(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2009년08월18일
(11) 등록번호 10-0912545
(24) 등록일자 2009년08월10일

(51) Int. Cl.
H04L 12/56 (2006.01)
(21) 출원번호 10-2007-0096880
(22) 출원일자 2007년09월21일
심사청구일자 2007년09월21일
(65) 공개번호 10-2009-0031059
(43) 공개일자 2009년03월25일
(56) 선행기술조사문헌
KR1020040095632 A
KR1020070059833 A

(73) 특허권자
한국전자통신연구원
대전 유성구 가정동 161번지
(72) 발명자
나용욱
대전 유성구 신성동 144-5번지 정자빌라 301호
최창호
대전 유성구 반석동 반석마을 703-1002
(뒷면에 계속)
(74) 대리인
특허법인 씨엔에스 · 로고스

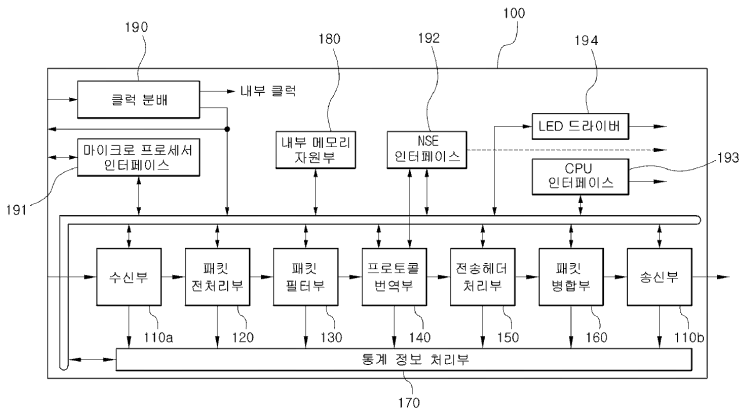
전체 청구항 수 : 총 22 항 심사관 : 김남인

(54) 패킷 처리 장치 및 방법

(57) 요 약

본 발명은, 패킷 처리 장치 및 방법에 관한 것으로서, 상기 패킷 처리 장치는 입력 패킷의 크기를 변환하고, 상기 입력 패킷을 분석하여 제2 계층 관련 처리를 수행하고, 상기 입력 패킷의 기본 전송 헤더를 생성하고, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하고, 상기 기본 전송 헤더를 추가한 입력 패킷의 헤더를 변형시켜서, 외부로 전송함으로써 추가적인 처리 작업 없이 다양한 패킷을 처리할 수 있으며, 네트워크 처리기의 사용 효율을 최적화시켜 패킷 처리 속도 및 성능을 향상시킬 수 있다.

대 표 도



(72) 발명자

안병준

대전 유성구 노은동 열매마을 807-1501

이경호

대전 유성구 어은동 한빛아파트 128-403

이 발명을 지원한 국가연구개발사업

과제고유번호	2006-S-061-02
부처명	정보통신부
연구사업명	IT신성장동력핵심기술개발사업
연구과제명	IPv6기반의 Qos 서비스 및 단말 이동성 지원 라우터기술개발
주관기관	한국전자통신연구원
연구기간	2006년 03월 01일 ~ 2009년 02월 28일

특허청구의 범위

청구항 1

입력 패킷을 내부 데이터 버스 크기로 변환하고, 데이터 제어 신호를 생성하는 수신부;

상기 수신부로부터 전달된 상기 입력 패킷을 분석하여 기본 전송 헤더를 생성하고, 제2 계층 관련 처리를 수행하는 패킷 전처리부;

상기 입력 패킷을 패킷 타입별로 처리하고, 필요로 하는 엔진 삽입 및 엔진 제거를 위한 모듈화된 프로토콜 엔진들을 포함하는 프로토콜 번역부;

상기 기본 전송 헤더에서 변환 헤더(TH)를 제거하고, 상기 입력 패킷의 패킷 타입에 따라 포인트 투 포인트 프로토콜 헤더(PPP)를 상기 입력 패킷에 적재시키는 헤더 처리부; 및

상기 헤더 처리부로부터 전달되는 상기 입력 패킷과 상기 수신부로부터 전달되는 상기 데이터 제어 신호를 변환하여 전송하는 전송부를 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 2

제1항에 있어서,

상기 패킷 전처리부로부터 전달된 상기 입력 패킷을 상기 제2 계층에 관련하여 필터링한 후 상기 입력 패킷을 상기 헤더 처리부로 전달하는 패킷 필터부를 더 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 3

제2항에 있어서,

상기 패킷 필터부는 상기 입력 패킷의 이더넷 관련 에러를 체크하고, 최대 수신 유니트 패킷 사이즈에 대한 필터링, 이더넷 분배 주소 필터링 또는 이더넷 소스 주소 필터링을 수행함을 특징으로 하는 상기 패킷 처리 장치.

청구항 4

제1항 또는 제2항에 있어서,

상기 헤더 처리부로부터 전달된 상기 입력 패킷에서 불필요한 바이트를 제거하고, 비어있는 바이트들을 병합하는 패킷 병합부를 더 포함하는 것을 특징으로 하는 것을 특징으로 하는 패킷 처리 장치.

청구항 5

제4항에 있어서,

상기 입력 패킷의 처리에 관련된 통계 정보를 처리하는 통계 정보 처리부를 더 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 6

제5항에 있어서,

내부 메모리 액세스를 수행하는 내부 메모리 자원부; 및

내부에서 사용되는 클럭 신호들을 생성하고, 상기 생성된 클럭 신호들을 분배하는 클럭 분배부를 더 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 7

제6항에 있어서,

상태 정보를 취합하여 외부로 전달하는 엘이디 드라이버;

외부의 중앙처리장치(CPU)와 인터페이스하는 마이크로처리기 인터페이스부;

상기 프로토콜 번역부로부터 추출된 키 값을 외부 상용 네트워크 검색 엔진(NSE)으로 전달하고, 상기 네트워트 검색 엔진(NSE)으로부터 얻은 룩업 결과를 상기 프로토콜 번역부로 전달하는 네트워크 검색 엔진(NSE) 인터페이스부;

외부 상용 매체 접근 제어(MAC) 칩을 제어하기 위한 중앙처리장치(CPU) 패스를 제공하는 중앙처리장치(CPU) 처리부를 더 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 8

제1항에 있어서, 상기 프로토콜 번역부는,

상기 입력 패킷이 IPv4 패킷이면, IPv4 패킷 헤더를 파싱하고 상기 입력 패킷의 헤더를 갱신하는 IPv4 처리 엔진부;

상기 입력 패킷이 멀티 프로토콜 레벨 스위칭(MPLS) 패킷이면, 새로운 전송 헤더를 생성하고 상기 입력 패킷의 헤더를 갱신하는 MPLS 처리 엔진부;

상기 입력 패킷이 터널 패킷이면, 터널 디캡슐레이션 및 터널 룩업을 수행하는 IPv4-t 처리 엔진부; 및

상기 입력 패킷이 IPv6 패킷이면, IPv6 패킷 헤더를 파싱하여 추출된 키 값을 외부 상용 네트워크 검색 엔진(NSE)으로 전달하고, 상기 외부 상용 네트워크 검색 엔진으로부터 룩업 결과들을 전달받아 상기 입력 패킷의 헤더를 정렬하는 IPv6 처리 엔진부를 포함하는 것을 특징으로 하는 패킷 처리 장치.

