고속 IP 주소 검색을 위한 다중 LC-트라이

비회원 황 현 숙*, 정회원 권 택 근*

Multiple LC-tries for Fast IP Address Lookup

Hyun-Sook Hwang* *Non-Member*, Taeck-Geun Kwon* *Regular Member*

요 약

IP 라우팅에서는 가장 긴 프리픽스가 일치 (longest prefix matching)되는 목적지 IP 주소를 사용한다. 따라서 고속 IP 라우팅을 위하여 IP 주소 검색이 고속으로 수행되어야 한다. 본 논문에서는 소프트웨어 기반의 유연한 IP 주소 검색을 효과적으로 수행하는 LC-트라이 알고리즘을 개선한 다중 LC-트라이 기법을 제안한다. 메모리 검색 회수를 줄이기 위하여 검색되는 IP 주소의 분포 패턴을 분석하여 자주 접근되는 IP 주소의 집합으로 별도의 LC-트라이를 구성함으로써 기존의 단일 LC-트라이 기법에 비하여 빠른 검색 결과를 실험을 통해 확인한다.

ABSTRACT

The IP routing uses the longest-matching prefix to determine the destination. Fast lookup should be required for the high speed routing. We propose a modified LC-trie, called multiple LC-trie, which is suitable data structure for fast address-lookups in software implementation. To reduce the number of memory accesses, our scheme analyzes the distribution of IP address access pattern, and constructs multiple LC-tries for frequently accessed IP addresses. Our experimental results show that our scheme can perform faster than the original LC-trie schemes.

I. 서 론

90년대 중반, WWW (World Wide Web)의 등장으로 인터넷 트래픽이 급격하게 증가되었고, 최근에는 비디오와 오디오 등 대용량, 실시간 미디어를 인터넷을 통해 전송하는 응용이 널리 보급되고 있다. 그리고 광 전송 기술을 사용한 WDM (Wavelength Division Multiplexing) 등 기술의 발전으로 물리적 데이터 전송 속도는 수십 Gbps로 급증하고 있는 실정이다.

이러한 인터넷의 증가는 라우터의 고속화를 촉진하여 최근에는 하드웨어를 이용한 고속 라우터가 등장하게 되었다. 이러한 고속 라우터는 인터넷 백본망을 구축하고 있고, 이러한 추세는 당분간 지속될 전망이다. 기존의 라우터에서는 소프트웨어를 통하여 IP 패킷을 처리하였으나, 고속 라우터는 IP 패킷을 하드웨어로 처리함으로써 물리적 전송 속도에 맞는 와이어 속도 (wire-speed)의 패킷 포워딩이 가능하다[1][2].

그러나 IP 주소는 하드웨어를 통한 고속 처리가 쉽지 않은데, 이러한 문제를 극복하기 위한 다양한 해결책이 제시되고 있다.

첫째, 네트워크 레벨의 IP 주소를 이용하지 않고 데이터 링크 레벨의 작고, 효율적인 식별자를 사용하는 방법이 있다. 이러한 방법은 ATM (Asynchronous Transfer Mode)나 MPLS (Multi-Protocol Label Switching) 등의 새로운 통신 기술을 적용하는 것으로서 보장형, 실시간 데이터 통신과 같은 새로운 트래픽 요구 사항을 수용하는 장점을 갖는다. 그러나 이 경우에 모든 라우터나 스위치를 대체하는 등의 통신 인프라를 새로이 구축해야 하는 부담이 따르므로, 최근의 급격한 인터넷 요구에 부응하

* 충남대학교 컴퓨터공학과 ({hshwang,tgkwon}@ce.cnu.ac.kr)
논문번호 : 010059-0410, 접수일자 : 2001년 4월 10일

지 못하는 점이 있다.

둘째, 새로운 IP 주소 검색 알고리즘을 하드웨어에서 처리하기 적합하도록 개발하여 기존의 IP 트래픽을 그대로 수용하되 처리 속도만을 향상시키는 방법이 있다. 이러한 방법은 인터넷에서 QoS (Quality of Service)를 만족시키기 어려운 문제점이 있으나, 기존의 인터넷 망을 활용할 수 있고 오버엔지니어링 (over-engineering)을 통하여 트래픽 병목을 어느정도 피할 수 있다[3].

새로운 IP 주소 검색 알고리즘을 통한 고속 라우터를 구현하는 방법으로는 일반적으로 자주 접근되는 IP 주소를 CAM (Content Addressable Memory)나 고속 캐쉬 등의 고속 메모리에 저장하는 방법이 사용된다[4]. 최근에는 고속 IP 주소 검색 하드웨어를 프로세서 내부에 포함시켜 보다 유연한 방법으로 IP 패킷을 처리하도록 하는 네트워크 프로세서를 라우터에 도입하려는 시도도 활발하다[5]. 이러한 네트워크 프로세서의 접근 방법은 하드웨어를 통한 IP 패킷 처리가 갖지 못하는 유연함 (flexibility)을 갖기 위해서 이다.

