



(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。 Int. Cl.

H04L 12/56 (2006.01)

(11) 공개번호

10-2007-0084760

(43) 공개일자

2007년08월27일

(21) 출원번호 10-2006-0016944

(22) 출원일자 2006년02월21일

심사청구일자 2006년02월21일

(71) 출원인

임혜숙

서울특별시 강남구 도곡동 91-5 도곡동삼성래미안아파트 106-1301

(72) 발명자

임혜숙

서울특별시 강남구 도곡동 91-5 도곡동삼성래미안아파트 106-1301

(74) 대리인

이헌수

유경열

전체 청구항 수 : 총 15 항

(54) 우선순위 기반의 영역분할을 이용한 패킷 분류 방법 및 이를 적용한 패킷 중계 장치

(57) 요약

본 발명은 영역분할을 이용한 패킷 분류 방법 및 이를 적용한 패킷 중계 장치에 관한 것으로, 본 발명에 따른 패킷 중계 장치의 패킷 분류 방법은, (a) 적어도 하나의 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 정의한 패킷 분류 테이블을 생성하는 단계와, (b) 생성된 패킷 분류 테이블에 포함된 각 규칙들에 대하여 트리를 생성하여 저장하는 단계와, (c) 패킷 중계 장치로 입력되는 패킷 중에서 분류에 사용될 비트를 추출하여 플로우 조건을 생성하는 단계와, (d) 플로우 조건에 매치하는 규칙을 찾기 위하여 저장된 트리를 검색하는 단계와, (e) 검색된 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 서비스를 제공하는 단계를 포함하며, 규칙들을 저장할 트리는 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 생성되는 것을 특징으로 한다. 이에 따라, 검색 시간을 줄일 수 있을 뿐만 아니라 소요되는 메모리의 크기도 줄일 수 있는 효율적인 패킷 분류 방법을 제공할 수 있다.

대표도

도 8

특허청구의 범위

청구항 1.

패킷 중계 장치의 패킷 분류 방법에 있어서,

(a) 적어도 하나의 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 정의한 패킷 분류 테이블을 생성하는 단계;

(b) 상기 생성된 패킷 분류 테이블에 포함된 각 규칙들에 대하여 트리를 생성하여 저장하는 단계;

- (c) 상기 패킷 중계 장치로 입력되는 패킷 중에서 분류에 사용될 비트를 추출하여 플로우 조건을 생성하는 단계;
 - (d) 상기 플로우 조건에 매치하는 규칙을 찾기 위하여 상기 저장된 트리를 검색하는 단계; 및
 - (e) 검색된 상기 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 서비스를 제공하는 단계를 포함하며,
- 상기 트리는 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 생성되는 것을 특징으로 하는 패킷 분류 방법.

청구항 2.

제1항에 있어서,

상기 (a) 단계의 패킷 분류 테이블을 구성하는 규칙은,

근원지 프리픽스, 목적지 프리픽스, 근원지 포트, 목적지 포트, 프로토콜 타입 중 적어도 하나의 패킷 필드와, 상기 적어도 하나의 패킷 필드에 기초하여 패킷을 분류하는 클래스 정보를 포함하는 것을 특징으로 하는 패킷 분류 방법.

청구항 3.

제1항에 있어서,

상기 (b) 단계의 트리를 생성하여 저장하는 방법은,

- (b1) 상기 적어도 하나의 패킷 필드에 기초하여 재귀적 영역분할(recursive area decomposition)을 수행하는 단계;
- (b2) 분할된 각 영역에 속하는 상기 규칙들 중 우선순위가 가장 높은 규칙을 먼저 노드에 저장하는 단계; 및
- (b3) 분할된 각 영역을 가로지르는 크로싱(crossing) 상태의 규칙은 상기 우선순위가 가장 높은 규칙을 저장한 노드에 연결 리스트(linked list)로 연결하여 저장하는 단계를 포함하는 것을 특징으로 하는 패킷 분류 방법.

청구항 4.

제3항에 있어서,

상기 (b) 단계의 트리를 생성하여 저장하는 방법은,

상기 생성된 패킷 분류 테이블에 포함된 모든 규칙들이 상기 트리의 각 노드들에 저장될 때까지 상기 (b1) 단계 내지 (b3) 단계를 반복하는 것을 특징으로 하는 패킷 분류 방법.

청구항 5.

제4항에 있어서,

상기 생성된 패킷 분류 테이블에 포함된 모든 규칙들에 대하여 생성된 트리를 구성하는 모든 노드들은 적어도 하나의 규칙을 저장하는 것을 특징으로 하는 패킷 분류 방법.

청구항 6.

제1항에 있어서,

상기 생성된 트리에 특정 규칙을 삭제하거나 추가하려는 경우, 삭제 또는 추가할 규칙의 영향을 상기 트리의 하나의 노드에 제한시켜 상기 트리를 업데이트하는 것을 특징으로 하는 패킷 분류 방법.

청구항 7.

제1항에 있어서,

상기 (c) 단계의 플로우 조건은,

근원지 프리픽스, 목적지 프리픽스, 근원지 포트, 목적지 포트, 프로토콜 타입 중 적어도 하나의 필드를 포함하는 것을 특징으로 하는 패킷 분류 방법.

청구항 8.

제1항에 있어서,

상기 (d) 단계의 저장된 트리를 검색하는 방법은,

(d1) 상기 플로우 조건에 매치하는 규칙을 찾기 위하여 영역분할 검색을 수행하는 단계;

(d2) 상기 (d1) 단계의 영역분할 검색 중 만약 하나의 노드에 연결 리스트로 연결된 복수의 규칙들이 저장된 경우 상기 플로우 조건에 매치하는 규칙을 찾기 위하여 상기 연결된 복수의 규칙들에 대하여 선형 검색을 수행하는 단계;

(d3) 상기 (d1) 단계 및 상기 (d2)단계의 검색결과 매치하는 규칙을 현재의 최선 정합 규칙(best matching rule: 이하 BMR이라 함)으로 저장하고, 상기 매치하는 규칙이 우선순위에 의해 저장된 경우라면 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하며, 상기 매치하는 규칙이 분할된 각 영역을 가로지르는 크로싱(crossing) 상태를 만족하여 저장된 경우라면 상기 플로우 조건의 다음 비트를 확인하고 상기 (d1) 단계로 진행하는 단계;

(d4) 상기 (d1) 단계 및 상기 (d2) 단계의 검색 결과 현재 저장된 최선 정합 규칙이 우선순위에 의하여 노드에 저장된 규칙보다 우선순위가 높은 경우 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하는 단계; 및

(d5) 저장된 트리의 리프(leaf) 노드에 도달하면 현재까지 저장된 최선 정합 규칙(BMR)을 출력하고 검색을 종료하는 단계를 포함하는 것을 특징으로 하는 패킷 분류 방법.

청구항 9.

제1항에 있어서,

상기 (e) 단계에서 제공되는 서비스는 상기 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 품질보장을 지원하는 서비스(QoS)인 것을 특징으로 하는 패킷 분류 방법.

청구항 10.

제1항에 있어서,

상기 (e) 단계에서 제공되는 서비스는 패킷 필터링(packet filtering), 트래픽 속도제한(traffic rate limit), 정책기반 라우팅(policy-based routing), 클래스에 따른 어카운팅(accounting), 클래스에 따른 요금부과(billing) 서비스 중 적어도 하나를 포함하는 것을 특징으로 하는 패킷 분류 방법.

청구항 11.

복수의 입력 포트 중 하나와 복수의 출력 포트 중 하나를 연결하는 패킷 중계 장치에 있어서,

입력 포트를 통해 수신된 패킷의 목적지 어드레스 정보를 참조하여 상기 수신된 패킷을 전송할 출력 포트를 지정하는 어드레스 룩업 처리부;

적어도 하나의 입력 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 포함하는 패킷 분류 테이블을 저장하고 상기 저장된 패킷 분류 테이블로부터 상기 입력 포트를 통해 수신된 패킷의 플로우 조건에 매치하는 규칙을 검색하는 패킷 분류 처리부; 및

상기 패킷 분류 처리부에서 검색된 상기 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 상기 지정된 출력 포트로의 상기 수신된 패킷의 전송을 제어하는 전송 제어부를 포함하며,

상기 패킷 분류 테이블은 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 트리로 생성되어 저장되는 것을 특징으로 하는 패킷 중계 장치.

청구항 12.

제11항에 있어서,

상기 패킷 분류 처리부는,

상기 패킷 분류 테이블에 포함된 모든 규칙들에 대하여 생성된 트리를 구성하는 모든 노드들이 적어도 하나의 규칙을 포함하도록, 상기 적어도 하나의 입력 패킷 필드에 기초하여 재귀적 영역분할(recursive area decomposition)을 수행하고, 분할된 각 영역에 속하는 상기 규칙들 중 우선순위가 가장 높은 규칙을 먼저 노드에 저장하며, 분할된 각 영역을 가로지르는 크로싱(crossing) 상태의 규칙은 상기 우선순위가 가장 높은 규칙에 대응하는 노드에 연결 리스트(linked list)로 연결하여 저장하는 것을 특징으로 하는 패킷 중계 장치.

청구항 13.

제11항에 있어서,

상기 패킷 분류 테이블을 구성하는 규칙은,

근원지 프리픽스, 목적지 프리픽스, 근원지 포트, 목적지 포트, 프로토콜 타입 중 적어도 하나의 패킷 필드와, 상기 적어도 하나의 패킷 필드에 기초하여 패킷을 분류하는 클래스 정보를 포함하는 것을 특징으로 하는 패킷 중계 장치.

청구항 14.

제11항에 있어서,

상기 전송 제어부는 상기 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 품질보장을 지원하는 서비스(QoS)를 제공하는 것을 특징으로 하는 패킷 중계 장치.

청구항 15.

제14항에 있어서,

상기 품질보장을 지원하는 서비스는 패킷 필터링(packet filtering), 트래픽 속도제한(traffic rate limit), 정책기반 라우팅(policy-based routing), 클래스에 따른 어카운팅(accounting), 클래스에 따른 요금부과(billing) 서비스 중 적어도 하나를 포함하는 것을 특징으로 하는 패킷 중계 장치.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명의 패킷 중계 장치에서의 패킷 분류 방법에 관한 것으로, 보다 구체적으로는 우선순위 기반의 영역분할 방식을 응용하여 품질 보증서비스를 제공하도록 패킷을 분류하는 방법에 관한 것이다.

