2-Level Trie를 이용한 고속 라우팅 검색

오승현°
동국대학교 컴퓨터학과

A High-Speed Routing Lookups Using 2-Level Trie

Seung-Hyun Oh°
Dept. of Computer Science, Dongguk University

요 약

라우터의 IP 주소검색은 라우터에 도착한 IP 패킷의 목적지 주소를 이용하여 적절한 출력링크를 검색하고 결정하는 것으로 고속 IP 주소검색은 초고속 라우터 개발에 필수적인 부분이다. 본 논문은 일반 PC에서도 고속의 라우팅 검색이 가능하도록 2-단계 트라이를 이용하는 트라이 기반의 IP 주소검색 자료구조를 소개한다. 2-단계 트라이는 최소 크기의 포워딩 테이블을 구축, 접근속도가 빠른 캐시 메모리에 저장함으로써 고속의 검색이 지원된다.

1. 서론

인터넷의 초고속 접고 서비스가 일반화됨에 따라 급격하게 증가하는 트래픽을 처리하기 위해 백본망의 초고속화가 진행되고 있으며, 고속 라우터 개발의 필요성 또한 증가되고 있는 실정이다. 그러나 라우터의 처리속도 증가는 백본망의 케이블 속도증가에 따라가이 어렵다. 그 이유는 IP 주소검색 알고리즘에서 O(1)의 복잡도를 얻기 힘들기 때문이다. 일반적으로 라우터는 세 가지 기능을 가지고 있다. 첫째, 라우터에 도착한 패킷의 처리순서를 결정하는 scheduling 기능이다. 둘째, 패킷을 출력단자(output port)로 전달하는 스위칭(Switching)이며, 마지막으로 세 번째 기능은 패킷의 목적지 IP 주소를 바탕으로 다음-홉(Next-hop) 주소를 결정하는 IP 주소검색기능이다.

IP 주소검색의 연구는 크게 하드웨어를 이용하는 방법[1,2]과 소프트웨어를 기반으로 하는 방법[3,4]으로 구분할 수 있다. 하드웨어를 이용하는 방법은 CAM(Content Access Memory)을 이용하는 방법[1], 주소검색용 하드웨어 로직을 구성하는 방법[2] 등이 있다. 소프트웨어를 이용하는 방법은 포워딩 테이블의 구조를 고속검색이 가능하도록 변경함으로써 IP 주소검색의 성능을 향상시키는 방법이다. 또한 최장일치검색을 일반적인 이진검색으로 변경[5]하거나 MPLS[6]와 같이 프로토콜을 변경하는 연구도 소프트웨어를 이용하는 범위에 속한다. 자료구조를 변경하는 연구의 경우 해시 테이블과 트라이 구조 등이 일반적으로 사용되었다. 참고로, 트라이는 사전과 같이 단어를 순서대로 나열하여 자료를 검색하는 형태에 적용되는 자료구조로서 대부분의 단어들의 프리픽스가 공유된다는 점을 이용하는 트리와 유사한 구조이다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2장에서는 비트-맵 트라이를 이용한 IP 주소검색에 대해 간략하게 살펴보고, 3장에서는 본 논문에서 제시한 자료구조와

관련 검색 알고리즘의 구조와 실험결과를 기술한다. 4장에서는 결론과 향후 연구방향에 대해 기술한다.

