Journal of Korea Multimedia Society Vol. 10, No. 5, May 2007(pp. 604-611)

RAM기반 자바카드 인스톨러를 이용한 로딩속도 개선

진민식[†], 최원호[††], 이동욱[†††], 김한나[††††], 정민수[†††††], 박규석[††††††]

요 약

자바카드는 스마트카드와 (U)SIM기술의 표준 기술로 받아들여지고 있으며, 그것은 하드웨어 독립성과 이를 통해 구현 가능한 어플리케이션의 사후발행기능으로 Native카드와 구별된다. 그러나 자바카드의 가장 큰 단점 중 하나는 하드웨어 자원의 제약과 자바 언어 자체에서 야기되는 늦은 실행 속도 문제이다. 본 논문에서는 카드 터미널 또는 SMS를 통한 어플리케이션 동적 다운로드시 속도를 개선하기 위해 논리주소를 물리주소로 바꾸는 Resolution작업에서 애플릿의 다운로드시 기존의 EEPROM 기반 심볼릭 참조를 EEPROM에 비해 약 100,000배 빠른 RAM에서의 직접참조가 가능한 자바카드 인스톨러를 설계 및 구현 하였다. 실험을 통해 확인한 결과 제안된 Resolution_In_RAM 기법이 적용된 자바카드 인스톨러를 통해 애플릿을 다운로드하면 EEPROM 기록 횟수가 37%, 다운로드 시간이 30% 이상 감소됨을 알 수 있었다.

An Improvement in Loading Speed Using RAM-based Java Card Installer

Min-sik Jin[†], Won-ho Choi[††], Dong-wook Lee[†††], Han-na Kim[††††], Min-soo Jung[†††††], kyoo-seok Park[††††††]

ABSTRACT

Java Card has gained general acceptance with standard for smart card and (U)SIM technology, and it is in distinction from native card by its post-issuance of an application and independence from hardware platforms. However, a main weak point of Java Card is its low execution speed caused by the hardware limitation and Java programming language itself. In this paper, we propose a new Java Card Installer to improve the download speed during the post-issuance of an application by resolving symbolic references to physical references in RAM. Our Resolution_In_RAM is based on the improved new RAM writing is 100,000 times faster than EEPROM writing and PageBuffer that is operated as block mode, rather than cell mode is used to write to EEPROM. Consequently, the total number of EEPROM writing are reduced 37%, and the times of downloading are reduced over 30% by using the Resolution_ In_RAM-based Java Card Installer.

Key words: *Smart Card*(스마트카드), *Java Card*(자바카드), *JavaCard Installer*(자바카드 인스톨러), Resolution(레졸루션), Post-issuance(사후발행)

※ 교신저자(Corresponding Author) : 진민식, 주소 : 마산시 월영 2동 449번지(631-701), 전화 : 055)249-2217, FAX : 055)248-2554, E-mail : msjin@komsco.com
접수일 : 2006년 11월 30일, 완료일 : 2007년 3월 29일
[†] 준회원, 한국조폐공사 정보기술연구팀
[††] 준회원, 경남대학교 컴퓨터공학부
(E-mail : hoya9499@kyungnam.ac.kr)
[†††] 경남대학교 컴퓨터공학부
(E-mail : smalldo2@hanmail.net)
[††††] 경남대학교 정보통신공학부
(E-mail : dasher83@kyungnam.ac.kr)
[†††††] 종신회원, 경남대학교 컴퓨터공학부
(E-mail : msjung@kyungnam.ac.kr)
[††††††] 종신회원, 경남대학교 컴퓨터공학부
(E-mail : kspark@kyungnam.ac.kr)
※ 본 연구는 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었음

1. 서 론

스마트카드는 일반적인 신용카드 크기와 모양이 매우 비슷하지만 카드자체에 일종의 프로세서를 내장하고 있어서 일정 수준의 정보 저장과 처리 능력을 가지는 하나의 작은 컴퓨터라고 할 수 있으며, 이러한 스마트카드의 저장 및 처리 능력 때문에 다양한 분야에서 활용되고 있다. 하지만, 기존 Native방식의 스마트카드에서는 탑재되는 어플리케이션들이 특정 하드웨어와 Card Operating System(COS)에 맞는 어셈블리어나 C언어로 작성되었다. 이것은 스마트카드 어플리케이션 개발에 많은 시간과 비용이 필요하도록 만들었으며, 일단 카드가 발급된 이후에는 스마트카드 어플리케이션을 업그레이드 하거나 추가하기가 쉽지 않았다. 이러한 폐쇄적인 스마트카드의 문제점을 극복하기 위해 하드웨어에 의존하지 않은 개방형 자바카드가 등장하게 되었다[1-3].

