임베디드 시스템을 위한 AODV기반 무선통신 테스트베드 구현

정왕부°, 정원도, 김종범, 박준성, 박찬흠, 서현곤, 김기형
영남대학교 컴퓨터 공학과
{kingrich°, yarang, kimjebi, post, shem, moses}@yumail.ac.kr, kkim@yu.ac.kr

AODV based Wireless Communication Testbed Implementation on Embedded Systems

Wang Boo Jeong°, Won Do Jung, Jong Beom Kim, Jun Sung Park, Chan Heum Park,
Hyun Gon Suh, Ki Hyung Kim
Dept. of Computer Engineering Yeungnam University

요 약

임베디드 시스템이란 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자 제어 시스템을 말한다. 임베디드 시스템은 다양한 분야에 적용될 수 있으며, 그 적용 분위는 날로 증가하고 있는 추세이다. 본 논문에서는 이동성을 가지는 임베디드 시스템을 구축하기 위한 테스트베드를 구현하였다. 본 논문에서 임베디드 시스템 사이의 무선 통신은 MANET의 가장 대표적인 요구기반 라우팅 프로토콜인 AODV를 이용하였고 개발한 테스트베드를 이용하여 MP3 스트리밍 서비스를 구현하였다.

1. 서 론

오늘날 우리 생활에 사용되는 각종 전자기기, 가전제품의 각종 제어장치에는 마이크로프로세서가 내장되어 있어 마이크로프로세서에 의하여 특정한 기능만을 수행하도록 프로그램이 되어있다. 마이크로프로세서에 의하여 이러한 특정한 기능만 수행하도록 만들어진 시스템을 임베디드 시스템(embedded system)이라 한다. 즉, 임베디드 시스템이란 마이크로프로세서가 내장되어 있어 동작하는 제어시스템을 말한다.

MANET(Mobile Ad Hoc Network)은 하부구조가 없이 이동 노드들로 구성된 네트워크를 말한다. 이들 사이의 통신은 자율적이고 즉흥적으로 이루어지는데, 이들의 활용도는 재난구조, 전쟁터 및 전시장 등과 같은 곳에서 지속적으로 증가하고 있고 공장 자동화 같은 전문분야의 임베디드 시스템에도 활용될 전망이다.

본 논문에서는 이동성이 있는 임베디드 시스템간의 무선 통신을 구현하기 위하여 AODV를 기반으로 한 테스트베드(Test bed)를 설계하고 구축하였다. 본 논문에서 구축한 테스트베드는 레드햇 리눅스 서버를 이용하여 노트북과 S3C2410 TK 보드를 사용하여 AODV를 기반으로 무선통신환경을 구축하여 실험하였다.

본 논문의 구성은 다음과 같다. 2장에서 본 연구와 관련된 기본 개념을 소개하고 3장에서는 테스트베드 시스템 개발 환경 및 구성방법을 소개한다. 4장에서 구현 및 실험 결과를 소개하고 끝으로 결론 및 향후 과제로 끝을 맺는다.

2. 기본개념

2.1 AODV(AD-HOC On-demand Distance Vector)

무선네트워크(wireless networks)는 기지국(base station)이나 AP(access point)와 같은 하부구조(infrastructure)를 갖는 네트워크와 하부구조가 없는(infrastructure less)네트워크로 분류된다. MANET은 하부구조 없이 이동노드들로 구성되는 대표적인 무선네트워크이다. MANET은 고정된 하부 구조가 없기 때문에 이동노드사이의 패킷 전달을 위해서 모든 이동노드들은 라우터 역할을 수행해야 한다. 따라서 이동 노드들은 경로설정 프로토콜을 실행함으로서 라우팅 테이블을 생성하고 이웃한 이동노드들과 경로 정보를 교환함으로 최신의 경로 정보를 유지해야 한다[1].

