여러 개의 SRAM과 해싱을 이용한 병렬 IP 어드레스 검색에 대한 연구

준회원 서 지 현*, 정회원 임 혜 숙**, 학생회원 정 여 진**, 정회원 이 승 준*

Parallel IP Address Lookup using Hashing with Multiple SRAMs

Ji-Hyun Seo* Associate Member, Hye-Sook Lim** Regular Member, Yeo-Jin Jung** Student Member, Seung-Jun Lee* Regular Member

요 약

컴퓨터간의 빠른 데이터 전송을 위해서는 링크 속도와 더불어 라우터에서의 패킷 전달율(forwarding rate) 이 중요하며, 이 중 어드레스 검색(address lookup) 은 패킷 전달 과정 중에서 매우 중요한 부분으로, 주요 시간 지연을 발생시키는 요인이라 할 것이다. 본 논문에서는 고속 라우터에 적합한 효율적인 어드레스 검색 하드웨어 구조를 제안하고자 한다. 본 논문에서 제안된 구조는 인터넷 어드레스의 프리픽스 길이별로 각각 다른 SRAM을 사용하여 여러 개의 어드레스 검색 테이블을 만들고, 각 테이블에 해싱(hashing)을 적용하여 동시에 어드레스 검색을 수행한 후, 각 테이블에서 일치되어 나온 엔트리 중 가장 길게 프리픽스가 일치하는 엔트리를 고르는 방식이다. 제안된 방식의 성능을 평가하기 위하여 MAE-WEST 라우터 예로 시뮬레이션을 수행하였다. 37000여 개의 라우팅 엔트리를 갖는 테이블 저장을 위해 약 300Kbyte의 메모리를 사용하였을 때, 패킷 당 1.93번의 메모리 접근 횟수를 갖는다.

ABSTRACT

One of the important design issues for IP routers responsible for packet forwarding in computer networks is the route-lookup mechanism. For each incoming packet, IP routing requires that a router performs a longest-prefix-match address lookup in order to determine the next hop that the incoming packet should be forwarded to. In this paper, we present a new scheme which applies the hashing function for IP address lookup. In the proposed scheme, the forwarding table is composed of multiple SRAMs, and each SRAM represents an address lookup table in each prefix. Hashing function is applied in order to find out the matching entries from the address lookup tables in parallel, and the entry with the longest prefix match among them is selected. Simulation using the MAE-WEST router example shows that a large routing table with 37000 entries can be compacted to a forwarding table of 300 Kbytes in the proposed scheme. It is also shown that the proposed scheme achieves one route lookup every 1.93 memory accesses in average.

Ⅰ. 서 론

인터넷이 좋은 서비스를 제공하기 위해서는 두 가지의 중요 사항이 요구된다. 링크 속도와 라우터 데이터 효율(throughput)과 관련된 패킷 전달율이 그것이다. 인터넷 라우터에서의 패킷 전달(packet forwarding)은 현재 기술로는 링크 속도의 증가만큼 빠르게 수행되고 있지 못하다. 특별히 패킷 전달을 수행하는데 있어서 가장 중요한 과정이라 할 수 있는 어드레스 검색(lookup)은 라우터에 입력된 패킷

* 이화여자대학교 과학기술대학원 정보통신학과 설계자동화 및 집적회로 연구실(ever77@ewha.ac.kr), ** 이화여자대학교 과학기술대학원 정보통신학과 컴퓨터 네트워킹 하드웨어 연구실

논문번호 : 020373-0830, 접수일자 : 2002년 9월 3일

※ 본 연구는 2002년도 이화여자대학교 교내 연구비 지원에 의하여 이루어졌음.

의 헤더에 있는 목적지 주소를 키로 사용하여 라우터에 저장되어 있는 전달 테이블(forwarding table) 내의 키와 부합하는 엔트리를 찾는 과정을 말하는데, 오늘날의 라우터에서 시간 지연을 발생시키는 주요 요인으로 작용하고 있다. 따라서 차세대 인터넷 라우터를 설계하는데 있어서 중요시되어야 할 문제 중 하나는 어드레스 검색을 얼마나 빠른 속도로 수행할 수 있느냐 하는 문제라고 할 것이다.

