# IP 주소 검색의 속도 향상을 위한 분할된 압축 트라이 구조

박 재 형[†]·장 익 현[††]·정 민 영[†††]·원 용 관[††††]


## 요 약

인터넷에서 IP 패킷 전송은 링크의 전송 속도와 더불어 라우터에서의 패킷 처리 속도에 영향을 받는다. 라우터는 외부 인터페이스에서 입력되는 패킷을 목적지로 보내기 위한 다음 홉을 결정하여 패킷을 전달하는 역할을 수행한다. 이 과정에서 주소 검색은 고성능의 라우터 설계에 중요한 요인이다. 본 논문에서는 트라이 자료 구조 기반의 IP 주소 검색 알고리즘의 성능을 향상시키기 위해서 경로 압축을 이용한 분할 압축 트라이 구조를 제안한다. 제안된 분할 압축 트라이에서는 IP 주소 프리픽스들을 여러 개의 분할 압축 트라이로 나누어서 하나의 분할된 압축 트라이에서만 검색이 이루어지도록 하여 압축 트라이에서 탐색하는데 드는 시간을 줄이는 방법이다. 분할을 함으로써 늘어나게 되는 메모리의 부담이 적음을 보여준다.


# A Partitioned Compressed-Trie for Speeding up IP Address Lookups

Jaehyung Park[†]·Ikhyeon Jang[††]·Min Young Chung[†††]·Yonggwan Won[††††]


## ABSTRACT

Packet processing speed of routers as well as transmission speed of physical links gives a great effect on IP packet transfer rate in Internet. The router forwards a packet after determining the next hop to the packet's destination. IP address lookup is a main design issue for high performance routers. In this paper, we propose a partitioned compressed-trie for speeding-up IP address lookup algorithms based on trie data structure by exploiting path compression. In the proposed scheme, IP prefixes are divided into several compressed-tries and lookup is performed on only one partitioned compressed-trie. Memory access time for IP address lookup is lessen due to compression technique and memory required for maintaining partition does not increased.


**키워드 : 인터넷 라우터(Internet Router), IP 주소 검색(IP Lookup), 프리픽스 트라이(Prefix Trie), IP 주소 체계(IP Addressing System), 압축 트라이(Compressed Trie)**

## 1. 서 론

인터넷을 통한 서비스 및 응용 프로그램이 다양해짐에 따라서 인터넷 사용자 수가 증가하여 트래픽의 양이 급격하게 증가하고 있다[7]. 이러한 인터넷 상황에서 패킷 전송율을 높임으로써 인터넷 상에서 지원되는 서비스의 품질을 적정 수준으로 유지할 수 있다. 패킷 전송율을 높이기 위해서는 물리적인 전송 매체의 속도 향상과 더불어 라우터의 패킷 처리 능력의 고속화가 요구된다. 전달 매체의 발달로 트래픽을 전달하는 링크는 높은 대역폭과 고속의 전송이 가능하게 됨에 따라서, 패킷을 처리하는 라우터의 성능이 인터넷 망의 패킷 전송율 증가에 병목 요인이 되고 있다[6].

인터넷 망의 성능에 큰 영향을 미치는 라우터는 입력 인터페이스를 통해서 들어오는 패킷의 목적지를 참조하여 IP 패킷이 목적지로 전달되기 위한 출력 인터페이스를 결정한 후 패킷을 전달한다. 이러한 과정 중에 패킷 목적지 주소와 출력 인터페이스가 사상된 포워딩 테이블을 검색하는 작업은 CIDR(Classless Inter-Domain Routing)[3]이 등장하면서 입력된 IP 패킷의 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 최장 프리픽스 일치(longest prefix matching)를 찾는 작업이 되었다. 최장 프리픽스 일치를 찾는 과정은 IP 주소의 처음 비트부터 순차적으로 비교하여 가장 긴 주소가 일치하는 프리픽스를 찾아야 한다. 따라서 하나의 주소를 검색하는데 여러 번의 메모리 접근을 필요로 하므로 고속의 IP 패킷 전달을 지원하는 라우터에서는 중요한 병목 요인이 되고 있다[1].

고속의 라우터를 구현하기 위해서는 최장 프리픽스 일치를 보다 빠르게 찾는 기법이 필요하다. 고속의 IP 주소 검색을 지원하기 위해서 크게 두 가지 방향으로 연구가 진행되었다. 하나의 방향은 메모리 접근 속도가 빠른 SRAM이나 CAM을 이용하는 하드웨어 기반 방법이다[4, 8, 16]. 하드웨어 기반 방법은 고속의 IP 주소 검색을 지원하지만, 고가의 메모리를 사용하므로 많은 비용이 든다는 단점이 있다. 다른 하나의 연구 방향은 트라이 자료 구조에 기반을 둔 소프트웨어 방법이다. 소프트웨어 기반 방법은 현존하는 라우터에서 IP 주소 검색을 위해서 사용되고 있는 방법이다[10, 14].

