A New Architecture for Packet Classification

이보미, 윤명희, 임혜숙*

이화여자대학교 SoC 설계 연구실
전화 02-3277-2809

EnBiT for Packet Classification

Bomi Lee, Myunghee Yoon, and Hyesook Lim

SoC Design Lab
Ewha Womans University
*E-mail : hlim@ ewha.ac.kr

Abstract

The process of categorizing packets into "class" in an Internet router is called packet classification. All packets with same class obey predefined rule specified in routing tables. Performing classification in real time on an arbitrary number of fields is a very challenge task. In this paper, we present a new algorithm named EnBiT-PC (EnBiT Packet Classification), and evaluate its performance against real classifiers in use today. We compare with previous algorithms, and found out that EnBiT-PC classify packets very efficiently and has relatively small storage requirements.

I. 서론

기존 인터넷 라우터의 기능은 들어온 패킷에 대하여 들어온 순서대로 패킷을 처리하는 best effort 를 기반으로 하였다. 하지만 인터넷 서비스 제공자(Internet Service Provider)나 인터넷 사용자들의 다양한 응용 프로그램에 대한 수요로 인해 인터넷 라우터에서는 응용 프로그램 별로 각기 다른 서비스를 제공하는 기능을 추가하게 되었다.

Packet classification 은 들어온 패킷을 flow 별로 categorize 하는 것으로, categorize 된 flow, 즉 class 별로 방화벽이나 QoS (Quality of Service)같은 best-of-the-best-effort 서비스를 제공한다. 클래스는 각 패킷의 여러 헤더 필드에 의해 구성된 룰에 의해 정의되며, 이 룰들의 집합을 classifier 라고 한다. 라우터는 classifier 를 구성하고 있는 룰들의 각 필드와 입력된 패킷의 해당 필드들이 모두 일치하는 룰을 찾은 뒤, 가장 순위가 높은 룰을 선택한다. 라우터는 이처럼 입력된 패킷을 classifier 에 따라 클래스를 나눈 뒤, 각 클래스 별로 다른 서비스를 제공하는데, 이러한 과정을 packet classification 이라 한다.

본 논문에서는 IP 주소 검색을 위해 사용되었던 EnBiT 구조를 packet classification 을 위해 적용하여 적절한 메모리 크기를 사용하면서 검색 시간을 줄일 수 있는 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2 장에서는 기존 packet classification 을 위해 제안되었던 scheme 들에 대해 살펴보고, 3 장에서는 본 논문에서 제안하는 구조에 대해 설명 한 후, 4 장에서는 실제 classifier 로 simulation 한 결과 및 다른 scheme 들과의 비교를 보이고, 마지막으로 결론을 맺는다.

II. Existing Schemes

최근 수년간 인터넷 라우터를 위해 다양한 packet

classification 방법들이 연구되었다. 여러가지 방법들을 접근 방식에 따라 크게 몇 가지로 구분하여 설명하면 다음과 같다.

첫번째는 trie 구조를 이용한 방식이다. 가장 기본이 되는 방법은 hierarchical trie[1]로, 룰을 구성하는 여러가지 필드 중 프리픽스로 표현되는 필드를 trie 로 구성하는 방식이다. IP 주소 검색에서 사용되었던 trie 구조는 1 가지 필드로 구성된 트리이고, 이것의 leaf 에 그 다음 필드를 계층적으로 연결시킴으로써 각 필드를 순차적으로 검색할 수 있게 한다. 하지만 이 방법은 패킷이 여러가지 룰에 매치할 때, trie 의 하나의 가지로 내려와서 매치하는 룰을 찾았다 하더라도, 더 높은 순위의 룰을 찾기 위해 back tracking 을 해야하는 단점이 있다. Back tracking 을 없애기 위해 set-prunig trie 에서는 trie 의 모든 path 에 매치하는 rule 을 복사하는 방법을 제시하였고, grid of tries 에서는 leaf 에 switching pointer 를 두어 다음 검색 되어져야 하는 rule 로 이동하는 방법을 제시하였다[2]. 이 방법들은 hierarchical trie 에 비해 검색시간이 짧고 back tracking 문제를 해결하였지만, 복잡한 선처리 과정이 요구되어 다이나믹한 classifier 에 적합하지 못하고 update 가 어려우며, 또한 두 개 이상의 필드에는 적용되기 어려운 단점이 있다.

