



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년09월20일
(11) 등록번호 10-2023475
(24) 등록일자 2019년09월16일

(51) 국제특허분류(Int. Cl.)
H04L 12/851 (2013.01) H04L 12/741 (2013.01)
H04L 12/753 (2013.01) H04L 29/06 (2006.01)
(52) CPC특허분류
H04L 47/2441 (2013.01)
H04L 45/48 (2013.01)
(21) 출원번호 10-2018-0036869
(22) 출원일자 2018년03월29일
심사청구일자 2018년03월29일
(56) 선행기술조사문헌
KR101331018 B1
WO2011108168 A1
KR101505845 B1

(73) 특허권자
계명대학교 산학협력단
대구광역시 달서구 달구벌대로 1095, 계명대학교
산학협력관 201호(신당동)
(72) 발명자
박우길
대구광역시 달서구 월서로 51 101동 1806호 (진천
동, 월배역 포스코 더샵 아파트)
(74) 대리인
김건우

전체 청구항 수 : 총 20 항

심사관 : 김대성

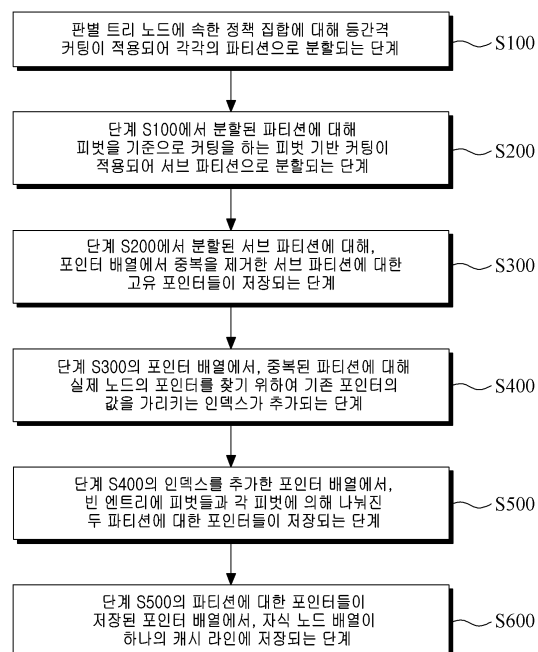
(54) 발명의 명칭 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템

(57) 요약

본 발명은 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에 관한 것으로서, 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 방법에 있어서, (1) 관별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할되는 단계; (2) 상기 단계 (1)에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하

(뒷면에 계속)

대표도 - 도2



는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 단계; (3) 상기 단계 (2)에서 분할된 서브 파티션들에 대한 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장되는 단계; (4) 상기 단계 (3)의 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가되는 단계; (5) 상기 단계 (4)의 인덱스를 추가한 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장되는 단계; 및 (6) 상기 단계 (5)의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장되는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

또한, 본 발명은 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템에 관한 것으로서, 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 시스템에 있어서, 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할하는 등간격 커팅 모듈; 상기 등간격 커팅 모듈에 의해 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅을 적용하여 서브 파티션으로 분할하는 피벗 기반 커팅 모듈; 상기 피벗 기반 커팅 모듈에 의해 분할된 서브 파티션들에 대한, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들을 저장하는 제1포인터 배열 저장 모듈; 상기 제1포인터 배열 저장 모듈에 의해 저장된 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스를 추가시키는 인덱스 추가 모듈; 상기 인덱스 추가 모듈에 의해 인덱스가 추가된 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들을 저장하는 제2포인터 배열 저장 모듈; 및 상기 제2포인터 배열 저장 모듈에 의해 포인터들이 저장된 포인터 배열에서, 자식 노드 배열을 하나의 캐시 라인에 저장하는 캐시 라인 저장 모듈을 포함하는 것을 그 구성상의 특징으로 한다.

본 발명에서 제안하고 있는 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템에 따르면, 정책 검색을 위한 매 단계에서 하나의 방법으로 커팅을 수행하는 대신, 두 가지 방식으로 동시에 커팅을 수행함으로써, 다양한 정책 집합의 특성에도 불구하고 빠르게 커팅을 할 수 있다.

또한, 본 발명에 따르면, 캐시 구조에 적합하도록 정교하게 설계된 노드를 사용하여 메모리 액세스 수를 최소화함으로써, 테이블 크기와 패킷 분류 성능을 동시에 개선할 수 있다.

(52) CPC특허분류

H04L 45/54 (2013.01)

H04L 63/20 (2013.01)

공지예외적용 : 있음

명세서

청구범위

청구항 1

커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 방법에 있어서,

- (1) 등간격 커팅 모듈에서, 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할되는 단계;
- (2) 피벗 기반 커팅 모듈에서, 상기 단계 (1)에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 단계;
- (3) 제1포인터 배열 저장 모듈에서, 상기 단계 (2)에서 분할된 서브 파티션들에 대한 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장되는 단계;
- (4) 인덱스 추가 모듈에서, 상기 단계 (3)의 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가되는 단계;
- (5) 제2포인터 배열 저장 모듈에서, 상기 단계 (4)의 인덱스를 추가한 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장되는 단계; 및
- (6) 캐시 라인 저장 모듈에서, 상기 단계 (5)의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장되는 단계를 포함하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 2

제1항에 있어서, 상기 단계 (1)은,

상기 등간격 커팅 모듈에서, 한 개 이상의 필드에 대해서 등간격으로 커팅을 수행하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 3

제1항에 있어서, 상기 단계 (2)는,

상기 피벗 기반 커팅 모듈에서, 상기 등간격 커팅에서 가장 많이 파티셔닝이 수행된 차원에 대하여 상기 피벗 기반 커팅이 적용되는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 4

제1항에 있어서, 상기 단계 (2)는,

상기 피벗 기반 커팅 모듈에서, 상기 단계 (1)에서 분할된 가장 큰 파티션들에 대해 적용되며 전체 파티션들의 가장 큰 크기를 최소화하는 값으로 커팅되는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 5

제1항에 있어서, 상기 피벗 기반 커팅은,

특정 값을 기준으로 작거나 같은 정책들과 큰 정책들로 구성된 서브파티션들로 커팅하는 것을 포함하는 것을 특

징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 6

제1항에 있어서, 상기 단계 (5)는,

상기 제2포인터 배열 저장 모듈에서, 자식 노드에 대해서는 32bit 포인터 값을 가지고 추가적인 정보를 제공하기 위해 4bit의 인덱스 값과 28bit의 포인터 값 혹은 피벗으로 구성되는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 7

