효율적인 패킷 필터링 시스템을 위한 CRG 알고리즘과 nTCAM

정회원 김 용 권*, 기 장 근*°, 이 순 석**, 종신회원 김 영 선**

CRG Algorithm and nTCAM for the Efficient Packet Filtering System

Yong-Kwon Kim*, Jang-Geun Ki*°, Soon-Seok Lee** *Regular Members*
Young-Sun Kim** *Lifelong Members*

요 약

본 논문에서는 TCAM을 이용해 패킷 필터링 시스템을 구현하는 경우 범위 규칙과 부정 규칙을 검색하는데 있어 기존의 방법보다 효율적으로 검색할 수 있는 방안을 제시하였다. 범위 규칙의 경우 그레이코드를 이용한 CRG(Converting Range rules using Gray code) 알고리즘을 제안하였으며, 부정 규칙을 효율적으로 검색하기 위한 방안으로는 nTCAM(TCAM with negation) 구조를 제안하였다. 또한 시뮬레이션을 통해 CRG 알고리즘과 nTCAM의 기능을 검증하였다. 성능 평가를 위해 제안 방안을 SNORT 규칙에 적용시킨 결과 IPv4와 IPv6 환경에서 기존의 방법과 비교할 때 각각 93%와 98%의 TCAM 엔트리를 절감하였다.

Key Words : Packet Filtering, TCAM, Range matching, Negation matching

Abstract

The general packet filtering system using TCAM has some limitations such as range and negation rules filtering, so this paper proposes efficient searching schemes than existing methods. CRG(Converting Range rules using Gray code) algorithm, in the case of range rules, that takes advantage of the gray code and TCAM characteristics to save a number of TCAM entries is proposed, and a nTCAM(TCAM with negation) architecture for negation rules is proposed, implemented using a FPGA design tool, and verified through the wave simulation. According to the simulation with the SNORT rules, the CRG algorithm and nTCAM save TCAM entries about 93% in IPv4 and 98% in IPv6 than the existing method.

Ⅰ. 서 론

패킷 필터링 시스템은 네트워크 장비에서 입력된 패킷을 검사하여 다음 수행해야할 일을 결정하기 위한 네트워크 기본 시스템으로써 다양한 네트워크 장비에 사용되고 있으며 장비의 성능을 좌우하는 중요한 요소이다[1]. 그러므로 증가하는 네트워크 속도를 위해서는 패킷 필터링 시스템의 성능이 뒷받침되어야 한다. 네트워크 속도에 따른 패킷 필터링 시스템의 성능은 IP 패킷의 최소 길이를 40bytes라고 할 때 네트워크 속도가 1Gbps이면 3.13Mpps(packet per second), 10Gbps일 때 31.3Mpps를 처리할 수 있어야한다. 이를 위해 가장 많이 사용되는 방법은 하드웨어 기반의 TCAM(Ternary Content Addressable Memory)을 이용하는 것이다. 그러므로 본 논문에서는 TCAM을 이용하여 패킷 필터링 시스템을 구현

* 공주대학교 정보통신공학부 (kjg@kongju.ac.kr) **한국전자통신연구원 (kig@mail.kongju.ac.kr) (° : 교신저자)
논문번호 : KICS2006-05-204, 접수일자 : 2006년 5월 10일, 최종논문접수일자 : 2006년 8월 3일

표 1. 응용 시스템별 사용 필드

Packet field	Router	Account	Traffic engineering	IDS	QoS
Src. addr.		○	○	○	○
Dst. addr.	○	○	○	○	○
Src. port		○		○	○
Dst. port		○		○	○
Protocol		○		○	○

하는 경우 문제점으로 제기 되는 범위 규칙과 부정 규칙에 대한 효율적인 검색 방안을 제시하고 검증하였다. 본 논문의 구성은 2장에서 기존 패킷 분류 알고리즘과 TCAM을 비교하였으며, 3장에서는 TCAM의 문제점을 고찰하고, 4장에서 TCAM의 문제점을 해결할 수 있는 CRG(Converting Range rules using Gray code)알고리즘과 nTCAM(TCAM with negation) 구조를 제안하였다. 5장에서는 제안 알고리즘과 nTCAM을 설계하고 시뮬레이션을 통해 검증한 내용을 기술하였으며, 6장에서 결론을 맺고 있다.

Ⅱ. 패킷 분류 알고리즘

2.1 패킷 매칭

패킷 매칭은 규정된 규칙과 네트워크로부터 입력된 패킷의 헤더 필드를 검색하여 규칙과의 부합여부를 판단하는 것이다. 패킷 매칭에 사용되는 패킷 헤더 필드는 발신지 주소와 포트번호, 목적지 주소와 포트번호, 프로토콜 등 일반적으로 5개의 헤더 필드이다. 표 1은 패킷 필터링 시스템이 사용되는 응용 시스템에 따라 매칭에 사용되는 패킷 헤더 필드를 나타낸 것이다[2]. 표에서 알 수 있듯이 라우터 시스템의 경우 목적지 주소만을 검색하지만 침입탐지 시스템(IDS)이나 과금시스템 등은 5개의 헤더 필드를 모두 사용한다.

패킷 필터링 규칙에 사용되는 필드는 특성에 따라 4가지로 분류할 수 있다. 먼저 1) 규칙의 필드와 입력 패킷의 해당 필드가 일치하는 경우 매칭되는 일반 필드와 2) 규칙의 필드와 입력 패킷의 필드가 서로 다른 경우 매칭되는 부정 필드, 3) 규칙 필드가 십진수의 범위로 규정되는 범위(negation) 필드, 4) 패킷의 해당 필드 값이 어떤 값이든 상관없는 무정(don't care)의 필드가 있다.

패킷 매칭 방법에는 규칙의 모든 비트를 검색하는 exact 매칭 방법과 규칙이 십진수의 범위로 규정되었을 때 사용하는 range 매칭 방법, 규칙이 prefix 형태로 주어진 경우 검색하는 prefix 매칭 방법이 있으며[3], 패킷 필터링 시스템은 3가지 매칭 방법을 모두 수행할 수 있어야 한다.

2.2 패킷 분류 알고리즘

패킷 분류를 위한 알고리즘은 크게 Trie 구조 방식과 기하학 구조 방식, 경험기반 방식, 하드웨어 기반 방식 등이 있다. Trie구조 방식은 Hierarchical tries와 set-pruning tries[4] 등의 알고리즘이 있으며, 기하학구조 방식에는 Grid-Of-Tries[5], EGT-PC[6], Cross-Producting[7], P^2C[8] 등의 알고리즘이 있고, 경험기반 방식에는 RFC[9], HiCut[10], HyperCuts[11], Tuple Space Search[12] 방식이 있다. 하드웨어 기반의 방식은 TCAM(Ternary Contents Addressable Memory)을 이용한 방법, WEITCAM[13], Bitmap-intersection 방법[14] 등이 있다. 위의 알고리즘 중 대표적인 알고리즘에 대해 간단히 살펴보면 다음과 같다. 설명에서 사용된 기호 N은 규칙 수를 의미하며, d는 매칭에 사용되는 검색 필드 수이고, W는 검색하는 필드의 비트 수를 의미한다.

