논문번호 05-01-07

임베디드 리눅스를 이용한 트래픽 미터링 시스템 설계

Design of Traffic Metering System using Embedded Linux

李興宰*, 田喜鈐*, 崔眞圭*, 李圭皓**
Heung-Jae Lee*, Hee-Jin Jeon*, Jin-Kyu Choe*, Kyou-Ho Lee**

요 약

네트워크 트래픽의 증가와 다양한 멀티미디어 응용 서비스가 등장함에 따라 네트워크의 자원을 효과적으로 관리, 운용 및 서비스 품질의 향상을 위해 네트워크 트래픽을 실시간으로 감시하고 분석하는 것이 요구되고 있다. 기존의 소프트웨어 기반의 트래픽 측정 방식으로는 광대역 트래픽의 정확한 측정이 어려움으로 캡처, 헤더 매칭 등의 기능을 효율적으로 수행하는 미터 구조에 대한 연구가 필요하게 되었다.

따라서, 본 논문에서는 좀 더 효율적인 패킷 캡처를 위하여 임베디드 리눅스 기반의 하드웨어 미터링 시스템을 설계 및 구현하였다. 또한 10 기가비트 네트워크의 패킷 캡처에 대해 AweSim을 이용하여 시뮬레이션 모델을 작성하고 시뮬레이션을 통해 요구되는 시스템 버스와 메모리 대역폭을 분석하였다.

Abstract

Increasing network traffic and multimedia application services need realtime analysis of network traffic for improvement of QoS and effective management of network resource. Because difficulty of measurement based on software method, study of meter architecture for efficient capture function is necessary.

Therefore we design and implement hardware metering system for efficient packet capture using embedded linux. And we analyze required bandwidth of system bus and memory for 10Gbps traffic through simulation.

Keyword: traffic measurement, meter, packet capture, network monitoring

Ⅰ. 서 론

네트워크 트래픽의 증가와 다양한 멀티미디어 응용 서비스의 등장은 네트워크에서 자원 부족과 폭주(congestion)를 유발하게 되어, 지연 또는 시간에 있어서 엄격한 품질 (Quality of Services, QoS) 보장이 요구되는 멀티미디어 관련 응용 서비스뿐만 아니라 전통적인 데이터 기반의 응용 서비스의 제공에 심각한 영향을 주게 되었다. 이에 따라, 네트워크의 자원을 효율적으로 관리, 운용함으로써, 서비스 품질을 향상시키기 위하여 네트워크 트래픽을 실시간으로 감시하고 분석하는 것이 요구되고 있다. 기존의 소프트웨어 기반의 트래픽 측정 방식으로는 광대역 트래픽을 정확히 측정할 수 없다. 따라서 고속 IP망에 적합한 효율적인 하드웨어 미터링 구조에 대한 연구가 필수적으로 수행되어야 한다. 트래픽을 캡처하고 분석하기위한 사용되고 있는 트래픽 분석 소

* 韓南大學校 情報通信멀티미디어工學部(School of Info. Tech. & Multimedia Eng., Hannam Univ.)
** 仁濟大學校 電子情報通信工學部(School of Info. Electronic and Telecommunication Engineering, Inje Univ.)
接受日:2005年 3月 31日, 修正完了日:2005年 12月 26日

프트웨어로 사용되는 MRTG가 있으며, Ciso 라우터로부터 발생되는 트래픽의 정보를 분석하여 주는 소프트웨어로는 FlowScan, Ciso NetFlow가 있다[9][10][11]. MRTG는 소프트웨어 만으로 구성이 되고 쉽게 사용할 수있는장점이 있지만 미터링 포인트에 각각의 서버를 위치시켜 원하는 패킷의 캡처를 수행하는 것이 필수 있다. FlowScan, NetFlow 역시 Ciso 라우터가 아닌 네트워크에서는 사용이 불가능 하다. 따라서 비용과 설치의 복잡
성으로 인하여 제한된 미터링 포인트만을 가지게 될 것이다.

