碩士學位論文

2개의 망 지원이 가능한 리눅스 라우터의 구현

Implementation of Linux Router Supporting Two Network Systems

仁川大學校 情報通信大學院

情 報 通 信 專 攻

金 錫 星

2001年 6 月 日

碩士學位論文

2개의 망 지원이 가능한 리눅스 라우터의 구현

Implementation of Linux Router Supporting Two Network Systems

指導敎授 趙 仲 彙

이 論文을 碩士學位 論文으로 提出함

2001年 6 月 日

仁川大學校 情報通信大學院

情 報 通 信 專 攻

金 錫 星

이 論文을 碩士學位 論文으로 認準함

2001年 6 月 日

主 審 

副 審 

副 審 

仁川大學校 情報通信大學院

목 차

표 목 차

그 림 목 차

국 문 요 약

본 논문에서는 개발비용이 거의 없는 리눅스 운영체제를 사용하여 새로운 리눅스 라우터를 구현하였다.

기존의 라우터는 RTOS를 구입하여 사용하기 때문에 개발비용의 부담으로 장비의 고가격화와 전용회선 1개의 망만을 사용하기 때문에 발생되는 불안정성과 속도개선을 위한 회선비용 부담의 문제점이 있다.

새로운 리눅스 라우터는 하드웨어적으로 2개의 이더넷 포트와 2개의 E1 포트로 설계하여 E1 포트는 전용회선 망으로 접속하고, 동시에 1개의 이더넷 포트는 ADSL/CABLE 망에 접속할 수 있게 하였고 또한 소프트웨어적으로는 QoS와 로드밸런싱 기능을 추가하여 RTOS 라우터가 가지고 있는 기능 이외에 2개의 망을 동시에 지원한다.

본 논문에서 구현한 리눅스 라우터는 개발단계에서 구입비용이 없는 리눅스 운영체제를 사용함으로서 개발비용이 RTOS와 현저하게 비교되었고 성능시험에서도 다른 RTOS 라우터에 비해 2~3배의 속도가 향상되었으며 2개의 망을 사용함으로서 안정성이 보장되어 그 효율성이 검증되었다.

또한 제품화에서도 기존 라우터에 비해 2~3배의 낮은 가격을 유지할 수 있어 가격적인 경쟁력이 입증되었다.

Ⅰ. 서 론

인터넷의 발전으로 세계가 하나의 데이터망으로 형성됨에 따라 여러 가지 요구 조건들이 생겨나게 되는데 그것은 동일한 IP상에서 이루어 질 수 있는 기술의 발전이다. 그 중에서 데이터의 안정성 및 품질의 확보는 가장 중요한 사항 중에 하나인데, 라우터[1]등 LAN[2]장비의 고성능화에 따라 인터넷에서 데이터의 안정성을 요구하는 QoS(Quality of Service)[3] 기술이 주목받고 있는 가운데 세계적으로 볼 때 QOS의 연구개발은 필연적이다.

이러한 QoS는 1998년부터 조금씩 세계 네트웍 시장에서 제품과 솔루션이 제안[4]되기 시작하였으며, 인터넷의 발전으로 형성되고 있는 다양한 요구의 근본적인 원인은 1) 사용자의 네트워크 구축비용의 절감. 2) 인터넷기반으로 한 데이터의 신뢰성 구축. 3) 통신비용의 절감이 그 원인이다.

대부분의 기업들은 라우터 장비의 개발을 위해 운영체제를 상용 RTOS(Real Time OS)[5]를 구입하여 개발의 목적으로 사용하는 경우가 대부분인데 개발비용, 개발시간, 기능구현에 있어서 경제적, 시간적인 제약이 부담으로 작용되면서 신뢰성과 안정성 및 다양한 기능을 구현하기가 매우 어렵다. 뿐만 아니라 이러한 전용회선 라우터를 사용하는 사용자에게는 속도개선을 위해 많은 회선비용의 부담도 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 구입비용이 전혀 없는 공개된 운영체제인 리눅스의 채택과 전용회선 망과 ADSL/CABLE 망을 동시에 사용할 수 있도록 하드웨어적으로는 1개의 이더넷 포트를 더 추가하는 것과 함께 소프트웨어적으로는 QoS 기능과 로드밸런싱 기능을 추가함으로서 개발비용의 부담해소와 2개의 망을 동시에 사용함으로서 안정성과 속도의 향상

그리고 제품화에 따른 장비의 저가격화를 그 연구목적으로 하며, RTOS 라우터와 2개의 망 지원이 가능한 리눅스 라우터를 여러 측면에서 비교하고 성능시험을 통하여 기존 RTOS 라우터보다 경제적, 기능적, 안정적, 경쟁력 등에서 효율성이 높음을 검증한다.

본 논문의 2장에서는 RTOS 라우터와 리눅스 라우터 를 이해, 비교 및 검토하고, 3장에서는 2개의 망 지원이 가능한 리눅스 라우터의 개발환경과 구현자료[5]등을 정리하고, 4장에서는 성능시험을 통하여 RTOS 라우터와의 차이점과 효율성을 비교한다. 5장에서는 본 논문에서 구현한 리눅스 라우터의 향후 연구방향을 제시하고 결론을 맺는다.

Ⅱ. RTOS 라우터와 리눅스 라우터의 이해

대부분의 라우터는 RTOS를 사용하여 라우팅 기능을 수행한다. 그러나 리눅스 라우터는 일반적으로 사용되는 RTOS를 사용하지 않고 공개된 운영체제인 리눅스를 사용하여 라우팅을 수행하는 라우터이다.

이러한 라우팅을 수행하는 RTOS 라우터와 리눅스 라우터의 개요와 특징들을 살펴보면 다음과 같다.

1. RTOS 라우터의 개요

라우터는 게이트웨이 역할을 하는 일반적인 하드웨어 장비를 말한다. 게이트웨이는 두개 이상의 물리적으로 구성되어 있는 네트워크상에서 서로의 네트워크를 연결해 주는 중계 호스트를 말한다. 이런 중계 호스트 역할을 하는 것이 바로 전용라우터 장비이다. 기술적인 용어의 라우터는 논리적으로 구성된 네트워크 사이에서 패킷을 전송하는 장치를 말한다. 그림 2.1은 네트워크 망에서의 라우터의 역할을 보여준다.

라우터는 OSI 7계층의 네트워크 계층에 해당되며 패킷에 있는 주소를 해석하여 패킷을 전달할 경로를 결정하고 목적지로 패킷을 전달하는 지능적인 장치이다. 이 과정에서 라우터는 LAN 및 WAN으로부터 입력된 TCP/IP, DECNET,IPX[1] 등 서로 다른 패킷을 소프트웨어에 의하여 해석하므로 라우터의 CPU는 부하가 많이 걸린다.

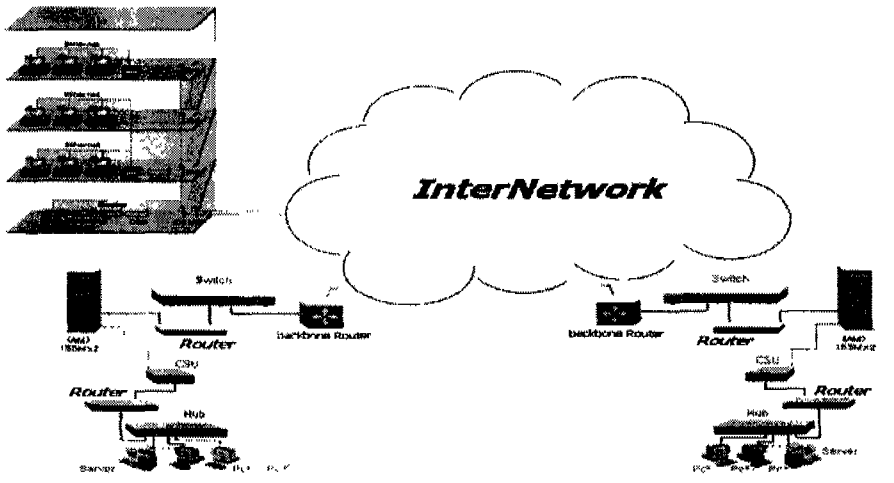

Figure 2.1 Internet network

라우터의 핵심은 소프트웨어로 프로토콜을 해석한다는데 있다. 다중 프로토콜[1]로 구성된 환경에서 복수개의 프로토콜을 소프트웨어에 의하여 해석하므로 하드웨어에 의하여 기능을 처리하는 허브, 스위칭 허브에 비하여 속도가 늦으며, 처리해야 할 프로토콜의 종류가 많을수록 CPU의 부하가 증가되므로 라우터의 성능은 떨어질 수밖에 없다. 라우터에는 LAN과 WAN[1]의 트래픽이 집중하므로 네트워크의 성능, 신뢰성, 확장성 등 모든 측면에 영향을 미치는 핵심적인 장비이다.

1) 성능

라우터의 성능은 여과와 전송속도를 기준으로 하여 PPS(Packet Per Second)[1]또는 FPS(Frame Per Second)[1]로 표시하는 것이 일반적이다.

실제로 사용하고 있는 네트워크 분석을 통하여 확인하면 대부분의 라우터에서는 CPU 부하가 대부분 30%이내의 안정적인 수준을 유지하고 있다. 즉,

대개의 라우터는 일상적인 트래픽을 감당하기에 충분한 성능을 가지고 있다.

2) 신뢰성

네트워크에서 라우터는 두 네트워크간의 데이터 전송경로를 제공하므로 라우터의 고장은 심각한 문제를 야기한다. 따라서 제품의 신뢰성은 정상적인 기능 못지 않게 중요하다. 신뢰성에 대해서는 평균고장시간(MTBF)[1], 평균 수리지연시간(MTTR)[1], 시스템 재구성 가능성, 소프트웨어 업그레이드 및 설치의 용이함 등과 같은 지표를 활용하여 제품의 평가기준으로 삼는다. 현재 국내외의 업체에서 개발하여 공급하고 있는 대부분의 라우터는 하드웨어 기술의 발달로 최상의 신뢰성을 보이고 있기 때문에 평균고장시간, 평균 수리 지연시간과 같은 장애의 측면보다는 소프트웨어 및 유지보수 측면이 더 중요하다.

2. RTOS 라우터의 특징

라우팅은 출발지로부터 목적지까지 인터넷을 통해 정보를 전송한다. 이런 작업에서는 적어도 한번 이상 중간 접속장비를 거친다. 라우팅은 종종 같은 목적을 달성하는 것처럼 보이는 브릿징과 비교된다.

둘 사이의 가장 큰 차이점인 브릿징은 OSI 참고모델의 Layer2(Link-Layer)[1]에서 동작하는 반면에 라우팅은 Layer 3에서 작동한다. 이러한 차이점은 라우팅과 브릿징이 출발지에서 목적지까지 정보를 전달하기 위하여, 다른 정보를 사용한다. 라우팅과 브릿징은 서로 다른 방식으로 목적을 달성하고 실제

로 다양한 종류의 라우팅과 브릿징 방식이 있다. 라우팅이 일반적으로 혹은 상업적으로 각광을 받기 시작한 것은 1980년대 중반이었다.

1) 경로결정[1]

경로길이와 같은 측정방법으로 목적지까지의 최적경로를 결정하는 라우팅 알고리즘이 표준적인 척도이다.

경로결정을 위해 라우팅 알고리즘은 경로에 대한 정보를 포함하는 라우팅 테이블을 유지하고, 초기화한다. 라우팅 정보는 사용되는 라우팅 알고리즘에 따라 다양하다. 라우팅 알고리즘은 다양한 정보로 라우팅 테이블을 채운다. 목적지/다음 도달점의 조합은 라우터가 패킷을 특정 목적지에 전달하기 위하여 그 목적지로 가는 경로에 있는 특정 라우터로 그 패킷을 보내야 한다는 사실을 알려준다.

To reach network:	Send to:
27	Node A
57	Node B
17	Node C
24	Node A
52	Node A
16	Node B
26	Node A
. . .	. . .

G1233a

Figure 2.2 Destination/next hop routing table

라우터가 패킷을 받으면, 라우터는 목적지주소를 체크하고 이 주소를 다음 도달점과 연관지으려고 시도한다. 그림 2.2는 목적지와 다음도달점의 라우팅 테이블의 예이다.

라우팅 테이블은 어떤 경로에 대한 적합성을 나타내는 정보를 가질 수 있다. 라우터는 최적의 경로를 결정하기 위하여 측정값들을 비교한다. 측정값들은 사용되어지는 라우팅 알고리즘의 디자인에 따라 다르다.

라우터들은 서로 라우팅 테이블을 교환하거나, 통신하기 위하여 다양한 메시지를 전송한다. 라우팅 업데이트 메시지가 이러한 메시지중의 하나이다. 라우팅 업데이트는 일반적으로 라우팅 테이블의 전체나 일부로 구성되어 있다. 전체 라우터로부터 들어온 라우팅 업데이트를 분석하여, 라우터는 네트워크 형태에 대한 구체적 그림을 그릴 수 있다. 연결상태도 라우터간 메시지들 중에 하나이다.

연결상태는 다른 라우터들에게 보낸 라우터의 연결상태를 알린다. 연결정보 또한 네트워크 형태에 대한 완전한 그림을 그리는데 사용할 수 있다. 네트워크의 형태가 한번 그려지고 나면, 라우터는 네트워크 목적지에 도달하는 최적의 경로를 결정한다.

2) 분배기능

분배기능 알고리즘[1]은 비교적 간단하고, 기본적으로 대부분의 라우팅 프로토콜[1]에서 같다.

대부분의 경우, 한 호스트는 다른 호스트에게 어떤 패킷을 보내야 한다는 것을 결정한다. 어떤 수단에 의해 라우터의 주소를 획득하면, 출발지 호스트는 목적지 호스트의 네트워크 계층의 프로토콜 주소가 아닌 라우터의 MAC-layer 주소가 붙여진 패킷을 전송한다. 패킷의 목적지 프로토콜 주소를 체크하여, 라우터는 그 패킷을 다음 도달점으로 어떻게 어떤 포트로 보내야 하는지를 아는지 혹은 모르는지를 결정한다.