- Hierarchical tries : d(>1) 차원 계층 트리를 구성하고 패킷이 입력되면 순환적인 이동(recursive traversals)을 통해 검색하는 알고리즘으로서 데이터 구조는 $O(NdW)$ 메모리 공간을 필요로 하며, 검색시간은 순환적 검색으로 인해 $O(W^d)$ 에 비례한다.

- set-pruning tries : set-pruning 알고리즘은 계층적 트리구조에서 반복적으로 검색하는 문제점을 보완한 알고리즘으로서 규칙을 미리 복사한 트리구조를 만들어 검색한다. 검색시간은 최악의 경우 $O(dW)$로서 계층트리구조의 $O(NdW)$ 보다 빠르지만 규칙을 복사해 사용하기 때문에 메모리 요구량이 증가한다.

- Grid-Of-Tries : 두개의 필드만을 고려한 알고리즘으로서 사용되는 데이터 구조는 계층 트리에서와 같이 각 규칙들이 오직 하나의 트리 노드에만 저장되도록하면서 일부 트리 노드에서 스위치 포인터(switch pointer)를 미리 계산해 저장해 놓음으로서 $O(dW)$ 검색시간을 가진다. 메모리 요구양은 $O(NdW)$이다.

- Cross-producting : 다차원 분류에 적용될 수 있으며, 각 차원에 대한 별도의 범위 검색 결과들을 조합하여 규칙이 들어있는 Cross-producting 테이블을 만든다. 패킷이 입력되면 각 필드의 범위를 찾은 후 범위에 해당하는 테이블을 검색해 매칭되는 규

칙을 찾는 알고리즘이다. 검색 시간은 $O(dW)$가 되고 메모리 요구량은 $O(N^d)$이 된다.

- HiCut : 분류기를 미리 처리(preprocessing)하여 결정 트리(decision tree) 데이터 구조를 구성한 후 패킷이 도착할 때마다 결정 트리를 검색하여 종단 노드를 찾는다. 종단 노드는 적은 수의 규칙들을 저장하고 있는 노드로, 이 노드에 들어있는 규칙들은 선형탐색(linear search)을 통해 매칭여부를 판단한다. 검색 시간은 $O(dW)$이며, 필요한 메모리는 $O(N^d)$가 된다.
- Tuple Space Search : 규칙의 필드들이 prefix 형태로 규정되었을 때 적용가능하며, 먼저 각 규칙을 prefix의 비트 길이로 구성된 튜플로 표현하고, 같은 튜플로 표현된 규칙들을 모아 하나의 해쉬 테이블로 만드는데 이때 해쉬 테이블의 키(key)는 각 차원(d)의 prefix를 차례로 연결하여 만든다. 패킷이 입력되면 독립적인 튜플마다 각각 구성되어 있는 해쉬 테이블들을 차례로 검색하여 매칭되는 규칙을 찾게 된다. 검색을 위한 튜플 수는 최악의 경우 $O(W^{d-1})$이 되고, 저장 공간은 $O(NdW)$이 된다.
- Bitmap-intersection : 각 차원을 범위로 나누고 나누어진 범위 마다 규칙 수와 같은 길이의 비트맵을 만든다. 패킷이 입력되면 패킷의 각 필드에 해당하는 범위를 검색하고 검색된 범위에 해당하는 비트맵을 AND 연산하여 규칙을 찾는 알고리즘으로서 검색 시간은 $O(dW\text{-}(N/mem_width))$이며, 메모리 요구량은 $O(dN^2)$이 된다.
- TCAM : 0, 1, X(don't care) 등 3가지 값을 검색할 수 있는 CAM으로서 하드웨어 구조상 prefix 형태의 규칙을 검색할 수 있다. TCAM의 검색 시간은 $O(1)$이며, 메모리 요구량은 $O(NdW)$가 사용된다.

위의 대표적인 8개 알고리즘 중 2개 필드만을 고려한 Grid-of-Tries 알고리즘을 제외하고 나머지 알고리즘을 검색 시간과 검색에 필요한 메모리 요구량으로 비교하면 그림 1과 같다. 그림은 IPv4 환경을 고려하여 5개의 필드를 모두 검사하고 검색 비트 수는 104bits, 규칙 수는 500개일 경우 메모리 요구량과 검색 시간을 이용하여 알고리즘을 비교한 것이다. 메모리 요구량에서 보면 Hierarchical tires와 Tuple space, TCAM이 동일하며, 검색 시간으로 보면 Bitmap-Intersection과 set-pruning, Cross-Producting, HiCut

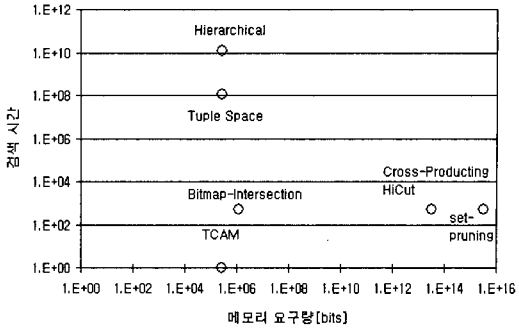

그림 1. 검색 시간과 메모리 요구량을 이용한 알고리즘 비교

알고리즘 등이 유사한 검색 시간을 보이고 있다. 메모리 요구량과 검색 시간을 모두 고려하는 경우 TCAM을 이용한 방법이 가장 우수하며, 하드웨어 기반의 Bitmap-Intersection 알고리즘의 경우 메모리 요구량은 TCAM과 유사하면서 검색 시간에서 TCAM을 제외한 알고리즘 중 가장 성능이 우수하다. 또한 알고리즘적인 방법은 IPv4 환경에 최적화 되어 있고 검색비트 수에 따라 검색 시간이 증가하게 된다. 그러므로 검색 비트 수가 증가하는 IPv6 환경에서는 성능이 급격히 떨어지는 문제점을 가지고 있다. 결과적으로 고속 트래픽 환경과 IPv6 환경에서 네트워크 속도를 유지시키기 위해서는 TCAM을 이용한 하드웨어 기반의 패킷 필터링 시스템이 가장 효율적인 것으로 판단된다.

TCAM은 엔트리의 집합으로 이루어져 있으며, 엔트리는 규칙을 저장하기 위한 값(value) 필드와 무정의 조건 검색을 위한 마스크(mask) 필드로 이루어져 있다. 마스크 필드의 각 비트는 0과 1값을 가지며, 0으로 설정되면 입력된 패킷의 해당 비트와 value 필드의 해당 비트를 검색하며, 1로 설정되면 무정의 조건을 의미하는 것으로 검색을 수행하지 않는다. 이러한 TCAM의 장·단점을 간략히 기술하면 다음과 같다.

- 다양한 응용 시스템에 사용 : TCAM은 구조가 간단하고 유연성과 확장성을 가지고 있어 다양한 응용 시스템에 사용 가능.
- 우수한 검색 성능 : 병렬처리를 통해 한 사이클에 매칭 수행.
- 하드웨어 구조상 prefix와 exact 매칭에 사용 : 범위 규칙과 부정 규칙에 비효율적임.
- 소비 전력이 크고 고가임 : TCAM은 무정의 조건을 검색하기 위해 한 비트 당 사용되는 트랜지스터의 개수가 일반 메모리에 비해 많음.

