발자가 서로 다른 스트림에 대해 서로 다른 오퍼레이션을 정의하고 사용할 수 있도록 유연성을 제공해야 한다. 예를 들어, 비디오 어플리케이션은 전형적으로 시작, 플레이, 중지, 되감기, 빨리감기 등의 오퍼레이션을 지원한다. 이와는 반대로 주식 인용(stock quote) 어플리케이션은 시작과 중지만을 지원한다. 스트림에 의해서 제공되는 오퍼레이션은 어플리케이션에 따라 다르기 때문에 스트리밍 서비스에서 제어 논리는 매우 유연해야 한다.

- 스트림 공급자와 수요자 상태의 유연한 관리 : 스트리밍 어플리케이션의 전송 컴포넌트는 종종 시스템의 현재 상태에 따라 동작을 변화해야 한다. 예를 들어, 비디오의 스트림 제어 인터페이스에서 플레이 오퍼레이션의 실행은 PLAYING 상태로 변하게 한다. 마찬가지로 중지 오퍼레이션은 STOPPED 상태로 변하게 한다. TAO에서는 이를 위해서 그림 6과 같이 상태 패턴을 사용하였다. 즉, 스트림 시작을 요청받았을 때 스트림의 상태에 따라 이 요청을 처리하는 방법은 다르다.

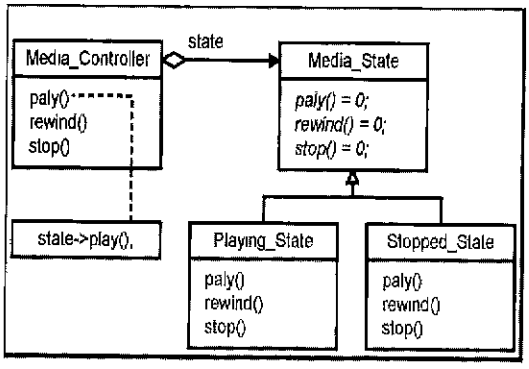


그림 6 상태 패턴

현재 TAO A/V 스트리밍 서비스는 경량 프로파일을 지원하고 있으며, TCP 상에서 구현된 스트리밍 어플리케이션과 성능 차이가 거의 나지 않음을 보이고 있다.

3.2 Orbix MediaStreams

전세계 ORB 시장의 선두를 점하고 있는 IONA 사에서는 Orbix에서의 실시간 지원, 멀티미디어 스트림, OMG의 A/V 스트림 사양의 구현, 다중 전송 프로토콜의 지원을 위하여 Orbix MediaStreams, OrbixTrader, OrbixNames, Orbix IIOP Engine, Orbix/MT 2.3으로 구성된 Orbix MX를 개발하고 있다[7]. Orbix MX는 개방형 서비스 시장에서 통신 서비스 제공자들의 목표를 현실화시킬 수 있도록 도와줄 것이다.

Orbix MX를 구성하고 있는 컴포넌트 중 Orbix MediaStreams는 경량 프로파일의 OMG A/V 스트리밍 서비스 사양을 구현한 것이다. Orbix MediaStreams는 비디오 영상회의, 주문형 미디어, 원격 의료시스템, 멀티미디어 서비스, 주문형 비디오, IDL 유형 데이터의 대량 전송 등과 같은 멀티미디어 스트리밍 어플리케이션 개발을 위한 프레임워크를 제공한다. 프레임워크는 스트림 설정, 제어, 종료를 제공하는 사양화된 IDL 인터페이스를 제공하며, 유니캐스트와 멀티캐스트 통신을 지원한다.

Orbix MediaStreams 프레임워크는 조합과 상속을 통하여 인터넷 기반의 RTP/RTCP (Real-time Control Protocol) 프로토콜 등과 같은 다른 표준들을 지원할 수 있도록 확장될 수 있다. 동일하게 IETF에서 표준화한 스트리밍 프로토콜인 RTSP(Real-time Streaming Protocol)는 Orbix MediaStreams 프레임워크에 매핑될 수도 있고, RTSP를 사용하는 미디어 어플리케이션에서 요구되는 기능들을 지원하기 위해 특정 인터페이스를 개발할 수도 있다. 이것은 RealSystems나 JavaMedia 같이 이러한 표준을 따르는 미디어 스트리밍 어플리케이션이 상호 운용성을 위해 Orbix MediaStreams의 개방성을 이용할 수 있도록 한다.

Orbix MediaStreams를 사용하여 두 개의 스트림 종점 사이의 스트림 연결은 그림 7과 같다.