본 논문에서는 시스템 운영체제로 임베디드 리눅스를 채택하였으며, 고속 패킷 캡처를 위해 패킷 처리 시간이 개선되도록 헤더 추출 시스템과 분석 시스템으로 분산된 구조로 하드웨어 미터링 시스템을 설계 및 구현하였다. 또한, 10 기가비트 네트워크에서 패킷 캡처에 대한 모델을 기술하고, 시뮬레이션을 통해 요구되는 시스템 버스와 메모리 속도에 대해 분석하였다.

본 논문의 구성은 2장에서는 실시간 트래픽 흐름 측정 구조와 고속 칩-to-칩 인터페이스에 대해 분석하고, 3장에서는 임베디드 리눅스를 이용한 미터링 시스템의 설계 및 구현에 대하여 상세히 기술하였다. 4장에서는 패킷 캡처 모델과 시뮬레이션 결과를 분석하였으며, 5장에서 이상의 결과들을 정리하였다.

II. 실시간 트래픽 측정

1. 실시간 트래픽 흐름 측정 구조

실시간 트래픽 흐름 측정은 IETF에서 표준화 중에 있으며, 네트워크 상의 트래픽을 실시간으로 수집하여 흐름 단위로 정보를 분석한다[1,2]. 실시간 트래픽 흐름 측정 구조는 매니저, 미터, 미터 리더, 분석 프로그램으로 구성되어 있다[3,4]. 매니저는 미터의 구성과 미터 리더를 제어하는 어플리케이션이다. 매니저는 미터와 미터 리더를 상황에 따라 적절한 제어와 구성을 통해 관리 감독한다. 미터는 네트워크에 지정된 미터링 포인트(metering point)에서 트래픽 흐름 데이터를 수집하기 위한 장치이다. 그림 1는 미터의 구조를 나타낸 것이다. 미터 리더는 매니저에 의해 정의된 간격마다 미터의 흐름 테이블에 저장된 흐름 데이터를 수집한다. 미터 리더는 미터로부터 흐름 데이터를 수집하여 파일을 생성하는 데 이것이 흐름 데이터 파일(Flow Data File)이다. 미터 리더는 이렇게 생성된 흐름 데이터 파일을 분석 프로그램에 전송한다. 분석 프로그램은 미터 리더로부터 수신된 데이터를 분석하여 사용자가 원하는 형태로 출력한다.

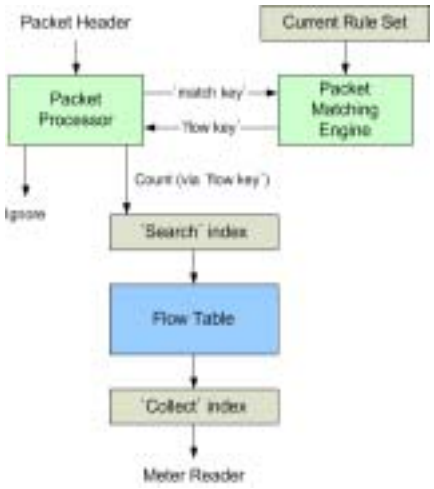


그림 1. 미터 구조
Fig. 1. Architecture of meter

2. 고속 칩-to-칩 인터페이스

트래픽 흐름 측정을 위한 미터링 시스템의 단일칩화 하기 위해서는 네트워크 라인 인터페이스와 미터링 시스템 사이의 고속 연결이 중요하다. 10기가로 입력되는 데이터는 직/병렬화기를 통과하는 과정이 필수이다. 이때 병렬화되어진 데이터 버스와 연결되는 인터페이스는 고속 처리가 필수이다. 고속의 칩간 인터페이스의 문제를 해결하기 위한 기술의 하나로 10 기가비트의 네트워크 시스템에서 라인 인터페이스와 시스템 인터페이스 사이의 고속 연결 문제를 해결하기 위해 래티스사의 FPSC (Field Programmable System Chip)에 대해 기술한다. 래티스사의 ORLI10G FPSC는 임베디드 ASIC 코어 안에 XSBI(10 Gigabit Sixteen-Bit Interface)를 통합시킨 것으로 라인과 시스템 사이에 위치하는 16비트 인터페이스이다[5]. FPSC를 사용하는 이유는 설계 시 보드 공간을 절약하고 트레이스 라우팅과 복잡성을 줄일 수 있는 소자이기 때문이다.