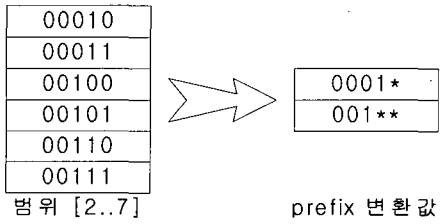

그림 2. 범위 필드의 TCAM 엔트리 확장 방법

Ⅲ. TCAM의 문제점 고찰

TCAM은 처리 속도 면에서 다른 알고리즘적인 방법에 비해 우수하기 때문에 수요가 증가하고 있으며, 이로 인해 제조 기술이 계속적으로 발전하고 있어 가격이나 소비전력문제가 점차 줄어들고 있는 추세이다[15][16]. TCAM의 다른 문제점은 하드웨어 구조상 prefix 형태의 규칙만을 검색할 수 있다는 것이다. 그러므로 규칙에 사용된 필드가 범위로 주어지거나 부정 필드로 주어지는 경우 하나의 TCAM 엔트리로 검색할 수 없기 때문에 복수 개의 TCAM 엔트리를 이용하여 검색하게 된다.

범위 필드는 주로 포트 번호 필드에 사용되며, TCAM을 이용하여 범위 필드을 검색하는 기존 방법은 이진 코드를 사용하여 범위를 대표할 수 있는 prefix 값으로 변환한 후 TCAM 엔트리에 저장해 매칭을 수행하는 것이다. 예를 들어 범위가 그림 2와 같이 [2..7]로 주어지는 경우 그림과 같이 2개의 prefix 값으로 변환할 수 있다.

그림 2와 같은 prefix 변환 방법을 이용하면 범위 필드의 길이가 W일 때 최대 2(W-1)개의 TCAM 엔트리가 필요하게 된다[17]. Prefix 형태로 변환되는 경우 가장 많은 개수로 변환되는 범위는 최상위 값과 최하위 값이 제외된 $[1..(2^W-2)]$가 되며, 이를 prefix 값으로 나타내면 {01*, 001*, 0001*, ..., $0^{W-1}1$, 10*, 110*,..., $1^{W-1}0$}이 된다. 예를 들어 W=6일 경우 가장 많은 prefix 값으로 변환되는 범위는 [1..62]가 되며, 변환된 prefix 값은 {01*, 001*, 0001*, 00001*, 000001, 10*, 110*, 1110*, 11110*, 111110}으로 10(=2x6-2)개의 prefix 값으로 변환된다. 그러므로 16bits인 포트번호 필드를 변환하는 경우 최대 30개의 TCAM 엔트리가 필요하며, 발신지와 목적지 포트를 모두 고려하는 경우 최대 900개의 엔트리가 필요하다.

TCAM의 다른 문제점은 규칙에 사용된 필드가 부정 필드인 경우이다. 부정 규칙은 규칙에 사용된 필드 중 부정 필드가 하나 이상 존재하는 규칙을 의미하며, 주로 주소 필드에 사용된다. 그림 3은 부정 필드를 검색하기 위한 기존 방법을 나타낸 것이다. 그림에서 부정 필드는 !203.253.37.0/16으로서 검색은 유효 주소 비트가 16bits이므로 상위 16비트만을 검색하게 된다. TCAM 엔트리 확장 방법은 규칙의 상위 16비트를 1의 보수화 한 후 그림과 같이 각 비트가 하나의 엔트리를 차지하도록 하는 것이다. 매칭 결과는 주소 필드의 상위 16비트를 확장된 TCAM에 매칭시켜 하나라도 매칭되면 규칙의 주소 값과 다른 헤더 필드 값이라고 판단하여 부정 규칙과 매칭된 것으로 판단한다.

규칙 : !203.253.37.0/16 $_{(10)}$
= !11001011.11111101.00100101.0000/16 $_{(2)}$

TCAM entris
0xxx xxxx.xxxx xxxx.xxxxxxxx.xxxxxxxx
x0xx xxxx.xxxx xxxx.xxxxxxxx.xxxxxxxx
xx1x xxxx.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxx1 xxxx.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxxx 0xxx.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxxx x1xx.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxxx xx0x.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxxx xxx0.xxxx xxxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.0xxx xxxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.x0xx xxxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.xx0x xxxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.xxx0 xxxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.xxxx 0xxx.xxxxxxxx.xxxxxxxx
xxxx xxxx.xxxx x0xx.xxxxxxxx.xxxxxxxx
xxxx xxxx.xxxx xx1x.xxxxxxxx.xxxxxxxx
xxxx xxxx.xxxx xxx0.xxxxxxxx.xxxxxxxx

그림 3. 부정 규칙의 TCAM 엔트리 확장 예

그림 3의 방법을 이용하게 되면 확장되는 엔트리 개수는 주소의 유효 비트 수와 동일하다. 그러므로 IPv4에서 하나의 주소 필드가 부정 필드인 경우 최대 32개의 TCAM 엔트리가 필요하며, 발신지 주소와 목적지 주소 모두 부정필드인 경우 하나의 규칙이 1,024개의 TCAM 엔트리를 사용하게 된다. IPv6 환경에서는 주소 비트가 4배로 증가하기 때문에 부정 필드 검색에 필요한 TCAM 엔트리의 개수가 4배로 증가하게 된다. 그러므로 부정 규칙에 대한 효율적인 검색 방안이 필요하다.

Ⅳ. CRG 알고리즘과 nTCAM

4.1 CRG 알고리즘

CRG(Converting Range rules using Gray code) 알고리즘[18]은 이진 코드가 아닌 그레이 코드와

표 2. 이진 코드와 그레이 코드

십진수	이진 코드	그레이 코드
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

TCAM의 매칭 특성을 이용하여 범위 규칙을 효율적으로 TCAM 엔트리 값으로 변환하는 알고리즘이다. 이진 코드를 그레이 코드로 변환하는 과정은 XOR 연산을 통해 쉽게 변환할 수 있으며, 변환된 그레이 코드는 대칭성을 가지고 있고 이웃한 값과 한 비트만 다르다. 표 2는 이진 코드와 그레이코드의 예를 나타낸 것으로 표에서 알 수 있듯이 그레이 코드는 0과 7의 중심값 4를 중심으로 상위 값을 접어서 내리면 최상위 비트만을 제외하고 일치한다. 다시 말해 이진 코드의 경우 0과 7을 하나의 값으로 묶을 수 없지만 그레이 코드는 '*00'으로 묶을 수 있게 된다. 또한 기존의 prefix 변환 방법은 무정의 조건이 항상 하위 비트에 있지만 그레이 코드를 사용하게 되면 무정의 조건이 임의의 위치에 있을 수 있으며 이는 TCAM의 특성상 매칭하는데 문제가 없다.

그레이 코드를 이용한 CRG 알고리즘은 먼저 주어진 범위 [a..b]를 서브 영영으로 분할하는 것이다. 주어진 범위는 최대 3개의 서브 영역(r_1, r_2, r_3)으로 분할되며, 이중 r_1, r_2 영역은 각각 중심값 m_1, m_2을 중심으로 대칭성을 가지도록 범위를 결정하고, r_3는 항상 r_1과 r_2 사이에 존재하기 때문에 쉽게 TCAM 엔트리 값으로 변환할 수 있다. 다음 단계는 분할된 서브 영역을 하나의 TCAM 엔트리 값으로 나타낼 수 있도록 다시 세부 영역으로 분할하는 것이며, 마지막으로 세부 영역을 대표하는 TCAM 엔트리 값으로 변환한다. 그림 4는 CRG 알고리즘에서 서브 영역을 분할하는 알고리즘을 순서도 형식으로 나타낸 것으로 그림에서 get_midval()은 이분법을 이용하여 중심값을 찾기 위한 알고리즘이다.

