표준 개방 분산환경에서의 멀티미디어 스트림 서비스의 설계

김종현, 김재일
동의공업대학 전자계산과
jhkim,kji@dit.ac.kr

Design of Multimedia Stream Service on a Standard Open Distributed Environments

Kim, Jong-Hyun, Kim, Jae-Il
Dongeui Institute of technology
jhkim,kji@dit.ac.kr

요 약

OMG에서 제안된 CORBA는 표준 개방 분산환경의 플래트폼을 제공하며 객체지향적인 방법으로 최근 주목을 받고 있다. 일반적으로 CORBA는 원격 데이터베이스 질의 등과 같은 요구/응답형의 응용에 잘 적용이 된다. 그러나 과도한 처리부담, 실시간 처리 기능과 QOS 지원 등의 결여로 멀티미디어 스트림과 같은 연속미디어의 처리에는 부적합하다. 본 논문에서는 실시간 제약을 갖는 멀티미디어 스트림의 효율적인 전송을 위하여 기존의 CORBA를 확장하여 멀티미디어 서비스를 설계한다.

I. 서 론

최근 고속 네트워크와 멀티미디어 컴퓨터 기술의 발전은 화상회의나 주문형 비디오 시스템(VOD) 등 새로운 형태의 멀티미디어 응용의 수행을 가능하게 하고 있다[1].

일반적으로 네트워크를 기반으로 하는 분산 시스템 환경에서 클라이언트, 서버 구조의 응용 프로그램을 구현할 때 이질성(heterogenity), 이식성(portability), 상호연동성(interogenity) 등으로 인해 많은 어려움이 있다. OMG(Object Management Group)에 의해 제안된 CORBA는 이러한 문제점을 해결하며, 객체지향적인 방법을 제공하므로서 표준 개방 분산 환경의 플래트폼을 제공하며 최근 주목을 받고 있다[2,3].

그러나 현재 대부분의 구현된 CORBA는 성능의 최적화, 실시간 기능과 QOS 기능의 결여로 분산 멀티미디어 응용에 효율적으로 적용되기가 어렵다. 따라서 표준 개방 분산 환경을 제공하는 CORBA 환경에서 멀티미디어 응용을 수행하기 위해서는 멀티미디어 스트림의 효율적인 처리를 위한 지원이 필요하다[4].

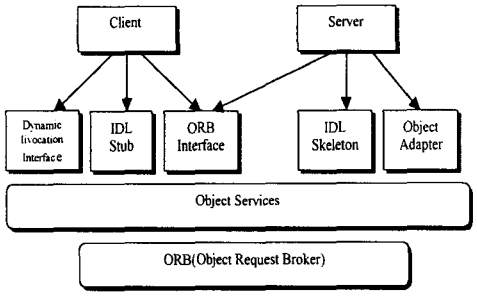
본 논문에서는 기존의 CORBA 환경을 확장하

여 멀티미디어 스트림을 효율적으로 처리하기 위한 서비스를 설계한다. 또한 사용자 수준의 QOS 명세를 정의하고, RM(Rate Monotonic) 정적 우선순위를 사용하는 주기적 스레드 모델에 기반한 스트림의 스케쥴링과 허용제어 정책을 제시한다.

II. 연구 배경

2.1 CORBA

CORBA 2.0은 위치의 투명성(location transparancy), 구현의 투명성(implementation trancparency) 등 프로그래머에게 표준화된 객체지향 분산 응용 개발 환경을 제공한다. (그림 1.)은 OMG에 의해 제안된 CORBA 2.0 표준 구조를 보여준다.



(그림 1) CORBA의 구조

1) Interface Definition Language(IDL)

CORBA IDL은 클라이언트 객체에서 호출되는 모든 서버 객체의 구현 부분에 대한 인터페이스를 정의하는 선언적 언어이다. 객체간의 인터페이스만 정의하면 구현 언어에 대해서는 독립성을 제공한다.

2) ORB(Object Request Broker)

클라이언트와 서버 객체간의 공통 통신 버스로 응용 객체간의 투명한 통신을 가능하게 하는 프레임워크를 제공한다.

3) Object Service

ORB 내부의 저차원 객체들에 대한 인터페이스로 구성된다. OMG에서 정의한 object service는 naming, events, life cycle, transaction, persistency, concurrency control 등이 있다.

