하드웨어 기반 집합연관 IP 주소 검색 방식

정회원 윤 상 균*

Hardware based set-associative IP address lookup scheme

Sang-Kyun Yun* ***Regular Member***

요 약

인터넷에서의 통신량이 증가함에 따라서 라우터의 고속 패킷 처리가 요구되며 IP 주소검색 이 라우터의 성능에 커다란 영향을 미친다. 인덱스 방식의 테이블을 사용하는 이전의 하드웨어 기반 IP 주소 검색 방법은 라우팅 프리픽스의 희소 분포로 인해서 메모리를 효율적으로 사용하지 못하여 메모리 요구량을 줄이는 데에 한계가 있다. 본 논문에서는 이전 방식보다 적은 양의 메모리를 가지고도 같은 IP주소 검색 속도를 제공하는 집합연관 IP 주소 검색 방식을 제안한다. 제안된 방식은 NHA 엔트리에 프리픽스와 다음경로 정보를 함께 저장하며 목적지 IP 주소를 연관 집합의 8개의 엔트리와 동시에 비교하여 일치하는 프리픽스를 찾도록 한다. 제안된 방식의 메모리 요구량은 Lin의 방식에 비해서 반 이하로 감소하여 효율적인 하드웨어 기반 IP 주소 검색방식임을 입증하였다.

Key Words : IP address lookup, router, LPM, forwardng, hardware-based lookup

ABSTRACT

IP lookup and forwarding process becomes the bottleneck of packet transmission as IP traffic increases. Previous hardware-based IP address lookup schemes using an index-based table are not memory-efficient due to sparse distribution of the routing prefixes. In this paper, we propose memory-efficient hardware based IP lookup scheme called set-associative IP address lookup scheme, which provides the same IP lookup speed with much smaller memory requirement. In the proposed scheme, an NHA entry stores the prefix and next hop together. The IP lookup procedure compares a destination IP address with eight entries in a corresponding set simultaneously and finds the longest matched prefix. The memory requirement of the proposed scheme is about 42% of that of Lin's scheme. Thus, the set-associative IP address lookup scheme is a memory-efficient hardware based IP address lookup scheme.

Ⅰ. 서 론

인터넷에서 통신량 증가 및 이에 따르는 통신회선의 대역폭 증가에 따라서 네트워크 라우터의 고속 패킷 처리가 필요하게 되었다. 라우터의 성능이 전체 인터넷 성능에 커다란 영향을 미치며, 라우터의 성능에 가장 영향을 미치는 요소는 IP 주소 검색을 수행하는 포워딩(forwarding) 엔진이다.

IP 주소는 네트워크 주소 부분과 호스트 주소 부분의 계층 구조를 가지는 데 과거의 클래스 기반 주소 방식에서는 네트워크 주소 부분, 즉 프리픽스(prefix)의 길이가 클래스에 따라서 8, 16 또는 24 비트로 정해져 있으며 IP 주소에 따라서 클래스가 정해져 있어서 IP 주소 검색이 간단하였다. 그렇지

* 연세대학교 문리대학 컴퓨터정보통신공학부(skyun@dragon.yonsei.ac.kr)
논문번호 : KICS2005-03-134, 접수일자 : 2005년 3월 31일
* Department of Computer and Telecommunication Engineering, Yonsei University
※ 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2003-041-D00507)

만 IP 주소의 사용 효율이 낮아서 제한된 자원인 IP 주소가 낭비되는 단점이 있었다.

IP 주소의 사용량이 증가함에 따라서 IP 주소를 효율적으로 사용하기 위해서 클래스가 없는 주소 방식인 CIDR(Classless Inter-Domain Routing) 방식이 1993년에 제시되었다[1]. CIDR 방식에서 네트워크 주소 부분인 프리픽스는 임의의 길이를 가질 수 있으며 여러 개의 프리픽스를 묶어서 더 짧은 하나의 프리픽스로 표현하여 백본 라우터에서 라우팅 테이블의 크기를 줄일 수 있는 장점을 가지고 있다.

그렇지만 CIDR 방식에서는 라우팅 테이블에 패킷의 목적지 IP 주소와 일치하는, 여러 개의 프리픽스가 존재할 수 있다. 이러한 경우에 목적지 IP 주소와 가장 길게 매칭이 되는 프리픽스에 대한 경로를 다음 경로로서 선택하며 이러한 선택방식을 LPM (longest prefix matching) 방식이라고 부른다. LPM 방식은 클래스 기반 방식보다 IP 주소 검색에 많은 연산을 필요로 한다.

