분산 환경에서 CM 스트림 인터페이스를 지원하는 계산 객체의 설계 및 구현 방안 연구

송 병 권[†] · 진 명 숙[††] · 김 건 웅[†††]

요 약

본 논문은 분산 환경에서 CM(Continuous Media) 스트림을 지원하는 계산 객체 모델의 제시 및 구현 방안에 대한 연구이다. 이를 위해 OMG CORBA 객체 모델을 바탕으로, QoS(Quality of Service)를 고려한 스트림 인터페이스를 지원하는 계산 객체 모델을 제시하며, 제시한 스트림 인터페이스의 구현 방법에 대해 기술한다. 스트림 인터페이스는 CM 데이터 전송 채널과 제어 채널로 구성된다. 본 논문에서는 CORBA 객체 간 통신을 스트림 데이터의 제어 채널로 사용하며, CM 데이터의 전송을 위해 다양한 트랜스포트 프로토콜을 사용할 수 있도록 하였다. QoS 지원을 위해, 스트림 인터페이스에 응용 QoS 정보를 포함하도록 하며 FIFO 큐와 타이머를 도입한 전송률, 지연(delay), 지터(jitter) 제어 기능의 구현 방안을 제시한다.

A Study on the Design and Implementation Method of Computational Object Supporting CM Stream Interface in the Distributed Environment

Byung-Kwen Song[†] · Myung-Sook Jin[††] · Geonung Kim[†††]

ABSTRACT

This paper presents a computational object model supporting CM(Continuous Media) stream interfaces including QoS(Quality of Service) required in the distributed applications. We introduce a new stream interface based on the object model of OMG CORBA and propose an implementation method for the proposed stream interface including QoS. A stream interface consists of a data channel and a control channel. In this paper, the CORBA supporting communication channel is used as the control channel and various transport protocols can be used as the data channel of the stream interface. Also, specifications of the application QoS are included in stream interface specification. In implementation, FIFO queues and timers are used to support transmission rate, delay and jitter control mechanisms of the stream interface.

1. 서 론

객체 지향 기법은 캡슐화, 재사용성 및 상속성을 바탕으로 다양한 소프트웨어 응용 영역에서 사용되고 있다. 분산 컴퓨팅에서의 객체 지향 기술은 특정 관점에 중점을 두고 시스템을 모델링하는 방법으로 활용된다. RM-ODP(Reference Model Of Open Distributed Processing)[16, 17], ANSA(Advanced Network System Architecture)[6] 등에서는 분산 시스템을 5 가지의 관점으로 나누어 서술하는데, 이는 엔터프라이즈(enterprise) 관점, 정보(information) 관점, 계산(computational) 관점, 공학(engineering) 관점, 기술(technology) 관점이다. 이러한 특정한 관점에 따른 시스템에 대한 시각은 복잡한

† 종신회원 : 서경대학교 정보통신공학과 교수
†† 정 회 원 : 경인여자대학 멀티미디어정보인산학부 교수
††† 종신회원 : 목포해양대학교 해양전자통신공학부 교수
논문접수 : 1999년 12월 17일, 심사완료 : 2000년 4월 24일

시스템의 기술, 분석, 결합에 유용하게 적용될 수 있다.

시스템에 대한 5가지 관점 중 정보 관점과 계산 관점은 시스템 개발자 측면에서 분산 응용 시스템에서 다루어지는 정보와 시스템의 명세를 위한 기본 개념을 제공한다. 이 중 계산 관점은 어플리케이션의 구조, 어플리케이션간의 인터페이스 등을 논리적으로 기술하며 이들 개념은 계산 객체로 나타난다. 계산 객체는 타 계산 객체를 위한 기능을 제공하는 창구로서 계산 인터페이스를 가진다. 계산 인터페이스는 단일 서비스의 요청과 응답으로 구성되는 오퍼레이션 인터페이스와 연속 미디어(CM : Contunuous Media) 데이터를 전송하는 스트림 인터페이스로 나누어진다[6, 16, 17, 19].

네트워크 기술의 발달을 바탕으로 분산 멀티미디어 컴퓨팅 관련 연구들이 점차 활기를 띠고 있으며 따라서 오디오, 비디오 등 CM 데이터를 전달하는 스트림 인터페이스의 중요성이 증대되고 있다. OMG(Object Management Group)의 경우, CORBA(Common Object Request Broker Architecture) IDL(Interface Definition Language)을 통해 계산 객체를 기술하는 명세가 제시되고 있으며, 이들의 구현 제품이 출시되고 있으나 스트림 인터페이스가 지원되지 못하고 있다. TINA[19], ANSA[6], RM-ODP[16, 17] 등에서는 스트림을 인터페이스의 한 종류로 정의하고 있으나 구현에 대한 구체적인 방안의 제시가 이루어지지 못하고 있다. IMA/MSS(Interactive *Multimedia Association/Multimedia System Services*)[11]는 스트림 인터페이스와 이를 지원하는 하드웨어 및 유무형의 자원을 계산 객체로 간주한다. 따라서 어플리케이션 개발 시 스트림 인터페이스를 위한 하부 자원에 대한 관리를 함께 수행해야 하는 불편함이 따른다.

