특집

패킷 포워딩 기술

임혜숙(이화여자대학교 공과대학 정보통신학과), 정여진(SoC 설계 연구실)

I. 서론

오늘날의 인터넷의 영향력은 기업이나 개인뿐 아니라 사회 각 분야에 걸쳐 빠르게 확산 되어가고 있으며, 이에 따라 여러가지 기술적인 난제를 안겨주고 있는데, 그 대표적인 것이 스위치나 라우터와 같은 스위칭 장비에서의 패킷 포워딩 기술이라 하겠다. 패킷 포워딩이란 다양한 네트웍들을 연결하는 스위칭 장비에서 수행되는 동작으로, 들어온 패킷의 헤더 정보를 이용하여 최종 목적지 네트웍을 향해 패킷을 내 보내주는 일련의 단계를 말한다.

패킷 포워딩 기술에서 해결해야 할 첫번째 문제는 인터넷 트래픽의 급속한 증가에서 비롯된다. 광섬유를 사용한 링크 기술은 인터넷 트래픽의 증가 속도에 맞추어 발전하고 있는데, 2.5Gbps(OC - 48), 10Gbps (OC - 192)등이 이미 사용되고 있으며, 40Gbps(OC - 768)로의 진행도 곧 이루어질 것으로 보인다. 그러나 스위칭 장비에서의 패킷 처리 속도는 이에 발 맞추어 발전하고 있지 않아 스위칭 장비에서의 패킷의 실시간 처리는 차세대 스위칭 장비 설계의 중요한 관건이 될 것으로 보인다.

두번째 문제는 네트웍의 다양화에서 비롯된다. Asymmetric digital subscriber line(ADSL) 이나 hybrid fiber coaxial(HFC) 케이블과 같은 네트웍 접근 기술의 발달은 기업이나 학교에서뿐 아니라 가정에서의 네트웍 접근을 가능하게 만들었다. 이에 따라 네트웍이 매우 다양화되어, 패킷 포워딩을 위해 스위칭 장비에서 참조해야 하는 데이타베이스의 크기가 기하 급수적으로 증가하였으며, 이에 따라 패킷 포워딩은 점점 더 어려운 작업이 되고 있다.

마지막으로는 다양한 서비스의 요구에서 비롯된다. 오늘날의 인터넷은 best - effort - service로 대변되며, 입력된 모든 패킷에 대해 동일한 서비스를 제공한다. 그러나 오디오나 비디오 스트리밍과 같이 전달 시간에 민감한 응용 프로그램의 등장은 quality - of - service(QoS)를 요구하고 있으며, 그 외에도 방화벽(firewall protection)이나 load balancing, 웹 스위칭, intrusion detection and security(IDS) 등의 다양한 서비스가 요구되고 있다. 스위칭 장비에 들어오는 패킷은 2계층에서 4계층까지의 헤더와 응용 프로그램의 데이타를 포함하는 세크먼트라고 볼 수 있는데, 초기 인터넷에서의 패킷 포워딩은 패킷의 2계층과 3

계층 헤더에 기초하여 이루어졌다. 오늘날의 스위칭 장비는 이에 더하여 4계층 헤더까지 참조하는 패킷 분류(packet classification)를 수행하고 있으며, 차세대 스위칭 장비에서는 앞서 말한 다양한 서비스를 제공하기 위하여 5계층 이상의 헤더(패킷의 payload)까지를 참조하여야 한다.

스위칭 장비로 들어온 패킷은 일련의 처리 과정을 거친 후 출력포트로 내보내어 지는데, 패킷 처리 과정을 단계별로 살펴보면 다음과 같다. 그 첫번째가 네트웍 미디엄을 통하여 들어오는 비트 스트림을 프레임이나 셀로 분리한 후, cyclic redundancy check(CRC)등을 보고 프레임에 에러가 있는지 검사하는 과정이다. 두번째로는 패킷의 여러 계층의 헤더로부터 네트웍 시스템의 protocol이나 응용 프로그램의 요구 사항에 맞는 정보를 도출하는 과정이다. 세번째가 패킷 분류로서 미리 정의된 데이타 베이스의 내용과 앞서 도출된 헤더 정보를 비교하는 단계이다. MAC 주소 검색 및 learning, IP 주소 검색, access control list(ACL) 필터링, 플로우 분류 및 connection 유지, 컨텐트 프로세싱등의 기능이 패킷 분류에 포함된다. 패킷 프로세싱의 네번째 과정은 패킷의 encryption 과 decryption 으로써, e - commerce 나 virtual private network(VPN)과 같은 서비스를 위해 필요하다. 다음은 패킷의 수정과 스케쥴링 과정으로서, 레이블이나 태그의 수정, type - of - service(ToS) 나 time - to - live(TTL) 필드의 수정, CRC의 수정 등이 포함되며, 패킷의 우선 순위에 따른 출력 버퍼의 관리가 이에 속한다. 마지막 과정은 네트웍 관리를 위해 필요한 정보를 모으는 일로서, 트래픽의 종류, 사용된 resource, 제공된 bandwidth나 서비스 등의 정보가 모아진다. 이러한 일련의 처리 과정을 거친 패킷은 마침내 출력 포트로 내보내어 진다.