그림 2와 같이 10Gbps의 데이터를 직/병렬화를 수행하며 다중 데이터 라인으로 구성되어 있는 LVDS(Low Voltage Differential Signaling) I/O를 사용한다. 기본 어플리케이션에서 각 데이터 라인의 최하 동작 속도는 622Mbps이다. 본 논문에서는 LVDS I/O를 이용한 16비트 송수신 데이터 라인, 1개 제어 수신 라인 그리고 각 방향으로 데이터와 함께 포워딩 되는 별도의 송수신 LVDS 클럭을 갖춘 인터페이스로 네트워크와 시스템을 연결하는 방식을 사용하여 10 기가비트 네트워크에서 패킷 캡처를 수행하는 미터링 시스템에 대해 시뮬레이션 하였다.

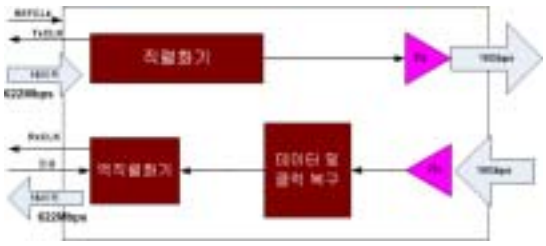


그림 2. 10Gps 라인 인터페이스
Fig. 2. 10Gps Line Interface

III. 미터링 시스템 설계

1. 미터링 시스템 구성

미터링 시스템을 위한 하드웨어로서는 PXA255 프로세서(400Mhz)와 10Mbps 이더넷 사양을 가지는 임베디드 시스템을 이용하여 구현을 하였다. 또한 소프트웨어로는 임베디드 리눅스 커널 2.4.19를 포팅 하였으며 패킷 캡처를 위한 소프트웨어로는 libpcap 0.8.3 버전을 사용하였다. PC의 리눅스와 임베디드 리눅스와의 교차 개발 환경을 구축을 하여 타겟으로 하는 하드웨어와 매칭이 될 수 있도록 리눅스 커널을 패치 및 변경하여 컴파일을 하였다. 패킷 캡처를 위한 libpcap 0.8.3을 임베디드 리눅스에서 적절히 동작할 수 있도록 라이브러리를 수정하였으며 libpcap을 실제 동작 시키게 하는 보다 상위의 소프트웨어를 개발 하였다. 휴대 가능한 임베디드 리눅스를 탑재한 미터링 시스템을 가지게 되면 비용과 설치의 복잡성은 현저히 줄어들게 된다. 그러나 임베디드 시스템만을 이용하여 패킷을 캡처하게 되면 임베디드 리눅스가 가지는 제한된 메모리로 제한된 프로세서의 성능 등으로 따른 한계상황에 직면하게 된다. 따라서 미터링 시스템이 캡처한 데이터는 최소한의 수정만을 거쳐 분석 시스템으로 보내는 분산된 방식을 선택하였다.

이와 같은 방식을 사용하게 됨으로서 원하는 미터링 포인트는 분석 시스템을 가지고 있는 네트워크가 아닌 또 다른 네트워크에 미터링 포인트에 미터를 둘 수 있는 장점을 가지고 있으며 또한 비용과 휴대 가능하다는 것을 큰 장점으로 가지고 있다고 할 수 있다.

