보안 인터페이스의 통합을 위한 객체 포장 모델 및 응용

정회원 김 영 수*, 최 흥 식**

Object Wrapping Model and Application for Integration of Security Interface

Young Soo Kim*, Heung Sik Choi** *Regular Member*

요 약

인터넷의 확산과 아울러 기존의 클라이언트/서버 환경을 발전시킨 분산 시스템이 보편화되면서 시스템을 상호 접속하고 일관되게 통합하기 위한 필요성이 증대되고 있다. 시스템을 통합하는 경우에 가장 문제가 되는 부분이 기존 시스템을 어떻게 재사용할 것인가 하는 것이다. 기존 시스템의 활용과 분산 환경으로의 이전은 기존 서비스를 분산객체로서 포장하는 것에 의해 해결할 수 있다. 코바와 객체 포장은 기존 시스템을 변경하지 않고 클라이언트에 대하여 일관된 표준 인터페이스를 제공하는 미들웨어로써 서비스를 제공할 수 있는 대안이다. 이러한 시스템 통합 기술은 적용하려는 분야에 필요한 기능을 구현하여 쉽게 확장할 수 있도록 응용되어야 한다. 이를 위해 본 논문에서는 코바로부터 분리 통합되는 형태로 관리되는 객체 포장 시스템에 대한 모델을 제안하고 이를 사용하여 보안 인터페이스를 통합하는 코바 응용 시스템을 구현하여 모델의 실용성을 검증하였다.

ABSTRACT

Along with the innovative enhancement of Internet technology and the emergency of distributed systems extended from client-server computing, it becomes indispensible and necessary to integrate and interconnect old legacy systems. Since building a distributed system requires consistency of integration, the proper reuse of incumbent systems is critical to successful integration of current systems to distributed ones. CORBA(Common Object Request Broker Architecture) and object wrapping technique can provide middleware solutions that extend the applications of a legacy system with little modification to the application level while keeping client consistency of standard interface. By using these techniques for system integration it is easier and faster to extend services on application development to distributed environments. We propose a model on object wrapping system that can manage, integrate, and separate the functions delivered from CORBA. We apply the object wrapping model specifically to integration of security system interfaces and also perform a test to verify the usability and the efficiency of our model.

I. 서 론

인터넷의 확산과 더불어 새롭게 개발되는 대부분의 애플리케이션은 클라이언트/서버 구조로 이루어진 분산처리시스템이다. 이렇게 분산되어 있는 시스템들이 서로 다른 네트워크 인터페이스와 프로토콜을 사용하고 있다면 상호운영성과 개발 효율은 저하되고 최악의 경우에는 시스템마다 고립되어 있는 정보의 섬을 유지하게 된다. 코바로 대표되는 분산객체기술은 객체지향의 이점을 분산 컴퓨팅에 적용하는 것에 의해 정보의 섬 문제를 해결하려 하고 있

* 국민대학교 정보관리학과, **국민대학교 비즈니스 IT학부
논문번호: 030113-0317, 접수일자: 2003년 3월 18일

다[3].

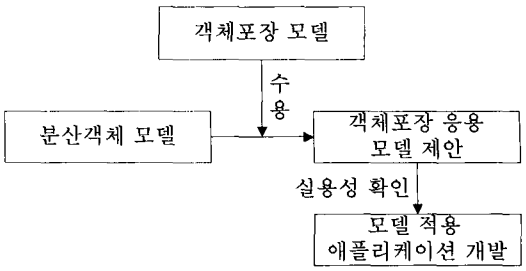
객체지향기술에서는 서비스를 제공하는 서버로서의 객체와 서비스를 이용하는 클라이언트가 객체의 인터페이스를 경계로 명확하게 구별되어 서버의 구현부를 클라이언트로부터 은폐시킨다. 또한 분산 기술은 네트워크 프로토콜을 은폐하고 기존 시스템의 서비스를 포장하는 것에 의해 기존 시스템의 활용을 실현하고 있다.

기존 시스템을 신규 분산객체 시스템으로 전환하기 위해서는 세 가지 방법을 사용할 수 있다[8]. 첫째, 기존 시스템을 폐기하고 분산객체기술을 사용하여 재개발한다. 이러한 접근법은 개발자에게 기존 시스템의 고려에 대한 부담을 감소시키는 반면 새로운 환경에서 새로운 언어를 사용하여 모든 기능이 구현되어야 하기 때문에 많은 비용과 시간이 소요된다. 두 번째 방법은 리엔지니어링 접근법이다. 이 방법은 기존 시스템의 기능을 다시 구현해야 하는 필요성이 없기 때문에 유리한 방법이다. 그러나 코드 전환이 쉽지 않고 이용가능한 툴과 방법이 부족하다는 단점이 있다. 세 번째 방법은 현존 시스템의 구성요소를 포장해서 분산객체 환경에서 이를 호출하여 사용하게 하는 방법이다. 객체포장은 클라이언트에게 서버 애플리케이션을 접근하기 위한 인터페이스를 제공하는 캡슐화의 한 방법이다.