여러가지 패킷 처리 과정중 패킷 포워딩의 핵심이 되면서도 어려운 기능이 패킷 분류라 할 수 있다. 패킷 분류는 입력 포트로 들어오는 모든 패킷에 대하여 실시간으로 수행되어야 하나 패킷당 여러번의 메모리 룩업이 요구되기 때문이다. 패킷 분류를 효율적으로 수행하기 위한 여러가지 패킷 분류 알고리듬 및 구조에 관한 연구가 활발하게 수행되어 오고 있다. 패킷 분류 알고리듬 및 구조들의 성능 평가의 가장 중요한 기준은 패킷 분류를 위해 요구되는 메모리 검색 횟수라 할 수 있는데, 메모리 검색 횟수는 전체 패킷 처리 속도의 척도가 되기 때문이다. 라우팅 테이블이 평균적으로 초당 100번 이상 갱신됨을 고려할때, 새로운 네트웍 정보를 추가하거나 쓰지 않는 네트웍 정보 삭제의 용이성도 중요한 척도가 되며, 정의된 데이타 베이스를 저장하기 위해 필요로 하는 메모리의 크기와 라우트의 수나 길이에 있어서의 확장 가능성도 중요한 척도가 된다[1].

본 고에서는 단일 필드 패킷 분류라 할 수 있는 IP 주소 검색 관련하여 기존에 연구된 구조들에 대해 먼저 알아보고, 이를 복수 개의 필드를 사용하는 패킷 분류를 위해 확장한 구조들에 관하여 알아보고자 한다. 기존의 연구 결과에 대하여 알아보고 성능을 비교 검토하는 일은 고속 패킷 처리 능력을 갖는 스위칭 장비 개발의 근간이 된다 할 것이다.

II. IP 주소 검색 구조

IP 주소는 네트웍을 지정하기 위한 프리픽스와 개개 호스트를 지정하기 위한 부분의 2계층 구조를 갖는다. 프리픽스란 같은 네트웍에 속한 모든 호스트들이 공통적으로 갖게 되는 IP 주소의 앞부분으로서, 과거의 클래스를 갖는 주소 구조

(classful addressing scheme)에서는 IP 주소의 프리픽스 길이를 8, 16, 혹은 24 로 고정시켜 사용하여 값 일치(exact matching)에 의한 어드레스 룩업이 가능하였다. 그러나 이러한 구조에서는 제한된 자원인 IP 주소가 낭비되는 단점이 있어 이를 해결하기 위해 클래스를 갖지 않는 주소 구조(Classless Inter - Domain Routing - CIDR)가 등장하였다. CIDR는 네트웍 부분이 임의의 길이를 가질 수 있으므로, 네트웍에 붙는 호스트의 갯수에 따른 다양한 크기의 네트웍 구성을 가능하게 하여 IP 주소 낭비를 방지하고, 또한 복수 개의 네트웍을 묶어(network aggregation) 하나의 커다란 네트웍으로 표현함으로써 백본 라우터에서의 라우팅 테이블의 크기가 증가하는 것을 방지할 수 있는 장점이 있다. 그러나 CIDR 개념하에서의 패킷은 네트웍의 계층 구조에 따라 여러개의 네트웍과 일치 할 수 있으므로, 특정 네트웍을 찾기 위하여 입력 패킷의 목적지 주소와 가장 길게 매치하는 엔트리를 찾아야 하며 이를 longest prefix matching(LPM) 이라 한다. LPM을 사용한 주소 검색은 과거의 <값>을 비교하는 일차원 검색에서, <값, 길이>를 비교하는 2차원 검색으로 전환된 점에 그 어려움이 있다 할 것이다[1].

LPM을 위한 여러가지 알고리듬과 구조들이 제안되어 왔는데, 크게 하드웨어에 기반한 방식과 소프트웨어에 기반한 방식으로 나눌 수 있다. 소프트웨어에 기반한 방식은 소규모 엑세스 라우터에서 주로 사용되는 방식으로, 패킷 포워딩 알고리듬을 소프트웨어로 프로그램하여, CPU가 패킷 포워딩을 담당하도록 한다. 저가의 스위칭 장비에 패킷 분류 기능을 추가하여 엑세스 라우터를 대치하고자 하는 노력들도 행하여 지고 있는데, 이런 경우에도 낮은 가격으로 구현하기 위하여 고속의 하드웨어가 아닌, 소프트웨어가 처리하도록 설계된다. 소프트웨어에 기반한 대부분의 기존의 방식들은 구현이 용이하고, 수정, 보완이 편리한 반면, 고속 라우터의 경우에는 패킷의 실시간 처리가 어려우며, 라우트의 갯수가 아닌, 프리픽스의 길이에 따라 메모리 엑세스 횟수가 결정되는 단점이 있다. 하드웨어에 기반한 방식은 백본 라우터등과 같이 패킷의 고속 처리가 관건이 되는 곳에서 주로 사용되며, 패킷 포워딩을 위한 전용 하드웨어를 ASIC으로 구현하여 사용하는 방식이다. 주로 TCAM 이나, 해쉉이 사용되며 어드레스 룩업을 위한 전용 엔진을 구현하여 패킷 포워딩을 위해 특별히 제작된 칩안에 내장하는데, 고속의 패킷 포워딩이 가능하나 알고리듬의 수정, 보완이 어렵다는 단점이 있다.

