IP 주소 검색을 위한 Priority Trie

임혜숙*, 문주형
이화여자대학교 정보통신학과

An Efficient IP address Lookup Algorithm Using a Priority-Trie

Hyesook Lim*, Ju Hyoung Mun
Information Electronics Engineering Department
Ewha Womans University
E-mail : *hlim@ewha.ac.kr

Abstract

Fast IP address lookup in routers is essential to achieve packet forwarding in wire-speed. The longest prefix matching for IP address lookup is more complex than exact matching because of its dual dimensions, length and value. By thoroughly studying the current proposals for IP address lookup, we find out that the binary search could be a low-cost solution while providing high performance. Most of the existing binary search algorithms based on trie have simple data structures which can be easily implemented, but they have some problems because of empty internal nodes. The proposed algorithm is based on trie structure, but empty internal nodes are replaced by priority prefixes. The best-matching-prefix search in the proposed algorithm is more efficiently performed since search can be finished earlier when input is matched with a priority prefix. The performance evaluation results show that the constructed priority-trie has very good performance in the lookup speed and the scalability.

I. 서론

인터넷 라우터에서의 IP 주소 검색은 패킷 포워딩을 위한 가장 기본적이면서도 실시간으로 처리되어야 하는 매우 중요한 기능으로서 효율적인 주소 검색 알고리듬에 관한 연구가 활발히 진행되어 오고 있다. [1] 현재의 CIDR (Classless inter-domain routing) 구조에서 프리픽스의 길이는 임의의 값을 가질 수 있기 때문에, IP 주소 검색은 들어온 패킷과 가장 길게 매치하는 프리픽스 (longest matching prefix)를 찾아야 한다. [2] 본 논문에서는 IP 주소 검색을 위한 프라이어티-트라이 구조를 제안한다. 제안하는 구조는 트라이 구조를 바탕으로 하되 트라이에서의 빈 노드를 제거한 구조로서, 검색 속도 및 라우팅 테이블을 저장하기 위해 필요로하는 메모리 사용량 등에서 매우 좋은 성능을 보여준다.

II. 본론

이진-트라이에서, 짧은 길이의 프리픽스들은 트라이의 상위 레벨에 저장된다. 따라서 가장 길게 매치하는 프리픽스를 찾기 위해서는 리프 노드를 만날 때까지 검색을 진행해야 한다. 긴 프리픽스를 트라이의 상위 레벨에 저장한다면, 매치하는 노드를 만났을 때 바로 검색을 종료할 수 있다. 그러나 프리픽스가 자신의 길이 i 보다 하위 레벨인 j 레벨에 저장되기 위해서는 2^{j-i} 개로 복사 되어야만 한다.

본 논문은 긴 프리픽스를 상위 레벨에 저장하기 위해, 트라이의 빈 노드를 사용할 것을 제안한다. 빈 노드에 그 빈 노드를 루트로 하는 서브-트리에서 가장 긴 프리픽스를 저장하면, 트라이의 빈 노드를 완전히 제거할 수 있다. 따라서 트라이의 깊이도 짧아지고, 검색 또한 효율적으로 수행할 수 있다. 자기를 루트로 하는 가장 긴 프리픽스로 대체된 빈 노드를 프라이어리티 노드로 정의한다. 제안하는 구조에서의 모든 프리픽스는 원래의 레벨보다 높거나 같은 레벨에 위치하게 되므로, 프리픽스가 여러 개로 복사되지 않는다. 그림 1 과 2 는 동일한 프리픽스에 대하여 이진-트라이와 제안하는 프라이어리티-트라이를 나타낸 것이다. 검게 칠해진 노드는 원래 위치에 저장된 노드를 나타내고, 굵은 선으로 표시된 노드는 프라이어리티 노드를 나타낸다. 프리픽스 P4 는 루트 노드에 속한 가장 긴 길이를 갖는 프리픽스이기 때문에 프라이어리티 노드로서 루트노드에 저장되었다. 빈 노드 0*의 경우는 자신의 서브 트리에 속

한 가장 긴 프리픽스 중 가장 왼쪽 프리픽스를 프라이어리티 노드로 이동하였다.