두번째는 기하학적 구조를 이용한 방법이다. 룰의 각 필드를 range 로 표현하고 필드 별로 검색하면서 패킷이 속하는 가장 정확한 range 를 찾는 방식이다. Cross-Producting[1]은 필드별로 range 를 검색한 후, cross-product table 로 가서 매치하는 룰을 찾는 방식이다. 임의의 필드 개수에 적용 가능한 장점이 있지만, cross-product table 을 저장하는 데서 오는 큰 메모리 소요와 update 가 어려운 단점이 있다.

세번째로는 classifier 의 경험적 특성을 이용한 방식이다. [3]에서 제안된 HiCut(Hierarchical Intelligent Cuttings)은 classifier 의 특성을 분석하여 여러 필드들을 분할하여 HiCut tree 를 만들고, leaf 에는 작은 개수의 룰을 저장하여 순차적으로 검색하는 방법이다. 각 노드는 기하학적 검색 범위의 일부분을 나타내며 root 노드는 전체 필드의 검색 범위를 나타낸다. root 노드는 작은 검색범위로 분할되며 더 이상 나누어지지 않을 때 (leaf 노드에 일정 개수의 룰이 저장될 때)까지 분할한다. HiCut 은 빠른 검색 시간과 쉬운 update 라는 장점이 있는 반면, classifier 의 특성에 따라 검색 시간의 의존도가 크다는 단점이 있다.

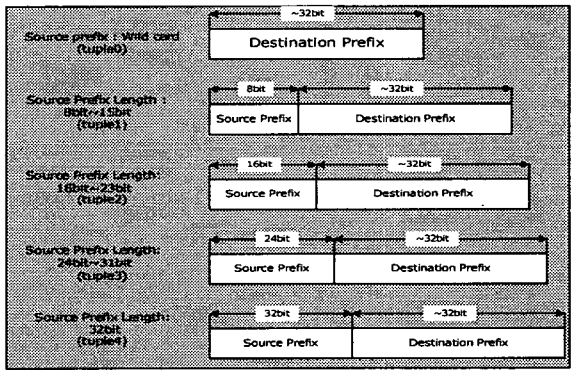
III. Proposed Architecture

. 본 논문에서는 IP 주소 검색을 위해 사용되었던 EnBiT 구조[4]를 packet classification 을 위한 구조로 확장하여 적용한 EnBiT-PC(EnBiT for Packet Classification)를 제안한다. EnBiT 은 서로 다른 길이의 프리픽스 간 검색이 가능한 구조이다.

1. Source, destination prefix pair

EnBiT 구조[4]에 2 dimension 검색을 적용하려면 source, destination 프리픽스를 함께 묶어 하나의 프리픽스처럼 취급하는데, 이것을 '프리픽스 pair'라고 한다. 그림 1은 본 논문에서 사용한 프리픽스 pair 의 종류를 나타낸 것으로 총 5 개로 구성되며 각각을 하나의 tuple 이라 명칭한다.

그림 1. 프리픽스 pair



2. EnBiT Memory

EnBiT memory 에는 EnBiT 의 구조에 따라 그림 1 에서 보인 프리픽스 pair 를 저장한다. 프리픽스 pair 가 인클로져인 경우 서브 트리를 가리키는 포인터와 룰 메모리를 연결하는 포인터를 가지고 있게된다.

