

論文 2000-37TC-1-9

초고속 IP 라우터를 위한 새로운 포워딩 Lookup 장치

(A Novel IP Forwarding Lookup Scheme for Fast Gigabit IP Routers)

姜昇旼 * , 宋在元 **

(Seung-Min Kang and Jae-Won Song)

요약

초고속이면서 소요 메모리의 크기를 극소화한 IP 라우터용 Lookup 알고리즘을 제안하고 성능을 분석하였다. 메모리 크기가 작으므로 고속/고가의 SRAM(10ns)을 사용할 수 있고, 구조가 간단하여 하드웨어로 구현 가능하였다. 본 장치는 1~3회의 메모리 접근을 통해 Lookup이 가능하고, IPMA 사이트에서 구한 40,000개의 라우팅 정보를 이용하여 시뮬레이션한 결과 대략 ~316KB의 포워딩 테이블용 메모리만이 소요된다. 이때 압축을 수행하는 옵션 임계치는 8이다. ALTERA EPM7256시리즈에 100MHz 클럭을 이용하여 모사시험한 결과 10ns 접근속도를 가진 SRAM 기준으로 2회의 메모리 접근만으로 Lookup하는 경우 45ns의 접근시간이 소요되며, 3회의 메모리 접근이 필요한 경우는 ~177ns의 접근시간이 소요된다.

Abstract

We have proposed and analysed a novel Lookup Algorithm which had a short switching speed and tiny memory size for IP router. This algorithm could simply be implemented by a hardware with SRAM because of simple structure. This Lookup scheme needs 1~3 memory access times. When we simulated with 40,000 routing record obtained from IPMA Website, the maximum memory size of this algorithm was 316KB(the offset threshold for compression algorithm was 8). When we simulated by HDL using ALTERA EPM7256 series and 100MHz clock and SRAM of 10ns access time, the total lookup time was 45ns for two memory access, 177ns for three memory access.

I. 서론

인터넷의 일반화는 지난 수년간에 걸쳐 급속히 진행되어 왔고, 그로 인해 인터넷의 트래픽은 급속히 증가되어 왔다. 앞으로 다양한 멀티미디어 정보에 대한 인

터넷 서비스 요구를 감안할 때 인터넷 트래픽의 증가는 지속될 것이다.

이러한 숫자 요구를 충족시키고, 인터넷 서비스의 질을 향상시키기 위해선 인터넷상에 존재하는 라우터의 발전이 수반되어야 하는데, 그 발전의 중요한 항목으로 1) 링크의 속도, 2) 라우터의 처리용량, 3) 패킷의 포워딩(Forwarding) 속도(율) 등이 부각되고 있다. 이중에 링크의 속도 문제는 광통신의 비약적인 발전으로 해결의 과정속에 있고, 라우터 처리량 문제도 라우터의 인터페이스 문제의 발전과 스위칭의 계층별 분산 기법으로 해결되고 있다.

본 논문은 3번째 문제에 대한 연구의 결과이다. 포워딩(Forwarding)이란 IP 패킷의 헤더(Header)부에 있는

* 正會員, 善隣大學 情報通信科

(Dept. of Information and communication., Sunlin College)

** 正會員, 慶北大學校 電子電氣工學部

(School of Electronical and Electric and Electronic Engineering, Kyungpook National University)

接受日字: 1999年10月8日, 수정완료일: 1999年12月2日

원격지 주소(Destination Address)를 분석하여 적절한 출력 인터페이스 링크로 해당 패킷을 전송하는 것을 지칭한다. 특히 다음 흙(Next Hop)으로의 패킷전달을 위해 최적의 메모리 구성과 접근을 포괄적으로 Lookup이라 한다. 그러나, 링크 속도의 비약적인 증가에 비해 라우터 내의 포워딩 속도는 그리 개선되지 못하고 있다.

본 연구에서는 최대 프리픽스(Prefix, 유효 네트워크 주소) 정합형 (Longest - prefix matching) 기술을 근간으로 저비용, 초고속의 새로운 Lookup 알고리즘을 제안한다. 비용 문제와 포워딩의 속도 문제는 상반되는 문제로 동시에 해결하기는 어렵지만, 가능한 이들 문제를 동시에 충족시키기자 했다. 따라서, 제안된 Lookup 알고리즘이 간단하고 소량의 메모리만을 요구 하므로 하드웨어로 가능하다. 하드웨어로 구현될 본 장치는 운영상 메모리로부터 정보 획득이 필수적이므로 메모리에 대한 접속수(Memory Access)와 메모리의 속도는 아주 중요한 변수가 된다. 그래서 가능한 메모리 접속 수가 적고 SRAM과 같은 고속의 메모리를 사용해야한다. 그러나 SRAM은 고가의 부품으로 라우터의 비용 상승 문제를 유발 할 수 있는데, 이를 위해 가능한 SRAM의 크기를 줄여야한다. 이러한 문제들이 논 논문에서 제안한 Lookup 알고리즘의 중요한 내용이다.

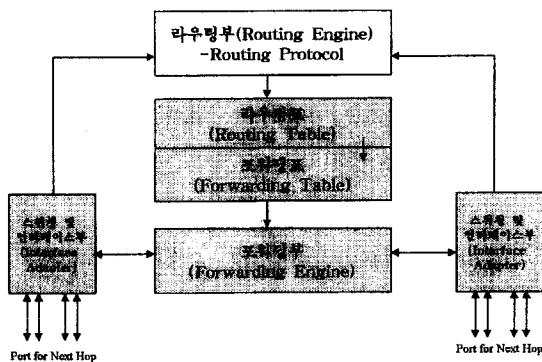


그림 1. IP 라우터 구조

Fig. 1. The structure of IP router.

