제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

트라이 기반의 2차원 필터 패킷 분류 기법

오승현
동국대학교 컴퓨터학과
e-mail:shoh@dongguk.ac.kr

A Trie Based Packet Classification Scheme for 2-Dimensional Filters

Seung-Hyun Oh
Dept of Computer Science, Dongguk University

요 약

라우터에서 패킷을 특정 "플로우"로 분류하는 과정을 패킷 분류라고 한다. 어떤 플로우에 속한 모든 패킷들은 미리 정의된 규칙을 따르며, 라우터는 비슷한 처리과정을 제공한다. 예를 들면, 같은 출발지 IP주소와 도착지 IP주소를 가진 패킷들은 하나의 플로우로 분류된다. 이러한 패킷 분류는 최선형(best-effort) 서비스가 아닌 품질보장(QoS), VPN, 방화벽 서비스 등과 같이 플로우와 플로우를 구별하여야하는 서비스 제공에 필요하다. 일반적으로 다수의 필터에 대한 패킷 분류는 매우 어려운 문제로 하드웨어 및 소프트웨어 기반의 다양한 알고리즘들이 제시되었다. 본 논문은 트라이를 기반으로 작은 메모리 공간을 사용하면서 빠른 패킷 분류를 할 수 있는 기법을 제시한다.

1. 서론

최근이전까지 라우터는 best-effort 개념의 최선형 서비스만을 제공하였다. 그러나 최근 들어 라우터들은 응용 프로그램 또는 서비스에 따라 차별화된 품질(quality)의 서비스를 제공하도록 요구받고 있다. 이런 차별화된 서비스 품질을 제공하기 위해서는 라우터에서 어드미션 제어(admission control), 자원예약, 플로우별 큐 서비스 및 공평한 스케줄링 기능을 구현하여야한다. 이러한 기능들은 모두 라우터가 패킷을 분류를 할 수 있어야 가능하다는 공통점이 있다.

플로우는 필터 또는 규칙(rule)에 의해 구분된다. 규칙의 집합을 필터 리스트 또는 분류자(classifier)라고도 부르며 규칙에 기술된 값과 일치하는 패킷 헤더의 필드(field)를 찾는 과정을 패킷 분류라고 한다. 라우터의 포워딩 기능은 패킷 헤더의 목적지 IP 주소를 라우팅 테이블의 목적지 프리픽스에 대해 최장프리픽스 검색을 하고, 검색결과로 선택된 다음-홉 주소로 패킷을 포워딩하는데 이 동작은 규칙에 따른 처리과정으로 볼 수 있으므로 1차원 패킷 분류의 특별한 경우로 구분할 수 있다.

패킷 분류에 적용되는 규칙이나 필터는 d개 요소를 갖는다. R[i]는 규칙 R의 i번째 요소를 의미하며, 패킷 헤더의 i번째 필드에 대한 값을 의미한다. 어떤 패킷 P와 규칙 R이 일치되었다는 것은 d개의 R[i] 요소가 패킷 P의 헤더 필드와 모두 일치하였다는 것이다. 패킷 분류에서 사용되는 필터의 종류는 목적지 및 송신지 IP주소, 목적지 및 송신지 포트번호와 프로토콜 번호의 다섯 가지를 들 수 있으며, 구분해야할 플로우의 성격에 따라 필요한 필터의 개수와 종류가 달라진다. 예를 들어, VPN 서비스는 목적지와 송신지의 IP주소 두 개 요소로 분류될 수 있고, 특정 ISP로부터 들어온 트래픽은 송신지의 링크계층 주소만을 사용한다. 참고로, 패킷 분류를 위해 사용되는 요소의 개수가 d개이면 d-차원 필터라고 한다.

패킷 분류 알고리즘의 성능은 다음과 같은 몇 가지 요소[1]로 평가할 수 있다. · 검색 속도-고속의 통신링크는 더 빠른 분류속도를 필요로 한다. · 작은 메모리 공간-작은 저장 공간은 고속의 SRAM 이용을 가능하게 한다. · 많은 규칙을 가진 실제 필터 리스트를 지원할 수 있어야한다. · 고속의 필터 리스트 수정-필터 리스트의 규칙이 수정되면 자료구조도 수정되어야 한다. · 다차원 규칙에 대한 비례성(scalability) · 유연성-규칙의 요소가 다양한 형태(프리픽스, 연산자, 와일드카드 등)를 지원할 수 있어야 한다.

