通信情報保護學會誌
第6卷 第2號, 1996. 6

CORBA 보안 구조

이 영 록*, 노 봉 남**

요 약

본 연구는 OMG/CORBA의 필요성과 OMG/OMA 구조, 그리고 OMG/CORBA에 대해 알아보고, 현재 분산처리의 기본 소프트웨어라 할 수 있는 OSF/DCE와 분산 객체중개자인 OMG/CORBA간의 차이를 비교한다. 또한 분산 시스템에서 객체 요청 중개자를 사용하는 CORBA 환경에서 있을 수 있는 보안 문제를 어떻게 다루어야 하는가를 서술하고, 이를 해결하는 방안 등을 제시한다.

1. 서 론

일괄처리와 처리기 공유라는 시분할 시대를 거쳐 개인 생산성 향상을 위한 개인용 컴퓨터 시대에 이르기까지 컴퓨터의 기술은 참으로 짧은 기간에도 불구하고 눈부신 발전을 거듭해왔다.

개인용 컴퓨터의 확산은 근거리 통신망(Local Area Network)을 불러들여 데이터 공유라는 서비스를 내놓게 하였으며, 연이은 80년대 말에는 개인용 컴퓨터와 여러 중·대형 컴퓨터를 원거리 통신망(Wide Area Network)로 연결하므로써 시스템을 통합하여 사용자들이 사용할 수 있게 하였다. 하지만 사용자들의 욕구는 처리기 공유나 데이터와 자원의 공유에 그치지 않고 응용 프로그램들을 통합 운용하여 모든 자원을 공유하려는 분산처리의 시대로 접어들게 하였다.

서로 다른 기종들이 널리 쓰이는 환경에서 분산처리를 행하기 위해서는 수많은 구성 요소들이 모여야 되는데 분산 응용간의 상호 운용성을 제공하기 위한 분산 시스템 소프트웨어도 그 중 일부이다. 이 분산 시스템 소프트웨어는 현재 분산처리 기본 소프트웨어라 할 수 있는 OSF/DCE나 분산 객체 중개자인 OMG/CORBA 위에서 구현되고 있는 상황이다.

한편, 개방 시스템화와 네트워킹화로 인해 기인하는 바이러스, 트로이목마, 불법 접근, 악의적인 파괴 등으로부터 정보보호가 절실한데, 지금까지는 보안을 요하는 분야가 제한되었고, 또 상업적으로도 시장성이 없어 기술이 미비한 실정이었다. 하지만 현재는 상업 영역에서도 보안 필요성이 널리 인식되고 있어 각국 국방부 주도의 보안이나 상업적으로 이루어지는 보안 둘다 결국 한 방향으로 수렴되어 보안에 관한 연구가 진행되고 있는 실정이다.

보안의 관점은 어느 측면에서 접근하느냐에 따라 네트워크 보안, 컴퓨터 보안, 정보 보안,

* 전남대학교 전산학과 박사과정
** 전남대학교 전산학과 교수

운영 보안(operation security), 직원 보안(personnel security), 물리적 보안(physical security) 등과 같이 여러 갈래로 나뉘어 논의되고 있다. 객체지향 기법이 도입되는 현 상태에서 기존의 보안 메커니즘과 보안 구조로 이러한 보안 문제를 해결하기에는 역부족이다.

본 연구에서는 OMG/CORBA의 필요성과 OMG/OMA 구조, 그리고 OMG/CORBA에 대해 알아보고, 현재 분산처리의 기본 소프트웨어라 할 수 있는 OSF/DCE와 분산 객체 중개자인 OMG/CORBA간의 차이를 비교한다. 또한 분산 시스템에서 객체 요청 중개자를 사용하는 CORBA 환경에서 있을 수 있는 보안 문제를 어떻게 다루어야 하는가를 서술하고, 이를 해결하는 한 방안 등을 제시한다.

2. 분산 컴퓨팅

분산 컴퓨팅이라는 용어는 서로 정보를 공유하도록 해주는 둘 이상의 소프트웨어로 정의하고 있다. 분산 환경에서 어떤 작업을 다수의 기계를 통해 수행하도록 하면 현재 설치되어 있는 수백만 개의 PC나 워크스테이션 상으로 작업량을 분담시킴으로써 수행 시간을 줄여 주고, 자원을 공유하게 해주므로 경제적일 뿐만 아니라 컴퓨팅 환경을 서서히 확장해 갈 수 있다는 장점이 있다.

분산 컴퓨팅을 행하는 방법은 크게 세가지 정도로 요약되는데[6], 한가지 방법은 소켓(socket)이라고 불리는 API를 통해서 전송 서비스를 엑세스 할 수 있도록 하는 것으로써 직접 전송 계층 프로토콜(TCP, UDP, TP4, SPX 등) 상에서 통신하도록 하는 것이다. 전송 계층상에서 소켓을 이용하여 분산 응용(distributed application)을 직접 개발하는 것은 단지 수개의 시스템 호출(system call)만을 이용하므로 간단하고, 속도 면에서 굉장히 효율적인데 반해 제공되는 서비스가 극히 제한적이며 보안을 행하기가 쉽지 않다는 단점이 있다.

두번째 방법으로는 좀더 높은 단계의 서비스를 제공하기 위해 RPC 메커니즘에 의존하는 방법이다. 물론 RPC 자체는 소켓이나 TLI/XTI를 이용하긴 하지만 응용 개발자들에게 더 고급수준의 추상화를 제공하며 이들에게 익숙해져 있는 프로시쥬어 호출과 같은 파라다임을 제공해 준다. 이 RPC도 어떤 RPC가 사용되는 가에 따라 둘로 나누어지는데 DCE와 ONC가 그것이다. DCE는 IBM이나 DEC, HP 등의 후원으로 구성된 OSF에 의해 정의된 것이고, ONC는 SunSoft사를 중심으로 한 단체에서 정의되었다.

마지막 세번째 방법은 분산 응용의 실행이 객체지향 방법으로 해결되도록 하는 것으로서 CORBA 메카니즘상에서 이루어진다. 하나 이상의 프로세스에 분포되어 있는 객체들과 서로 다른 기계에 존재하는 프로세스들을 가동해서 객체지향 방식으로 일을 수행하도록 하기 위해서는 최소한 두 가지를 해결해야만 된다[6].

첫째는 한 객체를 다른 객체에게 알려주는 인터페이스를 정의해야 한다. 우리가 어떤 객체를 접근하는 방법을 모르고서는 그 객체의 메소드에 도달하기가 어렵기 때문이다.

둘째는 객체들끼리의 질의와 응답을 주고받을 수 있는 기법이 필요한데, 이 인터페이스는 프로세스들 내에 있는 객체들 간의 대화를 주고받기 위해 사용되어져야하기 때문이다.

위의 두가지 필요한 일을 제공해 주기 위해 여러 제안들이 난립되는 현상을 막기위하여 OMG는 CORBA를 정의했다. RPC가 소켓보다도 더 높은 단계의 서비스를 제공해

주듯이 CORBA가 이들 DCE나 ONC+ 보다도 더 높은 수준의 서비스를 제공한다고 볼 수 있다[6].