AODV(Ad Hoc On-Demand Distance Vector)는 1999년 C. Perkins가 제안한 것으로 MANET에서 대표적인 요구기반 (on demand)라우팅 프로토콜이다[2]. MANET의 모든 이동노드들은 데이터 전달이 있는 라우팅 경로 정보만으로 라우팅 테이블의 유지 및 관리를 하게 된다. 데이터 전달이 필요한 소스노드는 요구기반 방식으로 목적지 노드까지의 최단경로를 라우팅 경로 탐색 과정을 통해 찾아낸다.

2.2 무선랜(WLAN:Wireless Lan)

무선랜(WLAN)은 기존의 유선랜(LAN) 허브에서 클라이언트까지 유선 대신 전파나 빛을 이용하여 네트워크를 구축하는 방식이다. 무선랜은 기존의 유선랜의 역할을 하는 무선랜카드, 무선 환경에서의 LAN의 허브역할을 하는 AP 그리고 분산된 무선랜 세그먼트를 연결시켜주는 무선 브리지로 구성된다.

현재 무선랜 기술은 IEEE 작업그룹이 개발한 규격으로 802.11, 802.11a, 802.11b, 그리고 802.11g 등이 개발 되어 있다. 이 규격들은 경로 공유를 위해 모두 이더넷 프로토콜인 CSMA/CD를 사용한다. 802.11b 표준이 초당 약 11 Mbps의 속도를 제공하는데 비해 가장 최근에 승인된 표준인 802.11g는 비교적 짧은 거리에서지만 최고 54 Mbps까지의 빠른 전송속도를 제공한다. 802.11g도 802.11b와 같이 2.4 GHz 대역에서 동작

하므로, 두 규격 사이에는 서로 호환성이 있다[3].

2.3 임베디드 시스템과 임베디드 운영체제

임베디드 시스템이란 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자 제어 시스템을 말한다[4]. 현재 우리 생활에 쓰이는 각종 전자기기 , 가전제품의 제어장치는 단순히 회로로만 구성된 것이 아니라 마이크로프로세서가 내장되어 있고, 그 마이크로프로세서를 구동해 특정한 기능을 위한 프로그램이 내장되어 있다. 즉 임베디드 시스템이란 프로세서가 들어가서 동작하는 제어 시스템을 말하며, 보통 32bit 이하의 마이크로프로세서를 사용한 시스템을 그 범위를 한정한다. 일반적인 요구사항까지 처리할 수 있는 마이크로프로세서의 등장은 소프트웨어만 변경하는 방식으로 다양한 시스템을 개발할 수 있는 환경을 제공했다. 여기서 임베디드 시스템의 특징을 들어보면 32bit이하의 마이크로프로세서를 사용하며 일반 컴퓨터에 비해 제한된 메모리와 I/O를 가지고 있으며, 시스템 구동에 반드시 필요한 주변장치만 있으며, 다양한 입출력 방식이 있다.

임베디드 시스템의 소프트웨어는 특정 장비에 맞추어 제작되며, 다른 장비와 호환이 되지 않는다. 임베디드 시스템은 아주 열악한 환경에서 동작하기도 하며, 높은 신뢰성을 요구하기도 한다. 임베디드 시스템을 운영하기 위한 운영체제는 초기의 단순한 임베디드 시스템의 구성에서는 8/16bit 컨트롤러에 제한된 동작을 하도록 하는 소프트웨어가 탑재된 시스템이 전부였다. 하지만 고성능의 마이크로프로세서와 DSP(Digital Signal Processing)칩이 일반적으로 사용됨에 따라 사용 영역이 넓어지고 그에 따른 소프트웨어도 발달하게 되었다.