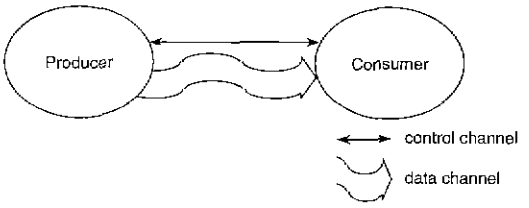
본 논문은 OMG 계산 객체를 바탕으로 CM 스트림 인터페이스를 지원하는 계산 객체 모델을 제시한다. 제안 모델은 스트림 인터페이스를 위해 계산 객체 형태의 스트림 관리자와 네트워크 관리자를 둔다. 스트림 및 네트워크 관리자를 통해, 어플리케이션 객체가 기존의 오퍼레이션 인터페이스를 사용할 경우와 같이 부과적인 자원의 관리 없이 스트림 인터페이스를 사용하도록 한다. 또한 제안 모델은 QoS(Quality of Service) 정책을 포함한 QoS 파라메타를 스트림 구조에 포함한다. 본 논문에서는 스트림 인터페이스의 구현 방안으로, CORBA 오퍼레이션 인터페이스를 스트림의 제어 채널로 사용하며, 외부 트랜스포트 프로토콜을 스트림 데이터 채널로 도입하는 방법을 제시한다. 구현에서는 QoS 제어 방법으로 FIFO 큐와 타이머를 사용하여 QoS 파라메타 중 전송률, 지연, 지터 제어 기능을 수행하는 방안을 제시한다.

논문의 구성은 2절에서 계산 객체 및 스트림 전송과 관련한 기존 연구들을 살펴보고 3절에서는 QoS를 포함한 스트림 인터페이스를 제공하는 계산 객체 모델을 제안한다. 4 절에서 제안 스트림 인터페이스의 구현 방법을 소개하고 5절에서는 구현 방법에 따른 CM 스트림 인터페이스의 적합성 검사를 위해 실제 CM 데이터를 사용한 스트림 전송의 결과를 분석하고 QoS 제어 방안을 살펴본다.

2. 관련 연구

계산 관점에서 분산 어플리케이션 시스템은 특정 기능을 제공하기 위해 상호 작용하는 객체들의 집합으로 보여진다. 이 때 계산 객체는 데이터와 프로세싱을 캡슐화하여 타 계산 객체에 의해 사용되는 기능(capability)을 제공하는 추상화 된 대상이다. 계산 객체는 타 계산 객체를 위한 기능을 제공하는 창구로서 인터페이스를 가지며 이는 타입(type)에 따라 오퍼레이션(operation) 인터페이스, 스트림 인터페이스 등으로 나누어진다[1].

오퍼레이션을 일회적인 이산 데이터의 전송, 즉 한번의 서비스 호출과 이에 대한 응답으로 간주할 때 스트림은 연속적인 데이터의 흐름으로 볼 수 있다. 따라서, CM 데이터는 시간에 따라 연속적으로 변화하는 데이터로 등시성(isochronous), 주기성을 가지는 비디오 데이터, 오디오 데이터, 일정 주기로 측정되는 센서의 값 등을 예로 들 수 있다. 일반적으로 스트림 인터페이스는 (그림 1)에서와 같이 데이터 채널과 데이터의 흐름을 관리하는 제어(control) 채널로 구성된다. 제어 채널은 양방향 채널이며 데이터 채널은 스트림의 생산자(producer)와 소비자(consumer) 사이의 단 방향 채널이다. 제어 채널을 통해 스트림 흐름의 중지, 재개, 매속 전송, QoS 제어 정보 등 다양한 제어 기능을 위한 정보가 전달된다. 스트림 인터페이스를 계산 모델에서 지원하기 위해서는 우선 CM 데이터 스트림에 대한 새로운 형식의 통신 추상화가 요구되는데, 이는 시간 축에 정보의 흐름을 표시하는 것으로 단일 호출(single invocation)로는 무리가 따를 수 있으며 스트림의 제어 기능이 오퍼레이션 형태로 함께 명세되어야 한다[3, 4, 9, 15].



(그림 1) 스트림 인터페이스

서비스 품질(QoS : Quality of Service)은 응용에서 요구되는 분산 멀티미디어 시스템의 질적, 양적 특성의 집합으로 나타난다. QoS는 처음에는 처리율, 전송 지연, 에러율 등 데이터 전송 기술의 특징 및 성능을 표현하기 위해 도입되었으나, 멀티미디어 응용에서 점차 CM 데이터를 요구함에 따라 전체 시스템 관점에서 QoS의 중요성이 증가되고 있다[2, 5, 13, 14, 20]. 따라서 스트림 인터페이스 또한 QoS측면에서, 사용자가 원하는 수준의 QoS를 명세할 수 있어야 하며 QoS를 동적으로 협상, 재협상할 수 있는 제어 기능도 함께 표현되어야 한다. 이밖에도 QoS의 변화를 사용자에게 통보하거나 자체적인 대처 기능 및 하부 자원의 예약, 관리 기능도 제공되어야 한다.

분산 시스템에서의 스트림은 객체 인터페이스의 한 형태로 간주되거나 독립된 객체로서 제어 인터페이스를 가지는 형태로 나타난다. 스트림을 객체의 인터페이스의 종류로 보는 연구는 TINA[19], ANSA[6], RM-ODP[16, 17] 등이 있으며, 스트림을 제어 오퍼레이션을 갖는 계산 객체의 한 종류로 간주하는 연구로는 IMA/MSS (Interactive Multimedia Association/Multimedia System Services)[11]가 있다.

OMG CORBA는 분산 환경에서 구현 기술이나 시스템의 이질성에 관계없이 객체가 서비스를 요청하고 응답을 받을 수 있도록 객체간의 상호작용을 가능하게 하는 하부구조이다. CORBA는 객체-구현(서버) 및 클라이언트에 상호운용성(interoperability)을 보장하며 한 객체가 제공할 서비스는 다수의 오퍼레이션으로 구성된 인터페이스로서 표현된다. 인터페이스 명세를 위해 IDL (Interface Definition Language)이 사용된다[18]. OMG IDL은 계산 명세를 위한 언어 개발의 바람직한 기본으로 인정받고 있으나 스트림 인터페이스에 대한 지원이 이루어지지 않고 있다.