소프트웨어 방법으로 트라이 자료 구조에 기반을 둔 최장 프리픽스를 찾기 위한 여러 가지 알고리즘들이 연구되고 있다. 그러한 알고리즘 중에 Patricia Trie[11]는 기수 트라이(radix trie)의 특별한 형태로서 현존하는 라우터에 구

※ 이 논문은 2002년도 전남대학교 학술연구비 지원에 의하여 연구되었음.
† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
†† 종신회원 : 동국대학교 정보통신공학과 교수
††† 종신회원 : 성균관대학교 정보통신공학부 교수
†††† 종신회원 : 전남대학교 정보통신공학부/의학부 교수
논문접수 : 2003년 7월 24일, 심사완료 : 2003년 9월 15일

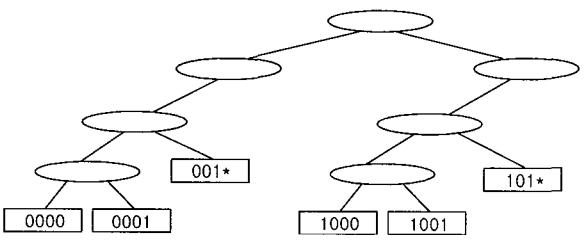
현되었다[14]. Patricia Trie는 하나의 자식 노드만 갖는 노드가 없는 검색경로 압축 트라이로서, 검색 시에 메모리 접근 횟수를 줄였다. 그러나, $W$가 트라이의 최대 높이라고 하면, 재귀적인 후진탐색이 필요하여 $O(W^2)$번의 메모리 접근이 요구된다. 이와 같은 재귀적인 후진탐색을 피하기 위해서 동적 프리픽스 트라이가 제시되었으며[2], 이 기법의 메모리 접근 횟수는 $O(W)$이다. 트라이 기반의 IP 주소 검색을 빠르게 수행하기 위해서 트라이 자료 구조의 변형들이 계속 연구 중이다[12, 13, 15].

본 논문에서는 고속의 패킷 처리를 위해서 트라이 자료 구조에 기반을 둔 IP 주소 검색 기법의 성능 향상을 지원하는 구조로 분할된 경로 압축 트라이 알고리즘을 제시한다. 제안된 분할 압축 트라이에서는 IP 주소 프리픽스들을 여러 개의 분할 압축 트라이로 나누어서, IP 주소 검색 시에 하나의 분할된 압축 트라이에서만 검색이 이루어지도록 하여 IP 주소를 검색하는데 드는 시간을 줄이고자 한다. 트라이 기반의 구조에서 IP 주소 검색에 드는 시간에 큰 영향을 미치는 메모리 접근 횟수가 감소함을 보이고, IP 주소 테이블을 트라이 데이터 구조로 표현하는데 추가로 요구되는 메모리량이 없음을 보인다.

본 논문의 구성은 다음과 같다. 2절에서는 트라이를 기반으로 제안된 알고리즘에 대해서 기술한다. 3절에서는 트라이 기반의 IP 검색 알고리즘의 성능 향상을 위한 분할 압축 트라이 구조와 포워딩 테이블의 IP 프리픽스를 분할하는 기법을 제안한다. 4절에서는 메모리 접근 시간과 메모리 요구량을 기준으로 제안된 분할 압축 트라이의 성능을 수치적인 방법으로 평가하고, 마지막으로 5절에서 결론을 맺는다.

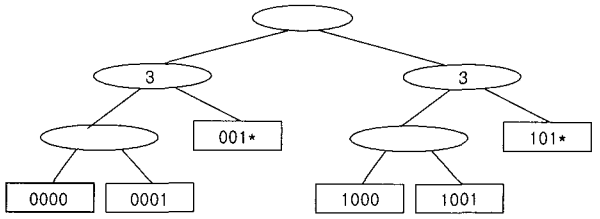
## 2. 트라이 기반 IP 검색 알고리즘

트라이는 트리와 유사한 자료 구조이며[5], 하나 이상의 자식 노드를 갖는 노드로 구성된다. 특정 키에 대한 검색은 주어진 키와 최장의 프리픽스가 일치하는 것을 찾기 위해서 트라이의 루트 노드에서 시작하여 트라이를 내려가면서 탐색한다. 각 노드에서는 특정 키의 일부분을 이용하여 다음에 탐색하여야 할 노드를 결정한다. (그림 1)은 0000, 0001, 001*, 1000, 1001, 101*의 6개의 프리픽스로 구성된 이진 트라이(binary trie)의 예를 보여준다. (그림 1)에서 프리픽스 1001을 찾는 과정은 루트노드로부터 시작하여 2개의 가지 중에서 1, 0, 0, 1 순으로 탐색한다. 그러므로, 트라이의 탐색 횟수는 프리픽스 길이에 비례하여, 프리픽스의 최대 길이가 $L$이라면 탐색 알고리즘은 $O(L)$의 성능을 갖는다.

