제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

압축 Trie에 기반한 IP 주소 탐색의 성능 향상을 위한 기법

장익현*, 박재형**, 정민영***
*동국대학교 정보통신공학과
**전남대학교 전자컴퓨터정보통신공학부, Advanced Information Technology Research Center(AITrc)
***성균관대학교 정보통신공학부
e-mail:ihjang@mail.dongguk.ac.kr, hyeoung@chonnam.ac.kr, mychung@ece.skku.ac.kr

A Performance Enhancement Scheme of IP Address Lookups based on Compressed-Tries

Ikhyeon Jang*, Jaehyung Park**, Min Young Chung***
*Department of Information Communication Engineering, Dongguk University
**Dept. of Electronics, Computer, Information Engineering, Chonnam National University, Advanced Information Technology Research Center(AITrc)
***School of Information and Communication Engineering, Sungkyunkwan University

요 약

포워딩 엔진은 외부 인터페이스를 통해서 들어오는 패킷에 대해서 IP 주소를 기반으로 목적지로 향하는 다음 홉을 결정한다. 이러한 고성능의 패킷 처리를 위한 포워딩 엔진을 설계함에 있어서 IP 주소 탐색은 중요한 요인이다. 본 논문에서는 검색경로 압축 트라이에 기반한 IP 주소 탐색 알고리즘의 성능을 향상시키는 분할 압축 탐색 기법을 제시한다. 제시된 분할 기법을 통해서 IP 주소 탐색의 메모리 접근 횟수를 줄일 수 있으며, 프로그램 가능한 네트워크 프로세서에 적용할 수 있다.

1. 서론

인터넷을 통한 서비스 및 응용 프로그램이 다양해짐에 따라서 인터넷 사용자 수가 증가하여 트래픽의 양이 급격하게 증가하고 있다. 전달 매체의 발달로 트래픽을 전달하는 링크는 높은 대역폭과 고속의 전송이 가능하게 됨에 따라서, 패킷을 처리하는 라우터가 인터넷 망의 성능에 병목이 되고 있다[5].

라우터는 라우팅 제어 프로세서, I/O 인터페이스들, 포워딩 엔진과 그들간의 연결을 지원하는 스위칭 패브릭으로 구성되어 있다. 그 중 패킷 전달을 담당하는 포워딩 엔진은 라우터에 있어서 중요한 역할을 수행한다. 포워딩 엔진은 입력 인터페이스를 통해서 들어오는 패킷을 목적지와 출력 인터페이스가 결합된 포워딩 테이블을 참조하여 패킷을 전달한다. 이러한 과정이 IP 주소 검색이며, 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 작업이다[2].

포워딩 엔진에서 패킷을 처리함에 있어서 성능에 큰 영향을 미치는 요소는 포워딩 테이블을 검색하는 IP 주소 검색이다. 고속의 패킷 처리를 지원하는 패킷 포워딩 엔진은 IP 주소를 검색하는 검색 엔진이

본 연구는 첨단정보기술 연구센터과제 "고성능 보안 미디어 네트워크"를 통하여 과학재단의 지원을 받았음.

따로 구성된다. MMC사의 NP시리즈[4]의 네트워크 프로세서에서는 Trie 구조의 IP 주소 검색 기법을 사용하고 있다.

고속의 IP 주소 검색을 위해서 많은 기법들이 제시되었다. 그러한 기법 중에 Patricia Trie[6]는 radix trie의 특별한 형태로서 현존하는 라우터에 구현되었다. Patricia Trie는 하나의 자식 노드를 갖는 노드가 없는 검색경로 압축 트라이로써, 검색시에 재귀적인 후진탐색이 필요하여 $O(W^2)$번의 메모리 접근이 요구된다, 이 때 W는 트라이의 최대 높이이다. 이와 같은 재귀적인 후진탐색을 피하기 위해서 동적인 프리픽스 트라이가 제시되었으며[1], 이 기법의 메모리 접근 횟수는 $O(W)$이다. 고속의 IP 주소 검색을 위해서 Patricia Trie의 변형들이 계속 연구중이다[7].

본 논문에서는 고성능의 패킷 포워딩 엔진을 위한 IP 주소 검색의 성능 향상을 지원하는 구조로 분할된 경로 압축 트라이 알고리즘을 제시한다. 제시된 기법은 검색경로 압축의 특성을 이용함으로써 IP 주소 검색 시에 메모리 참조 횟수를 줄일 수 있다. 메모리 접근 횟수를 줄임으로써 트라이 기반의 IP 주소 검색 기법의 성능을 향상시켜 높은 성능의 패킷 포워딩 엔진에 적용될 수 있다.