2.2 실시간 스트림 서비스

현재의 CORBA 2.0구현은 원격 데이터베이스 질의 등의 이산(discrete) 데이터의 상호 작용에 적합하도록 설계되었다. 따라서 과도한 내부 처리 부담, 실시간 처리를 위한 기능, QOS 지원 메커니즘 등이 결여되어 있다. 특히 멀티미디어 스트림을 지원하기 위한 연속 매체의 상호작용에 관한 시멘틱을 제공하지 못하고 있다.

따라서 다음과 같은 사항을 바탕으로 분산 멀티미디어 응용을 CORBA 환경하에서 효율적으로 지원하기 위한 실시간 스트림 서비스를 설계한다.

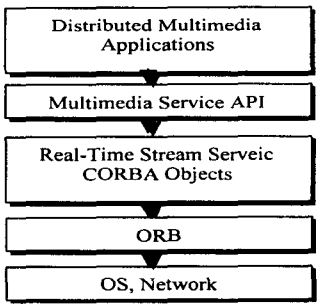
1) CORBA의 장점을 최대한 이용
2) 스트림 간의 상호작용을 지원하기 위한 시멘틱 정의
3) 표준 실시간 운영체제(POSIX)의 실시간 기능의 활용[5]
4) 고성능 네트워크(TCP/IP, ATM)의 활용
5) CORBA 처리부담의 최적화

III. 멀티미디어 서비스의 설계

3.1 멀티미디어 서비스의 계층적 구조

(그림 2) CORBA에 기반한 실시간 스트림 서비스의 계층화된 구조를 보여준다. 최상위 계층에는 화상회의나 주문형 비디오(VOD) 등과 같은 다양한 멀티미디어 응용이 존재한다.

두번째 계층에서는 상위계층인 멀티미디어 응용을 구성하는 멀티미디어 서비스 API를 제공한다. 이 계층은 객체지향적 방법으로 설계되며 멀티미디어 데이터를 전송하거나 다양한 디바이스들을 추상화한 클래스로 구성된다.



(그림 2)멀티미디어 스트림 서버의 계층적 구조

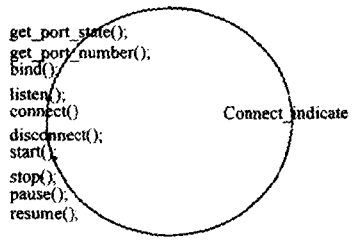
세번째 계층은 멀티미디어 응용을 지원하기 위

한 확장된 CORBA 서비스 객체로 구성된다. 확장된 CORBA 객체들은 name service, 스트림의 전송, 스트림 제어, 세션관리, 실시간 스케줄러와의 상호작용 등의 기능을 수행시간에 담당한다. 이 계층의 분산의 투명성(distribution transparency)은 하위 계층인 CORBA ORB(Object Request Broker)에 의해 자연스럽게 해결된다. 최하위 계층은 분산 멀티미디어 응용을 물리적을 연결하는 네트워크(TCP/IP, ATM)와 실시간 운영체제(POSIX, Solaris)가 존재한다. 이 계층은 실시간 기능을 제공하기 위해 세번째 계층의 일부 CORBA 서비스 객체들과 밀접하게 상호작용한다.

3.2 멀티미디어 서비스 객체

1) Port

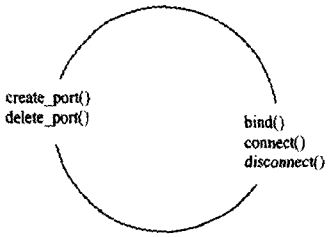
Port는 스트림 데이터를 전달하기 위한 end-to-end 통신 인터페이스이다. Port의 이름은 CORBA name service에 의하여 분산된 시스템내에서 구분이 되는 유일한 이름이다. 각 Port는 직접 접근이 가능한 스트림 버퍼를 유지하며, 버퍼의 크기, 버퍼 등의 QOS는 연결 설정시 인수로 명시할 수 있다. 이러한 QOS 명세는 스케줄링의 정보로 이용된다. (그림 3)은 Port의 인터페이스 정의를 나타낸다.



