컨텍스트 기반에서의 멀티미디어 스트림의 사건처리를 위한 Stream Reactor연구

박용희*, 강태성*, 임영환**
*숭실대학교 대학원 컴퓨터학과
**숭실대학교 정보과학대학 컴퓨터학부
e-mail: mirinae@archi.soongsil.ac.kr

A Study on Stream Reactor for the event processing of multimedia streams in context-based

Yonghee Park*, Taesung Kang*, Younghwan Lim**
*Dept. of Computing Soongsil Univ.
**School of Computing Soongsil Univ.

Abstract

기존의 멀티미디어 연구의 실현에 있어 가장 큰 문제라 할 수 있던 성능의 문제가 하드웨어의 급속한 발달로 해결되어 감에 따라 멀티미디어 및 제반 관련기술도 함께 발전되었으며 이에 기반한 multimedia stream에서의 event를 검출하기 위한 다양한 연구들이 진행되어 왔다. 그러나 지금까지의 연구는 주로 전송 및 저장, 검색에 집중되어 연구되어 왔으며 영상인식 등의 Vision관련 연구에서는 멀티미디어 스트리밍 기술과의 연동을 고려하지 않은 연구를 수행함에 따라 검출 가능한 event가 있다고 하더라도 응용영역에 종속적인 인테페이스만을 고려함에 따라 사용자가 이를 기술(記述, description)하거나, 사용자에게 검출 가능한 event를 제시하기 위해 일반화된 방법이 제시되어 있지 않았다. 본 연구에서는 사용자가 검출을 원하는 event를 기술하는 방법과, 시스템에서 검출 가능한 event를 제시하기 위한 방법을 제안하고, 제시되는 방법이 응용영역에 독립적이기 위해 요구되는 사항들과 객체 단위인 이벤트/행위와 처리기 사이의 인터페이스에 관하여 정의한 후 기본적인 동작방식을 제안한다.

1. 서론

지금까지의 멀티미디어 관련연구의 결과로 분산환경에서의 스트림 전송, 저장 및 검색에 관한 다양한 솔루션들이 제시되고 있다. 그러한 대표적인 서비스의 일환으로 VOD등의 서비스가 시범운용 하는 단계에 이르렀다. 하지만 사용자가 VOD서비스를 이용하여 검색을 원하는 경우, 즉 저장된 스트림내에서 원하는 스트림을 검색하고자 할 때 사용자의 의도는 비예측적이다. 이는 비단 검색뿐 아니라 실시간으로 전송되고 있는 스트림에도 발생할 수 있는 요구이며, 멀티미디어 응용서비스개발 과정에서 흔히 일어날 수 있는 문제이다.

멀티미디어 전송과 관련된 연구의 결과로 MuX, Microsoft DirectShow등의 다양한 멀티미디어 스트리밍을 지원할 수 있는 스트림 엔진들이 제시되고 있지만, 영상인식과 관련된 분야에서는 stand-alone 시스템을 고려한 연구가 주를 이루어 오면서 스트림 엔진과의 연동을 고려하지 않은 연구가 진행되어 왔다.

스트리밍 기술과 관련하여 전송, 저장, 검색 등의 문제가 중점적으로 연구되어 왔던 반면, 인식기술에서는 실시간 처리, 인식범위의 확대와 관련한 연구에 중점을 두어 진행함에 따라 두 기술간의 상호연동은 고려되지 않았다. 스트림 엔진들의 주된 입출력 형태가 추상수준의 프레임단위의 입출력 단위임에 비하여, 인식시스템의 입출력은 VFW(Video For Windows)기반의 raw- image 형태 또는 customized CODEC library 기반의 입력에 기초하며 출력은 고려되지 않는다.

따라서 두 기술을 연동시키기 위하여 인식모듈의 입출력형태 변경이 요구되지만, 솔루션의 설계시 고려되지 않았던 확장에 해당되어 수정을 위해 높은 비용을 야기시킨다

2. 정형화 및 인터페이스

2.1. 사건의 정형화

2.1.1. 사건의 정의

사건이란 임의의 시점에서 발생과 소멸중 한 가지 상태만을 갖는 객체를 의미한다.

사건은 발생과 소멸의 상태를 동시에 가질 수 없다.

사건의 초기상태 또한 발생 또는 소멸의 두 가지 상태 중 한 가지 상태이다.