본 논문은 하드웨어에 의한 고속 IP 패킷 라우팅의 구현상 복잡성과 다양한 서비스를 유연하게 지원하지 못하는 단점을 보완하기 위하여 소프트웨어 기반의 고속 IP 패킷 라우팅을 목표로 하고 있다. 따라서 본 논문에서는 IP 패킷 처리의 유연함을 최대한 제공하기 위하여 소프트웨어 기반으로 고속 IP 주소 검색 알고리즘을 분석하고, 최근 제안된 방법보다 성능이 우수한 새로운 알고리즘을 제안하고, 성능 실험을 통하여 그 우수성을 증명한다. 본 논문에서는, IP 주소 검색에 적용되는 트라이 구조를 개선한 LC (Level Compressed)-트라이에 대하여 분석하고, 성능 향상을 위하여 자주 검색되는 IP 주소로 새로운 LC-트라이를 구축하여 운용하는 다중 LC-트라이 구조를 제안한다. 본 연구에서 제안하는 다중 LC-트라이는 기존 LC-트라이에 비하여 트리의 깊이를 더욱 줄여 주소를 검색하는 시간이 기존의 LC-트라이에 비하여 1/2 수준으로 줄어들 수 있음을 실험을 통해 확인하였다.

본 논문의 구성은 다음과 같다. 2장과 3장에서는 본 연구의 연구 배경에 대하여 알아보고, IP 주소 검색의 문제점을 살펴본다. 그리고 본 연구의 기반이 되는 기존 LC-트라이의 원리에 대하여 알아본다. 4장에서는 본 논문에서 제안하는 다중 LC-트라이 알고리즘에 대하여 기술하고, 그 원리를 알아본다. 5장에서는 LC-트라이의 두 알고리즘 성능을 실제 트래픽 통계 데이터를 이용하여 실험한 결과를 제시한다. 본 연구에서는 두 알고리즘을 실제 구현하고 이를 실제로 운용되는 인터넷의 IP 주소 및 접근 통계를 이용해 시간을 실측하였다. 그리고 마지막으로 6장에서 추후 연구 내용과 함께 결론을 맺는다.

Ⅱ. 연구 배경

현재 인터넷에서 사용하고 있는 IP는 버전 4 (IPv4)로 32비트 길이의 주소를 갖는다. 따라서 2^32개의 주소인 약 40억개의 주소를 갖는다.

IP 주소는 네트워크 ID와 호스트 ID로 나누어지고, 초기 IP는 네트워크를 식별하기 위하여 3개의 클래스-A 클래스, B 클래스, C 클래스-로 나누었다. 이러한 클래스 기반의 네트워크 식별은 라우터의 동작을 단순화하는 이점이 있으나 주소가 비효율적으로 할당됨으로 인하여 심각한 IP 주소 고갈 문제를 야기하게 되었다. IP 주소 고갈 문제는 무선 단말, 가전 단말 등 다양한 형태의 단말을 통해 인터넷을 접속하는 환경에서 더욱 문제가 심각하다. 이러한 문제의 심각성에 의해 최근에는 IP 주소를 128비트로 확장한 IPv6가 제안되어 있다[6]. 하지만 모든 인터넷 라우터를 IPv6로 전환하는 데에는 상당한 시일이 소요될 것으로 보인다.

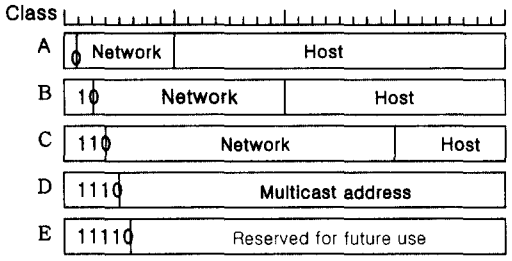


그림 1. A, B, C, D 클래스의 IP 주소 체계

이를 해결하기 위해 CIDR (Classless Inter-Domain Routing)이라는 방법이 제안되었다[7]. CIDR는 그림 1과 같은 기존의 IP 주소 체계에서 A, B, C, D 등으로 구분하던 클래스를 없애고, 가변 길이의 IP 주소를 네트워크 ID로 사용하는 방법이다. 클래스 A는 126 (= 2^7 - 2)개의 네크워크로 각각 2^24개의 호스트를 가지고, 클래스 B는 약 2^14개의 네크워크로 각각 2^16개의 호스트를 가지는 등 클래스로 구분한 IP 주소 체계는 통신망의 성장에 따라 심각한 문제를 야기한다. 따라서 통신