II. 본론

1. 초고속 IP 라우터의 구조

일반적으로 IP 라우터는 크게 스위칭부(Switching

Fabric), 포워딩부(Forwarding Engine), 라우팅부(Fabric), 포워딩부(Forwarding Engine), 라우팅부(Routing Engine)로 구분된다. 라우팅부는 라우팅 프로토콜(RIP, OSPF와 같은...)이 실행되어, 패킷이 최적의 경로를 찾는데 필요한 정보를 저장하는 기능을 한다. 부가적으로 이들은 라우팅 정보를 포워딩부가 포워딩 알고리즘을 수행하기 위해 적절한 형태로 자료를 재가공한다. 이러한 과정은 30~60초에 한번씩 동적으로 수행되며 수행시간은 내부 CPU와 관련 프로토콜에 따라 결정된다.

포워딩부는 패킷이 입력되면 헤더부를 분석하고 이들의 전송 경로를 포워딩표(Forwarding Table)를 이용하여 신속히 결정하고 해당 출력포트로 경로를 열어주는 역할을 한다. 이는 라우터의 속도를 결정하는 부분이다. 그외 헤더부의 정보를 새로 보완(TTL, Checksum...)하고 하위 계층간의 정보 캡슐등을 수행한다. 그림 1은 일반적인 IP 라우터의 구성도이다.

2. 제안된 IP Lookup 알고리즘

IP 라우터의 Lookup부를 구현하는 방법에는 여러 가지가 있다. 우선 그림 2와 같이 32비트 IP주소 전체를 포워딩표(일종의 메모리)의 주소부로 직접 사용하는 방식이다. 이는 한번의 메모리 접근으로 원하는 정보를 얻을 수 있으므로 속도 측면에서 아주 이상적이라 할 수 있다. 그러나 이러한 직접방식은 매우 큰 메모리를 ($2^{32} = 4\text{Gbyte}$) 요구하므로 현실성이 있다.

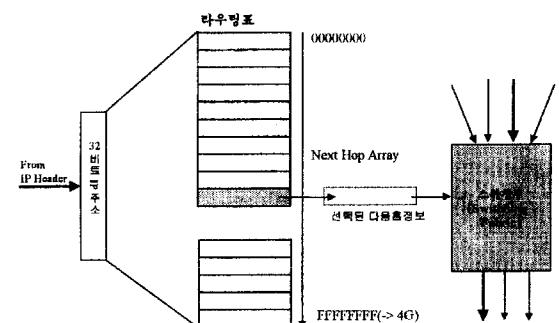


그림 2. 직접방식의 Lookup 기법

Fig. 2. Direct-lookup mechanism.

따라서, 메모리의 크기를 줄이기 위해 일반적으로 간접방식을 사용한다. 그림 3 처럼 각 IP의 주소는 두부분으로 나눈다. 1) 세그먼트(Segment : 16 bit)와 2) 오프셋(Offset : 16 bit). 세그먼트표의 크기는 64KB가 되고 세그먼트표의 내용은 출력포트의 직접적인 정보가

되거나 다음홉 배열(Next Hop Array : NHA)을 담고 있는 메모리의 초기주소를 지정하는 포인터(Pointer)가 된다. NHA의 주소로 IP주소의 오프셋부를 사용한다. NHA의 저장 자료는 다음홉의 정보가 된다. 즉, 원격지 IP 주소가 a.b.y.z 인 경우 a.b는 세그먼트표의 주소가 되고 y.z는 NHA의 인덱스(Index)로 사용된다. 세그먼트 ab에 대해 프리픽스의 길이가 16비트 이하이면 세그먼트의 내용이 직접적인 다음홉의 출력포트 정보가 되고, NHA가 불필요하게 된다. 이 경우 한번의 메모리 접근으로 패킷의 출력정보를 얻을 수 있다. Class B의 패킷이 이경우에 해당한다. 그러나, 프리픽스 길이가 16비트 이상이면, 올셀이 필요하고 하위 16비트 모두를 오셀 주소로 사용하므로 각 세그먼트별로 64KB의 NHA가 요구된다. 이 경우 IP Lookup 과정에서 최대 2번의 메모리 접근이 필요하다. 이러한 간접 Lookup 기법 중에 세그먼트부 주소를 24비트로 배당한 경우도 있다.^[11] 이 경우 세그먼트용 메모리로 16MB가 필요하다.

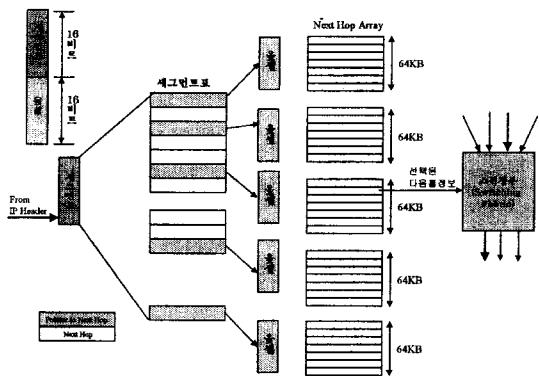


그림 3. 간접방식에 의한 Lookup 기법
Fig. 3. Indirect-lookup mechanism.

이러한 방식은 메모리의 접근 회수가 최대 2회로 한정되므로 Lookup 속도를 개선하고 동시에 메모리의 크기를 어느 정도 줄일 수 있다(직접방식에 비해)는 면에서 만족할만하다. 그러나 프리픽스의 길이가 가변적이므로 동일한 세그먼트 주소를 가진 IP들의 올셀 역시 가변적이라는 사실을 이용하지 못함으로 프리픽스가 16보다 큰 모든 경우에 대해 64KB의 NHA가 필요하게 된다. 현실적으로 동일한 세그먼트부 주소를 가진 패킷들은 세그먼트부는 같더라도 프리픽스의 길이는 다를 수 있다. 이때 동일한 세그먼트를 가진 일련의 집합에

서 가장 긴 프리픽스의 길이에 16을 뺀 수치를 올셀으로 하고(k_n), IP 주소부 중에 하위 16비트부터 $16-k_n$ 비트까지를 NHA의 주소로 사용하면, NHA의 크기를 줄일 수 있다. 그 결과 NHA의 메모리 크기는 2^{k_n} 이 된다. 일종의 가변 오셀방식(Variable Offset)이라 할 수 있다.