초기 임베디드 시스템은 단순해서 운영체제 없이 순차적인 프로그램으로 실행되었으며, 인터럽트가 발생할 경우에만 순차적인 프로그램에서 잠시 벗어났다. 하지만 최근의 임베디드 시스템은 시스템 자체가 커지고, 네트워크나 멀티미디어를 기본으로 장착하면서 다양한 기능이 추가되어 복잡해졌다. 더 이상 순차적인 프로그램 작성이 불가능하게 되면서 임베디드 시스템에서도 운영체제의 개념도 필요하게 되었고 임베디드 시스템의 특성상 실시간이라는 요소가 함께 도입되었다. 그러므로 임베디드 운영체제는 각 하드웨어에 맞는 최적화 코드를 사용한다. 본 논문의 Testbed에서는 레드햇 리눅스를 임베디드 운영체제로 사용하여 구현하였다[5].

3. 시스템 개발 환경

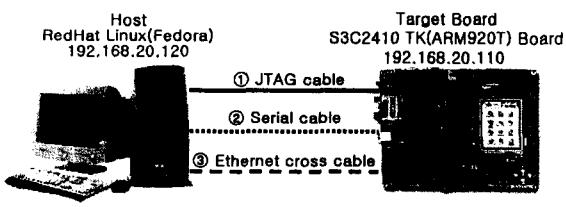

그림1 일반적인 교차개발 환경

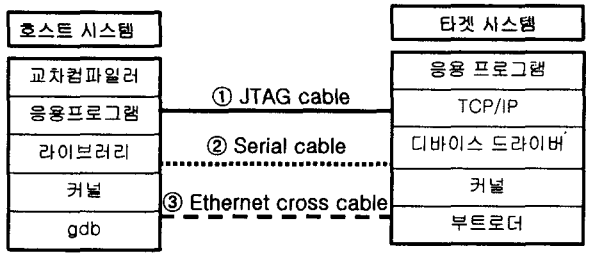

그림 2 교차 컴파일러(Cross Compiler)

임베디드 시스템 개발을 위해 생성된 타겟보드(target board)는 초기에 어떤 소프트웨어도 주기억장치에 없는 상태이며 전원을 켜도 아무런 동작을 할 수 없다. 그래서 하드웨어를 초기화하고 운영가능하게 하기위해서는 가장 먼저 타겟보드에서 실행 가능한 부트로더를 ROM이나 플래시 메모리에 장착하는 일이다. 초기화 되지 못한 하드웨어에 부트로더를 장착하는 일은 그림 1-①과 같이 JTAG(Joint Test Action Group) 인터페이스를 사용한다. JTAG은 원래 OCD(On Chip Debugging) 기능을 위해 개발되었으나 이 인터페이스를 통하여 외부의 프로그램이 마이크로 프로세서내의 레지스터 내용을 읽고 쓸 수 있으며 외부 메모리나 I/O디바이스의 접근이 가능하여 타겟보드에 부트로더 장착이 가능하다.

부트로더가 장착된 임베디드 시스템은 부트로더에 의해 그림 1-②과 같이 직렬 통신이 가능하게 되어 minicom등의 프로그램을 이용하여 호스트 컴퓨터에서 타겟보드에 접속하여 터미널 역할과 데이터 전송이 가능하다. 이때는 시리얼 통신으로 최소의 커널(Kernel) 이미지와 루트파일시스템(root file system)을 적재 시킬 수 있다. 이러한 과정이 끝나게 되면 임베디드 시스템에 리눅스가 실행되어져 Ethernet 장치를 초기화 하고 활성화 할 수 있어 그림 1-③을 통해 telnet, ftp, NFS (Network File System)와 같이 제어가 가능하다.

하드웨어가 처음 개발되었을 경우 메모리에는 어떠한 소프트웨어도 적재 되어있지 않아 타겟보드의 부트로더를 작성 하거나 기존 부트로더를 수정하고 컴파일 하여 타겟보드의 플래시 메모리나 롬에 저장시켜야 한다. 이러한 과정을 호스트 컴퓨터가 대신 처리할 때 이것을 교차 개발 환경(CDE : Cross Development Environment)이라 한다. 전형적인 개발 환경의 하드웨어 구성은 그림 1과 같다.