청구항 9

제8항에 있어서,

상기 IPv4 처리 엔진부는 802.2 LLC 패킷의 경우 패킷 길이 및 총 길이 필드를 확인하여 패딩이 되어 있는 경우 상기 패딩을 제거함을 특징으로 하는 패킷 처리 장치.

청구항 10

제8항에 있어서,

상기 IPv6 처리 엔진부는 미리 설정된 해쉬 알고리즘을 통해 해쉬 값을 생성하고, 상기 기본 전송 헤더의 제1 전송 헤더(DH_1 v4)를 제2 전송 헤더(DH_1 v6)로 변환함을 특징으로 하는 패킷 처리 장치.

청구항 11

입력 패킷의 크기를 변환하는 과정;

상기 입력 패킷을 분석하여 제2 계층 관련 처리를 수행하는 과정;

상기 입력 패킷의 기본 전송 헤더를 생성하는 과정;

상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정;

상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정의 처리 결과에 따라, 상기 입력 패킷의 헤더를 변형시키는 과정; 및

상기 변형된 헤더를 가지는 입력 패킷을 변환하여 전송하는 과정을 포함하여 패킷 처리 장치에서 추가적인 처리 작업 없이 다양한 패킷을 처리하는 것을 특징으로 하는 패킷 처리 방법.

청구항 12

제11항에 있어서,

상기 제2 계층 관련 처리 후 상기 입력 패킷을 상기 제2 계층에 관련하여 필터링하는 과정을 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 13

제11항 또는 제12항에 있어서,

상기 변형된 헤더를 가지는 입력 패킷에서 불필요한 바이트를 제거 및 비어있는 바이트들을 병합하는 과정을 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 14

제13항에 있어서,

상기 불필요한 바이트를 제거하기 위한 마스크를 조정하는 과정을 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 15

제11항에 있어서, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정은,

상기 입력 패킷의 헤더 정보를 추출하는 단계;

상기 추출된 헤더 정보에서 상기 입력 패킷의 패킷 타입이 IPv4 패킷을 나타내면 상기 IPv4 패킷을 파싱하여 분석하는 단계;

상기 IPv4 패킷의 헤더 필드를 세팅하는 단계;

상기 헤더 필드가 유효하면 헤더 체크섬을 계산하여 갱신하는 단계;

상기 IPv4 패킷의 기본 전송 헤더인 변환 헤더(TH) 및 제1 전송 헤더(DH_1 v4)를 갱신하는 단계; 및

상기 IPv4 패킷의 마스크를 조정하는 단계를 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 16

제15항에 있어서, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정은,

상기 헤더 필드가 유효하지 않는 경우 상기 기본 전송 헤더에서 상기 변환 헤더(TH)를 제외시키기 위한 제외 비트를 세팅하는 단계를 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 17

제11항에 있어서, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정은,

상기 입력 패킷의 헤더 정보를 추출하는 단계;

상기 추출된 헤더 정보에서 상기 입력 패킷의 패킷 타입이 멀티 프로토콜 레벨 스위칭(MPLS) 패킷을 나타내면 상기 멀티 프로토콜 레벨 스위칭(MPLS) 패킷의 기본 전송 헤더인 변환 헤더(TH) 및 제1 전송 헤더(DH_1 v4)의 일부 정보를 세팅하는 단계;

상기 기본 전송 헤더의 일부 정보를 갱신하는 단계;

상기 멀티 프로토콜 레벨 스위칭(MPLS) 패킷에 새로운 전송 헤더를 삽입하는 단계; 및

상기 멀티 프로토콜 레벨 스위칭(MPLS) 패킷의 헤더를 정렬하는 단계를 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 18

제11항에 있어서, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정은,

상기 입력 패킷의 헤더 정보를 추출하는 단계;

상기 추출된 헤더 정보에서 프로토콜 필드가 터널 패킷을 나타내면, 상기 터널 패킷의 터널 테이블 룩업을 수행하는 단계;

상기 추출된 헤더 정보에서 IPv4 분배 어드레스가 매치되면 상기 터널 패킷의 IPv4 터널 헤더를 디캡슐레이션시키는 단계; 및

상기 터널 패킷의 헤더를 정렬하는 단계를 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 19

제18항에 있어서,

상기 IPv4 분배 어드레스가 매치되지 않는 경우 상기 기본 전송 헤더인 변환 헤더(TH)의 제외(except) 비트를 세팅하는 단계를 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 20

제11항에 있어서, 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정은,

상기 입력 패킷의 헤더 정보를 추출하는 단계;

상기 추출된 헤더 정보에서 상기 입력 패킷의 패킷 타입이 IPv6 패킷을 나타내면, IPv6 패킷의 헤더를 파싱하여 키 값들을 추출하는 단계;

상기 추출된 키 값들을 이용하여 서비스 품질(QoS)/서비스 거부(DoS) 룩업을 수행하는 단계;

미리 설정된 해쉬 알고리즘을 통해 해쉬값을 생성 단계;

상기 IPv6 패킷의 전송 헤더에 상기 해쉬값을 삽입하는 단계;

상기 IPv6 패킷의 헤더를 정렬하는 단계를 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 21

제20항에 있어서,

상기 기본 전송 헤더의 제1 전송 헤더(DH_1 v4)를 제2 전송 헤더(DH_1 v6)로 변환하는 단계를 더 포함하는 것을 특징으로 하는 패킷 처리 방법.

청구항 22

제11항에 있어서, 상기 기본 전송 헤더를 추가한 입력 패킷의 헤더를 변형시키는 과정은,

상기 기본 전송 헤더에서 변환 헤더(TH)를 제거하는 단계; 및

포인 투 포인트 프로토콜(PPP) 헤더를 상기 변환 헤더(TH)가 제거된 입력 패킷에 적재시키는 단계를 포함하는 것을 특징으로 하는 패킷 처리 방법.

명 세 서

발명의 상세한 설명

기 술 분 야

<1> 본 발명은 패킷 스위치 시스템에서의 패킷 처리 장치 및 방법에 관한 것으로서, 특히 네트워크 처리기(Processer)나 패킷 처리기를 사용하는 패킷 처리 장치 및 방법에 관한 것이다.

<2> 본 발명은 정보통신부 IT신성장동력핵심기술개발사업의 일환으로 수행한 연구로부터 도출된 것이다[과제관리번호: 2006-S-061-02, 과제명: IPv6 기반의 QoS 서비스 및 단말 이동성 지원 라우터 기술개발].

배 경 기 술

<3> 인터넷의 대중화에 따른 폭발적인 데이터 트래픽(Traffic)과 유무선 인터넷 통합에 따른 새로운 형태의 서비스의 출현으로 인해 기존의 네트워크 장비는 패킷 처리 용량 및 성능에 문제점을 드러내기 시작했고, 이의 해결 방안으로 네트워크 처리기가 등장하였다. 그러나 이러한 네트워크 처리기를 이용한 네트워크 장비들은 모든 패킷 처리를 네트워크 처리기가 담당하게 함으로서 다양한 멀티미디어 서비스 및 애플리케이션을 지원하기 위한 기능들이 네트워크 처리기에 적재됨에 따라 성능저하 현상이 나타나게 되었다. 이러한 문제점을 해결하기 위해 종래의 패킷 처리 장치는 패킷 종류별로 구분을 두어 네트워크 처리기 앞단에서 버퍼를 관리하거나, 네트워크 처리기 자체 엔진을 가변적으로 사용할 수 있도록 하여 효율을 높였다.