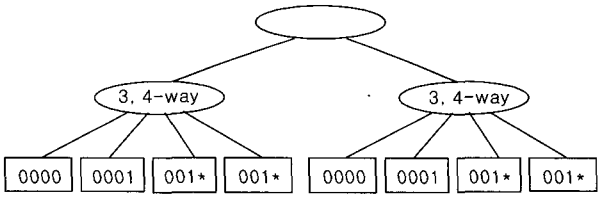



(그림 1) 이진 트라이

검색경로 압축 트라이(Path Compressed-Trie, PC-trie)[2]는 트라이를 기반으로 구성되며, (그림 1)의 예에 대한 검색경로 압축 트라이는 (그림 2)와 같이 표현된다. 하나의 자식 노드를 갖는 노드를 제거하기 위해서 각각의 가지 노드에 비트 번호가 부가된다. 부가된 비트 번호는 가지 노드에서 검색 시에 검사해야할 비트 번호를 의미한다. 검색경로 압축 트라이에서는 검사하지 않아도 되는 비트들을 제거함으로써 원래의 이진 트라이의 높이보다 더 낮은 트라이의 높이를 유지할 수 있다. 구성된 트라이의 높이를 줄임으로써 특정 키에 대한 검색 시에 메모리 접근 횟수를 줄이는 효과를 나타낸다. Patricia Trie[11]는 경로 압축 트라이의 전형적인 예이다. Patricia 트라이는 경로 압축 방법을 통해서 트라이의 높이 ($W$)를 줄였다. 즉, 이진 트라이의 높이를 $L$이라고 하면, $W \leq L$이다. 그러나, 프리픽스에 대한 탐색이 실패하였을 경우 재귀적인 후진탐색이 필요하여 최악의 경우 $O(W^2)$의 탐색이 요구된다. 동적 프리픽스 트라이[2]는 Patricia 트라이의 재귀적인 후진탐색의 단점을 극복하여 $O(W)$만에 탐색을 할 수 있는 구조이다.




(그림 2) 경로 압축 트라이




(그림 3) 단계 압축 트라이

(그림 3)은 단계 압축 트라이(Level Compressed-Trie, LC-trie)[12]의 예를 보여준다. 단계 압축 트라이에서는 PC-trie와 달리 노드에서의 n-way 가지를 허용한다. 그러한 특성 때문에, 노드가 몇 개의 가지를 갖는지를 표현하는 n값을 노드가 포함하여야 한다. LC-trie는 단계를 압축함으로써 트라이의 높이를 줄여 메모리 접근 횟수를 감소시킬 수 있다. 그러므로, 트라이의 높이를 $W$라면, 프리픽스를 탐색하기 위해서는 $O(W)$만큼의 시간이 소용된다. 한편, LC-trie는 노드마다 가지의 개수를 표현하는 n값이 상이하기 때문에 구현이 어렵다.

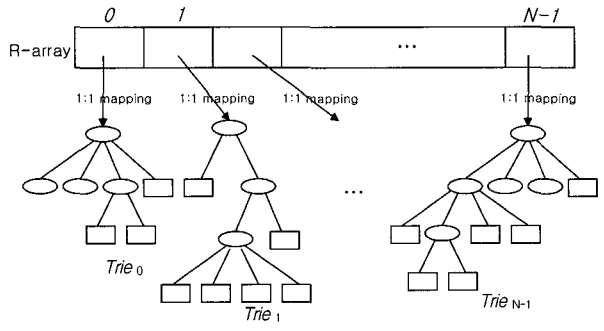
## 3. 분할된 경로 압축 트라이

본 절에서는 트라이 기반의 IP 주소 검색 알고리즘의 성

능을 높일 수 있는 분할된 경로 압축 트라이를 제안하고, 제안된 구조에서 IP 주소를 검색하는 과정을 기술한다. 또한, 포워딩 테이블에 존재하는 IP 주소를 분할하는 기법을 제안하고, 제안된 구조에서 포워딩 테이블에 대한 추가/삭제 연산 과정을 기술한다.

### 3.1 분할된 압축 트라이의 구조 및 검색

분할된 경로 압축 트라이는 $N$개의 압축 트라이와 $N$개의 트라이 노드를 저장할 수 있는 배열 R-array로 이루어진다. (그림 4)는 분할된 압축 트라이의 전체적인 구조를 보여주며, 각각 해당하는 IP 프리픽스들로 형성된 $N$개의 압축 트라이 ($Trie_i$)와 해당되는 트라이의 루트 노드를 저장하는 배열 R-array[ $i$ ]로 구성된다, $0 \le i \le N-1$.




(그림 4) 분할된 압축 트라이의 구조

분할된 압축 트라이에서 입력된 IP 패킷의 목적지 주소를 참조하여 패킷을 출력 인터페이스로 전달하기 위해서는 IP 주소를 검색하는 과정이 수행되어야 한다. 분할된 압축 트라이에서 IP 주소 검색은 찾고자 하는 IP 주소의 프리픽스가 $N$개의 압축 트라이 중에 어떤 트라이에 속해 있는지를 먼저 결정한 다음, 특정 압축 트라이에서 PC-trie 또는 LC-trie와 동일한 방법으로 트라이를 탐색하면서 최장 프리픽스 일치를 찾는 작업을 수행한다. 그러므로, 특정 압축 트라이의 루트 노드가 저장되어 있는 배열 R-array의 인덱스인 $i$를 구하는 계산이 선행되어야 한다.

배열 R-array의 인덱스 값 $i$를 구하는 계산은 포워딩 테이블에 존재하는 IP 프리픽스를 분할하는 기법에 의해서 결정된다. IP 프리픽스를 분할하는 기법을 제안하기에 앞서서 몇 가지 필요한 용어를 정의한다.

우선, IP 프리픽스 $p$는 이진수로 $p_1 p_2 \cdots p_m$와 같이 표현되고, IPv4 환경에서는 $m \le 32$이다. 1부터 $m$사이의 임의의 $k$개의 숫자를 $j_1, j_2, \cdots, j_k$와 같이 표현된다고 가정하자. 일반성을 잃지 않고, 임의의 $k$개의 숫자는 $j_1 < j_2 < \cdots < j_k$이라고 하자.

