임베디드 시스템을 위한 KVM 네트워크 설계 및 구현

전상호° 정근재 이정원 이철훈
충남대학교 컴퓨터공학과
(shjun°, gjjeong, booster, chlee)@cnu.ac.kr

The Design and Implementation of KVM Network for Embedded Systems

Shang-Ho Jeon°, Pil-Chang Son, Myoung-Jo, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.

요 약

최근 임베디드 디바이스에 여러 가지 장점을 제공하는 자바기술과 네트워크 기능은 필수적인 요소가 되었다. 임베디드 디바이스에 적용되는 자바기술은 대부분 J2ME 이며 J2ME 에서는 KVM 과 GUI, 네트워크 API 등을 명세하고 있다. 이 중 GUI 와 네트워크 부분은 구현 시 시스템에 의존적인 부분을 따로 구현해야 하는데, 이는 native 함수로 구현할 수 있다. 본 논문에서는 자바 스펙(J2ME)에서 정의된 네트워크 API 의 기능들을 분석하여 임베디드 시스템에 적합한 네트워크 API 의 native 함수를 구현하였다.

1. 서론

최근 들어 자바의 플랫폼 독립성, 보안성, 네트워크 이동성, 실행코드의 재사용성, 작은 실행 파일 크기, 동적 적응성, 이식성, 개발의 용이성 등의 장점들 때문에 자바 기술을 임베디드 디바이스에 탑재되는 자바 가상 머신이 네트워크 기능을 제공하기 위해 구현시 시스템에 의존적인 부분을 따로 구현해야 하는데, 이는 native 함수로 구현할 수 있다[1].

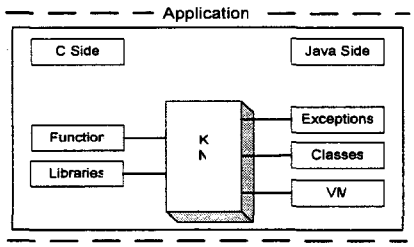
본 논문에서는 실시간 운영체제인 UbiFOS™의 네트워크 시스템과 KVM 의 네트워크 API 부분과의 상호 연동을 위한 native 함수를 구현하였다.

본 논문은 총 5 장으로 구성되어 있다. 2 장 관련연구에서 KNI 의 역할과 J2ME 의 CLDC, MIDP 에 대해서, 3 장에서는 임베디드 시스템의 자바 네트워크 설계 및 구현에 대해서, 4 장에서는 실험 환경 및 결과, 5 장에서는 결론 및 향후 연구 과제에 대해 기술한다.

2. 관련 연구

2.1 KNI (K Native Interface)*

KNI는 [그림2-1]에서 보는 바와 같이 자바와 자바 이외의 언어로 만들어진 애플리케이션이나 라이브러리가 상호 작동할 수 있도록 연결시켜주는 인터페이스로 자바에서 지원하지 못하는 플랫폼 종속적인 기능들과 이미 다른 언어로 만들어진 애플리케이션이나 라이브러리를 사용할 수 있도록 해주는 임베디드 시스템을 위한 인터페이스이다. native 언어를 이용한 자바 프로그램에서 자바로 된 부분은 여전히 플랫폼 독립적이지만 native 언어로 된 부분은 호스트 환경에 맞게 다시 컴파일 되어야 함을 의미한다[2].



[그림 2-1] KNI의 구성도

2.2 J2ME

SUN사의 J2ME 플랫폼은 다양한 제품과 임베디드 디바이스 시장을 목표로 플랫폼을 규정하고 있다.

J2ME 플랫폼에서는 CLDC(Connected Limited Device Configuration)와 CDC(Connected Device Configuration)가 사용되고 있으며, Profile로는 MIDP (Mobile Information Device Profile)만이 사용되고 있다[3].

2.3 MIDP

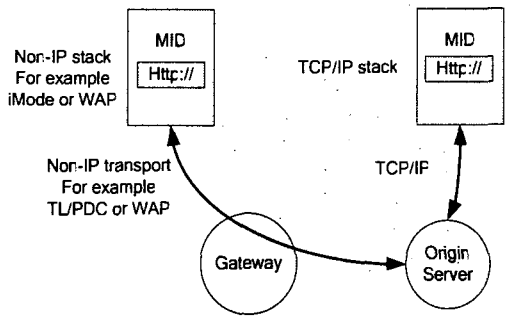
MIDP는 MID를 목표로 설계된 자바 클래스 라이브

* 본 논문은 국방과학연구소의 실시간 운영체제 인터페이스용 미들웨어 연구과제의 수행결과임

러리에 대한 명세로 CLDC를 기반으로 하고 있다. 그러므로 MIDP 명세는 CLDC의 명세를 구체화하거나, 확장 및 변경할 수 있다[3].

