Linux를 기반으로 한 Router 설계 및 구현

김성현*, 양진철*, 정기현*, 최경희**
*아주대 전자공학부
**아주대 정보 및 컴퓨터 공학부

Design and Implementation of a Router Based on Linux

Sung-Hyun Kim*, Jin-Chul Yang*, Ki-Hyun Chung*, Kyung-Hee Choi**
*Dept. of Electronic Engineering, A-jou Univ.
**Dept. of Information and Computer Engineering, A-jou Univ.

요 약

네트워크의 눈부신 발전 속에 정보 통신이 점차 대중화가 됨에 따라 통신망의 하부 구조가 계속 발전하고 있으며 수많은 네트워크 장비들이 개발되고 있다. 아주대 연구진은 여러 프로토콜들을 시험하고 높은 성능의 라우팅 알고리즘을 연구 개발하기 위한 네트워크에 매우 안정적인 Linux OS 가 탑재된 라우터를 설계 및 구현하였다. 본 논문에서는 라우터 시스템 하드웨어 구조를 설명하고 Linux 가 porting 되는 과정을 기술한다.

1. 서론

초고속 통신망을 요구하는 정보화 사회에서 컴퓨터 네트워크는 아주 급격한 발전을 이루었으며 더욱이 TCP/IP를 기반하는 인터넷의 발전은 놀랄만한 성장을 이룩했다. 인터넷의 발전은 많은 통신 사용자들의 다양한 서비스 요구에 따른 네트워크의 사용을 가져왔고 따라서 LAN과 WAN상에서의 많은 트래픽 초래 현상을 일으키고 있다. 이는 사용자의 요구를 수용하기 위해 네트워크 상에서 메시지를 최종 목적지까지 전달하기 위한 전송 경로에 대해 매우 안정적이고 효율적인 관리가 점점 필요로 되어지고 있다. 이를 해결하고 네트워크를 구축해 주는 장비가 라우터이다. 라우터는 네트워크 레벨에서 프로토콜 처리 능력을 갖는 LAN(WAN)간 접속장치로 논리적으로 또는 물리적으로 분리된 세그먼트를 하나의 세그먼트처럼 연결하여 데이터 패킷이 수신지로부터 목적지로 원활히 도달할 수 있게 하기 위해 사용하고자 하는 사용자의 요구에 의하여 발생된 개념을 구현한 장비이다. 라우터는 데이터링크층 이하가 다른 LAN들끼리도 중계가 가능하며 간선 LAN의 노드장치, 이기종 LAN의 접속, WAN의 접속, 경로의 이중화 등의 많은 목적으로 이용 가능하다. 여러 LAN을 상호 접속할 경우에 효율적인 경로를 설정하려는 라우팅 기능을 갖게 되며, 잘못된 패킷을 폐기할 수 있는 기능을 갖고 있으므로 장해 발생을 국지화할 수 있다.

본 시스템은 많은 트래픽 현상에서도 안정적이고 효율적인 하드웨어 구현과 요즘에 각광 받는 리눅스 OS(Operating System)를 라우터로 사용할 수 있도록 최적화된 LRP(Linux Router Project)를 porting함으로써 라우팅에 필요한 프로토콜 처리에 향상된 능력을 갖는 네트워크 장비를 설계 구현하고자 한다. 리눅스는 TCP/IP를 포함하고 있으므로, 인터넷 연동에 대한 확실한 해답이 될 수 있다. TCP/IP가 유닉스를 기반으로 개발 발전되었고, 리눅스는 유닉스를 기반으로 발전된 운영 체제이므로, 기타 인터넷 연동 솔루션에 비해 가장 발전된 해답을 제시할 수 있다. 이는 또한 임베디드 시스템에 PC환경의 OS인 리눅스를 porting한다는 것에 의미를 가질 수 있다. 리눅스가 비록 공개되어 있지만 실제 내장형 제품에 적용하기에는 많은 기술적 문제가 존재하고 손쉽게 적용되지 못하고 있는 것이 현실이기 때문이다. 본 논문에서는 시스템 구조와 메모리 관리, 디바이스 드라이버 설계에 대한 설명을 하였고 마지막으로 향후 연구과제를 언급하고자 한다.