(그림 3) Port 객체의 인터페이스

2) Factory

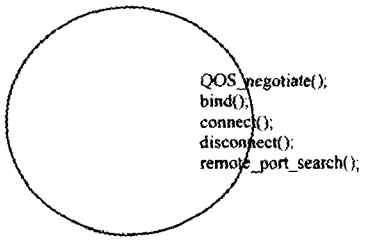
Factroy 객체는 Port 객체의 생성과 관리를 담당한다. 멀티미디어 응용은 연결설정시 제일 먼저 Factory 객체에 스트림 전송을 위한 자신의 Port 객체의 생성을 요구하게 되며, Factory 객체는 새로운 Port 객체를 생성하게 된다. (그림 4)는 Factory 객체의 인터페이스 정의를 나타낸다.



(그림 4) Factory 객체의 인터페이스

3) Connection Manager

Connection Manager 객체는 Port 들간의 실제적인 연결 설정과 QOS 협상을 담당한다. 서로 다른 두 Port 사이의 연결설정을 위하여 연결요청(connection request)이 오면 Connection Manager는 허용제어 후 실제적인 연결설정을 한다. (그림 5)는 Factory 객체의 인터페이스 정의를 나타낸다.



(그림 5) Connection Manager 객체의 인터페이스

3.3 QOS의 정의

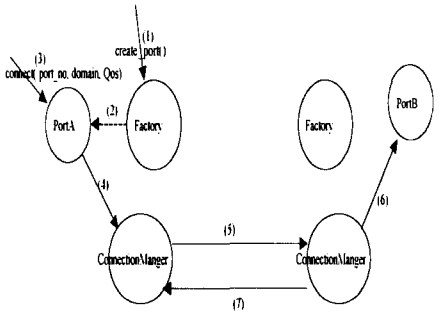
연결 설정시 사용자에 의해 명시되는 QOS는 아래의 구조를 가진다.

```
typedef  struct {
        RTS_proto protocol;
        short buffer_size;
        short buffer_rate;
        short actpp;
} RTS_qos ;

typedef enum {
        RTS_tcp,
        RTS_udp,
        RTS_all3, RTS_all4,
        RTS_all5
} RTS_proto;
```

RTS_proto 인수는 멀티미디어 스트림 데이터를 전달할 다양한 통신 프로토콜을 지정한다. RTS_buffer_size는 Port 내부의 스트림 버퍼의 크기를 지정한다. RTS_buffer_rate는 포트사이의 스트림 데이터 전달시 요구되는 속도율을 초당 프레임수로 명시한다. 즉 네트워크의 대역폭을 말한다. 예를 들면 full-motion 비디오 데이터를 전송할 때, buffer_size가 프레임의 최대 크기라면 초당 30 프레임의 buffer_rate가 요구된다. atcpp(approximate computation per period/msec)는 응용프로그램에서의 데이터의 처리와 통신 프로토콜 수행에 의한 CPU 소모 시간을 합한 값을 나타낸다. 이값은 통신 연결시 사용자에 의해 지정되는 근사값이며, 허용제어시의 중요한 정보로 이용된다.

3.4. 연결 설정 과정



(그림 6) QOS 협상과 연결 설정과정

(그림 6)은 QOS 협상과 연결설정 과정을 보여준다.

(1),(2) 멀티미디어 응용은 Factory 객체에 클라이언트/서버 각각의 Port 객체를 요청하여 생성한다.

(3) 멀티미디어 응용은 Port A에게 Port B로 연결을 요청(connection request)한다. 이때 Port B가 위치하는 도메인 이름, Port 번호, QOS를 인자로 넘긴다.

(4) Port A는 응용프로그램으로부터의 연결요청을 ConnectionManager에 넘기고, QOS 협상을 한다. 이때 응용프로그램에서 요청한 QOS를 만족할 경우 정상적으로 연결설정이 이루어 진다.

(5),(6) local ConnectionManger는 도메인 이름과 port 번호를 이용하여 remote ConnectionManger에게 연결설정을 요구하여 remote ConnectionManger는 해당 port를 찾는다.

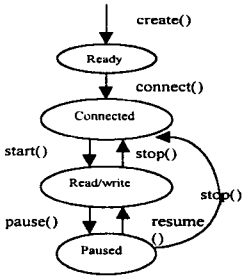
(7) Local ConnectionManger와 remote ConnectionManager는 서로 협력하여 필요한 정보를 전달하고 두 Port사이의 연결관계를 형성한다.