2. IP 주소검색

트라이[7]는 저장공간을 절감하기 위해 프리픽스 부분을 상호 공유하는 문자열들을 표현하는데 적합한 트리 형태이다. <프리픽스/길이> 형태인 라우팅 테이블의 프리픽스 엔트리가 비트열로 표현될 수 있고 많은 프리픽스 부분의 비트가 서로 중복되어 있기 때문에, 트라이는 라우팅 테이블을 압축하여 작고 효율적인 포워딩 테이블을 만들기에 좋은 자료구조이다. 그러나 트라이 구조는 필요한 메모리의 크기가 크므로 메모리 크기를 줄이기 위해 트라이를 깊이 방향으로 분할한다. 트라이의 분할은 메모리의 크기를 줄일 수 있게 하지만 구조가 복잡해짐으로써 표현이 어려워지고 메모리 접근회수가 증가한다. 본 논문에서는 최소한의 트라이 분할 즉, 2-단계 분할을 통해 메모리 크기와 접근회수를 적절하게 통제하고, 구축된 트라이를 비트-맵으로 압축하여 최소의 메모리 크기를 얻음으로써 고속의 캐시 검색 효과를 얻고자한다.

2-단계 트라이 구조는 트라이의 구조와 라우팅 정보 즉, 프리픽스 정보를 각각 하나의 비트로 표시하고 각 비트를 수집하여 비트맵을 만든다. 그림 1에서 검은색 노드는 루트에서 출발하여 해당 노드에 도착하는 경로와 일치하는 라우팅 프리픽스의 정보를 갖는다. 이러한 이유로 검은색 노드를 프리픽스 노드라고 명명한다. 트라이의 프리픽스 노드의 존재유무를 0과 1로 표현하고 이 정보를 가진 비트를 수집하여 프리픽스 비트맵을 구성한다. 프리픽스 비트맵에서 각 비트 1은 프리픽스 노드까지의 경로와 일치하는 프리픽스가 존재함을 의미한다. 비트맵에서 비트 1이 프리픽스의 존재를 의미하므로 비트맵의 좌측으로부터 비트 1을 카운트한 값 즉, 비트 1의 누적개수는 트라이 전체에서 프리픽스의 순서를 의미하고 프리픽스의 순서는 다음-홉 정보를 가진 해시 테이블의 인덱스가 된다.

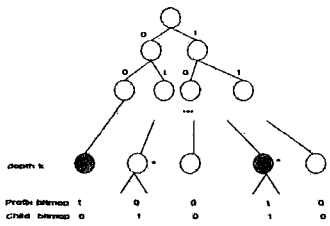

그림 1. 프리픽스 트라이와 Bitmap 구성 예제

두 번째 비트맵인 차일드 비트맵은 프리픽스 비트맵이 트라이 깊이 32에서 만들어진다면 필요 없다. 그러나 깊이 32가 아닐 경우에는 트라이의 구조를 저장하여야 하며, 이 정보가 차일드 비트-맵으로 만들어진다. 그림 1의 예제에서는 프리픽스 비트맵과 차일드 비트맵의 구성방법을 보여주고 있으며, 두 번째와 네 번째 노드가 차일드 링크를 갖고 있으므로 차일드 비트맵에서 비트 값 1이 할당됨을 확인할 수 있다. 참고로 그림 1의 트라이는 완전이진 트라이[4]로 변경된 후의 모양이다.

3. 2-단계 트라이 구조

트라이는 해시와 함께 IP 주소검색을 위해 많이 사용되는 자료구조이다. 라우팅 테이블의 IPv4 프리픽스 트라이는 최대 깊이가 32로 라우터에 도착한 IP 패킷의 목적지 주소와 일치하는 프리픽스 노드를 찾기 위해 한 비트씩 비교 검색할 경우 최대 32회의 메모리 검색이 발생하여 속도가 매우 느려지므로 트라이 레벨을 압축[3]하거나 한 번에 다수의 비트를 검색하는 방법이 연구되었다. 그러나 이러한 방법들의 문제점은 낮은 메모리 효율이다. 즉, 다수의 비트 k 비트를 한번에 비교 검색하기 위해서는 k 비트에 해당하는 트라이의 모든 정보를 저장하여야한다. 만일 k를 8로 가정하면 하나

의 IP 주소는 8 비트씩 4번의 검색을 하여야하며, 트라이의 8 비트에 해당하는 모든 정보를 저장하여야한다.