네트워크의 속도가 빨라짐에 따라서 라우터도 고속의 IP 주소 검색이 필요하게 되었으며 소프트웨어 만으로는 요구 속도를 만족시킬 수 없게 되었다. 이 때문에 고속 라우터는 하드웨어 기반의 IP 주소 검색을 사용한다. 여러 가지 방식의 하드웨어 기반 IP주소 검색 방법이 제시되었으며 이 방식들은 IP 주소 검색 속도의 향상과 IP 주소 검색 하드웨어에서 필요로 하는 메모리 크기를 줄이는 것 등을 목표로 하였다.

본 논문에서는 기존의 방식보다 적은 크기의 메모리를 사용하여 같은 IP 주소 검색 속도를 제공하는 하드웨어 기반의 IP 주소 검색 방법인 집합연관 IP 주소 검색 방법을 제안한다. 논문의 구성은 다음과 같다. 2장에서는 기존의 관련된 연구를 소개하고 문제점을 제시하며, 3장에서는 제안하는 방식인 집합연관 IP 주소 검색 방법을 설명한다. 4장에서는 제안된 방법에 대한 평가를 하고 5장에서 결론을 맺는다.

Ⅱ. 관련 연구와 문제점

IP 주소 검색은 입력 패킷에 대해서 라우팅 테이블을 참조하여 목적지 IP 주소로 전달하는 다음 홉(next hop)이라고 불리는 다음경로 정보를 얻는 과정이다. IP 주소 검색의 여러 방법에 대해서는 여러 문헌[2, 3]에서 소개되었으며 본 논문에서는 하드웨어 기반 IP 주소 검색 방법의 연구에 대해서 소개한다. 하드웨어 기반 IP 주소 검색 방법은 TCAM (ternary content address memory) 기반 방법, 해싱 (hashing) 기반 방법과 트라이(trie) 기반 방법으로 구분된다.

TCAM 기반 방법은 don't care 비교를 허용하는 CAM인 TCAM을 사용하는 방식으로서 TCAM에 라우팅 테이블을 저장한다[4]. 목적지 주소는 TCAM에 저장된 모든 엔트리의 프리픽스와 동시에 비교되어 가장 길게 일치하는 엔트리가 선택되며 선택된 엔트리에서 다음경로 정보를 얻는다. TCAM은 검색 속도가 매우 빠른 장점이 있으나, 가격이 비싸며 전력 소모가 매우 크며 기억 용량이 작아서 큰 라우팅 테이블을 저장할 수 없는 단점을 가지고 있다.

해싱 기반 방법은 프리픽스를 해싱 함수를 사용하여 짧은 길이의 값으로 변환하여 이 값이 가리키는 메모리의 엔트리에 프리픽스의 경로 정보를 저장하는 방식이다[5]. 이 방법은 목적지 IP 주소의 프리픽스의 길이를 미리 알지 못하므로 프리픽스 길이 별로 별도의 해싱 테이블을 만들어야 하는 단점이 있다.

트라이 기반 방법은 프리픽스 엔트리를 트리 구조의 단말노드에 저장하고 프리픽스의 비트들을 검사하면서 매칭되는 프리픽스 엔트리를 찾아가는 방법이다. 이 방법은 가장 많이 사용하는 방식으로서 소프트웨어 기반과 하드웨어 기반 IP 주소 검색 모두에 사용된다. 하드웨어 기반 검색에서는 한 번에 여러 비트를 검사하는 다중비트 트라이를 사용하여 IP 주소 검색을 2~3회에 끝낼 수 있도록 구성되어 있다.

Gupta의 24-8 DIR 방식[6]은 1단계와 2단계에서 각각 24비트와 8비트를 사용하여 테이블을 검색하는 방식으로서 검색 속도는 빠르지만 메모리 요구량이 너무 큰 단점이 있다. Huang의 방식[7]에서는 라우팅 테이블을 64K개의 엔트리를 갖는 세그먼트 테이블과 길이가 16보다 큰 프리픽스의 정보를 저장하는 NHA(next hop array)로 구성한다. 각 세그먼트에 대한 NHA 크기를 줄이기 위해서 여러 가지 압축 기법이 사용되며 최대 3번의 메모리 접근을 필요로 한다. Wang의 방식[8]에서는 같은 세그먼트 내의 프리픽스들의 공통 비트를 이용하여 NHA 크기를 줄여서 Huang의 방식보다 메모리 요구량이 작다. 이 방식 역시 최대 3번의 메모리 접근이 필요하다.