TINA ODL(Object Definition Language)은 OMG IDL을 확장시킨 언어로서 객체 템플릿을 명세할 수 있고 그 안에 객체간의 상호작용을 위한 다수의 인터페이스를 포함할 수 있다. 한 객체에 대해 복수 개의 인터페이스를 허용한다는 점이 OMG IDL과 다르다[18, 19]. TINA의 경우, 스트림을 객체의 인터페이스의 한 종류로 보지만 이에 대한 구체적인 사양의 제시가 이루어지지 못하고 있다.

IMA의 MSS 아키텍쳐는 다양한 이질 컴퓨팅 환경에서 분산 멀티미디어 어플리케이션 개발을 위한 표준화된 명세의 제시를 목표로 한다. MSS는 하드웨어, 소프트웨어, 연결(connection) 등 유, 무형의 자원을 추상화하여 가상 자원(virtual resource) 형태로 표시한다. MSS 가상 자원은 가상 장치(virtual device), 가상 연결(virtual connection), 가상 클럭(virtual clock) 등으로 나누어진다. 가상 자원들은 인터페이스를 가지는 계산 객체 형태로 사용자에게 제시된다. 가상 객체에 속하는 스트림 객체는 클라이언트에 스트림 내의 위치를 제공하는 오퍼레이션과 스트림의 멈춤, 재개 등의 오퍼레이션을 제공한다. MSS는 가상 자원 객체들을 명세하기 위해 OMG IDL을 사용하여 객체들의 인터페이스를 정의하며 분산된 객체 간의 상호 작용을 위해 OMG CORBA에 가반한 객체간 통신 방법을 사용한다[11]. MSS의 경우, 스트림 전송시 다양한 가상 객체들을 고려해야 하는 부가적인 자원의 관리가 필요하다.

3. 제안 계산 객체 모델

본 절에서는 OMG 객체 모델의 오퍼레이션 인터페이스에 스트림 인터페이스를 추가 한 객체 모델을 제안한다. 제안 모델은 스트림 전송을 위해 사용자가 부가적인 자원의 관리를 수행하지 않으며 오퍼레이션 인터페이스의 호출과 같은 단순한 방법으로 스트림 인터페이스 사용할 수 있는 모델이다.

3.1 스트림 인터페이스의 구조

OMG 객체 모델의 오퍼레이션 인터페이스에 추가하여 본 논문에서 제안한 스트림 인터페이스의 구조는 (그림 2)와 같다. 스트림 인터페이스는 스트림 식별자와 스트림 속성으로 구성되며 스트림 속성은 스트림 서비스의 역할 및 스트림 미디어의 타입(오디오, 비디오 등)과 어플리케이션 QoS를 포함한다.

```
<stream_dcl>        ::= "stream" <identifier> <stream_attr>
<stream_attr>       ::= <stream_init_role><stream_type>
                        [<ene2end_qos_attr>]
<stream_init_role>  ::= "producer"| "consumer"
<stream_type>       ::= <type_spec>
<end2end_qos_attr> ::= <qos_spec>
```

(그림 2) 스트림 인터페이스 구조

제안된 스트림 구조에 포함된 QoS는 (그림 3)과 같은 구조를 가지며 프레임(frame) 율과 프레임 크기, 손실 허용율, 지연, 지터 등으로 구성된다. 이외에도 QoS 정책으로 deterministic, statistical, best-effort 등이 포함된다 또한 요청된 QoS 수준을 유지하지 못할 경우, 재협상(renegotiation) 수행 여부와 단순 통보(indication), 연결 해제(disconnect) 또는 무시(no_action) 등의 행위를 포함할 수 있다

```
Typedef enum {deterministic, statistical, best_effort} commit,
Typedef enum {renegotiate, indication, disconnect,
              no_action} action ,
Struct end2endQoS {
  int no_QoS_Parameters ,
  string QoS_Parameter_Names[] ,
  string QoS_Parameter_Values[] ,
  commit QoS_Commitment[] ;
  action QoS_Adaptation[] ;
}
```

(그림 3) end-to-end QoS 구조

본 논문에서는 스트림 인터페이스를 지원하는 계산 객체로 스트림 관리자와 네트워크 관리자를 두어 하부 자원의 관리와 스트림 및 QoS를 제어하도록 한다. 스트림 관리자는 계산 객체의 요청에 따라 네트워크 관리자와 협력하여 스트림 데이터 채널의 설정과 해제, 스트림 데이터의 전송과 중지 등 제어기능을 수행한다. 그 외에도 스트림 관리자는 QoS 협상이나 재협상에 따른 QoS변화의 요구에 따라 네트워크 관리자를 통해 스트림 데이터 채널의 성능 제어 및 모니터링 기능을 수행한다. 네트워크 관리자는 스트림 관리자로부터 스트림 데이터 채널의 설정 요청에 따라 실제 데이터 채널을 설정한다. 네트워크 관리자는 다양한 하부 트랜스포트나 네트워크 프로토콜을 수용하여 스트림 관리자에게 전송 프로토콜에 대한 투명성(transparency)을 제공한다. 네트워크 관리자는 스트림 데이터 전송 채널의 설정과 해제 및 스트림 관리자로부터 생성된 SDU(Service Data Unit)들을 ORB(Object Request Broker)를 거치지 않고 직접 트랜스포트나 네트워크 계층으로 전달하는 역할을 한다.

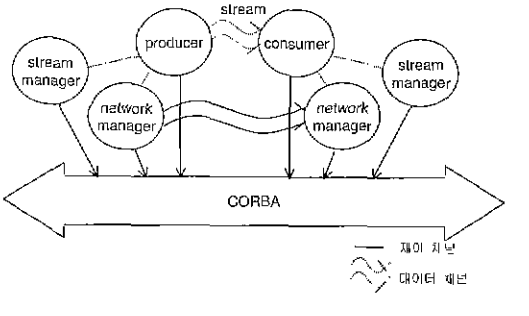
3.2 스트림 관리자와 네트워크 관리자

제안 모델에서의 스트림 인터페이스를 지원하는 스트림 관리자와 네트워크 관리자는 일반 계산 객체이다.

