(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년09월05일
(11) 등록번호 10-1061833
(24) 등록일자 2011년08월29일

(51) Int. Cl.
H04L 12/56 (2006.01)
(21) 출원번호 10-2010-0081704
(22) 출원일자 2010년08월24일
심사청구일자 2010년08월24일
(30) 우선권주장
1020100019209 2010년03월03일 대한민국(KR)
(56) 선행기술조사문헌
Tuple Pruning Using Bloom Filters for Packet Classification(대한전자공학회, ITC-CSCC : 2009, p.1358-1359)
acket Classification Algorithms Using Bloom Filters(Thesis(M.A.), 이화여자대학교 대학원 : 전자정보통신공학과 2008. 8)
전체 청구항 수 : 총 19 항

(73) 특허권자
이화여자대학교 산학협력단
서울 서대문구 대현동 11-1 이화여자대학교내
(72) 발명자
임혜숙
서울특별시 강남구 도곡2동 삼성래미안아파트 108-1102호
(74) 대리인
이철희

심사관 : 김세영

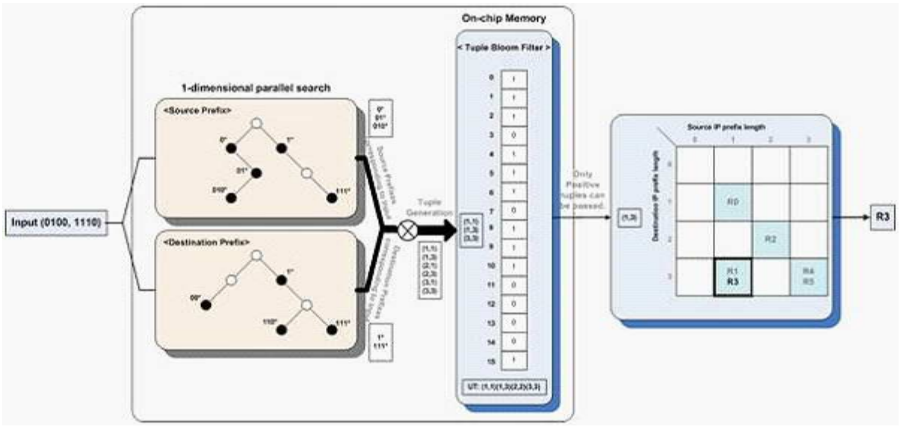
(54) 블룸 필터를 이용한 패킷 분류 방법 및 패킷 분류 장치

(57) 요 약

본 발명은 블룸 필터를 이용한 패킷 분류 방법 및 패킷 분류 장치에 관한 것으로, (a) 입력 패킷의 하나 이상의 필드에 대해 필드별 검색을 수행하여 상기 하나 이상의 필드 각각의 필드값과 룰 셋에 저장된 대응하는 필드의 필드값이 서로 일치하는 길이(이하, "일치 길이"라 함)를 판단하고, 상기 하나 이상의 필드 각각에 대한 상기 일치 길이의 조합(Combination)으로 이루어진 튜플(Tuple) 리스트를 생성하는 단계 (b) 상기 튜플 리스트에서 상기 룰 셋에 존재하는 튜플만을 선별하는 단계 (c) 블룸 필터(Bloom Filter)를 이용하여 선별된 튜플 각각에 대해 필터링을 수행하는 단계 및 (d) 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 하여 최우선 룰(BMR: Best Matching Rule)을 검색하는 단계를 포함한다.

본 발명에 의하면, 필드의 길이에 대한 정보 뿐만 아니라 필드의 값에 대한 정보를 이용하여 검색 대상 튜플 수를 줄일 수 있으므로, 메모리 접근 횟수를 줄여 검색 성능을 향상시킬 수 있다.

대 표 도 - 도2



특허청구의 범위

청구항 1

패킷 분류 방법에 있어서,

(a) 입력 패킷의 하나 이상의 필드에 대해 필드별 검색을 수행하여 상기 하나 이상의 필드 각각의 필드값과 룰 셋에 저장된 대응하는 필드의 필드값이 서로 일치하는 길이(이하, "일치 길이"라 함)를 판단하고, 상기 하나 이상의 필드 각각에 대한 상기 일치 길이의 조합(Combination)으로 이루어진 튜플(Tuple) 리스트를 생성하는 단계

(b) 상기 튜플 리스트에서 상기 룰 셋에 존재하는 튜플만을 선별하는 단계

(c) 블룸 필터(Bloom Filter)를 이용하여 선별된 튜플 각각에 대해 필터링을 수행하는 단계 및

(d) 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 하여 최우선 룰(BMR: Best Matching Rule)을 검색하는 단계

를 포함하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 2

제 1 항에 있어서, 상기 (c) 단계는,

(c1) 상기 선별된 튜플 각각에 대해 해싱 인덱스(Hashing Index)를 생성하는 단계

(c2) 상기 해싱 인덱스가 가리키는 상기 블룸 필터의 엔트리에 접근하여 상기 블룸 필터의 엔트리의 비트 벡터를 확인하는 단계 및

(c3) 상기 비트 벡터가 양성인 튜플만을 상기 검색 대상으로 선별하는 단계

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 3

제 2 항에 있어서,

상기 (d) 단계는, 상기 해싱 인덱스가 가리키는 해싱 테이블의 엔트리에 접근하여 상기 해싱 테이블의 엔트리에 저장된 룰을 검색하고, 검색된 룰 중에서 상기 최우선 룰을 결정하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 4

제 2 항에 있어서, 상기 (c1) 단계는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각에서 추출하여 서로 연결(Concatenation)함으로써 생성된 스트링(String)으로부터 상기 해싱 인덱스를 생성하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 5

제 2 항에 있어서, 상기 (c1) 단계는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각에서 추출하고 추출된 값 각각에 소정의 비트 패턴을 부가하여 고정길이 값으로 패딩(Padding)한 후 패딩된 값들을 서로 연결하여 생성한 스트링(String)으로부터 상기 해싱 인덱스를 생성하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 6

제 4 항 또는 제 5 항에 있어서, 상기 (c1) 단계는,

상기 스트링을 CRC 생성기(Cyclic Redundancy Check Generator)에 입력하여 고정 길이 스트링을 생성하는 단계

및

상기 고정 길이 스트링으로부터 복수의 상기 해싱 인덱스를 생성하는 단계

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 7

제 1 항에 있어서,

상기 하나 이상의 필드는 근원지 프리픽스(Source Prefix) 필드와 목적지 프리픽스(Destination Prefix) 필드인 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 8

제 1 항에 있어서,

상기 (a) 단계에서의 상기 필드별 검색은 Binary Search on Level을 이용한 1차원 검색인 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 9

제 2 항에 있어서, 상기 (d) 단계는,

(d1) 상기 검색 대상에 속한 튜플에 대해 해싱 테이블용 해싱 인덱스를 생성하는 단계 및

(d2) 상기 해싱 테이블용 해싱 인덱스가 가리키는 해싱 테이블의 엔트리에 접근하여 상기 해싱 테이블의 엔트리에 속한 룰을 검색하는 단계 및

(d3) 검색된 룰 중에서 상기 최우선 룰을 결정하는 단계

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 방법.

청구항 10

패킷 분류 장치에 있어서,

패킷 분류를 위한 룰을 구비한 해싱 테이블을 저장하는 메모리

입력 패킷의 하나 이상의 필드에 대해 필드별 검색을 수행하여 상기 하나 이상의 필드 각각의 필드값과 룰 셋에 저장된 대응하는 필드의 필드값이 서로 일치하는 길이(이하, "일치 길이"라 함)를 판단하고, 상기 하나 이상의 필드 각각에 대한 상기 일치 길이의 조합으로 이루어진 튜플(Tuple) 리스트를 생성하며, 상기 튜플 리스트에서 상기 룰 셋에 존재하는 튜플만을 선별하는 튜플 생성부 및

상기 튜플 생성부에 의해 선별된 튜플 각각에 대해 블룸 필터를 이용하여 필터링을 수행하고 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 하여 상기 해싱 테이블에 접근하여 최우선 룰(BMR: Best Matching Rule)을 검색하는 패킷 분류 제어부

를 포함하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 11

제 10 항에 있어서, 상기 패킷 분류 제어부는,

상기 선별된 튜플 각각에 대해 해싱 인덱스를 생성하는 해싱부

상기 해싱 인덱스가 가리키는 엔트리의 비트 벡터를 확인하여 상기 비트 벡터가 양성인 튜플만을 상기 검색 대상으로 선별하는 블룸 필터 및

상기 검색 대상에 속한 튜플에 대응하는 상기 해싱 인덱스를 이용하여 상기 해싱 테이블의 엔트리에 접근하여 상기 해싱 테이블의 엔트리에 저장된 룰을 검색하고, 검색된 룰 중에서 상기 최우선 룰을 결정하는 룰 검색부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 12

제 11 항에 있어서, 상기 해싱부는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각에서 추출하여 서로 연결(Concatenation)함으로써 생성된 스트링(String)을 입력받아 고정 길이 스트링을 생성하는 CRC 생성기(Cyclic Redundancy Check Generator); 및

상기 고정 길이 스트링으로부터 복수의 상기 해싱 인덱스를 생성하는 해싱 인덱스 생성부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 13

제 11 항에 있어서, 상기 해싱부는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각에서 추출하고 추출된 값 각각에 소정의 비트 패턴을 부가하여 고정길이 값으로 패딩(Padding)한 후 패딩된 값들을 서로 연결하여 생성한 스트링(String)을 입력받아 고정 길이 스트링을 생성하는 CRC 생성기(Cyclic Redundancy Check Generator); 및

상기 고정 길이 스트링으로부터 복수의 상기 해싱 인덱스를 생성하는 해싱 인덱스 생성부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 14