기존에 제시된 연구 성과들은 하드웨어 기반[2], 전통적인 검색방법[3], 기하학적 특성을 이용하는 방법[4], 기타[5,6] 등으로 구분할 수 있다. 병렬 하드웨어를 이용하여 빠른 검색을 지원하는 연구[2]에서는 프리픽스를 일정 구간마다 비트-벡터로 구성하여 각 요소를 검색하고, 검색결과를 모아서 비교함으로써 최종적으로 일치하는 규칙을 결정한다. 그러나 빠른 검색속도에 비해 비트-벡터의 계산과정이 필터 리스트의 점진적 수정을 지원하지 못한다. 전통적인 검색방법은 필터 리스트의 규칙들을 순차적으로 검색한다. 간단하고 효율적으로 메모리 공간을 사용하지만 비례성이 나쁘다. 계층적으로 트라이를 연결해서 검색하는 방법도 있다. d-차원의 규칙은 d개의 트라이가 계층적으로 연결되는 trie-of-trie 형태를 구성한다. Trie-of-trie는 재귀적 검색으로 검색시간이 길고, 공간이 커진다. 반면에 Set-pruning 트라이를 이용하는 연구[3]은 계층적 트라이 구조에서 중복해서 표시되는 규칙정보를 제거하여 재귀적 검색을 하지 않음으로써 검색시간을 단축하였다. 규칙이 갖는 기하학적 특성을 이용하는 연구[4]는 grid-of-trie 방법과 cross-producting 방법을 제안하였다. [4]는 이차원 필터를 검색 대상으로 switch pointer를 도입하여 O(W) (W=프리픽스 길이)의 속도를 제공한다. Cross-producting은 [3]에서 동시에 제안된 방법으로 필터 리스트에 포함된 모든 규칙을 조합하여 해시 테이블을 구성하였다. 해시 검색의 빠른 속도를 제공할 수 있지만 O(Nd) (N=규칙의 개수,d=요소의 개수) 의 큰 공간을 사용한다.

본 논문은 필터간 충돌[7]이 없는 상태에서 트라이를 기반으로 한 고속의 검색속도와 작은 크기를 제공하는 새로운 2차원 패킷 분류 기법 TBMF(trie based bit-map filtering)을 제안한다. TBMF는 목적지 IP주소와 송신지 IP주소의 두 개의 프리픽스 요소를 갖는 규칙에 대해 매우 작은 메모리 공간을 사용하는 자료구조를 제공함으로써 고속의 SRAM에서 패킷 분류를 할 수 있다. 자료구조의 압축 때문에 늘어난 계산 요구량은 빠르게 고속화되는 프로세서의 처리시간으로 희석되어 TBMF의 성능에 미치는 영향은 매우 미약한 것으로 파악된다.

2. 트라이 기반의 비트-맵 필터링 기법

IPv4 주소를 사용하는 2차원 필터 리스트를 검색하는 방법은 그림 1에서 보듯이 가능한 모든 IP 주소를 키로 테이블을 구성하는 것이다. 그러나 이 방법은 단 한번의 검색으로 패킷과 일치하는 규칙을 찾을 수 있지만 거대한 메모리를 요구하여 실현성이 없다.

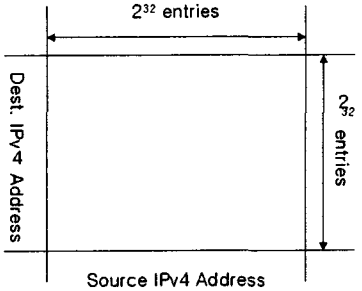

그림 1 단순한 이차원 프리픽스 규칙의 패킷 분류방법

전통적인 라우터의 포워딩 검색방법으로 트라이[8]를 많이 사용하였다. 트라이는 일종의 이진트리로 노드와 가지로 구성되며, 프리픽스 부분을 공유하는 비트열을 검색하는 방법으로 사용되는데 루트노드 아래의 가지들은 0 또는 1을 의미한다. 검색하고자 하는 비트열의 비트 값 0/1을 따라 트라이를 탐색한 후 리프노드를 만나면 리프노드는 다음-홉 주소와 같은 "행동규칙"에 해당하는 규칙을 찾을 수 있다. 이때 리프노드는 루트로부터 연결된 가지의 0과 1의 조합이 구성한 프리픽스와 일치하며 노드에 저장된 행동규칙은 검색된 프리픽스의 규칙이다.

표 1은 프리픽스로 구성된 이차원 필터 리스트 예제이고, 그림 2는 이 필터 리스트의 F1 요소에 대한 트라이를 각 노드에 대해 가지가 없거나 두 개의 가지를 갖는 완전이진 트라이[8]로 변경한 후의 구조이다. 완전이진 트라이 변경은 본 논문이 트라이

로부터 비트-맵을 구성하기 때문이다. 구성방법과 원리는 [9]을 참조할 수 있다. 그림 2에서 점선으로 표시된 노드와 가지는 완전이진 트라이의 규칙에 따라 확장된 노드를 의미하며, 동시에 선-점 라인은 노드의 확장에 따라 R2, R5 규칙도 확장되었음을 나타낸다.