스트림 관리자는 스트림 인터페이스의 설정과 제어를 위한 스트림의 제어 채널로 CORBA 오퍼레이션을 사용한다. 스트림 제어 오퍼레이션으로, 스트림 관리자가 지원하는 스트림 연결(streamConnectReq), 스트림 해제(*streamDisconnectReq*), 스트림 데이터 전송 중지(stream PauseReq), 스트림 데이터의 전송 재개(streamResumeReq) 오퍼레이션을 둔다. 스트림 제어와 마찬가지로 스트림 QoS의 제어를 위해서 스트림 관리자가 오퍼레이션 인터페이스를 제공하며 QoS 협상, QoS 재협상이 이에 해당한다.

스트림 제어와 별도로, 스트림 데이터의 전송을 위해서 네트워크 및 트랜스포트 프로토콜을 사용하는 외부 채널을 두며 이는 네트워크 관리자에 의해 제어된다. 본 논문에서는 스트림 전송 프로콜의 예로 UDP(User Datagram Protocol)을 사용한 구현 방안을 4절에서 제시한다.

(그림 4)는 스트림 관리자, 네트워크 관리자 및 스트림 송수신 객체인 *producer, consumer*의 관계를 보여준다. 그림에서 스트림 producer는 수신측 스트림 관리자에 스트림 인터페이스의 설정을 요청하며 스트림 관리자는 네트워크 관리자에 데이터 채널의 설정을 요구한다 네트워크 관리자는 송신측 네트워크 관리자와의 데이터 채널을 설정하고 스트림 관리자의 제어에 따라 스트림 전송을 수행한다.



(그림 4) 스트림 관리자와 네트워크 관리자

3.3 스트림 제어 인터페이스

제안 모델에서는 스트림 제어 채널로 CORBA 오퍼레이션 인터페이스를 사용한다. 다음은 이를 위한 스트림 관리자의 인터페이스이다.

- streamConnectReq(스트림 설정)

producer(consumer)가 스트림 관리자에게 스트림 데이터 채널의 설정을 요구하는 오퍼레이션이다. Producer(consumer)는 요구하는 consumer(producer)의 정보와 스트림 타입, 요구 QoS정보를 스트림 관리자에게 전달한다

- streamDisconnectReq(스트림 해제)

producer(consumer)가 스트림 관리자에게 스트림 데이터 채널의 해제를 요구하는 오퍼레이션이다. 요청을 받은 스트림 관리자는 네트워크 관리자에게 이를 통보하여 스트림 전송을 위해 할당된 자원을 해제한다.

- streamPauseReq(스트림 데이터의 전송 중지)

streamPauseReq는 스트림 전송을 잠시 멈추도록 스트림 관리자에게 요청하는 오퍼레이션이다. 다시 전송을 재개할 경우, 스트림 데이터의 전송 재개(streamResumeReq) 오퍼레이션을 호출한다.

스트림 QoS의 제어를 위해서는 스트림 관리자의 qosNegotiation(QoS 협상) 오퍼레이션을 사용하며 이는 요구하는 QoS 파라메타 값과 정책, 수행 방법(renegotiate, indication, disconnect, no_action 등)을 입력으로 하며 수행 방법에 따라 반환되는 값의 형태가 달라진다 QoS의 재협상 시에도 같은 qosNegotiation 오퍼레이션을 이용하며 이 때는 새로운 QoS 파라메타 값과 정책, 수행 방법이 입력으로 사용된다.

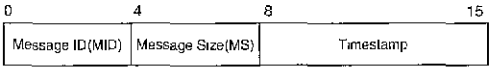
4 스트림 인터페이스의 구현 방안

본 절에서는 3절에서 제안한 계산 객체 스트림 인터페이스의 구현 방안에 대해 기술한다. 제안 모델은 스트림 관리자와 네트워크 관리자를 도입하고 CORBA의 객체 간 통신을 스트림의 제어 채널로 활용한다[7][8]. 스트림 데이터 전송을 위해, 네트워크 관리자 사이에 UDP을 사용한 데이터 채널을 따로 두며, 기본 QoS 제어 기능으로 스트림 관리자와 네트워크 관리자가 수행하는 전송률과 지터 제어, 에러 모니터링 기능의 구현 방안을 기술한다.

4.1 구현 환경

UDP는 비연결형 프로토콜이며 자원 예약에 따른 QoS의 보장이나 전송률 제어나 흐름 제어 방법 등이 제공되지 못한다. 그러나 모든 워크스테이션들이 UDP/IP 프로토콜과 표준화된 인터페이스를 가지고 있으며 프로토콜이 간단하여 오버헤드가 적다는 점과 CM 데이터 전송에 적합한 데이터그램 지향의 프로토콜인 점을 활용하여 스트림 인터페이스의 데이터 채널 구현 방안으로 도입하였다

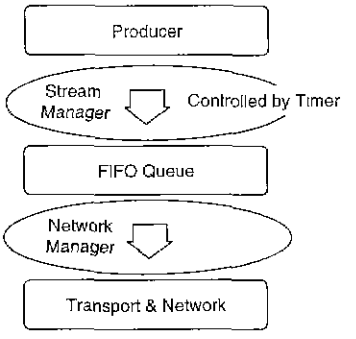
CM 스트림의 지원을 위한 SDU의 헤더 포멧을 (그림 5)와 같이 제안한다 최대 지연이나 지터 등 시간 차원의 지원을 받는 전송을 모니터링 하기 위해 타임스탬프(timestamp) 필드를 도입하며 메세지 식별자를 두어 패킷의 손실이나 에러을 제어를 수행한다. 메세지 크기 필드는 패킷의 전체 길이를 나타낸다.



