논문 2008-03-05

리눅스 환경에서 WIPI를 지원하기 위한 HAL (Handset Abstraction Layer) 이식

(Poring of WIPI HAL in Embedded Linux)

박우람, 김태웅, 박찬익*
(Woo-Ram Park, Tae-Woong Kim, Chan-Ik Park)

Abstract : This paper persents how to port HAL (Handset Abstraction Layer) on embedded Linux to support WIPI (Wireless Internet Platform for Interoperability). As smart phones are widespread nowdays the operating system is changing from a simple kernel like Qualcomm REX OS to more feature-rich Linux kernel. For this reason, we investigate the internal structure of HAL on REX OS and design how to port it to embedded Linux. Careful analysis leads us to identify several porting issues such as thread support, graphical user interface. In addition, we describe some problems discovered during the implementation process and propose alternative architecture of HAL for WIPI on Linux.

Keywords : WIPI, Linux, HAL

I. 서 론

WIPI (Wireless Internet Platform for Interoperability)는 모바일 무선인터넷 플랫폼으로 통일된 인터페이스를 제공함으로써 모바일 응용프로그램의 개발에 소요되는 불필요한 오버헤드를 줄일 수 있다. 이러한 WIPI는 WIPI 런타임 엔진과 HAL (Handset Abstraction Layer)로 크게 구성되어 있으며, HAL은 WPI 런타임 엔진에 단말기와 OS의 종류에 관계없이 동일한 인터페이스를 제공함으로써 WIPI 플랫폼을 이식할 때 독립성을 지원하고 있다.

현재 모바일 폰에서 동작하고 있는 WIPI 플랫폼의 구조는 < 그림 1 >과 같다. REX (Real-Time Executive) OS 상에서 WIPI 플랫폼이 동작하고 있으며, WIPI HAL은 REX OS에 맞게 포팅되어 있다.

* 교신저자(Corresponding Author)
논문접수 : 2008. 03. 13. 채택확정 : 2008. 03. 19.
박우람, 김태웅, 박찬익 : 포항공과대학교 컴퓨터공학과
※ 본 연구는 교육인적자원부의 BK21 사업과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음.
(IITA-2008-C1090-0801-0045)

하지만 최근 모바일 폰의 성능이 향상되면서 REX OS 대신 리눅스를 모바일 폰의 OS로 채택하여 보다 다양한 서비스를 효율적으로 제공하고자 하고 있다. 그에 따라 LiMo[1]와 같은 개발 기구가 조직되고, 모바일 폰 제조회사가 제휴하여 리눅스 모바일 폰의 개발이 진행되고 있다. [2-4]

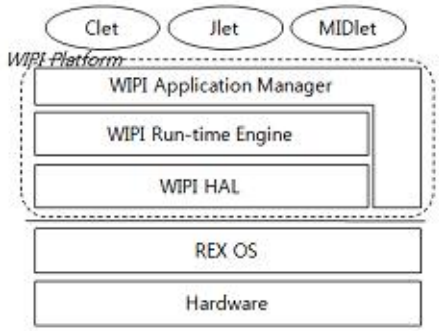

그림 1. WIPI 플랫폼의 구조
Fig 1. WIPI Platform Architecture

그에 맞게 모바일 무선인터넷 플랫폼인 WIPI를 리눅스 기반으로 재구성 하여야 할 필요성이 생겼으며, REX OS와 리눅스의 차이에 따라 발생할 수 있는 문제점을 파악하여 앞으로 리눅스 상에서의

WIPI 발전 방향에 대하여 고려하여야 한다.

이 논문에서는 ARM 기반의 임베디드 리눅스 모바일 플랫폼 상에서 WIPI HAL을 어떻게 포팅하고, thread 지원과 Graphic User Interface (GUI)를 어떻게 지원하였는지에 대하여 설명하고자 한다. 나아가 이 과정에서 발생할 수 있는 문제점과 앞으로 리눅스 상에서 WIPI HAL의 방향에 대하여 논하고자 한다.

II. 본 론

1. WIPI on Linux (WoL)의 구조

WIPI 플랫폼을 리눅스에서 동작시키기 위해서는 가장 하부에 위치한 HAL을 리눅스에 맞게 재개발 하여 리눅스에서 제공하는 서비스를 효과적으로 WIPI 런타임 엔진과 같은 상위 컴포넌트에서 사용할 수 있도록 하여야 한다.

그리고 WIPI HAL과 상위 컴포넌트를 지원하기 위하여 glibc, PTH[5] 라이브러리와 같은 리눅스 라이브러리를 추가적으로 사용하였다.