사건의 발생 및 소멸시점은 비예측적이며 상태천이 시점 또한 비예측적이다.

사건의 상태는 다른 사건의 상태에 대하여 독립적이다.

2.1.2 사건객체의 메써드 추출

2.1.2.1 속성 접근 및 제어를 위한 메써드

CEvent::GetDescription()

검출가능한 사건의 내용을 간략하게 제시하는 식별자를 얻는다. 이 식별자를 통하여 사용자에게 제시되어야 할 사건의 명칭을 알 수 있다.

CEvent::GetDuration()

사건의 속성중 시간 속성을 얻는다. 사건발생이 의미를 갖는 시작시간과 종료시간을 돌려준다.

CEvent::SetDuration()

사건의 시간 속성을 설정한다.

CEvent::GetSpatialID()

사건의 공간을 속성을 얻는다. 수치형으로 표현되며 수치의 의미는 제작자의 의도에 종속된다.

CEvent::SetSpatialID()

사건발생이 의미를 갖는 공간을 설정한다.

2.1.2.2 stream reactor 시스템과의 연동을 위한 메써드

CEvent::GetEvaluationProperty()

사건의 발생여부에 대한 판단주체를 의미한다.

CEvent::GetEventHandle()

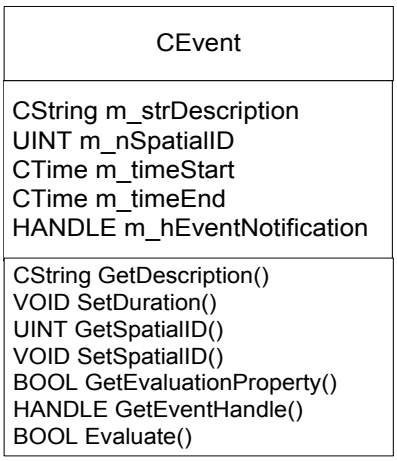
event-driven 평가방식을 가지는 경우 호출되며 이 경우 평가의 주체는 사건객체 자신이므로 사건의 발생을 알리기 위해 stream reactor에서 대기해야 하는 event handle을 돌려준다.

CEvent::Evaluate()

stream reactor에 의한 수동적 판단을 하는 사건 객체인 경우 stream reactor에 의해 주기적으로 호출된다.

2.1.2.3 사건 객체의 UML 표기

하나의 추상수준의 사건 객체는 다음과 같은 UML(Unified Modeling Language) 표기로 나타낼 수 있다.



[그림1] 사건객체의 UML 표기

2.1.3 Plug-in을 위한 DLL규정

windows에 대한 종속성이 없는 plug-in방법으로는 DLL을 이용한 plug-in 방법이 있다.

이러한 stream reactor와 사건 객체의 연동을 위하여 사건객체를 표현하는 모든 DLL은 다음과 같은 export table을 가져야 한다고 규정한다.

CEvent * AllocateEvent()

이 함수는 stream reactor에 의해서 탐색되며, 호출되는 경우 해당 사건객체를 상위 객체인 CEvent로서 돌려준다.

이때에 stream reactor는 전달받은 사건객체를 상위객체인 CEvent 객체로서 취급하고 하위객체는 상위객체의 메써드를 오버라이드(override)함으로써 stream reactor와 연동된다.

2.2 행위의 정형화

2.2.1 행위 정형화를 위한 속성 추출

2.2.1.1 행위의 정의

행위란 임의의 시점에서 수행과 정지중 한가지 상태만을 갖는 객체를 의미한다.

행위는 수행과 정지의 상태를 동시에 가질 수 없다.

행위의 초기상태는 정지 상태이다.

행위의 수행은 조건의 만족여부로 결정된다.

행위의 상태는 다른 행위의 상태에 종속적일 수

있다.

2.2.2 행위객체의 메써드 추출

2.2.2.1 속성 접근 및 제어를 위한 메써드

CAction::GetDescription()

수행되는 행위의 내용을 간략하게 제시하는 식별자를 얻는다. 이 식별자를 통하여 사용자에게 제시되어야 할 행위의 명칭을 알 수 있다.

CAction::HasParameter()

이 행위의 매개변수 필요여부를 알린다..

CAction::AskParameter()

매개변수가 필요한 행위에 한하여 호출되며 이 메써드가 호출될 때에 행위객체는 각 객체에게 필요한 사용자의 입력을 받을 수 있다.