2-단계 트라이 구조는 두 번의 검색으로 IP 주소 32비트를 검색한다. 즉, 상위 트라이 구조 U-Trie는 트라이 깊이 k에서 만들어진 비트맵이며, 하위 트라이 구조 L-Trie는 $32-k$ 비트의 정보를 저장한다. 그림 3은 깊이 k에서 분할된 2-단계 트라이의 전형을 보여주고 있다. U-Trie는 깊이 k에서 만들어진 비트맵을 16비트씩 분할하여 엔트리를 구성한 배열구조의 해시 테이블이므로 크기는 2^{k-4}이다. 그림 3에 표시된 것과 같이 IP 주소의 MSB $k-4$비트가 U-Trie 테이블의 인덱스가 되고 나머지 4비트가 비트별 위치 정보가 된다. L-Trie는 $32-k$비트에 대한 정보를 갖는데, U-Trie의 노드가 깊이 k 아래로 차일드 노드를 가질 때만 만들어지며 U-Trie와 동일하게 16비트씩 분할된 비트맵을 하나의 엔트리로 갖는 테이블 구조이므로 $32-k$비트 중에서 $32-k-4$비트는 L-Trie에 대한 인덱스로, 나머지 4비트는 비트별 위치정보가 된다. L-Trie는 U-Trie와 자식-부모의 관계가 있으므로 L-Trie를 검색할 때는 이미 검색한 U-Trie의 비트의 차일드 링크에 따라 만들어진 L-Trie를 검색하여야한다. 그림 3에서 child link index로 표시된 링크를 따라 L-Trie 테이블을 선택한 뒤 $32-k-4$비트의 L-Trie 인덱스가 사용되는 예를 볼 수 있다.

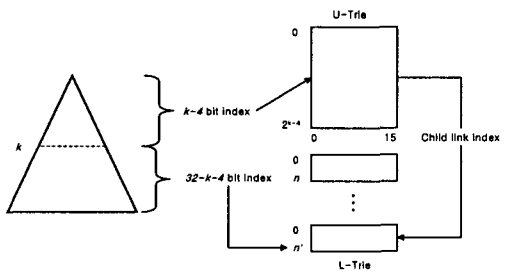

그림 2. 2-단계 트라이 구조

2-단계 트라이 구조는 적절한 k의 선택으로 최적의 메모리 크기와 검색속도를 얻기 위해 동적으로 트라이를 탐색하여 k에 따른 메모리의 크기가 L2 캐시보다 작고, U-Trie에 포함된 프리픽스 노드의 개수가 최대가 되는 k를 결정한다. 이때 프리픽스 길이 분포를 감안하여 k를 16부터 탐색하도록 하였다. k가 20일 때는 비트맵 하나의 크기가 2^{20}비트로 너무 커지므로 k의 상한선은 20으로 하였다. k가 21 이상 특히 24일 경우에는 거의 모든 주소검색이 U-Trie에서 종료되며 L-trie의 의미가 거의 없다고 볼 수 있으며, 특히 비트맵의 크기가 2^{21}비트 이상으로 커져서 L2 캐시보다 작은 메모리를 얻으려는 시도를 무의미하게 한다.

k의 선택에 따라 생성되는 L-trie의 크기가 커지는 단점이 있으므로 이를 해결하기 위해서는 일률적으로 L-Trie의 깊이를 32로 하지 않고 각각의 L-Trie에 따라 최대 깊이를 결정하여 사용하는 것이다. 예를 들어, 어떤 L-Trie의 트라이 구분 깊이가 k이고, 최대 깊이가 $maxd$라면 L-Trie의 비트-맵 크기는 2^{32-k}가 아니라 2^{maxd-k}로 변경됨을 의미한다. 예를 들어 k가 16일때 maxd가 24라면 2^{16} 비트맵이 2^8 비트맵으로 축소되어 매우 높은 메모리 절감효과를 얻게 된다.

