빠른 IP 주소 탐색을 위한 분할 압축 Trie

장익현*, 도재수**, 박재형***
*동국대학교 정보통신공학과,
**동국대학교 컴퓨터학과
***전남대학교 전자컴퓨터정보통신공학부, AITrc
e-mail:{ihjang,dojesu}@dongguk.ac.kr, hyeoung@chonnam.ac.kr

A Partitioned Compressed-Trie for Fast IP Address Lookups

Ikhyeon Jang*, Jae-su Do**, Jaehyung Park***
*Department of Info. Comm. Eng., Dongguk University,
**Department of Computer Science, Dongguk University,
***Dept. of Electronics, Computer, Information Engineering,
Chonnam National University, Advanced Information Technology Research Center(AITrc)

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 성능 요인이다. 본 논문에서는 검색 경로 압축 트라이에 기반한 IP 주소 탐색의 성능을 향상시키는 분할된 경로압축트라이 구조를 제안한다. 제안된 기법은 IP주소를 여러 개의 분할 압축 트라이로 나누어서 주소탐색이 하나의 분할된 압축트라이에서만 이루어지도록 하여 탐색시간을 줄인다.

1. 서론

인터넷을 통한 서비스가 다양해지고 사용이 증가됨에 따라 트래픽 양이 급격하게 증가하고 있다. 인터넷 서비스의 품질을 적절하게 유지하기 위해서는 전송과 교환의 고속화가 요구되고 있다. 전달 매체의 발달로 트래픽을 전달하는 링크의 고속화가 달성됨에 따라 패킷을 처리하는 라우터가 인터넷 망 성능의 병목요인이 되고 있다[5].

라우터는 라우팅 제어 프로세서, I/O 인터페이스, 포워딩 엔진과 그들 간의 연결을 지원하는 스위칭 패브릭으로 구성되어 있다. 포워딩 엔진은 입력 인터페이스를 통해서 들어오는 패킷을 목적지와 출력 인터페이스가 결합된 포워딩 테이블을 참조하여 패킷을 전달한다. 이러한 과정이 IP 주소 탐색이며, 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 작업이 되고 있다[3].

포워딩 엔진에서 패킷 처리 성능에 큰 영향을 미치는 요소는 포워딩 테이블을 검색하는 IP 주소 탐색이다. 고속의 패킷 처리를 지원하는 패킷 포워딩 엔진의 경우 IP 주소를 탐색하는 탐색 엔진이 따로 구성된다.

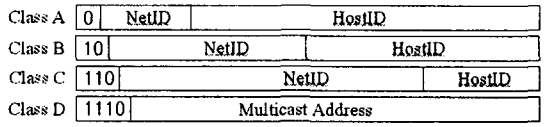
고속의 IP 주소 탐색을 위한 많은 기법들이 제시되고 있다. 그러한 기법 중에서 Patricia Trie[6]는 radix trie의 특별한 형태로 현존하는 라우터에 구현되었다. Patricia Trie는 자식 노드를 하나만 가지는 노드가 없는 탐색 경로 압축 트라이로, 탐색 시에 재귀적인 후진탐색이 필요하여 $O(W^2)$번의 메모리 접근이 요구된다, 단 W는 트라이의 최대 높이이다. 이와 같은 재귀적인 후진탐색을 피하기 위해서 동적 프리픽스 트라이가 제시되었으며[2], 이 기법의 메모리 접근 횟수는 $O(W)$ 이다. 고속의 IP 주소 탐색을 위한 Patricia Trie의 변형들이 계속적으로 연구되고 있다[7].

본 논문에서는 고속의 IP 주소 탐색을 위한 분할된 경로압축트라이를 제시한다. 제안된 방법은 IP주소를 여러 개의 분할 압축트라이로 구성하여 IP주소 탐색이 하나의 분할 압축트라이에서 이루어지도록 하여 IP 주소 탐색 시간을 줄인다. IP 주소 탐색에 큰 영향을 미치는 메모리참조횟수가 감소함을 보이고 추가로 요구되는 메모리도 거의 없음을 보인다.

2. 연구배경

2.1 IP 주소 체계

IP 주소 형태는 그림 1에서 보여주는 바와 같이 클래스 별로 클래스를 지칭하는 비트들과 네트워크 ID와 호스트 ID로 표현된다. A, B, C 각 클래스의 네트워크 ID를 나타내는 비트 수는 7, 14, 21이다.