전통적으로 인터넷 라우터는 들어오는 모든 패킷을 동일하게 처리하는 최선의 지원(best-effort-service)을 기본으로 하고 있다. 그러나 지난 수년간 인터넷 상에서 오디오나 비디오 데이터를 전송하는 네트워크 응용 프로그램이 급성장함에 따라 인터넷 서비스 제공자와 사용자들은 라우터에서 응용 프로그램의 종류와 목적에 따라 다양한 레벨의 품질보장(quality of service, QoS)을 제공할 것을 요구하게 되었다. 라우터에서 품질보장을 지원하기 위해서는 수락 제어(admission control), 자원 예약(resource reservation), 플로우별 큐잉(per-flow queuing), 출력 스케줄링(output scheduling) 등의 기능을 수행하여야 한다. 이러한 기능을 수행하기 위한 필수조건은 라우터로 들어오는 패킷들을 구별하여 서로 다른 플로우로 분류하는 패킷분류(packet classification) 기능이다. 이처럼 패킷분류 능력을 가진 라우터를 플로우-인지(flow-aware) 라우터라고 하는데, 라우터에 들어오는 패킷들의 플로우에 따라 클래스를 나누고, 각 클래스에 따라 다른 서비스를 제공한다는 점에서 전통적인 라우터와 구별된다.

각각의 클래스는 주로 패킷 헤더의 여러 필드들로 구성된 규칙(rule)들에 의해 정의되는데, 응용 프로그램들의 요구에 의해 결정된 규칙들의 집합을 분류테이블(classification table)이라 한다. 패킷분류는 패킷의 클래스들을 정의해 놓은 분류테이블에 따라 라우터로 들어오는 패킷의 클래스를 결정하는 것을 의미한다. 즉 패킷분류란 입력된 패킷의 해당 필드들과 분류테이블을 구성하고 있는 규칙들의 각 필드들이 모두 매치하는 규칙들을 찾고 그 중에서 가장 우선순위가 높은 규칙을 선택하여, 패킷을 선택된 규칙에서 규정하는 바와 같이 처리하는 과정이다.

패킷분류는 IP 주소 검색처럼 라우터에 입력되는 모든 패킷에 대하여 실시간으로 처리되어야 하지만, 네트워크 계층상 3계층(Network Layer)의 헤더 필드뿐 아니라, 4계층(Transport Layer)의 필드를 처리해야 하고 필드에 따라 각기 다른 종류의 연산이 요구되기 때문에 IP 주소 검색보다 데이터 구조나 검색 방법이 매우 복잡하고, 검색 시간이 많이 소요된다.

품질보장 지원의 요구에 따라 라우터에서의 패킷분류 기능이 더욱 강조됨에 따라 효율적인 패킷 분류를 위한 방법들이 널리 연구되고 있다. 패킷분류를 위해 제안된 방법들의 성능을 평가하기 위한 중요한 척도로는 검색 시간과 소요되는 메모리의 크기를 들 수 있다. 그동안 패킷분류를 위하여 여러 방법들이 제안되었지만, 각 방법들은 검색 시간과 소요되는 메모리 두 측면 중 한쪽에 치중된 경우가 많았다. 즉, 합당한 검색 시간을 제공하는 경우에는 분류테이블의 크기에 비하여 소요되는 메모리가 너무 크고, 반대로 적당한 메모리를 사용한 경우에는 검색 시간이 길다는 단점을 가지고 있었다.

이하, 보다 구체적으로 기존의 패킷분류 기법들을 살펴본다.

분류테이블의 검색시간을 줄이기 위하여 현재 시스코 등의 라우터장비 업체에서 가장 많이 사용하는 방법은 내용주소화 메모리(ternary content addressable memory, TCAM)를 사용하여 패킷분류를 구현하는 방법이다. TCAM은 한 번의 메

모리 접근으로 패킷분류를 가능하게 하는 매우 우수한 구조이나, 일반 메모리에 비하여 가격이 비쌌 뿐만 아니라, 면적에 있어 일반 메모리의 6배 정도를 차지하고, 전력소모가 매우 큰 단점이 있다. 이에 따라 일반 메모리를 사용하면서도 분류 테이블의 검색속도를 향상시킬 수 있는 여러 가지 패킷분류 방법들이 연구되고 있다.

기존의 패킷분류 방법은 그 접근 방법에 따라 크게 트라이(trie) 구조와 같은 기본적인 데이터 구조를 이용한 방법들과 경험적 특성을 이용한 방법, 그리고 규칙의 프리픽스 필드가 차지하는 영역에 따른 기하학적 구조를 이용한 방법들로 나누어 생각해 볼 수 있다.

1) 트라이(trie)에 기초한 방법

분류테이블을 구성하는 필드 중, IP 목적지 프리픽스(destination prefix)나 근원지 프리픽스(source prefix)와 같이 프리픽스로 표현되는 필드들을 사용하여 구성하는 방식이 트라이 구조를 이용한 방법이다. 계층트라이(hierarchical trie)는 다차원(multi-dimension)을 가지는 분류테이블에 대하여 각각의 차원에 대하여 트라이를 구성하고 개개의 트라이를 계층적으로 연결한 것이다. IP 주소 검색에서 사용되었던 트라이 구조는 한 가지 필드로 구성된 트라이이고, 패킷분류를 위해서는 이것의 리프(leaf)에 그 다음 필드를 계층적으로 연결시킴으로써 각 필드를 순차적으로 검색할 수 있게 한다.

그러나, 계층 트라이 방식은 패킷이 여러 가지 규칙에 매치할 때, 트라이의 하나의 가지로 내려와서 매치하는 규칙을 찾았다 하더라도 더 높은 순위의 규칙과 매치할 수 있으므로, 백-트래킹(back-tracking)을 해야 하는 단점이 있다. 이 때문에 프리픽스의 최대 길이를 W 라 하고 d 개의 필드로 구성된 경우, $O(W^d)$ 의 검색 시간과 $O(NdW)$ 의 메모리가 요구된다.

한편, 셋-프루닝 트라이(set-pruning trie) 방식은 트라이의 모든 경로에 매치하는 규칙을 복사하여 저장하는 방법으로 계층트라이 구조에서의 백-트래킹을 없앴으로써 검색 시간을 $O(dW)$ 로 개선한 것이다. 그러나, 향상된 검색 시간에 대한 패널티로 메모리 요구량이 $O(N^d)$ 로 증가하는 단점이 있다.

그 밖에도 전처리 과정과 스위칭포인트(switching pointer)를 통해 셋-프루닝 트라이의 이점인 단축 시간을 그대로 유지하되, 규칙을 복사하여 저장하는 단점을 해결한 방법인 그리드-오브-트라이(grid-of-trie) 방식이 있다. 그리드-오브-트라이는 메모리 요구량을 $O(NdW)$ 로 유지하면서도 검색 시간을 $O(W^{d-1})$ 로 줄인 방법이다.

그러나, 각 노드에 대하여 가장 잘 매치하는 규칙(best matching rule, BMR 또는 최선정합법칙 이라 함)을 미리 계산하여 저장하여야 하고, 각 노드에서 다른 노드로 이동하는 스위칭포인트를 미리 계산하는데 과도한 전 처리를 요구함으로써 트라이를 구성하고 업데이트하는데 시간이 많이 소요되는 단점이 있다.

요컨대, 앞서 설명한 트라이 구조들의 본질적인 문제는 트라이에 규칙을 저장하지 않는 비어있는 노드가 과도하게 존재하여 메모리 요구량이 증가하며 검색시간이 오래 걸린다는 점이다.

2) 경험적 특성을 이용한 방법

분류테이블의 경험적 특성을 이용한 방법으로는 HiCuts(Hierarchical Intelligent Cuttings)와 튜플-영역검색(tuple-space search) 방법 등이 있다.

먼저, HiCuts 방식은 규칙의 각 필드를 영역으로 표현한 다차원 검색 영역을 분할하고 특정 영역에 속하는 규칙만을 검색함으로써 검색해 보아야 하는 규칙의 개수를 줄이는데, 매 분할의 과정에서 분류테이블의 특성을 파악하여 영역을 분할할 차원과 몇 개의 영역으로 분할할 것인지를 정하는 방법이다. 이 방식은 분할된 영역을 바탕으로 판단트리(decision tree)를 구성하는데 판단트리의 각 리프는 미리 정해진 임계(threshold) 개수보다 적은 수의 규칙을 가지고 있게 된다. 검색 과정은 판단트리를 따라 내려가서 리프에 도달하고 나면, 그 리프에 저장된 규칙에 대하여는 순차적 검색을 하게 되는 방식이다.

그러나, 이 경우 판단트리 구성을 위하여 선처리 시간이 많이 요구되고, 분류테이블의 필드들이 가지는 범위 분포에 따라 검색 시간의 의존도가 크다는 단점을 가지고 있다.

한편, 분류테이블의 튜플(tuple)에 대한 특성을 고려한 방법으로는 튜플-영역검색이 있다. 튜플-영역검색은 분류테이블을 구성하는 필드 중 근원지주소와 목적지주소 쌍으로 구성된 튜플의 개수는 많지 않다는 관찰에 그 출발점을 두고 있다. 이 방식은 프리픽스의 길이 쌍으로 튜플을 구성하고 튜플들을 순차적으로 검색하되, 각 튜플에 대해서는 확정매치(exact match)가 가능하므로 해성을 적용하는 방식이다. 따라서 튜플의 수만큼 메모리 액세스가 요구된다.

최근 Srinivasan 등은 실제 라우터에서 사용되는 분류테이블의 성질을 분석하여 분류테이블을 이루고 있는 필드 중 근원지 프리픽스와 목적지 프리픽스만을 사용하여, 입력된 패킷과 매치하는 규칙들을 일차적으로 거른 후에, 걸러진 규칙들에 대해 선형검색을 사용하는 방법을 연구하였다. 이 논문에 따르면 근원지 프리픽스와 목적지 프리픽스를 사용하여 입력 패킷과 일치할 수 있는 후보규칙(candidate rule)을 고르면 5% 정도의 입력만이 20여 개의 후보규칙과 일치하고, 나머지 95%의 입력은 5개미만의 후보규칙과 일치하므로, 근원지 프리픽스와 목적지 프리픽스를 사용하는 효율적인 이차원 검색 알고리즘이 다차원 패킷분류에 그대로 적용될 수 있음을 시사 하였다.

3) 기하학적 구조를 이용하는 방법

기하학적 구조를 이용하는 방법은 규칙의 각 필드를 영역으로 표현하고 영역을 줄여가면서 패킷이 속하는 가장 구체적인 영역을 찾는 방법이다.

