2 차원 필터에 대한 빠른 패킷 분류 기법†

정상훈° 윤현수 조정완
한국과학기술원 전자 전산학과 전산학전공
{shchung°, hyoon, jwcho}@camars.kaist.ac.kr

A High-Speed Packet Classification Scheme For 2 Dimensional Filters

Sang-Hun Chung° Hyunsoo Yoon Jung-Wan Cho
Dept. of CS, Div. of EECS, Korea Advanced Institute of Science and Technology

요 약

패킷 분류는 품질보장(QoS), VPN(Virtual Private Network), 고성능 방화벽(high speed firewall), 인터넷 사용료 부과(pricing)를 제공하는 차세대 라우터에 반드시 필요한 기능이다. 라우터는 송신 주소, 수신 주소, 프로토콜 타입, 혹은 포트 번호와 같은 패킷 헤더의 여러 필드를 주어진 필터 리스트와 비교하여 패킷을 분류한다. 기존에 제시된 하드웨어 기반의 패킷 분류 기법은 빠른 검색 시간을 제공하지만 확장성과 테이블 갱신 면에서 문제점이 있다. 본 논문에서는 하드웨어 기반의 빠르고 확장성있고 갱신이 가능한 2차원 필드 검색 기법을 제시한다. 차후 연구에서는 본 기법을 보다 면밀히 분석하고 다차원 필터 검색이 가능하도록 확장한 기법을 제시하겠다.

1. 서 론

차세대 인터넷에서는 DiffServ(Differentiated Service)와 같은 품질 보장 기법, 인터넷 사용료 부과, 고성능 방화벽, VPN과 같은 향상된 서비스를 제공할 것이다. DiffServ는 사용자의 요구에 따라 차별화된 전송 서비스를 제공하는 인터넷 표준안으로 사용자의 class를 분류할 수 있는 에지 라우터(edge router)를 필요로 한다. 인터넷 사용료 부과는 ISP 간의 전송료 및 사용자의 사용료 부과를 말하는데 송신 주소와 수신 주소에 따라 패킷량 또는 사용 시간 같은 측정량을 분류해서 저장해야 한다. 인터넷 방화벽은 시큐리티와 관련된 접근 차단 서비스이고 VPN은 공공망에 가상의 사설망을 구축하는 서비스인데 둘 다 특정 플로우(flow)를 빠르게 분류할 수 있어야 한다. 위와 같은 차세대 인터넷의 서비스는 공통적으로 패킷 분류를 필요로 한다.

기존의 라우터는 패킷의 수신 주소만을 대상으로 IP 주소 검색을 수행하지만 차세대 라우터는 패킷 헤더의 정보를 이용하여 패킷을 분류할 수 있어야 한다. 패킷 분류는 특정 패킷 헤더 필드에 대해 미리 정의된 필터 리스트(classifiers라고도 불림) 중에서 일치하는 필터를 찾는 작업이다. 패킷 헤더 중에서 가장 중요한 필드는 송·수신 주소, 송·수신 포트 번호과 프로토콜 타입이다. 필터 리스트는 해당 필드값과 명령으로 이루어진다. 라우터는 패킷 헤더를 파싱해서 각 필드를 구하고 이들을 미리 정해진 필터 리스트와 비교함으로써 패킷을 분류한다. 예를 들어서, (수신 주소 프리픽스, 송신 주소 프리픽스, 수신 포트 번호 범위, 송신 포트 번호 범위, 프로토콜 타입)으로 표기되는 2개의 필터가 (143.248.150.*, 143.248.173.*, 21, *, TCP)와 (143.248.173.*,143.248.143.*, 27,*, TCP)으로 주어질 때, 들어온 패킷이 (143.248.150.113, 143.248.173.2, 21, 7000, TCP)라면 첫번째 필터에 일치하게 된다.

