고속의 라우터를 위한 병렬 IP 주소 검색 기법

박 재 형[†] · 정 민 영[††] · 김 진 수[†††] · 원 용 관[††††]

요 약

인터넷에서 IP 패킷을 목적지로 전달하는 라우터는 목적지의 주소에 따라서 다음 홉을 결정하는 IP 주소를 검색하는 과정을 수행한다. 패킷을 전달하는 과정에서 주소 검색은 고속의 라우터의 설계에 중요한 요인이다. 본 논문에서는 이미 하드웨어로 구현된 간접 IP 주소 검색 칩셋의 변경 없이 고속의 라우터의 설계를 위해서 여러 개의 검색 엔진으로 구성된 병렬 주소 검색 기법을 제안한다. 또한, 라우터 시스템 전체 포워딩 테이블에 존재하는 IP 프리픽스 엔트리를 각각의 검색 엔진에 분할하는 규칙을 제시한다. 본 논문에서 제안한 주소 검색 기법의 성능은 IP 프리픽스에 대한 포워딩 정보를 저장하는데 필요한 메모리의 양과 포워딩 테이블을 구성하기 위해 필요한 메모리 접근 횟수로 평가하였다. 본 논문에서 제안한 기법은 한 개의 검색 엔진을 사용하였을 때에 비해서 네 개의 검색 엔진을 사용할 경우, 하드웨어 로직의 도움으로 약 35%의 메모리 양의 감소와 80%의 메모리 접근 횟수의 감소를 보여준다.

A Parallel IP Address Lookup Scheme for High-Speed Routers

Jaehyung Park[†] · Min Young Chung[††] · Jinsoo Kim[†††] · Yonggwan Won[††††]

ABSTRACT

In order that routers forward a packet to its destination, they perform IP address lookup which determines the next hop according to the packet's destination address. In designing high speed routers, IP address lookup is an important issue. In order to design high speed routers, this paper proposes a parallel IP lookup scheme which consists of several IP lookup engines without any modification of already fabricated indirect IP lookup chipsets. Also, we propose a simple rule for partitioning IP prefix entries in an overall forwarding table among several IP lookup engines. And we evaluate the performance of the proposed scheme in terms of the memory size required for storing lookup information and the number of memory accesses on constructing the forwarding table. With additional hardware logics, the proposed scheme can reduce about 30% of the required memory size and 80% of the memory access counts.

키워드 : 라우터(Router), IP 주소 검색(IP Address Lookup), 병렬 기법(Parallel Scheme), 간접 주소 검색 기법(Indirect Lookup Scheme), 프리픽스 분할(Prefix Partition)

1. 서 론

최근 인터넷 사용자의 급격한 증가와 더불어 인터넷 백본(backbone)망의 트래픽이 폭발적으로 증가함에 따라 고속의 라우터를 개발하기 위한 연구가 활발히 진행되고 있다. 또한, 다양한 서비스 품질을 요구하는 응용 서비스가 폭넓게 확산되고, 고속을 지원하는 가입자망(access network)이 빠르게 보급되면서 폭증하는 트래픽은 인터넷 백본 용량을 기가비트 이상 급으로 확장하도록 요구하고 있다[1]. 이와 같은 인터넷 백본망의 트래픽 증가는 백본 라우터에서의 병목 현상을 해소하기 위해 기가비트 이상의 트래픽을 처리할 수 있는 고속의 라우터 개발에 동기를 부여하고 있다[9].

라우터는 입력 인터페이스를 통해서 들어오는 패킷의 목적지를 참조하여 IP 패킷이 목적지로 전달되기 위한 출력 인터페이스를 결정한 후 패킷을 전달한다. 이러한 과정 중에 패킷 목적지 주소와 출력 인터페이스가 사상된 포워딩 테이블을 검색하는 작업은 CIDR(Classless Inter-Domain Routing)[6]이 등장하면서 입력된 IP 패킷의 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 최장 프리픽스 일치(longest prefix matching)를 찾는 작업이 되었다. 최장 프리픽스 일치를 찾는 과정은 IP 주소의 처음 비트부터 순차적으로 비교하여 가장 긴 주소가 일치하는 프리픽스를 찾아야 한다. 따라서 하나의 주소를 검색하는데 여러 번의 메모리 접근을 필요로 하므로 고속의 IP 패킷 전달을 지원하는 라우터에서는 중요한 병목 요인이 되고 있다[1, 9].

고속의 라우터를 구현하기 위해서는 최장 프리픽스 일치

※ 이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2003-003-D00255).
† 종신회원 : 전남대학교 전자컴퓨터정보통신공학부 교수
†† 종신회원 : 성균관대학교 정보통신공학부 교수
††† 종신회원 : 건국대학교 컴퓨터 · 응용과학부 교수
†††† 종신회원 : 전남대학교 정보통신공학부/의학부 교수
논문접수 : 2004년 7월 9일, 심사완료 : 2004년 9월 17일

를 보다 빠르게 찾는 기법이 필요하다. 고속의 IP 주소 검색을 지원하기 위해서 크게 두 가지 방향으로 연구가 진행되었다[15]. 하나의 연구 방향은 트라이 자료 구조에 기반을 둔 소프트웨어 방법으로 초창기 라우터에서 IP 주소 검색을 위해서 사용된 방법이다[5, 12, 13]. 소프트웨어 방법은 IP 주소 검색을 고속으로 처리하는데 한계가 있기 때문에 고속의 라우터를 구현하는데 어려움이 있다. 다른 하나의 연구 방향은 고속의 처리를 지원하는 하드웨어 기반 방법이다[4, 7, 8, 11, 14, 17]. 하드웨어 기반의 IP 주소 검색 기법을 구현하는데 있어서 성능을 측정하는 세 가지의 요인이 있다. 첫째는 임의의 IP 주소에 대한 출력 인터페이스를 포워딩 테이블에서 찾는데 소요되는 메모리 접근 시간을 줄이는 것이고, 둘째는 IP 주소에 대한 포워딩 테이블을 구성하는데 요구되는 메모리의 양을 적게 사용하는 것이다. 셋째는 포워딩 테이블에서 특정 IP 주소에 대한 정보의 변경이 용이하여야 한다는 것이다.