먼저, 외적(cross-producing) 방법은 규칙을 구성하는 각 필드들에 대하여 필드를 구성하는 케이스들을 구하고, 이러한 케이스들의 조합을 사용하여 미리 외적(cross-product) 테이블을 구성하여 놓는다. 외적 테이블에는 각 엔트리에 매치하는 BMR을 미리 계산하여 놓는다. 검색 시에는 분류테이블의 모든 필드에 대하여 일차원 검색을 각각 수행한 후, 그 결과를 포인터로 하여 외적 테이블에 접근하여 매치하는 규칙을 얻는 방법이다. 각 조합에 대하여 BMR이 구해져 있으므로, 일단 각 필드에 대하여 검색을 수행하여 그 필드가 해당하는 케이스를 구하게 되면 외적 테이블로부터 바로 BMR을 얻을 수 있어 빠른 검색이 가능하다.

그러나 외적 테이블을 위하여 $O(N^d)$ 의 큰 메모리를 요구하게 되고, 업데이트가 어려운 단점이 있다.

한편, AQT(area-based quad tree)[4]는 매 단계에서 프리픽스로 표현된 두 필드를 동시에 보면서 전체 검색 범위를 1/4로 줄여가며 패킷이 해당하는 영역내의 규칙들을 얻는 방법이다. 즉, 매 단계에서 근원지 프리픽스(source prefix)와 목적지 프리픽스(destination address)가 각각의 영역에서 이등분되어 두 프리픽스로 표현된 정사각형 영역이 같은 크기의 네 개의 영역으로 분할된다. 분할 과정이 L번 반복되면, 동일한 크기의 4^L 개의 영역으로 분할되고, 분할된 각 영역은 L-bit 길이의 프리픽스 쌍으로 표현된다. AQT에서 취하고 있는 영역분할 방식은 두 프리픽스를 동시에 한 비트씩 진행함으로 각 영역은 같은 길이의 프리픽스로만 표현된다.

그러나 필드들의 프리픽스 길이가 각기 다를 수 있다. AQT에서는 규칙을 이루고 있는 두 프리픽스 중 한쪽이라도 검색의 영역이 나타내는 프리픽스 길이와 일치하고 규칙이 나타내는 영역이 그 영역에 완전히 속하면 해당 규칙은 그 영역의 CFS(crossing filter set)에 포함되는 것으로 정의한다. 검색 시에는 CFS에 속한 규칙들에 대한 선형 검색을 수행하게 된다.

AQT는 규칙이 속하는 기하학적 영역을 정의하고, 각 영역을 사분트리(quad-tree)의 노드에 매핑 시켰다는 점에서 매우 우수한 구조이나, 실제 패킷분류에 사용되기에는 다음과 같은 문제점을 지닌다. 예를 들어 최대 프리픽스의 길이가 W라고 할 때, AQT에서의 사분트리의 깊이는 W가 되고, 프리픽스의 길이가 긴 규칙들이 많은 경우 CFS를 만족할 때까지 평면을 분할해야 하므로 사분트리에 빈 노드가 많이 존재하게 된다. 트리의 깊이는 메모리 접근 횟수와 관련이 있기 때문에 트리의 깊이가 크면 검색시간이 길어지고, 노드의 개수는 메모리 사용량과 관계가 있으므로, 빈 노드가 많아지면 과도한 메모리를 사용하게 된다는 단점이 있다.

이상에서 살펴본 바와 같이, 트라이 구조를 이용한 패킷 분류 방법이나 경험적 특성을 이용한 패킷 분류 방법, 또는 기하학적 구조를 이용한 패킷 분류 방법 등 기존의 패킷 분류 방식들은 검색 시간과 소요되는 메모리라는 두 가지 중요한 측면 중 어느 하나에 대해서만 해결책을 제시할 뿐, 검색 시간을 줄이는 동시에 소요되는 메모리의 크기도 줄일 수 있는 우수한 패킷 분류 방법을 제시하지 못한다는 문제점이 있다.

발명이 이루고자 하는 기술적 과제

따라서, 전술한 문제점을 해결하기 위한 본 발명의 목적은 검색 시간을 줄일 수 있을 뿐만 아니라 소요되는 메모리의 크기도 줄일 수 있는 효율적인 패킷 분류 방법을 제공하는 것이다.

또한, 본 발명의 다른 목적은 검색 시간을 줄일 수 있을 뿐만 아니라 소요되는 메모리의 크기도 줄일 수 있는 패킷 분류 방법을 적용한 패킷 중계 장치를 제공하는 것이다.

보다 구체적으로는, 선형검색구조와 재귀적 영역분할 방식을 결합하여 우선순위에 기초한 검색을 수행하면서도 검색영역을 지수적으로 줄여나가는 새로운 패킷 분류 방법을 제안한다. 제안하는 구조는 검색 시간을 단축할 뿐만 아니라 소요되는 메모리의 크기를 줄일 수 있으며, 나아가 분류테이블의 갱신이 용이하고 확장성이 우수한 새로운 패킷 분류 방법을 제공한다.

발명의 구성

본 발명에 따라 전술한 목적은, 패킷 중계 장치의 패킷 분류 방법에 있어서 (a) 적어도 하나의 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 정의한 패킷 분류 테이블을 생성하는 단계; (b) 생성된 패킷 분류 테이블에 포함된 각 규칙들에 대하여 트리를 생성하여 저장하는 단계; (c) 패킷 중계 장치로 입력되는 패킷 중에서 분류에 사용될 비트를 추출하여 플로우 조건을 생성하는 단계; (d) 플로우 조건에 매치하는 규칙을 찾기 위하여 저장된 트리를 검색하는 단계; 및 (e) 검색된 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 서비스를 제공하는 단계를 포함하며, 트리는 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 생성되는 것을 특징으로 하는 패킷 분류 방법에 의해 달성된다.

특히, 전술한 (b) 단계의 트리를 생성하여 저장하는 방법은, (b1) 적어도 하나의 패킷 필드에 기초하여 재귀적 영역분할(recursive area decomposition)을 수행하는 단계; (b2) 분할된 각 영역에 속하는 규칙들 중 우선순위가 가장 높은 규칙을 먼저 노드에 저장하는 단계; 및 (b3) 분할된 각 영역을 가로지르는 크로싱(crossing) 상태의 규칙은 우선순위가 가장 높은 규칙을 저장한 노드에 연결 리스트(linked list)로 연결하여 저장하는 단계를 포함하는 것이 바람직하며,

생성된 패킷 분류 테이블에 포함된 모든 규칙들에 대하여 생성된 트리를 구성하는 모든 노드들은 적어도 하나의 규칙을 저장하는 것이 바람직하다.

또한, 전술한 (d) 단계의 저장된 트리를 검색하는 방법은, (d1) 플로우 조건에 매치하는 규칙을 찾기 위하여 영역분할 검색을 수행하는 단계; (d2) (d1) 단계의 영역분할 검색 중 만약 하나의 노드에 연결 리스트로 연결된 복수의 규칙들이 저장된 경우 플로우 조건에 매치하는 규칙을 찾기 위하여 연결된 복수의 규칙들에 대하여 선형 검색을 수행하는 단계; (d3) 앞의 (d1) 단계 및 (d2)단계의 검색결과 매치하는 규칙을 현재의 최선 정합 규칙(best matching rule: 이하 BMR이라 함)으로 저장하고, 매치하는 규칙이 우선순위에 의해 저장된 경우라면 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하며, 매치하는 규칙이 분할된 각 영역을 가로지르는 크로싱(crossing) 상태를 만족하여 저장된 경우라면 플로우 조건의 다음 비트를 확인하고 (d1) 단계로 진행하는 단계; (d4) 앞의 (d1) 단계 및 (d2) 단계의 검색 결과 현재 저장된 최선 정합 규칙이 우선순위에 의하여 노드에 저장된 규칙보다 우선순위가 높은 경우 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하는 단계; 및 (d5) 저장된 트리의 리프(leaf) 노드에 도달하면 현재까지 저장된 최선 정합 규칙(BMR)을 출력하고 검색을 종료하는 단계를 포함하는 것이 바람직하다.

한편, 본 발명의 다른 분야에 따르면 전술한 목적은, 복수의 입력 포트 중 하나와 복수의 출력 포트 중 하나를 연결하는 패킷 중계 장치에 있어서, 입력 포트를 통해 수신된 패킷의 목적지 어드레스 정보를 참조하여 수신된 패킷을 전송할 출력 포트를 지정하는 어드레스 룩업 처리부; 적어도 하나의 입력 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 포함하는 패킷 분류 테이블을 저장하고 저장된 패킷 분류 테이블로부터 입력 포트를 통해 수신된 패킷의 플로우 조건에 매치하는 규칙을 검색하는 패킷 분류 처리부; 및 패킷 분류 처리부에서 검색된 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 지정된 출력 포트로의 수신된 패킷의 전송을 제어하는 전송 제어부를 포함하며, 패킷 분류 테이블은 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 트리로 생성되어 저장되는 것을 특징으로 하는 패킷 중계 장치에 의해서 달성된다.

특히, 전술한 패킷 분류 처리부는, 패킷 분류 테이블에 포함된 모든 규칙들에 대하여 생성된 트리를 구성하는 모든 노드들이 적어도 하나의 규칙을 포함하도록, 적어도 하나의 입력 패킷 필드에 기초하여 재귀적 영역분할(recursive area

decomposition)을 수행하고, 분할된 각 영역에 속하는 규칙들 중 우선순위가 가장 높은 규칙을 먼저 노드에 저장하며, 분할된 각 영역을 가로지르는 크로싱(crossing) 상태의 규칙은 우선순위가 가장 높은 규칙에 대응하는 노드에 연결 리스트(linked list)로 연결하여 저장하는 것을 특징으로 하는 것이 바람직하다.

이하에서는 첨부한 도면을 참조하여 본 발명의 바람직한 실시예를 상세히 설명한다. 본 발명을 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 또한, 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

먼저 본 발명에서 이용되는 선형검색방법과 재귀적 영역분할방법을 상세히 살펴보고 본 발명에 따른 새로운 패킷 분류 방법을 설명한다.

1) 선형검색방법(Linear Search)

분류테이블을 저장하는 가장 기본적인 구조는 규칙들을 우선순위에 따라 저장하는 것이다. 선형검색은 우선순위에 따라 저장된 규칙들을 우선순위가 높은 규칙부터 순차적으로 검색해 나가는 방법으로 매치하는 규칙을 찾거나 리스트 끝에 도달하면 검색이 종료된다. 선형검색은 기존에 연구되어온 방식 중 유일하게 규칙들의 우선순위에 따라 분류테이블을 구성하고, 우선순위가 높은 규칙부터 검색하여 나가는 방법이다.