1. TCAM에 기초한 방식

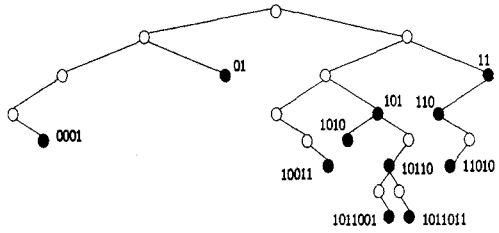
현재 상용화된 고속 라우터나 3계층 스위치에서 가장 많이 사용되는 방식으로서, ternary content addressable memory (TCAM)라는 특별히 제작된 메모리를 사용하여, 라우팅 테이블을 저장한다. 어드레스 검색은 입력된 패킷의 목적지 주소와, TCAM에 저장된 모든 엔트리의 프리픽스들이 동시에 비교되고, 일치하는 엔트리 중 가장 길게 일치하는 엔트리의 주소가 선택되는 방식이다. 이 주소는 포워딩 메모리를 가르키는데 사용되어, 포워딩 메모리로부터 출력 포트 정보와 다음 홉 주소의 정보를 얻게 된다. 구현이 간편하고, 매우 빠른 성능을 보이지만, TCAM은 일반 SRAM 에 비하여 차지하는 면적이 6배 정도 크고, 가격도 비쌀 뿐만아니라, 전력소모가 커서 패킷 포워딩을 위해 제작된 칩안에 내장되지 않는 단점이 있다. 또한 큰 라우팅 테이블이나, IPv6로의 확장에도 적합하지 않다[2].

2. Hashing 에 기초한 방식

해쉬이란 메모리 사용의 효율을 높이기 위하여 긴 길이의 IP 어드레스를 짧은 길이의 스트링으로 매핑한 후, 짧은 길이의 스트링을 메모리의 주소로 사용하여 검색을 수행하는 방식이다. 이때, 서로 다른 어드레스가 같은 스트링으로 매핑되는 경우가 발생할 수 있는데, 이를 충돌이라 하며, 충돌을 효율적으로 처리하여 주는 것이 관건이다. 해쉬은 값 일치를 사용하는 2계층 어드레스 룩업을 위해 많이 이용되어 왔으며, LPM을 이용하는 3 계층 어드레스 룩업을 위해서는 각 프리픽스 길이별로 별도의 해쉬을 수행하게 된다[1]. 프리픽스 길이별로 별도의 해쉬 테이블을 만들고, 충돌이 난 같은 길이의 프리픽스 들에 대하여 이진 검색을 적용한 방식과 복수의 해쉬을 적용하여 충돌을 줄인 방식도 제안되었다[3][4]. 라우팅 테이블을 4개의 그룹으로 나누어 별도의 메모리에 저장한 후, prefix 길이별로 긴 prefix 로 부터 순차적인 해쉬을 수행하는 방식도 있다[5].

3. Trie 에 기초한 방식

Trie는 <길이>에 대해서는 선형 검색을 적용한 방식으로서, 모든 프리픽스는 trie에서 하나의 노드에 위치하게 되며, 길이가 L인 프리픽스는 트리의 L번째 레벨에 존재하는 노드에 저장된다. 트리의 루트로부터 출발하여, 프리픽스의 한 비트씩 검사하여 프리픽스의 값이 0이면 왼쪽, 1이면 오른쪽 노드로 이동하는 방식이다. 그림 1은 임의의 프리픽스 집합에 대하여 이진 trie를 구성한 예를 보여준다. 어드레스 검색은 입력된 어드레스를 한 비트씩 검사해가면서, 일치하는 노드



〈그림 1〉 이진 trie

를 만날때마다 일치 정보를 업데이트하고, trie의 끝 단에서 종료된다. 가장 직관적인 구조이나, 프리픽스가 존재하는 노드뿐만 아니라 프리픽스가 존재하지 않는 빈 노드까지도 저장되어야 하는데서 오는 메모리 낭비가 있고, 메모리 검색 횟수는 O(W) (W는 어드레스 길이)에 비례하는 단점이 있다. 이러한 단점을 해결하기 위해 여러 가지 방식들이 제안되었는데, 메모리 엑세스 횟수를 줄이기 위해, 두개 이상의 가지를 갖도록 한 multi - bit trie가 있다. Patricia 트리는 하나의 가지를 가진 비어있는 노드들을 없앰으로서 몇몇 비트들의 검색을 바이패스 시킨 path - compressed trie이고, multi - bit trie와 path - compressed를 모두 결합한 level compressed trie 등이 있다[1]. Lulea 방식은 trie를 저장하는데 있어 작은 메모리를 사용하기 위해 제안된 방식으로, 노드 정보를 매우 축약하여 표현하였으나, 선처리를 많이 요구하여 라우팅 테이블의 갱신이 용이하지 않은 단점이 있다[1]. IP 주소 검색의 <길이> 차원을 없애는 가장 쉬운 방법은 모든 프리픽스의 길이를 32 비트로 expansion 하여 저장하는 것이다. 그러나 이러한 프리픽스를 저장하는데 너무나 큰 메모리가 요구됨으로, controlled prefix expansion 방법이 제안 되었는데, 이는 <길이> 차원의 종류를 줄임으로서 보다 빠른 검색을 시도한 방식과, 주어진 프리픽스 집