제1항에 있어서, 상기 단계 (5)는,

상기 제2포인터 배열 저장 모듈에서, 포인터 값과 피벗은 용도에 따라 다른 값으로 저장 되지만 공유되는 필드를 가지는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 8

제1항에 있어서, 상기 단계 (5)는,

상기 제2포인터 배열 저장 모듈에서, 자식 노드의 고유 포인터들은 배열의 첫 엔트리부터 연속적으로 배치된 후, 피벗 기반 커팅을 위한 피벗과 피벗 기반 커팅을 통해 얻은 추가 노드에 대한 포인터가 배치되는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 9

제1항에 있어서, 상기 단계 (6)은,

상기 캐시 라인 저장 모듈에서, 하나의 캐시 라인 안에서는 단 한 번의 메모리 액세스만 이루어지는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 10

제1항에 있어서, 상기 단계 (6)은,

상기 캐시 라인 저장 모듈에서, 메모리 액세스 수를 줄이기 위해서 하나의 캐시 라인 사이즈마다 새로운 헤더를 추가하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법.

청구항 11

커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 시스템에 있어서,

판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할하는 등간격 커팅 모듈;

상기 등간격 커팅 모듈에 의해 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅을 적용하여 서브 파티션으로 분할하는 피벗 기반 커팅 모듈;

상기 피벗 기반 커팅 모듈에 의해 분할된 서브 파티션들에 대한, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들을 저장하는 제1포인터 배열 저장 모듈;

상기 제1포인터 배열 저장 모듈에 의해 저장된 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를

찾기 위하여 기존 포인터의 값을 가리키는 인덱스를 추가시키는 인덱스 추가 모듈;

상기 인덱스 추가 모듈에 의해 인덱스가 추가된 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉘진 두 파티션에 대한 포인터들을 저장하는 제2포인터 배열 저장 모듈; 및

상기 제2포인터 배열 저장 모듈에 의해 포인터들이 저장된 포인터 배열에서, 자식 노드 배열을 하나의 캐시 라인에 저장하는 캐시 라인 저장 모듈을 포함하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 12

제11항에 있어서, 상기 등간격 커팅 모듈은,

한 개 이상의 필드에 대해서 등간격으로 커팅을 수행하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 13

제11항에 있어서, 상기 피벗 기반 커팅 모듈,

상기 등간격 커팅에서 가장 많이 파티셔닝이 수행된 차원에 대하여 상기 피벗 기반 커팅을 하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 14

제11항에 있어서, 상기 피벗 기반 커팅 모듈은,

상기 등간격 커팅 모듈에 의해 분할된 가장 큰 파티션들에 대해 적용되며 전체 파티션들의 가장 큰 크기를 최소화하는 값으로 커팅하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 15

제11항에 있어서, 상기 피벗 기반 커팅 모듈은,

특정 값을 기준으로 작거나 같은 정책들과 큰 정책들로 구성된 서브파티션들로 커팅하는 것을 포함하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 16

제11항에 있어서, 상기 제2포인터 배열 저장 모듈은,

자식 노드에 대해서는 32bit 포인터 값을 가지고 추가적인 정보를 제공하기 위해 4bit의 인덱스 값과 28bit의 포인터 값 혹은 피벗으로 구성하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 17

제11항에 있어서, 상기 제2포인터 배열 저장 모듈은,

포인터 값과 피벗은 용도에 따라 다른 값으로 저장되지만 공유되는 필드를 가지는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 18

제11항에 있어서, 상기 제2포인터 배열 저장 모듈은,

자식 노드의 고유 포인터들을 배열의 첫 엔트리부터 연속적으로 배치한 후, 피벗 기반 커팅을 위한 피벗과 피벗 기반 커팅을 통해 얻은 추가 노드에 대한 포인터를 배치하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 19

제11항에 있어서, 상기 캐시 라인 저장 모듈은,

하나의 캐시 라인 안에서는 단 한 번의 메모리 액세스만 이루어지도록 하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

청구항 20

제11항에 있어서, 상기 캐시 라인 저장 모듈은,

메모리 액세스 수를 줄이기 위해서 하나의 캐시 라인 사이즈마다 새로운 헤더를 추가하는 것을 특징으로 하는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템.

발명의 설명

기술 분야

[0001] 본 발명은 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 방법 및 시스템에 관한 것으로서, 보다 구체적으로는 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템에 관한 것이다.

배경 기술

[0002] 네트워크 보안 장비들은 다양한 보안 기능을 제공하기 위해 정책을 사용한다. 이 경우 패킷 분류 알고리즘이 사용되어 수신한 패킷들에 대해 매칭되는 보안 정책을 찾고 이에 따라 패킷을 처리하게 된다. 일반적으로 패킷 분류 속도와 테이블의 크기는 비례하는 상관관계를 가지고 있다. 즉, 현재 개발된 많은 고속 패킷 분류 알고리즘들은 빠른 정책 검색을 위해서 매우 큰 테이블을 사용하고 있다. 하지만 대형 테이블은 정책 검색 속도에는 유리하지만 테이블을 생성하거나 변경할 때 지나치게 오랜 시간이 걸린다는 단점이 있다.

[0003] 또한, 네트워크의 크기가 대형화되면서 다양한 보안 공격을 막기 위해 보안 정책들이 복잡해지고 있으며 보안 정책의 수가 빠르게 증가하고 있다. 이에 따라 기존 대형 테이블을 사용하는 패킷 분류 알고리즘의 경우 정책의 수가 증가함에 따라 테이블의 크기가 지나치게 커지면서 대용량 정책을 지원하는데 큰 어려움이 발생하게 된다. 따라서 네트워크 보안 장비를 개발하는데 소형 테이블을 사용하면서 높은 검색 성능을 갖는 패킷 분류 알고리즘에 대한 요구가 커지고 있다. 하지만, 앞서 설명한 것처럼 테이블의 크기와 분류 속도의 관계로 인해 패킷 분류 알고리즘의 속도와 테이블 크기를 동시에 개선하는 것은 기술적으로 매우 어렵다고 말할 수 있다.