다차원 필드의 패킷 분류는 IP 주소 검색보다 상위의 문제인 동시에 계산 기하학(Computational Geometry)의 포인트 위치 문제보다 제한적인 문제이다. 이는 프리픽스일치(prefix match)와 정일치(exact match)가 범위 일치(range match)보다 제한적이라는 점에 기인한다. 2차원 필드의 패킷분류는 5차원 필드 중에서 복잡한 형태의 프리픽스를 다룬다는 점에서 단순한 IP 주소 검색의 일반화가 아니다. 기존의 논문에서 제안된 2차원 패킷 분류는 IP 주소 검색보다 4 배이상의 검색시간이 걸리거나 필터의 수에 대해 확장성이 크게 떨어지는 문제점이 있다.

기존에 발표된 논문들은 크게 4가지로 분류될 수 있는데 트라이 기반 검색[4], 해쉬 기반 검색[5], 기하학 기반 검색[6]과 하드웨어 기반 검색[1][2]으로 나눌 수 있다. 그 중에 가장 빠른 검색을 보이는 것은 하드웨어 기반 검색이다. 논문 [1]에서는 필터를 특정 범위마다 비트 벡터화하여 저장하고 병렬적으로 검색하는 기법을 발표하였다. 이 기법은 필터 수에 대한 확장성이 크게 떨어지고 테이블의 부분 갱신이 불가능하다. 논문 [2]는 필터 수에 대한 확장성을 개선하기 위해 비트 벡터에 대한 통합적인 비트 벡터를 제시하였다. 그러나 필터 수에 대해 메모리 접근 수 측면의 확장성은 보완하였지만 부분 갱신이 불가능하다는 문제점이 존재한다.

본 논문은 충돌없는 필터[7]에서의 2차원 패킷분류 기법을 제시하고 다음의 목적을 추구한다. 1) 와이어 속도에 맞추어 패킷을 빠르게 분류한다. 2) 필터의 추가에 대해 테이블의 갱신이 가능하도록한다. 3) 테이블의 크기를 작게함으로써 필터 수에 대한 확장성을 높인다. 위의 목적을 이루기

† 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았고 대학 IT연구센터 육성·지원사업의 연구결과로수행되었음.

위해 3가지의 방법을 제시하였는데 첫째는 다단계 테이블 구조이고, 둘째는 각 필터의 길이를 이용한 테이블을 구성이고, 셋째는 중간 비트를 건너뜀으로써 테이블의 수를 줄이는 것이다.

논문의 구조는 다음과 같다. 2절에서는 본 논문에서 제안 패킷 분류 기법에 대해 자세히 설명하고 3절에서는 중간 시뮬레이션 결과에 대해 기술하겠다. 4절에서는 요약과 앞으로의 계획에 대해 논의하겠다.

2. 2차원 필드의 패킷 분류 기법

2차원 필드의 IPv4 패킷 분류에 대한 가장 단순한 방법은 그림 1에서 보듯이 32 bit의 송신 IP 주소와 수신 IP주소를 키로 하는 테이블을 구성하는 것이다. 테이블의 엔트리는 필터 id를 저장한다. 이렇게 테이블을 구성하면 한번의 메모리 접근으로 분류가 가능하지만 테이블의 크기가 최소한 2^{64} (=2^{32}*2^{32}*1bytes) bytes로 현재의 기술로 테이블 구성이 불가능하고 테이블 갱신 또한 마찬가지이다.

