LMHTL 기법에 의한 고속 IP주소 검색

오 승 현[0]
동국 대학교 컴퓨터학과
shoh[0]@dongguk.ac.kr

High-speed IP address lookup using LMHTL technique

Seung-Hyun Oh[0]
Dept. of Computer Science, Dongguk University

요 약

본 연구는 포워딩 테이블의 자료구조와 검색 알고리즘을 개선하여 고속 IP주소 검색을 구현하는 소프트웨어 기반의 연구로써 목적지 IP주소에서 검색할 길이를 결정한 후 길이별로 준비된 해시 테이블 검색을 통해 O(1)의 속도로 IP주소 검색을 실시할 수 있다. IP주소 검색은 목적지 IP주소와 다수의 포워딩 테이블 엔트리 중에서 가장 많은 비트가 일치하는 엔트리를 검색하는 과정으로 일반적인 완전일치 검색이 적용되기 어렵다. 본 연구에서는 포워딩 테이블 엔트리로 구성된 트라이를 기반으로 길이별 다중 해시 테이블을 구성하여 평균 $O(\log_2 N)$,N=4 의 속도로 IP주소 검색을 한다. 이때 최악의 검색시간은 4회의 주 메모리 접근시간이며 더 빠른 검색을 위해 본 논문은 각 프리픽스의 첫 8비트를 키로 256개의 그룹을 만들고 각 그룹별 최대 프리픽스 길이를 기록한 테이블을 캐쉬에 저장함으로써 길이별 해시 테이블 검색 시 N값을 99.9%의 확률로 3이하로 제한할 수 있다. 이것은 포워딩 테이블의 프리픽스 길이 분포에 의한 결과이며, 99.9%의 확률에서 최악의 검색시간을 3회의 주 메모리 접근시간으로 할 수 있다. 주 메모리 접근시간 50ns를 적용하면 150ns의 검색속도는 512B 패킷을 가정할 때 약 27Gbps의 검색속도를 지원할 수 있다.

1. 서론

최근의 인터넷 환경은 초고속 인터넷 접속환경의 급격한 보급과 폭증하는 사용자와 정보에 따라 백본의 대용량화가 빠르게 진행되고 있다. 이에 발맞추어 백본의 대용량 고속화에 필수적인 라우터(router)가 Mbps급에서 Gbps급으로 변화되고 있다. 그러나 하드웨어 기반의 스위치(switch)와는 다르게 IP주소 검색이 필요한 라우터의 성능은 그다지 빠른 진보를 보이지 못하는 실정이다. 이것은 IP주소 검색에 최장 프리픽스 일치(longest prefix matching)검색 조건을 만족하여야 하는 점이 라우터 성능에서 일종의 병목(bottleneck)으로 작용하기 때문이다. 라우터의 고속화 연구는 다양한 방법으로 시도되어 왔는데 소프트웨어 기반의 연구는 저렴한 비용과 손쉬운 구현성 때문에 많은 관심을 끌어왔다. 본 연구도 포워딩 테이블의 자료구조와 검색 알고리즘을 새로이 구현함으로써 고속의 검색속도를 얻는 소프트웨어 기반의 연구이다.

기존의 소프트웨어 기반의 연구는 프리픽스 엔트리를 트라이(trie)로 구현하여 비트별 검색을 실시하는 연구들[1]과 해시 테이블을 이용한 연구[2], 프리픽스 길이의 확장을 통한 이진 검색의 적용 연구[3] 등이 있다. 참고로 소프트웨어 기반의 연구 이외의 IP검색 연구분류는 고속의 메모리용 하드웨어를 적용하는 하드웨어 기반의 연구[4]와 IP검색 부담을 경감하거나 제거할 수 있는 새 네트워크 계층 및 이하의 계층에 적용 가능한 프로토콜을 개발하는 프로토콜 기반의 연구로 정리할 수 있다.