만약 라우터가 그 패킷을 어떻게 보내야 하는지를 모를 경우, 일반적으로

라우터는 그 패킷을 버린다. 혹은 라우터가 그 패킷을 어떻게 보내야 하는지를 알 경우, 라우터는 패킷의 목적지 MAC주소를 다음 도달점의 MAC주소로 변환하여 전송한다. 그 다음 도달점은 최종의 목적지일 수도 있고, 아닐 수도 있다. 그렇지 않다면 다음 도달점은 같은 과정의 분배처리를 하는 다른 라우터이다. 패킷이 인터넷을 통해 옮겨질 때, 그 패킷의 MAC주소는 변하지만 프로토콜 주소는 변하지 않는다.

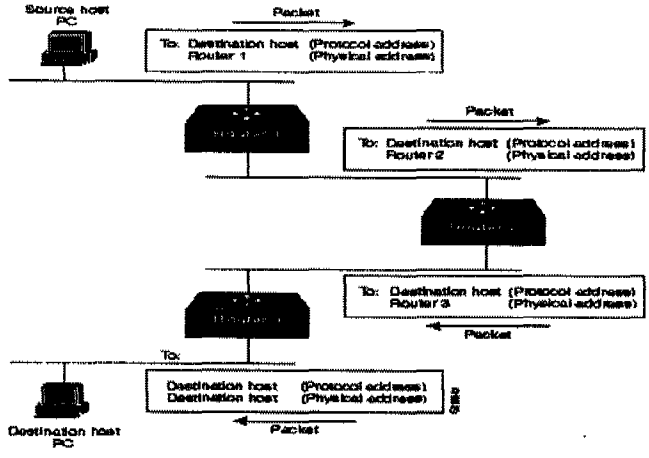

Figure 2.3 Switching process

이 과정은 그림 2.3에서와 같이 출발지와 목적지 시스템간에 분배처리에 대하여 설명한다. International Organization for Standardization(ISO) 는 이 과정을 설명하는데 유용한 계층적인 용어를 개발하였다. 이 용어에 의하면 서브네트워크간에 패킷을 전달할 수 없는 네트워크 종류를 EndSystems(ESs)[1]이라 하며, 패킷을 전달할 수 있는 네트워크 종류를 Intermediate Systems(ISs)라고 한다. ISs는 라우팅 도메인 내에서 통신할 수 있는 장비와 라우팅 도메인 내에서나 라우팅 도메인간에 통신할 수 있는 장비로 구분한다. 라우팅 도메인은 일반적으로 일반적인 관리의 권한 하에 있는 인터넷의 한 부분인데,

이는 특별한 구성으로 정의된다.

실제 프로토콜에서 라우팅 도메인은 라우팅 지역들로 나누어질 수 있으며, 내부 도메인 라우팅 프로토콜은 지역사이 혹은 지역내의 분배만 사용한다.

3) 라우팅 알고리즘

라우팅 알고리즘[1]은 유연성을 가지고 있어야 하며 다양한 네트워크의 환경 변화에 빠르고 정확하게 적응해야 한다.

어떤 네트워크의 한 부분이 불통되었다면 이 문제를 발견한 순간, 대부분의 라우팅 알고리즘은 빠르게 보통 이 한 부분을 이용하던 경로를 차선의 경로로 전환한다. 라우팅 알고리즘은 네트워크 대역폭이나 라우터 대기하는 패킷 크기, 네트워크 지연, 또는 다른 변수의 변화에 적응하도록 프로그램 된다.

라우팅 알고리즘은 몇 가지 중요한 특징에 의해 달라진다. 우선 알고리즘 디자인너의 특별한 목표가 결과적으로 라우팅 프로토콜의 작동에 영향을 준다. 또한 여러 종류의 라우팅 알고리즘이 있으며, 각 알고리즘은 네트워크와 라우터에 다른 영향을 준다.

라우팅 알고리즘은 최적의 경로를 계산하는데 영향을 주는 다양한 측정방법을 사용하며 몇 가지 타입으로 분류된다.

① 정적 또는 동적

② 단독경로 또는 다중경로

③ 비계층적 또는 계층적

④ 호스트의 처리 또는 라우터의 처리

① 정적 또는 동적

정적 라우팅 알고리즘은 정적 라우팅 테이블 연결이 라우팅이 시작되기 전

에 네트워크 관리자에 의해 우선 만들어진다. 이 테이블은 네트워크 관리자가 그 값들을 바꾸어 주기 전에는 바뀌지 않는다. 정적 라우팅 알고리즘은 네트워크 트래픽이 비교적 예견되어지거나, 네트워크 디자인이 비교적 간단한 환경에서 잘 작동한다. 정적 라우팅 시스템은 네트워크 환경변화에 반응할 수 없기 때문에, 거대한 네트워크 환경에는 부적합하다.

1990년대에 출현한 대부분의 라우팅 알고리즘은 동적이다. 동적 라우팅 알고리즘은 거의 실시간으로 네트워크 환경변화에 적응한다. 또한 라우팅 업데이트 메세지를 분석해서 작동한다. 그 메시지가 네트워크 환경 변화를 요구하면, 라우팅 소프트웨어는 새로운 경로를 계산하고 라우팅 업데이트 메세지를 전송한다. 이러한 메시지는 네트워크에 퍼지고, 다른 라우터들이 라우팅 알고리즘을 다시 작동시켜, 라우팅 테이블을 동시에 변화시킨다.

어떤 경우에 마지막에 의지하는 라우터가 지정되면 이 라우터는 모든 언라우터블 패킷에 대한 보관창고로 작동을 하며 어떤 방법으로든 모든 메시지를 처리한다. 이 경우에는 정적 라우팅이 일반적으로 사용된다.

② 단독경로 또는 다중경로

몇몇의 잘 설계된 라우팅 프로토콜은 몇몇 목적지에 대해 다중의 경로를 제공한다.

이런 다중경로 알고리즘은 단독경로 알고리즘에서는 불가능한 다중의 라인을 통해 트래픽의 분산을 허용한다.

③ 비계층적 또는 계층적

라우팅 알고리즘은 비계층적과 계층적으로 분류된다. 비계층적 라우팅 시스템에서 라우터는 모두 다른 라우터와 통신한다.

계층적 라우팅 시스템에서는 어떤 라우터는 라우팅 백본을 형성한다. 백본

을 형성하지 않은 라우터에 발생한 패킷은 백본 라우터로 전송하며, 백본 라우터는 그 패킷이 목적지가 있는 지역에 도달할 수 있도록 백본을 통해 전송한다. 목적지가 있는 지역의 백본 라우터는 이 패킷을 하나 이상의 백본이 아닌 라우터를 거쳐 최종 목적지로 전송한다.

라우팅 시스템은 도메인이라고 불리는 매듭들의 논리적인 그룹을 지정한다. 이는 자율적인 시스템 또는 지역이라고 한다. 계층적 시스템에서 도메인내의 어떤 라우터들은 다른 도메인의 라우터와 통신하는 반면, 어떤 라우터들은 동일한 도메인내의 라우터들끼리만 통신을 한다. 큰 네트워크에서는 추가적인 계층적 레벨이 존재한다. 계층구조의 최고 레벨에 위치한 라우터는 라우팅 백본을 형성한다.

계층적 라우팅의 주요한 장점은 시스템이 대부분 회사의 조직체계를 따름으로써, 그들의 트래픽 패턴을 잘 지원한다는 것이다. 대부분의 네트워크 통신은 작은 회사그룹에서 발생한다. 내부도메인 라우터만이 그 도메인 내의 라우터들에 대해서 알고 있기 때문에 라우팅 알고리즘은 간단하다. 결국 사용되는 라우팅 알고리즘에 의해 라우팅 업데이트 트래픽도 감소된다.

④ 호스트의 처리 또는 라우터의 처리

라우팅 알고리즘은 출발지의 마지막 매듭이 전체의 라우팅 경로를 결정하도록 하고 이에 따르는데 이를 출발지 라우팅이라고 한다.

출발지 라우팅 시스템에서 라우터는 다음전송지로 전송에만 동작하고, 다음 장비에 대해서는 상관하지 않는다. 다른 알고리즘들은 호스트의 경로에 대하여 모른다. 이러한 알고리즘에서는 라우터가 그들의 계산 결과에 따라 인터넷을 통해 경로를 결정한다.

결국 전자의 방식에서는 호스트가 경로계산을 하고, 후자의 방식에서는 라우터가 경로 계산을 한다. 호스트의 처리와 라우터의 처리 사이의 차이점은

최적의 경로인가인데 트래픽오버헤드인가의 차이이다. 호스트의 처리는 패킷을 실제로 전송하기 전에 목적지에 도달하는 모든 가능경로를 찾기 때문에 좀더 좋은 경로를 선택하기 쉽다. 또한 이 알고리즘에서는 특정 시스템에 대한 최적의 정의에 기초하여 가장 좋은 경로를 선택한다.

3. 리눅스 라우터의 개요 및 특징

1) 리눅스 라우터의 개요

라우터는 임베디드 시스템[7]이다. 임베디드 시스템이란 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자제어 시스템을 말하며, 필요에 따라서는 일부 기계가 포함된다.

예를 들어 PC, TV, 냉장고, 세탁기, 전자레인지 같은 전자 가전제품 외에 핸드폰, PDA, 그리고 사이버 아파트의 홈 관리시스템, 홈 네트워크 게이트웨이, 그 밖의 교통 관리시스템, 주차 관리시스템, 홈 관리시스템, 엘리베이터 시스템, 현금 지급기, 항공 관제시스템, 우주선 제어장치, 군사용 제어장치 등이 있다.

이러한 임베디드 시스템을 이용한 네트워크 장비개발의 경우 리눅스는 개발 시간단축에 유리하다. 리눅스는 거의 모든 네트워크 기능을 제공한다. 네트워크 장비의 대부분의 응용 소프트웨어는 TCP/IP stack을 사용한다. 리눅스에서 이러한 대부분의 네트워크 관련 소프트웨어를 지원한다.

새로운 네트워크 기능을 네트워크 장비에 추가할 경우, RTOS를 이용하게 되면 먼저 고가의 프로그램 소스를 사거나 직접 프로그래밍 하여야 한다. 그리고 개발장비에서 직접 시험을 해야 한다. 그러나 리눅스는 먼저 안정된 PC

위에서 시험 할 수 있다. TCP/IP stack은 임베디드 시스템과 PC에 적용되기 때문에 PC에서 모든 시험을 하고 임베디드 시스템에 적용한다.

2) 리눅스 라우터의 특징

① 커널 컴파일

리눅스 커널 컴파일은 압축된 커널 이미지를 만들 때 PC와는 다르다. PC에서는 make bzImage라고 명령을 주지만 리눅스에서는 make zImage.initrd라는 명령어를 준다.

차이점은 bzImage는 커널만을 압축한 것이고 zImage.inird는 압축된 커널을 풀어주는 프로그램과 압축된 커널 그리고 압축된 RAM disk image를 포함한다. 대부분의 임베디드 시스템은 HDD와 같은 저장 장치를 가지고 있지 않기 때문에 RAM disk를 사용한다.

리눅스 커널을 컴파일하기 전에 커널의 환경설정을 한다. 임베디드 시스템에 맞는 커널의 환경설정은 PC의 환경설정과 다르다. 부록 자료 1은 QoS 라우터에 해당하는 환경설정파일 내용이다.

② 기본동작

그림 2.4는 리눅스를 올릴 수 있는 최소한의 하드웨어의 구조이다.

그림 2.5는 ROM의 내부 내용이다. Start.s를 수행하면 하드웨어는 DRAM과 UART를 읽고 쓴다. Loader.c는 ROM에 있는 linux soruce부터 생성된 이미지를 ROM에서 읽어서 DRAM에 복사한다.

head.s는 serial.c와 misc.c를 수행하기 위한 준비작업인 stack pointer와 특정 메모리영역의 정의를 한다. tty.c는 UART를 초기화 하고 Misc.c는 압축

된 커널을 푼다. head.s,tty.c는 하드웨어 구조에 따라 다르게 작성된다.

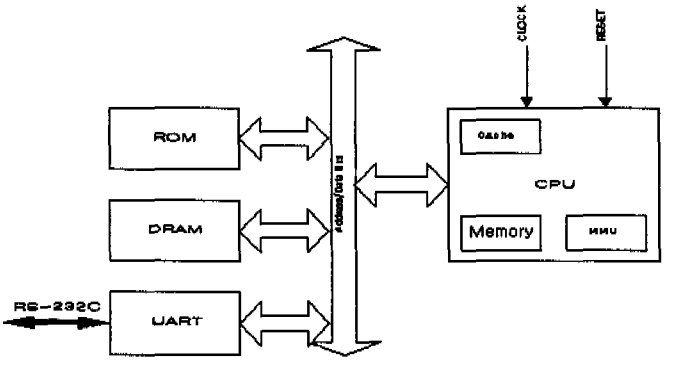

Figure 2.4 Basic structure of hardware

리눅스 커널소스(/usr/src/linux)에서 arch/ppc에 xxboot라는 디렉터리가 있다. 해당 디렉터리의 서브 디렉터리에는 일반적으로 널리 알려진 상용 하드웨어에 대한 head.s,tty.c,m isc.c의 소스가 있다. 이를 참고하여 해당 임베디드 시스템에 맞는 init.s,tty.cmisc.c을 만든다.