2.3.1 MIDP 네트워킹

MIDP에서는 CLDC의 커넥션 프레임 워크에 대해 확장된 네트워킹에 대한 정의와 구현을 제공한다. MIDP 2.0 명세서는 HTTP 1.1 프로토콜의 서브셋을 지원하고 있고, 이것은 TCP/IP 네트워크의 IP 프로토콜에 대한 구현과, WAP 및 iMode의 게이트웨이를 통한 비IP(non-IP) 프로토콜에 대한 구현을 포함한다.



[그림 2-1] Http 네트워크 연결

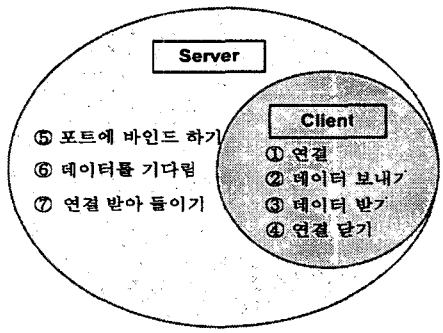
[그림 2-1]을 보면, 한 쪽은 WAP/iMode Gateway를 통해서, 다른 한 쪽은 TCP/ IP로 직접 오리진(Origin) 서버에 접속하고 있다. 클라이언트 프로그램이나 오리진 서버는 이러한 연결 방법의 차이에 대해서 알아야 할 필요가 없다. 클라이언트 프로그램은 URL을 통해 네트워크 연결을 요청할 뿐이고, 웹 서버는 HTTP 요청에 대한 응답을 전송하는 고전적인 임무를 수행하면 된다. 무선망의 연결방법에 의한 차이는 MIDP의 설계자에 의해 해결해야 할 문제일 뿐, MIDP 애플리케이션 개발자는 네트워크 연결방법에 대해서는 고려하지 않아도 된다[3].

3. 설계 및 구현

본 논문에서는 UbiFOS™의 네트워크 시스템과 KVM 의 네트워크 API 부분과의 상호 연동을 위한 native 함수를 구현하여, TCP 와 UDP 소켓을 이용한 네트워크 통신을 가능하게 하였다. UbiFOS™는 TCP, UDP 통신이 지원되며, 이것은 POSIX 규약에 맞게 정의되어 있다.

3.1 KNI를 사용한 TCP 소켓

하나의 소켓은 두 호스트 사이의 연결을 말하며, [그림 3-1]과 같은 7가지 기능을 수행한다.



[그림 3-1] TCP 서버 / 클라이언트

소켓의 기능 중 ①~④에 해당하는 메소드는 서버/클라이언트 양쪽에서 필요하며, connect(), write(), read(), close()API를 구현하였고, 서버 쪽에서만 필요한 ⑤~⑦의 기능에 해당하는 메소드는 bind(), listen(), accept()API를 구현하였다. 이러한 API를 구현하기 위해 필요한 추가 API인 getIpNumber(), getsockopt(), setsockopt(), getport()을 구현 하였다. 위와 같은 함수를 사용하여 TCP 통신이 가능하도록 native 함수를 구현하였다.

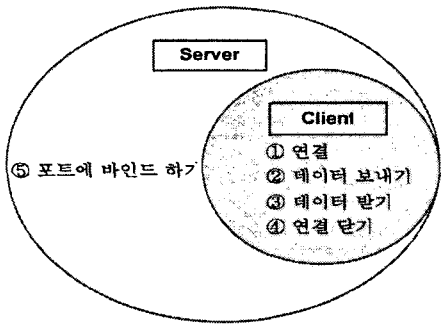
```
int open(char* name, int port, char **exception){
  ..
  fd = socket(AF_INET, SOCK_STREAM, 0);
  addr.sin_family      = AF_INET;
  addr.sin_port        = htons((unsigned short)port);
  addr.sin_addr.s_addr = ipn;
  res = connect(fd,(struct sockaddr *)&addr, sizeof(addr));
}
```

[그림 3-2] connect 함수의 사용

[그림 3-2]은 클라이언트가 서버로 접속하게 하는 connect() API의 사용 예이다.

3.2 KNI를 사용한 UDP 소켓

UDP 프로토콜에서는 소켓은 기다리는 포트와 수신 포트만 알면 된다. [그림 3-3]은 UDP 소켓의 기능을 나타내며, 프로토콜 특성상 신뢰성의 보장 없이 데이터를 보내기만 하므로 TCP 소켓과 차이점이 있다. 각각의 메소드는 connect(), write(), read(), close(), bind()의 API로 구현하였다. 또한 추가적으로 getMaximumLength(), getNominalLength()의 API를 구현하였다. 위와 같은 함수를 사용하여 UDP 통신이 가능하도록 native 함수를 구현하였다.