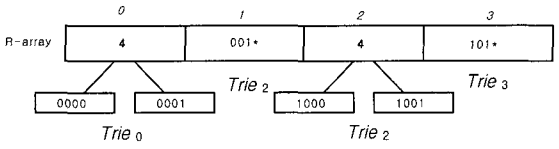
**[정의 1]** $p_k$는 이진수로 표현된 IP 프리픽스 $p_1 p_2 \cdots p_m$의 임의의 일부분으로 $p_{j_1} p_{j_2} \cdots p_{j_k}$과 같은 이진수로 정의한다. 이 때, $k$는 $m$보다 작으며, $1 \le l \le k$, 모든 $l$에 대해서 $1 \le j_l \le m$이다.

**[정의 2]** $P_k$는 포워딩 테이블에 존재하는 모든 프리픽스 $p$에 대해서 같은 $p_k$값을 갖는 프리픽스들의 집합으로 정의한다.

### 3.2 임의의 $k$비트를 이용한 분할 기법

임의의 $k$개의 비트를 이용한 분할 기법(including class-bits 기법)에서는 포워딩 테이블에 존재하는 모든 프리픽스를 특정한 비트 위치의 $k$개의 비트 값이 같은 프리픽스들로 분할한다. [정의 1]과 [정의 2]를 이용하여, 압축 트라이 $Trie_i$는 포워딩 테이블에 존재하는 모든 프리픽스에 대해서 $p_k$값이 $i$인 프리픽스 집합인 $P_i$들로 구성된 압축 트라이다. 이 때, 압축 트라이의 개수 $N$은 $2^k$이다.

(그림 5)는 (그림 2)의 예에서 프리픽스의 첫 번째와 세 번째 비트들을 이용하여 분할한 기법으로 $N(4=2^2)$개의 압축 트라이로 구성된다. PC-trie와 LC-trie에서와 같이 노드에서 비교를 해야하는 비트 번호( $Trie_0$의 루트 노드인 R-array[0]에 포함된 4)를 포함하고 있다. (그림 5)에서 보는 바와 같이, 각각 분할된 프리픽스들로 경로 압축 트라이를 구성하면, 임의의 $Trie_i$, $0 \le i \le N-1$의 높이는 $k$만큼 감소한다. 그 이유는 각각 분할된 프리픽스는 특정한 비트 위치의 $k$개의 비트가 같기 때문이다.




(그림 5) 분할 압축 트라이의 예

임의의 $k$개의 비트를 이용한 분할 기법에서는 다음과 같은 두 가지의 경우가 발생한다. 첫 번째는 IP 프리픽스의 길이, $length(p)$가 $j_l, 1 \le l \le k$보다 크거나 같은 경우이고, 두 번째는 IP 프리픽스의 길이, $length(p)$가 $j_l$보다 작은 경우이다. 첫 번째의 경우에는 주어진 IP 프리픽스에 대한 분할 기법을 그대로 적용할 수 있으나, 두 번째의 경우에는 주어진 IP 프리픽스를 $length(p)$보다 큰 모든 $j_l, 1 \le l \le k$에 대해서 해당되는 비트 위치의 값을 0인 프리픽스와 1인 프리픽스를 갖는 두 개의 프리픽스로 확장을 하여야 한다. 즉, 포워딩 테이블에 추가와 삭제 연산에는 두 가지 경우에 따라서 첫 번째 경우의 프리픽스는 추가 및 삭제 연산이 $p_k$ 값이 $i$인 압축 트라이 $Trie_i$에서 수행된다. 그러나, 두 번째 경우의 프리픽스는 프리픽스를 확장한 후, 확장된 개수만큼의 압축 트라이 $Trie_i$에서 각각 수행되어야 한다.

### 3.3 IP 주소 체계를 고려한 분할 기법

IP 주소 체계를 고려한 분할 기법(excluding class-bits 기법)에서는 IP 주소는 네트워크 번호와 호스트 번호로 나뉘어지는데, IP 주소의 최고 높은 비트들 값에 따라서 해당 IP 주소가 어느 클래스에 속하는지 결정되기 때문에 클래스를 의미하는 비트들을 분할하는데 사용하지 않는 기법이다.

IP 주소 체계를 고려한 분할 기법에서는 IP 주소 클래스별로 클래스 A에는 $k_a$개, 클래스 B에는 $k_b$개, 클래스 C에는 $k_c$개의 임의의 비트를 이용하여 분할한다. 클래스 A에 선정된 $k_a$개에는 IP 프리픽스의 첫 번째 비트는 제외되고, 클래스 B에 선정된 $k_b$개에는 첫 번째와 두 번째 비트들이 제외되고, 클래스 C에 선정된 $k_c$개에는 첫 번째, 두 번째, 세 번째 비트가 제외된다. 즉, 클래스 별로 선정된 $k$값을 이용하여 임의의 $k$개의 비트를 이용한 분할 기법을 적용한다. 그러면, 각각의 분할된 압축 트라이의 개수는 클래스 A에는 $2^{k_a}$개, 클래스 B에는 $2^{k_b}$개, 클래스 C에는 $2^{k_c}$개다.

