논문 01-26-4A-19 한국통신학회논문지 '01-4 Vol.26 No.4A

고속 LPM 탐색을 위한 파이프라인CAM 구조

정회원 안 희 일*, 조 태 원*

PICAM:A Pipelined Architecture of CAM for high speed LPM Searching

Hee Il Ahn*, Tae Won Cho* *Regular Members*

요 약

라우터의 인터넷 패킷 처리에서 가장 많은 시간이 걸리는 부분이 IP(Internet protocol) 패킷 주소 룩업(address lookup) 중 LPM(longest prefix matching) 탐색이다. 기존의 LPM 탐색 방법은 라우터의 룩업 테이블의 갱신속도에 대한 고려 없이, LPM 탐색율만 높이는데 주력해왔다. 룩업 테이블의 느린 갱신속도로 인해 룩업정지(lookup blocking) 또는 시효가 지난 경로(obsolete route)에 의한 부정확한 라우팅이 발생할 수 있어, 고속의 라우팅에서는 갱신시간이 짧은 LPM 탐색방법이 절실히 필요하게 되었다. 특히 기존 CAM(content addressable memory, 내용주소화 메모리)을 이용한 LPM 탐색에서는 LPM 탐색율이 높으면서 동시에 복잡도도 높지 않은 방식은 룩업 테이블의 갱신시간이 O(n)으로 오래 걸렸다. 본 논문에서는 룩업 테이블의 갱신시간이 O(1)으로 짧으면서도, LPM 탐색율이 높고, 복잡도도 높지 않은 새로운 방식인 파이프라인 CAM 구조(PICAM)를 제안한다.

ABSTRACT

LPM searching in IP address lookup is a major bottleneck of IP packet processing in high speed router. Existing LPM methods are focused on only lookup speed without considering lookup table update. So their slow update can lead to lookup blocking or wrong routing decision based on obsolete routes. Especially existing LPM methods based on CAM(content addressable memory) have slow update of O(n) cycles in spite of their high throughput and low area complexity on LPM searching. In this paper we proposes PICAM, a new pipelined CAM architecture for fast update of O(1) cycle of lookup table and high throughput and low area complexity on LPM searching.

Ⅰ. 서 론

인터넷 패킷 통신에서 패킷의 행선지 주소(destination address)에 따라 패킷이 전달될 출력 포트를 결정하는 것을 IP 주소 룩업(address lookup)이라 한다. IP 주소 룩업은 인터넷 통신에서 패킷 처리 능력을 결정하는 중요한 요소이다. 인터넷의 속도는 통신선로가 아무리 빠르더라도 패킷을 행선지 주소에 따라 적절한 곳(출력포트)으로 라우팅 해주는 IP 주소 룩업 처리 능력이 작으면, 고속의 인터넷 통신이 불가능 해진다. 최근 기가비트급이상의 초고속 인터넷 통신에서는 IP 주소 룩업 처리 능력이 병목현상을 일으키고 있다.

인터넷 통신의 초기에는 IP 주소는 3가지 class가 있었고, 각각 8 비트, 16비트 및 24비트의 고정길이의 IP 주소를 사용하였다. 고정길이의 IP 주소의 룩업에는 패킷의 목적지 IP주소와 정확하게 일치하는 엔트리를 룩업 테이블(lookup table)에서 찾아내는 EM(exact matching) 탐색을 사용하였다[1, 2]. EM 탐색은 perfect hashing, binary search 또는 일반적인 CAM을 사용해 빠르게 수행할 수 있었다. 특히 EM 탐색은 하드웨어적으로 일반적인 CAM을 사용

* 충북대학교 전기전자공학부
논문번호 : K01048-0201, 접수일자 : 2001년 2월 1일

하면 한 번의 메모리 접근으로 EM 탐색을 할 수 있어 매우 빠르게 IP 주소 룩업을 수행할 수 있다. 또한 룩업 테이블의 갱신도 일반적인 CAM을 사용하면 한번의 메모리 접근으로 완료할 수 있어 매우 빨랐다.

고정길이의 IP 주소로 인해 IP 주소를 많이 낭비되는 문제가 발생하였으며, 1990년대 초반에 CIDR (classless interdomain routing)이 도입되었다. 이에 따라 IP 주소는 가변길이를 갖게 되었다. 라우터에서 동일한 출력포트(output port)로 도달될 수 있는 IP 주소들을 하나의 그룹으로 묶어서 하나의 prefix로 표현하여 라우터 룩업 테이블의 한 엔트리로 사용함으로써, 룩업 테이블의 크기를 대폭 줄일 수 있게 되었다. 그러나 가변길이의 IP 주소 탐색을 하려면, EM 탐색에 비해 시간이 많이 걸리는 LPM 탐색을 수행해야 하는 문제점이 발생했다[7, 13].

CIDR을 사용하면 라우터의 룩업 테이블은 {IP 주소/매스크길이, 출력포트}의 triple로 표현되는 엔트리들로 구성된다. 여기서 매스크길이는 prefix의 길이를 나타내며, IP주소/매스크길이는 IP route를 나타내는 prefix이다. 만일 입력된 패킷의 IP주소가 128.32.195.1이라 할 때, 룩업테이블에서 매치되는 엔트리를 {128.32.1.5/16, port1}, {128.32.225.0/18, port3}, {128.0.0.0/8, port5}이라 하자. 이들 중 왼쪽으로부터 제일 길게 매치되는 엔트리는 두 번째의 {128.32.225.0/18, port3}이며, 이를 LMP(longest matching prefix) 또는 BMP(best matching prefix)라 부른다. 이 LMP를 라우터 룩업 테이블에서 찾는 작업을 LPM 탐색이라 한다. IP 패킷을 라우팅할 때는 룩업 테이블에서 이 LMP를 찾아 그에 해당하는 출력 포트인 port3로 입력패킷을 출력함으로써 라우팅을 수행한다.

LPM 탐색의 어려움은 입력 패킷의 IP 주소의 prefix 길이를 미리 알 수 없다는데 있다. 더구나 입력 패킷에 대해 여러 개의 매치 가능한 prefix중 가장 길게 매치되는 prefix를 찾아내야 하기 때문에 탐색시간이 오래 걸려, 고속 IP 주소 룩업에서 LPM 탐색은 병목현상을 일으키고 있다. 또한 LPM 탐색에는 perfect hashing, binary search 또는 일반적인 CAM과 같은 EM 탐색에 사용하던 방법들을 적용할 수 없게 되었다.