IP(Internet Protocol)주소 검색에 요구되는 longest prefix matching의 수행을 위해 여러 가지 빠른 검색(routing-lookup) 알고리즘들이 제안되어 왔는데, 살펴보면 다음과 같다.

IP 주소 검색은 IP 주소의 계층적 구조와 CIDR (Classless Inter-Domain Routing) 구조로부터 오는 특이성으로 인하여 Trie를 이용한 구조가 연구되어 왔다[1]. Degermark 등에 의해 고안된 전달 검색 구조(forwarding-lookup structure)[2]는 150~160 Kbyte 정도의 메모리를 사용하여 4만여 개의 엔트리를 가지는 큰 라우팅 테이블을 구성한 후, 소프트웨어적인 접근을 시도하는 방식인데, 이것을 하드웨어로 구현하려면 어드레스 검색을 위해서 최소 2번에서 최대 9번까지의 메모리 접근(memory access)이 요구되는 단점이 있다.

Gupta 등이 제안한 빠른 검색 구조(fast routing - lookup scheme)[3]는 큰 DRAM을 사용하는데, 어드레스 검색을 위해 최대 2번의 메모리 접근을 필요로 하는 장점이 있지만 33Mbyte 라는 큰 저장공간을 필요로 한다. 중간길이(intermediate-length) 테이블을 덧붙이는 방법으로 전달 테이블의 크기를 9 Mbyte까지 줄이는 경우에는, 메모리 접근 회수가 3번으로 증가하게 된다. 이 논문에서는 또한 파이프라인 기법의 하드웨어 방식을 사용하여 매 메모리 접근마다 라우트 검색이 가능함을 제안하고 있다.

Waldvogel 등에 의해 제안된 binary search mechanism을 기본으로 한 방법[4]은 어드레스와 라우팅 테이블의 크기가 증가되어도 scaling이 용이한 장점을 지닌다. 그러나 최악의 경우, log2 (address bits)의 해쉬 검색이 요구된다. 소프트웨어 기반의 이 방식은 multi-way, multi-column search 방식[5]과 캐쉬 구조를 이용하여 좀 더 보완될 수 있다고 하지만, 이 방식은 완전 해싱(perfect hashing)을 가정하는 것으로 인해 하드웨어로의 구현이 쉽지 않다는 단점을 지니고 있다.

Contents Addressable Memories(CAMs)을 사용한 방식은 대표적으로 사용되는 하드웨어 구현방식으로, 메모리에 저장된 값과 들어오는 키 값이 병렬적으로 직접 비교된다. 그러나 CAM 방식은 기존의 메모리보다 속도가 느리고 비용이 비싸며, 저장공간이 작은 편이다. 따라서 128 비트 어드레스를 사용하는 IPv6에서 CAM 방식을 적용하기에는 비용 부담이 너무 크다는 단점을 지닌다.

앞에서 정리한 여러 방법들을 잘 살펴보면, 라우팅 방식을 정하는데 있어서 고려해야 할 점들이 여럿 눈에 뜨인다. 메모리 접근 횟수가 평균 어느 정도 되는가, 어느 정도의 저장공간을 요구하는가, update는 용이한가, preprocessing을 요구하지는 않는가, 하드웨어 구현은 가능한가 등을 고려해야 할 것으로 보인다.