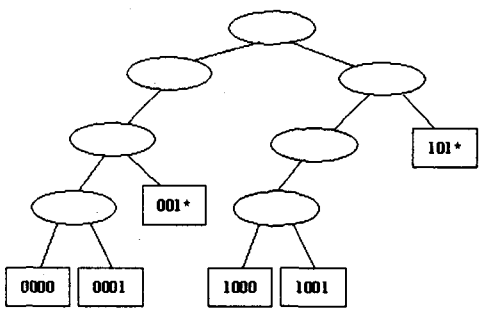


[그림 1] IP 주소 클래스

IP 주소 부족 문제를 해결하기 위해 제시된 CIDR은 클래스 개념을 없애고 프리픽스 개념을 도입하여 가변길이의 IP주소 부분을 네트워크번호로 사용한다. IP 주소 프리픽스는 클래스를 나타내는 비트를 포함한다.

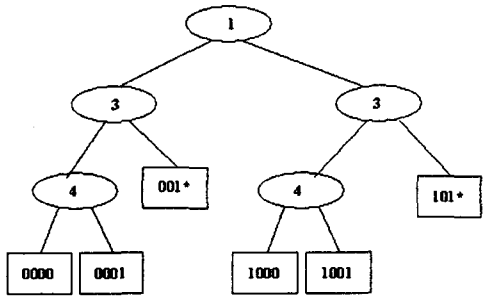
2.2 경로압축트라이

트라이는 트리와 유사한 자료 구조이며[4], 하나 이상의 자식 노드를 갖는 노드로 구성된다. 키에 대한 검색은 트라이의 루트에서 시작하여 가장 긴 것과 일치하는 것을 찾기 위해서 내려가면서 탐색한다. 각 노드에서는 키의 일부분을 이용하여 다음에 탐색하여야 할 노드를 결정한다. 그림 2는 0000, 0001, 001*, 1000, 1001, 101*의 6개의 프리픽스로 구성된 이진 트라이의 예를 보여준다.



[그림 2] 이진트라이의 예

경로압축트라이[2]는 트라이를 기반으로 구성되며, 그림 2의 예에 대한 경로압축트라이는 그림 3과 같이 표현된다. 하나의 자식 노드를 갖는 노드를 제거하기 위해서 각각의 가지 노드에 비트 번호가 부여된다. 이렇게 함으로써 원래의 이진 트라이의 높이보다 더 낮은 높이를 유지할 수 있다. 즉, 특정 키에 대한 탐색 시에 메모리 접근 횟수를 줄이는 요인이 된다. 경로압축트라이의 노드에 부여된 비트 번호는 가지 노드에서 탐색 시에 검사해야 할 비트 번호를 의미한다.

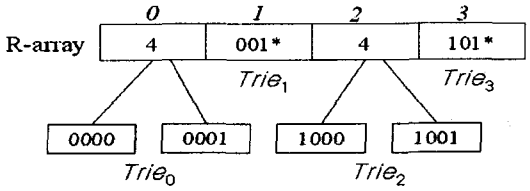


[그림 3] 경로압축트라이의 예

3. 분할된 경로압축트라이

3.1 분할된 경로압축트라이의 구조

분할된 경로압축트라이는 N 개의 압축트라이와 N 개의 항목을 저장하는 $R-array$ 로 구성되어 있다. 그림 4는 그림 3의 예에서 프리픽스의 첫 번째와 세 번째 비트를 이용하여 경로압축트라이를 분할하는 방법으로 $4(N=2^2)$개의 압축 트라이로 구성된 분할된 경로압축트라이의 예를 보여준다. 그림 4에서 분할된 각 IP 프리픽스로 구성된 N 개의 경로 압축 트라이 ($Trie_i$)와 해당되는 트라이의 루트 노드는 $R-array[i]$ 에 저장된다, $0 \le i \le N-1$.



[그림 4] 분할된 압축 트라이의 예

3.2 임의의 k 비트를 이용한 분할 기법

임의의 k 개의 비트를 이용한 분할 기법에서는 라우팅 테이블에 존재하는 모든 프리픽스를 특정한

비트 위치의 k개의 비트 값이 같은 프리픽스들로 분할하며, 분할된 압축 트라이의 수 N은 2^k개가 된다. 분할된 프리픽스들로 경로압축트라이를 구성하면, 임의의 $Trie_i(0 \le i \le N-1)$의 높이는 k만큼 감소한다. 그 이유는 분할된 각각의 프리픽스는 특정한 비트 위치의 k개의 비트가 같기 때문이다.