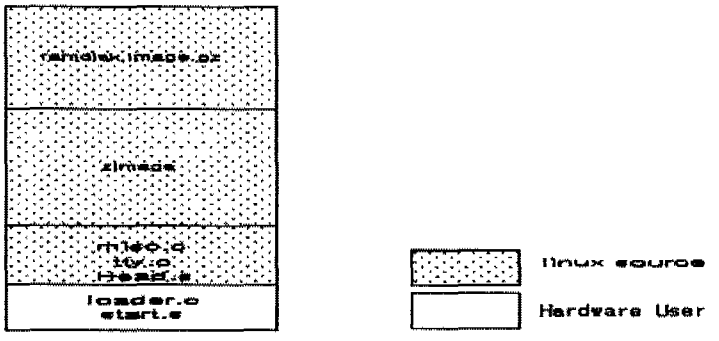

Figure 2.5 Having ROM of data structure

새로운 리눅스 라우터 개발에는 mbxboot를 사용하였다. 커널의 압축된 이

미지가 풀리고 커널을 수행하면 그림 2.4에서와 같이 RS-232C통해서 내용이 전달된다. 커널에서는 CPU와 UART을 초기화하고 RAM disk를 생성한 후 압축된 RAM disk 이미지를 풀고 이를 RAM disk에 복사한다.

4. RTOS 라우터와 리눅스 라우터의 비교

리눅스의 장점은 네트워크 구현이 뛰어나고 POSIX,HTTP등의 업계표준을 지원한다. 그리고 초기 구입비용과 라이센스 비용부담이 없으며, 소스가 공개되어 있어 손쉽게 하드웨어를 최적화한다.

리눅스 운영체제와 RTOS인 VxWORKS의 경제적인 비용에 대하여 표 2.1에서 비교하였다.

Table 2.1 Linux OS & real time OS(VxWORKS)

	Linux OS	Real Time OS(VxWORKS)
Base OS	$0	$50000
Device Driver & Software	$0	$100000
Total	$0	$150000

리눅스는 여러 CPU를 지원하기 때문에 다른 기종간의 이식도 수월하다. 또한 일반적인 환경에서 요구되는 메모리 및 기타 제약사항에 대한 만족할 만한 해결책이 있다. 기존에 산업전반에 사용되는 RTOS에 비해 저가의 비용으로도 적용될 수 있고, 개발 기간 단축은 물론 경쟁력에서도 우위에 있다. 현

재 관련분야에서 상용 RTOS 도입에 따른 엄청난 로열티가 지급되고 있다. 어떠한 운영체제가 탑재되든지 간에 핵심이 되는 기술은 하드웨어의 설계기술이다.

기존의 대표적인 기업인 Cisco의 router 2501제품과 리눅스 라우터를 하드웨어적, 소프트웨어적인 부분과 가격적인 부분에서 표 2.2에서와 같이 비교하였다.

Table 2.2 Linux router & Cisco router 2501

	SPEC		Linux router	Cisco router 2501
Hardware	CPU		MPC860ENZ50C1 (50MHz)	MC68360 (20MHz)
	Memory (Byte)	DRAM	16M(Up to 32M)	4M(Up to 16M)
		Flash	4M(Up to 8M)	8M
		Boot	512K	
	Ethernet port		10Mbps X 2	10Mbps X 1
	WAN port		E1 X 2	E1 X 2
OS	OS		Linux(2.2.14)	Cisco IOS
Protocol	Static		Possibility	Possibility
	RIP v1/2		Possibility	Possibility
	OSPF v1/2		Possibility	Possibility
	BGP		Possibility	Possibility
	PPP		Possibility	Possibility
	HDLC		Possibility	Possibility
Software	NAT		Possibility	Possibility
	PAT		Possibility	Possibility
	DHCP		Possibility	Possibility
	Load Sharing		Possibility	Possibility
	Telnet		Possibility	Possibility
	SNMP		Possibility	Possibility
Price	$		400	1400

Ⅲ. 새로운 리눅스 라우터의 구현

새로운 리눅스 라우터는 공개된 운영체제인 리눅스를 사용하였으며 기본적인 라우팅 이외에 QoS[3]와 로드밸런싱[12]기능을 추가한 새로운 기능을 구현한 라우터이다.

새로운 리눅스 라우터를 구현하기 위한 새로운 기능과 하드웨어 구조 및 개발환경을 살펴보면 다음과 같다.

1. QoS와 로드밸런싱의 기능

1) QoS(Quality of Service)

인터넷의 발전에 따라 사용자가 서비스에 대한 요구가 다양해지는 동시 서비스의 품질에 대한 요구가 높아지고 있다. 특히 멀티미디어 서비스의 도입에 따라 화상 전송과 같은 넓은 대역폭을 요구하는 서비스, 영상, 방송과 같은 높은 대역폭과 실시간 전송을 요구하는 서비스가 많이 도입된다. 많은 사용자가 요구하는 멀티미디어 서비스를 지원하기 위하여 망을 확장하는 동시에 기존 망을 효율적으로 최대한 활용할 수 있는 방안의 연구가 활발히 진행되고 있다.

기존의 인터넷은 best effort 서비스만 지원하며 서비스의 품질에 대한 고려는 미약하다. Ipv4에서는 패킷의 헤더 중 여러 종류의 서비스 클래스를 지원할 수 있는 TOS라는 8비트 필드가 규정되어 있지만, 현재까지 많이 사용

하지 않는다. 서비스의 품질을 고려하지 않은 경우에 패킷을 빨리 처리할 수 있는 장점이 있지만 데이터 전송의 지연을 보장해 줄 수 없다. 인터넷의 사용자가 요구하는 서비스 품질은 목적에 따라 많이 다르다. 예를 들어 일반 PC 통신 사용자는 연결만 요구하는 반면 인터넷 폰, 화상 회의와 같은 사용자는 실시간 데이터의 전송에 대한 보장을 요구한다.

인터넷의 표준화 기구인 IETF에서는 현재 인터넷 상에서 QoS를 제공하는 방법에 대해 표준화 작업을 진행하고 있다. IntServ(Integrated Services)모델, DiffServ(Differentiated Services)모델 등은 IETF에서 제안된 대표적인 인터넷 QoS 방안이다.

① IntServ모델

IntServ모델[9]은 best effort 서비스 외에 두 개의 새로운 서비스클래스를 추가한다. 1) guaranteed 서비스는 수용할 수 있는 최대지연 내 서비스를 제공해 주어야 하는 서비스이고, 2) predictive 서비스는 통계적으로 수용할 수 있는 지연 내를 제공하여야 할 서비스이다.

이러한 서비스를 추가하기 위하여 라우터에서는 특정한 패킷 stream 또는 flow에 대하여 요구된 서비스 품질을 보장하는데 필요한 자원을 확보하여야 한다. 즉, 하나의 stream 또는 flow가 설정하기 전에 경유된 라우터들에게서 사용자가 요구한 자원을 확보하여야 한다. 확보된 자원은 대역폭, 메모리 등이 포함된다.

stream 또는 flow를 설정하기 위하여 라우터들 간의 자원을 확보하기 위한 시그널링 프로토콜이 필요하다. IntServ모델에서 시스널링 프로토콜을 사용한다. RSVP[10][11]를 사용해서 자원을 할당하는 과정은 그림 3.1과 같다.

발신자가 연결을 설정할 때에 요구된 트래픽 특성 정보를 포함하는 PATH 메시지를 수신측에 보낸다. 중간에서 경과된 라우터들은 PATH 메시지를 받

을 때 라우팅 테이블을 이용하여 다음 허브로 전송한다. 수신자가 PATH 메시지를 수신한 후 RESV 메시지를 답장처럼 수신측으로 보내며 자원의 할당을 요청한다. 중간 라우터들은 RESV을 수신한 후에 요구된 자원을 제공할 것인지에 따라 RESV 메시지를 허락하거나 거부할 수 있다.

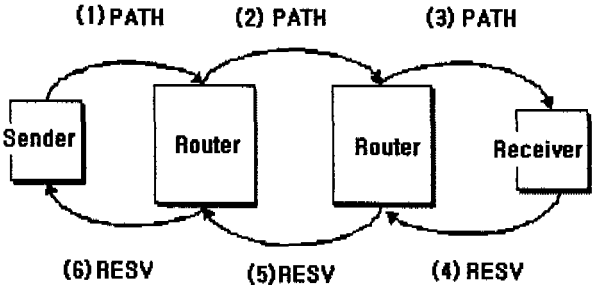

Figure 3.1 RSVP signaling

RESV 요구를 거부하는 라우터는 에러 메시지를 수신자에 전송하면 연결 설정은 실패로 종료된다. 모든 중간 라우터에서 요구된 연결을 설정이 가능한 경우에는 각 라우터에서 경로를 설정할 때에 요구된 대역폭 및 메모리를 할당하고 연결 상태에 관련된 정보를 저장한다.

IntServ 모델을 지원하기 위하여 각 중간 라우터에서 flow의 연결 상태에 관련된 정보를 저장하여야 하기 때문에 라우터에서 이런 정보를 저장하는 공간을 요구할 뿐만 아니라 라우터의 처리 오버헤드가 커질 수 있다. 인터넷 백본 라우터인 경우 전송 속도가 상당히 높고 연결된 flow의 개수가 많음으로 IntServ모델을 지원하기는 힘들다. IntServ모델을 지원하기 위하여 라우터마다 RSVP, 연결 관리, 패킷의 스케줄링 등의 기능을 지원하며 라우터에 대한 요구가 높다. 그리고 IntServ 모델을 지원하기 위하여 모든 중간 라우터에서

IntServ모델을 지원한다.

② DiffServ모델

확장성 문제 때문에 IntServ모델은 구현과 도입하는데 문제가 있다. IETF에서 이러한 문제를 해결하기 위하여 DiffServ모델[4]을 도입한다.

DiffServ 모델에서는 다양한 flow가 많지 않은 서비스 클래스로 분류되며 중간 라우터에서는 이러한 서비스 클래스별로 처리한다. 모든 라우터에 대하여 flow상태 관리 및 시그널링을 요구하지 않는다. Diffserv는 IP 해더의 TOS의 부분중의 6비트를 QoS를 정하는 부분으로 바꿨다. 이러한 방법은 모든 트래픽을 요구하는 QoS에 따라 나누고 이에 따라 트래픽을 집합함으로써 스케쥴링 문제를 해결하였다.

RSVP와는 달리 목적지와 원천지간의 어떠한 QoS 요구사항에 대한 정보교환이 일어나지 않게 함으로써 RSVP가 가지고 있던 연결설정 비용에 대한 문제를 제거하였다. Diffserv의 short-lived flow는 응답성을 개선하였으며 다른 호스트와의 빠른 처리를 할 때 발생하던 과부하를 줄였다. Diffserv에서 사용되는 트래픽 집합모델은 예측성이 떨어진다는 단점이 있다.

reservation, signaling mechanism, traffic shaping이 Diffserv에서는 없기 때문에 트래픽의 수준은 매우 동적이다. 이러한 이유 때문에 Diffserv에서 특정한 수준의 서비스를 보장하는 것은 매우 어려운 일이 되었다. 따라서 Diffserv에서는 어떤 수준의 서비스 보장보다는 각각의 집합에 대한 규칙에 근거하여 상대적으로 서비스가 제공되도록 한다. 즉 어떤 집합은 다른 집합보다 더 데이터를 잘 받거나 못 받도록 한다.

③ IntServ모델과 DiffServ모델

확장성의 문제 때문에 IntServ는 WAN상에서는 거의 구현되지 않는다. 그

러나 IntServ는 기업체의 네트워크에서 구현될 수 있다. 그리고 WAN상에서는 DiffServ가 구현된다. WAN은 DiffServ이고 LAN은 DiffServ와 IntServ가 혼합된 형태의 망에서 패킷을 보내고 받는 사이의 경로에서 WAN과 LAN의 자원을 사용할 때 QoS의 제공에 대해 살펴보면 다음과 같다.

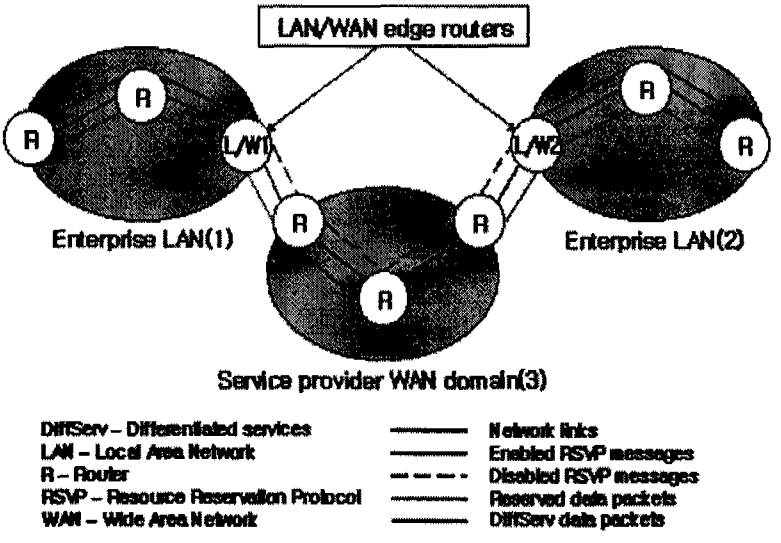

Figure 3.2 WAN with RSVP and RESV message

만약 호스트가 IntServ를 사용한다면 문제가 있다. 그리고 DiffServ에서도 이 문제 해결을 위한 제안을 하였다. 그것은 LAN(1)에서는 대역폭을 예약하기 위하여 RSVP를 사용하고 그림 3.2에서 L/W1로 표시된 LAN과 WAN의 경계에서는 WAN 라우터가 패킷에 있는 RSVP 정보를 무시할 수 있도록 RSVP 경로와 RESV 메시지를 수정하는 것이다. 그리고 패킷이 LAN(2)에 도착하면은 패킷은 원래의 형태로 복구된다. 그리고 경로와 RESV메시지는 자원을 계속해서 예약한다.

과정을 요약하면 다음과 같다. 패킷이 RSVP를 사용하여 LAN(1)을 통하여

LAN/WAN edge(L/W1)에 도착한다. 이 라우터는 예약된 자원을 만족할 수 있는 코드값을 패킷에 할당한다. 그리고 이것이 LAN/WAN edge(L/W2)에 도달하면 이 라우터는 패킷의 RSVP의 형태로 복구 시키면 이 패킷은 LAN(2)에서 RSVP의 표준에 따라 서비스를 받게 된다.