사용자의 변화하는 요구사항과 기술의 변화에 대한 신속한 대응을 필요로 하는 분산시스템의 경우에는 개발지연을 허용하지 않기 때문에 첫 번째와 두 번째 방법은 적합한 방법이 아니다. 세 번째 방법인 객체포장은 현존기술을 사용하여 쉽고 빠르게 구현할 수 있기 때문에 보다 효과적인 방법이다.

분산객체기술인 코바와 객체 포장은 기존 시스템을 변경하지 않고 클라이언트에 대하여 일관된 표준 인터페이스를 제공하는 미들웨어로써 서비스를 제공할 수 있는 대안이다. 이러한 분산객체기술은 적용하려는 분야에 필요한 기능을 구현하여 쉽게 확장할 수 있도록 응용되어야 한다. 이를 위하여 본 논문에서는 <그림 1>과 같은 방법에 따라 코바로부터 분리 통합되는 형태로 관리되는 객체 포장 시스템에 대한 모델을 제안하고 이를 사용하여 기존 시스템을 통합하는 코바 응용 시스템을 구현하여 모델의 실용성을 검증하였다.

본 논문은 다음과 같이 구성한다. 2장에서는 본 논문의 모델과 응용의 이해를 위해 코바기술과 객체포장기술을 간략하게 소개한다. 3장에서 기존 시스템을 코바 기반 분산 환경으로 전환하기 위한 객체포장 모델과 응용을 제안하고 4장에서는 모델의 적합성을 검증하기 위하여 통합시스템을 구현한다. 5장에서는 결론과 시사점을 기술한다.



〈그림 1〉 연구 모델

Ⅱ. 분산객체 기술과 객체포장 모델

현재의 이질적인 분산환경에서 분산객체기술과 객체포장 기술은 분산시스템의 통합을 위하여 폭넓게 사용된다. 서로 다른 하드웨어, 운영체제, 네트워크에 위치하며 서로 다른 언어로 쓰여진 소프트웨어에게 상호운영성을 제공하는 코바의 이점과 기존 시스템에 대한 잘 정의된 인터페이스를 제공함으로써 타 시스템에서 기존 시스템의 서비스를 이용할 수 있도록 해주는 객체포장기술을 연결하여 사용함으로써 분산시스템의 통합은 효과적으로 실현할 수 있다.

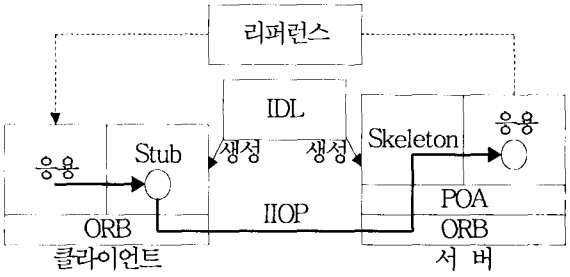
2.1 분산객체 모델

분산객체기술로 대표되는 코바(CORBA: Common Object Request Broker Architecture)는 응용계층에서 애플리케이션을 통합하기 위하여 비영리단체인 객체관리그룹(OMG: Object Management Group)에 의하여 개발된 표준안으로 다양한 언어와 구현, 플랫폼을 통합하는 것을 목표로 한다[6].

<그림 2>는 코바의 클라이언트-서버 모델를 표현하고 있다. 객체 리퍼런스(Object Reference)는 객체의 식별을 위한 정보를 포함한 데이터이고 객체 생성시 만들어지기 때문에 클라이언트는 생성된 객체 리퍼런스를 획득하여야 객체의 서비스를 이용할 수 있다. 클라이언트는 획득한 객체 리퍼런스를 사용하여 원격 객체를 마치 동일한 기억공간에 있는 것처럼 사용한다. 이는 운영체제에 의하여 지원되는 가상 메모리 기법처럼 가상의 실체인 코바 객체를 클라이언트가 호출하면 구현객체로의 매핑은 ORB

(Object Request Broker)에 의하여 수행된다.

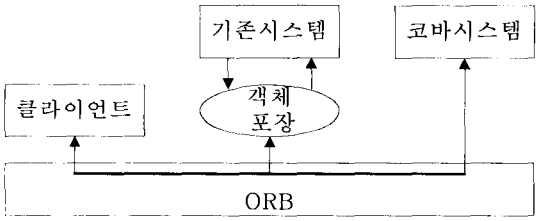
클라이언트에 의한 서버 객체의 호출은 IDL (Interface Definition Language)화일를 컴파일하여 생성된 클라이언트 스터브(Stub)와 서버로서의 역할을 수행하는 스켈리톤(Skeleton)이 서버 객체에 대한 메서드의 호출과 응답을 메시지의 형태로 포장하고 언마샬링한다. ORB는 메시지에 IIOP (Internet Inter ORB Protocol)헤더를 부착하거나 역캡슐화하고 POA(Portable Object Adapter)는 서버 객체에 대한 인스턴스의 생성과 제거를 수행한다.