현재의 자바카드 기술은 스마트카드 영역을 넘어서 (U)SIM 및 Ubiquitous환경의 표준 플랫폼으로 자리 매김하고 있다[4,5]. 무선 환경에서 모바일 폰 사용자는 SIM 카드로 불리는 손톱만한 크기의 자바 기술이 탑재된 스마트칩을 사용한다. 이 모바일 폰 안의 SIM 칩에는 사용자 및 모바일 폰 인증을 위한 인증 데이터 및 Key값을 저장할 뿐만 아니라 모바일 뱅킹 및 티켓팅과 같은 거래 서비스를 지원하기 위한 어플리케이션 까지도 탑재된다. 이러한 자바카드 기술이 탑재된 SIM카드는 이미 전 세계의 모바일 폰 사용자들에게 혁명적인 서비스를 제공하고 있다[2,3].

자바카드는 개방형 운영체제와 플랫폼 독립성과 같은 여러 장점들을 가지고 있지만, 제약적인 하드웨어 환경으로 인해 느린 실행 속도 문제를 가지고 있다. 그리고 어셈블리와 같은 프로그래밍 언어에 비해 자바언어 자체가 가지고 있는 인터프리터 방식의 언어에서 야기되는 속도 문제가 자바카드를 더욱 느리게 만드는 주원인이 되고 있다[1,2,6].

본 논문에서는 이러한 하드웨어와 자바카드 언어 측면에서 야기되는 속도 문제를 극복하기 위해 애플릿의 다운로드 동안 심볼릭(간접)참조를 EEPROM이 아닌 기능 및 용량 면에서 개선된 새로운 RAM환경을 기반으로 직접(물리)참조로 바꾸는 Resolution_In_RAM 기법 기반 자바카드 인스톨러(Installer)를 설계/구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로써 자바카드 시스템의 개요와 자바카드 인스톨러에 대하여 알아본 후, 3장에서는 본 논문에서 제시하는 Resolution_In_RAM 기법을 기술하며, 4장에서는 CalmCore16bits MPU가 장착된 자바카드 환경에서 Resolution_In_RAM 기법이 적용된 자바카드 인스톨러의 성능평가 및 결과를 기술한다. 마지막으로 5장에서는 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

2.1 자바카드 시스템의 개요

자바카드는 자바 기술을 IC 카드에 대해 최적화하여 구현하고 있다. 자바카드는 일반적인 IC 카드에 적용되는 모든 표준을 따르는 전형적인 IC 카드인데 하위의 운영체제 위에 존재하는 자바카드 가상 기계가 자바카드 애플릿의 바이트 코드(bytecode)를 수행하고 메모리, I/O 같은 카드 내의 모든 자원에 대한 접근을 제어한다는 점에서 차이가 난다[1,7].

자바카드 기술은 플랫폼간에 이진 코드의 이식성 즉, 상호 운용성이 뛰어나고 타입 검사 등에 의해 악의적 코드에 대한 보안성을 지닌 자바 언어를 IC 카드의 실시간 환경에 대해 최적화하고 있다. 자바카드에서의 가상 기계 이용에 의한 장점은 응용 프로그램과 운영체제를 분리하는 개방형 운영체제를 가능하게 한다는 점과, 하드웨어 의존적인 어셈블리코드가 아닌 상위 언어인 자바 언어로 쉽게 응용 프로그램을 작성 및 수행할 수 있다는 점이다. 아울러 카드가 최종 사용자에게 발급된 이후에도 서비스 변경에 따른 응용 프로그램을 IC 카드에 적재할 수 있다는 점도 장점이라 할 수 있다. 결국 다양한 다수의 응용 프로그램을 수용할 수 있는 유연성을 가지게 한다[1].

