



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년03월14일
(11) 등록번호 10-0814077
(24) 등록일자 2008년03월10일

(51) Int. Cl.

H04L 12/28 (2006.01) H04L 29/06 (2006.01)

H04L 12/56 (2006.01)

(21) 출원번호 10-2006-0098166

(22) 출원일자 2006년10월09일

심사청구일자 2006년10월09일

(56) 선행기술조사문헌

KR1020010098110 A

(뒷면에 계속)

전체 청구항 수 : 총 13 항

(73) 특허권자

이화여자대학교 산학협력단

서울 서대문구 대현동 11-1 이화여자대학교내

(72) 발명자

임혜숙

서울특별시 강남구 도곡동 91-5 도곡동삼성래미안
아파트 106-1301

(74) 대리인

유정열, 이현수, 정영미, 천성훈

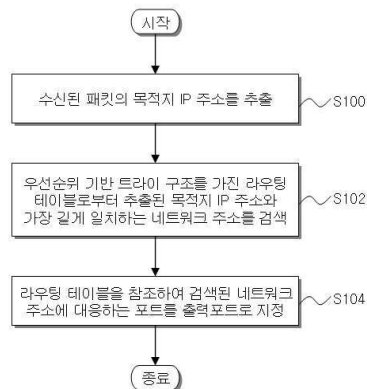
심사관 : 김병성

(54) 우선순위 트라이를 이용한 I P 주소 검색 방법 및 이를제공하는 패킷 중계 장치

(57) 요약

본 발명은 패킷 포워딩을 위한 IP 주소 검색 방법에 관한 것으로, 본 발명에 따른 IP 주소 검색 방법은, 수신된 패킷의 목적지 IP 주소를 추출하는 단계와, 네트워크 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 추출된 주소와 가장 길게 일치하는 네트워크 주소를 검색하는 단계와, 검색된 네트워크 주소에 대응하는 출력 포트를 수신된 패킷을 전송할 출력 포트로서 지정하는 단계를 포함하며, 상기 검색하는 단계는 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조를 이용하여 이루어지는 것을 특징으로 한다. 여기서, 우선순위 트라이 구조는 길이가 긴 프리픽스 일수록 먼저 비교되도록 일반 트라이 구조의 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당함으로써 구축되는 것이 바람직하다. 이에 따라, 검색 시간이 적고 소요되는 메모리의 크기도 줄일 수 있는 효과적인 IP 주소 검색 방법을 제공할 수 있다.

대표도 - 도5



(56) 선행기술조사문헌

KR1020030044506 A

KR1020040003258 A

KR1020050036457 A

US6237061 B1

특허청구의 범위

청구항 1

패킷 포워딩을 위한 IP 주소 검색 방법에 있어서,

- (a) 수신된 패킷의 목적지 IP 주소를 추출하는 단계;
- (b) 네트워크 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소를 검색하는 단계; 및
- (c) 상기 검색된 네트워크 주소에 대응하는 출력 포트를 상기 수신된 패킷을 전송할 출력 포트에 지정하는 단계를 포함하며,

상기 검색하는 단계는 상기 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조를 이용하여 이루어지는 것을 특징으로 하는 IP 주소 검색 방법.

청구항 2

제1항에 있어서,

상기 우선순위 트라이 구조는 길이가 긴 프리픽스 일수록 먼저 비교되도록 일반 트라이 구조의 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당함으로써 구축되는 것을 특징으로 하는 IP 주소 검색 방법.

청구항 3

제1항에 있어서,

상기 우선순위 트라이 구조는 일반 트라이 구조의 각각의 빈 내부 노드를 해당 빈 내부 노드가 루트 노드인 서브 트리에 속하는 프리픽스들 중 가장 긴 프리픽스로 대체함으로써 구축되는 것을 특징으로 하는 IP 주소 검색 방법.

청구항 4

제1항에 있어서, 상기 (b) 단계는,

- (b1) 상기 추출된 목적지 IP 주소를 상기 우선순위 트라이 구조의 각 노드들과 비교하는 단계;
- (b2) 비교결과 상기 추출된 목적지 IP 주소가 일반 노드와 일치하는 경우에는 상기 일치하는 일반 노드의 프리픽스를 현재까지 일치된 가장 긴 프리픽스로서 저장하고 검색을 계속하며,
- (b3) 비교결과 상기 추출된 목적지 IP 주소가 상기 우선순위 트라이 구조에서 빈 노드에 대체된 우선순위 프리픽스와 일치하거나 또는 리프(leaf)를 만나면 검색을 종료하는 단계; 및
- (b4) 상기 우선순위 프리픽스와 일치하여 검색이 종료된 경우에는 해당 우선순위 프리픽스를 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소로 반환하고, 상기 리프를 만나 종료된 경우에는 상기 저장된 현재까지 일치된 가장 긴 프리픽스를 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소로 반환하는 단계를 포함하는 것을 특징으로 하는 IP 주소 검색 방법.

청구항 5

IP 주소 검색을 위한 라우팅 테이블을 구축하는 방법에 있어서,

- (a) 네트워크 주소를 나타내는 프리픽스들을 각각 자신의 길이에 해당하는 레벨의 노드들로서 저장하여 일반 트라이 구조를 구축하는 단계; 및
- (b) 상기 일반 트라이 구조로부터 길이가 긴 프리픽스 일수록 먼저 비교되도록 구조된 우선순위 기반 트라이 구조를 재구축하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 구축 방법.

청구항 6

제5항에 있어서, 상기 (b) 단계는,

상기 (a) 단계에서 구축된 일반 트라이 구조의 경로에서 첫 번째로 만난 빈 노드에 가장 길이가 긴 프리픽스를 저장하고 상기 빈 노드에 저장된 프리픽스를 우선순위 프리픽스로 마크하는 과정을 상기 일반 트라이 구조에 빈 노드가 없어질 때까지 반복하는 것을 특징으로 하는 라우팅 테이블 구축 방법.

청구항 7

IP 주소 검색을 위한 라우팅 테이블을 업데이트하는 방법에 있어서,

(a) 상기 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조에서 하나의 프리픽스를 삭제하는 경우 상기 삭제될 프리픽스를 빈 내부 노드로 만드는 단계; 및

(b) 상기 빈 내부 노드를 제거하기 위하여 소정의 주기로 상기 라우팅 테이블에 대하여 우선순위 기반 트라이 구조를 재구축하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 업데이트 방법.