선형검색의 경우 N 개의 규칙을 저장하기 위하여 규칙을 한 번씩만 저장하므로 $O(N)$ 의 메모리가 사용되는 장점을 갖는다. 그러나 선형검색은 한 번의 테이블 룩업으로 단 하나의 엔트리만이 검색영역에서 제외되므로, 검색영역이 선형적으로 감소하여 $O(N)$ 의 검색 시간이 소요되며, 이로 인하여 확장성(scalability)이 매우 나쁘다는 단점을 가진다.

본 발명에서는 이러한 선형검색의 장점은 살리고 단점을 보완한 새로운 패킷 분류 방법을 제안한다.

2) 재귀적 영역분할방법(Recursive Range Decomposition)

규칙을 구성하는 필드들 중 프리픽스로 표현되는 근원지 프리픽스 또는 목적지 프리픽스는 기하학적인 관점으로 보면 영역(range)으로 해석될 수 있다. 규칙을 이루는 하나의 프리픽스 필드만을 고려하면 규칙은 일차원 직선상에서의 영역으로 표현된다. 만약 근원지와 목적지 두 개의 필드를 동시에 고려하면, 규칙은 이차원 평면에서 사각형으로 표현되며, 만약 d 개의 필드를 동시에 고려하면 d-차원 다면체(hyper-rectangle)로 표현될 수 있다.

본 발명에서는 설명의 편의를 위하여, 규칙을 이루는 필드 중 프리픽스로 이루어진 필드만을 고려하는 경우를 예로 들어 설명한다. 즉, 하나의 프리픽스 필드를 고려하는 일차원 이진영역분할구조와 두 개의 프리픽스 필드를 동시에 고려하는 이차원 사분영역분할구조를 예로 들어 설명하기로 한다. 그러나, 본 발명의 사상은 일차원 이진영역분할구조나 이차원 사분영역분할구조에 한정되는 것이 아니며 동일한 방식이 다차원 영역분할구조로 확장 적용될 수 있다.

예를 들어 W를 IP 주소의 길이라고 할 때, i-bit 길이의 근원지 프리픽스를 갖는 규칙은 크기가 2^W 인 직선상에서 크기가 2^{W-i} 인 직선영역으로 표현된다(CASE 1). 한편 i-bit 길이의 근원지 프리픽스와 j-bit 길이의 목적지 프리픽스를 모두 고려하는 경우라면 이 규칙은 $2^W \times 2^W$ 평면 내에서 $2^{W-i} \times 2^{W-j}$ 크기의 사각형 영역으로 표현된다(CASE 2).

입력된 패킷은 근원지 주소 (또는 목적지 주소)만을 고려하면, 크기가 2^W 인 직선상에서 하나의 점으로 정의된다. 한편 근원지 주소와 목적지 주소를 모두 고려하면 크기가 $2^W \times 2^W$ 평면상에서 하나의 점으로 정의된다. 이 점을 포함하게 되는 규칙이 입력패킷에 매치하는 규칙이 된다. 규칙을 표현하는 영역들은 서로 겹쳐질 수 있기 때문에 패킷은 여러 개의 규칙과 매치할 수 있다. 이때는 가장 우선순위가 높은 규칙을 최선 정합 규칙(BMR)으로 선정한다.

주소 평면을 반복적으로 분할함에 의하여 일차원 직선정보는 이진트리(binary tree)를 이용하여 나타낼 수 있다. 한편 이차원 평면정보는 사분트리(quadruple tree)를 사용하여 나타낼 수 있다. 이진트리는 직선을 반복적으로 분할함에 있어서, 각 레벨에서 같은 크기의 두개의 직선으로 나누는 재귀적인 영역분할(recursive range decomposition)에 기초한다. 트리의 각 노드는 현재 분할되는 직선에 대응되고, 두 개의 자식 노드는 분할하여 얻어진 두 개의 직선에 대응된다.

한편 사분트리는 평면을 반복적으로 분할함에 있어서, 각 레벨에서 같은 크기의 네 개의 평면으로 나누는 재귀적인 영역분할 (recursive space decomposition)에 기초한다. 트리의 각 노드는 현재 분할되는 평면에 대응되고, 네 개의 자식 노드는 분할하여 얻어진 네 개의 평면에 대응된다.

표 1은 12개의 규칙들로 이루어진 분류테이블의 예를 보여준다. 근원지 프리픽스와 목적지 프리픽스, 근원지 포트번호와 목적지 포트번호, 그리고 프로토콜 필드를 규칙을 규정하는 필드로서 사용한 예가 도시되어 있다. 규칙의 번호가 작은 것이 규칙의 우선순위가 높다고 가정하였다. 표 1의 분류테이블을 예로 하여, 재귀적 영역분할에 의한 트리구조를 설명하기로 한다.

[표 1]
분류 테이블의 일 예

Rule No	Src Prefix	Dst Prefix	Src Port	Dst Prot	Protocol Type	Class Info
0	10100*	*	>1024	>1024	*	Permit
1	11110*	*	80	80	TCP	Permit
2	0*	11100*	80	80	TCP	Permit
3	0*	11000*	88	88	TCP	Permit
4	111*	11110*	*	>1024	UDP	Deny
5	01110*	0111*	80	80	TCP	Permit
6	1*	10011*	21	21	TCP	Permit
7	*	10101*	80	80	*	Permit
8	1100*	1100*	88	88	TCP	Permit
9	1101*	1101*	*	*	*	Deny
10	010*	010*	>1024	>1024	UDP	Deny
11	0110*	10*	80	80	*	Permit

도 1은 근원지 프리픽스를 이용한 영역분할의 일 예이다.

도 1을 참조하면, 전술한 표 1의 분류테이블을 이루는 규칙들의 근원지 프리픽스만을 고려하여 각 규칙들이 직선상에서 차지하는 영역을 표현한 예가 도시된다.

도 2는 종래의 근원지 프리픽스를 이용한 이진트리 구조의 일 예이다.

도 2를 참조하면, 재귀적 영역분할에 의해 각 규칙들을 트리의 노드에 매핑시킨 이진검색 트리구조가 도시된다. 즉, 규칙을 이루는 근원지 프리픽스의 길이에 따라 규칙이 위치하게 되는 트리의 레벨이 결정되며, 트리의 레벨은 입력패킷과 그 규칙이 비교되기 위하여 거쳐야 되는 메모리 접근 횟수와도 일치한다. 예를 들어 도시된 그림에서 규칙 R0은 우선순위가 가장 높은 규칙임에도 불구하고, 근원지 프리픽스의 길이가 5이므로, 트리의 5번째 레벨에 위치하게 되고, 입력패킷과 비교되기 위해서는 다섯 번의 메모리 검색이 필요하다. 이와 같이 기존의 일차원 영역분할방법에 의하면 우선순위가 높은 규칙이라도 여러 번의 메모리 접근을 시도해야 하는 문제점이 있다.

한편, 도 3은 근원지 프리픽스와 목적지 프리픽스를 이용한 이차원 영역분할의 일 예이다.

도 3을 참조하면, 표 1의 분류테이블을 이루는 규칙들의 근원지 프리픽스와 목적지 프리픽스를 동시에 고려하여, 각 규칙들이 이차원 평면상에서 차지하는 영역을 표현한 예가 도시된다.

도 4는 종래의 근원지 프리픽스와 목적지 프리픽스를 이용한 AQT 사분트리 구조의 일 예이다.

도 4를 참조하면, AQT에서 제안한 재귀적 영역분할에 의해 각 규칙들을 사분트리의 노드에 매핑시킨 사분 트리구조가 도시된다. 루트노드는 전체 영역에 대응되고, 4개의 자식노드들은 각각 전체 사각형 영역을 같은 크기로 분할한 네 개의 사각형 영역에 대응되는 것으로서, 좌측상단(top-left), 좌측하단(bottom-left), 우측상단(top-right), 우측하단(bottom-right)의 사각형이 각각 사분트리의 0번, 1번, 2번, 3번 노드에 대응된다.

특정 사각형 영역을 가로지르는 규칙들을 이 사각형 영역에 대응되는 노드에 위치시키고, 모든 규칙들이 노드에 매핑되기까지 재귀적으로 분할을 반복한다. 이와 같이 하면 규칙을 이루는 근원지 프리픽스 길이와 목적지 프리픽스 길이 중 짧은 길이에 해당하는 레벨에 규칙이 위치하게 된다. 예를 들어 도 4에서 규칙 R0과 R1은 목적지 프리픽스의 길이가 0이므로 사분트리의 루트노드에 위치하였고, 규칙 R7은 근원지 프리픽스의 길이가 0이므로 루트 노드에 위치하였다.

이러한 사분트리 방식도 일차원 이진트리 방식의 경우와 마찬가지로, 해당 규칙이 가지는 우선순위와는 무관하게, 규칙을 구성하는 프리픽스의 길이에 의하여 규칙을 갖는 노드의 레벨이 결정되므로 검색이 효율적이지 못한 단점을 갖는다. 또한 도 4에 도시된 바와 같이 규칙을 갖지 않는 빈 노드가 생기게 되며 이러한 빈 노드는 규칙을 구성하는 프리픽스의 길이가 길어질수록 많아지게 되어 검색 시간을 늘리게 되는 문제점이 있다.

이하, 전술한 기존의 선형검색 및 영역분할 방식의 단점을 해결한 본 발명에 따른 새로운 패킷 분류 방법 및 이를 적용한 패킷 중계 장치를 상세히 설명한다.

먼저, 패킷 중계 장치의 구조를 살펴보고, 본 발명에 따른 패킷 분류 방법을 설명한다.

도 5는 본 발명의 일 실시예에 따른 패킷 중계 장치의 블록도이다. 패킷 중계 장치는 복수의 입력 포트 중 하나와 복수의 출력 포트 중 하나를 연결하여 입력된 패킷을 지정된 출력 포트에 출력하는 중계 장치로서, 라우터(router)가 대표적이다.

도 5를 참조하면, 본 발명에 따른 패킷 중계 장치는 복수의 인터페이스(10)와 스위치 패브릭(switch fabric)(20)을 구비한다. 그 밖에 관리 콘트롤러(management controller)나 루트 콘트롤러(route controller)와 같은 제어부가 더 포함될 수 있다. 패킷 중계 장치는 복수의 입력 포트와 복수의 출력 포트가 구비되어 복수의 입력 포트 중 하나로 입력된 패킷을 스위치 패브릭(20)을 거쳐 지정된 출력 포트에 출력할 수 있다. 본 발명에 따른 패킷 분류 기능은 인터페이스(10)에서 담당한다.

