고속 라우터를 위한 IP 주소 검색 기법†

정상훈[0] 권위남 권보섭 진성기 윤현수 조정완
한국과학기술원 전자전산학과 전산학전공
{shchung, wnkwon, bskwon, skjean, hyoon, jwcho}@camars.kaist.ac.kr

A IP Address Lookup Scheme For High Speed Routers

S.H. Chung[0] W.N. Kwon B.S. Kwon S.K. Jean H. Yoon J.W. Cho
Dept. of EECS, Division of Computer Science, KAIST, AITrc

요 약

현재 인터넷은 매우 빠른 속도로 커가고 있으며 기존의 인터넷 하부구조(infrastructure), 특히 라우터에 커다란 부담이 되고 있다. IP 주소 검색은 라우터에 들어오는 패킷의 출력 링크를 찾기 위해 전송 테이블에서 가장 길게 일치하는 프리픽스를 찾는 것이다. 이러한 작업은 매우 복잡하고 고속의 라우터에 커다란 병목이 되고 있으며 이를 해소하기 위해서는 하드웨어기반의 빠른 IP 주소 검색 기법이 필요하다. 본 논문은 유니 캐스트 상에서 전송 테이블의 크기와 검색 시간을 줄이고 점차적인 갱신이 가능한 하드웨어기반의 알고리즘을 제시하고 다른 하드웨어 기반의 알고리즘과 성능을 비교한다. 제시한 알고리즘은 작은 크기의 SRAM과 단순한 로직의 하드웨어로 구현되기 때문에 값이 싸고 파이프라인으로의 구성이 가능하기 때문에 빠른 IP 주소 검색이 가능하다. 10ns의 SRAM으로 구현할 경우, 초당 100×10^6의 검색이 가능하고 이는 지금까지 제안된 알고리즘보다 빠른 검색을 제공할 수 있다.

1. 서론

WWW(World Wide Web) 및 인터넷을 통한 서비스 및 응용 프로그램이 다양해짐에 따라 인터넷에 접속된 컴퓨터 수와 트래픽의 양이 기하급수적으로 증가하고 있는 추세이다. 따라서 기존의 인터넷 하부구조는 커다란 압박을 받게 되었고 이를 해결하기 위해 큰 대역폭의 전송 매체와 고속 라우터가 필요하게 되었다. 광통신 섬유 기술의 발전으로 전송매체는 충분한 양의 대역폭을 제공할 수 있지만 라우터는 아직도 과제로 남아있다.

패킷은 라우터의 IP 주소 검색을 통해서 목적지 주소로 전송된다. 패킷이 라우터의 입력 포트에 도착하면 IP 주소 검색을 통해 어떤 출력 포트로 전송될지 결정되고 스위칭 회로와 출력 포트를 지나 스케줄링을 통해 라우터를 떠난다. IP 주소검색은 <프리픽스/프리픽스 길이, 출력포트>의 정보를 가진 전송 테이블에서 패킷의 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 작업이다.

고속 라우터에서 IP 주소 검색은 라우터 성능, 곧 패킷의 전송 속도를 제한하는 연산이다. 패킷의 평균 길이가 1000 bits라고 할 때 라우터가 1Gb/s의 전송 속도를 내기 위해서는 초당 10^6개의 패킷을 전송해야 하는데 이는 패킷당 주소 검색이 1μs 내에 수행되어야 함을 의미한다. 인터넷의 코어(core) 네트웍에 쓰일 OC-192(9.953 Gb/s)는 초당 10×10^6개의 패킷을 전송해야 하고 이를 위해서 패킷당 주소 검색은 100ns내에 처리되어야 한다. 앞으로 보다 빠른 패킷 전송이 요구되므로 고속의 IP 주소 검색 알고리즘이 절실히 필요하다. 또한, 프리픽스는 네트웍의 상황에 따라 동적으로 변화하기 때문에 전송 테이블은 이를 잘 반영할 수 있도록 프리픽스의 삽입과 삭제를 효율적으로 수행할 수 있어야 한다.