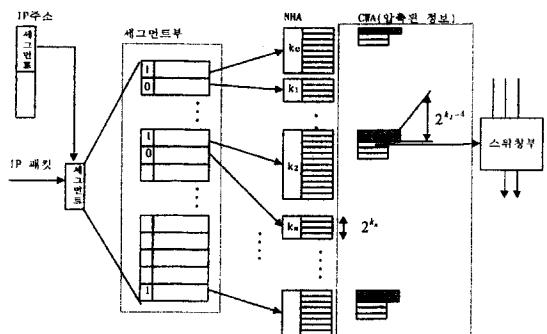


그림 4. 가변 올셀방식에 의한 간접 Lookup 기법
Fig. 4. Indirect-lookup mechanism with variable offset length.

본 알고리즘은 이상의 최대 프리픽스 정합형(Longest-Prefix Matching)형을 기반으로하면서 “가변 올셀” 기법에 “압축 알고리즘”을 적용한 복합적인 알고리즘이다. 여기에 압축된 정보의 형태를 새롭게 최적화하여 다양한 IP에 대해 메모리의 접근 회수를 최대한 줄인다.

1) 세그먼트 테이블의 구조

세그먼트는 크게 3부분으로 나눈다. 최상위 1비트는 현재의 IP에 대해 압축알고리즘 채택 여부($d=1$ 이면,

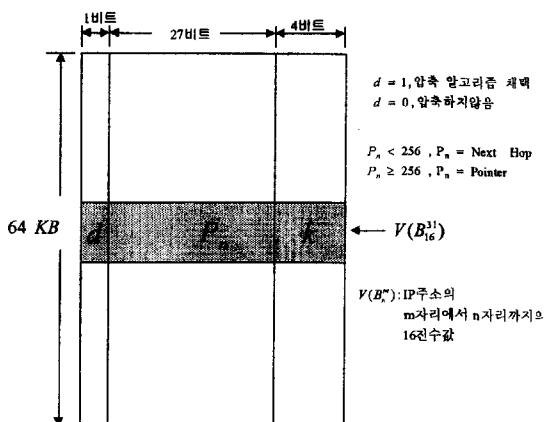


그림 5. 세그먼트 테이블의 구조
Fig. 5. The structure of segment table.

압축함)를 지시하는 비트이다. 중간의 27비트는 CWA 용 포인터이거나 다음홉정보이다. 이값이 256보다 작으면 “다음홉정보”이고 아니면 포인터가 된다. 최하위 1비트(k)는 윈셀의 길이($k+1$)를 나타낸다. 일반적인 Lookup 알고리즘에서 윈셀(k)이 크면 소요 메모리 크기가 증가하므로 압축기법을 사용한다. 그러나 본 알고리즘에서는 어떤 임계치의 k 를 미리 정하고 이값을 넘으면 압축기법을 적용하는 것이 아니라, 메모리 크기와 메모리 접근 회수, 프리픽스길이의 분포등을 고려하여 압축유무를 결정한다.

2) NHA 산출 기법

일련의 동일 세그먼트를 가진 프리픽스 집합의 NHA를 계산하는 알고리즘은 다음과 같다.

- 1) 동일 세그먼트를 갖는 프리픽스들이 라우팅 테이블로 부터 입력되면($P=\{p_0, p_1, \dots, p_{m-1}\}$)
- 2) 프리픽스 길이에 따라 순서적으로 분류한다. i 번째 IP의 프리픽스 길이를 l_i , j 번째 IP의 프리픽스 길이를 l_j 라 하면, $l_i \leq l_j$ 인 경우 p_i 를 먼저두고 p_j 를 나중에 두는 방식으로 분류한다.
- 3) 집합 P 의 “최대 프리픽스 길이”에 16을 뺀 값을 윈셀(kn)으로 정한다. 이때 NHA은 1에서 2^{kn} 까지의 일차원적 배열로 표현된다(즉, NHA는 2^{kn} 의 크기를 가진다).

4) 각 프리픽스(p_i) 별로 전체 NHA 배열내에서 해당 프리픽스의 구간(시작주소 끝주소)을 계산한다. 이 구간에 해당 프리픽스를 위한 출력포트값을 배당한다. 구간 계산방법은 다음과 같다. 해당 프리픽스가 $a.b.x.y$ 라할 때, $x.y$ 의 이진수 표현을 $x_0x_1x_2\dots x_{k-1}$ 로 한다. 윈셀이 k 라면, k 비트의 시작점 마스크 s_0, s_1, \dots, s_{k-1} 와 끝점 마스크 e_0, e_1, \dots, e_{k-1} 를 우선 다음과 같은 방법으로 구한다. k 비트 중에 자신의 프리픽스 길이보다 낮은 자리는 “1”로, 그외는 “0”으로 하여 시작점 마스크를 구하고, 자신의 프리픽스 길이보다 높은 자리는 “1”로 낮은 자리는 “0”으로하여 끝점 마스크를 구한다. 시작점 마스크와 $x.y$ 를 AND취하여 배열내의 “시작점 주소”을 구하고 끝점 마스크와 $x.y$ 를 OR취하여 “끝점 주소”를 구한다.

