# 네트워크 프로세서를 이용한 기가비트 패킷 헤더 수집기

최 판 안[†] · 최 경 희[††] · 정 기 현[†††] · 심 재 홍[†††]


## 요 약

본 논문에서는 기가비트 트래픽에서도 높은 패킷 헤더 수집률(packet header collection ratio)을 보이는 멀티프로세서(multi-processor), 멀티쓰레드(multi-thread)를 채용한 네트워크 프로세서 기반의 패킷 헤더 수집기를 제안한다. 제안 패킷 수집기는 기가비트 트래픽 패킷 헤더를 분리하여 여러 대의 100Mbps MAC 포트로 분산하여 전달할 수 있는 구조를 가지고 있다. 제안된 구조는 고속 트래픽 처리를 위해 독창적인 버퍼관리 기법과 프로세서간 부하 분산 기법을 사용하고 있으며, 충분한 실험을 통해 그 성능을 검증하였다.


# A Gigabit Rate Packet Header Collector using Network Processor

Pan-an Choi[†] · Kyung-hee Choi[††] · Gi-hyun Jung[†††] · Jae-hong Sim[††††]


## Abstract

This paper proposes a packet header collector, based on a network processor with multi-processor and multi-threads, that shows a high throughput on gigabit network. The proposed collector has an architecture to separate packets coming from gigabit network into headers and payloads, and distribute them to multiple 100Mbit MAC ports. The architecture hiring a unique buffer management method and load distribution strategy among multiple processors is evaluated empirically in depth.


**키워드 : 네트워크 프로세서(network processor), 패킷 헤더 수집기(packet header collector), 부하 분산(load distribution)**

## 1. 서 론

인터넷 사용자의 계속된 증가와 다양한 광 대역 접근 매체(broadband access media)의 발달로 트래픽의 양이 급속하게 증가하고 있다. 많은 양의 트래픽을 수용하기 위해서 인터넷 서비스 공급업체, 기업망, 또는 학내망들은 백본 랜으로 기가비트 이더넷[1]을 도입하고 있다. 이러한 기가비트 이더넷 망의 도입으로 네트워크 관리자들은 네트워크에 대한 다양한 정보를 얻기 위해서 고성능 인터넷 모니터링 시스템을 새로이 구축해야 할 필요가 생겼다. 네트워크의 감시는 보안과 직결된 사안이므로 망의 유지, 보수에 필수사항이기 때문이다.

네트워크 모니터링은 프로토콜 분석, 네트워크 사용 어카운팅(accounting), 침입 탐지, 통계 등과 같은 많은 응용들을 위한 기초적인 기술 중의 하나이며, 주요 목적은 네트워크 관리자의 의사 결정을 돕기 위해 네트워크로부터 필요한 정보를 얻는 것이다. 모니터링 시스템의 성능은 실제 소프트웨어의 알고리즘에 따라 많이 좌우되지만 또한 하드웨어로부터 상위 소프트웨어로 이어지는 구조의 효율적 디자인에 의해서도 많은 차이가 난다. 대부분의 모니터링 시스템은 범용 하드웨어 시스템을 기반으로 하기 때문에 이미 정형화되어 만들어져 있는 네트워크 인터페이스 카드(network interface card)를 사용하고, 높은 지연이 유발되는 I/O 버스를 사용할 수밖에 없다. 이것은 모니터링 시스템에서 중요한 기능 중 하나인 패킷 헤더 수집의 성능을 감소시키는 중요한 요인이 된다.

침입탐지 시스템(intrusion detection system)이나 방화벽(firewall)들은 고성능의 플랫폼에 탑재되어 사용됨에도 불구하고 여전히 기가비트 속도에서 모든 패킷 헤더를 수집, 분석하지 못하는 것으로 알려져 있다. 단일 프로세서의 처리능력 한계를 극복하기 위한 멀티프로세서(multiprocessor) 기반의 시스템이 도입되고는 있으나 프로그램의 효과적인 연동이 되지 않아 비례적인 수치가 나오지 않고 있다. 또 다른 방식인 패킷 헤더 수집과 모니터링 기능을 서로 다른 시스템에서 분산하여 실행하는 부하 분산 모니터링 시스템[2, 3]의 경우, 역할 분담을 통해 성능을 높일 수 있다는 장점은 있으나, 비용절감을 위해서 정형화된 소프트웨어에만 의존했기 때문에 성능 면에서 한계를 가지고 있다.

† 준 회 원 : 아주대학교 정보통신전문대학원
†† 정 회 원 : 아주대학교 정보통신전문대학원 교수
††† 정 회 원 : 아주대학교 전자공학부 교수
†††† 정 회 원 : 조선대학교 인터넷소프트웨어공학부 교수
논문접수 : 2004년 1월 17일, 심사완료 : 2004년 5월 11일

본 논문에서는 기가비트 속도 트래픽에서 우수한 성능의 인터넷 모니터링 시스템을 구성하기 위해 네트워크 프로세서를 이용한 새로운 패킷 헤더 수집기를 제안한다. 네트워크 프로세서는 프로그래밍이 가능한 유연성을 가지고 있으면서도 네트워크 데이터 처리를 위해 특화된 멀티프로세서(multiprocessor), 멀티쓰레드(multithread) 기반으로 동작하므로 저가의 고성능 시스템을 제작하는데 적합하다. 이를 이용하면 기가비트 트래픽 패킷 헤더를 분리하여 여러 대의 100Mbps MAC 포트로 분산하여 전달할 수 있어 기존 장비의 활용과 더불어 고속의 트래픽을 감당할 수 있게 된다.

본 논문의 구성은 다음과 같다. 2절에서 관련연구 및 연구 동기를 소개하고, 3절에서는 네트워크 프로세서를 이용한 패킷 헤더 수집기의 설계 및 구현에 대해 기술한다. 4절에서는 실험환경에 대해 기술하고 실험결과를 통해 제안시스템과 기존시스템의 성능을 비교 평가하며, 5절에서 결론을 맺는다.

## 2. 관련 연구 및 연구 배경

효율적인 네트워크 모니터링 시스템 및 패킷 수집 시스템 구축을 위한 많은 연구들이 수행되었다. 본 절에서는 이들을 소프트웨어에 의한 접근방법, 하드웨어에 의한 접근방법, 부하 분산 모니터링 시스템으로 나누어서 살펴보고, 본 연구의 배경에 대해 논의하고자 한다.

