Journal of The Korea Society of Computer and Information
Vol. 20 No. 12, pp. 45-52, December 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.12.045

A Hybrid Prefix Cashing Scheme for Efficient IP Address Lookup

Jinsoo Kim*, Junghwan Kim**

Abstract

We propose a hybrid prefix caching scheme to enable high speed IP address lookup. All prefixes loaded in a prefix cache should not be overlapped in address range for correct IP lookup. So, every non-leaf prefix needs to be expanded not so as to be overlapped. The shorter expanded prefix is more preferable because it can cover wider address range just as an single entry in a prefix cache. We exploits advantages of two dynamic prefix expansion techniques, bounded prefix expansion technique and bitmap-based prefix expansion technique. The proposed scheme uses dual bound values whereas just one bound value is used in bounded prefix expansion. Our elaborated technique make the dual bound values be associated with several subtries flexibly using bitmap information, rather than with fixed subtries. We evaluate the performance of the proposed scheme in terms of the average length of the expanded prefixes and cache miss ratio. The experiment results show the proposed scheme has lower cache miss ratio than other previous schemes including both bounded prefix expansion and bitmap-based expansion irrespective of the cache size.

▸ Keyword : IP address lookup, hybrid prefix cashing, prefix expansion, bitmap, dual bounds

I. Introduction

IP 주소 검색은 라우터의 핵심 기능으로, 입력 패킷의 목적지 IP 주소와 매치되는 prefix를 라우팅 테이블에서 검색하고, 이를 바탕으로 다음 홉(next hop)과 출력 링크를 찾는 것이다[1]. IP 주소의 검색은 가변 길이의 prefix를 허용하기 때문에 목적지 주소에 매치되는 prefix를 여러 개 찾을 수 있고, 이들 중 가장 길게 매치되는 prefix인 LMP(Longest Matching Prefix)를 최종 선택한다. 또한 라우팅 테이블은 수십만 개의 prefix를 가지고 있으며 그 수가 지속적으로 늘어나고 있기 때문에, IP 주소 검색의 효율성은 라우터의 성능에 지대한 영향을 미친다.

IP 주소 검색의 성능을 향상시키기 위해, SRAM과 TCAM (Ternary Content Addressable Memory)을 사용하는 많은 기법들이 제안되었다[1-4]. 또한 기본적인 IP 주소 검색 방법으로 선택된 결과를 caching하여 속도를 더욱 향상시키는 기법들도 개발되어 왔다. Cache는 지역성(locality)을 활용함으로써 그 효율성을 증대시킬 수 있는데, IP 주소 검색에서 시간적 지역성과 공간적 지역성이 모두 활용 가능하다.

인터넷의 트래픽 특성상 라우터에 입력되는 근처 패킷들의 목적지 IP 주소 또는 목적지 네트워크 주소가 동일한 경우들이 다수 존재할 수 있다[5]. 전자의 경우 인터넷 트래픽에서 시간적 지역성이 있음을 의미하고, 후자의 경우는 공간적 지역성도 존재함을 의미한다. IP 주소 검색에서의 caching 기법도 크게 IP 주소 caching[6]과 prefx caching으로 구분할 수 있다. IP 주소 caching은 32비트 IP 주소를 cache에 적재하여 시간적 지역성만을 활용할 수 있지만, prefix caching은 네트워크 주소인 prefix를 cache에 적재하여 시간적 지역성 뿐아니라 공간적 지역성도 활용할 수 있다.

네트워크 주소는 다수의 IP 주소를 포함하고 있으므로, prefix caching이 IP 주소 caching에 비해 cache 적중률이 높

• First Author: Jinsoo Kim, Corresponding Author: Junghwan Kim
*Jinsoo Kim(jinsoo@kku.ac.kr), Dept. of Computer Engineering, Konkuk University
**Junghwan Kim (jhkim@kku.ac.kr), Dept. of Computer Engineering, Konkuk University
• Received: 2015. 11. 30, Revised: 2015. 12. 10, Accepted: 2015. 12. 22.
• This paper was written as part of Konkuk University's research support program for its faculty on sabbatical leave in 2014.

으며 성능이 우수하다. 그러나 prefix caching에서 non-leaf prefix를 caching할 경우 잘못된 검색결과를 산출하는 문제가 있기 때문에, non-leaf prefix는 그대로 caching하면 안 된다. Non-leaf prefix 문제를 해결하는 합리적인 방안은 그의 후손 prefix가 없도록 해당 prefix 길이를 확장하는 prefix 확장 기법이다[7-12].

Prefix 확장은 미리 라우팅 테이블 자체에서 확장하는 정적 확장[7, 9]과 caching할 때 확장하는 동적 확장[8, 10, 11, 12]으로 구분할 수 있다. 정적 확장은 확장된 prefix가 모두 라우팅 테이블에 미리 저장되어 있어야 하므로 라우팅 테이블의 엔트리 수가 매우 늘어나 IP 주소 검색 자체의 성능을 저하시키는 요인이 된다. 반면에 동적 확장은 prefix 확장을 위해 라우팅 테이블에 prefix를 별도로 추가하지 않기 때문에 더 효율적이다.

Bounded prefix 확장[10]와 bitmap 기반의 prefix caching[12]은 동적 확장 방식이며, 유용한 정보를 이용하여 효율적으로 prefix를 확장한다. 본 논문에서는 non-leaf prefix의 후손 prefix가 많은 경우 유리한 bound 방식과 적은 경우 유리한 bitmap 기반 prefix caching의 장점을 취한 혼합형 prefix caching 방법을 제안한다. 기존 bitmap 방식에서 긴 prefix 확장이 필요한 경우에 IP 주소 자체를 caching해야 하는 제한점을 효율적으로 해결하기 위해, 두 개의 bound 값을 사용하여 prefix를 확장한다. 제안하는 방식에서는 bitmap 정보를 최대한 활용하여 두 개의 bound 값을 유연하고 효과적으로 사용한다.