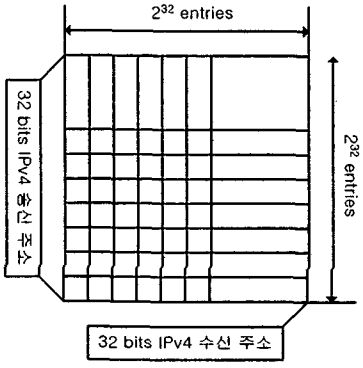

그림 1. 단순한 패킷 분류 기법

필터 리스트으로부터 구성된 테이블(이하 필터 테이블)의 크기를 줄기 위해서 다단계의 작은 필터 테이블들로 구성해야한다. 다단계의 필터 테이블은 그림 2처럼 4단계로 이루어진다. 각 단계의 필터 테이블은 64K개(=2^8* 2^8)의 엔트리를 갖고 각 엔트리에는 필터 id 또는 다음 단계의 필터 테이블을 가리키는 포인터가 저장된다. 엔트리의 크기는 필터의 수와 관련있다. 단계 k의 필터 테이블은 2개의 k번째 작은 주소(sub-address, SA)로 접근한다. (SA는 IP 주소를 4 개로 쪼갠 주소 일부이다.) 수신 주소가 a.b.c.d이고 송신 주소가 w.x.y.z일 때 검색하는 예를 보자. (a, w)를 키로 단계 1의 필터 테이블을 접근하면 가로 a와 세로 w가 만나는 엔트리를 구하게 된다. 이 엔트리에 저장된 값이 필터 id이면 이를 리턴한 후 종료하고 포인터이면 포인터가 가리키는 다음 단계의 필터 테이블에 대해서 앞의 과정을 반복한다.

다단계 테이블 구조는 최대 4번의 메모리 접근이 필요하므로 매우 빠른 검색이 가능하지만 시뮬레이션 결과 필터의 수가 커짐에 따라 메모리의 사용량이 매우 커지는 문제점이 있다. 테이블 단계의 수를 늘리면 메모리의 사용량이 줄어들게 되는 반면 메모리 접근 수가 증가하게 된다. 이는 접근 회수와 메모리 사용량간에 trade-off를 보여준다. 메모리 사용량을 더욱 줄이기 위해서 2가지의 방법을 고안하였다.

첫째, SA의 길이를 단계마다 같게 고정시키지 않고 필터의 프리픽스에 따라 다른 길이로 한다. 그림 3은 첫번째 방법의 유용성을 보여주는 예이다. 빠른 이해를 위해서 주소의 최대 길이는 4bits로 축소하였다. 각 필터는 서로 다른 길이의 송

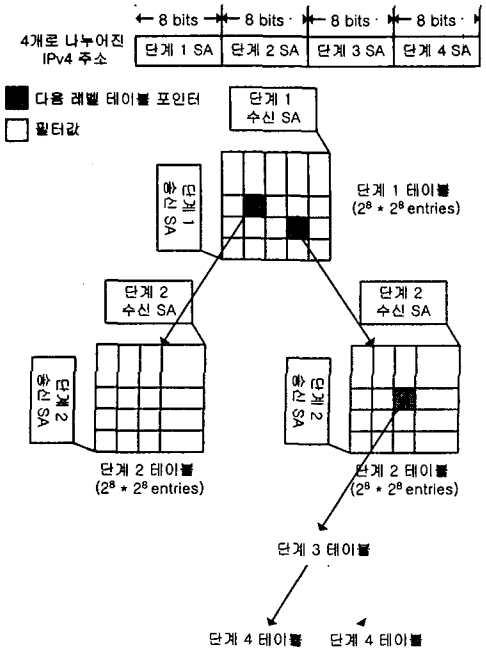

그림 2. 다단계 테이블 구조

신 프리픽스들로 구성되고 두 차원 필드의 프리픽스 길이를 고려하면 테이블의 크기를 줄일 수 있다. 먼저 테이블을 구성하는 방법에 대해 설명하겠다. 필터 F1= (000*,000*)은 첫 두비트가 (00,00)이므로 단계 1 테이블의 (00,00) 엔트리에 해당한다. 그리고, (00, 00) 엔트리에는 다음 단계 테이블을 가리키는 포인터가 저장된다. F1의 남은 SA인 (0*, 0*)은 (00, 00), (00, 01), (01, 00)과 (01, 01)에 해당하므로 4개의 엔트리에 필터 아이디가 저장된다.

두 차원의 필드 중에 짧은 길이의 프리픽스는 아래 단계로 내려갈 수록 더 이상의 검색에 필요하지 않는 경우가 있다. 필터 F8=(110*, 0*)의 경우가 이에 해당한다. 송신주소 프리픽스의 길이가 1 (<2)이고 F8과 프리픽스를 공유하는 필터가 없으므로 송신 주소 필드는 단계 2 테이블 검색에 필요가 없게 된다. 테이블 엔트리는 포인터 정보와 함께 다음 테이블 검색에 필요한 주소 길이 정보도 포함한다. F8은 첫 두 비트가 (11, 0*)이므로 (11,00)과 (11, 01) 엔트리에 해당한다. (11,00) 과 (11,01) 엔트리는 같은 값의 테이블 포인터가 저장된다. F8의 남은 SA인 (0*,*)으로부터 수신 주소 필드 0만 비교되므로 단계 1 테이블 (11,00)과 (11,01) 엔트리에 키길이를 표시하는 (1,0)이라는 정보도 저장한다. 단계 2 테이블은 일차원으로 구성되고 0 엔트리에 필터 아이디 8이 저장된다.