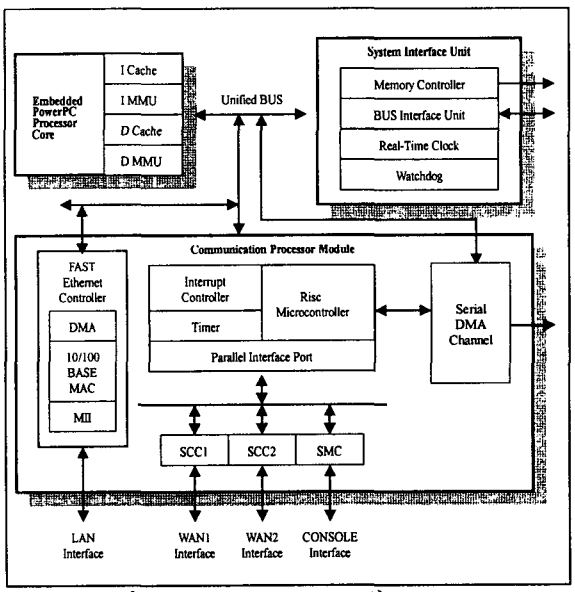
2. Processor Unit과 Memory

라우터에 내장된 CPU의 역할은 PC에서 사용하는 것과 같은 역할을 담당한다. 본 시스템에서는 모토로라 계열의 PowerPC인 MPC860T를 사용하였으며 이는 범용 Communication Processor이다. Processor unit은 PowerPC CPU Core, SIU(System Interface Unit), CPM(Communication Processor Module) 이렇게 세 unit으로 나뉜다.

CPU Core는 일반적인 Embedded PowerPC Core로서 4Kbyte Data, Instruction 캐쉬 구조를 포함하고 있다.

SIU(System Interface Unit)는 기본적으로 Memory controller 기능과 내부 버스와 외부 버스의 인터페이스를 담당할 뿐 아니라 Watchdog과 Real-Time Clock 기능을 제공한다.

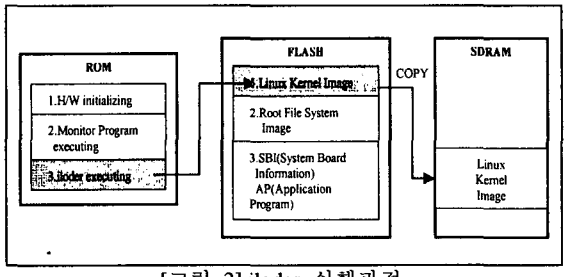
CPM(Communication Processor Module)은 MII(Media Independent Interface)에 10/100Mbps ethernet용 LAN port, 2개의 SCC(Serial Communication Controllers)를 WAN port, SMC(Serial Management Controller)에 콘솔용 UART, 이렇게 4개의 독립적인 통신 장치로 구성되어 데이터를 송수신하는 unit이다. CPM은 각각의 통신 프로토콜 스택을 지원하기 위해 별도의 RISC를 중심으로 구성되어 있다. 통신 처리에 있어서 PowerPC가 상위 layer의 기능을 최대화 하기위해 사용된다면 RISC는 수신된 데이터를 메모리로 옮기고, 실제 통신을 제어하는 하위 layer의 기능을 처리할 수 있다. 시리얼 DMA은 각각의 다른 port에서 송수신되는 네트워크 데이터를 Core 및 SIU에 연결시켜 준다. 그러나 100Mbps를 지원하는 MII은 다른 네트워크 디바이스보다 월등히 빠른 속도로 인해 패킷량이 더 많이 송수신 되기에 따로 DMA controller를 가지고 있다.