Lin은 최대 2번의 메모리 접근과 이전 방식보다

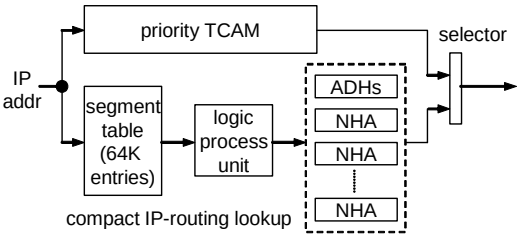

그림 1. 우선순위 TCAM IP-라우팅 검색 방식
Fig. 1. Priority TCAM IP-routing lookup scheme

적은 크기의 메모리로 IP 주소 검색을 수행하는 '우선순위 TCAM IP 라우팅 검색 방식'을 제안하였다[9]. 이 방식은 그림 1과 같이 세그먼트 테이블과 NHA의 2단계 테이블로 이루어진 compact IP-routing lookup 블록과 우선순위 TCAM으로 구성되며 길이가 24보다 큰 프리픽스 정보는 TCAM에 저장하고 길이가 24이하인 프리픽스 정보는 세그먼트 테이블 또는 NHA에 저장한다. 목적지 주소가 TCAM과 compact IP-routing lookup 블록에서 모두 프리픽스가 일치하는 엔트리를 찾은 경우에는 TCAM의 엔트리가 길이가 더 긴 프리픽스에 해당하므로 TCAM의 결과를 선택하며 이 때문에 우선순위 TCAM이라고 부른다.

표 1. 세그먼트의 프리픽스들과 NHA 크기
Table 1. Segment prefixes and NHA size

p(17:24)	NHA 인덱스	p(17:24)	NHA 인덱스
000101xx	010, 011	000101xx	001010, 001011
010101xx	110, 111	011001xx	110010, 110011
0100010x	100	0100100x	100100
공통비트: 4비트(17, 19, 21, 22번째 비트) 인 덱 스: 3비트 NHA 크기: 8 엔트리		공통비트: 1비트(17번째 비트) 인 덱 스: 6비트 NHA 크기: 64 엔트리	

Lin 방식을 비롯한 이전 방식에서는 프리픽스 정보를 NHA에 저장할 때에 인덱스 방식을 사용한다. 프리픽스의 상위 16비트를 세그먼트 테이블 인덱스로 사용하고 나머지 부분을 NHA의 인덱스로 사용한다. Lin 방식에서는 길이가 24보다 큰 프리픽스는 TCAM에 저장되므로 NHA의 인덱스의 크기는 최대 8비트이다. 인덱스 크기는 세그먼트 내의 프리픽스들 중 최장 길이를 갖는 프리픽스에 의해서 정해지며 길이가 짧은 프리픽스들은 최장 길이에 맞추어 확장되어 하나의 프리픽스가 여러 개의 인덱스에 대응되므로 메모리 낭비의 요인이 된다. Lin 방식에서는 공통비트를 최대한 이용하여 NHA 크기를 줄이는 방법을 사용하지만 NHA를 인덱스 방식으로만 구성하는 경우에 메모리의 낭비는 줄이는 데에는 한계가 있다.

인덱스 방식의 NHA에서의 메모리 낭비를 보이기 위해서 3개의 프리픽스를 가지고 있는 두 세그먼트를 예로 들었다. 표 1은 3개의 프리픽스를 갖는 세그먼트에서 프리픽스들의 값에 따른 NHA 크기를 보여준다. 여기서 p(x:y)는 프리픽스의 왼쪽 x번째부터 y번째까지의 비트열을 나타낸다. 왼쪽의 첫번째 예는 공통비트가 많아서 NHA 크기가 8이지만 오른쪽의 두 번째 예는 공통비트가 1비트이어서 NHA 크기는 64가 된다. 이처럼 프리픽스의 개수가 적더라도 프리픽스의 값의 분포에 따라서 NHA 크기가 커질 수 있다.

인터넷에 연결되는 장치들이 급격히 증가하고 있으며 이에 따라서 라우팅 프리픽스 수도 함께 증가하고 있을지라도 프리픽스는 여전히 희소 분포를 보이고 있다. 현재의 백본 라우터에서는 10만개 이상의 프리픽스가 존재한다[10]. 이들을 분석해보면 2^{16}개의 세그먼트 중에서 약 35% 만이 프리픽스를 가지고 있으며, 프리픽스가 있는 세그먼트 중에서 약 35% 만이 NHA를 가지고 있다. 그리고 NHA를 가진 세그먼트 중에서 50% 이상은 8개 이하의 프리픽스를 가지고 있으며, 전체 프리픽스들 중 대부분의 길이가 24 이하이다. 이처럼 적은 수의 프리픽스를 가지는 세그먼트가 다수를 차지하며 이러한 소규모 세그먼트에 대한 NHA 크기를 줄이는 것이 전체 메모리 요구량을 줄이는 데 매우 중요하다.