하드웨어 기반의 IP 주소 검색 기법 중에서, 가장 간단하고 직관적인 방법은 직접 IP 주소 검색 기법으로 IPv4의 주소 비트인 32비트의 모든 IP 주소로 구성된 메모리를 포워딩 테이블로 구성하는 방법이다[7]. 이와 같은 직접 IP 검색 기법은 처리하는 IP 패킷에 대한 목적지 IP 주소를 인덱스로 하여 포워딩 테이블을 단 한번의 메모리 접근으로 출력 인터페이스에 대한 정보를 얻을 수 있다. 그러나, 이 기법은 $4G(=2^{32})$개의 포워딩 엔트리를 저장할 수 있는 메모리가 필요하므로 구현하기 어렵다는 단점이 있다. 포워딩 테이블을 구성하는 메모리의 양을 줄이기 위해서 간접 IP 주소 검색 기법들이 제안되었다[4, 8, 14, 17]. 이러한 간접 IP 검색 기법들은 사용되는 메모리의 양을 줄이기 위해서 계층적 메모리 구조를 갖는 방법으로 구성되었다. 특히, IQ2200 칩셋[16]은 간접 IP 주소 검색 기법을 하드웨어로 구현된 칩으로 고속의 라우터를 개발하는데 사용하고 있다. Chung[3]은 IQ2200에 적용된 IP 검색의 성능을 향상시키는 방법을 제시하였으나, 고성능을 위해서는 이미 하드웨어로 구현된 IQ2200의 IP 검색 칩의 메모리 구조를 수정하여야 한다.

본 논문에서는 이미 구현된 칩을 수정하지 않고 고속의 IP 주소 검색을 지원하는 병렬 검색 기법을 제시한다. 제안된 병렬 IP 검색 기법은 간접 IP 주소 검색 기법이 적용된 여러 개의 IP 검색 엔진으로 구성되며, IP 프리픽스를 분할하는 방법에 따라서 특정 IP 주소를 여러 개의 IP 검색 엔진들 중에서 단 하나의 검색 엔진에 저장한다. 임의의 IP 주소에 검색과 수정이 분할 방법에 의하여 단 하나의 IP 검색 엔진에서 간접 IP 주소 검색 기법에 따라서 수행된다. 또한, 포워딩 테이블을 구성하는데 요구되는 메모리의 양과 소요되는 메모리 접근 횟수 측면에서 제안된 병렬 IP 검색 기법의 성능을 평가하였다. 본 논문에서 제시한 기법은 4개의 IP 검색 엔진을 사용하였을 경우, 35% 정도의 메모리 양의 감소와 80% 정도의 메모리 접근 횟수의 감소를 보였다. 제시된 기법은 이미 구현된 IQ2200 칩셋의 수정 없이 분할하는 방법을 구현하는 로직만을 추가하여 병렬 IP 검색 기법으로 적용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 IQ2200 칩셋에 하드웨어 방식으로 구현된 간접 IP 주소 검색 기법에 대해서 기술한다. 3장에서는 고속의 패킷 처리 가능한 라우터를 위한 병렬 IP 검색 기법과 포워딩 테이블의 IP 프리픽스를 분할하는 기법을 제안한다. 4장에서는 메모리 접근 시간과 메모리 요구량을 기준으로 제안된 병렬 IP 검색 기법의 성능을 평가하고, 마지막으로 5장에서 결론을 맺는다.

2. 간접 IP 주소 검색 기법

본 절에서는 IQ2200 칩셋에 구현된 간접 IP 주소 검색 기법의 전체적인 구조와 임의의 IP 주소에 대한 정보 검색 및 변경하는 방법에 대해서 기술한다.

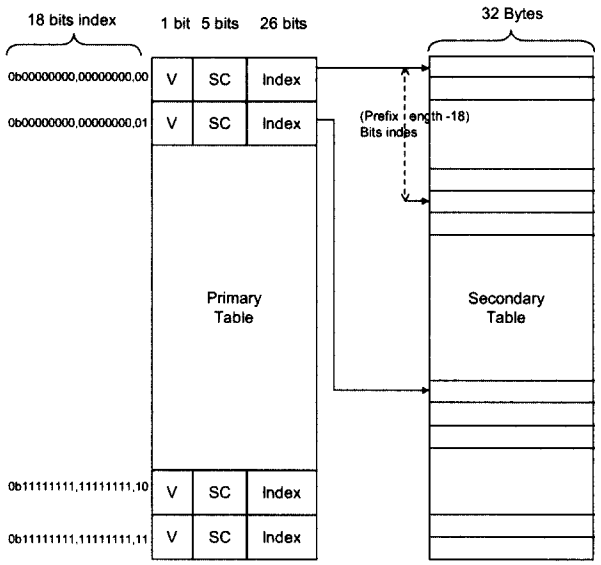
2.1 간접 IP 주소 검색 기법의 메모리 구조

IQ2200 칩셋에 구현된 간접 IP 주소 검색 기법은 대부분의 간접 IP 검색과 유사하게 포워딩 테이블은 두 단계의 메모리로 구성되어있다. 그 두 단계의 메모리는 각각 일차 메모리와 이차 메모리로 불리워진다. (그림 1)은 하드웨어 기반의 간접 IP 주소 검색 기법에서 일차 메모리와 이차 메모리에 대한 전체적인 구조를 보여준다. (그림 1)에서 일차 메모리의 크기를 결정하는 확장 비트 수, L_{exp}는 18이다.