논문의 나머지 구성은 다음과 같다. 2절에서 prefix caching에 대한 기존 연구들의 기법에 대해 설명한다. 3절에서는 cache를 이용한 IP 주소 검색 구조에 대해 소개하고, 동적 prefix 확장을 이용한 기존의 prefix caching 기법으로 bound 방식과 bitmap기반 prefix caching에 대해 비교 정리한다. 4절에서는 bound 방식의 장점과 bitmap 기반 prefix의 장점을 혼합한 하이브리드 prefix caching에 대해 기술한다. 즉 하이브리드 prefix 확장 방법과 하이브리드 caching 구조에 대해 설명한다. 그리고 5절에서는 제안한 prefix caching의 성능을 기존의 prefix 확장 방법들과 비교 실험한다. 마지막으로 6절에서 결론을 맺는다.

II. Related Works

Prefix는 여러 개의 IP 주소들을 포함하는 네트워크 주소이므로, prefix caching은 IP 주소 caching에 비해 효율적이다. 하지만, prefix caching은 non-leaf prefix를 cache에 그대로 적재할 수 없기 때문에 이를 효과적으로 해결해야만 한다. 이를 위해 NPE(no prefix expansion)[7]과 같이, non-leaf prefix 대신에 IP 주소를 cache에 적재하는 방법이 있으나, 이는 공간적 지역성을 활용하는데 상당한 제한을 준다. 다른 방법으로, non-leaf prefix 뿐만 아니라 그의 모든 후손 prefix를 같이 적재하면 잘못된 검색결과를 방지할 수 있다. 그러나 이 방법은 cache에서 다중 매치되는 prefix들의 처리를 위한 순위(ordering) 유지 논리와 우선순위 인코더가 추가되어야 하는 부담이 있다.

Prefix 확장은 non-leaf prefix의 길이를 확장하여 후손 prefix가 없도록 만든다. 즉, 확장된 prefix는 leaf prefix와 같이 볼 수 있으므로 잘못된 검색결과를 산출하지 않는다. 기존의 prefix 확장은 정적인 확장 방법과 동적인 확장 방법이 있다.

정적인 확장은 라우팅테이블에서 non-leaf prefix를 사전에 확장하는 방법으로 확장을 통해 라우팅 테이블 내의 모든 prefix가 leaf prefix가 되도록 만드는 것이다. Liu는 non-leaf prefix를 처리하기 위해, CPTE(completely prefix tree expansion), PPTE(partially prefix tree expansion), NPE를 제안했다[7]. CPTE는 non-leaf prefix의 모든 후손 노드의 차수가 0이나 2가 되도록 완전히 확장한다. 반면에, PPTE는 non-leaf prefix의 왼쪽 또는 오른쪽 자식노드 중 하나만 없을 경우 자식노드가 없는 쪽으로 1비트만 부분적으로 확장한다. CPTE는 cache 적중률 측면에서는 바람직한 방법이나, 라우팅 테이블의 prefix 개수를 상당히 증가시키며, 테이블의 갱신이 힘들다는 근본적인 문제를 지니고 있다. Prefix 개수의 증가는 자체 IP 주소 검색의 시간을 증가시켜 효율을 저하시킨다는 것을 의미한다. Kasnavi 등은 prefix 확장의 부담을 줄이기 위해 짧은 길이의 prefix만 확장하는 SPE(short prefix expansion)를 제안하였다[9]. 또한 SPE 개념을 통해 길이가 짧은 prefix는 prefix 확장을 사용하고, 긴 prefix에 대해서는 IP 주소 caching을 사용하는 MPC(multi-zone pipelined cache)를 개발하였다. 라우팅 테이블에서 확장하는 방법은 공통적으로 prefix의 개수가 증가하고 테이블 갱신에 어려움을 지니고 있다.

동적으로 확장하는 방법은 라우팅 테이블에서 확장을 하지 않고 non-leaf prefix를 cache에 적재하기 직전에 확장한다. 따라서 라우팅 테이블에서 prefix 개수가 전혀 증가하지 않는다. RRC-ME(reverse routing cache-minimal expansion)[8]은 non-leaf prefix와 다른 임의의 prefix 간의 경로 상에 확장된 prefix가 존재하지 않도록 하는 길이까지 non-leaf prefix를 확장하여 cache에 적재한다. 이 방식은 prefix 확장을 위한 유용한 정보가 없기 때문에, prefix 확장을 위해 non-leaf prefix와 확장된 prefix 간의 거리만큼 추가적인 메모리 접근이 필요하다는 단점을 지니고 있다. 반면에 bounded prefix 확장 방식과 bitmap 기반의 prefix caching은 저장된 정보를 기반으로 하여 효율적으로 prefix 확장을 할 수 있다.