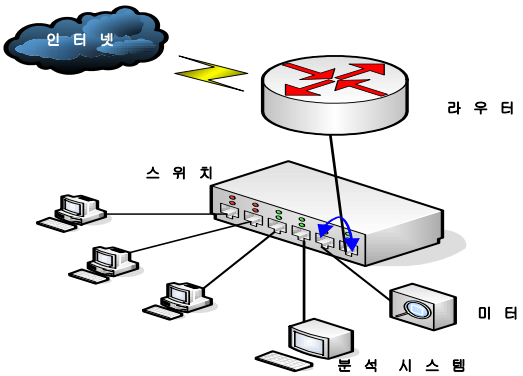


그림 4. 미터링 시스템과 분석 시스템
Fig. 4. Metering System and Traffic Analyzer

그림 5. 구현된 미터링 하드웨어
Fig. 5. Implemented metering hardware

그림 5는 구현된 미터링 하드웨어의 그림이고, 그림 4는 미터링 시스템과 분석 시스템의 네트워크 구조를 보여주는 그림이다.

2. libpcap 라이브러리

libpcap(Portable Packet Capturing Library)는 패킷 캡처를 위한 운영 체제에 종속되지 않은 라이브러리이다. libpcap은 리눅스, 유닉스, 윈도우등의 운영 체제에 관계없이 사용할 수 있는 API를 제공한다. 따라서 어떠한 운영체제의 어플리케이션이라도 libpcap의 API를 이용하여 원하는 어플리케이션을 제작할 수 있다는 것을 의미한다.

libpcap 라이브러리에 의해서 캡처가 되는 단계는 디멀티플렉싱 과정을 거치 이전의 패킷이다. 디멀티플렉싱이 되지 않은 완전한 구조체의 포인터를 넘겨받기 때

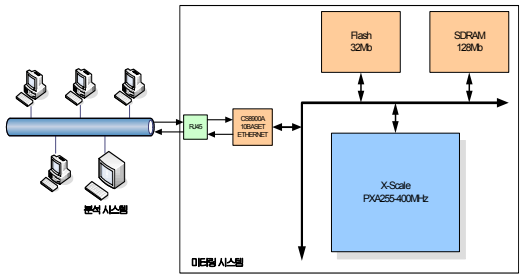
문에 각 계층의 데이터를 읽어 올 수 있다.

그림 6 포트 80으로 들어오는 패킷 캡처 화면
Fig. 6 Packet dump in port 80

그림 6은 libpcap을 이용하여 프로토타입으로 작성한 어플리케이션에 의해서 특정 포트인 80 포트에 도착하는 패킷만을 캡처하는 결과를 보여주는 그림이다. 어플리케이션은 IP 버전, 헤더 길이, TTL, 식별자과 같은 IP 헤더의 모든 내용을 표시할 수 있으며 실제의 데이터를 덤프하여 주는 기능을 가지고 있다.

실제 미터링 시스템을 위한 소프트웨어는 속도를 위하여 현재 상황에 대한 어떠한 데이터도 보여주지 않는다. 또한 분석 시스템의 소프트웨어에 의한 정의에 따르는 수동적인 프로그램으로 작성이 되었다.

그림 7은 미터링 시스템의 소프트웨어와 하드웨어 구조를 보여주는 그림이다. 하드웨어는 10M 이더넷을 가지며 128Mbyte의 SDRAM을 가진 PXA255 시스템이다. 미터링 시스템의 속도 향상을 위하여 각 패킷에 대한 흐름 테이블을 생성하지 않는다. 또한 패킷 전송 시 생길 수 있는 오버헤드를 감소하기 시키기 위하여 버퍼의 한 개의 용량은 1024 바이트로 설정하였다. 패킷 매칭 엔진은 미터링 분석 시스템에 의해서 정의가 되어진다. 미터링 시스템의 패킷 분석 프로그램은 분석 시스템에 의해서 정의된 룰 셋을 수신 받은 후에 패킷 매칭 엔진에 적용을 한다. 미터링 시스템은 32 바이트의 데이터를 생성하여 1024 바이트 버퍼에 저장을 한다. 32개의 매칭 데이터가 생성이 될 때 어플리케이션은 패킷 분석 프로그램에 TCP를 이용하여 전송 하는 방식으로 구현이 되어 있다. 이때 생성되어 분석 시스템으로 보내지는 데이터는 그림 8과 같다.