포워딩 테이블을 구성하는 일차 메모리는 $2^{L_{exp}}$개의 포워딩 엔트리를 포함할 수 있으며, 4바이트 크기의 하나의 엔트리는 유효 비트(valid bit), 쉬프트 개수(shift count), 인덱스(index)라는 세 개의 정보로 구성된다. 유효 비트는 엔트리의 내용이 유효한지 그렇지 않는지를 가리키고, 쉬프트 개수는 검색하게될 IP 주소의 비트를 L_{exp} 부터 얼마까지 사용할 것인지를 나타낸다. 그리고, 인덱스는 L_{exp}까지 같은 IP 주소에 대해서 이차 메모리의 기본 주소 값을 의미한다.

포워딩 테이블을 구성하는 이차 메모리는 임의의 개수의 엔트리를 갖는 블록으로 구성된다. 각 블록을 구성하는 엔트리의 크기는 32바이트로 고정적이며, 포워딩 엔트리가 가져야하는 정보에 따라서 그 크기가 정해진다. 일반적으로 IP

주소 검색에 대한 정보는 다음 홉의 주소, 출력 인터페이스, 서비스 형태를 포함하고 있다.



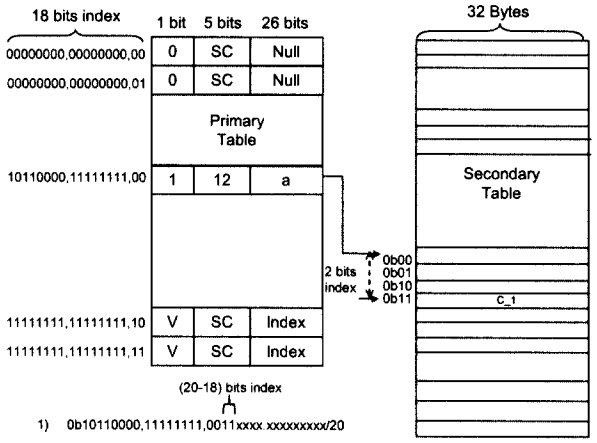
(그림 1) 간접 IP 검색 기법의 두단계 메모리 구조

2.2 간접 IP 검색 기법의 주소 검색 및 변경

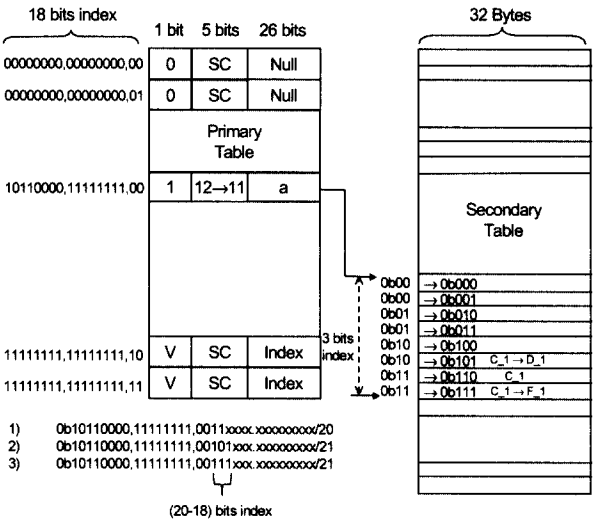
전달을 해야할 IP 패킷이 도착하면 IP 목적지 주소의 L_{exp} 비트까지 추출하여 얻은 값을 주소 값으로 지정하여 일차 메모리를 참조한다. 참조된 엔트리의 정보 중에서 유효 비트를 검사하여 무효로 지칭되었을 경우에는 라우터의 default router로 전송한다. 참조된 엔트리가 유효하다고 지칭되었다면, L_{exp} 비트부터 쉬프트 개수만큼의 비트들을 추출하여 얻은 값과 인덱스에 있는 이차 메모리의 기본 주소 값과 합한 다음 이차 메모리 주소를 참조하여 정보를 구한다. 이렇게 구한 정보에는 다음 홉의 주소 또는 출력 인터페이스 정보가 포함되어 있다. 이와 같이 임의의 IP 주소에 대한 정보를 검색하는 데에는 두 번의 메모리 접근이 요구된다.