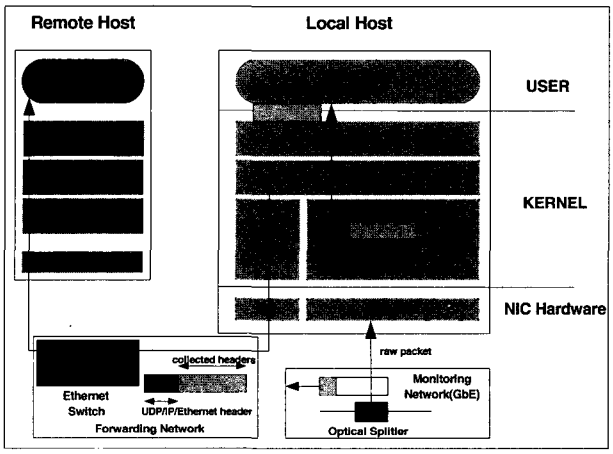
### 2.1 관련 연구

BPF(BSD Packet Filter)[4]는 유닉스 기반 운영체제(예, Free BSD, NetBSD 등)의 커널 내에 위치하고 있는 소프트웨어 방식의 대표적인 패킷 수집 프레임 워크이다. 패킷이 네트워크 인터페이스에 도착했을 때, 네트워크 디바이스 드라이브는 일반적으로 시스템 프로토콜 스택(system protocol stack)에 패킷을 전달한다. 그러나 BPF가 이 인터페이스에서 기다리고 있을 때에는 네트워크 디바이스 드라이브는 먼저 BPF를 호출하고 패킷은 각각의 어플리케이션 필터에게 전달된다. 사용자에 의해 정의된 필터는 각각의 필터와 관련된 버퍼에 필요한 패킷 정보를 복사한다. BPF는 몇 개의 패킷을 한 단위로 포장(encapsulation)하여 이를 모니터링 어플리케이션에게 전달한다.

MRTG(Multi-Router Traffic Grapher)[5]는 네트워크 링크 간의 트래픽 부하량을 측정하는 도구로서 HTML을 이용하여 네트워크 트래픽을 쉽게 파악 할 수 있는 GIF 이미지를 생성한다. SNMP[6]를 이용하여 트래픽 정보를 읽어들여서 트래픽 정보를 분석한다. 출력은 그래프 형태로 나타나게 되는데 웹 페이지(web page)에 포함되므로 단순히 웹 브라우저(web browser)만 있으면 사용자는 언제 어디서나 그 정보를 읽을 수 있다.

TCPdump[7]는 네트워크 상에서 전송되는 패킷의 헤더 정보를 분석하여 보여주는 역할을 하는데 사용자는 그 정보를 수동으로 분석하여 네트워크의 상태를 파악해야 한다. 따라서 장시간의 네트워크 상태를 파악하는 데는 적합하지 않다. TCPdump가 여러 가지 다양한 옵션들을 가지고 패킷 정보를 자세하게 보여주는 장점은 있지만 텍스트 기반이고 분석 기능을 제공하지 않는다는 것과 출력 결과가 매우 복잡하다는 것이 단점이다. 그 외에도 Etherfind[8], NFSwatch [9], Argus[10] 등 네트워크 모니터링을 위한 다양한 소프트웨어 툴들이 있다.

소프트웨어적인 패킷 헤더 수집방식 이외에도 OC3MON [11], OC12MON와 같은 OC3/ATM, OC12/ATM에 기초한 하드웨어 기반 시스템들도 등장하였다. 이는 ATM(Asynchronous Transfer Mode)과 같은 고속 시스템에 비해 프로세서의 성능이 충분하지 않아 소프트웨어적 접근 방법이 적합하지 않음을 고려한 것으로서, Unix(Coral Reef[12])에 포팅되었다.




(그림 1) Hasegawa에 의해 제안된 패킷 헤드 수집기의 구조

#### 2.1.1 부하 분산 모니터링 시스템

일반적으로 기가 속도의 트래픽 상에서 패킷 헤더 수집은 너무나 고 부하이므로 같은 호스트에서 다른 모니터링 기능을 실행할 여유가 없다. 따라서 헤더 수집은 다른 모니터링 기능과 독립적으로 실행될 필요가 있다. Hasegawa와 그의 동료들은 저비용으로 고속의 트래픽에 적용될 수 있는 모니터링 시스템[2]을 설계하였으며, 부하 분담의 개념을 도입하고 기가 속도의 트래픽상에서 고속으로 패킷 헤더를 수집하는데 초점을 맞추었다. 이 시스템은 부하 분산을 위해 패킷 헤더 수집과 모니터링 기능을 다른 시스템에서 실행되게 하였다((그림 1) 참조).

헤더 수집은 GbE NIC(Gigabit Ethernet Network Interface Card)과 수정된 디바이스 드라이브(customized device driver)에 의해서 이루어진다. NIC를 통해 수신된 패킷은 수정된 디바이스 드라이브에서 헤더가 분리되어 네트워크 버퍼(예, BSD의 Mbuf[13], 리눅스의 sk_buff[14])의 데이터 영역에 분

리하여 저장된다. 수집된 헤더 정보의 그룹은 UDP 메시지로 포장되어 네트워크를 통해서 리모트 호스트(remote host)로의 전송은 물론 로컬 호스트(local host)로도 전송이 가능하다. 패킷 헤더 수집 시스템으로부터 분리된 모니터링 시스템은 UDP/IP 스택을 통해서 헤더 정보를 받게 된다.

부하 분산의 개념을 도입한 또 다른 모니터링 시스템은 WebTrafMon II[3]이다. 이 시스템의 가장 큰 특징은 부하 분산 방법을 사용했다는 점이다. 즉, 패킷을 수집해서 파일에 저장하는 패킷 수집 모듈(probe)과 파일에 저장된 패킷 정보를 분석해서 데이터베이스에 저장한 뒤 매시, 매일, 매달, 매년 단위로 분석하는 패킷 분석 모듈(analyzer), 그리고 분석된 정보를 웹에서 보여주는 웹 뷰어(web viewer) 모듈로 구성되어 있다. 이 세 모듈은 각각의 시스템에서 동작함으로서 여러 네트워크 노드(network node)에서 패킷 수집을 할 수 있고, 패킷 분석 시의 고 부하가 패킷 수집에 영향을 주지 않는다.