3. OMG/CORBA

3.1 CORBA의 필요성

한 조직은 여러 부서로 되어 있고, 다양한 자원의 정보를 다루기 위해 각 부서는 다양한 소프트웨어를 사용하고 있다. 하지만 회사의 최고 관리자는 네트워크를 통해 각 부서의 다양한 데이터에 접근해서 자료를 가져와 그 관리자가 제일 선호하는 스프레드시트를 띄워서 작업하려 할 것이다. 이런 일이 가능하도록 하기 위해서는 수많은 서로 다른 작업 구성 요소들을 통합해야 하는데, CORBA를 이용하면 현존하는 하드웨어나 소프트웨어, 네트워크 등의 하부구조를 수정하지 않고도 PC에서 돌아가는 응용들을 기업의 다른 시스템에 연결할 수 있도록 해준다.

CORBA를 이용하면 다음과 같은 일을 가능하게 해준다[7].

- 널리 쓰이고 있는 데스크 탑형 응용들로부터 생성되어 널리 산재해 있는 정보나 자원에 접근할 수 있다.
- 현존하는 업무 데이터나 시스템을 네트워크의 자원으로서 이용할 수 있다.
- 널리 쓰이는 데스크 탑형 도구들이나 응용들을 특정한 업무를 위한 주요 기능으로 사용할 수 있다.
- 새로운 토폴로지나 자원을 네트워크 기반 시스템들에 반영할 수 있다.

CORBA는 분산 객체 컴퓨팅에 기반을 둔 것임으로 CORBA 사양에 맞게 구현된 응용들끼리는 서로간의 객체에 접근하여 공유할 수 있도록 해주는데, 이는 CORBA가 중개자(broker)를 두어 클라이언트와 서버간의 요청을 다루어 주기 때문이다.

CORBA는 통신용 미들웨어(middleware) 사양(specification)인데 미들웨어라함은 응용들을 기계에 종속되는 낮은 단계의 세부 사항으로부터 분리시켜 응용 개발자로 하여금 기계마다 복잡한 많은 세부 정보를 알 필요 없도록 하는 것이다. CORBA는 통신커널의 세부 사항으로부터 응용을 분리시켜 주는 소프트웨어 사양이라 할 수 있다[11].

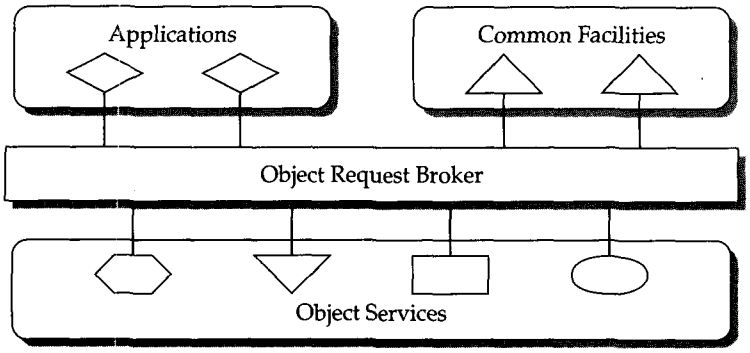
3.2 OMG/OMA

객체 기술에 관한 전문적 기술과 지식을 소유하고 있는 SunSoft, HP, Digital Equipment Corp, IBM, HyperDeck 등의 회사와 사용자, 그리고 S/W 제공자들이 모여 OMG(Object Management Group)라는 그룹을 만들고 분산 객체 컴퓨팅을 위한 해결책으로 CORBA라는 사양을 정의했다.

OMG는 CORBA 사양을 공표하기 전에 CORBA의 기초가 되는 OMA를 발표했다. OMA는 그림 3.1과 같이 4개의 구성 요소로 되어 있다[2].

3.2.1 객체 요청 중개자

CORBA(Common Object Request Broker)라는 이름이 ORB에서 나왔다고 할 수 있는데, 객체 요청 중개자(Object Request Broker)는 시스템에 있는 모든 객체들에 대한 통신허브(communication hub)를 제공하는 것으로 하드웨어 버스 개념과 유사하다.

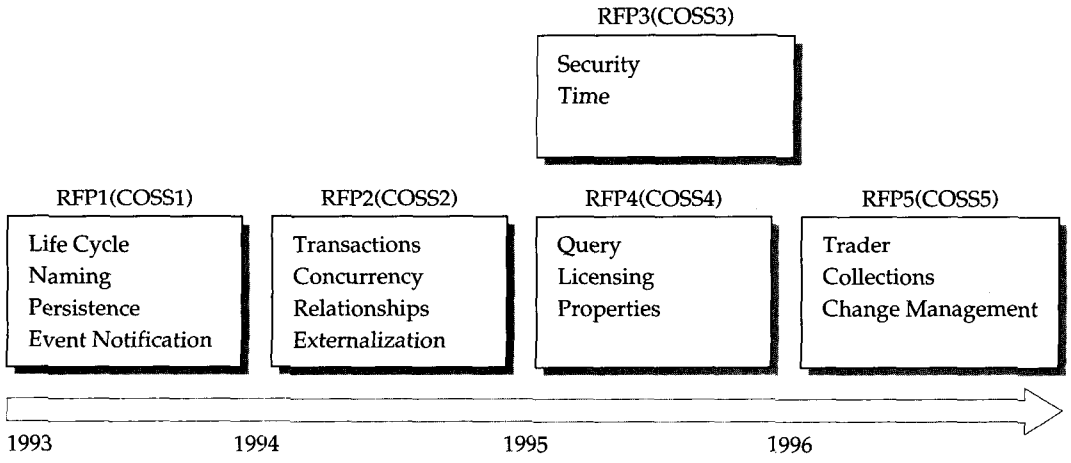


〈그림 3.1〉 OMG/OMA 구조

객체들 간에 전달되는 모든 정보는 이 통로를 거쳐간다. OMG는 곧이은 CORBA라는 사양에서 이 개념을 자세히 정의하고 있다. 응용 레벨에서 사용되는 통신 하부구조가 CORBA이기 때문에 CORBA에서는 객체 요청 중개자와 기본 객체 적응자(Basic Object Adapter), 인터페이스 저장소(interface repository) 등의 개념이 함께 정의된다.

3.2.2 객체 서비스

객체들을 위한 시스템 서비스들로서 객체의 위치를 알게 해주는 객체 이름 서비스, 한 객체가 그 자신의 상태 변경 상황을 상대 객체들에게 알리기 위해 객체들 간의 사건(event)을 나르는 객체 사건 서비스, 그리고 객체들을 생성하고 또한 생성된 객체를 저장하거나 객체에 대한 접근을 제어하는 등의 서비스가 있다[3]. OMG는 이들 서비스들의 정의 사양을 COSS(Common Object Service Specification) vol.1에 RFP1부터 RFP4 까지 4개의 시리즈로 공표했다. 객체 서비스들을 간단히 나타내보면 그림 3.2와 같다[10].