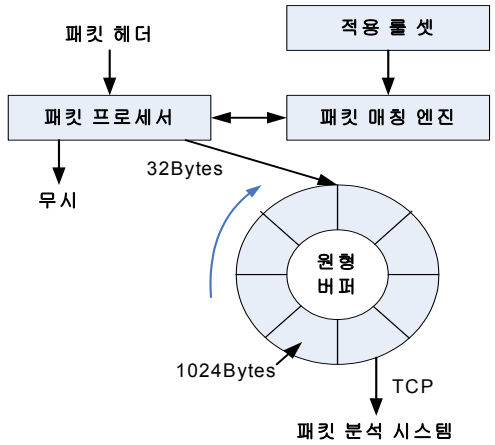


그림 7 미터링 시스템 구조
Fig. 7 Structure of metering System

그림 8 가공된 데이터 포맷
Fig. 8 Processing Data format

2. 분석 시스템의 구현

원하는 미터링 포인트에서 위치한 원격의 미터링 시스템에서 캡처한 데이터는 디멀티플렉싱을 거치지 않은

데이터 이다. 또한 미터링 시스템으로 얻어진 데이터를 미터링 시스템에서 직접 분석하는 것은 미터링 시스템이 가진 성능의 문제가 발생할 수 있다. 따라서 미터링 시스템에서 캡처한 데이터는 분석 시스템으로 보내져 데이터를 분석하게 된다.

미터링 시스템에서 데이터를 받아 분석하는 분석 시스템은 Intel Pentium 2.4GHz, 512MB 램과 100Mbps 이더넷 사양의 PC를 사용하였다. 분석 시스템 프로그램은 Visual C++ 6.0과 MFC를 사용하여 구현하였으며, 수신된 데이터를 분석하여 사용자가 알아보기 쉽게 실시간으로 데이터를 도식화하여 출력하도록 하였다. 분석 소프트웨어는 먼저 미터링 시스템과의 연결과정을 거쳐야만 한다. 분석 소프트웨어가 미터링 시스템과 연결되는 과정에서 미터링 시스템이 취해야 할 행위를 정의하게 된다. 그림 9는 분석 시스템의 분석 프로그램의 설정 화면을 보여주는 그림이다. 그림 9에서 먼저 미터링 시스템이 존재하는 IP의 주소를 설정하고, 3에 정의된 룰에 따른 데이터를 얼마만큼 받을 것인가를 2에 의해서 결정한 후 4의 Make Ruleset 버튼을 이용하여 미터링 시스템과의 초기화 과정을 거치게 된다.

그림 9 분석 프로그램 설정 화면
Fig. 9 Configure screen of analyzer program

그림 9에 의해서 설정되고 분석 프로그램에서 의해서 미터링 시스템과의 제어 패킷들은 그림 10에 나타난 로그 화면에 상세히 보여주게 된다. 미터링 분석 시스템 역시 시스템의 부하를 최소화하는 것이 중요하므로 로그화면을 보여주는 것을 비활성 할 수 있다. 로그 화면에서는 미터링 시스템으로부터 받은 자세한 정보를 보여주며 그림 11과 12는 미터링 시스템으로부터 받은 데이터를 분류 하여 차트 형식으로 보여준다. 그림 11과 그림 12의 차트 형식은 각 프로토콜에 따라 분류를 하여 퍼센티지 형식으로 보여준다. 이때 보여 지는 퍼센티지는 상대적인 수치로서, 총 도착한 패킷의 몇 퍼센트임을 보여주는 것이다.