### 2.2 연구 배경

고속의 네트워크에서는 전송되는 패킷의 수가 매우 많다. 그러한 데이터를 분석하는 것은 굉장히 많은 처리시간을 요한다. 따라서 패킷 수집 과정은 최대한 효율적으로 작성되어야 한다. 모든 패킷을 수집할 수 있을 정도로 시스템 자체 속도가 빠르지 못하다면 모니터링 분석 결과를 완전히 신뢰할 수 없을 것이다. 그러나 기가비트 속도로 전송되는 모든 패킷을 제대로 수집한다는 것은 매우 어려운 일이다. 소프트웨어나 하드웨어에 의한 패킷 수집 방안에 관한 많은 연구가 있었지만, 그 자체만으로 고속의 네트워크 속도를 따라 잡기에는 한계가 있었다.

부하 분산의 개념을 도입하여 패킷 헤더 수집 기능과 모니터링 기능을 서로 다른 시스템에서 실행시키면서, CPU 부하를 덜고 패킷 헤더 수집률을 높이며 모니터링 성능도 향상시킬 수 있었다[2]. 이러한 부하 분산 모니터링 시스템에서는 패킷 헤더 수집기의 성능이나 기능이 무엇보다 중요한 역할을 하게 된다. 부하 분산 시스템은 유입되는 패킷의 양에 따라서 시스템의 수를 증가시킴으로써 부족한 성능을 보충 할 수 있지만, 패킷 헤더 수집기는 실시간으로 패킷을 수집해야 하기 때문에 기가 속도의 고속 네트워크에서는 분석기능 없이 패킷 수집만 하는 시스템이라도 패킷을 분실할 가능성이 크다. 망의 고속화와 트래픽의 양이 급속하게 증가하고 있는 인터넷 동향을 볼 때, 모니터링 시스템에서도 부하 분산과 고속의 패킷 헤더 수집기의 필요성이 급증하고 있는 추세이다.

본 논문에서는 네트워크 데이터 처리를 위해 특화된 네트워크 프로세서를 이용하여, 저가의 패킷 헤더 수집기를 설계 및 구현하였다. 네트워크 프로세서는 낮은 가격에 우수한 성능을 낼 수 있는 네트워크 시스템 개발에 활용되고 시장 변화에 의해 요구되는 기능들을 빠르게 수용할 수 있다. 기존의 범용 프로세서 기반 방식과 달리 네트워크에 특성화된 내부 구조를 가짐으로써 고속의 데이터 처리를 요구하는 네트워크 장비에 사용할 수 있다. 또한 프로그램에 의해 동작하므로 기존의 알고리즘을 업그레이드하거나 새로운 기능의 추가가 용이하다는 점에서 주문형 반도체 기반 방식이 가지지 못한 유연성을 가진다.

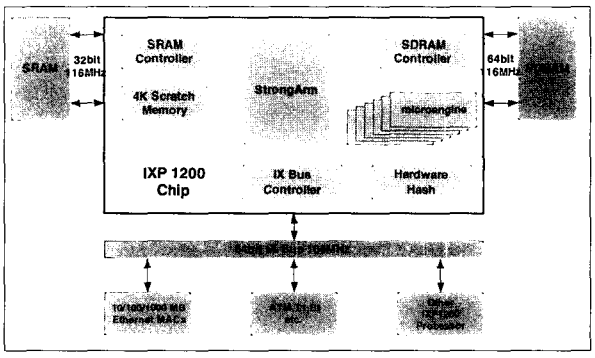
본 연구에서는 대표적인 네트워크 프로세서인 인텔(Intel)의 IXP1200을 기반으로 한다. IXP1200은 네트워크 데이터 처리를 위한 6개의 프로그래밍이 가능한 프로세서(마이크로엔진)와 프로세서마다 4개의 쓰레드(thread)를 가지는 멀티프로세서, 멀티쓰레드 기반으로 고속으로 패킷을 처리 할 수 있다. 또한 6개의 네트워크 데이터 처리를 위한 프로세서는 프로그래밍이 가능하여 사용자의 목적에 맞게 다양한 시스템 구현이 가능하며, 고성능 시뮬레이터의 제공으로 프로그래밍 및 테스트를 효율적으로 할 수 있다. 본 논문에서는 IXP1200 네트워크 프로세서를 이용한 고속의 패킷 헤더 수집기를 구현하기 위한 효율적인 방안을 모색하고 이를 설계 및 구현하며, 실험을 통해 기존시스템과의 성능을 비교 평가한다.

## 3. 고속 패킷 헤더 수집을 위한 제안 모델

본 절에서는 고속의 패킷 헤더 수집기를 구현하기 위해 사용되는 IXP1200 네트워크 프로세서의 구조와 이를 기반으로 한 설계 및 구현 원리에 대해 기술한다.

### 3.1 네트워크 프로세서 구조

(그림 2)는 IXP1200의 블록 다이어그램으로서, 실제 패킷 처리를 수행하는 6개의 마이크로 엔진(microengine, $\mu$E)과 메모리 유닛(memory unit), FBI 유닛, 그리고 코어(core)로서 이들 유닛을 관리하는 스트롱암(strongarm) 프로세서가 결합된 형태로 이루어져 있다. 스트롱암 프로세서는 각 기능 유닛에 대한 관리 기능을 수행한다.




(그림 2) IXP1200 블록 다이어그램

FBI 유닛은 마이크로 엔진에서 전달된 명령어에 의해 동

작하는 유닛으로서, IX 버스 슬레이브 장치와 IX 버스를 통해 연결된 각 물리 포트와의 패킷 입출력을 전담한다. 포트를 통해 들어온 패킷은 메인 메모리에 전달되고, 이에 대한 정보를 마이크로 엔진에 전달하여 마이크로 엔진이 패킷을 처리할 수 있도록 한다. 처리된 패킷은 다시 FBI 유닛을 통해 목적지 정보에 따라 해당 IX 버스 슬레이브 장치의 물리 포트로 전달된다. IX 버스는 인텔의 독자적인 버스 규격으로서, 64bit 데이터 대역폭, 최대 동작 주파수 104Mhz에, 4.4Gbps의 데이터 교환량을 가진다.

6개의 마이크로 엔진들은 각각 독립적인 RISC 코어로서, 레지스터 파일과 32비트 산술논리연산 유닛, 그리고 독자적인 명령어 세트를 가지고 있다. 패킷 처리를 전담하는 기능 유닛으로서, 4개의 프로그램 카운터를 내장하여 4개의 쓰레드들이 수행될 수 있도록 설계되어 있으며, 명령어 세트는 패킷 처리를 효과적으로 수행하기 위해 관련된 연산 명령어들을 포함하고 있다.