예를들어 $p=a.b.60.50$, 프리픽스 길이(l_i)가 26, 윈셀 k 가 12(해당 집합에서 최대 프리픽스가 28이라 가정하면, $28-16=12$ 이므로), 출력 포트가 2인 경우를 살펴본다. $k=12$ 이므로 전체 배열은 0에서 $4095(=2^{12})$ 까지 존재한다. 60.50을 12비트로 표시하면 011000000101 이다. 자

표 1. 각 프리픽스별 구간

Table 1. Interval presentation of Prefix.

프리픽스	시작점주소	끝점주소
192.168.16/1	0	65535
192.168.58/18/2	0	16383
192.168.92/24/1	23552	23807
192.168.58.32/26/3	14848	14911
192.168.255.240/28/5	65520	65535
192.168.58.36/32/8	14884	14884

표 2. 다음홉배열의 예

Table 2. Next Hop Array

주소(시작)	주소(끝)	출력포트
0	14847	2
14848	14883	3
14884	14884	8
14885	14911	3
14912	16383	2
16384	65519	1
65520	65535	5

신의 프리픽스 길이가 26이므로 $26-16 = 10$ 이므로 10비트자리 까지는 “1” 나머지 2비트는 “0”을 채워 시작점 마스크를 구한다. 그래서, 시작점 마스크는 111111111100, 끝점 마스크는 000000000011. 따라서 시작점은 60.50과 시작점 마스크를 AND취하면 011000000101 = 1540, 끝점은 1543이 된다. 전체 NHA배열중에서 이 구간(1540~1543)내에 출력포트값 2를 배당한다.

5) 4)번 과정을 전 프리픽스에 대해 적용한다. 표 1은 본 과정의 구체적인 예이다.

예를들어, 동일 세그먼트를 가진 프리픽스 집합을 (프리픽스/프리픽스길이/다음홉출력포트)의 형태로

{192.168.16/1,
192.168.58/18/2,
192.168.92/24/1,
192.168.58.32/26/3,
192.168.255.240/28/5,
192.168.58.36/32/8}

표현하면, 최대 프리픽스길이는 32 이므로 윈셀길이(k)

$(= 16 \times 32 \times 16)$ 이다. 따라서 NHA의 크기는 2^{16} 이다. 각각의 프리픽스에 대한 NHA내의 구간을 앞의 과정을 이용하여 구하면 표 1, 2 와 같다. 표 1은 각 프리픽스에 대한 시작 주소와 끝점 주소를 나타낸 것이다. 각 프리픽스별로 겹치는 부분이 있는데, 이를 일차원적으로 다시 배열하여 NHA를 표시한 것이 표 2이다.

3) 압축 알고리즘을 이용한 CWA의 유도

k 가 특정치 이상인 경우 NHA의 크기가 증가하여 메모리를 많이 소모하므로 압축기법을 적용하여 메모리의 크기를 더 줄인다. 본 Lookup 알고리즘에 적용할 압축알고리즘은 다음과 같다. 우선 압축된 정보는 CWA(Code word array)로 표현되고, 이는 다시 CBM(Compression Bit Map)과 CNHA(Compressed NHA)로 구성된다. 압축은 CBM과 CNHA를 유도하는 과정으로 시작한다. 일단 NHA를 구하고 NHA의 i번째 값을 a_i , CBM의 i번째 값을 b_i , CNHA의 j번째 값을 c_j 라 하면

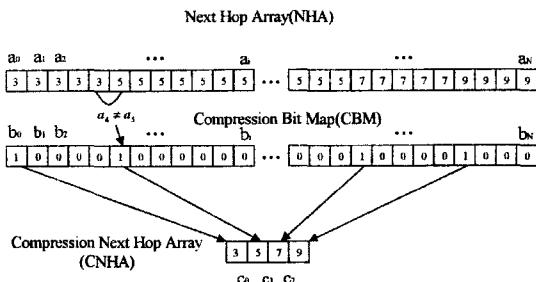


그림 6. 압축 알고리즘의 예

Fig. 6. The example of a compression algorithm.

$$\begin{aligned} b_{i+1} &= 0, \sim b_1 = 1 && \text{if } a_i = a_{i+1}, \\ b_{i+1} &= 1, \sim c_j = a_{i+1}, && j = i+1 \quad \text{if } \sim a_i \neq a_{i+1}. \end{aligned} \quad (1)$$

와같이 CBM과 CNHA를 구할 수 있다. 즉, NHA 값의 일차원적 배열을 순차적으로 스캔하면서 이웃하는 값이 서로 다를 경우 CBM의 해당 자리에 1을 배당하고 같을 경우는 0을 배당하여 압축비트맵 CBM을 작성한다. CBM이 1이될 때마다 CNHA에 해당 NHA 값을 채워나간다. 그림 6은 이러한 압축기법을 그림으로 나타냈다.

CBM과 CNHA를 이용하여 CWA(Code word Array)를 메모리상에 구현한다. CWA는 CBM을 16비트 단위로 나누어(압축비율을 16:1로 하기 위해)하여 배열하고 이를 Map으로 명명한다. 이전 워드(Word)의

Map들에 누적된 “1”的 개수를 Base라 명명하여 마찬가지로 같은 주소상에 배열한다. 그럼 8은 이러한 CWA의 구조를 나타낸다. Base는 다시 CNHA로부터 다음홉정보를 얻는데 사용된다. 여기서 유의해야 할 사항은 현재의 Map자료가 $0000_{(16)}$ 이면 Base자리에 다음홉(NextHop)의 출력포트번호를 저장한다는 사실이다. Map이 $0000_{(16)}$ 이라는 것은 이 구간에서의 다음홉이 동일한 값이라는 것을 의미한다. 따라서 굳이 Base값을 저장할 필요가 없고, 이 Base 자리에 다음홉을 저장한다. 왜냐하면 계속 이어지는 Map에서 $0000_{(16)}$ 이 아닌 최초 지점의 Base 자리만으로 누적된“1”的 정보를

