2002년 한국정보처리학회 춘계학술발표논문집 제 9 권 제 1 호

자바 카드에서의 원격 메소드 호출

김영진*, 정용화*, 정교일**
*한국전자통신연구원 생체인식기술연구팀
** 한국전자통신연구원 정보보호기반연구부
e-mail : youngjk@etri.re.kr

Remote Method Invocation in Java Card

Young-Jin Kim*, Yong-Wha Chung**, Kyo-Il Chung*
*Biometrics Technology Research Team, ETRI
**Information Security Basic Department, ETRI

요 약

자바 카드 기술은 자바 기술을 스마트 카드에 최적화시키면서 자바 카드 플랫폼을 상위 자바 플랫폼에 접합시키는 방향으로 발전하고 있는 추세이다. 최근 발표된 SUN 의 자바 카드 2.2 에서 지원되는 카드(서버)와 카드 리더(클라이언트)간의 원격 메소드 호출 인터페이스가 대표적인 예이다. 자바의 원격 메소드 호출은 서버와 클라이언트간의 상호 정의된 원격 객체의 메소드 호출에 대한 인터페이스를 제공하여, 다른 주소에 존재하는 자바 가상 기계들 사이에 메소드 호출이 가능하게 하는 것이다. 본 논문에서는 자바 카드에서의 원격 메소드 호출 인터페이스에 대해 살펴보고, 구현 및 이용 방법에 대해서도 살펴보고자 한다.

1. 서론

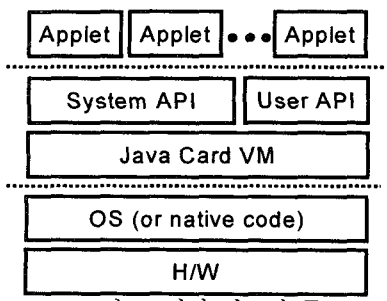
자바 카드는 자바 언어 및 수행 환경을 스마트 카드에 최적화하여 탑재한 것으로, 현재 유럽의 GSM 내의 SIM 카드 등에서 많은 사용량을 보이고 있으며 점차 시장 점유율이 커질 것으로 예상된다. [1]

자바 카드는 전통적인 스마트 카드에 비해 플랫폼간에 이진 코드의 이식성(portability) 즉, 상호 운용성(interoperability)이 뛰어나며, 타입 검사, 접근 제한 등에 의한 보안성(security)을 지닌, 객체 지향 언어인 자바 언어를 스마트 카드 환경에 대해 최적화하고 있다. 구조는 H/W 와 그 위의 전용 운영체제 또는 네이티브(native) 코드 계층, 자바 카드 가상 기계, 라이브러리인 API(Application Programming Interface)와 응용 프로그램인 애플릿(Applet)이 존재한다. (그림 1)은 이러한 내용을 보이고 있다.

자바 카드는 SUN 에 의해 1996 년 산업 표준이 발표된 이후에 지속적인 개선 작업을 통해, 최근에 2.2 베타 버전까지 공표되었다. 자바 카드 2.2 에서는 이전에는 미흡하거나 제외되었던 내용을 대폭 추가함으로써 선구적인 개방형 스마트 카드 운영 체제로 자리매김되고 있다. 추가된 내용을 요약하면 다음과 같다.[2]

- 애플릿 및 API 삭제 기능
- 객체 삭제 기능(제한된 쓰레기 수집)
- 자바 카드 원격 메소드 호출
- 카드 관리 API
- 근접 비접촉 카드 지원

자바 카드 원격 메소드 호출은 일반 자바 플랫폼에서 제공되던 메커니즘을 자바 카드에 맞게 변환한 것이다. 이는 자바 카드의 발전 추세가 스마트 카드에의 최적화된 자바 기술 탑재와 더불어 상위 플랫폼 기술과의 융합되는 형태를 보이는 것이다.

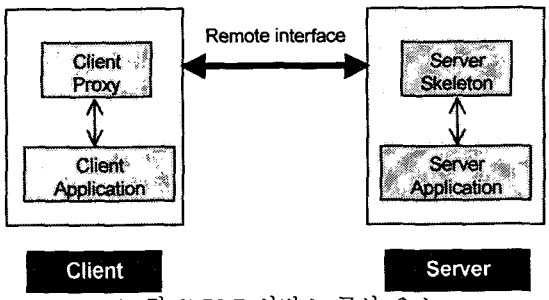


(그림 1) 자바 카드의 구조

본 논문에서는 자바 카드 2.2 에서 도입하고 있는 자바 카드 원격 메소드 호출(Java Card Remote Method Invocation, JCRMI; 이하 JCRMI 로 지칭함)에 대해 전반적으로 살펴보고, 호출 인터페이스 구현 과정과 이용에 대해 언급하고자 한다.