이 기법에서의 R-array의 인덱스 $i$는 식 (1)과 같이 계산된다.

$$i = \begin{cases} i_a, & \text{if prefix} \in \text{Class A}, \\ i_b + 2^{k_a}, & \text{if prefix} \in \text{Class B}, \\ i_c + 2^{k_a} + 2^{k_b}, & \text{if prefix} \in \text{Class C}. \end{cases} \quad (1)$$

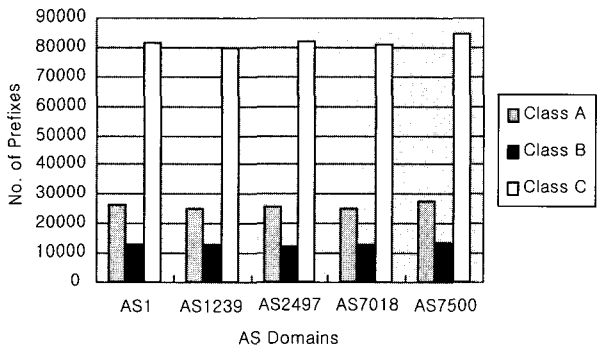
식 (1)에서, 숫자 $i_a$, $i_b$, $i_c$는 각각의 클래스 내에서 주어진 프리픽스에 대한 $p_{k_a}$값, $p_{k_b}$값, $p_{k_c}$값이다.

## 4. 성능 평가

본 절에서는 제안한 분할된 압축 트라이 구조의 성능을 수치적인 방법으로 평가한다. 메모리 접근 횟수와 포워딩 테이블을 제안된 트라이 구조로 표현하였을 때 요구되는 메모리 크기를 기준으로 제안된 구조의 성능을 평가한다. 우선, CIDR 환경에서의 라우터의 포워딩 테이블에 포함된 IP 주소 프리픽스의 특성을 알아본다.

### 4.1 IP 주소 프리픽스의 특성

(그림 6)은 5개의 AS 도메인 - ASN-BBN(AS1), ASN-ICM-INRIA (AS1239), ASN-JNIC-ASN-ASNBLOCK (AS2497), ASN-ATT-INTERNET4 (AS7018), DNS-IN (AS7500) - 의 BGP 라우팅 테이블에서 추출한 클래스에 따른 프리픽스의 분포이다. 약 70%의 IP 주소 프리픽스가 클래스 C에 포함되고, 클래스 A와 B에는 약 20%와 10%씩 정도만이 포함되어 있음을 알 수 있다. 즉, IP 주소 프리픽스가 클래스별로 균일하게 분포되어있지 않고 특정 클래스에 집중되어 있음을 알 수 있다.




(그림 6) IP 주소 프리픽스 분포

<표 1>은 5개의 AS 도메인에서 추출된 클래스별 프리픽스 분포를 프리픽스 길이로 나누어 보여준다. <표 1>에서 보여진 바와 같이 IP 주소 프리픽스의 특성은 프리픽스의 길이가 7이하인 IP 주소 프리픽스는 존재하지 않으며, 특히 클래스 C인 경우 프리픽스 길이가 10이하인 IP 주소 프리픽스는 전체 중에서 거의 미미한 개수만 존재한다. 대부분의 프리픽스가 16에서 24 길이이며, 그러한 IP 주소 프리픽스가 전체 중 99%를 차지한다.

〈표 1〉 IP 주소 클래스 별 프리픽스 길이 분포

| IP 주소 프리픽스 길이 | AS1 | | | AS1239 | | | AS2497 | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| 1~7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8~10 | 21 | 3 | 3 | 22 | 4 | 3 | 23 | 4 | 3 |
| 11~15 | 272 | 243 | 358 | 263 | 244 | 350 | 275 | 245 | 371 |
| 16~23 | 15040 | 8380 | 30402 | 14242 | 8274 | 29769 | 14454 | 7912 | 30658 |
| 24 | 11195 | 4535 | 50910 | 11037 | 4431 | 49575 | 11193 | 3952 | 51356 |
| 합 계 | 26528 | 13161 | 81673 | 25564 | 12953 | 79697 | 25945 | 12513 | 82388 |

### 4.2 메모리 접근 횟수

제안된 분할 압축 트라이 구조에서의 메모리 접근 횟수는 분할된 압축 트라이의 높이에 비례한다. 임의의 $k$개의 비트를 이용한 분할 기법으로 분할된 압축 트라이의 높이 ( $W_{incb}$)는 평균적으로 식 (2)와 같이 유도된다.

$$W_{incb} = \frac{1}{N}\sum_{i=0}^{N-1} W_i \le \frac{1}{N}\sum_{i=0}^{N-1}(W-k) = W-k. \quad (2)$$

식 (2)에서, $W_i$는 분할된 압축 트라이 $i$의 높이이고, $W$는 포워딩 테이블을 하나의 압축 트라이로 구성하였을 때의 높이이다. 분할된 압축 트라이는 하나의 경로 압축 트라이에서 $k$비트를 압축한 트라이이므로 제안된 분할 압축 트라이 $Trie_i$의 높이는 $(W-k)$보다 적거나 같다. 그러므로, 한번의 IP 주소 검색을 위해서 접근하는 메모리 접근 횟수가 $k$만큼 감소함을 알 수 있다.