제 10 항에 있어서, 상기 패킷 분류 제어부는,

상기 선별된 튜플 각각에 대해 블룸 필터용 해싱 인덱스와 해싱 테이블용 해싱 인덱스를 생성하는 해싱부

상기 블룸 필터용 해싱 인덱스가 가리키는 엔트리의 비트 벡터를 확인하여 상기 비트 벡터가 양성인 튜플만을 검색 대상으로 선별하는 블룸 필터 및

상기 검색 대상에 속한 튜플에 대응하는 상기 해싱 테이블용 해싱 인덱스를 이용하여 상기 해싱 테이블의 엔트리에 접근하여 상기 해싱 테이블의 엔트리에 저장된 룰을 검색하고, 검색된 룰 중에서 상기 최우선 룰을 결정하는 룰 검색부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 15

제 14 항에 있어서, 상기 해싱부는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각의 필드값에서 추출하여 서로 연결(Concatenation)함으로써 생성된 스트링(String)을 입력받아 고정 길이 스트링을 생성하는 CRC 생성기(Cyclic Redundancy Check Generator); 및

상기 고정 길이 스트링으로부터 복수의 상기 블룸 필터용 해싱 인덱스 및 하나의 상기 해싱 테이블용 해싱 인덱스를 생성하는 해싱 인덱스 생성부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 16

제 14 항에 있어서, 상기 해싱부는,

상기 선별된 튜플이 나타내는 상기 일치 길이만큼의 상위 비트를 상기 하나 이상의 필드 각각에서 추출하고 추출된 값 각각에 소정의 비트 패턴을 부가하여 고정길이 값으로 패딩(Padding)한 후 패딩된 값들을 서로 연결하여 생성한 스트링(String)을 입력받아 고정 길이 스트링을 생성하는 CRC 생성기(Cyclic Redundancy Check Generator); 및

상기 고정 길이 스트링으로부터 복수의 상기 블룸 필터용 해싱 인덱스 및 하나의 상기 해싱 테이블용 해싱 인덱

스를 생성하는 해싱 인덱스 생성부

를 포함하는 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 17

제 10 항에 있어서,

상기 하나 이상의 필드는 근원지 프리픽스(Source Prefix) 필드와 목적지 프리픽스(Destination Prefix) 필드인 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 18

제 10 항에 있어서,

상기 하나 이상의 필드는 근원지 포트 넘버 필드와 목적지 포트 넘버 필드인 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

청구항 19

제 10 항에 있어서,

상기 튜플 생성부에서 수행되는 상기 필드별 검색은 Binary Search on Level을 이용한 1차원 검색인 것을 특징으로 하는 블룸 필터를 이용한 패킷 분류 장치.

명 세 서

기 술 분 야

[0001] 본 발명은 블룸 필터(Bloom Filter)를 이용한 패킷 분류 방법 및 패킷 분류 장치에 관한 것이다. 더욱 상세하게는, 입력 패킷의 필드 길이 일치 정보를 근거로 생성된 튜플 리스트를 블룸 필터를 이용하여 필터링함으로써 룰셋과 일치할 가능성이 있는 튜플에 대해서만 검색을 실시하는 패킷 분류 방법 및 패킷 분류 장치에 관한 것이다.

배 경 기 술

[0002] 이 부분에 기술된 내용은 단순히 본발명의 실시예에 대한 배경 정보를 제공할 뿐 종래기술을 구성하는 것은 아니다.

[0003] 응용 프로그램이 요구하는 다양한 서비스를 제공하기 위해, 인터넷 라우터는 입력 패킷의 헤더 정보를 이용하여 패킷(Packet)의 클래스(Class)를 결정하고, 각 클래스에 해당하는 적절한 서비스를 제공해야 한다. 패킷이 속하는 클래스를 결정하는 작업을 패킷 분류라고 하는데, 패킷 분류는 주로 근원지(Source) IP 주소, 목적지(Destination) IP 주소, 근원지 포트넘버, 목적지 포트 넘버, 프로토콜 필드 등을 주로 사용하여 수행된다.

[0004] 패킷 분류의 어려움은 다음과 같다.

[0005] 첫째, 패킷 분류는 다차원 검색이 필수적인데, 각 필드별 검색 방법이 다르다. IP 주소 필드는 최장 길이 일치 방식, 포트 넘버 필드는 영역 일치 방식 그리고 프로토콜 필드는 확정 일치를 수행해야 한다. 각기 다른 방식의 검색을 동시에 수행하여야 하기 때문에 검색이 복잡하고 검색 속도가 저하된다.

[0006] 둘째, 검색을 통해 일치하는 모든 룰을 찾고, 그 중에서 가장 최상위 룰을 선택하는 과정이 부가적으로 필요하다.

[0007] 셋째, 이러한 모든 동작은 입력되는 패킷의 속도에 맞추어 선-속도로 처리되어야 한다.

[0008] 따라서 많은 연구들이 라우터의 패킷 처리 속도를 높이기 위해 진행되고 있다. 특히 다차원 검색의 어려움을 극복하기 위해 모든 필드를 사용하여 패킷을 분류하기 보다는 일부 필드만을 사용하여 검색 속도를 개선하는 방법들이 제안되고 있으며, 대표적인 것이 튜플(Tuple) 공간 검색 알고리즘이다.

표 1

[0009]

Rule No.	Source Prefix	Destination Prefix	Tuple
R0	1*	1*	(1,1)
R1	1*	110*	(1,3)
R2	01*	00*	(2,2)
R3	0*	111*	(1,3)
R4	111*	110*	(3,3)
R5	010*	110*	(3,3)

[0010] 튜플 공간이란 각 필드의 길이의 조합에 의한 공간으로서, 예를 들어 표 1과 같은 예시 룰 셋에서 R0는 근원지 프리픽스의 길이가 1이고 목적지 프리픽스의 길이가 1이므로 (1,1)튜플에 속하고, R1은 근원지 프리픽스의 길이가 1이고 목적지 프리픽스의 길이가 3이므로 (1,3)튜플에 속하게 된다.

[0011] 표 1에서와 같이 근원지 프리픽스와 목적지 프리픽스의 길이 조합으로 이루어진 튜플 공간을 정의한다면, 프리픽스는 0에서 32까지의 길이를 가질 수 있으므로, 최대 33*33=1089 개의 튜플을 만들 수 있다. 따라서 라우터에 입력된 각각의 입력 패킷에 대해 룰이 하나라도 존재하는 모든 튜플을 선형적으로 검색하기에는 많은 시간이 소요되므로, 검색해야 하는 튜플의 수를 줄이기 위한 많은 연구가 진행되어 왔는데, 대표적인 것이 튜플 프루닝 알고리즘(Tuple Pruning Algorithm)이다.

[0012] 튜플 프루닝 알고리즘은 각 필드 별 검색 결과를 사용하여 검색에 불필요한 튜플을 제거함으로써 검색성능을 개선한다. 근원지 프리픽스 필드와 목적지 프리픽스 필드에 대해 각각 1차원 검색을 진행하고, 그 결과로 얻어진 일치하는 프리픽스 길이 정보를 조합하여 튜플을 형성한다. 두 필드의 길이 정보를 조합하기 위해 크로스-프로덕팅(Cross-Producting) 과정이 필요한데, 이것은 각각의 필드 검색 결과 입력과 일치하는 프리픽스들의 길이들에 대하여 모든 가능한 튜플 경우의 수를 생성하는 과정이다. 그러나 크로스-프로덕팅의 결과로 얻어진 튜플들은 각 필드 별로 일치하는 프리픽스 길이의 조합이기 때문에 실제로 룰 셋에 존재하지 않는 튜플들이 생성될 수 있다. 따라서 룰 셋에 존재하지 않는 튜플들을 제거하여 최종적으로 검색하여야 하는 튜플들을 결정한다.

[0013] 도 1은 종래의 튜플 프루닝 알고리즘을 설명하기 위한 도면이다.

[0014] 표 1의 룰 셋에 대하여 1차원 검색을 위한 트라이(Trie)를 만들고, 튜플 공간에 룰을 저장하였다. 입력 패킷의 근원지 프리픽스와 목적지 프리픽스가 (0100, 1110)인 경우에 대한 검색 과정을 예로 들어 설명한다.

[0015] 먼저 근원지 프리픽스 필드와 목적지 프리픽스 필드에 대해 개별적인 검색을 동시에 진행하여 일치하는 모든 길이를 찾는다. 근원지 프리픽스 필드 검색 결과 0*, 01*, 010*과 일치하므로 길이 1, 2, 3이 일치 길이로 판단되고, 목적지 프리픽스 필드에서는 1*와 111*이 일치하므로 길이 1, 3이 일치 길이로 판단된다.

[0016] 1차원 검색 결과로부터 일치하는 프리픽스의 길이 정보를 크로스-프로덕팅을 수행하면 (1,1), (1,3), (2,1), (2,3), (3,1), (3,3) 총 6개의 튜플이 형성된다. 이 때 미리 저장된 튜플 리스트와의 비교를 통해 형성된 튜플들 중에서 룰 셋에 존재하지 않는 튜플 (2,1), (2,3), (3,1)을 제거할 수 있다. 따라서 최종적으로 남게 되는 튜플은 (1,1), (1,3), (3,3) 총 3개가 된다.

[0017] 이 세 개의 튜플에 대하여 입력 (0100, 1110)에 대한 검색은, 첫째로 (1,1) 튜플에 대하여 첫번째 필드의 첫 비트와 두번째 필드의 첫 비트를 연결한 스트링인 '01'를 해시키로 사용하여 해싱을 사용한 검색을 수행한다. 다음은 (1,3) 튜플에 대하여 마찬가지 방법으로 '0111'를 검색하고, 마지막으로 (3,3) 튜플에 대하여 '010111'를 검색하여 일치되는 최우선 룰(BMR: Best Matching Rule)인 R3을 찾게 된다. 이처럼 주어진 입력에 대하여 필드 별 검색을 통해, 검색이 불필요한 튜플을 제거하는 방식을 사용함으로써 검색 속도를 향상시킬 수 있다.