합에 대해 서로 디스조인트한 관계를 갖도록 트리의 끝단으로 프리픽스를 밀어내기(leaf pushing)하는 방식이 있다. 그러나 프리픽스 확장의 경우 하나의 프리픽스가 여러개의 프리픽스로 전환됨에 따라 프리픽스 정보의 추가, 삭제가 용이하지 않은 단점을 갖는다[1].

4. 영역 검색 (range search) 에 기초한 방식

영역 검색은 모든 프리픽스를 32 비트로 확장하되, 호스트 부분을 모두 0을 넣어 확장한 것과, 모두 1을 넣어 확장한 두개의 프리픽스 만으로 구성하여, IP 주소 영역을 프리픽스들의 디스조인트한 기본 구간으로 나눈다. IP 주소 검색은 기본 구간의 경계점에 기초하여 수행되는 방식으로, LPM의 <길이> 차원을 완전히 없앴다는 점에서 매우 우수한 구조라 하겠다[1].

5. 이진 검색에 기초한 방식

기존의 이진 검색은 길이가 같은 스트링간의 크기 비교에 기초하여 이루어졌으므로, LPM에 적용하는데 어려움이 있었다. 최근 LPM을 수행하는 IP 주소 검색을 위하여 이진 검색을 적용하는 새로운 개념이 제안되었다[6][7]. Yazdani [6]는 길이가 다른 프리픽스들 간의 크기 비교를 가능하게 하는 몇가지 정의를 제안하였는데 다음과 같다.

정의 1 **비교** (comparison):

두 프리픽스의 길이가 같으면 숫자 값이 비교된다.

예) A=1001, B=1100인 경우, A<B.

두 프리픽스의 길이가 다른 경우 짧은 길이까지만 비교된다. 짧은 길이까지의 스트링이 같은 경우, 다음 비트(bit)가 1이면 B>A, 아니면 B<A이다.

예) A=1001, B=110000인 경우, A<B

예) A=1001, B=100110인 경우, A<B

예) A=1001, B=100100인 경우, A>B

정의 2 **매치** (match):

하나의 프리픽스가 다른 프리픽스의 substring이면 두 프리픽스는 매치한다.

예) A=1001, B=100100인 경우, A와 B는 매치한다.

정의 3 **디스조인트** (disjoint):

두 프리픽스가 매치하지 않으면 A와 B는 disjoint 하다.

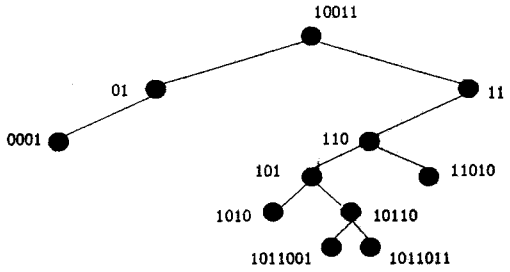
예) A=1001, B=111인 경우, A와 B는 disjoint 하다.

정의 4 **인클로저 프리픽스** (enclosure prefix):

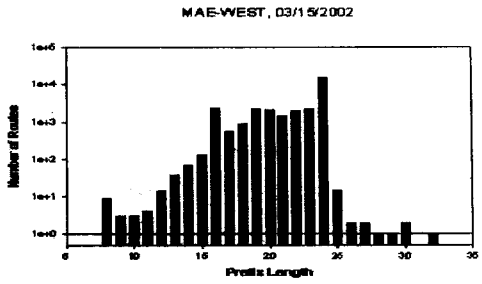
프리픽스 A를 프리픽스로 갖는 다른 프리픽스가 하나라도 존재하면 A는 인클로저 프리픽스이다.

예) A=1001, B=100100인 경우, A는 B의 인클로져 프리픽스이다.

그림 2에서는 위의 정의를 사용하여 그림 1의 이진 trie를 구성한 같은 프리픽스 집합에 대하여 트리를 구성한 이진 프리픽스 트리(binary prefix tree - BPT)의 예를 보여준다. 트리의 모양이 한쪽으로 쏠린 불균형적인 트리를 구성한 것을 알 수 있는데, 이는 LPM의 특성상, 인클로져 프리픽스는 자신을 프리픽스로 하는 더 긴 프리픽스보다 상위에 위치하여야 하는 제약점 때문이다. 이러한 제약점만을 제외한다면, 이진 프리픽스



〈그림 2〉 이진 프리픽스 트리 (BPT)