[0004] 최근에는 RFC(Recursive Flow Classification)와 같은 cross-producting 기반 알고리즘의 대형 검색 테이블을 사용함으로써 라우팅뿐만 아니라 5 튜플을 사용하는 보안 기능 내 정책 검색 기능까지 지원이 가능하다. RFC를 사용하면 무엇보다도 정책의 수에 상관없이 고정된 검색 성능을 갖는다는 장점이 있다. 하지만 정책이 커질 경우 테이블의 크기가 증가하게 되어 테이블 생성 시간이 길어져 실제 네트워크상에서 사용할 수 없다는 단점이 있다.

[0005] 도 1은 HyperCuts 알고리즘에 의해 커팅이 수행되어 8개의 파티션으로 분할된 모습을 도시한 도면이다. HyperCuts 알고리즘은 패킷 분류 알고리즘 중 하나로, 하나 이상의 차원(필드)에 대해 등간격으로 커팅을 수행한다. 도 1에 도시된 바와 같이, 판별 트리 노드는 HyperCuts 알고리즘에 의해 X 차원에서 4개, Y 차원에서 2개로 분할되어 총 8개의 파티션으로 분할될 수 있다. HyperCuts 알고리즘은 매우 간단하여, 기존의 패킷 분류 알고리즘에 비해 훨씬 높은 성능을 얻을 수 있다. 그러나 이러한 HyperCuts 알고리즘은 커팅에 의해 중복된 파티션들이 많이 생길 수 있다는 문제점이 있다. 도 1에 도시된 바와 같이, P1, P2, P4, P5 및 P6은 동일한 정책을 포함하여 중복된 파티션에 해당하므로 사실상 하나의 파티션에 해당한다. 따라서 HyperCuts 알고리즘에 의해 커팅되어 실제로 분할된 파티션은 4개에 해당한다. HyperCuts 알고리즘과 같이 등간격으로 커팅을 수행하는 경우 해당 차원 상에서 정책 집합의 분포에 대한 정보를 고려하지 않고 단순히 커팅할 범위와 커팅 수에 의해 분할되는 위치가 결정된다는 문제점이 있다. 따라서 최적으로 분할이 이루어지지 않게 되며, 정책 집합이 각각의 차원에 균일하게 분산되어 있지 않으면 중복된 파티션들이 더욱 많이 발생하게 된다. 이러한 특성으로 인해, HyperCuts 알고리즘에 의한 패킷 분류는 검색 속도 저하와 테이블 크기 증가 문제가 빈번하게 발생하고 있다.

[0006] 이와 관련된 선행기술로는, 등록특허 제10-1331018호(발명의 명칭: 패킷 분류 방법 및 그 장치)와 등록특허 제10-0451788호(발명의 명칭: VPN라우터의 LNS 성능향상을 위한 L2TP 패킷고속 수신방법) 등이 제시된 바 있다.

발명의 내용

해결하려는 과제

[0007] 본 발명은 기존에 제안된 방법들의 상기와 같은 문제점들을 해결하기 위해 제안된 것으로서, 정책 검색을 위한 때 단계에서 하나의 방법으로 커팅을 수행하는 대신, 두 가지 방식으로 동시에 커팅을 수행함으로써, 다양한 정책 집합의 특성에도 불구하고 빠르게 커팅을 할 수 있는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템을 제공하는 것을 목적으로 한다.

[0008] 또한, 본 발명은, 캐시 구조에 적합하도록 정교하게 설계된 노드를 사용하여 메모리 액세스 수를 최소화함으로써, 테이블 크기와 패킷 분류 성능을 동시에 개선할 수 있는, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템을 제공하는 것을 다른 목적으로 한다.

과제의 해결 수단

[0009] 상기한 목적을 달성하기 위한 본 발명의 특징에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법은,

[0010] 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 방법에 있어서,

[0011] (1) 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할되는 단계;

[0012] (2) 상기 단계 (1)에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 단계;

[0013] (3) 상기 단계 (2)에서 분할된 서브 파티션에 대해, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장되는 단계;

[0014] (4) 상기 단계 (3)의 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가되는 단계;

[0015] (5) 상기 단계 (4)의 인덱스를 추가한 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장되는 단계; 및

[0016] (6) 상기 단계 (5)의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장되는 단계를 포함하는 것을 그 구성상의 특징으로 한다.

- [0017] 바람직하게는, 상기 단계 (1)은,
- [0018] 한 개 이상의 필드에 대해서 등간격으로 커팅을 수행할 수 있다.
- [0019] 바람직하게는, 상기 단계 (2)는,
- [0020] 상기 등간격 커팅에서 가장 많이 파티셔닝이 수행된 차원에 대하여 상기 피벗 기반 커팅이 적용될 수 있다.
- [0021] 바람직하게는, 상기 단계 (2)는,
- [0022] 상기 단계 (1)에서 분할된 가장 큰 파티션들에 대해 적용되며 전체 파티션들의 가장 큰 크기를 최소화하는 값으로 커팅될 수 있다.
- [0023] 바람직하게는, 상기 피벗 기반 커팅은,
- [0024] 특정 값을 기준으로 작거나 같은 정책들과 큰 정책들로 구성된 서브파티션들로 커팅할 수 있다.
- [0025] 바람직하게는, 상기 단계 (5)는,
- [0026] 자식 노드에 대해서는 32bit 포인터 값을 가지고 추가적인 정보를 제공하기 위해 4bit의 인덱스 값과 28bit의 포인터 값 혹은 피벗으로 구성될 수 있다.
- [0027] 바람직하게는, 상기 단계 (5)는,
- [0028] 포인터 값과 피벗은 용도에 따라 다른 값으로 저장 되지만 공유되는 필드를 가질 수 있다.
- [0029] 바람직하게는, 상기 단계 (5)는,
- [0030] 자식 노드의 고유 포인터들은 배열의 첫 엔트리부터 연속적으로 배치된 후, 피벗 기반 커팅을 위한 피벗과 피벗 기반 커팅을 통해 얻은 추가 노드에 대한 포인터가 배치될 수 있다.
- [0031] 바람직하게는, 상기 단계 (6)은,
- [0032] 하나의 캐시 라인 안에서는 단 한 번의 메모리 액세스만 이루어질 수 있다.
- [0033] 바람직하게는, 상기 단계 (6)은,
- [0034] 메모리 액세스 수를 줄이기 위해서 하나의 캐시 라인 사이즈마다 새로운 헤더를 추가할 수 있다.
- [0035] 상기한 목적을 달성하기 위한 본 발명의 특징에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템은,
- [0036] 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 시스템에 있어서,
- [0037] 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할하는 등간격 커팅 모듈;
- [0038] 상기 등간격 커팅 모듈에 의해 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅을 적용하여 서브 파티션으로 분할하는 피벗 기반 커팅 모듈;
- [0039] 상기 피벗 기반 커팅 모듈에 의해 분할된 서브 파티션에 대해, 포인터 배열에서 중복을 제거한 서브 파티션에