〈그림 2〉 코바 클라이언트-서버 모델

2.2 객체 포장 기법

객체포장은 <그림 3>과 같이 기존 시스템과 신규 시스템을 통합하기 위한 방안을 제공한다. 객체 포장은 계층화(Layering), 마이그레이션(Migration), 재공학(Reengineering), 캡슐화(Encapsulation)와 같은 기법을 사용한다[4].



〈그림 3〉 객체포장 시스템

① 계층화는 상이한 개발 환경을 위하여 응용 프로그램 인터페이스(API: Application Program Interface)를 매핑하는 작업이다. 이러한 API의 계층 구조는 기능 변경 등이 발생될 경우 인접 계층에 대해서만 영향이 있을 뿐 그 외의 계층은 각각 독립적으로 정의되어 있으므로 시스템 전체에 영향을 미치지 못하는 장점을 제공한다.

② 마이그레이션은 기존 시스템에서 사용하는 데이터를 객체지향 DB와 같은 다른 데이터 모델로 이동하는 것을 말한다. 때때로 상이한 공급업자간의 데이터베이스 시스템과의 매핑을 위해 사용된다. 객체 포장은 다른 데이터베이스에로의 접근을 제공하기 위하여 계층화된 코드를 추가하는 것이다. 이러한 접근법은 기존시스템의 기능보다 데이터의 재사용에 초점을 맞춘다.

③ 재공학은 시스템의 유지 비용의 감소와 성능의 향상 그리고 유지보수의 용이성을 제공하기 위하여 수행된다. 재공학 활동은 시스템의 분석, 재구성(Recomposition), 역공학(Reverse Engineering), 이식(Porting) 활동 등으로 구분할 수 있다. 소프트웨어 재공학의 일반적인 개념은 데이터와 기능들의 개조 및 개선을 통해 유지 보수 용이성을 향상시키자는 것이다. 기존 시스템의 어떤 부분을 분산객체 기술을 사용하여 재구현할 것인가는 기존 시스템에 대한 분석을 바탕으로 하여 결정하여 분산객체기술에 의한 신규 서브시스템을 설계한 후 기존시스템의 나머지 컴포넌트와 인터페이싱되도록 하여야 한다. 객체포장은 기존 시스템의 컴포넌트를 객체지향 컴포넌트로 대체하여 사용할 수 있게끔 한다.

④ 캡슐화는 객체포장의 가장 일반적인 형태로 인터페이스와 구현을 분리한다. 캡슐화는 프로그래밍 언어, 위치 운영체제, 알고리즘 데이터 구조체의 차이를 IDL를 사용하여 숨긴다. 캡슐화는 기존 시스템의 구현코드에 대한 접근이 불가능하여 변경할 수 없을 때 사용되어질 수 있다. 이러한 경우에 모든 액세스는 캡슐화에 의하여 포장된 객체를 통하여 제공된다. 캡슐화는 또한 기존시스템을 컴포넌트로 분할하는데 사용될 수 있다. 각각의 컴포넌트는 별도로 캡슐화하여 분산객체기술을 사용하여 재통합된다.

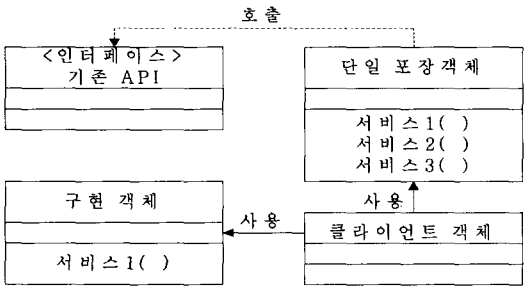
2.3 객체 포장 모델

기존 시스템을 점진적으로 신규 시스템으로 교체하는 방법에는 단일 포장객체를 사용하여 기존 시스템의 모든 서비스를 일괄 포장하는 방법과 별도의 포장객체를 사용하여 개개 서비스를 포장하는 다중 객체포장 모델이 있다.

① 단일 객체포장 모델

기존 시스템을 코바 시스템으로 통합하는 단일 객체포장 모델은 <그림 4>와 같다[5]. 이 모델에서 단일 포장객체는 기존 API를 호출하는 모든 메서드를 포함하고 있다. 클라이언트 객체가 기존 시스템

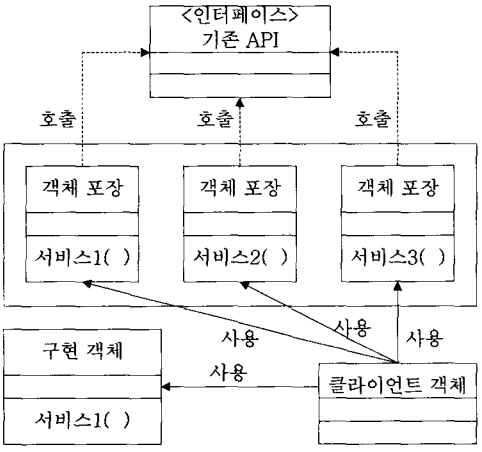
의 기능을 필요로 하는 경우 포장객체에게 메시지를 보내면 포장객체는 기존 시스템의 API를 호출한다. 또한 기존 서비스를 확장하고 있는 구현개체를 개발하여 이를 이용할 수 있다. 즉 단일 객체포장 모델은 기존 시스템의 모든 서비스를 포장하는 대신 유용한 서비스만을 선별하여 일괄 포장하는 메커니즘을 사용할 수 있다.