2.2.2.2 stream reactor 시스템과의 연동을 위한 메써드

CAction::IsRunning()

행위가 현재 수행중인지의 여부를 알린다.

CAction::IsMultiplicable()

한 시점에서 복수개의 수행이 가능한 행위객체인지의 여부를 알린다.

CAction::Run()

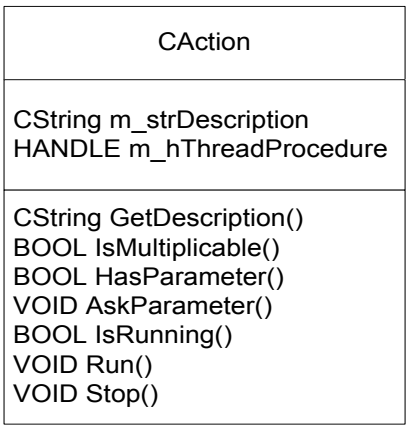
조건절의 조건이 만족되는 경우 stream reactor에 의하여 행위시작을 알리기 위해 호출되며 행위객체는 이 메써드의 호출시 행위 수행을 시작한다.

CAction::Stop()

stream reactor에 의하여 행위가 중단되어야 한다고 판단될 때에 호출되며 행위객체는 이 메써드의 호출시 행위의 수행을 중단하기 위한 절차들을 수행한다.

2.2.2.3 행위 객체의 UML 표기

하나의 추상수준의 행위 객체는 다음과 같은 UML 표기로 나타낼 수 있다.



[그림2] 행위객체의 UML 표기

2.2.3 plug-in을 위한 DLL규정

stream reactor와 행위 객체의 연동을 위하여 사건객체를 표현하는 모든 DLL은 다음과 같은 export table을 가져야 한다고 규정한다.

CAction * AllocateAction()

이 함수는 stream reactor에 의해서 탐색되며, 호출되는 경우 해당 행위객체를 상위 객체인 CAction 형태로 돌려준다.

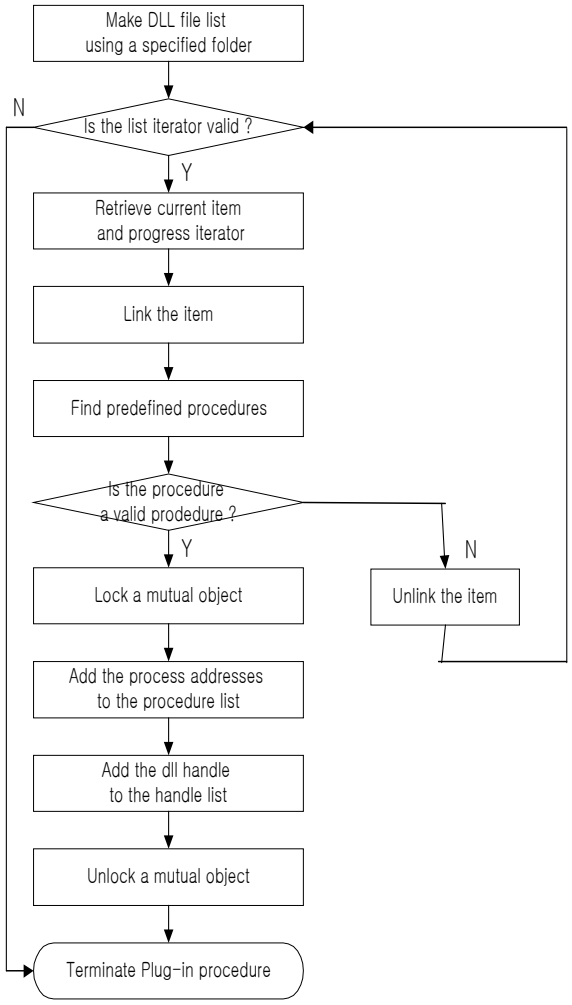
이때에 stream reactor는 전달 받은 행위객체를 상위객체인 CAction 객체로서 취급하고 하위객체는 상위객체의 메써드를 오버라이드(override)함으로써 stream reactor와 연동기능과 함께 해당 행위만의 기능을 수행하게 된다.