본 논문에서는 라우팅 테이블을 인덱스 방식으로 구성할 때에 발생하는 메모리 낭비를 줄이기 위해서 적은 수의 프리픽스를 갖는 세그먼트에 대해서는 NHA를 인덱스 방식으로 구성하는 것 대신에 CAM과 비슷한 방식으로 엔트리를 저장하고 여러 개의 엔트리와 동시에 비교를 하도록 하는 집합연관 IP 주소 검색 방법을 제안한다.

Ⅲ. 집합연관 IP 주소 검색 방식

3.1 IP 주소 검색 방식의 구조

본 논문에서 제안하는 집합연관 IP 주소 검색 방식의 구조는 그림 2와 같다. 기본 구조는 Lin이 제안한 그림 1과 같은 '우선순위 TCAM IP 주소 검색 방식'을 따르지만 'compact IP-routing lookup

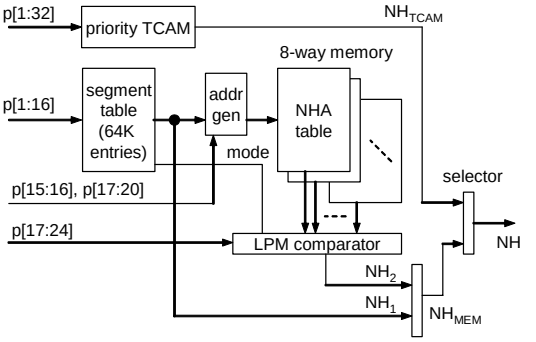

그림 2. 집합연관 IP주소 검색 방식의 구조
Fig. 2. Organization of set-associative IP address lookup scheme

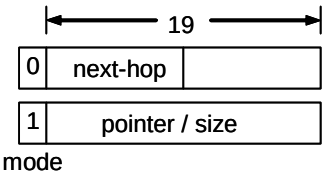

그림 3. 세그먼트 테이블 엔트리 형식
Fig. 3. Segment table entry format

블록' 대신에 새로 제안하는 '집합연관 IP 주소 검색 블록'을 사용한다. 이 블록은 세그먼트 테이블, 8-way 메모리와 LPM 비교기를 포함한 부수 장치들로 구성된다.

길이가 24보다 큰 프리픽스들의 정보는 모두 TCAM에 저장되며 집합연관 IP 주소 검색 블록에는 길이가 24이하의 프리픽스들의 정보만 저장된다. 세그먼트 테이블은 2^{16}개의 엔트리를 포함하며 엔트리는 그림 3과 같은 형식을 사용하며 각 세그먼트에 대한 다음경로 정보 또는 NHA에 대한 포인터가 저장되어 있다. 길이가 16보다 큰 프리픽스가 존재하지 않으면 세그먼트 테이블에 다음경로 정보가 저장되고, 그렇지 않으면 NHA에 다음경로 정보가 저장되며 세그먼트 테이블에는 NHA의 위치를 나타내는 포인터가 저장된다. 세그먼트에 속한 프리픽스의 개수와 값의 분포에 따라서 NHA의 크기가 영향을 받으며 NHA 크기 정보가 포인터와 함께 저장된다.

세그먼트에 속한 프리픽스의 정보들이 NHA에 저장이 될 때에 각 프리픽스들은 8개의 NHA 엔트리로 구성된 집합과 연관이 되어 이 집합의 엔트리에 저장된다. NHA 테이블은 8-way 메모리로 구성되어 집합에 속한 모든 엔트리들은 동시에 접근할 수 있다. IP 주소 검색을 할 때에 목적지 IP 주소는

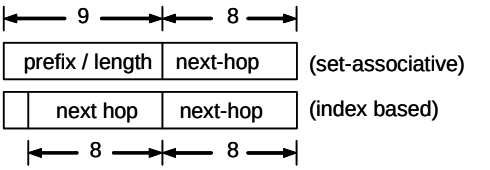

그림 4. NHA 엔트리 형식
Fig. 4. NHA entry format

그 주소와 연관된 집합의 모든 엔트리와 동시에 비교를 하게 된다. NHA 엔트리는 그림 4와 같이 17비트 크기이며 집합연관 방식으로 사용될 때에는 9비트의 프리픽스/길이와 8비트의 다음경로 정보를 저장하며, 프리픽스가 많은 경우에는 인덱스 기반 방식으로 사용되어 두 개의 8비트 다음경로 정보를 저장한다.

세그먼트가 8개 이하의 프리픽스를 가지면 관련된 NHA는 하나의 집합으로 이루어지며 모든 프리픽스는 이 집합에 저장된다. 프리픽스의 수가 8개보다 많으면 더 많은 집합을 필요로 하며 이러한 경우에 세그먼트는 프리픽스의 개수와 값의 분포에 따라서 2, 4, 8 또는 16개의 집합을 이루어진 NHA를 사용한다. 8개 이하의 집합 크기의 NHA는 집합연관 방식을 사용하며 16개의 집합 크기의 NHA는 인덱스 방식을 사용한다.