### 3.2 멀티프로세서, 멀티쓰레드 기반의 부하 분산

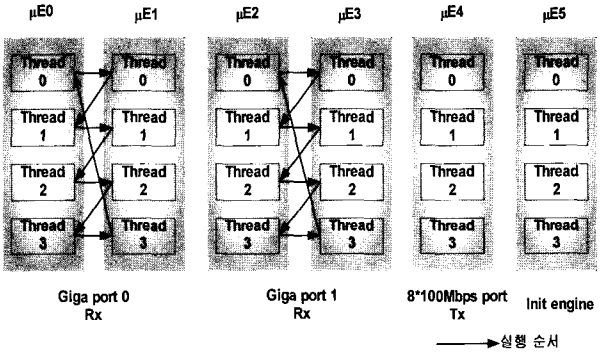
본 연구에서는 패킷을 수신하기 위해 2개의 기가 포트(Giga Port)를 사용하고 있으며, 패킷 헤더를 전송하기 위해서 8개의 100Mbps MAC 포트를 사용하고 있다. 2개의 기가 포트를 사용하는 것은 양방향으로 흐르는 패킷을 각각 수신하기 위함이다. 본 절에서는 2개의 기가 포트를 통해 수신되는 패킷들을 효율적으로 처리하기 위한 쓰레드 구성방안에 대해 논의한다.

IXP1200 네트워크 프로세서는 멀티프로세서, 멀티쓰레드를 기반으로 하는 프로세서로서 메모리 접근에 의한 유휴기(idle period)에서도 문맥교환(context switching)을 통한 마이크로 엔진 파이프라인을 활성화시켜 성능을 배가시키는 특징을 가지고 있다. 이는 메모리 접근에 의한 유휴기가 전체 사이클의 75%를 차지한다는 것을 감안할 때 문맥교환에 의한 오버헤더에도 불구하고 그 효율성은 의심할 여지가 없다. 따라서 시스템의 목적에 따른 적절한 프로세서와 쓰레드의 구성은 시스템 성능을 높이는데 중요한 역할을 할 것이다.

본 연구에서 기가비트 네트워크에서의 효과적인 패킷 헤더 수집을 위해 설계한 6개의 마이크로 엔진들에 의해 수행되는 쓰레드들의 구성은 (그림 3)과 같다. $\mu$E0-$\mu$E3에서 2개의 기가 포트(0, 1)에서 들어오는 패킷 수신, 패킷 헤더 분리, 큐잉(Queuing) 등을 수행하는 역할을 담당하며, $\mu$E4에서는 8개의 100Mbps MAC 포트로의 패킷 전송을, $\mu$E5에서는 하드웨어 초기화 및 쓰레드 초기화를 담당하고 있다.

$\mu$E0-$\mu$E3에서 $\mu$E0-$\mu$E1은 기가 포트 0에서, $\mu$E2-$\mu$E3은 기가 포트 1에서 패킷을 수신하여 처리한다. $\mu$E0-$\mu$E1와 $\mu$E2-$\mu$E3은 처리하고자 하는 패킷의 수신 포트만 다르고 4개의 마이크로 엔진 각각의 쓰레드들은 같은 코드로 이루어져 있기 때문에 각 쓰레드들은 같은 실행 순서로 구성되어 있다. 이와 같이 기가비트 입력 데이터 처리를 위해 4개의 마이크로 엔진을 할당하는 이유는 고속의 데이터를 수신하여 시스템의 목적에 맞게 처리하는 작업부하가 본 시스템에서 제일 크기 때문에 가능한 많은 프로세싱 자원을 할당하기 위함이다.

(그림 3)에서 화살표는 $\mu$E0-$\mu$E1, $\mu$E2-$\mu$E3에서 쓰레드들의 실행순서를 나타낸다. 각 쓰레드는 하나의 패킷 수신 시 실행순서에 의해 동일 패킷을 64바이트씩 나누어 처리함으로써, 서로 부하를 분담하고 쓰레드당 하나씩 있는 메일박스(mailbox)를 통해 서로간의 패킷에 대한 정보(버퍼위치, 현재까지 받은 element의 수 등)를 화살표 순서대로 주고받으며 공유하게 된다. 이러한 라운드 로빈(round robin) 방식의 패킷 처리 메커니즘은 간단하면서도 4개의 마이크로 엔진과 쓰레드들이 기가 속도의 패킷 수신 및 처리 부하를 균등하게 받게 함으로써 성능을 향상시킬 수 있는 장점을 가진다. 각 쓰레드가 패킷 단위가 아닌 64바이트 단위로 동일 패킷의 부하를 분담하는 것은 패킷 단위로 처리할 경우 복잡한 메커니즘으로 인해 오히려 큰 성능 저하가 발생하기 때문이다.

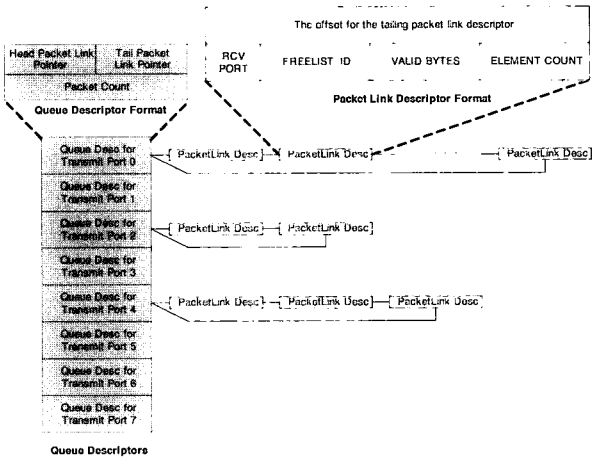



(그림 3) 마이크로 엔진들에 의해 수행되는 쓰레드들의 구성

$\mu$E4는 8개의 100Mbps MAC 포트로 패킷 헤더를 전송하는 역할을 담당하며, 쓰레드 0이 스케줄링, 쓰레드 1~3은 패킷 헤더 전송을 담당하는데, 여기에서 스케줄링은 라운드 로빈 방식으로 쓰레드 1~3에게 작업을 할당함으로써 8개의 100Mbps MAC 포트로의 패킷 헤더 전송을 관리하게 된다. 8개 포트로의 패킷 헤더 전송을 위해 마이크로 엔진 하나만을 할당한 것은 8개 포트의 물리적인 성능을 감안할 때 하나의 마이크로 엔진만으로도 이를 여유 있게 처리할 수 있기 때문이다. 중요한 것은 2개의 기가 포트에서 2Gbps의 속도로 수신되는 패킷을 8개의 100Mbps MAC 포트, 즉 800Mbps의 속도로 전송이 가능한가이다. 이는 비록 2Gbps의 패킷이 수신되더라도 단지 헤더에서 필요한 정보만을 분리하여 전송하기 때문에 충분히 가능하다.