<4> 이와 같은 일반적인 패킷 처리 장치들 중 네트워크 처리기를 이용한 IP 패킷 처리 장치를 예를 들 수 있는데,

이를 첨부된 도 1을 참조하여 설명하기로 한다.

<5> 네트워크 처리기를 이용한 IP 패킷 처리 장치는, 상기 도 1에 도시된 바와 같이, 네트워크 처리기의 8개의 마이크로 엔진들(11 내지 18)로 구성되며, 이러한 마이크로 엔진들(11 내지 18)은 패킷 수신부(Packet Rx)(11), 패킷 분류부(Ethernet Decap/Classify)(12), IPv4 패킷 및 IPv6 패킷 포워더(IPv6 Unicast/Multicast Forwarder)(13), 패킷 큐 관리부(Packet Queue Manager)(14), 패킷 스케쥴러(Packet Scheduler)(15), 멀티캐스트 패킷 복사부(Multicast Packet Copier)(16), 패킷 송신부(Packet Tx and Ethernet Encap)(17)로 사용될 수 있다.

<6> 일반적인 패킷 처리 장치의 다른 예로 첨부된 도 2에 도시된 바와 같은 패킷 처리 장치가 있는데, 이는 포인트 투 포인트 프로토콜(Point-to-Point Protocol 이하, PPP라 칭함) 헤더에서 프로토콜 필드 룩업부(21), 버퍼 관리부(22), 데이터 전송에 필요한 버퍼(23 내지 24) 및 네트워크 처리기(Network Processor) 내의 IPv4 및 IPv6 처리부(25, 26)로 이루어져 있다. 여기서 버퍼는 IPv4 패킷을 저장하기 위한 버퍼 1(23)과 IPv6 패킷용 버퍼 2(24)의 두 개의 버퍼가 사용된다. 이러한 버퍼 1(23)은 IPv4 패킷을 저장하기 위한 버퍼이고, 버퍼 2(24)는 IPv6 패킷을 저장하기 위한 버퍼이다.

<7> 또한, 일반적인 패킷 처리 장치의 또 다른 예로 첨부된 도 3에 도시된 바와 같은 PPP 데이터 프레임 처리 장치가 있는데, 상기 PPP 데이터프레임 처리 장치(30)는 외부의 스위치보드와의 데이터교환을 위한 패브릭 인터페이스(31)와, 패브릭인터페이스(31)를 통해 수신된 PPP 데이터프레임을 분석하기 위한 제1 및 제2 PPP데이터 분석모듈(32, 33)과, PPP 데이터프레임의 서브컨트롤을 제어하기 위한 호스트(34)를 포함한다. 여기서 제1 PPP 데이터 분석모듈(32)은 PPP데이터를 분석하여 PPP데이터에 포함된 서브컨트롤에 따라 데이터를 복원하는 제1처리기와 PPP데이터의 헤더를 제거하는 제2처리기를 포함한다. 그리고 상기 제2 PPP데이터 분석모듈(33)은 IP데이터에 헤더를 추가시키는 제3처리기와, IP데이터를 PPP데이터 형식으로 변환시키는 제4처리기를 포함한다.

<8> 이와 같은 패킷 처리 장치들의 기능 및 동작에 대해서는 통상의 지식을 가진자들에게 일반적으로 잘 알려진 것으로서 구체적인 설명을 생략하기로 한다. 그리고 이와 같은 종래의 패킷 처리 장치는 IPv4 패킷 및 IPv6 패킷을 처리하는데, 이러한 IPv4 패킷 및 IPv6 패킷의 기본 헤더는 IPv4 및 IPv6 표준 프로토콜의 헤더로서, 이들 헤더 필드 포맷 구조는 첨부된 도 4 및 도 5에 도시된 바와 같으며, 이 또한 통상의 지식을 가진자들에게 일반적으로 잘 알려진 것으로서 구체적인 설명을 생략하기로 한다.

<9> 하지만, 상기 도 1 내지 도 3에 도시된 바와 같은 종래의 기술의 패킷 처리 장치들은 다음과 같은 문제점들이 있다.

<10> 상기 도 1에 도시된 바와 같은 네트워크 처리기를 사용한 IP 패킷 처리 장치는 네트워크 프로세의 마이크로 엔진들을 고정적으로 할당하지 않고, 가변 기능 마이크로 엔진을 두어 수신되는 패킷의 종류별 패킷량에 따른 동적인 자원 할당을 가능하게 함으로써 패킷 처리를 위해 사용되는 네트워크 처리기를 효율적으로 사용할 수 있게 하였다. 그러나 이는 실제 패킷 처리를 위한 모든 패킷 처리를 네트워크 처리기가 담당함으로써 여전히 지원하는 프로토콜들이 늘어나게 될수록 네트워크 처리기의 성능을 저하시키는 문제점이 발생한다.

<11> 상기 도 2에 도시된 바와 같은 패킷 처리 장치는 프로토콜별로 패킷을 구분해서 버퍼링함으로써 네트워크 처리기에서 프로토콜에 따른 패킷별로 독립적인 처리가 가능하게 하고, 지원하지 않는 네트워크 프로토콜에 관련된 처리과정을 생략할 수 있어 자원낭비를 방지할 수 있다. 또한, 상기 패킷 처리 장치는 버퍼 매니저를 통해서 각 네트워크 프로토콜별로 할당하는 버퍼 크기를 동적으로 조정할 수 있게 함으로써 수신되는 패킷 량, 수신되는 패킷 량의 변화 등에 따른 효율적인 통신이 가능하도록 한다. 하지만, 상기 패킷 처리 장치는 단순히 프로토콜 룩업에 의한 버퍼 관리를 함으로써 실제 패킷 처리에 대한 처리를 하지 않음으로 인한 네트워크 처리기의 패킷 처리 처리량은 변하지 않게 된다. 더욱이, 상기 패킷 처리 장치는 네트워크 처리기에 라우팅 프로토콜이나 멀티 프로토콜 레벨 스위칭(MPLS : Multi Protocol Label Switching) 등 다양한 프로토콜 등이 적재되면 실제 시스템의 성능 저하 현상을 유발하게 된다.

<12> 상기 도 3에 도시된 바와 같은 포인트 투 포인트 프로토콜(Point to Point Protocol 이하, PPP라 칭함) 데이터 프레임 처리 장치는 각기 다른 기능을 수행하는 한편 데이터 프레임의 처리가 가능한 복수개의 처리기를 포함하는 네트워크 처리기를 이용하여 하드웨어적으로 PPP 데이터 프레임을 생성 및 분석하도록 함으로써, 대량의 PPP 데이터를 고속 처리하는 것을 가능케 하였다. 그러나 PPP 데이터프레임 처리 장치는 일반적으로 전체네트워크의 80% 이상을 이루고 있는 이더넷 프레임을 달고 들어오는 패킷을 처리하지 못한다. 따라서 이러한 처리를 하기 위해서는 또 다른 처리기의 처리 능력을 요구하고 이더넷 패킷 처리 기능이 추가되므로 PPP 데이터프레임 처리

장치는 실제 네트워크 처리기의 부담을 증가시켜서 시스템의 성능에 영향을 주게 된다.

