소형 디지털기기용 내장 마이크로브라우저의 설계 및 구현

최 원 호*, 진 민 식*, 정 준 영*, 김 도 우*, 정 민 수*
*경남대학교 컴퓨터공학과

Design and Implementation of Embedded MicroBrowser for Small Size Digital Appliance

Won-Ho Choi*, Min-Sik Jin*, Jun-young Jung*, Do-Woo Kim*, Min-Soo Jung*
*Dept. of Computer Engineering, Kyungnam University

요 약

무선 단말기를 이용한 무선 인터넷 시장은 점점 커져간다. 또한, 무선 단말기에 내장 가능한 프로그램 개발도 발전하고 있다. 이 중에서도 무선 인터넷에 접속하게 하는 마이크로브라우저에 대한 개발도 이루어 졌다. 하지만, 마이크로브라우저에 관한 기능은 아직까지 미비하다. 마이크로브라우저는 HTTP 브라우저와 WAP 브라우저로 나뉘어져 있고, 현재 이들 브라우저의 제약점들을 개선하기 위해서 자바 이용하는 브라우저가 개발되고 있다.

본 논문에서 자바를 기반으로 하는 이동단말기용 마이크로브라우저에 대하여 설계 및 구현을 하였다. 자바를 사용하기 때문에 마이크로브라우저는 무선 인터넷 상에서 필요한 자바 응용 프로그램들을 다운로드 받을 수 있고, 또 이 응용 프로그램을 단말기 내에서 실행시킬 수 있다. 따라서 자바를 기반으로 하는 마이크로브라우저는 보다 낳은 무선 인터넷 항해를 할 수 있도록 해준다.

제 1 장 서론

우리는 양방향 페이저, 셀룰러폰, 무선기능이 탑재된 PDA 등 선으로부터 자유로운 통신의 시대를 맞이하게 되었다. 모바일 인터넷 환경과 모바일 단말기는 현재의 데스크톱 기준으로 접근하기에는 아직 전력 소모량, 메모리 크기, 디스플레이 크기, 전송 속도, 안정성 등에서 많은 어려움이 있다. 따라서 기존 표준을 가능하면 따르면서 무선 환경에 적합한 프로토콜을 만들려는 움직임이 일어나게 됐다. 현재 사용되고 있는 모바일 익스플로러나 WAP 브라우저를 장착한 무선 단말기가 시판되고 있지만 그 사용사의 기능들은 아직까지 미흡하다. 하지만 자바는 이러한 모바일 솔루션을 대표하는 기술로서 주목 받고 있다. 자바는 이미 시장 지배적인 기술일 뿐만아니라, 그 변화의 속도가 엄청 빠르다. 그 중에서도 가장 변화의 속도가 빠른 분야가 무선 분야이다. 무선 시장의 주요 제조업체들이 자바를 선택하는 이유로는 동적인 응용 프로그램의 다운로드, 크로스 플랫폼 호환성, 향상된 사용자 경험과 역동성, 비연결성, 보안 문제 등이 있다 본 연구에서는 소형 디지털 단말기기에 적합한 자바 플랫폼인 J2ME 플랫폼의 CLDC(Connected, Limited Device Configuration)를 기반으로 하는 KVM(K Virtual Machine)을 사용하는 소형 브라우저를 설계하고

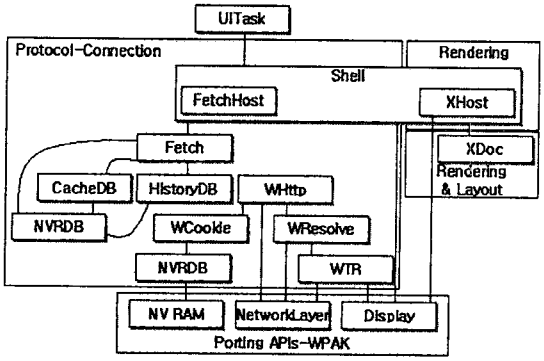
† 정보통신부에서 지원하는 2000 대학기초 연구지원사업으로 수행

구현한다.