청구항 8

제7항에 있어서, 상기 (b) 단계는,

상기 라우팅 테이블에 포함된 길이가 긴 프리픽스 일수록 먼저 비교되도록 상기 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당하여 우선순위 기반 트라이 구조를 재구축하는 것을 특징으로 하는 라우팅 테이블 업데이트 방법.

청구항 9

IP 주소 검색을 위한 라우팅 테이블을 업데이트하는 방법에 있어서,

(a) 상기 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조에서 하나의 프리픽스를 삽입하는 경우, 상기 삽입될 프리픽스가 우선순위 프리픽스와 일치하면서 그 길이가 상기 우선순위 프리픽스의 길이보다 긴 경우 상기 삽입될 프리픽스가 상기 우선순위 프리픽스의 위치를 차지하도록 배치하는 단계; 및

(b) 상기 삽입될 프리픽스의 위치를 이미 다른 우선순위 프리픽스가 차지하고 있는 경우 상기 삽입될 프리픽스가 상기 다른 우선순위 프리픽스의 위치를 차지하도록 배치하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 업데이트 방법.

청구항 10

IP 주소 검색을 위한 패킷 중계 장치에 있어서,

수신된 패킷의 목적지 IP 주소를 추출하는 프레임 해석부; 및

네트워크 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소를 검색하고 상기 검색된 네트워크 주소에 대응하는 출력 포트를 상기 수신된 패킷을 전송할 출력 포트에 지정하는 어드레스 룩업 처리부를 포함하며,

상기 어드레스 룩업 처리부는 상기 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조를 이용하여 상기 네트워크 주소 검색을 수행하는 것을 특징으로 하는 패킷 중계 장치.

청구항 11

제10항에 있어서,

상기 우선순위 트라이 구조는 길이가 긴 프리픽스 일수록 먼저 비교되도록 일반 트라이 구조의 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당함으로써 구축되는 것을 특징으로 하는 패킷 중계 장치.

청구항 12

제10항에 있어서,

상기 우선순위 트라이 구조는 일반 트라이 구조의 각각의 빈 내부 노드를 해당 빈 내부 노드가 루트 노드인 서브 트리에 속하는 프리픽스들 중 가장 긴 프리픽스로 대체하여 구축되는 것을 특징으로 하는 패킷 중계 장치.

청구항 13

제10항에 있어서, 상기 어드레스 룩업 처리부는,

상기 추출된 목적지 IP 주소를 상기 우선순위 트라이 구조의 각 노드들과 비교하여 상기 추출된 목적지 IP 주소가 일반 노드와 일치하는 경우에는 상기 일치하는 일반 노드의 프리픽스를 현재까지 일치된 가장 긴 프리픽스로서 저장하고 검색을 계속하고 상기 추출된 목적지 IP 주소가 상기 우선순위 트라이 구조에서 빈 노드에 대체된 우선순위 프리픽스와 일치하거나 또는 리프(leaf)를 만나면 검색을 종료하되,

상기 우선순위 프리픽스와 일치하여 검색이 종료된 경우에는 해당 우선순위 프리픽스를 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소로 반환하고, 상기 리프를 만나 종료된 경우에는 상기 저장된 현재까지 일치된 가장 긴 프리픽스를 상기 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소로 반환하는 것을 특징으로 하는 패킷 중계 장치.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

- <16> 본 발명은 패킷 중계 장치에서의 IP 주소 검색(IP address lookup) 방법에 관한 것으로, 우선순위 트라이를 이용하여 효과적으로 IP 주소를 검색하는 방법을 제안한다.
- <17> 최근 인터넷 사용의 폭발적인 증가에 따라 인터넷 트래픽이 빠르게 증가하고 있다. 이에 따라 라우터(router)와 같은 패킷 중계 장치가 고속으로 패킷 포워딩(packet forwarding)을 수행할 것이 요구된다. 고속의 패킷 포워딩을 위해서는 라우터의 IP 주소 검색 효율을 개선하는 것이 필수적이다. 패킷들의 도착 비율과 라우팅 테이블의 크기가 급격히 증가하는 환경 하에서 입력되는 각 패킷이 회선 속도(wire-speed)로 처리될 것이 요구되기 때문이다.
- <18> 여기서 IP 주소 검색(IP address lookup)이란, 입력되는 패킷이 최종 목적지에 포워딩 되도록 수신된 패킷의 목적지 IP 주소(인터넷 프로토콜 주소)를 이용하여 출력 포트를 결정하는 것을 의미한다. IP 주소는 네트워크 부분과 호스트 부분의 2 단계 계층구조를 가지며, 이중 네트워크 부분을 프리픽스(prefix)라고 부른다. 이하에서는 IP 주소 중 네트워크 부분을 프리픽스로 통칭한다.
- <19> 종래에는 클래스별로 일정한 길이를 가지는 프리픽스를 사용하였으나 고정길이를 가진 클래스 구조로 인해 IP 주소가 낭비되는 문제점이 있었다. 이를 개선하여 최근에는 클래스 구분 없는 라우팅 구조(Classless inter-domain routing : 이하 CIDR 이라 칭함)를 사용하여 임의의 길이의 프리픽스와 주소의 결합을 허용하게 되었다. 따라서 IP 주소에서 프리픽스의 길이가 가변함에 따라 라우터가 IP 주소의 특정 네트워크 경로를 찾기 위해서는 라우팅 테이블에서 입력 패킷의 목적지 주소와 일치하는 가장 긴 프리픽스를 검색할 것이 요구된다. 이 때 IP 주소 검색을 위하여 가장 길게 일치하는 프리픽스(the longest matching prefix: 이하 LMP 라 약칭함) 또는 가장 잘 일치하는 프리픽스(the best matching prefix : 이하 BMP 라 약칭함)를 결정하는 것이 필요하며, 이를 위해 프리픽스의 길이와 값이라는 2 가지 요소를 모두 고려해야 한다.
- <20> CIDR 구조의 IP 주소 검색을 위해 가장 대표적으로 사용되는 종래 기술은 트라이(trie) 구조를 이용한 이진 검색(binary search) 방법이다. 트라이는 길이에 따라 선형검색을 적용하는 트리 기반의 데이터 구조를 가리킨다. CIDR 형태의 IP 주소에서 프리픽스의 길이와 값을 표현하기 위해 트라이 구조를 사용한다. 즉, 각 프리픽스는 트라이의 하나의 노드에 존재하며 해당 노드의 레벨은 프리픽스의 길이에 상당한다. 트라이 구조에서의 검색은 트라이의 루트 노드로부터 출발하여 입력된 패킷의 목적지 IP 주소의 MSB(Most Significant Bit)로부터 순차적으로 검사하여 검사된 비트가 0이면 왼쪽 자식 노드를 가르키는 포인터를 따라, 1이면 오른쪽 자식 노드를 가르키는 포인터를 따라 내려가며, 트라이의 리프를 만날 때 까지 같은 과정을 반복한다.
- <21> 한편, 도 1은 IP 주소 검색 방법을 설명하기 위한 프리픽스 집합의 일 예를 나타내고, 도 2는 상기 프리픽스 집합을 이용하여 종래의 이진 트라이 구조를 구축한 일 예를 도시한다.
- <22> 도 2에서 검은색으로 표시된 노드들은 프리픽스들을 나타내며, 흰색으로 표시된 노드들은 프리픽스가 지정되지