R1	00*	00*
R2	0*	01*
R3	1*	0*
R4	00*	0*
R5	0*	1*
R6	*	1*

표 1 프리픽스로 구성된
이차원 필터 리스트 예제

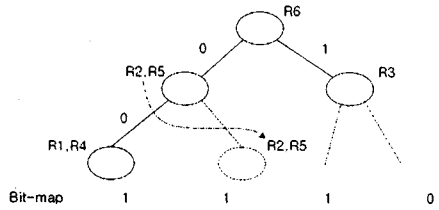

그림 2 표 1의 F1 프리픽스에 대한 완전이진
트라이

TBMF는 트라이를 기반으로 구성된 비트-맵을 이용하는 새로운 이차원 패킷 분류기법으로 기존의 연구들에 비해 더 작은 메모리 공간과 규칙요소의 증가에 비례하는 검색속도를 제공한다. TBMF는 기존의 많은 연구에서 사용된 트라이로 목적지 IP 프리픽스와 출발지 IP 프리픽스를 표현한다. IP 프리픽스의 비트 값을 따라 그림 2와 같이 구성되는 트라이는 비트열과 일치하는 곳에 위치한 노드에는 규칙을 저장한다. 따라서 트라이의 노드는 규칙노드-규칙을 저장한 노드-와 그렇지 않은 빈 노드로 분리된다. IPv4 주소 프리픽스의 길이 W가 최대 32이므로 트라이의 최대 깊이는 32이며, 최대 2^{32}개의 리프노드가 만들어진다. 최대 2^{32}개의 리프노드에 대한 정보를 테이블로 저장하는 것은 거대한 공간이 필요하므로 실현성이 없다.

메모리 공간을 줄이기 위해서 TBMF는 리프노드 정보를 하나의 비트로 표현한다. 즉, 비트 1은 규칙노드를 의미하고, 비트 0은 빈 노드를 표시하고, 이렇게 정해진 모든 비트를 수집하여 비트-맵을 구성한다. 그림 2에서 비트 값이 할당된 예제를 볼 수 있다. 트라이의 깊이가 2이므로 비트열의 길이는 4이고, R3을 저장하고 있는 노드는 두 비트를 할당하며 이때 좌측의 노드만 1을 할당하면 충분하다. 그러나 깊이 32에서 비트-맵을 구성하면 한번의 검색으로 노드에 저장된 규칙을 찾을 수 있지만 약 4GB의 메모리가 필요하므로 실질적인 방법으로 볼 수 없으므로 트라이를 여러 단계로 분리하여 비트-맵을 구성하여야 한다.

비트-맵에서의 검색은 비트-맵의 각 비트 1이 규칙노드의 순서 즉, 필터 리스트 테이블의 인덱스이므로 비트-맵의 첫 번째 비트부터 검색할 프리픽스 비트열 값이 지정하는 곳 사이에 있는 비트 1의 합이 필터 리스트의 인덱스이다. 그림 2의 비트-맵 '1110'에서 프리픽스 '01'을 검색하는 경우, '01'의 값은 1이므로 '1110'에서 좌측으로부터 2번째 위치까지 비트 1의 개수가 2이므로 표1에서 R2가 검색된다. 이때 루트노드에 있는 R6은 검색되지 않는다. 이것은 "숨은 규칙" 문제로 그림 2의 트라이를 두 비트 단위로 검색했기 때문이다. 숨은 규칙을 해결하는 방법은 비트-맵 구성단위 즉, 스트라이드(stride)를 결정할 때 규칙의 분포를 고려하여야 한다.

TBMF는 규칙노드가 존재하는 모든 레벨-트라이에서 동일한 깊이를 갖는 모든 노드를 의미-에서 분할을 하여 비트-맵을 구성한다. Srinivasan의 연구[4]에서는 이 문제를 리프노드에 숨은 규칙을 저장하는 별도의 변수를 둠으로서 해결하고 있으나 자료구조의 크기를 증가시킨다. 참고로, 트라이를 분할하여 비트-맵을 구성할 때는 트라이의 구조를 표시하기 위한 자료구조로 추가적인 비트-맵이 사용되며, 그 크기는 규칙노드를 위한 비트-맵과 같다.

TBMF에서 해결해야 할 또 하나의 문제는 어떤 규칙노드에서는 두 개 이상의 규칙이 존재하는 경우이다. 그림 2에서 <R2,R5>, <R1,R4>를 가진 노드가 그 예이다. 각 노드를 하나의 비트로 표시하므로 발생하는 문제이며, 완전이진 트라이를 구성할 때 증가한 규칙노드의 개수를 보완하기 위해 사용하는 중간 테이블에 링크 필드를 추가하여 두 개 이상의 규칙을 연결한다.