2) 로드밸런싱(Load balancing)

로드밸런싱[12]은 하나의 컴퓨터에서 동시에 여러 개의 처리장치가 작업을 수행하는 다중 병렬 시스템에서 각각의 처리장치에 대해 작업을 균형 있게 할당하고 부하를 조정하는 방법이다.

만일 다른 장치에 비해 하나의 처리장치에 지나치게 많은 작업을 할당하게 되면 병목현상이 발생하게 되며 결과적으로 전체 시스템의 작업성능이 떨어지게 된다.

라우팅에서 수신 주소로부터 동일한 거리에 있는 모든 네트워크 포트를 통해 트래픽을 배포할 수 있는 라우터의 능력인 뛰어난 부하 분배 알고리즘은 라인 속도와 신뢰성 정보를 모두 사용한다. 부하 분배를 하면 네트워크 세그먼트 사용이 증가하며, 따라서 효과적인 네트워크 대역폭도 증가한다.

2. 하드웨어 구조와 개발 환경

새로운 리눅스 라우터의 하드웨어 구조와 개발환경을 살펴보면 다음과 같다. CPU는 motorola의 PowerPC계열인 MPC860ENZ50C1을 사용하였으며 50Mhz로 동작한다.SDRAM은 16Mbyte, RAM disk flash 4Mbyte의 메모리를

사용한다. 2개의 망 지원을 위하여 2개의 이더넷(10Mbps) 포트와 2개의 E1(2Mbps) 포트가 적용된다.

1) 하드웨어 사양

① 블록다이어그램[13]

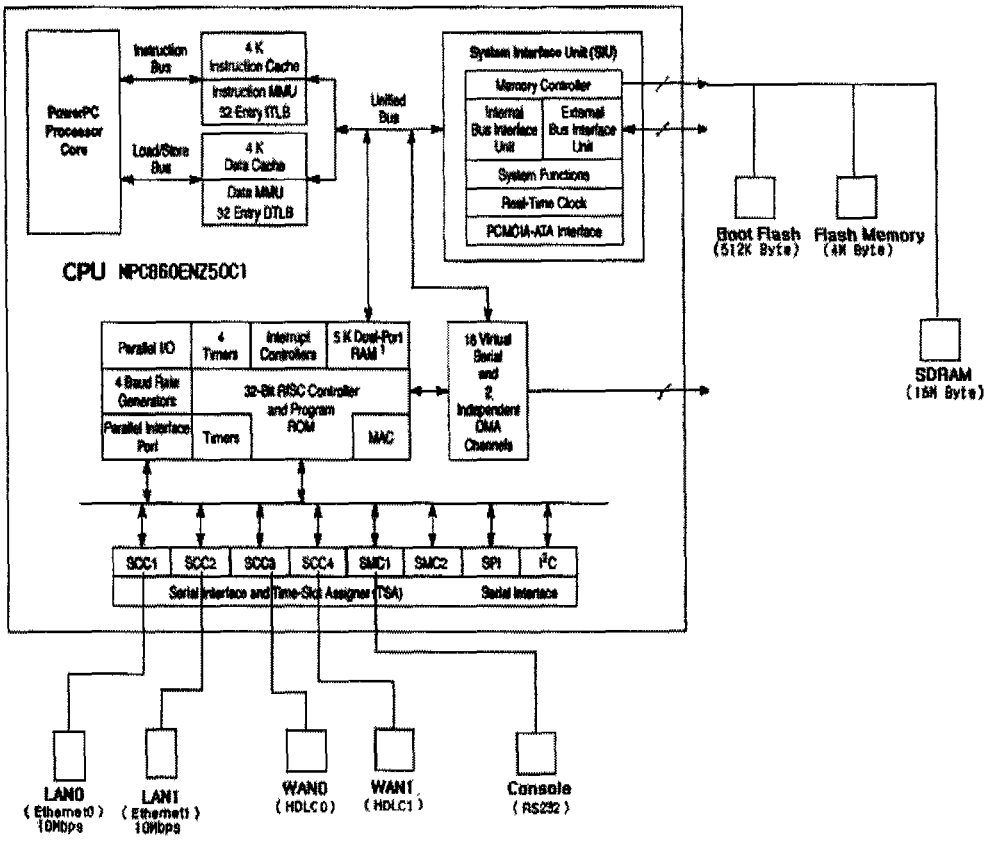

Figure 3.3 Linux router block diagram

② 하드웨어 규격

Table 3.1 Hardware size

	Spec		Linux router
Hardware	CPU		MPC860ENZ50C1(50MHz)
	Memory (Byte)	DRAM	16M(Up to 32M)
		Flash	4M(Up to 8M)
		Boot	512K
	Ethernet port		10Mbps X 2
	WAN port		E1 X 2

③ 새로운 리눅스 라우터의 보드 사진

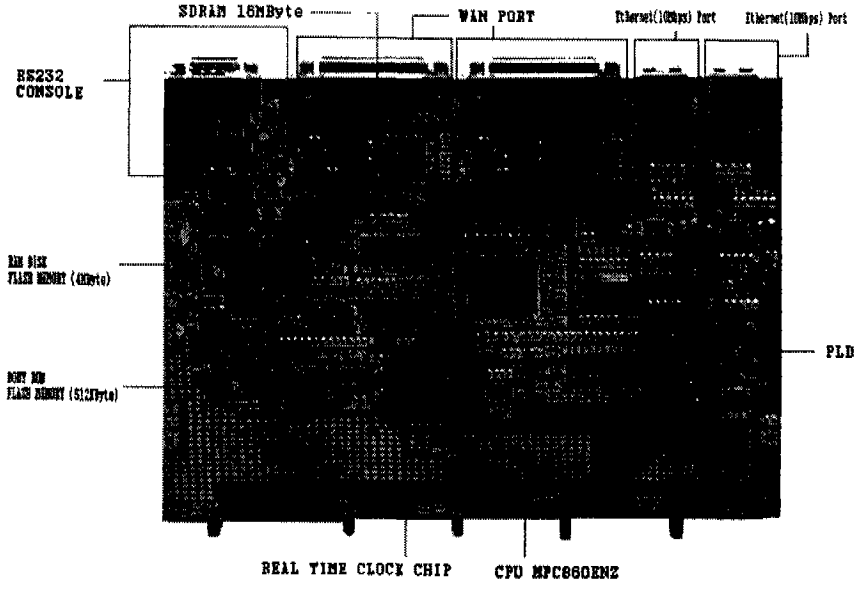

Figure 3.4 Real picture of new linux router board

2) 새로운 리눅스 라우터 보드의 CPLD

CPLD는 VHDL를 사용하였으며 device는 Xilinx의 36 cell을 가지고있는

XC9536-7VQ44C를 사용하였으며 소스내용은 부록 자료 2에 수록되어 있다.

CPLD의 신호들은 CPLD로 들어가는 입력신호와 CPLD에서 생성되어서 각 chip에 전달되는 출력신호들이며 각 신호들의 동작은 그림 3.5와 같이 시뮬레이션 화면에서 알 수 있다.

① 입력신호

입력신호는 대부분 CPU의 외부신호들이다. pclkout 신호는 시스템 clock으로서 50Mhz의 clock이며 이 신호에 맞춰 각각의 신호들이 동작한다. clk10hz는 CPU의 외부신호가 아니고 wan tx,rx 신호의 동작을 led로 보여주기 위하여 0.2초의 시간을 구현하기 위한 NE555 chip에서 공급되는 clock이다. poreset은 CPU에 들어가는 reset 신호이다. cs1은 chip select 신호로서 해당되는 chip인 RTC chip에 동작가능 신호를 CPU에서 내보내는 신호이다. cs7은 chip select 신호로서 CPLD chip에 동작가능 내보내는 신호이다. paddr4,5,6,7은 wan tx,rx 신호의 동작을 led로 보여주기 위한 address신호이다. paddr23은 RTC chip에 address나 data신호를 구분해주는 address신호이다. pts는 transfer start 신호로 CPU의 동작을 보여준다. pta는 transfer acknowledge 신호로 CPU의 동작을 보여준다. prw는 read/write 신호로서 address 또는 data 신호를 읽고 쓰게끔 할 수 있는 동작신호이다. hreset은 CPU가 동작 후 주변 chip들에게 보내는 reset 신호이다.

②출력신호

출력신호는 CPLD에서 생성시킨 신호이다. lporest은 flash memory에 보내는 reset신호이다. ltstart는 power on시에 동작하는 reset chip에 pts신호가 인가되게 된다. ds는 data strobe 신호로서 cs1이 동작 시 RTC chip에 data 인가 신호이다. as는 address strobe 신호로서 cs1이 동작시 RTC chip에

address인가 신호이다. cs는 cs1신호를 RTC chip에 그대로 보내준다. rw는 prw신호를 RTC chip에 그대로 보내준다. run_led는 정상적인 시스템 동작상태를 보여주는 신호이다. fail_led는 시스템 비정상적인 동작상태를 보여주는 신호이다.

w1_txled,w1_rxled는 wan1 포트의 동작이 진행될 때 tx,rx신호를 led로 점등시켜 보여주는 신호이다. w2_txled,w2_rxled는 wan2 포트의 동작이 진행될 때 tx,rx신호를 led로 점등 시켜 보여주는 신호이다.

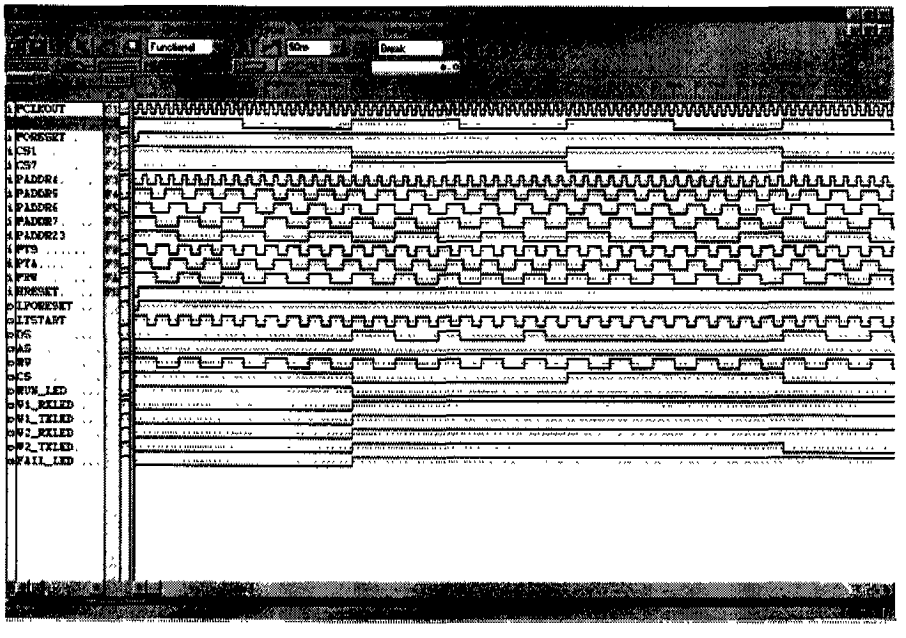

Figure 3.5 CPLD VHDL simulation

3) 개발환경

새로운 리눅스 라우터를 개발하기 위해서 모니터 프로그램을 만들어야 한다. 모니터 프로그램은 실행 파일을 외부로부터 다운로드 후 수행하고 수행결과를 확인할 수 있는 기능을 제공한다. 그림 3.6은 개발 환경이며 그림 3.7은

모니터 프로그램의 화면과 명령어이다.

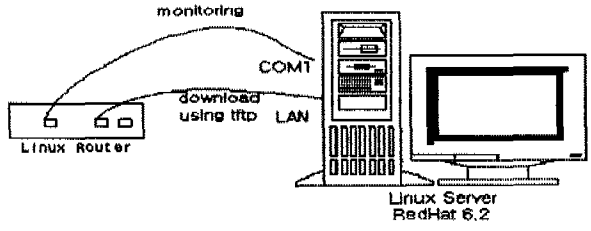

Figure 3.6 Development environment for linux router

이 모니터 프로그램을 이용하면 커널 컴파일 시간을 제외하고 프로그램 수행 결과를 보는데 1분 정도가 걸린다.

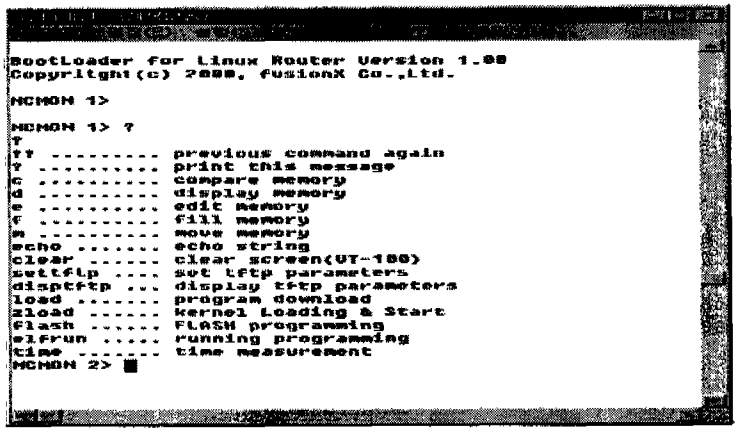

Figure 3.7 Monitor program screen

새로운 리눅스 라우터는 4~8MB 크기의 RAM Disk을 사용하기 때문에 필요한 소프트웨어만을 RAM disk에 저장한다. 필요한 실행 파일을 수행하기 위해서 필요한 공유 라이브러리를 체크해서 넣어 준다. BlueCat의 mkrootfs 라는 실행 파일이 RAM disk image를 작성하게 해준다.

Ⅳ. 시험 및 고찰

2개의 망 지원이 가능한 새로운 리눅스 라우터는 공개된 운영체제인 리눅스를 사용하였기 때문에 기능설정과 포트설정은 리눅스 명령어를 사용한다. 또한 PC에서 구동하여 사용되는 리눅스 명령어와도 동일하다. 초기화는 버전에 따라 PC와는 다르지만 BIOS의 개념으로 해석하면 된다.