〈그림 3.2〉 객체 서비스들

3.2.3 공동 설비(Common Facility)

공동설비는 많은 응용이나 S/W 제공자들의 요구를 직접적으로 만족시키도록 기본적인 객체 서비스들을 보다 풍부한 인터페이스로 확장한 것으로서 OMG내의 CFTF(Common Facilities Task Force)라는 상임위원회가 관장하여 정의하고 있다. 제공되는 서비스들로는 객체간의 문서교환, 전자우편, 클립보드, 객체 연결이나 객체 내장(object embedding)과 같은 서비스들이 있다.

3.2.4 응용 객체들(application objects)

모든 사용자들이 작성해 놓은 응용프로그램들이나 기존에 개발된 특정 사용자들의 응용프로그램, 그리고 클래스의 객체들이 여기에 해당하며 워드프로세서, 스프레드쉬트, 멀티미디어 응용객체 등을 예로 들 수 있다.

3.3 CORBA 개요

CORBA의 객체지향 인터페이스를 사용하는 클라이언트(client) 객체는 어떻게 서버 객체가 그의 작업을 수행하는지를 알 필요 없이 그 서버 객체상에서 수행되어지는 한 연산에 어떻게 요청을 보내는가의 방법만 알면 되는 것이다[7].

CORBA는 클라이언트와 서버를 분리하기 위해 요청이라 불리는 메시지로 둘 간의 통신을 제어하는데 클라이언트가 요청을 보내면 서버가 이에 대한 결과를 응답함으로써 상호간 대화한다.

요청이라는 메시지는 기본적인 형태를 지녀야 하는데 이 메시지에 포함되어야 하는 정보들로는 ① 수행되어야 할 서버의 연산(operation)을 지시해 주어야 하고, ② 그 연산이 어느 객체에 위치해 있는지의 객체 참조(reference)점을 알려주어야 하고, ③ 요청에 의해 수행되는 결과에 따른 예외 사항 정보를 되돌려 줄 수 있는 메커니즘이 있어야 하고, ④ 요청 메시지가 전달시 동시에 알려져야 하는 부가적인 정보인 내용(context) 객체의 참조점이 나타내져 있어야 하고, ⑤ 그 연산이 수행될 때 필요한 정보인 매개변수들이 0개 이상 있어야 된다.

CORBA 시스템은 위에서 보인 다섯 가지 정보를 포함하도록 클라이언트의 요청 메시지를 만들어야 한다. 클라이언트 응용 프로그램을 작성할 때, 클라이언트 응용 개발자는 분산되어 있는 서버에서 수행되어지는 연산의 종류나 각 연산에 필요한 매개변수들을 알고 있어야 하는데, CORBA는 서버 개발자가 구축해 놓은 그 서버상에서 사용할 수 있는 연산을 OMG IDL(Interface Definition Language) 언어를 사용해 인터페이스로 기술하도록 하고 있다.

다시 말해 CORBA는 클라이언트나 서버 응용을 작성할 수 있기 전에 먼저 *OMG IDL* 화일을 작성하거나, 인터페이스 정의들이 포함되어 있는 인터페이스 저장소에 접근할 수 있도록 정의하고 있어 클라이언트가 이용하려는 서버의 연산 이름만 알면된다.

OMG IDL 화일은 데이터 타입(data type), 연산(operation), 객체(object)들을 기술하고 있는데, 클라이언트는 요청을 위해 이 IDL을 이용하고, 서버는 IDL에서 정의된 객체와 그 객체 내에 있는 연산의 수행과 관련된 작업을 구현해야 한다.

CORBA는 하고자 하는 작업과 관련된 인터페이스를 OMG IDL 언어로 기술하게 함으로써 클라이언트나 서버 프로그램을 개발하는 개발자가 어떤 종류의 프로그래밍 언어를 사용하여 코딩하더라도 OMG IDL 컴파일러가

IDL 화일을 컴파일할 때 클라이언트 스터브(client stub)와 서버 스켈튼(server skeleton)을 각 개발자들에 맞는 특정언어로 변환시켜 줌으로써 개발자들은 특정한 프로그래밍 언어만을 사용해도 된다.

CORBA에서는 특정한 인스턴스를 나타내기 위해 객체 참조(object reference)를 이용한다. 객체 구현(object implementation)은 특정한 객체상의 연산들을 행하도록 요하는 클라이언트를 만족시켜 주는 서버측의 부분으로 서버응용에 위치하고 있으며, 요청되는 작업을 행하는 일련의 코드들로 된 하나 이상의 메소드(method)들을 포함하고 있다.

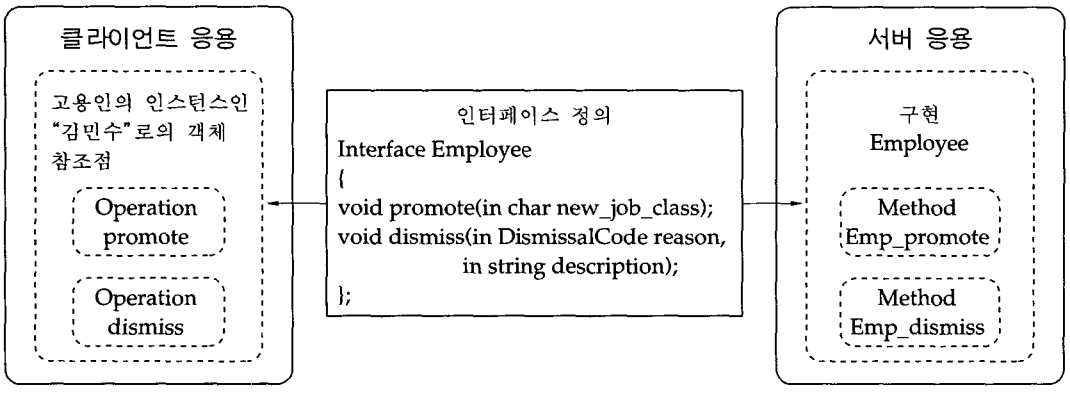
그림 3.3은 클라이언트 응용과 서버상의 관련 구현이 IDL 인터페이스를 사이에 두고 어떻게 작동되는지를 간단히 보여주고 있다. 이 그림에서는 "정보통신부서" 소속의 직위가 "과장"인 "김민수"라는 특정 객체에 'promote'와 'dismiss'에 관한 연산을 행할 수 있도록 서버의 객체 구현을 설명해 주고 있다. "김민수"는 고용인이라는 객체중 특정한 객체 인스턴스이므로 고용인 인터페이스에서 정의해 놓은 두 연산을 "김민수" 객체 위에서 행할 수 있는 것이다. 실지 "김민수" 객체에서 행해지는 연산은 서버응용에 실지 구현되어 있는 객체 구현에서 행해지게 되고 이 두 연산은 두 메소드를 실행한다.

모든 요청은 앞에서 기술한 바와 같이 서버에서 그 요청이 성공적으로 수행됐는지를 나타내는 예외 정보를 객체 구현이 반환해 주도록 해주는 메커니즘을 포함하고 있는데, 이 정보는 객체 구현이나 객체 요청 중개자에 의해서 반환될 수 있다. 클라이언트는 이 예외 정보를 보고 보냈던 요청을 재시도할 것인지를 결정하는데 클라이언트는 이 예외 정보를 읽을 수만 있도록 되어 있다.