[그림 1] MPC860T Processor와 Device Unit

본 시스템에서의 외부 메모리는 ROM, FLASH, SDRAM 이렇게 세 가지가 쓰인다. ROM에는 하드웨어 초기화인 캐쉬 불가능한 부트 코드 영역과 리눅스 부팅 개시를 포함하고 있다. 또 라우터에 필요한 옵션을 설정하는 모니터 프로그램이 내장되어 있다. 리눅스가 부팅되는 시점부터 ROM은 사용되지 않으므로 캐쉬화 하지 않아도 시스템의 성능을 저하시키지 않는다. FLASH는 세 개의 영역으로 나뉜다. 전형적인 Embedded System에는 플로피와 하드디스크를 갖고 있지 않으므로 LKI(Linux Kernel Image)와 RFS(Root File System) 이미지를 저장할 수 있는 공간이 필요하다. 그 역할을 해 주는 것이 FLASH이다. 그러므로 위의 두 개의 이미지를 저장하는 각각의 공간과 나머지 라우터에 필요한 AP(Application Program)와 SBI(System Board Information)를 저장하는 곳 이렇게 세 부분으로 나뉜다. SBI는 System의 메모리 크기와 MAC, IP, Getway address, Netmask 등의 정보를 말한다. FLASH 각각의 이미지와 AP, SBI는 ROM의 모니터 프로그램에 의해 PC에서 시스템의 콘솔을 이용해 다운로딩 된다. SDRAM은 실제 리눅스가 움직이는 공간이며, Processor Unit중 SIU의 Memory controller에 의해 작동된다.

리눅스 부팅에 있어 메모리 구조는 다음과 같다. 부팅을 두 개의 분리된 로더를 지원하는 2단계로 쪼갤 수 있다. 하나는 LKI 로더로 칭하는 iloder이고 다른 하나는 리눅스 커널 로더인 kloder이다. iloder는 ROM 안에 내장되어 있다. 즉, 시스템 파워를 넣고 필수적인 하드웨어 초기화 및 모니터 프로그램이 가동 되면서 iloder가 실행된다. 이는 FLASH 첫 번째 영역에 있는 LKI를 SDRAM 어느 영역 적절한 위치에 옮기게 된다.



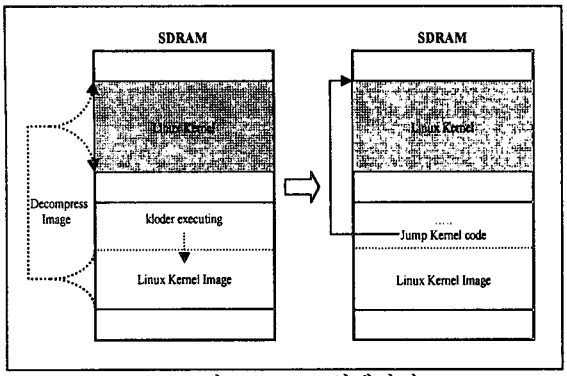
[그림 2] iloder 실행과정

kloder는 iloder가 끝나면 실행을 시작한다. kloder는 LKI 앞에 위치하며 LKI와 같이 FLASH에 저장된다.

우선 kloder는 타이머와 인터럽트, 트랩, 시그날처리 등의 더 많은 하드웨어를 초기화한다. 그리고 나서 LKI의 압축을 풀어 리눅스 커널 부팅을 위한 셋업을 한다. 마지막으로 kloder는 리눅스 개시 시퀀스를 시작하기 위해 커널 코드로 점프한다. 아직 SDRAM에 남아 있는 LKI는 이제 필요가 없으므로 지워져도 상관없다.

실제 커널이 부팅되면서 커널은 RFS를 찾는다. 이때 FLASH에 있는 RFS 이미지를 커널이 지정한 어느 영역에 압축을 풀게 되며 init이라는 초기 타스크를

실행하면서 리눅스 부팅은 끝나게 된다.

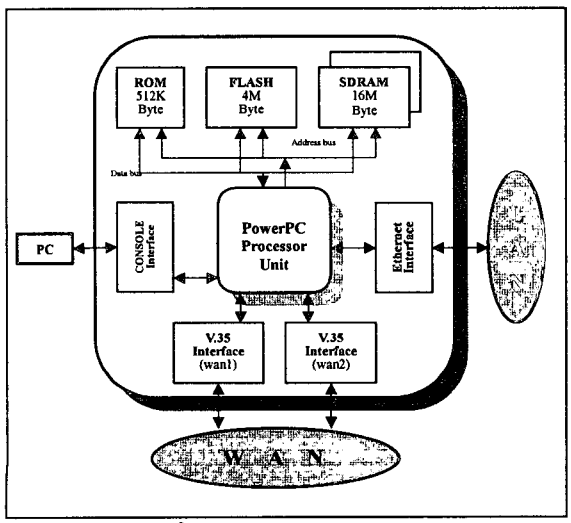


[그림 3] kloder 실행과정

가상 메모리 관리와 페이징 같은 메모리 관리를 위해서는 특정 메모리 크기와 범위를 설정하고 커널이 시작되는 동안 메모리를 재정렬하며, PowerPC BAT (Block Address Translation) 레지스터 쌍을 사용한다. 이는 물리적 어드레스와 가상 어드레스간의 메모리 맵핑을 위한 것이다.