본 논문의 LMHTL(Length-based Multiple Hash Table Lookup) 방법은 프리픽스의 길이에 따라 별도로 구성된 다수의 해서 테이블 검색방법이다. 해시 테이블은 8, 16, 24 및 32-비트 길이의 4가지 종류로 구성된다. 이때 해시 테이블의 크기와 버켓(bucket) 다중적재(overload)를 최소화하기 위해서 해시 테이블은 다시 프리픽스 그룹별로 나누어서 별도로 구성된다. 예를 들어 12.0*/16과 24.108*/16(편의를 위해 점십진(dot-decimal) 표기법에 프리픽스의 길이 표시를 첨가한 프리픽스 표기임) 프리픽스에 대해 16-비트 길이의 해시 테이블은 첫 8-비트 값 12와 24가 다르므로 각각 12와 24를 위한 다른 해시 테이블에 저장된다. 물론 IP주소 검색 시에도 첫 8-비트 값에 따라 적절한 해시 테이블이 검색되어야한다.

최장 프리픽스 일치 검색조건을 만족시키는 LMHTL의 최악의 검색속도는 최대 4회의 주 메모리 접근속도이다. 이것은 해시 테이블이 8, 16, 24 및 32-비트 길이 단위로 만들어지기 때문에 최악의 경우 모든 길이의 해시 테이블을 검색하여야하기 때문이다. 물론 백본 포워딩 테이블의 프리픽스 길이별 분포[5]를 살펴보면 8~32-비트 길이 분포를 보이지만 이 같은 분포를 그대로 적용할 경우 최악의 검색시간이 24회의 주 메모리 접근시간으로 악화되므로 채택할 수 없다. 반대로 2 단계 또는 1 단계의 해시 테이블 구성은 일반적인 주 메모리 크기로 감당하기 어려운 너무 큰 저장공간을 요구하므로 현실성이 없다. 따라서 LMHTL에서는 메모리 크기와 검색속도의 상호관계를 고려하여 4단계의 해시 테이블 길이 단위를 채택하였다. 이때 포워딩 테이블의 프리픽스들이 모두 8의 배수 길이가 아니므로 프리픽스 확장이 필요하다. 프리픽스 확장은 해시 테이블 구성 길이 단위가 커지면 지수적으로 확장의 크기가 증가하므로 적절한 크기에서 제한할 필요가 있다. 본 연구에서는 임의로 8의 배수 단위를 사용하였다. 최적의 길이 단위를 구하는 문제는 차후의 연구로 미뤄진 상태이다.

본 연구의 LMHTL 기법은 해시 테이블을 저장하기 위해 대량의 주 메모리를 사용하는 대신 CPU 및 캐쉬에 비해 매우 느린 주 메모리의 접근 회수를 감소시키는 것이 기본적인 목표이다. 기존의 소프트웨어 기반 연구들이 포워딩 테이블의 크기를 시스템의 캐쉬 크기 이하로 압축하여 빠른 캐쉬 접근속도를 기반으로 다소 복잡하고 반복적인 캐쉬 접근을 시도하는 방법과는 다른 방법이라고 할 수 있다. 2장에서는 LMHTL의 구조에 대해 기술하고 3장에서는 검색 알고리즘에 대해 설명한다. 4장에서는 결론과 향후 연구과제에 대해 기술한다.

2. LMHTL 구조

LMHTL 기법은 두 가지 종류의 테이블을 사용한다. 첫 번째

테이블은 4단계의 프리픽스 길이별로 구성된 다중 해시 테이블이다. 해시 테이블은 주 메모리에 저장된다. 두 번째 테이블은 캐쉬에 저장되는 그룹별 프리픽스 길이를 기록한 GLT(Group Length Table)이다. 두 가지 종류의 테이블을 구성하기 위해 포워딩 테이블의 프리픽스를 이용한 트라이가 사용된다.