Prefix 변환 방법을 이용할 경우 가장 많은 prefix 값으로 변환되는 범위 $[1..(2^W-2)]$를 CRG 알고리즘에 적용시키면 대칭적인 특징으로 인해 $W-1$개의

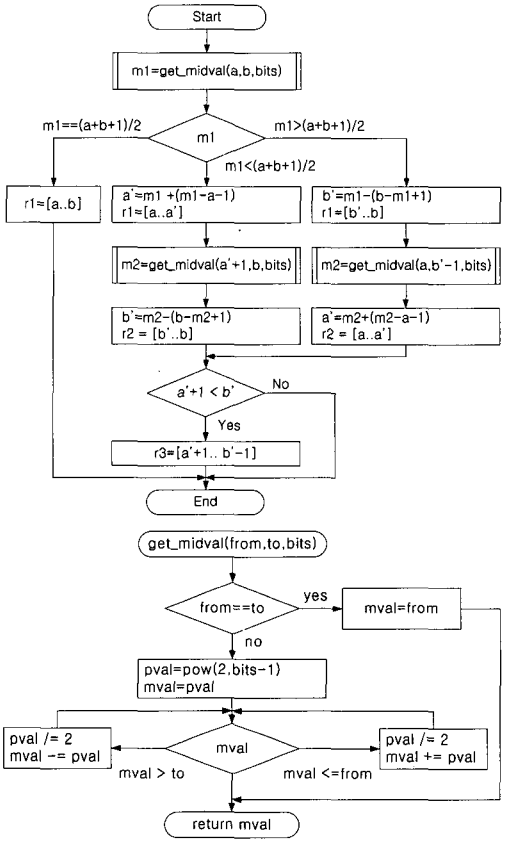

그림 4. 범위 [a..b]에 대한 영역분할 알고리즘

TCAM 엔트리 값으로 변환된다. 그레이 코드를 이용하는 경우 가장 많은 TCAM 엔트리 값으로 변환되는 범위는 대칭적인 특징이 최소화 되는 범위로서 $[1..(\frac{3}{4}2^W-2)]$ 또는 $[(2^{W-2}-1)..(2^W-2)]$가 되며, 주어진 범위는 알고리즘에 의해 항상 두개의 범위로 분할된다. 범위 $[1..(\frac{3}{4}2^W-2)]$을 두개의 범위로 분할하면 다음과 같다.

$$r_1 = [1..(2^{W-1}-2)]$$

$$r_2 = [(2^{W-2}+1)..(\frac{3}{4}2^W-2)]$$

분할된 r_1, r_2범위는 $2^{W-1}-2$개의 값으로 이루어져 있으며, 각 범위를 TCAM 엔트리 값으로 변환하는 경우 엔트리의 개수는 아래 식의 항수가 된다.

$$\{2^{W-2}, 2^{W-3}, ..., 2^{W-i}\}\ (i = W-1)$$

그러므로 분할된 영역을 TCAM 엔트리 값으로 변환하면 각각 $(W-2)$개로 변환되며, 최종 TCAM

엔트리 개수는 $2(W-2)$개가 된다. CRG 알고리즘을 이용하여 범위 $[1..(2^W-2^{W-2}-2)]$을 TCAM 엔트리 값으로 변환하면

$$\begin{Bmatrix} 0*1*, 0*01*, \ldots, 0*0^{W-3}1, *10*, \\ *111*, *1101*, \ldots, *110^{W-4}1 \end{Bmatrix}$$

가 된다. 예를 들어 W=6일 경우 가장 많은 TCAM 엔트리 값으로 변환되는 범위는 [1..46]이며, 이때 변환된 TCAM 엔트리 값은 {0*1*, 0*01*, 0*001*, 0*0001*, *10*, *111*, *1101, *11001*}이 되어 총 8(=2x6-4)개의 엔트리로 변환된다.

결과적으로 prefix 변환 방법은 범위가 W비트일 때 최대 $2W-2$개의 엔트리가 필요하며, 제안 알고리즘은 $2W-4$개의 TCAM 엔트리가 필요하게 된다. 두 방법이 모두 $O(W)$개로 범위를 변환하기 때문에 범위의 비트 수가 커지는 경우 제안 알고리즘에 의한 이득이 적어진다. 하지만 일반적으로 범위 필드가 16비트인 포트번호 필드에 사용되므로 제안 알고리즘을 하나의 포트 번호 필드에 적용시키면 prefix 변환 방법 보다 평균 7%의 TCAM 엔트리를 절감할 수 있으며, 발신지와 수신지 포드 번호 필드를 동시에 고려하는 경우 평균 14%의 엔트리를 절약할 수 있다.

4.2 nTCAM

TCAM을 이용하여 부정(negation) 주소 필드을 검색하는 경우 유효 주소 비트 수 만큼의 TCAM 엔트리가 필요하며, 부정 포트번호 필드의 경우 16개의 TCAM 엔트리가 필요하다. 이러한 부정 규칙을 검색하기 위해서는 기존 TCAM 구조로는 효율적이지 못하기 때문에 본 논문에서는 일반 규칙은 물론 부정 규칙도 효율적으로 검색할 수 있는 nTCAM 구조를 제안하였다.

먼저 부정 규칙을 검색하기 위해서는 부정 규칙을 TCAM 엔트리로 확장해야 한다. 본 논문에서 제안한 방법은 먼저 규칙내에 범위 필드가 있는 경우 CRG 알고리즘을 이용해 일반 필드로 변환한다. 다음은 필드를 부정 필드와 일반 필드로 분리한 후 하나의 부정 필드가 하나의 TCAM 엔트리 값을 가지도록하고, 마지막으로 부정 필드를 제외한 일반 필드들을 하나의 엔트리에 저장하여 부정 규칙을 TCAM 엔트리로 확장한다.

그림 5는 부정 규칙의 TCAM 엔트리 확장 예로서 부정 필드가 3개이고, 일반 필드가 있으므로 검색을 위해 필요한 TCAM 엔트리 수는 4개가 된다.

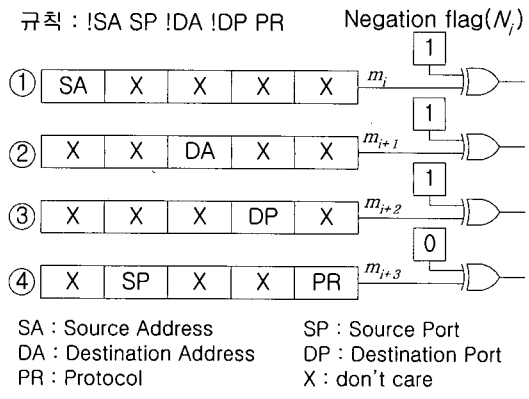

그림 5. negation 플래그를 이용한 매칭

첫 번째 엔트리에는 첫 번째 부정 필드인 SA를 TCAM 엔트리의 해당 비트에 입력하고 나머지 자리에는 무정의 조건(don't care)을 입력한다. 두 번째 엔트리에는 DA를 입력하고 나머지 비트는 무정의 조건을 입력한다. 이와 같은 방법으로 TCAM 엔트리 값을 설정한다.