자바카드는 그림 1에서 보는 것처럼, 자바카드 API와 가상기계(Virtual Machine)로 이루어진 자바카드 수행 환경(Java Card Runtime Environment: JCRE), Chip Operating System(COS), H/W 및 응용 프로그램인 애플릿으로 구성된다[1,7]. 그림 2는 분리된 가상 기계 구조를 가진 자바카드에서, 카드 밖에서 애플릿을 컴파일하고 변환한 뒤, 카드리더를 통해 카드로 적재하여 수행하는 과정을 보인 것이다[1,8].

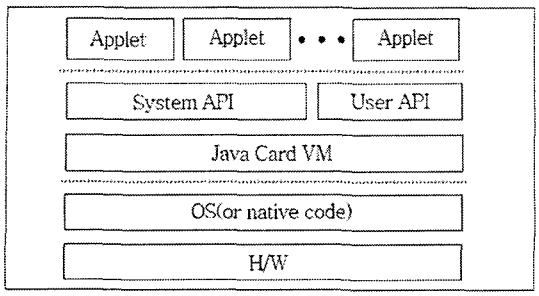


그림 1. 자바카드 구조

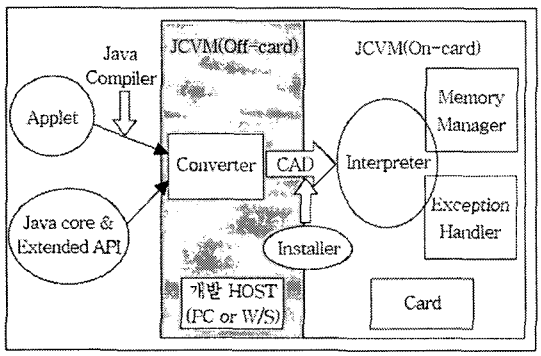


그림 2. 자바카드 수행 및 적재 과정

2.2 자바카드 인스톨러의 구성

자바카드 인스톨러는 JVM 에서 ClassLoader 의 기능을 제한적으로 수행한다. 즉, 컴파일 후 자바카드 용으로 변환된 애플릿을 자바카드 용 내부 메모리 구조에 로드하고 링크작업을 수행 한다[1,9,10]. 그림 3은 Card Reader 측에서의 post-issuance APDU 명령을 중심으로 하여, 인스톨러의 수행 절차를 보이고 있다.

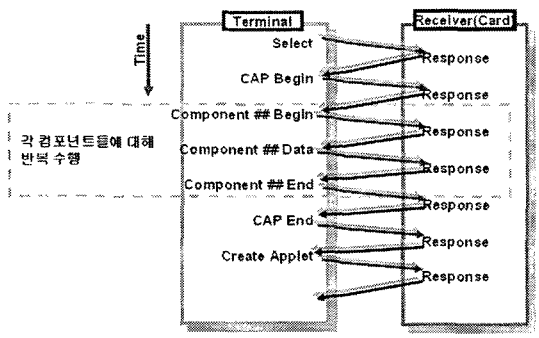


그림 3. 자바카드 인스톨러 애플릿이 새로운 CAP파일을 다운로드 하는 과정

2.3 자바카드 시스템의 Resolution과 링크과정

자바 시스템에서는 간접 참조를 직접 참조로 바꾸는 Resolution과 링크의 과정을 자바 프로그램 실행 중에 수행 하지만, 자바카드 시스템에서는 이 과정이 CAP파일 다운로딩 동안 인스톨러에 의해 수행한다.

Method컴포넌트의 모든 바이트 코드들은 Off-Card의 Converter를 통해 생성된다. Off-Card에서는 애플릿이 호출하는 실제 카드에 있는 API의 메소드들에 대한 물리 주소를 알 수 없다. Converter가 바이트 코드를 변환하는 동안 자바카드 API에 있는 메소드 호출의 경우는 토큰 형태의 간접 참조로 변경된다. 이것은 자바 시스템에서는 Unicode형태의 String을 사용하는 것과는 달리 자바카드는 String에 대한 제약사항 때문에 간접참조를 토큰형태로 사용하기 때문이다.

Method컴포넌트 이후에 다운로드 되는 ConstantPool컴포넌트는 CAP파일에 사용되는 모든 Constant들에 대한 토큰형태로 되어 있는 간접주소를 가지고 있다. 자바의 ConstantPool은 스트링을 사용하여 Constant 항목을 표현하지만 자바카드에서는 토큰 방식의 Constant 항목들을 표시한다.