2. JCRMI 개요

자바 원격 메소드 호출(Java RMI; 이하 RMI 로 지칭함)은 분산된 객체들이 메소드 호출에 의해 통신할 수 있는 메커니즘을 제공한다. 원격 메소드 호출은 서버의 원격 객체의 메소드를, 클라이언트의 자바 가상 기계 위에 존재하는 객체에서 호출하는 동작을 말한다.[3] 자바에서의 RMI 서비스 구성은 (그림 2)에서 보는 바와 같이, 클라이언트 응용 프로그램과 클라이언트 proxy, 서버 skeleton 과 서버 응용 프로그램, 그리고 원격 호출 인터페이스 들로 이루어진다. proxy 는 클라이언트 측의 대리자로서, 원격 메소드 호출을 원격 객체 호출용 인터페이스를 이용해서 기동하고 결과를 받아서 전달한다. 서버의 skeleton 은 서버의 원격 메소드 호출 수행의 대리자로서, 인자를 넘겨 받아서 원격 메소드를 호출하고 결과를 반환하는 역할을 한다.[4] RMI 에서는 원격 메소드 호출을 위한 기반 API 로서 java.rmi.Remote 인터페이스와 예외 처리를 위한 java.rmi.RemoteException 클래스를 제공하고 있다. JCRMI 는 이러한 자바 RMI 의 부분 집합 API 들을 이용하고 카드측의 라이브러리를 추가하여 인터페이스를 제공한다.



(그림 2) RMI 서비스 구성 요소

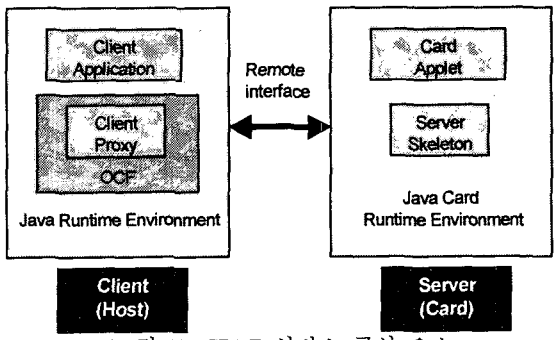
JCRMI 는, 카드리더가 포함된 호스트에서 수행되는 클라이언트 응용 프로그램이 카드상의 원격 객체의 메소드를 호출하는 메커니즘을 제공한다.[5]

JCRMI 는 자바 카드에서 구현되므로 자바 카드 언어의 제한에 귀속되어 원격 메소드 호출을 지원하게 된다. 카드 통신시 신속성과 안정성을 위해 원격 메소드 호출 인자와 결과값에 있어 직렬화(serialization)을 지원하지 않으며, boolean, byte, short, int 및 이들의 배열 등으로 제한하여 지원하고 있다. 또한, 서버의 자바 카드 가상 기계에 대응하는 클라이언트의 자바 가상 기계는 호스트에 존재하는 일반 자바 가상 기계를 이용한다. 즉, proxy 와 클라이언트 응용 프로그램을 수행하기 위한 환경은 호스트 상에 존재하는 자바 가상 기계를 이용한다. 이때, proxy 는 카드 리더와의 통신을 수행해야 하므로, 스마트 카드와 호스트간의 통신을 가능하게 하는 자바로 된 라이브러리인 opencard framework(OCF)를 사용하고 있다. [7]

JCRMI 서비스의 구성 요소는 다음과 같다.

- 카드측(서버)
  (1) JCRMI 기반 API
  (2) JCRMI 구현을 위한 인터페이스
  (3) 위 인터페이스를 구현하는 카드 애플릿
  (4) 위 클래스를 기반으로하는 skeleton
- 호스트측(클라이언트)
  (1) skeleton 을 기반으로 하는 proxy
  (2) proxy 를 기반으로 하는 클라이언트 응용 프로그램

JCRMI 에서 원격 메소드는 바로 원격 객체의 메소드를 지칭한다. 원격 객체는 클라이언트로부터 그 메소드가 호출되는 객체를 지칭하며, 원격 객체는 하나 또는 그 이상의 원격 인터페이스에 의해 기술된다. 원격 인터페이스는 JCRMI 기반 API 인 java.rmi.Remote 인터페이스를 직간접으로 계승하는 인터페이스로, JCRMI 구현을 위한 인터페이스이다.[5] skeleton 은 이 원격 인터페이스를 구현하는 카드 애플릿으로부터 생성되며, 카드 애플릿에 대해 카드상의 원격 메소드 호출을 수행하는 역할을 한다. 즉, 호스트 측의 proxy 와의 통신 프로토콜인 APDU(Application Protocol Data Unit) 내용을 분석하여 원격 메소드의 호출을 야기한다. 호스트 측의 proxy 는 클라이언트 응용 프로그램에 의한 원격 메소드 호출을 카드와 카드 리더와의 프로토콜인 APDU 로 변환하고 카드측에 전달하며 결과를 넘겨 받는 역할을 한다. (그림 3)은 JCRMI 서비스의 구성 요소들을 도식화한 것이다.