(그림 5) 메세지 헤더 구조

4.1.1 송신(producer) 데이터 채널

송신측 스트림 관리자는 전송을 위한 버퍼를 두며 이 버퍼에 일정 주기로 패킷을 채운다. 이 주기는 타이머에 의해 제어된다(그림 6). 버퍼는 FIFO 큐 공유 메모리로 큐의 접근 제어는 시스템에서 제공한 세마포어(semaphore)를 사용한다.



(그림 6) 전송률 제어를 위한 스트림 관리자 구조

4.1.2 수신(consumer) 데이터 채널

수신측의 스트림 관리자 역시 공유 메모리 큐를 사용하여 지터 제어를 수행한다 네트워크 관리자가 네트워

크로부터 전송된 패킷을 FIFO 큐에 보관하면 스트림 관리자는 일정 주기로 큐로부터 버퍼를 읽어 네트워크로 보낸다. 이 주기는 타이머에 의해서 제어되며 따라서 지터 값의 완화는 시스템 클럭(clock)의 정확도에 따라 결정된다.

처리율은 연결 설정 과정에서 협상된다. 처리율은 송신 측과 수신 측에서 측정될 수 있다 네트워크 네트워크 내의 클럭들이 동기를 이루고 있으면 지연과 지터는 수신 측에서 측정할 수 있으며 동기를 맞출 수 없으면 지연은 round-trip 시간으로 측정될 수 있다. 에러율은 패킷 헤더 내의 메세지 식별자를 사용하여 수신 측에서 측정한다. 성능 모니터링을 위해 주기적으로 처리율, 왕복 지연, 에러율, 지터 제어를 위한 버퍼의 크기 등 성능 파라메타에 대한 통계적인 정보를 주고 받는다.

4.2 스트림 인터페이스의 구현

4.2.1 스트림 관리자

(그림 7)은 스트림 관리자의 주 알고리즘이다. 계산 객체 및 네트워크 관리자와의 제어 정보의 전달을 위한 채널의 설정과 네트워크 객체와의 스트림 데이터 전송을 위한 공유 메모리 및 세마포어를 초기화 한 후 계산 객체로부터 신호를 기다리는 반복 실행으로 들어간다.

```
Signal(sig_id, sig_func1)   /* from producer/consumer */
Create named pipe as control channel
   / *with computational object*/
open named pipe as control channel
   /* with network manager */
shared memory & semaphore initiate
   /* for stream data transfer to(from) network manager */
event list initiate
   /* for rate control, if Comp. object is stream producer
     for jitter control, if Comp. object is stream consumer
   */
while ()
     : /* wait for signal from application */
endwhile
```

(그림 7) 스트림 관리자 알고리즘

(그림 8)은 스트림 관리자에서, 계산 객체의 내부 메쏘드로부터 전달된 신호를 처리하는 함수이다. 스트림 데이터 채널의 설정이나 해제, 전송의 중지 등은 네트워크 관리자에게 한번의 신호를 보냄으로써 해결된다. 스트림 데이터 전송 제개는 타이머의 도움을 받아 연속적이고 주기적으로 데이터 패킷을 공유 메모리로 쓰거나 공유 메모리로부터 읽는 동작을 반복한다.

```
Function sig_func1
Begin
 Flag = signal  /  from computational object */
 Switch(flag)
  Case connectStreamReq
       Send signal to network manager
       get connection information and return it to
          computational object,
    case resumeStreamReq
       call timer function
    case pauseStreamReq
       send signal network manager to pause writing
          (reading) data packets into(from) network
    case disconnectStreamReq
       send signal network manager to disconnect stream
          data channel
    case
       .
  endswitch
end
```

(그림 8) 스트림 관리자 신호 처리 함수

4.2.2 타이머와 공유 메모리 FIFO 큐

전송률과 지터 제어를 위해 타이머와 공유 메모리를 이용한 FIFO 큐를 사용한다. 전송률 제어의 경우 잘 정의된 주기로 전송할 패킷을 생성해야 하며 이를 위해 사용되는 것이 타이머이다. 타이머는 이벤트들의 배열로 구현되며 배열의 요소는 이벤트 처리 함수와 이벤트가 실행되기까지 남은 시간(*remain_time*), 실행 후 같은 이벤트의 다음 실행까지의 시간 간격(*call_interval*)으로 구성된다(그림 9).

```
struct timer_t {
    int  (*func)();
    int  call_interval,
    int  remain_time,
} timer_str[CONTROL_FUNC_NUM]
```

(그림 9) 이벤트 구조

(그림 10)은 타이머 함수이다 만료값(*remainTime*)은 이벤트 리스트의 맨 앞의 요소의 *remain_time* 값으로 초기화 된다 *timeUnit*은 타이머 시간의 단위 크기(granularity)를 나타낸다. *timeUnit* 시간 만큼 휴면

(sleep) 상태를 유지하다 *remainTime*이 줄어들어 0이 되면 첫번째 타이머 이벤트에 대응되는 이벤트 처리 함수가 호출된다. 처리된 이벤트를 제외한 타이머 이벤트들의 *remain_time* 값이 수정되고 만료된 타이머 이벤트의 *call_interval* 값이 0이 아니면 *remain_time* 값은 *call_interval* 값이 된다. 다시 *remainTime*이 타이머 이벤트 배열의 처음 요소의 *remain_time* 으로 초기화 되고 반복 실행으로 들어간다.

```
RemainTime = diffTime = timer_value of first element
TimeUnit = system time granularity
While()
  Sleep(timeUnit)
  RemainTime -= timeUnit
  If(remainTime is 0)
    call handler function
    correct time_value of all timer events in list using
diffTime
    reschedule timer events
    remainTime = diffTime = time_value of first list
element
  fi
endwhile
```

(그림 10) 타이머 알고리즘

스트림 관리자와 네트워크 관리자간의 통신은 공유 메모리를 사용한다. 공유 메모리 내부를 FIFO 큐 형식으로 하며 큐에 동시 접근 제어를 위해 시스템 V 세미포어를 사용한다.