않은 빈 내부 노드들을 나타낸다. 예를 들어 도 1의 프리픽스 집합에서 P2인 프리픽스 1000*은 트라이 구조를 이용하면 도 2의 P2로 표시된 검은색 노드로 표현된다. P2 노드는 1000*이라는 값을 가지며 그 길이가 4이므로 루트 노드로부터 4번째 레벨에 위치한다.

<23> 이러한 트라이 구조의 첫 번째 문제점은, 프리픽스의 길이가 트라이의 레벨에 해당하므로 짧은 길이의 프리픽스들은 트라이의 상위 레벨에 저장되고 길이가 긴 프리픽스들은 트라이의 하위 레벨에 저장된다는 점이다. 즉, 짧은 프리픽스일수록 상위 레벨에 저장되어 길이가 더 긴 프리픽스보다 먼저 비교되므로, 검색시 일치하는 노드를 발견한 경우라도 리프 노드에 도달할 때까지 검색이 계속 수행되어야 한다. 중간에 일치하는 프리픽스 노드를 만나더라도 하위 레벨에 더 길게 일치하는 프리픽스가 존재할 가능성이 있기 때문이다. 따라서 매번 리프 노드까지 검색을 수행해야 하므로 메모리 액세스 횟수가 증가하고 검색 시간이 길어지는 문제점이 있다. 또한, 트라이에 빈 내부 노드들이 다수 포함됨으로써 메모리 액세스 횟수가 늘어 검색시간이 길어지며 메모리 공간을 낭비하는 문제점도 발생한다.

발명이 이루고자 하는 기술적 과제

<24> 따라서 전술한 문제점을 해결하기 위한 본 발명의 목적은 검색 시간과 요구되는 메모리 크기를 줄일 수 있는 효과적인 IP 주소 검색 방법을 제공하는 것이다.

<25> 또한, 본 발명의 다른 목적은 검색 시간과 요구되는 메모리 크기를 줄일 수 있는 효과적인 IP 주소 검색 방법을 적용한 패킷 중계 장치를 제공하는 것이다.

발명의 구성 및 작용

<26> 본 발명에 따라 전술한 목적은, 패킷 포워딩을 위한 IP 주소 검색 방법에 있어서 (a) 수신된 패킷의 목적지 IP 주소를 추출하는 단계; (b) 목적지 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 추출된 목적지 주소와 가장 길게 일치하는 네트워크 주소를 검색하는 단계; 및 (c) 검색된 네트워크 주소에 대응하는 출력 포트를 수신된 패킷을 전송할 출력 포트에 지정하는 단계를 포함하며, 검색하는 단계는 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조를 이용하여 이루어지는 것을 특징으로 하는 IP 주소 검색 방법에 의해 달성된다.

<27> 여기서, 우선순위 트라이 구조는 길이가 긴 프리픽스일수록 먼저 비교되도록 일반 트라이 구조의 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당함으로써 구축되는 것이 바람직하며,

<28> 또한, 우선순위 트라이 구조는 일반 트라이 구조의 각각의 빈 내부 노드를 해당 빈 내부 노드가 루트 노드인 서브 트리에 속하는 프리픽스들 중 가장 긴 프리픽스로 대체함으로써 구축되는 것이 특히 바람직하다.

<29> 나아가, 네트워크 주소를 검색하는 단계는 (b1) 추출된 목적지 주소를 우선순위 트라이 구조의 각 노드들과 비교하는 단계; (b2) 비교결과 추출된 목적지 주소가 일반 노드와 일치하는 경우에는 일치하는 일반 노드의 프리픽스를 현재까지 일치된 가장 긴 프리픽스로서 저장하고 검색을 계속하며, (b3) 비교결과 추출된 목적지 주소가 우선순위 트라이 구조에서 빈 노드에 대체된 우선순위 프리픽스와 일치하거나 또는 리프(leaf)를 만나면 검색을 종료하는 단계; 및 (b4) 우선순위 프리픽스와 일치하여 검색이 종료된 경우에는 해당 우선순위 프리픽스를 추출된 목적지 주소와 가장 길게 일치하는 프리픽스로 반환하고, 리프를 만나 종료된 경우에는 저장된 현재까지 일치된 가장 긴 프리픽스를 추출된 목적지 주소와 가장 길게 일치하는 프리픽스로 반환하는 단계를 포함하는 것이 바람직하다.

<30> 한편, 본 발명의 일 양상에 따르면 전술한 목적은, IP 주소 검색을 위한 라우팅 테이블을 구축하는 방법에 있어서 (a) 네트워크 주소를 나타내는 프리픽스들을 각각 자신의 길이에 해당하는 레벨의 노드들로서 저장하여 일반 트라이 구조를 구축하는 단계; 및 (b) 일반 트라이 구조로부터 길이가 긴 프리픽스일수록 먼저 비교되도록 구조된 우선순위 기반 트라이 구조를 재구축하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 구축 방법에 의해 달성된다.

<31> 한편, 본 발명의 일 양상에 따르면 전술한 목적은, IP 주소 검색을 위한 라우팅 테이블을 업데이트하는 방법에 있어서 (a) 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조에서 하나의 프리픽스를 삭제하는 경우 삭제될 노드를 빈 내부 노드로 만드는 단계; 및 (b) 빈 내부 노드를 제거하기 위하여 소정의 주기로 라우팅 테이블에 대하여 우선순위 기반 트라이 구조를 재구축하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 업데이트 방법에 의해 달성된다.