그림에서 1, 2, 3번 엔트리는 부정 필드를 가지고 있고, 4번째 엔트리는 일반 규칙을 포함하고 있으므로 입력된 패킷을 검색한 후 1, 2, 3번 엔트리의 매칭 결과가 0이고, 4번 엔트리 매칭 결과가 1이면 입력된 패킷이 부정 규칙에 매칭된 것으로 판단한다. 하지만 1, 2, 3번 엔트리 중 하나라도 매칭 결과가 1이거나 4번 엔트리의 매칭 결과가 0이면 부정 규칙과 매칭되지 않았다는 의미가 된다. 그러므로 본 논문에서는 그림과 같이 negation 플래그와 XOR 게이트를 이용하여 부정 필드를 검색할 수 있도록 하였다. Negation 플래그는 부정 필드를 저장한 TCAM 엔트리의 경우 1이 되고, 일반 필드를 저장한 TCAM 엔트리는 0이 된다.

부정 규칙은 위의 그림에서 알 수 있듯이 규칙내의 부정 필드 수에 따라 매칭에 필요한 엔트리 개수가 변하게 된다. 그러므로 본 논문에서는 부정 규칙 검색에 필요한 엔트리 개수에 따라 유연성 있게 최종 매칭 결과를 계산할 수 있는 nTCAM 구조를 제안하였다. 그림 6은 제안한 nTCAM의 기본 RC(Result Calculation) 블록을 나타낸 것으로 입력은 TCAM 엔트리 중간 매칭 결과 값(m_i)과 negation 플래그(N_i), 확장된 TCAM 엔트리 중 첫 번째 엔트리임을 나타내는 start 플래그(S_i), $i+1$번째 엔트리의 결과 값(C_{i+1})이며, 출력 값으로는 최종 매칭 결과(M_i)와 i번째 계산 결과를 $i-1$번째 엔트리 계산 블록에 전달하기 위한 C_i가 있다. 이들 입출력간의

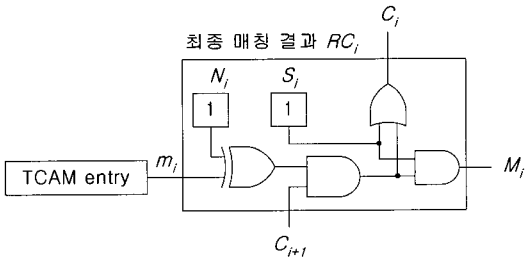

그림 6. nTCAM의 기본 RC 회로

관계는 다음 식으로 나타낼 수 있다.

$$M_i = (N_i \oplus m_i) \cdot C_{i+1} \cdot S_i$$

$$C_i = \begin{cases} (N_i \oplus m_i) \cdot C_{i+1} & \text{if } S_i = 0 \\ 1 & \text{if } S_i = 1 \end{cases}$$

Start 플래그 값은 현재 계산한 결과가 최종 결과 값인지 아닌지를 판단하기 위한 값으로 S_i=0이면 규칙의 시작 엔트리가 자신이 아니므로 i번째 엔트리에서 계산된 $(N_i \oplus m_i) \cdot C_{i+1}$ 값을 $i-1$번째 계산 블록으로 출력한다. S_i가 1이면 i번째 엔트리에서 계산한 결과 값을 최종 결과 값으로 판단하여 출력하고, C_i=1로 출력한다. C_i는 상단의 $i-1$번째 엔트리의 최종 결과 값인 M_{i-1} 값에 영향을 미치지만 AND연산에 의해 값을 계산하기 때문에 $C_i = 1$이 되면 M_{i-1}을 계산하는데 영향을 주지 않게 된다.

그림 7은 그림 5의 규칙을 nTCAM에 입력한 예를 나타낸 것이다. 그림의 왼쪽이 TCAM 엔트리이고, 오른쪽이 RC블록이며, 규칙 내에 부정 필드가 3개 있기 때문에 4개의 엔트리를 사용한다.

최종 매칭 결과를 계산하기 위한 식은 다음과 같다.

$$M_i = (m_i \oplus N_i) \cdot ((m_{i+1} \oplus N_{i+1}) \cdot ((m_{i+2} \oplus N_{i+2}) \cdot ((m_{i+3} \oplus N_{i+3}) \cdot C_{i+4} + S_{i+3}) + S_{i+2}) + S_{i+1}) \cdot S_i$$

식에서 S_i는 새로운 규칙이 시작되는 엔트리이므로 1이 되고, $S_{i+1}, S_{i+2}, S_{i+3} = 0$이 되며, C_{i+4}은 $i+4$번째 엔트리부터 새로운 규칙이 시작($S_{i+4} = 1$) 되기 때문에 1이 된다. 그러므로 최종 매칭 결과는 각 엔트리의 중간 매칭 결과(m_n)와 N_n의 XOR 연산 값을 AND 연산한 결과가 된다. 결국 최종 매칭 결과는 부정 규칙에 따라 확장되는 엔트리 개수만큼의 AND 연산으로 아래의 식과 같이 나타낼 수 있다.

$$M_i = (N_i \oplus m_i) \cdot (N_{i+1} \oplus m_{i+1}) \cdot \cdot \cdot (N_{i+n-1} \oplus m_{i+n-1})$$

n = 확장된 TCAM 엔트리 수

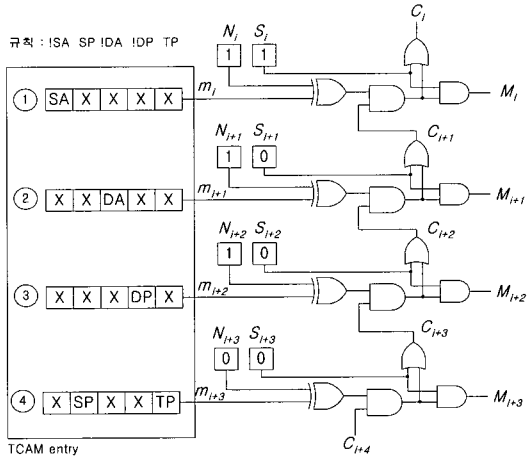

그림 7. nTCAM 구조

하드웨어 측에서 볼 때 TCAM은 무정의 조건을 검색하기 위해 한 비트당 16개의 트랜지스터가 사용된다[19][20]. 그러므로 IPv6 환경에서 하나의 패킷을 하나의 엔트리에서 처리한다고 가정하면 하나의 엔트리는 296bits이며, 사용되는 트렌지스터는 4,736개가 된다. 반면 nTCAM에서 최종 매칭 결과를 계산하기 위해 추가적으로 필요한 트랜지스터는 32개이므로[21][22] 기존 TCAM에 비해 0.7%의 추가적인 트랜지스터가 필요하다.

V. 시뮬레이션 결과

5.1 FPGA 설계 툴을 이용한 시뮬레이션

제안된 TCAM 구조의 기능 검증을 위해 Altera 사의 Quartus 툴을 이용하여 nTCAM을 설계하였다. 설계에 사용된 기본 CAM[23]은 Quartus 툴에서 제공하는 CAM으로서 256x96bits 용량을 가지고 있다. 즉 TCAM 엔트리는 256개 이며 각 엔트리는 96bits로 이루어져 있어 발신지/목적지 주소와 포트 번호을 검색할 수 있도록 설계하였다. 사용된 디바이스는 CAM을 내장하고 있는 APEX II 계열의 칩을 사용하였다.