IP 주소 체계를 고려하여 프리픽스를 분할하는 기법으로 제안된 분할 압축 트라이의 높이의 평균( $W_{excb}$)은 식 (2)를 응용하여 다음과 같이 유도된다.

$$\begin{aligned} W_{excb} &= W - \left( \frac{2^{k_a}}{N}\sum_{i=0}^{2^{k_a}-1}\frac{1}{2^{k_a}}(k_a+1) + \frac{2^{k_b}}{N}\sum_{i=0}^{2^{k_b}-1}\frac{1}{2^{k_b}}(k_b+2) \right. \\ &\quad \left. + \frac{2^{k_c}}{N}\sum_{i=0}^{2^{k_c}-1}\frac{1}{2^{k_c}}(k_c+3) \right) \quad (3) \\ &= W - \left( \frac{2^{k_a}(k_a+1) + 2^{k_b}(k_b+2) + 2^{k_c}(k_c+3)}{N} \right). \end{aligned}$$

식 (3)에서, $N$은 $2^{k_a}+2^{k_b}+2^{k_c}$이다. 이 방법에서는 클래스를 표시하는 비트들이 - 클래스 A, B, C일 때 각각 1, 2, 3 개의 비트 - 미리 경로 압축에 포함된다. 예를 들어, (그림 6)에서 보여주는 바와 같이 클래스 A의 프리픽스 수( $2^{k_a}$), 클래스 B의 프리픽스 수( $2^{k_b}$), 클래스 C의 프리픽스 수( $2^{k_c}$)의 비율이 약 2 : 1 : 8이므로, 트라이 높이 $W_{excb}$는 다음과

같이 유도할 수 있다.

$$W-\left(\frac{2\times 2^x(x-2+1)+2^x(x-3+2)+8\times 2^x(x+3)}{11\times 2^x}\right) \quad (4)$$
$$= W-\left(\frac{11x+21}{11}\right) \cong W-(x+2).$$

식 (4)에서, $2^{k_a}:2^{k_b}:2^{k_c}$가 $2:1:8$이므로, $k_a:k_b:k_c$는 $x-2:x-3:x$로 유도된다. 따라서 분할된 압축 트라이 $Trie_i$의 높이는 $(W-(x+2))$보다 적거나 같다.

라우터의 포워딩 테이블을 경로 압축 기법을 이용하여 트라이를 구성하는 경우 트라이의 높이가 $W$라면, 본 논문에서 제시한 임의의 $k$개의 비트를 사용하여 분할하는 기법을 이용하면 트라이의 높이를 $W-k$로 줄일 수 있다. 특히, IP 주소 체계를 이용하여 분할하는 기법은 임의의 $k$개의 비트를 사용하는 기법에 비해서 같은 개수의 $x$비트를 분할할 경우 2만큼의 높이를 더 줄일 수 있어서 메모리 접근 시간을 단축시킬 수 있다.

### 4.3 요구되는 메모리 크기

라우터의 포워딩 테이블을 분할된 압축 트라이로 구성하는데 요구되는 메모리 크기를 계산하기 위해서 임의의 압축 트라이 $Trie_i$의 메모리 크기를 $M_i^{P_k}$와 같이 정의한다. 즉, 임의의 $k$개의 비트를 사용하는 기법에서 $P_k$에 포함된 프리픽스들로만 이루어진 $Trie_i$의 메모리 크기를 나타낸다. 그러면, 임의의 $k$개의 비트를 사용하는 기법에서 요구되는 메모리 크기($M_{\text{incb}}$)는 식 (5)와 같이 유도된다.

$$M_{\text{incb}} = M^{R-array} + \sum_{i=0}^{N-1} M_i^{P_k} + M^{\text{Exp}}. \quad (5)$$

식 (5)에서, $M^{R-array}$는 루트 노드를 저장하는 배열의 크기이고, $M^{\text{Exp}}$는 적은 프리픽스 길이로 인하여 생성되는 확장된 프리픽스를 저장하는 메모리 크기이다. 임의의 $k$개의 숫자들 $j_0, j_1, \cdots, j_{k-1}$ 중에서 임의의 IP 프리픽스의 길이 $length(p)$보다 큰 숫자들의 개수를 $n$이라고 하면, 확장을 함으로써 늘어나게 되는 노드의 수는 $2^n$이 된다. 그러므로, 메모리 크기 $M_{\text{incb}}$은 다음과 같이 유도할 수 있다.

$$M_{\text{incb}} = N+\sum_{i=0}^{N-1} M_i^{P_k}, \text{ if } j_k < length(p) \text{ for all prefixes } p \text{ or,}$$
$$M_{\text{incb}} \le N+\sum_{i=0}^{N-1} M_i^{P_k}+2^n, \text{ otherwise.} \quad (6)$$

만약, 프리픽스를 확장을 하지 않는 경우에는 전체의 포워딩 테이블을 하나의 압축 트라이로 구성하였을 경우의 크기($M$)과 같다. 왜냐하면, 임의의 압축 트라이 $Trie_i$의 메모리 크기 $M_i^{P_k}$는 임의의 $k$개의 비트 열이 균일하게 분할할 경우 압축 트라이의 크기는 $\frac{M-N}{N}(=\frac{M}{N}-1)$ 값을 갖는다.