### 3.3 패킷 버퍼 관리

본 절에서는 기가 포트로부터 패킷 헤더를 분리한 후, 이를 8개의 100Mbps MAC 포트로 전송하기 위한 패킷 버퍼 관리 방식에 대해 논의한다.




( 그림 4 ) 패킷 버퍼 관리

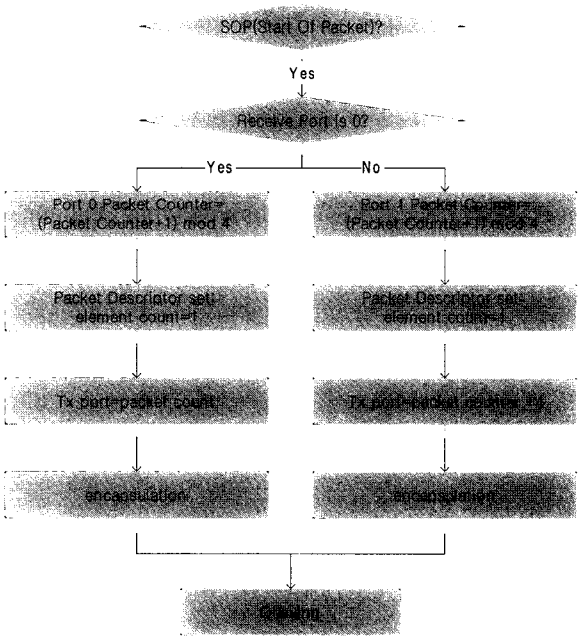
(그림 4)는 패킷 버퍼 관리를 위해 본 논문에서 사용하고 있는 구조를 나타내고 있으며, 크게 큐 기술자(queue descriptor)와 패킷 링크 기술자(packet link descriptor)로 구성되어 있다. 큐 기술자는 패킷을 전송할 수 있는 포트당 하나씩 존재한다. Packet Count는 해당 포트로 전송해야 하는 패킷의 수를 나타내며 Head Packet Link Pointer, Tail Packet Link Pointer은 각각 해당 포트로 전송해야 하는 패킷의 정보를 가지고 있는 첫 번째와 마지막 패킷 링크 기술자의 포인터 값을 가지고 있다.

패킷 링크 기술자는 전송해야 하는 패킷당 하나씩 존재하며, 서로 연결 리스트(linked list)로 연결된다. The offset for the tailing packet link descriptor가 서로 연결된 다른 패킷 링크 기술자의 포인트를 가지고 있으며, RCV PORT는 해당 패킷이 수신된 포트이다. FREELIST_ID는 패킷 버퍼를 여러 개로 나누어서 사용할 때 해당 패킷이 저장되어 있는 버퍼이며, VALID BYTES는 해당 패킷의 마지막 64바이트에서 유효 데이터를 바이트로 나타낸 것이다. ELEMENT COUNT는 해당 패킷이 64바이트 단위로 몇 개인지를 나타낸다.

$\mu$E0-$\mu$E3은 패킷을 수신한 후 전송해야 할 포트를 결정하고 패킷 링크 기술자를 작성한 후, 이를 해당 전송 포트의 큐 기술자의 맨 마지막에 연결한다. 그 후 $\mu$E4에 있는 패킷을 전송하는 쓰레드는 각 포트의 큐 기술자를 확인하고 해당 포트로 전송해야 하는 패킷이 있으면, 그 패킷 링크 기술자로부터 해당 패킷에 대한 정보를 얻어서 패킷을 전송하게 된다.

### 3.4 패킷 헤더 전송을 위한 부하 분산

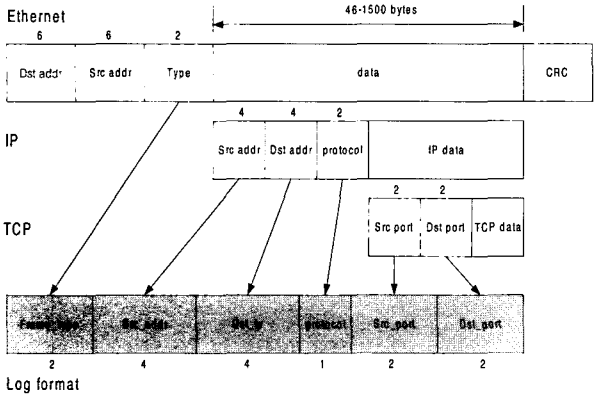
본 논문에서 제안하는 고속 패킷 헤더 수집기에서 고려해야 할 또 다른 문제는 2개의 기가 포트로부터 2Gbps의 속도로 패킷을 수신할 때 패킷 헤더를 분리하여 8개의 100Mbps MAC 포트로 균등하게 패킷을 분배해야 한다는 것이다. $\mu$E0-$\mu$E1, $\mu$E2-$\mu$E3은 각각 기가 포트로부터 패킷 수신 시 패킷의 첫 64바이트(Start of Packet; SOP)일 경우에만 SRAM에 있는 해당 포트의 패킷 카운터(packet counter)를 증가시킨다. 이때 $\mu$E의 id를 이용하여 패킷의 수신 포트를 구분하게 되는데, $\mu$E0-$\mu$E1은 기가 포트 0을, $\mu$E2-$\mu$E3은 기가 포트 1을 담당한다. 수신 포트가 0일 경우 패킷 카운트의 값(0-3)에 따라서 100Mbps MAC 포트의 해당 포트(0-3)로 패킷을 전송하며, 포트 1일 경우에는 패킷 카운트의 값(0-3)에 4를 더해서 100Mbps MAC 포트의 해당 포트(4-7)로 패킷을 전송한다. 이러한 전략을 사용함으로써 패킷 헤드 전송의 균등한 부하 분산을 이룰 수 있다. 이 때 2개의 기가 포트로부터 받은 패킷은 할당 받은 SDRAM의 버퍼 위치에 버퍼링 된다. 그러나 패킷 링크 기술자의 ELEMENT COUNT 필드를 1로 설정함으로써 SOP만을 큐잉시키기 때문에 $\mu$E4에서는 패킷 헤더 정보만을 전송시킬 수 있다. (그림 5)는 $\mu$E0-$\mu$E3이 패킷의 SOP 수신 후 처리하는 과정을 간단한 순서도로 나타낸 것이다.