망을 분할하여 서브넷 (subnet)으로 구분하는 기술이 도입되었다. 나아가 IP 주소를 통신망에 할당할 때 클래스 단위로 할당하지 않고 2^i개의 IP 주소를 할당하는 CIDR이 널리 적용되고 있다. 즉, 2000개의 IP 주소를 할당할 때, B 클래스의 네크워크 IP 주소를 할당하는 것이 아니라 인접한 8개의 C 클래스 주소를 할당한다. 여기서 네트워크 프리픽스 (network-prefix) 개념을 도입하여 임의 길이의 IP 주소를 네트워트 ID로 사용한다. CIDR 모델에서는 IP 주소를 이용하여 패킷을 라우팅하기 위하여 목적지 IP 주소뿐 아니라 유효한 네트워크 ID를 추출하기 위한 비트 마스크 (bit mask) 또는 프리픽스 길이를 같이 표현해야 한다. 예를 들어, 최대 16개의 호스트로 서브네트워크를 구축하는 경우에 네트워크 주소로 168.188.46.80을 할당하면, 비트 마스크는 255.255.255.240이므로 프리픽스의 길이는 28이 되고, 이 서브네트워크에 접속되는 호스트는 168.188.46.80부터 168.188.46.95까지 할당될 수 있다. 이 경우에 라우터는 패킷의 목적지 IP 주소로부터 가변 길이의 네트워크 ID를 찾아야 하므로 일치되는 네트워크 ID 중에서 가장 길게 일치하는 프리픽스가 목적지의 물리적 포트와 연관된 정보임을 알아야 한다.

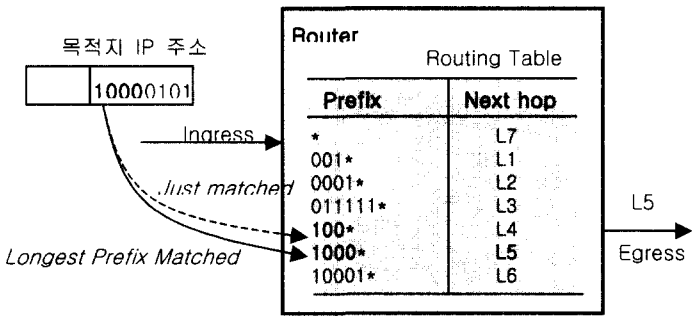


그림 2. IP 라우팅 및 LPM 예

IP 주소 고갈 문제가 심각해지고, 따라서 서브넷 마스킹과 CIDR의 사용이 보편화됨에 따라 가장 긴 프리픽스에 일치하는 LPM (Longest Prefix Matching)이 라우터의 성능을 좌우하게 되었다[8]. LPM을 위하여 일반적으로 IP 주소를 MSB (Most Significant Bit)부터 순차적으로 비교하여 가장 긴 주소가 일치되는 네트워크 ID를 찾아야 한다. 따라서 하나의 주소를 검색하는데 여러 번의 메모리 액세스를 필요로 한다. 초기 라우터에서 이러한 과정을 소프트웨어로 수행하였고, 데이터의 처리 속도가 심각한 문제로 등장하기 전에는 IP 주소 검색이 라우터의 중요한 병목이 되지 못하였다. 그러나 백본망을 중심으로 고속 IP 포워팅의 필요성이 증대되고, 최근의 라우터 구조가 스위치 패브릭을 통한 분산 병렬 처리 시스템으로 발전함에 따라 이러한 IP 주소 검색은 라우터의 중요한 병목이 되고 있는 실정이다[9][10]. 본 논문에서는 기존의 LC-트라이 방법의 성능을 개선하여 소프트웨어 기반의 고속 라우터를 구성하는데 그 목표를 둔다.

Ⅲ. IP 주소 검색

인터넷에서 IP 패킷은 데이터그램을 구성하는 기본 단위로 목적지 IP 주소가 라우팅에 이용된다. 여기서 IP 주소는 현재 인터넷에 접속된 모든 호스트에 대한 유일한 식별이 가능해야 하므로 ATM의 VPI/VCI나 MPLS의 레이블과는 그 특성이 다르다. 따라서 라우터는 라우팅이 가능한 모든 서브네트워크의 프리픽스를 라우팅 테이블에서 유지하여야 한다. 그러므로 라우터에서 관리하는 라우팅 테이블의 크기는 최대 수십만 호스트나 서브네트워크의 IP 주소를 갖는다.

그림 2는 IP 라우팅의 원리를 설명하고 이를 위한 LPM의 예를 보이고 있다. 예를 들어, 입력 패킷의 목적지 IP 주소가 10000101인 경우에 라우팅 테이블에 일치되는 엔트리는 100과 1000 두 가지가 존재한다. 그러나 가장 길게 일치되는 IP 주소는 1000이므로 해당 라우팅 테이블 엔트리의 다음 홉 (next hop)인 L5로 라우팅하게 된다.

IP 라우팅에 가장 널리 사용되는 알고리즘은 패트리샤 (Patricia: Practical Algorithm to Retrieve Information Coded in Alphanumeric) 트리를 이용한 검색 방법이다[11][12]. IP 주소와 같은 이진수 표현의 데이터 검색에 이진 트리 형태의 이진 트라이를 사용한다. 그러나 성능 향상을 위하여 차수가 1인 노드를 제거하여 압축 이진 트라이를 사용할 수 있고, 중간 노드와 단말 노드를 동일하게 구성한 패트리샤를 IP 주소 검색에 사용한다. 그림 3은 그림 2에서의 IP 주소로 구성된 패트리샤의 예를 보이고 있다. 그림 2의 프리픽스는 트리 탐색을 위하여 가변 길이의 비트로 표현되는데, 그림 3에서는 편의상 프리픽스와 함께 탐색할 키 값을 같이 표현하였다.