분할을 하는 과정에서 다음과 같은 두 가지 경우가 발생한다. 첫 번째는 IP 프리픽스의 길이가 k개의 특정한 비트 위치를 가리키는 가장 큰 위치 값보다 큰 경우이고, 두 번째는 IP 프리픽스의 크기가 k개의 특정한 비트 위치를 가리키는 가장 큰 위치 값보다 작은 경우이다. 첫 번째 경우는 주어진 프리픽스에 대한 분할 기법을 그대로 적용할 수 있으나, 두 번째 경우에는 주어진 프리픽스를 프리픽스의 길이보다 큰 위치에 해당되는 모든 비트에 대해 비트 위치가 0과 1인 프리픽스로 확장해야 한다.

3.3 IP 주소체계를 고려한 분할 기법

IP 주소체계의 특성을 이용한 분할 기법에서는 IP 프리픽스를 클래스를 나타내는 비트와 분할 기준으로 사용할 비트의 두 부류로 나누어 고려한다.

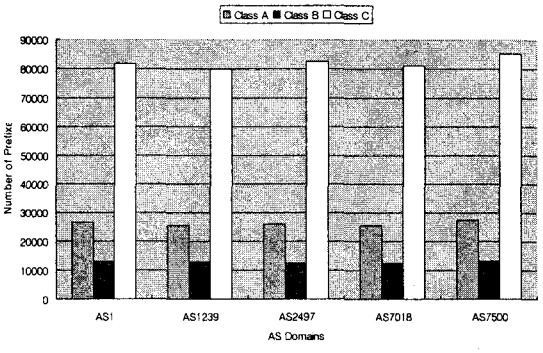
IP 주소체계를 고려한 분할 기법에서는 주소 클래스 별로 클래스 A는 K_A개, 클래스 B는 K_B개, 클래스 C는 K_C개의 임의의 비트를 이용하여 분할한다. 그러면 각각의 분할된 압축트라이의 수는 클래스 A에는 2^{K_A}개, 클래스 B에는 2^{K_B}개, 클래스 C에는 2^{K_C}개가 된다. 따라서 분할된 압축트라이의 전체 개수 N은 $2^{K_A}+2^{K_B}+2^{K_C}$이 된다. 이 기법에서 $R-array$의 인덱스 i는 다음과 같이 계산된다.

$$i=\begin{cases} i_a, & \text{if 프리픽스} \in \text{Class A} \\ i_b+2^{K_A}, & \text{if 프리픽스} \in \text{Class B} \\ i_c+2^{K_A}+2^{K_B}, & \text{if 프리픽스} \in \text{Class C} \end{cases}$$

이때 i_a, i_b, i_c는 각 클래스 내에서 사용한 임의의 K_A, K_B, K_C개의 비트로 만들어지는 값이다.

4. 성능 평가

우선 CIDR 환경에서 현존하는 라우터의 IP 주소 프리픽스의 특성을 알아보기로 하자. 그림 5와 표 1은 2003년 7월 1일에 서로 다른 5개의 AS에서 수집된 IP주소 프리픽스와 프리픽스 길이의 클래스별 분포를 보여주는 자료이다[1].



[그림 5] IP 주소 프리픽스의 분포

그림 5를 보면 대부분의 IP주소는 클래스 C에 속해 있으며 CIDR의 도입에도 불구하고 클래스 구조가 완전히 사라지지 않았음을 알 수 있다.

[표 1] IP주소 프리픽스 길이의 클래스별 분포

prefix	AS1			AS1239			AS2497		
길이	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
25-32	0	0	0	0	0	0	0	0	0
계	26528	13161	81673	25564	12953	79697	25945	12513	82388

표 1을 보면 프리픽스의 길이가 8보다 작거나 25보다 큰 경우는 없으며, 길이가 11보다 작은 것도 무시할 수 있을 정도로 작으며, 대부분의 프리픽스 길이는 16에서 24사이에 있음을 알 수 있다.

4.1 메모리 접근 횟수

제안된 분할압축트라이의 메모리 접근 횟수는 분할된 압축트라이의 높이에 비례한다. 모든 IP 프리픽스에 대해 검색할 확률이 동일하다고 가정하면, 모든 분할된 압축 트라이가 선택될 확률 또한 동일하므로, 임의의 k개의 비트를 이용한 분할 기법으로 분할된 압축 트라이의 높이는 다음과 같다.

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

여기서 W_i는 분할된 압축 트라이 i 의 높이이고, W는 전체를 하나의 압축트라이로 구성하였을 때의 높이이다. 분할된 압축트라이는 하나의 경로압축트라 압축트라이 i 의 높이는 $(W-k)$보다 작거나 같다. 따라서 한번의 IP 주소 탐색을 위한 메모리 접근 횟

수가 k만큼 감소함을 알 수 있다.