3. Rule Memory

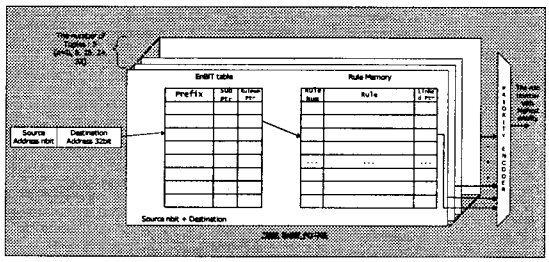
Rule memory 에는 EnBiT memory 에서 저장한 두 가지 필드(source, destination 프리픽스)를 제외한 나머지 필드인 port number 와 protocol 을 저장한다. 프리픽스 pair 가 같은 같은 룰들은 linked list 로 연결되어 룰은 순위가 높은 순서대로 저장되어 있으므로 상위 레벨에서 매치하는 룰을 찾은 경우 linked list 를 따라 하위 레벨로 내려갈 필요가 없게된다.

4. Build Procedure

제안하는 구조는 classifier 를 EnBiT memory 와 Rule memory 에 저장하는 과정을 거친다. rule 의 source 프리픽스 길이를 보고 어느 tuple 에 속하는지 판단한후, tuple type 에 맞춰 source 프리픽스 뒷부분을 자른다. 분리된 source 프리픽스와 destination 프리픽스를 붙여 프리픽스 pair 를 만든다. 완성된 프리픽스 pair 로 EnBiT 을 구성하고, 나머지 source 프리픽스와 나머지 필드들을 rule memory 에 저장한다. Source 프리픽스를 일정 길이로 잘랐기 때문에 같은 프리픽스 pair 를 가지는 rule 이 있을 수 있는데, 이는 linked list 로 연결한다.

그림 2 는 EnBiT-PC 의 구조를 나타낸 그림으로, 위에서 설명한 EnBiT memory 와 Rule memory 가 있고, 5 개 tuple 에서 병렬로 검색을 수행하는 것을 보여준다.

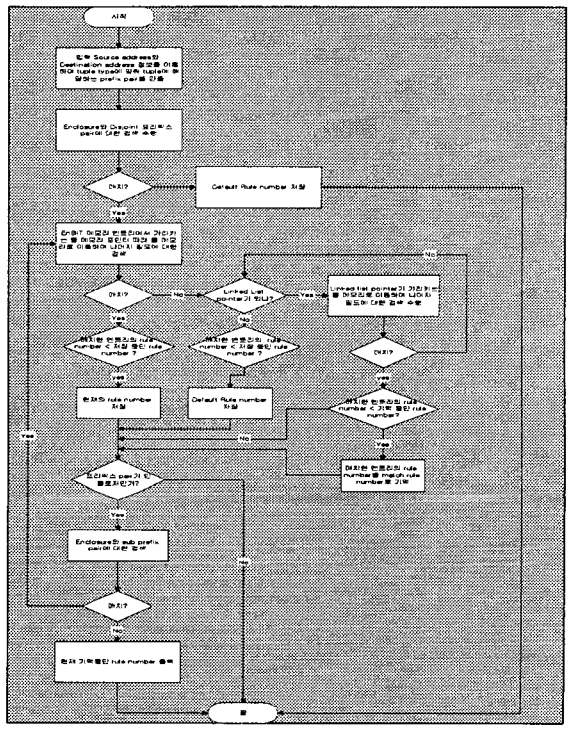
그림 2. EnBiT-PC 의 구조



4. Search Procedure

제안하는 구조의 검색은 다음과 같다. 먼저 패킷이 입력되면 source prefix 와 destination prefix 를 tuple 에 맞춰 5 종류의 프리픽스 pair 를 만든다. 입력된 패킷의 프리픽스 길이를 알 수 없으므로 5 개 tuple 의 프리픽스 pair 를 모두 만들어야 한다. 프리픽스 pair 는 각 tuple 의 EnBiT 메모리에서 먼저 검색된다. EnBiT 메모리에서 일치하는 엔트리가 발견된 경우 엔트리의 룰 메모리 포인터를 따라 룰메모리로 이동하여 나머지 필드에 대한 검색을 수행한다. 그림 3 는 본논문에서 제안하는 구조의 검색 과정을 나타내는 flow chart 이다.

그림 3. EnBiT-PC 의 검색 flow chart



IV. Simulation and Performance Evaluation

제안하는 구조를 C 언어를 사용, 실제 라우터에서 사용되는 classifier 로 simulation 을 수행하였다.