2.1 트라이

트라이[6]는 포워딩 테이블 엔트리인 프리픽스를 비트열(bit stream)로 사용하여 구성되는 이진 트리와 유사한 트리이다. 즉, 비트 값 0을 좌측 가지(edge)로, 1을 우측 가지로 사용하여 이진 트리를 구성하고, 프리픽스의 마지막 비트 값에 해당하는 노드(node)에 프리픽스 정보를 저장하는 트리이다. 또한 한 노드의 차일드 노드가 좌측 또는 우측에만 존재한다면 비어있는 방향에도 차일드 노드를 형성하는 즉, 모든 노드에는 차일드 노드가 없거나 두 개가 있는 완전 이진 트라이 확장을 한다. 완전 이진 트라이 확장은 [6]에서 자세히 참고할 수 있다. 본 연구에서 사용하는 트라이는 비트열의 길이가 8의 배수가 되는 지점 즉, 트라이의 깊이가 8, 16, 24 및 32가 되는 부분에 프리픽스 정보가 저장되도록 변형된 트라이이다. 포워딩 테이블의 프리픽스 엔트리들이 모두 8의 배수 길이가 아니므로 프리픽스 확장이 필요하다. 즉, 프리픽스의 길이가 8인 경우는 확장 없이 사용되고, 프리픽스의 길이가 14라면 길이 16에 프리픽스 정보를 저장할 수 있도록 2비트의 프리픽스 확장이 필요하며 2^2개의 프리픽스가 추가로 발생한다. 그러나 포워딩 테이블의 포워딩 정보를 훼손하지 않도록 확장된 2^2개의 프리픽스는 모두 확장이전의 프리픽스가 가진 출력 링크정보를 갖는다. 자세한 트라이 구성방법 및 정보는 [6]에서 찾을 수 있다. 참고로 이러한 프리픽스 확장은 트라이를 만들 때 실제로 시행되지는 않고 나중에 해시 테이블을 구성할 때 논리적으로 처리된다.

구성이 완료된 트라이는 해시 테이블과 GLT를 구성하기 위한 기준으로 사용된다. 트라이에서 추출되어야할 첫 번째 정보는 트라이의 8의 배수의 깊이 즉, 8, 16, 24 및 32의 노드에 저장된 프리픽스 정보이다. 각 깊이에서 수집된 프리픽스를 길이를 기준으로 해시 테이블에 저장하면 4단계의 길이별 다중 해시 테이블이 구성된다. 트라이에서 얻을 두 번째 정보는 프리픽스의 MSB를 기준으로 첫 번째 8-비트 값이 같은 프리픽스 중에서 최대 프리픽스 길이를 구하는 것이다. 최대 프리픽스 길이는 다중 해시 테이블의 검색시간을 단축하기 위한 최적화 기법에 사용된다.

2.2 다중 해시 테이블

트라이에서 4단계 길이를 기준으로 구성된 4개의 해시 테이블의 크기는 2^{16}개의 엔트리를 가진 배열로 구성된다. 각 엔트리의 길이를 4-바이트의 프리픽스와 1-바이트의 출력 링크 정보로 구성하면 약 328KB이므로 총 4개의 테이블에 대해 1.3MB 크기가 된다. 이 크기는 주 메모리에 저장하는데 전혀 문제가 되지 않는 매우 작은 크기이다. 하지만 프리픽스의 길이별 분포를 감안할 때 4개 테이블의 크기가 모두 같다는 것은 하나의 문제를 일으킬 수 있다. 즉 해시 테이블의 버켓에 하나 이상의 프리픽스가 적재되는 다중적재(overloading)가 불균등하게 발생할 수 있다는 것이다. 그림 1은 4개의 해시 테이블에 Mae-East[7] 포워딩 테이블을 저장했을 때 발생한 버켓별 다중적재 분포를 정리한 것이다. 그림 1은 해시 테이블의 크기를 2^8~2^{16}까지 변경하면서 발생하는 최대 다중적재 개수를 표시하고 있다. 해시 테이블에서 높은 다중적재 발생은 해시 테이블의 일반적인 접근속도인 O(1)의 속도를 악화시킨다. 즉 모든 버켓에서 다중적재가 발생하지 않아야 O(1)의 속도를 제공할 수 있으므로 가능한 한 다중적재를 최소화할 필요가 있다.