3. 구현

3.1 처리 모델

3.1.1 stream reactor 처리모델 개괄

최초 stream reactor는 초기화 단계에서 외부 응용프 로그램 에게 제공 가능한 조건과 행위 리스트를 설정한다. DLL 형태의 조건, 행위들을 동적 연결하여 규정된 export table을 참조함으로써 조건, 행위

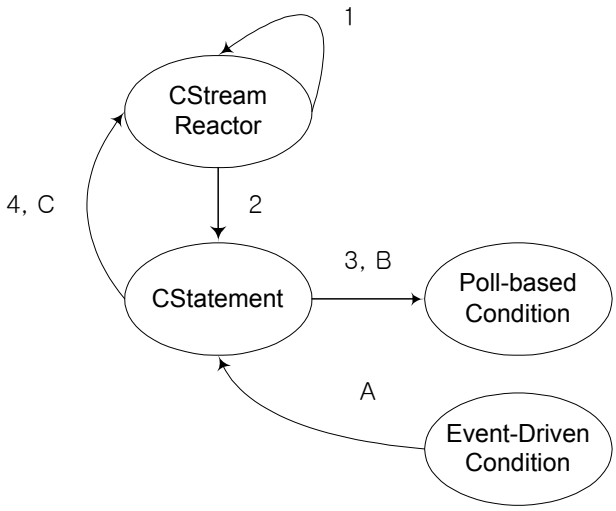


[그림3] plug-in객체 적재과정

객체간의 연결이 이루어진다. 또한 조건 및 행위 객체에게 요구되는 스트림 내용 참조를 위하여 스트림의 구성 또한 stream reactor에서 이루어지며 조건 및 행위의 스트림 참조를 위한 인터페이스가 조건 및 행위 객체에게 제공된다. [그림3]는 시스템의 초기화 과정에서 plug-in 객체의 초기 적재 과정에 대한 도식이다.

3.1.2 조건 판단부 처리 모델

조건 판단을 위하여 고려될 수 있는 조건평가 방식은 poll-based 평가 방식과 event-driven 평가 방식으로 구분될 수 있다. stream reactor 시스템은 두 가지 평가방식을 모두 지원할 수 있어야 하며 조건 객체의 평가방식은 해당 객체의 속성으로 결정된다. stream reactor 시스템의 스케줄러와 구문 객체에 의한 혼재된 평가방식은 다음의 도식으로 나타난다.

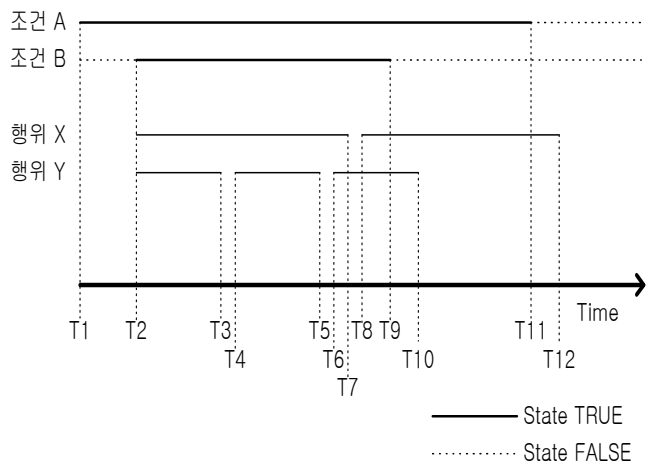


[그림4] 조건 평가

최초 시스템은 스케줄러에 의해 발생된 Tick(1)에 의해 구문 개체를 결정하고, 결정된 구문으로부터 poll-based 조건리스트를 얻는다(2), 획득한 조건 리스트를 순환하여 평가함으로 poll-based 조건의 만족여부를 알수 있으며, 조건판단기간 내에 event-driven 조건의 사건발생이 없으면 구문의 평가치는 거짓이 된다. 그 기간 내에 event-driven 조건의 사건발생이 있는 경우 구문의 평가치는 참이 되고 stream reactor에게 사건의 발생을 알린다(4). event-driven 조건의 사건이 발생한 시점에서는 그 조건개체를 포함하고 있는 구문개체에게 사건의 발생을 알린다(A). 구문개체는 복수 사건의 통지를 받아 복수 event-driven조건이 모두 만족된 경우 poll-based 조건을 평가한다(B). 평가치의 결과로서 구문내의 모든 조건개체가 참으로 평가된 경우 stream reactor시스템에 조건의 만족을 알리고, 행위의 수행을 요구한다(C).