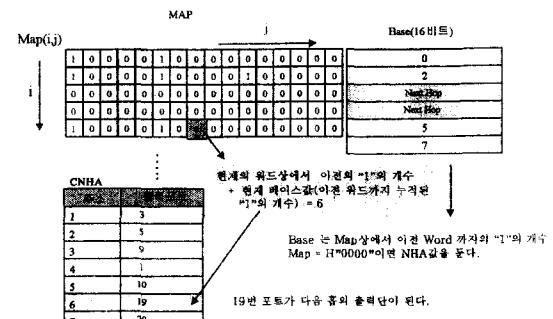


그림 7. CWA 구조

Fig. 7. The structure of CWA.

$$\begin{aligned} \text{Base}(i) &= \sum_{n=1}^{i-1} \sum_{k=1}^{16} \text{Map}(n, k), \\ &\quad \text{if } \text{Map}_i \neq 0000_{(16)} \\ &= \text{NextHop}, \quad \text{if } \text{Map}_i = 0000_{(16)} \end{aligned} \quad (2)$$

충분히 알 수 있기 때문이다. 이는 본 알고리즘의 중요 특징으로, 많은 경우에 걸쳐 2회의 메모리 접근으로 다음홉을 구할 수 있게 된다. 이렇게 Map과 Base가 완성되면 CNHA를 뒤이어 연속적으로 배열한다. 즉, 16비트 단위로 나누어 저장하였으므로 우리는 NHA를 16:1로 압축하였고, 따라서 윈셀주소로 사용하는 k비트중에 상위의 k-4비트를 CWA의 주소로 활용하여 Map 값을 읽어낸다. 만약 이때의 Map값이 $0000_{(16)}$ 이면 현재의 Base값은 다음 흙의 출력 포트값이 되고, 아니면 윈셀주소(k비트)중에 하위 4비트 주소의 십진수 값을 w(0에서 15)라 했을때, 현재의 Map의 0자리부터 w 자리까지 누적된 “1”的 개수($|w|$)를 다시 구한다. 이를 현재의 Base값에 더하여, 그 결과를 CNHA를 찾기위한 주소로 사용한다.

결과 CNHA의 주소(A_{CNHA})는

$$A_{CNHA} = P_n + 2^{k-4} + B_c + \lfloor u \rfloor \quad (3)$$

이된다. P_n 은 세그먼트에 저장된 포인터이고, 2^{k-4} 는 CWA의 최대 길이(CNHA를 제외한). B_c 는 현재 워드의 Base값이다.

4) 압축알고리즘 수행의 근거

압축알고리즘의 수행 여부는 다음 몇 가지 조건을 충족해야 한다. 첫째, 초기 설계 과정에서 배당된(확정된) 메모리 크기에 비해 만들어진(재가공된) 포워딩표의 크기가 작아야 된다. 이 조건을 충족시키기 위해 압축수행을 위한 윈셀임계치(k_t)를 조절해야 한다. 즉, 재가공된 포워딩표의 크기가 하드웨어로 결정된 메모리에 비해 여유가 있으면 압축 알고리즘을 적용할 k의 임계치를 상향 조절하여(예를들면 $k=8$ 인 경우는 프리픽스가 24비트로 인터넷상에서 네트워크 주소로 가장 많이 사용) 압축 알고리즘을 적용하지 않아 단지 2번의 메모리 접근만으로 출력포트값을 얻을 수 있는 IP영역을 확대한다. 그러나 포워딩표의 크기가 증가하면 압축 알고리즘을 적용할 임계치 k 값을 낮추어(예를들면 4이상) 많은 범위에 걸쳐 압축알고리즘을 수행한다. 이 경우 메모리 접근 횟수가 3회인 경우가 확율적으로 높아진다.

값을 윈셀임계치로 설정한다. 그럼 10에서 임계윈셀이 0이면 소요 메모리의 크기가 1MB로 급격히 증가함을 알 수 있다.

세째, 일단 압축된 CWA 중에서도 2회의 메모리 접근으로 원하는 정보를 얻을 수 있는 경우(Map=0000₍₁₆₎)와 3회의 메모리 접근으로 원하는 정보를 구할 수 있는 것으로 구분되는데, 이들간의 비율이 특정수치 이상이 되어서는 안된다. 압축을 통해 메모리의 크기를 줄이는 것은 효과적인 설계이지만 그 결과 3회의 메모리 접근을 통해서 원하는 다음Hop정보를 얻는 비율이 높아지면 초고속 스위칭이라는 최초의 목표를 달성하기가 어려워지기 때문이다. 즉, 2회와 3회의 메모리 접속을 단순히 1 회차 일뿐이라는 판단은 잘못된 것이다. 보사실험과정에서 1 회의 메모리 접속은 20ns, 2번의 메모리 접속은 45ns, 3번의 메모리 접속은 177ns정도가 소요되어(가산기, 병렬16비트 가산기의 소요 시간 때문에) 하드웨어 구현시 이들간의 차이는 확연히 들어난다. 따라서 이들간의 Lookup 비용은 달리 설정하여야 한다.