3.4 CORBA 구조

앞절에서 설명한 개념을 이용하여 메시지 전달과정을 나타내보면 그림 3.4와 같다[7].

먼저 정의해 놓아야 할 객체들로서 한 회사원이 일하고 있는 "department" 객체와 일할 수 있는 능력이 있는 사람들의 개인 기록을 모아 놓은 "personnel" 객체, 그리고 한 회사에 채용되어 실지 일하고 있는 회사원의 인사기록을 모아 놓은 "employee" 객체 등이 서로 다른 서버객체에 구현되어 있다고 가정하자. 이때 "personnel" 객체에서 행해줄 수 있는 연산인 "hire"와 "employee" 객체에서 행해줄 수

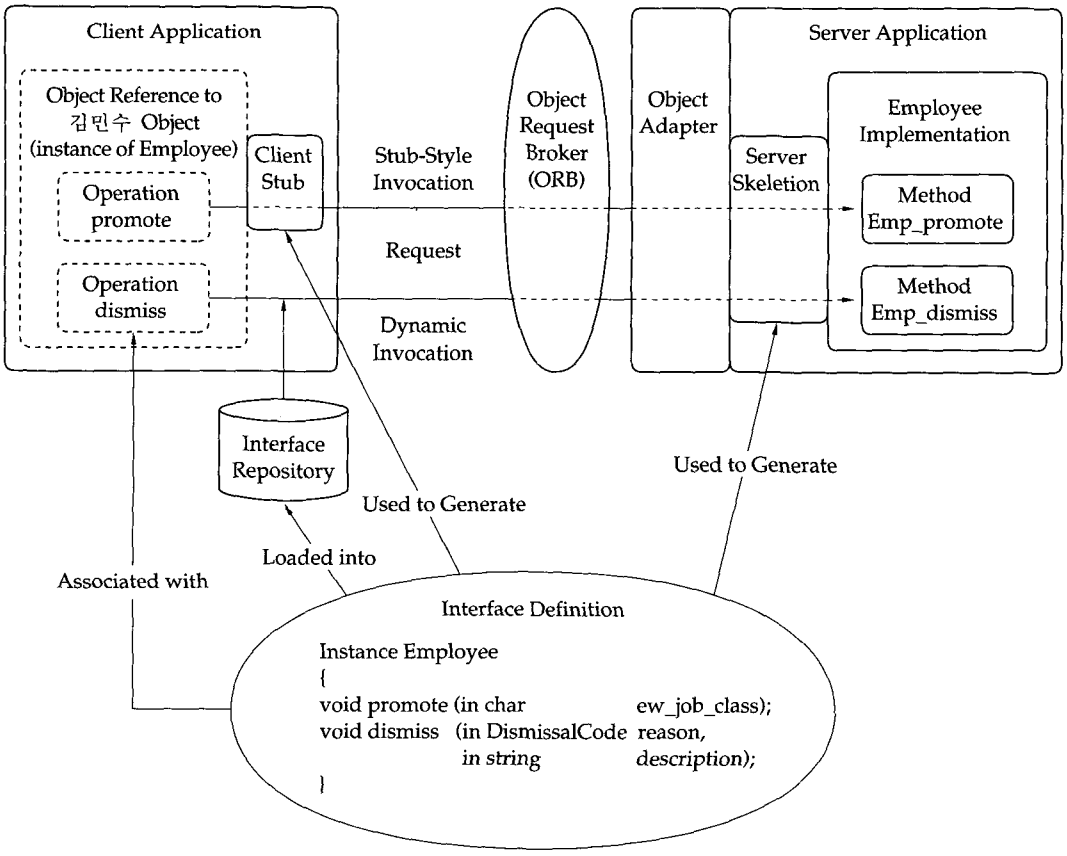


〈그림 3.3〉 클라이언트와 그와 관련된 서버상의 객체구현

있는 연산인 "promote"과 "dismiss" 등이 메소드로 구현되어 있다고 하자.

클라이언트 응용에 연결되어 있는 사용자는 "personnel" 객체에서 어떤 경력을 지닌 인사를 한 회사의 한 부서에 과장으로 채용하고자 한다고 하자. 이 사용자는 클라이언트 실행 화일명과 매개변수로서 필요한 데이터인 고용될 자의 이름(김민수), 고용될 부서(통신부), 그리고 직위(과장)와 일일 근무시간(7시간) 등을 입력하게 된다. 그러면 클라이언트 응용은 자기의 기계에 있는 "김민수"라는 객체 참조(object reference)와 "정보부"라는 객체 참조를 알아내서 클라이언트 측 객체 요청 중개자에 알린다.

클라이언트측 객체 요청 중개자는 서버로 보낼 요청 메시지를 만들기 위해 객체 요청 중개자에 정의된 연산인 CORBA_ORB_string_to_object( ) 연산을 사용해 실지 객체 요청 중개자가 접근할 수 있는 "김민수"라는 "personnel" 객체의 주소와 "정보부"라는 "department" 객체의 주소를 얻어내 "hire" 연산을 수행하도록 하는 요청 메시지를 완성해 서버측 객체 요청 중개자에 보낸다. 서버측 객체 요청 중개자는 고용인 객체 중 "김민수"라는 객체에 대한 구현을 포함하고 있는 서버를 알아내서 그 서버 구현을 활성화하도록 기본 객체 적응자(Basic Object Adapter)에게 시킨다. 그러면 그 서버의 DB에서 "김민수"라는 객체를



〈그림 3.4〉 요청 메시지가 전달되는 과정

읽어 와서 "hire" 연산을 수행하는 메소드를 실행시켜 그 결과를 역으로 보내면 클라이언트 응용은 사용자에게 그 "hire"이라는 연산이 "김민수" 객체에서 성공적으로 수행되었다는 메시지를 디스플레이 해주면 되는 것이다.

4. CORBA와 DCE 비교

CORBA나 DCE는 둘 다 분산 컴퓨팅을 위한 사양으로서 통신 미들웨어 해결책을 제공해 주기 위해서 설계된 분산 시스템을 구축하는데 도움이 되는 도구들이다. 따라서 수행하는 기능에 있어서는 여러가지로 유사할 수밖에 없다. 그러나 근본적인 차이점은 각각이 사용하는 분산 모델이 다르고, 그들 분산 모델이 지원하는 통신 스타일이 다르다고 할 수 있다. DCE 분산 모델은 프로시쥬어 지향(procedure oriented) 분산 모델을 사용한다. 반면에 CORBA 분산 모델은 객체 사용에 기반을 둔 객체지향(object-oriented) 분산 모델을 사용한다.

DCE가 현존하는 소프트웨어들을 통합함으로써 만들어진 반면, OMG는 문서화 작업부터 시작하여 기존의 시스템들이 가져오는 기능들에 대한 부담없이 객체지향의 구조를 구현할 수 있도록 시도하였으므로, 객체 이론 중 어떤 기능이 필요하고, 유용하며 또 해를 끼치는가에 대한 많은 기본적인 문제들을 안고서 표준을 개발하는 접근 방식을 택하고 있다.