Bounded prefix 확장 방식[10]은 non-leaf prefix의 가장 긴 직계 후손 prefix (longest immediate descendant prefix)의 길이인 bound 값을 이용하여, prefix를 그의 bound 값까지만 간단히 확장하는 것이다. 이 방식은 직계 후손 prefix의 개수가 많고 이들의 길이가 비슷한 경우 매우 효율적이고 간편하

게 prefix를 확장할 수 있다. Controlled bounded prefix 확장 방식[11]에서는 컨트롤 정보를 이용하여 bound 값의 한계를 보완하는 기법을 제안하고 있으나, bound 값을 한 개만 이용함으로써 효과가 제한적이다. Bitmap 기반의 prefix caching[12]는 non-leaf prefix의 하위 레벨에 해당하는 2의 승수 개의 비트 값을 기준으로 그 non-leaf prefix를 확장한다. 해당 비트가 1이면 32 비트 IP 주소로 확장하고 해당 비트가 0이면 해당 레벨만큼만 추가 확장한다. 따라서 이 bitmap 기반의 prefix caching은 non-leaf prefix의 후손 prefix가 드문 경우에 효율적이다.

III. Previous Prefix Caching Schemes based on Dynamic Prefix Expansion

1. Prefix caching scheme

라우팅 테이블은 현재 수십만 개의 prefix를 가지고 있기 때문에 가변크기 주소 방식을 적용하는 IP 주소 검색은 빠른 속도로 수행되어야 한다. 이때 라우터에 들어오는 IP 주소들간에 지역성이 존재한다면 cache를 이용하여 IP 주소 검색을 신속히 완료할 수 있다. 그림 1에서 목적지 IP 주소는 1차적으로 cache에서 검색되고, cache miss가 발생할 경우 기존 IP 주소 검색 모듈에서 검색을 수행한다.

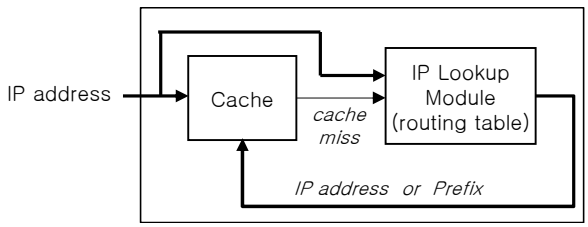

Fig. 1. IP lookup Architecture with Cache

기존 IP 주소 검색 모듈에 있는 라우팅 테이블의 구성은 정적 prefix 확장을 사용하는지 아니면 동적 prefix 확장을 사용하는지에 따라 달라진다. 정적 prefix 확장을 사용하는 경우, 라우팅 테이블은 이미 확장된 prefix들로 구성된다. 이들 prefix들이 검색되었을 때 바로 cache로 적재되거나 또는 경우에 따라 32 비트로 확장되어 적재된다. 동적 prefix 확장의 경우, 라우팅 테이블은 원래 prefix 그대로 구성된다. Prefix가 라우팅 테이블에서 검색되었을 때, 동적으로 길이 확장이 일어나며, 확장된 prefix가 cache에 적재된다.

정적 prefix 확장 기법은 라우팅 테이블에 확장된 prefix가 저장됨으로 인해 저장되는 prefix 개수가 크게 증가하는 문제를 안고 있다. 이와 같은 라우팅 테이블 크기의 증가는 라우팅 테이블 검색 속도를 저하시키는 요인이 된다.

2. Bounded prefix expansion

동적 prefix 확장 기법으로 제안된 bound 방식은 각 prefix마다 확장될 길이를 사전에 계산하여 저장하고 있다. 이러한 확장될 길이 값을 bound라고 부르며, cache에 적재될 prefix는 bound 값이 나타내는 길이만큼 확장되게 된다. 이때 bound 값은 가장 긴 직계 후손 prefix(longest immediate descendant prefix)의 길이로 정의된다.

그림 2에서 prefix p (= 0110*)는 길이가 4인데, 가장 긴 직계 후손 prefix는 q이므로, p의 bound 값은 q의 길이인 9가 된다. 이것은 다음과 같은 것을 의미한다. 어떤 IP 주소가 p에 매치되었을 때, 그 IP 주소를 길이 9로 잘라 prefix를 만들면(즉, p를 길이 9로 확장하면), 이 prefix에 매치되는 어떤 IP 주소도 p의 후손인 q, r, s에는 매치되지 않는다는 것이 보장된다. 그림 2에서 32-비트 주소 ip_1, ip_2는 각각 prefix p에 매치되는데, p를 길이 9까지 확장한 $e1$, $e2$는 p의 후손인 q, r, s 중 어느 누구와도 중첩되지 않는다. 따라서 동적 확장된 prefix $e1$이나 $e2$를 cache에 적재하는 것은 q, r, s에 매칭되는 IP 주소 영역에 아무런 영향을 미치지 않는다.

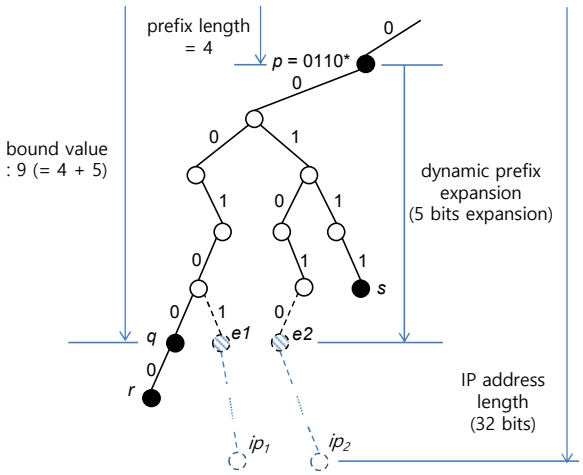

Fig. 2. Dynamic Prefix Expansion using Bound

Cache에 적재되는 prefix는 길이가 짧을수록 더 넓은 영역을 나타내므로 더 높은 cache hit율을 보이게 된다. bound 방식은 non-leaf prefix에 대한 확장 길이를 32 비트가 아닌 bound 값으로 한정함으로써 cache hit율을 높일 수 있다.