〈그림 3〉 MAE-WEST 라우트 분포

트리는 LPM의 <값, 길이>의 2차원 문제를 <값> 만을 고려하는 1차원으로의 변환을 시도하였다는 점에서 매우 우수한 구조라고 할 수 있다. 그러나 불균형 트리는 라우트의 분포 형태에 따라 메모리 검색 횟수가 최악의 경우 W (W는 어드레스 길이) 번까지 가는 단점이 있다. BPT가 불균형 트리를 생성하는 단점을 해결하기 위해 [7]에서는 라우팅 테이블을 디스조인트 프리픽스들로만 구성된 여러개의 독립된 트리들로 나누어 저장하는 방식을 제안하였다. 하나의 트리에 속한 모든 프리픽스들은 서로 디스조인트한 관계를 갖도록 하면 균형 트리가 생성되는 점에 착안한 구조로서, 하드웨어 파이프라인 방식을 적용하여 한번의 메모리 검색으로 IP 주소 룩업이 가능한 구조이다.

그림 3은 약 3만여개의 라우트를 갖는 MAE - WEST 백본 라우터의 라우트 분포를 보여준다. 그림에서 보면 프리픽스는 길이 8에서 32까지 분포되어 있으며, 프리픽스 길이 24가 전체의 반 정도를 차지함을 알 수 있다.

표 1은 CIDR 에서의 프리픽스 통합(prefix aggregation)을 통한 프리픽스의 계층 구조를 알아보기 위하여 몇몇 백본 라우터에서의 실제 라우트 데이타를 분석한 것이다. 이러한 분석을 통하여 라우팅 데이타는 두 레벨 이상의 계층 구조를 갖으므로, 입력된 패킷은 복수개의 프리픽스와 일치할 수 있어 LPM이 수행되어야 하고, 완전 균형 트리의 구성이 어려움을 알 수 있다.

III. 패킷 분류 구조

패킷 분류 능력을 가진 라우터를 flow - aware 라우터라고 하는데, 라우터에 들어 오는 패킷들의 클래스를 나누고, 각 클래스에 따라 다른 서비스를 제공한다는 점에서 전통적인 라우터와 구별 된다. 각각의 클래스는 주로 패킷 헤더의 여러 필드들로 구성된 룰들에 의해 정의되며, 특정 응용 프로그램의 요구에 의해 결정된 룰들의 집합을 classifier라고 한다.

패킷 분류는 패킷의 클래스들을 정의해 놓은 classifier에 따라 라우터로 들어오는 패킷의 클래스를 결정하는 것을 의미한다.

즉, 패킷 분류란, classifier를 구성하고 있는 룰들이 F개의 필드로 이루어져 있을 때, 룰의 각 필드들과 패킷의 해당 필드들이 모두 일치하는 룰들을 찾고 그 중에서 가장 순위가 높은 룰을 선택하여, 패킷을 선택된 룰에서 규정하는 바와 같이 처리하는 과정이다. 패킷 분류 능력을 가진 flow - aware 라우터는 packet filtering, traffic rate limiting, policy based routing 등의 서비스를 제

〈표 1〉 라우팅 데이타 분석 (Enclosure/Disjoint Prefix)

MAE-WEST	14937	315/13083	5/1526	0/8	-	-	320/14617
AADS	20204	371/17734	1/2097	0/1	-	-	372/19832
PAC BELL	20519	360/18152	2/2003	0/2	-	-	362/20157
MAE-EAST	39464	1120/30838	28/7427	0/51	-	-	1148/38316
FUNET	40905	1288/24417	297/14034	12/835	0/22	-	1597/39308

공할 수 있고 또한, 클래스에 따라 accounting과 billing도 가능하게 한다.

Classifier를 이루는 룰의 각 필드들은 네트워크 계층 헤더에서 어플리케이션 계층 헤더까지 다양한 헤더 필드들을 가질 수 있다. 각각의 필드는 IP주소 검색에서 사용되는 것과 같이 <값, 길이>를 가질 수도 있고, '1034보다 큰', '23과 같은', '6070부터 6080사이인'과 같이 좀더 일반적인 표현도 가능하다. 여기에 특정 필드는 그 값을 고려하지 않겠다는 와일드카드 연산이 추가된다.

Classifier의 검색 시간을 줄이기 위하여 여러 가지 패킷 분류 방법들이 제안되고 있는데, 그 접근 방법에 따라 크게 trie 구조와 같은 기본적인 데이터 구조를 이용한 방법들과 룰의 한 필드가 차지하는 영역에 따른 기하학적 구조를 이용한 방법, 그리고 경험적 특성을 이용한 방법들로 분류될 수 있다.

1. 선형 검색 (linear search)

Classifier를 저장하는 가장 기본적인 구조는 룰들을 우선 순위에 따라 구성하는 것이다. 선형 검색은 룰들을 순차적으로 검색해 나가는 방법으로 매치하는 룰을 찾거나 리스트에 끝에 도달하면 검색이 종료된다. 선형 검색의 경우 N개의 룰을 저장하기 위하여 O(N)의 메모리와 O(N)의 검색 시간을 요구한다. 선형 검색 방법은 O(N)의 검색 시간 때문에 확장성이 나쁜 단점을 가지고 있다.