DCE의 클라이언트 응용은 RPC로 하여금 서버에 있는 대응 기능을 사상하도록 하는 프로시쥬어들을 호출하기 때문에, 클라이언트 개발자는 무슨 서버가 이용가능한지, 그리고 그 서버에 접근하는데 필요한 RPC 호출에 대한 정확한 지식을 가지고 있어야 한다. 그러나 CORBA 클라이언트는 객체 요청 중개자에 요청을 보내면 객체 요청 중개자가 알아서 처리하도록 하므로 클라이언트 개발자로 하여금 세부적인 지식을 알 필요가 없도록 해준다.

CORBA를 이용하면 기존의 CORBA 클라이언트 응용의 어떠한 코드의 수정없이도 이 클라이언트가 이용할 수 있는 서버를 첨가시키는 것이 가능하고, 또한 역으로 서버의 응용을 수정하지 않고도 기존의 서버를 이용하는 새로운 클라이언트를 첨가시키는 것이 가능하지만, DCE를 이용하면 클라이언트 개발자가 새로운 서버의 향상된 기능을 사용하려할 때마다 클라이언트 코드를 변경해야만 하는 어려움이 있게 된다.

CORBA는 동기적(synchronous) 통신과 비동기적(asynchronous) 통신을 지원하지만, DCE는 동기적 통신을 지원하고, 만일 비동기적으로 통신하려면 다수의 동기적 쓰레드(thread)들을 사용하여 해결해야 한다.

위에서 살펴본 것처럼 우리가 구축하고자 하는 분산 시스템이 자주 확장되어지고 수정되어질 필요가 있고, 또 여러개의 쓰레드를 사용하지 않으면서 비동기적 통신을 원하고, 객체지향 분석과 설계의 장점을 지니도록 하려면 CORBA를 택해야 하지만, 꼭 그럴 필요가 없는 단순하고도 안정성 있는 시스템을 원할 때는 DCE를 택하는 것이 좋은 선택이라 할 수 있다.

5. CORBA 보안

5.1 보안 용어

보안 서비스 용어들 중 데이터 비밀성(data confidentiality)이라는 것은 데이터에 접근할 권한이 없는 자에게는 그 데이터가 노출되어서도, 또 이용되어서도 안된다는 것이고, 데이터 무결성(data confidentiality)이라는 것은 데이터가 화일 시스템에 있건, 메모리에 있건 또는 통신선을 통해 보내지건, 권한 밖의 사용자

로부터 변경과 파괴를 방지할 수 있어야 한다는 개념이다.

사용자가 참으로 권리를 행사할 수 있는지를 시스템에게 확인시켜 주는 것은 식별과 인증(identification and authentication) 개념이고, 접근 제어(access control)이라는 것은 정보 자원에 대한 접근을 허용해 줄 것인가에 관한 것으로 불법적인 방법으로는 자원을 사용 못하도록 하자는 정책으로 임의적 접근 제어(Discretionary Access Control)와 강제적 접근 제어(Mandatory Access Control) 정책이 있다.

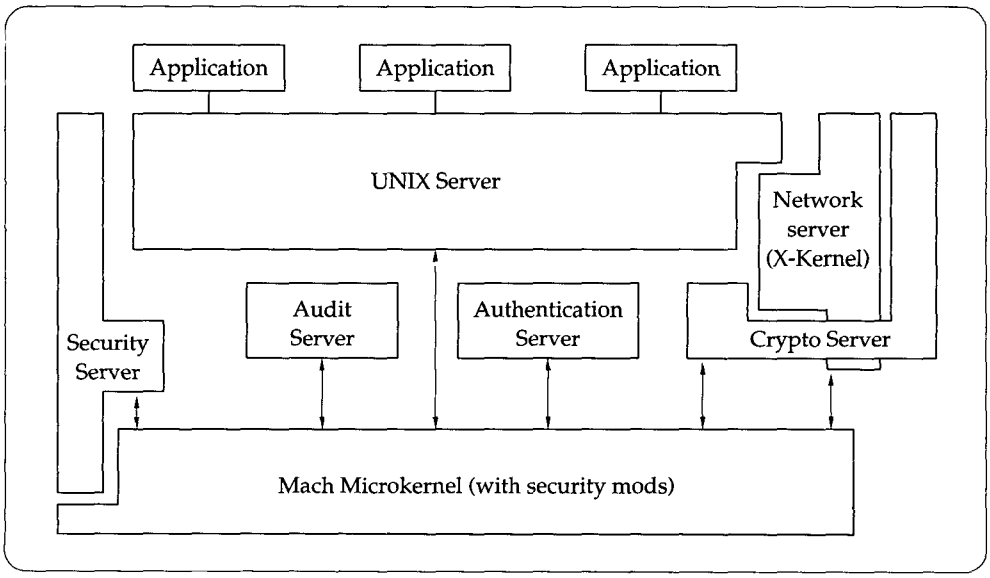
감사(auditing)라는 것은 만일의 사태를 대비하여 보안 관련 사건(event)들을 모아 놓는 개념을 말하며, 부인 봉쇄(non-repudiation)라는 것은 수행한 행위를 나중에 부인하지 못하도록 하는 것으로 디지털 서명이나 공개키 암호화 메커니즘이 구현 방법으로 쓰인다.

또 다른 용어들 중 통신 보안이라는 것은 전송시에 데이터 보호에 관련된 것으로 데이터 무결성이나 비밀성이 이루어져야 하고, 또한 데이터의 노출이나 재시도도 막아야 되는 개념이다.

5.2 SYNERGY

현존하는 보안 모델들과 보안 구조들로는 분산 시스템의 요구 사항을 만족할 수 없으므로 NSA(National Security Agency)에서는 SYNERGY라는 보안 구조를 제안했는데[10], 이 SYNERGY는 이식성이 좋은 마이크로커널(microkernel)에 기반을 둔 것으로 수많은 보안 정책을 지원할 수 있을 만큼 융통성 있고, 네트워크화된 다단계 보안 시스템을 지원하도록 계획되었다.

SYNERGY 구조는 개별적인 서버로 구성되어 있는데, 이 서버들은 기계에 종속되지 않는 공통 인터페이스를 제공해 주고 있는 마이크로커널을 통해서 접근될 수 있다. SYNERGY 구조는 그림 5.1에 나타나 있는바와 같이 6개의 서버로 구성되어 있다.



〈그림 5.1〉 SYNERGY 구조

보안 서버(security server)는 접근 제어를 제공해 주고, 감사 서버(audit server)는 시스템의 모든 영역으로부터 오는 메시지를 받아 감사하도록 하는 것이고, 암호 서버(cryptographic server)는 오가는 메시지의 암호화와 복호화를 담당하도록 한다. 네트워크 서버는 X-커널 프로토콜을 사용하여 안전한 다단계 통신을 제공할 수 있도록 하고 있으며, 인증 서버(authentication server)는 사용자 인증과 시스템 인증을 제공하도록 하고 있다.