프리픽스를 공유하는 필터들로 단계 2 테이블을 구성할 때, 각 필드의 긴 길이의 프리픽스에 맞추어 테이블의 크기가 정해진다. 그림 3의 F6과 F7을 살펴보면 F6의 송신 주소 프리픽스가 4이므로 단계 2 테이블의 송신 주소 축은 $4(2^{4-2})$가 되고 수신 주소 축은 둘의 길이가 모두 3이므로 $2(2^{3-2})$가 된다.

필터 리스트의 프리픽스가 차지하는 공간은 전체 2차원 필드의 전체 주소 공간에 비하면 매우 희소하다. 기존 연구에

서 필터들이 서로 겹치는 부분이 대단히 적다고 발표하였다.

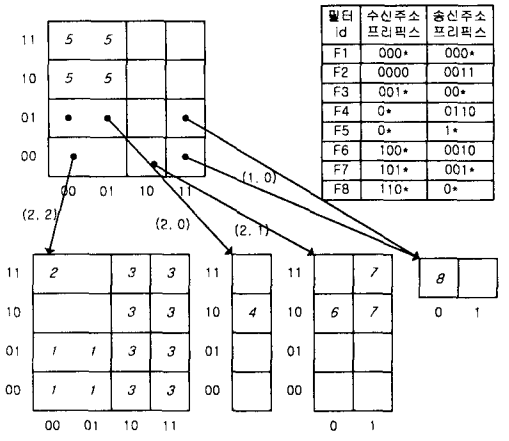

그림 3. 각 필터마다 다른 키 길이를 갖는 필터 테이블 예

이는 공유하는 프리픽스를 가진 필터의 수가 적다는 것을 의미한다. 즉, 단계 1 테이블에서 다음 단계로 내려갈수록 한 테이블에 저장된 필터의 수는 점점 줄어들게 된다. 검색의 중간 경로에 있는 테이블 중에 적은 수의 필터를 저장하기 위해 2^8x2^8 크기의 메모리를 할당하는 것은 낭비일 수 있다. 그래서 두번째 아이디어는 []에서 제안한 것을 2차원 필드 검색으로 확장하여 중간 경로의 프리픽스 비교를 건너뛰거나 특정비트에만 수행하고 가장 아래 단계의 테이블에 프리픽스 정보를 저장하여 마지막에 비교한다는 것이다. 이는 중간 단계의 테이블의 수와 테이블의 크기를 줄이게 되므로 메모리 요구량을 크게 줄일 수 있다.

그림 4는 중간 비트를 생략하여 만든 테이블 구조를 보여준다. 테이블 엔트리는 건너뛸 수 있는 중간 비트수에 대한 정보를 추가적으로 저장한다. 수신 주소가 11로 시작되고 송신 주소가 00으로 시작되는 패킷은 필터 F8과 일치하거나 일치하지 않거나 둘 중에 하나이다. F8가 일치하는지 알기 위해 중간 비트에 해당하는 테이블을 두는 것은 메모리의 낭비를 초래한다. 차라리 그림 4처럼 중간 비트는 생략해서 마지막 테이블 엔트리에 저장된 비트(skip bits)와 비교하는 것이 메모리의 낭비도 줄이고 메모리 접근 시간도 줄일 수 있다.