3.5. 스트림 제어

(그림 7)은 Port의 상태 전이도를 나타낸다. start() 연산의 수행은 실제 스트림 데이터를 주고받기 위한 스레드를 수행하기 위하여 운영체제에 의해 실시간 스레드를 생성(fork)하는 상태 전이를 말한다. 이것은 현재 포트가 연결되어 상태에서만 적용된다.

stop() 연산은 Port 객체가 생성된 실시간 스레드를 종료하고 connected 상태로 돌아간다. 이것은 스트림 데이터가 전송되고 있는 상태(read/write)나 connected 상태에서 모두 적용된다. Port가 read/write 상태에서 pause() 연산이 적용되면 pauesed 상태로 바뀌게 되며, resume() 은 반대의 역할을 한다.

RTS_state RTS_port::strat()
RTS_state RTS_port::stop()
RTS_state RTS_port::pause()
RTS_state RTS_port::resume()



(그림 7) Port의 상태전이도

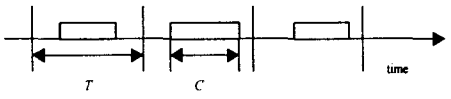
IV. 스트림 스케줄링과 허용제어

4.1 실시간 스트림 스케줄링

각 원격 호스트에 존재하는 멀티미디어 스트림 서버스에는 각각 시간제약을 가지는 다수의 스트림들이 수행을 기다리게 된다. 이때 스트림 스케줄러는 각 스트림의 시간 정보를 유지하여 모든 스트림이 시간제약을 만족하도록 스케줄링을 해야 한다.

본 논문에서의 스케줄링 방법은 자원의 활용도와 가용한 용량에 기반한 CBS(Capacity Based Secheduling)방법을 사용한다[8]. CBS는RM(Rate Monotonic) 정적 우선순위에 기반한 주기적 스레드 모델로 주기적 스트림 스레드(타스크)는 주기에 따라 우선순위가 고정된다. 즉, 가장 작은 주기를 가진 스레드가 가장 높은 우선순위를 가지게 된다.

RM에 기반한 스케줄링은 시스템이 예측할 수 없는 동기화나 통신 부하의 경우에도 EDF(Earlist Deadlein First) 스케줄링 알고리즘에 비해 안정된 성능을 보장한다. 높은 우선순위를 가진 타스크는 수행이 보장되는 반면, 낮은 우선순위의 타스크는 항상 시스템 부하때 종료시한을 어기게 된다. 따라서 높은 우선순위의 타스크의 수행보장이 중요한 경우 시스템이 과부하된 상태에서도 안정된 성능을 보장한다. 따라서 RM에 기반한 기법은 빈번한 문맥교환(context switching)으로 인한 동기화나 통신부하를 피할 수 없는 분산 멀티미디어 환경에 적합한 스케줄링 방법이다.



C : Computation time of Thread
T : period
(그림 8) 주기적 타스크

여기서 C는 한 주기내에서의 스레드의 CPU 소모시간이며, T는 CPU 소모시간을 포함하는 주기이다. 주기적 스레드에는 스트림 처리를 위한 응용프로그램 처리코드(processing code)와 통신 프로토콜의 처리 코드(protocol code)가 모두 포함된다.

이때 우선순위 P는 사용자가 명시한 QOS로 직접 부터 얻을수 있다.

$$P = buffer\_rate$$

C, T, P의 값이 주어졌을 때 POSIX 실시간 스레드에 기반한 스케줄러의 pseudo 코드는 다음과 같다.

```
real_time_thread(void *data)
   cast data to approach structure;
   block unwanted signal;
   set priority to P;
   create periodic timer T1;
   create quantom timer T2;
   while(1) {
      set timer t2 to C;
      if (type == RT_read) {
         read_from_network(buffer,
                           buffer_size);
          application_data_handler(buffer,
                           buffer_size);
       } else
         application_data_handler(buffer,
                           buffer_size);
         write_to_network(buffer,
                           buffer_size);
    }
    nanosleep(T);
}
```

(그림 9) 실시간 스케줄러의 pseudo 코드

4.2 허용제어(admission control)

허용제어는 연결 시간동안 CPU, 메모리와 네트워크 자원을 대상으로 가용성을 검사하여, 새로운 연결 요구를 만족할 때 연결을 허용한다. 만약 그렇지 않을 경우 해당 스레드의 연결은 거부되고 적절한 예외조치나 일정한 주기를 연장하여 새로운 우선순위로 다시 스케줄된다. 일단 새로운 연결이 성공하면 이에 필요한 자원들은 예역되어 연결이 종료될 때까지 계속 사용된다.