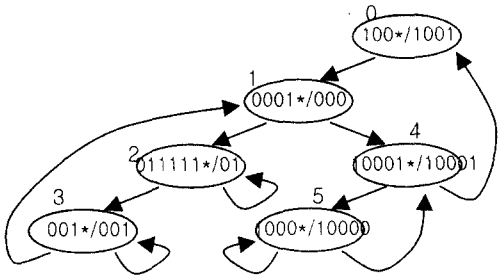


그림 3. 패트리샤 트리 구성의 예

가장 위에 존재하는 노드가 헤더 노드이고 트리의 탐색은 헤더 노드의 왼쪽 노드부터 시작한다. 트리 탐색은 자신의 노드나 상위 노드의 검색시 종료되고, 트리의 압축을 위하여 각 노드는 검색할 비트의 위치를 갖는다. 예를 들어 001의 키 값을 찾기 위하여 헤더의 왼쪽 서브노드부터 시작하여 1, 2, 3번째 비트를 차례로 검사하는데, 마지막에서 자신의 노드를 검색하므로 종료한다. 같은 방법으로 1001의 키 값을 찾기 위하여 헤더의 왼쪽 노드에서 1번 비트를 검사하여 오른쪽으로 분기하고, 4번 비트를 검사하여 오른쪽으로 분기하면 상위 노드를 검색하므로 검색을 종료한다.

이와 같이 패트리샤 트리를 이용한 IP 주소 검색 방법은 IP 주소의 길이를 M이라 할 때, 최대 M번의 메모리 액세스가 필요하다. 라우팅 테이블에서 관리되는 IP 주소의 수는 라우터의 규모에 따라 다르지만 백본 라우터의 경우에 수 십만 개의 IP 주소를 관리하는 것이 일반적이다. 따라서 SRAM과 같은 고속 메모리를 사용하기 보다는 DRAM과 같은 메모리를 사용하는 것이 일반적인 방법이고 이 경우에 한 번의 메모리 액세스에 수십 ns의 접근 시간이 소요된다. 현재 물리적 인터페이스의 속도는 수백 Mbps에서 수십 Gbps에 이르고 있으므로 와이어 속도의 처리를 위하여 인터페이스마다 최대 수백 Kpps (packet per second)에서 수십 Mpps에 이르는 처리율을 가져야 한다. 즉, 1 ms이하에서 수백 ns이하의 시간에 하나의 패킷을 처리해야 하므로 고속 라우터의 경우에 기존의 패트리샤 트리를 이용하는 방법에는 한계가 있다.

시스코의 CEF (Cisco Express Forwarding)[13]에 따르면 단순한 형태의 트라이로 라우팅 테이블을 구축하고 있다. 하지만 고속 라우터의 구현을 위해 IP 주소 검색 성능 향상이 절실하여 개선된 알고리즘에 의한 라우팅 테이블 구축이 활발해지고 있다.

LC-트라이 알고리즘은 기본적으로 패트리샤 트라이를 바탕으로 동작하는데, 하나의 노드에서 한 비트 정보를 통해 분기하는 것이 아니라, 한번에 여러 비트의 정보를 통해 분기함으로써 전체 트라이의 깊이를 축약한다. 이렇게 트리의 레벨을 축약함으로써 메모리의 접근 회수를 줄일 수 있는데, 이로써 고속 라우팅이 가능하다. 실제로 LC-트라이에 의한 메모리 접근은 평균 2-3회의 접근으로 IP 주소 검색이 가능하고, 최악의 경우에도 5-6회의 접근으로 LPM에 의한 IP 주소 검색이 가능하다. IP 주소 검색이 고속 라우터 구현에서의 병목이 되는 상황을 고려할 때, 이러한 메모리 접근 회수의 감소는 라우터 성능을 결정하는 중요한 요인이 된다.

표 1은 16개의 엔트리를 갖는 LC-트라이를 구성하기 위한 예제 프리픽스를 보인 것이다. 이를 이용하여 16개의 엔트리로 LC-트라이를 구성하면 그림 4에서 보는 바와 같다. Skip 값은 패트리샤 트리에서도 사용된 값으로 더 이상의 분기가 IP 주소 검색에 의미없는 비트를 검사하지 않음으로써 탐색의 경로 압축 (path compression) 효과를 얻을 수 있다. LC-트라이에서는 이러한 경로 압축에 추가하여 한번에 분기할 수 있는 2^N개의 서브트라이를 찾고 이때의 N값을 분기 인자 (branching factor)로 하여 2^N개의 서브노드로 분기한다.