세그먼트가 여러 개의 집합을 사용하는 경우에 세그먼트에 속한 프리픽스는 그들 중 하나의 집합과 연관되며 연관집합은 프리픽스의 일부 비트 값에 의해서 정해진다. 연관집합의 결정에 사용되는 비트의 위치는 프리픽스를 가능한 한 균일하게 각 집합에 분산시킬 수 있도록 정해야 한다. 비트 위치를 잘못 선택하면 다른 집합에는 엔트리의 여유가 있을지라도 일부 집합에 8개보다 많은 프리픽스가 연관되어 이 프리픽스들을 모두 저장할 수 없기 때문에 집합의 개수를 증가시켜야 하는 경우가 발생할 수 있다. CIDR report 사이트[10]에서 얻은 프리픽스 데이터를 사용하여 분석한 결과를 바탕으로 하여 본 논문에서는 2개의 집합 크기의 NHA에 대해서는 p[19]가, 4개의 집합 크기의 NHA에 대해서는 p[19:20]이, 8개의 집합 크기의 NHA에 대해서는 p[19:21]이 연관 집합 결정에 사용되도록 정하였다. 이에 대한 분석 결과는 4장에서 제시한다. 프리픽스의 길이가 작아서 해당 위치의 값이 don't care 이면 프리픽스는 비트값이 0과 1인 두 집합에 모두 연관이 되어 두 집합에 모두 저장된다.

예를 들어서 세그먼트 192.168의 NHA가 4개의

표 2. 포인터/크기 필드 형식과 NHA 크기와의 관계
Table 2. Relationship between pointer/size field format and NHA size

세그먼트테이블		NHA 테이블			
pointer (비트)	size	NHA 크기	집합 선택	엔트리 선택	방식
17	11	1/4 집합	pointer	p[15:16]	집합 연관
17	01	1/2 집합	pointer	p[16]	
17	10	1 집합	pointer		
16	100	2 집합	pointer, p[19]		
15	1000	4 집합	pointer, p[19:20]		
14	10000	8 집합	pointer, p[19:21]		
13	100000	16 집합	pointer, p[19:22]	p[17:18] p[23:24]	인덱스 기반

집합 크기인 경우에 p[19:20]이 연관 집합 결정에 사용되어서 두 프리픽스 192.168.20/22 (p[17:24]=000101xx)와 192.168.180/24 (p[17:24]=10101010)은 각각 집합 1과 집합 2에 연관되며 프리픽스 192.168.160/19 (p[17:24]=101xxxxx)는 20번째 비트가 don't care이므로 집합 2와 3에 동시에 연관된다.

메모리 공간의 추가적인 절약을 위해서 프리픽스가 매우 적은 경우에 세그먼트가 집합의 8개 엔트리들 중에서 2개 또는 4개만 사용할 수 있도록 하였다. 이렇게 하면 적은 수의 프리픽스를 갖는 여러 개의 세그먼트들이 한 집합을 공유할 수 있다. 한 집합 내에서 세그먼트가 사용할 수 있는 엔트리 위치는 세그먼트의 일부 비트값에 의해서 정할 수 있으며 본 논문에서는 4개의 엔트리를 사용하는 경우는 p[16]을, 2개의 엔트리를 사용하는 경우는 p[15:16]을 사용한다.

16개 집합 크기의 NHA은 기존 방식들과 같이 인덱스 방식을 사용하여 엔트리를 저장한다. 엔트리에는 8비트의 다음경로 정보만 저장하고 프리픽스 값에 따라서 엔트리 위치가 정해진다. 각 집합은 16개의 8비트 엔트리를 저장하므로 16개 집합 크기의 NHA는 총 256개의 엔트리를 저장할 수 있다. 프리픽스의 p[19:22] 값은 집합 위치를 결정하고 p[17:18]와 p[23:24]는 집합 내에서 8비트 엔트리 위치를 정한다. 이처럼 프리픽스의 p[17:24]에 의해서 NHA의 8비트 엔트리가 선택된다.

3.2 간결한 엔트리 형식

NHA의 크기는 1/4, 1/2, 1, 2, 4, 8, 16 집합의 7 종류가 있다. 크기를 나타내려면 3비트가 필요된다. 그런데 집합의 개수가 2개 이상일 때에 NHA는 집합 크기에 맞추어서 정렬된 위치의 메모리를 할당받아서 포인터의 최하위 비트들은 0이 된다. 값이 0인, 포인터의 최하위 비트들의 위치를 NHA의 크기를 나타내는 데에 사용하면 포인터 크기에서 2비트를 추가하여 크기 정보를나타낼 수 있다. 표 2는 각 NHA 크기에 대한 세그먼트 테이블 엔트리의 포인터/크기 필드 값과 NHA 내에서 집합 및 엔트리를 선택하는 데 사용되는 비트 위치를 보여준다. 세그먼트 테이블에서 NHA 테이블의 집합을 선택하는 데 17비트 포인터를 사용하는 데, 집합의 크기는 8개의 17비트 엔트리 크기인 136 비트(17 B)이므로 17비트 포인터는 최대 2.125 MB(=17×217 B) 크기의 NHA를 지원할 수 있다. 4장에서 제시되는 실제로 필요한 NHA의 총 크기는 이보다 훨씬 작다.