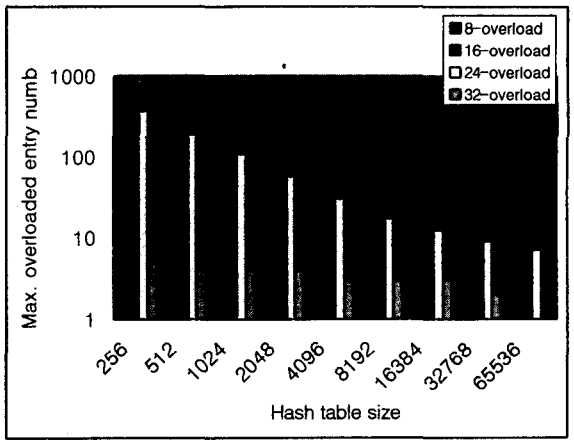

그림 1 4개의 해시 테이블에 대한 다중적재 분포

이 같은 다중적재 현상을 개선하기 위한 방법의 하나는 해시 테이블에 저장될 엔트리 즉, 프리픽스의 개수를 감소시키는 것이다. 이것은 프리픽스의 개수가 변함이 없으므로 반대로 해시 테이블의 개수를 증가시킴으로써 해시 테이블에 저장될 프리픽스를 다수의 해시 테이블로 분산시킬 수 있다. 다만 프리픽스 길이별 분포를 감안할 때 길이 8과 32의 경우는 프리픽스의 개수가 매우 작으므로 해시 테이블을 분리할 필요가 없다고 판단되므로 본 연구에서는 길이 16과 24의 경우만 해시 테이블 분리 대상으로 하였다.

해시 테이블 분리는 깊이 16과 24에 형성된 프리픽스 정보 중에서 프리픽스의 MSB 기준 첫 번째 8-비트 값이 동일한 프리픽스들을 하나의 해시 테이블로 분리 구성한다. 이와 같은 해시 테이블 분리는 16과 24 깊이에 각각 256개의 해시 테이블을 만든다. 즉, 프리픽스의 첫 8-비트 값을 키로 해시 테이블이 분리되고, IP주소 검색 시에도 주어진 IP주소의 첫 8-비트 값을 키로 검색할 해시 테이블이 결정된다.

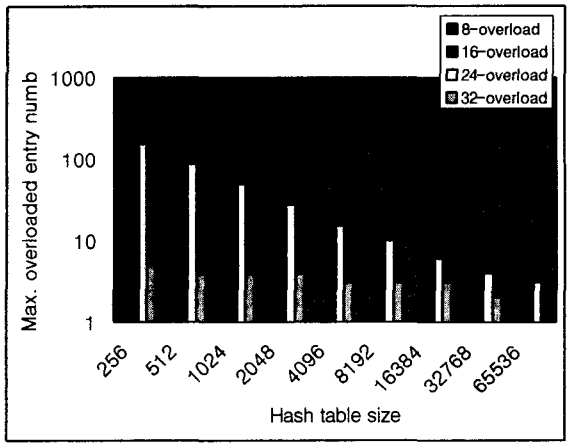

그림 2 배열 크기에 따른 해시 버켓 다중적재 분포도

분리된 해시 테이블의 개수는 깊이 8에서 1개, 16과 24에서 각각 256개, 마지막으로 32에서 1개로 총 514개이다. 해시 테이블을 2^{16}개의 엔트리를 가진 배열로 만들고, 엔트리 길이 5-바이트를 적용하면 필요한 총 메모리 크기는 약 168MB 이다.

메모리 크기를 줄이기 위해 배열의 크기를 2^{15}개로 축소하면 총 메모리 크기는 약 84MB 이다. 이러한 배열 크기의 선택은 해시 버켓의 다중적재 분포도와 메모리 사용가능 정도이다. 그림 2에서 이러한 배열의 크기가 변화할 때 그룹별로 분리된 다중 해시 테이블에 발생한 Mae-East 포워딩 테이블의 다중적재 분포도를 보여주고 있다. 본 연구에서 사용한 해시 함수는 XOR(exclusive-OR) 연산을 이용한다. XOR 해시 함수는 [8]에서 보고한 바와 같이 연산결과 값의 분포도가 양호하며 동시에 매우 손쉽게 구현할 수 있으므로 본 연구에서 채택하였다.