백본 라우터(backbone router)에 사용되고 있는 룩업 테이블의 엔트리의 개수는 45,000개 정도인 것으로 알려져 있다[14]. 앞으로 계속 증가 추세가 이어질 것으로 보인다. 인터넷상에서 라우터의 룩업 테이블은 계속 갱신되고 있다. 조사에 의하면 초당 1,000번 이상의 룩업 테이블 갱신이 발생하기도 한다. 신속한 룩업 테이블의 갱신이 이루어지지 않으면, 룩업 정지(lookup blocking)가 발생하거나, 시효가 지난(obsolete) 루우트에 의한 부정확한 라우팅이 발생할 수 있다. 연달아 발생하는(consecutive) 두 IP 주소 룩업 동안에 룩업 테이블 갱신이 완료되지 않으면 나중에 수행되는 IP 주소 룩업은 부정확한 라우팅을 야기시킬 수 있다. 룩업 테이블의 갱신속도가 빠르면, 이러한 문제들이 해결될 수 있다[13].

기존의 LPM 탐색 방법에는 소프트웨어적인 방법과 하드웨어적인 방법들이 있다. 소프트웨어적인 방법은 Trie 또는 binary search를 기반으로 하여 LPM 탐색을 처리하는 방법들이다. 소프트웨어적인 방법들은 여러 차례 메모리를 참조해야 하기 때문에 처리 능력이 낮다. 라우터의 룩업 테이블을 압축하여 캐쉬나 main memory에 탑재하여 속도를 올리고 있으나, 룩업 테이블의 갱신을 쉽게 할 수 없다는 단점이 있다[2, 4, 5, 6, 7, 8, 9, 10, 11]. 룩업 테이블의 갱신에 보통 수백ms나 걸리며, 룩업 테이블의 크기가 매우 큰 백본 라우터에서는 수초의 시간이 걸리기도 한다[8, 11]. 하드웨어적인 방법은 CAM을 이용한 방법이 있다[1, 12]. 그러나 이 방법들도 갱신할 때마다 룩업 테이블의 갱신이 쉽지 않거나 복잡도가 너무 높다는 갖고있다[1].

IP 주소 룩업에서 중요한 것은 LPM 탐색 처리율이 높으면서 동시에 룩업 테이블의 갱신 속도가 빨라야 하며, 아울러 복잡도도 높지 않아야 한다는 점이다. 그러나 기존의 LPM 탐색에 대한 연구들에서는 룩업 테이블의 효율적인 갱신은 고려하지 않고, LPM 탐색의 처리능력의 향상에만 초점을 맞춰왔다[2, 4, 5, 13].

CAM은 RAM에 비해 집적도가 떨어지고, LPM의 경우에는 특별한 구조의 CAM을 사용해야 한다는 단점 때문에, CAM을 이용한 방법에 대한 연구는 활발하지 않았다. 그러나 최근 반도체기술의 계속적인 발전으로 CAM을 적용할 정도로 직접도가 높아졌다.

따라서 본 논문에서는 룩업 테이블의 갱신속도가 높으면서도 LPM 탐색율이 높으며, 또한 복잡도도 높지 않은 새로운 고속 LPM 탐색을 위한 CAM구조(PICAM)를 제안한다. II장에서는 LPM 탐색에 CAM을 이용한 기존연구에 대해 살펴본 다음, III장에서 새로운 방법인 PICAM을 기술하고, IV장에서는 PICAM을 기존 방법들과 비교 분석하겠다.

Ⅱ. 기존 CAM을 이용한 LPM 탐색 방법

CAM을 이용한 IP 주소 룩업 방법에 대해 McAuley[1, 12]가 B1, B2, B3, T1, T2 및 T3의 6가지 방법을 제안했다. B1, B2, B3는 binary CAM을 사용하는 방법이고, T1, T2, 및 T3는 ternary CAM을 이용하는 방법이다. B1은 일반적인 CAM을 사용해 EM 탐색을 수행하는 방법이며, 그 외의 방법은 LPM 탐색에 적용할 수 있는 방법이다. T2는 B2보다 2배의 셀을 필요로 하면서도 탐색 처리능력에서는 B2와 비슷하고, T3는 룩업 테이블의 갱신속도를 개선한 방법이나 B3보다 2배의 셀이 필요한 복잡도가 매우 높은 방법이다. 따라서 본 논문에서는 B2, B3 및 T1에 대해서만 자세히 살펴보고자 한다.

2.1 B2방법

일반적인 CAM 구조를 그대로 이용해 비교적 쉽고 간단하게 구성할 수 있는 반면에, 여러 사이클에 걸쳐 LPM 탐색을 수행하기 때문에 탐색율이 저하되는 방법이다. IP 패킷의 LPM 탐색 수행에는 최악의 경우에 IP 주소 길이에 해당하는 수(IP version 4인 경우 32, IP version 6인 경우 128)의 사이클이 걸릴 수 있다[1]. 이 방법은 패킷 처리율이 매우 낮으므로 고속의 IP 패킷의 라우팅에는 적용할 수 없는 방법이다.

● 룩업 테이블 구성

룩업 테이블의 각 엔트리는 {IP 주소/매스크길이, 출력포트}이다. IP 주소를 왼쪽부터 매스크 길이에 해당하는 비트까지는 IP 주소 패턴을 취하고, 그 이후의 비트부터 끝까지는 '0'으로 세트한다. 이러면 prefix를 제외한 나머지 비트는 '0'이 된다. 예를 들어 ..., {128.32.1.5/16, port1}, ..., {128.32.225.0/18, port3}, ..., {128.0.0.0/8, port5}, ... 의 룩업 테이블은 그림 1와 같이 매핑된다. 키필드에는 prefix들이 들어가고, 데이터필드에는 포트번호들이 해당되는 행에 저장된다. {128.32.1.5/16, port1}에 해당하는 키필드의 주소표현은 왼쪽에서 16비트까지는 "128.32.1.5의 2진표현 중 왼쪽에서 16번째 비트까지의 비트패턴인 '10000000 00100000'을 취하고 나머지 16비트는 '0'으로 채워 만든다. 이와 같은 방법으로 나머지 엔트리에도 적용하여 모든 키필드의 데이터를 구성한다.

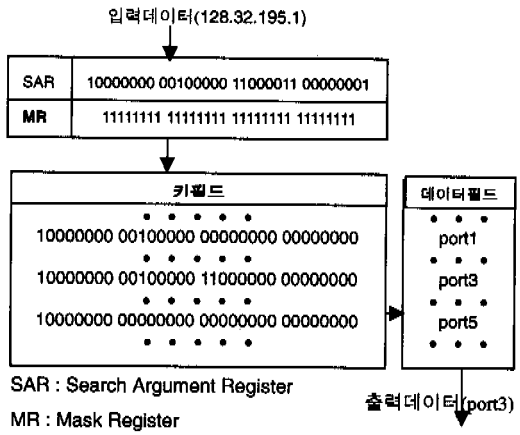

그림 1. B2방법의 룩업 테이블 구성 예

● LPM 탐색과정

LPM 탐색과정을 알고리즘으로 표현하면 그림 2와 같다. 입력 패킷의 IP 주소가 그림 1과 같이 '128.32.195.1'이라 하면, 이 주소의 이진표현 '10000000 00100000 11000011 00000001'를 SAR에 가하고 MR에 제일 긴 매스크인 '11111111 11111111 11111111 11111111'과 같은 패턴을 가한 후 탐색을 한다. 만일 히트 되면 탐색을 중단하고 그 워드에 해당하는 포트 번호를 출력하게 되고, 히트가 되지 않으면 MR에 앞보다 매스크 길이가 하나 짧은 즉 앞의 매스크를 shift left한 패턴을 가한 후

```
procedure  LPM_search_B2
begin
    SAR := input_packet_addr;  // load input address //
    MR := FFFFFFFF;                          // maximum mask length //
    repeat
        search_keyfield;                     // search CAM array //
        shift_left(MR);                      // reduce mask length by 1 //
    until (HIT = 1  or  MR = 0);
    output(hit_data);                  // output the result //
end; {LPM_search}
```