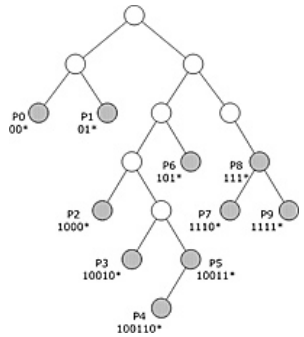

그림 1 The example set of prefixes

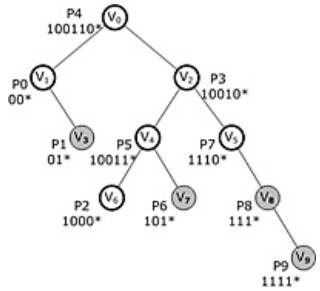

그림 2 The proposed priority-trie

2.1 빌드

제안하는 구조를 빌드하는 방법은 다음과 같다. 먼저 이진-트라이를 만들고, 이진-트라이의 가장 긴 프리픽스중 제일 왼쪽에 위치한 프리픽스로부터 하나씩 제거 한 후, 제거된 프리픽스의 원래 노드까지의 경로에서 처음 만나게 되는 빈 노드에 제거된 프리픽스를 프라이어티 노드로서 저장한다. 이와 같은 과정을 이진-트라이에 빈 노드가 없어질 때까지 반복한다 002E

2.2 검C0C9

이진 트라이에서와 동일하게 입력된 주소의 한 비트씩 검사하여 0 이면 왼쪽으로, 1 이면 오른쪽으로 검색을 진행한다. 각 노드에서 입력된 주소를 노드에 저장되어 있는 프리픽스의 길이까지만을 비교하여 일치하는 경우 매치로서 정의한다. 일반노드와 매치하는 경우 현재까지의 가장 긴 매치로서 기억하고 검색을 계속 진행한다. 프라이어리티 노드와 매치하거나, 리프를 만나면 검색을 종료한다.

Ⅲ. 실험 결과

여러가지 크기의 실제 라우팅 테이블에 대한 실험 결과를 Table I, II에 보였다. Table I은 이진 프라이어리티-트라이를 구성한 경우이며, Table II는 다중 프라이어리티-트라이를 구성한 경우이다. 타 구조와의 성능 비교는 Table III 에 보였다. 테이블에서 N 은 라우팅 테이블의 크기, N_p는 프라이어티 노드의 개수, D_p는 프라이어티 트라이의 깊이, T_A 는 평균 메모리 접근 횟수, T_{max} 는 최대 메모리 접근 횟수, N_{extra} 는 별도로 요구되는 노드의 개수, M은 요구되는 메모리의 크기이다.

TABLE I. Performance of the Proposed Priority-binary Trie

Routing Table	N	N_p	D_p	T_A	M (Kbyte)
MAE-West1	14,553	14,199	24	16.66	127.9
Aads	20,204	19,568	24	17.43	177.6
MAE-West2	29,584	26,671	24	18.22	260.0
PORT 80	112,310	50,091	28	20.35	987.1
Grouptlcom	170,601	70,525	24	20.76	1.46M
Telstra	227,223	119,149	32	22.86	1.95M

TABLE II. Performance of the Proposed Priority-Multibit Trie

Routing Table	N	N_p	N_{extra}	D_p	T_A	M (Kbyte)
MAE-West1	14,553	14,388	2,337	12	9.70	214.4
Aads	20,204	19,124	3,208	12	10.05	297.2
MAE-West2	29,584	23,016	6,273	13	10.54	455.2
PORT 80	112,310	36,362	21,630	16	11.27	1.79M
Grouptlcom	170,601	47,073	32,732	13	11.45	2.71M
Telstra	227,223	75,929	41,850	16	12.51	3.85M

참고문헌

[1] H. Jonathan Chao, “Next generation routers,” Proceedings of the IEEE, vol. 90, no. 9, pp. 1518-1558, Sep. 2002.

[2] M.A. Ruiz-Sanchex, E.W. Biersack, and W. Dabbous, “Survey and taxonomy of IP address lookup algorithms,” IEEE Network, pp.8-23, March/April 2001.

[3] George Varghese, “Network algorithmics: An interdiciplinary approach to designing fast networked devices,” Morgan Kaufmann Publishers, Elsevier Inc, 2005

TABLE III. Comparison with other Algorithms

Algorithm	incremental update	Port80 (112,310)				Telstra (227,223)			
		T_{max}	T_A	M (MB)	N_{Extra}	T_{max}	T_A	M (MB)	N_{Extra}
Binary trie [2]	yes	32	22.15	1.29	112,907	32	24.64	2.59	225,682
BPT [2]	no	44	25.82	1.25	0	66	30.80	2.60	0
WPT	no	36	20.44	1.25	0	39	23.96	2.60	0
BSR	no	18	11.42	0.96	72,063	19	11.07	1.76	124,795
Proposed binary	yes	28	20.35	0.99	0	32	22.86	1.95	0
Proposed multi-bit	yes	16	11.27	1.79	21,630	16	12.51	3.85	41,850