본 논문에서는 IP 어드레스 검색을 위하여 MAC (Media Access Control)어드레스의 exact matching을 위해 활발하게 사용되어 온 해싱 기법을 병렬적으로 적용하기 위한 방식을 제안한다. 먼저 여러 개의 SRAM을 사용하여 프리픽스 길이 별로 여러 개의 어드레스 검색 테이블을 만들고, 각 테이블에 대하여 해싱 기법을 적용하여 동시에 어드레스 검색을 수행한 후, 각 테이블에서 일치되어 나온 엔트리 중 가장 길게 프리픽스가 일치하는 엔트리를 고르는 방식이다. 해싱 하드웨어는 프리픽스 길이 별로 별도의 하드웨어가 필요하므로, 하드웨어의 부담이 아주 작은 Exclusive-OR 논리를 사용하였다. Exclusive-OR를 이용한 해싱 하드웨어는 완전 해싱(perfect hashing)[7]을 보장하지 않으므로 충돌 현상을 일으킬 것이고, 이러한 충돌 현상이 있을 때의 어드레스 검색을 위해서는 binary 검색을 사용하는 방식을 제안하였다.

본 논문은 다음과 같이 구성되어 있다. 먼저 II장에서는 본 논문에서 제안하는 방식과 이의 구현을 위한 하드웨어 구조를 설명한다. III장에서는 MAE-WEST 라우터의 예를 사용하여 본 논문에서 제안하는 구조의 성능을 평가하고 다른 구조들과 비교한다. IV장에서는 간단한 결론을 맺는다.

II. 제안하는 구조

가장 직관적인 어드레스 검색 구조는 IP 어드레스 자체를 메모리의 포인터로 사용하는 것이라 할 것이다. 다시 말하면, IP 목적지 어드레스가 32 비

트이므로 232개의 엔트리를 갖는 메모리를 설정하여, 임의의 패킷이 들어왔을 때, 패킷의 목적지 IP 어드레스를 키로 사용하여 목적지 IP 어드레스가 가리키는 메모리의 엔트리에 그 패킷에 해당하는 전달 정보(forwarding information)를 저장하는 것이다. 그러나 이 방식은 한 엔트리의 크기를 3 byte라 가정했을 경우 12 Gbyte 라는 엄청난 양의 메모리를 필요로 하므로 실용적이지 못하다.

앞 절에서 소개한, 기존에 연구되어 왔던 여러 어드레스 검색 구조 중에서 Waldvogel 등에 의해 제안된 검색 구조[4]에서는 프리픽스 길이 별로 구성된 해쉬 테이블들을 linear search 하는 방법을 제시하였다. 이 검색 구조는 가장 긴 길이를 가지는 것에서부터 차례로 linear search를 수행하여 best matching prefix(BMP)를 찾아내는 방식이다. 그러나 Waldvogel의 논문에서는 완전 해쉬을 가정하여 충돌 현상의 발생에 대한 고려를 전혀 하지 않고 있다. 그러나 실제로 완전 해쉬을 위한 하드웨어의 구현은 거의 불가능하기 때문에 실제 사용 가능한 해쉬 하드웨어로 구현을 하게 되면 충돌 현상에 대해 충분히 고려를 해주어야만 한다. 그리고 linear 방식으로 검색을 진행하게 되면 최악의 경우 W가 서로 다른 프리픽스 종류라 할 때, O(W) 만큼의 시간이 필요하게 되므로 32 비트의 IP 어드레스 검색을 하는데 최대 32 번의 검색횟수가 요구된다.

본 논문에서 제안하는 구조는 먼저 32 비트의 IP 어드레스는 8 비트에서 32 비트의 프리픽스들을 갖으며[6], 해쉬 함수는 하드웨어를 사용하여 간단하게 구현될 수 있다[7]는 데에서 착안하여, 각각의 프리픽스별로 별도의 해쉬 하드웨어, 별도의 주 테이블과 별도의 보조 테이블을 갖는 구조로서, 각 테이블에서 병렬적으로 검색을 수행하여 일치되는 엔트리 중에서 가장 길게 일치되는 엔트리를 택하는 방식이다. 이러한 구조를 택함으로 해서 longest prefix matching 의 문제가 exact matching 문제로의 전환이 가능하게 되었다.