그림 2. B2방법에서의 LPM 탐색과정

위의 탐색과정을 반복한다. 이런 과정을 히트 될 때까지 반복한다. 그러면 제일 길게 매치되는 데이터가 제일 먼저 히트하게 되어 위의 과정이 끝나게 되며, 이 때 히트 된 워드가 찾고자 하는 패턴과 왼쪽부터 가장 길게 매치되는 워드가 되며, 이 워드가 LMP가 된다[1]. 그림 1에서는 생략되지 않고 명확하게 표현된 줄 중에서 2번째 줄이 되며 IP 패킷을 port 3으로 출력함으로써 LPM 탐색과정이 종료된다.

2.2 B3 방법

이 방법은 그림 3와 같이 여러 개의 CAM 모듈을 병렬로 동시에 동작시켜 입력 IP 주소와 매치되는 엔트리를 한 사이클에 찾는 방법이다. 한 사이클에 찾을 수 있는 반면에 키필드의 폭이 각기 1, 2, ..., w인 CAM모듈이 w개 필요하다. IP version 4인 경우 w=32, IP version 6인 경우 w=128이 된다. B3는 룩업 테이블을 갱신할 때마다 엔트리들을 매스크 길이별로 분류하여 해당되는 CAM모듈에 저장하나, 각 CAM모듈내의 엔트리들은 정렬되지 않아도 된다[1].

● 룩업 테이블의 구성

그림 3 및 그림 4와 같이 룩업 테이블의 각 엔트리들을 매스크 길이 별로 분류해, 각기 매스크 길이별로 나뉘어지는 CAM모듈의 키필드에 각 엔트리의 왼쪽에서 매스크길이에 해당하는 비트패턴을 저장한다. 룩업 테이블의 구성에 필요한 CAM모듈은 IP version 4인 경우는 32개이고 IP version 6인 경우에는 128개나 소요된다[11].

● LPM 탐색 과정

그림 3에서는 외부에서 들어오는 입력 패킷의 IP 주소가 동시에 모든 CAM모듈에 가해진다. 그림 4에서 보는 바와 같이 8비트 CAM 모듈, 16비트 CAM 모듈, 18비트 CAM모듈만이 들어온 IP 패킷 주소와 일부 매치되는 패턴을 갖고 있고, 나머지는 매치되는 패턴이 없다.

매치되는 CAM모듈의 출력은 각각 port 5, port 1, port 3 이다. 이들 출력들 중에 우선순위처리기는 매스크길이가 긴 CAM 모듈에서 나오는 출력을 선택하여 최종적으로 IP 패킷을 port3으로 출력한다[1, 12]. 탐색과정을 알고리즘으로 표현하면 그림 5와 같다.

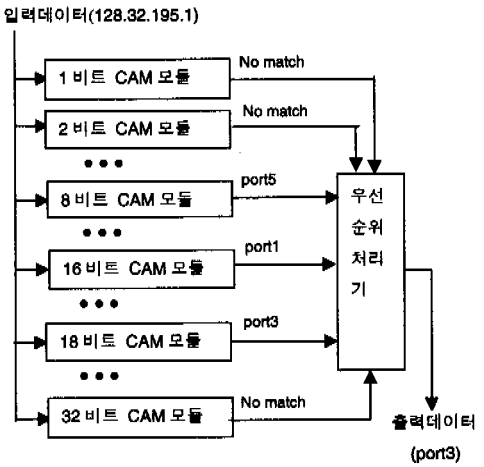

그림 3. B3방법의 룩업 테이블 구성 및 LPM 탐색 예

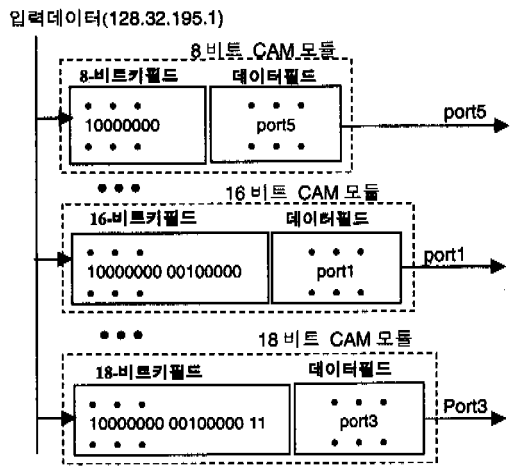

그림 4. B3방법의 CAM 모듈 내에서의 룩업 테이블 예

```
procedure  search_CAM_module(var k)
begin
     SAR := input_packet_addr;   // load input address //
     search CAM(k);              // search k-th CAM module //
    if (match =1)
          then output the matched return field data;
end; {search_CAM_module}

procedure LPM_search_B3
begin
     for I = 1 to m do          // parallel CAM module search //
          parbegin  search_CAM_module(i)  parend;
    output the CAM module output which has the longest mask;
end;{LPM search_B3}
```

그림 5. B2방법에서의 LPM 탐색과정

2.3 T1방법

이 방법은 단일 ternary CAM 모듈을 사용하는 방법이다. 한 개의 ternary CAM 셀은 두개의 일반적인(binary) CAM 셀로 구성하다. Ternary CAM은 각 엔트리(워드)가 매스크를 내장한 것과 같은 기능을 함으로써, 여러 개의 binary CAM 모듈을 하나의 ternary CAM 모듈로 대치하였다. 그러나 ternary CAM의 키필드에 엔트리들은 매스크 길이 순서대로 정렬되어 저장되어야 한다[1]. 따라서 이 방법은 B3와 같이 한 사이클에 LPM 탐색을 수행할 수 있는 반면에, 총 CAM 메모리 셀 수는 B2의 두 배인 2wn의 필요하다. 또한 룩업 테이블을 갱신할 때 마다 ternary CAM내의 엔트리들을 정렬해야 하므로, 룩업 테이블의 갱신에 $O(n)$ 사이클이 걸린다.

```
procedure  LPM_search_T1
begin
    SAR := input_packet_addr;          // load input address //
    search_keyfield;                   // search
CAM array //
    select the lowest row among the multiple hit data;
    output the selected data;          // output the result //
end; {LPM_search_T1}
```

그림 7. T1방법에서의 LPM 탐색과정

● 룩업 테이블의 구성

룩업 테이블의 구성방법은 그림 6과 같이 키필드에 엔트리들이 매스크 길이 순서대로 정렬되어 저장된다는 점을 제외하고는 B2와 동일하다.