제 2 장 관련 연구

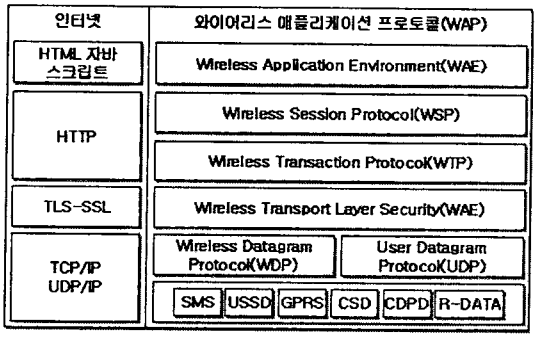
2.1 모바일 익스플로러

모바일 익스플로러는 모질라(Mozilla)를 기반으로 하고, 네 개의 컴포넌트(component) 아키텍처를 구성하고, 각각의 컴포넌트 아키텍처(architecture)는 네비게이션(navigation) 컴포넌트, 프로토콜(protocol) 컴포넌트, 파싱과 레이아웃(layout) 컴포넌트, 렌더링(rendering) 컴포넌트이고 <그림 2.1>와 같다. 네비게이션 컴포넌트는 사용자가 URL 에 연결할 수 있거나 연결된 링크로 이동할 수 있게 한다. 브라우저의 프로토콜 컴포넌트는 URL 스키마와 호스트를 찾기 위해 URL 을 파싱한 다음 IP 를 검증하고 호스트에 연결한다. 또한 프로토콜 컴포넌트는 요청을 호스트에 보내고 요청에 대한 결과를 받는다. 파싱 컴포넌트는 전달받은 내용을 장치의 기능에 맞게 전환하고 렌더링 컴포넌트가 변환된 값을 화면에 보여준다.



<그림 2.1> 모바일 익스플로러 컴포넌트 아키텍처

<그림 2.2> WAP 프로토콜의 구조



2.2 WAP 브라우저

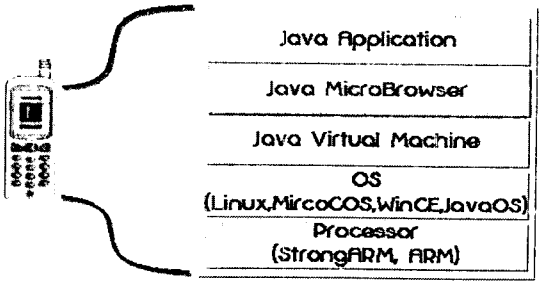
WAP 의 구조는 이동통신 장비의 애플리케이션 개발을 위해 확장 가능한 환경을 제공한다. 이것은 전체 프로토콜이 계층화된 구조를 가지고 있기 때문에 가능하다. 각 계층은 그 보다 상위 계층에 의해 접근할 수 있고 다른 외부 서비스나 애플리케이션에 의해서도 직접 접근할 수 있다. WAP 의 구조는 <그림 2.3>와 같이 WAE(Wireless Application Environment), WSP(Wireless Session Protocol), WTP(Wireless Transaction Protocol), WTLS(Wireless Transport Layer Security), WDP(Wireless Datagram Protocol), Bearers 로 구성된다.

제 3 장 마이크로브라우저 설계

3.1 마이크로브라우저의 플랫폼 설계

KVM 을 탑재한 마이크로브라우저는 무선 단말기 플랫폼을 목표로 설계되고, HTTP 프로토콜을 이용해 네트워크 망에 접속하여 필요한 자바 응용 프로그램을 다운로드 받고, 이를 가상 기계을 통해 실행 시킬수 있도록 마이크로브라우저를 설계 및 구현한다.

마이크로브라우저는 선 마이크로시스템즈에서 발표한 소형 디지털기기를 위한 J2ME 을 기본 플랫폼으로 한다. 그리고, 128K ~ 512K 의 메모리 여유 공간과 16 ~ 32 비트 프로세서, 주로 배터리를 사용하는 저전력 소모, 9600bps 이하의 제한된 대역의 네트워크 연결성을 가진 디바이스들을 목표로 설계된 CLDC 컨피큐레이션을 사용하고, 자바 응용 프로그램을 실행시킬수 있도록 가상 기계인 KVM 을 이용할 수 있도록 KVM 을 탑재한 마이크로브라우저의 플랫폼을 설계한다. 다음의 <그림 3.1>과 같은 플랫폼을 가진다.