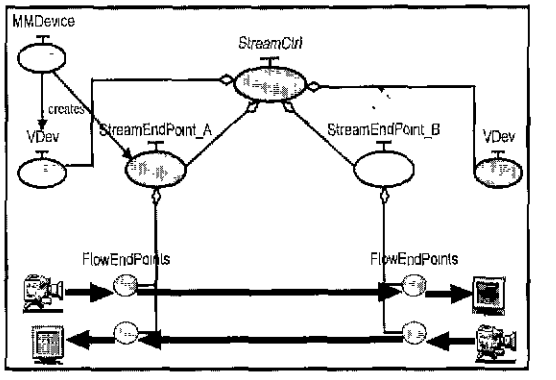


그림 7 Orbix MediaStreams 스트림 구조

MMDevice 객체는 StreamCtrl 객체로부터 bind_devs() 요구를 수신했을 때 Vdev와 StreamEndPoint를 생성하는 팩토리의 역할을 한다. 왼쪽의 VDev가 MPEG 형식의 비디오를 생성한다면, 이것은 스트림 설정 시에 오른쪽의 VDev가 MPEG 형식의 비디오를 처리할 수 있는지를 점사하게 된다. StreamEndPoint는 멀티미디어 데이터의 전송을 담당한다.

*Orbix MediaStreams*는 어플리케이션 레벨, 네트워크 레벨에서 서로 다른 기술에 의해서 요구되는 서로 다른 서비스 레벨 사이의 QoS를 변환하는 QoS 관리 기능을 제공한다.

4. 결 론

OMG에서 제정한 A/V 스트리밍 서비스 사양은 표준 분산객체 미들웨어인 CORBA 환경에서 스트림 제어와 관리 객체에 대한 인터페이스와 각 스트림 단말의 스트림 인터페이스 제어 객체에 대한 표준 인터페이스를 정의하고 있다. 그리고, 멀티미디어 데이터의 전송은 *CORBA GIOP/IIOP*를 사용하지 않고 RTP와 같이 멀티미디어 데이터 전송에 적합한 전송 프로토콜을 사용할 수 있도록 함으로써 CORBA를 사용하지 않은 미디어 스트림 프레임워크와 비슷한 미디어 전송 성능을 발휘할 수 있도록 하였다. 앞으로 더 많은 어플리케이션들이 멀티미디어 서비스를 제공할 것으로 예상됨에 따라 OMG A/V 스트리밍 서비스 사

양은 주문형 오디오/비디오 서비스, 영상 회의, 인터넷 방송, 원격 의료 시스템, 대용량 데이터 전송에서 분산 이기종 시스템을 통합하는 핵심 기술이 될 것으로 예측된다.

감사의 글

본 기고문은 정보통신부 국제공동연구사업(과제번호: TRP-9803-6)의 지원을 받아 작성되었다.

참고문헌

[1] Microsoft, NetShow, http://www.microsoft.com/windows/windowsmedia/.

[2] RealNetworks, RealSystems, http://www.real. com.

[3] Object Management Group, The Common Object Request Broker: Architecture and Specification, 2.2 ed., Feb. 1998.

[4] Object Management Group, Control and Management of A/V Streams specification, OMG Document telecom /98-10-05 ed., October 1998.

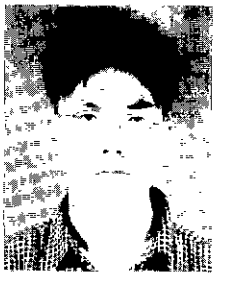
[5] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design and Performance of Real-Time Object Request Brokers", Computer Co- mmunications, vol. 21, pp. 294-324, Apr. 1998.

[6] S. Mungee, N. Surendran, D. C. Schmidt, "The Design and Performance of a CORBA Audio/Video Streaming Service", HICSS-32 International Conference on System Sciences, minitrack on Multimedia DBMS and the WWW, Jan. 1999.

[7] IONA, Orbix MX, http://www.iona.com.

[8] Microsoft, DCOM (Distributed Component Object Model), http://www.microsoft.com/com.

[9] Sun, Java RMI, http://java.sun.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.



홍 영 래

1996 경희대학교 전자계산공학과 학사
1998 경희대학교 전자계산공학과 석사
1998~현재 경희대학교 전자계산공학과 박사과정 재학중
관심 분야 : 실시간 통신시스템, 실시간 CORBA, 임베디드 시스템
E-mail : gomterl@oslab.kyunghee.ac.kr

이 승 룡

1978 고려대학교 재료공학과 학사
1986 Illinois Institute of Technology 전산학과 석사
1991 Illinois Institute of Technology 전산학과 박사
1992~1993 Governors State University 조교수
1993~현재 경희대학교 전자계산공학과 부교수
관심 분야 : 실시간 시스템, 실시간 고장허용시스템, 멀티미디어 시스템
E-mail : sylee@oslab.kyunghee.ac.kr