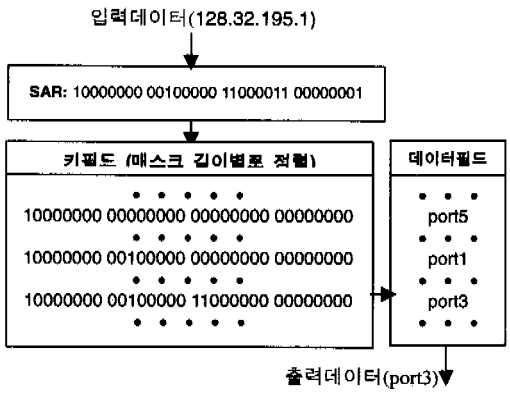

그림 6. T1방법의 룩업 테이블 구성 및 LPM 탐색 예

● LPM 탐색과정

입력패킷의 IP 주소를 SAR에 가하면 이와 매치되는 엔트리들이 여러 개 존재할 수 있다. 엔트리들이 매스크길이 순서대로 정렬되어 있으므로 매치되는 엔트리들 중 LMP는 제일 아래에 있는 엔트리가 된다. 따라서 아래쪽에 있는 엔트리에 출력의 우선순위를 주어 선택하면 longest matching prefix를 찾아낼 수 있고 이에 해당하는 출력포트로 IP 패킷을 출력한다. 탐색과정을 알고리즘으로 표현하면 그림 7와 같다.

Ⅲ. 파이프라인 CAM의 구조(PICAM)

McAuley의 B3는 B2에 비해 LPM 탐색율을 올릴 수 있으나, 복잡도가 높다. T1은 B3와 같은 LPM 탐색율을 가지나, 룩업 테이블의 갱신속도가 느리다. 여기서는 B3나 T1보다 LPM 탐색율이 높으면서도 룩업 테이블의 갱신속도가 빠르고 복잡도가 높지않은 새로운 CAM 구조인 PICAM을 제안한다.

B3는 여러 개의 binary CAM 모듈을 병렬로 동시에 작동시켜 복수개의 출력이 나오면, 이 출력 중에서 최선의 것을 한 개 선택해 LPM 탐색의 결과로 하는 형태이고, T1은 한 개의 ternary CAM 모듈을 작동시키나 복수개의 출력 중에서 최선의 것을 한 개 선택해 LPM 탐색의 결과로 하는 형태이다. 본 PICAM은 binary CAM의 키필드와 데이터필드를 분리해 3개의 단계로 구성되는 파이프라인 처리 형태이다. 파이프라인 처리이기 때문에 하나의 입력 IP 패킷에 대한 LPM 탐색 시간은 더 오래 걸리지만, 단위시간 당 처리하는 패킷 수인 LPM 탐색율(throughput)은 더 올릴 수 있다.

파이프라인 단계는 단계1, 단계2와 단계3으로 구성된다. 룩업 테이블의 IP 주소 데이터는 단계1의 키필드와 단계2의 키필드에 중복 저장된다. 룩업 테이블의 출력포트 데이터들은 단계3의 데이터 필드에 저장된다. 단계1과 단계2의 키필드는 수평적으로 여러 개의 블록으로 분리된다. 이때 각 블록들을 키필드블록이라 한다.

LPM 탐색은 단계1의 블록단위 탐색, 단계2의 비트단위 탐색, 그리고 단계3의 데이터 필드 출력으로

진행된다. 단계1에서는 블록단위의 탐색으로 입력된 IP 주소와 어느 키필드 블록까지 매치되는지 찾아내는 매치블록 탐지가 목표이다. 단계2에서는 비트단위 탐색으로 앞에서 탐지한 첫번째 비매칭 키필드블록에 대해 어느 비트까지 매치되는지를 찾아내는 매치비트 탐지가 목표이다. 매치블록과 매치비트가 결정되면 키필드에서 어느 비트까지 매치되는지 결정되고 이에 해당되는 엔트리가 히트된다. 단계3에서는 히트결과에 의거 데이터필드의 관련데이터를 출력한다.

3.1 PICAM의 구조

PICAM은 그림 8와 같이 3단계의 파이프라인 단계로 이루어진다.

단계1은 m개의 키필드블록과 1개의 제어모듈로 이루어진다. 키필드블록의 폭을 b, IP주소의 길이를 w라하면, w = m × b이다. 키필드블록은 b × n개의 CAM 메모리 셀로 이루어진다. 이러한 키필드 블록이 m개 있으므로, 단계1은 m × b × n = w × n개의 CAM 메모리 셀로 이루어진다. 여기서 n는 룩업테이블의 엔트리 개수이다. 여기서 입력된 2진수로 표현된 IP주소를 왼쪽에서부터 b비트씩 나누어, 각 키필드블록은 블록단위의 매치여부를 탐지하고, 모든 키필드블록이 동시에 동작한다. 제어모듈은 각 키필드블록에서의 매치여부를 기반으로 어느 키필드블록까지 연속적으로 매치가 되는지 탐지하여 매치되지 않는 첫번째 키필드블록의 정보와 그때까지의 매치벡터를 다음 단계로 보낸다. 매치벡터는 처음에서 연속적으로 블록매치 되는 마지막 키필드블록까지의 각 엔트리별로 매치여부를 나타내는 어레이를 말한다. 단계1에서는 한 파이프라인 클록 주기 즉 한 사이클 동안에 한 개의 입력 IP 주소만 처리하며, 한번의 CAM 어레이 접근이 발생한다.

단계2는 m개의 키필드블록과 각 키필드블록을 제어하는 m개의 제어모듈로 이루어진다. 앞 단계에서 지정해준 키필드블록에 대해서만 어느 비트까지 매치되는지 탐지한다. 매치비트 탐지에는 $\log_2 b$의 회수만큼의 CAM 어레이 접근이 일어난다. 키필드블록이 하나의 입력 패킷을 처리하는데 $\log_2 b$의 사이클이 소요된다. 그러나 키필드블록들이 m개가 있어 연속되어 같은 키필드블록이 지정되지 않으면 병렬로 동시에 작동될 수 있다. 최대 m개의 키필드블록이 동시에 매치비트 탐지를 수행할 수 있다. 따라서 한 사이클 당 처리되는 최대 입력 패킷 수는 $m / (\log_2 b)$ 이다. 하나의 키필드블록에서 패킷을 처리하고 있는데 중간에 동일한 키필드블록으로 패킷이 입력되는 패킷충돌(packet conflict)이 발생될 수 있다. 이 경우에 원활한 처리를 위해, 각 키필드블록의 입력단에는 패킷버퍼가 있다. 각 키필드블록의 제어모듈은 앞 단계에서 입력된 매치벡터와 현재의 키필드블록에서 탐지한 매치비트를 기반으로 히트벡터를 생성한다. 히트벡터는 각 엔트리별로 제일 길게 매치되는지를 나타낸다. 히트벡터는 LPM 매치되는 엔트리의 위치를 나타내며, 이는 다음 단계로 출력된다.