표 1에서 모든 IP 주소의 접근이 균등할 경우에 평균과 최대 프리픽스의 길이는 각각 5.75와 11이 된다. 그러나 그림 4에서 보는 바와 같이 LC-트라이를 구성하는 경우에 메모리의 접근 회수는 평균과 최대가 각각 2.19와 4이다. 이와 같이 트라이의

표 1. 예제 프리픽스

번호	프리픽스
0	0000000
1	0000001
2	001
3	0100
4	0101
5	011
6	100000
7	1000010
8	1000011
9	100010
10	100011
11	1001
12	101
13	110
14	11111101000
15	11111101001

레벨을 줄임으로써 메모리 접근 회수를 줄이게 되고, 나아가 IP 주소 검색 시간을 줄여 고속 라우팅이 가능해진다.

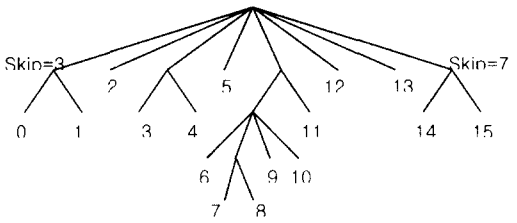


그림 4. 표 1에 의한 LC-트라이 구성

실제 IP 트래픽을 분석해보면 IP 프리픽스의 길이 분포나 접근 패턴이 균등하지 않다. IP 트래픽에 대한 실제 운영 결과를 통계로 제공하는 MAE-EAST와 FUNET에 따르면 라우팅에 유효한 프리픽스의 길이가 8-15비트는 1.1%, 16-24비트가 98.8%, 나머지 25-32비트가 0.1%이다[14]. CIDR가 도입되었으나 여전히 B 클래스와 C 클래스의 서버네트워크로 라우팅되는 경우가 많음을 의미한다. 그리고 전체 41,709개의 엔트리를 검색하는 과정에서 192로 시작하는 프리픽스와 같이 특정 엔트리의 접근이 편중되어 있음을 알 수 있다. 이는 전체의 라우팅 테이블에서 특정 프리픽스로 시작되는 IP 주소로의 패킷 포워딩이 자주 일어나 이에 해당하는 엔트리의 검색이 다른 엔트리에 비하여 자주 수행됨을 의미한다. 실제로 192, 199, 204, 198, 203, 194인 6개의 8비트 프리픽스로 시작하는 IP 패킷이 전체 트래픽의 52%에 해당한다. 본 논문에서는 이러한 IP 주소의 편향된 접근 패턴을 이용하여 LC-트라이의 레벨을 더욱 축약하는 방법을 제시하는데 자세한 알고리즘은 다음 절에 설명한다.

라우팅 테이블 내용의 변경은 라우터가 관리하는 서브네트워크의 상태가 변경되는 경우에 이루어진다. 이때 RIP (Routing Internet Protocol)나 OSPF (Open Shortest Path First)와 같은 라우팅 프로토콜을 이용하기도 한다. 실제로 서브네트워크의 상태는 자주 발생하지 않고, 많은 경우에 고정 라우팅 테이블을 이용하기도 한다. 따라서 서브네트워크의 구성이 완료되고 본격적으로 운용되는 상황에서 라우팅 테이블의 내용은 자주 변경되지 않고, 변경되는 경우에도 전체적인 트래픽의 성향에는 큰 변화가 없다고 볼 수 있다. 예를 들어, 정상적인 망 운용과정에서 192의 프리픽스를 갖는 서브네트워크로 라우팅되던 많은 트래픽이 갑자기 소멸되거나 다른 프리픽스의 망으로 변경되는 것은 자주 발생하지 않는다.

Ⅳ. 다중 LC 트라이

LC-트라이는 기존 패트리샤 트리에서 동시에 분기가 가능한 2^N개의 노드를 한 레벨로 축약함으로써 분기의 회수, 즉, 메모리의 접근 회수를 줄였다. IPv4의 주소 검색에 적용할 경우에 최대 깊이가 5이고 평균 2를 넘지 않음을 보이고 있다[15]. 하지만 이 실험에서는 LC-트라이의 레벨을 줄이기 위하여 실제 엔트리의 50%, 즉, fill factor가 0.5인 경우에 분기가 수행되도록 하여 IP 주소 검색 시간을 줄이기 위하여 메모리의 낭비는 감수해야 한다.

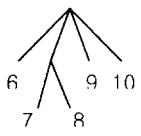
본 논문에서 제안하는 다중 LC-트라이 알고리즘은 하나 이상의 LC-트라이로 IP 주소를 검색하는 알고리즘이다. 비록 단일 LC-트라이의 평균 레벨이 2 이하라 할지라도 라우터의 성능은 IP 검색에 걸리는 최대 시간에 영향을 받게되고, 이 경우에 5번의 메모리 액세스가 이루어 진다. 이는 ATM이나 MPLS와 같이 데이터 링크 계층의 연결 ID를 갖는 경우에 비하여 메모리 액세스 비용이 여전히 높음을 알 수 있다. 그리고 메모리 액세스 회수를 줄이기 위하여 fill factor를 0.5로 설정하였는데, 이는 최대 50%의 라우팅 테이블이 낭비됨을 의미한다. 앞에서 기술한 바와 같이 IP 라우팅 테이블은 현재 서비스 중인 데이터 플로우 (flow)의 정보뿐 아니라 모든 라우팅 가능한 호스트 및 서브네트워크에 대한 정보를 가지고 있으므로 50%까지의 메모리의 낭비는 심각하다. 따라서 다중 LC-트라이는 LC-트라이의 레벨을 보다 축소하여 IP 주소 검색 시간을 단축하거나, 같은 성능을 얻기 위하여 fill factor를 높여 메모리 효율을 높이는데 그 목적이 있다.