라우터에서 수행되는 라우팅 프로토콜의 네트워크 정보가 변경됨에 따라서 포워딩 테이블을 구성하는 일차 메모리와 이차 메모리의 내용은 변경되어야 한다. 만약, 포워딩 테이블에 존재하는 엔트리의 다음 홉 또는 출력 인터페이스에 대한 정보만 변경된다면, 해당 엔트리에 대한 정보를 이차 메모리에서 찾은 후 그 엔트리의 내용만을 수정하면 된다. 그렇지만, 포워딩 테이블에 새로운 포워딩 엔트리를 추가하거나 기존 포워딩 엔트리를 삭제하는 경우에는 이차 메모리의 엔트리를 추가하거나 삭제하여야 한다. 그와 더불어 일차 메모리의 엔트리도 변경하여야 하는 경우가 발생한다. 포워딩 테이블을 구성하는 일차 메모리와 이차 메모리의 엔트리의 변경은 추가/삭제될 엔트리와 포워딩 테이블에 존재하는 엔트리간의 상관관계에 의해서 결정된다.

(그림 2)는 새로운 엔트리를 추가하는 과정을 보여준다. (그림 2)(a)는 20비트의 프리픽스가 0b10110000,11111111,0011이고 다음 홉 정보가 C_1인 엔트리를 추가한 결과를 보여준다. (그림 2)(b)는 21비트의 프리픽스가 0b10110000,11111111,00101이고 다음 홉 정보가 D_1인 엔트리와 21비트의 프리픽스가 0b10110000,11111111,00111이고 다음 홉 정보가 F_1인 엔트리 두개를 추가한 결과를 보여준다.



(a)



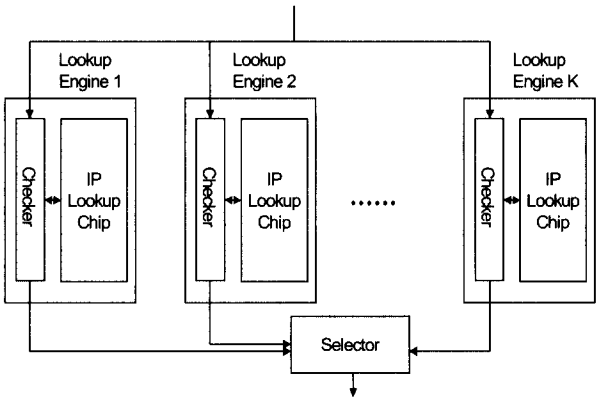
(b)

(그림 2) 포워딩 테이블에 새로운 엔트리를 추가하는 과정

3. 병렬 IP 주소 검색 기법

본 논문에서 제안하는 병렬 IP 주소 검색 기법은 (그림 3)에서와 같이 K개의 검색 엔진과 선택기로 구성되어 있다. 각각의 검색 엔진에는 IP 주소 검색을 수행하는 칩과 검사기로 이루어져 있으며, IP 주소 검색 칩은 하드웨어 기반의

간접 IP 검색 기법으로 구현되었다. 검사기는 입력으로 들어오는 패킷의 IP 목적지 주소에 대한 프리픽스가 자신의 IP 검색 엔진에 포함되어 있는지 그렇지 않는지를 판단한다. 자신의 IP 검색 엔진에 존재한다면, 해당 검색 엔진은 포워딩 엔트리에 대한 검색과 변경을 수행하고, 그렇지 않는 경우에는 검색 엔진의 동작을 중지시키고 널(NULL)을 출력한다. 최종적으로 선택기는 찾고자 하는 IP 주소에 대한 결과를 모든 검색 엔진으로부터 받아서 널이 아닌 정보를 추출한다.



(그림 3) 병렬 IP 주소 검색 기법의 구조

입력되는 IP 패킷의 목적지 주소로 포워딩 테이블에서 출력 인터페이스에 대한 정보를 얻는 과정은 해당 주소 프리픽스를 포함하고 있는 단 하나의 검색 엔진에서만 수행된다. 해당 검색 엔진에서는 2.2절에서 설명한 하드웨어 기반의 간접 IP 검색 기법에 의해서 정보를 획득한다. 또한, 임의의 포워딩 엔트리에 대한 추가/삭제를 포함하는 변경시에도 2.2절에서 설명한 방법에 의해서 수행된다.

본 논문에서 제안한 병렬 IP 주소 검색 기법에서 IP 프리픽스에 대한 연산을 수행하기 위해서는 포워딩 테이블을 구성하는 엔트리들을 K개의 검색 엔진으로 분할하는 방법이 필요하다. 이와 같은 IP 포워딩 엔트리를 분할하기 위해서 IP 주소 32비트 중에서 k개의 비트들을 추출하여 분할하는 방법을 사용한다. 이 때, K는 2^k이다. 이러한 분할 방법에 의해서 검사기는 단지 IP 주소 중에서 k개의 비트들만을 추출하여 검사하면 해당 주소가 자신의 검색 엔진에 존재하는지 그렇지 않는지를 판별할 수 있다.

이러한 분할 방법에 따르면, 하나의 IP 검색 엔진에 존재하는 모든 포워딩 엔트리의 k개의 비트들은 동일하므로 간접 IP 주소 검색 기법에 의미 없는 비트들이다. 그러므로, 임의의 포워딩 엔트리에 대한 검색과 변경을 수행하는 경우에 일차 메모리 또는 이차 메모리의 주소를 찾는데 사용되지 않는다. 그렇게 하기 위해서는 검사기에서 자신의 IP 검색 엔진에 해당 IP 주소가 존재하는지 판별하여 존재할 경우에 k개의 비트들을 제거한 비트열로 간접 IP 주소 검색을 수행하여야 한다. k개의 해당하는 비트열을 제거하는 기능을 포함하는 검사기를 사용하면, 이미 구현된 IQ2200의 간접 IP 주소 검색 칩셋을 변경하지 않아도 된다.