- <32> 한편, 본 발명의 다른 양상에 따르면 전술한 목적은, IP 주소 검색을 위한 라우팅 테이블을 업데이트하는 방법에 있어서 (a) 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조에서 하나의 프리픽스를 삽입하는 경우, 삽입될 프리픽스의 길이가 자신과 일치하는 우선순위 프리픽스의 길이보다 긴 경우 삽입될 프리픽스가 우선순위 프리픽스의 위치를 차지하도록 배치하는 단계; 및 (b) 삽입될 프리픽스의 일반적인 위치를 이미 다른 우선순위 프리픽스가 차지하고 있는 경우 삽입될 프리픽스가 다른 우선순위 프리픽스의 위치를 차지하도록 재배치하는 단계를 포함하는 것을 특징으로 하는 라우팅 테이블 업데이트 방법에 의해 달성된다.
- <33> 한편, 본 발명의 일 양상에 따르면 전술한 목적은, IP 주소 검색을 위한 패킷 중계 장치에 있어서, 수신된 패킷의 목적지 IP 주소를 추출하는 프레임 해석부; 및 네트워크 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소를 검색하고 검색된 네트워크 주소에 대응하는 출력 포트를 수신된 패킷을 전송할 출력 포트에 지정하는 어드레스 룩업 처리부를 포함하며, 어드레스 룩업 처리부는 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조를 이용하여 IP 주소 검색을 수행하는 것을 특징으로 하는 패킷 중계 장치에 의해 달성된다.
- <34> 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.
- <35> 먼저, 패킷 중계 장치의 구조를 살펴보고, 본 발명에 따른 IP 주소 검색 방법을 상세히 설명한다.
- <36> 도 3은 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.
- <37> 도 3을 참조하면, 본 발명에 따른 패킷 중계 장치는 복수의 인터페이스(10)와 스위치 패브릭(switch fabric)(20)을 구비한다. 그 밖에 관리 컨트롤러(management controller)나 루트 컨트롤러(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 IP 주소 검색 기능은 인터페이스(10)에서 담당한다.
- <38> 도 4는 상기 패킷 중계 장치에 포함된 인터페이스의 상세 블록도이다. 도 4를 참조하면, 인터페이스(10)는 패킷 송수신부(transponder/transceiver)(11), 프레임 해석부(framer)(12), 어드레스 룩업 처리부(lookup processor)(13) 및 전송 제어부(traffic manager)(15)를 구비한다. 그 밖에 패킷 분류 처리부(packet classification processor)(14), CPU, 메모리를 더 구비한다.
- <39> 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 분리하며 근원지 주소, 목적지 주소, 근원지 포트, 목적지 포트 등을 해석한다. 특히, 패킷 포워딩을 위해 필요한 목적지 IP 주소를 추출해 낸다.
- <40> 어드레스 룩업 처리부(lookup processor)(13)는 입력 포트를 통해 수신된 패킷의 목적지 IP 주소를 참조하여 상기 수신된 패킷을 전송할 출력 포트를 지정한다. 어드레스 룩업 처리부(13)는 네트워크 주소에 따른 출력 포트가 지정된 라우팅 테이블로부터 수신된 패킷에 포함된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소를 검색하며, 본 발명에 따른 우선순위 트라이(priority-trie) 구조를 이용하여 IP 주소 검색을 수행한다. 구체적인 우선순위 기반 트라이 구조는 후술한다.
- <41> 전송 제어부(traffic manager)(15)는 어드레스 룩업 처리부(13)에서 검색된 네트워크 주소에 대응하는 출력포트를 제어한다. 그 밖에 패킷 분류 처리부(packet classification processor)(14)는 패킷을 분류하는 규칙들이 저장된 패킷 분류 테이블로부터 수신된 패킷의 플로우 조건에 매치하는 규칙을 검색한다. 라우터와 같은 패킷 중계 장치에 패킷 분류 처리부를 포함할 것인지 여부는 설계자의 선택 사항이다.
- <42> 이상에서 살펴본 패킷 중계 장치의 구조에 기초하여 본 발명에 따른 우선순위 트라이를 이용한 IP 주소 검색 방법을 설명한다.