대한 고유 포인터들을 저장하는 제1포인터 배열 저장 모듈;

- [0040] 상기 제1포인터 배열 저장 모듈에 의해 저장된 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스를 추가시키는 인덱스 추가 모듈;
- [0041] 상기 인덱스 추가 모듈에 의해 인덱스가 추가된 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜진 두 파티션에 대한 포인터들을 저장하는 제2포인터 배열 저장 모듈; 및
- [0042] 제2포인터 배열 저장 모듈에 의해 포인터들이 저장된 포인터 배열에서, 자식 노드 배열을 하나의 캐시 라인에 저장하는 캐시 라인 저장 모듈을 포함하는 것을 그 구성상의 특징으로 한다.
- [0043] 바람직하게는, 상기 등간격 커팅 모듈은,
- [0044] 한 개 이상의 필드에 대해서 등간격으로 커팅을 수행할 수 있다.
- [0045] 바람직하게는, 상기 피벗 기반 커팅 모듈,
- [0046] 상기 등간격 커팅에서 가장 많이 파티셔닝이 수행된 차원에 대하여 상기 피벗 기반 커팅을 할 수 있다.
- [0047] 바람직하게는, 상기 피벗 기반 커팅 모듈은,
- [0048] 상기 등간격 커팅 모듈에 의해 분할된 가장 큰 파티션들에 대해 적용되며 전체 파티션들의 가장 큰 크기를 최소화하는 값으로 커팅할 수 있다.
- [0049] 바람직하게는, 기 피벗 기반 커팅 모듈은,
- [0050] 특정 값을 기준으로 작거나 같은 정책들과 큰 정책들로 구성된 서브파티션들로 커팅할 수 있다.
- [0051] 바람직하게는, 상기 제2포인터 배열 저장 모듈은,
- [0052] 자식 노드에 대해서는 32bit 포인터 값을 가지고 추가적인 정보를 제공하기 위해 4bit의 인덱스 값과 28bit의 포인터 값 혹은 피벗으로 구성할 수 있다.
- [0053] 바람직하게는, 상기 제2포인터 배열 저장 모듈은,
- [0054] 포인터 값과 피벗은 용도에 따라 다른 값으로 저장 되지만 공유되는 필드를 가질 수 있다.
- [0055] 바람직하게는, 상기 제2포인터 배열 저장 모듈은,
- [0056] 자식 노드의 고유 포인터들을 배열의 첫 엔트리부터 연속적으로 배치한 후, 피벗 기반 커팅을 위한 피벗과 피벗 기반 커팅을 통해 얻은 추가 노드에 대한 포인터를 배치할 수 있다.
- [0057] 바람직하게는, 상기 캐시 라인 저장 모듈은,
- [0058] 하나의 캐시 라인 안에서는 단 한 번의 메모리 액세스만 이루어질 수 있다.
- [0059] 바람직하게는, 상기 캐시 라인 저장 모듈은,
- [0060] 메모리 액세스 수를 줄이기 위해서 하나의 캐시 라인 사이즈마다 새로운 헤더를 추가할 수 있다.

발명의 효과

- [0061] 본 발명에서 제안하고 있는 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템에 따르면, 정책 검색을 위한 매 단계에서 하나의 방법으로 커팅을 수행하는 대신, 두 가지 방식으로 동시에 커팅을 수행함으로써, 다양한 정책 집합의 특성에도 불구하고 빠르게 커팅을 할 수 있다.
- [0062] 또한, 본 발명에 따르면, 캐시 구조에 적합하도록 정교하게 설계된 노드를 사용하여 메모리 액세스 수를 최소화함으로써, 테이블 크기와 패킷 분류 성능을 동시에 개선할 수 있다.

도면의 간단한 설명

- [0063] 도 1은 HyperCuts 알고리즘에 의해 커팅이 수행되어 8개의 파티션으로 분할된 모습을 도시한 도면.
- 도 2는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 구성을 도시한 도면.
- 도 3은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 도 1에서 분할된 파티션에 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 모습을 도시한 도면.
- 도 4는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 결정트리 상 서브 파티션 노드에 대한 포인트 배열 구조를 도시한 도면.
- 도 5는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 중복된 서브 파티션 포인터들을 제거하고 노드 포인터에 대한 인덱스를 추가한 포인터 배열 구조를 도시한 도면.
- 도 6은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 피벗과 피벗 기반 파티션 포인터를 추가한 포인터 배열 구조를 도시한 도면.
- 도 7은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 캐시 최적화된 포인터 배열 구조를 도시한 도면.
- 도 8은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 정책 종류에 따른 정책 크기별 상대적인 메모리 액세스 양을 도시한 도면.
- 도 9는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 캐시를 고려한 정책 종류에 따른 정책 크기별 상대적인 메모리 액세스 양을 도시한 도면.
- 도 10은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 정책 종류에 따른 정책 크기별 테이블 상대 크기를 도시한 도면.
- 도 11은 본 발명의 다른 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템(100)의 세부 구성을 도시한 도면.

발명을 실시하기 위한 구체적인 내용

- [0064] 이하에서는 첨부된 도면을 참조하여 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 본 발명을 용이하게 실시할 수 있도록 바람직한 실시예를 상세히 설명한다. 다만, 본 발명의 바람직한 실시예를 상세하게 설명함에 있어, 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략한다. 또한, 유사한 기능 및 작용을 하는 부분에 대해서는 도면 전체에 걸쳐 동일 또는 유사한 부호를 사용한다.
- [0065] 덧붙여, 명세서 전체에서, 어떤 부분이 다른 부분과 ‘연결’되어 있다고 할 때, 이는 ‘직접적으로 연결’되어 있는 경우뿐만 아니라, 그 중간에 다른 소자를 사이에 두고 ‘간접적으로 연결’되어 있는 경우도 포함한다. 또한, 어떤 구성요소를 ‘포함’ 한다는 것은, 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있다는 것을 의미한다.