( 그림 5 ) 패킷 수신 및 처리 순서도

패킷 수신 후 분석 시스템으로의 전송을 위해 UDP 메시지를 사용하고 있다. 패킷의 헤더 정보는 분리되어 UDP 메시지로 포장되어 전송된다. 로그 정보 포맷은 (그림 6)과 같다. Frame_type은 이더넷 헤더에서 정보를 가져오고 src_ip, dst_ip, protocol 등은 IP 프로토콜 헤더에서, src_port, dst_port 등은 TCP/UDP 헤더에서 정보를 가져온다. 로그 포맷의 일정 필드를 사용하지 않는 프로토콜은 사용하지 않는 필드를 0으로 채운다. 예를 들어,

ARP와 같이 IP 기반이 아닌 프로토콜은 frame_type 등의 값만 저장한 뒤, 사용하지 않는 src_ip, dst_ip, protocol, src_port, dst_port 등은 모두 0으로 설정한다. 또 ICMP와 같이 IP기반이지만 포트 정보를 사용하지 않는 프로토콜은 frame_type, src_ip, dst_ip 등의 값만 저장한 뒤, src_port, dst_port 등을 0으로 설정한다. 포트 번호 중 0번이 있으나, 분석할 때 TCP/UDP 프로토콜 여부를 먼저 체크한 뒤 분석하면 되기 때문에 TCP/UDP의 포트 번호 0번과 빈 값을 채워 넣는 0번과 구분할 수 있다. 수집된 로그 정보는 UDP 메시지를 통해서 로컬 호스트뿐만 아니라 리모트 호스트에게로 전송된다.




(그림 6) 패킷 헤더 수집기의 로그 정보

## 4. 성능 평가

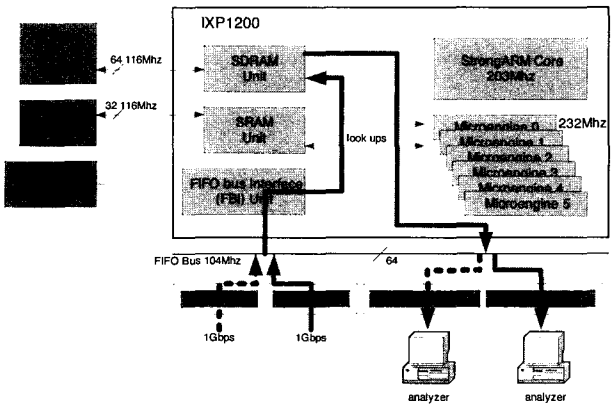
본 절에서는 본 연구에서 설계, 구현된 멀티프로세서, 멀티쓰레드 기반의 패킷 헤더 수집기의 패킷 헤더 수집률(packet header collection ratio)을 측정하기 위한 실험 환경에 대해 기술하고, 실험 결과로서 그 성능을 비교 평가한다.

### 4.1 실험 환경

실험을 위해 인텔사에서 제공하는 시뮬레이터(intel IXP 1200 network processor family developer workbench)를 사용하였다. 이것은 실제 플래폼에서 실험하는 것과의 오차율이 0.5% 이하이기 때문에 신뢰할 수 있는 환경이다. SDRAM은 24Mbyte, SRAM은 2Mbyte의 크기를 사용하였다. 그리고 2개의 기가 포트와 8개의 100Mbps MAC 포트가 사용되었다.

네트워크 시스템의 성능을 측정하는 요소로는 처리량, 버퍼 크기, 최대 버퍼량, 전달 지연 시간 등 여러 가지가 있으나 본 연구에서는 처리량을 기준으로 한다. 이것은 기가 속도로 패킷을 수신하였을 때 버퍼 오버 플로우와 패킷 분실 없이 수신된 모든 패킷의 헤더를 분리하여 분석 시스템으로 전송하는 패킷 헤더 수집률로서 제안 시스템의 성능을 가장 잘 보여 줄 수 있는 요소이다.

본 연구의 성능 측정은 두 가지 경우에 대해서 패킷 헤더 수집률로 판단한다. 첫 번째는 다양한 패킷 크기(64, 128, 192, 256, 384, 512, 768, 1024, 1518바이트)별로 실험을 하는 것이다. 이는 패킷 헤더 수집기는 패킷 수신 후 패킷 헤더 정보만을 분석 시스템으로 전송하는 시스템으로, 패킷 크기가 시스템 성능을 많이 좌우하기에 패킷 크기에 따른 패킷 헤더 수집률을 측정하면 본 시스템의 성능이 어느 정도인지 비교 판단할 수 있기 때문이다. 두 번째는 제안 시스템의 실제 네트워크(실망)에 적용 가능성을 판단하기 위한 실험이다. 즉 다양한 시간, 다양한 실망에서 측정된 패킷 크기 분포를 바탕으로 같은 분포를 가지는 패킷을 시뮬레이터상에서 만들어 실험하는 것이다. 이는 제안 시스템의 실망에서 적용 가능성을 조금 더 확실히 확인할 수 있을 것이다.



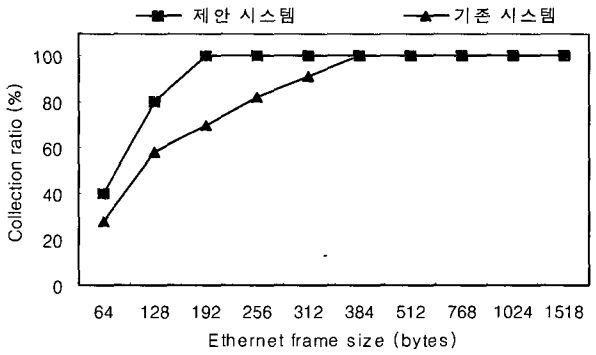

(그림 7) 실험 환경

입력 데이터는 시뮬레이터에서 제공하는 패킷 발생 기능을 사용하였다. 2개의 기가 포트로부터 각각 1Gbps로 패킷을 수신하여 8개의 100Mbps MAC 포트들로 전송하였으며, 패킷 크기당 10,000,000개의 패킷을 수신 및 처리하여 전송하게 하였다. (그림 7)은 실험 환경을 보여준다.