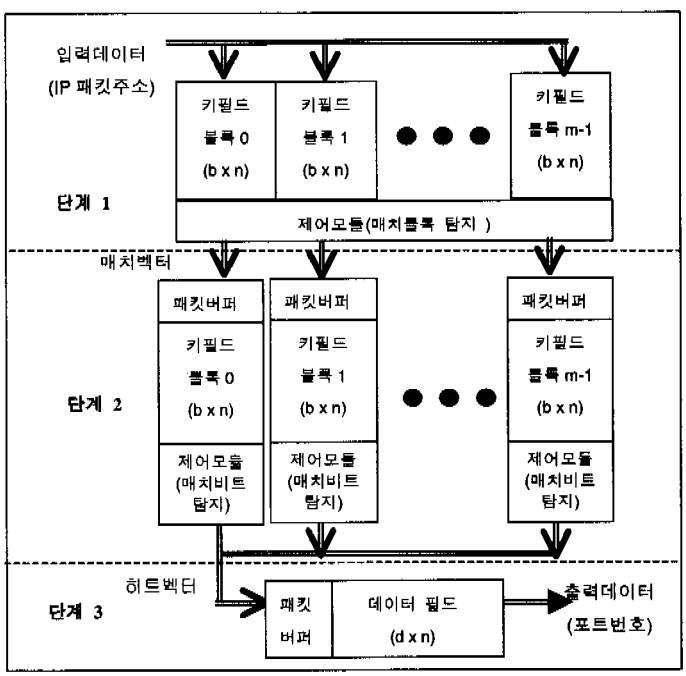

그림 8. PICAM의 구조

단계3은 데이터필드로 구성되어 있다. 바로 앞 단계에서 입력된 히트벡터를 기반으로 데이터필드에서 해당되는 출력데이터를 외부로 출력한다. 데이터 필드의 접근이 한번 일어나므로 한 사이클에 하나의 패킷이 처리된다. 앞 단계의 키필드블록들이 병렬로 동작하므로 한번에 하나이상의 패킷이 입력되는 패킷충돌이 발생될 수 있다. 이 경우에 원활한 처리를 위해, 입력단에 패킷버퍼가 있다.

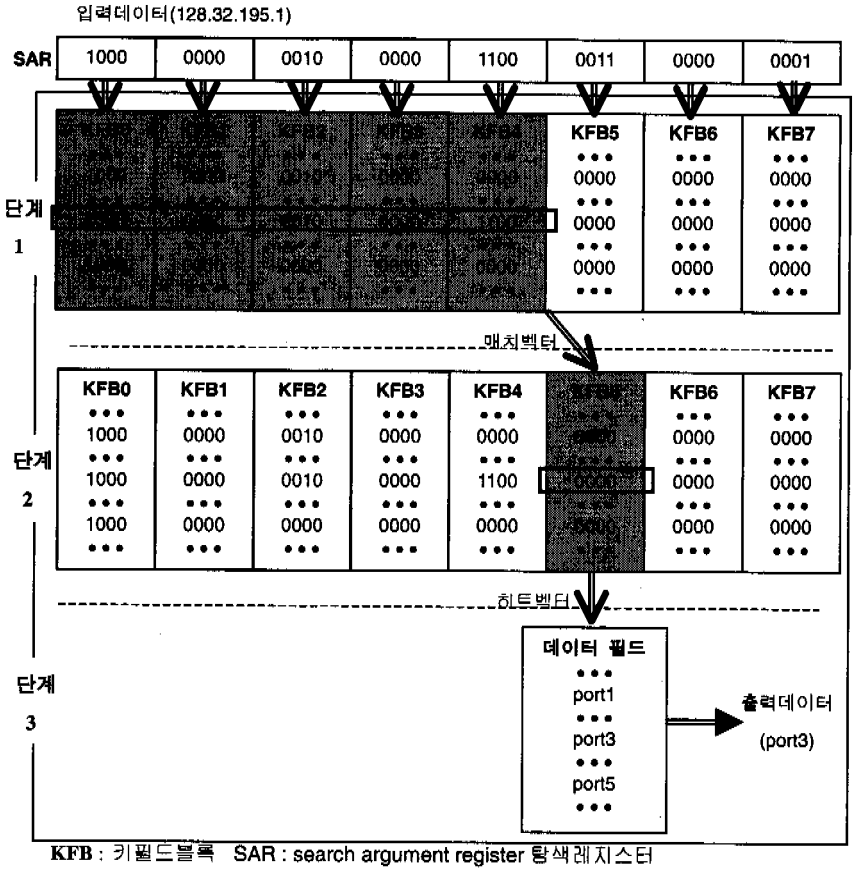

그림 9. PICAM에서의 룩업 테이블 구성 및 LPM 탐색 예

3.2 PICAM의 동작 예

IP version 4, m = 8인 경우의 PICAM에서의 룩업 테이블 구성과 LPM 탐색은 다음과 같다.

● 룩업 테이블 구성

룩업 테이블의 각 엔트리 중 IP 주소는 32 비트의 폭을 갖는다. 그림9와 같이 각 엔트리의 IP 주소는 단계1의 키필드블록과 단계2의 키필드블록의 같은 행으로 저장하고, IP 주소를 왼쪽부터 4비트씩 나누어 차례대로 해당되는 키필드블록의 행에 저장된다. 예를 들면 엔트리가 {128.32.1.5/16, port1}이면 저장되는 IP 주소는 '10000000 00100000 00000000 00000000'이며 1000은 키필드블록0, 0000은 키필드블록1, 0010은 키필드블록2, 0000은 키필드블록3, 0000은 키필드블록4, 와 같이 단계1의 키필드블록과 단계2의 키필드블록의 같은 행에 중복 저장된다.

또한 각 엔트리의 출력데이터는 단계3에 있는 데이터필드의 같은 행에 저장된다. 예를 들어 엔트리가 {128.32.1.5/16, port1}이면 port1의 포트번호가 데이터필드에 저장된다.