〈그림 4〉 단일 객체포장 모델

② 다중객체 포장 모델

다수 포장 객체를 사용하여 기존시스템을 신규시스템으로 통합하기 위해서는 포장할 기존 시스템의 컴포넌트를 재공학 기법을 사용하여 분리하는 것은 중요하다[7]. <그림 5>는 다수 포장 객체를 사용하여 기존 시스템이 수행하는 특정 서비스를 분리하여 캡슐화하는 것을 보여준다. 이 모델에서의 포장객체는 기존 시스템의 API를 분리하여 구현함으로써 서비스를 분리하여 제공한다. 클라이언트 객체가 이용하고자 하는 서비스를 캡슐화하고 있는 포장객체를 호출하면 포장객체는 기존 시스템의 API를 호출한다. 기존 서비스에 대한 완전 대체로 신규 서비스가 구현되는 경우에는 해당 객체포장은 폐기된다.



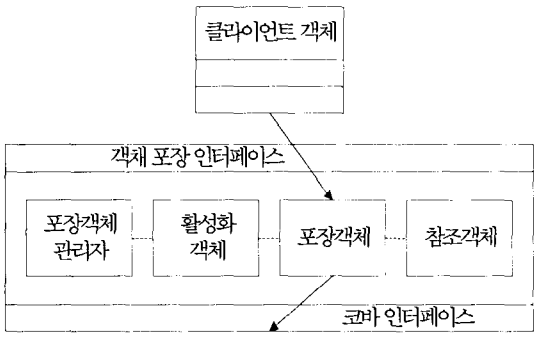
〈그림 5〉 다중 객체 포장 모델

Ⅲ. 객체포장 응용 모델

3.1 시스템 아키텍처

기존 시스템과 분산객체시스템의 통합을 위하여 미들웨어계층으로 기존의 객체포장 모델을 사용하여 통합하는 경우에 애플리케이션 개발자는 포장객체를 관리하는데 많은 시간과 노력을 필요로 한다. 더욱이 기존 보안 시스템의 인터페이스를 객체포장 모델을 사용하여 통합하는 경우에는 보안에 대한 전문지식을 애플리케이션 개발자는 구비하고 있어야 한다[10]. 이는 애플리케이션 개발자에게 상당한 부담으로 작용한다. 따라서 개발자의 역할 분리와 포장객체의 통합 관리를 위한 객체포장을 위한 서브 시스템 모델을 제안한다. 이는 개발의 복잡성을 감소시키고 서비스의 품질을 향상시키는데 많은 도움을 준다.

<그림 6>은 객체포장을 응용한 서브 시스템에 대한 모델을 묘사하고 있다. 응용 모델의 구성요소는 포장객체 관리자, 활성화객체, 포장객체, 그리고 참조객체로 구성되어 있다. 클라이언트 객체는 참조객체로부터 포장객체에 대한 리퍼런스를 획득하여 기존 시스템의 서비스를 이용한다. 포장객체 관리자는 포장객체에 대한 인스턴스(Instance)를 생성하고 활성화객체로 하여금 활성화 정보를 유지하게 한다. 인스턴스화된 포장객체는 기존 시스템의 서비스를 제공한다.

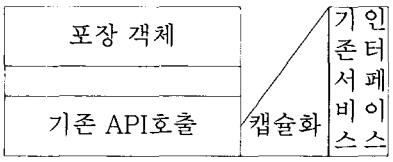


〈그림 6〉 객체 포장 응용 시스템 모델

3.2 시스템 컴포넌트

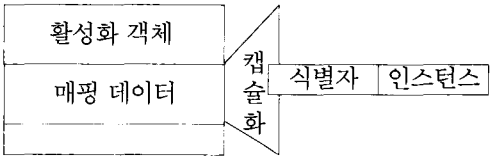
포장객체는 <그림 7>과 같은 구조를 사용하여 기존 시스템의 서비스에 대한 인터페이스를 캡슐화하고 있는 객체이다. 클라이언트 객체는 포장객체에 대한 인스턴스를 통하여 기존 시스템의 서비스를

이용한다.



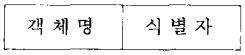
〈그림 7〉 포장 객체의 구조

활성화객체는 <그림 8>과 같고 포장객체의 식별자와 포장객체의 인스턴스를 매핑하는 기능을 수행하는 컴포넌트로서 포장객체의 인스턴스 생성과 제거에 중요한 역할을 담당한다.



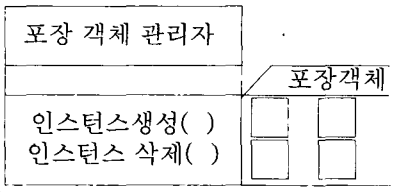
〈그림 8〉 활성화 객체의 구조

<그림 9>는 참조객체의 리퍼런스를 표현하고 있다. 참조객체는 클라이언트 객체가 포장 객체를 통해 기존시스템의 서비스를 이용할 수 있도록 해준다. 즉 클라이언트 객체가 기존 시스템의 서비스를 캡슐화하고 있는 포장 객체에 대한 참조객체를 획득하여 기존 시스템의 서비스를 이용한다.