4.2.3 전송률과 지터 제어

전송률 제어는 스트림 공급자(producer) 측의 스트림 관리자에 의해 이루어진다. 스트림 관리자는 패킷을 생성하여 FIFO 큐에 채우는 함수를 타이머에 등록한다 *call_interval* 값은 패킷의 생성 주기에 맞추고 패킷 생성 함수를 등록한 후 타이머의 주 반복 작업으로 들어간다. 지터 제어는 스트림 수신측에서 네트워크 관리자가 네트워크에서 공유 메모리 큐로 단순히 패킷을 전달하면 스트림 관리자는 타이머를 사용하여 큐로부터 지터를 조절한 후, 패킷을 응용 프로그램이 지정하는 처리 장치로 보낸다. 스트림 관리자는 실행을 시작하기에 앞서 패킷 처리 함수를 타이머에 등록한다 타이머의 제어 시간 간격은 패킷의 처리율에 맞추어진다. 타이머에 등록 후 타이머 시스템의 주 함수를 호출한다.

4.2.4 QoS 파라메터 모니터링

처리율은 송신측과 수신측의 스트림 관리자에서 모두 측정될 수 있으며 각 SDU(Service Data Unit)가 전송될 때 송신자는 시간 값 t_i를 얻는다. 단위 시간당 SDU 수는 다음과 같다.

$$R_i = \frac{1}{t_i - t_{i-1}} \tag{1}$$

따라서 처리율은 식 (2)와 같다

$$TP_i = R_i \times SDUsize \tag{2}$$

이 값은 로그 파일에 등록이 되며 모니터링 함수는 이 값들을 읽어 smoothed 처리율(STP)을 얻을 수 있다.

SDU 지연은 round-trip 시간을 사용하여 측정한다. 각 SDU는 타임 스탬프 값을 포함하며 수신 측 스트림 관리자는 송신 측 타임 스탬프 값과 수신 측 타임 스탬프 값을 모아두어 테이블이 가득 차면 이를 송신 측으로 보낸다

〈표 1〉 수신측 타임 스탬프 테이블

TS_S^i	TS_R^i
TS_S^{i+1}	TS_R^{i+1}
...	...
$TS_S^{i+len-1}$	$TS_R^{i+len-1}$

테이블의 크기는 컴파일 시에 결정되며 큰 테이블은 네트워크 부하를 감소시키나 송신측에서 제어 정보를 얻는 시간 간격을 늘이는 길과를 초래한다. 송신측에서는 타임 스탬프 데이블을 받을 때 또 하나의 타임 스탬프 TS 값을 얻는다. 이때 $k \in [i; i+len-1]$인 각 SDU k에 대한 지연은 다음과 같이 계산되어진다.

$$Delay_k = \frac{1}{2}(TS - TS_S^k - (TS_R^{i+len-1} - TS_R^k)) \tag{3}$$

송신측에서 이러한 값들을 로그 파일에 기록하여 두면 모니터링 함수는 처리율의 경우와 마찬가지로 같은 smoothing 함수를 쓸 수 있다. 에러율은 수신 측에서 계산된다. 유실된 패킷을 알아내기 위해 메시지 식별자가 사용된다. SDU 식별자는 0에서부터 1씩 증가하는 값을 가진다. 수신자는 유실된 패킷과 한계 이상의 지

연을 가진 SDU의 수를 측정하여 에러율을 다음과 같이 계산한다. 식 (4)는 최종 패킷이 손실되었다는 가정하에 얻어진다.

$$ErrorRate = \frac{Nloss + Ndelay}{Ntotal} \quad (4)$$

5 구현된 CM스트림 인터페이스의 적합성 검사

본 절에서는 CBR(Constant Bit Rate) CM 데이터를 사용하여 구현한 CM 스트림 데이터 채널의 적합성을 검사한다. 테스트 환경은 10BaseT LAN 상의 Sun UltraStation(Solaris 2.5) 2대를 사용하였으며, <표 2>의 표준 스트림 타입을 중심으로 스트림 인터페이스의 성능을 측정하였다.

<표 2> 표준 스트림 타입[12]

Stream type	Bandwidth	Jitter	Delay	Traffic type	Error
Standard Video	25 Mbps	10 ms	250 ms	Probabilistic	10^{-7}
Slow Scan Video	1 Mbps	10 ms	250 ms	Probabilistic	10^{-7}
MPEG Video	10 Mbps	1ms	250 ms	Deterministic	10^{-9}
Voice Audio	64 Kbps	10 ms	250 ms	Probabilistic	10^{-3}
HiFi Audio	2 Mbps	5ms	500 ms	Deterministic	10^{-5}

<표 3>은 테스트 데이터로 각 미디어 타입에 대해 SDU의 크기와 전송률을 달리하면서 실제 지연과 처리율을 측정하여 그 분포를 알아볼 수 있도록 구성하였다.