자바 시스템과는 달리 자바카드 시스템에서는 스트링을 지원하지 않으며, ClassLoader역시 존재하지 않는다. 그래서 자바카드에서는 스트링이 아닌 토큰기반의 심볼릭 참조를 사용한다. 여러 개의 컴포넌트로 되어 있는 CAP파일 중 ConstantPool컴포넌트는 토큰으로 되어 있는 Constant 엔트리들을 가지고 있다.

3. RAM기반 자바카드 인스톨러

8, 16비트용 스마트카드에 내장되는 RAM은 1~2K정도의 용량을 가지고 있다. 충분하지 않은 RAM 용량으로 자바카드 어플리케이션을 수행하기 위해서 대부분의 기능과 필요한 데이터들을 EEPROM기반으로 수행하였다. 하지만, 최근 하드웨어 기술의 발달로 인해 최근의 스마트카드용 32비트 MPU는 6~10K 크기의 RAM용량을 가지고 있다. 본 논문에서는 EEPROM기반으로 수행되는 애플릿의 로딩을 RAM 용량의 증가를 통한 RAM기반 고속의 자바카드 인스톨러를 개발하였다.

3.1 기존 자바카드 인스톨러

자바카드 시스템은 MPU와 메모리 같은 하드웨어에 대하여 다른 컴퓨터 시스템과 달리 많은 제약사항을 가지고 있다. 그리고 Card Acceptance Device (CAD)로부터 애플릿을 다운로드 받는 경우나 무선을 통해 SMS로 애플릿을 동적으로 다운로드 받는 자바카드기반 SIM카드의 경우 다운로드 속도 및 시간이 가장 중요한 요소가 된다. 그러나 기존의 자바카드 인스톨러를 분석한 결과 CAD로 부터 동적인 애플릿 다운로드 시 많은 EEPROM 데이터 기록 과정을 수행하였다. 앞에서 언급한 것 처럼, 페이지버퍼를 이용한 EEPROM 기록 과정은 애플릿 로딩 속도를 더욱 느리게 만드는 주요 원인이 된다.

CAD로 부터 전달되는 12개의 컴포넌트 중 Class, Method, Export, ConstantPool컴포넌트는 컴포넌트의 모든 내용을 EEPROM의 힙 영역에 그대로 적재한 후 저장한다. 이것은 위 4개의 컴포넌트만을 이용하여 하나의 애플릿 인스턴스를 생성할 수 있음을 의미한다. 그 외의 나머지 컴포넌트들은 인스톨러로 전달된 후 EEPROM에 적재되지 않고 단지 인스톨러에 의해 파싱된 후 필요한 정보를 설정한 후 소멸된다.

Resolution과 링크 과정은 EEPROM에 저장된 4개의 컴포넌트 데이터에서 논리주소로 되어 있는 심블릭참조를 타겟플랫폼에 맞는 물리주소로 변경하는 과정이다. 일반적으로 토큰기반으로 되어 있는 논리주소는 애플릿 로딩 시 Resolution과 링킹 과정을 수행한 후 1~2바이트의 물리주소로 변경된다. 즉, Resolution은 이미 EEPROM에 로딩 된 토큰기반의 Constant 엔트리들을 물리주소로 바꾸는 과정이다. 또, 링크 과정은 ReferenceLocation의 크기만큼 이미 EEPROM에 로딩 되어 있는 Method컴포넌트 바이트코드를 1바이트 또는 2바이트씩 계속적으로 변경하는 과정이다. 이러한 이유로써, CAP파일 다운로딩 동안 가장 많은 시간을 소비하는 컴포넌트들은 ConstantPool과 ReferenceLocation컴포넌트이다.

그림 4는 CAP파일 다운로드 시 인스톨러가 Resolution과 링크 과정에 관련된 3개의 컴포넌트에 대하여 수행 하는 과정을 RAM과 EEPROM의 관점에서 도식화하고 있다.