4. 성능 평가

본 장에서는 제안한 병렬 IP 주소 검색 기법의 성능을 포워딩 엔트리들을 저장하는데 요구되는 메모리의 양과 포워딩 테이블을 구성하는데 소요되는 메모리 접근 시간 측면에서 분석한다. 라우팅 테이블을 구성하는데 소요되는 메모리 접근 시간은 포워딩 엔트리를 저장하는데 소요되는 메모리 접근 횟수에 의존하므로 본 절에서는 메모리 접근 횟수 측면에서 분석한다. 또한, 실질적인 라우팅 테이블에 대해 제안한 병렬 IP 주소 기법의 성능을 평가한다.

4.1 수학적인 분석

포워딩 테이블을 구성하는데 필요한 메모리의 양과 소요되는 메모리 접근 횟수를 계산하기 위해서, 다음과 같은 용어를 정의한다. 윗첨자로 표현된 k는 각각의 k개 비트들의 값이 모두 동일한 IP 프리픽스를 나타낸다. 하나의 IP 검색 엔진의 확장 비트는 $L_{\exp}$이고 k개 비트의 위치는 모두 $L_{\exp}$보다 적다고 가정한다.

- N_n^k : 포워딩 테이블에 존재하는 프리픽스의 길이가 n인 엔트리의 개수
- $N_{n,D}^k$: 프리픽스의 길이가 n이고, 해당 프리픽스의 ($L_{\exp}+k$) 비트들이 포워딩 테이블에 존재하는 다른 엔트리의 프리픽스와 상이한 엔트리의 개수
- $N_{n,S,h}^k$: 프리픽스의 길이가 n이고, 해당 프리픽스의 ($L_{\exp}+k$) 비트들이 포워딩 테이블에 존재하는 다른 엔트리의 프리픽스와 동일한 엔트리의 개수, 이 때, h는 그 프리픽스들 중에서 가장 긴 프리픽스 길이를 지칭함.

동일한 k개 비트 값을 가지는 IP 검색 엔진 하나에 대한 일차 메모리의 크기는 식 (1)과 같이 계산된다. 이 때, B_{1st}는 일차 메모리의 엔트리 하나의 크기(바이트)를 나타낸다.

$$M_{1st}^{one} = 2^{L_{exp}} \times B_{1st}. \quad (1)$$

하나의 IP 검색 엔진의 이차 메모리의 크기를 구하기 위

해서는, 길이가 n이고 동일한 k개 비트 값을 가지는 IP 프리픽스에 대한 엔트리의 개수로부터 식 (2) 같이 계산된다.

$$M_{2nd}^{one}(n)=\begin{cases} N_n^k, & \text{if } k\le n\le L_{\exp}+k, \\ N_{n,D}^k\times 2^{\,n-L_{\exp}-k}, & \text{if } L_{\exp}+k<n\le 32. \end{cases} \quad (2)$$

식 (2)로부터 포워딩 테이블을 저장하는데 필요한 메모리의 크기는 모든 n에 대해서 $M_{2nd}^{one}(n)$의 총합으로 구할 수 있다. 그러므로 하나의 IP 검색 엔진에서 포워딩 엔트리를 저장하는데 요구되는 일차 및 이차 메모리의 크기는 식 (3)과 같이 계산된다. 이 때, B_{2nd}는 이차 메모리의 엔트리 하나의 크기(바이트)를 나타낸다.

$$M^{one}=2^{L_{\exp}}\times B_{1st}+\sum_{n=k}^{L_{\exp}+k}N_n^k\times B_{2nd}+\sum_{n=L_{\exp}+k+1}^{32}N_{n,D}^k \times 2^{\,n-L_{\exp}-k}\times B_{2nd}. \quad (3)$$

병렬 IP 검색 기법은 K개의 IP 검색 엔진으로 구성되어 있으므로, 전체적인 메모리의 크기는 모든 검색 엔진의 메모리 크기를 합한 것으로 표현된다. IP 프리픽스의 길이(n)이 k개의 비트들의 위치 값보다 적을 경우에는 여러 개의 IP 검색 엔진에 저장하여야 한다. 그렇지만, k개의 비트들의 위치 값을 프리픽스의 길이보다 적게 정하는 경우에는 요구되는 메모리의 총량은 변함이 없음을 보여준다.

포워딩 테이블에 있는 엔트리가 변경이 될 때마다, 특정 IP 검색 엔진의 메모리의 내용은 변경되어야 한다. 이와 같은 포워딩 엔트리의 변경에 소요되는 메모리 접근 횟수는 IP 프리픽스 길이, 확장 비트 크기, 저장할 프리픽스의 특성에 따라 결정된다. 병렬 IP 주소 검색 기법의 메모리 접근 횟수의 성능을 분석하기 위해서 최악의 상황에서 포워딩 엔트리의 추가/삭제를 수행할 경우에 대해서 고려한다.