도 6은 도 5에 도시된 인터페이스의 상세 블록도이다. 도 6을 참조하면, 인터페이스(10)는 패킷 송수신부(transponder/transceiver)(11), 프레임 해석부(framer)(12), 어드레스 룩업 처리부(lookup processor)(13), 패킷 분류 처리부(packet classification processor)(14) 및 전송 제어부(traffic manager)(15)를 구비한다. 그 밖에 CPU와 메모리를 더 구비한다.

패킷 송수신부(transponder/transceiver)(11)는 입력 포트를 통해 패킷을 수신하고 출력 포트를 통해 패킷을 송신한다. 프레임 해석부(framer)(12)는 입력된 패킷의 시작과 끝을 알아내고 패킷을 해석하여 각 계층의 헤더를 분리하는데, 패킷 분류를 위한 필요한 근원지 프리픽스, 목적지 프리픽스, 근원지 포트, 목적지 포트, 프로토콜 타입 등의 패킷 필드 등도 이곳에서 알아낸다. 어드레스 룩업 처리부(lookup processor)(13)는 입력 포트를 통해 수신된 패킷의 목적지 프리픽스 정보를 참조하여 상기 수신된 패킷을 전송할 출력 포트를 지정한다.

한편, 패킷 분류 처리부(packet classification processor)(14)는 적어도 하나의 입력 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 포함하는 패킷 분류 테이블을 저장하고 저장된 패킷 분류 테이블로부터 입력 포트를 통해 수신된 패킷의 플로우 조건에 매치하는 규칙을 검색한다. 이때, 패킷 분류 테이블은 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 트리로 생성되어 저장되는 것을 특징으로 한다.

구체적으로 패킷 분류 처리부(14)는 패킷 분류 테이블에 포함된 모든 규칙들에 대하여 생성된 트리를 구성하는 모든 노드들이 적어도 하나의 규칙을 포함하도록, 적어도 하나의 입력 패킷 필드에 기초하여 재귀적 영역분할(recursive area decomposition)을 수행하고, 분할된 각 영역에 속하는 규칙들 중 우선순위가 가장 높은 규칙을 먼저 노드에 저장하며, 분할된 각 영역을 가로지르는 크로싱(crossing) 상태의 규칙은 우선순위가 가장 높은 규칙에 대응하는 노드에 연결 리스트(linked list)로 연결하여 저장하는 것을 특징으로 한다. 보다 구체적인 트리 구조와 그 검색 방법은 후술한다.

전송 제어부(traffic manager)(15)는 패킷 분류 처리부(14)에서 검색된 플로우 조건에 매치하는 규칙에 포함된 클래스 정보(Class Info)에 기초하여 지정된 출력 포트에 수신된 패킷의 전송을 제어한다. 즉, 입력 패킷의 플로우 조건에 매치하는 규칙에 정의된 클래스 정보에 따라 품질보장을 지원하는 서비스를 제공한다. 예를 들면, 품질보장을 지원하는 서비스에

는 패킷 필터링(packet filtering), 트래픽 속도제한(traffic rate limit), 정책기반 라우팅(policy-based routing), 클래스에 따른 어카운팅(accounting), 클래스에 따른 요금부과(billing) 등의 서비스가 포함될 수 있으며, 그 밖에 클래스 정보에 기초하여 다양한 품질보장을 지원하는 서비스를 제공할 수 있다.

한편, 우선순위 기반의 영역검색 구조를 가지는 본 발명에 따른 패킷 분류 방법을 상세히 설명한다.

도 9는 본 발명의 일 실시예에 따른 패킷 분류 방법을 도시한 흐름도이다.

도 9를 참조하면 본 발명에 따르면 효율적으로 패킷을 분류하기 위하여, 적어도 하나의 패킷 필드에 기초하여 패킷의 클래스를 구분하는 규칙들을 정의한 패킷 분류 테이블을 생성하고(S100), 생성된 패킷 분류 테이블에 포함된 각 규칙들에 대하여 트리를 생성하여 저장하며(S120), 패킷 중계 장치로 입력되는 패킷 중에서 분류에 사용될 비트를 추출하여 플로우 조건을 생성하고(S140), 플로우 조건에 매치하는 규칙을 찾기 위하여 저장된 트리를 검색하며(S160), 검색된 플로우 조건에 매치하는 규칙에 포함된 클래스 정보에 기초하여 서비스를 제공한다(S180). 이때, 트리는 미리 정의된 규칙들의 우선순위에 기초하여 영역분할 방식을 이용하여 생성되는 것을 특징으로 한다.

이하 본 발명에 따른 패킷 분류 방법을 보다 상세히 설명한다.

1) 트리의 생성(Building Tree)

본 발명에 따른 이진트리의 노드에는 각 노드에 대응되는 영역에 속하는 규칙 중 가장 우선순위가 높은 규칙이 가장 먼저 저장되며, 이에 더하여 영역을 가로지르는 crossing 규칙들은 그 우선순위에 따라 연결리스트(linked list)로 연결하여 저장된다.

(1) 이진트리의 구성

먼저, 규칙을 이루는 필드 중 근원지 프리픽스를 이용하는 경우로서 일차원 영역분할에 기초한 이진트리의 구성을 예로 들어 설명한다.

우선, 분류테이블에 포함된 규칙 중 가장 우선순위가 높은 규칙 R0을 루트 노드에 저장한다. 만약 근원지 프리픽스에 와일드카드를 갖는 규칙(들)이 있다면, 이 규칙(들)은 전체 영역을 가로지르는(crossing) 규칙이므로, 이 규칙(들)을 함께 저장한다. 왜냐하면 해당 영역을 가로지르는 규칙을 같이 저장하지 않는다면 이들 규칙이 해당 노드보다 아래 레벨의 노드에 저장되며, 이때 규칙이 속하게 되는 모든 영역에 규칙을 복사하여 저장해야 하는 문제점이 발생하기 때문이다. 따라서, 해당 영역을 가로지르는 규칙을 우선순위가 가장 높은 규칙과 함께 저장하는 것이 바람직하다. 하나의 노드에 여러 개의 규칙이 저장되는 경우 규칙들은 우선순위에 따라 연결리스트(linked list)에 의하여 연결되도록 한다.

다음으로 루트의 첫 번째 자식 노드에 대해서도 같은 방식으로 해당영역에 포함된 규칙 중 가장 우선순위가 높은 규칙을 저장하고, 해당영역을 가로지르는 규칙이 있다면 연결 리스트로 연결하여 저장한다. 분류테이블의 규칙들의 집합은 노드에 규칙이 저장될 때마다 저장된 규칙들을 지워나가는 방식으로 갱신되며, 이와 같은 검색 평면 분할 과정을 분류테이블의 마지막 규칙이 저장될 때까지 반복한다.

한편, 앞서 설명한 기존의 영역분할방법(AQT)에 의하면 최대 프리픽스의 길이를 W라고 할 때 AQT에서의 트리의 깊이는 W가 된다. 또한 프리픽스의 길이가 긴 규칙들이 많은 경우 크로싱(crossing) 상태를 만족할 때까지 평면을 분할해야 하므로 트리에 빈 노드가 많아지게 되는 문제점이 발생한다.

그러나, 본 발명에 따른 패킷 분류 방법에 있어서 트리 생성 방법을 이용하면, 각 노드에서 크로싱(crossing) 상태를 만족하지 않아도 현재의 규칙 집합에 있는 규칙 중, 그 영역에 속하면서 우선순위가 가장 높은 규칙을 저장하기 때문에 트리에 빈 노드가 없어지게 되며, 따라서 트리의 깊이도 줄어든다. 이는 트리의 검색시간이 단축될 수 있음을 시사한다. 실제 라우터에서 사용되는 분류테이블들을 분석하면, 규칙을 이루는 IP 주소의 근원지와 목적지 프리픽스의 길이는 대부분 0, 16, 24, 32에 분포되는 경향이 있다. 따라서 기존의 영역분할방법(AQT)을 실제 분류테이블에 적용하면 트리 깊이가 매우 크게 되는 문제점이 있는데 반해, 본 발명에 따른 우선순위에 기반을 둔 영역분할방법(PQT)을 이용하면 프리픽스 길이가 16 이상인 규칙들도 낮은 레벨의 노드에 저장되어 트리 깊이가 줄어들어 검색시간을 단축할 수 있다.

도 7은 본 발명에 따른 근원지 프리픽스를 이용한 이진트리 구조의 일 예를 나타낸다. 즉, 표 1에 도시된 분류테이블의 규칙들을 도 1에 도시된 바와 같이 근원지 프리픽스를 이용하여 영역 분할한 결과를 기초로 이진트리로 생성한 결과이다.

도 1 및 도 7을 참조하면, 큰 글씨로 표시된 규칙들은 우선순위에 의하여 저장된 규칙들을 나타내며, 작은 글씨로 표시된 규칙들은 크로싱 상태를 만족하여 저장된 규칙들을 나타낸다. 예를 들면, R0은 우선순위가 해당 영역에서 가장 높아 저장된 규칙이고 R7은 근원지 프리픽스가 크로싱 상태를 만족하여 저장된 규칙이다. 규칙 R2의 경우 루트의 왼쪽 영역에서 가장 우선순위가 높은 규칙이면서 동시에 크로싱 상태를 만족하는데, 우선순위조건이 먼저 고려되므로 우선순위에 의하여 저장된다. 규칙 R3은 크로싱 상태를 만족하여 규칙 R2와 함께 저장된다.

이와 같이 본 발명에 따른 패킷 분류 방법에 따라 생성된 이진트리를 도 2에 도시된 기존의 이진트리와 비교하면, 트리의 레벨이 5에서 4로 줄어든 것을 확인할 수 있다. 이에 따라 리프(leaf)까지 검색해야 하는 경우에도 검색시간을 단축할 수 있다.

나아가, 도 2에 도시된 기존의 이진트리에는 빈 노드들이 다수 포함된 데 비해, 본 발명에 따른 이진트리에는 빈 노드가 완전히 없어진 것을 확인할 수 있다. 즉, 본 발명에 따른 이진트리구조에서는 우선순위가 높은 규칙들일 수록 상위 레벨의 노드에 저장되므로 입력 패킷의 플로우와 먼저 비교하게 되어 검색 시간을 단축할 수 있다. 또한, 빈 노드가 없어지므로 메모리 공간을 절약하게 된다.