- <43> 도 5를 참조하면, 먼저 패킷 중계 장치로 수신된 패킷을 해석하여 목적지 IP 주소를 추출하고(S100), 후술하는 우선순위 기반 트라이 구조를 가진 라우팅 테이블로부터 추출된 목적지 IP 주소와 가장 길게 일치하는 네트워크 주소를 검색한다(S102). 라우팅 테이블은 네트워크 주소에 따른 출력 포트를 지정한 테이블로서, 각 엔트리들은 본 발명에 따른 우선순위 기반 트라이 구조로 저장되어 있다. 즉, 각 엔트리들은 길이가 긴 프리픽스 일수록 먼저 비교되도록 일반 트라이 구조의 빈 내부 노드를 이용하여 길이가 긴 프리픽스를 높은 레벨에 할당한 우선순위 기반 트라이 구조에 따라 저장된다. 이에 따라 IP 주소 검색 시에 트라이의 리프 노드까지 도달하기 전이라도 우선순위 프리픽스와 일치하는 노드를 발견하면 검색을 종료할 수 있다. 보다 구체적인 검색방법은 후술한다. 이제 라우팅 테이블을 참조하여 검색된 네트워크 주소에 대응하는 출력포트를 수신된 패킷을 출력할 포트로 지정한다(S104).
- <44> 이하, 본 발명에 따른 우선순위 트라이 구조를 보다 구체적으로 살펴본다.
- <45> 본 발명에서는 보다 긴 프리픽스를 보다 높은 레벨에 할당하기 위하여 일반 트라이에 존재하는 빈 내부 노드를 사용할 것을 제안한다. 즉, 빈 노드를 그 빈 노드가 루트인 서브 트리에 속하는 프리픽스들 중 가장 길이가 긴 프리픽스에 결합하는 것이다. 이에 따라 빈 내부 노드들은 모두 제거되며 트라이의 깊이는 감소하고 검색은 보다 효율적으로 수행된다. 이 때 빈 내부 노드에 결합된 프리픽스들을 우선순위 프리픽스(priority-prefix)라고 칭한다. 이와 같이 빈 내부 노드들을 그 자신의 길이보다 같거나 보다 길이가 긴 프리픽스들에 할당하기 때문에 본 발명에 따르면 프리픽스가 중복될 필요가 없다.
- <46> 도 6은 도 1에 도시된 프리픽스 집합을 이용하여 본 발명에 따른 우선순위 트라이 구조를 구축한 일 예이다. 도 6을 참조하면, 검은 색 노드들은 자신의 레벨에 배치된 일반 프리픽스(ordinary prefix)들을 나타내고, 흰색 노드들은 빈 내부 노드를 이용하여 보다 긴 프리픽스를 보다 높은 레벨에 할당한 우선순위 프리픽스(priority prefix)를 나타낸다.
- <47> 한편, 전술한 도 2에는 도 1에 도시된 동일한 프리픽스 집합을 이용하여 종래의 일반 트라이 구조를 구축한 예가 도시되어 있다. 먼저 도 2에 도시된 일반 트라이에서 가장 길이가 긴 프리픽스는 P4로 100110*이라는 값을 가지며 가장 하위 레벨에 저장되어 있다. 이를 본 발명에 따른 우선순위 트라이 구조로 재구축한 결과가 도 6에 도시되어 있다. 도 6을 참조하면, 이제 가장 하위 레벨에 저장되어 있던 프리픽스 P4가 빈 내부 노드인 루트 노드에 할당되며 우선순위 프리픽스가 된 것을 확인할 수 있다.
- <48> 이와 같이 빈 노드를 그 빈 노드가 루트인 서브 트리에 속하는 프리픽스들 중 가장 길이가 긴 프리픽스에 결합하는 방식으로 도 2에 도시된 일반 트라이를 재구축하여 우선순위 트라이를 구축하면 도 6과 같은 결과가 얻어진다. 즉, 본 발명에 따르면 길이가 긴 프리픽스 일수록 상위 레벨의 빈 노드에 저장된다. 도 2 및 도 6을 대비하여 보면 프리픽스 P0, P2, P3, P4, P5, P7이 빈 내부 노드에 할당되어 우선순위 프리픽스가 된 것을 알 수 있다. 여기서 동일한 길이를 가진 두 개의 프리픽스가 존재하는 경우 좌측 프리픽스를 우선하는 것으로 가정한다. 이에 따라 빈 내부 노드들이 모두 제거되고 트라이의 깊이가 감소하여 보다 효율적인 검색을 수행할 수 있다.
- <49> 도 7은 도 6에 도시된 우선순위 기반 트라이 구조를 가진 라우팅 테이블의 일 예가 도시되어 있다. 도 7을 참조하면, 첫 번째 열은 메모리 주소이며 임의의 값이 될 수 있다. 두 번째 열은 프리픽스가 우선순위 프리픽스(1)인지 아니면 일반 프리픽스(0)인지를 나타낸다. 프리픽스와 그 길이는 각각 세 번째 열과 네 번째 열에 저장된다. 다섯 번째 열과 여섯 번째 열은 현재 노드의 왼쪽 및 오른쪽에 있는 자식노드가 저장된 메모리 주소를 의미한다. 마지막 열은 프리픽스에 상응하는 출력 포트이며, 본 예에서는 설명의 편의를 위해 간단히 프리픽스 이름을 저장하였다. 도시된 바와 같이 제안된 알고리즘에서 라우팅 테이블의 엔트리 수는 N으로서, N개의 프리픽스를 저장하기 위하여 N개의 엔트리를 사용하였다. 이는 내부에 빈 노드가 하나도 존재하지 않음을 나타낸다.
- <50> 도 8은 종래의 이진 트라이 구조와 본 발명에 따른 우선순위 트라이 구조의 메모리 액세스 횟수를 비교한 도면이다.
- <51> 도 8을 참조하면, 도 2에 도시된 일반 트라이 구조의 메모리 액세스 횟수에 비하여 도 6에 도시된 우선순위 트라이 구조의 메모리 액세스 횟수가 훨씬 줄어든 것을 확인할 수 있다. 특히, 우선순위 프리픽스로 마크된 프리픽스 P0, P2, P3, P4, P5, P7이 상위 레벨의 빈 내부 노드들에 할당됨에 따라 메모리 액세스 횟수가 상당히 줄어든 것을 알 수 있다. 이와 같이 본 발명에 따른 우선순위 트라이 구조를 이용하면 빈 내부 노드들이 모두 제거되고 트라이의 깊이가 감소하여 메모리 공간을 절약할 뿐 아니라 메모리 액세스 횟수를 줄일 수 있어 보다 효율적인 검색을 수행할 수 있다.