3. 하드웨어 구조

그림 8은 본 알고리즘을 구현하기 위한 하드웨어 구조이다. 우선 세그먼트 테이블을 위한 메모리부가 있다. IP 주소의 상위 16비트를 이용하여 세그먼트 테이블 자료를 읽어낸다. 포인터(P_n) 값이 256보다 작으면 이값을 곧장 출력단의 선택기를 통해 출력시킨다. 만약 256보다 크면, 이를 CWA값을 읽기위한 포인터로 사용한다. 이 상황에서 최상위 1비트(d) 값이 0이면, 압축 알고리즘을 채택하지 않은 것이므로 CWA의 Map, Base를 참조하지 않고 직접 NHA 정보를 읽어 출력시킨다. $d=1$ 이면 압축알고리즘을 채택한 것이므로 포인터와 $V(B_{15-k+4}^{15})$ 를 더한 주소에서 Map, Base를 읽어낸다. 메모리 접근을 위한 타이밍은 내부 제어기에서 발생된다. Map값이 0000₍₁₆₎이면 이때의 Base자리의 수치가 NH(Next Hop) 정보이므로 이를 출력시킨다. 0000₍₁₆₎이 아니면, 0에서 $V(B_{15-k+4}^{15})$ 자리까지 “1”로 채운 마스크 바이트를 만들고 이를 Map(M_m)과 AND한다. 이 결과치 내부에 존재하는 “1”的 개수를 구하기 위해 병렬 가산기(16개의 1비트 가산기)를 구현한다. 병렬 가산기의 결과를 현재의 Base(B_m), 포인터(P_n)와 더하여 새로운 CWA의 주소를 만든다. 이 주소를 이용하여 CNHA에 접근하여 자료를 읽어내어 출력한다.

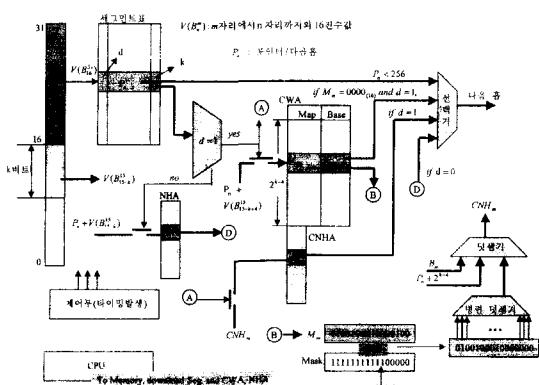


그림 8. 하드웨어 구조

Fig. 8. The hardware structure .

둘째, 라우팅 테이블을 이용하여 포워딩 테이블을 재가공할 때 압축을 수행하는 임계 윈셀의 값을 증가시켜본다. 그러면 소요 메모리가 점차 증가하지만, 그 증가치는 일정 구간에서 미약하다. 임계윈셀이 특정 값이 되면 갑자기 소요 메모리가 급격히 증가하는데, 이때의

III. 성능분석

1. 메모리 크기와 메모리 접근회수

본 알고리즘의 성능을 분석하기 위해 현재 운용중인 IP 라우터의 라우팅테이블 정보가 필요하다. 이는 IPMA(Internet Performance Measurement and Analysis) 프로젝트의 웹사이트^[4]에서 구할 수 있다. IPMA

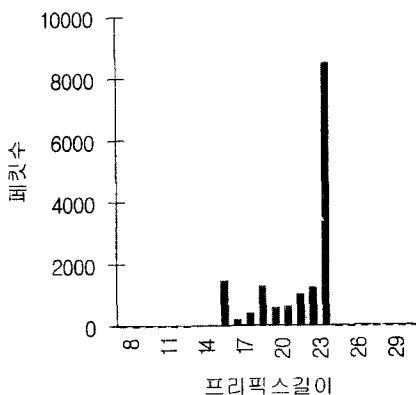


그림 9. 프리픽스 길이별 분포

Fig. 9. The distribution of Prefix length.

표 3. 여러 Lookup 기법들의 비교

Table 3. The comparison of Lookup schemes.

Lookup기법	메모리 접근수	비율	포워딩 테이블 크기	SRAM사용
본 기법	1	15%	~316KB	Yes
	2	80%		
	3	5%		
DIR-24-8	1/2	50%	33MB	No
DIR-21-3-8	1/3	-	9MB	No
SFT	2/9	-	150~160KB	Yes

인터넷 사이트는 실시간으로 라우팅테이블 자료를 제공한다. 그림 9는 IPMA사이트에서 얻은 프리픽스의 길이 분포를 그래프로 표시한 것이다. 인터넷 라우터를 통해 전달되는 패킷의 프리픽스 길이는 통계적으로 24인 경우가 가장 많다(대부분 24이내이다). 따라서 이러한 근거에 맞는 라우팅 테이블을 웹사이트에서 제공받아 본 알고리즘을 검증한다.

본 알고리즘의 성능을 3가지 관련 알고리즘 -DIR-

24-8, DIR-21-3-8, SFT^{[2][3]}과의 비교를 통해 분석하고자 한다. 표 3은 기존의 기법과 본 기법간의 기준의

표 4. 포워딩테이블의 소요 메모리(단위 : KB)

Table 4. The forwarding table memory size.

월/년 날짜	4	5	6	7	8	9
'99 9 28	255	258	263	278	310	989
'99 9 29	257	260	265	280	313	997
'99 9 30	220	222	226	236	256	850
'99 10.1	259	262	268	283	316	1000
'99 10.2	258	261	267	282	315	1000

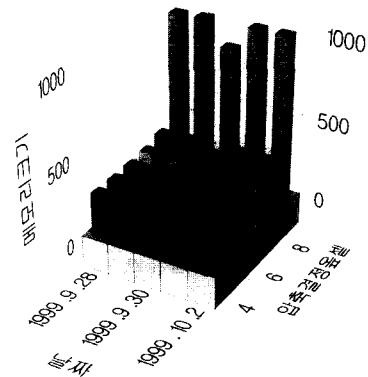


그림 10. 압축수행을 결정하는 임계값에 따른 소요 메모리의 변화

Fig. 10. The memory size distribution by threshold offset value which determine compression algorithm.