(2) 사분트리의 구성

한편, 규칙을 이루는 필드 중 근원지 프리픽스와 목적지 프리픽스를 이용하는 경우로서 이차원 영역분할에 기초한 사분트리의 구성이 도 8에 도시되어 있다. 즉, 표 1에 도시된 분류테이블의 규칙들을 도 3에 도시된 바와 같이 근원지 프리픽스와 목적지 프리픽스를 이용하여 영역 분할한 결과를 기초로 사분트리로 생성한 결과이다.

도 3 및 도 8을 참조하면, 우선, 분류테이블에 포함된 규칙 중 가장 우선순위가 높은 규칙 R0이 루트 노드에 저장된다. 또한 전체 영역을 가로지르는 크로싱 상태의 규칙 R1 및 R7이 R0에 연결리스트로 연결되어 저장된다.

다음으로 루트의 첫 번째 자식 노드에 대해서도 같은 방식으로 해당영역에 포함된 규칙 중 가장 우선순위가 높은 규칙을 저장하고, 해당영역을 가로지르는 규칙이 있다면 연결 리스트로 연결하여 저장한다. 분류테이블의 규칙들의 집합은 노드에 규칙이 저장될 때마다 저장된 규칙들을 지워나가는 방식으로 갱신되며, 이와 같은 검색 평면 분할 과정을 분류테이블의 마지막 규칙이 저장될 때까지 반복한다.

이에 따라 생성된 사분트리구조가 도 8에 도시되어 있으며, 이를 도 4에 도시된 기존의 사분트리와 비교해 보면, 트리의 깊이가 4에서 3으로 줄어들어 검색 시간이 단축됨을 알 수 있다. 나아가, 도 4에 도시된 기존의 사분트리에는 다수의 빈 노드들이 포함되었으나, 도 8에 도시된 본 발명에 따른 사분트리에는 빈 노드가 하나도 포함되지 않아 검색시간을 단축하고 메모리 공간을 절약하게 된다.

2) 이진트리를 저장한 메모리의 구조(Table Structures)

본 발명에 따른 이진트리 구조를 저장한 메모리의 구조가 표 2에 도시되어 있다.

[표 2]
이진트리 구조를 저장한 테이블

Rule No	Stored Priority	Src Prefix	Dst Prefix	Src Port	Dst Port	Protocol Type	Child #1		Child #2		Linked Ptr
							Valid	Ptr	Valid	Ptr	
0	1	10100*	*	>1024	>1024	*	1	2	1	1	7
1	1	11110*	*	80	80	TCP	0	-	1	4	6
2	1	0*	11100*	80	80	TCP	0	-	1	5	3
3	0	0*	11000*	88	88	TCP	0	-	1	5	-
4	1	111*	11110*	*	>1024	UDP	1	8	0	-	-
5	1	01110*	0111*	80	80	TCP	1	10	1	11	-
6	0	1*	10011*	21	21	TCP	0	-	1	4	-
7	0	*	10101*	80	80	*	1	2	1	1	-
8	1	1100*	1100*	88	88	TCP	0	-	1	9	-

9	1	1101*	1101*	*	*	*	0	-	0	-	-
10	1	010*	010*	>1024	>1024	UDP	0	-	0	-	-
11	1	0110*	10*	80	80	*	0	-	0	-	-

표 2에 도시된 바와 같이 본 발명에 따른 패킷 분류 방법에서의 규칙들은 우선순위에 따라 차례대로 메모리에 저장되며, 각 엔트리에는 규칙의 번호와 함께, 현재의 규칙이 우선순위에 의하여 저장되었는지 또는 크로싱(crossing) 상태를 만족하여 저장되었는지를 나타내는 1 비트 정보(Stored Priority)가 저장된다. 이 정보는 후술할 트리의 검색과정에서 사용된다.

그 다음으로는 규칙의 나머지 필드들이 차례대로 저장된 후 자식 노드를 가리키는 필드(Child #1 및 Child #2)가 저장된다. 자식노드가 없으면, 유효(valid) 비트는 0이 되고, 포인터(ptr)는 유효하지 않으며, 자식노드가 있으면, 유효(valid) 비트는 1이 되고, 포인터(ptr)는 자식노드가 저장된 메모리의 주소를 가리킨다. 연결 포인터(linked ptr)는 같은 노드에 해당하는 다음 규칙이 저장된 메모리의 주소를 가리킨다.

표 2에 도시된 바와 같이 본 발명에 따른 패킷 분류 방법에서는 선형검색 구조와 마찬가지로 규칙의 개수와 같은 수의 메모리 엔트리를 필요로 하며 규칙이 우선순위에 따라 저장되어 있다. 다만, 이진트리구조를 표현하기 위하여 자식노드를 가리키는 두개의 포인터(child #1의 ptr 및 child #2의 ptr)와 같은 노드에 저장된 다른 규칙을 가르치는 포인터(linked ptr)만이 추가로 저장된 것이 기존의 선형검색과의 차이점이다.

동일한 방식으로 본 발명에 따른 패킷 분류 방법을 위한 사분트리구조를 메모리에 저장할 수 있다. 이 때, 네 개의 자식노드를 가리키기 위해 자식 노드를 위한 포인터가 4개로 추가될 뿐 다른 구조는 이진트리구조의 경우와 동일하다. 따라서, 사분트리구조를 위한 메모리 구조에 대한 설명은 생략한다.

3) 검색과정(Search Procedure)

패킷에 매치하는 규칙을 찾기 위한 검색 과정은 앞서 보인 이진트리 메모리 구조를 이용하여 설명한다. 검색은 이진트리 구조를 저장한 트리 메모리의 첫 번째 엔트리, 즉 트리의 루트 노드로부터 시작한다. 제안하는 구조의 검색 과정이 선형검색과 다른 점은 검색 영역을 1/2로 줄여가면서 검색을 한다는 것이다. 아래 레벨의 노드로 하나씩 내려가면서 검색 영역이 1/2로 줄기 때문에 검색을 빠르게 할 수 있다는 장점이 있다.

하나의 노드에 여러 개의 규칙이 저장된 경우, 저장된 규칙들에 대해서는 연결리스트에 의한 선형 검색을 한다. 실제 분류 테이블의 특성상 루트 노드를 제외한 나머지 노드들에는 크로싱 상태를 만족하는 규칙보다 우선순위에 의하여 규칙이 저장될 가능성이 매우 크기 때문에, 대부분의 노드에 하나의 규칙만이 저장되게 되어 선형 검색에 대한 검색 시간은 문제되지 않는다.

도 10은 본 발명의 일 실시예에 따른 검색 방법을 도시한 흐름도이다.

도 10을 참조하면, 먼저 입력된 패킷의 헤더필드가 루트 노드, 즉 첫 번째 엔트리에 저장된 규칙과 매치하는지 비교한다(S1604). 매치하지 않는다면 연결리스트가 존재하는지 확인하여(S1610) 존재한다면 연결리스트가 가르치는 메모리의 주소로 이동하여 같은 과정을 반복한다(S1614). 매치하는 경우, 매치하는 규칙을 현재의 최선 정합 규칙(best matching rule, 이하 BMR이라 함)으로 저장하고(S1606), 매치하는 규칙이 우선순위에 의하여 저장된 규칙이라면(S1608) 가장 우선순위가 높은 매치하는 규칙을 찾은 것이므로 검색을 완전히 종료한다(S1618).

만약 매치하는 규칙이 우선순위에 의하여 저장된 규칙이 아니라면, 즉 크로싱 상태를 만족하기 때문에 저장된 규칙이라면(S1608), 아래 노드에 우선순위가 더 높은 규칙이 존재할 가능성이 있으므로 입력패킷의 근원지 프리픽스의 다음 비트를 확인하여 0이라면 왼쪽 자식노드가, 1이라면 오른쪽 자식노드가 존재하는지 확인한다(S1612). 자식노드가 존재한다면, 자식 노드의 포인터가 가르치는 메모리의 주소로 이동하여 같은 과정을 반복한다(S1616).

각 노드마다 저장된 규칙 중 우선순위가 가장 높은 규칙이 먼저 비교되는데, 노드에 저장된 규칙 중 우선순위에 의해 저장된 규칙보다 현재의 최선 정합 규칙(BMR)이 우선순위가 높으면(S1602) 검색을 완전히 종료하고, 최선 정합 규칙(BMR)을

가장 우선순위가 높은 매치하는 규칙으로 판단한다(S1618). 노드에 저장된 규칙의 우선순위가 최선 정합 규칙(BMR)보다 더 높으면(S1602) 검색을 진행하고(S1604), 현재의 최선 정합 규칙(BMR)보다 우선순위가 더 높은 매치하는 규칙을 발견하면(S1604) 최선 정합 규칙(BMR)을 업데이트 한다(S1606).

검색이 중간에 중단되지 않고 리프 노드까지 검색을 하고 나면(S1612), 현재 기억된 최선 정합 규칙(BMR)을 매치하는 규칙으로 판단하고 검색을 종료한다(S1618).

이상의 검색 방법을 정리하면, 저장된 트리를 검색하기 위하여, 플로우 조건에 매치하는 규칙을 찾기 위하여 영역분할 검색을 수행하고, 영역분할 검색 중 만약 하나의 노드에 연결 리스트로 연결된 복수의 규칙들이 저장된 경우 플로우 조건에 매치하는 규칙을 찾기 위하여 연결된 복수의 규칙들에 대하여 선형 검색을 수행하며, 검색결과 매치하는 규칙을 현재의 최선 정합 규칙(best matching rule: 이하 BMR이라 함)으로 저장하고, 매치하는 규칙이 우선순위에 의해 저장된 경우라면 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하며, 매치하는 규칙이 분할된 각 영역을 가로지르는 크로싱(crossing) 상태를 만족하여 저장된 경우라면 플로우 조건의 다음 비트를 확인하고 전술한 검색과정을 반복한다. 만약, 검색 결과 현재 저장된 최선 정합 규칙이 우선순위에 의하여 노드에 저장된 규칙보다 우선순위가 높은 경우 현재의 최선 정합 규칙(BMR)을 출력하고 검색을 종료하며, 저장된 트리의 리프(leaf) 노드에 도달하면 현재까지 저장된 최선 정합 규칙(BMR) 중 우선순위가 가장 높은 규칙을 출력하고 검색을 종료한다.

4) 분류테이블의 갱신(Update)

본 발명에 따른 패킷 분류 방법은 분류테이블의 부가적 갱신(incremental update)을 지원한다. 먼저 이미 존재하는 규칙을 삭제하고자 하는 경우, 규칙을 검색하여 자식노드로 가는 포인터와 연결리스트 포인터를 제외한 다른 필드들을 삭제한다. 이는 자식노드와 연결리스트는 현재의 규칙이 삭제되어도 사용되어야 하기 때문이다. 많은 규칙을 삭제한 경우 비어있는 엔트리가 증가할 것이나, 일정한 시간 간격으로 전체 분류테이블이 갱신됨을 고려할 때 문제가 되지 않는다.