[0018] 그러나 튜플 프루닝 알고리즘에서는 튜플이 프리픽스의 길이의 조합으로 결정되기 때문에 프리픽스의 길이 조합은 존재하지만, 프리픽스의 값이 다른 경우를 걸러낼 수 없다. 앞서 보인 예시에서 제거되고 남은 튜플 (1,1), (1,3), (3,3) 중 (1,1)과 (3,3) 튜플에 저장된 룰은 주어진 입력과 전혀 일치되지 않음에도 불구하고 제거되지 않고 남아서 불필요한 튜플 검색을 유발하였다.

[0019] 즉, 튜플 프루닝 알고리즘은 프리픽스의 길이 정보만으로 튜플을 제거하기 때문에 불필요한 튜플 검색을 유발함으로써, 검색 성능을 저하시키는 단점이 있다.

발명의 내용

해결하려는 과제

[0020] 이러한 문제점을 해결하기 위해, 본 발명은 입력 패킷의 필드 길이 정보를 근거로 생성된 튜플 리스트를 블룸 필터를 이용하여 필터링함으로써 룰 셋과 일치할 가능성이 있는 튜플에 대해서만 검색을 실시하는 패킷 분류 방법 및 패킷 분류 장치를 제공함을 목적으로 한다.

과제의 해결 수단

[0021] 이러한 목적을 달성하기 위해, 본 발명은 (a) 입력 패킷의 하나 이상의 필드에 대해 필드별 검색을 수행하여 상기 하나 이상의 필드 각각의 필드값과 룰 셋에 저장된 대응하는 필드의 필드값이 서로 일치하는 길이(이하, "일치 길이"라 함)를 판단하고, 상기 하나 이상의 필드 각각에 대한 상기 일치 길이의 조합(Combination)으로 이루어진 튜플(Tuple) 리스트를 생성하는 단계 (b) 상기 튜플 리스트에서 상기 룰 셋에 존재하는 튜플만을 선별하는 단계 (c) 블룸 필터(Bloom Filter)를 이용하여 선별된 튜플 각각에 대해 필터링을 수행하는 단계 및 (d) 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 하여 최우선 룰(BMR: Best Matching Rule)을 검색하는 단계를 포함하는 패킷 분류 방법을 제공한다.

[0022] 또한, 본 발명은, 패킷 분류를 위한 룰을 구비한 해싱 테이블을 저장하는 메모리 입력 패킷의 하나 이상의 필드에 대해 필드별 검색을 수행하여 상기 하나 이상의 필드 각각의 필드값과 룰 셋에 저장된 대응하는 필드의 필드값이 서로 일치하는 길이(이하, "일치 길이"라 함)를 판단하고, 상기 하나 이상의 필드 각각에 대한 상기 일치 길이의 조합으로 이루어진 튜플(Tuple) 리스트를 생성하며, 상기 튜플 리스트에서 상기 룰 셋에 존재하는 튜플만을 선별하는 튜플 생성부 및 상기 튜플 생성부에 의해 선별된 튜플 각각에 대해 블룸 필터를 이용하여 필터링을 수행하고 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 하여 상기 해싱 테이블에 접근하여 최우선 룰(BMR: Best Matching Rule)을 검색하는 패킷 분류 제어부를 포함하는 패킷 분류 장치를 제공한다.

발명의 효과

[0023] 본 발명에 의하면, 프리픽스의 길이 정보뿐만 아니라 프리픽스 값 정보를 이용하여 검색 대상 튜플의 수를 줄일 수 있으므로, 메모리 접근 횟수가 감소하여 검색 성능을 향상시킬 수 있다.

도면의 간단한 설명

[0024] 도 1은 종래의 튜플 프루닝 알고리즘을 설명하기 위한 도면,

도 2는 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 방법을 설명하기 위한 도면,

도 3은 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 장치를 도시한 도면,

도 4는 본 발명의 제1 실시예에 따른 패킷 분류 제어부의 구성을 나타내는 도면,

도 5는 본 발명의 제2 실시예에 따른 패킷 분류 제어부의 구성을 나타내는 도면,

도 6은 본 발명의 실시예에서 사용한 32비트 CRC 생성기의 구조를 나타내는 도면,

도 7 및 도 8은 종래의 튜플 프루닝 알고리즘과 본 발명의 실시예에 따른 패킷 분류 방법 간의 메모리 접근 횟수에 대한 성능 비교 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하, 본 발명의 일부 실시예들을 예시적인 도면을 통해 상세하게 설명한다. 각 도면의 구성요소들에 참조부호를 부가함에 있어서, 동일한 구성요소들에 대해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 한다. 또한, 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한설명은 생략한다.

[0026] 또한, 본 발명의 구성 요소를 설명하는 데 있어서, 제 1, 제 2, A, B, (a), (b) 등의 용어를 사용할 수 있다. 이러한 용어는 그 구성 요소를 다른 구성 요소와 구별하기 위한 것일 뿐, 그 용어에 의해 해당 구성 요소의 본질이나 차례 또는 순서 등이 한정되지 않는다. 어떤 구성 요소가 다른 구성요소에 "연결", "결합" 또는 "접속"된다고 기재된 경우, 그 구성 요소는 그 다른 구성요소에 직접적으로 연결되거나 접속될 수 있지만, 각 구성 요

소 사이에 또 다른 구성 요소가 "연결", "결합" 또는 "접속"될 수도 있다고 이해되어야 할 것이다.

[0027] **<발명의 구성>**

[0028] 본 발명의 실시예에 따른 패킷 분류 방법은, (a) 튜플 리스트 생성 단계, (b) 튜플 리스트에서 룰 셋에 존재하는 튜플만을 선별하는 단계, (c) 블룸 필터를 이용한 필터링 단계 및 (d) 룰 검색 단계를 포함하여 구성된다.

[0029] 이하, 도 2를 참조하여 각 단계를 보다 상세히 설명한다.

[0030] 도 2는 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 방법을 설명하기 위한 도면이다.

[0031] 본 발명의 실시예에서는 튜플 생성 및 룰 검색을 위해 근원지 프리픽스 필드와 목적지 프리픽스 필드를 이용하는 경우를 예로 들어 설명한다. 그러나, 이는 설명의 편의를 위한 예시에 불과하며, 본 발명의 사상을 한정하기 위한 것이 아니다. 즉, 튜플 생성 및 룰 검색을 위해 사용하는 필드의 종류나 필드의 수는 적절히 선택할 수 있고, 어떤 종류의 필드를 사용하든 또는 몇 개의 필드를 사용하든 모두 본 발명의 기술적 사상에 포함된다고 할 것이다. 예컨대, 두 개의 필드를 사용하는 경우, 근원지 포트 넘버 필드와 목적지 포트 넘버 필드를 사용하는 것도 가능하며, 세 개의 필드를 사용하는 경우에는, 근원지 프리픽스 필드, 목적지 프리픽스 필드 및 근원지 포트 넘버 필드 등을 사용할 수 있다. 또한, 네 개 또는 다섯 개의 필드를 사용하는 것도 가능하다.

[0032] (a) 튜플 리스트 생성 단계

[0033] 튜플 리스트 생성 단계에서는, 입력 패킷의 근원지 프리픽스 필드와 목적지 프리픽스 필드 각각에 대해 필드별 검색을 수행하여, 필드별 일치 길이의 조합으로 이루어진 튜플 리스트를 생성한다.

[0034] 즉, 입력 패킷의 근원지 프리픽스 필드의 필드값과 룰 셋에 저장된 근원지 프리픽스의 값을 비교하여 값이 서로 일치하는 일치 길이를 판단하고, 입력 패킷의 목적지 프리픽스 필드의 필드값과 룰 셋의 목적지 프리픽스의 값을 서로 비교하여 일치 길이를 판단한다.

[0035] 예컨대, 도 2에 도시된 바와 같이, 입력 패킷의 근원지 프리픽스와 목적지 프리픽스가 (0100, 1110)이고, 룰 셋이 표 1과 같다고 할 때, 입력 패킷의 근원지 프리픽스는 표 1의 룰 셋의 근원지 프리픽스 0*, 01*, 010*과 일치하므로 일치 길이는 1, 2, 3이 된다. 또한, 입력 패킷의 목적지 프리픽스는 룰 셋의 목적지 프리픽스 1*, 111*과 일치하므로, 일치 길이는 1, 3이 된다.

[0036] 따라서, 근원지 프리필스 필드와 목적지 프리픽스 필드의 일치 길이의 조합으로 이루어진 튜플은 (1,1), (1,3), (2,1), (2,3), (3,1), (3,3)이 되고, 이 튜플들이 튜플 리스트로 생성된다.

[0037] 여기서, 각 필드에 대해 수행되는 필드별 검색은 Binary Search on Level의 1차원 검색에 의해 수행될 수 있다.

[0038] (b) 튜플 선별 단계

[0039] 튜플 선별 단계에서는, 튜플 리스트에 속한 튜플 중에서 룰 셋에 존재하는 튜플들만 선별한다.

[0040] 예컨대, 표 1의 룰 셋에는 (1,1), (1,3), (2,2), (3,3)의 튜플들만 존재한다. 따라서, 입력 (0100, 1110)에 대해 생성된 (1,1), (1,3), (2,1), (2,3), (3,1), (3,3) 총 6개의 튜플에서 룰 셋에 존재하지 않는 튜플 (2,1), (2,3), (3,1)을 제거하고, 도 2에 도시된 바와 같이, (1,1), (1,3), (3,3)의 튜플만을 선별하게 된다.

[0041] (c) 블룸 필터를 이용한 필터링 단계

[0042] 블룸 필터는 (b) 단계에서 선별된 튜플들을 필터링하여 필터링 결과가 양성(Positive)인 튜플만을 검색 대상으로 선별한다.