[0066] 도 2는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 구성을 도시한 도면이다. 도 2에 도시된 바와 같이, 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법은, 커팅 방식을 이용하여 패킷을 분류하는 패킷 분류 방법에 있어서, (1) 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할되는 단계(S100); (2) 단계 S100에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 단계(S200); (3) 단계 S200에서 분할된 서브 파티션에 대해, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장되는 단계(S300); (4) 단계 S300의 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가되는 단계(S400); (5) 단계 S400의 인덱스를 추가한 포인터 배열에서, 빈 엔트리에 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장되는 단계(S500); 및 (6) 단계 S500의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장되는 단계(S600)를 포함할 수 있다. 이하에서는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 각각의 구성에 대해 상세히 설명하기로 한다.

[0067] 단계 S100에서는, 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할될 수 있다. 등간격 커팅은, 정책집합을 작은 정책집합으로 나누는 커팅 방법일 수 있다. 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 등간격 커팅은, HyperCuts 알고리즘과 유사한 방식으로, 한 개 이상의 차원(필드)에 대해서 등간격으로 커팅을 수행할 수 있다.

[0068] 단계 S200에서는, 단계 S100에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할될 수 있다. 피벗 기반 커팅은, 특정 값을 기준으로 작거나 같은 정책 집합과 큰 정책 집합으로 구성된 파티션들로 커팅하는 분할 방법일 수 있다. 이때, 커팅에 사용되는 특정 값을 피벗이라고 부를 수 있다. 피벗 기반 커팅은, 등간격 커팅이 적용되어 가장 많은 수로 커팅된 파티션을 가지는 차원에 대해서 피벗 기반 커팅이 적용될 수 있다. 도 3은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 도 1에서 분할된 파티션에 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 모습을 도시한 도면이다. 도 3에 도시된 바와 같이, 등간격 커팅으로 인해 X 차원에 대해서 4개, Y 차원에 대해서 2개의 파티션으로 커팅되었으므로, X 차원에 대해서 피벗 기반 커팅이 적용될 수 있다.

[0069] 단계 S300에서는, 단계 S200에서 분할된 서브 파티션들에 대한 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장될 수 있고 단계 S400에서는, 단계 S300의 포인터 배열에서 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가될 수 있다. 도 4는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 결정트리상 서브 파티션 노드에 대한 포인트 배열 구조를 도시한 도면이고, 도 5는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 중복된 서브 파티션 포인터들을 제거하고 노드 포인터에 대한 인덱스를 추가한 포인터 배열 구조를 도시한 도면이다. 도 4 및 도 5에서 PTRk는 k번째 고유 파티션에 대한 포인터를 나타낸다. 도 4 및 도 5에 도시된 바와 같이, 피벗 기반 커팅을 한 후에 총 파티션의 수(N)는 8개이지만, 실제로는 단 4개의 파티션만을 얻게 된다. 이를 해결하기 위해 도 4의 배열에서 중복을 제거한 파티션에 대한 고유 포인터들만 저장하도록 할 수 있다. 이때, 8개의 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위해 도 5와 같이, 기존 포인터의 값을 가리키는 인덱스를 추가하도록 할 수 있다. 예를 들면, 3번째 엔트리에 PTR3이 저장되어 있지만 실제 필요한 값은 PTR2이다. 이를 얻기 위해서는 인덱스 1을 통해 두 번째 엔트리에 실제 값이 저장되어 있음을 알 수 있고 PTR2를 찾을 수 있다.

[0070] 단계 S500에서는, 단계 S400의 인덱스를 추가한 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장될 수 있다. 이 경우, 공유 포인터의 수를 u 라 하고 중복 포함한 전체 파티션의 수를 N 이라고 할 때, 하나의 피벗으로 두 개의 파티션을 생성하므로 총 파티션의 수는 $u + [2/3 \times (N - u)]$ 이 되므로, 결국 최종 파티션의 수는 $\Theta(N)$ 이 된다.

[0071] 설명을 쉽게 하기 위해서 기존의 HyperCuts 알고리즘과 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 자식 노드에 대한 포인터 배열의 k번째 엔트리를 각각 $c[k]$ 와 $C[k]$ 로 표현하기로 한다. 기존 HyperCuts에서 $c[k]$ 는 자식 노드에 대한 32bit 포인터 값이지만, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 노드의 크기를 HyperCuts과 동일하게 유지하면서 추가적인 정보를 제공하기 위해서는 $C[k]$ 는 4 bit 인덱스 값과 28bit 포인터 값 혹은 피벗으로 구성될 수 있다. 이를 각각 $C[k].idx$, $C[k].ptr$ 와, $C[k].pivot$ 으로 부르기로 하면, 포인터 값과 피벗은 용도에 따라 다른 값으로 저장되지만 공유되는 필드를 가질 수 있다. 또한 $C[k].ptr=c[k]>>2$ 로 설정될 수 있다. 여기서 ‘>>’는 오른쪽 비트 이동 연산자이다. 이는 곧 자식 노드의 주소 범위가 2^{30} 즉, 1GB로 제한된다는 것을 의미하며, 결국 총 테이블의 크기는 1GB로 제한될 수 있다. 이는 대형 정책을 지원하는데 한계가 될 수도 있지만 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서는 전체 테이블의 크기를 HyperCuts에 비해 크게 줄인다면 이 문제를 상쇄시킬 수 있다.

[0072] 자식 노드의 고유 포인터들은 배열의 첫 엔트리부터 연속적으로 배치된 후, 피벗 기반 커팅을 위한 피벗과 피벗 기반 커팅을 통해 얻은 추가 노드에 대한 포인터들이 배치될 수 있다. 도 6은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 피벗과 피벗 기반 파티션 포인터를 추가한 포인터 배열 구조를 도시한 도면이다. 도 6에 도시된 바와 같이, 피벗 기반 커팅에 대한 값들이 저장되어 있는 첫 엔트리(PVT1이 저장된 엔트리)의 위치, 즉 2는 배열의 첫 인덱스에 저장될 수 있다.