그러나, bound 값은 확장된 prefix가 cache에 적재될 수 있음을 보장해주는 길이일 뿐, 이보다 더 짧은 길이의 prefix가 cache에 적재될 수 없는 것은 아니다. 가령, 그림 2에서 ip_2가 라우터에 들어왔을 때, p로부터 동적 확장된 $e2$ = 0110 0101 0* 보다 짧은 0110 010*를 cache에 적재하는 것이 가능하다. prefix 0110 010*는 여전히 prefix s와 영역이 겹치지 않는다.

3. Bitmap based dynamic prefix expansion

Bitmap 기반의 동적 prefix 확장은 8-비트 bitmap 정보에

기반하여 prefix 길이를 3 비트만 확장한다. 라우팅 테이블에 있는 각 prefix는 8-비트 bitmap 정보를 갖고 있다.

그림 3에서 prefix p의 bitmap은 $b_0b_1b_2b_3b_4b_5b_6b_7$ = 01010000 으로 주어져 있다. 각 비트는 해당 위치의 후손으로 prefix가 존재하는지를 나타낸다. 가령, b_0가 나타내는 위치 아래로는 후손 prefix가 없으므로 b_0 = 0이며, b_1의 경우 후손으로 q, r이 있으므로 b_1 = 1이 된다.

그림 3에서 32-비트 IP 주소 ip_1이 주어졌다고 하면, ip_1과 매치되는 prefix p의 bitmap은 $b_0b_1b_2b_3b_4b_5b_6b_7$ = 01010000와 같다. 이때 ip_1은 b_1이 나타내는 위치 아래에 있고, b_1 = 1이므로 후손 prefix가 존재함을 나타낸다. 따라서 p를 3 비트 확장한 prefix(b_1 위치)는 cache에 적재될 수 없다. 이 경우 p는 32 비트로 확장한, 즉, ip_1 자체를 cache에 적재하게 된다. 또 다른 경우로 ip_2가 주어졌을 경우, bitmap에서 해당하는 비트는 b_2 = 0이 된다. 이 경우에는 p를 3 비트 확장한 $e2$를 cache에 적재할 수 있다.

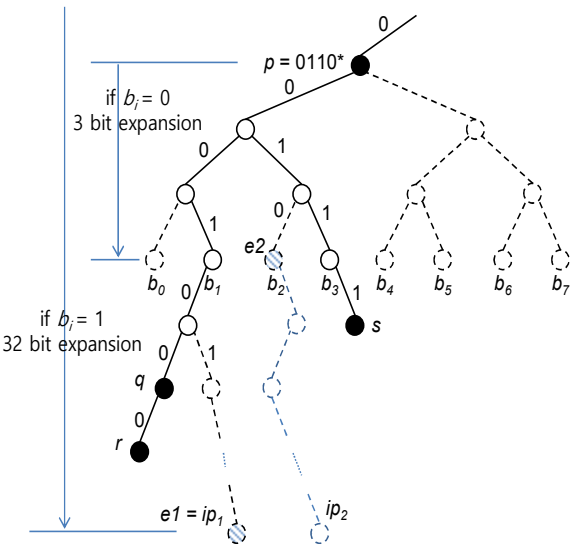

Fig. 3. Dynamic Prefix Expansion in Bitmap

동적 prefix 확장에 bitmap을 이용하는 기법은 b_i = 0인 경우 확장 길이가 3에 불과해 대부분 bound 방식보다 좋지만, b_i = 1이 되면 32 비트로 확장을 하게 되어 prefix caching의 장점을 잃어버리게 된다.

1. Hybrid prefix expansion technique

먼저, 본 논문에서 제안하는 하이브리드 prefix caching을 위한 prefix 확장과 관련된 용어를 설명하고, prefix 확장 방법에 대해 기술한다.

Prefix p의 bitmap은 8-비트 스트링 $bm(p)$ = $b_0b_1\cdots b_7$로 표현하는데, b_i 가 나타내는 위치 아래로 후손 prefix가 존재하면 b_i = 1이고, 그렇지 않으면 b_i = 0이다[12]. 그림 4에서, b_1이 나타내는 위치인 0110001은 후손 prefix q=011000100*를 가지고 있으므로 b_1 = 1이다. 반면에, b_2가 나타내는 위치 아래로 후손 prefix가 없으므로 b_2 = 0이다. 따라서 p=0110*의 bitmap은 bm(p) = 01010000이다.

제안하는 하이브리드 방식은 bitmap의 b_i = 1인 경우에 실질적인 확장을 할 수 없다는 단점을 개선하기 위해 bound 값을 두 개를 사용하고 있다. 이들 bound 값을 다음과 같이 정의한다.

[정의1] Prefix p에 대한 bitmap에서. b_i = 1인 것의 개수를 n이라고 하자. 이들 b_i = 1인 노드들 중에서 앞에서부터 $\lceil n/2 \rceil$ 개 노드의 아래에 있는 가장 긴 직계후손 prefix의 길이를 앞부분 bound인 $fBound(p)$라고 정의하고, b_i = 1인 그 이후 뒷부분 $\lfloor n/2 \rfloor$ 개 노드의 아래에 있는 가장 긴 직계후손 prefix의 길이를 뒷부분 bound인 $rBound(p)$라고 정의한다.