- <52> 도 9 내지 도 11은 본 발명에 따른 우선순위 트라이를 구축하는 방법, 검색하는 방법, 및 업데이트 하는 방법을 구현한 의사 코드(pseudo code)의 일 예이다.
- <53> 도 9를 참조하면, 본 발명에 따른 우선순위 기반의 트라이를 구축하는 방법은 첫째 프리픽스의 길이가 노드의 레벨에 해당하는 일반 이진 트라이를 구축하는 과정(BuildBinaryTrie), 둘째 보다 긴 프리픽스를 보다 높은 레벨에 할당하기 위하여 일반 이진 트라이에 존재하는 빈 내부 노드를 사용하여 우선순위 트라이를 재구축하는 과정(BuildPriorityTrie)으로 이루어진다.
- <54> 여기서 우선순위 트라이를 재구축하는 과정을 도 9를 참조하여 상세히 살펴보면, 먼저 일반 이진 트라이를 검색(traverse)하여 만약 빈 노드가 아닌 노드(non-empty node)를 발견하면 일반 노드로 마크하고, 만약 빈 노드(empty node)를 발견하면 그 빈 노드가 루트인 서브 트리, 즉 $E(x)$ 에 속하는 프리픽스들을 정렬하고 그 정렬된 프리픽스들 중 가장 길이가 긴 프리픽스, 즉 $P(x)$ 를 트라이에서 제거한 후에 $P(x)$ 를 그 빈 노드에 저장한다. 이제 $P(x)$ 가 저장된 노드는 우선순위 노드(priority node)로 마크된다. 이 때 $P(x)$ 는 현재 자신의 프리픽스 길이보다 높은 레벨로 이동하여야 함에 주의한다. 이와 같은 과정을 일반 이진 트라이의 모든 노드를 검색하거나 또는 일반 이진 트라이에 더 이상 빈 노드가 포함되어 있지 않을 때까지 반복한다. 우선순위 트라이 재구축 과정이 완료되면 빈 노드가 모두 제거되기 때문에 트라이에 포함된 노드의 수는 프리픽스들의 수와 같아지게 된다.
- <55> 한편, 도 10은 본 발명에 따른 우선순위 트라이를 검색하는 방법을 구현한 의사 코드의 일 예가 도시된다.
- <56> 도 10을 참조하면, 먼저 루트 노드에서 시작하여 입력된 IP 주소를 우선순위 트라이에 저장된 노드들과 비교한다. 비교결과 입력된 주소가 일반 노드와 일치하는 경우에는 일치하는 일반 노드의 프리픽스를 현재까지 일치된 가장 긴 프리픽스로서 저장하고 검색을 계속한다.
- <57> 비교결과 추출된 주소가 우선순위 트라이 구조에서 빈 노드에 대체된 우선순위 프리픽스와 일치하거나 또는 리프(leaf)를 만나면 검색을 종료한다.
- <58> 이 때, 우선순위 프리픽스와 일치하여 검색이 종료된 경우에는 해당 우선순위 프리픽스를 상기 추출된 주소와 가장 길게 일치하는 네트워크 주소로 반환하면 된다. 본 발명에 따른 우선순위 기반 트라이는 길이가 긴 프리픽스가 보다 상위 레벨에 저장되도록 재구축되어 있기 때문에, 검색 중 우선순위 프리픽스와 일치하였다는 것은 해당 서브 트리에서 가장 길게 일치하는 노드임을 보장하게 되며, 따라서 더 이상 하위 노드들과 비교할 필요가 없어지기 때문이다. 이에 따라 검색을 위한 메모리 액세스 횟수를 획기적으로 줄일 수 있게 된다.
- <59> 한편, 리프를 만나 종료된 경우에는 현재까지 일치된 가장 긴 프리픽스를 상기 추출된 주소와 가장 길게 일치하는 네트워크 주소로 반환하면 된다.
- <60> 한편, 도 11은 본 발명에 따른 우선순위 트라이를 업데이트하는 방법을 구현한 의사 코드의 일 예가 도시된다.
- <61> 참고로, 도시되지는 않았지만, 본 발명에 따른 우선순위 트라이에서 하나의 프리픽스를 삭제하기 위하여 삭제될 노드를 빈 내부 노드로 만들고, 빈 내부 노드를 제거하기 위하여 소정의 주기로 상기 라우팅 테이블에 대하여 우선순위 기반 트라이 구조를 재구축하는 것이 바람직하다. 구체적으로는 삭제할 프리픽스를 찾아 도 7에 도시된 라우팅 테이블의 엔트리 중 처음 세 필드, 즉 메모리 주소, 일반 노드인지 우선순위 노드인지를 나타내는 마크 정보 및 해당 프리픽스 값을 삭제하여 빈 노드로 만든다. 이 노드는 하위 레벨까지 검색을 계속할 수 있도록 여전히 자식 노드에 대한 포인터를 저장하고 있기 때문에 그 노드를 유지해야 한다. 만약 적절한 주기로 전체 라우팅 테이블을 재구축한다고 가정하면 프리픽스를 삭제함에 따라 빈 노드의 숫자가 문제가 되진 않을 것이다.
- <62> 한편, 제안된 알고리즘에서 하나의 프리픽스를 삽입하기 위하여 도 11에 도시된 업데이트 절차를 수행한다.
- <63> 도 11을 참조하면, 라우팅 테이블에 포함된 네트워크 주소들에 기초하여 생성된 우선순위 트라이(priority-trie) 구조에서 하나의 프리픽스를 삽입하는 경우, 삽입에 의해 복수의 노드들이 영향을 받는 두 가지 경우가 있다. 첫 번째는 삽입될 프리픽스가 우선순위 프리픽스와 일치하면서 그 길이가 우선순위 프리픽스보다 긴 경우이다. 두 번째는 삽입될 프리픽스의 일반적인 위치를 다른 우선순위 프리픽스가 이미 차지하고 있는 경우이다. 이러한 경우들에서 삽입될 프리픽스는 우선순위 프리픽스의 위치를 차지한다. 삽입될 프리픽스에 의하여 자리를 빼앗긴 우선순위 프리픽스들에 대하여 동일한 절차가 반복된다.
- <64> 다른 모든 경우들에서 삽입될 프리픽스는 리프 노드에 저장되거나 또는 라우팅 테이블에 있는 기존의 프리픽스

가 대체된다.

- <65> 도 12 내지 도 14는 본 발명에 따른 우선순위 트라이 구조의 성능을 평가한 실험 결과들을 도시한다.
- <66> 도시된 실험결과들은 백본 라우터의 실제 라우팅 테이블에 대하여 시뮬레이션을 수행한 결과를 나타낸다. 도 12 및 도 13은 제안된 우선순위 이진 트라이 및 우선순위 기반 멀티 비트 트라이(2 비트의 경우)의 성능 평가 결과를 보여준다.
- <67> 각각 N 은 라우팅 프리픽스의 수, M 은 라우팅 프리픽스들 중 우선순위 프리픽스의 수, D 는 최대 프리픽스 길이, D_0 는 제안된 우선순위 트라이의 깊이, T_a 는 주소 검색을 위한 평균 메모리 액세스 횟수, M 은 다양한 크기의 라우팅 데이터를 위해 요구되는 메모리의 크기를 나타낸다. 처음 3개의 라우팅 테이블에 대하여 도 12의 우선순위 프리픽스의 수에 나타난 바와 같이, 프리픽스들의 90% 이상이 우선순위 프리픽스로 저장된다. 이는 일반 이진 트라이가 많은 빈 노드들을 가지고 있음을 의미한다.
- <68> 본 제안된 알고리즘에 따르면 우선순위 프리픽스 노드들이 많아질수록 검색 성능은 더 좋아질 것으로 예측된다. 평균 메모리 액세스 횟수도 약 16-23이며, 라우팅 테이블의 크기가 증가하더라도 크게 나빠지지 않는다. 도 13에서 2 비트는 2 비트 트라이에서와 동일한 것으로 간주된다. 2 비트 트라이에서는 별도의 노드들이 생성되는 것을 알 수 있다. 트라이 깊이 면에서 제안된 알고리즘의 성능, 즉 평균 메모리 액세스 횟수, 요구되는 메모리 크기는 라우팅 테이블의 크기가 커지더라도 크게 나빠지지 않는다. 즉, 제안된 우선순위 트라이 구조는 커다란 라우팅 데이터에 대하여 확장성(scalability) 면에서 좋은 성능을 가진다.
- <69> 한편, 도 14는 112K 엔트리와 225K 엔트리를 가진 이진 검색 구조를 시뮬레이션한 결과를 도시한다. 최악의 경우의 메모리 액세스 횟수(T_{max}), 평균 메모리 액세스 횟수(T_a), 요구되는 메모리 크기(M), 및 필요한 엑스트라 노드의 수(N_{extra}) 면에서의 성능 비교를 나타낸다. 도시된 바와 같이 제안된 우선순위 기반의 멀티 비트 트라이는 최악의 경우의 메모리 액세스 횟수 면(T_{max})에서 최고의 성능을 나타내며, 종래의 BSR(Binary search on range) 및 본 발명에서 제안된 우선순위 기반의 멀티 비트 트라이는 평균 메모리 액세스 횟수 면(T_a)에서 최고의 성능을 나타낸다. 요구되는 메모리의 크기 면(M)에서, BSR 및 제안된 우선순위 기반의 이진 트라이가 최고의 성능을 나타낸다.
- <70> 한편, 종래의 BSR 알고리즘은 각 분리 구간별 최선 정합 프리픽스를 미리 계산하기 때문에 인크리멘탈 업데이트(incremental update)를 제공하지 못한다. 반면 제안된 알고리즘은 인크리멘탈 업데이트(incremental update)를 제공할 수 있다.
- <71> 전술한 본 발명에 따른 IP 주소 검색 방법은 패킷 중계 장치 내에서 하드웨어적으로 구현될 수 있다. 패킷 중계 장치에서의 IP 주소 검색은 입력된 모든 패킷에 대하여 실시간으로 처리되어야 하므로, 실시간으로 IP 주소 검색을 수행하기 위하여 본 발명을 하드웨어 구조로 설계 구현하는 것이 적절하다.
- <72> 한편, 본 발명에 따른 IP 주소 검색 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 IP 주소 검색 방법을 구현한다. 상기 정보저장매체는 자기 기록매체, 광 기록매체, 및 캐리어 웨이브 매체를 포함한다.

