OSI 관리 기술을 이용한 CORBA 트래픽 모니터링 프록시 설계

박 재 성*, 송 왕 철*

The Design of Proxy for CORBA Traffic Monitoring Using OSI management technology

Jae-Seong Park*, Wang-Cheol Song*

요 약

시스템 관리 영역의 한 부분인 트래픽 모니터링은 시스템의 상황을 파악하는데 있어서 매우 중요하다. 현재 CORBA 표준화를 담당하는 OMG에서는 CORBA 시스템 관리의 표준화를 진행 중에 있으며, 많은 회사와 연구소들은 필요에 의해 CORBA에서의 시스템 관리에 대해 개발하고 연구하고 있다.
본 논문에서는 표준화된 OSI 관리 방식을 통하여 CORBA 트래픽 모니터링을 하기 위한 프록시를 설계하였다. 이를 위해, CORBA 자원을 관리하기 위한 6개의 파라미터를 정의하고, 이들을 관리객체로 만들었다. 본 논문에서 모니터링 시스템의 구성은 CORBA 서버, 프록시 객체, 그리고 MIB로 되어 있다. CORBA 서버는 서비스 제공자, 프록시 서버, 그리고 이벤트 서버로 구성되었다. 프록시 객체는 CORBA 객체인 프로세스로 작동을 하며, 프록시 클라이언트, 이벤트 클라이언트, IPC 서버로 구성되었다.

Abstract

Traffic monitoring, a part of the system management, is a vital function for the proper operation of a system in use. Currently OMG has been trying to standardize CORBA system management. Besides, many companies and research laboratories have been developing and studying CORBA system management.

* 제주대학교 정보공학과
접수일자 : 1999년 1월 14일

In this paper, we have designed the proxy to monitor the CORBA traffic using the OSI management technology. To manage CORBA traffic resources, 6 parameters have been made into managed objects. The monitoring system consists of a CORBA server, a proxy object and an MIB. The CORBA server is made up of a service provider, a proxy server, and a event server. The proxy object acts as a process of a CORBA object, and is made up of a proxy client, a event client, and an IPC server.

I. 서 론

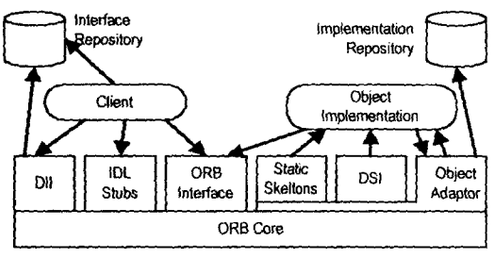
몇 년 전부터 컴퓨터 분야에서는 응용 프로그램의 통합이나 분산 처리의 해결책으로 미들웨어라는 분산 객체 시스템을 채택하고 있다. 최근 OMG(Object Management Group)에서 CORBA(Common Object Request Broker Architecture)라는 표준안을 만들면서 주목받기 시작했다. 이런 분산 환경에서는 매우 많은 객체들이 동작하게 되는데, 이들을 관리해줄 표준안 기반의 접근이 거의 없었다[1]. 그리고, 기존의 시장에서는 CMIP(Common Management Information Protocol)과 SNMP(Simple Network Management Protocol) 기반의 제품들이 나오고 있었다. 이에 분산 환경을 위한 관리 표준안이 필요하게 되고, 현재 연구와 제품개발이 활발히 진행 중에 있다[2,3,4,5]. 그러나, 대부분의 관리 프로그램들은 특정 ORB(Object Request Broker)에서의 관리에 국한된다는 것이다. 그리고, 트래픽 모니터링은 관리함에 있어서 매우 중요한 부분 중에 하나로 시스템의 특성이나 정보를 얻기 위한 것 외에도 시스템과 관련된 모든 내용을 파악하여 최적의 환경을 구성하거나 관리하기 위한 정보를 얻는데 있다. 본 논문에서는 CORBA에서 트래픽 모니터링하는 것에 있어서 표준화된 OSI(Open System Interconnection) 관리 방식을 적용하기 위한 프록시를 설계하였다. 그리고, 각각의 트래픽 정보를 저장하기 위하여 트래픽 파라미터를 추출하고, OSI 관리에서 관리객체(MO: Managed Object)들을 정의하고 구성하였다.