[0073] 도 6에서, 회색으로 표시된 인덱스 2는 두 번째 엔트리부터 피벗, 피벗으로 인한 두 서브 파티션 포인터 등 총 세 개의 값들이 연속해서 저장되어 있음을 나타낸다. 편의상 피벗 인덱스가 저장되는 첫 엔트리를 헤더라 부르면, 앞서 설명한대로 $0 < k < c[0].idx$ 인 $c[k]$ 에 대해서 $c[k]$ 값은 원래 저장되어야 하는 포인터 값이 존재하는 엔트리의 인덱스로 설정될 수 있다. 따라서 $c[k]=C[C[k].idx].ptr < 2$ 일 수 있다. 여기서, ‘<<’는 왼쪽 비트 이동 연산자를 의미할 수 있다.

[0074] $k \geq c[0].idx$ 인 경우에는 커팅을 위한 피벗과 피벗보다 큰 값으로 구성된 파티션 노드에 대한 포인터들이 순서대로 저장될 수 있다. SIP나 DIP 필드에 대한 피벗 값의 경우 일반 포인터와 마찬가지로 32bit 값이어야 하지만 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서는 28bit 값으로 사용할 수 있다. 이를 위해 다음과 같이 동작할 수 있다.

[0075] 현재 노드에서 선택된 최대 커팅 차원, D에 대해 모든 부모 노드로부터 현재 노드까지 커팅을 한 수가 24, 즉 16을 넘는 경우와 넘지 않는 경우에 대해서 다른 값으로 설정될 수 있다. 현재 노드에서 값 기반 커팅을 할 것에 대해 매칭되는 D의 범위가 D_s, D_c 라고 하면, $C[k].pivot$ 은 다음의 수학적 식 1에 의해 설정될 수 있다.

수학적 식 1

$$C[k].pivot = \begin{cases} P - D_s & \text{if } N_C^D \geq 16 \\ (P - D_s) \gg \lfloor \log_2 N_C^D \rfloor & \text{if } N_C^D < 16 \end{cases}$$

[0077] 단계 S600에서는, 단계 S500의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장될 수 있다. 또한, 하나의 캐시 라인 안에서는 단 한 번의 메모리 액세스만이 이루어질 수 있다. 이때, 포인터 배열의 크기는 파티션 수에 따라 결정되므로 특정 환경에서는 포인터 배열의 크기가 캐시 라인 사이즈를 넘을 수 있으므로, 메모리 액세스 수를 줄이기 위해서 하나의 캐시 라인 사이즈마다 새로운 헤더를 추가하도록 할 수 있다. 캐시 라인 사이즈가 64바이트인 경우, 엔트리의 수가 16개 이하인 경우에는 하나의 헤더만으로 충분할 수 있다. 하지만 엔트리의 수가 16개를 넘는 경우, 16개의 엔트리마다 1개의 헤더가 추가로 존재할 수 있다.

- [0078] 도 7은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 캐시 최적화된 포인터 배열 구조를 도시한 도면이다. 도 7에 도시된 바와 같이, 메모리 액세스 시 함께 캐싱되는 메모리 영역 내에 필요한 엔트리들이 배치됨으로써, 메모리 액세스 수를 크게 감소시킬 수 있다.
- [0079] 이하에서는 도 8 내지 9를 참조하여, 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 성능을 비교한다. 성능을 비교하기 위해서 Classbench를 이용하여 생성한 정책집합을 사용하였다. 정책집합은 Access Control List(ACL), Firewall(Fw)와 IP Chain(IPC) 등 총 세 가지를 이용하였으며 테이블 크기, 메모리 액세스 량, 캐시라인 기준 메모리 액세스 수를 측정한다. 또한, 정책 수에 따라 성능을 측정함으로써, 정책 크기에 따른 경향성을 비교하였다.
- [0080] 도 8은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 정책 종류에 따른 정책 크기별 상대적인 메모리 액세스 양을 도시한 도면이다. 일반적으로 메모리 성능이 전체 성능의 병목이 되므로 메모리 액세스 양을 줄이는 것이 성능 향상에서 가장 중요하다. 도8 도시된 바와 같이, 메모리 액세스 양은 기존 HyperCuts 대비 30% 이상 감소함을 알 수 있다. 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법은 한 노드에서 액세스하는 양이 증가를 하지만 디지전 트리의 높이 자체를 낮추기 때문에 전체적인 메모리 액세스 양은 오히려 감소하는 것을 알 수 있다. 따라서 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법이 기존 HyperCuts의 단점을 효과적으로 개선하고 있음을 알 수 있다.
- [0081] 도 9는 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 캐시를 고려한 정책 종류에 따른 정책 크기별 상대적인 메모리 액세스 양을 도시한 도면이다. 본 성능평가에서 캐시 라인 사이즈는 64바이트이다. 메모리 액세스 양은 실제로 캐시에 의한 영향을 고려하고 있지 않으므로 정확한 성능을 반영하지는 못한다. 따라서 정확한 분석을 위해서 캐시라인 액세스 수를 측정하는 것이 필요하다. 도 9에 도시된 바와 같이, 실제 메모리 액세스 수는 기존 HyperCuts 대비 55% 이상 감소되어 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법이 기존 방법에 비해 2배 이상의 성능을 제공할 수 있음을 보여준다. 이는 캐시라인을 고려하지 않는 경우에 비해 더욱 성능이 개선되고 있음을 의미한다. 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 캐시 지향으로 설계방식이 성능 개선에 매우 유리함을 증명한다.
- [0082] 도 10은 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법에서 정책 종류에 따른 정책 크기별 테이블 상대 크기를 도시한 도면이다. 테이블 크기는 지원하는 정책의 수에 영향을 주기 때문에 매우 중요한 성능 측정 요소이다. 도 10에 도시된 바와 같이, 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법은 모든 정책 종류와 정책 크기에 대해 테이블 크기를 크게 개선함을 알 수 있다.
- [0083] 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법은 노드의 크기를 기존 대비 증가하지 않도록 설계하였기 때문에 테이블 크기를 줄이도록 설계한 것은 아니다. 하지만 이중 커팅이 내부적으로 커팅 효율을 크게 개선함으로써 실제 필요한 자식 노드의 수를 크게 줄여 테이블 크기가 크게 감소할 수 있다. 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법과 HyperCuts를 비교할 때, ACL과 IPC의 경우에는 25%, FW의 경우 35%가 감소된 것을 확인할 수 있다. 따라서 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법이 내부적으로 28비트 포인터를 사용함으로써 1GB로 테이블 크기 제약도 사실상 없음을 알 수 있다.
- [0084] 도 11은 본 발명의 다른 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템(100)의 세부 구성을 도시한 도면이다. 도 11에 도시된 바와 같이, 본 발명의 다른 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템은, 등간격 커팅 모듈(110), 피벗 기반 커팅 모듈(120), 제1포인터 배열 저장 모듈(130), 인덱스 추가 모듈(140), 제2포인터 배열 저장 모듈(150), 및 캐시 라인 저장 모듈(160)을 포함할 수 있다.