발명의 효과

- <73> 전술한 바와 같이 본 발명에 따르면, 검색 시간을 줄일 수 있을 뿐만 아니라 소요되는 메모리의 크기도 줄일 수 있는 효과적인 IP 주소 검색 방법 및 이를 적용한 패킷 중계 장치가 제공된다.
- <74> 본 발명에 따르면 우선순위 트라이를 이용한 새로운 IP 주소 검색 방법이 제공된다. 즉, 제안된 알고리즘은 트리 구조에 기초하지만 빈 내부노드들이 우선순위 프리픽스(priority prefix)로 대체된다. 따라서, 제안된 알고리즘에서 가장 길게 일치하는 프리픽스(LMP)의 검색은, 입력된 프리픽스가 일반 프리픽스 노드보다 우선순위 프리픽스 노드에 일치하는 경우에 더 일찍 종료되기 때문에 보다 효율적으로 수행된다. 효율성 평가 결과는 구조된 우선순위 트라이가 요구되는 메모리, 검색 속도 및 확장성 면에서 매우 좋은 성능을 가짐을 보여준다.
- <75> 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

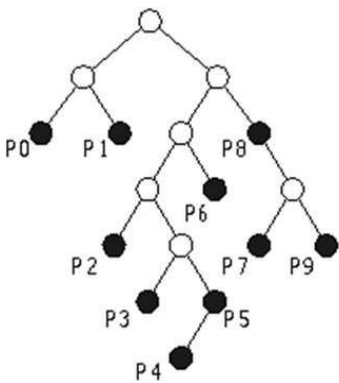
- <1> 도 1은 IP 주소 검색 방법을 설명하기 위한 프리픽스 집합의 일 예,
 - <2> 도 2는 상기 프리픽스 집합을 이용하여 종래의 이진 트라이 구조를 구축한 일 예,
 - <3> 도 3은 본 발명의 일 실시예에 따른 패킷 중계 장치의 블록도,
 - <4> 도 4는 상기 패킷 중계 장치에 포함된 인터페이스의 상세 블록도,
 - <5> 도 5는 본 발명의 일 실시예에 따른 우선순위 트라이를 이용한 IP 주소 검색 방법을 도시한 흐름도,
 - <6> 도 6은 도 1에 도시된 프리픽스 집합을 이용하여 본 발명에 따른 우선순위 트라이 구조를 구축한 일 예,
 - <7> 도 7은 본 발명에 따른 우선순위 기반 트라이 구조를 가진 라우팅 테이블의 일 예,
 - <8> 도 8은 종래의 이진 트라이 구조와 본 발명에 따른 우선순위 트라이 구조의 메모리 액세스 횟수를 비교한 도면
 - <9> 도 9 내지 도 11은 본 발명에 따른 우선순위 트라이를 구축하는 방법, 검색하는 방법 및 업데이트 하는 방법을 구현한 의사 코드(pseudo code)의 일 예,
 - <10> 도 12 내지 도 14는 본 발명에 따른 우선순위 트라이 구조의 성능을 평가한 실험 결과들을 도시한다.
 - <11> <도면의 주요부분에 대한 부호의 설명>
 - <12> 10: 인터페이스 20: 스위치 패브릭
 - <13> 11: 패킷 송수신부 12: 프레임 해석부
 - <14> 13: 어드레스 룩업 처리부 14: 패킷 분류 처리부
 - <15> 15: 전송 제어부

도면

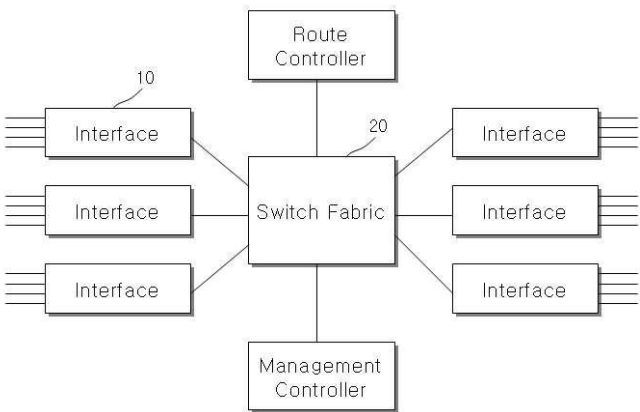
도면1

P_i	프리픽스
P0	00+
P1	01+
P2	1000+
P3	10010+
P4	100110+
P5	10011+
P6	101+
P7	11+
P8	1110+
P9	1111+