〈그림 9〉 참조객체의 구조

포장객체 관리자는 <그림 10>과 같고 클라이언트의 요청에 의하여 포장객체의 인스턴스를 동적으로 생성 관리하는 책임을 수행한다. 즉 클라이언트의 요청 메시지에 있는 포장객체의 식별자가 활성화객체에 등록되어 있는지를 검색하여 존재하지 않으면 객체식별자를 등록하고 포장 객체의 인스턴스를 생성하여 매핑시켜 주는 역할을 수행한다.

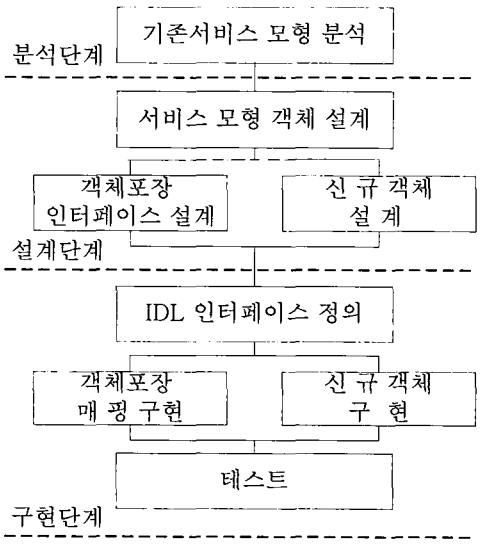


〈그림 10〉

포장 객체 관리자의 구조

Ⅳ. 응용 시스템의 설계 및 모델의 검증

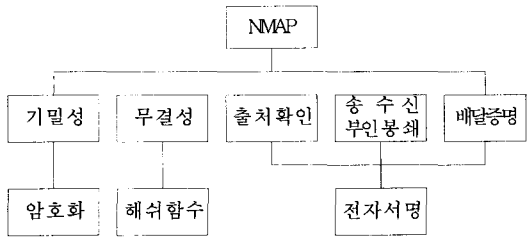
기존 시스템의 객체포장 응용 모델에 대한 적용 사례로서 메시지 보안 시스템의 인증 프로토콜 설계 및 구현[11]에 관한 연구 개발 시스템인 NMAP (New Message Authentication Protocol)을 선정하였다. 객체포장 응용 시스템의 개발 과정은 <그림 11>과 같이 기존 서비스 분석, 서비스모형 객체 설계 및 객체포장과 신규 인터페이스 설계 그리고 기존 시스템의 인터페이스 구현으로 구성된다[2]. 분석단계에서는 기존 시스템이 제공하는 서비스를 분석하여 재사용할 영역과 수정할 부분을 분리하여 정의한다. 설계단계에서는 기존 시스템의 API를 객체 단위로 모듈화하고 수정이 필요한 API에 대해서는 새로운 객체를 설계한다. 구현단계에서는 이들 객체를 코바의 IDL 인터페이스를 사용하여 캡슐화하고 구현한다.



〈그림 11〉 객체포장 응용 개발과정

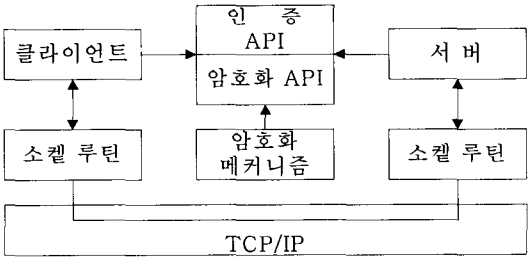
4.1 기존 NMAP 시스템의 서비스 구조

NMAP는 공개되어 있는 정보만을 이용하여 암호화 메시지를 구성하여 불특정 다수에게 메시지를 안전하게 전송할 수 있는 실체 기반 암호화 프로토콜[1,9]로 메시지에 부가되어 보안성을 제공해주는 인증 헤더의 설계를 구현한 시스템으로 <그림 12>와 같이 기밀성, 메시지 출처확인, 무결성, 송수신 부인봉쇄, 배달증명 서비스를 제공한다.



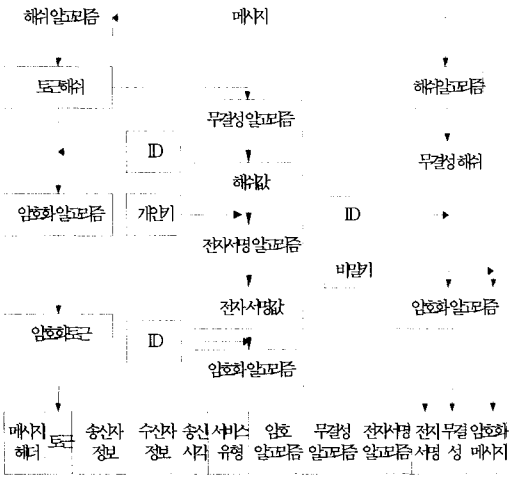
〈그림 12〉 NMAP의 서비스 모델

NMAP 시스템의 기본 구조는 <그림 13>과 같이 암호화 메커니즘과 소켈 인터페이스를 사용하여 송신측 클라이언트와 수신측 서버는 인증정보가 탑재된 메시지를 전송하여 인증을 실현하고 있다.