<표 3> 테스트 데이터

Media Type	Case	SDU size (byte)	SDU rate (ms/SDU)	Throughput (bytes/sec)
Voice Audio	Case 1	320	40	8000
	Case 2	400	50	8000
Slow Scan Video	Case 3	1250	10	125000
	Case 4	2500	20	125000
HiFI Audio	Case 5	2500	10	250000

<표 4> 지연과 처리율의 측정 결과

Media Type	Case	Mean SDU Delay (msec)	Mean Throughput (bytes/sec)
Voice Audio	Case 1	10.2	8210
	Case 2	10.9	8030
Slow Scan Video	Case 3	14.5	124600
	Case 4	26.5	125500
HiFI Audio	Case 5	27.6	245500

<표 3>의 표준 CM 데이터를 4절에서 구현한 스트림 데이터 채널로 전송한 경우, 처리율과 지연에 대한 결과를 <표 4>에 담았다. 결과는 표준 미디어 데이터로 수용할 수 있는 수준의 처리율을 보였다.

QoS 파라메타 중 지연, 지터, 에러률 제어하기 위한 방안으로 SDU 크기, 전송율에 따른 각 파라메타 값을 측정하여 보았다.

<표 5>는 패킷의 크기와 전송 지연과의 관계를 나타낸다. SDU 율을 20 msec/SDU로 고정하고 지연을 측정한 결과, 패킷의 크기가 클수록 긴 지연 시간을 갖는 것으로 나타났다. 이는 큰 패킷 일수록 사용자 영역에서 커널 영역으로, 네트워크 어댑터로 복사에 시간이 많이 소요됨을 보여준다. 또한 최대 이더넷 MTU는 1500byte이므로 이를 초과하는 SDU는 분할과 재합성 과정이 요구되므로 지연 시간이 더 길어진 것으로 판단된다. SDU 율과 지연과는 별 상관 관계가 없는 것으로 나타났다.

<표 5> SDU의 크기에 따른 지연값

Case	SDU size (byte)	Mean SDU Delay (msec)
Case 1	320	10.1
Case 2	800	13.9
Case 3	1750	24.0
Case 4	2000	26.5
Case 5	2500	30.2

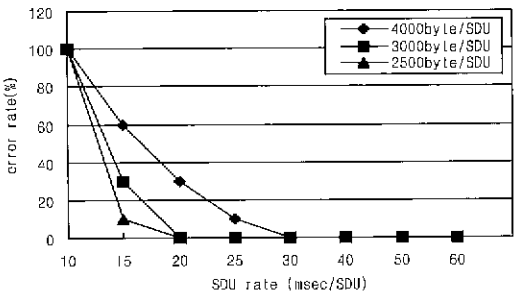
<표 6>은 지터값과 SDU 크기 및 SDU 율(msec/SDU)과의 관계를 보여주는 것으로 지터값은 SDU의 크기 보다는 SDU 율에 의존하는 것으로 나타났다. <표 6>은 전송 시작부터 40개의 SDU만을 평균한 값으로 40번째 이후, 즉 일정 시간 경과 후에는 안정된 상태, 즉 지터값이 1msec 이하로 유지되는 것으로 나타났다.

<표 6> SDU 크기와 전송률에 따른 지터 값

Case	SDU size (byte)	SDU rate (ms/SDU)	Mean Jitter (msec)
Case 1	400	50	1.1
Case 2	2500	20	10.9
Case 3	4000	30	1.2

에러율은 SDU의 크기와 SDU율에 모두 영향을 받는 것으로 나타났다. SDU 크기가 4000 byte일 경우, SDU

율이 30 msec/SDU면 에러율이 0이었으나 20msec/SDU에서는 30%, 15 msec/SDU에서는 60% 이상으로 측정되었다. SDU의 크기가 2500 byte에서는 20msec/SDU에서 에러율이 0, 15msec/SDU에서 30%로 나타났다(그림 11). 따라서 SDU의 크기가 작을 경우 SDU 간 전송 간격이 조밀해도 무방하나(최소 10msec/SDU 이상) SDU의 크기가 클 경우 SDU율에 크게 영향을 받는 것으로 나타나 SDU율과 크기의 조절에 의해 에러율의 조절이 가능한 것으로 분석된다.



(그림 11) SDU의 크기와 SDU 율에 따른 에러율

표준 CM 데이터를 사용하여 구현 스트림 인터페이스의 적합성 여부와 QoS제어를 위한 방안으로 지연, 지터, 에러율과 SDU의 크기 및 SDU의 생성율과의 관계를 살펴보았다. 사용한 표준 오디오, 비디오 데이터에 대체 요구 처리율에 근접하는 결과를 보여주었으며, SDU의 크기와 생성율에 따라 QoS의 제어가 가능함을 알 수 있었다.

6 결 론

본 논문에서는 분산 응용에서 요구되는 QoS를 고려한 CM 데이터 스트림 인터페이스를 지원하는 계산 객체 모델을 설계하고, 스트림 인터페이스의 구현 방안을 제시하였다.

제안 객체 모델은 CORBA의 객체 간 통신을 스트림 데이터의 제어 채널로 사용하며 CM 데이터의 전송을 위해 트랜스포트 프로토콜을 사용한 스트림 데이터 채널을 사용한다. QoS 제어와 스트림 인터페이스 및 하부 자원의 관리를 위해 스트림 관리자와 네트워크 관리자를 둔다. QoS의 지원을 위해, 스트림 인터페이스에 전송율, 지터 등 응용 QoS 정보를 스트림 인터페이스에 포함하도록 하였다.

본 논문에서는 객체 모델의 설계와 함께 제안 스트림 인터페이스의 구현 방안을 제시하였다. 이를 위한 기반 플랫폼으로 OMG CORBA를 사용하였다. CORBA의 객체 간 통신을 스트림의 제어 채널로 사용하였으며 스트림의 데이터의 전송을 위해 인터넷 환경의 UDP/IP 프로토콜을 사용하는 방안을 제안하였다. 또한 FIFO 큐와 타이머를 도입하여 전송률, 지연, 지터 제어 등 QoS를 고려한 스트림 인터페이스의 구현 방안을 제시하였다.