2. 연구배경

본 절에서는 경로 압축 트라이의 구조에 대해서 기술하고, IP 주소의 체계에 대해서 살펴본다.

2.1 경로 압축 트라이

트라이는 트리와 유사한 자료 구조이며[3], 하나 이상의 자식 노드를 갖는 노드로 구성된다. 특정 키에 대한 검색은 트라이의 루트에서 시작하여 가장 긴 것과 일치하는 것을 찾기 위해서 내려가면서 탐색한다. 각 노드에서는 특정 키의 일부분을 이용하여 다음에 탐색하여야 할 노드를 결정한다. 그림 1은 0000, 0001, 001*, 1000, 1001, 11*의 6개의 프리픽스로 구성된 이진 트라이의 예를 보여준다.

검색경로 압축 트라이는 트라이를 기반으로 구성되며, 그림 1의 예에 대한 검색경로 압축 트라이는 그림 2와 같이 표현된다. 하나의 자식 노드를 갖는 노드를 제거하기 위해서 각각의 가지 노드에 비트 번호가 부가된다. 이렇게 함으로써 원래의 이진 트라이의 높이보다 더 낮은 높이를 유지할 수 있다. 즉, 특정 키에 대한 탐색시에 메모리 접근 횟수를

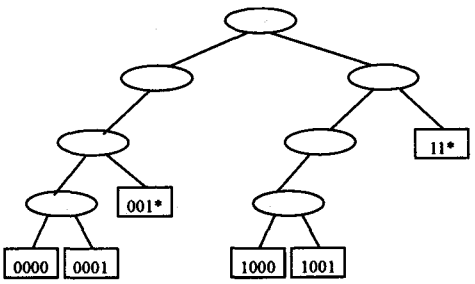

그림 1 이진 트라이의 예

줄이는 요인이 된다. 검색경로 압축 트라이의 노드에 부가된 비트 번호는 가지 노드에서 탐색시에 검사해야할 비트 번호를 의미한다.

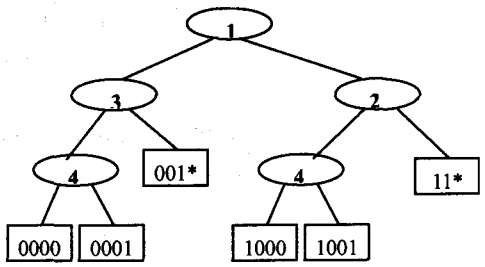

그림 2 검색경로 압축 트라이의 예

2.2 IP 주소 체계

IP 주소는 크게 4개의 클래스로 구분된다. 클래스 A, B, C는 일반적으로 사용하는 주소이고, 클래스 D는 멀티캐스트를 위해서 특별한 목적으로 사용하는 주소이다. 본 논문에서는 클래스 D에 대한 내용은 제외한다. 왜냐하면 대부분의 라우터에서 멀티캐스트 용 라우팅 테이블이 따로 관리되기 때문이다. 각각의 주소의 특성은 표 2에 보여진 것과 같이 총 32비트 중에서 앞쪽의 비트는 클래스를 나타내는데 쓰이고 나머지 비트들이 IP 주소의 프리픽스로 쓰인다.

클래스	형식
Class A	0nnnnnnnhh...h
Class B	10nnnnnnnnnnnnnnhh...h
Class C	110nnnnnnnnnnnnnnnnnnnnnhh...h
Class D	1110mm...m

표 1 IP 주소 클래스

IP 주소 형태는 표 1에서 보여주는 바와 같이 클래스 별로 클래스를 지칭하는 비트들과 네트워크 주소, 그리고 호스트의 주소로 표현된다. 각각 클래스 별로 네트워크 주소를 나타내는 비트가 7, 14, 21이다.

3. 분할된 경로 압축 트라이

본 절에서는 경로 압축 트라이 기반의 IP 주소 검색 알고리즘의 성능을 높일 수 있는 분할된 경로 압축 트라이를 제안한다.

3.1 분할된 압축 트라이의 구조

분할된 경로 압축 트라이는 N개의 압축 트라이와 N개의 항목을 저장하는 $R-array$로 구성되어 있다. 그림 3에서와 같이 각각에 해당하는 IP 프리픽스들로 구성된 N개의 압축 트라이($Trie_i$)와 해당되는 트라이의 루트 노드를 $R-array[i]$에 저장한다, $0 \le i \le N-1$. 그림 3은 그림 2의 예에서 프리픽스의 앞의 2 비트들을 이용하여 분할한 기법으로 $N(4=2^2)$개의 압축 트라이로 구성된다.