[0085] 등간격 커팅 모듈(110)은, 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅을 적용하여 각각의 파티션으로 분할할 수 있다. 피벗 기반 커팅 모듈(120)은, 상기 등간격 커팅 모듈(110)에 의해 분할된 파티션에 대해 피벗 기반 커팅을 적용하여 서브 파티션으로 분할할 수 있다. 제1포인터 배열 저장 모듈(130)은, 상기 피벗 기반 커팅 모듈(120)에 의해 분할된 서브 파티션들에 대한, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들을 저장할 수 있다. 인덱스 추가 모듈(140)은, 상기 제1포인터 배열 저장 모듈(130)에 의해 저장된 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스를 추가 할 수 있다. 제2포인터 배열 저장 모듈(150)은, 상기 인덱스 추가 모듈(140)에 의해 인덱스가 추가된 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들을 저장할 수 있다. 캐시 라인 저장 모듈(160)은, 상기 제2포인터 배열 저장 모듈(150)에 의해 포인터들이 저장된 포인터 배열에서, 자식 노드 배열을 하나의 캐시 라인에 저장할 수 있다. 각각의 모듈에서 수행되는 정보 처리는, 전술한 본 발명의 일 실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법의 각각의 단계에서 수행되는 정보 처리와 대응될 수 있다.

[0086] 전술한 바와 같이, 본 발명에서 제안하고 있는 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 방법 및 시스템에 따르면, 정책 검색을 위한 매 단계에서 하나의 방법으로 커팅을 수행하는 대신, 두 가지 방식으로 동시에 커팅을 수행함으로써, 다양한 정책 집합의 특성에도 불구하고 빠르게 커팅을 할 수 있다. 또한 본 발명에 따르면, 캐시 구조에 적합하도록 정교하게 설계된 노드를 사용하여 메모리 액세스 수를 최소화함으로써, 테이블 크기와 패킷 분류 성능을 동시에 개선할 수 있다.

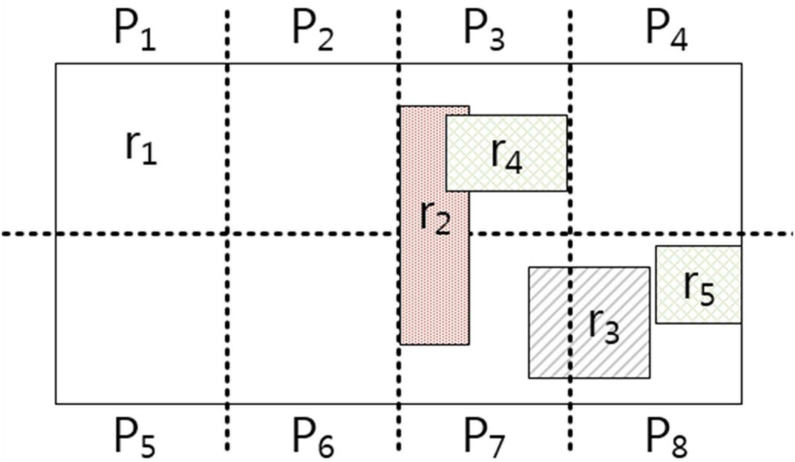
[0087] 이상 설명한 본 발명은 본 발명이 속한 기술분야에서 통상의 지식을 가진 자에 의하여 다양한 변형이나 응용이 가능하며, 본 발명에 따른 기술적 사상의 범위는 아래의 특허청구범위에 의하여 정해져야 할 것이다.

부호의 설명

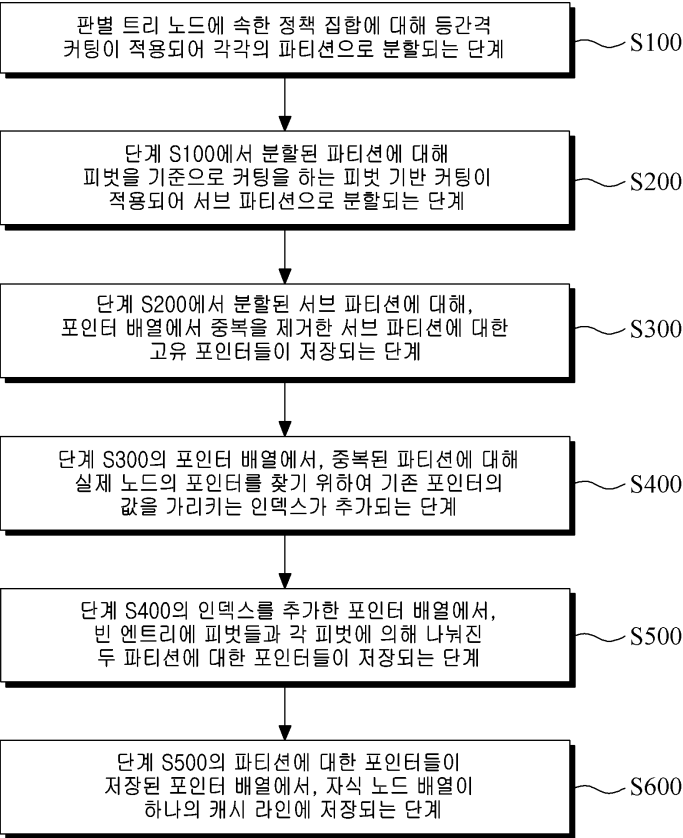
[0088] 100: 본 발명의 일실시예에 따른 고속 보안 정책 검색을 위한 이중 커팅 기반 패킷 분류 시스템
 110: 등간격 커팅 모듈
 120: 피벗 기반 커팅 모듈
 130: 제1포인터 배열 저장 모듈
 140: 인덱스 추가 모듈
 150: 제2포인터 배열 저장 모듈
 160: 캐시 라인 저장 모듈
 S100: 판별 트리 노드에 속한 정책 집합에 대해 등간격 커팅이 적용되어 각각의 파티션으로 분할되는 단계
 S200: 단계 S100에서 분할된 파티션에 대해 피벗을 기준으로 커팅을 하는 피벗 기반 커팅이 적용되어 서브 파티션으로 분할되는 단계
 S300: 단계 S200에서 분할된 서브 파티션에 대해, 포인터 배열에서 중복을 제거한 서브 파티션에 대한 고유 포인터들이 저장되는 단계
 S400: 단계 S300의 포인터 배열에서, 중복된 파티션에 대해 실제 노드의 포인터를 찾기 위하여 기존 포인터의 값을 가리키는 인덱스가 추가되는 단계
 S500: 단계 S400의 인덱스를 추가한 포인터 배열에서, 빈 엔트리의 피벗들과 각 피벗에 의해 나뉜 두 파티션에 대한 포인터들이 저장되는 단계
 S600: 단계 S500의 파티션에 대한 포인터들이 저장된 포인터 배열에서, 자식 노드 배열이 하나의 캐시 라인에 저장되는 단계