II. CORBA

1. CORBA 개요

OMG에서는 특정 네트워크 아키텍처나 운영체제 아키텍처에 의존하지 않고, 보다 추상화 레벨이 높은 단일의 아키텍처를 제공하는 것에 의해 이식성과 상호 운용성을 실현하고 있다[6].

ORB는 클라이언트에서 서버로의 요구를 넘겨주는 역할을 한다. ORB는 그림 1에서 보여주듯이 ORB 코어, OA(Object Adopter), ORB 인터페이스, 동적 기동 인터페이스, 동적 스켈레톤 인터페이스, IDL(Interface Definition Language), 스터브(Stub), 정적 스켈레톤(Skeleton), 인터페이스 저장소(Interface Repository) 및 구현 저장소(Implementation Repository)로 구성된다[7].



DII: Dynamic Interface Invocation
DSI: Dynamic Skeleton Invocation
ORB: Object Request Broker
IDL: Interface Definition Language

그림 1. ORB 기능구조
Fig. 1 The framework of ORB

전체적인 메시지의 흐름을 보면, 클라이언트는 요구 메시지를 객체 구현 쪽으로 발행해서, 결과를 얻어 오게 된다. 클라이언트와 서버는 ORB 인터페이스를 통해서 ORB 관련된 부분을 초기화한다. 클라이언트가 메시지를 DII(Dynamic Interface Invocation)나 IDL 스터브를 통해 ORB로 보내면, ORB는 서버측으로 메시지를 넘겨주게 된다. 서버측에서는 BOA(Basic Object Adapter)에서 메시지를 받아서, 적당한 서버를 찾아서 정적 스켈레톤이

나 DSI(Dynamic Skeleton Interface)통해서 메시지를 넘겨주게 된다. 서버에서 응답 메시지를 보낼 경우에도 앞 순서의 반대로 이루어지게 된다.

2. GIOP

본 절에서는 CORBA 가운데에서 중점적으로 다룰 GIOP(General Inter-ORB protocol) 부분을 살펴보겠다. 이 부분은 서로 상이한 CORBA 제품간의 접속을 위한 데이터 표현 방식, 메시지 포맷, 그리고 전송 방식에 대해 규정하고 있다. 각각의 클라이언트에서 발생하는 메시지는 GIOP에서 정한 메시지 타입에 맞추어서 ORB 상으로 보내지게 된다[8].

2.1 GIOP 메시지 헤더

본 논문에서는 GIOP 메시지들에서 트래픽 정보를 추출하였다. 그래서, GIOP 스팩 중에서 본 논문에서 사용할 메시지 포맷에 대해 알아보겠다. 먼저, 그림 2는 GIOP 메시지 헤더부분 나타낸 것이다. 이 부분이 모든 메시지의 기본적인 부분으로 GIOP에서의 메시지의 기본적인 정보를 포함하고 있다. 먼저 GIOP 메시지임을 표시하는 Magic이라는 부분이 있다. 이 곳에는 4 바이트 크기의 GIOP라는 글자가 대문자 형태로 인코딩되어 들어가 있다. 그리고, GIOP 메시지 헤더 부분에 Version부분에는 GIOP 버전 v1.1, 또는 v1.0을 가리킨다. 이는 CORBA의 버전과는 일치하지는 않지만, 현재 CORBA 2.0에서는 GIOP 1.0과 1.1를 사용할 수 있다. Byte order는 바이트 정렬에 관련 된 것으로 v1.0에의 바이트 정렬이 big-endian 형태인지, little-endian으로 되어 있는지를 boolean형태로 표시하고 있다. 반면에 GIOP v1.1에서는 플래그라는 8비트 옥테트 부분의 최상위 비트에 바이트 정렬을 표시하고 있다. 두 번째 최상위 비트에는 Fragment의 유무를 표시하는 부분이고, 나머지는 현재 예약되어 있다. 그리고, Msg type는 GIOP 메시지 헤더 뒤에 따라오는 메시지의 타입 식별하기 위한 것으로 GIOP 메시지 타입 중에 하나를 가리킨다. GIOP 1.0에서의 메시지 타입으로는 요구, 응답, 요구취소, 위치요구, 연결종결, 그리고 메시지 에러 메시지가 있고, 1.1에서는 프래그먼트 메시지가 추가되어 있다. 마지막으로 Msg size는 메시지 헤더의 뒷부분에 포함된 메시지의 크기를 옥테트 단위로 표시하고 있다.