IP 주소 체계를 고려하여 프리픽스를 분할하는 기법으로 제안된 분할 압축 기법에서 요구되는 메모리 크기($M_{\text{excb}}$)는 식 (5)와 식 (6)을 응용하여 다음과 같이 유도된다.

$$\begin{aligned} M_{\text{excb}} &= 2^{k_a}+\sum_{i=0}^{2^{k_a}-1} M_i^{P_{k_a}}+M_a^{\text{Exp}}+2^{k_b}+\sum_{i=0}^{2^{k_b}-1} M_i^{P_{k_b}}+M_b^{\text{Exp}} \\ &\quad +2^{k_c}+\sum_{i=0}^{2^{k_c}-1} M_i^{P_{k_c}}+M_c^{\text{Exp}} \\ &= (2^{k_a}+2^{k_b}+2^{k_c})+\sum_{i=0}^{2^{k_a}-1} M_i^{P_{k_a}}+\sum_{i=0}^{2^{k_b}-1} M_i^{P_{k_b}}+\sum_{i=0}^{2^{k_c}-1} M_i^{P_{k_c}} \\ &\quad +(M_a^{\text{Exp}}+M_b^{\text{Exp}}+M_c^{\text{Exp}}) \\ &= N+\sum_{i=0}^{2^{k_a}-1} M_i^{P_{k_a}}+\sum_{i=0}^{2^{k_b}-1} M_i^{P_{k_b}}+\sum_{i=0}^{2^{k_c}-1} M_i^{P_{k_c}}+M^{\text{Exp}}. \end{aligned} \quad (7)$$

식 (7)에서, $M_a^{\text{Exp}}$, $M_b^{\text{Exp}}$, $M_c^{\text{Exp}}$는 각각의 클래스에서 확장하는데 증가되는 노드의 개수로 모두 합한 값을 $M^{\text{Exp}}$으로 표현하였다. IP 주소 체계를 고려하여 프리픽스를 분할하는 기법도 프리픽스 확장을 하지 않는 경우에는 $M$과 같다.

프리픽스를 확장을 하지 않는 경우에는 포워딩 테이블을 분할 압축 트라이로 구성하는데 추가로 요구되는 메모리의 량은 늘어나지 않는다. IP 주소를 검색하기 위해서 소요되는 메모리 접근 횟수가 IP 주소 체계를 고려하여 분할하는 기법이 그렇지 않는 기법에 비해서 더 적음을 알 수 있다.

IPv4 환경에서는 $m$이 32이므로, 임의의 $k$ 또는 $x(x_a, x_b, x_c)$를 32보다 적은 임의의 값을 선택할 수 있지만, IP 프리픽스의 길이가 $j_l, 1 \le l \le k$ 보다 적은 경우 프리픽스를 확장하여야 하는 경우가 발생한다. 이러한 경우에는 주어진 IP 프리픽스에 대해서 중복된 프리픽스가 여러 개의 분할 트라이에 존재하게 되어 메모리를 낭비하게 된다. <표 1>에서 보여진 바와 같이 1에서 7까지의 길이를 갖는 프리픽스가 없기 때문에 프리픽스를 확장을 할 필요가 없어서 추가되는 메모리는 없다. 또한, 8에서 10까지도 전체 프리픽스 중에 약 0.02% 정도이기 때문에 프리픽스를 확장하는 경우가 발생하지만 프리픽스의 개수가 적기 때문에 필요한 메모리의 요구량이 적다. <표 1>의 특성으로부터 임의의 $k$ 또는 $x(x_a, x_b, x_c)$개의 비트를 사용하는 기법에서 $k$를 7로 정하고, $j_1=1, j_2=2, \cdots, j_k=7$로 정하면 IP 프리픽스를 확장하지 않으므로 추가로 요구되는 메모리 없이 7만큼 트라이의 높이를 줄일 수 있다. 한편, IP 주소 체계를 고려하여 프리픽스를 분할하는 기법에서 $x$를 7로 하면, 클래스 A인 $x_a$는 2부터 6까지 5개이고 클래스 B인 $x_b$는 3부터 6까지 4개이고 클래스 C인 $x_c$는 4부터 10까지 7개이다. 이 경우 클래스 C는 확장을 해야할 IP 프리픽스가 존재하지만 비율이 극히 미미하여 추가되는 메모리가 무시할 정도로 적다. IP 주소 체계를 고려한 방법은 9만큼 트라이의 높이를 줄일 수 있다.

## 5. 결 론

IP 트래픽의 폭발적인 증가로 인하여 라우터의 IP 패킷 전달율이 인터넷의 병목 요인이다. 본 논문에서는 IP 패킷 전달율에 큰 영향을 주는 요인인 IP 주소 검색 기법의 속

도 향상을 위한 방법을 제시하였다. 트라이 자료 구조 기반의 IP 주소 검색 기법의 속도 향상을 위해서 포워딩 테이블에 있는 IP 프리픽스를 여러 개의 압축 트라이로 분할하고, 주어진 IP 목적지 주소에 대해서 하나의 압축 트라이에서만 검색이 수행되도록 하는 분할된 압축 트라이 구조를 제안하였다. 제안된 구조는 압축 트라이의 근본적인 경로 압축 효과를 응용하여 IP 주소 검색에 소요되는 메모리 접근 횟수를 줄였으며, 포워딩 테이블 저장에 필요한 메모리의 크기가 기존 압축 트라이와 같음을 보였다. 또한, IP 주소를 분할하는 기법으로 IP 주소 체계 특성을 고려한 분할 방법이 고려하지 않는 방법에 비해서 메모리 참조 횟수가 적어짐을 보였다. 본 논문에서 제안한 분할된 압축 트라이는 기존의 트라이 자료 구조 기반의 IP 주소 검색을 사용하는 라우터의 성능 향상에 적용할 수 있으며, IPv6인 경우 호스트의 주소할당 체계가 특정 영역별로 구별지어져 있으므로 IPv6 환경에서도 적용할 수 있다.