<13> 최근에는 IPv6을 지원하는 수많은 어플리케이션 등장하고, 지속적으로 증가하는 트래픽 요구에 점점 더 회선 속도가 빨라지고, 최근에는 기가비트 스위치가 주력으로 부상하고 있으며, 10기가비트 포트가 업 링크용으로 지원되는 스위치 구조가 대두되고 있다. 이에 따라 다양한 애플리케이션을 지원하는 시스템의 성능은 대단히 중요한 문제로 부각되고 있다.

발명의 내용

해결 하고자하는 과제

<14> 따라서 본 발명의 목적은 네트워크 처리기나 패킷 처리기가 이더넷을 지원하지 않고 PPP와 MPLS 프로토콜만을 처리하도록 구성되어 있는 경우, 성능 저하의 원인이 되는 추가적인 처리 작업 없이 다양한 이더넷 패킷을 처리하기 위한 패킷 처리 장치 및 방법을 제공함에 있다.

<15> 본 발명의 다른 목적은 기본적인 패킷 처리를 순수 하드웨어 로직으로 처리하고, 서비스 품질(QoS) 및 다양한 프로토콜 등을 지원하기 위하여 라인속도를 저하시키지 않는 전송 헤더(Delivery Header)의 추가와 패킷 헤더를 변형시키는 패킷 처리 장치 및 방법을 제공함에 있다.

과제 해결수단

<16> 상기 이러한 본 발명의 목적들을 달성하기 위한 패킷 처리 장치는, 입력 패킷을 내부 데이터 버스 크기로 변환하고, 데이터 제어 신호를 생성하는 수신부; 상기 수신부로부터 전달된 상기 입력 패킷을 분석하여 기본 전송 헤더를 생성하고, 제2 계층 관련 처리를 수행하는 패킷 전처리부; 상기 입력 패킷을 패킷 타입별로 처리하고, 필요로 하는 엔진 삽입 및 엔진 제거를 위한 모듈화된 프로토콜 엔진들을 포함하는 프로토콜 번역부; 상기 기본 전송 헤더에서 변환 헤더(TH)를 제거하고, 상기 입력 패킷의 패킷 타입에 따라 포인트 투 포인트 프로토콜 헤더(PPP)를 상기 입력 패킷에 적재시키는 헤더 처리부; 및 상기 헤더 처리부로부터 전달되는 상기 입력 패킷과 상기 수신부로부터 전달되는 상기 데이터 제어 신호를 변환하여 전송하는 전송부를 포함하는 것을 특징으로 한다.

<17> 상기 본 발명의 목적들을 달성하기 위한 패킷 처리 방법은, 입력 패킷의 크기를 변환하는 과정; 상기 입력 패킷을 분석하여 제2 계층 관련 처리를 수행하는 과정; 상기 입력 패킷의 기본 전송 헤더를 생성하는 과정; 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정; 상기 생성된 기본 전송 헤더를 적용한 상기 입력 패킷을 타입별로 처리하는 과정에 따라, 상기 입력 패킷의 헤더를 변형시키는 과정; 및 상기 변형된 헤더를 가지는 입력 패킷을 변환하여 전송하는 과정을 포함하여 패킷 처리 장치에서 추가적인 처리 작업 없이 다양한 패킷을 처리하는 것을 특징으로 한다.

효 과

<18> 상술한 바와 같이 본 발명은 추가적인 처리 작업 없이 다양한 이더넷 패킷을 처리 할 수 있으며, 전송 헤더의 추가 및 패킷 헤더를 변형시킴으로써 시스템의 라인 속도(Line Speed) 저하를 방지할 수 있으며, 이에 따라 네트워크 처리기의 사용 효율을 최적화시켜 멀티미디어 응용에서 패킷 처리 속도 및 성능을 향상시킬 수 있는 효과가 있다.

발명의 실시를 위한 구체적인 내용

<19> 이하, 본 발명의 바람직한 실시 예를 첨부한 도면을 참조하여 상세히 설명한다. 본 발명을 설명함에 있어, 관련된 공지 기능 혹은 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

<20> 본 발명의 실시예에서는 패킷 처리 장치가 네트워크 처리기나 패킷 처리기를 이용하며, 네트워크 처리기나 패킷 처리기가 이더넷을 지원하지 않고 포인트 투 포인트 프로토콜(Point-to-Point Protocol 이하, PPP라 칭함)와 멀티 프로토콜 레벨 스위칭(Multi Protocol Label Switching 이하, MPLS라 칭함) 프로토콜만을 처리하도록 구성되며, 네트워크 처리기나 패킷 처리기에서 추가적인 처리 작업 없이 이더넷 패킷을 처리한다. 그러면 상기 패킷 처리 장치의 구조에 대해 첨부된 도면을 참조하여 구체적으로 설명하기로 한다.

<21> 도 6은 본 발명의 실시예에 따른 패킷 처리 장치의 구성을 도시한 도면이다.

<22> 상기 도 6을 참조하면, 패킷 처리 장치(100)는 기본적인 처리를 순수 하드웨어적인 로직으로 처리하고, 서비스

품질((Qualitiy of Service 이하, QoS라 칭함) 및 다양한 프로토콜 등을 지원하기 위하여 라인 속도(Wire Speed)를 저하시키지 않는 전송 헤더(DH : Delivery Header)를 추가하고, 패킷 헤더를 변형시킨다. 그리고 상기 패킷 처리 장치(100)는 송/수신부(110a, 110b)와, 패킷 전처리부(120), 패킷 필터부(130), 프로토콜 번역부(140), 전송 헤더 처리부(150), 패킷 병합부(160), 통계 정보 처리부(170), 내부 메모리 자원부(180), 클럭 분배부(190) 및 다양한 인터페이스들을 포함하여 구성된다. 여기서 상기 다양한 인터페이스들은 마이크로처리기 인터페이스(191), 네트워크 검색 엔진(Network search Engine 이하, NSE라 칭함) 인터페이스(192), 중앙 처리 장치(CPU) 인터페이스(193), LED 드라이버들(194) 등이다.

<23> 상기 수신부(110a)는 1 기가비트 이더넷(Gigabit Ethernet) 또는 10 기가비트 이더넷 외부 상용 매체 접근 제어(Media Access Control 이하, MAC이라 칭함) 칩으로부터 SPI 인터페이스를 경유하여 입력되는 패킷(Ingress Packet)을 수신하여 64비트 즉, 데이터 버스 크기로 변환하고, 데이터 제어 신호들을 생성한다.

<24> 상기 패킷 전처리부(120)는 상기 수신부(110a)에서 생성된 각종 데이터 제어 신호를 수신하고, 입력되는 패킷의 제2 계층(Layer 2) 헤더를 분석하여 변환 헤더(Transitional Header 이하, TH라 칭함), 제1 전송 헤더(DH_1 v4 : Delivery Header_1 v4)와 같은 기본적인 전송 헤더들을 생성하고, 패딩 제거 기능 및 L2 터널을 포함한 제2 계층(Layer 2) 관련 처리를 수행한다.