시뮬레이션의 메인 주파수는 50MHz로서 50Mpps의 패킷을 처리할 수 있기 때문에 IP 패킷의 최소 길이를 40bytes라고 가정하면 16Gbps 네트워크에 사용될 수 있다.

nTCAM의 검색 기능 검증을 위한 시뮬레이션은 임의의 규칙을 만들어 입력한 후 임의의 패킷 헤더 필드를 TCAM에 입력하여 수행하였다. 그림 8은 기능 검증의 이해를 돕기 위해 시뮬레이션에 사용된

```
        SA          SP          DA          DP
R1 : Home_Net    any  -> Home_Net    2702
R2 : External_Net any  -> Home_Net    1812
R3 : Home_Net    993  -> External_Net any
R4 : External_Net !80  -> Home_Net    21544
R5 : External_Net any  -> External_Net 139
* External_Net = !Home_Net
* any = 무정의 필드
```

그림 8. 시뮬레이션을 위한 SNORT 규칙의 예

표 3. 시뮬레이션을 위한 TCAM 엔트리와 플래그 값

규칙		발신지		목적지		flag	TCAM addr.
		address	port	address	port		
R1	patten	0A000100	0000	0A000100	0A8E	01	0
	mask	000000FF	FFFF	000000FF	0000		0
R2	patten	0A000100	0000	0A000100	032C	11	1
	mask	000000FF	FFFF	FFFFFFFF	FFFF		1
	patten	0A000100	0000	0A000100	032C	00	2
	mask	FFFFFFFF	FFFF	000000FF	0000		2
R3	patten	0A000100	03E1	0A000100	0000	11	3
	mask	FFFFFFFF	FFFF	000000FF	FFFF		3
	patten	0A000100	03E1	0A000100	0000	00	4
	mask	000000FF	0000	FFFFFFFF	FFFF		4
R4	patten	0A000100	0050	0A000100	5428	11	5
	mask	000000FF	FFFF	FFFFFFFF	FFFF		5
	patten	0A000100	0050	0A000100	5428	10	6
	mask	FFFFFFFF	0000	FFFFFFFF	FFFF		6
	patten	0A000100	0050	0A000100	5428	00	7
	mask	FFFFFFFF	FFFF	000000FF	0000		7
R5	patten	0A000100	0000	0A000100	008B	11	8
	mask	000000FF	FFFF	FFFFFFFF	FFFF		8
	patten	0A000100	0000	0A000100	008B	10	9
	mask	FFFFFFFF	FFFF	000000FF	FFFF		9
	patten	0A000100	0000	0A000100	008B	00	10
	mask	FFFFFFFF	FFFF	FFFFFFFF	0000		10

일부 규칙을 나타낸 것이다. 규칙은 침입 탐지 프로그램인 SNORT[24] 규칙 중 일부이다.

규칙을 TCAM에 저장하기 위해 Home_net 주소를 10.0.1.0/24로 가정하였다. 그러므로 부정 필드인 External_Net의 값은 !10.0.1.0/24가 된다. 표 3은 그림 8 규칙에 대한 TCAM 엔트리 값과 무정의 조건 검색을 위한 마스크 값 그리고 플래그 값을 나타낸 것이다. 마스크 값은 0인 경우 해당 비트를 검색하고, 1인 경우(무정의 조건) 검색을 수행하지 않는다. 예를 들어 규칙 R4는 발신지 주소와 발신지 포트 번호가 부정 필드인 규칙으로서 3개의 엔트리를 이용하여 검색하게 된다. 그러므로 R4를 위한 첫 번째 엔트리는 발신지 주소만을 검색하도록 설정하며, negation플래그는 1이 되고, start 플래그도 1이 된다. 두 번째 엔트리는 발신지 포트 번호만을 검색하며, negation 플래그는 1이 되고, start 플래그는 규칙이 시작하는 첫 번째 엔트리가 아니므로 0이 된다. 마지막 엔트리는 일반 필드인 목적지 주소와 목적지 포트번호를 같이 검색하며, 플래그는 '00'으로 설정한다. 이와 같은 방법으로 다른 규칙도 TCAM 엔트리 값을 설정하면 된다.

표 4. 시뮬레이션에 사용된 입력 패턴 및 매칭 결과

pat-tern	발신지		목적지		최종매칭결과	rule	비고
	address	port	address	port			
P1	0A000100	00EE	0A000100	0A8E	1	R1	R1 규칙
P2	0B000B00	00EE	0A000100	0A8E	0	-	R1, SA가 다름
P3	0A000100	00EE	0A000100	E000	0	-	R1, DP가 다름
P4	0B000B00	00AA	0A000100	0714	1	R2	R2 규칙
P5	0B000B00	00AA	0A000100	E000	0	-	R2, DP가 다름
P6	0A000100	00AA	0A000100	0714	0	-	R2, SA가 다름
P7	0A000100	03E1	0B000B00	00AA	1	R3	R3 규칙
P8	0C000C00	03E1	0B000B00	00AA	0	-	R3, SA가 다름
P9	0A000100	E000	0B000B00	00AA	0	-	R3, SP가 다름
P10	0A000100	03E1	0A000100	00AA	0	-	R3, DA가 다름
P11	0B00010B	00AA	0A000100	5428	1	R4	R4 규칙
P12	0A000100	00AA	0A000100	5428	0	-	R4, SA가 다름
P13	0B000B00	0050	0A000100	5428	0	-	R4, SP가 다름
P14	0B000B00	00AA	0C000C00	5428	0	-	R4, DA가 다름
P15	0B000B00	00AA	0A000100	E000	0	-	R4, DP가 다름
P16	0B000B00	00EE	0C000C00	008B	1	R5	R5 규칙
P17	0A000100	00EE	0C000C00	008B	0	-	R5, SA가 다름
P18	0B000B00	00EE	0A000100	008B	0	-	R5, DA가 다름
P19	0B000B00	00EE	0C000C00	E000	0	-	R5, DP가 다름
P20	0A00E100	03E1	0A000100	0714	1	R2	R2 규칙

표 4는 이해를 돕기 위해 시뮬레이션에 사용된 입력 패킷의 패턴과 매칭 결과의 예를 표 형식으로 나타낸 것이며, 규칙에 매칭되는 패턴과 매칭 되지 않는 패턴으로 나누어 20개의 입력 패턴 예를 나타내었다.