표 1 은 본 논문에서 제안한 구조에서의 메모리 검색 횟수를 나타낸 표이다. 메모리 횟수는 EnBiT memory 와 Rule memory 의 접근 횟수를 모두 포함한다. 4 개의 Rule(acl00, acl67, acl01, acl61)에 대해 평균, 최대, 최소 접근 횟수를 나타내었다.

표 1. rule 에 따른 메모리 접근 횟수

Rule	No. of Rules	average	Max.	Min.
acl00	2250	13.8	21	6
acl67	2375	30.2	58	8
acl01	2783	17.6	47	7
acl61	4710	40.3	43	13

표 2 은 본 논문에서 제안한 구조의 각 rule 별 메모리 크기를 나타낸 표이다. 제안된 구조는 N 을 rule 의 개수라 할 때, O(N)의 memory 를 요구한다. 각 Tuple 5 개에 해당하는 EnBiT table 의 크기와 Rule memory 의 크기를 합산하면 각 rule 의 메모리 크기가 나온다.

표 2. 제안하는 구조의 메모리 크기 (단위: KByte)

	acl 00		acl 67		acl 01		acl 61	
	EnBiT	Rule	EnBiT	Rule	EnBiT	Rule	EnBiT	Rule
sub total	15.3	331.1	16.4	36.8	7.2	42.5	33.8	79.6
total	48.4		53.3		49.7		113.4	

표 3 은 그동안 packet classification 을 위해 제안되었던 scheme 들과 본 논문에서 제안한 구조의 memory 접근 횟수와 memory 크기를 비교한 표이다. [5]

표 3. worst case memory 접근 횟수와 total memory 크기 비교 (HiCut: spfac =1.4)

No. of rules = 2790	RFC	HiCut -4	HiCut -1	BV	ABV	EGT	our scheme
Memory Access	12	82	172	846	190	154	47
Total memory (KByte)	2989	471	102	1106	1140	301	49

V. 결론

Packet classification 은 차세대 라우터에서 해야할 매우 중요한 기능으로 들어온 패킷을 class 별로 분류하여 각기 다른 서비스를 제공해 주는 것을 의미한다. 다양한 응용 프로그램의 전개와 QoS 의 요구로 인해 packet classification 의 중요성이 강조됨에 따라 그동안 여러가지 방법들이 연구되어 왔다. 본 논문에서는 효율적인 메모리 사용과 동시에 빠른 검색시간을 갖는 EnBiT-PC 구조를 제안하고 실제 classifier 로 simulation 한 결과와 다른 구조와의 비교를 하였다. 기존 방식들은 다양한 연산을 처리하지 못하고, 제한된 필드에 대한 검색만을 지원하였다. 또, 여러 필드를 동시에 검사하는 경우 빠른 검색 시간과 적절한 메모리 크기를 모두 만족하지 못하였다. EnBiT-PC 는 이러한 문제점을 해결하여, 2783 개의 룰에서 약 50KB 의 메모리를 사용하여 최대 47 번/평균 17.6 번의 메모리 검색이라는 효율적인 결과를 보였다.

References

[1] P.Gupta and N.McKeown, " Algorithms for Packet Classification" , IEEE Network, vol.15, pp24-32, 2001

[2] H.Jonathan Chao, " Next Generation Routers" , Proceedings of IEEE, vol. 90 pp1518-1558, 2002

[3] P.Gupta and N.NcKeown, " Packet Classification Using Hierarchical Intelligent Cuttings" , IEEE Micro, vol.20 no.1, pp34-41, 2000

[4] Bomi Lee and Hyesook Lim, "A New Pipelined Binary Search Architecture for IP Address Lookup", 한국통신학회논문지, 29 권, 1B, pp18-28, 2003

[5] Florin Baboescu, Sumeet Singh, George Varghese, "Packet Classification for Core Routers: Is there alternative to CAMs?", IEEE INFOCOM, vol.1 pp53-63, 2003