V1.0

Magic	Version	Byte order	Msg type	Msg size

V1.1

Magic	Version	Flags	Msg type	Msg size

그림 2. GIOP 메시지 헤더 정보

Fig. 2 GIOP message header format

2.2 메시지 구조

본 논문에서는 GIOP 메시지 타입 중에서 요구 메시지와 응답 메시지를 가지고 트래픽 정보를 추출하였기에 이들에 대해 간략히 살펴보겠다. 기본적으로 이들 메시지의 처음 부분에는 GIOP 메시지 헤더가 들어가 있고, 다음으로 각 메시지의 헤더와 몸체 부분이 들어가게 된다.

GIOP msg header	Request Header	Request Body

그림 3. 요구 메시지 구조

Fig. 3 Request message format

V1.0

Service context	Request ID	Response expected	Object key	Operation	Requesting principal

V1.1

Service context	Request ID	Response expected	Reserved	Object key	Operation	Requesting principal

그림 4. 요구 메시지 헤더 정보

Fig. 4 Request message header format

요구 메시지 헤더는 클라이언트가 서버로 보내는 요구를 담은 메시지로 그림 3과 같은 형태를 가지고 있다. 그림 4에서 요구 메시지 헤더 정보를 살펴보면, Service context는 클라이언트에서 서버로 가는 ORB 서비스 데이터가 포함된 부분이고, Request ID는 메시지 식별자로 응답 메시지와 같이 사용하게 된다. Response-expected는 요구 메시지가 오기를

원하는지, 원하지 않는지를 표시한다. Reserved는 v1.1에만 있는 것으로 0으로 세팅되어 있고, 앞으로 사용을 위해 남겨두었다. Object key는 호출할 객체를 식별하기 위한 것이다. Operation은 클라이언트에서 호출된 오퍼레이션 이름이 들어가 있는 부분이다. Requesting principal은 요구에 대한 대표자를 식별하기 위한 값이 들어가 있다.

Service context	Request ID	Reply status

그림 5. 응답 메시지 헤더 정보
Fig. 5 Reply message header format

GIOP msg header	Reply Header	Reply Body

그림 6. 응답 메시지 구조
Fig. 6 Reply message format

그림 5에 보이는 Reply 메시지는 서버로부터의 결과를 클라이언트로 돌려줄 때 사용하는 메시지이다. 그림 6에서 보여주는 Reply 메시지 헤더 정보를 살펴보면, Service context는 요구 메시지의 것과 같고, Request ID는 특정 요구 메시지에 대한 응답이라는 것을 가리키는 것이다. Reply status는 응답 메시지의 완료된 상태를 표시하고 있다.

Ⅲ. OSI 관리

망 관리는 망 자원의 사용과 활동을 모니터링, 계획, 조절하는 것을 말한다. 망 관리의 첫 번째 목표가 관리대상이 되는 모든 하위 시스템들을 관리자에 의해 관리하는 것이다. 그리고, OSI 관리 표준안의 목적은 일관된 관리 방법을 제시함으로써 통합된 망 관리 시스템을 정의하는데 있다[9,10].