1. 기능설정과 포트설정

1) 기능설정

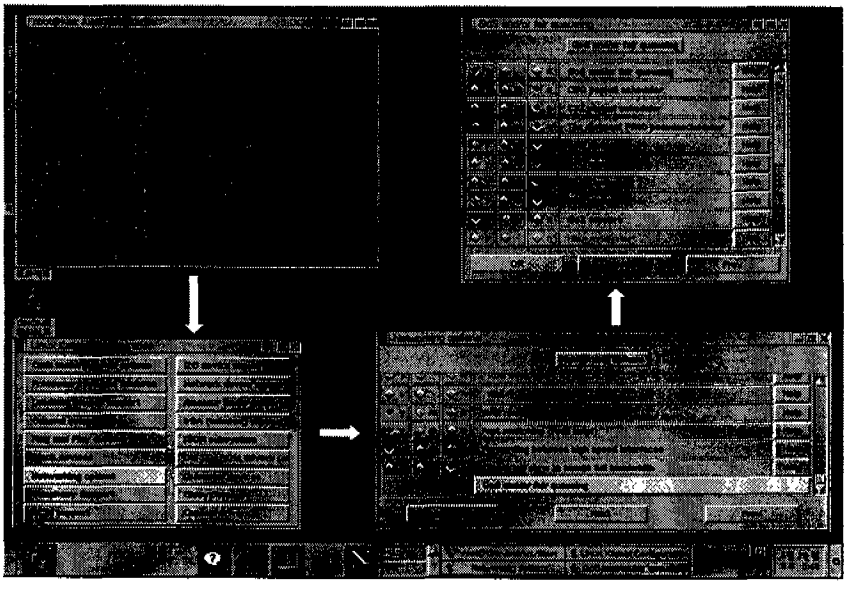

Figure 4.1 QoS setting screen

그림 4.1은 리눅스에서 QoS의 설정화면으로 커널 컴파일 하기 전에 이러한 설정을 하여주면 커널 내에 QoS기능이 설정이 된다. 리눅스의 장점인 PC내에서 기능설정 후에 장비에 다운로드 하면 장비에서 기능이 구현된다는 것이며 이러한 다양한 기능의 지원이 가능하다.

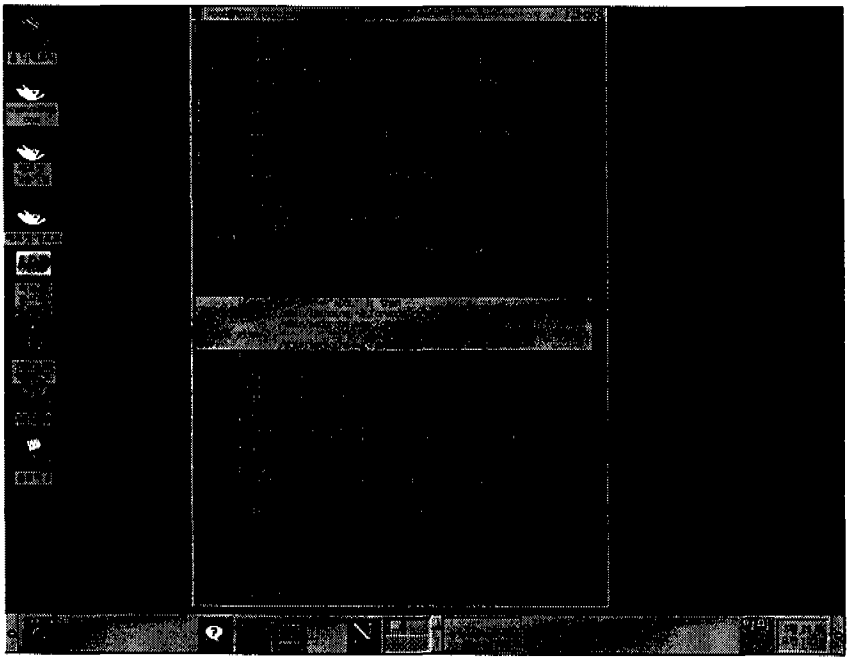

Figure 4.2 Load balancing setting screen

그림 4.2는 리눅스에서 로드밸런싱의 설정화면으로 커널 컴파일하기 전에 특정화일(rc.eth0)내에 ipchain 명령어를 사용하여 부록자료 3과 같이 설정하면 커널 내에 로드밸런싱 기능이 설정된다.

그림 4.3은 리눅스에서 커널 컴파일 화면으로 커널 내에 QoS와 로드밸런싱의 설정 후 커널 컴파일을 하면 바이너리 파일이 생성되게 된다. 생성된 파일을 장비에 다운로드 하면 QoS와 로드밸런싱 기능이 가능한 리눅스 라우터가 된다.

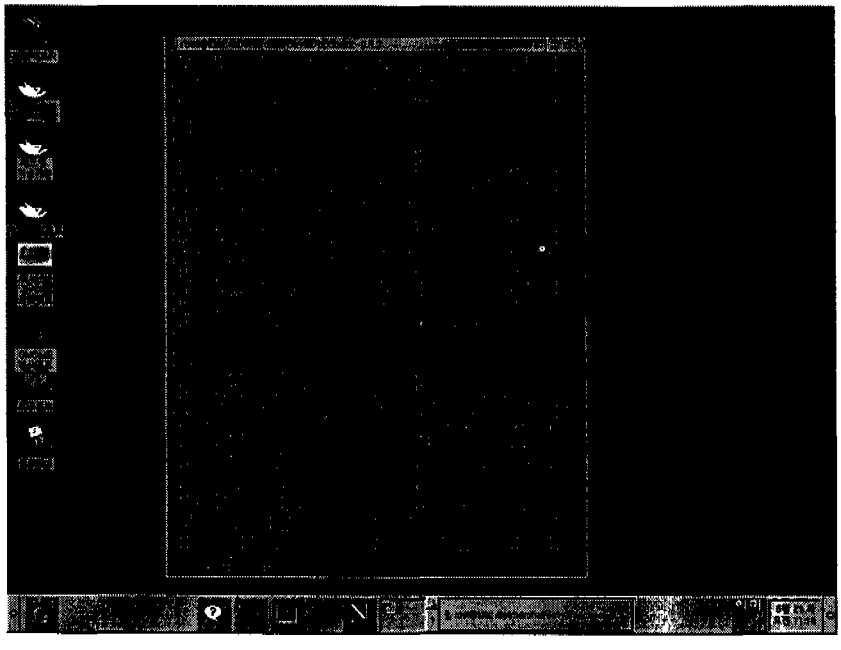

Figure 4.3 Kernel compile screen

2) 포트설정

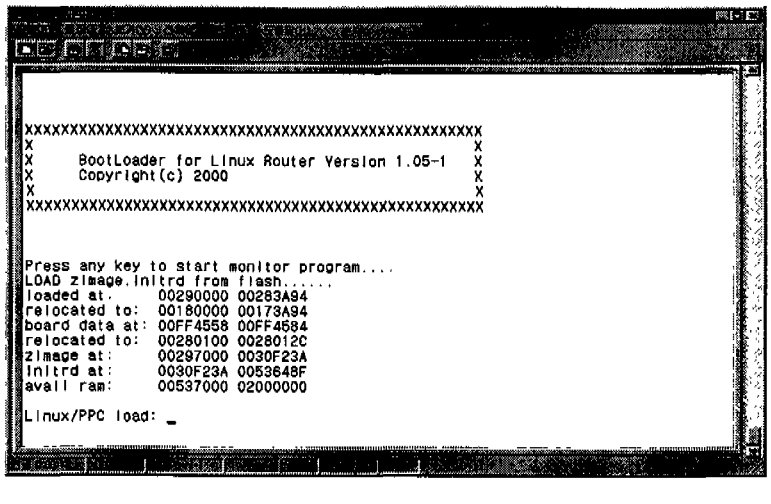

Figure 4.4 Router booting screen

라우터와 PC의 터미널을 연결한 후 라우터의 스위치를 켜면 터미널 프로그램에 그림 4.4와 같은 부팅화면이 나온다. 이 초기화면은 부트로더의 버전에 따라 조금씩 다를 수 있다

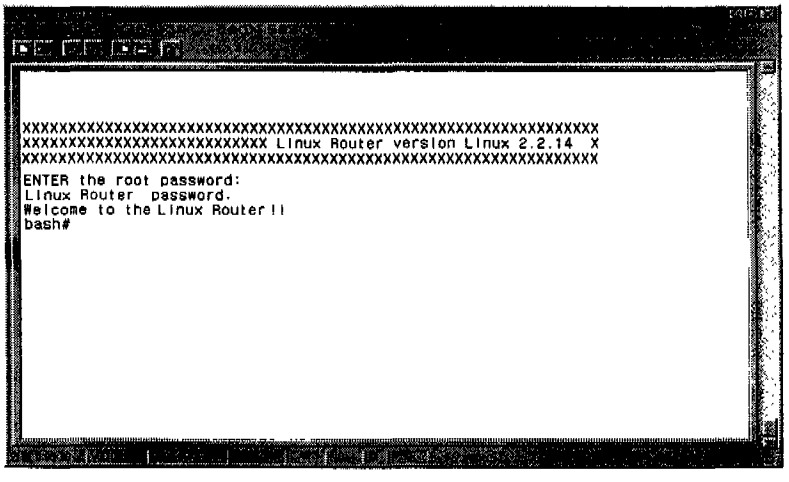

Figure 4.5 Router login screen

위 그림 4.5는 로그인 화면으로 id는 root이다. 이렇게 로그인이 되고 나면 사용자 환경에 맞게 기본설정 및 고급설정을 할 수 있는 bash# 환경모드가 된다.

① LAN 포트 설정

예를 들어서 ethernet0을 10.10.10.10 netmask를 255.255.255.0로 setting을 하려고 한다면 bash prompt에서 bash# ifconfig eth0 10.10.10.10 netmask 255.255.255.0으로 설정한다.

LAN포트 설정이 제대로 이루어 졌는지 확인하기 위해서는 리눅스의 명령어인 ifconfig를 사용한다. 이 명령어를 사용하면 설정된 모든 포트들이 화면에 나타난다. 그림 4.6은 LAN포트 설정화면이다.

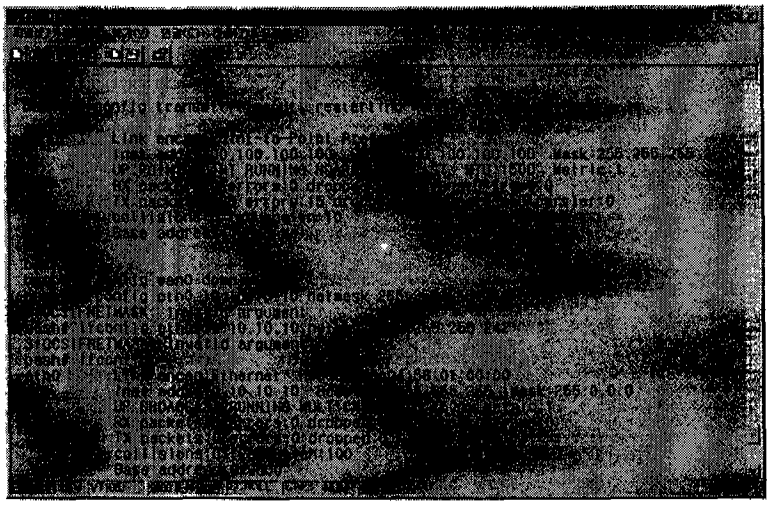

Figure 4.6 LAN setup screen

② WAN 포트 설정

WAN 포트는 그림 4.7과 같이 ifconfig 명령어를 사용하여 설정 해준다.

bash# ifconfig wan0 10.10.10.1 netmask 255.255.255.0

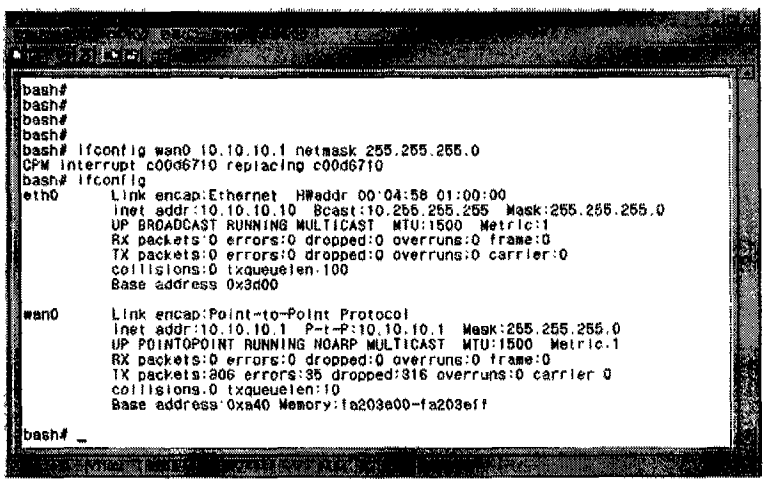

Figure 4.7 Confirmation method of wan port setup

WAN 포트에서 HDLC와 PPP의 프로토콜 중에서 사용하는 프로토콜을 선택하여 사용하는데 PPP 프로토콜을 사용하려면 그림 4.8과 같이 설정해 주면 된다.

bash# wanctl wan0 link ppp

다시 HDLC 프로토콜 사용을 위해 변환 하고자 할 때에는 WAN 포트를 한번 다운을 시킨다.

bash# ifconfig wan0 down

이렇게 되면 WAN포트가 설정이 해제되고 설정하고자 하는 프로토콜로 선택하여 다음과 같이 설정하면 된다.