길이가 n인 IP 프리픽스를 변경하는데 소요되는 일차 및 이차 메모리의 접근 횟수는 식 (4)와 식 (5)와 같이 계산된다.

$$A_{1st}^{one}(n)=\begin{cases} N_n^k\times 2^{\,L_{\exp}+k-n}, & \text{if } k\le n\le L_{\exp}+k, \\ \sum_{h=n+1}^{32}N_{n,S,h}^k, & \text{if } L_{\exp}+k<n\le 32. \end{cases} \quad (4)$$

$$A_{2nd}^{one}(n)=\begin{cases} N_n^k, & \text{if } k\le n<L_{\exp}+k, \\ \sum_{h=n+1}^{32}N_{n,S,h}^k\times 2^{h-n}, & \text{if } n=L_{\exp}+k, \\ N_{n,D}^k\times 2^{\,n-L_{\exp}-k}+\sum_{h=n+1}^{32}N_{n,S,h}^k\times 2^{h-n}, & \text{if } L_{\exp}+k<n\le 32. \end{cases} \quad (5)$$

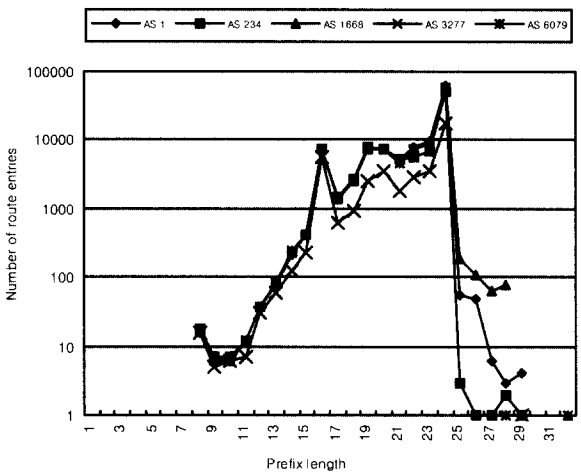
식 (4)와 식 (5)로부터 포워딩 테이블에서 임의의 IP 프리픽스에 대한 정보를 추가/삭제하는데 소요되는 총 메모리 접근 횟수는 식 (6)과 같이 계산된다.

$$A^{one}=\sum_{n=1}^{32}[A_{1st}^{one}(n)+A_{2nd}^{one}(n)]. \quad (6)$$

제안한 병렬 IP 주소 검색 기법의 포워딩 테이블에서 엔트리의 추가/삭제를 수행하는데 소요되는 메모리 접근 횟수는 모든 IP 검색 엔진에 균등하게 분산되어 있을 경우, 하나의 IP 검색 엔진에서 소요되는 횟수와 동일하다. 그러므로 포워딩 테이블에서 엔트리를 변경하는데 필요한 메모리 접근 횟수는 A^{one}과 동일하다.

4.2 실험 결과

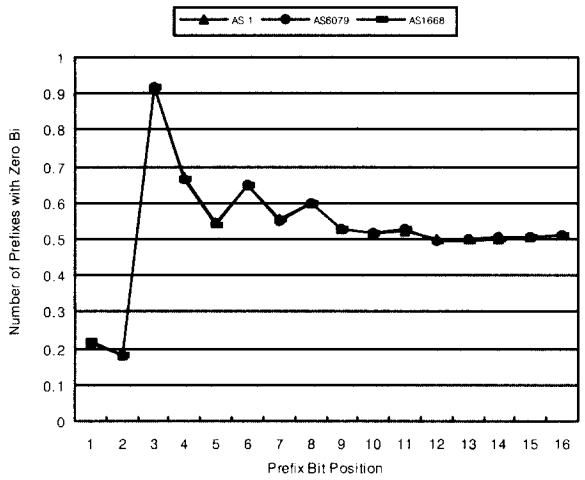
제안된 병렬 IP 주소 검색 기법의 포워딩 테이블 구성에 필요한 메모리의 양과 접근 횟수를 측정하기 위해서 서로 5개의 AS들로부터 얻은 라우팅 테이블을 사용하여 실험하였다[2]. IQ2200 칩셋에 구현된 간접 IP 검색 기법에와 마찬가지로 확장 비트의 수는 18이고, 일차 메모리의 엔트리의 크기인 B_{1st}는 4바이트, 이차 메모리의 엔트리의 크기인 B_{2nd}는 32바이트를 가정한다.



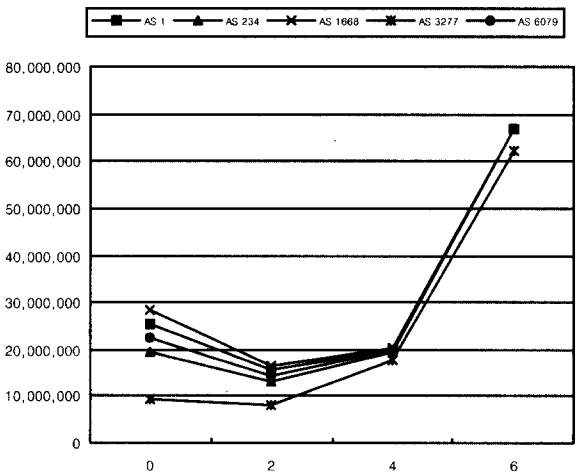
(그림 4) 5개의 AS에 대한 IP 프리픽스 길이의 분포도

(그림 4)는 5개의 AS의 라우팅 테이블에 포함되어 있는 IP 프리픽스 길이의 분포를 보여준다. AS 1, 234, 1668, 3277, 6079에서 프리픽스의 길이가 24이하인 것의 비율이 거의 99.9%에 가깝고 프리픽스의 길이가 8이하인 것은 모든 AS에 대해서 존재하지 않는다. 결과적으로, 8이하인 값으로 k개의 비트들의 위치를 선택하면 메모리의 크기가 증가하지 않는다는 것을 보여준다. (그림 5)는 AS 1, 1668, 6079에

포함된 IP 프리픽스에 대해서 임의의 비트 위치에서 그 값이 0인 경우의 비율을 보여준다. IP 프리픽스들 중에서 첫 번째와 두 번째 비트의 값이 0인 비율은 20% 이하이고, 세 번째 비트의 값이 0인 비율은 90% 정도이다. (그림 5)에서 보는 바와 같이, IP 프리픽스에서 임의의 비트 위치의 값이 0인 비율은 비트의 위치가 8보다 적은 경우 대부분이 불균형적으로 분포된다는 것을 보여준다. 그러나, 비트 위치가 9 이상인 경우에는 거의 50%에 이르러 균등하게 분할될 수 있음을 보여준다. 그러므로, 본 실험에서는 비트의 위치를 9 이상인 값으로 선택한다. 예를 들어, k가 2인 경우 12와 13 번째를 선택하여 실험하였다.