1. 관리 구조

OSI 관리 모델은 관리자가 하나 또는 여러 개의 관리 프로세스와 대리자라고 불리는 프로세스들간의 상호 작용으로 기술된다. 그림 7은 OSI 관리 모델을 보여주고 있다. 여기서, 대리자는 관리 관점에 포함되는 관리객체들을 조절하며, 관리의 관점에서 관리객체들은 모든 자원들의 추상적 표현이라 할 수 있다. 그리고, 모든 관리객체 정보들은 MIB (Management Information Base)라는 개념적인 저장소에 저장된다. 관리 프로세스는 대리자에 의해서 유지되는 정보들을 CMIS(Common Management Information Service)에서 제공되는 서비스를 이용하여 CMIP을 통해 전송하게 된다.

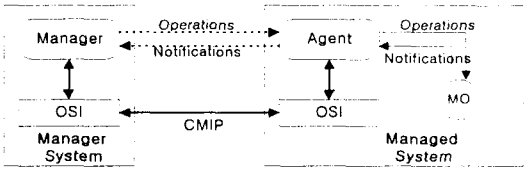


그림 7. 시스템 관리 모델
Fig. 7 The model of system management

Ⅳ. 시스템 설계

현재 CORBA에서 트래픽 모니터링하기 위한 프로그램의 개발과 연구가 이루어져오고 있다. Lakshmikanth은 ORB 소스 코드를 변경하여 메시지를 복제하여 기록하는 방법을 사용하여 CORBA에서 어플리케이션의 작동을 살펴본바 있다[11]. 이 연구는 특정 프로그램의 작동상황과 성능을 측정하기 위한 것이었다. 그리고, Inprise사와 Iona사에서는 ORB와 어플리케이션 사이에 메시지를 가로채는 기능을 삽입하여 현재 CORBA에 대한 정보를 추출하기 위해 사용되어 지거나, 특정 메쏘드을 사용하여 메시지를 추출하는 방법을 연구하고 있다[2,3]. 현재 Visibroker의 Manager에서 성능 모니터 도구가 나와 있지만, 외부 프로토콜 지원이 부족하여 기존 시장에 나와 있는 CMIP, SNMP 관리 제품과 연동하기에는 부족함이 많다. Iona회사의 Orbix Manager에서도 SNMP 지원은 되고 있지만, TMM 환경으로의 통합을 위한 CMIP은 지원되지 않고 있다.

표준화된 관리방식에 대한 연구로는 MAScOTTE (Management Services for Object Oriented Distributed Systems)가 활발히 진행 중에 있으며, 이 프로젝트에서는 관리 퍼실리티를 제공함으로서

CORBA 기반의 관리 프로그램이나 현존하는 표준안을 기반으로 하는 관리 도구들에 의해 CORBA에 대한 관리를 시도하고 있으며[4], 현재 OSI 관리 방식의 구현을 추진 중에 있다.

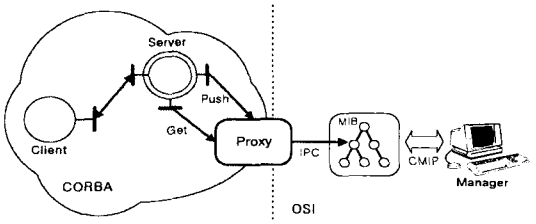


그림 8. 모니터링 시스템 프로토타입
Fig. 8 The prototype of monitoring system

현재 구현하고자 하는 시스템은 CORBA의 트래픽 모니터링에 있어서 OSI 관리 기법을 적용하여 OSI 관리자에 의해 모니터링 할 수 있도록 하는데 있다. 시스템의 개괄적인 흐름을 보면(그림 8), 클라이언트가 적절한 서버를 선택하여 요구 메시지를 보낸다. 서버는 받은 메시지를 해당 메시지 타입을 처리하는 메쏘드로 보내지고, 이곳에서 트래픽 정보를 추출하게 된다. 그렇게 한 후, 서버는 CORBA 객체인 프록시로 메시지에 관한 트래픽 정보를 전송하게 된다. 프록시는 대리자에 있는 MIB의 관리객체의 값들을 갱신하기 위한 정보를 보낸다. 그리고, 관리자가 대리자로 정보를 요구하면, 대리자는 적절한 값을 MIB에서 찾아서 되돌려 준다.