그림 1을 통하여 제안하는 구조를 살펴보면, 입력으로 사용되는 각각의 프리픽스 길이별로 별도의 해쉬 하드웨어에서 나온 값은 주 테이블 중 하나의 엔트리를 가리키고, 주 테이블에 저장된 정보 중 일부분이 또 다시 보조 테이블을 가리키게 된다. 주 테이블과 보조 테이블은 하나의 SRAM으로 구성될 수 있다. 즉, 프리픽스 길이별로 어드레스 검색 테이블이 따로 구성 되고, 각각의 SRAM에 저장된 테이블에서 어드레스 검색이 병렬적으로 수행된다.

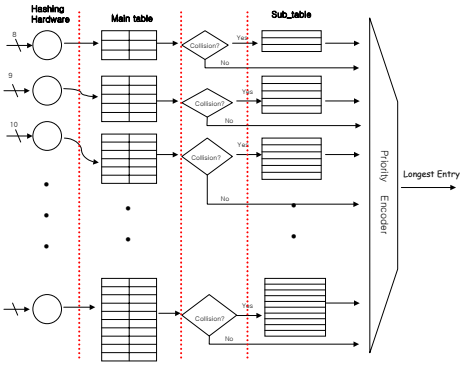

그림 4 제안한 구조의 전체 블록 다이어 그램

제안하는 구조를 수행하는 알고리즘은 다음과 같다.

```
Function Search_Prefix // 입력된 어드레스 D와
                       일치되는 prefix search
Do parallel ( L = 8 ~ 32 )
   Extracts the first L bits of D into D´;
   Table_pointer = Hash( D´);
         // Hash( D´)은 Exclusive-OR 수행하고
         // 그 결과를 table pointer로 이용
   Compare D´and an entry value pointed
   by Table_pointer;
   if ( not same ) begin      // 충돌현상
      Use pointer to the Sub_table and
      # of entries to be searched;
      Perform Binary search for the entries;
   end
   Send Search_result and ACK
   to the Prioity Encoder;
End Do parallel
Select the entry with longest prefix;
```

그림 2에서는 한 예로 입력 프리픽스 길이가 24 비트, 그리고 계산된 해쉬 값이 14 비트인 해쉬 함수와 그와 연결된 주 테이블, 보조 테이블의 구조를 보여준다. (1)의 예시는 주 테이블에서 검색이 끝나는 경우이다. 이 경우에는 엔트리에 저장된 다음 홉 정보와 출력포트(output interface) 정보가 priority encoder로 보내진다. 그러나 해쉬 값이 가리키는 지점에 저장된 다음 홉 정보와 프리픽스가 입력된 IP 어드레스의 프리픽스 값이 아닌 다른 어드레스의

프리픽스라면, 이는 충돌현상이 발생한 경우(2)로서 그 엔트리에 저장된 정보 중 보조 테이블을 가리키는 포인터와 이 엔트리는 몇 개의 충돌 현상을 갖는지에 대한 정보를 검색에 이용해야 한다. 본 논문에서는 포인터가 가리키는 보조 테이블로 옮겨서 포인터가 가리키는 지점으로부터 충돌 현상에 해당되는 엔트리들에 대하여 binary 검색을 적용하여 찾아내는 방식을 제안한다.

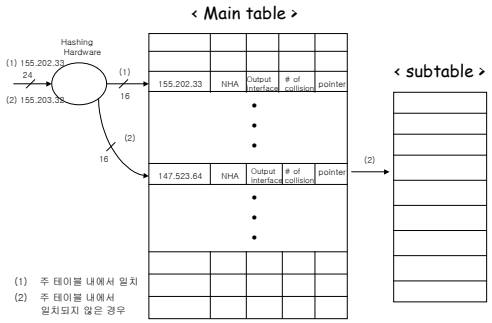

그림 6 각 SRAM 별 테이블 검색 구조
(예. 프리픽스 24의 경우)

이 모든 과정은 프리픽스 길이에 따른 SRAM별로 병렬적으로 수행된다. Priority Encoder에서는 임의의 한 IP 어드레스에 대해 모든 테이블에서의 검색이 끝나기를 기다렸다가, 프리픽스가 일치되는 엔트리 중에서 가장 긴 프리픽스 일치를 보이는 엔트리를 선택하며, 그 엔트리의 다음 홉 어드레스, 출력 인터페이스 정보가 패킷 전송에 이용된다. 또한 본 논문에서 제안한 방식은 각 프리픽스 별로 각기 다른 SRAM에 어드레스 검색 테이블을 구성하였기 때문에 하나의 메모리에 어드레스 전달 테이블을 구성하는 방식에 비해 새로운 엔트리를 추가, 삭제하는 테이블 갱신이 용이하다.