기법과 본 기법간의 비교 분석표이다. SFT(Small Forwarding-Table)^[2] 기법은 캐쉬 메모리에 저장된 Lookup 정보를 CPU를 이용하여 추적하는 방식으로서 최소의 메모리 크기를 요구하고, SRAM으로 구현 가능하지만 메모리의 접근회수가 2에서 9번까지로 초고속 Lookup에는 적합하지 않다. DIR-24-8 기법은 프리픽스를 24비트로 고정하여, 각 세그먼트당 2^{16} 비트의 NHA를 배당한 방법으로, 메모리 접근 횟수는 1에서 2번 사이로 초고속 라우팅에는 적합하나, 메모리의 크기가 33MB로 비현실적으로 크기 때문에 하드웨어로 구현하기가 불가능하다. 다. 그래서 SRAM으로 구현이 불가능하여 DRAM으로 구현할 수밖에 없는데, 이로 인

해 고속의 스위칭이 가능한 장점을 살릴 수 없다. DIR-21-3-8 기법 역시 프리피스의 길이를 조절한 DIR-24-8과 유사한 기법으로 메모리 크기를 9MB까지 줄였으나 최대 3번의 메모리 접근을 요구하고 여전히 SRAM 사용이 불가능한 기법이다. 이들 두 방법은 가변성 있는 프리피스 길이 분포에 관계없이 대용량의 메

표 5. 압축된 CWA에서 메모리 접근회수의 비교례
Table 5. The comparison example for memory access times with a compression CWA.

워드 번호	Map	Base	2회	3회	전체 N H
1	1000000000000000	0		16	16
2	0000000000000000	3(NH)	16		:
3~9	:	3(NH)	112		:
10	1000000000000000	1		16	
11	1000000000000000	2		16	
12	0000000000000000	1(NH)	16		
13~19	:	1(NH)	112		
20	000000010001000	3		16	
21	0000000000000000	4(NH)	16		
22~32	:	4(NH)	176		
33	1100000000000000	5		16	
34	0000000000000000	1(NH)	16		
35~64	:	1(NH)	480		
2^6			944	80	1024
비율			92%	8%	

모리를 요구한다. 즉, 프리피스의 길이가 24보다 작은 경우에도 각각 16, 8MB의 메모리를 요구한다. 그러나 본 기법은 1999년 9월 28일부터 1999년 10월 2일까지 IPMA사이트에서 구한 자료를 이용하여 계산한 결과 최대 316KB의 SRAM으로 구현 가능하므로 저비용으로 초고속 라우팅이 가능함을 알 수 있다. 그림 10은 1999년 9월 28일부터 10월 2일까지의 라우팅정보를 이용하여 압축알고리즘 수행을 결정하는 임계 올셀값(k_t)을 4에서 9까지 가변하면서 소요 메모리량을 구하여

표현한 것이다. k_t 를 8보다 작게하면 소요 메모리의 용량이 거의 비슷(~316KB)하지만 9로 설정하면 소요 메모리가 1MB로 크게 증가한다. 따라서 이경우는 k_t 를 8로 설정한다. 본 기법을 사용하면 95%이상 2회의 메모리 접근만으로 다음홉정보를 얻을수 있다. 그러나 만약 CWA 구조에서 Map=000₍₁₆₎인 경우를 구분하여 활용하지 않으면 2회 메모리 접근 비율이 67%로 떨어진다. 프리피스 길이가 24보다 큰 경우를 각 프리피스마다 최소 하나 이상 삽입한 라우팅 테이블을 본 알고리듬에 적용하면 소요 메모리는 최대 2MB까지 증가한다. 사실 그림 10에 알 수 있듯이 이러한 경우는 라우터에서 확률적으로 발생빈도가 낮으나 본 알고리즘의 성능을 좀 더 극단적인 경우 적용해봄으로서 그 가치를 인정받기 위해서이다.

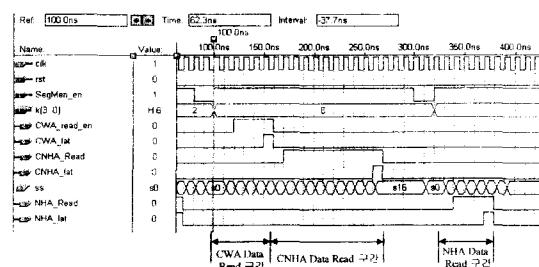


그림 11. 제어부의 타이밍
Fig. 11. The timing of controller.

표 5는 임의 프리피스 집합을 선택하여 CWA를 구한 예이다. 올셀이 10비트이므로 전체 CWA워드수는 $2^6(=64)$ 이다. 여기서 주의해야할 사항은 비록 압축하더라도 2회의 메모리 접근만으로 출력포트를 구할 수 있는 비율이 92%라는 사실이다. 이는 Map = 0000₍₁₆₎인 워드의 Base자리에 “다음홉정보”를 저장함으로 가능해진 사실인데, 이러한 기법이 전체 포워딩테이블에 적용되면 단지 2회의 메모리 접근 비율이 95%이상으로 증가하게 되어 스위칭속도가 전반적으로 증가하게 된다. 이것이 본 논문의 주요 특징이다.

2. 하드웨어의 HDL 구현과 시뮬레이션

HDL(Hardware Description Language)을 이용하여 본 논문에서 제안한 하드웨어를 모델링하고 시뮬레이션하였다. 그림 11은 제어부의 내부 타이밍을 시뮬레이션한 결과이다. 올셀에 따라 발생 타이밍이 다른데, $k=2$ 인 경우 NHA관련 타이밍이 발생하고 $k=6$ 인 경우 CWA/CNHA 메모리 접근용 타이밍이 발생된다. 그림

12^회 k=2인 경우로 10ns의 SRAM을 사용하였다고 가정하였을 때의 타이밍 결과이다. 2회의 메모리 접근으로 원하는 출력을 얻을 수 있고, 소요 시간은 100MHz를 시스템 클럭으로 사용하였을 때 37ns이다.