마지막으로 Qtopia[6]를 이용하여 GUI를 지원하였으며 리눅스 응용프로그램과 WIPI 응용프로그램 간의 멀티프로세싱 환경을 구축하였다. < 그림 2 >는 실제 구현된 WIPI on Linux 플랫폼의 구조이다.

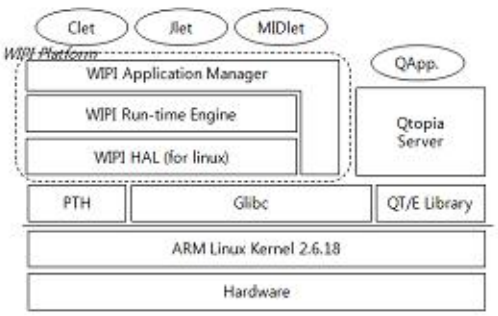

그림 2. WIPI on Linux 플랫폼의 구조
Fig. 2. WIPI on Linux Platform Architecture

2. WIPI on Linux의 구현

REX OS에서 동작하는 WIPI HAL을 리눅스로 포팅하기 위해서는 우선 HAL에서 제공하는 API 중 REX OS에 의존성을 가지는 API를 구분하는 것이 필요하다. 디바이스 (문자입력, 미디어, 프레임 버퍼, 시리얼 등)나 시스템 정보에 관련된 API는 WIPI 플랫폼이 동작할 하드웨어에 의존성을 가지는 API로 하드웨어 디바이스 드라이버를 리눅스 버전으로 재구성함으로써 쉽게 사용하는 것이 가능하다.

파일 서비스와 메모리 관리, 타이머에 관련된 부분은 REX OS에 의존성을 가지고 있는 API로 리눅스에서 사용하기 위해서는 기존의 WIPI HAL을 수정해야만 한다. 메모리 관리와 타이머는 REX OS와 리눅스 간의 변수 타입이나 기준 시각이나 같은 기본 값이 다르기 때문에 이를 수정해줌으로써 쉽게 사용이 가능하다. 반면 파일 서비스는 REX OS와 리눅스에서 사용하는 파일 시스템이 서로 다르기 때문에 WIPI 2.0 표준[7]에 따라 HAL을 재구현 해주어야 한다. 이 때 REX OS와 리눅스 간의 파일 정책 상의 차이점에 대해 생각을 하여야 한다. 대표적으로 REX OS의 경우 파일이 open 되어 있을 때, remove 명령이 들어올 경우 STATUS_BUSY를 반환하고 삭제를 하지 못 하도록 되어있지만, 리눅스의 경우 파일이 open 되어 있을 경우 우선 success를 반환한 뒤, 파일이 close 된 후 해당 명령이 수행되어 삭제되도록 하고 있다. 이러한 차이가 WIPI 플랫폼의 상위 컴포넌트에서는 혼선을 가져올 수 있기 때문에 lock이나 swap 파일 등을 이용하여 문제를 해결하여야 한다.

또한 WIPI 플랫폼에서 사용되는 task를 wait 시키거나 wakeup이 가능하도록 지원해주는 API가 REX OS에 의존성을 가지고 있다. 이 부분은 scheduling과 밀접한 관계를 가지고 있으며, 상위 컴포넌트에서 사용하고 있는 user-level scheduling을 지원하기 위하여 사용한 PTH 라이브러리를 이용하여 재구성하였다.

리눅스 응용프로그램과 WIPI 응용프로그램 간의 멀티프로세싱을 보다 효과적으로 사용할 수 있도록 Qtopia-arm 공개버전을 이용하여 GUI 환경을 추가적으로 구축하였다. Qtopia에는 WIPI 플랫폼을 동작시키기 위한 WIPI Qtopia Application을 작성하였다. 이 응용프로그램을 동작시키면 Qtopia server에 event block signal을 전송하여 qtopia로 전달되는 이벤트를 일시적으로 막아서 WIPI와 이벤트가 중복 처리되는 것을 막아주며, 그 후 WIPI 플랫폼을 동작시키게 된다. 즉 백그라운드에서 qtopia 응용프로그램이 동작하고 있지만, 실제 화면에 보여 지는 것은 WIPI 응용프로그램만이 보여지게 된다. 이것은 Qtopia에서 사용하는 kernel-level scheduling 라이브러리인 pthread와 WIPI on Linux에서 사용하는 user-level scheduling 라이브러리인 PTH 라이브러리의 충돌

을 막기 위한 구조로 이에 대해서는 3장에 자세히 설명하겠다.