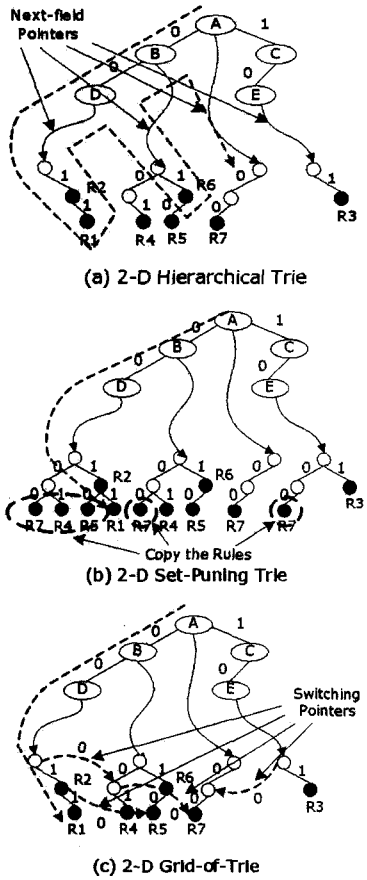
2. Trie에 기초한 방식

Classifier를 구성하는 필드 중 프리픽스로 표현되는 필드들(IP 주소의 destination prefix, source prefix)를 구성하는 방식으로 많이 연구되고 있는 것이 trie 구조를 이용한 방법이다. 그림 4는 trie 구조에 기초한 방식들의 예를 보이고 있다. Hierarchical trie는 d - dimension을 가지는 classifier를 각각의 dimension에 대하여 trie를 구성하고 D 개의 trie를 계층적으로 연결한 것으로 그림 4의 (a)가 여기에 해당한다. 2 - D hierarchical trie의 경우, F1 필드 만으로 구성된 하나의 F1 - trie와 동일한 F1 값을 가지는 룰들로 구성된 여러 개의 F2 - trie들로 이루어진다. F1 - trie의 각 노드들은 그 노드에 해당하는 F2 - trie를 가리키는 next - field 포인터를 갖게 된다. Packet classification에서 매치하는 룰을 찾는 것은 IP 주소 검색과 같이 LPM을 찾는 것이 아니라 패킷이 매치할 수 있는 여러 룰 중 가장

순위가 높은 룰을 찾아야 하기때문에 trie 상에서 가능한 모든 경로를 따라 검색을 수행해야 하므로 'backtracking'이 요구되며, 이 때문에 프리픽스의 최대 길이를 W라 하고 d개의 필드로 구성된 경우, $O(W^d)$의 검색 시간이 요구된다. 그림 4의 (b)에 해당하는 set - pruning trie는 룰을 복사한 후 저장하여 hierarchical trie에서의 backtracking을 없앰으로서 검색 시간을 개선한 것으로, F2 - trie를 구성함에 있어 자신의 노드에 해당하는 룰뿐만이아니라 그 노드의 프리픽스를 포함하는 상위 프리픽스가 가지는 룰들을 복사하여 모두 포함하는 방법이다. 따라서 검색 시간을 hierarchical trie의 $O(W^d)$에서 O(dW)로 줄일 수 있다는 장점을 지닌다. 그러나 set - pruning trie는 향상된 검색 시간에 대한 패널티로 메모리 요구량이 $O(N^dW)$로 증가하게 된다. 이밖에도 전처리 과정과 switching pointer를 통해 set - pruning trie의 이점인 단축시간을 그대로 유지하되, 룰을 복사하여 저장하는 단점을 해결한 방법인 grid - of - trie(그림 4의 (c))가 있다. Grid - of - trie는 메모리 요구량을 O(N)으로 유지하면서도 검색시간을 set - pruning trie와 같이 O(dW)로 줄인 방법이다. 그러나 각 노드의 룰들을 BMR(Best Matching Rule)로 업데이트하고 각 노드에서 switching pointer를 계산하는데 전처리가 과도하게 요구됨으로써 trie를 구성하고 업데이트하는데 시간이 많이 소요되게 된다.

Filter	F1	F2
R1	00*	11*
R2	00*	1*
R3	10*	1*
R4	0*	01*
R5	0*	10*
R6	0*	1*
R7	*	00*

(001, 110)



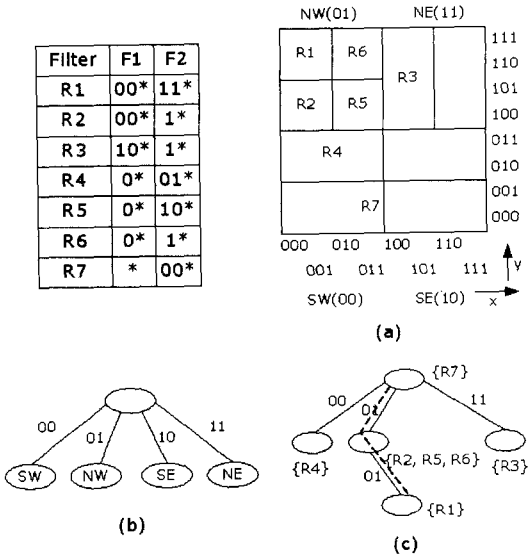
〈그림 4〉 2-D Tries [8]