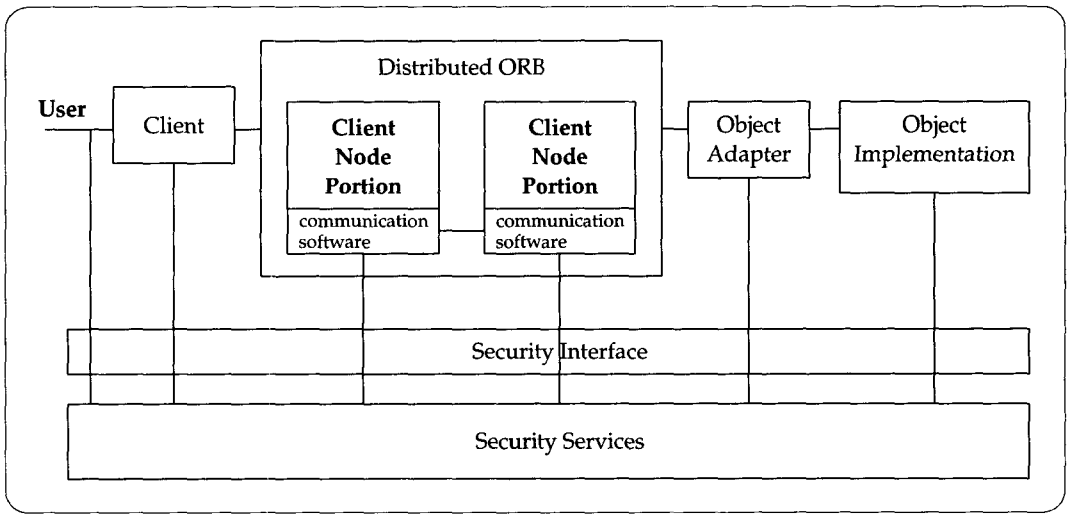
사용자나 프로토콜이 인증 서버나 보안 서버, 암호 담당 서버에 접근하기 위해서는 하부적인 메커니즘이나 기술에 종속되지 않는 일반적인 보안 서비스를 제공해 주는 GSS-API(Generic Security Service Application Program Interface)를 이용하도록 하고 있다.

SYNERGY 구조의 서버들이 GSS-API를 제공하므로 이를 CORBA 구조에 그대로 적용시키면 매우 잘 맞게 된다. 이는 GSS-API가 객체지향의 특성과 단순성의 성질을 지니고 있기 때문이다. GSS-API에서 사용되는 보안 메커니즘들은 클라이언트로 부터 숨겨지게 되므로 클라이언트나 서버가 GSS-API를 사용하기 위해 보안 토큰이나 보안 기술을 일일이 이해할 필요가 없도록 해준다.

5.3 보안 백서

OMG는 분산 객체 컴퓨팅 환경에서 사용자의 실수나 고의적인 침입에 대하여 안전성을 부여하기 위하여 보안구조와 제공되어야 할 보안 서비스들에 대한 작업을 진행하고 있다. 현재 OMG에서는 CORBA의 보안과 관련하여 보안 백서(White Paper On Security)와 보안 객체 서비스인 RFP3를 공표하였다.

백서에서는 분산 객체 컴퓨팅 환경에서 요구되는 보안 성질들을 일반적 요구와 기능적 요구들로 나누어 기술하였고, CORBA의 구성요소들이 보안성질에 대한 요구를 만족하기 위하여 서비스들을 이용하는 구조를 보였다[4]. 그리고 RFP3에서는 CORBA에 제공될 보안 서비스의 사양을 표준화하기 위한 제안을 하고 있다[5].



〈그림 5.2〉 백서의 보안구조

그림 5.2는 OMG 백서 보안 구조를 나타낸 것이다.

이 그림에서 보안 인터페이스의 위치는 제안자의 입장에 따라서 구현상의 방법이 여러 가지 있을 수 있게 된다. 서버 스켈톤 측면에서 보안 서비스를 제공하는 구현 구조를 예로 들어보면, CORBA에서는 사용자 뿐만 아니라 객체들도 접근제어가 이루어져야 하므로, 접근 제어는 객체 요청 중개자 자체가 수행하도록 하거나 객체 적응자(object adapter)에 의해서 행해지도록 하거나, 또는 객체 자신이 하도록 하거나 아니면 여러가지가 혼합된 형태로 접근제어가 이루어지도록 한다. 그러나 객체에 대한 보안등급이 알려져 있지 않는 경우에는 객체 요청 중개자나 기본 객체 적응자에 의해서만 이루어지도록 해야만이 트로이목마 같은 객체가 안전하지 않은 객체 요청 중개자나 기본 객체 적응자에 연결되는 것을 막을 수 있다.

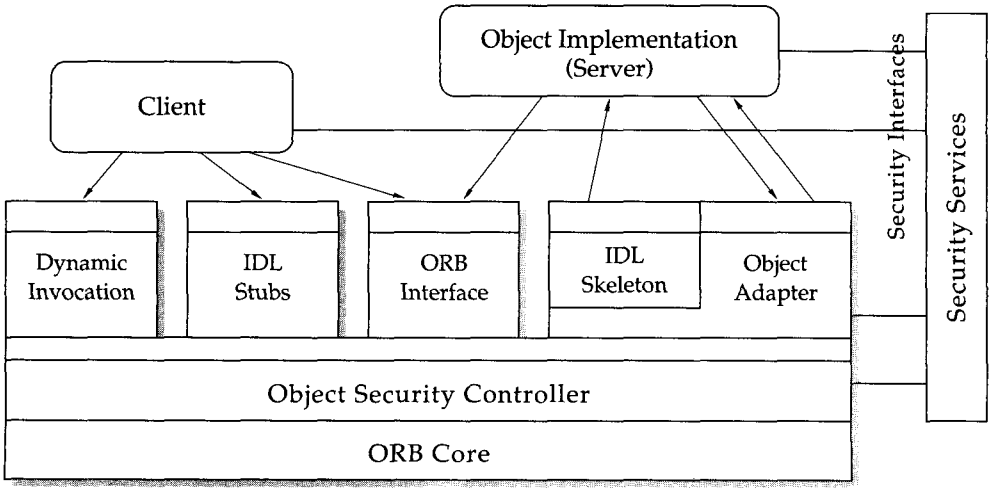
완전히 안전한 구조를 만들기 위해서는 객체 요청 중개자나 객체구현들이 안전한 운영체제에서 수행되어야만 가능하므로 객체 요청 중개자나 운영체제, 통신채널들이나 객체 적응자, 그리고 GSS-API 같은 보안 메커니즘들이 TCB(Trusted Computing Base)의 일부가 되어야 하는 것은 중요한 일이다.

5.4 CORBA 보안구조

OMG의 보안구조에서 보안 서비스들을 제공할 주체를 어떤 것으로 정할 것인가는 아직 확정된 결론이 없지만, CORBA 환경에서 요구되는 보안 서비스를 제공하기 위해서는 안전한 객체 요청 중개자를 설계하는 방법과 현재 구현된 객체 요청 중개자를 이용하여 CORBA에 보안성을 부여하는 방법이 있을 수 있다. 두번째의 방법은 보안서비스를 객체서비스, 공동 설비, 응용 등에 위치시킬 수 있는 것이다.