### 4.2 실험 결과 분석

본 연구의 패킷 헤더 수집 방식은 부하 분산 모니터링 시스템의 분석 시스템이 리모트 또는 로컬 네트워크에 존재한다고 가정한다. 수신된 패킷에서 필요한 헤더 정보를 분리하여 UDP 메시지로 전송하는 방식으로서 2개의 기가 포트로부터 각각 1Gbps의 속도로 패킷을 수신한다. 기가 포트 0으로 수신된 패킷은 100Mbps MAC 포트 0-3으로 전송하고, 기가 포트 1로 수신된 패킷은 포트 4-7로 전송한다. 성능 비교 대상으로는 앞서 2절에서 기술한 Hasegawa에 의해 제안된 고속 패킷 헤더 수집기[2]를 선택하였다. 이는 저가의 고속 패킷 헤더 수집기를 설계 및 구현하고자 하는 목적이 본 연구와 일치하며, 패킷 헤더 수집기에 초점을 맞춘 연구로서 본 연구와 같은 의도를 가지고 있기 때문이다. 이 연구[2]는 실험 환경으로서 기가 이더넷으로부터 단 방향으로 1Gbps 속도로 패킷이 수신 될 때, 패킷 크기당 패킷 헤더 수집률을

측정해 패킷 분실 없이 패킷 헤더를 수집하여 UDP 메시지로서 성공적으로 전송할 수 있는 패킷 크기를 결정한다. 제안 시스템과 기존 시스템[2]의 성능 비교를 위해서는 같은 실험 조건을 유지해야 한다. 따라서 제안 시스템의 성능 실험 결과에서는 단지 기가 포트 0으로 수신된 패킷의 처리 결과만을 기존 시스템의 실험 결과와 비교 평가하였다.




( 그림 8 ) 패킷 헤더 수집 비율

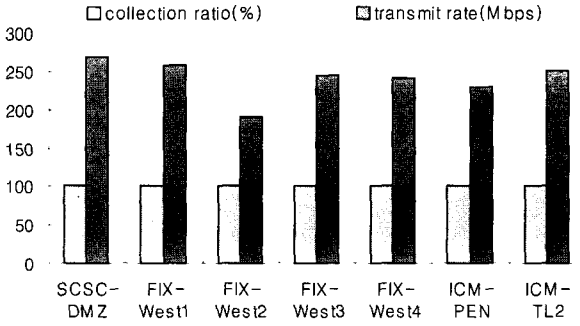
(그림 8)은 본 논문에서 제안하는 패킷 헤더 수집기와 기존의 패킷 헤더 수집기[2]의 성능을 비교하여 보여 주고 있다. X축은 패킷 크기이며 Y축은 패킷 크기당 패킷 헤더 수집률을 나타내고 있다. 기존 시스템은 패킷 크기 384byte와 그 이상의 경우에는 100% 패킷 헤더를 수집할 수 있으나 그 이하의 크기에서는 패킷 크기가 작아질수록 패킷 헤더 수집률은 더 떨어진다. 이에 비해 제안 시스템의 패킷 헤더 수집률은 190 바이트 이상의 크기에서는 100% 패킷 헤더를 수집하고 있다. 이는 1Gbps로 패킷이 수신될 때 패킷 크기가 커질수록 패킷 개수는 적어지게 되어, $\mu$E0-$\mu$E1에서 수행해야 할 작업량(패킷 헤더 분리, encapsulation)이 줄어들기 때문이다. 반대로 패킷 크기가 작아질수록 패킷 개수는 많아지게 되고, $\mu$E0-$\mu$E1에서 수행해야 할 작업량이 늘어나게 된다. 따라서 패킷 크기가 커질수록 패킷 헤더 수집률은 높아지는 현상을 보여주는 것이다. 두 시스템의 실험 결과에서 단지 처리할 수 있는 패킷 크기만으로 비교하더라도 제안 시스템이 기존 시스템의 성능에 비해 월등히 우수하다는 것을 알 수 있다. 이는 기존 시스템이 단지 최적화된 소프트웨어(수정된 디바이스 드라이브)에만 의존했지만, 제안 시스템은 네트워크 데이터 처리를 위해 특화된 멀티프로세서, 멀티쓰레드를 기반으로 한 병렬 처리 방식의 하드웨어와 소프트웨어를 사용했기 때문이다. 인터넷 백본 트래픽의 연구[15, 16]에 따르면 평균 IP 패킷 크기는 384-512바이트이다. 이는 본 연구의 실험 결과와 비교해 볼 때 실제 네트워크에서도 제안 시스템이 효과적으로 사용될 수 있음을 증명한다.

다음 실험은 제안 시스템의 실제 망에서의 사용 가능성을 판단하기 위한 것으로서 트래픽 전문 측정 기관(NLANR)의 데이터를 이용하였다. <표 1>은 NLANR에서 제공하는 서로 다른 시간, 다른 망에서 측정한 데이터로서 각 망에서 대부분 비율을 차지하는 패킷 크기 분포를 보여주고 있다.

(그림 9)는 <표 1>에서의 패킷 크기 분포를 시뮬레이터로 실행시킨 결과로서 X축은 측정망의 이름을 나타내며, Y축은 패킷 헤더 수집률과 전송 속도를 보여주고 있다. 실망에서의 패킷 크기 분포로 구성된 패킷은 임의로 선택되어 1Gbps 기가 포트로 유입되며, 필요한 헤더정보만 분리되어 UDP 메시지로서 100Mbps MAC 포트로 전송된다. 실험 결과 모두 100% 패킷 헤더 수집률을 나타내고 있으며, 단지 전송속도에서만 조금씩 차이를 보이고 있다. 전송속도에서의 차이는 <표 1>에서의 평균 패킷 크기별로 조금씩 다른 결과를 보이고 있는데, 이는 패킷 크기가 커질수록 패킷 헤더 수는 적어지며, 전송해야 하는 UDP 메시지도 적어지기 때문이다. 따라서 실망에서의 패킷 크기 분포와 같이 패킷을 구성하여 실험한 결과를 통해 본 논문에서 제안하는 시스템은 실망에서도 충분히 사용 가능하다는 것을 알 수 있다.

〈표 1〉 패킷 크기 분포

| 패킷 크기 (바이트) | | 48 | 64 | 100 | 256 | 512 | 600 | 1100 | 1500 | 평균크기 (바이트) |
|---|---|---|---|---|---|---|---|---|---|---|
| 비율 (%) | 940714-SDSC-DMZ | 39 | 5 | 11 | 3 | 6 | 27 | 2 | 0 | 292 |
| | 950621-FIX-West | 38 | 4 | 8 | 7 | 8 | 29 | 1 | 1 | 306 |
| | 960918-FIX-West | 30 | 3 | 5 | 13 | 10 | 21 | 5 | 8 | 426 |
| | 960926-FIX-West | 32 | 5 | 6 | 21 | 5 | 18 | 3 | 4 | 321 |
| | 970109-FIX-West | 24 | 8 | 5 | 25 | 7 | 15 | 4 | 4 | 322 |
| | 970112-ICM-PEN-10 | 50 | 3 | 5 | 4 | 6 | 18 | 3 | 8 | 347 |
| | 970112-ICM-TL2-11 | 54 | 3 | 4 | 4 | 6 | 18 | 2 | 7 | 319 |




( 그림 9 ) 패킷 크기 분포에 따른 패킷 헤더 수집률과 전송 속도

## 5. 결 론

망의 고속화와 트래픽 양이 급속하게 증가하고 있는 인터넷 동향을 볼 때 모니터링 시스템에서도 부하 분산과 고속의 패킷 헤더 수집기의 필요성이 더욱더 중요해졌다. 본 논문에서는 분산 모니터링 시스템을 위한 패킷 헤더 수집기의 성능 개선을 위해서 멀티프로세서, 멀티쓰레드의 네트워크 프로세서 기반 위에 고속 패킷 헤더 수집기를 제안하였다.