Ⅲ. 시뮬레이션 결과 및 성능 비교

본 논문에서 제안된 구조에서 요구되는 메모리의 크기 및 평균 메모리 접근 횟수 등의 성능을 평가하기 위해서 MAE-WEST 라우터의 예[6]를 가지고 시뮬레이션을 수행하였다. 해쉉 하드웨어로는 Exclusive-OR 논리[7]를 사용하였다. 그림 3에서는 2002년 3월 15일의 MAE-WEST에서 나타나는 프리픽스 길이 별 라우트 수에 대한 분포를 보여준다. 그림 3으로부터 MAE-WEST 라우터에 들어오는 IP 패킷은 프리픽스 길이가 8부터 32까지(31 제외)를 갖는 것을 알 수 있다. 주 테이블에서의 어드레스 검색과 보조 테이블에서의 어드레스 검색은 순차적으로 수행되므로 주 테이블과 보조 테이블은 하나의 SRAM에 구성될 수 있다. 그러므로 각 프리픽스 길이 별 어드레스 검색을 위해 24개의 SRAM을 사용하여 시뮬레이션을 수행하였다.

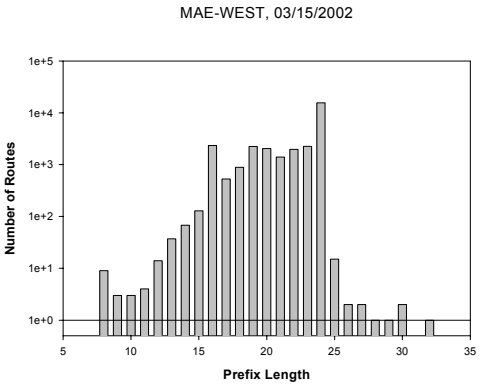

그림 8 프리픽스 길이에 따른 라우트의 분포

본 논문에서 수행한 실험에서는 MAE-WEST 분포에 나타나는 각 프리픽스 길이 별 라우트의 수에 따라 메모리의 크기를 조절하고, 이에 따라 해쉉에 사용되는 비트 수를 계산하였다.

해쉉 하드웨어는 프리픽스의 비트 수를 해쉉의 결과로 나오기를 원하는 비트 수 만큼씩 묶어서 Exclusive-OR를 수행하는 방식을 사용하였다. 만약 프리픽스 길이가 해쉉의 결과 비트의 수로 묶이지 않는 경우, 앞에서부터 결과 비트만큼 묶고 남은 비트는 결과 비트에 모자란 비트 수만큼 임의의 비트로 채운 후 Exclusive-OR를 수행하였다. 남은 비트를 채울 때에는 뒤에서부터 1010…의 방식으로 채워 넣었다. 예를 들어 프리픽스 8의 경우, 해쉉의 결과로 2 비트를 사용하므로 해쉬값(8) = [7:6]^[5:4]^[3:2]^[1:0]의 방식으로 계산을 하고, 프리픽스 14의 경우 해쉉의 결과로 6 비트가 만들어지기를 원하므로 이 때 해쉬값(14) = [13:8]^[7:2]^{[1:0], 0,1, 0,1} 과 같은 방식으로 수행한다.