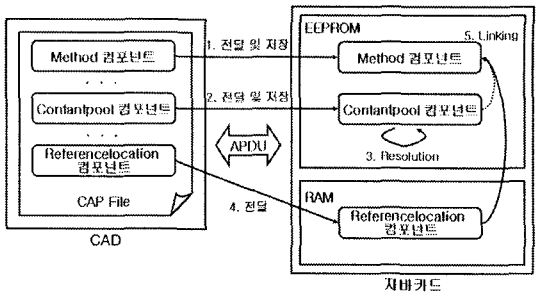


그림 4. 컴포넌트의 Resolution과 링크 과정

3.2 RAM기반 자바카드 인스톨러

자바카드와 같은 개방형 카드 플랫폼에서는 유/무선을 통해 동적으로 자바카드와 SIM카드에 프로그램을 설치하는 어플리케이션 사후 발행 기능이 지원된다. 만약 자바카드 및 SIM카드 사용자가 어플리케이션을 다운로드 받기 위해 많은 시간을 소비한다면 다운로드 중에 설치를 중단해 버리는 일이 많이 발생할 것이다.

본 논문에서는 어플리케이션의 다운로드 및 설치를 담당하는 자바카드 인스톨러를 분석한 후 EEPROM에서 수행되는 Resolution과 링크 과정을 RAM에서 수행하는 Resolution_In_RAM기법을 개발하였다. 이것은 페이지 버퍼를 이용한 EEPROM의 쓰기 방식과 EEPROM의 쓰기 속도가 RAM에 비해 100,000배 이상 느리다는 특징을 바탕으로 하고 있다.

CAP파일의 적재 시 가장 많은 시간이 소비되는 ReferenceLocation컴포넌트는 그 크기가 200Bytes면 EEPROM에 기록을 200번 수행한다. 즉, EEPROM의 기록횟수는 ReferenceLocation컴포넌트 바이트 수에 비례한다. 하지만 Resolution_In_RAM 기법이 적용된 Installer의 기록횟수는 Method 컴포넌트의 크기에 비례하므로 EEPROM의 기록 횟수를 줄일 수 있다.

그림 5는 본 논문에서 제시한 Resolution_In_RAM기법을 RAM과 EEPROM의 관점에서 보이고 있다. 우선 기존의 CAP파일 다운로딩과 마찬가지로 Method컴포넌트는 그대로 전달된 후 EEPROM에 저장되고, ConstantPool컴포넌트는 EEPROM이 아니라 RAM으로 전달된 후 저장된다. 이 컴포넌트는 토큰기반의 Constant 엔트리들을 가지고 있으므로 이것들을 RAM에서 물리주소로 바꾸는 Resolution을 수행한다. 마지막으로 ReferenceLocation컴포넌트가 RAM으로 전달되면 Method컴포넌트로부터

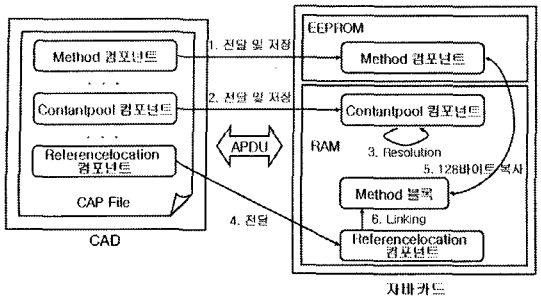


그림 5. 자바카드 인스톨러의 Resolution In RAM 기법

128바이트를 RAM으로 복사한다. 여기서 128바이트는 페이지 버퍼의 크기이다.

12개의 컴포넌트 중 마지막 컴포넌트인 ReferenceLocation컴포넌트는 Method컴포넌트의 바이트코드들 중 링크 과정이 필요한 오프셋을 가지고 있다. 자바카드 인스톨러는 전달된 ReferenceLocation컴포넌트의 오프셋들을 가지고 RAM에 복사된 Method컴포넌트의 128바이트에서 오프셋의 위치를 계산한 후 링크 작업을 수행한다. 만약, 오프셋이 128바이트를 넘어서면 RAM에 있는 링킹 작업을 마친 128바이트를 EEPROM의 Method컴포넌트의 원래 위치로 되돌린 후 다음 128바이트 블록을 RAM으로 복사한다. 위와 같은 과정을 ReferenceLocation컴포넌트의 끝까지 수행한다. Resolution과 링크 과정 모두를 RAM기반에서 수행하게 된다.

아래는 타겟 플랫폼의 페이지 크기에 따라 RAM 영역에서 Resolution과 링킹을 수행하기 위한 절차를 보이고 있다. 본 논문에 적용된 S전자의 타겟 플랫폼의 페이지 크기는 128바이트이므로 RAM영역에 128바이트의 메모리를 동적으로 할당한 후 Resolution과 링킹 작업을 수행하였다.