[0043] 블룸 필터는 입력이 특정 집합에 속하는 요소인지 여부를 확인하는 필터로서, 그 특정 집합에 속한 요소들의 비트 벡터를 1로 프로그래밍해 놓은 상태에서, 어떤 입력값이 들어오면 쿼리(Quarry) 과정을 통해 그 입력값에 대응하는 비트 벡터가 1인지 여부를 판단한다. 비트 벡터가 1이면 그 입력값이 특정 집합에 속할 가능성이 있다고 판단하고(이 경우를 "양성(Positive)"라고 함), 비트 벡터가 0이면 그 입력은 특정 집합에 속하지 않는다고 판단하게 된다(이 경우를 "음성(Negative)"라고 함).

[0044] 블룸 필터를 이용한 필터링을 수행하기 위해서는, 먼저 룰 셋에 속하는 각 룰을 블룸 필터에 프로그래밍하는 과정이 필요한데, 이에 대해서는 표 2를 참조하여 이하에서 설명한다.

표 2

[0045]

Rule No.	Source prefix	Destination prefix	Tuple	Concatenated String	Hashing code	BF index (MSB, LSB 4bit 씩)
R0	1*	1*	(1, 1)	11	01101001	6, 9
R1	1*	110*	(1, 3)	1110	10000010	8, 2
R2	01*	00*	(2, 2)	0100	01101100	6, 10
R3	0*	111*	(1, 3)	0111	00000101	0, 5
R4	111*	110*	(3, 3)	111110	00010100	1, 4
R5	010*	110*	(3, 3)	010110	10101111	10, 15

[0046] 표 2는, 표 1의 룰 셋에 대해 블룸 필터를 프로그래밍하는 방법을 설명하기 위한 표이다.

[0047] 먼저 각 룰의 근원지 프리픽스와 목적지 프리픽스를 연결(Concatenation)하여 하나의 스트링을 생성한다. 예컨대, 룰 R0의 경우 근원지 프리픽스가 1*이고 목적지 프리픽스가 1*이므로, R0에 대한 스트링은 11이 된다. R1의 경우에는, 근원지 프리픽스와 목적지 프리픽스가 각각 1*, 110*이므로 이들을 서로 연결하여 1110의 스트링을 생성하게 된다. 여기서 두 개의 프리픽스를 연결하여 사용하였으나 이에 한정된 것은 아니다.

[0048] 이렇게 하여 모든 룰에 대해 스트링을 만들면, 이를 해시함수의 입력으로 사용하여 해싱 코드를 생성하게 된다. 본 발명의 실시예에서는 해시함수로서CRC 생성기(Cyclic Redundancy Check Generator)를 사용하였다.

[0049] CRC 생성기는 다양한 길이의 스트링을 입력으로 받아 고정 길이의 해싱 코드를 생성할 수 있다. 따라서 CRC 생성기에서 생성된 고정 길이의 해싱 코드로부터 원하는 길이, 원하는 개수만큼의 해싱 인덱스를 추출할 수 있게 된다.

[0050] CRC 생성기의 동작을 설명하기 위해, 32비트 CRC 생성기의 구조를 도시한 도 6을 예로 들어 설명하면, CRC 생성기 내의 모든 플립플롭(Flip-flop)은 0으로 초기화되어 있으며, 입력값이 한 비트식 들어오면 플립플롭에 저장된 값과 입력된 비트값에 대해 XOR 연산을 수행하게 되고, 마지막 비트까지 들어오면 CRC 생성기의 동작이 멈추면서 각각의 플립플롭에 저장된 값들이 CRC 코드로 출력됨으로써, 32비트의 고정 길이 해싱 코드를 얻을 수 있다.

[0051] 표 2에서 "Hashing Code"항목은 각각의 룰에 대해 두 프리픽스를 연결하여 생성한 스트링을 8비트 CRC 생성기에 입력하여 얻은 CRC 코드이다.

[0052] 해싱 코드가 얻어지면, 이로부터 블룸 필터를 프로그래밍하기 위한 해싱 인덱스를 생성하게 된다. 예컨대, R0에 대해 CRC 생성기로부터 얻어진 해싱 코드 "01101001"의 상위 4비트와 하위 4비트를 추출하면, "0110"(6)과 "1001"(9)를 얻을 수 있고, 따라서 블룸 필터의 6번째 엔트리와 9번째 엔트리의 비트 벡터를 1로 프로그램하게 된다.

[0053] 표 2의 "BF index"는 위와 같은 방법으로 생성한 해싱 인덱스를 나타낸 것이고, 이 해싱 인덱스에 의해 프로그래밍된 블룸 필터가 도 2에 도시되어 있다.

[0054] 이렇게 프로그래밍 된 블룸 필터를 이용하여 (b) 단계에서 선별된 튜플에 대해 필터링을 수행하게 된다.

[0055] 입력 (0100, 1110)과 관련하여, (b) 단계에서 선별된 튜플 (1,1), (1,3), (3,3)에 대한 필터링 과정을 예로 들어 설명하면, 먼저 각각의 튜플에 대해 해싱 인덱스를 생성하게 된다. 예컨대, 튜플 (1,1)은 근원지 프리픽스와 목적지 프리픽스의 일치 길이가 모두 1이라는 것을 나타내므로, 입력 (0100, 1110)의 근원지 프리픽스와 목적지 프리픽스에서 각각 상위 1비트씩을 추출하여 해시함수의 입력으로 사용할 스트링 "01"을 생성한다. 튜플 (1,3)의 경우에는 근원지 프리픽스와 목적지 프리픽스의 일치 길이가 각각 1, 3이므로 입력의 근원지 프리픽스에서 상위 1비트, 목적지 프리픽스에서 상위 3비트를 추출하여 스트링 "0111"을 생성한다. 튜플 (3,3)에 대해서도 동일한 과정을 수행하면 스트링 "010111"을 생성하게 된다.

[0056] 튜플 (1,1), (1,3) 및 (3,3) 각각에 해당하는 스트링 "01", "0111" 및 "010111"을 8비트의 CRC 생성기에 입력하면, "10110001", "00000101", "00011110"인 해싱 코드를 얻게 된다.

[0057] 튜플 (1,1)의 스트링 "01"에 의해 생성된 해싱 코드 "10110001"에서 상위 4비트와 하위 4비트를 각각 추출하면"1011"(11)과 "0001"(1)의 해싱 인덱스를 얻을 수 있다. 따라서 이 해싱 인덱스를 사용하여 블룸 필터의 1번

째 엔트리와 11번째 엔트리에 접근하여, 각 엔트리의 비트 벡터를 확인하게 되는데, 1번째 엔트리의 비트 벡터는 1인 반면, 11번째 엔트리의 비트 벡터는 0이므로, 튜플 (1,1)은 룰 셋에 포함되지 않는 것으로 판단할 수 있다. 따라서 튜플 (1,1)은 검색 대상에서 제거된다.

[0058] 다음으로, 튜플 (1,3)의 스트링 "0111"에 의해 생성된 해싱 코드 "00000101"에서, 위와 마찬가지 방법으로, 상위 4비트와 하위 4비트를 각각 추출하면 "0000"(0)과 "0101"(5)의 해싱 인덱스를 얻을 수 있다. 이 해싱 인덱스를 이용하여 블룸 필터의 0번째 엔트리와 5번째 엔트리에 접근하여 비트 벡터를 확인하면, 0번째 엔트리와 5번째 엔트리 모두 비트 벡터가 1이므로, 튜플 (1,3)이 나타내는 (0*,111*)이 룰 셋에 존재할 가능성이 있게 된다. 따라서 튜플 (1,3)은 검색 대상 튜플이 된다.

[0059] 마지막으로, 튜플 (3,3)에 대해 위와 동일한 방법으로 필터링을 하면, 해싱 코드 "00011110"의 상위 4비트 해싱 인덱스가 나타내는 1번째 엔트리의 비트 벡터는 1이지만 하위 4비트 해싱 인덱스가 나타내는 14번째 엔트리의 비트 벡터는 0이므로, 룰 셋에 포함되지 않는 것으로 판단할 수 있고, 튜플(3,3)은 검색 대상에서 제거된다.

[0060] (d) 룰 검색 단계

[0061] 룰 검색 단계는 블룸 필터의 필터링 결과가 양성인 튜플만을 검색대상으로 하여 입력 패킷과 가장 일치하는 최우선 룰(BMR: Best Matching Rule)을 결정하는 단계이다.

[0062] 표 2를 참조하여 보다 상세히 설명하면, 먼저 블룸 필터를 프로그래밍하기 위해 사용했던 해싱 인덱스를 이용하여 해싱 테이블의 엔트리에 해당 룰이 저장하게 된다. 여기서, 블룸 필터를 프로그래밍하기 위해 사용했던 두 개의 해싱 인덱스를 모두 사용하는 것도 가능하나, 본 실시예에서는 그 중 하나의 해싱 인덱스만을 사용하였다. 예컨대, 해싱 코드에서 상위 4비트를 추출하여 생성한 해싱 인덱스를 이용하여 룰을 저장한다고 가정하면, 해싱 테이블의 6번째 엔트리에는 룰 R0가 저장되고, 8번째 엔트리에는 R1이 저장되며, 룰 R2~R5 모두 첫번째 해싱 인덱스가 가리키는 각 엔트리에 저장되어 있다.

[0063] 이전 단계인 (c) 단계에서 튜플 (1,3)만이 검색 대상으로 선별되었다. 따라서 튜플 (1,3)에 대한 해싱 인덱스인 "0000"(0)을 이용하여 검색을 수행하게 되는데, 해싱 테이블의 0번째의 엔트리에 저장된 룰은 R3이고, R3는 입력 패킷 (0*,111*)과 일치하므로, R3가 최우선 룰로서 결정된다.

[0064] 이상에서 설명한 (d) 룰 검색 단계에서는 블룸 필터에 접근하기 위한 해싱 인덱스와 해싱 테이블에의 접근을 위한 해싱 인덱스를 동일하게 사용하였다. 그러나 본 발명은 이에 한정되는 것은 아니며, 블룸 필터용 해싱 인덱스와 해싱 테이블용 해싱 인덱스를 다른 값으로 사용하여도 무방하다.