<25> 상기 패킷 필터부(130)는 MAC 필터, 프레임 검사 순서(FCS : Frame Check Sequence) 필터, 최대 수신 유니트(MRU : Maximum Receive Unit) 필터, 가상 랜(Virtual LAN) 필터 등을 포함하며, 이더넷 FCS 에러를 체크하여 마킹하고, MRU 패킷 사이즈에 대한 필터링, 이더넷 분배 주소(Destination Address) 필터링, 소스 포트(Source Port), VLAN ID, 이더넷 소스 주소(Source Address)에 근거한 이더넷 소스 주소 필터링을 수행한다.

<26> 상기 프로토콜 번역부(140)는 IPv4 패킷, IPv6 패킷, MPLS 패킷, IPv4 터널 패킷등과 같은 각종 패킷 타입별로 패킷 처리 기능을 수행하는 모듈화된 프로토콜 엔진들을 포함한다. 이에 대한 구체적인 구조는 이하, 첨부된 도 7의 설명에서 구체적으로 설명하기로 한다.

<27> 상기 전송 헤더 처리부(150)는 상기 패킷 전처리부(120)에서 생성된 전송 헤더 중 TH를 제거하고, 각각의 패킷 타입에 따라 PPP 헤더를 적재시킨다.

<28> 상기 패킷 병합부(160)는 입력된 패킷의 불필요한 바이트를 제거하고, 상기 입력된 패킷의 비어있는 바이트들을 병합(Packing)하며, 제거된 바이트에 따라 유효한 데이터의 EOP(End of Packet) 위치를 조정한다.

<29> 상기 송신부(110b)는 64비트 내부 데이터 버스 및 제어 신호를 SPI 인터페이스로 변환하여 네트워크 처리기나 패킷 처리기에 전달한다.

<30> 상기 통계 정보 처리부(170)는 패킷 카운터 및 바이트 카운터, 에러 카운터 등의 각종 통계 정보를 처리한다.

<31> 상기 내부 메모리 자원부(180)는 내부 레지스터 및 모듈을 제어하며, 읽기/쓰기 레지스터 및 DPRAM(Dual Port RAM)과 같은 메모리 액세스를 위한 기능을 수행한다.

<32> 상기 클럭 분배부(190)는 패킷 처리 장치(100) 내에서 사용되는 모든 클럭 신호들을 생성하고, 각 블록에 락킹(Locking)된 클럭 신호들을 분배한다.

<33> 상기 마이크로소프트 인터페이스(191)는 외부 사용 CPU와의 인터페이스를 가지며, 입력된 어드레스를 디코딩하여 각 모듈의 레지스터에 칩 선택 및 읽기/쓰기 인에이블(Enable) 등을 생성한다.

<34> 상기 NSE 인터페이스부(192)는 외부 상용 NSE와 인터페이스를 가지며, 상기 프로토콜 번역부(140) 내의 IPv6 처리 엔진에서 추출된 키값을 전달받아 상기 NSE로 전달하고, 상기 NSE로부터의 LPM(Longest Prefix Matching)/QoS/서비스 거부(DoS : Denial of Service) 룩업 후 결과물로 나온 인덱스(index)들을 다시 상기 프로토콜 번역부(140) 내의 IPv6 처리 엔진으로 전달한다.

<35> 상기 CPU 인터페이스(193)는 외부 상용 MAC 칩을 제어하기 위하여 CPU 패킷을 제공한다.

<36> 상기 LED 드라이버부(194)는 각종 상태 정보를 용이하게 알 수 있도록 취합하여 외부로 전달한다.

<37> 도 7은 본 발명의 실시예에 따른 프로토콜 번역부의 일예를 도시한 도면이다.

<38> 상기 도 7을 참조하면, 상기 프로토콜 번역부(140)는 크게 IPv4 패킷에 대해 처리를 하는 IPv4 처리 엔진부(Processing Engine)(141), MPLS 패킷에 대해 처리를 하는 MPLS 처리 엔진부(Precessing Engine)(142), IPv4 터널링 패킷에 대해 처리를 하는 IPv4-t 처리 엔진부(Processing Engine)(143), IPv6 패킷에 대해 처리를 하는

IPv6 처리 엔진부(Processing Engine)(144) 등으로 구성될 수 있다. 이러한 모든 엔진부들 간의 인터페이스는 동일하며, 각각의 엔진부들은 모듈화되어 있어 특정 엔진부만을 요구 시에 TH의 Evld[1:0] 신호의 업데이트에 따라 필요로 하는 각 엔진들의 삽입과 제거가 가능하도록 되어 있다.

<39> 상기 IPv4 처리 엔진부(141)는 상기 패킷 필터부(130)로부터 입력된 패킷 즉, 필터링된 패킷이 전송 헤더의 패킷 타입과 AHL(Aggregate Header Length)를 분석하고, 입력된 패킷이 IPv4 패킷이면 IPv4 헤더를 파싱(Parsing)하고, 헤더 체크섬(Checksum)을 계산하여 업데이트를 수행한다. 또한, 상기 IPv4 처리 엔진부(141)는 802.2 LLC 패킷에 대해서는 패킷 길이(Packet Length)와 총 길이(Total Length) 필드를 체크하여 패딩이 되어 있으면, 마스크(Mask)와 EOP(End of Packet)를 조정하여 패딩을 제거하는 기능을 수행하고, IPv4 패킷이 아니면 바이패스(Bypass)한다.

<40> 상기 MPLS 처리 엔진부(142)는 상기 패킷 필터부(130)로부터 입력된 패킷 즉, 필터링된 패킷이 MPLS 패킷이면 제3 및 제4 전송 헤더(DH_2 v4 : Delivery Header_2 v4, DH_2 v6 : Delivery Header_2 v6)를 생성하고, LPOPed MPLS 라벨(label)을 위한 TTL(Time-to-Live) Copy, MPLS stack bit 등 TH 및 DH_1 v4를 업데이트 하며, 반면, MPLS 패킷이 아니면 바이패스(Bypass)한다.

<41> 상기 IPv4-t 처리 엔진부(143)는 상기 패킷 필터부(130)로부터 입력된 패킷 즉, 필터링된 패킷이 터널 패킷이면 6 to 4 터널, 6 over 4 터널을 디캡슐레이션하고, 내부 캠(CAM)을 사용하여 터널을 룩업하며, 반면, 터널 패킷이 아니면 바이패스(Bypass)한다.

<42> IPv6 처리 엔진부(144)는 상기 패킷 필터부(130)로부터 입력된 패킷 즉, 필터링된 패킷이 IPv6 패킷이면 IPv6 패킷 헤더를 파싱(Parsing)하여 다양한 키값들을 추출하고, 외부 상용 NSE 룩업을 위해 그 키값들을 적절히 조합하여 NSE 인터페이스부(192)로 전달하며, 해쉬 알고리즘을 통해 20 비트 해쉬값(Hash value)을 생성한다. 또한, IPv6 처리 엔진부(144)는 DH_1 v4를 제2 전송 헤더(DH_1 v6 : Delivery Header_1 v6)로 변환하고, NSE 인터페이스부(192)로부터 전달받은 룩업 결과들을 가지고 IPv6 패킷 헤더를 재정렬한다.

<43> 그러면 이와 같은 구조를 갖는 패킷 처리 장치에서 라인속도를 저하시키지 않기 위해 추가되는 전송 헤더(Delivery Header)와 패킷 헤더를 변환시키는 과정에 대해 첨부된 도면들을 참조하여 구체적으로 설명하기로 한다.