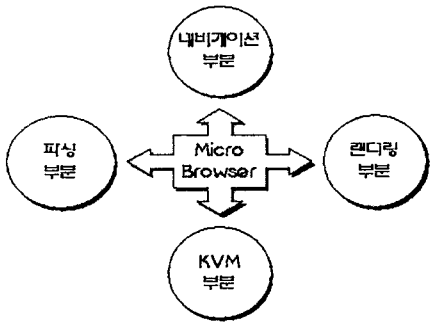


<그림 3.1> 마이크로브라우저의 플랫폼

3.2 마이크로브라우저의 구성

마이크로브라우저의 시스템 구성은 네비게이션 부분, 파싱 부분, 렌더링 부분, 가상 기계 부분의 4 개의

모듈로 나뉘어진다. 각 모듈들은 서로간의 상호작용을 통해 사용자가 요청한 데이터를 받아서 처리한다. 다음 <그림 3.2>는 마이크로브라우저에 대한 시스템 구성도이다.



<그림 3.2> 마이크로브라우저의 시스템 설계도

① 네비게이션 부분

네비게이션(Navigation) 부분은 사용자가 요청한 웹서버의 URL 을 받아서 이를 실제 IP 주소로 변환시키고, 그 주소값을 이용해서 웹서버에 연결하여 문서를 받아온다. 그리고 현재 페이지 상태에서 이전 페이지로의 이동과 앞 페이지로 이동의 기능을 갖는다.

② 파싱 부분

파싱(Parsing) 부분은 네비게이션 부분에서 웹서버로부터 받아온 HTML 문서를 파싱한다. 파싱은 HTML 명세를 따라서 수행하고, 각 태그(tag)에 대해서는 별도의 함수들을 통해 처리하고, 그 결과를 렌더링 부분으로 전달한다.

③ 렌더링 부분

렌더링(Rendering) 부분은 파싱 부분에서 전달받은 파싱 결과를 화면에 표시하는 기능을 한다.

④ KVM 부분

KVM 부분은 HTML 문서를 파싱할 때 발생하는 애플릿(applet) 태그의 처리를 하는 부분이다. 지정된 경로부터 자바 클래스 파일을 다운로드 받고, 이를 KVM 에 전달하여 응용 프로그램을 수행하도록 한다.

3.3 네비게이션 부분의 설계

본 논문에서 설계된 마이크로브라우저는 일반 퍼스널 컴퓨터에서 사용되는 브라우저의 네비게이션 부분을 축약했다고 할 수 있다. 마이크로브라우저의 네비게이션 기능은 모질라(Mozilla) 1.0 의 기능을 축약시켜서 설계되었다. 마이크로브라우저에서는 HTTP 1.0 을 지원하고 TCP/IP 를 기반으로 하고 있다.

3.4 파싱 부분의 설계

파싱 부분은 W3 에서 제공하는 'HTML 3.2 Reference Specification'을 참조하여 파서(parser)를 만들든다. HTML 태크에 따라서 토큰을 만들고 트리를 만든다. 생성된 구문에 대해서 화면에 표시할 수 있도록 *Parsing()*함수를 만든다.

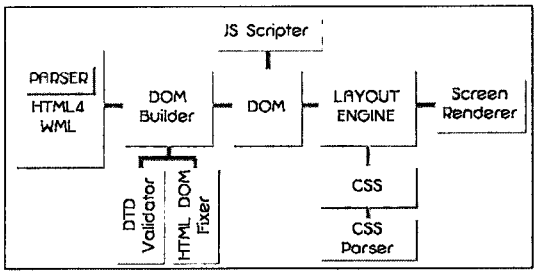
3.5 렌더링 부분의 설계

렌더링(Rendering) 부분은 웹서버로부터 읽어온 HTML 문서를 파싱하고, 파싱된 결과를 브라우저의 디스플레이 화면에 맞게 웹문서를 표시한다.

3.6 KVM 과의 인터페이스 설계

마이크로브라우저에서 가장 중요한 부분은 마이크로브라우저와 KVM 과의 연결에 있다. 본 논문에서 사용하는 KVM 은 일반적인 자바 가상 기계과는 클래스를 로딩하는 부분과 JAR 파일을 읽어 들인다는 것에서 차이점이 있다. 이 차이점들을 브라우저에서 해결을 해야지만 KVM 에서 자바 응용 프로그램을 실행시킬 수 있다. 브라우저와 KVM 간의 인터페이스에서 *extractClass()*를 이용해서 JAR 파일내의 클래스 파일을 추출해내고, *CheckVerify ()*를 사용해 클래스 파일의 사전검증에 대한 정보를 스택맵에 검증정보를 추가한다. 그리고 클래스 파일을 *KVMStart ()*을 이용해 KVM 에 전달한다.

KVM 은 사전검증에서 스택맵(StackMap)이라는 검증정보를 클래스 파일에 추가하여 사용자에서 검증시에 검증정보를 이용하여 자바 가상 기계에서의 검증보다 빠르게 수행한다.