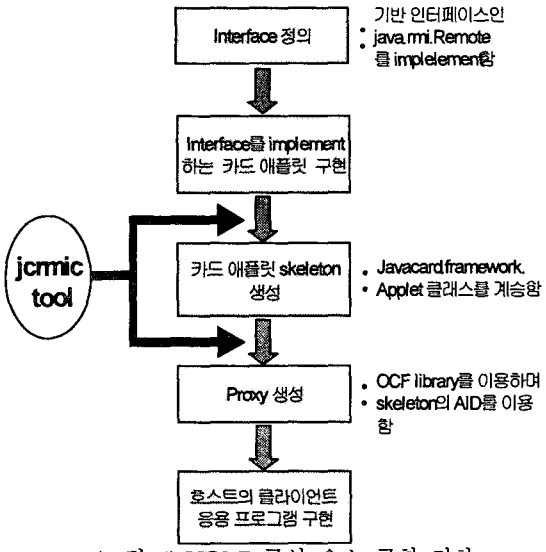
(그림 3) JCRMI 서비스 구성 요소

3. JCRMI 구현 및 수행

3.1 JCRMI 구현

JCRMI 구현을 위해서 가장 중요한 내용은 원격 메소드 호출 인터페이스의 두 축을 이루는 proxy 와 skeleton 을 생성하고, 이를 호스트 및 카드상의 응용 프로그램과 연결하는 것이다. RMI 구현을 위해 SUN 에서는 proxy 와 skeleton 을 자동적으로 생성하는 도구인 rmic 를 제공하고 있다. JCRMI 에서도 이와 비슷하

게 jcrmic 를 이용하여 호스트상의 proxy 와 카드상의 카드 애플릿 skeleton 을 생성할 수 있다. (그림 4)는 JCRMI 서비스 이용을 위한 구성 요소 구현 절차를 도식으로 보인 것이다.



(그림 4) JCRMI 구성 요소 구현 절차

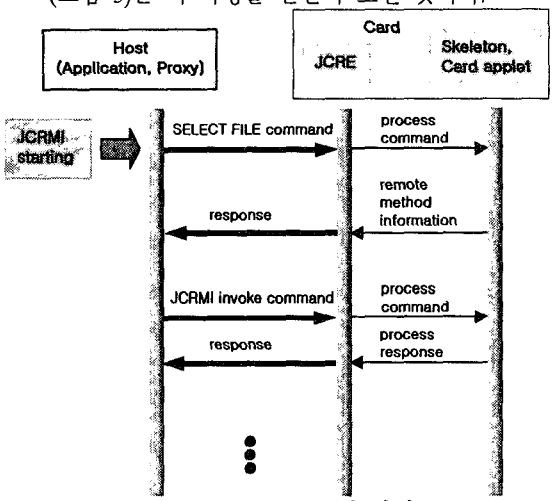
JCRMI 구현을 위한 첫단계는 java.rmi.Remote 을 계승하는 인터페이스를 정의하는 것에서 출발한다. 이 인터페이스로부터 원격 메소드를 소유하는 카드 애플릿을 구현한다. 그리고, jcrmic 를 사용하여 원격 객체 호출을 APDU 프로토콜로 내재시켜 호스트측과 통신을 수행할 skeleton 을 생성한다. 이때, skeleton 은 JCRE 에 의해, 카드 수행초기에 객체가 생성되고 선택되어 수행될 수 있도록 javacard.framework.Applet 클래스를 계승해야 한다. 즉, 카드에의 entry 애플릿이 되어야 한다. 이때, 사용되는 Applet Identifer(AID)를 이용하여 OCF 라이브러리로부터 호스트상의 proxy 를 생성한다. OCF 는 호스트측의 운영체제에 관계없이 자바 가상 기계가 탑재된 플랫폼이면 어디서나 카드 리더를 통해 스마트 카드와 통신하여 응용 서비스를 수행할 수 있는 인터페이스 및 이를 구현한 라이브러리이다.[8] OCF 는 스마트카드와의 통신시에 호스트를 대리하는 proxy 를 생성하여 이용하는데, jcrmic 에서는 이 생성 라이브러리를 이용한다. 또, 생성된 proxy 는 OCF 에서 이용됨과 동시에 JCRMI 에서 이용된다.