3. 기하학적 구조에 기초한 방식

기하학적 구조를 이용하는 방법은 룰의 각 필드를 range로 표현하고 range를 줄여가면서 패킷이 속하는 가장 구체적인 range를 찾는 방법이다. 대표적인 것으로는 cross - producting과 AQT(area - based quadtree)가 있다. Cross - producting은 각 필드에 대해 일차원 검색을 각각 수행한 후 그 결과를 포인터로 하여 미리 계산된 cross - product 테이블에서 매치하는 룰을

얻는 방법이다. 우선, 룰을 구성하는 각 필드들에 대하여, 필드를 구성하는 케이스들을 구한다. 그리고 이러한 케이스들의 조합을 가지고 cross - producting 테이블을 구성하게 된다. 각 조합에 대하여 BMR이 구해져 있으므로, 일단 각 필드에 대하여 검색을 수행하여 그 필드가 해당하는 케이스를 구하게 되면 cross - producting 테이블로부터 바로 BMR을 얻을 수 있어 빠른 검색이 가능하다. 각 필드에 대한 케이스 검색 시간을 tRL이라고 했을 때, $O(dt_{RL})$의 검색 시간이 소요되고 병렬 수행시에는 검색 시간이 $O(t_{RL})$로 단축된다. 그러나 cross - producting테이블을 위하여 $O(N^d)$의 큰 메모리를 요구하게 된다. 더군다나 이 방식은 룰들이 가지는 모든 케이스에 대한 테이블을 가지므로 새로운 룰이 추가되거나 삭제되는 경우 cross - producting 테이블을 새로 구성해야하는 단점을 가지게 된다. 따라서 적은 수의 룰들로 구성된 static classifier에 적합한 방식이라 할 수 있다.

AQT는 매 단계에서 프리픽스로 표현된 두 필드를 동시에 보면서 전체 검색 범위를 1/4로 줄여가며 패킷이 해당하는 range 내의 룰들을 얻는 방법이다. 즉 매 단계에서 source prefix와 destination prefix 각각의 range가 이등분되어 두 프리픽스로 표현된 정사각 영역이 같은 크기의 네 개의 영역으로 분할된다. 분할 과정이 L번 반복되면, 동일한 크기의 4^L개의 영역으로 분할되고, 분할된 각 영역은 Lbit 길이의 프리픽스 쌍으로 표현된다. AQT에서 취하고 있는 검색 영역 분할 방식은 두 프리픽스를 동시에 한 비트씩 진행함으로 각 영역은 같은 길이의 프리픽스로만 표현된다. 그러나 룰들은 필드들의 프리픽스 길이가 각기 다를 수 있다. AQT에서는 룰을 이루고 있는 두 프리픽스의 길이 중 한 쪽이라도 검색의 영역이 나타내는 프리픽스 길이와 일치하고 룰이 나타내는 영역이 그 영역에 완전히 속하면 해당 룰을 그 영역의 CFS(Crossing Fliter Set)으로 정의하여 각 영역에서 CFS에 대한 선형 검색을 수행하게 된다. 그림 5는 AQT 구조를 예를 들어 보이고 있다. AQT 구조에서 각 룰들은 특정 노드에서 1번씩만 저장되므로 N개의 룰에 대하여 O(N)의 메모리를 요구하며, 프리픽스의 길이가 W일 때, 최대 O(WN)의 검색 시간을 요하게 된다.



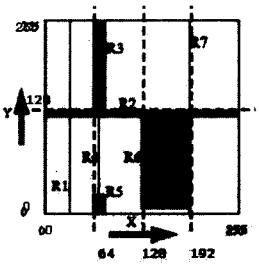
〈그림 5〉 AQT 구조 [8]

4. 경험적 특성을 이용한 방식

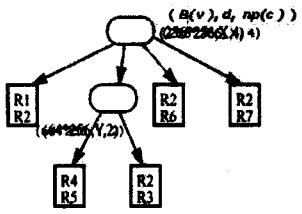
다차원 검색 영역을 분할하고 특정 영역에 속하는 룰만을 검색함으로써 검색해 보아야 하는 룰의 개수를 줄이는 방법 중 하나인 AQT에 대하여 위에서 언급하였다. HiCuts(Hierarchical Intelligent Cuttings) 역시 같은 개념을 바탕으로

Rule	X range	Y range
R1	0-31	0-255
R2	0-255	128-131
R3	64-71	128-255
R4	67-67	0-127
R5	64-71	0-15
R6	128-191	4-131
R7	192-192	0-255

(a) Example Classifier



(b) Space Division



(c) Decision Tree

〈그림 6〉 Hierarchical Intelligent Cuttings [8]

제안된 방식으로, AQT가 전체 검색 영역을 동일한 크기의 정사각형 4개로 영역으로 일괄적으로 분할하는 데 비하여 HiCuts의 경우에는 매 분할의 과정에서 classifier의 특성을 파악하여 영역을 분할할 dimension과 몇 개의 영역으로 분할할 것인지를 정하게 된다. HiCuts는 분할된 영역을 바탕으로 decision 트리를 구성하는데 decision 트리의 각 leaf는 미리 정해진 적은 수의 룰을 가지고 있게 된다. 검색 과정은 decision 트리를 따라 내려가서 leaf에 도달하면 그 leaf에 저장된 룰에 대하여 순차적 검색을 하게 되는 방식이다. 그림 6은 HiCuts의 예이다. (a)의 작은 classifier는 (b)와 같은 검색영역으로 표현되고, HiCuts에 의해 (c)와 같은 decision 트리로 구성된다. HiCuts의 경우 decision 트리 구성을 위하여 선처리 시간이 많이 요구되고, classifier의 필드들이 가지는 범위 분포에 따라 검색 시간의 의존도가 크다는 단점을 가지고 있다.