● LPM 탐색 과정

입력 패킷 IP주소는 그림 9과 같이 4비트씩 분할되어 단계1의 키필드블록0에서 키필드블록7의 8개 블록으로 총 32비트가 인가된다. 예를 들어 입력 패킷 IP주소가 128.32.195.1이라 하면, 이는 2진표현으로 '10000000 00100000 11000011 00000001'이 된다. 키필드블록 0에는 '1000', 키필드블록 1에는 '0000', 키필드블록2에는 '0010', 키필드블록3에는 '0000', 키필드블록4에는 '1100', 키필드블록5에는 '0011', 키필드블록6에는 '0000', 키필드블록7에는 '0001'이 인가된다. 단계1의 각 키필드블록은 각각

에게 인가된 4비트 데이터가 키필드블록 내의 CAM메모리 어레이에 존재하는지 여부를 알아내는 탐색을 동시에 수행한다. 각 키필드블록은 키필드블록 내에 인가된 4비트의 데이터가 있는지 여부를 탐지해 결과를 매치블록을 탐지하는 제어모듈에 인가한다. 제어모듈이 8개의 키필드블록의 결과를 바탕으로 블록단위로 어느 블록까지 매칭이 되었는지 탐지하면, 매칭이 되지 않는 블록 중 맨 왼쪽의 블록이 다음 단계에서 비트단위로 세부적으로 탐색할 블록이 된다.

예를 들면 그림 9에서 입력 패킷의 IP 주소와 가장 길게 매칭하는 엔트리는 '10000000 00100000 11000000 00000000'이다. 이를 4비트씩 분할해 보면 키필드블록4까지는 완전히 매칭이 되고 키필드블록5부터 매칭이 되지 않는다. 매치벡터는 n개의 셀로 구성되며, 각 값은 입력 IP 주소가 키필드블록 0에서 키필드블록4까지 매치되는 엔트리에 해당하는 셀은 1이고 매치되지 않는 엔트리에 해당하는 셀은 0이 된다. 매치벡터가 단계2의 키필드블록5로 출력된다. 그림 9의 단계1에서 굵은 선으로 둘러싸인 부분이 입력 패킷과 블록매치 되는 행이다.

단계2에서는 키필드블록5에 대해 비트단위로 매치여부를 탐지한다. 키필드블록의 폭이 4이므로 키필드내의 CAM 어레이를 두 번 접근해 매치비트를 판별한다. 즉 매치비트 탐지에는 두 사이클이 걸린다. 예를 들면 앞 단계에서 입력된 매치벡터에서 셀이 1의 값을 갖는 엔트리에 대해 키필드블록5의 입력 패턴 '0011'과 비트매치 되는지를 탐지한다. 비트매치 결과와 매치벡터를 결합하여 히트벡터가 생성된다. 히트벡터는 n개의 셀로 구성되며, 각 값은 현재까지 모든 비트가 매치되면 1이고, 아니면 0이다. 히트벡터에서 1의 값을 갖는 셀에 해당하는 엔트리가 LPM 탐색에서 찾고자 하는 엔트리이다. 히트벡터는 다음 단계로 출력된다. 여기서는 두 번째 비트까지 비트매치 되며, 키필드블록5안에서 왼쪽에서 2번째까지 매칭이 된다. 그림 9의 단계2에서 굵은 선으로 둘러싸인 부분이 비트매치 되는 행이다.

단계3은 앞 단계에서 입력된 히트벡터를 사용해 데이터필드에서 관련된 데이터를 즉 출력포트의 번호를 LPM 탐색의 결과로 출력한다. 여기서는 port3에 해당하는 출력포트의 번호를 출력한다.

PICAM에서의 LPM 탐색과정을 알고리즘으로 표현하면 그림 10과 같다.

Ⅵ. PICAM의 분석

IP version 4, m=8인 경우에 대해 LPM 탐색율, 룩업 테이블 갱신시간 및 복잡도(면적)면에서 PICAM을 기존의 CAM을 이용한 방법들과 비교 분석한다.

4.1 LPM 탐색율(패킷처리율)

기존의 CAM의 처리 사이클은 키필드 탐색과 데이터 필드 접근으로 이루어져 있는 반면에, PICAM의 처리 사이클은 키필드 탐색과 데이터 필드 접근을 분할하여 파이프라인 처리하므로, 한 단계의 처리시간은 기존의 CAM을 이용한 방법의 처리 사이클보다 훨씬 줄어든다. 따라서 하나의 패킷이 처리되는 시간은 더 길어지지만, 패킷처리율 즉 LPM 탐색율은 더 높아진다. McAuley의 방법들과 PICAM방법에서의 LPM 탐색율은 다음과 같다.

● **B2방법**

최대 32번의 CAM 메모리 접근이 필요하다. 패

```
procedure LPM_search_PICAM
begin
    // first stage //
        for i = 0 to m-1 do                    // parallel KFB search for all KFBs //
            parbegin  search KFB(i)  parend;
        find the first unmatched KFB;
        generate match vector;
    // second stage //
        for the first unmatched KFB find match_bit;
        generate hit vector based on match vector and match bit;
    // third stage //
        find return data by hit vector;
        output return data // the result of LPM searching //
end;{LPM_search_B3}
```

그림 10. PICAM에서의 LPM 탐색과정

킷처리시간을 T_{B2}, 한번의 CAM 메모리 접근시간을 T_{CAM}이라 하면, $T_{CAM} \leq T_{B2} \leq 32T_{CAM}$ 이다.

패킷처리율 P_{B2} 라 하면, $1/(32T_{CAM}) \leq P_{B2} \leq 1/T_{CAM}$ 이 된다. $P = 1/T_{CAM}$ 라 하면, $(1/32)P < P_{B2} < P$ 가 된다.

● B3방법

한번의 CAM 메모리 접근으로 LPM 탐색이 수행된다. CAM 메모리 접근시간을 T_{B3}라 하자. 여러 개의 CAM 모듈이 매치되면 이들의 우선순위를 처리하는 시간이 추가된다. 따라서 $T_{B3} > T_{CAM}$ 이다. 패킷처리율을 P_{B3} 라 하면, $P_{B3} = 1/T_{B3} < 1/T_{CAM}$ 이 된다. $P = 1/T_{CAM}$ 이므로, $P_{B3} < P$ 가 된다.

● T1방법

이 경우도 LPM 탐색은 ternary CAM의 한번 접근으로 완료된다. ternary CAM은 일반 CAM과 접근시간에서 크게 차이가 나지 않는다. 따라서 ternary CAM 메모리 접근시간을 T_{T1}라 하면, $T_{T1} = T_{CAM}$ 이 성립된다. 패킷 처리율을 P_{T1} 라 하면, $P_{T1} = 1/T_{T1} = 1/T_{CAM} = P$ 가 된다.

● PICAM의 경우

PICAM에서는 단계 1에서는 한번에 한 패킷만 처리되고, 단계 2에서는 키필드블록이 중복되지 않으면 최대 8개의 패킷이 동시에 처리될 수 있다. 따라서 단계 2에서 패킷이 2 사이클이 걸림에도 불구하고 각 키필드블록이 병렬로 처리함으로써, 단계1의 패킷처리율에 맞추어 단계2에서 패킷처리를 할 수 있다.

