한국컴퓨터정보학회 하계학술대회 논문집 제20권 제2호 (2012. 7)

스크립트 파일 기반 AFC 단말기의 카드처리 방법 설계

남현우°, 장병근*, 박영욱*
°*한국스마트카드 T-money 기술연구소
e-mail: {hw.nam, bgjang, yw.park}@koreasmartcard.com°*

A Design of Card Processing Method based on Script File for AFC Terminal

Hyunwoo Nam°, Byunggeun Jang*, Youngwook Park*
°*T-money Tech. Unit, Korea Smart Card

● 요 약 ●

AFC 시스템은 요금징수단말기로서 버스, 철도, 택시와 같은 다양한 대중교통 수단에서 사용되고 있다. 기존 AFC 단말기는 C, C++ 언어로 작성되었으며, 기능 추가 및 수정을 위해 전체 Application의 실행파일이 전부 교체되어야 했다. 본 논문의 목표는 빈번하게 수정되는 주요 기능들을 스크립트 파일로 모듈화 하는 것이다. 이를 위해 스크립트 파일 기반 시스템을 설계하고 일부 기능을 구현하여 시스템을 평가하였다.

키워드: 스크립트(Script), AFC(Automatic Fare Collection), 스마트카드(Smart Card)

I. 서 론

AFC(Automated Fare Collection) 시스템은 요금징수를 위한 시스템으로 대중교통에서 사용되는 버스단말기, 철도 게이트, 택시 단말기 등을 통해 사용자들은 접하고 있다. AFC 시스템은 지불 매체의 종류 및 카드 종류, 요금제를 비롯한 비즈니스 환경에 따라 다양한 형태로 구현될 수 있다.

일반적으로 해당 노선에서 선택한 요금제 및 비즈니스 환경, 지원 카드의 종류에 따라 전용 Application을 설계하고 개발하고 있다. 이에 따라 단말기나 운영 지역 및 운수사에 따라 각기 다른 소스코드와 실행바이너리 파일이 생성되어 운영되고 있다. 만약 요금제나 비즈니스 또는 지원 카드 타입의 수정 또는 기능이 추가될 경우 일부 기능의 변경을 위해 전체 Application의 소스를 수정하고 빌드해야만 했다. 이와 같은 구조에서는 매번 전체 Application을 대상으로 변경 작업을 수행했어야 했으며, 따라서 업데이트 때마다 전체 시스템의 재검증이 요구되었다. 또한 비즈니스 변경에 따른 유연한 시스템의 구성이 쉽지 않았다.

따라서 본 논문에서는 변경이 빈번하게 발생하거나 적용 지역 및 비즈니스 환경에 따라 달라지는 카드 처리 부분을 스크립트 파일 모듈로 대체하는 것을 목표로 한다. 만약 스크립트 파일 기반으로 단말기가 운영될 경우 기능 변경이나 추가시 Application은 건드리지 않고, 해당 기능의 스크립트 파일만을 교체하여 변경 작업을 적용할 수 있다. 따라서 업그레이드나 패치가 필요할 경우 Application의 중단 없이도 스크립트 파일의 변경만으로 기능 변경 및 추가가 가능해진다.

본 논문에서는 스크립트 파일 기반으로 카드처리를 수행하는 AFC 단말기의 S/W를 설계하였고, 시스템의 구현 가능성을 테스트하기 위하여 일부 기능을 실제 구현하여 테스트를 수행하였다. 2장에서는 관련 연구 자료를 정리하며, 3장에서는 시스템에 대한 설계를 4장에서는 제안하는 스크립트 기반 시스템의 성능을 측정하였다.

II. 관련 연구

1. 스크립트 엔진

스크립트 엔진은 지원하고자 하는 스크립트 언어에 따라 달라질 수 있다. 예를 들어 게임 프로그래밍의 경우 Lua, 루비와 같은 스크립트 언어를 사용한다. 그리고 대중적으로 웹 환경에서 많이 쓰이는 자바스크립트의 경우 기존 웹 브라우저에서 사용되며, 다양한 자바 스크립트 엔진들을 일반 Application에서도 내장하여 사용할 수가 있다.