그림 4에서, bm(p=0110*)=01010000이므로, 1의 개수인 n은 2이다. 그리고 bm(p)에서 앞에서부터 첫 번째 1에 해당하는 것이 b_1=1이고 관련된 prefix가 q이므로 fBound(p) = 9이다. 마찬가지로 뒷부분 1에 해당하는 것이 b_3=1이고 관련된 prefix는 s이므로, rBound(p) = 8이다. 추가로 예를 들어, 만일 non-leaf prefix인 t=11*에 대해, bitmap bm(t)=11110001이라고 가정하자. 1의 전체 개수는 5이고, 앞부분 3개의 1에 대한 index가 각각 0, 1, 2이므로 노드 11*000*, 11*001*, 11*010* 의 후손들에 대해서만 bound값을 산출하여 fBound(t)를 구한다. 또한, rBound(t)는 뒷부분 2개의 1에 대한 index가 각각 3, 7이므로 노드 11*011*, 11*111*의 후손들에 대해서만 bound값을 산출하면 된다. 여기서 주목할 점은 bitmap에서 1의 분포를 활용해야만 두 개의 bound값인 fBound와 rBound를 균형 있고 유연하게 사용할 수 있다는 점이고, 제안한 하이브리드 방식의 장점이라고 할 수 있다.

Ⅳ. Proposed Hybrid Prefix Caching

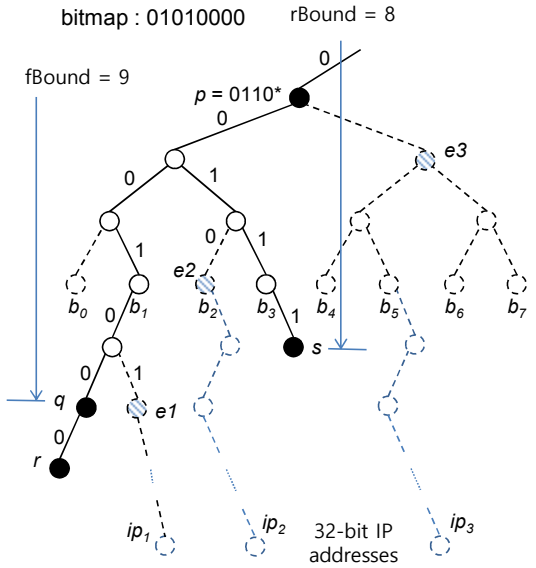

Fig. 4. Hybrid Prefix Expansion Example

하이브리드 prefix 확장은 bitmap 방식과 bound 방식의 장점을 효과적으로 혼합하여 발전시킨 기법이다. Non-leaf prefix p를 캐시에 적재 직전 확장을 하는데, 먼저 해당 IP 주소에서 p의 길이 이후 3비트 값을 바탕으로 하여 p의 bitmap을 살펴본다. 해당 비트가 0이면 최대 3비트까지 덧붙여 확장을 한다. 만일 해당비트가 1이면 bitmap에서 1의 전체 개수와 해당 비트의 위치에 따라 fBound(p) 또는 rBound(p)까지 확장한다.

그림 4에서 하이브리드 방식의 prefix 확장 예를 볼 수 있다. 만일 ip_2의 경우, prefix p가 매치되며, ip_2에서 p의 길이 이후 3비트가 010, 즉 2이다. bitmap은 bm(p) = 01010000이므로 010에 대응되는 값은 b_2 = 0이다. 따라서 3비트를 추가하여 확장한다. 그림 4에서 ip_1에 매치되는 prefix도 p이다. 그러나 ip_1에서 p의 길이 이후 3비트는 001, 즉 1이다. 또한 bitmap에서 b_1 = 1이므로, bitmap에서 전체 1의 개수와 위치를 살펴본다. bm(p) = 01010000에서 1의 개수가 2개이고, b_1은 앞부분에 있는 1이다. 따라서 fBound(p) (=9)를 사용하여 p의 길이를 9로 확장한다.

제안된 기법에서는 bitmap 정보와 두 개의 bound 값, 즉 fBound(p)과 rBound(p)을 사용하여, 가능하면 확장되는 길이를 최대한 줄여 효율을 높이고 있다. 만일 bitmap을 활용하지 않고 단순히 두 개의 bound 값을 사용한다면 임의의 prefix에 대해 왼쪽 subtrie와 오른쪽 subtrie에 대한 bound 값을 사용할 수 밖에 없고, 이 경우 bound 정보를 충분히 활용하지 못해 효율 향상이 제한적이다. 예를 들어, 그림 4와 같이 LMP p의 직계 후손 prefix가 왼쪽 한쪽으로 모여 있는데 bitmap을 활용하지 않는다면, 두 개의 bound 값 모두를 유용하게 사용할 수 있는 방법이 없다. 즉 bitmap을 bound 값과 혼합하여 사용하는 주요 장점은 두 개의 bound 값인 fBound(p)와 rBound(p)의 사용 효과를 높일 수 있다는 점이다.

2. Hybrid prefix caching structure

하이브리드 caching을 사용한 IP 주소 검색기의 구조는 그림 5와 같이 cache, 기존 IP 주소 검색 모듈, 하이브리드 prefix 확장 모듈로 구성되어 있다. Cache에 있는 메모리는 TCAM으로 구성되어 병렬로 신속하게 검색할 수 있다. 하나의 엔트리는 확장된 prefix와 출력 번호 필드로 구성되어 있다. 그리고 다중 매치가 발생하지 않으므로 우선순위 인코더가 필요가 없고 단순하다.

Cache에서 IP주소에 대한 검색이 실패하면, cache miss가 되어 기존 IP 주소 검색 모듈에서 추가로 검색을 한다. 본 하이브리드 방식에서는, 기존의 IP 주소 검색 모듈은 방식이 어떤 것인지가 제한이 되지 않으며, 검색 결과가 사용된다. 검색을 위해 사용되는 라우팅 테이블은 prefix, 출력 번호, bitmap, fBound, rBound 등의 5개 필드로 구성되어 있다.