## 참 고 문 헌


[1] J. Aweya, "On the Design of IP Routers Part 1 : Router Architectures," Journal of Systems Architecture, Vol.46, No.6, pp.483-511, Apr., 2000.

[2] W. Doeringer, G. Karjoth and M. Nassehi, "Routing on Longest Matching Prefixes," IEEE/ACM Transaction on Networking, Vol.4, pp.86-97, Feb., 1996.

[3] V. Fuller, T. Li, J. Yu and K. Varadhan, "Classless Inter-Domain Routing (CIDR) and Address Assignment and Aggregation Strategy," RFC1519, Sep., 1993.

[4] P. Gupta, S. Lin and N. Mckweon, "Routing Lookups in Hardware at Memory Access Speeds," Proc. of INFOCOM, pp.1240-1247, 1998.

[5] E. Horowitz and S. Sahni, "Fundamentals of Data Structures in C," Computer Science Press, 1993.

[6] S. Keshav and R. Rharma, "Issues and Trends in Router Design," IEEE Communications Magazine, Vol.36, No.5, pp. 144-151, May, 1998.

[7] 이인복, 박근수, 최양희, 정성권, "세그먼트 트리를 이용한 IP 주소 검색", 정보과학회지 : 시스템 및 이론, 제28권 제11호, pp.613-619, 2001.

[8] A. J. McAuley and P. Francis, "Fast Routing Table Lookup using CAMs," Proc. of IEEE Infocom, pp.1382-1391, 1993.

[9] "BGP Table Statistics," http://bgp.potaroo.net.

[10] MMC Networks Co., "EPIF4-L3 Reference Manual," Oct., 1998.

[11] D. Morrison, "PATRICIA-Practical Algorithm To Retrieve Information Coded In Alphanumeric," Journal of ACM, Vol. 5, No.4, pp.514-534, Oct., 1968.

[12] S. Nilsson and G. Karlsson, "IP-Address Lookup using LC-Tries," Journal of Selected Areas in Communications, Vol. 17, No.6, pp.1083-1092, Jun., 1999.

[13] V. Srinivasan and G. Varghese, "Fast Address Lookups using Controlled Prefix Expansion," ACM Trans. on Computer Systems, Vol.17, No.1, pp.1-40, Feb., 1999.

[14] K. Sklower, "A Tree-based Routing Table for Berkeley Unix," Technical Report, Univ. of California, Berkeley, 1993.

[15] H. Tzeng and T. Przygienda, "On Fast Address-Lookup Algorithms," IEEE Journal of Selected Areas in Communications, Vol.17, No.6, pp.1067-1082, Jun., 1999.

[16] P.-C. Wang, C.-T. Chan and Y.-C. Cheng, "A Fast IP Routing Lookup Scheme," Proc. of ICC, pp.1140-1144, 2000.



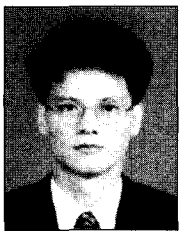
**박 재 형**

e-mail : hyeoung@chonnam.ac.kr
1987년~1991년 연세대학교 전산학과 (학사)
1991년~1993년 KAIST 전산학과(석사)
1993년~1997년 KAIST 전산학과(박사)
1997년~1998년 KAIST 인공지능연구센터 Post-Doc 연수연구원
1998년~2002년 ETRI 네트워크연구소 선임연구원
2002년~현재 전남대학교 전자컴퓨터정보통신공학부 조교수
관심분야 : MPLS, 인터넷 프로토콜, 인터넷 시스템, 라우터 구조, 이더넷 네트워크 연결망, 병렬처리, 멀티캐스트



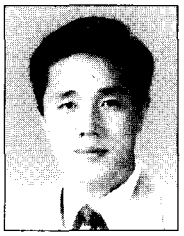
**장 익 현**

e-mail : ihjang@mail.dongguk.ac.kr
1980년~1984년 서울대학교 계산통계학과 (학사)
1984년~1986년 KAIST 전산학과(석사)
1993년~1998년 KAIST 전산학과(박사)
1986년~1999년 (주)데이콤 종합연구소 책임연구원
1999년~현재 동국대학교 정보통신공학과 조교수
관심분야 : 인터넷 프로토콜, 인터넷 서비스, 병렬처리, 분산 시스템


**정 민 영**

e-mail : mychung@ece.skku.ac.kr
1986년~1990년 KAIST 전기및전자공학과 (학사)
1992년~1994년 KAIST 전기및전자공학과 (석사)
1994년~1999년 KAIST 전기및전자공학과 (박사)
1999년~2002년 ETRI 네트워크연구소 선임연구원
2002년~현재 성균관대학교 정보통신공학부 조교수
관심분야 : 차세대 인터넷, 광/홈/Ad hoc 네트워크, 이동통신망 성능 및 신뢰성 분석


**원 용 관**

e-mail : ykwon@chonnam.ac.kr
1986년 한양대학교 전자공학과 학사
1991년 Univ. of Missouri, 컴퓨터공학 석사
1995년 Univ. of Missouri, 컴퓨터공학 박사
1991년~1995년 Univ. of Missouri, Research/Teaching Assistant
1995년~1996년 한국전자통신연구원
1996년~1999년 한국통신 연구개발본부
1999년~현재 전남대학교 정보통신공학부/의학부 부교수
2002년~현재 한국생물정보학회 Proteome Informatics 연구회장
관심분야 : 네트워크관리 및 보안, 컴퓨터비젼, 바이오정보학