3. HAL 포팅 이슈

WIPI HAL을 리눅스로 포팅하기 위해서는 크게 thread 지원과 GUI 지원에 대하여 고려를 하여야 한다.

REX OS에서는 thread를 지원하지 않기 때문에 그동안 WIPI 런타임 엔진에서 thread를 생성하고 이를 user-level에서 scheduling 하고 있다. 반면 리눅스에서는 thread와 kernel-level scheduling을 지원하고 있기 때문에 대다수의 리눅스 응용프로그램은 pthread와 같은 라이브러리를 이용하여 이 기능을 지원받고 있다.

WIPI 런타임 엔진의 구조적인 변경을 막기 위하여 리눅스에서 user-level scheduling을 지원하는 PTH 라이브러리를 사용하여 WIPI on Linux 플랫폼을 구축하였다. 이를 지원하기 위해서 glibc를 수정하여 pthread 대신 PTH 라이브러리를 사용하도록 변경해 주어야 한다. 별도로 스케줄링 되는 리눅스 응용프로그램이 없이 WIPI on Linux가 독립적으로 동작하여 WIPI 응용프로그램을 동작시킬 때는 kernel-level scheduling과 충돌이 일어날 경우가 없기 때문에 문제없이 동작이 가능하다.

하지만 사용자에게 보다 편리한 인터페이스를 제공하고, 리눅스 응용프로그램과 WIPI 응용프로그램을 보다 쉽게 멀티프로세싱하기 위하여 Qtopia와 같은 리눅스 기반의 GUI를 제공하고자 하면, kernel-level과 user-level scheduling 간에 충돌을 일으켜서 정상적인 scheduling이 수행되지 않는 문제가 발생한다. 그렇기 때문에 pthread 라이브러리를 사용하고 있는 Qtopia server에서 리눅스 응용프로그램과 PTH 라이브러리를 사용하고 있는 WIPI on Linux를 멀티 프로세싱 하기 위해서는 둘 간의 충돌을 막기 위하여 WIPI 응용프로그램을 동작시키기 전에 Qtopia 응용프로그램을 일시적으로 suspend 시켜주어야 한다. 그리고 WIPI on Linux 플랫폼의 동작이 완료된 후에는 기존의 상태로 복구시켜주어야 하기 때문에 Qtopia server와 리눅스 응용프로그램을 완전히 suspend 시킨 후 다시 동작시키는 것이 아니라, 이벤트의 입력만을 막아주고 풀어줌으로써 쉽게 기존 상태로 복구해줄 수 있다.

4. WIPI on Linux의 발전 방향

3장에서 언급하였듯이 WIPI on Linux 플랫폼 상에서 WIPI 응용프로그램 혹은 리눅스 응용프로그램을 독립적으로 동작시키는 데는 문제가 없지만, 둘 간의 멀티프로세싱이 필요한 경우에는 user-level scheduling과 kernel-level scheduling 간의 충돌이 발생하게 된다. 이것은 기존 WIPI 플랫폼이 REX OS 상에서 동작을 해야 했기 때문에 어쩔 수 없이 WIPI 런타임 엔진에서 scheduling을 제공해야 하는 문제 때문에 발생하였다고 볼 수 있다.

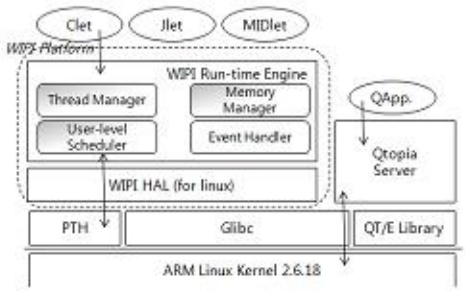

그림 3. WIPI on Linux Scheduling 흐름도
Fig 3. WIPI on Linux Scheduling Flow

< 그림 3 >은 현재 구성된 WIPI on Linux 플랫폼의 scheduling 흐름도이다. WIPI 런타임 엔진에서 thread와 scheduling을 지원하기 위한 각각의 manager가 존재하며, 또한 thread를 구성하고 관리하며, WIPI 플랫폼에서 사용할 memory를 관리하는 memory manager가 존재한다. 하지만 이 기능은 현재 리눅스 커널에서 상위 컴포넌트에 동일한 서비스를 제공하고 있기 때문에 이것을 WIPI 플랫폼에서 이용할 수 있다. 그렇기 때문에 앞으로 모바일 리눅스 플랫폼 상에서 리눅스 응용프로그램과 WIPI 응용프로그램을 멀티프로세싱 하기 위해서는 WIPI HAL에 있는 task API를 확장하여 WIPI 응용프로그램의 scheduling을 관리하고 현재 상태를 파악하여 user-level scheduling과 kernel-level scheduling 간에 발생하는 충돌을 줄여나가야 할 것이다.