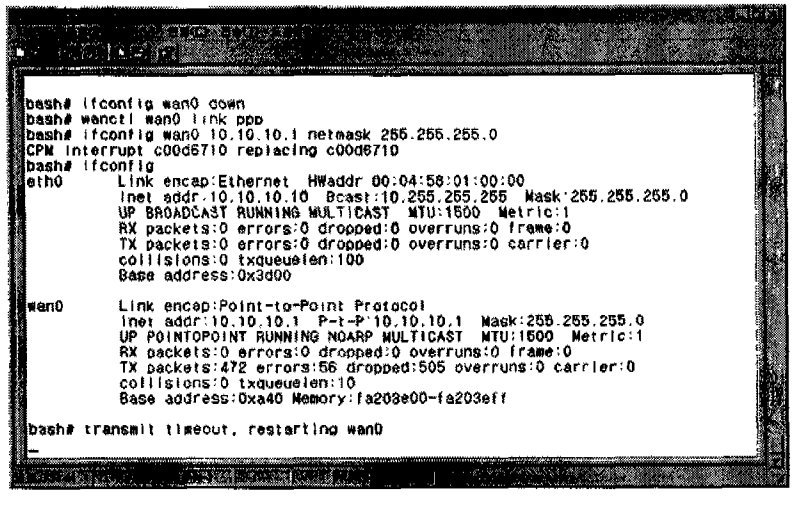

Figure 4.8 Protocol setup & confirmation

bash# wanctl wan0 link hdlc를 해주고 WAN0을 다시 up을 해주면 된다.

bash# ifconfig wan0 10.10.10.1 netmask 255.255.255.0

를 해주고 확인을 하려면 ifconfig 명령어로 확인을 하면 된다.

bash# ifconfig를 하면 그림 4.8과 같이 확인이 가능하다.

2. 성능시험 및 고찰

RTOS 라우터나 일반적인 라우터는 전용회선 망을 통해 인터넷에 접속하여 내부 망에 연결하여 라우팅을 수행한다. 본 논문에서 구현한 새로운 리눅스 라우터도 전용회선 망도 사용하므로 전용회선 서비스에서도 같은 결과가 나온다.

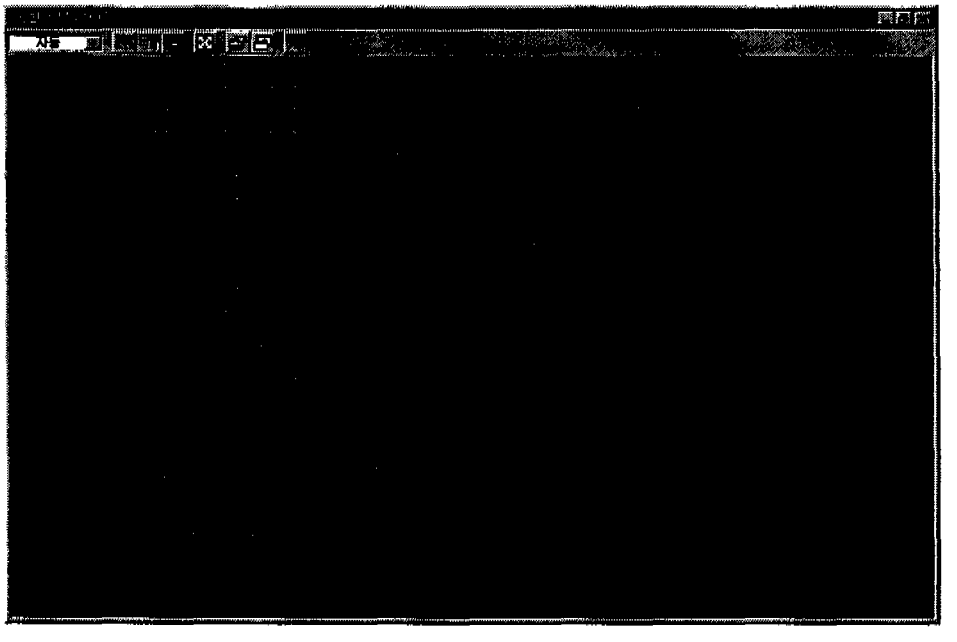

Figure 4.9 Ping test for QoS

새로운 리눅스 라우터는 새로운 기능 추가로 전용회선 망과 ADSL/CABLE 망을 동시에 사용함으로서 낮은 가격으로 고속의 속도를 보장한다. 라우터의 시험방법은 연결시험(ping test)과 속도시험(speed test)이 통상적인데 새로운 리눅스 라우터의 시험은 2개의 망을 동시에 사용하기 때문에 전용회선 망과

ADSL/CABLE 망을 구분하여 시험한다.

패킷 손실률을 없애기 위한 QoS기능은 그림 4.9와 같이 연결시험을 통하여 시험결과를 확인할 수 있는데 1400Byte 패킷을 www.yahoo.co.kr[211.32.119.151]에 1401번 보냈고 1401번을 받았는데 손실률은 0%이다. 그러므로 QoS의 패킷 손실률의 보상기능이 이루어졌다.

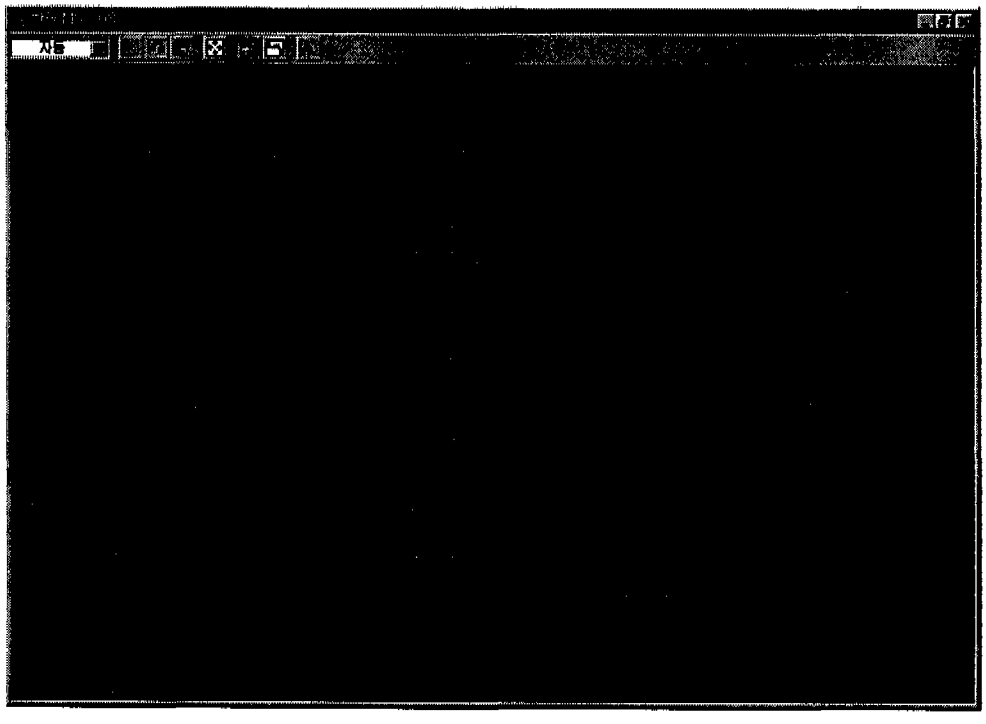

Figure 4.10 Routing processing for network of exclusive line

로드밸런싱 기능의 성능시험은 전용회선 망과 ADSL/CABLE 망을 구분하여 시험을 하는데 라우터가 2개의 망 중에서 어느 망을 쓰는지 경로를 분석하여 보고 속도시험을 통하여 2개의 망의 속도차이를 비교한다.

그림 4.10은 반드시 1개의 공인된 IP를 사용해야 하는 인터넷 서비스를 받고자 원할 경우 라우터의 내부 망의 게이트웨이인 192.168.1.1에 우선 연결되

고 라우터의 전용회선 망의 게이트웨이인 210.114.208.65로 접속이 된다. 다음은 신비로 라우터[203.240.236.77]로 접속되고 202.30.241.17->202.30.243.51->210.120.248.231->210.120.192.206->210.92.194.202->128.134.40.73->211.217.32.222->168.126.109.190->211.106.67.46의 경로를 통하여 http://hangame.naver.com[211.218.152.16]으로 연결된다.

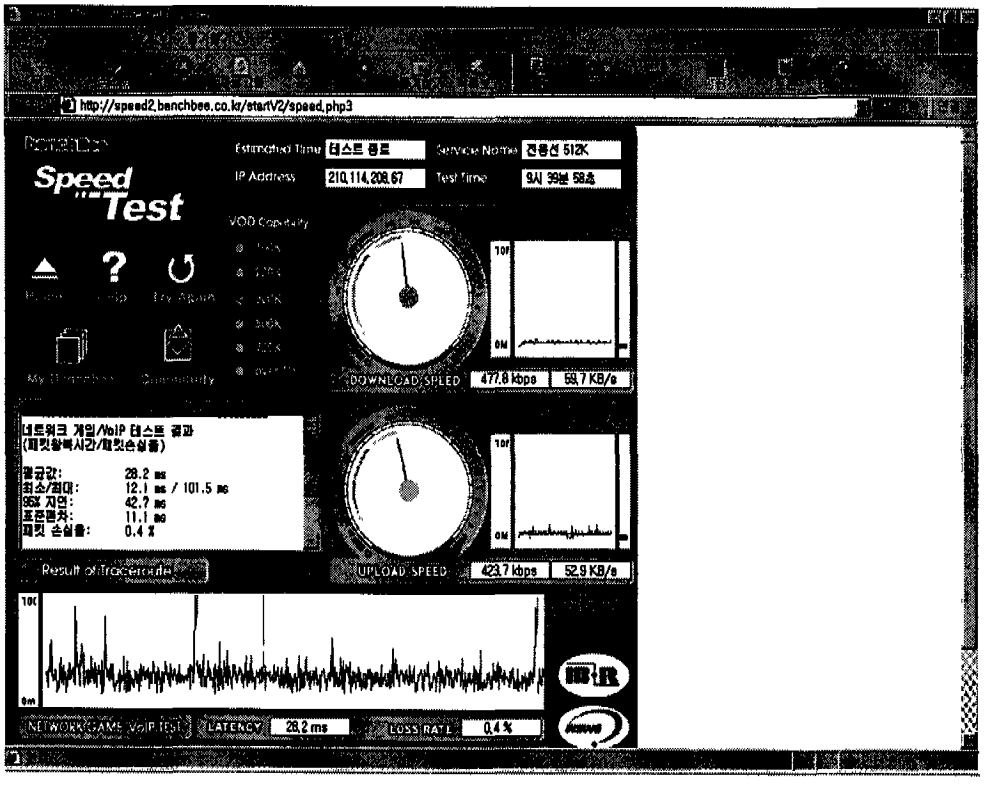

Figure 4.11 Speed test for network of exclusive line

그림 4.11은 전용회선 망을 사용하였을 때의 속도를 측정한 화면으로 새로운 리눅스 라우터가 전용회선 망으로 접속이 될 경우 내부 IP는 210.114.208.67이고 512Kbps의 전용회선 서비스를 사용한다.

전용회선 망의 속도는 512Kbps의 전용회선 서비스를 받아도 망의 여러 가

지 환경여건에 따라 속도의 차이가 있다. 그림 4.11도 이러한 점을 감안하여 측정결과를 보면 다운로드 속도는 477.8Kbps,업로드 속도는 423.7Kbps의 속도로 측정되었다. RTOS 라우터와 기존의 라우터도 전용회선 망만을 사용하기 때문에 같은 측정결과가 나온다.

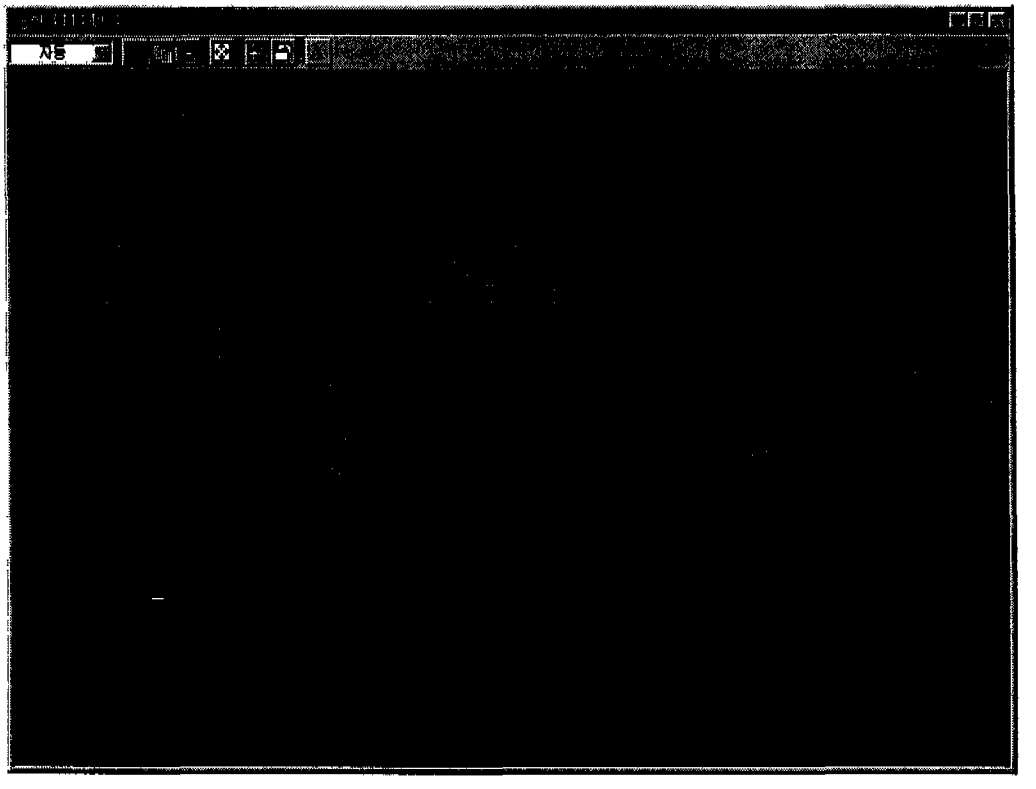

Figure 4.12 Routing processing for network of ADSL/CABLE

그림 4.12는 반드시 1개의 공인된 IP를 사용하지 않아도 되는 일반적인 인터넷 서비스를 받고자 원할 경우 라우터의 내부 망의 게이트웨이인 192.168.1.1에 우선 연결되고 라우터의 ADSL/CABLE 망 게이트웨이인 210.114.208.94에 접속한다. 다음에 211.221.63.126->168.126.136.2->211.195.72.157->168.126.109.9->211.192.46.72의 경로를 통하여 http://ns.korner.net[168.126.63.1

]으로 연결된다.