소프트웨어 구성은 그림2와 같은 형태를 취하고 있다. 개발자는 호스트 컴퓨터에서 타겟보드의 프로그래밍 코드와 컴파일을 거쳐 타겟보드에 적재하는데 이를 교차 컴파일러(Cross Compiler)라 한다. 교차 컴파일환경이란, 호스트 시스템에서 임베디드 시스템용 리눅스를 개발하기 위해 구축한 개발 환경을 말한다. 타겟보드와 호스트 컴퓨터는 서로 다른 프로세스를 사용하기 때문에 임베디드 시스템용 프로그램을 만들기 위해서는 타겟보드 내에서 직접 컴파일하거나 호스트에서 타겟보드 프로세스용으로 대신 컴파일 해야 한다. 하지만 자원의 제약이 큰 타겟보드 내에서는 컴파일 한다는 것은 바람직하지 않다. 타겟보드의 경우 메모리 및 저장 공간 제약 때문에 컴파일에 필요한 라이브러리까지 포팅하기 어려우며, CPU 속도가 느리기 때문에 개발 시간도 많이 소요된다. 이런 문제를 해결하기 위해

호스트 컴퓨터에서 그림 2와 같이 타겟보드에 적재된 프로그램을 대신 컴파일 한다. 표 1에서는 일반적인 교차 컴파일러의 환경 도구 모임이다. 표 1에서는 본 논문에 구현된 임베디드 시스템을 위한 ARM용 컴파일러와 라이브러리, 커널, 직렬 통신과 Ethernet 통신을 위한 프로그램도 포함되어있다.

표 1 Cross Complier 환경 도구 모음

Cross Compiler
cross-armv4l-binutils-2.10-3mz.i386.rpm
cross-armv4l-gcc-2.95.2-10mz.i386.rpm
cross-armv4l-gcc-c++-2.95.2-10mz.i386.rpm
cross-armv4l-gdb-5.2.1-1mz.i686.rpm
cross-armv4l-glibc-2.2.1s-3mz.i386.rpm
cross-armv4l-jpeg-6b-2mz.i386.rpm
cross-armv4l-jpeg-devel-6b-2mz.i386.rpm
cross-armv4l-kernel-headers-2.4.5_rmk7_np2-1mz.i386.rpm
cross-armv4l-libfloat-1.0-3mz.i386.rpm
cross-armv4l-zlibcross-armv4l-zlib-1.1.3-5mz.i386.rpm-1.1.3-5mz.i386.rpm
linuette_sdk_arm-1.3.1-1mz.noarch.rpm
linuette_sdk_x86-1.3.1-1mz.noarch.rpm
minicom-1.82.1-8mz.i386.rpm
qtE-custom-1.3.1-2mz.i586.rpm
qtE-custom-armv4l-1.3.1-2mz.i586.rpm
ztelnet-0.9.1-7mz.i386.rpm

4. 구현

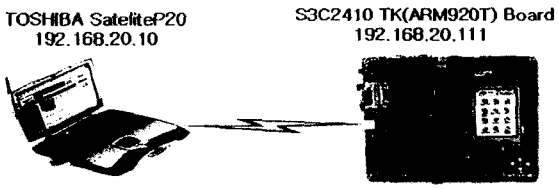

그림 3 구현 시스템

AODV 기반 무선 임베디드 시스템의 구축을 위한 테스트베드의 하드웨어 시스템은 그림 3과 같다. 그림 3에서와 같이 테스트베드는 도시바 노트북과 임베디드 mizi kit인 S3C2410 TK Board와 레드햇 리눅스 서버(Fedora, 커널 버젼 : 2.4.22-1.2115.nptl)로 호스트 컴퓨터를 구성하였다. 호스트 컴퓨터는 임베디드 보드의 상황을 모니터링하고, 교차 컴파일 환경, 노트북과 타겟보드사이에 AODV를 이용한 통신 환경을 만드는데 사용하였다.