도면

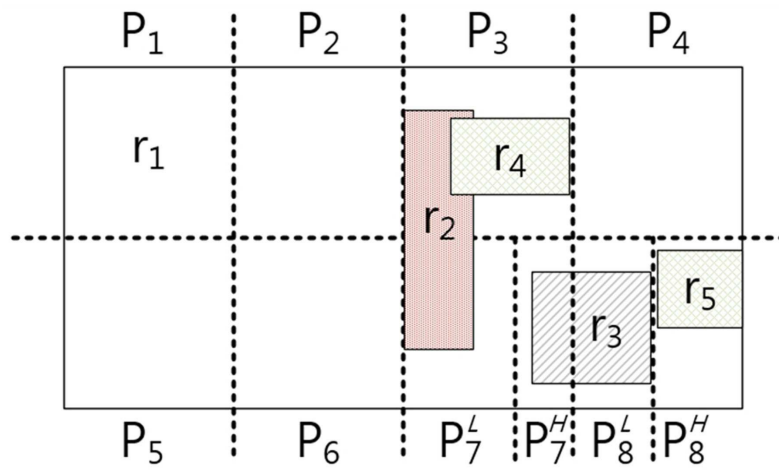
도면1



도면2



도면3



도면4

PTR ₁	PTR ₁	PTR ₂	PTR ₁
PTR ₁	PTR ₁	PTR ₃	PTR ₄

도면5

0	PTR ₁	0	PTR ₂	1	PTR ₃	0	PTR ₄
0		0		2		3	

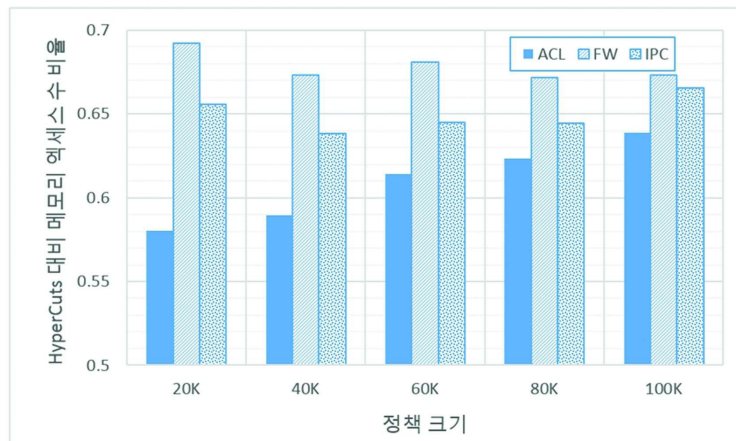
도면6

2	PTR ₁	0	PTR ₂	1	PVT ₁	0	PTR ₃ ^L
0	PTR ₃ ^H	0	PVT ₂	2	PTR ₄ ^L	3	PTR ₄ ^H

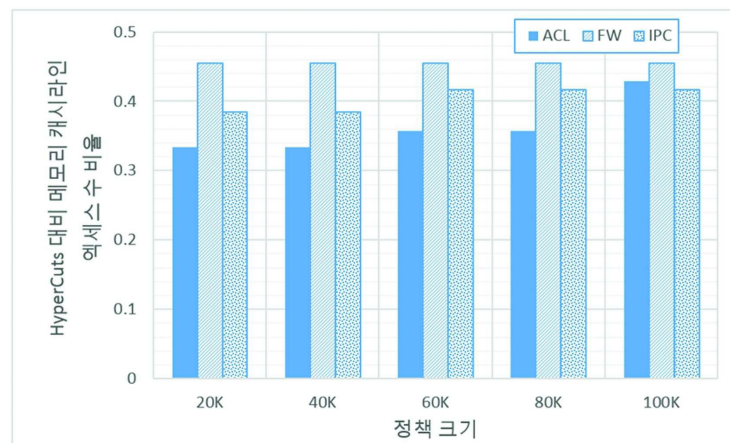
도면7

2	PTR ₁	0	PTR ₂	1	PVT ₁	0	PTR ₃ ^L
0	PTR ₃ ^H	0	PVT ₂	2	PTR ₄ ^L	3	PTR ₄ ^H
4	PTR ₅	0	PTR ₆	2	PTR ₇	1	PTR ₈
1	PVT ₃	1	PTR ₉ ^L	3	PTR ₉ ^H	1	

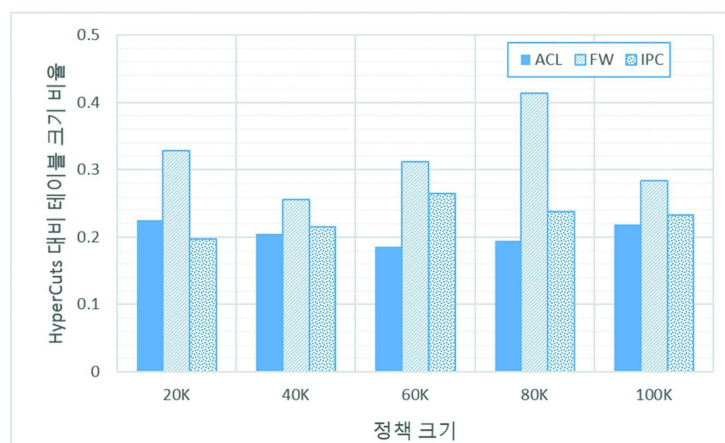
도면8



도면9



도면10



도면11