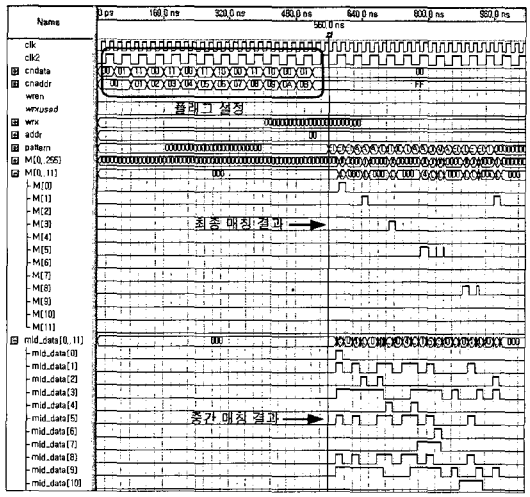

그림 9. 파형 시뮬레이션 결과

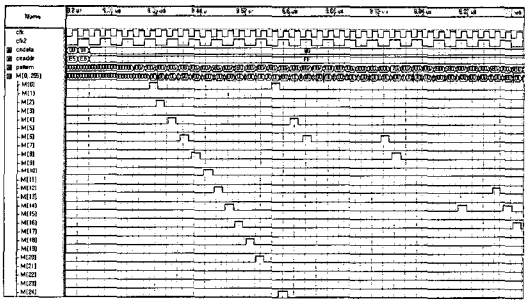

그림 10. nTCAM을 이용한 SNORT 규칙의 파형 시뮬레이션 결과

예를 들어 입력 패킷 패턴 P11은 발신지 주소와 발신지포트 번호가 R4 규칙의 부정 필드 값과 불일치하면서 목적지 주소와 포트 번호는 R4 규칙과 일치하여 최종 매칭 결과는 1이 된다. P12 패턴은 발신지 주소는 R1과 R3 규칙과 일치하지만 발신지 포트번호는 R1, R2, R4, R5 규칙과 일치한다. 그러므로 발신지 주소와 발신지 포트번호는 R1과 일치하지만 목적지 포트번호가 R1 규칙과 일치하지 않으므로 최종 매칭 결과는 0이 된다.

그림 9는 표 3과 표 4의 규칙과 입력 패턴을 이용하여 시뮬레이션을 수행한 결과 파형이다. 파형의 앞부분은 negation과 start 플래그를 설정하기 위한 부분이고, 뒷부분의 하단은 TCAM 엔트리의 중간 매칭 결과 값이며, 중간에 있는 파형이 최종 매칭결과을 나타낸 것이다.

그림 10은 SNORT 2.4 규칙을 nTCAM에 입력하여 시뮬레이션한 결과이다. TCAM의 제한으로 인해 115개의 규칙만을 TCAM 엔트리에 입력하여 시뮬레이션을 수행하였다. 그림에서 앞부분의 신호는

표 5. 범위 규칙에 대한 기존 방법과 제안 방법과의 비교

구분	부정 필드가 있는 경우		부정 필드가 없는 경우		최종 엔트리 수
	규칙수	변환 개수	규칙 수	변환 개수	
Prefix 방법	31	169	1	10	179
CRG 알고리즘	31	152	1	9	161

표 6. SNOR 규칙에 적용시킨 결과

	기존 규칙 수	확장 규칙 수	최종 규칙 수	최종 엔트리 수
기존 방법	RR : 32 NRR : 291	RR : 179 NRR : 291	NR : 444 GR : 26	IPv4 : 12,338 IPv6 : 40,538
CRG & nTCAM	RR : 32 NRR : 291	RR : 161 NRR : 291	NR : 427 GR : 25	IPv4 : 882 IPv6 : 882

* NR : Negation rule, NRR : No Range Rule
GR : General rule, RR : Range rule

플래그 설정을 위한 값이며, 뒷부분이 입력 패턴과의 매칭 결과이다.

앞에서 기술한 것과 같이 파형 시뮬레이션을 통해 nTCAM의 기능을 검증하였으며, 특히 실제 사용되고 있는 SNORT 규칙을 본 논문에서 제안한 부정 규칙의 TCAM 엔트리 확장 절차에 따라 TCAM 엔트리 값을 설정하고, 임의의 입력 패턴을 이용하여 시뮬레이션을 수행하였다. 수행결과 nTCAM이 일반 규칙은 물론 부정 규칙에 대해 정상적으로 검색하는 것을 확인하였다.

5.2 SNORT 규칙에 적용시킨 결과

표 5는 SNORT 규칙 중 포트번호가 범위 필드인 규칙만을 대상으로 시뮬레이션을 수행한 결과이다. SNORT 규칙 중 범위 규칙은 32개 이며, 이중 31개가 발신지 혹은 목적지 주소 중 하나가 부정 필드로 되어 있는 부정 규칙이고, 나머지 하나의 범위 규칙은 부정 필드가 없다. 32개의 범위 규칙을 CRG 알고리즘을 이용해 변환하면 161개의 TCAM 엔트리 값으로 나타낼 수 있으며, 기존의 prefix 방법을 이용하면 179개의 prefix 값으로 나타낼 수 있다. 결과적으로 범위 규칙의 경우 CRG 알고리즘을 이용하여 범위 규칙을 변환하면 기존의 방법보다 더 적은 수의 TCAM 엔트리로 패킷 매칭을 수행할 수 있다.

표 6은 SNORT 전체 규칙을 이용하여 기존 방법과 본 논문에서 제시한 방법을 비교한 것이다. SNORT의 3,771개 규칙 중 서로 다른 패턴의 규칙은 323개이며, 규칙 중 포트번호가 범위 필드인 규칙은 32개이고, 범위가 아닌 규칙은 291개이다. 범위 규칙에 대해 기존의 prefix 변환 방법을 이용하면 179개의 규칙으로 확장되며, CRG 알고리즘을 이용하면 161개의 규칙으로 확장된다. 그러므로 최종 규칙 수는 prefix 변환 방법을 적용시키면 470개가 되고, CRG 알고리즘을 적용시키면 452개가 된다. 부정 규칙 중 발신지와 목적지 주소가 모두 부정 필드인 규칙은 3개이며 나머지는 한쪽만 부정 필드인 규칙이다. 유효 주소 비트가 24bits인 IPv4환경을 가정하면 기존 방법과 nTCAM을 적용시키면 최종 엔트리 개수는 각각 12,338개와 882개의 TCAM 엔트리로 확장된다. 그러므로 제안 TCAM은 기존 방법에서 필요한 엔트리의 7%인 882개의 TCAM 엔트리만 있으면 SNORT 규칙을 검색할 수 있게 된다. IPv6환경(유효 주소 비트 : 64bits)에 적용시킬 경우 CRG 알고리즘과 nTCAM이 기존 방법에 비해 98%의 TCAM 엔트리를 절약 할 수 있다.

기존의 알고리즘적인 방법은 검색 비트 수가 증가하는 IPv6 환경에서는 메모리 뿐만 아니라 검색 시간이 비례하여 증가한다. 결국 검색 속도와 메모리 용량면에서 TCAM이 보다 효율적이다. nTCAM은 표 6의 결과에서 알 수 있듯이 IPv6 환경에서 검색 속도는 기존 TCAM과 동일하면서 TCAM 엔트리 수를 줄일 수 있는 효율적인 TCAM 구조이다.

Ⅵ. 결 론

본 논문에서는 TCAM을 이용하여 패킷 필터링 시스템을 구현하는 경우 기존의 방법보다 효율적으로 범위 규칙과 부정 규칙을 검색할 수 있는 방안을 제시하고 시뮬레이션을 통해 검증하였다. 범위 규칙의 경우 기존의 prefix 변환 방법이 아닌 그레이 코드를 이용한 CRG 알고리즘을 제안하였으며, 부정 규칙은 기존의 TCAM 구조로는 효율적으로 검색을 할 수 없기 때문에 nTCAM 구조를 제안하였다. nTCAM은 기존의 TCAM 기능은 그대로 유지하면서 부정 규칙을 효율적으로 검색하기 위한 RC블록을 포함하고 있다. 추가된 RC 블록은 적은 게이트를 이용하여 부정 규칙 검색에 필요한 엔트리 개수에 따라 유연성 있게 최종 매칭 결과를 계산할 수 있는 구조이다. 제안된 nTCAM은 FPGA 설계 툴을 이용하여 설계하고, 시뮬레이션을 통해 기능을 검증하였다. 또한 SNORT 규칙에 CRG 알고리즘과 nTCAM을 적용시킨 결과 IPv4 환경에서는 기존 방법에 비해 93%의 TCAM 엔트리를 절감하였으며, IPv6환경에서는 약 98%의 TCAM 엔트리를 절약하였다. 결론적으로 본 논문에서 제안한 CRG 알고리즘과 nTCAM 구조는 TCAM을 이용하여 패킷 필터링을 수행하는 경우 범위 규칙과 부정 규칙에 필요한 TCAM 엔트리를 절감함으로서 패킷 필터링 시스템의 비용을 낮출 수 있다.