① Method 컴포넌트를 EEPROM 영역에 저장
② ConstantPool 컴포넌트를 RAM 영역에 저장
③ ConstantPool의 상수들에 대한 Resolution 수행
④ ReferenceLocation컴포넌트를 RAM 영역에 저장
⑤ Method컴포넌트의 128바이트를 RAM 영역으로 복사
⑥ ReferenceLocation컴포넌트 데이터로 부터 링킹이 필요한 오프셋을 구함
⑦ 오프셋 값에 따라 ⑤ 데이터에 대해 링킹 작업을 수행
⑧ 128바이트를 넘으면 새로운 128바이트를 복사 후 Method컴포넌트의 끝까지 링킹 작업 수행

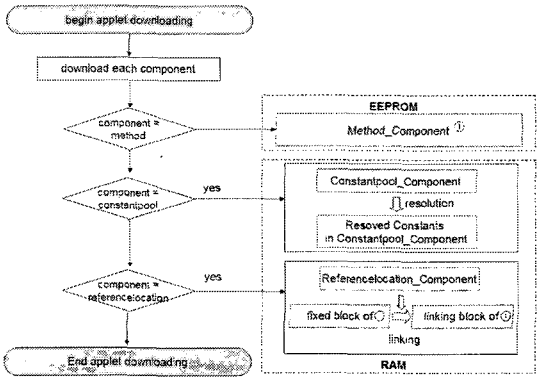


그림 6. Resolution In RAM 알고리즘

그림 7은 Sun에서 제공하는 "HelloWorld" 애플릿 샘플을 이용하여 Resolution_In_RAM 기법이 적용된 Installer가 Method컴포넌트, Constantpool컴포넌트, ReferenceLocation컴포넌트에 대해 어떻게 Resolution과 링크 과정을 수행하는지 보이고 있다.

먼저 Method컴포넌트는 EEPROM에 다운로드되고, Constantpool컴포넌트와 ReferenceLocation 컴포넌트는 RAM에 다운로드 된다. Constantpool컴포넌트가 물리주소로 변환하는 Resolution과정을 거치고, EEPROM에 저장된 Method컴포넌트에서 128바이트씩 RAM의 Method Block로 복사한다. ReferenceLocation컴포넌트가 가지는 오프셋 값으로 Method Block에 복사된 값의 링크과정을 수행한다. ReferenceLocation컴포넌트의 "01 04 06"은 복사된 Method Block에서 1번째, 5번째, 11번째 위치에서 링크 과정이 필요함을 의미한다. 따라서 Method Block의 1번째와 5번째, 11번째 위치인 "00 00, 00 01, 00 02"는 각각 "80 38, 80 3A, 80 86"으로 변경되는

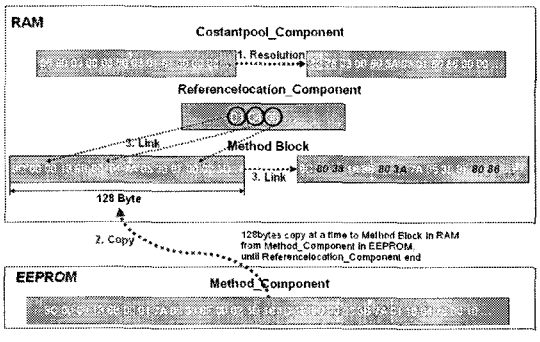


그림 7. Resolution_In_RAM 기법의 Resolution과 링크 과정

데, 이 과정을 ReferenceLocation컴포넌트 끝까지 수행하면 Resolution과정과 링크작업이 완료된다.

4. 실 험

본 논문에서 제시한 Resolution_In_RAM기법에 대한 정확한 측정을 위해 S전자에서 생산하는 스마트카드용 칩인 CalmCore16bits MPU를 이용하여 실험하였다. 이 칩은 160K ROM, 6K RAM, 64K EEPROM의 메모리를 가지고 있다[11,12].

본 실험에서 16개의 자바카드용 애플릿들을 실험하였다. 일반적으로 각 CAP파일들의 Method컴포넌트 크기는 애플릿 사이즈의 50%이상을 차지한다. 각 실험을 통하여 우리는 각각의 CAP파일이 다운로드되는 동안의 EEPROM 쓰기 횟수와 실행 속도에 대하여 알아보았다.