본 논문에서는 고속의 라우터를 위한 하드웨어기반의 IP 주소 검색 알고리즘을 제안한다. 제시되는 알고리즘은 평균 검색 시간을 줄이고 전송 테이블의 점차적인 갱신이 가능하고 하드웨어 구현이 용이하다는 장점을 가진다. 본 논문의 구성은 다음과 같다. 2절에서는 기존에 제안된 IP 주소 검색 기법을 살펴보고 3절에서 새로운 하드웨어기반의 IP 주소 검색 기법인 UCBM (Updatable Compression Bit Map)을 제안한다. 4절에서는 시뮬레이션을 통해 기존의 연구들과 성능을 비교하고 5절에서는 결론을 내린다.

2. 관련연구

빠른 IP 주소 검색 알고리즘을 개발하기 위한 방향은 메모리 접근 횟수를 줄이는 것과 작은 전송 테이블을 만들어 빠른 메모리를 이용하는 것이다[1,2,3,4]. 최근에 고속의 라우터를 지원하는 여러 가지 알고리즘이 등장하였고 하드웨어 기반의 빠른 알고리즘도 제안되었다.

DIR-24-8 기법은 매우 큰 메모리를 필요로 하는 하드웨어 기반의 알고리즘으로서 최악의 경우 두 번의 메모리 접근이 필요하다[1]. Huang의 알고리즘은 CBM(Compression Bit Map) 기법[3]을 적용한 하드웨어 기반의 알고리즘이다. 이 알고리즘은 전송 테이블을 3단계로 나누어 저장하고 16보다 긴 길이의 프리픽스를 압축하여 메모리 요구량과 접근 횟수를 줄였다. 메모리 요구량을 줄이기 위해서 테이블을 여러 종류의 크기로 구성하였다. 이로 인해서 전송 테이블이 갱신될 때 테이블 크기가 변하면 대부분의 메모리를 다시 써야 하고 그동안 검색을 수행할 수 없기 때문에 검색 속도에 영향을 미칠 수 있다. Huang의 알고리즘에서는 이러한 점을 완화하기 위해 듀얼 메모리 뱅크(dual memory bank)를 쓰는데 이는 메모리 요구량을 2배로 증가시킨다. 본 논문에서 제시하는 UCBM은 테이블을 특정한 크기로 고정함으로써 갱신할 때

†본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음

메모리에 쓰는 양을 줄였으며 요구되는 메모리 양의 측면에서도 Huang의 알고리즘보다 뛰어나다.

3. IP 주소 검색 기법(Updatable CMB 기법)

본 절에서는 점차적인 갱신이 가능하고 평균 검색시간을 줄이고 메모리 요구량이 작은 알고리즘인 UCBM에 대해서 자세히 살펴본다. UCBM은 [그림 1]과 같은 간접적 검색 기법을 기반으로 하였고 메모리 요구량을 줄이기 위해 CBM 기법을 적용하였다. 16 비트이하의 프리픽스는 레벨 1의 SEGT(Segmentation Table)에 저장하고 17~24 비트의 프리픽스는 레벨 2의 NHA(Next Hop Array), NPA(Next hop/Pointer Array)에 저장하고 25 비트이상의 프리픽스는 레벨 3의 NHA에 저장한다. NHA는 출력 정보(<256)만을 저장한 테이블로 엔트리는 1 Byte이고 NPA는 포인터 정보와 출력 정보를 함께 저장한 테이블로 엔트리는 2 Byte이다.

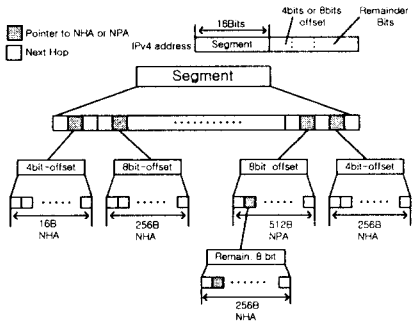

그림 1. 간접적 검색 기법

라우터의 프리픽스는 대부분 16에서 24이하의 길이를 가지기 때문에 256 Byte의 NHA와 512 Byte의 NPA는 많은 메모리를 차지한다. 따라서 CBM 기법을 적용해서 압축하여 CWA(Code Word Array)와 CNHA(Compressed NHA)(또는 CNPA)로 만들어 메모리 요구량을 줄인다. 전체적인 테이블의 구조는 [그림 3]과 같다. 논문의 표기상 32 비트의 IP 주소를 16비트의 세그먼트와 다음 4비트 혹은 8비트의 오프셋으로 나누고 DA[a:b]는 패킷 주소의 a 비트에서 b 비트($1 \le a \le b \le 32$)까지 값이라 한다.