이와 같이 WIPI on Linux의 HAL을 변경함으로써 개발의 효율성과 호환성을 높여줄 수 있으며, 리눅스 응용프로그램을 WIPI 플랫폼으로 포팅할 필요없이 쉽게 사용을 함으로써 보다 많은 서비스를 제공해줄 수 있을 것으로 생각된다.

< 그림 4 >는 4장에서 언급한 WIPI on Linux 플랫폼의 발전 방향을 표현한 것이다.

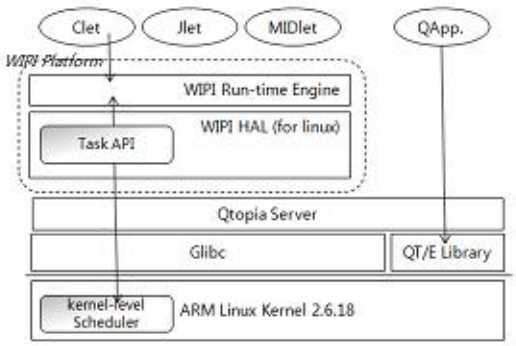

그림 4. WIPI on Linux 플랫폼의 발전 방향
Fig 4. The development direction of WIPI on Linux

III. 결 론

이 논문에서는 앞으로 리눅스 모바일 폰 시장의 확대와 맞물려 WIPI 플랫폼의 리눅스

이식의 필요성을 느끼고 이를 ARM 보드 상에서 구현하였다. WIPI on Linux 플랫폼을 구현할 때 중요한 이슈 중 하나는 REX OS와 리눅스 간의 정책의 차이, 그리고 사용하는 라이브러리의 차이에 따라 HAL을 어떻게 리눅스에 맞게 포팅할 것인가이다. 그리고 HAL을 최적화 하여 기존 WIPI 플랫폼에서 WIPI 응용프로그램이 동작할 때와 유사하거나 혹은 향상된 성능을 보일 수 있도록 디자인하는 것이 필요하다.

또 하나의 이슈는 WIPI 응용프로그램과 리눅스 응용프로그램 간의 멀티프로세싱이다. 멀티프로세싱을 통해 리눅스에서 개발된 다양한 응용프로그램을 WIPI 응용프로그램과 같이 사용하고 서로 정보를 교환함으로써 보다 다양한 서비스를 사용자에게 제공하는 것이 가능할 것이다. 하지만 현재 WIPI 플랫폼의 구조상으로는 user-level scheduling과 kernel-level scheduling의 충돌로 인하여 문제가 발생할 여지가 많이 있다. 그렇기 때문에 새로운 WIPI on Linux의 발전 방향을 제시함으로써 보다 효율적이고 리눅스에 호환성이 높은 WIPI on Linux 플랫폼을 이 논문에서 제안하고 있다.

참고문헌

[1] LiMo Foundation,
https://www.limofoundation.org/
[2] 삼성전자, "리눅스 모바일 플랫폼 공동개발", 전자신문, 2007.01.25.
[3] LG경제연구원, "휴대폰의 승부처, 플랫폼 경쟁", 전자신문, 2007.05.10.
[4] Jaeho Lee, Sunja Kim, Sangyun Lee, Woosik Kim, Hwangu Lee, "Implementing WIPI for Linux-based Smartphone", ICACT 2005, Volume 1, p692-696, 2005.
[5] GNU Pth - The GNU Portable Threads,
http://www.gnu.org/software/pth/
[6] Qtopia - Trolltech,
http://trolltech.com/products/qtopia/
[7] KWISF, WIPI 2.0 Specification,
http://www.wipi.or.kr/

저 자 소 개

박 우 람

2006년 고려대 컴퓨터학과 학사. 현재, 포항공대 컴퓨터공학과 통합과정.
Email: wizrampa@postech.ac.kr

김 태 웅

2007년 포항공대 컴퓨터공학과 학사. 현재 포항공대 컴퓨터공학과 석사과정.
Email: ehoto@postech.ac.kr

박 찬 익

1983년 서울대 전기공학 학사. 1985년 KAIST 전자전기공학 석사. 1988년 KAIST 전자전기공학 박사. 현재 포항공대 교수.
Email: cipark@postech.ac.kr