실험 결과를 통해 본 논문에서 제안하는 고속 패킷 헤더 수집기가 기존의 패킷 헤더 수집기보다 높은 성능을 보이는 것을 확인하였다. 기존의 패킷 헤더 수집기가 384바이트 이

상의 패킷 크기에서만 100% 패킷 헤더를 수집했지만, 제안 시스템은 190바이트 이상의 패킷 크기에서도 100% 헤더 정보를 수집할 수 있었다. 이는 제안 시스템이 네트워크 데이터 처리를 위해 특화된 네트워크 프로세서의 데이터 병렬 처리 방식에 크게 의존하고 있으며, 이와 함께 병렬 프로세서 시스템에 적합한 구조로 설계된 소프트웨어를 사용했기 때문이다. 또한 실망에서의 패킷 크기 분포를 바탕으로 한 실험에서 100% 패킷 헤더 수집률을 보임으로써 제안 시스템이 실망에서도 적용 가능하다는 것을 확인할 수 있었다.

보다 정확한 실험을 위해서는 패킷 손실률을 정확히 측정할 수 있는 측정 도구로부터 생성되는 모든 패킷의 양과 전송되는 동안 네트워크에서 손실되는 패킷의 양을 정확하게 측정할 수 있는 도구를 활용해 실제 플래폼에서의 실험이 필요하다. 또한 최적화된 쓰레드 구조와 큐잉 구조에 의해 성능 향상 방안도 연구되어야 한다.

## 참 고 문 헌


[1] IEEE Computer Society. "IEEE STD 802.3, 2000 Edition," Oct.2000. http://standards.ieee.org/getieee802/.

[2] Teruyuki Hasegawa, Tomohiko Ogishi, and Toru Hasegawa. "A Framework on Gigabit Rate Packet Header Collection for Low-cost Internet Monitoring System," May 2002.

[3] 홍순화, 김재영, 홍원기, "로드 분산 방법을 이용한 대용량 네트워크 트래픽 모니터링 및 분석", KNOM Review, Vol.4, No.2, pp.17-27, Dec., 2001.

[4] McCanne, S., and Jacobson, V., "The BSD Packet Filter: A New Architecture for User-level Packet Capture," Proc. of the 1993 Winter USENIX Technical Conference, Jan. 1993.

[5] Tobias Oetiker and Dave Rand, "MRTG: MultiRouter Traffic Grapher", http://eestaff.ethz.ch/~oetiker/webtools/mrtg/mrtg. html.

[6] David Perkins and Evan McGinnes, Understanding SNMP MIBs, Prentice-Hall, 1997.

[7] Lawrence Berkley National Laboratory, "tcpdump 3.4a6," ftp://ftp.ee.lbl.gov.

[8] Craig Hunt, TCP/IP Network Administration, O'Reilly and Associates, Inc., 1992.

[9] Dave Curry and Jeff Mogul, "nfswatch-4.3," http://ftp.lip6.fr/pub2/networking/nfs/.

[10] Carter Bullard, "argus-1.7.beta.1b," ftp://ftp.sei.cmu.edu/pub/argus.

[11] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder. "OC3MON: Flexible, Affordable, High-Performance Statistics Collection," Proc. Of INET'97, June 1997. http://www.nlanr.net/ NA/Oc3mon/.

[12] Keys, K., Moore, D., Koga, R., Lagache, E., Tesch, M., and Claffy, K., "The Architecture of CoralReef: An Internet Traffic Monitoring Software Suite," Proc. of PAM2001, Apr., 2001. http://www.caida.org/outreach/papers/pam2001/coralreef. xml.

[13] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman. "The Design and Implementation of the 4.4BSD Operating System," Addison-Wesley, Reading, MA, 1996.

[14] A. Cox. "Network Buffers and Memory Management," Linux Journal, Issue #30, Oct. 1996. http://www2.linuxjournal.com/ljissues/issue30/1312.html.

[15] Claffy, K., "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," Proc. of INET'98, July 1998. http://www.caida.org/outreach/papers/Inet98/.

[16] McCreary, S., and Claffy, K., "Trends in Wide Area IP Traffic Patterns-A View from the Ames Internet Exchange," May, 2000. http://www.caida.org/outreach/papers/AIX0005/.



**최 경 희**

e-mail : khchoi_at_madang.ajou.ac.kr

1976년 서울대학교 사범대학 수학교육과(학사)

1979년 프랑스 그랑데꼴 Enseeiht 정보공학과(공학석사)

1982년 프랑스 Paul Sabatier 정보공학과(공학박사)

1982년~현재 아주대학교 정보통신전문대학원 교수

관심분야 : 운영체제, 분산시스템, 실시간 및 멀티미디어시스템 등



**최 판 안**

e-mail : banana@lgphilips-lcd.com

2002년 아주대학교 공과대학 전자공학부(학사)

2004년 아주대학교 정보통신전문대학원 정보통신공학과(공학 석사)

관심분야 : 실시간 시스템



**정 기 현**

1984년 서강대학교 공과대학 전자공학과(학사)

1988년 미국Illinois주립대 EECS(공학석사)

1990년 미국Purdue대학 전기전자공학부(공학박사)

1991년~1992년 현대전자 반도체 연구소

1993년~현재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI설계, 멀티미디어 및 실시간 시스템 등



**심 재 홍**

1987년 서울대학교 전산과학과(학사)

1989년 아주대학교 컴퓨터공학과(공학석사)

1989년~1994년 서울시스템㈜ 공학연구소

2001년 아주대학교 컴퓨터공학과(공학박사)

2001년~2001.9. 아주대학교 정보통신전문대학원 BK21 전임연구원

2002년~현재 조선대학교 인터넷소프트웨어공학부 교수

관심분야 : 운영 체제, 분산 시스템, 실시간 및 멀티미디어시스템 등