전체적인 모니터링 시스템의 구성은 서버 객체, 프록시 객체, 그리고 MIB로 구성되어 있다. 중간에 프록시 객체를 두어 OSIMIS와 CORBA사이의 트래픽 정보 중계 역할을 한다. 트래픽 정보를 서버에서 추출하여 MIB로 넘겨주게 된다.

1. CORBA 모니터링 시스템

그림 9는 서버 객체의 인스탄스화된 모습을 보여 주고 있다. 서버 객체는 서비스 제공자, 프록시 서버, 그리고 이벤트 서버로 구성되어 있다. 일반적인 CORBA 서비스를 클라이언트에게 제공하는 서비스 제공자와 트래픽 정보를 수집하여 프록시 객체에게 트래픽 정보를 넘겨주는 역할을 하는 프록시 서버로 구성되어 있다. 그리고, 서버에서 발생하는 예외 처리 정보들은 이벤트 서버를 사용하여 프록시로 전송하게 했다. 본 논문에서 서버 객체에서 추가한 부분이 프록시 서버와 이벤트 서버 부분이다. 이들을 라이브러리화 하였으며, 서버 객체 생성할 때 같이 링크할 수 있도록 했다.

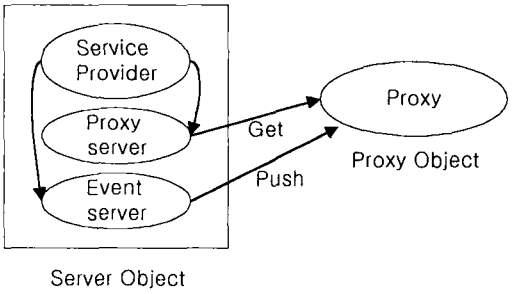


그림 9. 서버 객체
Fig. 9 Server object

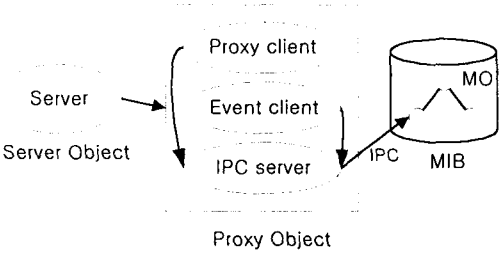


그림 10. 프록시 객체
Fig. 10 Proxy object

그림 10에서 프록시 객체의 경우에는 프록시 클라이언트, 이벤트 클라이언트, 그리고 IPC 서버로 구성되어 있다. 프록시 클라이언트는 서버 객체로부터 트래픽 정보를 수신 받는다. 서버에서 예외처리 발생할 때 이벤트 서버가 보내는 예외 정보를 받는 이벤트 클라이언트가 구성된다. 그리고, MIB에 있는 관리객체에게 트래픽 정보를 전송하기 위해 프로세스간 통신을 담당하는 IPC 서버가 존재한다. 그리고, OSI 대리자는 MIB내에 관리객체들의 트래픽 정보를 갱신하기 위해 IPC 서버와 프로세스간 통신을 통하여 트래픽 정보를 수신한다.