주 테이블의 엔트리에는 비교에 사용되는 프리픽스, 다음 홉 주소, 출력포트, 보조 테이블로 가는 포인터, 그리고 이 해쉉 엔트리에 해당하는 충돌 횟수 등을 갖는 필드를 포함하며, 보조 테이블은 프리픽스, 다음 홉 주소, 그리고 출력포트 필드만을 포함

한다. 각 필드의 크기는 출력포트와 다음 홉 주소를 위해 28 비트를 할당하였고[8], 보조 테이블을 가리키는 포인터를 위해 13 비트, 보조 테이블에서의 충돌 횟수를 위해 4 비트를 할당하였다. 해싱 결과 비트 수는 앞서 설명한 바와 같이 메모리의 크기와 직접적인 연관을 갖는다. 실험에 사용된 전체 테이블의 총 엔트리 수는 37000여 개이다.

표 1에서는 다른 검색 방식들과의 요구되는 메모리 크기와 메모리 접근 횟수에 있어서의 성능을 비교하였다.

어드레스 검색 방식	메모리 접근 횟수 (최소/최대)	전달 테이블의 크기
SFT[2]	2/9	150KB ~ 160KB
Fast Routing -lookup Scheme[3] (DIR-24-8 -> DIR-21-3-8)	1/2 -> 1/3	33MB -> 9MB
Noble IP-Routing Lookup Scheme[8]	1/3	450KB ~ 470KB
제안한 구조	1/5	300KB

표 1 다른 검색 구조와의 성능 비교

본 구조에서 요구되는 메모리의 크기는 표 1에서 보는 바와 같이 SFT[2]를 제외하고는 가장 작은 크기의 메모리를 요구한다. 본 구조는 24개의 SRAM을 사용하므로 이에 따른 overhead도 고려되어야 하며, 각 프리픽스 길이 별로 별도의 해싱 하드웨어를 사용하여야 한다는 단점이 있으나, Exclusive-OR 논리를 사용한 해싱은 하드웨어로의 구현 부담이 적다고 할 수 있다. 또한 패킷 당 평균 메모리 접근 수는 1.93번으로서, 메모리를 많이 사용한 다른 하드웨어에 기초한 방식과 비슷한 성능을 가지는 것을 볼 수 있다.

그림 4는 메모리 접근 분포를 나타내고 있다. 분포에서 보이듯이 78% 이상의 라우트가 두 번의 메모리 접근으로, 95% 이상의 라우트가 세 번의 메모리 접근으로 검색이 가능한 것을 알 수 있다.

표 1에서 보면 본 논문에서 제안된 구조의 경우 최대 메모리 접근 횟수가 5번인 것을 볼 수 있는데, 이는 주 테이블이 상대적으로 작아 충돌 현상이 많이 발생한 경우로서, 주 테이블을 위하여 메모리를

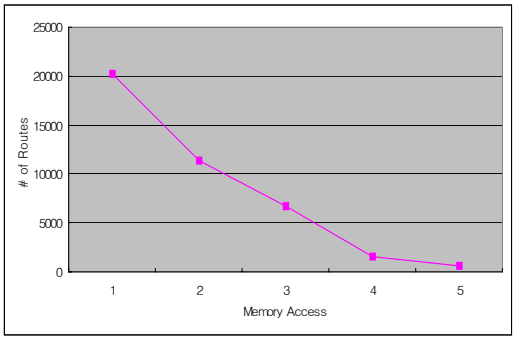

그림 10 메모리 접근 분포

좀 더 할당하는 방식으로 최대 메모리 접근 수를 조절 할 수 있다. 또한 하드웨어 파이프라인 기법을 이용하면 어드레스 검색 throughput을 향상시킬 수 있다. 예를 들어 주 테이블과 보조 테이블을 별개의 SRAM을 사용하며 구성한다면, 주 테이블에서의 어드레스 검색과 보조 테이블에서의 어드레스 검색이 순차적으로 들어오는 패킷들에 대해 병렬적으로 수행될 수 있기 때문이다.