하이브리드 prefix 확장 모듈은 그림 5에서 보는 바와 같이, 라우팅 테이블에서 매칭된 결과를 활용하여 prefix를 확장하는 기능을 수행하고 그 결과를 cache에 적재한다. 이 모듈은 prefix 길이, IP 주소, bitmap, fBound, rBound를 입력으로 한다. 이들 입력을 이용하여 prefix에서 확장할 길이를 구한다. 그리고 IP 주소에서 확장 길이만큼 masking하여 확장된 prefix를 cache에 적재하게 된다.

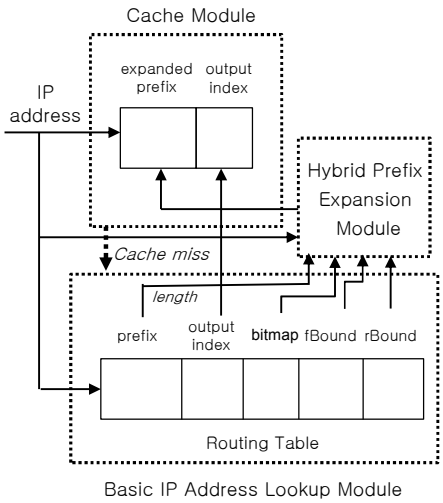

Fig. 5. Hybrid Prefix Caching Structure

3. Determination of prefix expansion length

하이브리드 prefix 확장 기법의 핵심은 prefix를 확장하는 길이를 산출하는 것이다. 그림 6는 prefix의 확장 길이를 구하는 알고리즘을 보이고 있다. 그림 6의 라인 1과 같이, 먼저 IP 주소에서 prefix의 길이만큼 앞부분을 제외하고 다음 3비트 값 x를 구한다. 그 값에 해당하는 bitmap, 즉 b_x가 0이면 최대 3까지 추가 확장하고, b_x가 1이면 *fBound* 또는 *rBound*로 확장한다.

```
Hybrid_Pfx_Expansion(prefix length h, IP_addr,
bitmap b0b1… b7, fBound, rBound)

1. x(=x0x1x2) ← rhrh+1rh+2,
        where IP_addr = r0r1… rh-1rhrh+1…r31
2.  y ← x0x1x2' ; z ← x0x1'x2 ; w ← x0x1'x2'
3.  if (bx = 0) // if the related bit in bitmap is 0
4.      if (by = 1) leng ← 3
5.      else if (bz∨ bw = 1) leng ← 2
6.      else leng ← 1
7.      expansion_leng ← h + leng
8.  else // if the related bit in bitmap is 1 (bx=1)
9.      n ← the number of 1 in bitmap b0b1…b7
10.     if (bx is one of the front ⌈ n/2 ⌉ 1's)
11.         expansion_leng ← fBound
12.     else expansion_leng ← rBound
13. return expansion_leng
```

Fig. 6. Algorithm for the Prefix Expansion

그림 6에서, b_x가 0이면 라인 3~7을 수행하는데, bitmap의 해당 위치인 x를 기준으로 1에서 3까지 추가 확장 길이가 정해진다. 그림 4에서, 입력 IP 주소 ip_2에 대한 예를 들어 보자. 그림 6의 알고리즘에서 보면, bitmap 01010000에서의 위치 x는 2(=010)이다. 따라서 y = 011이고, 이어서 b_2가 0이고 b_3은 1이다. 즉 라인 4에 해당되어, *leng*이 3이 된다. 또 그림 4에서, IP 주소 ip_3에 대해, bitmap의 위치는 5(=101)이다. 그림 6의 알고리즘에서 보면, x = 101이므로, y = 100, z = 111, w = 110이다. 따라서 라인 6에 해당되어, *leng*이 1이 된다. 결과적으로, ip_3에 대한 확장 prefix는 $e3$ = 01101*가 되며, p = 0110*에서 1비트 확장한 것이다.

그림 6의 라인 8~12는 b_x가 1인 경우로 긴 확장이 필요하다. 먼저 bitmap에서 전체 1의 개수를 n으로 찾고, x 에 해당하는 비트 1이 앞부분에 있으면 *fBound* 값을, 뒷부분에 있으면 *rBound* 값을 확장 길이 *expansion_leng*에 저장한다. 그림 4에서, IP 주소 ip_1에 대해, bitmap의 위치는 1이고, b_1 = 1이다. 또한 bitmap이 01010000이므로, 1의 전체 개수 n = 2이고, b_1이 앞부분에 있으므로, 확장 길이 *expansion_leng*은 *fBound* 값인 9가 된다.

V. Simulation Results

그림 7은 제안한 하이브리드 prefix 확장 기법이 원래의 prefix를 어느 정도 확장하는지 보여준다. 그림에서 'Bound', 'Bitmap', 'Hybrid'는 각각 bounded prefix 확장, bitmap 기반 prefix 확장, 본 논문에서 제안하는 하이브리드 prefix 확장을 나타낸다. 라우팅 테이블은 APNIC[13]의 실제 테이블을 사용하였다. 그림에서 x축은 원래 prefix 길이를 나타내고, y축은 주어진 prefix에 대해 IP 주소가 균등하게 분포되어 들어왔을 때, 원래 길이에 대해 추가적으로 연장된 평균 길이를 나타낸다. 하이브리드 prefix 확장은 다른 두 개의 기법보다 0.1 ~ 6 정도 길이가 더 짧게 확장함을 알 수 있다.