자바 스크립트를 지원하는 주요 스크립트 엔진으로는 크게 Rhino, Spider Monkey, V8 등이 있다[1]. Rhino는 자바 언어로 작성되었으며 ECMA-262 표준을 지원하고 있다. Spider Monkey[2]는 Rhino와 마찬가지로 Mozila사에서 개발되었으며, Filefox 웹브라우저에서 사용되었다. 또한 ECMA-262 표준을 지원하지만 Rhino와 달리 C, C++로 작성된 엔진이다. V8은 구글의 크롬 웹브라우저에서 사용된 엔진으로 성능 측면에서 타 엔진보다 강점을 가지고 있다. Spider Monkey와 동일하게 C, C++언

어로 작성되었으며 현재 EABI 컴파일러만을 지원하고 있어 EABI가 지원되지 않는 구형 단말기에서는 적용하기가 어려웠다.

2. 스마트 디바이스의 앱(APP)

최근 스마트 디바이스들의 시장점유율이 많아지면서 일반 PC의 Application이 아닌 스마트폰, 스마트TV와 같은 디바이스를 위한 Applcation의 제작이 증가되고 있다. 하지만 스마트 디바이스의 제조사 및 플랫폼에 따라 개발 환경 및 방법이 많이 달라져, 동일한 Application을 다양한 디바이스에 적용하기 위해서는 지원하는 디바이스 타입 수에 비례하여 개별적으로 개발이 필요하였다.

이와 같은 문제점을 해결하기 위해 HTML5와 Javascript 언어를 사용하여 플랫폼에 독립적으로 개발을 수행할 수 있는 개발 방법들이 생겨나고 있다. 대표적으로 웹브라우저 환경에서 수행되는 웹앱 형태와 기존 웹 기술에 Native 코드가 혼용된 하이브리드 웹앱 기술이 있다. 대표적인 기술로는 폰갭[1], 앱스프레소, 타이타늄의 개발도구들이 있으며, HTML 과 Javascript 스크립트 언어만으로 작성된 하나의 소스코드만으로 다양한 스마트 디바이스들을 지원할 수 있게 되었다. 또한 웹 기술뿐만 아니라 Native 코드 또한 사용이 가능하여 스크립트 파일을 사용하여 개발 편리성을 제공하면서도, Native 코드 사용을 통한 성능상의 이점도 가져갈 수 있게 되었다.

3. RF, SAM 디바이스

RF 디바이스는 13.56Mhz 주파수 대역에서 단말기와 스마트카드 간 통신을 수행하기 위해 사용되는 H/W 모듈로, 이론상 최대 10cm 인식거리 안에서 카드 인식이 가능하다.

SAM 디바이스는 USIM과 같은 형태로 제공되는 보안 처리를 위한 H/W 모듈이며, SAM에는 고유 키 값을 가지고 있어 선불카드의 지불이나 충전 등의 기능을 수행할 때 카드의 인증 작업시 사용된다. 현재 버스 단말기에는 지원하는 선불 카드에 따라 티머니 SAM 뿐만 아니라 타사의 SAM이 같이 설치되어 있다.

III. 스크립트 기반 단말기의 설계

1. 시스템 구성도

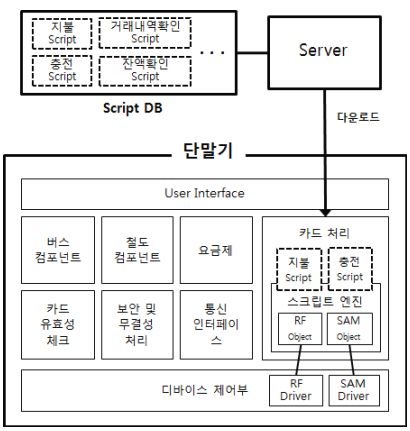

그림 1. 스크립트 파일 기반 AFC 단말기 시스템

그림1은 스크립트 파일 기반의 시스템 구성을 보여준다. 단말기에는 화면 디스플레이를 위한 UI 인터페이스와 버스, 철도에서 사용되는 주요 기능들의 컴포넌트를 지원하고 있으며 카드의 유효성 및 보안관련 기능들과 다양한 통신 방법을 지원하는 통신 인터페이스를 제공한다. 본 논문에서는 이 중 카드처리 부분을 스크립트 파일을 사용하여 모듈화 형태로 제공하는 방법을 제안하고 있다.

스크립트 파일로는 지불, 충전, 거래내역 확인, 잔액확인 등의 기존에 사용되던 다양한 카드 처리 기능들을 구현할 수 있으며, 이 스크립트 파일은 서버의 DB에 저장되어있다. 단말기는 서버로부터 필요한 기능의 스크립트 파일을 다운로드 받아 단말기에 저장할 수도 있다.