<44> 도 8은 본 발명의 실시예에 따른 전송 헤더의 일예를 도시한 도면이다.

<45> 상기 도 8을 참조하면, 전송 헤더는 상기 기본 전송 헤더인 TH, DH_1 v4와, DH_1 v6와, 새로운 전송 헤더인 DH_2 v4, DH_2 v6 등으로 구분할 수 있다.

<46> 상기 TH는 VLAN 인덱스(index)가 유효한지를 나타내는 신호인 vidx_vld, 상기 vidx_vld 신호가 하이(High)이면 유효한 7비트의 규정된(provisioned) VLAN 인덱스를 나타내는 vidx[6:0], 프로토콜 번역부(140)에서 다양한 타입의 패킷 처리를 용이하게 하기 위해 준비된 8비트의 ahl(Aggregate Header Length)[7:0], 3비트의 MPLS 라벨 수를 나타내는 num_labels[2:0], 1비트의 transparent flag, 1 비트의 eprx, 상기 TH는 프로토콜 번역부(140)의 엔진들의 가용여부를 나타내는 2비트의 Evld[1:0], 1 비트의 Except 비트, 입력되는 패킷의 타입을 나타내는 4비트의 pkt_type[3:0], 1비트의 ttl_copy, DH_1 v4 와 DH_1 v6 헤더의 위치를 나타내는 2 비트의 loc[1:0], 1 비트의 pkt_err로 구성된다.

<47> 상기 DH_1 v4는 2비트의 ones[1:0], 패킷이 VLAN 태그(tag)를 가졌는지를 나타내며 만약 VLAN 태그를 가졌다면 VLAN ID 와 PRI 필드를 DH_1 v4 의 vid_or_mac_sa[11:0]와 pri[2:0]에 매핑을 시키기 위한 1비트의 vlan flag, 입력되는 패킷이 터널링된 포트로부터의 입력인지를 나타내는 1비트의 tunnel 비트, 4비트의 port[3:0], 입력되는 패킷이 VLAN 태그를 가지고 있으면 12 비트의 VLAN ID, 그렇지 않으면 12 비트의 LSB(Least Significant Bit) 소스 주소(Source Address)를 매핑시키는 vid_or_mac_sa[11:0], 입력되는 패킷이 VLAN Tag를 가지고 있으면 VLAN PRI 비트들이 매핑되고, 그렇지 않으면 제로(zero)로 세팅되는 3비트의 pri[2:0], MPLS stack 비트를 나타내는 1비트의 stack, 두 비트의 Spare, 1비트의 Trunk, 실제 MPLS 패킷인지를 나타내는 1비트의 lpop, ipck[2] 는 IHL 이 5가 아니면 세팅되고, ipck[1]은 checksum 결과가 유효하지 않으면 세팅, ipck[0]는 total length 가 20 보다 작으면 세팅되는 IPv4 패킷의 헤더 체크 결과를 표시하는 ipck[2:0], DH_1 v4 를 DH_1 v6 로 업데이트를 알리는 1 비트의 ipv6 flag 로 구성되어 있다.

<48> 상기 DH_1 v6은 해쉬 알고리즘을 통해 얻은 20비트의 해쉬값을 나타내는 hash_val[19:0], VLAN 태그가 세팅되어 있으면 VLAN PRI가 매핑되고, 그렇지 않으면 제로(zero)로 세팅되는 pri[2:0], MPLS Stack 비트와 동일한 1비트의 stack, 4비트의 port[3:0], 1비트의 lpop, 1비트의 spare, 만약 Flow Label 필드가 제로(zero)이면,

flow_label_zero flag가 세팅되고, 플로우(flow)를 위한 해싱(Hashing)에서 다른 IP/TCP/UDP 필드들이 사용됨을 나타내는 1비트의 flow_label_zero비트, IPv6 패킷인지를 나타내는 1비트의 ipv6 비트로 구성된다.

<49> DH_2 v4 와 DH_2 v6 헤더는 해쉬(Hash) 와 터널 플래그(Tunnel Flag) 테이블로부터 추출된 해쉬 플레그(Hash Fla)에 근거하여 추가되고, 만약 DH_1 필드의 해쉬 조건이 투명(Transparent)하면 LPOP 라벨 값(Label value)은 20'h00000이 되고, 그렇지 않으면 20'hFFFFB가 된다. 해쉬 조건은 EPRX, 해쉬 및 터널 테이블 룩업 결과, LPOP 플래그(flag), MPLS 라벨 수, 다음 패킷 타입 순서대로 검색하는 것이다. 이러한 상기 DH_2 v4 또는 DH_2 v6 헤더는 실제 MPLS LPOP 레벨(label)이 존재하면 MPLS 처리 엔진부(142)에서 업데이트 되지 않는다. 이때, 상기 MPLS 처리 엔진부(142)는 DH_2 v4 또는 DH_2 v6 헤더가 아닌 DH_1 v4 의 LPOP flag 비트를 업데이트 한다.

<50> 도 9는 본 발명의 실시예에 따라 패킷 처리 장치에서의 패킷 변환 과정을 도시한 도면이다. 상기 도 9에서는 입력된 패킷이 IPv6/MPLS/이더넷(Ethernet)인 패킷인 경우의 일예를 보여주고 있다.

<51> 패킷 포맷 A(310)는 입력된 패킷이 SPI 인터페이스를 통하여 수신부(110a)를 거친 후 64 비트 포맷으로 바뀐 직후의 구성이다.

<52> 패킷 포맷 B(320)는 패킷 처리부(120)를 거친 후에 제 2계층(Layer 2) 헤더를 분석하여 TH 및 DH_1 v4 헤더를 삽입한 후의 구성이다. 상기 패킷 포맷 B(320)는 패킷 필터부(130)까지 유지된 후 IPv4 처리 엔진부(141)를 거친 후 TH 및 DH_1 v4 헤더가 갱신된다.

<53> 패킷 포맷 C(330)는 MPLS 처리 엔진부(142)에서 TH 및 DH_1 v4 헤더를 분석하여 DH_2 v4 헤더를 삽입한 구성이다.

<54> 패킷 포맷 D(940)는 입력된 패킷이 실제 MPLS 패킷인 경우에 헤더 구성을 재조정한 구성이며, 이 포맷 그대로 IPv4-t 처리 엔진부(143)로 포워딩 된다. IPv4-t 처리 엔진부(143)에서는 터널 패킷이 아니므로 포맷 변경이 이루어 지지 않으며, '6 to 4' 터널이나 '6 over 4' 터널 패킷인 경우에는 IPv4 헤더를 제거한 후에 IPv6 처리 엔진부(144)로 패킷을 포워딩한다.

<55> 패킷 포맷 E(950)는 룩업 결과물들을 삽입한 후, 상용 네트워크 처리기나 패킷 처리기의 헤더 분석 방식에 따라 패킷을 재정렬하는 중간 과정의 구성이다. 여기서 상기 룩업 결과물들은 IPv6 처리 엔진부(144)에서 DH_1 v4 헤더를 해쉬 알고리즘을 통해 얻은 20비트 해쉬 값을 삽입하여 DH_1 v6 헤더로 변경한 뒤에 추출된 키값들을 외부 상용 NSE으로 보낸 후 얻은 라우터 아이디(Route ID), 서비스 거부 아이디(DoS ID), 서비스 품질 아이디(QoS ID) 등의 정보들이다.