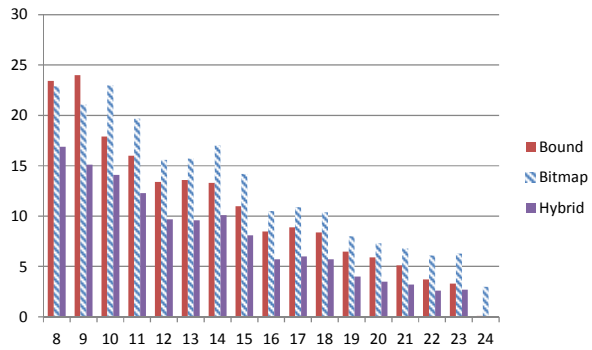

Fig. 7. Comparison of Expansion Length

Cache miss율 비교를 위해 마찬가지로 APNIC의 실제 라우팅 테이블을 사용하였으며, 트래픽 데이터의 경우는 공개적으로 접근 가능한 트래픽은 실제 IP 주소를 알 수 없게 되어 있으므로, 인터넷 트래픽의 시간적 공간적 지역성을 유지하고 라우팅 테이블에 적용 가능하도록 랜덤 생성한 트래픽 데이터 2개 종류를 사용하였다. CPTE와 PPTE의 경우는 해당 기법에 맞게 prefix를 확장하고, 확장된 prefix를 라우팅 테이블에 추가하여 재 생성하였다.

비교 대상이 되는 기법들은 bounded prefix 확장, controlled bounded prefix 확장, bitmap 기반 prefix caching 방식, CPTE, PPTE, NPE를 대상으로 하였고, 실험 결과에서는 각각, 'Bound', 'CTR', 'Bitmap', 'CPTE', 'PPTE', 'NPE'로 방식을 표시하였다.

그림 8과 9는 이들 트래픽 데이터에 대한 시뮬레이션 결과를 보이고 있다. 이들 그림에서, 'Hybrid'는 본 논문에서 제안한 하이브리드 방식을 나타낸다. Bitmap 방식은 8-비트 bitmap을 사용하여 시뮬레이션 하였다.

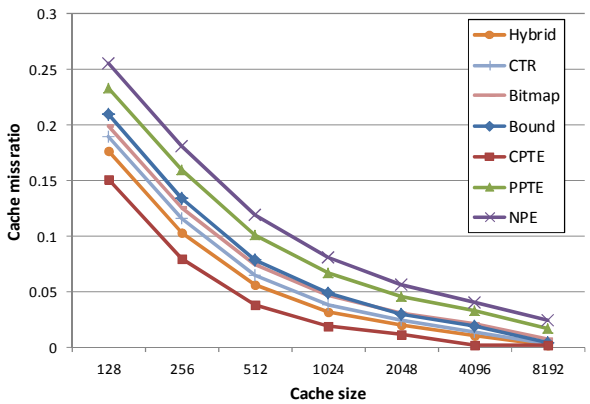

Fig. 8. Cache Miss Ratio for Traffic Data 1

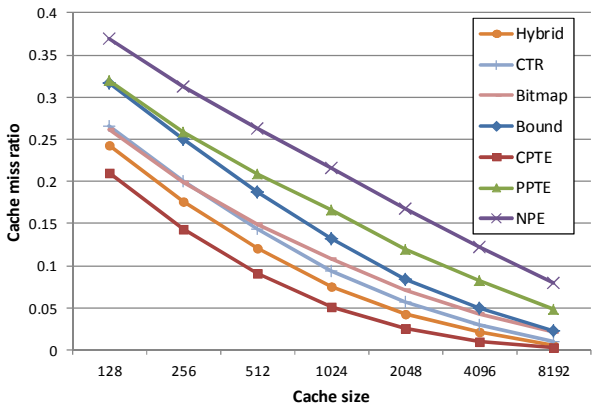

Fig. 9. Cache Miss Ratio for Traffic Data 2

시뮬레이션에서, cache 크기를 128부터 8192까지 다양하게 바꾸어 가면서 실험을 하였다. 실험 결과는 그림 8과 9에서 보는 바와 같이, 본 논문에서 제안한 하이브리드 방식이 CPTE를 제외한 모든 다른 방식에 비해, cache 크기에 상관없이 더 낮은 cache miss율을 보였다. CPTE는 모든 prefix에 대해 leaf가 되도록 확장하기 때문에 낮은 cache miss율을 보이게 된다. 그런데 본 실험에서 하이브리드 prefix 확장 기법의 라우팅 테이블은 552,981개의 prefix를 포함하며 원래 APNIC 라우팅 테이블과 동일하다. 반면에 CPTE의 라우팅 테이블 크기는 729,949로 크게 증가하여, 제안된 기법보다 무려 17만 6천개 이상의 prefix가 추가로 늘어난다. 즉 CPTE는 cache miss율이 우수하지만, 라우팅 테이블이 이와 같이 매우 커지기 때문에 IP 주소검색 자체의 성능과 라우팅 테이블 갱신에 상당한 어려움이 발생한다. 그림들에서, 하이브리드 방식이 bound 방식과 bitmap 방식에 비해 대부분의 경우 성능 차이를 많이 보이고 있다. 그림 9를 보면, 작은 cache에서 bitmap 방식에 대한 성능의 향상도가 크지 않았으나, cahce 크기가 증가함에 따라 성능 차이는 점차 더욱 벌어짐을 확인하였다.