그림 4.13은 ADSL/CABLE 망을 사용하였을 때의 속도를 측정한 화면으로 새로운 리눅스 라우터가 ADSL/CABLE 망으로 접속이 될 경우 유동 IP를 사용하는 ADSL회선 서비스를 사용한다.

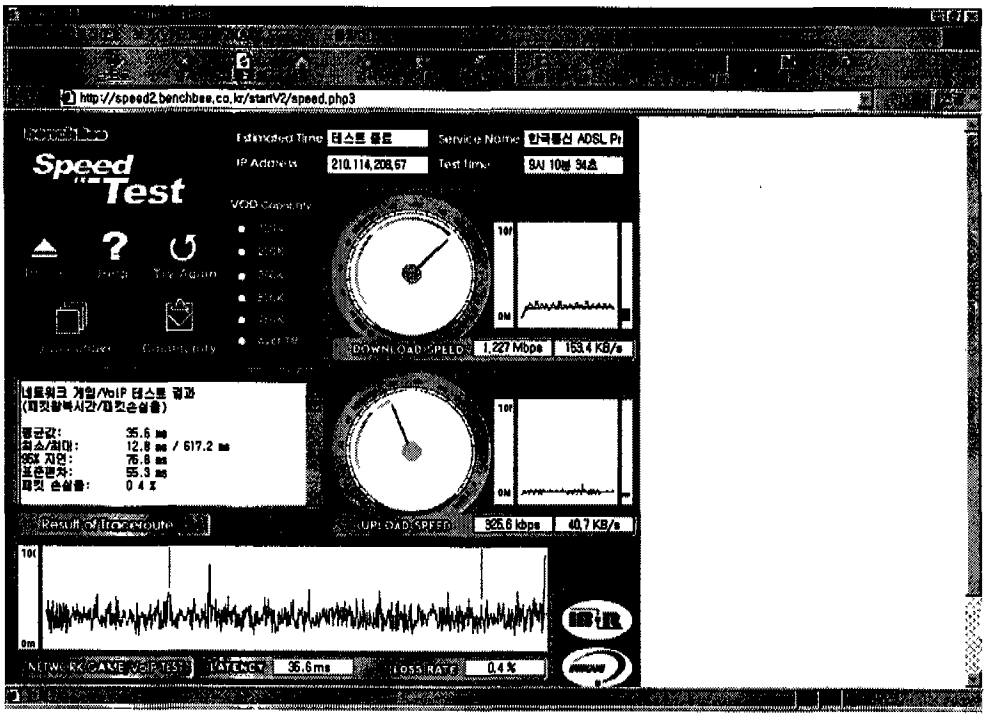

Figure 4.13 Speed test for network of ADSL/CABLE

ADSL/CABLE 망의 속도는 1Mbps~3Mbps 이상으로 전용회선 서비스에 비해 속도는 빠르나 안정성에 대해서는 그다지 좋은 조건은 아니다. 또한 여러 가지 환경여건에 따라 속도의 차이가 있다. 그림 4.13도 이러한 점을 감안하여 측정결과를 보면 다운로드 속도는 1.227Mbps,업로드 속도는 325.6Kbps의 속도로 측정되었다. RTOS라우터와 기존의 라우터는 512Kbps를 기준으로 전용회선 망만을 사용하기 때문에 이러한 고속의 측정결과가 나오기가 어렵

다.

2개의 망 중에서 1개의 망이 접속이 끊기는 경우에는 다른 접속 가능한 망으로 호스트들은 접속한다. 그림 4.14는 ADSL/CABLE 망이 끊길 경우 전용회선 망으로 접속이 되는 화면으로 내부 게이트웨이인 192.168.1.1을 거쳐 라우터에서 ADSL/CABLE 망의 게이트웨이인 210.114.208.94를 통해 접속을 시도하다가 다시 전용회선 망 게이트웨이인 210.114.208.65로 접속한다

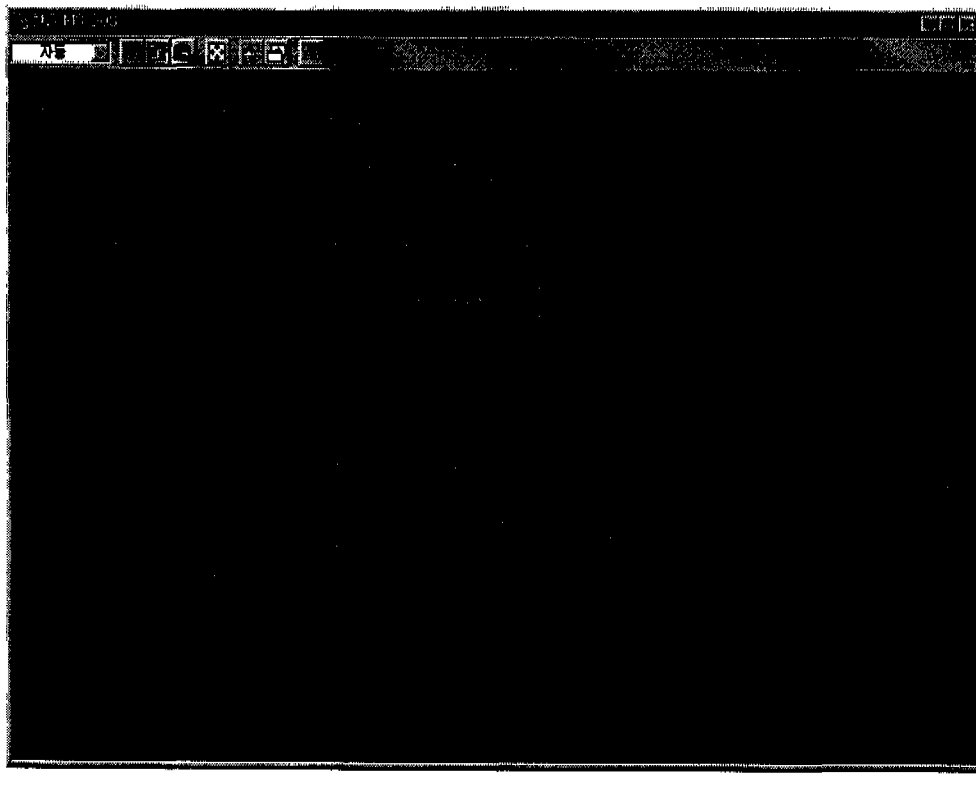

Figure 4.14 Routing processing of autoconnection

RTOS 라우터나 기타의 라우터의 대부분은 1개의 이더넷 포트와 2개의 E1 포트를 가지고 있는데 E1 포트는 전용회선 망으로 연결하고, 1개의 이더넷 포트는 내부 망에 연결하여 라우팅을 수행한다.

본 논문에서 구현한 리눅스 라우터는 2개의 이더넷 포트와 2개의 E1 포트로 설계함으로서 E1 포트는 전용회선 망으로 연결하고, 동시에 1개의 이더넷 포트는 ADSL/CABLE 망과 나머지 1개의 이더넷 포트는 내부 망에 연결시켜서 QoS와 로드밸런싱 기능을 추가하였다.

그러므로 표 2.1,표 2.2에서 언급한 개발비용과 장비가격을 비교하면 2~3배 이상으로 효율적이며, 성능시험을 통하여 2개의 망을 사용함으로서 표 4.1, 표 4.2와 같이 512Kbps 비용과 낮은 ADSL/CABLE 망 회선비용으로 T1,E1급 이상의 속도를 충족시킴으로서 2~3배 이상의 효과가 있다.

Table 4.1 Service charge of exclusive line and speed

Service	Price	Speed
512K	₩ 580,250	512kbps
T1	₩ 1,298,000	1Mbps
E1	₩ 1,458,000	2Mbps
ADSL/CABLE	₩ 28,000	1~3Mbps Up

Table 4.2 Service charge & speed between RTOS router & linux router

	Service	Price	Speed
RTOS router	512K	₩ 580,250	512Kbps
Linux router	512K + ADSL/CABLE	₩ 608,250	1~3Mbps Up

그림 4.15는 새로운 리눅스 라우터의 시스템 구성으로서 허브에 연결된 호

스트들은 2개의 망을 모두 사용한다.

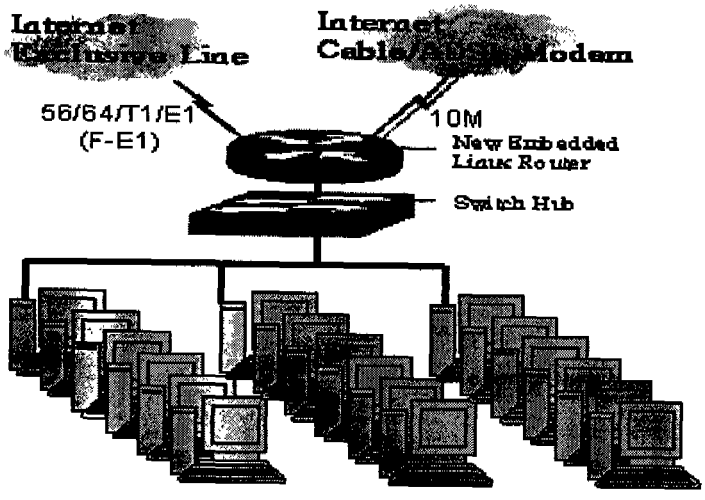

Figure 4.15 New linux router system

본 논문에서 구현한 2개의 망 지원이 가능한 리눅스 라우터는 장비의 낮은 가격으로 경쟁력이 있으며 새로운 하드웨어 설계 및 기능을 구현함으로서 기존의 RTOS 라우터나 기타 라우터보다 기능과 가격적인 부분에서 우월한 라우터라 생각한다.

V. 결 론

본 논문에서는 리눅스를 사용하여 2개의 망 지원이 가능한 리눅스 라우터를 하드웨어로 구현하고 성능시험을 통해 기능을 검증하고 그 효율성을 입증하였다.

일반적으로 라우터에 사용되는 운영체제는 상용 RTOS나 기업에서 자체 개발한 운영체제를 사용하게 되는데 경제적, 시간적, 기능적인 문제가 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 공개된 운영체제인 리눅스를 사용하였다. 또한 하드웨어적으로 2개의 망 지원을 위해 설계하였으며, 소프트웨어적으로 인터넷을 통해 전달되는 패킷의 손실을 보장할 수 있는 QoS 기능과 로드밸런싱 기능을 추가하여 시스템 개발환경을 구축하였다.

그 결과 기존 RTOS 라우터와 여러 측면에서 비교하여 볼 때 OS 구입비용은 $50000 이상 차이가 있으며 제품화에 따른 장비의 가격도 2~3배 이상 차이가 있다. 또한 성능시험에서도 2개의 망을 사용하기 때문에 보다 안정적이며 회선비용 부담에서도 512Kbps와 낮은 ADSL/CABLE 망 회선비용으로 2~3배의 속도인 T1, E1이상 속도보장의 효과가 있어 경제적, 기능적, 안정적으로 효율성이 높음을 검증하였다.

리눅스 라우터는 리눅스 운영체제만이 가지고 있는 공개된 소스를 응용하여 여러 기능을 부여하였을 경우에 다양한 어플리케이션의 구현에 있어 기대효과가 클 것이다.

구현할 수 있는 여러 가지 기능 중에 최근 급속한 인터넷의 발전으로 야기되는 확실한 보안(Fire wall)문제는 앞으로 해결할 과제로서 지속적인 연구가 되어야 할 것이다.

참 고 문 헌

[1] 채기준(1998), MPOA를 이용한 가상 라우터 및 멀티프로토콜 수용 방안에 관한 연구, 《이화여자대학교》.

[2] 이영천(1999), SOHO라우터 기반 홈 네트워킹, 《전자공학회지》 184, PP. 27~32.

[3] S. Shenker et al.(1997), "Specification of Guaranteed Quality of Service", *RFC* 2212.

[4] Y. Berenet, et. al.(1998), "A Framework for End-to-End QoS Combining RSVP/Int-Serv and Differentiated Services", *Internet Draft <draft-bernet-intdiff-00.txt>.*

[5] 이지영(1998), 라우터 시험 시스템의 설계 및 구현, 《정보과학회 논문지》,C 4.4, PP.601~610.

[6] 오주일(1995), 《리눅스(Linux)》,인포북.

[7] 한선형(1999), 임베디드 시스템의 구성 , 《전자기술첨단》 134 , PP.81~85.

[8] 주성순(2000), 차세대 인터넷 라우터, *Telecommunications Review* 10, PP.470~480.

[9] J. Wroclaswski(1997), "The Use of RSVP with IETF Integrated Services", *RFC* 2210.

[10] R. Branden et al.(1997), "Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification", *RFC* 2205.

[11] P. P. White(1997), "RSVP and Integrated Services in the Internet: A Tutorial", *IEEE Comm. Magazine.*

[12] 케이블트론 시스템즈 기술지원부(1999), 서버 로드밸런싱과 스위치 라우터의 역할, 《경영과 컴퓨터》,PP.266~269.

[13] Motorola (1999), *MPC860 User Manual*, Motorola,INC.

ABSTRACT

Implementation of Linux Router Supporting Two Network Systems

KIM SEOG SEONG

Information and Telecommunications Major

Graduate school of Information and Telecommunications

University of Incheon

Making low in price of development expense router, A Linux is an essential condition. Most of routers have RTOS that it's expensive solution and insecurity. The Linux router has two ethernet ports and two E1 port. The E1 port connected network of exclusive line , one ethernet port connected ADSL/CABLE.