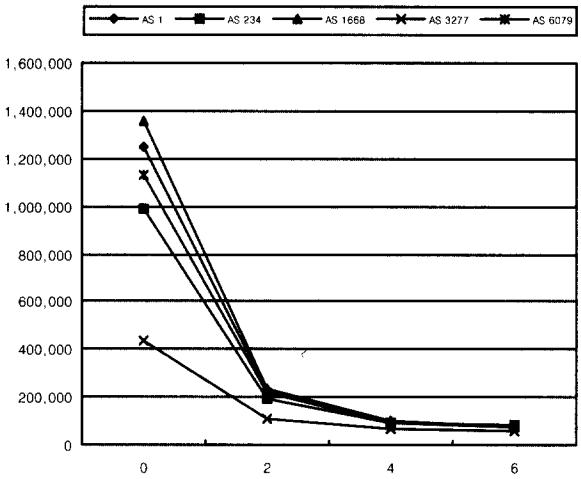


(그림 5) AS에 포함된 IP 프리픽스의 비트 위치에 대해서 0값을 갖는비율



(그림 6) 포워딩 테이블을 저장하는데 요구되는 메모리의 양(Bytes)

(그림 6)은 제안한 병렬 IP 검색 기법의 검색 엔진의 개수를 결정짓는 k의 값을 변경하면서 각각의 AS에 대한 포워딩 테이블을 저장하는데 필요한 메모리의 크기를 보여준다. (그림 6)에서 보는 바와 같이, 일차 메모리를 참조하는데 사용하는 IP 프리픽스의 비트열이 커져서 확장되는 프리픽스의 개수가 증가하여 k의 값이 4보다 큰 경우에는 메모리의 크기가 오히려 증가한다. 특히, k의 값이 2인 병렬 IP 주소 검색 기법과 하나의 검색 엔진으로 구현된 IQ2200 칩셋($k=0$)과 비교하면, 각각의 AS 라우팅 테이블을 저장하는데 필요한 메모리의 크기가 약 35% 정도 감소한다.



(그림 7) 포워딩 테이블의 엔트리 변경에 소요되는 메모리 접근 횟수(count)

(그림 7)은 제안한 병렬 IP 검색 기법에서 k의 값을 변경하면서 포워딩 테이블에서 임의의 엔트리의 추가/삭제를 수행하는데 소요되는 메모리 접근 횟수를 보여준다. k의 값이 증가함에 따라, 소요되는 접근 횟수는 감소한다. 특히, k의 값이 2인 병렬 IP 주소 검색 기법은 IQ2200 칩셋에 구현된 기법에 비해서 80% 정도의 횟수가 감소함을 보여준다. (그림 6)과 (그림 7)의 결과를 보면, 4개의 IQ2200 칩셋의 검색 엔진으로 구성한 병렬 IP 주소 검색 기법이 임의의 비트열을 검사하고 제거하는 검사기와 특정 결과를 선택하는 선택기가 필요한 반면, 메모리 양과 메모리 접근 횟수 측면에서 성능이 우수함을 보여준다.

5. 결 론

IP 트래픽의 폭발적인 증가로 인하여 인터넷에서는 고속의 패킷 처리가 가능한 라우터가 요구되고 있다. 본 논문에서는 고속의 라우터를 위한 병렬 IP 주소 검색 기법을 제시하였다. 제시된 병렬 IP 검색 기법은 여러 개의 간접 IP 검색 기법을 사용하는 검색 엔진으로 구성되어 있다. IP 프리픽스에 대한 검색은 특정 하나의 IP 검색 엔진에서 수행되

도록 지원하기 위해서 IP 프리픽스를 여러 개의 IP 검색 엔진으로 분할하는 방법도 제시하였다. 본 논문에서 제시한 병렬 IP 검색 기법의 성능은 4개의 검색 엔진을 가지는 경우에 기존의 IQ2200 칩셋에 구현된 방법에 비해서 비트열 검사기와 결과 선택기의 하드웨어 로직이 필요한 반면, 약 35%의 메모리 양이 감소하고 약 80%의 메모리 접근 횟수가 감소하여 성능이 우수함을 보여주었다. 제시된 병렬 IP 검색 기법은 이미 구현된 IQ2200 칩셋의 변경 없이 확장하여 적용할 수 있는 장점이 있다. 향후 간접 IP 검색 기법을 사용한 병렬 IP 검색 기법에 적용할 수 있는 포워딩 엔트리의 추가 및 삭제가 용이한 구조에 대한 연구가 필요할 것이다.

참 고 문 헌

[1] J. Aweya, "On the Design of IP Routers Part 1 : Router Architectures," Journal of Systems Architecture, Vol.46, No.6, pp.483-511, Apr., 2000.