단말기에는 스크립트 파일을 처리하기 위하여 스크립트 엔진이 포함된다. 스크립트 엔진은 스크립트 파일을 해석하여 코드를 수행하고, 카드처리를 위한 RF 디바이스와 SAM 디바이스 오브젝트를 제공한다. 각 오브젝트는 실제 하드웨어를 제어하는 디바이스 드라이버와 연결되어 스크립트 파일에서 하드웨어 사용을 가능하게 해준다. 이외에도 스크립트 엔진은 환경파일을 읽거나 로그를 기록하기 위하여 파일 오브젝트와 에러 로그 오브젝트를 제공한다.

2. 카드 처리 스크립트 파일의 수행 과정

그림2는 스크립트 파일 기반 시스템의 카드 처리 과정만을 도식화한 그림이다. 스크립트 파일은 Application의 초기화시 로드되거나 필요에 따라 운영 중에도 로드될 수 있다.

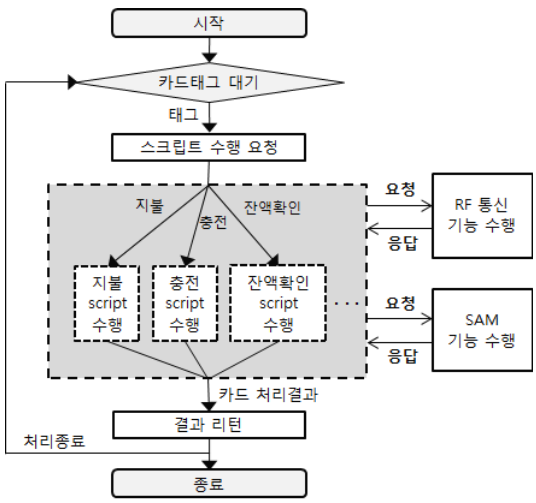

그림 2. 카드처리 스크립트 파일의 수행 과정

RF 오브젝트는 카드의 대기 상태로 운영 중이다가 카드 인식거리 이내에 카드가 존재하면 카드의 충돌 검사 및 초기화 과정을 수행한다. 초기화 과정이 완료되면 해당 스크립트 파일로 분기하여 카드처리를 수행한다. 만약 지불 기능을 수행한다고 한다면 지불 스크립트 파일로 분기하고 이에 대한 처리 결과를 리턴한 후 스크립트 수행을 종료한다.

그리고 스크립트 수행시 실제 카드의 처리를 위하여 RF와

SAM 디바이스를 제어하기 위한 RF 오브젝트와 SAM 오브젝트를 사용한다.

표1의 코드는 스크립트 엔진 기반에서 교통 카드의 잔액을 확인하기 위하여 테스트로 작성한 스크립트 파일이다.

스크립트 파일에서 RF, SAM 오브젝트에서 제공하는 자바스크립트 API인 lapdu() 함수를 사용하여 APDU 명령을 전달하며, 처리 결과를 리턴 받는다. lapdu() 함수를 호출하면 RF 오브젝트를 통해 실제RF 하드웨어에 APDU 명령 요청을 전달하여 카드와 통신이 이뤄진다. 리턴 받은 값은 잔액 확인 요청에 대한 잔액 값으로 실제 값을 디코딩 해보면 잔액을 확인할 수 있다.

표 1. 잔액 확인 스크립트 파일 예제

```
load("tools/wbsutil.js");

var retVal = lapdu(0x90, 0xx4C, 0x00, 0x00, 0x00, 0x04)

// APDU 리턴값
var val1 = retVal[0];
var val2 = retVal[1];
var val3 = retVal[2];
var val4 = retVal[3];
var sw1 = retVal[4];
var sw2 = retVal[5];

if(sw1 == 0 && sw2 == 0) {
        print("APDU 리턴값 없음");
} else {
        sum = (val1 〈〈 24) + (val2 〈〈 16) + (val3 〈〈 8) + val4;
        print("신카드 잔액 : " + sum);
}
```

이와 같은 잔액 확인 스크립트 파일 뿐만 아니라, 이외에 기존 C 언어로 작성된 카드처리 관련 기능들을 스크립트 파일 형태로 구현이 가능하다.

IV. 시스템 평가

제안하는 스크립트 파일 기반 AFC 단말기의 구현 가능성을 검토하기 위하여 기존 단말기에 스크립트 엔진을 탑재하고 RF 오브젝트와 SAM 오브젝트를 구현하였다. 구현된 시스템을 기반으로 기본 동작 테스트와 성능을 측정 하였으며 실험환경은 표2와 같다.