3. Device driver Architecture

Processor, 메모리 등 극히 일부의 다른 자원을 제외하고는 모든 디바이스 제어 작업은 지정된 디바이스에 따른 코드에 의해서 수행된다. 이 코드를 디바이스 드라이버라고 부른다. 리눅스는 문자, 블록, 네트워크 인터페이스 이렇게 세 가지 종류가 있는데 본 시스템에서는 크게 저장에 관계된 디바이스와 네트워크에 관련된 디바이스 드라이버로 나눌 수 있다. 저장에 해당하는 FLASH 메모리는 마치 하드디스크나 플로피와 같은 역할을 하는 것으로 블록 단위로 접근하는 블록 디바이스이다. UART를 이용하는 콘솔용 SMC와 WAN port SCC1과 SCC2는 문자 디바이스, LAN port는 네트워크 디바이스로 구분한다. [그림 4]는 라우터 전체 블록도로 Processor unit과 메모리, 그 외 네트워크 디바이스를 보여주고 있다.



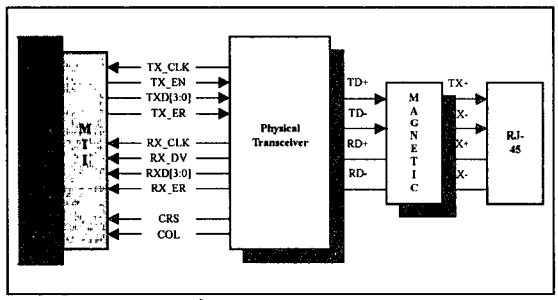
[그림 4] ROUTER Architecture

3.1 FLASH Device driver

앞서 설명하였듯이 세 개의 영역으로 나뉘어져 저장 매체로 사용되는 블록 디바이스이다. 이는 리눅스 상에서 sfd0, sfd1, sfd2라는 이름의 파티션으로 존재하며 각각 마운트하여 사용할 수 있다.

3.2 10/100Mbps Ethernet device

CPM내에 MII와 연결되며, 10/100Mbps ethernet controller로 physical transceiver와 마그네틱으로 구성된다. physical transceiver는 실질적인 물리계층으로 시그날링 드라이빙하는 PLS(Physical Layer Signaling) 기능과 코드 변환, 데이터 송수신, collision detection, 루프백을 지원하는 MAU(Media Attachment Unit)로 나뉜다. 마그네틱은 과전류 방지를 위한 절연 기능을 하는 부분이다.



[그림 5] MII DATA Interface

MII data는 TXD[0:3], RXD[0:3]으로 시리얼이 아닌 병렬로 nibble 데이터가 오고 간다. 그러나 10Mbps로 쓰일 때는 TXD0, RXD0만 사용된다. 실제 리눅스에서는 FEC(Fast Ethernet Controller)라는 디바이스 이름을 가지고 있으며 '#ifconfig fec' 명령으로 커널에 ethernet을 등록할 수 있다.

본 시스템에서 라우터 기능의 LAN 서비스는 다음을 만족한다.

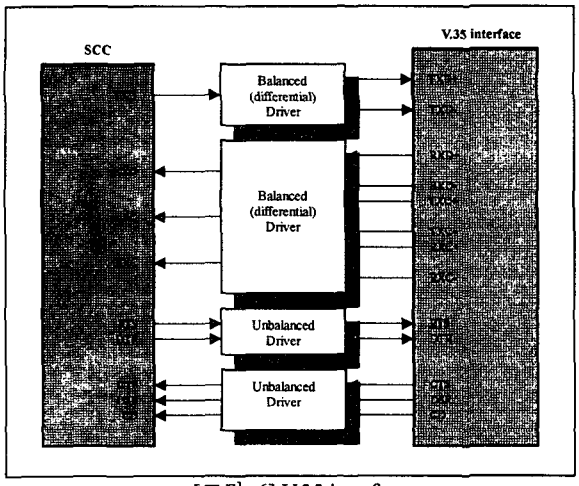
- IEEE802.3 ethernet을 지원하며 접속 형태는 10/100Base 포트를 지원한다.
- 라우터와 호스트 간에 telnet, rlogin 통신이 가능하다.
- TCP/IP, IPX를 지원한다.
- PING, ARP, RARP 등으로 ethernet 상태를 확인한다.