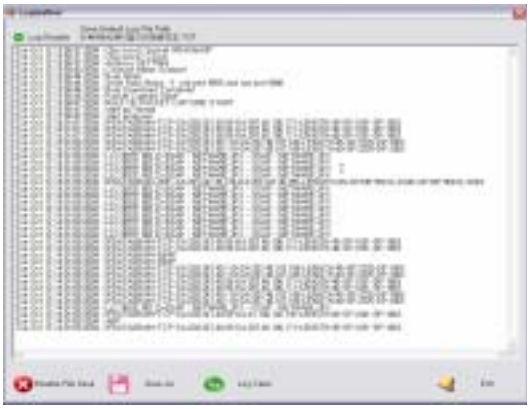


그림 10. 분석 시스템 로그 화면
Fig. 10. Log screen of analyzer

그림 11. 바 차트 화면
Fig. 11. Bar chart screen

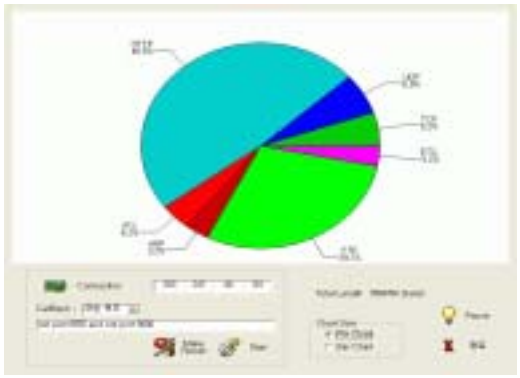


그림 12. 파이 차트 화면
Fig. 12. Pie chart screen

3. 분석 시스템의 검증

본 논문에서 작성한 분석 시스템과 미터링 시스템이 적절한 동작을 하는가에 대한 검증이 필요하다. 시스템 검증을 위하여 네트워크에 32바이트의 각종 프로토콜 트래픽 64,000개 트래픽을 발생하여 구현한 미터링 시스템과 분석 시스템이 적절히 동작하는 것을 검증 하였다. 패킷 발생을 위하여 패킷 발생 프로그램은 인게이지 시큐어러티(Engage security)사의 rafaleX를 사용하였다. 패킷 생성기는 포트 80인 HTTP 패킷, UDP, IPX, FTP, ETC을 발생하게 설정하였다. 그림 13은 본 논문에서 구현한 분석 시스템이 작성한 캡처 데이터를 보여주는 그림이다. 패킷 발생기에 의해서 발생한 패킷을 100% 캡처한 것을 보여주며, 총 수신한 페이로드가 2048000 바이트임을 보여준다.

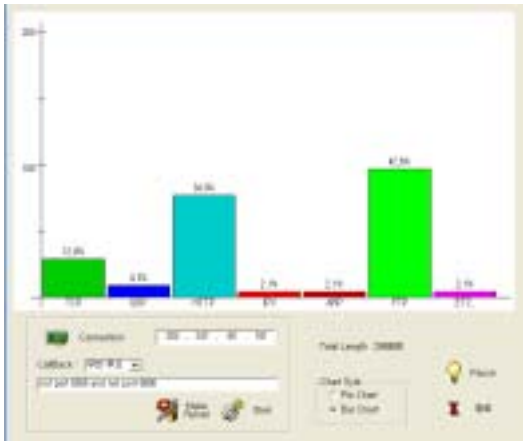


그림 13 분석 프로그램 패킷 분석 화면
Fig. 13 Packet analysis screen of analyzer program

Ⅳ. 시뮬레이션 모델링 및 분석

1. 시뮬레이션 모델링

본 논문에서 수행한 트래픽 모니터링을 위하여 구현된 하드웨어와 소프트웨어는 10 메가비트 네트워크에서 패킷 캡처를 수행 하였다. 본 논문에서 구현된 시스템을 고속의 네트워크에서 동작하는 하드웨어를 개발하기 위해서는 시스템의 스펙을 결정해야 한다. 따라서 시뮬레이션은 본 절에서는 구현되어진 시스템을 10기가 네트워크에 적용하기 위해서 필요한 시스템 스펙을 시뮬레이션 한다. 10 기가비트 네트워크에서 패킷 캡처 수행 시 요구되는 시스템 버스와 메모리의 대역폭을 분석하기 위해 AweSim을 이용하여 시뮬레이션을 하였다. 그림 14는 10기가비트 네트워크에서 패킷 캡처를 위한 시뮬레이션 모델의 블록도이다.