NHA의 집합연관 방식을 사용하는 엔트리 형식에서 프리픽스와 길이 정보를 그림 4에서와 같이 9비트 크기로 함께 나타내었다. 저장해야하는 프리픽스의 길이는 최소 0비트부터 최대 8비트 크기의 9가지이며 이를 나타내기 위해서는 4비트가 필요하다. 그런데 프리픽스의 길이가 짧은 경우에 사용하지 않는 나머지 비트들을 활용하면 1비트를 추가한 9비트로 프리픽스와 길이 정보를 함께 나타낼 수 있다. 예를 들어서 m비트 프리픽스에 대해서 나머지 (9-m)비트를 10...0으로 나타낸다. 표 3은 각 프리픽스 길이에 대한 프리픽스/길이 필드의 값을 보여준다. 길이가 16비트 이하인 프리픽스는 17비트 이상의 부분이 없으므로 NHA 엔트리에 모든 비트를 0으로 나타내는 0비트 프리픽스 필드 값을 저장한다. 이 프리픽스는 해당 세그먼트에서 길이가 더 긴 프리픽스와 일치되지 않는 IP 주소에 대한 기본(default) 경로를 제공하는 데 사용된다.

표 3. 프리픽스/길이 필드의 형식
Table 3. Prefix/length field format

프리픽스 총 길이	프리픽스 필드	길이 필드
24 비트	8 비트	1
23 비트	7 비트	10
22 비트	6 비트	100
…	…	…
18 비트	2 비트	1000000
17 비트	1 비트	10000000
16 비트	0 비트	00000000

3.3 IP 주소 검색 동작

집합연관 IP 주소 검색 방식에서 집합연관 IP 주소 검색 블록은 다음과 같이 입력된 패킷에 대해서 IP주소 검색을 수행하여 다음경로 정보인 NH값을 얻는다.

1단계: 패킷의 목적지 IP 주소의 p[1:16]이 가리키는 세그먼트 테이블 엔트리를 읽는다.

1.1 mode가 0이면 엔트리에 저장된 값은 다음경로 정보로서 이 값을 NH값으로하고 이 블록에서의 검색 동작을 종료한다.

1.2 mode가 1이면 엔트리에 저장된 값은 NHA 포인터/크기 정보로서 표 2에서 제시한 대로 포인터와 주소의 일부 비트를 사용하여 NHA의 연관집합의 주소를 얻고 2단계 넘어간다.

2단계: 1단계에서 얻은 주소의 집합에 있는 8개의 엔트리를 동시에 읽는다.

2.1 NHA의 크기 필드가 '100000' (16 집합)이 아닌 경우에 목적지 주소를 8개 엔트리의 프리픽스 값들과 동시에 비교하여 가장 길게 일치하는 LPM 엔트리를 결정한다. 비교되는 비트 수는 길이 필드의 패턴에 의해서 정해진다. 1/4 또는 1/2 집합과 같이 집합의 일부를 사용하는 경우에는 주소의 p[16] 또는 p[15:16]에 의해서 선택되는 엔트리의 비교 결과만 사용한다. 선택된 LPM 엔트리의 다음경로 정보가 NH로서 출력된다.

2.2 NHA의 크기 필드가 '100000'이면 읽혀진 16개의 8비트 엔트리 값들 중에서 주소의 p[17:18]와 p[23:24]의 4비트 값에 의해서 선택되는 8비트 엔트리 값이 NH로서 출력된다.

위의 동작과 함께 목적지 IP 주소는 우선순위 TCAM을 검색하여 일치하는 프리픽스를 찾는다. 일치하는 프리픽스가 있으면 해당 프리픽스에 대한 다음경로 정보를 최종 NH값으로 사용한다. 그렇지 않으면 집합연관 IP 주소 검색 블록의 1단계 또는 2단계에서 얻은 NH값을 최종 NH값으로 한다.

Ⅳ. 평 가

2005년 1월에 수집한, CIDR report 사이트에서 제공하는 AS1221, AS4637, AS6447의 라우팅 테이블의 프리픽스 정보를 평가, 분석에 사용하였다. 이 라우팅 테이블의 프리픽스 수는 모두 15만 개 이상이지만 여러 개의 프리픽스가 하나의 짧은 프리픽스로 묶여서 집약된(aggregated) 프리픽스의 개수는 약 10만 개 정도이다.