<56> 패킷 포맷 F(960)는 헤더 처리부(150)를 거쳐 TH가 제거되고, 2바이트 PPP 헤더가 삽입된 후 패킷 병합부(160)를 거쳐 패킷의 불필요한 바이트를 제거한 후 병합(Packing)된 구성이다.

<57> 이와 같은 각 포맷들의 우측면에 표시된 클럭 사이클(clock cycle)은 실제 내부 처리 시의 1 Tick을 보여주며, 실제 소요된 처리 클럭 사이클이 감소되거나 유지되어 라인속도(Wire Speed)를 유지함을 알 수 있다.

<58> 한편, 본 발명의 실시예에 따른 패킷 처리 장치의 출력단 IPv6 패킷 포맷 구조는 첨부된 도 10에 도시된 바와 같으며, 출력단 IPv6 패킷 포맷에서 2바이트 PPP 헤더와 다음의 4바이트 DH_2는 네트워크 처리기나 패킷 처리기가 받는 순간 정보 획득 후 제거되는 필드이다. 만약, Flow Label 필드가 제로(Zero)이고, 다음 헤더(Next Header)가 TCP 또는 UDP이면 SPORT[7:4]가 Flow_label[19:16]에 복사되고, DPORT[15:0]이 Flow_label[15:0]으로 복사되어 재정렬된다.

<59> 그러면 상술한 바와 같은 패킷 처리 장치에서 패킷을 처리하기 위한 방법에 대해 첨부된 도면을 참조하여 구체적으로 설명하기로 한다.

<60> 도 11a 내지 도 11d는 본 발명의 실시예에 따라 패킷 처리 장치에서의 패킷 처리 절차를 도시한 도면이다.

<61> 상기 도 11a 내지 도 11d를 참조하면, 패킷 처리 장치는 410단계에서 입력된 패킷으로부터 패킷 헤더 정보를 추출한 후 420단계에서 제외 플래그(Except Flag)가 세팅 되어 있는지를 확인한다. 확인 결과, 상기 제외 플래그가 세팅되어 있으면 패킷 처리 장치는 470단계로 진행하고, 그렇지 않은 경우 430단계에서 패킷 타입(Packet Type)이 IPv4 패킷인지를 확인한다.

<62> 상기 430단계에서 확인한 결과, IPv4 패킷인 경우에는 431단계에서 IPv4 패킷 헤더를 파싱(Parsing) 및 분석한 후, 432단계에서 분석 결과에 따라 DH_1 v4 헤더 필드를 세팅한다. 그런 다음 패킷 처리 장치는 433단계에서 DH_1 v4 헤더 필드가 유효한지를 확인하여 유효하면 434단계에서 헤더 체크섬을 계산하여 계산된 결과를 상기

IPv4 패킷에 갱신하고, 435단계에서 상기 IPv4 패킷의 TH 및 DH_1 v4 헤더를 갱신한다. 이후, 패킷 처리 장치는 436단계에서 상기 IPv4 패킷이 802.2 LLC 패킷이면 패딩을 제거하기 위해 마스크(Mask)를 조정한 후 440단계로 진행한다. 반면, 상기 DH_1 v4 헤더 필드가 유효하지 않으면, 437단계에서 상기 TH의 제외(Except) 비트를 세팅한 후 패킷 처리 동작을 종료한다.

<63> 상기 430단계에서 확인한 결과, 입력된 패킷이 IPv4 패킷이 아니면 패킷 처리 장치는 440단계에서 패킷 타입이 MPLS 패킷인지를 확인하여 입력된 패킷이 MPLS 패킷이면 441단계에서 TH에서 num_label[2:0]을 세팅한다. 그런 다음 442단계에서 패킷 처리 장치는 상기 MPLS 패킷의 DH_1 v4 헤더에서 LPOP 비트를 세팅하고, 443단계에서 TH에서 ahl[7:0]과 loc[1:0]을 갱신한다. 이후, 패킷 처리 장치는 444단계에서 상기 갱신된 MPLS 패킷에 DH_2 v4 또는 v6 헤더를 삽입한 후, 445단계에서 MPLS 패킷의 헤더를 재정렬(Rearrangement)한다.

<64> 상기 440단계에서 확인한 결과, 입력된 패킷이 MPLS 패킷이 아니면 패킷 처리 장치는 450단계에서 프로토콜 필드가 터널인지를 확인하여 프로토콜 필드가 터널이면, 입력된 패킷이 터널 패킷으로서, 451단계에서 터널 테이블 룩업을 수행한다. 그런 다음 패킷 처리 장치는 452단계에서 목적지 주소(Destination Address)가 매치되어 있는지를 확인하여 매치되어 있지 않은 경우 437단계로 진행하고, 매치된 경우에는 453단계에서 MF 필드가 "1" 즉, 세팅되었는지를 확인한다. 확인 결과, MF가 세팅되어 있으면 패킷 처리 장치는 마스크를 조정하기 위해 470단계로 진행하고, 그렇지 않은 경우에는 454단계에서 IPv4 터널 헤더를 디캡슐레이션(Decapsulation)시킨 후 455단계에서 상기 터널 패킷의 헤더를 정렬(Alignment)한 후 460단계로 진행한다.

<65> 상기 450단계에서 확인한 결과, 터널 패킷이 아닌 경우 패킷 처리 장치는 460단계에서 입력된 패킷의 타입이 IPv6인지 확인한다. 확인 결과, IPv6 패킷인 경우 패킷 처리 장치는 461단계에서 IPv6 패킷 헤더를 파싱(Parsing)하고, 룩업을 위한 다양한 키 값들을 추출한 후, IPv6 목적지 주소를 사용하여 LPM(Longest Prefix Matching) 룩업을 수행한다. 그런 다음 패킷 처리 장치는 463단계에서 다양한 키 값들을 이용하여 QoS/DoS 룩업을 수행하고, 464단계에서 해쉬 알고리즘을 통하여 20비트 해쉬 값을 생성한다. 이후, 465단계에서 패킷 처리 장치는 상기 IPv6 패킷의 DH_1 v6 헤더에 상기 생성된 해쉬 값을 삽입하고, 466단계에서 상기 IPv6 패킷의 DH_1 v6 아래에 상용 NSE로부터 획득한 Route ID, QoS/DoS ID를 삽입한 후, 467단계에서 IPv6 패킷의 헤더를 재정렬(rearrangement)한다.

<66> 상기 460단계에서 확인한 결과, IPv6 패킷이 아니면 470단계에서 불필요한 바이트를 제거하기 위해 마스크(Mask)를 조정한다. 그런 다음 패킷 처리 장치는 471단계에서 패킷 타입에 따라 2 바이트 PPP 헤더를 세팅하고, 472단계에서 상기 세팅된 PPP 헤더를 삽입하고, TH를 제거한다. 이후, 패킷 처리 장치는 473단계에서 상기 PPP 헤더가 삽입되고 TH가 제거된 패킷에서 불필요한 바이트를 제거하고, EOP(End of Packet) 조정 및 패킷을 병합(Packing)한 후 패킷 처리 동작을 종료한다.