10Gbps의 전송 속도의 네트워크와 미터링 시스템의 인터페이스는 래티스사의 LVDS I/O 방식의 라인 인터페이스로 고속 연결 방식을 사용하였으며, 패킷 생성은 10Gbps일 때의 최소 IPG 9.6ns를 적용하여 생성하였다. 또한, 최소 패킷 크기인 40Bytes인 패킷이 최소 IPG 간격으로 들어온다고 가정하였다. CPU는 다음 패킷이 들어오기 전에 전 패킷을 처리할 수 있는 능력이 있다고 가정하였으며, CPU에서 처리된 패킷은 64비트 시스템 버스를 통해 메모리에 저장된다. 따라서 9.6ns마다 40Bytes의 패킷을 생성하여 입력하는 방식으로 최악 조건인 경우에 대해 시뮬레이션 하였다.

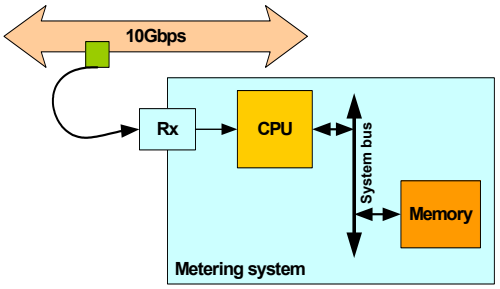


그림 14. 시뮬레이션 모델 블록도
Fig. 14. Block diagram of simulation model

2. 분석

앞 장에서 서술한 기본 시뮬레이션 모델에 의해 10기가비트 네트워크에서 패킷 캡처를 수행할 때 요구되는 시스템 버스와 메모리 대역폭을 분석하기 위해 시스템 속도와 메모리 대역폭의 변화에 따른 패킷 손실을 시뮬레이션 하였다. 그림 15는 시스템 버스 대역폭에 따른 패킷 손실의 변화를 도시한 것이다.

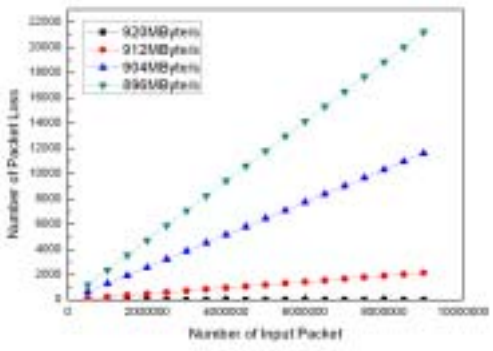


그림 15. 시스템 버스 대역폭에 따른 패킷 손실
Fig. 15. Packet loss vs. system bus bandwidth

시스템 버스 대역폭이 920MBytes/s 미만일 경우 시스템 버스의 지연으로 인해 패킷을 캡처하지 못하고 손실이 일어나는 것을 확인할 수 있었다. 10 기가비트 네트워크에서 패킷 캡처를 수행할 경우 시스템 버스가 920MBytes/s 이상의 대역폭이 요구되는 것으로 분석되었다. 데이터 전송의 크기가 64비트일 때 버스 클럭이 115MHz 이상이 되어야 하고 메모리 엑세스 시간은 최소 8.695nsec 이하가 되어야 한다. 따라서 메모리 엑세스 시간에 맞추려면 버스 크기를 128비트 이상이 되어야 한다. 표 1은 시스템 버스 처리량이 920MBps일 경우 시스템 버스 데이터 전송 크기와 버스 클럭, 메모리 엑세스 타임을 나타낸 것이다.

표 1. 시스템 버스 처리량(920MBps일 경우)
Table 1. Data processing in system bus (case 950 MBps)

시스템 버스	버스 클럭	메모리 엑세스 시간
32bits	203MHz	4.348nsec
64bits	115MHz	8.695nsec
128bits	57.5MHz	17.391nsec

V. 결 론

본 논문에서는 기존의 소프트웨어 기반의 트래픽 측정 방식이 아닌 임베디드 리눅스 기반의 하드웨어 미터링 시스템을 설계하였다. 10Mbps의 네트워크에서 패킷 캡처를 수행하는 임베디드 리눅스 미터링 시스템과 분석 시스템을 설계 및 구현하였다.