다음은 새로운 규칙을 추가하고자 하는 경우이다. 트리 메모리를 검색하여, 추가하고자 하는 규칙이 저장되어야 하는 노드를 찾는다. 추가하려는 규칙이 우선순위에 의하여 해당 노드에 저장되어야 한다면, 해당 노드에 저장되어 있던 기존의 규칙 중 우선순위에 의하여 저장되었던 규칙은 트리 메모리의 비어있는 엔트리로 이동하고, 추가하고자 하는 규칙은 이 엔트리를 대체하며, 연결리스트에 의하여 이동된 규칙을 가리킨다.

만약 추가하려는 규칙이 크로싱 상태를 만족하기 때문에 해당 노드에 저장되어야 한다면, 트리 메모리의 비어있는 엔트리에 추가되고, 해당 노드에 속하는 규칙들의 연결 리스트들을 우선순위에 의하여 재조정한다. 즉 본 발명에 따른 패킷 분류 방법에 따르면 규칙을 삭제하거나 추가하려는 경우, 삭제 혹은 추가하려는 규칙의 영향을 하나의 노드로 국한시켜 효율적으로 분류 테이블을 업데이트할 수 있다.

5) 실험결과 및 성능 비교(Simulation Result and Performance Evaluation)

기존의 선형검색구조 및 AQT구조와, 본 발명에 따른 우선순위에 기초한 영역분할 구조를 이용한 패킷분류방법을 C 언어를 사용하여 각각 구현하고, 실험을 통해 성능을 비교해 보았다. 성능 실험을 위한 분류테이블로 실제 라우터에서 사용하는 주소검색테이블로부터 랜덤하게 근원지 프리픽스를 추출하여 일차원 패킷분류를 실험하였고, 근원지 프리픽스와 목적지 프리픽스 쌍을 추출하여 이차원 패킷분류구조를 실험하였다. 그리고 각각의 규칙에 매치하는 패킷들이 동일한 확률로 들어온다고 가정하고, 검색을 위한 입력 패킷으로 각 규칙에 매치하는 정보들을 갖는 패킷을 사용하였다. 규칙의 개수는 50개부터 10000개까지 5가지 경우로 증가시키면서 실험하였다. 검색에 걸리는 시간은 메모리 접근횟수와 매우 밀접한 관계를 가지므로, 주어진 규칙을 검색하는데 소요된 메모리 접근 횟수를 비교하였다.

[표 3]
규칙 검색을 위한 최대 메모리 접근횟수와 평균 메모리접근 횟수 비교

Number of Rules	선형검색		AQT				본 발명에 따른 구조			
			일차원 이진트리		이차원 사분트리		일차원 이진트리		이차원 사분트리	
	최대 횟수	평균 횟수	최대 횟수	평균 횟수	최대 횟수	평균 횟수	최대 횟수	평균 횟수	최대 횟수	평균 횟수
50	50	25.5	32	27.26	31	25.32	10	5.98	8	4.28
100	100	50.5	32	27.32	31	25.24	14	7.42	8	4.94

1000	1000	500.5	32	27.21	32	25.12	18	11.10	12	7.21
5000	5000	2500.5	32	27.29	32	25.28	23	13.94	14	8.48
10000	10000	5000.5	33	27.34	32	25.29	25	15.25	15	9.04

표 3은 검색 과정에서의 최대 메모리 접근횟수와 평균 메모리 접근횟수를 나타낸 것이다. 선형검색의 경우 최대 규칙 개수 만큼의 메모리 접근 횟수를 갖으며, 하나의 입력이 여러 개의 규칙과 일치할 수 있고, 이 경우 가장 우선순위가 높은 규칙이 선택되므로, 테이블에 나타난 바와 같은 평균 검색횟수를 갖는다. AQT의 경우 규칙이 두 프리픽스의 길이 중 짧은 것의 길이에 해당하는 깊이를 가진 노드에 저장되기 때문에 규칙의 개수에 관계없이 트리의 깊이가 크게 나타났다. AQT의 경우 프리픽스의 길이가 긴 규칙들이 트리의 아랫부분에 저장되어 있기 때문에, 루트 노드부터 매치하는 규칙이 저장된 노드까지 한 레벨씩 내려가면서 계속적으로 메모리에 접근하여야 할 뿐 아니라, 매치하는 규칙을 발견하였다 하더라도, 하위 노드에 우선순위가 더 높은 규칙과 일치할 가능성이 있으므로, 트리의 리프를 만날 때까지 검색을 진행하여야 하므로, 평균 메모리 접근 횟수가 크게 나타난 것을 알 수 있다.

한편, 본 발명에 따른 우선순위에 기초하여 영역분할을 이용한 패킷 분류 방법에 의하면, 트리에 빈 노드가 없이 규칙이 저장되기 때문에 규칙의 개수가 증가하여도 트리의 깊이가 깊어지지 않는 것을 확인할 수 있다. 본 발명에 따른 트리 구조의 경우에는 프리픽스의 길이보다는 규칙의 우선순위에 따라 규칙이 저장될 가능성이 크고, 우선순위에 의하여 저장된 규칙과 매치하거나, 현재의 최선 정합 규칙(BMR)이 우선순위에 의하여 저장된 규칙보다 우선순위가 높은 경우 리프노드가 아니어도 검색을 완전히 종료하므로, 평균 메모리 접근 횟수가 매우 적게 나타났다.

【표 4】

규칙 테이블 저장을 위해 요구되는 메모리 크기 비교 (Kbyte)

Number of Rules	선형검색 (21byte width)	AQT		본 발명에 따른 구조	
		일차원 이진트리 (27byte width)	이차원 사분트리 (32byte width)	일차원 이진트리 (27byte width)	이차원 사분트리 (32byte width)
50	1.05	27.6	33.8	1.35	1.6
100	2.1	49.8	65.6	2.7	3.2
1000	21.0	394.5	584.8	27	32.0
5000	105.0	1495.9	2753.7	135	160.0

표 4는 규칙 테이블을 저장하기 위해 필요한 메모리의 크기를 비교한 결과를 나타낸 것이다. 선형검색의 경우 자식노드로 가는 포인터가 전혀 필요하지 않으므로 메모리 사용량이 가장 적은 것을 알 수 있다. 그러나 AQT의 경우 규칙이 저장되어 있지는 않으나 자식노드로 가는 포인터를 저장하는 빈 노드가 존재하기 때문에 매우 큰 메모리를 요구하는 것을 알 수 있다.

한편, 본 발명에 따른 구조는 선형검색구조에 비하여 약 30%의 메모리를 더 사용하여, 선형검색구조나 AQT 구조에 비해 매우 빠른 검색 성능을 보이며, AQT에 비하여는 매우 작은 메모리 크기를 사용하고 있어 매우 우수한 구조라고 할 수 있다.

정리하면, 본 발명에 따르면 기존의 선형검색 방식과 영역분할 방식의 단점을 보완한 새로운 패킷분류 기법이 제공된다. 기존의 선형검색 방식에서는 매 테이블 록업 당 검색영역이 선형적으로 감소하는 단점을 갖는데, 본 발명에 따른 구조에서는 검색영역을 지수적으로 줄이는 방법으로서 재귀적 영역분할 방식을 이용하였다. 특히, 규칙을 저장할 때 크로싱 상태를 만족하지 않아도 노드에 해당하는 영역에 속하는 규칙 중 우선순위가 가장 높은 규칙을 저장하여 트리의 빈 노드를 완전히 없애고 트리의 깊이를 줄여 검색속도를 향상시키고 소요되는 메모리의 크기도 줄인 구조이다. 본 발명에 따라 저장된 분류 테이블을 예로 실험한 결과 본 발명에 따른 방식은 패킷분류 속도나 소요되는 메모리, 분류테이블 갱신의 용이성, 확장성의 다양한 측면에서 매우 우수한 결과를 나타냄을 확인할 수 있었다.

전술한 본 발명에 따른 패킷 분류 방법은 패킷 중계 장치 내에서 하드웨어적으로 구현될 수 있다. 패킷 중계 장치에서의 패킷 분류는 입력된 모든 패킷에 대하여 실시간으로 처리되어야 하므로, 실시간으로 패킷 분류를 수행하기 위하여 본 발명을 하드웨어 구조로 설계 구현하는 것이 적절하다.

한편, 본 발명에 따른 패킷 분류 방법은 컴퓨터 프로그램으로도 작성 가능하다. 상기 프로그램을 구성하는 코드들 및 코드 세그먼트들은 당해 분야의 컴퓨터 프로그래머에 의하여 용이하게 추론될 수 있다. 또한, 상기 프로그램은 컴퓨터가 읽을 수 있는 정보저장매체(computer readable media)에 저장되고, 컴퓨터에 의하여 읽혀지고 실행됨으로써 패킷 분류 방법을 구현한다. 상기 정보저장매체는 자기 기록매체, 광 기록매체, 및 캐리어 웨이브 매체를 포함한다.

발명의 효과

전술한 바와 같이 본 발명에 따르면, 검색 시간을 줄일 수 있을 뿐만 아니라 소요되는 메모리의 크기도 줄일 수 있는 효율적인 패킷 분류 방법 및 이를 적용한 패킷 중계 장치가 제공된다.

기존의 패킷 분류 방법들은 빠른 검색 속도와 적은 메모리 소요라는 요구를 동시에 만족시키지 못하는 제한이 있었다. 그러나, 본 발명에 따른 패킷 분류 방법은 선형검색구조에 재귀적 영역분할방법을 응용함으로써, 선형검색구조에서의 단점인 검색속도를 향상시키고 재귀적 영역분할방법의 단점인 요구되는 메모리 크기를 개선시킬 수 있다.

즉, 선형검색구조의 단점을 보완하기 위해 본 발명에 따른 패킷 분류 방법은 검색속도를 향상시키기 위하여 재귀적 영역분할 방법을 응용함으로써 검색범위를 지수적으로 줄일 수 있다. 또한, 재귀적 영역분할 방식의 단점을 보완하기 위해 본 발명에 따른 패킷 분류 방법은 이진트리 혹은 사분트리의 노드에 규칙을 저장할 때 크로싱(crossing) 상태를 만족하지 않아도 노드에 해당하는 영역에 속하는 규칙 중 우선순위가 가장 높은 규칙을 저장하도록 함으로써 트리의 빈 노드를 완전히 없애고 트리의 깊이를 줄여 검색속도를 향상시키며, 소요되는 메모리의 크기도 줄일 수 있다.