3.2 JCRMI 의 수행

카드 수행 측면에서 본다면, JCRMI 의 수행은 카드 리더(또는 호스트)로부터 원격 메소드 호출을 내재하는 APDU 명령을 카드에서 수신하고 APDU 로부터 원격 메소드 호출 내용을 복원하여 원하는 메소드를 호출하게 된다. 또한, 메소드를 수행하고 난 뒤에 결과를 응답 APDU 에 실어 호스트측으로 보내는 수행 구조를 가진다.

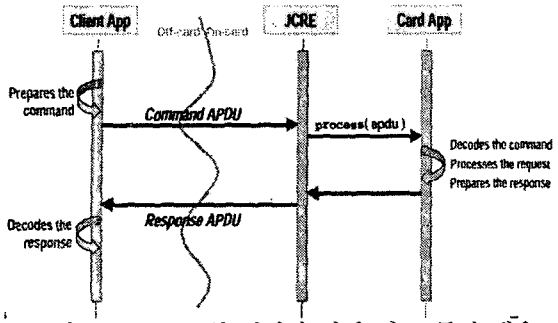
JCRMI 수행을 위한 내용을 정리하면 다음과 같다. [5]

- 원격 메소드 호출을 위해서는 카드측에서 미리 제시한 원격 객체에 대한 참조를 (호스트측으로부터) 전달받는 것이 필요하며, 전달받은 후에는 원하는 원격 메소드 호출 요구 명령을 카드에 송신해야 한다.
- 원격 객체의 참조를 얻기 위해서는 ISO 7816 표준에 정의된 SELECT FILE APDU 명령을 사용한다. 이 명령은 TLV(tag-length-value) 구조로 되어 있으므로, 원격 메소드 호출에 대한 인자 전달이 용이한 구조이다.
- 원격 메소드 호출시에 호스트에서는 원격 객체 ID, 호출될 메소드 ID, 인자값들을 알고 있어야 하며 카드내에서 정의된 형식을 취한다.
- 원격 메소드 호출은, 먼저 SELECT FILE APDU 명령에 의해 현재 선택된 애플릿의 원격 객체 및 메소드에 대한 정보를 카드에서 호스트로 송신한다.
- 호스트는 상호 정의된 원격 객체 호출용 APDU 명령(즉, 원격 객체 호출에 대한 내용을 특정 APDU 명령에 함축시킨 APDU 명령)을 카드로 보내면, 카드에서는 메소드 호출 내용을 추출하여 원격 메소드를 수행하고 결과를 반환하게 된다. (그림 5)는 이 과정을 간단히 보인 것이다.

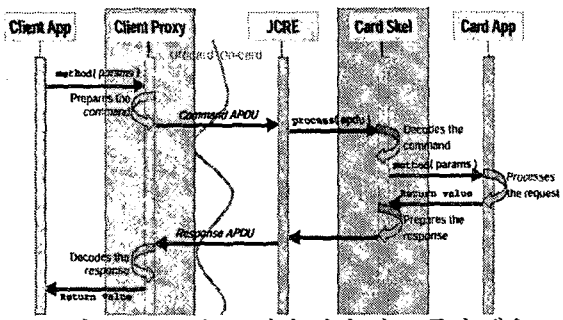


(그림 5) JCRMI 수행 과정

(그림 6)과 (그림 7)은 각각 JCRMI 도입 이전과 도입 이후에 카드와 카드 리더를 포함한 호스트의 통신 구성 요소 및 통신 프로토콜의 변화를 일반적인 형태로 보이고 있다.[6] 자바 카드 수행 환경(Java Card Runtime Environment, JCRE)은 전용 운영체제 위에 존재하는 가상 기계와 API 들을 포함하며 APDU 통신 프로토콜을 송수신하여 카드 애플릿에 전달하여 수행되도록 한다. JCRMI 구현 및 사용에 의해 원격 메소드 호출이 호스트측의 proxy 에 APDU 프로토콜에 내재되고 카드 skeleton 에 의해 다시 복원되어 원격 메소드를 호출함을 알 수 있다.