구현된 미터링 시스템은 기존 미터 구조에서 패킷 캡처 부분과 패킷 분석 부분을 분산시켜 설계함으로써 패킷 캡처의 속도가 개선되는 장점을 지닌다. 또한 본 논문에서는 10 기가비트 네트워크에서 패킷 캡처를 수행시 요구되는 시스템 버스와 메모리 속도에 대해 시뮬레이션 하였다.

시뮬레이션 결과, 10 기가비트 네트워크에서 모든 패킷의 원활한 캡처를 위해서는 920MBytes/s 이상의 시스템 버스와 460MBytes/s 이상의 메모리 대역폭이 요구되는 것으로 분석되었다. 향후 광대역 트래픽에 적합한 고속의 하드웨어 미터링 시스템을 SoC(System on chip)화 방안에 대한 연구가 필요하다.

참 고 문 헌

[1] S. Handelman, S. Stibler, N. Brownlee and G. Ruth, “RTFM : New Attributes for Traffic Flow Measurement,” RFC 2724, October, 1999.
[2] N. Brownlee, “RTFM : Applicability Statement,” RFC 2721, October 1999.
[3] N. Brownlee, “Traffic Flow Measurement : Experiences with NeTraMet,” RFC 2123, March 1997.
[4] N. Brownlee, C. Mills, and G. Ruth, “Traffic Flow Measurement : Architecture,” RFC2722, October 1999.
[5] Lattice Semiconductor Corporation, http://www.latticesemi.com
[6] TCPDUMP/LIBPCAP, http://www.tcpdump.org
[7] Intel, *Intel PXA255 processor Developer's Manual*, March, 2003.
[8] Thmas Lindh, "A New Approach to Performance Monitoring in IP Networks Combin- ing Active and Passive Methods," Proc. of PAM2002 Workshop, Colorado, USA, Mar., 2002.
[9] MRTG, http://www.mrtg.org
[10] FlowScan, http://www.caida.org/tools/utilities/flowscan/index.xml/.
[11] CoralReef, http://anala.caida.org/CoralReef/Demos/cerfnet/link/.

저 자 소 개

이 홍 재 (학생회원)

1968년 2월 27일 생.
1998년 한남대학교 전자공학과 졸업(공학사)
2000년 한남대학교 대학원 전자공학과 졸업(공학석사)
2002년~현재 한남대학교 대학원 전자공학과(박사수료)
<주관심분야> Embedded system, Ad-hoc 네트워크

전 희 진 (정회원)

2003년 한남대학교 전자공학과 졸업(공학사)
2005년 한남대학교 대학원 전자공학과 졸업(공학석사)
<주관심분야> Embedded system

최 진 규 (정회원)

1980년 고려대학교 전자공학과 졸업(공학사)
1982년 고려대학교 대학원 전자공학과 졸업(공학석사)
1987년 고려대학교 대학원 전자공학과 졸업(공학박사)

1987년 9월-1990년 8월 대전공업대학 조교수
1999년-2000년 미국 Oregon State University 방문교수
1990년 8월 - 현재 한남대학교 정보통신멀티미디어공학부 교수
<주관심분야> 통신망 성능평가, 디지털시스템 설계, Embedded system

이 규 호 (정회원)

1980년 경북대학교 전자공학과 졸업(공학사)
1982년 경북대학교 대학원 전자공학과 졸업(공학석사)
1998년 The University of Gent, Belgium, 정보/컴퓨터 공학과(공학박사)

1986년-1988년 미국 AIT Inc. 연구원
1983년-현재 한국전자통신연구원(ETRI) 책임 연구원
<주관심분야> 고속인터넷 기술, IP 응용기술, 고성능 네트워크프로세서, 고성능 라우터 기술