[0065] 본 발명에서 사용한 블룸 필터는 특정 값이 특정 셋에 존재하지 않는데 존재한다고 판단할 가능성은 있어도(이를 "거짓 양성(False Positive)"라 함), 특정 값이 특정 셋에 존재하는데 존재하지 않는다고 판단하는 경우는 없다. 따라서, 본 발명이 제안하는 블룸 필터를 이용한 패킷 분류 방법을 사용하면, 불필요한 튜플 검색으로 인한 검색 성능 저하를 막을 수 있다.

[0066] 표 1의 예시 룰 셋에 대한 결과에서 알 수 있는 바와 같이, 기존의 튜플 프루닝 알고리즘은 튜플 (1,1), (1,3) 및 (3,3)을 모두 검색 대상으로 하고 있으나, 본 발명에서는 블룸 필터를 이용함으로써 튜플 (1,3)만을 검색 대상으로 하고 있다. 따라서 룰 검색을 위한 외부 메모리 접근 횟수를 줄일 수 있어 패킷 분류를 위한 검색 성능을 향상시킬 수 있는 것이다.

[0067] 한편, 전술한 실시예에서는 각각의 튜플에 대해 두 개의 해싱 인덱스를 생성하는 것으로 설명하였으나, 이에 한정되는 것은 아니며, 해싱 인덱스를 하나만 사용할 수도 있고 세 개 이상을 사용할 수도 있다. 즉, 해싱 인덱스의 개수와는 상관없이 전술한 패킷 분류 방법과 동일 내지 균등한 방법을 사용하는 한, 본 발명의 기술적 사상에 포함된다고 할 것이다.

[0068] 아래의 표 3은 3개의 해싱 인덱스를 사용한 경우의 예이다.

표 3

[0069]

Rule No.	Source prefix	Destination prefix	Tuple	Concatenated String	CRC 코드	BF index
R0	1*	1*	(1, 1)	11	01101001	6, 10, 9
R1	1*	110*	(1, 3)	1110	10000010	8, 0, 2
R2	01*	00*	(2, 2)	0100	01101100	6, 11, 12

R3	0*	111*	(1, 3)	0111	00000101	0, 1, 5
R4	111*	110*	(3, 3)	111110	00010100	1, 5, 4
R5	010*	110*	(3, 3)	010110	10101111	10, 11, 15

[0070] 블룸필터의 크기를 M, 블룸필터에 저장할 요소의 수를N, 프로그래밍에 사용할 해시함수의 수를 k로 정의할 때, 최적화된 해시함수의 수는 수학식 1과 같음이 알려져 있다(Sarang Dharmapurikar, Praveen Krishamurthy, and David E. Taylor, "Longest prefix matching using bloom filters," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 397-409, April 2006).

수학식 1

[0071] $$k = \frac{M}{N} ln2$$

[0072] 따라서, 블룸필터의 크기(M)는 서로 다른 튜픔의 개수(N)인 4의 4배를 사용해 16 bit를 가정하면, 블룸필터를 프로그래밍하기 위한 최적의 해시함수의 수는 위 수학식 1에 따라 3개로 결정된다. 표 3은 8비트 CRC 생성기의 출력값 중, [7:4], [5:2], [3:0]비트를 사용하여 생성한 해싱 인덱스를 나타내고 있다. 예컨대, R0의 경우, 8비트 CRC 생성기의 출력값인 CRC 코드 "01101001" 중, [7:4]비트는 "0110", [5:2]비트는 "1010", [3:0]비트는 "1001"이므로, 해싱 인덱스는 6, 10, 9가 된다.

[0073] 이하에서는, 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 방법을 구현한 패킷 분류 장치를 설명한다.

[0074] 도 3은 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 장치를 도시한 도면이다.

[0075] 본 발명의 실시예에 따른 블룸 필터를 이용한 패킷 분류 장치는 튜플 생성부(310), 패킷 분류 제어부(320) 및 메모리부(330)를 포함하여 구성된다.

[0076] 튜플 생성부(310)는 기본적으로 전술한 패킷 분류 방법 중 (a) 튜플 리스트생성 단계 및 (b) 튜플 선별 단계를 수행하는 모듈이다. 즉, 입력 패킷의 근원지 프리픽스 필드와 목적지 프리픽스 필드 각각에 대해 필드별 검색을 수행하여, 룰 셋의 대응하는 필드의 필드값과의 일치 길이를 판단하여 각 필드의 일치 길이의 조합으로 이루어진 튜플 리스트를 생성하고, 생성된 튜플 리스트에서 룰 셋에 존재하는 튜플만을 선별한다.

[0077] 여기서, 튜플 생성부(310)가 수행하는 필드별 검색은 1차원 검색으로서 가장 성능이 우수하다고 알려진 Binary Search on Level을 이용할 수 있으나, 이에 한정된 것은 아니다.

[0078] 패킷 분류 제어부(320)는 기본적으로 전술한 패킷 분류 방법 중 (c) 블룸 필터를 이용한 필터링 단계 및 (d) 룰 검색 단계를 수행하는 모듈이다. 즉, 튜플 생성부(310)에서 선별된 튜플 각각에 대해 블룸 필터를 이용한 필터링을 수행하고 필터링 결과가 양성인 튜플만을 검색 대상으로 하여 메모리부(330)에 저장된 해싱 테이블에 접근함으로써 해싱 테이블에 저장된 룰 중 최우선 룰을 검색 및 결정한다. 패킷 분류 제어부(320)의 보다 상세한 구성은 도 4 및 도 5를 참조하여 후술한다.

[0079] 메모리부(330)에는 패킷 분류를 위한 룰을 구비한 해싱 테이블이 저장되어 있다.

[0080] 도 4는 본 발명의 제1 실시예에 따른 패킷 분류 제어부의 구성을 나타내는 도면이다.

[0081] 본 발명의 제1 실시예에 따른 패킷 분류 제어부(320)는 해싱부(410), 블룸 필터(420) 및 룰 검색부(430)를 포함한다.

[0082] 해싱부(410)는 튜플 생성부(310)에서 선별된 튜플 각각에 대해 해싱 인덱스를 생성하는데, CRC 생성기(412) 및 해싱 인덱스 생성부(414)를 포함하여 구성될 수 있다.

[0083] CRC 생성기(412)는 튜플 생성부(310)에서 선별된 튜플이 나타내는 일치 길이만큼의 상위 비트를 입력 패킷의 근원지 프리픽스 및 목적지 프리픽스에서 추출하여 서로 연결(Concatenation)함으로써 생성된 스트링을 입력 받아 고정 길이의 해싱 코드를 생성한다.

[0084] 해싱 인덱스 생성부(414)는 CRC 생성기(412)에서 생성된 고정 길이 해싱 코드로부터 해싱 인덱스를 생성한다.

해싱 인덱스 생성부(414)는 고정 길이 해싱 코드로부터 일정 비트(예컨대, 상위 몇 비트 또는/및 하위 몇 비트)를 추출하여 해싱 인덱스를 생성할 수 있으며, 하나 또는 다수의 해싱 인덱스를 생성할 수 있다.

[0085] 블룸 필터(420)는 해싱 인덱스 생성부(414)에서 생성된 해싱 인덱스가 가리키는 엔트리의 비트 벡터를 확인하여 비트 벡터가 양성인 튜플만을 검색 대상으로 선별한다.

[0086] 룰 검색부(430)는 블룸 필터(420)에서 검색 대상으로 선별된 튜플의 해싱 인덱스가 가리키는 해싱 테이블의 엔트리에 접근하여 룰 검색을 수행하고, 입력 패킷과 일치하는 룰 중 최우선 룰을 결정한다.

[0087] 도 5는 본 발명의 제2 실시예에 따른 패킷 분류 제어부의 구성을 나타내는 도면이다.

[0088] 본 발명의 제2 실시예에 따른 패킷 분류 제어부(320)의 해싱부(510)은 블룸 필터용 해싱 인덱스와 해싱 테이블용 해싱 인덱스를 별도로 생성한다는 점에서 제1 실시예와 다르다.

[0089] 즉, 해싱 인덱스 생성부(514)는 CRC 생성기(512)에서 생성된 고정 길이 해싱 코드를 이용하여 블룸 필터용 해싱 인덱스와 해싱 테이블용 해싱 인덱스를 개별적으로 생성한다. 여기서, 블룸 필터용 해싱 인덱스는 하나 또는 복수 개일 수 있고 해싱 테이블용 해싱 인덱스는 한 개이다.

[0090] 블룸 필터(520)는 블룸 필터용 해싱 인덱스가 가리키는 엔트리의 비트 벡터를 확인하여 비트 벡터가 양성인 튜플만을 검색 대상으로 선별하고, 룰 검색부(530)는 검색 대상으로 선별된 튜플의 해싱 테이블용 해싱 인덱스가 가리키는 해싱 테이블 엔트리에 접근하여, 입력 패킷과 일치하는 룰 중 최우선 룰을 검색 및 결정한다.

[0091] 이상에서 설명한, 본 발명의 일 실시예에서는 해싱 인덱스를 생성하기 위한 해시함수(CRC 생성기)의 입력값으로, 튜플이 나타내는 일치 길이만큼의 상위 비트를 근원지 프리픽스와 목적지 프리픽스에서 각각 추출하여 서로 연결한 스트링을 사용하였다. 그러나, 이는 하나의 실시예일 뿐, 본 발명의 사상이 이에 한정되는 것은 아니며, 해싱 인덱스를 생성하기 위한 해시함수의 입력값으로 다양한 입력값들이 사용될 수 있다.

[0092] 예컨대, 블룸 필터에 의한 필터링시에 발생하는 거짓 양성(False Positive)의 수를 줄이기 위한 방법으로서, 튜플이 나타내는 일치 길이만큼의 상위 비트를 근원지 프리픽스와 목적지 프리픽스에서 각각 추출하고, 추출된 값에 소정의 비트 패턴을 부가하여 고정길이 값으로 패딩(Padding)한 후, 패딩된 값들을 서로 연결한 값을 해시함수의 입력값으로 사용할 수 있다.