(그림 6) JCRMI 도입 이전의 자바 카드 통신 내용



(그림 7) JCRMI 를 포함한 자바 카드 통신 내용

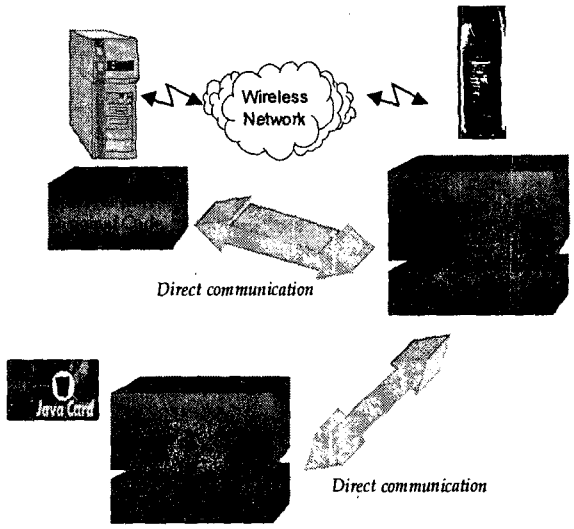
4. JCRMI 의 효과 및 이용

자바 카드에서의 원격 메소드 호출 인터페이스 제공으로 인해 얻을 수 있는 잇점은 다음과 같이 정리될 수 있다.

- 사용자 측면
  - 일반 자바 메소드를 이용하는 것처럼, 메소드 호출만으로 원하는 카드 응용 서비스를 이용할 수 있음
- 개발자 측면
  - 하위 APDU 프로토콜에 의존하지 않고 분산된 응용 프로그램을 개발할 수 있음
  - 코드의 분산으로 코드 크기를 줄일 수 있으며 프로그램 개발 및 유지가 편리함
  - 호스트측에서 적은 비용으로 손쉬운 네트워크 연결 및 사용이 가능함

이러한 잇점을 지닌 JCRMI 의 응용 중 가장 쉽고 널리 사용될 것으로 보이는 것은 휴대폰에 탑재된 자바 카드의 경우이다. 휴대폰이 호스트 역할을 하며 무선 네트워크로 서비스 서버에 연결되는 경우, 서비스 서버에서의 카드로의 사용자 인증, 뱅킹 등에 대한 원격 메소드 호출을 이용할 수 있을 것이다. (그림 8)은 자바 카드에서의 원격 메소드 호출을 이용하는 예를 그림으로 보인 것이다.

이외에도 추상화된 각 디바이스들을 네트워크로 연결하기 위한 서비스인 Jini 등에 연결하여 자바 카드 시스템(카드리더, 호스트 포함)을 하나의 네트워크용 정보 가전 형태로 사용할 수도 있다.[6]



(그림 8) JCRMI 의 이용 예 :
휴대폰과 네트워크의 연결 [6]

5. 결론

자바 카드 기술의 발전 동향 중의 하나는, 카드 H/W 성능 향상에 힘입어 상위 자바 플랫폼의 기술을 적용하는 것이다. 원격 메소드 호출의 자바 카드 적용도 이러한 맥락으로 설명될 수 있다.

자바 카드에서의 원격 메소드 호출은 스마트 카드에의 접근을 원격의 상위 프로토콜 호출로써 가능하게 하고 있다. 이로 인해 자바 카드는 카드 리더와 결합되어 추상화된 형태로 네트워크에 접속 및 이용이 가능해져, 사용자의 편의를 제공하며 코드 개발 및 유지가 용이해질 것으로 예상된다. 따라서, 향후 자바 카드 시스템은 뱅킹, 신분 확인 등의 기존의 스마트 카드 기능뿐 아니라, 네트워크 연동 기능을 탑재한 정보 가전 형태로 발전할 것으로 전망된다.

참고문헌

[1] 한국전자통신연구원, *스마트 카드 기술 시장 보고서*, 2001.12.

[2] http://java.sun.com/products/javacard/javacard22.html.

[3] M. Avvenuti, A.Vecchio, "Embedding Remote Object Mobility in Java RMI", Proceedings Eighth IEEE Workshop on Future Trends of Distributed Computing Systems(FTDCS 2001), pp.98-104. Los Alamitos, CA, USA.

[4] *Java™ Remote Method Invocation Specification*, Sun Microsystems, Inc., J2SE SDK v.1.4.

[5] *Java Card™ 2.2 Runtime Environment Specification*, Sun Microsystems, Inc., Beta Release, 2002.

[6] E. Vetillard, "Tools for Integrating the Java Card™ API into Jini™ Connection Technology", javaone conf., 2000.

[7] *Java Card 2.1 Remote Method Invocation User's Guide*, Gemplus, 2001.

[8] O. Fodor and V. Hassler, "JavaCard and OpenCard Framework: a tutorial", Proceedings of ETFA '99 IEEE, vol.1, pp.13-22, Piscataway, NJ, USA.