다중 LC-트라이의 원리는 IP 주소 검색이 특정 IP들에 편향되어 있다는 점을 이용하여 별도의 LC-트라이를 유지하는 데 있다. 즉, 일부 백본 라우터에서 자주 검색되는 IP 주소를 링크 인터페이스 모듈로 캐쉬하여 분산, 병렬 처리함으로써 전체 시스템의 성능을 향상시키는 기법을 LC-트라이에 적용하여 여러 개의 LC-트라이를 관리하는 것이다. 이를 위하여 현재 라우팅 테이블에 있는 IP 주소의 액세스 패턴을 분석한다. 자주 검색되는 IP 주소의 공통 프리픽스를 찾고, 이 공통 프리픽스를 가진 모든 IP 주소 (G)-실제로는 네트워크 ID에 해당하는 프리픽스-로 별도의 LC-트라이를 구성한다. 이렇게

구성된 LC-트라이는 전체 IP 주소의 부분 집합 (F)에 해당하므로 단일 LC-트라이의 레벨보다 작은 레벨의 트라이를 구성한다. 그리고 나머지 IP 주소의 집합 (G - F)로 LC-트라이를 구성한다. 두 개의 LC-트라이로 다중 LC-트라이를 구성하는 경우에 검색하고자 하는 IP 주소가 F에 속하는지 G - F에 속하는지 한번의 판단이 필요하므로 F로 구성된 LC-트라이의 평균 레벨 L_avg(F)와 G - F로 구성된 LC-트라이의 평균 레벨 L_avg(G - F)가 원래의 G로 구성된 LC-트라이의 평균 레벨 L_avg(G) 보다 1 이상 작다면 IP 주소 검색의 성능이 향상된다. 표 1의 예제 프리픽스를 다중 LC-트라이에 적용한 경우를 살펴보자. 최상위 4비트를 공통 프리픽스 길이로 보고, 이 공통 프리픽스 중 가장 접근이 잦은 것을 1000이라고 하자. (예에서 모든 IP 주소의 접근 패턴이 균등한 것으로 가정하였으나 특정 공통 프리픽스에 편중된 경우도 같은 효과를 갖는다.) 따라서 공통 프리픽스 1000으로 시작하는 나머지 프리픽스, 즉 F로 LC-트라이를 구성하고 (그림 5 (a) 참조), G - F로 다른 LC-트라이를 구성한다 (그림 5 (b) 참조). 그림 4에서 L_avg(G)는 2.19이다. 하지만 새로 구성된 LC-트라이의 L_avg(F)와 L_avg (G - F)는 각각 1.4와 1.55이다.

자주 검색되는 IP 주소로 별도의 LC-트라이를 구성함으로써 캐쉬 효과의 이득을 얻을 수 있고, 단일 LC-트라이에 비하여 최대 레벨을 축소할 수 있다. 하지만 라우터에 입력된 IP 패킷의 목적지 주소가 F에 속하는지 G - F에 속하는지 판단하는 과정이 필요하다. 따라서 다음 절에서 실제 구현을 통하여 두 가지 기법의 주소 검색 시간을 측정하였다.

(a) 공통 프리픽스 1000을 가진 IP 주소에 대한 LC-트라이



(b) 나머지 IP 주소의 LC-트라이

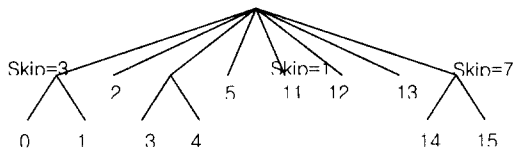


그림 5. 다중 LC-트라이

V. 성능 평가

본 논문에서는 LC-트라이의 성능 평가에 사용된 동일한 파라메터를 적용하였다. 즉, fill factor는 0.5로 하고, 루트 노드의 분기 인자 (branching factor)는 [15]에서 사용한 16을 실험에 사용하였다. 이 경우에 16비트의 프리픽스로 루트 노드에서 분기하므로 65,536개의 노드가 필요한데 실제 IP 주소의 수 41,709개를 초과한다. 따라서 본 실험에서는 분기 인자를 16외에 8, 4, 2로 다양하게 적용하였다. LC 트라이의 fill factor는 라우팅 테이블의 크기와 트리의 평균, 최대 깊이와 관련이 있다. 즉 [15]에 따르면 fill factor가 커질수록 트리의 깊이가 커져 lookup 시간이 길어지는 반면, 트리의 크기는 줄어든다. 본 연구에서는 FUNET의 실제 라우팅 테이블의 통계 정보를 이용하였고, Pentium III 550 MHz, RAM 128 MB, Linux 2.2 환경 하에서 실험하였다.