2.3 GLT

트라이의 4가지 길이 분류에 따라 구성된 해시 테이블로 IP 주소를 검색할 때 우리는 주어진 IP주소와 일치하는 최장 프리픽스의 길이를 미리 알 수 없다. 따라서 4가지 종류의 길이 모두에 대해 검색을 해야 한다. 다만 이때 이진검색 기법을 적용하면 평균 $O(\log_2 n)$,n=4 의 속도를 얻을 수 있다. 그러나 최악의 검색시간은 4번의 주 메모리 검색이 된다. 프리픽스 길이 분포를 살펴보면 프리픽스의 길이가 32인 경우는 매우 적은 숫자에 불과하며, 대부분의 프리픽스가 16 또는 24-비트 길이를 중심으로 분포되어있다. 또한 본 연구에서는 트라이를 이용하여 모든 프리픽스의 길이를 4가지로 단순화하였음을 상기할 때 대다수의 프리픽스가 16이나 24 길이에 모임을 알 수 있다.

이러한 점을 이용하여 우리는 LMHTL 기법의 최악 검색시간을 단축할 수 있다. 즉, 최악의 경우 4가지 해시 테이블 종류를 모두 검색하지 않고 주어진 IP주소의 최대 프리픽스 길이 이하에 대해서만 검색을 하는 것이다. 이때 주어진 IP주소의 최대 프리픽스 길이는 트라이에서 4가지 깊이에서 프리픽스 정보를 수집하여 해시 테이블을 구성할 때 프리픽스의 첫 8-비트 값에 대해 수집한다. 즉, 총 256개의 8-비트 값으로 대표되는 프리픽스 그룹의 최대 프리픽스 길이를 테이블에 기록하는 것이다. 이 테이블이 GLT(Group Length Table)이다.

GLT는 IP주소의 첫 8-비트를 키로 구성되는 각 그룹의 최대 프리픽스 길이를 저장함으로써 주어진 IP주소 검색 시에 검색할 해시 테이블의 범위를 최소한으로 제한한다. GLT는 각 해시 테이블의 메모리 주소를 포함한다. GLT의 한 엔트리 당 최대 해시 주소 개수는 4개의 해시 길이 단위에 대해 하나씩의 주소가 필요하므로 총 4개, 16-바이트가 필요하다. GLT의 크기는 4.3KB로 매우 작고 캐쉬에 저장된다. IP주소 검색 시 GLT 검색에 필요한 1회의 캐쉬 읽기 시간이 사용된다. 결과적으로 프리픽스 길이 분포 상 LMHTL 기법의 최악 검색시간은 99.9%의 확률로 주 메모리 접근 3회가 된다.

3. LMHTL 검색 알고리즘

길이별 다중 해시 테이블에 대한 IP주소 검색은 IP주소의 첫 8-비트 값을 키로 GLT를 검색하여 해당 그룹의 최대 프리픽스 길이를 구하는 것으로부터 시작한다. x.y.z.k를 검색대상 IP주소로 할 때 x가 대표하는 프리픽스 그룹의 최대 길이를 구하기 위해 x를 GLT의 인덱스로 사용하여 x 그룹의 길이를 구한다. 이때 길이는 8, 16, 24, 32 중의 하나이다. IP주소 검색의 두 번째 단계는 결정된 최대 길이를 최대 범위 즉 $\log_2 n$의 n값으로 설정하여 이진검색의 순서로 검색할 해시 테이블의 길이를 검색한다. 참고로 이때 사용되는 해시 함수는 XOR 연산을 이용한 함수이다.