[그림 3-3] UDP 서버 / 클라이언트

[표 3-1]은 TCP/UDP 의 추가 API 를 간단히 설명한 것이다.

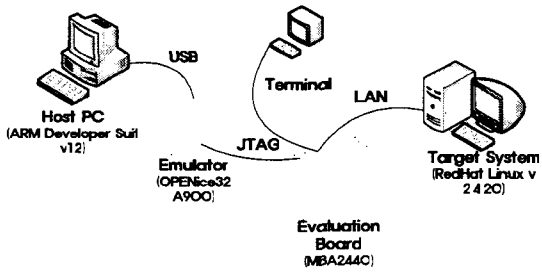
[표 3-1] TCP/UDP의 추가 API

함수	설명
getIpNumber()	IP주소를 얻어옴
getsockopt()	소켓 옵션 value를 얻어옴
setsockopt()	소켓 옵션 value를 설정
getport()	포트 넘버를 얻어옴
getMaximumLength()	데이타그램의 최대 길이를 얻어옴
getNominalLength()	데이타그램의 일반 길이를 얻어옴

4. 테스트 환경 및 결과

본 논문의 테스트 환경은 MBA2440 보드상에 실시간 운영체제로 UbiFOS™를 사용하였고, 개발도구로 ARM SDT v2.51 을 사용하였다. 또한 OPENice-A900 을 사용하여 디버깅 하였다.

[그림 4-1]은 테스트 환경을 나타내고 있다.

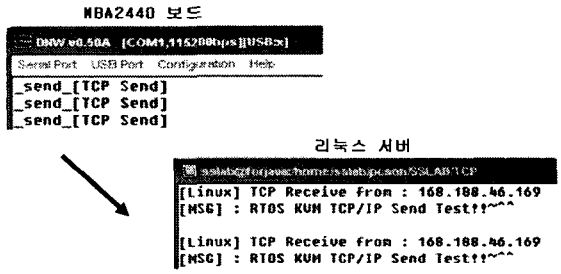


[그림 4-1] 테스트 환경

본 논문에서의 테스트는 MBA2440 보드와 리눅스 서버간의 문자열 전송을 확인하는 방법으로 수행되었다.

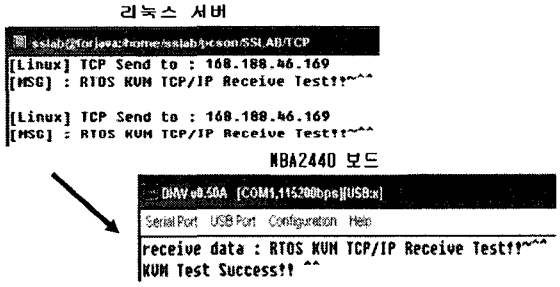
MBA2440 보드에서 리눅스 서버로의 문자열을 전송하면, 리눅스 서버에서는 문자열을 받고, 받은 문자열을 출력한다.

[그림 4-2]는 TCP 를 사용하여 MBA2440 보드에서 리눅스 서버로 문자열이 잘 전송됨을 확인할 수 있다.



[그림 4-2] 구현 시스템에서 리눅스 서버로 TCP 문자열 전송

또한 리눅스 서버에서 MBA2440 보드로 문자열을 전송하면, 보드는 문자열을 받아 출력한다. **[그림 4-3]**에서 보는 바와 같이 TCP 를 사용하여 리눅스 서버에서 MBA2440 보드로 문자열이 잘 전송됨을 확인할 수 있다.



[그림 4-3] 리눅스 서버에서 구현 시스템으로 TCP 문자열 전송

UDP 문자열 전송도 TCP 와 마찬가지로 잘 전송 됨을 확인할 수 있었다.

5. 결론 및 향후 연구과제

본 논문에서는 제한된 리소스를 사용하는 임베디드 시스템에 네트워크 통신이 가능하도록 UbiFOS™의 네트워크 시스템과 KVM 의 네트워크 API 부분과의 상호 연동을 위한 native 함수를 구현하였으며, 테스트 결과 TCP, UDP 송수신이 제대로 되는지 확인할 수 있었다.

향후 연구과제는 본 논문에서 구현한 TCP/UDP 통신의 유용성을 높이기 위해 MBA2440 보드에서 동작하는 telnet, FTP, SMTP 등을 구현하는 것이다.

참고 문헌

[1] 손필창, 정명조, 유용선, 최인범, " *The Analysis and Design of Java Network API for Embedded System*" 산업공학회, 2004

[2] sun Microsystems, Inc., K Native Interface Specification, 2002.

[3] Sun Microsystems, " *Mobile Information Device Profile(JSR-37)*" , 2000