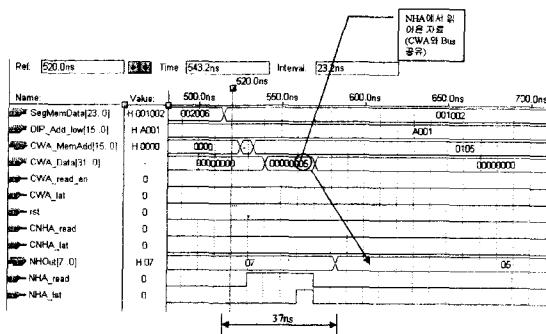


그림 12. 시스템 타이밍(k=2, 2회의 메모리 접근 과정)
Fig. 12. The system timing(k=2, two times memory access).

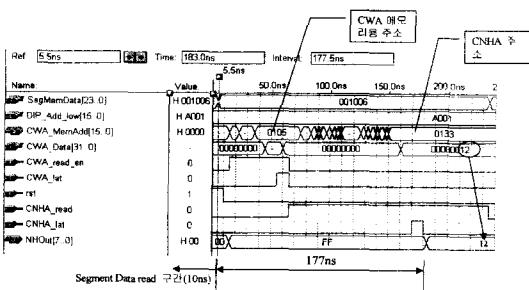


그림 13. 시스템 타이밍(k=6, Map = 0000₍₁₆₎)
Fig. 13. The system timing(k=6, Map = 0000₍₁₆₎).

그림 13은 k=6^회면서 Map값이 0000₍₁₆₎가 아닌 경우로 압축알고리즘이 적용되어 CWA와 CNHA를 동시에 이용하는 경우로 3회의 메모리 접근이 소요된다. 그림 14는 Map이 0000₍₆₎인 경우로 2번의 메모리 접근으로 원하는 출력을 얻는 경우이다. 같은 k값에 대해 2번의 메모리 접근은 45ns의 시간이 소요되지만 3번의 메모리 접근시에는 내부 16비트 병렬가산기와 여러번의 가산기의 동작 시간이 누적되어 177ns의 시간이 소요된다. 본 시뮬레이션은 ALTERA사의 MPM7256시리즈를 기초로 HDL을 이용한 결과이다.

IV. 결 론

본 연구는 메모리용량을 최소화하고, 하드웨어로 쉽

게 구현가능한 새로운 초고속 IP 라우터용 Lookup 알고리즘에 대해 소개한 것이다. 주요 특징으로 첫째, 세그먼트 테이블에 윤셀 필드를 두고, 이를 근거로 세그먼트별로 NHA의 크기를 가변하여 메모리크기를 줄였고, 둘째, 프리픽스 길이가 긴 경우 압축알고리즘을 수용하여 메모리 크기를 더욱 줄였다. 셋째, 압축 기법을 적용하여 새로 가공된 정보는 CWA/CNHA에 저장되는데, CWA에서 MAP이 0000₍₁₆₎인 경우 Base자리에 다음 흙의 포트값을 둠으로서 2회의 메모리 접근의 비율을 극대화하였다. 넷째, 압축 알고리즘의 적용을 결정하는 윤셀의 임계치를 고정하지 않고 현재의 메모리 상태와 프리픽스 길이의 분포등을 고려하여 결정하는 적응형(Adaptive) 기법을 도입하였다.

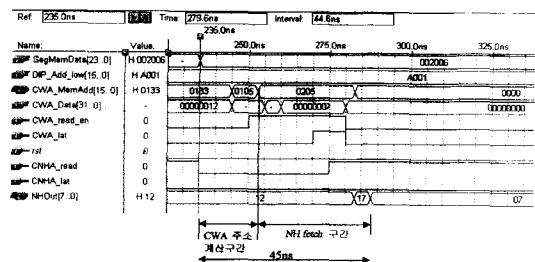


그림 14. 시스템 타이밍(k=6, Map = 0000₍₁₆₎)
Fig. 14. The system timing(k=6, Map = 0000₍₁₆₎).

그 결과 IPMA 웹사이트에서 얻은 실시간 태이밍 테이블(프리픽스 길이가 대부분 24이내)을 본 기법에 적용한 결과 ~316KB의 메모리가 소요됨을 확인하였고, 이들의 메모리 접근 회수는 2회 이내가 95%로 초고속 스위칭이 가능함을 확인하였다. 10ns의 SRAM과 ALTERA사의 EEPROM 기반의 MPM7256시리즈에 100MHz 클럭을 적용할 경우 최대 3회의 메모리 접속으로 “다음 흙 출력정보(NextHop)”를 얻을 수 있는 경우 177ns의 시간이 소요된다. 만약 1회의 메모리 접속인 경우(20ns 이므로) 초당 50×10^6 번 Lookup이 가능하다. 좀 더 빠른 ASIC기법(200MHz 이상에서 동작)을 이용할 경우 이보다 더 빠른 결과(1Gbps)를 도출할 수 있을 것이다.

참 고 문 헌

- [1] M. Doegerman, A. Brodnik, "Small forwarding tables for fast routing lookups," in Proc. ACM

- SIGCOMM'97, France.
- [2] P. Gupta, "Routing lookups in hardware at memory access speeds," IN PROC IEEE infocom'98, Session 10b-1, San Francisco, CA, pp. 1240-1247.
- [3] Nen-Fu Huang, Shi-Ming Zhao, "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers," IEEE Journal of Selected Areas in communications, Vol. 17, No. 6, June 1999, 1093-1104.
- [4] Michigan University and Merit Network, Internet Performance Measurement and Analysis(IPMA) Project [Online]. Available <http://nic.merit.edu/~ipma/>.

저자소개



姜昇旼(正會員)

1987년 경북대학교 전자공학과(학사). 1990년 경북대학교 전자공학과(석사). 1990~1996년 국방과학연구소 연구원. 1996년 3월~현재 : 포항선린대학 정보통신과 전임강사.
1999년 경북대학교 전자공학과 박사 수료. 광통신, 인터넷 라우터, 컴퓨터 네트워크

宋在元(正會員) 論文 第36卷 D編 第9號 參照