이진검색 순서로 결정된 검색 대상 길이에 맞는 해시 테이블을 검색하고 적합한 프리픽스를 찾은 경우에는 검색이 종료된다. 만일 적합한 프리픽스를 차지 못한 경우에는 이진검색 순서에 따라 선택된 다음 길이의 해시 테이블을 검색한다. 이때 주의할 사항은 최장 길이 일치검색 원칙을 만족시키기 위해서 만일 검색된 프리픽스가 최대 길이에서 검색된 경우가 아닌 중간길이에서 검색된 경우 더 긴 길이의 프리픽스가 존재하는지 반드시 추가 검색이 실시되어야한다. LMHTL의 최악 검색시간은 캐쉬와 주 메모리 접근시간이 각각 10ns와 50ns일 때 1회의 캐쉬 읽기, 3회의 주 메모리 검색으로 160ns가 된다. 평균 검색 시간은 10ns+$\log_2 n$(n=3) 즉, 80ns이다. 그림 3은 LMHTL 기법의 검색 알고리즘을 간략하게 정리한 것이다.

1. IP주소 x.y.z.k에서 첫 8-비트 x값을 추출
2. 최대 프리픽스 길이 MaxL=GLT[x].length
3. 해시 테이블 주소[]=GLT[x].addr_hash[]
4. 이진검색 순서에 의한 첫 검색 대상 길이 선정
5. 해시 테이블 인덱스=hash function(IP주소)에 의해 해시 검색
6. 검색 성공이면, 2에서 설정된 최대 길이에서 검색이 실시되었는지 확인하여 만일 실시되지 않았으면 이진검색에서 더 긴 길이 해시 검색을 위해 low bound 값 조정 후 4항으로 이동
7. 6항에서 더 긴 길이에서 검색이 실시된 후면 검색종료
8. 5항 검색이 실패면 더 짧은 길이 해시 검색을 위해 upper bound 조정 후 4항으로 이동
9. 모든 이진검색 후 검색된 프리픽스가 없으면 IP주소 검색을 실패처리

알고리즘 1 LMHTL 검색 알고리즘

4. 결론 및 향후 연구과제

본 연구는 프리픽스의 길이를 키로 다중 해시 테이블을 구성하여 해시 버켓의 다중적재를 최소화하면서 평균 80ns의 고속 검색속도를 달성하고 있다. 해시 테이블은 O(1)의 빠른 검색이 가능하지만 버켓 당 저장되는 엔트리의 개수가 잘 분포되어야 한다. 그렇지 않고 다중적재가 많이 발생할 경우 버켓 내부에서 최악의 경우 순차검색이 추가로 발생할 수 있다. 본 연구는 이러한 다중적재의 가능성을 최소화하기 위해서 프리픽스를 그룹과 프리픽스 길이로 분류하여 해시 테이블을 그룹 및 길이별로 복합적·다중적으로 생성함으로써 다중적재률을 낮추었다. 향후의 연구는 프리픽스 길이의 분류를 단순한 8의 배수가 아닌 최적화된 길이 그룹을 구할 것이다. 이것은 프리픽스 확장 크기를 최적화하고 결과적으로 해시 다중적재를 최소화하는데 기여할 것이다. 또 다른 과제는 해시함수의 최적화이다. XOR 연산을 이용한 해시함수보다 더 분포도가 우수하고 메모리 효율이 좋은 해시함수를 찾는 것은 중요한 과제이다.

참고문헌

[1]Donald R. Morrison, "PATRICIA - Practical Algorithm to Retrieve Information Coded In Alfanumeric," *journal of the ACM*, 15(4):514-534, October 1968

[2]Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner, "Scalable High Speed IP Routing Lookups," *In Proceedings of ACM SIGCOMM'97*, October 1997

[3]S. Venkatachary and G. Varghese, "Faster IP Lookups using Controlled Prefix Expansion," *In Proceedings of ACM Sigmetrics'98*, June 1998

[4]A. Moestedt, et al., "IP Address Lookup in Hardware for High-Speed Routing," *Hot Interconnects*, August 1998

[5]P. Gupta, et al., "Routing Lookups in Hardware at Memory Access Speeds," *In Proceedings of IEEE Infocom'98*, San Francisco, April 1998

[6]오승현, 안종석, "고속 라우터의 기가비트 포워딩 검색을 위한 비트-맵 트라이 구조", 정보과학회 논문지:정보통신, 2001

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

[8] R. Jain, "A Comparison of Hashing Schemes for Address Lookup in Computer Networks", IEEE Transactions on Communications, vol. 40, no. 10, pp.1570-1573, 1992