Also the Linux router has QoS and road balancing.

Furthermore, the router includes all functions of RTOS and can use two internet networks at the same time(It is tested).

This Linux router is 2~3 time fast than RTOS. The price for commercial product of the Linux router is advanced(2~3 time cheaper than the other).

부 록

자료 1 . QoS라우터의 config 파일

```
[daniel@prj linux]$ cat .config
# CONFIG_RPXCLASSIC is not set
# Automatically generated make config: don't edit
# Platform support
CONFIG_PPC=y
CONFIG_8xx=y
CONFIG_MPC860=y
CONFIG_SERIAL_CONSOLE=y
CONFIG_MBX=y
CONFIG_MACH_SPECIFIC=y
CONFIG_MATH_EMULATION=y
# General setup
CONFIG_EXPERIMENTAL=y
CONFIG_MODULES=y
CONFIG_KMOD=y
CONFIG_NET=y
CONFIG_SYSCTL=y
CONFIG_SYSVIPC=y
CONFIG_BINFMT_ELF=y
CONFIG_KERNEL_ELF=y
# Additional Block Devices
```

```
CONFIG_BLK_DEV_LOOP=y
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_INITRD=y
CONFIG_PARIDE_PARPORT=y
# networking options
CONFIG_packet=y
CONFIG_FIREWALL=y
CONFIG_FILTER=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_IP_FIREWALL=y
CONFIG_IP_MASQUERADE=y
# protocol-specific masquerading support will be built as modules.
CONFIG_IP_MASQUERADE_ICMP=y
# protocol-specific masquerading support will be built as modules.
CONFIG_IP_MASQUERADE_MOD=y
CONFIG_IP_MASQUERADE_IPAUTOFW=y
CONFIG_IP_MASQUERADE_IPPOREAL TIMEFW=y
CONFIG_IP_MASQUERADE_MFW=y
CONFIG_IP_router=y
CONFIG_IP_ALIAS=y
# network device support
CONFIG_NETDEVICES=y
CONFIG_PPPOX=y
# Ethernet (10 or 100Mbit)
CONFIG_NET_ETHERNET=y
```

```
# CONFIG_BMAC is not set
  CONFIG_PPP=y
# Filesystems
  CONFIG_MINIX_FS=y
  CONFIG_PROC_FS=y
  CONFIG_DEVPTS_FS=y
  CONFIG_EXT2_FS=y
# MPC8xx CPM Options
  CONFIG_SCC_ENET=y
  CONFIG_SCC1_ENET=y
#CONFIG_BMAC is not set
  CONFIG_PPP=y
#Filesystems
  CONFIG_MINIX_FS=y
  CONFIG_PROC_FS=y
  CONFIG_DEVPTS_FS=y
  CONFIG_EXT2_FS=y
#MPC8xx CPM Options
CONFIG_SCC_ENET=y
CONFIG_SCC1_ENET=y
```

자료 2 CPLD의 소스

```
--      Module      : linuxrouter.vhd
--      Title       : Main Decoder for MPC860EN-50Mhz
--      Part no.    : XC9536-7VQ44C
--      Device      : Xilinx
```

```
--          Modification Record
--                  2001.02.25          Created by s.s kim
library IEEE;
library SYNOPSYS;
use IEEE.std_logic_1164.all;
use SYNOPSYS.attributes.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

ENTITY linuxrouter IS PORT (
  pclkout                    : IN STD_LOGIC;
  clk10hz                    : IN STD_LOGIC;
  poreset                    : IN STD_LOGIC;
  cs1,cs7                    : IN STD_LOGIC;
  paddr4, paddr5, paddr6, paddr7, paddr23: IN STD_LOGIC;
  pts,pta,prw                : IN STD_LOGIC;
  hreset                : IN STD_LOGIC;
  lporeset                          : BUFFER STD_LOGIC;
  ltstart               : BUFFER STD_LOGIC;
  ds                         : BUFFER STD_LOGIC;
  as                         : BUFFER STD_LOGIC;
  cs                         : BUFFER STD_LOGIC;
  rw                           : BUFFER STD_LOGIC;
  run_led                    : BUFFER STD_LOGIC;
  fail_led                   : BUFFER STD_LOGIC;
  w1_txled                   : BUFFER STD_LOGIC;
```

```
  w1_rxled                    : BUFFER STD_LOGIC;
  w2_txled                    : BUFFER STD_LOGIC;
  w2_rxled                    : BUFFER STD_LOGIC);


END linuxrouter ;


ARCHITECTURE cpu_arch OF linuxrouter IS
        SIGNAL  paddr4_7            : STD_LOGIC_VECTOR(3 DOWNTO
0);
        SIGNAL  watch_count         : STD_LOGIC_VECTOR(5 DOWNTO
0);
        SIGNAL  watch_enable,watch_strobe          : STD_LOGIC;
        SIGNAL  hw_reset,synreset,clk5hz,hreset_f  : STD_LOGIC;
        SIGNAL  w1_tx,w1_rx,w2_tx,w2_rx            : STD_LOGIC;
        SIGNAL  w1_tx_f,w1_rx_f,w2_tx_f,w2_rx_f    : STD_LOGIC;
BEGIN
        paddr4_7(3) <= paddr4;
        paddr4_7(2) <= paddr5;
        paddr4_7(1) <= paddr6;
        paddr4_7(0) <= paddr7;
-- *****:       RTC  DAllAS DS12887 Selection: PROCESS        :*****
                as <= '1' WHEN ( cs1 = '0' AND paddr23 = '1')
                      ELSE '0';
                ds <= '1' WHEN ( cs1 = '0' AND paddr23 = '0')
                      ELSE '0';
```

```
--*****      Poreset Syncronization : PROCESS          *****--
        PROCESS
          BEGIN
               wait until pclkout = '1';
                        synreset <= poreset;
                        lporeset <= poreset;
        END PROCESS;
--*****  Clk 2 times divided : PROCESS                  *****--
        PROCESS
          BEGIN
               wait until clk10hz = '1';
                        clk5hz <= NOT clk5hz;
        END PROCESS;
--*****          Watchdog Counting : PROCESS          *****--
PROCESS (clk5hz,synreset,watch_strobe,watch_enable)
  BEGIN
IF(synreset = '0' OR watch_strobe = '0' OR watch_enable = '1') THEN
                        watch_count <= "000000";
               ELSIF (clk5hz'EVENT AND clk5hz = '1') THEN
                        watch_count <= watch_count+1;
               END iF;
 END PROCESS;
----*****          Watchdog,   Hw_reset : PROCESS          *****--
PROCESS
 BEGIN
  wait until pclkout = '1';
```

```
IF(synreset = '0' OR (synreset = '1' AND prw='0' AND pta='1'
            AND cs7 = '0' AND paddr4_7 = "0000") ) THEN
                        watch_strobe <= '0';
                ELSE
                        watch_strobe <= '1';
                END IF;
IF( synreset = '0' OR (synreset = '1' AND prw='0' AND pta='1'
            AND cs7 = '0' AND paddr4_7 = "0010") ) THEN
                        watch_enable <= '1';
ELSIF (synreset = '1' AND prw='0' AND pta='1' AND cs7 = '0'
                        AND paddr4_7 = "0001") THEN
                        watch_enable <= '0';
END IF;
IF(synreset = '0') THEN
                        hw_reset <= '1';
ELSIF( (synreset = '1' AND prw = '0' AND pta = '1' AND cs7 = '0'
                AND paddr4_7 = "1111") OR (watch_count="111111"))
                THEN  hw_reset <= '0';
                END IF;
        END PROCESS;
--*****          Ltstrart Signal : PROCESS          *****--
        PROCESS(synreset,hw_reset,pts)
            begin
                IF( synreset = '1' AND hw_reset = '0' ) THEN
                        ltstart <= '0';
                ELSE
```

```
                    ltstart <= pts;
                END IF;
    END PROCESS;
--*****              wan LED Selection              *****--
    w1_tx <= '0' WHEN ( synreset = '1' AND prw='0' AND pta='1'
                    AND cs7 = '0' AND paddr4_7 = "0100" ) ELSE
                '1';
    w1_rx <= '0' WHEN ( synreset = '1' AND prw='0' AND pta='1'
                    AND cs7 = '0' AND paddr4_7 = "0101" ) ELSE
                '1';
    w2_tx <= '0' WHEN ( synreset = '1' AND prw='0' AND pta='1'
                    AND cs7 = '0' AND paddr4_7 = "0110" ) ELSE
                '1';
    w2_rx <= '0' WHEN ( synreset = '1' AND prw='0' AND pta='1'
                    AND cs7 = '0' AND paddr4_7 = "0111" ) ELSE
                '1';
--*****              wan LED Flags, hreset_f        *****--
    w1_tx_f <='1' WHEN ( ( synreset = '1' ) and ( (w1_tx='0')
      OR (w1_txled = '1' and w1_tx_f = '1') ) )   ELSE
            '0';
    w1_rx_f <='1' WHEN ( ( synreset = '1' ) and ( (w1_rx='0'
             OR (w1_rxled = '1' and w1_rx_f = '1') ) )   ELSE
            '0';
    w2_tx_f <='1' WHEN ( ( synreset = '1' ) and ( (w2_tx='0')
             OR (w2_txled = '1' and w2_tx_f = '1') ) )   ELSE
            '0';
```

```
        w2_rx_f <='1' WHEN ( ( synreset = '1' ) and  ( (w2_rx='0'
                    OR (w2_rxled = '1' and w2_rx_f = '1') ) )   ELSE
                   '0';
        hreset_f <= '1' WHEN ( hreset = '0'
                          OR (fail_led = '1' and hreset_f = '1') ) ELSE
                   '0';
--*****            Fail_LED,  wan LED : PROCESS       *****--
        LED: PROCESS
            BEGIN
                wait until clk5hz = '1';
                    IF ( poreset = '0' OR hreset_f = '1') THEN
                            fail_led <= '0';
                        ELSE
                            fail_led <= '1';
                        END IF;
                        IF ( w1_tx_f = '1') THEN
                            w1_txled <= '0';
                        ELSE
                            w1_txled <= '1';
                        END IF;
                        IF ( w1_rx_f = '1') THEN
                            w1_rxled <= '0';
                        ELSE
                            w1_rxled <= '1';
                        END IF;
                        IF ( w2_tx_f = '1') THEN
```

```
                    w2_txled <= '0';
                ELSE
                    w2_txled <= '1';
                END IF;
                IF ( w2_rx_f = '1') THEN
                    w2_rxled <= '0';
                ELSE
                    w2_rxled <= '1';
                END IF;
        END PROCESS;
--******                Run_LED                *****--
        run_led <= not fail_led;
      rw  <= prw;
       cs <= cs1
END cpu_arch;
```

자료 3 로드밸런싱 설정문

```
#!/bin/sh
. /setup/flb100.conf

ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 dev lo
#       eth0 setup
if `ifconfig | grep -q ^eth0`
then
        ifconfig eth0 down
```

```
        ifconfig eth0:0 down
fi
ifconfig eth0 $ETH0_IP netmask $ETH0_MASK
#       DHCP Server
if [ "$ETH0_IP" = "192.168.1.1" ]; then
        sed s/RANGE1/192.168.1.100/ /etc/dhcpd.conf.org | \
        sed s/RANGE2/192.168.1.250/ > dhcpd.conf.tmp
        mv -f dhcpd.conf.tmp /etc/dhcpd.conf
        if [ -n "$DNS" ]; then
                sed s/168.126.63.1,168.126.63.2/$DNS/ /etc/dhcpd.conf >
dhcpd.conf.tmp
                mv -f dhcpd.conf.tmp /etc/dhcpd.conf
        fi
        dhcpd eth0
else
        if [ -n "$RANGE1" ] && [ -n "$RANGE2" ]; then
                sed s/RANGE1/$RANGE1/ /etc/dhcpd.conf.org | \
                sed s/RANGE2/$RANGE2/ | \
                sed s/192.168.1.1/$ETH0_IP/ > dhcpd.conf.tmp
                mv -f dhcpd.conf.tmp /etc/dhcpd.conf
                if [ -n "$DNS" ];then
                        sed s/168.126.63.1,168.126.63.2/$DNS/ /etc/dhcpd.conf >
dhcpd.conf.tmp
                mv -f dhcpd.conf.tmp /etc/dhcpd.conf
                fi
                dhcpd eth0
```

```
        else
                rm -f /etc/dhcpd.conf
        fi
fi
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "0" > /proc/sys/net/ipv4/conf/all/send_redirects
echo "0" > /proc/sys/net/ipv4/conf/eth0/send_redirects
ETH0_NET=`route -n -A inet | grep eth0 | awk '{print $1}'`
if [ -n "$SBRP_GW" ]; then
/sbrp.cgi FLUSH
/sbin/ipchains -A input -p tcp -s $ETH0_NET/$ETH0_MASK -d 0/0 20:21 -m
2
   if [ -z "$WWW" ]; then
        /sbin/ipchains -A input -p tcp -s $ETH0_NET/$ETH0_MASK -d 0/0
80 -m 2
   fi
if [ -n "$NAT" ]; then
        if [ "$ETH1_IP" = "ADSL" ]; then
                /sbin/ipchains -A forward -i ppp0 -j MASQ
                /sbin/modprobe mssclampfw
        else
                /sbin/ipchains -A forward -i eth1 -j MASQ
        fi
        modprobe ip_masq_ftp
        modprobe ip_masq_irc
        modprobe ip_masq_raudio
```

```
        # dialpad
        /sbin/ipmasqadm autofw -A -v -u -r udp 51200 51201 -c tcp 7175
        /sbin/ipmasqadm autofw -A -v -u -r tcp 51210 51210 -c tcp 7175
        #wow call
        # starcraft
        /sbin/ipmasqadm autofw -A -r udp 6112 6112 -c tcp 6112
fi
ifconfig eth0:0 192.168.1.254 netmask 255.255.255.0
```