LC-트라이의 수를 1, 2, 3, 4로 늘려가며 실제 IP 주소 검색에 소요되는 시간을 측정하였는데, LC-트라이의 수가 1인 경우는 기존의 LC-트라이를 이용한 IP 주소 검색에 해당한다. 표 2는 LC-트라이 수 (M = 1, 2, 3, 4)에 따라 잦은 빈도로 접근되는 IP 주소의 집합 Fi (i = 1, 2, ..., M)로 다중 LC-트라이를 구성하는 경우 (a)는 트라이 수 (M)에 따른 M 개의 트라이의 평균 레벨을 나타낸 것이고, (b)는 이 M 개의 트라이 중에서 가장 자주 접근되는 IP로 구성된 트라이의 평균 레벨을 나타낸 것이다.

단일 LC-트라이의 평균 레벨은 [15]에서 지적한 바와 같이 분기 인자가 16인 경우에 급격하게 줄어든다. 그러나 분기 인자가 4에서 8로 늘어나는 경우에도 IP 주소의 분포에 따라 레벨 수가 줄어들지 않는 경우도 있다. 다중 LC-트라이의 평균 레벨 수

표 2. 단일 LC-트라이 및 다중 LC-트라이에서

(a) 트라이 수 (M)에 따른 전체 트라이의 평균 레벨

분기 인자	Lavg(F_1)	Lavg(F_1UF_2)	Lavg ($F_1UF_2UF_3$)	Lavg($F_1UF_2UF_3UF_4$)
2	3.98	4.02	4.00	4.04
4	3.36	3.32	3.28	3.40
8	3.62	3.17	3.28	3.13
16	1.73	1.66	1.60	1.53

(b) 접근 빈도가 높은 IP로 구성된 트라이의 평균 레벨

분기인자	Lavg(F1)	Lavg(F2)	Lavg(F3)	Lavg(F4)
2	3.98	3.10	3.33	3.28
4	3.36	3.09	3.20	3.16
8	3.62	2.55	2.62	2.58
16	1.73	1.00	1.00	1.00

는 단일 LC-트라이의 평균 레벨, 즉, L_avg(F1)보다 줄어들지만 분기 인자 2, 4의 경우에서 보는 바와 같이 약간 증가하는 경우도 있다. 그러나 자주 접근되는 IP 주소 집합, 즉, Fi로 구성된 LC-트라이의 평균 레벨 수는 항상 L_avg(F1)보다 작음을 볼 수 있다. 이를 통하여 다중 LC-트라이의 IP 검색 효율이 다소 향상됨을 볼 수 있는데, 실제 IP 접근에 걸리는 시간을 측정해보면 본 논문에서 제안하는 다중 LC-트라이의 방법이 기존의 단일 LC-트라이에 비하여 성능이 크게 향상됨을 확인할 수 있다. 그림 6은 표 2의 LC-트라이로 fill factor를 0.5로 하여 실제 IP 검색에 걸리는 시간을 실험한 결과이다.

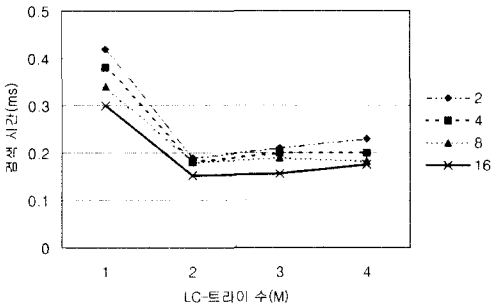


그림 6. LC-트라이 수 (M)에 따른 IP 주소 검색 시간 (ms)

분기 인자가 커질수록 루트 노드에서 분기되는 서브트라이의 수가 증가하고, 따라서 트라이의 레벨이 줄어든다. 그러나 자주 검색되는 IP 주소로 별도의 LC-트라이를 구성하는 경우에 단일 LC-트라이에 비하여 검색 시간은 반으로 줄어든다. 즉, 기존의 LC-트라이에서 IP 주소의 평균 검색 시간은 분기 인자 2, 4, 8, 16에 대하여 각각 0.42, 0.38, 0.34, 0.30 ms인데, 본 논문에서 제안하는 다중 LC-트라이에서 M의 값이 2일때 평균 검색 시간은 각각 0.19, 0.18, 0.18. 0.15 ms로 검색 시간이 2배 이상 향상되었다. LC-트라이의 수를 늘일수록 라우터가 입력 IP 패킷의 목적지 주소가 어떤 LC-트라이에 속하는 주소인지 판단하는 시간이 증가하므로 LC-트라이 레벨의 축소에 대한 이득을 얻지 못한다. 따라서 본 연구의 실험에서는 LC-트라이 수를 2로 실험한 결과에서 최적의 성능을 확인할 수 있었다.