IP 주소체계를 고려한 프리픽스 분할 기법으로 제안된 분할 압축트라이의 높이는 위의 식을 응용하여 다음과 같이 유도된다.

$$W-\left(\frac{1}{N}\sum_{i=0}^{N_A-1}(K_A+1)+\frac{1}{N}\sum_{i=0}^{N_B-1}(K_B+2)+\frac{1}{N}\sum_{i=0}^{N_C-1}(K_C+3)\right)$$
$$= W-\left(\frac{N_A(K_A+1)+N_B(K_B+2)+N_C(K_C+3)}{N}\right)$$

여기서는 클래스 A, B, C를 표시하는 각각 1, 2, 3개의 비트들이 경로압축에 포함되어 있으며 N은 $N_A + N_B + N_C$이다. 그림 5에서 보여주는 바와 같이 클래스 A, B, C의 프리픽스 수 N_A, N_B, N_C의 비율이 약 2:1:8이므로 압축트라이의 높이를 맞추기 위해 각 클래스의 분할비트 K_A, K_B, K_C는 κ-2, κ-3, κ로 할당할 수 있다. 따라서 위의 식은 다음과 같이 간략화 할 수 있다.

$$W-\frac{2N_B(x-1)+N_B(x-1)+8N_B(x+3)}{11N_B} = W-\frac{11x+21}{11} = W-(x+2)$$

여기서 k를 κ와 같도록 하면 분할된 압축 트라이 i의 높이는 $W-(x+2)$보다 적거나 같게 된다. 따라서 IP 주소 체계를 고려하여 분할하는 기법이 임의의 비트를 사용하는 기법에 비해서 같은 개수의 k비트를 분할할 경우 2만큼의 높이를 줄일 수 있어서 메모리 접근 시간을 더욱 단축시킬 수 있다.

4.2 요구되는 메모리 크기

라우터의 포워딩 테이블을 분할된 압축트라이로 구성하는데 요구되는 메모리 크기를 계산하기 위해 임의의 압축트라이 $Trie_i$ 의 메모리 크기를 $M_i^{P_k}$로 정의한다. 임의의 k개의 비트를 사용하는 기법에서 요구되는 메모리 크기는 다음과 같이 유도된다.

$$M^{Ra}+\sum_{i=0}^{N-1}M_i^{P_k}+M^{Exp} = N+\sum_{i=0}^{N-1}M_i^{P_k}+M^{Exp}$$

여기서 M^{Ra}는 루트노드를 저장하는 배열의 크기이고 M^{Exp}는 작은 프리픽스 길이로 인하여 생성되는 확장된 프리픽스를 저장하는 메모리 크기이다. 프리픽스를 확장하지 않을 경우의 메모리 크기는 전체의 포워딩 테이블을 하나의 압축트라이로 구성할 경우와 같고, n비트를 확장하는 경우에는 $M^{Exp}= 2^n$ 정도 메모리가 더 필요하게 된다.

IP 주소체계를 고려한 기법의 경우에 요구되는 메모리 크기는 다음과 같다.

$$2^{K_A}+\sum_{i=0}^{N-1}M_i^{P_{k_A}}+M_A^{Exp}+2^{K_B}+\sum_{i=0}^{N-1}M_i^{P_{k_B}}+M_B^{Exp}$$
$$= N+\sum_{i=0}^{N-1}(M_i^{P_{k_A}}+M_i^{P_{k_B}}+M_i^{P_{k_C}})+M^{Exp}+2^{K_C}+\sum_{i=0}^{N-1}M_i^{P_{k_C}}+M_C^{Exp}$$

여기서 $M_A^{Exp}, M_B^{Exp}, M_C^{Exp}$ 는 각 클래스에서 확장에 따라 증가되는 노드의 수로 모두 합하여 M^{Exp}로 표현하였으며 N은 $2^{K_A}+2^{K_B}+2^{K_C}$ 이다. 이 기법에서도 프리픽스를 확장하지 않을 경우에는 요구되는 메모리는 하나의 트라이로 구성한 경우와 같으며 확장할 경우에는 확장되는 비트의 수에 따라 약간 증가함을 알 수 있다.

5. 결론

본 논문에서는 고성능의 패킷 포워딩 엔진을 설계하는데 있어서 병목 요인이 되는 IP 주소 탐색의 속도를 향상시킬 수 있는 구조를 제안하였다. 검색 경로 압축 트라이에 기초한 분할된 압축 트라이 기법을 제시하였으며 검색 경로압축트라이의 본질적인 특성을 통해서 IP 주소 검색을 위한 메모리 접근 횟수가 감소되고 메모리 증가도 거의 없음을 보였다.

참고문헌

[1] BGP Table Statistics; http://bgp.potaroo.net

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

[5] S. Keshave and R. Rharma, "Issues and Trends in 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.