(1) CPU

CPU 허용제어를 위한 주기(T)와 CPU의 소모시간(C)는 사용자 수준의 QOS 명세로부터 직접 얻을 수 있다. T는 다음과 같이 해석된다.

$$T = 1/buffer\_rate$$

같은 방법으로 CPU 계산시간 C는 다음과 같이 얻을 수 있다.

$$C = actpp$$

여기서 *actpp*는 사용자가 명시한 하나의 주기내의 스레드의 근사 CPU 소모시간(milisecond)을 말한다. 위에서 얻은 분석으로부터 CPU 허용제어는 아래의 방정식을 만족하면 새로운 스트림 연결은 허용된다. 그러나 만족하지 못할 경우 연결은 거부되며, 해당 스트림은 주기를 일정량 연장하여 낮은 우선순위로 다시 스케쥴링 된다.

$$\sum_{i=1} \frac{C_i}{T_i} \leq 0.88$$

(2) 메모리

메모리 자원의 요구는 각 호스트에서 사용자에 의해 제공되는 *buffer_size*의 요구량을 말한다. 이는 각 Port에 존재하는 버퍼 크기이다. 연결시간 동안 스트림이 요구하는 양의 버퍼가 할당된다. 만약 시스템 메모리의 부족으로 할당이 불가능하면 연결은 거부된다.

(3) 네트워크

네트워크 자원의 허용제어에서 라우터 기반의 TCP/IP프로토콜일 경우는 자원의 사전 할당이 불가능하므로 best effort 방식을 사용한다. ATM의 경우 ATM 적응층 인터페이스에서 제공하는 native API를 이용한 사전 자원 할당이 가능하다. 주어진 연결당ATM 네트워크의 최대 대역폭의 요구는 QOS 명세의*buffer_rate*와 *buffer_size*로 부터 다음과 같이 표시된다.

$$B=(8Xbuffer\_sizeXbuffer\_rate/1000Kbps)$$

최대 대역폭(peak bandwidth)에서 보낸 패캣의 평균 크기(average packet size)인 mean burst length(*mbl*)은 다음과 같이 표시된다.

$$mbl=(8Xbuffer\_size)/1000+1(Kbits)$$

V. 결론 및 향후 과제

본 논문에서는 표준화된 개방 분산환경을 제공하는 CORBA를 기반으로 멀티미디어 스트림 서비스를 설계하였다. 설계된 스트림 서비스는 end-to-end 스트림 전송을 위한 화상회의 시스템 등에 적용될 수 있을 것이다. 현재 CORBA 2.0 표준 명세를 제공하는 IONA사의 Orbix 2.0[9]과 Solaris Realtime Class를 이용한 프로토타입 시스템을 구현중에 있다. 향후 실시간 처리 기능 및 허용제어 등의 성능 평가와 multipoint 통신 기능 등의 확장에 관한 연구가 필요하다.

참고 문헌

1) Guojun Lu, 1996, Communication and Computing for Distributed Multimedia Systems, Artech House, Boston. London.
2) Jon Siegel, 1996, CORBA Fundamental and Programming, John Wieley & Sons, Inc.
3) Object Management Group Inc., 1994, The Common Object Request Broker : Architecture and Specification, Revision 2.0, OMG TC Document.
4) OMG, 1996, Control and Management of A/V Streams Request For Proposal. OMG TC Document.
5) Khanna, S., Sebree, M., and Zolnowsky J., 1992, Realtime Scheduling in SunOS 5.0, Proc. USENIX Winter Conf.
6) Coulson, G., Blair, G.S., and Robin, P.,1993, Micro-kernel Supprot for Continus Media in Distributed Systems, An Internal Report No. MPG-93-04, Department of Computing, Lancaster University, UK.
7) T.H. Yun, J. Y. Kong and J. Won-Ki Hong, 1997, A CORBA-based Distributed Multimedia System", Proc. Of 1997 Pacific Workshop on Distributed Multimedia Systems, pp. 1-8.
8) Mercer, C.W., Savage, S., and Tokuda, H., 1993, Processor Capacity Reservers for Multimedia Operating Systems, an Internal Technical Report No. CMU-CS-93-157, School of Computer Science, Carnegie Mellon University
9) IONA, 1997, Orbix Programmer's Guide