Ⅵ. 결 론

최근, 고속 라우터를 구현할 때, 하드웨어를 이용하여 고속 IP 검색을 시도하고 있으나, 다양한 제어 기능을 구현하지 못하는 문제를 갖고 있다. LC-트라이는 기존의 소프트웨어 기반 라우터에서 적용하던 패트리샤 트리의 구조에서 메모리 접근을 크게 줄인 고속 IP 검색을 위한 데이터 구조로 기가비트 속도의 고속 IP 주소 검색이 가능하다. 본 논문에서 기존 LC-트라이 기법에서 IP 주소의 검색 패턴에 따라 다수의 LC-트라이를 구성하도록 함으로써 검색 시간면에서 상당한 진전이 있었다. 제안한 다중 LC-트라이는 라우팅 테이블을 구성하는 메모리 용량면에서는 동일하지만, 자주 접근되는 IP 주소로 작고 빠른 라우팅 테이블을 관리함으로써 기존 운영체제하에서도 성능 향상을 꾀할 수 있었다.

PC를 기반으로 실험한 결과를 실제 라우터에 적용하기엔 무리가 있으나, 기존 LC-트라이에 비해 상대적으로 우수한 결과를 얻었고, 라우터 전용 운영체제하에서 보다 최적화하여 운영하는 경우에 와이어 속도의 IP 포워딩이 가능할 것으로 보인다. 그리고, 실제 라우터에서는 자주 접근하는 IP 주소를 캐쉬에서 관리하거나 인터페이스별로 분산 처리하는 등 다양한 방법으로 라우팅의 성능 향상을 꾀하고 있다. 추후 이러한 점을 고려한 다중 LC-트라이 기법의 개선을 통하여 보다 빠른 IP 라우팅을 구현할 수 있을 것이다.

참 고 문 헌

[1] P. Gupta, S. Lin, M, McKeown, "Routing Lookups in Hardware at Memory Access Speeds," *IEEE INFOCOM,* Mar. 1998.

[2] H. H.-Y. Tzeng, T. Prizygienda, "On Fast Address-Lookup Algorithms," *IEEE JSAC* 17(6), pp. 1067-1082, Jun. 1999.

[3] G. Huston, *Internet Performance: Survival Guide,* Wiley, pp. 431-436, 2000.

[4] A. McAuley, P. Francis, "Fast Routing Table Lookup Using CAMs," *IEEE INFOCOM*, pp. 1382-1391, Mar. 1993.

[5] W. Bux, W. E. Denzel, et al., "Technologies and Building Blocks for Fast Packet Forwarding," *IEEE Comm. Mag.*, Vol. 39, pp. 70-84, Jan. 2001.

[6] Y. Rekhter, P. Lothberg, R. Hinden, S. Deering, J. Postel, "An IPv6 Provider-Based Unicast Address Format," *RFC 2073*, IETF, Jan. 1997.

[7] V. Fuller, T. Li, J. Yu, K. Varadhan, "Classless Inter-Domain Routing (CIDR): An address Assignment and Aggregation Strategy," *RFC 1519*, IETF, Sep. 1993.

[8] M. Degermark, A. Brodnik, S. Carlsson, S. Pink, "Small Forwarding Tables for Fast Routing Lookups," *Comp. Comm. Review*, Vol. 27, pp. 3-14, Oct. 1997.

[9] S. Keshav, R. Sharma, "Issues and Trends in Router Design," *IEEE Comm. Mag.*, Vol. 36, pp. 144-151, May 1998.

[10] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, "Scalable High Speed IP Routing Lookups," *ACM Comp. Comm. Rev.*, Vol. 27, pp. 25-36, Oct. 1997.

[11] R. Sedgewick, *Algorithms in C*, Addison-Wesley, pp. 253, 1990.

[12] W. Doeringer, G. Karjoth, M Nassehi, "Routing on Longest Matching Prefixes," *IEEE/ACM Trans. Networking*, Vol. 4, pp. 86-97, Feb. 1996.

[13] Cisco, How to Choose the Best Router Switching Path for Your Network, http://www.cisco.com/warp/public/105/20.html

[14] IPMA, Internet Performance Measurement and Analysis, http://www.merit.edu/ipma.

[15] S. Nilsson, G. Karlsson, "IP-Address Lookup Using LC-Tries," *IEEE JSAC*, 17(6), pp. 1083 - 1092, Jun. 1999.

황 현 숙(Hyun-Sook Hwang) 비회원

1998년 2월 : 배재대학교 컴퓨터공학과 졸업

2001년 2월 : 충남대학교 교육대학원 컴퓨터공학교육 석사

<주관심 분야> 초고속 통신망, 멀티미디어 통신

권 택 근(Taeck-Geun Kwon) 정회원

1988년 2월 : 서울대학교 컴퓨터공학과 졸업

1990년 2월 : 서울대학교 컴퓨터공학과 석사

1996년 2월 : 서울대학교 컴퓨터공학과 박사

1992년 1월~1998년 8월: LG전자 연구원

1998년 9월~현재 : 충남대학교 정보통신공학부 조교수

<주관심 분야> 초고속 통신망, 멀티미디어 통신, 멀티미디어 서버