표 2. 실험 환경

구분	기존 시스템	스크립트 기반
CPU	s3c2440(400Mhz)	s3c2440(400Mhz)
OS	Linux-2.6.21	Linux-2.6.21
개발언어	C	C, 자바스크립트
Script 엔진	없음	Spider Monkey

실험은 표1의 잔액 확인용 스크립트 파일에 대해 세부 기능 단위로 수행 시간을 측정 하였다. 크게 하드웨어 디바이스의 코드를 수행하는 RF 디바이스 초기화 부분과 APDU 명령어를 전송/수신하는 부분에서의 시간 측정, 그리고 순수 스크립트 코드를 수행하기 위해 소요된 시간으로 구분하여 측정하였다.

표 3. 스크립트 파일의 실행 속도 측정 결과

구분	측정 시간
스크립트 처리 시간	19 ms
RF Detect 처리	61 ms
APDU 명령어 처리	21 ms
총 수행 시간	101 ms

표3은 스크립트 파일을 크게 세 부분으로 나누어 실행 시간을 측정한 결과이다. 결과를 보면 처음 스크립트 파일이 로드되고 카드 태그를 처리하는 RF Detect 처리에 61ms 정도의 시간이 소요되었다. 그리고 단말기와 카드가 APDU 명령어를 사용한 통신 처리에 21ms 정도 소요되는 것을 확인하였다. 그리고 이외에 순수 스크립트 처리에 19ms 정도 소요되었다. 결론적으로 잔액을 확인하는 스크립트 파일의 처리시간은 총 101ms 가 걸렸다.

크게 RF 오브젝트를 통해 H/W 디바이스 드라이버에서 소요되는 RF Detect 처리부와 APDU 명령어 처리부에 약 80% 정도 이상의 시간이 소요되었다. 이 부분들은 자바 스크립트에서 C로 구현된 디바이스 드라이버를 호출하는 구조이기 때문에 함수 호출로 인한 약간의 지연 외에는 C와 동일할 것이다.

이와 비교하여 순수 계산 로직인 스크립트 처리에 소요된 시간은 19ms 정도로 상대적으로 빠르게 처리되는 것을 확인 하였다. 물론 실험에서 사용한 스크립트 파일은 비트 연산과 출력 함수를 사용하는 정도의 간단한 구조이기 때문에 복잡한 스크립트 파일의 경우 스크립트 수행속도가 증가할 것이다.

하지만 스크립트 처리에 소요되는 시간보다는 RF 처리와 같이 하드웨어에 의존적인 부분에서의 처리 속도가 전체 성능에 더 큰 영향을 준다는 것을 확인할 수 있었다. 따라서 기존 C로 작성된 Application과 스크립트로 작성된 Application과의 차이는 크게 나지 않을 것으로 예상된다.

V. 결 론

본 논문은 스크립트 파일 기반의 AFC 단말기의 카드처리 부분을 설계 하였고, 실제 스크립트 기반으로 잔액확인 기능을 구현하여 시스템의 구현 가능성을 확인 하였다. 또한 성능 측정을 하여 세부 기능별 수행 속도를 확인하였다. 측정 결과를 통해 기존 C로 작성된 Application을 스크립트 기반으로 변경하는 것이 현재의 카드처리 수준의 범위에서는 성능적으로 문제가 없음을 확인하였다.

향후 이와 같은 결과를 통해 AFC 단말기의 카드처리 기능을 스크립트 파일 기반으로 운영할 경우 유연하게 시스템을 운영할 수 있으며, 신규 카드의 추가나 비즈니스 정책의 변경시 쉽게 적용이 가능할 것으로 예상할 수 있다.

향후 이번 실험에서 테스트하였던 잔액 확인 스크립트뿐만 아니라 향후 추가적으로 지불, 충전과 같은 다양한 처리를 수행할 수 있는 스크립트의 제작 및 검증이 필요할 것이다.

감사의 글

이 논문은 2010년도 지식경제부의 지원을 받아 수행된 연구 사업임(WBS AFC 표준 SW 솔루션 개발, 과제번호 10038474)

참고문헌

[1] W. Jung, S. Lee, H. Oh, S. Moon, “Performance Evaluation of Javascript Engines”, Korea Information Processing Society 2009 Fall Conference, Nov, 2009.

[2] http://phonegap.com/

[3] http://www.mozilla.org/js/spidermonkey