Ⅵ. Conclusion

본 논문에서는 IP 주소 검색을 신속하게 수행하기 위한 prefix cache로서, 하이브리드 기법을 연구하였다. Prefix caching에서 non-leaf prefix 문제 해결을 위한 동적 prefix 확장 방법들이 이미 제안되었는데, 본 논문에서는 non-leaf prefix의 후손 prefix가 많은 경우 유리한 bounded prefix 확장 방식과 적은 경우 유리한 bitmap 기반 prefix caching의 장점을 취한 혼합형 prefix caching 방법을 제안하였다. 제안한 기법은 bitmap의 해당 위치 값에 따라 그 값이 0이면 3이하의 짧은 확장을 하고, 1이면 bound 값의 개념을 이용하여 미리 정해진 적절한 값까지 확장을 한다. 기존의 bitmap 방식은 값이 1인 경우 IP 주소 자체를 cache에 적재하기 때문에, 공간적 지역성 활용에 제한이 있었다.

본 연구에서는 동적 prefix 확장 시 길이를 더욱 단축하기 위해 bound 값을 두 개 사용한다. 이때 bitmap 정보가 없다면, bound 값들은 단순히 왼쪽과 오른쪽 subtrie에 고정적으로 적용될 수 밖에 없어 효과가 제한적이다. 본 하이브리드 방식에서는 bitmap에서 1의 분포를 활용하여 앞부분을 담당하는 bound 값과 뒷부분을 담당하는 bound 값을 제안하여, prefix 확장 시 유연하고 효과적으로 사용하였다. 즉, bitmap에서 1의 개수와 위치 등의 정보를 최대한 활용함으로써, 이들 bound 값들을 유연하게 할당하여 이용하고 있다.

본 논문에서 제안된 기법의 성능을 다른 기법들과 비교 평가하기 위해, 실제 라우팅 테이블과 IP 트래픽의 지역성을 반영하는 트래픽을 생성하여 사용하여 시뮬레이션 하였다. 시뮬레이션 결과, cache 크기에 관계없이, 제안된 기법이 bounded 확장방식과 bitmap 방식을 포함한 대부분의 다른 기법들보다 낮은 cache miss율을 보이고 있다. Cache miss율이 낮다는 것은 cache를 더 많이 사용하여 IP 주소 검색 속도가 증가한다는 것을 의미한다.

REFERENCE

[1] G. Varghese, “Network Algorithmics: An Interdisciplinary Approach to Designing Fast Networked Devices,” Morgan Kaufmann, 2005.

[2] Mi. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," IEEE Network, March/April 2001.

[3] J. Kim and J. Kim, “Fast Prefix Deletion for Parallel TCAM-Based IP Address Lookup,” Journal of The Korea Society of Computer and Information, Vol. 15, No. 12, pp. 93-100, Dec. 2010.

[4] L. Luo, G. Xie, Y. Xie, L. Mathy, and K. Salamatian, “A Hybrid Hardware Architecture for High-Speed IP Lookups and Fast Route Updates,” Proceedings of IEEE INFOCOM, pp. 2435-2443, 2012.

[5] W. Shi, M. MacGregor, P. Gburzynski, “On temporal locality in IP address sequences,” IEICE Transactions on Communications, E86-B (11), pp. 3352-3354, 2003.

[6] B. Talbot, T. Sherwood, and B. Lin, “IP caching for terabit speed routers,” Proc. of Global Telecommunications Conf., pp. 1565-1569, 1999.

[7] H. Liu, “Routing prefix caching in network processor design,” Proc. of International Conf. on Computer Communications and Networks, pp. 18-23, Oct. 2001.

[8] M. J. Akhbarizadeh and M. Nourani, “Efficient prefix cache for network processors,” Proc. of 12th Annual IEEE Symp. on High Performance

Interconnects, pp. 41-46, Aug. 2004.
[9] S. Kasnavi, P. Berube, V. Gaudet, and J. N. Amaral, “A cache-based internet protocol address lookup architecture,” Computer Networks, vol. 52, no. 2, pp. 303-326, Feb. 2008.
[10] J. Kim, M. Park, S. Han, and J. Kim, “An efficient prefix caching scheme with bounded prefix expansion for high-speed IP lookup,” IEICE Transactions on Communications, vol. E95-B, no. 10, pp.3298-3301, Oct. 2012.
[11] J Kim, M Park, S Han, and J Kim, "A Novel Controlled Bound Prefix Expansion for Prefix Caching in High-Speed IP Lookup," Journal of Next Generation Information Technology, Vol. 4, No. 7, pp. 61-69, Sep. 2013.
[12] J. Kim, M.-C. Ko, J. Nam, and J. Kim, “Bitmap-based Prefix Caching for Fast IP Lookup,” KSII Tr. on Internet and Information Systems, vol. 8, no. 3, pp.873-889, Mar. 2014.
[13] APNIC prefix table, http://thyme.apnic.net/ap-data/2015/01/01

Authors

Jinsoo Kim received the B.S. degree from Seoul National University, Seoul, in 1983, and the M.S. and Ph.D degrees from Korea Advanced Institute of Science and Technology (KAIST), in 1985 and 1998, respectively, all in computer engineering. Dr. Kim joined Korea Telecom, where he was a senior researcher in 1985. He has been a professor of Konkuk University since 2000. His research interests include parallel computing architectures, high-speed networking, wireless sensor networks, and packet processing systems.

Junghwan Kim received the B.S., M.S. and Ph.D degrees from Seoul National University, Seoul, in 1991, 1993 and 1999, respectively, all in computer science.
Dr. Kim joined Samsung Electronics as a senior researcher in 1999. He joined the faculty of Konkuk University in 2001. His research interests are in the areas of parallel computing, communication networking, GPU computing, and design of efficient algorithms.