표 1에 있는 수치는 카드를 CAD에 넣은 후 카드를 초기화하고 각각의 애플릿을 다운로드 한 후 설치할 때까지의 EEPROM 쓰기 횟수이다. 실험 결과로 Resolution_In_RAM기법이 탑재된 자바카드 인스톨러를 사용할 경우 평균 37%이상의 EEPROM 기록 횟수를 줄일 수 있음을 알 수 있다.

또한, 표 2는 각각의 CAP파일이 CAD를 통해 자바카드 인스톨러에 의해서 카드에 다운로딩되고 설치되는 동안의 시간을 나타내고 있다. RAM기반의 자바카드 인스톨러가 탑재된 자바카드에서는 평균 30% 이상 다운로드 시간이 단축됨을 알 수 있다.

표 1. 애플릿 다운로드 동안 EEPROM 쓰기 횟수의 비교

애플릿	애플릿 크기	EEPROM 쓰기 횟수		Reduced rate (%)
		JavaCard 2.2.1	Our Approach	
Channel Demo	5k	7,552	4,788	36.59
JavaLoyalty	3k	7,291	3,966	45.60
JavaPurse	7k	22,712	11,898	47.61
ObjDelDemo	10k	16,416	9,075	44.71
PackageA	5k	9,685	6,000	38.04
PackageB	4k	7,698	4,218	45.20
PackageC	3k	3,497	2,235	36.08
Photocard	4k	6,737	4,227	37.25
RMIDemo	4k	6,119	3,783	38.17
Wallet	4k	5,641	3,570	36.71
EMV small Applet	5k	6,721	4,257	36.66
EMV Large Applet	17k	11,461	7,299	36.31
Sample-Library	3k	6,326	4,833	23.60
Seed	11k	18,140	12,303	32.17
Crypto	15k	19,435	12,702	34.64
Testhash	13k	19,243	12,342	35.86
평균				**37.06**

5. 결 론

본 논문에서는 애플릿의 다운로드 및 실행 속도를 높이기 위해 심볼릭(간접)참조를 EEPROM이 아닌 RAM에서 물리(직접)참조로 바꾸는 Resolution_In_RAM 기술을 가지는 자바카드 인스톨러를 설계/구현하였다. 자바카드와 SIM카드에서 사용되는 애플릿의 사후발행 기능의 성능 요인은 빠른 다운로드 속도가 가장 중요한 요소이다. 그러나 카드 하드웨

표 2. 애플릿 다운로드 및 설치 동안의 속도 비교

애플릿	애플릿 크기	다운로드 및 설치 속도(단위:초)		Reduced rate (%)
		JavaCard 2.2.1	Our Approach	
Channel Demo	5k	76	51	32.8
JavaLoyalty	3k	72	50	30.5
JavaPurse	7k	232	170	26.7
ObjDelDemo	10k	199	160	19.5
PackageA	5k	90	56	37.7
PackageB	4k	74	51	31.0
PackageC	3k	32	21	34.3
Photocard	4k	64	41	35.9
RMIDemo	4k	57	36	36.8
Wallet	4k	63	50	21.0
EMV small Applet	5k	61	39	36.0
EMV Large Applet	17k	119	79	33.6
SampleLibrary	3k	69	54	21.7
Seed	11k	247	163	34.0
Crypto	15k	280	207	26.0
Testhash	13k	256	183	28.5
평균				**30.38**

어의 제약사항과 다른 프로그래밍 언어에 비해 자바 언어의 느린 실행속도에서 야기되는 문제점으로 자바카드 개발 회사에서는 자바카드의 고유한 특성인 플랫폼 독립성을 위반해 가며 자바카드 인스톨러를 자바언어가 아닌 Native모듈로 작성하고 있는 실정이다.

본 논문에서는 자바카드용 애플릿으로 동작하는 자바카드 인스톨러를 분석 후 빠른 다운로딩과 실행 속도를 가지는 인스톨러를 개발하였다. 이것은 RAM의 쓰기 속도가 EEPROM에 비해 100,000배 이상 빠르다는 것과, EEPROM은 페이지 버퍼를 이용한 블록 단위의 쓰기를 수행 하며, 일반적인 스마트 카드 시스템은 트랜잭션을 처리하기 위해 EEPROM의 값이 변경 될 때 마다 트랜잭션 버퍼에 계속적으로 이전 데이터를 저장하는 것에 기반한다. 따라서, 본 논문에서 제시한 Resolution_In_RAM기법 기반 자바카드 인스톨러를 이용하여 EEPROM의 쓰기 횟수와 애플릿의 다운로드 시간이 30% 이상 감소됨을 확인할 수 있다.