3.1. 전송 테이블의 구조

레벨 1의 SEGT(segmentation Table)은 세그먼트로 구분 가능한 0.0~255.255의 프리픽스(2^{16}개)를 저장한다. SEGT 엔트리는 출력 정보나 레벨 2의 NHA나 CWA/CNHA(CNPA)를 가리키는 포인터를 저장한다. 레벨 2 테이블은 17~20비트의 프리픽스를 저장하는 NHA와 17~24비트의 프리픽스를 저장하고 출력 정보만을 가진 CWA/CNHA와, 출력 정보와 포인터를 저장한 CWA/CNPA로 이루어진다. 레벨 3의 NHA는 24보다 길이가 길고 8비트로 구분 가능한 모든 프리픽스의 출력 정보를 저장한다. SEGT와 NHA를 구성하는 방법은 [2]와 같이 한다.

3.2. CWA/CNHA(CNPA) 구성

CWA와 CNHA를 구성하기 위해서 같은 세그먼트 값을 가지는 프리픽스 그룹에 대해 각 프리픽스의 범위를 구하고 이를 노드로 하는 시블링 트리 구조를 구성한다. 프리픽스 p_i의 범위는 시작점인 $S(p_i)$와 $E(p_i)$로 표시된다. 예를 들어 p_1=143. 248.0/17은 143.248.0/24에서 143.248.127/24의 프리픽스를 포함하므로 범위는 $S(p_1)$=0, $E(p_1)$=127이다. 프리픽스 범위 종속 관계에 대해 시블링 트리를 구성할 수 있다. 부모 노드는 자식 노드의 범위를 포함하고 시블링들은 서로 범위를 포함하지 않는다. 각 시블링 노드는 시작점이 작은 것부터 왼쪽에 있다. [그림 2]는 프리픽스 범위에 대한 시블링 트리 구성 예를 보여준다.

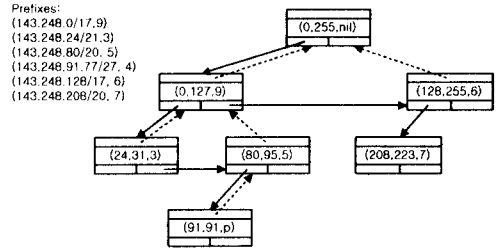

그림 2. 프리픽스 그룹에 대한 시블링 트리 구성 예

시블링 트리를 레벨 별로 방문하면서 CWA와 CNHA(CNPA)를 구성한다. CWA는 출력 정보가 저장되는 범위를 0과 1의 비트열로 표현하는데 시작점을 1로 하고 나머지를 0으로 하여 표현한다. [그림 3]의 중간 부분은 CWA/CPNA에 대한 예를 보여준다.

3.3. 전체 테이블 구성 예

[그림 3]의 예를 통해서 각 레벨의 테이블의 검색하는 방법을 설명한다. SEGT 엔트리는 마지막 2 비트에 EO(Encoded offset length)를 저장하는데, EO는 출력 정보, 포인터, 검사할 DA(Destination Address)의 폭을 나타낸다.

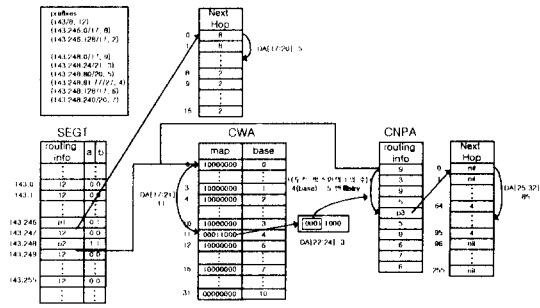

그림 3. 전체 테이블 구성 예

IP 패킷의 목적지 주소가 143.248.91.85의 경우, 세그먼트는 143.248이고 엔트리의 EO=11인데 DA[17:21](=11)로 CWA의 엔트리를 찾는다. CWA 엔트리 내에서 첫 비트부터 DA[22:24]번째 비트까지 1로 된 비트의 수를 더하고 여기에 베이스의 값을 더하면 CNPA의 인덱스가 된다. CNPA에는 포인터가 저장되어 있으므로 DA[25:32](=85)로 레벨 4를 검색하면 출력 정보 4를 얻는다.