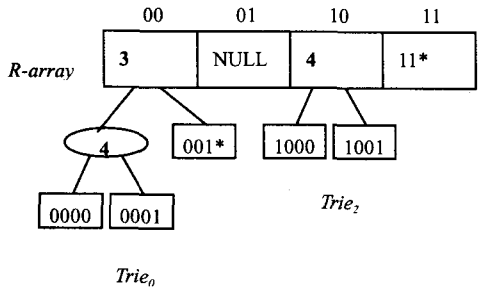

그림 3 분할된 압축 트라이

3.2 임의의 k비트를 이용한 분할 기법

임의의 k개의 비트를 이용한 분할 기법에서는 라우팅 테이블에 존재하는 모든 프리픽스를 특정한 비트 위치의 k개의 비트 값이 같은 프리픽스들로 분할한다. 그러면 N은 2^k가 성립한다. 각각 분할된 프리픽스들로 경로 압축 트라이를 구성하면, 임의의 $Trie_i, 0 \le i \le N-1,$의 높이는 k만큼 감소한다. 그 이유는 각각 분할된 프리픽스는 특정한 비트 위치의 k개의 비트가 같기 때문이다.

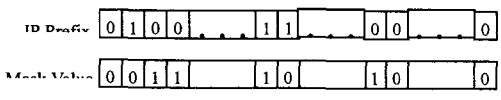

그림 4 임의의 k개의 비트로 분할하는 방법

IP 프리픽스를 N개의 압축 트라이($Trie_i$)로 분할을 하기 위해서는 그림 4와 같이 주어진 IP 프리픽스에 대해서 마스크 값과 논리-AND를 수행한다. 이 때, 마스크 값은 특정한 비트 위치의 k개의 비트 값이 모두 1로 되어 있다. 그리고, 해당되는 트라이의 번호 i를 구하기 위해서는 논리-AND의 결과에서 마스크 값에 1로 셋팅되어 있는 값만을 추출하여 숫자화한다. 이러한 숫자화에는 비트의 SHIFT연산으로 이루어진다.

분할을 하는 과정에서 다음과 같은 두 가지의 경우가 발생한다.

① IP 프리픽스의 크기가 k개의 특정한 비트 위치를 가리키는 위치 값보다 큰 경우

② IP 프리픽스의 크기가 k개의 특정한 비트 위치를 가리키는 위치 값보다 작은 경우

①번의 경우에는 주어진 프리픽스에 대한 분할 기법은 아무런 변형없이 가능하나, ②번의 경우에는 주어진 프리픽스를 해당되는 비트 위치가 0인 프리픽스와 1인 프리픽스로 확장을 하여야 한다.

3.3 IP 주소 체계의 특성을 이용한 분할 기법

IP 주소 체계의 특성을 이용한 분할 기법에서는 IP 주소의 최고 높은 비트에 주소의 클래스를 의미하는 비트들을 함께 고려하는 기법이다. IP 주소가 어느 클래스에 속해 있느냐에 따라서 프리픽스의 길이가 정해지기 때문에 임의의 k개의 비트를 이용한 분할 기법을 사용하면 프리픽스를 확장하여야 하는 부담이 많아진다.

IP 주소 특성을 이용한 분할 기법에서는 주소 클래스 별로 클래스 A에는 x_a개, 클래스 B에는 x_b개, 클래스 C에는 x_c개의 임의의 비트를 이용하여 분할한다. 그러면, 각각의 분할된 압축 트라이의 수는 클래스 A에는 2^{x_a}개, 클래스 B에는 2^{x_b}개, 클래스 C에는 2^{x_c}개다. 그리고, 분할된 압축 트라이의 전체 개수 N은 $2^{x_a}+2^{x_b}+2^{x_c}$이다. 이 기법에서의 $R-array$의 인덱스 i는 다음과 같이 계산된다.

$$i_b\, i = \begin{cases} i_a, & \text{if prefix} \in \text{Class A} \\ i_b + 2^{x_a}, & \text{if prefix} \in \text{Class B} \\ i_c + 2^{x_a} + 2^{x_b}, & \text{if prefix} \in \text{Class C} \end{cases} .$$

이 때, 숫자 i_a, , i_c는 각각의 클래스 내에서 사용한 임의의 x_a, x_b, x_c를 마스크 값과 논리-AND를 취하고 SHIFT연산을 수행한 후 얻은 값이다.

4. 산술적인 평가

우선 CIDR 환경에서 현존하는 라우터의 IP 주소 프리픽스의 특성을 알아보기로 하자. 그림 5는 네트워크 엑세스 포인트 중의 중요한 3지점인 MaeEast, AADS, PacBell에서 추출한 프리픽스의 분포이다. 대부분의 IP 주소 프리픽스가 24 비트에 포함된다.