3.1.3 행위 수행부 처리 모델

stream reactor시스템은 해당기간 내에 행위의 무한 반복을 허용하며 현재 행위가 수행중인 경우는 행위의 수행종료를 대기하여 재수행시키는 방법을 택하고 있으며 이는 아래의 도식으로 표현된다.



[그림5] 행위 수행 처리 예

위의 도식은 시구간 내에서 조건 A, B로 구성된 구문에 의하여 행위 X, Y가 수행되는 경우를 나타낸다.

구문은 T2 ~ T9 까지 참으로 평가되고, 구문이 참으로 평가되는 T2 ~ T9까지 행위 X, Y가 각기 다른 수행시간을 가지고 수행된다.

행위 X는 T2 ~ T7구간 동안 1회 수행되고, 수행종료 후 구문은 '참'인 평가 상태이므로 T8 - T7 만큼의 Context Switch 지연을 가진 후 T8시점에서 재수행되어 T12 시점에서 수행을 마치고 비활성화된다.

행위Y는 T2시점에서 최초 실행된 후 T3시점에서 수행을 마친다. T3 시점에서 구문은 참인 상태이므로 다시 T4-T3만큼의 지연을 가진 후 T4시점에서 T5까지 재수행된다. T5시점에서 구문은 계속해서 참인 상태를 유지하고 있으므로 T6 시점에서 3차 수행을 시작하여 T10시점에서 수행이 종료된 후 비활성화된다. T10에서의 4차 수행이 종료된 후 비활성화된다.

T9~T12 까지 구문의 평가상태와 행위의 상태가 비동기적일 수 있으며, 이는 종료시점의 동기화를 위한 행위의 강제종료로 유발될 수 있는 system crash와, 객체작성의 난이도 증가를 막기 위함이다.

3.2 시스템 접근을 위한 API

stream reactor 시스템은 다음과 같은 API 리스트를 제공함으로써 응용프로그램에서 필요한 기능을 제공하게 된다.

CStreamReactor::CStreamReactor();

stream reactor 객체를 생성한다.

VOID CStreamReactor::LoadConditions(VOID);

조건 객체의 plug-in 프로시져를 시작한다.

VOID CStreamReactor::LoadActions(VOID);

행위 객체의 plug-in 프로시져를 시작한다.

CString CStreamReactor::GetDescriptionOfAction(UINT nIndex);

인덱스에 해당하는 행위 개체의 서술부를 얻는다. 응용 프로그램에서는 서술부를 사용자 인터페이스에 이용할 수 있다.

UINT CStreamReactor::GetNumberOfActions(VOID);

stream reactor 시스템에서 제공 가능한 행위 객체의 수를 돌려준다.

CString CStreamReactor::GetDescriptionOfCondition(UINT nIndex);

인덱스에 해당하는 조건 개체의 서술부를 얻는다. 응용 프로그램에서는 서술부를 사용자 인터페이스에 이용할 수 있다.

UINT CStreamReactor::GetNumberOfCondition(VOID);

stream reactor 시스템에서 제공 가능한 조건 객체의 수를 돌려준다.

CObList* CStreamReactor::GetConditionList(VOID);

stream reactor 시스템에서 제공 가능한 조건 객체의 리스트를 돌려준다.

CObList* CStreamReactor::GetActionList(VOID);

stream reactor 시스템에서 제공 가능한 행위 객체의 리스트를 돌려준다.

VOID CStreamReactor::SetRelay(unsigned char cValue);

릴레이 상태 값을 설정한다.

BOOL CStreamReactor::GetRelayStatus(UINT nRelayIndex);

릴레이 장치의 현재 상태를 돌려준다.

VOID CStreamReactor::RegisterRelayCallback (VOID (*RelayCallback)(UINT));

릴레이 상태변경을 통지 받을 프로시져를 등록한다.

VOID CStreamReactor::SetRelayAutomation(CTime timeStart , CTime timeEnd , UINT nRelayIndex, BOOL bOnOff);

릴레이 제어장치의 상태를 자동 제어한다.