실제 라우터의 주소검색테이블에서 랜덤하게 생성된 규칙 10000개를 갖는 분류테이블에 대해 본 발명에 따른 패킷 분류 방법을 적용한 실험 결과, 선형검색구조에 비하여 약 30%의 메모리를 더 사용하나, 최대 혹은 평균 메모리 접근횟수를 획기적으로 줄임으로써 검색시간을 훨씬 단축할 수 있었다. 또한, 재귀적 영역분할 방식과 비교하면, 사용되는 메모리의 측면이나 검색속도 모두를 획기적으로 개선할 수 있었다. 나아가, 본 발명에 따른 패킷 분류 방법은 분류테이블의 부가적인 갱신을 지원함으로써 분류테이블이 자주 변경되는 경우에도 업데이트가 용이할 뿐 아니라, 분류테이블의 확장성도 매우 뛰어남을 확인할 수 있다.

이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

도면의 간단한 설명

도 1은 근원지 프리픽스를 이용한 영역분할의 일 예,

도 2는 종래의 근원지 프리픽스를 이용한 이진트리 구조의 일 예,

도 3은 근원지 프리픽스와 목적지 프리픽스를 이용한 이차원 영역분할의 일 예,

도 4는 종래의 근원지 프리픽스와 목적지 프리픽스를 이용한 AQT 사분트리 구조의 일 예,

도 5는 본 발명의 일 실시예에 따른 패킷 중계 장치의 블록도,

도 6은 도 5에 도시된 인터페이스의 상세 블록도,

도 7은 본 발명에 따른 근원지 프리픽스를 이용한 이진트리 구조의 일 예,

도 8은 본 발명에 따른 근원지 프리픽스와 목적지 프리픽스를 이용한 사분트리 구조의 일 예,

도 9는 본 발명의 일 실시예에 따른 패킷 분류 방법을 도시한 흐름도,

도 10은 본 발명의 일 실시예에 따른 검색 방법을 도시한 흐름도이다.

<도면의 주요부분에 대한 부호의 설명>

10: 인터페이스 20: 스위치 패브릭

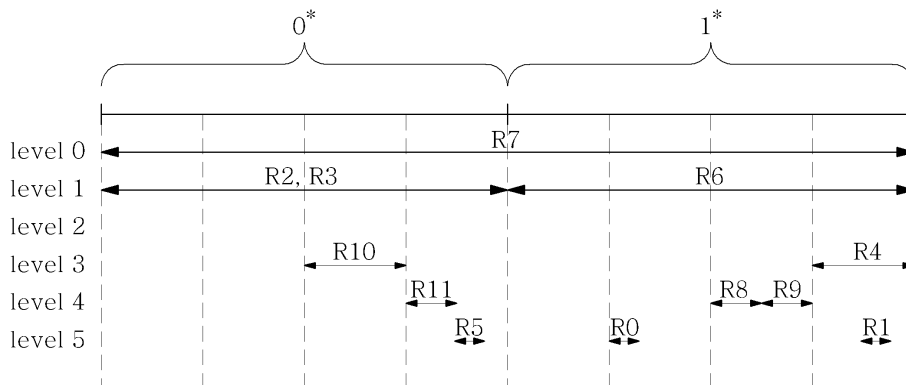
11: 패킷 송수신부 12: 프레임 해석부

13: 어드레스 룩업 처리부 14: 패킷 분류 처리부

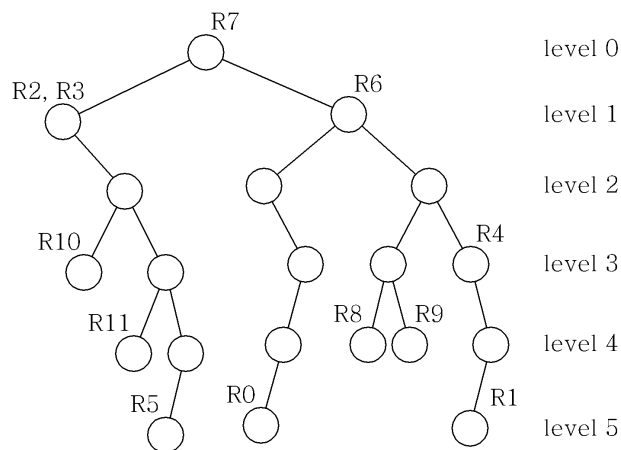
15: 전송 제어부

도면

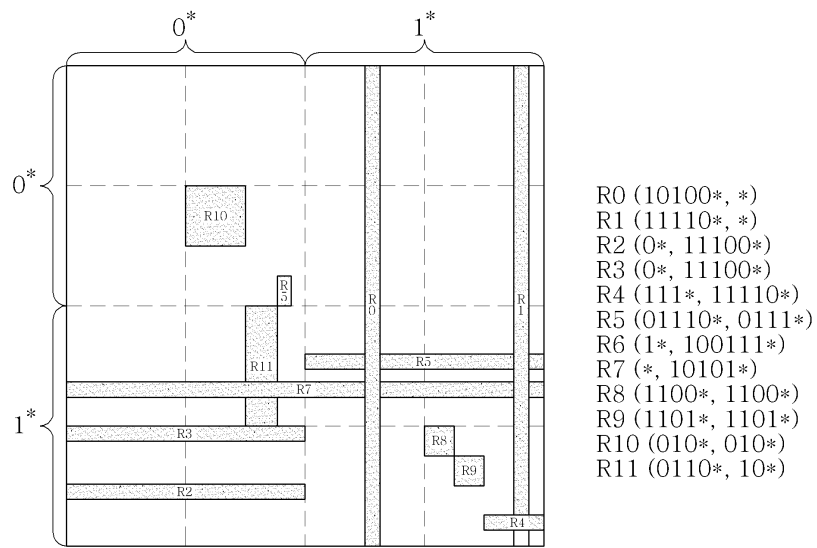
도면1



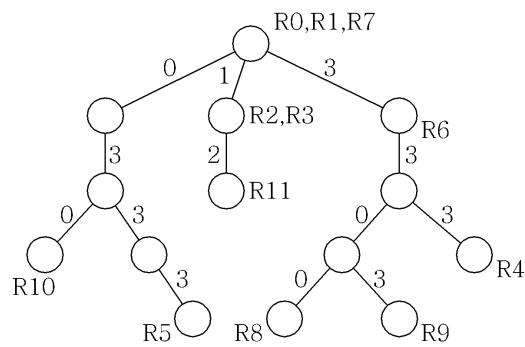
도면2



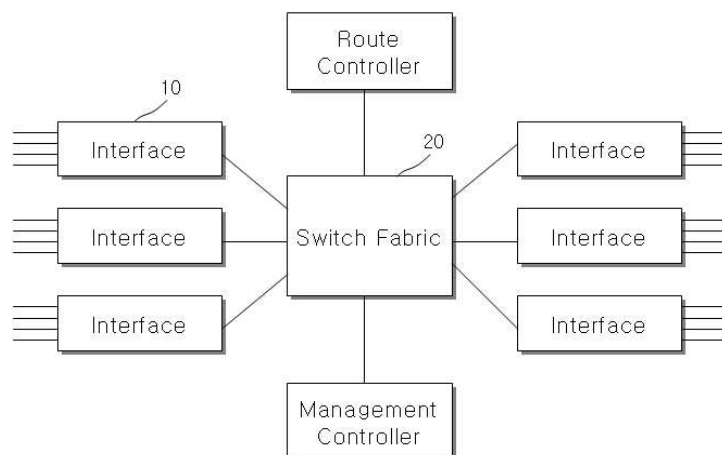
도면3



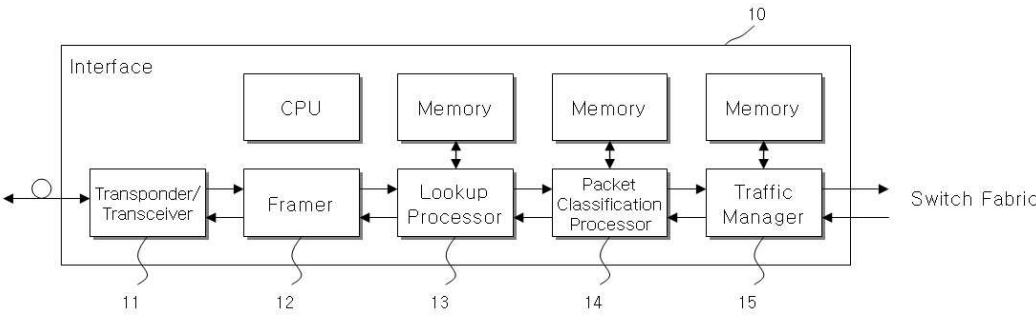
도면4



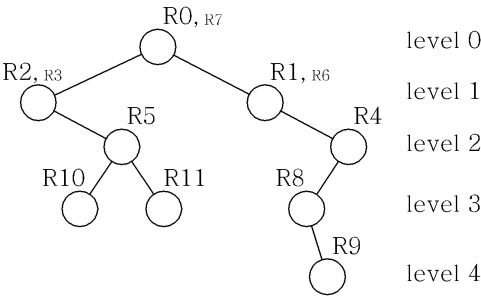
도면5



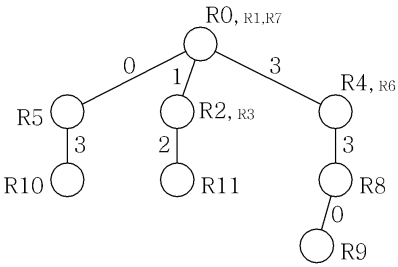
도면6



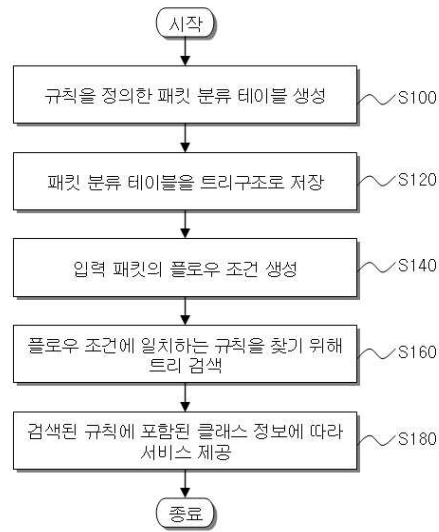
도면7



도면8



도면9



도면10