3.4. 하드웨어 설계

[그림 4]는 본 논문에서 제안하는 검색 알고리즘의 하드웨어 설계도이다. 전송 테이블은 네 개의 메모리 뱅크에 나누어 저장되어 검색시 파이프라인이 가능하다. 하드웨어 설계시 가장 많은 시간을 소모하는 부분은 CWA를 계산하는 부분인데, 엔트리의 비트가 길어질수록 시간은 더 증가한다.

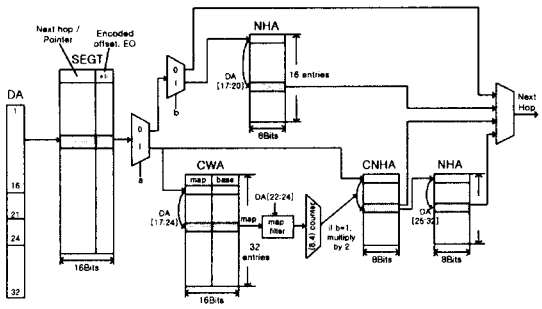

그림 4 하드웨어 설계도

3.5. 갱신 알고리즘

사용 가능한 메모리를 64 Byte와 32 Byte 의 블럭으로 나누어 관리하므로 점차적인 갱신이 가능하다. 예를 들어 1 블럭의 CWA와 CNHA에 프리픽스가 삽입되어 1 블럭이 더 필요할 경우에는 메모리 풀에서 CWA와 CNHA에 각각 2 블럭을 재할당하고 1 블럭을 메모리 풀에 넣는다. CWA/CNHA에서 프리픽스 삽입이 일어날 경우 다시 써야하는 메모리의 양은 최악의 경우 8개 블럭의 CNHA와 1개 블럭의 CWA로 총 64*9 = 576 Byte 이다. SRAM 의 한 행(row)이 2 Byte 이고 SRAM의 접근시간이 10ns 이면 대략 3 ms 이면 갱신을 마칠 수 있다. 평균적인 갱신시간은 이보다 더 빠르다. 그리고 SEGT와 NHA을 갱신하는 경우에는 [1]에서 제안한 범위 갱신(subrange-update)을 쓰면 보다 효율적이다.

4. 성능평가

본 논문에서 제안하는 알고리즘의 성능을 평가하기 위해 IPMA (Internet Performance Measurement and Analysis) 프로젝트에서 제공하는 라우팅 테이블[5]을 사용하여 시뮬레이션하였다. 전송 테이블의 크기, 메모리 접근 횟수, 검색 시간에 대해 하드웨어 기반의 검색 기법인 Huang의 알고리즘, DIR-24-BASIC과 비교하였다.

[표 1]은 각 기법마다 생성하는 전송 테이블의 크기를 보여준다. DIR-24-BASIC은 고정적으로 33MB의 메모리를 필요로하고 Huang의 알고리즘은 338KB-624KB의 전송 테이블을 만들지만 갱신을 위해 듀얼 메모리를 쓰므로 2배의 메모리가 필요하다. UCBM은 세 알고리즘 중에서 가장 작은 크기의 전송 테이블을 생성한다.

Site	Prefix	Proposed Scheme	Huang's	DIR-24-BASIC
Mae-East	52,791	558KB	624KB	33MB
Mae-West	32,652	449KB	496KB	33MB
AADS	20,693	374KB	408KB	33MB
Pac Bell	29,775	428KB	459KB	33MB
Paix	11,360	301KB	338KB	33MB

표1. 전송 테이블 크기 비교

메모리 접근 횟수는 최악의 경우에 대하여 DIR-24-BASIC은 2번, Huang의 알고리즘은 3번, UCBM은 4번이 필요하다. UCBM은 메모리 접근이 Huang의 알고리즘보다 한 번 더 많은 대신 전송 테이블의 크기가 작고, DIR-24-BASIC은 메모리가 많이 필요한 대신 접근 횟수가 가장 작다.