Ⅳ. 결 론

본 논문에서는 24개의 SRAM에 프리픽스 길이별로 여러 개의 어드레스 검색 테이블을 만들고, 각 테이블에 해싱 기법을 적용하여 동시에 어드레스 검색을 수행한 후, 각 테이블에서 일치된 엔트리 중 가장 길게 프리픽스가 일치하는 것을 고르는 방식을 제안하고, MAE-WEST 라우터의 예[6]를 이용하여 시뮬레이션을 수행한 후 제안한 방식의 성능을 평가하였다. 제안된 구조는 약 300Kbyte의 저장 공간을 사용한 경우, 메모리 접근 횟수는 평균 1.93번, 최소 1번, 최대 5번의 성능을 보인다. 여러 개의 작은 메모리를 사용하여, 전체적으로 작은 저장 공간을 요구하면서도 , 적은 횟수의 메모리 접근을 통하여 어드레스를 검색할 수 있는, 하드웨어로 설계하기에 우수한 구조라고 할 것이다.

참고문헌

[1] M.A. Ruiz-Sanchez, E.W. Biersack, W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms", IEEE Network, Vol. 15 No. 2, pp. 8-23, 2001

[2] M. Degermark, A. Brodnik, S. Carls

son, S. Pink, "Small Forwarding Tables for Fast Routing Lookups", Proc. ACM SIGCOMM, pp. 3-14. 1997

[3] P. Gupta, S. Lin, N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds", Proc. IEEE INFOCOM, pp. 1240-1247, 1998

[4] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, "Scalable High Speed IP Routing Lookups", Proc. ACM SIGCOMM, pp. 25-36, 1997

[5] B. Lampson, V. Srinivasan, G. Varghese, "IP Lookups Using Multiway and Multicolumn Search", IEEE/ACM Transactions on Networking, vol. 7 No.3, pp. 324-334, 1999

[6] Merit Networks, Inc. http://www.merit.edu

[7] R. Jain, "A Comparison of Hashing Schemes for Address Lookup in Computer Networks : Technical report", Digital Equipment Corporation, 1989

[8] N. Huang, S. Ming, "A Noble IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers", IEEE Journal on selected areas in communications, vol. 17, pp. 1093-1104, 1999

서 지 현(Ji-Hyun Seo) 준회원

ever77@ewha.ac.kr

2001년 2월 이화여자대학교 정보통신학과 학사

2001년 3월~ 현재이화여자대학교 과학기술대학원 정보통신학과 석사과정

<관심분야> 통신용 반도체 설계, 컴퓨터 네트워킹을 위한 스위치/라우터 칩의 설계

임 혜 숙(Hye-Sook Lim) 정회원

hlim@ewha.ac.kr

1986년 2월 서울대학교 제어계측공학과 학사

1986년 8월~1989년 2월 삼성휴렛 팩커드 연구원

1989년 3월~1991년 2월 서울대학교 제어계측공학과 석사, 신호처리 전공

1992년 1월~1996년 12월 The University of Texas at Austin, Electrical and Computer Engineering 박사

1996년 11월~2000년 7월 Lucent Technologies Member of Technical Staff

2000년 7월~2002년 2월 Cisco Systems Hardware Engineer

2002년 3월~현재 이화여자대학교 정보통신학과 조교수

<관심분야> 컴퓨터 네트워킹을 위한 스위치/라우터 칩의 설계, TCP/IP 관련 하드웨어 설계, MPLS등.

정 여 진(Yeo-Jin Jung) 학생회원

whitehole99@hotmail.com

1999년 3월~ 현재이화여자대학교 정보통신학과 재학중

<관심분야> 컴퓨터 네트워킹을 위한 스위치/라우터 칩의 설계, TCP/IP 관련 하드웨어 설계, MPLS등

이 승 준(Seung-Jun Lee) 정회원

slee@ewha.ac.kr

1986년 서울대학교 전자공학과 학사

1989년 University of California, Berkeley 전자공학과 석사

1993년 University of California, Berkeley 전자공학과 박사

1992년 1월 ~ 1998년 10월 현대전자 시스템IC 연구소

1999년 3월 ~ 현재 이화여자대학교 전자공학과 조교수

<관심분야> 통신용 반도체 설계, CA