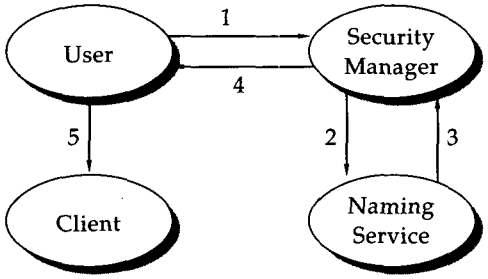
첫번째의 방법에 가까운 시도로는 Robert H. 등이 제안한 보안 구조로서 그림 5.3과 같다[8].

그림 5.3의 구조는 객체 보안 제어기(Object Security Controller)라는 시스템 보안 객체를 ORB 핵심위에 두어 보안 결정을 내릴 수 있



〈그림 5.3〉 CORBA의 보안구조 1

도록 했는데, 객체 보안 제어기가 클라이언트의 요청메시지를 분석해 인증하고, 서버 객체의 접근 제어 정보에 근거하여 요청을 승인할지를 결정하는 것이다.



〈그림 5.4〉 클라이언트의 생성과정

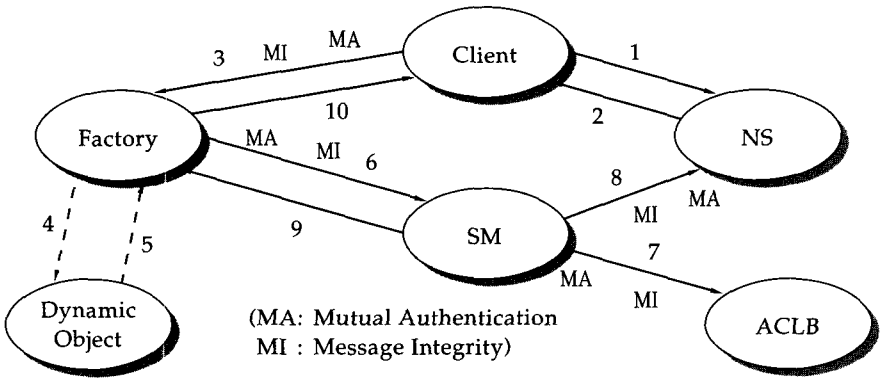
클라이언트의 생성과정은 그림 5.4와 같이 사용자는 보안관리자에게 인증받기 위해 로그인 요청 항목을 보내면 보안관리자는 이름 서버에게 사용자 인증을 위한 정보를 요청한다. 보안관리자는 응답받은 보안 정보 메시지의 항목과 로그인 요청 항목을 비교해보고 같으면 인증한다는 내용의 메시지를 사용자에게 보낸다. 그러면 비로소 사용자는 클라이언트를 생성한다.

동적 객체의 생성은 그림 5.5와 같다. 클라이언트는 원하는 조건의 객체를 실지 생성해 주는 공장(factory)의 참조를 얻기위해 이름 서버로 요청을 보내면 이름 서버는 클라이언트에게 조건에 적합한 공장객체의 참조를 보낸다. 이어 클라이언트는 공장객체에게 제약조건과 함께 객체를 생성해 주라는 요청을 보내면 공장객체는 객체를 생성하고, 생성된 객체는 비밀키와 공개키를 만들어 비밀키는 자기 자신이 안전한 장소에 저장하고, 공개키 등은 공장객체에 보낸다.

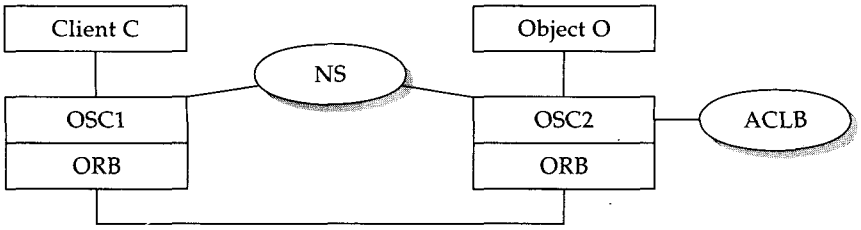
공장객체는 이 객체에 대한 보안 정보를 보안 관리자에게 보내 필요한 안전한 장소에 저장해줄 것을 요청하고 보안 관리자는 이들 보안정보를 이름 서버와 접근 제어 베이스에 저장하고 성공적으로 끝났음을 공장객체에게 알려준다. 마지막으로 공장객체는 생성된 객체의 참조를 클라이언트에게 전달해 준다.

위에서 말한 작업이 이루어진 후에는 그림 5.6과 같이 응용은 보안 서비스를 받을 수 있다.

그림 5.6에서 클라이언트는 원하는 객체와의 인증을 위해 객체 보안 제어기1을 통해 이름 서버로 클라이언트가 정당한 사용자임을 나타내는 증명서를 보낸다. 이름 서버는 클라



〈그림 5.5〉 동적객체의 생성



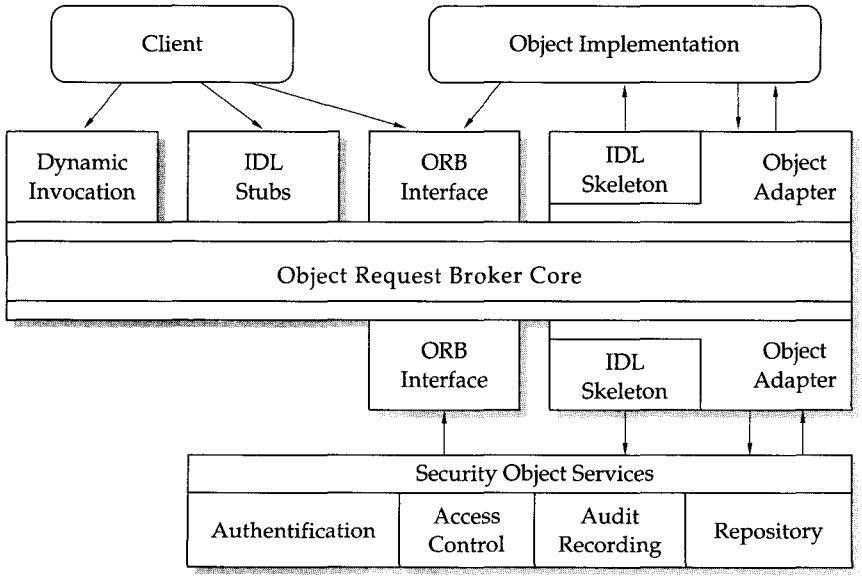
〈그림 5.6〉 응용의 시나리오

한 객체의 공개키를 알아내서 인증에 필요한 토큰을 만들어 클라이언트에 보낸다. 클라이언트는 이 토큰을 원하는 객체가 있는 노드의 객체 보안 제어기2에게 보낸다. 객체 보안 제어기2는 요청메시지의 암호화를 풀어서 클라이언트의 요청메시지가 원하는 객체에 접근될 수 있는지를 판단한 후 허용여부를 결정한다.