다음으로 각 알고리즘의 검색 시간을 비교해본다. UCBM과 Huang의 알고리즘은 SRAM으로 구현 가능하고 DIR-24-BASIC은 DRAM으로 구현 가능하다. SRAM으로 구현된 하드웨어의 경우 메모리 접근 지연시간 뿐만 아니라 로직에 걸리는 시간까지 고려해야 검색에 소요되는 시간을 알 수 있다. HD 0.65 um two-metal 3.3V CMOS technology를 사용할 때, 게이트 지연시간(gate delay)은 0.21ns이다. Huang의 알고리즘은 전체 로직 지연시간은 14.7.ns가 소요되고 UCBM은 7.77ns가 소요되며 DIR-24-BASIC은 0.21ns 가 소요된다. UCBM은 8 비트의 맵을 쓰는 반면 Huang의 알고리즘은 16 비트의 맵을 쓰고 전체적으로 덧셈기의 수와 비트가 크다. 메모리 접근 지연시간을 포함하면 Huang의 알고리즘은 44.7ns, UCBM 은 47.77ns, DIR -24-BASIC은 100.21ns 가 소요된다. 평균적인 검색 시간을 비교하기 위해 트래픽을 생성하여 모의 실험하였다. 라우팅 테이블이 차지하는 IP 주소 공간에서 랜덤하게 IP 주소를 골라 1,000,000 번의 검색을 수행하였다. [표 2]는 평균적인 검색 시간을 나타낸다.

Site	UCBM	Huang's	DIR-24-BASIC
Mae-East	16.00	19.58	50.21
Mae-West	15.22	18.67	50.21
AADS	15.83	19.38	50.21
Pac Bell	15.56	19.07	50.21
Paix	16.87	20.67	50.22

표2. 평균적인 검색 시간 비교, 단위 ns

UCBM가 Huang의 알고리즘보다 대략 3.5 ns 정도 빠르고 DIR-24-BASIC은 대략 50.2 ns 로 가장 느리다. 패킷의 크기가 1000 bits라고 가정할 때 평균적으로 UCBM은 52.4~62.5 Gb/s, Huang의 알고리즘은 48.4~59.3 Gb/s 로 6~17 Gb/s 만큼 성능이 뛰어나다. 결론적으로 UCBM은 최악의 경우에는 Huang의 알고리즘보다 검색 시간이 더 소요되지만 메모리의 요구량과 평균 검색 시간면에서 Huang의 알고리즘과 DIR-24-BASIC보다 뛰어나고 메모리의 일정부분만 갱신하므로 갱신 오버헤드가 작다.

5. 결론

본 논문에서는 유니 캐스트(unicast)를 위한 IP 주소 검색 기법인 UCBM 을 제안하였다. UCBM 은 CBM[3] 과 Huang 의 알고리즘을 수정하여 테이블 크기와 검색 시간을 최적화하고 점차적인 갱신이 가능한 알고리즘이다. UCBM 은 작은 크기의 SRAM 과 간단한 로직으로 구현되고 검색 시간이 작기 때문에 값싸고 빠른 전송 엔진을 만들 수 있다. 현재 인터넷에서 쓰이는 가장 큰 라우팅 테이블을 적용해도 560 KB 이하의 메모리로 구현가능하다. 10ns SRAM을 이용해서 구현할 경우 최고 경우 초당 100×10^6번의 검색이 가능하고 시뮬레이션 결과 평균적으로 초당 62.5×10^6번의 검색이 가능하다. 이는 기존의 알고리즘보다 빠르다.

6. 참고 문헌

[1] P. Gupta , S . Lin , and Nick Mckeown , "Routing Lookups in Hardware at Memory Access Speeds ", Proceedings of IEEE INFOCOM'98, pp 1240-1247

[2] V. Srinivasan , a nd George Varghese , "Faster address Lookups using Controlled Prefix Expansion ", ACM Transactions on Computer Systems, Vol 17, No 1, pp1-40, Feb, 1999

[3] M . Degermark , A . Brodnik , S. Carlsson , and S . Pink , "Small Forwarding Tables for Fast Routing Lookups ", Proceedings of the ACM SIGCOMM'97, Vol 27, No. 4, pp 3-14,Oct, 1998

[4] N. Huang, and S. Zhao, "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers ", IEEE JSAC., Vol 17, No 6, pp1093-1104, Jun, 1999

[5] Internet Routing Table Statistics, http://www.merit.edu/ipma/routing_table/