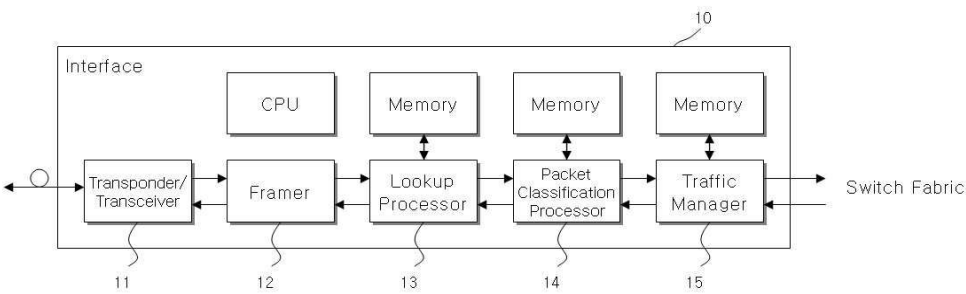
도면2



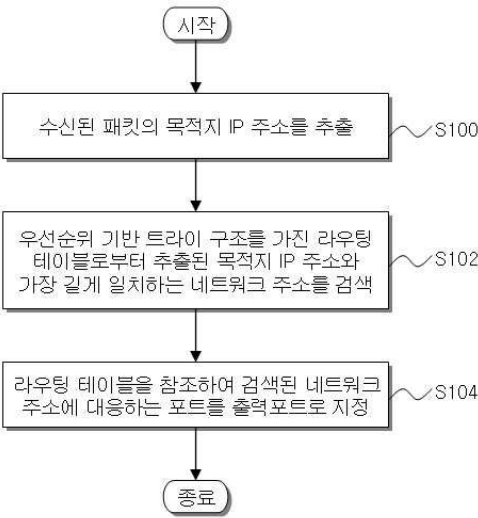
도면3



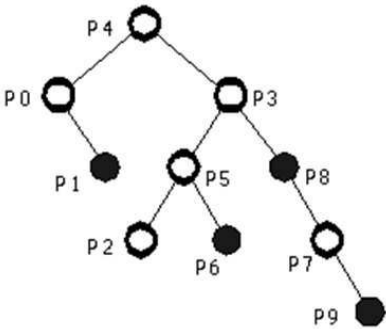
도면4



도면5



도면6



도면7

memory address	priority /ordinary	prefix	length	leftPtr	rightPtr	Out port
0	1	100110*	6	1	2	P4
1	1	00*	2	-	3	P0
2	1	10010*	5	4	5	P3
3	0	01*	2	-	-	P1
4	1	10011*	5	6	7	P5
5	0	11*	2	-	8	P8
6	1	1000*	4	-	-	P2
7	0	101*	3	-	-	P6
8	1	1110*	4	-	9	P7
9	0	1111*	4	-	-	P9

도면8

BMP	Binary Trie	Priority Trie
P0	3	2
P1	3	3
P2	5	4
P3	6	2
P4	7	1
P5	6	3
P6	4	4
P7	5	4
P8	3	3
P9	5	5

도면9

```

Build ( ) {
    BuildBinaryTrie (root);
    BuildPriorityTrie (root);
}

BuildPriorityTrie (node *ptr) {
    x = ptr;
    if (x is empty) {
        if (E(x) is empty) //no prefix enclosed in the sub-trie of x
            remove x;
        else // E(x) is not empty
            sort E(x);
            remove P(x);
            // P(x) is the prefix with the highest priority in E(x)
            store P(x) at x and mark x as a priority node;
    }
    else // non-empty node
        mark x as an ordinary node;
    if (x has a left child)
        BuildPriorityTrie (LeftChild);
    if (x has a right child)
        BuildPriorityTrie (RightChild);
    return;
}

```

도면10

```

Search (input) {
    BMP = *;
    ptr = index(root); //start at root
    do {
        if (input == prefix(ptr)) { // input matches to a prefix
            BMP = prefix(ptr);
            if (priority(ptr) == '1')
                break; //matched prefix is a priority prefix
        }
        if (nextBit(input) == '0')
            ptr = leftPtr(ptr); // follow the left pointer
        else ptr = rightPtr(ptr); //follow the right pointer;
    } while (ptr != NULL)
    return BMP;
}

```

도면11

```

Update (inputPrefix) {
    curLevel = 0;
    newPfx = inputPrefix;
    ptr = index(root); //start at the root
    do {
        if (newPfx == prefix(ptr)) { // a match
            if (priority(ptr) == '1') //match a priority prefix
                if ( (length(newPfx) > length(ptr))
                    or (length(newPfx) == curLevel) ) {
                    tmp = prefix(ptr);
                    store newPfx into the ptr;
                    if (length(newPfx) > curLevel)
                        priority(ptr) = 1;
                    else priority(ptr) = 0;
                    newPfx = tmp;
                }
            else if (length(newPfx) == length(ptr)) {
                store newPfx into the ptr;
                break;
            }
        }
        else //match an ordinary prefix
            if (length(newPfx) == length(ptr)) {
                store newPfx into the ptr;
                break;
            }
    }
    if (nextBit(newPfx) == '0') ptr = leftPtr(ptr);
    else ptr = rightPtr(ptr);
    curLevel++;
    if (ptr == NULL) {
        create a ptr;
        store newPfx into the ptr;
        priority(ptr) = 1;
    }
    } while (ptr != NULL)
}

```

도면12

Routing Table	N	N_p	D	D_p	T_a	M (Kbyte)
MAE-West1	14,553	14,199	32	24	16.66	128
Aads	20,204	19,568	32	24	17.43	178
MAE-West2	29,584	26,671	32	24	18.22	260
PORT 80	112,310	50,091	32	28	20.35	987
Grouptlcom	170,601	70,525	32	24	20.76	1,460
Telstra	227,223	119,149	32	32	22.86	1,950

도면13

Routing Table	N	N_p	N_{extra}	D	D_p	T_a	M (Kbyte)
MAE-West1	14,553	14,388	2,337	16	12	9.70	214
Aads	20,204	19,124	3,208	16	12	10.05	297
MAE-West2	29,584	23,016	6,273	16	13	10.54	455
PORT 80	112,310	36,362	21,630	16	16	11.27	1,790
Groupptlcom	170,601	47,073	32,732	16	13	11.45	2,710
Telstra	227,223	75,929	41,850	16	16	12.51	3,850

도면14

Algorithm	Binary trie [2]	BPT [16]	WPT [17]	BSR [6]	Priority binary	Priority multi-bit
Incremental update	yes	no	no	no	yes	yes
T_{max}	32	44	36	18	28	16
Port80 (112,310)	T_a 22.15	25.82	20.44	11.42	20.35	11.27
M (Mbyte)	1.29	1.25	1.25	0.96	0.99	1.79
N_{extra}	112,907	0	0	72,063	0	21,630
T_{max}	32	66	39	19	32	16
Telstra (227,223)	T_a 24.64	30.80	23.96	11.07	22.86	12.51
M (Mbyte)	2.59	2.60	2.60	1.76	1.95	3.85
N_{extra}	225,082	0	0	124,795	0	41,850