[2] BGP Reports, http://bgp.potaroo.net.

[3] M. Y. Chung, J. Park, B. J. Ahn, N. Ko and J. H. Kim, "Performance Analysis of an IP Lookup Algorithm for High Speed Router Systems," Lecture Note on Computer Science, Vol.2662, pp.35-45, 2003.

[4] M. Degermark, A. Bordnick, S. Carlsson and S. Pink, "Small Forwarding Tables for Fast Routing Lookups," Proceedings of ACM SIGCOMM, pp.3-14, 1997.

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

[6] V. Fuller, T. Li, J. Yu and K. Varadhan, "Classless Inter-Domain Routing (CIDR) and Address Assignment and Aggregation Strategy," RFC1519, Sep., 1993.

[7] P. Gupta, S. Lin and N. Mckweon, "Routing Lookups in Hardware at Memory Access Speeds," Proceedings of IEEE INFOCOM, pp.1240-1247, 1998.

[8] N.-F. Huang and S.-M. Zhao, "A Novel IP Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers," IEEE Journal of Selected Areas in Communications, Vol.17, No.6, pp.1093-1104, 1993.

[9] S. Keshav and R. Rharma, "Issues and Trends in Router Design," IEEE Communications Magazine, Vol.36, No.5, pp.144-151, May, 1998.

[10] S. Jean, S.-H. Chung, J. W. Cho and H. Yoon, "A Scalable and Small Forwarding Table for Fast IP Lookups," Proceedings of Int'l Conference on Computer Networks and Mobile Computing, pp.413-418, 2001.

[11] A. J. McAuley and P. Francis, "Fast Routing Table Look-up using CAMs," Proceedings of IEEE INFOCOM, pp. 1382-1391, 1993.

[12] D. Morrison, "PATRICIA-Practical Algorithm To Retrieve Information Coded In Alphanumeric," Journal of ACM, Vol.5, No.4, pp.514-534, Oct., 1968.

[13] S. Nilsson and G. Karlsson, "IP-Address Lookup using LC-Tries," IEEE Journal of Selected Areas in Communications, Vol.17, No.6, pp.1083-1092, Jun., 1999.

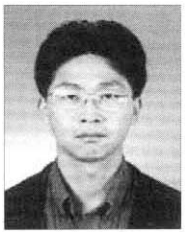
[14] V. Srinivasan and G. Varghese, "Fast Address Lookups using Controlled Prefix Expansion," ACM Trans. on Computer Systems, Vol.17, No.1, pp.1-40, Feb., 1999.

[15] M.A. Ruiz-Sanchez, E.W. Biersack and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," IEEE Network, Vol.15, No.2, pp.8-23, 2001.

[16] Vitesse Semiconductor Co., "Design Manual : IQ2200, Family of Network Processors," 2002.

[17] P.-C. Wang, C.-T. Chan and Y.-C. Cheng, "A Fast IP Routing Lookup Scheme," Proceedings of ICC, pp.1140-1144, 2000.

박 재 형



e-mail : hyeoung@chonnam.ac.kr

1987년~1991년 연세대학교 전산학과 (학사)

1991년~1993년 KAIST 전산학과(석사)

1993년~1997년 KAIST 전산학과(박사)

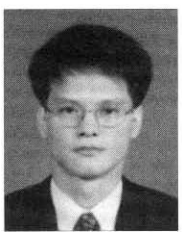
1997년~1998년 KAIST 인공지능연구센터 Post-Doc 연수연구원

1998년~2002년 ETRI 네트워크연구소 선임연구원

2002년~현재 전남대학교 전자컴퓨터정보통신공학부 조교수

관심분야 : MPLS, 인터넷 프로토콜, 인터넷 시스템, 라우터 구조, 이더넷 네트워크 연결망, 병렬처리, 멀티캐스트

정 민 영



e-mail : mychung@ece.skku.ac.kr

1986년~1990년 KAIST 전기및전자공학과 (학사)

1992년~1994년 KAIST 전기및전자공학과 (석사)

1994년~1999년 KAIST 전기및전자공학과 (박사)

1999년~2002년 ETRI 네트워크연구소 선임연구원

2002년~현재 성균관대학교 정보통신공학부 조교수

관심분야 : 차세대 인터넷, 광/홈/Ad hoc 네트워크, 이동통신망 성능 및 신뢰성 분석

김 진 수

e-mail : jinsoo@kku.ac.kr
1979년~1983년 서울대학교 컴퓨터공학과 (학사)
1983년~1985년 KAIST 전산학과(석사)
1993년~1998년 KAIST 전산학과(박사)
1985년~2000년 한국전기통신공사 선임 연구원
2000년~현재 건국대학교 컴퓨터·응용과학부 조교수
관심분야 : Interconnection Network, Parallel Computer Systems, Network Systems, Network Security, Mobile Computing

원 용 관

e-mail : ykwon@chonnam.ac.kr
1986년 한양대학교 전자공학과 학사
1991년 Univ. of Missouri, 컴퓨터공학 석사
1995년 Univ. of Missouri, 컴퓨터공학 박사
1991년~1995년 Univ. of Missouri, Research/Teaching Assistant
1995년~1996년 한국전자통신연구원
1996년~1999년 한국통신 연구개발본부
1999년~현재 전남대학교 정보통신공학부/의학부 부교수
2002년~현재 한국생물정보학회 Proteome Informatics 연구회장
관심분야 : 네트워크관리 및 보안, 컴퓨터비젼, 바이오정보학