표 4. NHA 메모리 요구량의 비교
Table 4. Comparison of NHA memory requirements

		AS1221	AS4637	AS6447
프리픽스 수		104,668	103,139	115,496
프리픽스를 가진 세그먼트 수		22,492	22,447	22,482
NHA를 가진 세그먼트 수		7,801	7,760	8,190
길이가 24보다 큰 프리픽스 수		217	1	1,485
NHA 메모리 요구량 (bytes)	Lin 방식[9]	1,176,644	1,172,240	1,234,588
	제안한 방식	499,881	495,682	549,309

표 4는 Lin의 방식[9]과 본 논문에서 제안한 집합연관 방식을 적용했을 때의 NHA의 메모리 요구량을 3개의 라우팅 테이블의 특성과 함께 보여준다. 집합연관 방식에서의 NHA 테이블의 메모리 요구량은 Lin의 방식에 비해서 약 42%에 불과하다. 메모리 요구량 산출에서 다음경로를 8비트로 나타내는 것으로 가정하였는 데, Lin의 방식은 다음경로를 8비트 또는 4비트로 선택적으로 사용할 수 있다. 그렇지만 Lin방식에서 다음경로를 모두 4비트로 나타내는 경우와 비교하더라도 집합연관 방식의 메모리 요구량이 더 적다. 그리고 Lin의 방식에서는 세그먼트 테이블의 엔트리 크기가 32비트로서 집합연관 방식의 20비트보다 12비트가 더 크며 이를 고려하면 전체 메모리요구량의 차이는 96KB(=64K×12비트)가 더 커진다. 제안한 방식에서 IP 주소 검색은 메모리에 최대 2회 접근하여 수행할 수 있다. 따라서 제안한 집합연관 방식은 Lin의 방식보다 훨씬 적은 양의 메모리를 사용하면서도 같은 검색속도를 제공한다.

라우팅 테이블의 구성 및 갱신 속도도 NHA의 크기와 엔트리 개수에 비례하므로 엔트리 수가 현저하게 적은 집합연관 IP 주소 검색방식이 테이블 구성 및 갱신 속도에 있어서도 기존의 방식에 비해서 더 더 빠르다.

그림 5는 집합연관 방식에서 3개의 라우팅 테이

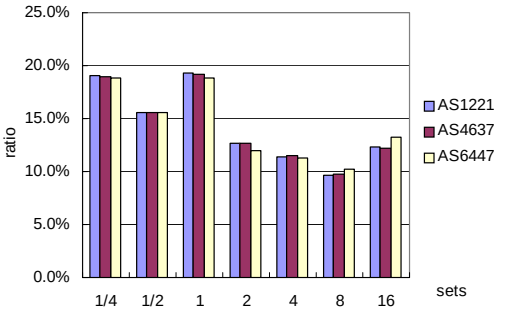

그림 5. NHA 크기의 분포
Fig. 5. Distribution of NHA sizes

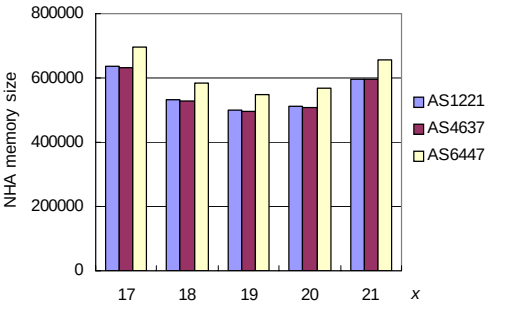

그림 6. 연관집합 선택에 사용되는 비트 위치와 메모리 요구량의 관계
Fig. 6. Relationship between selection bit locations of associative set and memory requirement.

블에 적용하여 얻은 NHA 크기 분포의 비율을 나타낸다. 적은 수의 프리픽스를 가진 세그먼트들이 많기 때문에 50% 이상의 세그먼트는 1개 집합 이하의 크기를 가지며 전체의 약 13% 만이 인덱스 방식을 사용하는 16개 집합 크기를 가진다. 이처럼 집합연관 방식은 NHA 크기를 최소화하는 데 많은 기여를 한다.