2. 트래픽 파라미터

본 논문에서 추출하는 트래픽 정보로는 오퍼레이션, 바인딩 시간, 예외처리, 메시지 크기, 타임 스탬프, 종료 시간이 있다. 각각의 파라미터 특징을 살펴보면, 오퍼레이션은 클라이언트가 객체 구현 쪽으로 요구하는 오퍼레이션 명을 말한다. 이 정보는 요구 메시지 부분에 저장되어 있다. 이를 통하여 어떤 종류의 오퍼레이션이 빈번히 사용되는지 알 수가 있다. 예외처리는 클라이언트와 객체구현 쪽에서 예외가 생길 경우, 어떤 예외가 생겼는지는 저장한다. 예외가 생겼다는 것은 시스템의 오류 등을 파악할 수 있다. 바인딩 시간은 클라이언트가 요구 메시지를 보내기 위해 객체 구현과 연결 설정하는 동안의 시간을 말한다. 즉, 얼마나 빠르게 연결 설정이 되는지를 알 수 있고, 서버의 처리 능력 등을 추측할 수 있다. 타임 스탬프는 특정 객체 구현에서 클라이언트에서 오는 요구메시지 도착 시간을 기록하고 있다. 그래서, 객체구현이 언제 가장 많이 사용하고 있는지를 알 수가 있다. 종료시간은 객체 구현에서 클라이언트에서 온 요구를 처리를 마치고 종료할 때의 시간을 기록한다. 이를 통해 현재 서버가 메시지가 처리되는 시간을 알 수 있다. 메시지 크기는 서버에서 송수신 되는 메시지의 크기를 저장한다. 이 부분에서는 현재 서버에서 처리하는 메시지의 양을 가늠할 수 있다.

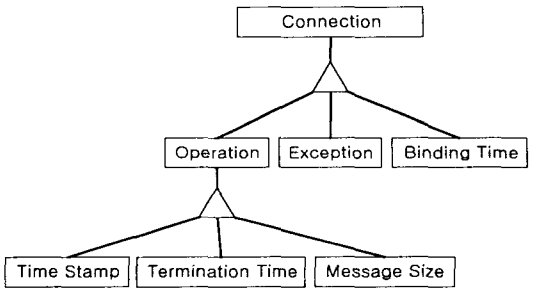


그림 11. 포함 트리

Fig. 11 The containment tree

각각의 트래픽 파라미터들은 맨 처음을 연결(Connection)으로 두어 그림 11과 같은 포함 트리 형태로 구성했다. 즉, 하나의 연결에는 오퍼레이션, 예외, 그리고 바인딩 타임이 생기게 된다. 그리고, 오퍼레이션에서는 타임 스탬프, 종료시간, 그리고 메시지 크기가 포함된다. 그리고, 각 트래픽 파라미터들은 관리객체화 되어 그림 11같은 형태의 구조로 MIB 내에 구성되어 진다.

여섯 개의 관리객체와 트래픽 파라미터들은 서로 일대일 대응이 된다. 즉, 각 트래픽 파라미터들은 MIB의 관리객체로 추상화되어 각 관리객체가 실체화되면서 트래픽 정보들이 저장되어 진다. 그러면, 관리자는 대리자를 통하여 트래픽 관련 정보를 획득할 수 있게 된다.

V. 프로토타입 구현

사용한 툴로는 CORBA 환경은 Inprise사의 VisiBroker를 사용하였으며, 사용한 시스템은 Sun사의 Solaris 2.5.1으로 개발언어는 C++을 사용하였다. OSI 관리 플랫폼은 UCL의 OSIMIS를 사용했다. 그리고, 모든 트래픽 모니터링은 서버에서 입출력되는 메시지를 추출하였다.

구현 부분은 크게 서버 객체, 프록시 객체, 그리고 관리객체로 나누어진다. 각 객체들은 몇 개의 모듈형태로 구성하였으며, 각 모듈들은 하나의 객체로 서로 협력하여 동작하게 된다.

서버 객체는 서비스 제공자, 프록시 서버, 그리고 이벤트 서버을 각각 모듈로 구성했다. 프록시 서버와 이벤트 서버는 서버 객체가 동작할 때에 같이 구동되어 진다. 특히, 서버 객체에서 프록시 서버와 이벤트 서버는 라이브러리 형태로 구성되어 서버 코드에 아무런 변경 없이 같이 링크하여 서버 객체를 생성할 수 있게 했다. 이 라이브러리에서 중간에 오는 메시지를 가로채서 특정 형태의 메시지들이 들어올 때, 특정 메쏘드를 호출하여 트래픽을 처리하고 있다.