3.3 Console과 WAN interface

Processor Unit의 CPM내의 SMC와 연결된 콘솔은 RS-232C 시리얼 인터페이스를 사용하여 PC와 통신을 한다. 이 콘솔 port를 통해 라우터의 설정과 이미지 등을 FLASH에 저장 시킬 수 있다.

SCC에 연결된 WAN port는 V.35 인터페이스를 사용

하며 이는 외부적으로 CSU/DSU 모뎀에 연결된다. 물리적 신호 V.35 인터페이스는 데이터와 clock은 balanced(differential) 시그날로서 5V의 신호를 +12V와 -12V로 differential하게 전환해 주기 위해 전송할 때와 수신할 때 각기 다른 드라이버가 사용되며, control 시그날은 unbalanced이다. 이 밖에 신호의 과전압 방지를 위한 surge protection 디바이스도 연결되어 있다.



[그림 6] V.35 interface

본 시스템에서는 두 가지의 모드를 사용하고 있는데, PPP 모드는 WAN port를 이용하여 TCP/IP 통신을 하기 위한 것이고, HDLC 모드는 패킷 데이터 통신망에서 사용되는 프레임 계층의 전송 방식으로 CSU/DSU을 연결하여 최대 T1(1.544Mbps)/E1(2.048Mbps)의 속도로 인터넷 전용망에 연결한다.

리눅스 상에서 WAN port는 파일처럼 /dev/tty 파일 시스템 노드라는 방법으로 접근하여 open, close, read, write 등의 시스템 콜을 구현한다.

4. 결론 및 향후 연구과제

본 논문에서는 라우터의 네트워크 디바이스 설계 및 구현 방안을 제시하였다. CPM의 RISC 칩은 많은 네트워크 패킷 양을 빨리 처리 해 준다. 기존의 10Mbps ethernet controller를 벗어나 FEC를 지원하므로 LAN상에서의 많은 병목 현상들을 줄일 수 있을 뿐 아니라 리눅스의 운영 체제는 네트워크에 매우 안정적이고 효율적인 시스템을 구현하는데 향상된 능력을 가져다 준다.

본 시스템에서의 FEC는 MAC 어드레스의 빠른 처리를 요구한다. 이는 100Mbps의 성능에 밀접한 관계가 있다. 즉 상위 layer에서 자기의 어드레스가 아니면 제거 시키는 기능을 하드웨어적으로 외부 메모리에 어드레스만을 위한 캐쉬가 있다면 정상적인 네트워크 속도를 이룰 수 있을 것이다.

현재 콘솔로서의 LKI, AP 등의 다운로드와 모니터링은 라우터의 편리한 관리와 쉽게 제어를 발휘할 수 있도록 웹에서의 작업이 필요하다.

리눅스 그 자체로도 라우터의 기능을 사용할 수 있으나 표준 라우터 프로토콜인 RIP, OSPF 뿐만 아니라 다양한 라우팅 알고리즘과 프로토콜 지원은 네트워크 발전에 꼭 필요한 요소이다. 또한 NMS(Network Management System)을 포함한 강력한 보안 기능도 지원해야 하며 이는 별도의 방화벽 서버없이 패킷 access control 기능을 통해 방화벽을 구성해야 한다.

이 모든 것은 어떠한 네트워크 장애에도 항상 신뢰성 향상이 요구되어짐을 말하고 있다.

참고문헌

[1] Paul Mackerras. "Porting Linux to the Power Macintosh", 5th Annual Linux Expo, 1999
[2] Nichael Beck, Robert Magnus., "Linux Kernel internals", 2nd Ed. Addison-Wesley, pp428-439, 1998
[3] Remy Card, Eric Dumas, "The Linux Kernel Book", wiley, 1998
[4] Alessandro Rubini, "Linux Device Drivers", O'RELLY, pp3-5, 41-68, 301-306, 1998
[5] Motorola Inc. "MPC860 PowerQUICC ", 1998
[6] Tom Shaughnessy, Toby Vente, "Cisco A Beginner's Guide", Osborne/McGraw-Hill, 2000