시스템의 운영체제는 리눅스를 사용하였으며, 타겟보드의 모니터링 도구로는 호스트 컴퓨터에서 직렬 통신을 이용한 minicom을 사용하여 네트워크 환경에 영향을 최소화 하였다. 타겟보드에서는 AODV-UU를 수정하여 교차 컴파일하여 타겟보드에 적합한 모듈을 만들어 올렸으며, 노트북에는 AODV 통신을 위해서 AODV-UU의 모듈과 통신 데몬을 적재하였다. 타겟보드의 무선랜 지원을 위하여 커널과 타겟보드의 시스템 환경을 구축하였다. 그리고 임베디드 시스템의 타겟보드는 ROM 타입의 파일 시스템을 가지고 있어 새로운 작업을 할 때 마다 부트로더, 커널, 루트 파일 시스템을 새로운 이미지 파일로 만들어 적재해야 하는 불편함이 있어서 추가 하거나 갱신하기에 어려움이 많았다. 특히 타겟보드에서 AODV 모듈을 실행하기 위해서는 ROM 타입의 파일 시스템으로는 커널의 모듈에 대한 정보를 갱신 할 수 없는 문제점이 있다. 이 문제의 해결을 위해서는 읽고, 쓰기가 가능한 파일 시스템이 필요했으며, 해결방안으로 NFS(Network File System)를 이용하였다.

NFS 파일 시스템을 사용한다면 개발 호스트상의 파일이 리눅스 파일 시스템 위에서 접근이 가능하고 실행이 가능하다. 또한 램 디스크 상에서 올리기에 너무 큰 파일도 NFS상에서는 호스트의 기억용량에 의존하기 때문에 쉽게 처리할 수 있다. 즉, NFS는 RPC를 이용하여 리모트 호스트상의 파일을 유저가 마치 로컬 파일에 접근 하듯이 접근 할 수 있게 한다. 그 후, 호스트 컴퓨터에서 이미지 파일을 제작하여 타겟보드에 적재하여 구현하였다.

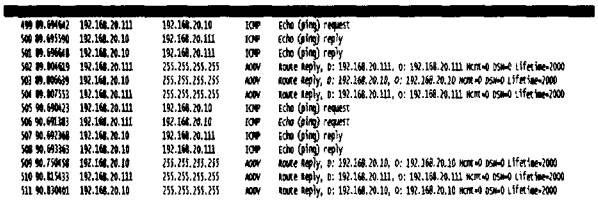

그림 4 라우팅 메시지

5. 결론 및 향후 연구 방향

임베디드 시스템에 리눅스를 이식하기위해서는 시스템의 특성에 대한 이해도가 중요하며, 이를 위해서는 임베디드 하드웨어에 대한 충분한 이해가 선행 되어야 한다. 본 논문에서는 최근에 빠르게 연구되고 있는 리눅스 기반의 임베디드 시스템에 모바일 라우팅 프로토콜인 AODV모듈을 탑재하여 무선랜으로 통신할 수 있도록 구축 하였다. 향후 이를 통하여 리눅스 임베디드 시스템에서 좀 더 많은 서비스 지원에 대해서 연구할 계획이다.

참고 문헌

[1] C. K. Toh, *"Ad Hoc Mobile Wireless Networks"*, Prentice Hall PTR, pp. 27-37, 2002.

[2] C. E. Perkins, *"Ad Hoc On-Demand Distance Vector (AODV) Routing,"* Internet Draft, IETF MANET Working Group, draft-ietf-manet-aodv-12.txt, November 2002.

[3] ANSI/IEEE Std. 802.11b, Part 11 : Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) Specifications, Amendment 2 : Higherspeed Physical Layer (PHY) extension in the2.4 GHz band, 2000.

[4] MIZI Research, Inc., "Linu@ SDK Install Guide for SMDK2410" pp. 4-9, March. 2003.

[5] http://fedora.redhat.com