참 고 문 헌

[1] Zhiqun Chan, *Java Card Technology for Smart Cards: Architecture and programmer's guide.* Addison Wesley, Reading, Massachusetts, 2001.

[2] Wolfgang Rankl and Wolfgang Effing, *Smart Card Handbook Third Edition*, John Wiley & Sons, 2001.

[3] Min Sik Jin, Won Ho Choi, Yoon-Sim Yang, and Min Soo Jung, "A Study on fast JCVM with new transaction mechanism and Caching-Buffer based on Java Objects with a high locality," *UISW Workshops*, LNCS 3823, pp. 91-100, 2005.

[4] The 3rd Generation Partnership Project, *Technical Specification Group Terminals Security Mechanisms for the (U)SIM application toolkit*, 3GPP, 2002.

[5] SIMAlliance, http://www.simalliance.org.

[6] 김영진, 전용성, 전성익, 정교일, "Java Card Platform을 내장한 Smart Card의 구현," 정보과학회지, ISSN 1229-6821, pp. 33-45, 2001.

[7] Sun Microsystems, Inc., *JavaCard 2.2.1 Virtual Machine Specification,* Sun Microsystems, Inc., http://java.sun.com/products/javacard, 2003.

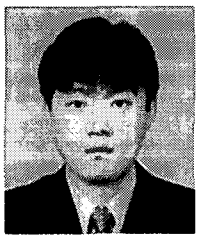
[8] Sun Microsystems, Inc., *JavaCard 2.2.1 Runtime Environment Specification,* Sun Microsystems, Inc., http://java.sun.com/products/javacard, 2003.

[9] Xavier Leroy, "Bytecode verification for Java smart card," *Software Practice & Experience*, pp. 319-340, 2002.

[10] Xavier Leroy, "On-Card Bytecode Verification for Java Card," *E-smart 2001*, LNCS 2140, pp. 150-164, 2001.

[11] MCULAND, http://mculand.com/e/sub1/s1-main.htm.

[12] SAMSUNG, http://www.samsung.com/Products/Semiconductor

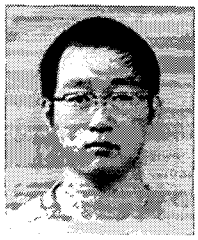


진 민 식

2000년 경남대학교 컴퓨터공학과 학사
2002년 경남대학교 컴퓨터공학과 석사
2006년 경남대학교 컴퓨터공학과 박사
2007.3~현재 한국조폐공사 정보기술연구팀
관심분야 : Ubiquitous, Smart Card, Java Card

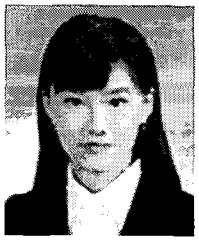
최 원 호

1999년 경남대학교 전산통계학과 학사
2001년 경남대학교 컴퓨터공학과 석사
2005년 경남대학교 컴퓨터공학과 박사
관심분야 : Embedded, Java Card, JavaMachine



이 동 욱

2006년 2월 경남대학교 컴퓨터공학부 학사
2006년 3월~현재 경남대학교 컴퓨터공학과 석사과정
관심분야 : Java Technology, Java Card, Home-Network



김 한 나

2006년 2월 경남대학교 정보통신공학부 학사
2006년 3월~현재 경남대학교 정보통신공학과 석사과정
관심분야 : Smart Card, Data Communication, 초고속 정보통신

정 민 수

1986년 서울대학교 컴퓨터공학과 학사
1988년 한국과학기술원 전산학과 석사
1994년 학국과학기술원 전산학과 박사
1990년~현재 경남대학교 컴퓨터공학부 교수
관심분야 : Java Technology, JavaMachine, Home-Networking

박 규 석

1980년 중앙대학교 대학원 전자계산학과 석사
1988년 중앙대학교 대학원 전자계산학과 박사
1982년~현재 경남대학교 컴퓨터공학부 교수
관심분야 : 분산처리 시스템, 멀티미디어 시스템, u-지능형스페이스