구현 스트림 인터페이스의 테스트를 위해 CBR CM 데이터를 사용하여 정량적인 분석을 행하였다. 테스트의 결과, 측정한 CM 데이터의 경우, 요구 처리율에 근접하는 결과를 얻었으며, SDU의 크기, 생성율에 따른 지연, 지터, 에러율을 조절하는 방안을 얻었다.

본 연구를 통해 스트림 인터페이스를 기존의 오퍼레이션 인터페이스와 같이 사용자 수준에서 부가적인 자원의 관리없이 객체의 인터페이스로 다루는 방안을 제안하였다. 본 연구의 결과는 QoS의 제어를 포함한 CM 스트림 데이터 전송을 요구하는 다양한 품질의 분산 응용 시스템의 개발에 활용이 가능할 것으로 판단된다. 본 연구를 바탕으로 향후 연구에서는 멀티미디어를 위한 다양한 고속 전송 프로토콜들을 사용하여 다양한 QoS의 지원 및 실시간 응용을 대상으로 한 CM 데이터의 전송에 대한 연구가 추진되어야 할 것이다.

참 고 문 헌

[1] P. Adcock, N. Davies, G. S. Blair, "Supporting Continuous Media in Open Distributed Systems Architectures," DCE - The OSF Distributed Computing Environment, International DCE Workshop, Springer-Verlag, 1993

[2] C. Aurrecoechea, "A. Campbell and L. Hauw, A Review of Quality of Service Architectures." ACM Multimedia Systems Journal, November, 1995

[3] Wulfdieter Bauerfeld, Horst Westbrok, "Multumedia Communication with High-speed Protocols," Computer Networks and ISDN Systems 23, pp.413-151, 1991

[4] A. Campbell, G. Coulson, F Garcia and D. Hutchison, "Resource Management in Multimedia Communication Stacks," Proc. 4th IEE Conference on Telecommunications, Manchester, UK, April 1993.

[5] A. Campbell, G. Coulson and D. Hutchison, "A

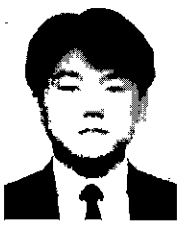
Quality of Service Architecture," ACM Computer Communication Review, April 1994.
[6] G. Coulson, G. S. Blair, N. Davies, N Williams, "Extensions to ANSA for Multimedia Computing," Computer Networks and ISDN Systems(25), pp305-323, 1992
[7] ObjectBroker Reference Manual, DEC, August, 1994
[8] ObjectBroker2.5 Release Notes, DEC, August, 1994
[9] Luca Delgrossi, Christian Halstrick, Dietmar Hehmann, Ralf Guido Herrtwich, Oliver Krone, "Media Scaling in a Multimedia Communication System," ACM Multimedia Systems, pp.66-75, March 1994
[10] B Furht, "Multimedia Systems : An Overview," IEEE Multimedia, Vol.1, No.1, Spring 1994.
[11] Interactive Multimedia Association, Multimedia System Services Version 1.0, (HuwlettPackard, IBM, SunSoft), June 1993
[12] LeGall, "MPEG : A Video Compression Standard for Multimedia Applications," Comm ACM, Vol. 34, No.4, Apr. 1991
[13] Carsten Vogt, "Quality-of-Service Management for Multimedia Streams with Fixed Arrival Periods and Variable Frame Sizes," ACM Multimedia Systems, 3, 1995, pp.66-75
[14] Klara Nahrstedt, Jonathan M. Smith, "The QOS Broker, IEEE Multimedia," Vol.2, No.1, Spring, 1995
[15] Klara Nahrstedt, Ralf Steinmetz, "Resource Management in Networked Multimedia Systems," IEEE Computer, Vol.28, No.5, May 1995
[16] ISO/IEC JTC1/SC21 N7054 : ISO Working Draft on Basic Reference Model of Open Distributed Processing - Part 2 : Descriptive Model, Jul., 1992
[17] ISO/IEC JTC1/SC21 N7055 : ISO Working Draft on Basic Reference Model of Open Distributed Processing-Part 3 : Prescriptive Model, Aug., 1992
[18] The Common Object Request Broker : Architecture and Specification, OMG Document 93.XX.YY, Revision 1.2, December 1993
[19] TINA DPE Overall Concepts and Principles of TINA, TB_MDC.018_1.0_94, February 1995
[20] Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann and Jan Gecsei, "Distributed Multimedia and QOS : A Survey," IEEE Multimedia, Vol.2, No.2, Summer, 1995

송 병 권

e-mail : bksong@bukak.seokyeong.ac.kr
1984년 고려대학교 전자공학과 졸업(공학사)
1986년 고려대학교 대학원 전자공학과(공학석사)
1995년 고려대학교 대학원 전자공학과(공학박사)
1984년~1997년 삼성종합기술원 선임연구원
1995년~현재 서경대학교 정보통신공학과 교수
관심분야 : High-Speed Network, 분산처리시스템, Mobile Computing

진 명 숙

e-mail : jnms@dove.kyungin-c.ac.kr
1990년 고려대학교 전자전산공학과 졸업(공학사)
1992년 고려대학교 대학원 전자공학과(공학석사)
1997년 고려대학교 대학원 전자공학과(공학박사)
1996년~현재 경인여자대학 멀티미디어정보전산학부 조교수
관심분야 : 분산 시스템, 멀티미디어 컴퓨팅, 객체 지향 시스템



김 건 웅

e-mail : kgu@mail.mmu.ac.kr
1990년 고려대학교 전자전산공학과 졸업(공학사)
1994년 고려대학교 대학원 전자공학과 졸업(공학석사)
1998년 고려대학교 대학원 전자공학과 졸업(공학박사)
1999년~현재 목포해양대학교 해양전자통신공학부 전임강사
관심분야 : 네트워크 프로토콜, 망 관리 시스템, 지능망, 정보망