<그림 3.3> 마이크로브라우저의 아키텍처

제 4 장 마이크로브라우저의 구현

4.1 구현 환경

본 논문에서 구현한 KVM 을 탑재한 마이크로브라우저의 구현 환경은 다음과 같다.

윈도우 운영체제인 WinNT4.0 환경하에서 ANSI C 와

비주얼 C++ 6.0 ServicePack 3.0 을 이용하여 코딩하고 컴파일하였다. 브라우저 소스로는 마이크로소프트에서 제공하는 ActiveX 와 공개소스인 모질라(Mozilla) M18 의 일부를 수정해서 사용하였다. 그리고, KVM 소스로는 썬마이크로시스템즈에서 공개된 J2ME 플랫폼의 CLDC 1.0 내의 KVM 소스를 사용했다.

4.2 네비게이션 부분의 구현

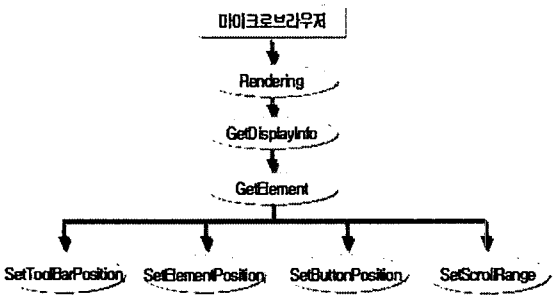
네비게이션 부분은 마이크로브라우저에서 중요한 부분으로서 사용자로부터 직접 URL 을 입력받거나 링크(link)를 통해 요청한 웹페이지로 이동한다. 마이크로브라우저 내에는 히스토리(History) 데이터베이스가 있었서 이미 방문한 웹페이지의 URL 이나 사용자로부터 요청 받은 URL 을 저장할 수 있다. 네비게이션은 입력받은 URL 을 URL 스키마를 통해서 실제 IP 주소를 파싱하고, 그런후에 IP 를 분석하고 웹페이지에 접속하게 한다. 네비게이션을 통해서 컨텐트(content)를 요청하고 응답 받을 수 있다.

4.3 파싱 부분의 구현

파싱 부분은 마이크로브라우저가 요청 웹페이지에서 읽어온 HTML 문서를 HTML3.2 명세서에서 마이크로브라우저에 맞게 축약한 HTML 태그들을 기준으로 파싱을 한다. 그리고, 파싱 문서에 서술되지 않은 HTML 태그들에 대해서는 파서(Parser)가 처리하지 않는다. HTML 문서에서 지정한 태그들만 선택하여 파서 트리를 만들고, 이를 화면에 표시할 때 각 태그가 갖는 기능을 수행할 수 있도록 태그에 대해서 함수를 작성한다. 그리고, 이미지 파일과 자바 클래스 파일을 문서로부터 추출하여 저장하고, 자바 클래스 파일인 경우에는 KVM 에서 컴파일할 수 있도록 전달한다.

4.4 렌더링 부분의 구현

렌더링 부분은 요청한 웹페이지를 브라우저 화면에 알맞게 문자열과 이미지, 그리고 자바 클래스 파일의 실행 결과들을 배치한다. 그리고 화면상에 버튼들의 위치와 스크롤의 범위를 정한다.

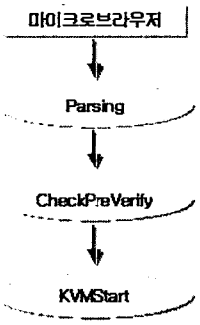


<그림 4.1> 렌더링 부분의 구성

4.5 KVM 과의 인터페이스 구현

KVM 과 브라우저와의 연동 기술은 본 논문에서 가장 중요한 핵심적인 부분이다. KVM 과 브라우저에서는 일반적으로 사용하고 있는 브라우저에서 자바 가상 기계에 자바 클래스 파일을 전달함으로서 자바 가상 기계 호출하는 방식과 유사하다. 하지만 자바 가상 기계를 소형화한 KVM 에서는 한단계의 작업을 더 거쳐야만 한다. KVM 에서는 자바 클래스 파일에 대한 사전 검증을 한다. 그런후에 자바 클래스 파일에 대한 검증 정보를 스택맵에 저장하고, 이 스택맵을 클래스 파일에 추가한다. 이 추가된 정보는 KVM 이 디바이스 검증을 할 때 바이트코드를 다시 검증하는 것이 아니라 추가된 스택맵을 이용해서 검증하여 자바 클래스 파일이 올바른지를 확인하고 클래스 파일을 수행시킨다.