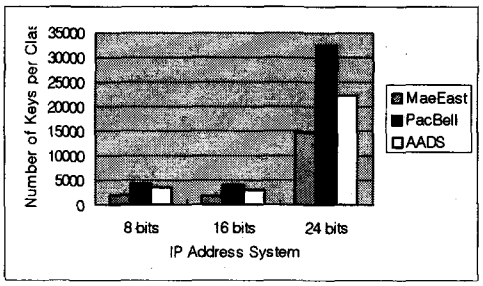

그림 5. IP 주소 프리픽스의 분포

제안된 분할 압축 트라이의 성능을 평가하기 위해서 메모리의 접근 횟수를 비교하여 보자. 트라이 구조에서의 메모리 접근 횟수는 분할된 압축 트라이의 높이에 비례한다. 모든 IP 프리픽스에 대해서 검색할 확률이 동일하다고 가정한다면, 모든 분할된 압축 트라이가 선택될 확률 또한 동일하므로, 분할된 압축 트라이의 높이는 다음 식과 같이 유도된다.

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

이 때, W_i는 분할된 압축 트라이 i의 높이이고, W는 하나의 압축 트라이로 구성하였을 때의 높이이다. 분할된 압축 트라이는 하나의 경로 압축 트라이에서 k비트를 압축한 트라이이므로 제안된 분할 압축 트라이 i의 높이는 $(W-k)$보다 적거나 같다. 그러므로, 한번의 IP 주소 검색을 위해서 접근하는 메모리 접근 횟수가 k만큼 감소함을 알 수 있다.

IP 주소 체계를 이용하여 프리픽스를 분할하는 기법으로 제안된 분할 압축 트라이의 높이는 위의 식을 응용하여 다음과 같이 유도된다.

$$W-\left(\frac{2^{x_a}}{N}\sum_{i=0}^{2^{x_a}}\frac{1}{2^{x_a}}(x_a+1) + \frac{2^{x_b}}{N}\sum_{i=0}^{2^{x_b}}\frac{1}{2^{x_b}}(x_b+2)+\frac{2^{x_c}}{N}\sum_{i=0}^{2^{x_c}}\frac{1}{2^{x_c}}(x_c+3)\right)$$
$$= W-\left(\frac{2^{x_a}(x_a+1)+2^{x_b}(x_b+2)+2^{x_c}(x_c+3)}{N}\right)$$

클래스를 표시하는 비트들- 클래스 A, B, C일 때 각각 1, 2, 3 개의 비트 -이 미리 경로 압축에 포함되기 때문이다. 그림 5에서 보여주는 바와 같이 클래스 A의 프리픽스 수(2^{x_a}), 클래스 B의 프리픽스 수(2^{x_b}), 클래스 C의 프리픽스 수(2^{x_c})의 비율이 약 1:1:8이므로 위의 식은 다음과 같이 수식화 할 수 있다.

$$W-\left(\frac{2^x(x+1)+2^x(x+2)+8\times 2^x((x+3)+3)}{10\times 2^x}\right)$$
$$= W-\left(\frac{10x+51}{10}\right) = W-(x+5.1)$$

$2^{x_a}:2^{x_b}:2^{x_c}$가 1:1:8이므로 $x_a:x_b:x_c$는 $x:x+1:x+3$이다. 따라서 분할된 압축 트라이 i의 높이는 $(W-(x+5.1))$보다 적거나 같다. IP 주소 체계를 이용하여 분할하는 기법은 임의의 비트를 사용하는 기법에 비해서 같은 개수의 k비트를 분할할 경우 5.1만큼의 높이를 줄일 수 있어서 메모리 접근 시간을 단축시킬 수 있다.

5. 결론

본 논문에서는 고성능의 패킷 포워딩 엔진을 설계하는데 있어서 병목 요인이 되는 IP 주소 검색의 속도를 향상시킬 수 있는 구조와 기법에 대해서 제시하였다. 검색경로 압축 트라이에 기반하여 분할된 압축 트라이 기법을 제시하였으며 검색경로 압축 트라이의 본질적인 특성을 통해서 IP 주소 검색을 위한 메모리 접근 횟수를 감소시켰다.

참고문헌

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

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

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

[4] MMC Networks, "EPIF4-L3 Reference Manual", 1998.

[5] S. Keshave and R. Rharma, "Issues and Trends on Router Design", IEEE Communication Magazine, vol.36, no.5, pp.144-151, May 1998.

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

[7] S. Nilsson and G. Karlsson, " IP-Address Lookup using LC-Tries", Journal of Selected Areas in Communications, vol.17, pp.1083-1092, Jun. 1999.