〈그림 13〉 NMAP의 기본 구조

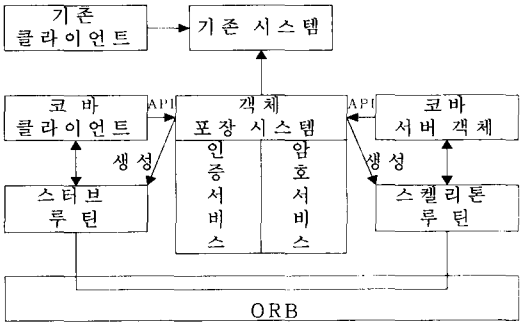
NMAP는 암호방식에 기초를 둔 인증 프로토콜로 <그림 14>와 같이 암호화 시스템을 통하여 전송할 메시지를 구성한다. 송신자가 기밀성 서비스를 요청하면 메시지를 비밀키로 암호화하고 무결성 서비스를 요청하면 메시지와 토큰의 해쉬코드를 계산하여 토큰을 형성한다. 또한 디지털 서명 서비스를 선택하면 토큰의 해쉬 값에 대해 디지털서명 값을 계산한다. 메시지 전송시점에서 일괄적으로 전자봉투화하여 전송함으로써 개인키를 가지고 있는 수신자만이 이를 열람하여 메시지를 처리할 수 있도록 하고 있다.



〈그림 14〉 NMAP 프로토콜 구조

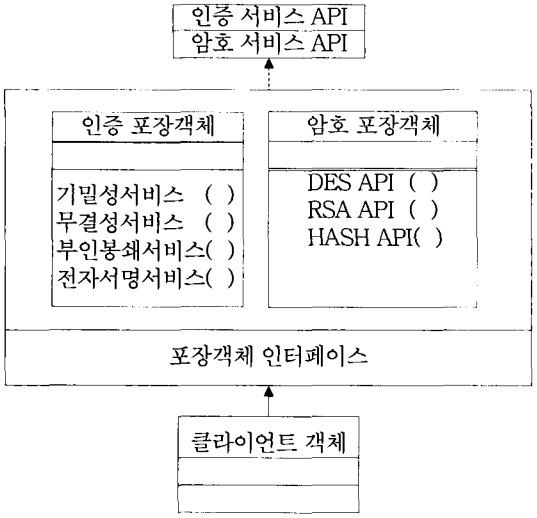
4.2 코바 응용 시스템의 설계

<그림 15>는 기존 시스템의 서비스 제공을 위한 인터페이스와 설계 내용을 토대로 제안 모델에 따라 재구성한 시스템의 구조이다. 기존 클라이언트는 기존 시스템에 대한 서비스를 변경없이 그대로 이용한다. 반면에 코바 클라이언트는 기존 시스템의 인증서비스와 암호서비스를 캡슐화하고 있는 객체포장시스템을 통하여 기존 시스템의 서비스를 이용하고 있다.



〈그림 15〉 코바 응용 시스템 구조

<그림 16>은 다수 포장 객체를 사용하여 기존 시스템이 수행하는 특정 서비스를 분리하여 캡슐화하고 있다. 기존 시스템의 기밀성서비스, 무결성서비스, 부인봉쇄서비스, 전자서명서비스는 인증 포장객체로 캡슐화하고 DES(Data Encryption Standard), RSA(Rivest-Shamir-Adleman), 해쉬(Hash)등의 암호



〈그림 16〉 객체 포장화한 NMAP 서비스

서비스는 암호 포장객체로 캡슐화하여 클라이언트 객체에게 기존시스템의 서비스를 이용하도록 하고 있다.

<그림 17>은 객체포장시스템을 이용하는 클라이언트와 서버 객체간에 송수신되는 메시지 토큰의 구성을 보여주고 있다. 메시지 토큰은 인증정보를 탑재할 수 있도록 OMG의 IDL로 만든 인터페이스

메시지토큰											
식별정보			서비스	메커니즘의 유형			키정보	세션정보	해쉬 시퀀스		
송신식별자	수신식별자	송신시각	서비스유형	공개키알고리즘확인자	비밀키알고리즘확인자	해쉬알고리즘확인자	암호화초기값	비밀키	토큰해쉬값	무결성해쉬값	디지탈서명값