TBMF는 이차원 필터 F1, F2에 대해 각각의 트

라이와 비트-맵을 구성하고 검색을 수행한다. 각각의 필터검색 결과가 산출되면 두 개의 결과로 비트열을 구성하여 비트-AND를 수행한다. 결과 비트열에서 나온 규칙의 번호 중에서 가장 높은 우선순위의 규칙이 최종적으로 패킷 필터의 결과가 된다. 그림 2에서 F1:'01', F2:'01'을 검색하면 <F1:R2,R5,R6>과 <F2:R2,R3,R4>가 검색되고, 규칙 번호가 비트 1이되는 비트열 '0100011'과 '011100'의 비트-AND는 '010000'이므로 규칙 2가 패킷 분류의 결과가 된다.

TBMF는 두 개의 IP 주소 필터에 대해 별개의 트라이를 구성한다. W 길이의 필터 2개로 구성된 N개의 규칙을 가진 필터 리스트에 대해 최악 공간 복잡도는 2·NW 공간을 사용하므로 O(NW)이다. 이것은 trie-of-trie와 [4]의 grid-of-trie의 O(NdW)보다 좋은 성능을 갖는다. 트라이에서 비트맵이 구성되면 실제 공간 복잡도는 $\sum_{k=1}^{n} nNode \cdot 2^{k\text{th stride}}$ 이다. 이때 n은 규칙노드가 존재하는 레벨의 개수 즉, 비트-맵이 구성되는 개수이고, k'th stride는 각 레벨의 넓이로 분할된 트라이의 높이와 같다. nNode는 분할된 트라이에서 상하위의 트라이를 연결하는 노드의 개수이다. 즉, 상위 트라이의 리프노드 중에서 하위 트라이로 연결되는 리프노드마다 하위 트라이가 연결되기 때문이다.

TBMF의 검색성능은 각각의 필터 요소에 대해 순차적으로 검색하고 비트열 연산에 의해 일치된 규칙을 결정하는 경우 하나의 트라이를 검색하는데 2·W 시간이 소요되므로 O(W)의 시간복잡도를 갖는다. IPv4 주소의 길이가 32비트 이므로 최대 32회의 메모리 검색을 수행한다. 이것은 매우 느린 속도이며, 실제로는 비트-맵 분할 개수 n에 의존적이므로 O(n)이며 n<W이다. 또한 n번의 메모리 검색은 O(NW)의 자료구조 크기가 SRAM 크기보다 작으므로 저속의 메모리 검색이 고속의 SRAM 검색으로 대치될 수 있다. 참고로, 두 개의 필터에 대한 검색이 병렬로 진행될 경우 검색에 필요한 시간은 절반으로 감소한다.

3. 결론 및 향후연구

패킷 필터 검색은 최신 라우터의 다양한 차별성 서비스 제공에 필수적 기능이다. 또한 방화벽, IDS와 같은 보안 서비스에서도 매우 중요한 요소로 부각되고 있다. 본 연구의 TBMF 이차원 필터 검색 알고리즘은 비트-맵을 이용하여 SRAM 크기보다 작은 크기로 압축하여 느린 메모리 검색대신 고속의 SRAM 검색으로 빠른 패킷 검색을 지원할 수 있다.

향후의 연구방향은 설계된 알고리즘을 실제 필터 리스트를 이용하여 구현하고 계산성능을 검증하고, 5차원 이상의 다차원 필터에 대해 알고리즘을 확장하는 것이다.

참고문헌

[1] P. Gupta and N. McKeown "Algorithms for Packet Classification", IEEE Network, March/April 2001, pp. 24-32

[2] T.V. Laskman and D. Stiliadis "High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching", Proc. ACM Sigcomm, Sept. 1998, pp. 191-202

[3] P. Tsuchiya "A Search Algorithm for Table Entries with Non-contiguous Wildcarding," unpublished report, Bellcore.

[4] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel "Fast and Scalable Layer four Switching," Proc. ACM Sigcomm, Sept. 1998, pp. 203-214.

[5] M. M. Buddhikot, S. Suri, and M. Waldvogel "Space Decomposition Techniques for Fast Layer-4 Switching," Proc. Conf. Protocols for High Speed Networks, Aug. 1999, pp. 25-41.

[6] A. Feldman and S. Muthukrishnan, "Tradeoffs for Packet Classification," Proc. IEEE Infocom, vol. 3, Mar. 2000, pp. 1193-1202.

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

[8] S. Nillson and G. Karlsson "Fast Address Lookup for Internet Routers", Proc. IEEE Broadband Comm., April 1998

[9] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink "Small Forwarding tables for Fast Routing Lookups", Proc. ACM Sigcomm, Oct. 1997