KVM 과 마이크로브라우저의 인터페이스 구현은 자바 클래스 파일을 KVM 에 전달하기 위해 자바 클래스 파일의 사전 검증 여부를 확인한다. 그리고, 사전 검증이 되어 있지 않으면 자바 클래스 파일을 사전 검증을 한후에 클래스 파일이 저장되어 있는 경로와 파일명을 KVM 에 넘긴다. 실제로 사용되는 가상 기계는 CLDC 내의 KVM 소스를 그대로 사용했다.



<그림 4.4> KVM 과의 인터페이스 구성

4.6 구현 비교 분석

본 논문에서 구현한 KVM 을 탑재한 마이크로브라우저는 기존의 HTTP 브라우저, WAP 브라우저와 비교를 하면 다음의 <표 4.1>과 같다.

<표 4.1> 무선 단말기용 마이크로브라우저 비교 분석

	HTTP 브라우저	WAP 브라우저	KVM 을 탑재한 브라우저
인터넷 접속	온라인만 지원	온라인만 지원	온/오프라인 지원
자바 프로그램 실행	지원하지 못함	지원하지 못함	지원
플랫폼 호환성	HTTP 를 사용하는 플랫폼만 가능	WAP 사용하는 플랫폼만 가능	플랫폼에 제한 없음

현재 사용되고 있는 무선 단말기용 마이크로브라우저는 인터넷 연결만이 가능하지, 필요한 응용 프로그램들을 다운로드 하지 못한다. 그러나 KVM 을 탑재한 마이크로브라우저는 기존 마이크로브라우저가 갖는 기능들을 가지면서, 기존 마이크로브라우저에서 미약한 기능을 자바라는 언어를 사용함으로서 개선할 수 있다.

제 5 장 결론 및 향후 연구 방향

본 논문에서 구현한 'KVM 을 탑재한 마이크로브라우저'는 TCP/IP 기반의 HTTP 프로토콜을 이용해서 무선 인터넷에 접속하여 자바 응용 프로그램을 다운로드 받을 수 있고, 오프라인 상태에서도 자바 응용 프로그램을 실행시킬 수 있다.

무선 단말기에 KVM 이 탑재된 마이크로브라우저를 장착하면 HTTP 브라우저나 WAP 브라우저가 갖는 기능은 뿐만 아니라 오프라인 인터넷을 사용할 수 있다는 장점을 가질 수 있다.

앞으로는 KVM 이 탑재된 마이크로브라우저가 지니를 기반으로 하는 홈네트워크에도 접속하여 서비스를 받을 수 있도록 KVM 의 성능을 향상시키는 연구와 네트워크 연결성과 보안 기능을 강화시키는 연구가 요구된다. 그리고 브라우저의 크기를 좀 더 소형화하는 기술에 대한 연구가 기대된다.

[참고 문헌]

[1] CLDC 1.0 Specification, http://java.sun.com/products/cldc

[2] MIDP 0.9 Specification, http://java.sun.com/products/midp

[3] KVM WhitePaper, http://java.sun.com/products/ kvm

[4] J2ME, http://java.sun.com/j2me

[5] WAP 포럼, http://www.wapforum.com

[6] 무선 인터넷 & WAP 개발포탈, http://mobile.daesang.co.kr

[7] WIPS, http://wap.sejong.edu

[8] B.Venners, Inside Java Virtual Machine, McGraw-Hill, 1997

[9] 곽용재, 자바가상머신 프로그래밍, 인포북, 1999

[10] Charles Arehart, Professional WAP, Wrox Press, 2000

[11] William R. Stanek, Netscape Mozilla Source code guide, IDG Books, 1999

[12] A. Taivalsaari, *Implementation a Java Virtual Machine in the java programming Language*, SUN Lab, 1997.

[13] Rational Software Co, *Rational Rose User Guide*, 1997.

[14] Symantec Corporation, *Cafe™ Companion*, 1996

[15] 이종동, 정민수 "자바 메쏘드 호출 관계 표시기의 설계 및 구현", 한국정보과학회 98 봄학술발표논문집, 제 25 권 1 호, pp. 74-76, 1997.

[16] *http://www.artima.com/*, ARTIMA SOFTWARE COMPANY

[17] *http://java.sun.com/*, Sun Microsystem, Java Home Page