시스템 구성의 두 번째 구성 요소인 프록시 객체는 일반적인 CORBA 객체로 시스템에서 하나의 프로세스 형태로 동작한다. 프록시 객체 부분도 프록시 클라이언트, 이벤트 클라이언트, 그리고 IPC 서버로 각각 모듈로 구성하였다. IPC 서버는 프로세스간 통신을 사용하여 프록시 객체와 관리객체 사이의 트래픽 정보의 전송이 이루어진다. 사용되

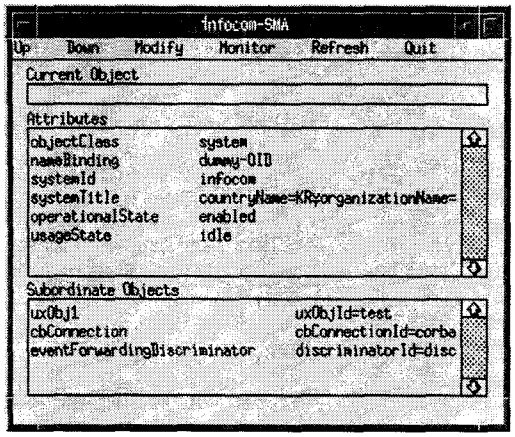


그림 12. OSI 대리자에 있는 객체들
Fig. 12 Objects in the OSI agent

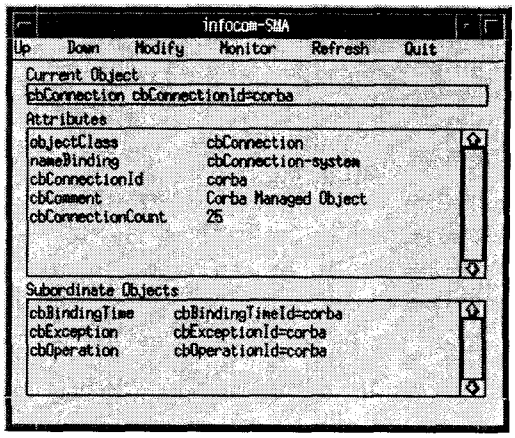


그림 13. cbConnection에 있는 하위 객체들
Fig. 13 Subobjects of cbConnection

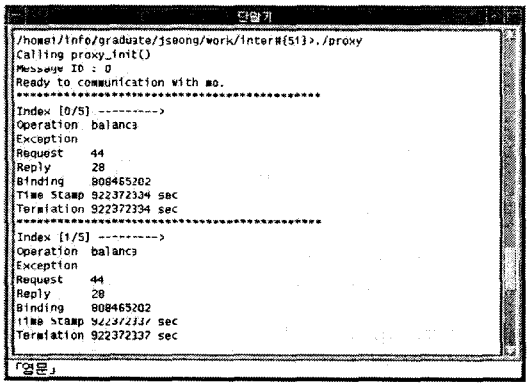


그림 14. 프록시에서 트래픽 데이터
Fig. 14 Traffic data at the proxy

어진 IPC는 메시지 큐이다.

MIB에 있는 관리객체들은 GDMO의 템플릿에 맞추어 작성한 후, 생성된 소스를 가지고 구현하였다[12].

각 트래픽 정보와 관련된 관리객체들은 프록시 객체와 프로세스간 통신을 통하여 트래픽 정보를 획득한다. 이러한 트래픽 정보의 확인을 위해서 사용한 OSI 관리자는 OSIMIS에서 제공되는 CMIS browser를 사용하였다. 그림 12와 그림 13이 CMIS browser로 본 관리객체들이다. 그림 12는 하위 객체에 있는 cbConnection는 Connection이 관리객체로 인스탄스된 것을 보여주고 있다. 그리고, 그림 11에서 보여주는 트리 형태로 관리객체들이 구성하게 되는데, 트리 구성에서도 보여주고 있듯이 cbConnection의 하위에 구성되어진 관리객체 객체를 보여주는 것이 그림 13이다. 즉, 하위 관리객체로 cbOperation, cbException, 그리고 cbBindingTime 관리객체들이 인스탄스화 되어 있는 것을 보여주고 있다. 그리고, 그림 14은 프록시에서 수집되는 실제 트래픽 테이터들을 보여주고 있다.