단계1에는 한번의 CAM 키필드 접근이 있고, 단계2에는 2번의 CAM 키필드 접근이 있다. 일반 CAM 메모리 접근시간은 CAM 키필드(키필드) 접근시간과 데이터필드 접근시간으로 구성되어 있기 때문에 단계1과 단계2에서의 CAM 키필드 접근시간은 일반 CAM 메모리 접근시간 보다 작다. 일반 CAM 메모리 접근시간을 T_{CAM}, 단계 1의 패킷 처리시간을 T_{ST1}, 단계 2에서의 패킷 처리시간을 T_{ST2} 라 하자. 그러면 $T_{ST1} < T_{CAM}$, $(T_{ST2})/2 < T_{CAM}$ 이 된다.

전체 파이프라인 흐름에 걸어주는 클록의 주기 T는 T_{ST1}과 $(T_{ST2})/2$ 중에서 큰 것으로 택한다. 그러면 단계1에서의 패킷 처리시간은 T로 되고, 단계2에서의 패킷 처리시간은 2T가 된다. T는 한번의 CAM 키필드 접근시간만으로 이루어지며, 데이터필드 접근시간이 제외되기 때문에 제어모듈에서의 탐지시간이 데이터필드 접근시간 보다 작게 될 수 있다면, CAM 메모리 접근시간보다 작게 할 수 있다. 따라서 $T < T_{CAM}$ 이다.

단계1에서의 패킷처리율은 1/T이고, 단계2에서의 패킷처리율은 최대 1/(2T) 8 = 4/T가 된다. 단계2에서는 패킷처리시간이 단계1보다 2배 길기 때문에, 패킷충돌이 발생할 수 있다. 단계2의 각 키필드블록의 입력단에 있는 패킷버퍼가 패킷충돌이 발생할 경우 파이프라인의 흐름을 깨지 않고 처리한다. 따라서 단계2에서는 단계1의 패킷처리율에 맞추어 처리한다.

단계3에서의 처리시간은 CAM 키필드 접근시간은 없고, 단지 데이터필드 접근시간만 있다. 따라서 단계1이나 단계2에 비해 패킷처리시간은 적게 걸리며 전체 파이프라인 클록의 주기 T에 맞추어 동작한다. 여기서도 앞 단계와 같이 패킷충돌이 발생할 수 있으나, 입력단에 있는 패킷버퍼로 파이프라인 흐름을 깨지 않고 처리한다. 단계3에서의 패킷처리시간은 T이며, 패킷처리율은 1/T가 된다.

이상 각 단계의 패킷 처리율을 종합하면, PICAM에서의 패킷 처리율을 P_{PICAM} 이라면, $P_{PICAM} = 1/T$ 가 된다. 또한 $T < T_{CAM}$ 이므로, $P_{PICAM} > P_{CAM}$ 이된다.

4.2 룩업 테이블의 갱신시간

기존의 CAM을 사용한 방법들과 PICAM에서 룩업 테이블의 갱신속도를 살피면 다음과 같다.

● B2방법, B3방법

이 경우는 룩업 테이블을 정렬할 필요가 없다. B3의 경우에는 매스크 길이별로 구분해 CAM모듈에 저장하거나 삭제하면 된다. 따라서 한 사이클에 룩업 테이블의 갱신을 완료할 수 있으므로 갱신시간은 $O(1)$이다.

● T1방법

룩업 테이블이 매스크 길이 별로 정렬되어 ternary CAM 키필드에 저장된다. 매번 룩업 테이블을 갱신할 때마다 룩업 테이블을 정렬하므로, 갱신시간은 $O(n)$사이클이 된다.

● PICAM의 경우

단계1과 단계2의 키필드블록에 엔트리들이 정렬될 필요가 없다. 룩업 테이블 갱신할 때는 각 단계에 있는 룩업 테이블을 동시에 갱신하므로, 한 사이클에 룩업 테이블의 갱신을 완료된다. 따라서 갱신시간은 $O(1)$사이클이다.

4.3 면적(복잡도)

면적 복잡도는 VLSI로 구현할 때 필요한 면적을 나타낸다. 이 중 CAM 키필드의 셀의 개수 및 제어회로의 개수로 비교한다.

(1) CAM 키필드의 셀 수

셀의 개수 면에서 기존 CAM을 이용한 방법들과 PICAM을 살펴보면 다음과 같다.

● B2방법

n개의 엔트리가 있으므로 필요한 CAM 메모리 셀은 32n 비트이다.

● B3방법

각 매스크 길이 마다 하나의 CAM 모듈이 필요하므로, 최대 32개의 CAM 모듈이 필요하다. 각 CAM 모듈의 엔트리 수는 미리 알 수 없다. k비트의 매스크길이를 갖는 CAM모듈에 들어갈 수 있는 엔트리 수는 2^k와 n중 작은 것이다. 따라서 B3의 총 CAM 메모리 셀 수는 $\sum_{k=1}^{32} k \bullet \min(2^k, n)$ 이다.

$n = 2^{16}$라 하면, k=16까지는 2^k < n이다.

$\sum_{k=1}^{32} k \bullet \min(2^k, n) = \sum_{k=1}^{16} k2^k + \sum_{k=17}^{32} kn \cong 422n$ 이다.

B3의 총 CAM메모리 셀 수는 약 422n 이며, B2의 총 CAM 메모리 셀수 32n에 비해 매우 커진다.

● T1방법

ternary CAM을 사용하므로, binary CAM 경우보다 2배의 CAM 메모리 셀이 필요하다. CAM 메모리 셀 수는 2 × 32 × n = 64n 비트이다.

● PICAM의 경우

단계1의 키필드와 단계2의 키필드에 동일한 키들이 중복되어 저장되므로, 필요한 CAM 메모리 셀은 2 × 32 × n = 64n비트이다. T1과 같은 개수의 CAM 메모리 셀이 필요하다.

(2) 제어회로의 개수

B3는 32개의 CAM모듈 제어회로가 필요하다. 반면에 T1에서는 1개의 CAM모듈 제어회로가 필요하다. PICAM에서는 단계 1과 단계2 각각 8개씩 총 16개의 CAM모듈 제어회로가 필요하다. 따라서 PICAM 제어회로 복잡도는 B3보다는 간단하고 T1보다는 복잡하다.

4.4 종합

지금까지 분석한 내용을 정리하면 아래 표1과 같다. 기존 CAM을 이용한 방법 중에서 B2는 갱신시간은 짧으나, LPM 탐색율이 너무 낮다. B3는 갱신시간은 짧고, LPM 탐색율은 높으나, 복잡도가 너무 높다. T1은 LPM 탐색율은 높고, 복잡도도 크지 않으나, 갱신시간이 너무 길다. PICAM은 기존의 CAM을 이용한 B3과 T1보다 LPM 탐색율이 크고, 갱신시간은 T1보다 매우 짧다. PICAM이 CAM메모리 셀의 개수에서는 B3 및 T1보다 작거나 대등하고, 제어회로의 개수에서는 T1보다는 크나 B3보다는 작다. PICAM은 기존의 방법보다 룩업 테이블의 갱신이 매우 빠르며, LPM 탐색율도 높으며, 복잡도도 작다.