Classifier의 tuple에 대한 특성을 고려한 방법으로는 tuple space search가 있다. Tuple space search는 classifier를 구성하는 필드 중 (src_pfx_length, dst_pfx_length) 쌍으로 구성된 tuple의 개수는 많지 않다는 관찰에 그 출발점을 두고 있다. 이 방법은 프리픽스의 길이 쌍으로 tuple을 구성하고 tuple들을 순차적으로 검색하되, 각 tuple에 대해서는 해쉬을 적용하는 방식이다. 따라서 tuple의 수만큼 메모리 액세스가 요구된다.

Ⅳ. 결언

패킷 포워딩 기술의 난제와 이를 해결하기 위하여 연구되어온 IP 주소 검색 및 패킷 분류 구조들에 대하여 살펴보았다. IP 주소 검색과 패킷 분류는 multi - protocol label switching(MPLS)과 같이 고정된 길이를 갖는 레이블에 의한 스위칭 방식이 널리 사용되지 않는 한 앞으로도 당분간 고속 백본 라우터 설계의 이슈로 남아있을 전망이다. 이는 스위칭 장비에 입력된 모든 패킷에 대해 실시간으로 수행되어야 할 뿐만 아니라, 매우 큰 메모리를 요구하며, 수행되어야 하는 작업의 복잡성이 크기 때문이다. 특별히 load balancing이나 웹 스위칭과 같이 payload에 기초한 content 스위칭을 요구하는 다양한 응용 프로그램의 등장으로 classifier의 크기가 필드수에 있어서나 룰 수에 있어서 빠르게 증가하고 있어 패킷 분류 작업은 더욱 어려워지고 있다. IP 주소 검색과 패킷 분류를 위한 효율적인 알고리듬과 구조에 관한 연구가 활발히 진행되어야 할 것이다.

참고문헌

[1] M.A. Ruiz - Sanchex, E.W. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," IEEE Network, pp.8 - 23, March/April 2001.

[2] Mohammad Peyravian, Gordon Davis, and Jean Calvignac, "Search Engine Implications for Network Processor Efficiency," IEEE Network, pp.12 - 20, July/August 2003.

[3] 정여진, 이보미, 임혜숙, "복수의 해쉬 함수를 이용한 병렬 IP 어드레스 검색 구조," 한국통신학회 논문지, 29권, 2B호, pp.158 - 166, 2004.

[4] H. Lim and Y. Jung, "A Parallel Multiple Hashing Architecture for IP Address Lookup," Proc. IEEE HPSR2004, pp.91 - 98, Apr. 2004.

[5] D. Yu, B. C. Smith, and B. Wei, "Forwarding Engine for Fast Routing Lookups and Updates," Proc.IEEE GLOBECOM' 99, pp.1556 - 1564, 1999.

[6] N. Yazdani and P. S. Min, "Fast and Scalable Schemes for the IP Address Lookup Problem," Proc. IEEE HPSR2000, pp 83 - 92, 2000.

[7] H. Lim and B. Lee, "A New Pipelined Binary Search Architecture for IP Address Lookup", Proc. IEEE HPSR2004, pp.86 - 90, Apr. 2004.

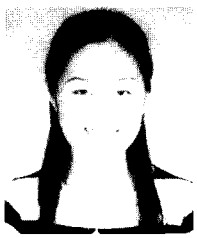
[8] H.J. Chao, "Next Generation Routers," Proceedings of the IEEE, Vol.90, No.9, pp.1518 - 1558, Sep, 2002.

[9] J. van Lunteren and A.P.J. Engbersen, "Multi - field packet classification using Ternary CAM," Electronics Letters, 3rd January 2002.

저자소개

임 혜 숙

1986년 서울대학교 공과대학 제어계측공학과 학사
1986년 - 1989년 연구원, 삼성 Hewlett Packard
1991년 서울대학교 공과대학 제어계측공학과 석사, 신호처리 전공
1996년 Electrical and Computer Engineering, The University of Texas at Austin, Austin, Tx, Ph.D.
1996년 - 2000년 Member of Technical Staff, Bell Labs, Lucent Technologies, Murray Hill, NJ
2000년 - 2002년 Hardware Engineer, Cisco Systems, San Jose, CA
2002년 - 현 재 이화여자대학교 공과대학 정보통신학과 조교수
주관심분야 Internet Router/Switch Design, MPLS, Scalable Video Coding



정 여 진

2003년 이화여자대학교 공과대학 정보통신학과 학사
주관심분야 Internet Router/Switch Design