[0093] 예를 들어, 근원지 프리픽스에서 추출된 값을 "A"라고 할 때, A 뒤에 3비트의 "101"을 트레일러로 붙인 "A101" 패턴을 반복적으로 사용하여 32비트까지 확장하고, 목적지 프리픽스에서 추출한 B에 대해서도 동일한 방식으로 32비트까지 확장한 후, 이들을 서로 연결한 64비트의 값을 해시함수의 입력값으로 사용할 수 있다.

[0094] 입력 패킷이 룰과 서로 일치하지 않더라도 해시함수의 입력값이 서로 같으면, 동일한 해싱 인덱스가 생성되므로, 블룸필터에 의한 필터링 결과가 거짓 양성이 되는 경우가 발생하게 된다.

[0095] 예를 들어, 저장된 룰이 (1*, *)이라고 가정하면, 튜플이 나타내는 일치 길이만큼의 상위 비트를 근원지 프리픽스와 목적지 프리픽스에서 각각 추출하여 서로 연결한 스트링을 해시함수의 입력값으로 사용하는 실시예에 기반할 때, 해시함수의 입력 스트링은 '1' 이 된다. 만약 입력이 (1100, 1000)이고 필드별 검색 결과로 튜플 (0,1)을 얻게 된다면 해당 입력의 해싱 인덱스를 얻기 위한 해시함수의 입력 스트링은 '1' 이 되어앞서 저장된 룰과 동일한 해싱 인덱스를 얻게 된다. 하지만 실제로 저장된 룰과 입력은 튜플이 일치하지 않으므로 해당 입력의 튜플 (0,1)은 거짓 양성임을 알 수 있다. 또 다른 예로서, 저장된 룰이 (1*, 0*)이라면 이 룰의 해시함수 입력 스트링은 '10' 이 되고, 이전 예시와 동일한 필드별 검색 결과로 튜플 (0,2)를 얻게 되는 경우에는 해당 입력의 해시함수 입력 스트링 역시 '10' 이 되어저장된 룰과 동일한 해싱 인덱스를 얻게 된다. 이 경우 해당 입력이 튜플과 값 모두 저장된 룰과 다름에도 불구하고 동일한 해싱 인덱스를 얻게 되어 거짓 양성을 일으키게 된다.

[0096] 패딩을 사용한 방법은 서로 다른 튜플에 대해 서로 다른 해시함수의 입력값이 생성될 확률을 높여 거짓 양성이 발생하는 빈도를 낮추게 된다. 따라서. 최우선 룰을 검색하기 위한 메모리 접근 횟수를 더욱 줄일 수 있게 된다.

[0097] **<본 발명의 성능 평가>**

[0098] 실제 백-본 라우터에서 사용되는 룰 셋과 비슷한 특성을 갖는다고 알려진 Classbench를 사용하여 약 5000개의 룰을 갖는 ACL, IPC, FW 셋을 가지고 본 발명의 실시예에서 제안하는 구조의 성능을 평가하였다.

[0099] 표 4는 실험에 사용한 각 룰 셋의 특성에 관한 것으로, 룰 셋에 포함된 룰의 개수, 서로 다른 튜플의 개수 및 서로 다른 필드 값의 조합의 개수를 나타내었다. ACL 5K 룰 셋은 다른 룰 셋에 비해 적은 수의 튜플 종류가 포함되어 서로 다른 튜플의 개수가 적지만, 하나의 튜플 내에 서로 다른 프리픽스 값의 조합을 갖는 룰이 많아 서로 다른 필드 값의 조합의 수가 크다는 특징을 갖는다. 반면 FW 5K는 튜플 종류가 다양한 반면 프리픽스 조합이 같은 룰이 많기 때문에 서로 다른 필드 값의 조합의 수는 다른 룰 셋에 비해 적은 특징을 보인다.

표 4

[0100]

Rule set	No. of rules	No. of distinct tuples	No. of distinct combinations in prefix values
ACL 5K	4660	102	2453
IPC 5K	4486	680	2933
FW 5K	4351	579	1244

[0101] 표 5는 튜플 블룸 필터를 적용한 튜플 프루닝 알고리즘의 성능을 나태낸 것으로, 튜플이 나타내는 일치 길이만큼의 상위 비트를 근원지 프리픽스와 목적지 프리픽스에서 각각 추출하여 서로 연결한 스트링을 해시함수의 입력값으로 사용한 실시예에 대한 성능 평가 결과이다.

표 5

[0102]

Rule Set	No. of input	T_{Pruned}	Tuple Bloom Filter					Tuple Pruning Rate by Tuple BF(%)	True match (T_{true_match})
			M	No. of Indices	T_{Neg} Negative	Positive(T_{BF_Pruned})			
						True ($T_{BF_2\text{-}field}$)	False (T_{BF_false})		
ACL 5K	13980	6.87	N	1	1.79	3.87	1.22	26.01	3.29
			$2N$	1	2.36		0.65	34.28	
			$4N$	3	2.90		0.11	42.12	
			$8N$	6	2.96		0.04	43.11	
			$16N$	11	2.96		0.04	43.12	
			$32N$	22	2.94		0.07	42.73	
IPC 5K	20610	15.29	N	1	3.77	5.41	6.11	24.62	4.00
			$2N$	1	4.95		4.93	32.35	
			$4N$	3	5.78		4.10	37.81	
			$8N$	6	5.95		3.93	38.89	
			$16N$	11	5.95		3.93	38.90	
			$32N$	22	5.85		4.03	38.26	
FW 5K	13054	8.37	N	1	0.54	6.14	1.70	6.39	5.82
			$2N$	1	0.92		1.31	11.03	
			$4N$	3	1.08		1.15	12.92	
			$8N$	6	1.10		1.14	13.08	
			$16N$	11	1.05		1.18	12.50	
			$32N$	22	1.10		1.14	13.08	

[0103] 이 실험에서 서로 다른 튜플의 개수를 N 이라고 할 때 튜플 블룸 필터의 크기 단위 N' 는 $2^{\lceil \log_2 N \rceil}$으로 하였다. 해시함수는 CRC-64 생성기를 사용하였고, 해싱 인덱스의 수(k)는 블룸 필터의 크기(M)에 따라 증가시켰다. 필드별 1차원 검색은 가장 성능이 우수한 것으로 알려진 Binary Search on Level(BSL)을 사용하였다.

[0104] 표 5에서 T_{Pruned}는 기존의 튜플 프루닝 알고리즘만을 적용하였을 때 남는 튜플의 평균 개수를 나타낸다. T_{BF_Pruned}는 튜플 블룸 필터의 크기에 따른 메모리 접근 성능을 나타내는데, 튜플 블룸 필터가 걸러내는 평균 음성 튜플

의 수 (T_{Neg}), 양성 결과 중에서 저장된 룰과 튜플을 만드는데 사용한 두 필드가 일치하는 평균 참된 양성 튜플의 수 ($T_{BF_2\text{-}field}$), 그리고 튜플 블룸 필터를 통과하였으나 저장된 룰과 일치하지 않는 평균 거짓 양성 튜플의 수 (T_{BF_false})를 나타낸다. 튜플 블룸 필터가 기존의 튜플 프루닝 알고리즘에서 추가적으로 걸러내는 튜플을 퍼센트(Tuple Pruning Rate by Tuple BF), 최종적으로 하나의 입력에 대한 여러 튜플 검색 결과 중에서 저장된 룰과 완전히 일치하는 평균 튜플의 수 (T_{true_match})를 나타내었다.

[0105] 표 5의 데이터를 통해, 제안하는 튜플 블룸 필터가 튜플 프루닝 알고리즘의 결과로써 남은 튜플(T_{Pruned}) 중에서 룰 셋에 존재하지 않는 튜플을 추가적으로 제거함으로써 실제 메모리에 접근하는 튜플의 수(T_{BF_Pruned})가 감소되었음을 알 수 있다. 또한 블룸 필터의 크기(M)가 커질수록 음성 결과의 비율은 증가되고, 따라서 튜플 블룸 필터를 통과하는 튜플의 개수(T_{BF_Pruned})는 점차 감소함을 알 수 있다. 블룸 필터의 특성인 거짓 양성 역시 블룸 필터의 크기(M)에 의존하는데, 블룸 필터의 크기가 커질수록 거짓 양성은 감소하는 경향을 보인다.

[0106] 여기서 참된 양성 ($T_{BF_2\text{-}field}$)이란 블룸 필터를 통과한 양성 결과 중에서 해시 테이블에 접근하여 저장된 룰과 비교한 결과 입력의 근원지와 목적지 프리픽스의 값이 일치한 경우이다. 평균 참된 양성 튜플의 수는 블룸 필터의 크기와 관계 없이 일정하지만 평균 거짓 양성 튜플의 수는 블룸 필터의 크기에 의존하여 변화됨을 알 수 있다.

[0107] 표 5에서 튜플 블룸 필터를 적용한 튜플 프루닝 알고리즘은 추가적인 튜플 제거 기능을 수행하여 외부 메모리 접근 횟수가 감소함을 알 수 있다. 하지만 블룸 필터의 특성상 튜플 블룸 필터의 크기가 증가됨에 따라 거짓 양성의 수는 감소되어야 하지만, 특정 크기에서부터는 거짓 양성이 더 이상 감소되지 못하고 수렴되는 것을 볼 수 있다. 특히 IPC 5K나 FW 5K 룰 셋의 경우에는 블룸 필터의 크기가 커짐에 따른 거짓 양성 튜플 수의 감소폭이 매우 적음을 알 수 있다.

[0108] 이러한 거짓 양성의 수를 줄이기 위해, 전술한 바와 같이, 튜플 나타내는 일치 길이만큼의 상위 비트를 근원지 프리픽스와 목적지 프리픽스에서 각각 추출하고, 추출된 값에 소정의 비트 패턴을 부가하여 고정길이 값으로 패딩(Padding)한 후, 패딩된 값들을 서로 연결한 값을 해시함수의 입력값으로 사용할 수 있고, 그 결과가 표 6에 나타나 있다.