VI. 결 론

CORBA는 분산 시스템에서 업계표준으로 자리잡고 있는 상황에서, 많은 연구소와 대학에서 연구와 개발이 되어지고 있다. 최근 들어, 분산 환경에서의 수 많은 객체들의 관리에 관심이 많아지면서, CORBA에서 시스템 관리의 연구가 고조되고 있다. CORBA 시스템 관리에서 시스템 모니터링을 하는 것이 단지 정보수집만을 위한 것이 아니라, 더욱 효율적인 CORBA 시스템 관리를 하고자 하는데 있다. 그리하여, 본 논문에서는 CORBA 트래픽을 모니터링하기 위해 표준화된 OSI 관리 방식의 프록시를 설계하였다.

본 논문에서는 서버 객체에 예외처리를 위한 이벤트 서버와 트래픽 정보 전달을 위한 프록시 서버부분을 구현함으로서 예외 처리 발생 할 때 처리의 효율을 높이고, 데이터들은 프록시 서버에서

수집하게 했다. 그리고, MIB에 있는 트래픽 관련 관리객체들에 트래픽 정보를 저장하고, 이 정보들은 프록시 객체의 IPC 서버에서 가져온다. 각각의 처리들을 모듈화하여 처리하여 작업의 효율을 높였다. 각 트래픽 파라미터들의 속성에 따라 서로간의 관계를 두어 구성하였고, 관리객체도 마찬가지 형태로 구성하였다. 그래서, 관리자는 좀더 구조적으로 값을 획득할 수 있게 하였다.

구현 시스템은 CORBA 트래픽에 대하여 OSI 관리 방식을 적용하기 위한 프록시를 설계한 것으로, 향후 시스템 관리 응용으로의 개발이 기대된다. 표준화된 CORBA 모니터링 시스템은 CORBA 시스템 관리에 있어서 중요한 역할을 할 것이고, 나아가 분산 처리 환경에서 망 관리에 활용되리라 사료된다.

참고문헌

[1] OMG, "System management: Common Management Facilities, Volume1, Version 2", X/Open Company, Dec, 1995.

[2] Inprise co., "Visibroker Manager", http://www.inprise.com/visibroker/products/vbroker /tools/, 1997.

[3] Iona, "Orbix Manager", 1997.

[4] White Paper, "Introduction to MAScOTTE", MAScOTTE project, 1998.

[5] BLACK & WHITE co., http://www.blackwhite.com/, 1999.

[6] H. Onazawa, "분산 오브젝트 지향 기술 CORBA", 홍릉과학 출판사, 1997.

[7] Robert Orfali and Dan Harkey, "Client/Server Programming with JAVA and CORBA 2nd Edition", Joh Wieg & Sons, Inc., 1998.

[8] OMG, "CORBA/IIOP 2.2 Specification", 1998.

[9] Adrian Tang, Sophia Scoggins, "OPEN NETWORKING WITH OSI", 1994.

[10] William Stallings, "SNMP SNMPv2 and CMIP", Addison-Wesely, 1993.

[11] Lakshmikanth S. Jnnalagada, "The role of network traffic statistics in devising object migration policies", Dissertation, The State University of New Jersey, 1997.

[12] G. Pavlou, K. MacCarthy, S. Bhatti, and G. Knight, "The OSIMIS Platform: Making OSI Management Simple". Proceeding of the 4th International Symposium on Integrated Network Management, 1995.

박 재 성(Jae-Seong Park)
1997년 2월 제주대학교 에너지공학과 공학사
1999년 2월 제주대학교 정보공학과 공학석사
*주관심분야 : 분산망 관리, CORBA 등

송 왕 철(Wang-Cheol Song)
1989년 2월 연세대학교 전자공학과 공학사
1991년 2월 연세대학교 전자공학과 공학석사
1995년 2월 연세대학교 전자공학과 공학박사
1996년 2월~현재 제주대학교 정보공학과
* 주관심분야 : TMN, 분산망 관리, CORBA, mobile agent 등