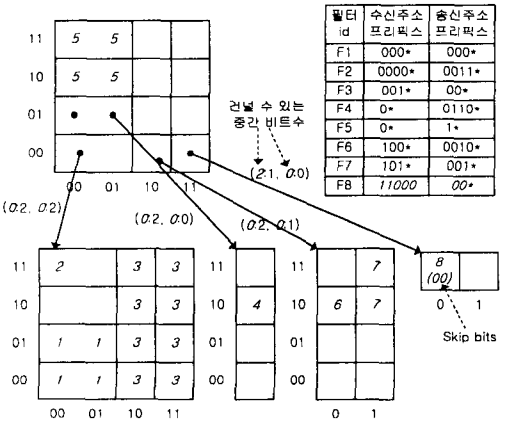

그림 4. 중간 비트를 건너뛰는 필터 테이블 예

3. 중간 시뮬레이션 결과

현재 시뮬레이션은 계속 진행 중인 상태인데, 앞서 기술한 2가지 개선 방안을 설계하고 있다. 기본 아이디어인 SA의 크기를 8-8-8-8(BASIC-1)로 하였을 때 필요한 메모리 요구량과 SA의 크기를 8-4-4-4-4-4-4(BASIC-2)로 하였을 때 필요한 메모리 요구량에 대한 시뮬레이션 결과를 기술하겠다. 시뮬레이션에 사용한 필터는 MAE-EAST[8]의 프리픽스를 이용해서 인위적으로 만들었고 필터의 수는 500이하로 한정하였다.

놀랍게도 테이블 1에서 보는 바와 같이 BASIC-1과 BASIC-2의 테이블의 크기는 수십배정도의 차이가 난다. BASIC-1은 메모리 접근 횟수가 최악의 상황에서 3번 덜 드는 장점이 있지만 많은 메모리가 필요하기 때문에 필터의 수가 500인 환경에도 부적합해 보인다. BASIC-2는 메모리 요구량면에서 BASIC-1보다 탁월한데 2개의 개선 방안을 적용하면 테이블 크기를 더욱 줄일 것으로 예상된다.

테이블 1. 두 방법의 필터 수에 대한 테이블의 크기 (MB)

	100	200	300	400	500
BASIC-1	23.8	45.9	68.8	88.6	113
BASIC-2	0.457	0.75	1.06	1.32	1.67

4. 요약 및 향후 연구 방향

패킷 분류는 차세대 라우터에 반드시 필요한 기능이다. 느린 패킷 분류는 네트웍에 병목을 야기할 것으로 예상되므로 빠른 패킷 분류 기법에 대한 연구가 필요하다. 또한 현재 라우터처럼 많은 횟수의 갱신과 확장성도 염두해야한다. 본 논문은 2 차원 필드에 대한 패킷 분류 기법을 제시하였다. 빠른 검색과 확장성과 갱신을 위해 크게 3가지의 방법을 제시하였다. 첫째는 다단계 테이블 구조이고 둘째는 각 필터의 길이에 맞게 테이블 을 구성하는 것이고, 셋째는 중간 비트를 건너뜀으로써 테이블의 수를 줄이는 것이다. 차후 연구에서는 2가지 개선 방법을 구현하여 확장성면에 대해 보다 면밀히 분석하고 효율적인 갱신 알고리즘도 제시하겠다. 그리고 다차원 필터 검색이 가능하도록 확장할 계획이다.

참고 문헌

[1] T.V. Lakshman and D. Stiliadis, "*High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching*", SIGCOMM'98.

[2] F. Baboescu and G. Varghese, "*Scalable Packet Classification*", SIGCOMM'01.

[3] S. Jean, S-H. Chung, J. Kim, and H. Yoon, "*Scalable IP lookup scheme with small forwarding table for gigabit routers*", IEE electronics letters, vol. 38, no. 6, pp. 298-299, March, 2002.

[4] V.Srinivasan, G. Vargese, S. Suri, and M. Waldvogel, "*Fast and Scalable Layer Four Switching*", SIGCOMM'98.

[5] V.Srinivasan, S. Suri, and G. Vargese, "*Packet Class-ification using Tuple Space Search*", SIGCOMM'99.

[6] A. Feldmann and S. Muthukrishnan, "*Tradeoffs for Packet Classification*", IEEE Infocom'2000.

[7] A. Hari, S. Suri, and G. Parulkar, "*Detecting and Resolving Packet Filter Conflicts*", IEEE Infocom'2000.

[8] IPMA Statistics, http://www.merit.edu/ipma.