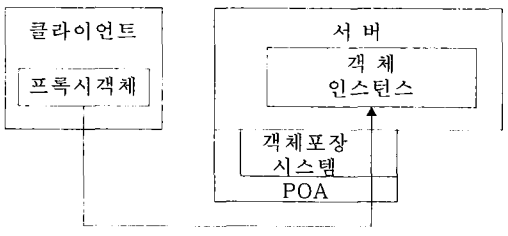
〈그림 17〉 메시지 토큰 구조

에 정의되어 있다. 송?수신자의 식별자와 송신시각을 나타내는 타임스탬프 와 서비스의 유형 그리고 공개키 암호화 알고리즘, 비밀키 암호화 알고리즘, 해쉬 알고리즘의 확인자를 규정한다. 그리고 메시지 토큰 자체를 해쉬 함수로 계산한 해쉬 코드를 첨부하여 토큰의 변경여부를 확인할 수 있도록 하였다. 해쉬 계산 과정은 메시지 토큰내의 인증정보로부터 토큰 해쉬를 만들고 메시지로부터 무결성해쉬를 산출하고 토큰해쉬와 서명정보를 결합하여 디지털서명 해쉬를 생성하여 전송함으로써 수신자가 인증 서비스를 검증하도록 하였다.

4.3 모델의 검증

객체 포장 응용 모델의 실용성을 확인하기 위하여 어댑터 방식과 래퍼방식을 사용하여 코바응용시스템을 구현하고 비교분석하였다. <그림 18>는 객체포장시스템을 코바시스템에 플러그인(Plug in)하여 사용하는 어댑터 접근 방식을 보여주고 있다. 어댑터 접근 방식은 객체포장시스템을 POA의 구현 클래스로부터 직접 상속을 받도록 선언하여 구현한다.

어댑터 접근 방식을 사용한 코바 시스템은 포장객체에 대한 클라이언트 요청이 있는 경우에 자동으로 포장객체의 인스턴스를 생성하여 서비스를 제공하기 때문에 처리속도의 향상을 높일수 있는 반면 영속객체의 형태로 객체지향형 DB에 저장하는 경우에 불필요한 상속클래스의 내용까지도 저장되는 단점이 존재한다.

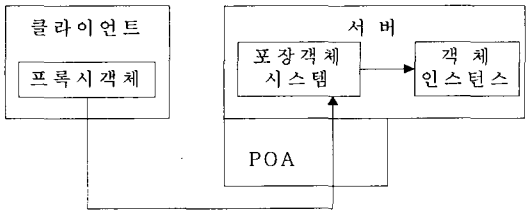


〈그림 18〉 Adapter 접근 방식

<그림 19>는 객체포장시스템을 구현클래스의 대표(delegation) 객체로 선언하여 인터페이스를 구현하는 래퍼 접근 방식이다. 래퍼 접근방식은 인터페이스 클래스에 대한 요청을 구현클래스에 전달하는 대표기법을 사용한다. 대표기법을 사용하는 시스템

에서 각 객체들은 상속의 관계를 나타내는 부모리스트를 유지하는 대신 대표리스트를 유지한다. 따라서 대표기법의 핵심은 임의의 객체가 처리할 수 없는 메시지를 잠정적으로 처리 가능한 객체에게 전가한다는 점이다.

래퍼 접근 방식을 사용한 코바 시스템은 포장 객체에 대한 클라이언트 요청이 있는 경우에 포장객체에 대한 인스턴스 과정을 필요로 하기 때문에 서비스의 처리속도는 약간 지연되는 반면 포장객체를 객체지향형 DB에 저장하는 경우에 상속을 사용하지 않기 때문에 포장객체의 내용만을 저장할 수 있는 이점이 있다.



〈그림 19〉 래퍼 접근 방식

객체 포장 응용 모델의 실용성을 확인하기 위한 모의실험은 어댑터 방식과 래퍼방식에 의한 구현과의 비교를 통하여 수행하였고 셀러론 850MHz와 윈도우즈 2000 운영체제 환경하에서 메시지의 크기와 키의 길이를 상이하게 하여 인증 메시지를 구성하는데 소요되는 시간을 측정하여 실험하였다. 파일은 영문자의 알파벳을 사용하여 100K, 200K, 300K, 400K, 500K 바이트 크기별로 RSA 키 길이는 512비트, 1024비트, 2048비트로 구별하여 실험하였다.

<표 1>에서 보는 것과 같이 코바 시스템에 플러그인하여 객체포장시스템을 구현하는 어댑터 방식이 대표기법을 사용하여 객체포장시스템을 구현하는 래퍼접근방식보다 인증 메시지를 구성하는데 약간 우수하다는 것을 알 수 있다. 이는 어댑터 접근방식의 경우에는 포장 객체의 인스턴스 과정이 불필요한 반면 래퍼접근방식의 경우에는 래퍼 객체의 인스턴스 과정이 서버에서 수행되어야 하기 때문에 수행속도의 면에서 어댑터 접근방식의 결과치가 약간 우수하게 나왔음을 알 수 있다.