표 6

[0109]

Rule Set	No. of input	T_{Pruned}	Tuple Bloom Filter					Pruned Tuple Ratio by Tuple BF(%)	True match (T_{true_match})
			M	No. of Indices	Negative	Positive(T_{BF_Pruned})			
						True ($T_{BF_2\text{-}field}$)	False (T_{BF_false})		
ACL 5K	13980	6.875	*N*	1	1.707	3.868	1.301	24.82	3.29
			2*N*	1	2.313		0.693	33.65	
			4*N*	3	2.918		0.089	42.44	
			8*N*	6	3.007		0.000	43.74	
			16*N*	11	3.007		0.000	43.74	
			32*N*	22	3.007		0.000	43.74	
IPC 5K	20610	15.290	*N*	1	4.979	5.411	4.901	32.56	4.00
			2*N*	1	5.811		4.070	38.00	
			4*N*	3	9.013		0.867	58.95	
			8*N*	6	9.775		0.105	63.93	
			16*N*	11	9.778		0.102	63.95	
			32*N*	22	9.778		0.102	63.95	
FW 5K	13054	8.374	*N*	1	0.618	6.142	1.613	7.39	5.82

[0110] 표 6에서는, 각 필드의 프리픽스를 고정길이 32-bit로 확장하는 패딩을 사용하였다. 이 때 패딩의 길이와 값은 임의로 결정할 수 있으며, 본 실험에서는 각 필드의 프리픽스 끝에 3-bit의 '101'을 트레일러로 붙인 패턴, 즉 {prefix, 101} 패턴을 반복적으로 32 비트에 이르기까지 패딩하였다.

[0111] 표 6에서 보는 바와 같이, 해시함수의 입력을 모두 동일하게 64-bit로 확장하여 얻은 색인으로부터 블룸 필터를 프로그래밍하고 검색을 진행한 결과, 모든 룰 셋이 블룸 필터의 크기가 $4N'$ 일 때 이미 거짓 양성 튜플의 수가 1개 이하가 되었음을 알 수 있다.

[0112] 도 7 및 도 8은 종래의 튜플 프루닝 알고리즘과 본 발명의 실시예에 따른 패킷 분류 방법 간의 메모리 접근 횟수에 대한 성능 비교 그래프이다.

[0113] 도 7은 본 발명의 실시예 중 패딩을 사용하지 않은 경우에 대한 결과를 종래의 튜플 프루닝 알고리즘과 비교한 것이다.

[0114] 각 알고리즘의 성능 평가를 위해서 사용한 해시함수인 64비트 CRC 함수는 튜플의 종류에 관계 없이 모두 동일한 길이의 해싱 코드를 얻을 수 있고, 그로부터 특정 길이의 해싱 인덱스를 뽑아낼 수 있다는 장점이 있지만 완전 해시함수가 아니기 때문에 서로 다른 입력에 대해서 동일한 해싱 인덱스가 발생하는 충돌이 발생할 수 있다. 성능 평가 실험에서는 해싱 인덱스의 충돌이 발생하였을 경우에 linked-list로 연결하여 저장하였고, 검색 시 linked-list에 의하여 저장된 엔트리들에 대해서는 선형 검색을 진행하였다. 도 7을 통해 제안하는 구조가 기존의 튜플 프루닝 알고리즘에서 프리픽스 길이 정보의 조합으로 된 튜플로만 검색 영역을 제한하는 것에 더 나아가 적은 크기(2~4Kbyte)의 on-chip memory인 블룸 필터를 사용하여 프리픽스 값 정보에 대해 한 번 더 검색 영역을 제한함으로써 더 빠른 패킷 분류를 가능하게 하는 효과적인 구조임을 알 수 있다.

[0115] 도 8은 본 발명의 실시예 중 패딩을 사용한 경우의 결과를 종래의 튜플 프루닝 알고리즘과 비교한 것이다.

[0116] 여기서, 도 8(a) 해시함수가 충돌이 발생하지 않는 완전한 해시함수라고 가정한 경우에 대한 비교 그래프인데, 도 8(a)에서 보는 바와 같이, 최대 40%까지 외부 메모리 접근 횟수가 감소함을 알 수 있다.

[0117] 한편, CRC 함수는, 위에서 설명한 바와 같이, 완전한 해시함수가 아니므로, 해싱 인덱스의 충돌이 발생하였을 경우에 linked-list로 연결하여 저장하고, 검색 시 linked-list에 의하여 저장된 엔트리들에 대해서는 선형 검색을 진행하였는데, 그 결과가 도 8(b)에 도시되어 있다.

[0118] 도 8(b)를 참조하면, 완전한 해시함수를 가정한 도 8(a)보다 성능 향상이 더욱 증가함을 알 수 있다.

[0119] 또한, 패딩을 사용하지 않은 도 7과 패딩을 사용한 도 8(b)를 비교하여 볼 때, 패딩을 사용한 경우의 성능 향상이 더욱 증가함을 알 수 있다.

[0120] 이상에서, 본 발명의 실시예를 구성하는 모든 구성 요소들이 하나로 결합하거나 결합하여 동작하는 것으로 설명되었다고 해서, 본 발명이 반드시 이러한 실시예에 한정되는 것은 아니다. 즉, 본 발명의 목적 범위 안에서라면, 그 모든 구성 요소들이 하나 이상으로 선택적으로 결합하여 동작할 수도 있다. 또한, 그 모든 구성 요소들이 각각 하나의 독립적인 하드웨어로 구현될 수 있지만, 각 구성 요소들의 그 일부 또는 전부가 선택적으로 조합되어 하나 또는 복수 개의 하드웨어에서 조합된 일부 또는 전부의 기능을 수행하는 프로그램 모듈을 갖는 컴퓨터 프로그램으로서 구현될 수도 있다. 그 컴퓨터 프로그램을 구성하는 코드들 및 코드 세그먼트들은 본 발명의 기술 분야의 당업자에 의해 용이하게 추론될 수 있을 것이다. 이러한 컴퓨터 프로그램은 컴퓨터가 읽을 수 있는 저장매체(Computer Readable Media)에 저장되어 컴퓨터에 의하여 읽혀지고 실행됨으로써, 본 발명의 실시예를 구현할 수 있다. 컴퓨터 프로그램의 저장매체로서는 자기 기록매체, 광 기록매체, 캐리어 웨이브 매체 등이 포함될 수 있다.

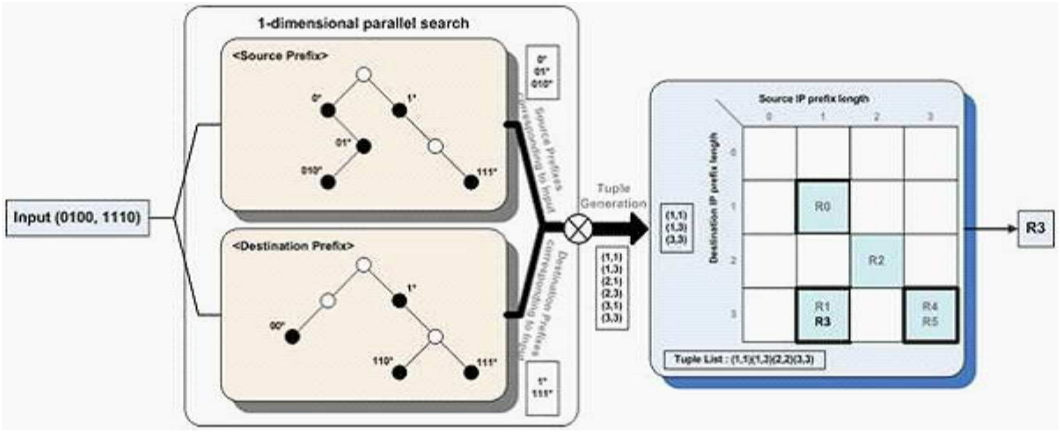
[0121] 또한, 이상에서 기재된 "포함하다", "구성하다" 또는 "가지다" 등의 용어는, 특별히 반대되는 기재가 없는 한, 해당 구성 요소가 내재할 수 있음을 의미하는 것이므로, 다른 구성 요소를 제외하는 것이 아니라 다른 구성 요소를 더 포함할 수 있는 것으로 해석되어야 한다. 기술적이거나 과학적인 용어를 포함한 모든 용어들은, 다르게 정의되지 않는 한, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미가 있다. 사전에 정의된 용어와 같이 일반적으로 사용되는 용어들은 관련 기술의 문맥상의 의미와 일치하는 것으로 해석되어야 하며, 본 발명에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0122] 이상의 설명은 본 발명의 기술 사상을 예시적으로 설명한 것에 불과한 것으로서, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 수정 및 변형이 가능할 것이다. 따라서, 본 발명에 개시된 실시예들은 본 발명의 기술 사상을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예에 의하여 본 발명의 기술 사상의 범위가 한정되는 것은 아니다. 본 발명의 보호 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술 사상은 본 발명의

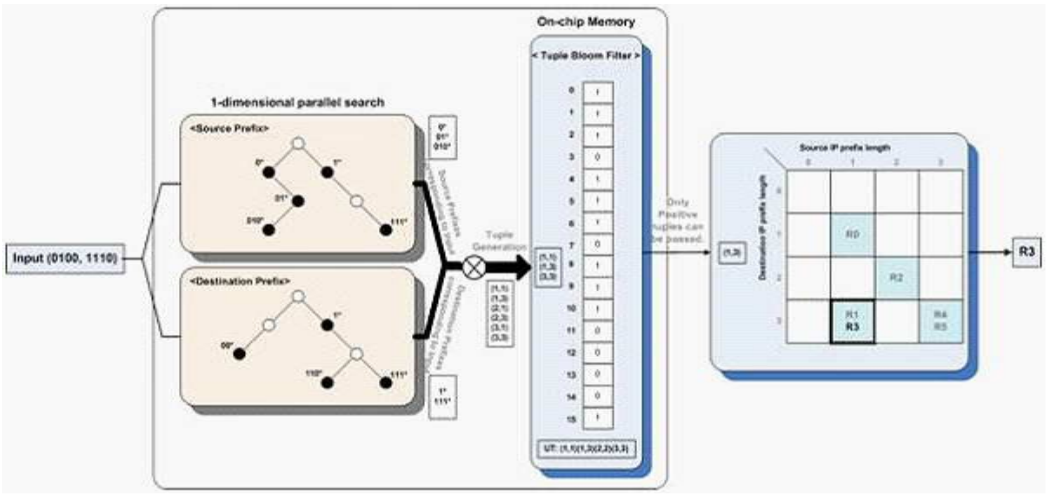
권리범위에 포함되는 것으로 해석되어야 할 것이다.