세그먼트와 연관된 NHA가 2, 4, 또는 8개의 집합으로 구성되어 있을 때에 세그먼트에 속한 각 프리픽스의 연관집합은 프리픽스의 일부 비트 값에 의해서 선택되었다. 연관집합을 선택하는 데에 사용하는 비트 위치는 프리픽스들이 비교적 균일하게 분포되어 각 집합과 연관이 되도록 선택해야 하며 이 때의 NHA 크기가 가장 작다. 그림 6은 연관집합을 선택하는 데에 집합의 개수에 따라서 IP 주소의 p[x], p[x+1], p[x+2]을 순서대로 사용할 때에 각 x값에 대한 메모리 요구량을 보여준다. 이 그림에서 x가 19일 때에 메모리 요구량이 가장 적으므로 본 논문에서는 표 2에서 제시한 바와 같이 이 비트 위치를 연관 집합 선택에 사용하였다.

Ⅴ. 결 론

인터넷의 사용이 증가함에 따라서 라우팅 테이블의 프리픽스의 개수가 10만개 이상으로 증가되었을지라도 라우팅 테이블의 많은 세그먼트가 비어있고, 비어있지 않은 세그먼트들도 대부분이 적은 수의 프리픽스로 구성되어 있는 희소 분포를 보인다. 이러한 희소 분포를 가지는 세그먼트의 NHA를 인덱스 방식으로 구성하면 프리픽스의 확장으로 인해서 메모리 요구량이 증가한다.

본 논문에서는 프리픽스 수가 적은 세그먼트는 NHA의 엔트리에 프리픽스/길이와 다음경로 정보를 함께 저장한 후에 목적지 IP 주소를 연관되는 집합에 속한 8개 엔트리의 프리픽스와 동시에 비교하여 주소 검색을 하는 집합연관 검색 방법을 제시하였다. 이 방식에서 엔트리의 크기가 이전 방식의 2배가 될 지라도 엔트리 수가 상당히 감소하여 전체적인 메모리 요구량은 반 이하로 감소하였다. 주소 검색 속도는 최대 2회의 메모리 접근이 필요하여 이전 방법과 같다. NHA 엔트리 수가 적으므로 테이블 구성 및 갱신 속도도 기존의 방법보다 빠르다. 이처럼 집합연관 IP 주소 검색 방식은 적은 양의 메모리로도 같은 검색속도와 빠른 구성 및 갱신 속도를 제공하는 효율적인 하드웨어 기반 IP 주소 검색 방법이다.

참 고 문 헌

〔1〕 V. Fuller, T. Li, J. Yu, and K. Varadhan, “Classless inter-domain routing (CIDR): and address assignment and aggregation strategy,” RFC1519, Sep. 1993.

〔2〕 M. A. Ruiz-sanchez, E.W. Biersack, and W. Dabbous, “Survey and taxonomy of IP address look up algorithms,” *IEEE Network*, 15(2), pp. 8-23, Mar.-Apr. 2001.

〔3〕 임혜숙, 정여진, “패킷 포워딩 기술,” 대한전자공학회지, 31(8), pp. 830-840, 2004. 8.

〔4〕 F. Zane, G. Narlikar; A. Basu, “Coolcams: power-efficient TCAMs for forwarding engines,” *Proc. INFOCOM 2003*, IEEE, vol. 1, pp. 42-52, Mar. 2003.

〔5〕 H. Lim, J.-H Seo, and Y.-J Jung, “High speed

IP address lookup architecture using hashing," *IEEE Communications Letters*, 7(10), pp. 502-504, Oct. 2003.

〔6〕 P. Gupta, S. Lin, and N. Mckeown, "Routing lookups in hardware at memory access speeds," *Proc. IEEE INFOCOMM 1998*, Mar. 1998.

〔7〕 N.-F. Huang and S.-M. Zhao, "A novel IP-routing lookup scheme and hardware architecture for multigigabit switching routers," *IEEE J. Selected Areas in Communications.*, 17(6), pp. 1093-1104, June 1999.

〔8〕 P.-C. Wang, C.-T. Chan, and Y.-C. Chen, "High-performance IP routing table lookup," *Computer Communications*, 25, pp. 303-312, 2002.

〔9〕 P.-C. Lin and C.-J. Chang, "A priority TCAM IP-routing lookup scheme," *IEEE Communications Letters*, 7(7), pp.337-339, July 2003.

〔10〕 CIDR report, http://www.cidr-report.org, Jan. 2005.

윤 상 균(Sang-Kyun Yun) 정회원

1984년 2월 서울대학교 전자공학과(공학사)
1986년 2월 한국과학기술원 전기 및 전자공학과(공학석사)
1995년 8월 한국과학기술원 전기 및 전자공학과(공학박사)
1984년 1월~1990년 2월 현대전자(주) 정보사업본부
1992년 3월~2001년 8월 서원대학교 전자계산학과
1998년 2월~1999년 1월 University of Michigan Ann Arbor, Visiting scholar
2001년 9월~현재 연세대학교 문리대학 컴퓨터정보통신공학부 부교수
<관심분야> 컴퓨터시스템, 컴퓨터네트워크, 라우터구조, 임베디드시스템