참 고 문 헌

[1] Myung-Sup Kim, Young J. Won, and James Won-Ki Hong, "Application-Level Traffic Monitoring and an Analysis on IP Networks," ETRI Journal, Vol.27, No.1, pp.22-42, Feb. 2005.

[2] J. Quittek, T. Zseby, B. Claise, S. Zander, "Requirements for IP Flow Information Export," IETF IPFIX working group, RFC3917, Oct. 2004.

[3] Huan Liu, "Efficient Mapping of Range Classifier into Ternary-CAM,' Proc. 10th Hot Interconnects (HOTI), 2002.

[4] Paul Francis Tsuchiya, "A Search Algorithm for Table Entries with Non-contiguous Wildcarding," http : //citeseer.ist.psu.edu/tsuchiya91search.html, 1991.

[5] V. Srinivasan, George Varghese, Subhash Suri, Marcel Waldvogel, "Fast and Scalable Layer Four Switching," Proceedings of ACM SIGCOMM '98, pp.203-214, Sept., 1998.

[6] Florin Baboescu, Sumeet Singh, George Varghese, "Packet Classification for Core Routers : Is there an alternative to CAMs?," INFOCOM 2003.

[7] V. Srinivasan, G. Varghese, S. Suri, and M. Wald-vogel, "Fast and scalable layer four switching," In Proceedings of the ACM SIGCOMM '98, pp.191-202. ACM Press, 1998.

[8] Jan van Lunteren, Ton Engbersen, "Fast and scalable packet classification," IEEE Journal on Selected Areas in Communications, Vol.21, No.4, pp.560-571, May, 2003.

[9] Pankaj Gupta and Nick McKeown, "Packet classification on multiple fields," In SIGCOMM, pp.147-160, 1999.

[10] P. Gupta and N. McKeown, "Packet Classification Using Hierarchical Intelligent Cuttings," Proc. Hot Interconnects VII, Aug. 1999; also available in IEEE Micro, Vol.20, No.1, pp.34-41, Jan./Feg., 2000.
[11] Sumeet Singh, Florin Baboescu, George Varghese, and Jia Wang, "Packet Classification Using Multidimensional Cutting," Proc. ACM Sigcomm, Aug., 2003.
[12] V. Srinivasan, S. Suri and G. Varghese, "Packet Classification using Tuple Space Search," Proc. ACM Sigcomm, pp.135-146, Sept., 1999.
[13] H. Che, Y. Wang, and Z. Wang, "A Rule Grouping Technique for Weight-Based TCAM Coprocessors," Proc. 11th Hot Interconnects (HOTI), 2003.
[14] T. V. Lakshman and Dimitrios Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," In SIGCOMM, pp.203-214, 1998.
[15] A. Natarajan, D. Jasinski, W. Burleson, and R. Tessier, "A Hybrid Adiabatic Content Addressable Memory for Ultra-Low Power Applications," in the Proceedings of the IEEE/ACM Great Lakes Symposium on VLSI, Apr., 2003.
[16] I.Arsovski and A. Sheikholeslami, "A Mismatch-Dependent Power Allocation Technique for Match-Line Sensing in Content-Addressable Memories IEEE Journal of Solid-State Circuits," Vol.38, No.11, pp.1958-1966, Nov., 2003.
[17] Karthik Lakshminarayanan, Anand Rangarajan, Srinivasan Venkatachary, "Algorithms for Advanced Packet Classification with Ternary CAMs," SIGCOMM2005, Aug., 2005.
[18] Yong Kwon Kim, Jang Geun Ki, Kyou Ho Lee, "A Novel Scheme for Range Rule Matching in Ternary CAM," ITC-CSCC 2005, Vol.4, pp. 1427-1428, July, 2005.
[19] David E. Taylor, Edward W. Spitznagel, "On using content addressable memory for packet classification," Technical Report WUCSE-2005-9, Wachington Univ., March 2005.
[20] Mohammad J. Akhbarizadeh, Mehrdad Nourani, Cyrus D. Cantrell, "Segregating the Encompassing Prefixes to Enhance the Performance of Packet Forwarding Engines," IEEE Communications Society Globecom pp.1612-1616, 2004.
[21] Reto Zimmermann and Wolfgang Fichtner, "Low-Power Logic Styles : CMOS Versus Pass-Transistor Logic," IEEE Journal of Solid-State Circuits, Vol.32, No.7, pp.1079-1090, July, 1997.
[22] All About Circuits, "CMOS gate Circuitry," at http://www.allaboutcircuits.com/vol_4/chpt_3/8.html/cmos_gate_circuitry.
[23] Altera, "Implementing High-Speed Search Applications with Altera CAM," Application note 119 at http://www.altera.com, July, 2001.
[24] SNORT, at http://www.snort.org.

김 용 권 (Yong-Kwon Kim) 정회원

1999년 공주대학교 전자공학과 공학사
2001년 공주대학교 전기전자정보공학과 공학석사
2006년 공주대학교 전기전자정보공학과 공학박사
<관심분야> 차세대 인터넷 기술, 인터넷 망 관리

기 장 근 (Jang-Geun Ki) 정회원

1986년 고려대학교 전자공학과 공학사
1988년 고려대학교 대학원 전자공학과 공학석사
1992년 고려대학교 대학원 전자공학과 공학박사
1992년~현재 공주대학교 정보통신공학부 교수
2002년~2003년 미국 Univ. of Arizona 방문교수
<관심분야> 컴퓨터 네트워크, 차세대 네트워크, 멀티미디어 통신

이 순 석 (Soon-Seok Lee) 정회원

1988년 성균관대학교 산업공학과 공학사
1990년 성균관대학교 산업공학과 공학석사
1993년 8월 성균관대학교 산업공학과 공학박사
1993년 7월~2002년 7월 현재 한국전자통신연구원 선임연구원
2002년 7월~현재 한국전자통신연구원 책임연구원
2005년 2월~현재 한국전자통신연구원 광대역통합망연구단 BcN설계팀 팀장
<관심분야> 차세대 네트워크 및 서비스 아키텍쳐, 네트워크 진화전략, 네트워크 구조 및 최적설계, 트래픽 엔지니어링, 네트워크 및 통신시스템 성능평가

김 영 선 (Young-Sun Kim) 종신회원

1980년 고려대학교 전자공학과 공학사
1982년 고려대학교 전자공학과 공학석사
1991년 고려대학교 전자공학과 공학박사
1982년~현재 한국전자통신연구원 광대역통합망연구단 네트워크연구그룹장.
1989년~현재 한국통신학회 상임이사
<관심분야> 광인터넷 네트워크, BcN네트워크 엔지니어링, SLA, 품질보장형라우터 등