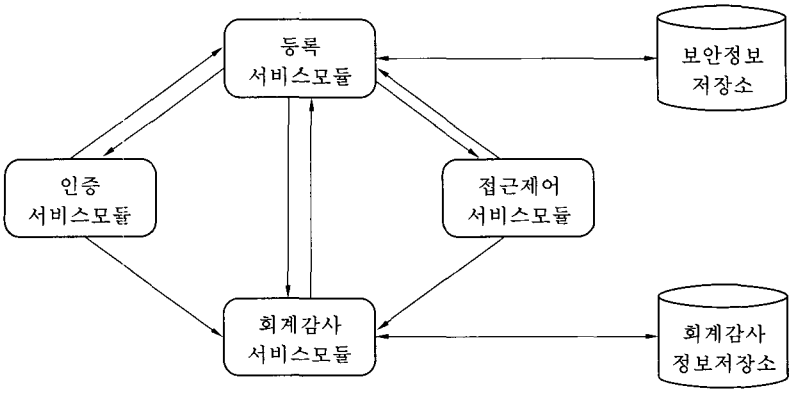
CORBA에 보안서비스를 제공하기 위한 또 다른 방법은 OMG에서 제안한 보안구조를 따르는 것으로 그림 5.7과 같이 다시 나타낼 수 있다.

위의 보안 구조가 보안 서비스를 제공하기 위해서는 몇몇 모듈들로 나누어 보안 서비스를 제공하도록 하는 인터페이스를 정의하면 된다[12].

- 등록 서비스 모듈은 시스템의 모든 보안 정보를 저장하고 있는 보안 정보 저장소를 관리하고 회계감사를 위해서 보안 정보 저장소에 대한 모든 접근을 회계감사 서비스 모듈에 보고한다.
- 인증 서비스 모듈은 보안 정보 저장소로부터 사용자 인증정보를 얻기 위해 등록 서비스 모듈을 이용하고 회계감사정보



〈그림 5.7〉 CORBA의 보안구조2



〈그림 5.8〉 보안서비스 모듈간의 관계

저장소에 회계감사 정보를 저장하기 위해서 회계감사 서비스 모듈을 이용한다.

- 접근 제어 서비스 모듈은 서버로부터의 접근제어 서비스 요청을 처리하기 위해서 등록서비스 모듈을 이용하여 접근제어에 관련된 정보를 얻고 회계감사 정보의 저장을 위해서 회계감사 서비스 모듈을 이용한다.
- 회계감사 서비스 모듈은 다른 서비스 모듈로부터의 회계감사 정보를 제공받아 회계감사 정보 저장소에 저장하고 보안 관리자에게만 회계감사 정보를 제공한다.

이들 서비스 모듈간의 관계를 간단하게 나타내면 그림 5.8과 같다.

보안 서버는 안전한 객체 요청 중계자보다 실제로 구현하기가 더 쉽고, 기존의 보안 서비스 모듈을 수정하지 않고도 새로운 모듈을 확장해 갈 수 있으므로 보안 서버를 확장하기가 용이하여 쉽게 CORBA 환경에 적용할 수 있다는 장점이 있는 반면에 독립된 보안 서버만으로는 완전한 보안기능을 부여하지는 못하는 단점이 있다.

6. 결 론

정보 처리 기술과 정보 통신 기술의 발달은 분산 컴퓨팅이라는 새로운 분야를 낳았고, 이를 객체지향방식과 통합하여 해결하려는 시도로서 분산 객체 컴퓨팅 환경이 만들어졌다. 분산 객체 컴퓨팅 환경의 여러 난립하는 문제들을 해결하기 위한 표준을 위해 OMG에서는 분산 객체 컴퓨팅의 표준 구조로 CORBA를 제안하고 있다. CORBA는 이기종 환경에서 응용 프로그램간의 상호 운용성과 다중 객체 시스템의 상호 연결성을 제공해주는 미들웨어로서 분산 시스템을 구축하는데 도움이 되는 도구이다.

그러나 CORBA가 분산 객체 컴퓨팅 환경의 실제적인 표준으로 자리잡아가고 있는 추세에서 보안에 관한 연구가 거의 이루어 지지 않고 있는 것은 분산 응용에 치명적인 문제를 일으킬 수 있다. 본 연구에서는 OMG/CORBA에 대해 개념적, 그리고 구조적으로 고찰하였으며, 현재 분산처리의 기본 소프트웨어라 할 수 있는 OSF/DCE와 분산 객체 중개자인 OMG/CORBA간의 장단점을 비교했고, 분산 시스템에서 ORB를 사용하는 CORBA 환경에서 보안 문제와 이의 해결방안들에 대해 기술하였다.

안전한 객체 요청 중개자를 설계하기 위해 고려되어야 하는 사항들과 보안 서버에 갖추어야 하는 보안 서비스 모듈들에 관한 연구가 현재 진행되고있다.

참 고 문 헌

[1] Object Management Group, Framingham, "Common Object Request Broker : Architecture and Specification", Document number 91.12.1, 1991.

[2] Object Management Group, Framingham, "Object Management Architecture Guide", Document number 92.11.1, 1992.

[3] Object Management Group, Framingham, "Common Object Services Specification, Volume 1", 1992.

[4] Object Management Group, Framingham, "White Paper on Security", Document number 94.4.16, May 1994.

[5] Object Management Group, Framingham, "Object Services Request For Proposal 3", Jun 1994.

[6] David Chappell, "Distributed Object Computing with CORBA", NETWORLD & INTEROP'94, 1994.

[7] Randy Otte, Paul P., Mark Roy, "Understanding CORBA", Prentice Hall,inc. 1996.

[8] Robert H.Deng, Shailendra K.bhonsle, Weiguo Wang, Aurel A.Lazar, "Integrating Security in CORBA Based Object Architectures", proceedings of 1995 IEEE Symposium on Security and Privacy, Oakland, California, May,1995, pp50-61.

[9] Susan L.Chapin, William R.Herndon, LouAnna Notargiacomo, "Security For The Common Object Request Broker Architecture", Tenth Annual Computer Security Applications Conference, Orlando, Floriad, December, 1994, pp21-29.

[10] Thomas J. Mowbray, "The Essential CORBA : Systems Integration Using Distributed Objects", John Wiley & Sons, Inc. 1994.

[11] 한국전자통신연구소, "국책기술개발 과제 결과 발표회 : 분산시스템 소프트웨어 기술", Feb., 1996.

[12] 노봉남 외 2인, "CORBA 환경에서 보안 서버와 그 인터페이스 설계", 한국통신정보보호학회 논문집, 5권 1호, 1995. 11. pp31-40.

□ 著者紹介

이 영 록(李泳錄, Youngrok Lee)

1986년 전남대학교 계산통계학과 학사
1990년 전남대학교 전산통계학과 석사
1993년~현재 전남대학교 전산통계학과 박사과정

※ 주관심분야 : 분산시스템, 객체지향시스템, 정보보안 등

노 봉 남(盧奉男, Bongnam Noh)

1978년 전남대학교 수학교육과 이학사
1982년 한국과학기술원 석사
1994년 전북대학교 전산학과 박사
1983년~현재 전남대학교 전산학과 교수

※ 주관심 분야 : 객체지향시스템, 통신망관리, 정보보안, 컴퓨터와 정보사회 등