2-단계 트라이 구조의 실험은 IPMA[9] 프로젝트의 백본 라우팅 테이블을 대상으로 하였다. 실험에 투입된 IP 주소는 랜덤으로 만들어진 주소이며, 한번의 실험에 백만 개의 주소를 투입하였고 얻어진 IP 주소검색 시간은 총 수행시간을 투입된 IP 주소의 개수로 나눈 평균값이다. 실험에 사용된 플랫폼은 펜티엄 II 450MHz CPU에 L2 캐시 크기는 512KB이다. 표 1은 라우팅 테이블별로 k의 변화에 따르는 검색성능을 보여준다. 이표에서 검색성능은 테이블의 크기와

는 무관하며 자료구조의 크기와 밀접한 관련이 있다는 것을 알 수 있다. 표 2는 실험에 사용된 라우팅 테이블의 종류와 각 테이블에서 생성된 2-단계 트라이 구조의 멤리 크기를 보여준다. k가 17과 18일 때는 모두 L2 캐시보다 작은 크기이지만 성능의 차이가 크게 나타났다. 이것은 k가 18일 때의 자료구조 크기가 L2 캐시 보다는 작지만 실제로는 이 자료구조가 저장되는데 실제 사용된 캐시의 크기보다 커져서 캐시 미스가 많이 발생했기 때문이다.

표 1. IPMA 라우팅 테이블의 평균 검색성능: 단위(ns)

k	Mae-East Lookup Time	Mae-West Lookup Time	PacBell Lookup Time	Paix Lookup Time	Aads Lookup Time
16	87	90	80	90	90
17	83	80	80	80	80
18	103	100	110	100	90
19	116	110	110	110	110
20	139	140	140	130	140

표 2. 2-단계 트라이 구조의 크기: Size(KB)

k	Mae-East	Mae-West	PacBell	Paix	Aads
	Size	Size	Size	Size	Size
16	371	320	270	178	224
17	374	322	285	208	246
18	461	409	382	318	347
19	688	639	618	565	588
20	1192	1146	1127	1081	1101

4. 결론

본 연구에서 개발한 2-단계 트라이 구조는 적절한 자료구조의 크기로 고속의 IP 주소검색을 지원할 수 있으며, 트라이를 2-단계로 분할할 때 자료구조의 크기를 동적으로 측정하여 고속의 SRAM 캐시 검색을 최대한 사용할 수 있는 최적의 분할 깊이 k를 결정할 수 있었다. 2-단계 트라이 구조의 향후 연구는 IPv6의 128비트 주소에 대해 다단계 비트맵 구조를 적용할 수 있는지를 실험하고, 최적의 단계수와 단계의 분할 깊이를 동적으로 측정하는 것이다.

[참고문헌]

[1] A.J. McAuley and P. Francis, Fast routing table lookup using CAMs, Proc. IEEE Infocom '93, San Francisco, 1993.

[2] N. Huang and S. Zhao, "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers", IEEE JSAC, Vol. 17, No. 6, Jun. 1999.

[3] S. Nilsson and G. Karlsson, "Fast Address Look-up for Internet Routers", Proc. of IEEE Broadband Communications 98, Apr. 1998.

[4] M. Degermark, A. Brodnik, S. Carlsson and S. Pink, "Small Forwarding Tables for Fast Routing Lookups," Proc. of ACM SIGCOMM'97, Oct. 1997.

[5] B. Lampson, V. Srinivasan and G. Varghese, "IP Lookups using Multiway and Multicolumn Search", Proc. of INFOCOM', Mar. 1998.

[6] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, IETF, Jan. 2001.

[7] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, "Data Structure and Algorithms," *Addison-Wesley*, 1983

[8] S. Venkatachary and G. Varghese, "Faster IP Lookups using Controlled Prefix Expansion," Proc. of ACM Sigmetrics, Sep. 1998.

[9] Michigan University and merit Network, Internet Performance Management and Analysis (IPMA) project, http://nic.merit.edu/~ipma.