여기서 $P=1/T_{CAM}$이고, T_{CAM}은 일반 CAM메모리 접근시간이다. n는 2^{16}로서, 룩업 테이블의 엔트리의 개수이다.

기존의 소프트웨어를 이용한 방법들은 LPM 탐색에 기본적으로 이진 탐색을 이용하기 때문에 패킷 처리시간이 $O(\log n)$이 걸린다. 또한 룩업 테이블이

표 1. 기존 CAM을 이용한 방법들과 본 논문의 PICAM방법의 비교

		기존 CAM을 이용한 방법들(McAuley)			본 논문의 PICAM 방법
		B2방법	B3방법	T1방법	
LPM 탐색율		(1/32)P ~ P	P보다 작음	P	P보다 큼
갱신시간(사이클)		O(1)	O(1)	O(n)	O(1)
복잡도	CAM메모리 셀 수	32n	422n	64n	64n
	제어회로 개수	1	32	1	16

정렬된 상태를 유지해야 하므로 룩업 테이블의 갱신시간도 $O(\log n)$이나 걸린다. 본 논문의 PICAM 방법은 기존의 소프트웨어를 이용한 방법들 보다 LPM 탐색율이 매우 높으며, 룩업 테이블의 갱신속도도 매우 빠르다.

V. 결 론

기존의 LPM 탐색 방법들은 룩업 테이블의 갱신에 대한 고려 없이 오직 패킷처리율 즉 LPM 탐색율만 높이는데 주력해, 고속 라우팅에서 룩업정지나 시효가 지난 경로에 의한 부정확한 라우팅을 하게 되는 문제가 있었다. 이런 문제들을 없애려면 룩업 테이블의 빠른 갱신이 무엇보다 필요하다. 또한 복잡도가 높지 않아야 실용적인 LPM 탐색 방법이 될 수 있다.

본 논문에서는 인터넷 라우터에서 기존 CAM을 이용한 방법보다 라우터의 룩업 테이블의 갱신이 매우 빠르고, LPM 탐색율도 높으면서도, 복잡도도 높지 않은 새로운 파이프라인 CAM 구조인 PICAM방법을 제안했다.. PICAM방법의 룩업 테이블 갱신시간은 $O(1)$사이클로서 기존의 LPM 탐색에서의 $O(n)$사이클보다 훨씬 빠르다. . 또한 PICAM 방법은 LPM 탐색율도 높아 라우터의 룩업 테이블의 갱신이 매우 빈번한 백본 라우터에서 더욱 유용하다.

PICAM 구조는 ASIC(application specific IC)이나 주문형 칩으로 구현되어 초고속 라우터의 고속 LPM 탐색에 사용될 수 있다

참 고 문 헌

[1] Anthony J. McAuley and Paul Francis, "Fast Routing Table Lookup Using CAMs", IEEE INFOCOM'93, vol. 3, pp 1392-1392, March 1993

[2] Nen-Fu Huang and Shi-Ming Zhao, "A Novel IP-Routing Lookup Scheme and Hardware Architecture for Multigigabit Switching Routers", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 6, June 1999, pp. 1093 - 1104

[3] Y. Rekhter and T. Li. "An Architecture for IP Address Allocation with CIDR." RFC 1518, Sept. 1993

[4] Henry Hong-Yi Tzeng and Tony Przygienda, "On Fast Address-Lookup Algorithms", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 6, June 1999, pp. 1067-1082

[5] Pankaj Gupta, Steven Lin, and Nick McKeown, "Routing Lookups in Hardware at Memory Speeds", Proc. of INFOCOM '98, Session 10B-1, San Francisco, CA, pp. 1240-1247

[6] Andreas Moestedt and Peter Sjodin, "IP Address Lookup in Hardware for High-Speed Routing", Proceeding of Hot Interconnects, August 1998, pp. 1-9

[7] Marcel W. Waldvogel, George Varghese, Jon Turner, Bernhard Platner, "Scalable High Speed IP Routing Lookups", Proc. of ACM SIGCOM '97, France, pp. 25-36

[8] Mikael Degermark, Andrew Brodnik, Svante Carlsson, and Stephen Pink, "Small Forwarding Tables for Fast Routing Lookups", Proc. of ACM SIGCOMM'97, France, pp. 3-14

[9] Wen-Shyen E. Chen and Chung-Ting Justine Tsai, "A Fast and Scalable IP Lookup Scheme for High-Speed Networks", Proc. of IEEE International Conference on Networks (ICON'99), pp. 211-218

[10] Stefan Nilsson and Gunnar Karlsson, "Fast address lookup for Internet routers", http://www.nada.kth.se/~snilsson/public/papers.html

[11] Butler Lampson, Venkatachary Srinivasan, and George Varghese, "IP Lookups Using Multiway and Multicolumn Search", IEEE Transaction on Networking, Vol. 7, No. 3, June 1999, pp. 324-334

[12] Anthony J. McAuley, Paul F. Tsuchiya, and Daniel V. Wilson. "Fast multilevel hierarchical routing table using content-addressable memory", U. S. Patent serial number 034444. Assignee Bell Communications research Inc Livingston NJ, January 1995.

[13] Daxiao Yu, Brandon C. Smith, and Belle Wei, "Forwarding Engine For Fast Routing Lookups and Updates", IEEE Global Telecommunications Conference, proceeding on GLOBECOM '99, vol. 2, pp. 1556-1564, 1999

[14] Scott Bradner, "Next Generation routers

Overview," proceeding of Networld Interop 97, 1997

안 희 일(Hee Il Ahn) 정회원

1973년 2월 : 서울대학교 전자공학과 졸업(공학사)
1996년 2월 : 충북대학교 대학원 전자공학과 졸업 (공학석사)
1976년 11월~1978년 3월 : 한국과학기술연구원 연구원
1978년 1월~1999년 4월 : 한국전자통신연구원 책임연구원/실장
1999년 9월~현재 : 충북대학교 전기전자공학부 객원교수, 강사
<주관심 분야> 컴퓨터구조, 라우터, 유전자알고리즘

조 태 원(Tae Won Cho) 정회원

1973년 2월 : 서울대학교 전자공학과 졸업(공학사)
1986년 5월 : 미국 루이빌대 전자공학과 졸업 (공학석사)
1992년 5월 : 미국 켄터키 주립대 전자공학과 졸업 (공학박사)
1973년 8월~1983년 10월 : 금성전선㈜
1977년 1월~1977년 3월 : 영국 및 프랑스의 ITT계열사 연수
1992년 9월~현재 : 충북대학교 전기전자공학부 부교수
<주관심 분야> 집적회로설계, 컴퓨터구조, 저전력회로설계, DSP core 설계