〈표 1〉 인증 메시지 제출 시간

구분	Wrapper			Adapter			Wrapper-Adapter
	512	1024	2048	512	1024	2048	
100K	0.43	1.03	2.84	0.42	1.01	2.82	0.0158
200K	0.87	2.06	5.68	0.83	2.03	5.65	0.0338
300K	1.30	3.10	8.52	1.25	3.05	8.48	0.0498
400K	1.74	4.13	11.3	1.67	4.06	11.3	0.0676
500K	2.17	5.16	14.2	2.09	5.08	14.1	0.0836

V. 결 론

코바와 객체 포장 기술은 기존 시스템을 변경하지 않고 클라이언트에 대하여 일관된 표준 인터페이스를 제공하는 미들웨어로서 서비스를 제공할 수 있는 대안이다. 코바는 상호 운용성, 프로그램의 일관성, 코드 중복 회피, 유지보수의 용이성 등 많은 이점을 제공한다. 그리고 객체포장기술은 기존 시스템의 내부구조를 클라이언트에게 은폐시킬 수 있기 때문에기존 시스템과의 통합을 위한 근간을 제공한다.

이러한 코바와 객체포장 기술은 적용하려는 분야에 필요한 기능을 구현하여 쉽게 확장할 수 있도록 응용되어야 한다. 이를 위하여 객체포장 응용 모델을 제안하고 이의 실용성을 확인하기 위하여 기존 시스템의 사례로 메시지 보안 시스템을 선정하여 모델링을 수행하여 실험실에서 모의 실험하였다. 어댑터 접근 방식과 래퍼 접근 방식에 의하여 구현한 코바 통합시스템의 비교 결과치는 성능면에서 차이가 거의 없음을 확인하였다.

두 가지 접근 방식은 각기 장단점을 가지고 있기 때문에 상황에 따라 적절하게 사용하여 기존 시스템의 통합과 개발의 용이성을 실현하여야 한다. 또한 래퍼 접근 방식의 경우에는 래퍼 객체의 인스턴스 과정이 필요하기 때문에 어느 정도의 처리속도의 저하는 불가피하지만 인프라의 성능 향상에 따라 그 차이가 유의하지 않은 수준으로 갈 것이다. 연구의 한계로는 기존 시스템에 존재하는 기능상의 한계를 그대로 상속하고 있으며 기존시스템이 제공하는 서비스를 고정되게 필요하다는 가정하에 기존 서비스를 확장하고 있는 구현 객체를 모델링하지 않고 있다는 점이다. 향후에는 객체포장 응용 시스

템의 개발을 위한 자동화 도구에 대한 연구와 개발을 수행할 필요가 있다 .

참 고 문 헌

[1] Boneh, D. and M. Franklin, "Identity based encryption from the Weil paring", Advances in Cryptology: Crypto, 2001(LNCS 2139), 2001, pp. 213-229

[2] DaeSeug Kang, NamHong Hong and SunHyung Cho. "Object Wrapping as a special middleware of distributed computing in internet environments", '97 Internatioanl Conference Multimedia Database on Internet, 1997, pp. 302-318

[3] Ganti, N. and W. Brayman, The Transition of Legacy Systems to a Distributed Architecture, John Wiley & Sons, 1995

[4] Mowbray T. J. and R. Zahavi, The Essential CORBA: System Integration Using Distributed Objects, John Wiley & Sons, 1995

[5] J. Mowbray, Thomas, and Raphael C. Malveau, CORBA Design Pattern, John Wiley & Songs, New York, 1997

[6] Orfali, R. and D. Harkey, Client/Server Programming with JAVA and CORBA, John Wiley & Sons, 1998

[7] Ronald, C. E. Aronica and Ronald, Jr. Rimel, "Wrapper Your Legacy Systems." in Datamation, June 15, 1996

[8] Sneed, H.M. and R. Majnar, "A Case Study in Software Wrapping" in Proc. of ICSM'98, ieee Computer Society, 1998, pp. 86-93

[9] Shamir, A., "Identity-based cryptosystems and signature schemes", Advances in Cryptology : Crypto, 1984(LNCS 196), 1985, pp 47-53

[10] Souder, T. and S. Mancoridis, "A Tool For Securely Integrating Legacy Systems into a Distributed Environment", in Proc. of Sixth Working Conf. on Reverse Engineering, 1999, pp47-55

[11] 김영수, 메시지보안시스템의 인증 프로토콜 설계 및 검증" ,박사학위논문, 국민대학교 대학원, 2003.

김 영 수 (Young Soo Kim) 정회원

1989년 2월 : 전북대학교 회계학과 졸업 (경영학 학사)

1992년 2월 : 경희대학교 경영학과 졸업 (경영학 석사)

2003년 8월 : 국민대학교 정보관리학과 졸업(정보관리학박사)

<주관심분야> 전자상거래, 인터넷 응용, 분산정보시스템, 정보보안

최 흥 식 (Heung Sik Choi) 정회원

1983년 2월 : 한양대학교 산업공학과 졸업

1985년 2월 : 한국과학기술원 경영과학과 석사

1991년 2월 : University of Rochester, 경영학 석사

1995년 2월 : University of Rochester, Computers and Information Systems, 경영학박사

1985년 3월 : 1988년 6월:데이콤정보통신연구소 연구원

1995년 3월 : 현재 : 국민대학교 비즈니스IT학부 교수

<주관심분야> 통신경영, 통신정책전략, 네트워크 설계