도면

도면1



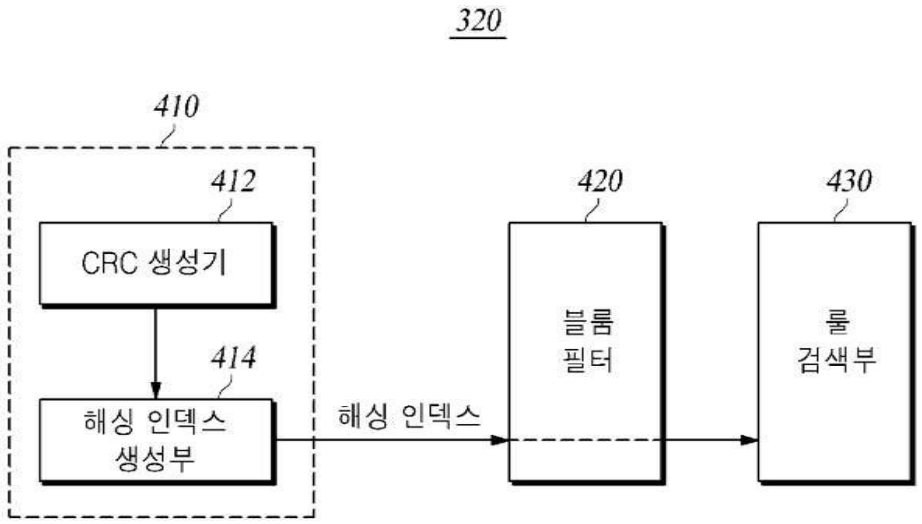
도면2



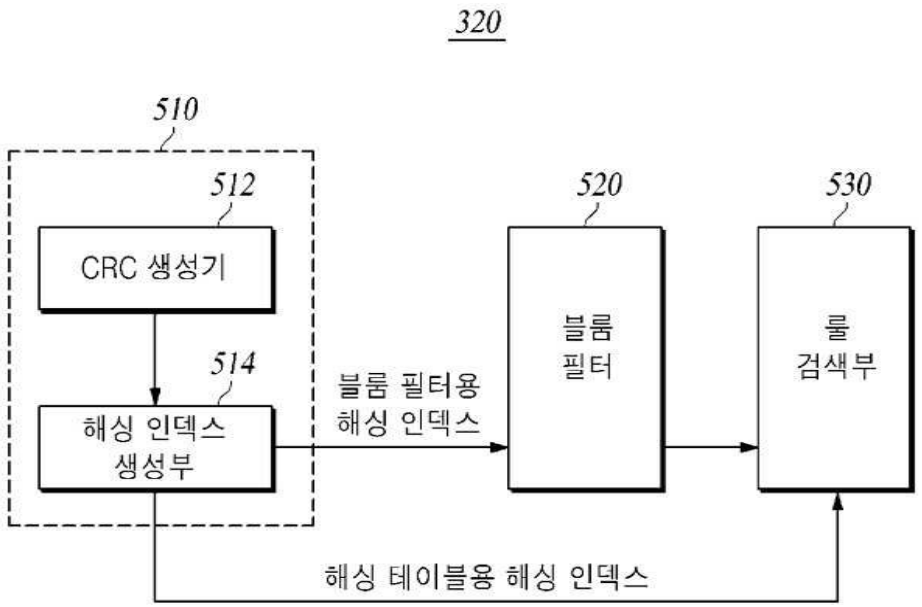
도면3



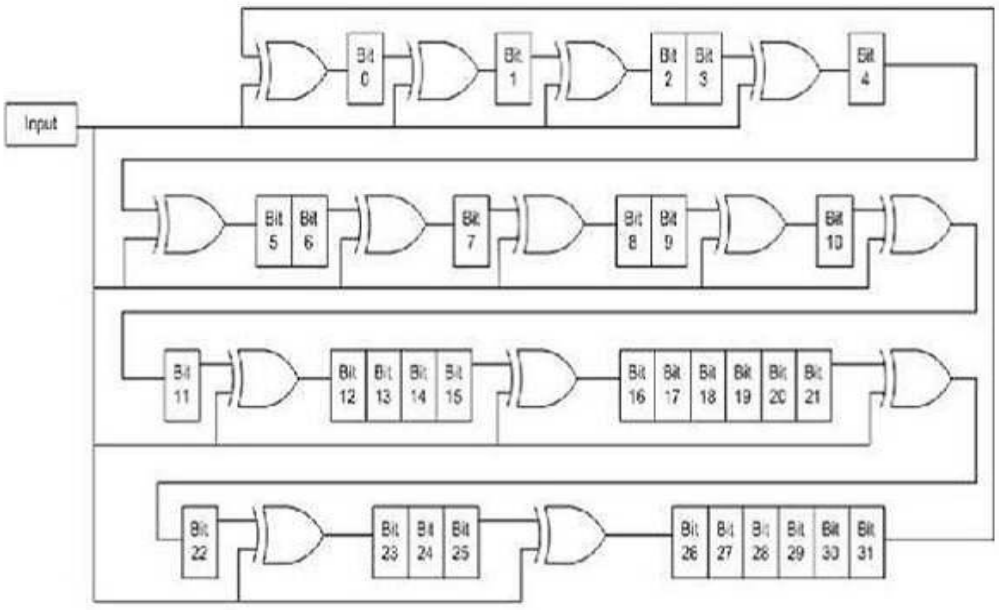
도면4


320
410
412
CRC 생성기
414
해싱 인덱스
생성부
해싱 인덱스
420
블룸
필터
430
룰
검색부

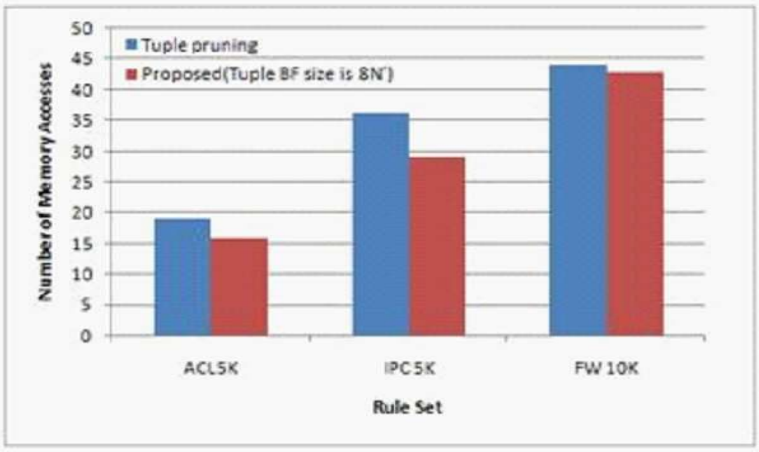
도면5


320
510
512
CRC 생성기
514
해싱 인덱스
생성부
블룸 필터용
해싱 인덱스
520
블룸
필터
530
룰
검색부
해싱 테이블용 해싱 인덱스

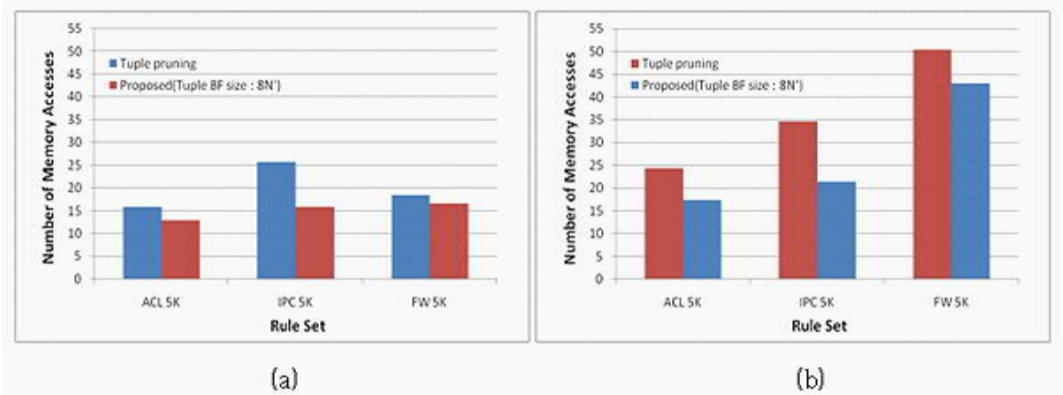
도면6


Input
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15
Bit 16
Bit 17
Bit 18
Bit 19
Bit 20
Bit 21
Bit 22
Bit 23
Bit 24
Bit 25
Bit 26
Bit 27
Bit 28
Bit 29
Bit 30
Bit 31

도면7


Tuple pruning
Proposed(Tuple BF size is 8N')
Number of Memory Accesses
50
45
40
35
30
25
20
15
10
5
0
ACL5K
IPC5K
FW10K
Rule Set

도면8


Tuple pruning
Proposed(Tuple BF size : 8N')
Number of Memory Accesses
55
50
45
40
35
30
25
20
15
10
5
0
ACL 5K
IPC 5K
FW 5K
Rule Set
(a)
(b)