VOID CStreamReactor::DeleteRelayAutomation(UINT nRelayIndex);

설정된 릴레이 자동제어를 취소한다.
VOID CStreamReactor::RegisterEventCallback(VOID EventCallback(UINT , CString));
사건발생을 통지 받을 프로시져를 등록한다.
UINT CStreamReactor::AddConditionAndAction(CTime timeStartTime, CTime timeEndTime, UINT *pIndexOfConditions, UINT nCountOfConditions, UINT *pIndexOfActions, UINT nCountOfActions);
구문의 내용을 통해 조건/행위를 추가한다.
VOID CStreamReactor::AddStatement(CStatement *pStatement);
주어진 구문개체를 시스템의 구문리스트에 추가한다.
CObList * CStreamReactor::GetStatementList(VOID);
시스템에 등록된 구문리스트를 돌려준다.
CStatement* CStreamReactor::GetStatement(UINT nStatementIndex);
stream reactor 시스템에서 지정된 인덱스에 해당하는 구문개체를 돌려준다.
VOID CStreamReactor::DeleteStatement(UINT nIndex);
VOID CStreamReactor::DeleteStatement(CStatement *pStatement);
조건/행위 구문을 삭제한다.

4. 결론

멀티미디어 스트림의 컨텍스트 기반 사건 처리를 위한 사건 및 반응 객체를 모델링하고 각 개체들의 상호작용을 모델링함으로써 stream reactor 시스템을 설계하였으며, 이의 구현을 통하여 다양한 사건 및 반응의 처리가 가능한 스트림 반응기 시스템을 구축하였다.
stream reactor 시스템은 응용영역에 독립적일 수 있도록 추상수준의 조건 및 반응 객체를 모델링하고 하위 객체들에 대한 Plug-in을 지원하도록 하였다. 이를 통해 응용 영역의 요구에 따른 조건 및 행위 객체들을 제작/추가함으로써 응용영역의 변화에 적응적인 서비스가 가능하도록 하였다.
또한 기존의 스트리밍 엔진인 MuX를 개선하여 인식기술의 적용이 가능한 필터메커니즘을 추가함으로써 인식기술과 스트리밍 기술을 접목시킬 수 있는 방법을 제시하였으며, 그러한 스트리밍 엔진과의 연동을 통해 분산환경에 대한 seamless stream service를 제공하는 시스템으로 구축하였다.
본 문에서 제시된 시스템을 이용하여 향후 사용자의 다양한 요구를 만족시킬 수 있는 고품질의 멀티미디어 스트림 서비스가 가능해질 것으로 예측되며. 실시간 반응이 요구되는 분야를 비롯한 여러 멀티미디어 스트림 컨텍스트 기반 서비스에 적용 가능할 것으로 예측된다.

5. 참고문헌

[1] 임영환 등, 멀티미디어 컴퓨터공동연구개발 연구보고서, 1994. 7.
[2] Baker R, A. Downing. K. Finn, E. Rennison, D.H. Kim, and Y.H. Lim, "Multimedia Processing Model for a Distributed Multimedia I/O System," 3rd International Workshop on Network and Operating System Support for Audio/Video, 1993.
[3] 임영환, ComBiStation : 분산 멀티미디어 컴퓨팅 환경을 위한 컴퓨터플랫폼, 정보과학회 논문지, 제 2권, 제 1호, 1996, pp.160-181.
[4] C.W.Mercer and H.Tokuda, "The ARTS Real-Time Object Model," IEEE Real-Time System Symposium, 1990, pp.2-10.
[5] International Standards Organization, Hypermedia/Time-base Document Structuring Language(HyTime), ISO/IEC, 1992
[6] Martin Fowler and Kendall Scott, UML Distilled Applying the Standard Object Modeling Language, Addison Wesley, 1997
[7] 함재욱 외 4명, “멀티미디어 프리젠테이션 동기화를 위한 선행 스케쥴링 기법,” 정보과학회 논문지, 제21권, 제12호, 1994, pp.2187-2197.
[8] T.Tsang, R.Lai, "Time-Estelle : An Extended Estelle Capable of Expressing Multimedia QoS Parameters," 1997, pp.311-318.
[9] Doo-Hyun Kim, Sang-Hwan Kung, and Chee-Hang Park, "Empirical Delay Comparisons of Push and Pull Implementation Models for Local Audio and Video Streams,"IEEE PROMS-MmNet97, Santiago, Chile, 1997, pp.24-26.