<67> 상술한 바와 같은 본 발명은 종래 네트워크 처리기나 패킷 처리기가 이더넷을 지원하지 않고 PPP와 MPLS 프로토콜만을 처리하도록 구성되었을 때, 네트워크 처리기나 패킷처리기의 추가적인 처리 작업 없이 다양한 이더넷 패킷을 처리 할 수 있으며, 기본적인 패킷 처리 처리를 순수 하드웨어 로직으로 처리한다.

<68> 또한, 본 발명은 QoS 및 다양한 프로토콜 등을 지원하기 위하여 라인속도(Wire Speed)를 저하시키지 않는 전송 헤더의 추가 및 패킷 헤더를 변형시킴으로써, 네트워크 처리기의 사용 효율을 최적화시킬 수 있다.

<69> 게다가, 본 발명은 프로토콜 엔진들이 모듈화되어 있어 특정 프로토콜만을 탑재할 수가 있어 시스템의 변경을 필요로 하지 않고, 특정화된 프로토콜을 지원하는 시스템을 구성할 수 있다.

<70> 한편, 본 발명의 상세한 설명에서는 구체적인 실시 예에 관하여 설명하였으나, 본 발명의 범위에서 벗어나지 않는 한도 내에서 여러 가지 변형이 가능함은 물론이다. 그러므로 본 발명의 범위는 설명된 실시 예에 국한되어 정해져서는 안되며 후술하는 발명청구의 범위뿐 만 아니라 이 발명청구의 범위와 균등한 것들에 의해 정해져야 한다.

도면의 간단한 설명

<71> 도 1은 네트워크 처리기를 이용한 IP 패킷 처리 장치의 일반적인 구성을 도시한 도면,

<72> 도 2는 일반적인 패킷 처리 장치의 다른 예를 도시한 도면,

<73> 도 3은 일반적인 PPP 데이터 프레임 처리 장치의 구성을 도시한 도면,

<74> 도 4는 일반적인 IPv4 기본 헤더 포맷 구조를 도시한 도면,

<75> 도 5는 일반적인 IPv6 기본 헤더 포맷 구조를 도시한 도면,

<76> 도 6은 본 발명의 실시예에 따른 패킷 처리 장치의 구성을 도시한 도면,

<77> 도 7은 본 발명의 실시예에 따른 프로토콜 번역부의 일예를 도시한 도면,

<78> 도 8은 본 발명의 실시예에 따른 전송 헤더의 일예를 도시한 도면,

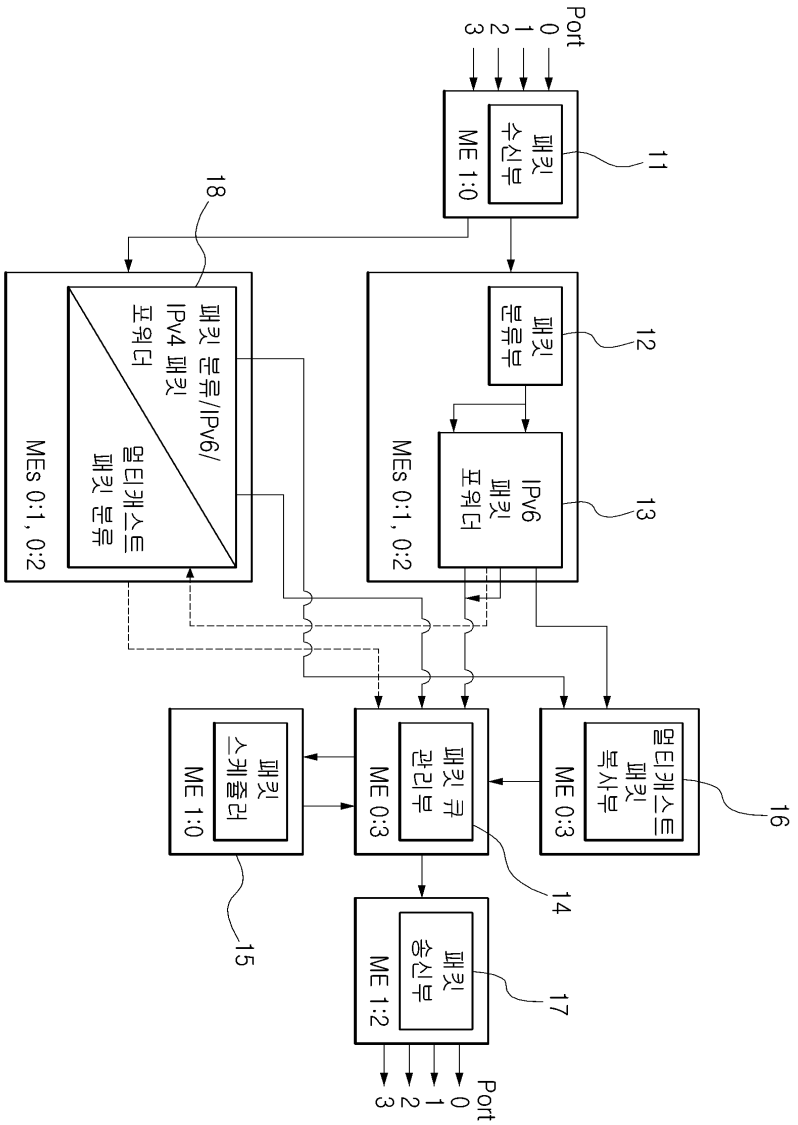
<79> 도 9는 본 발명의 실시예에 따라 패킷 처리 장치에서의 패킷 변환 과정을 도시한 도면,

<80> 도 10은 본 발명의 실시예에 따른 패킷 처리 장치의 출력단 IPv6 패킷 포맷 구조를 도시한 도면,

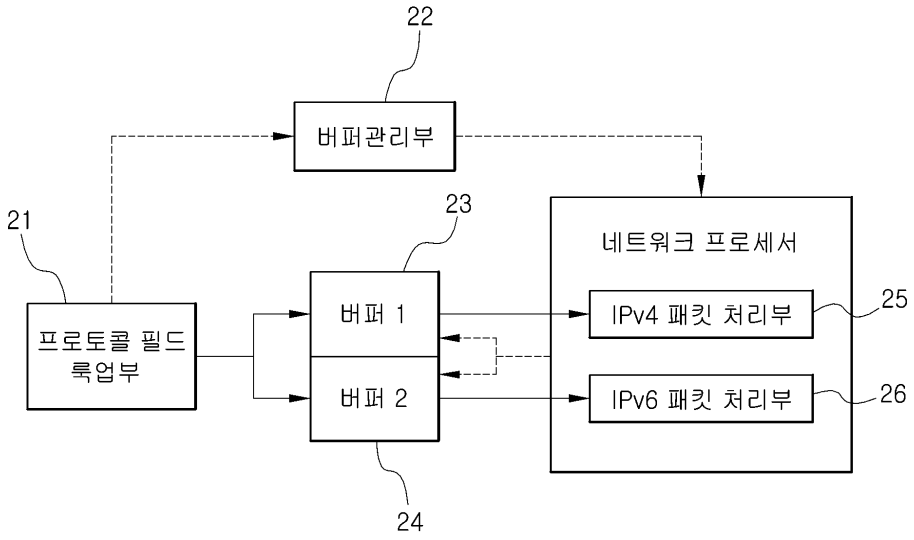
<81> 도 11a 및 도 11d는 본 발명의 실시예에 따라 패킷 처리 장치에서의 패킷 처리 절차를 도시한 도면.

도면

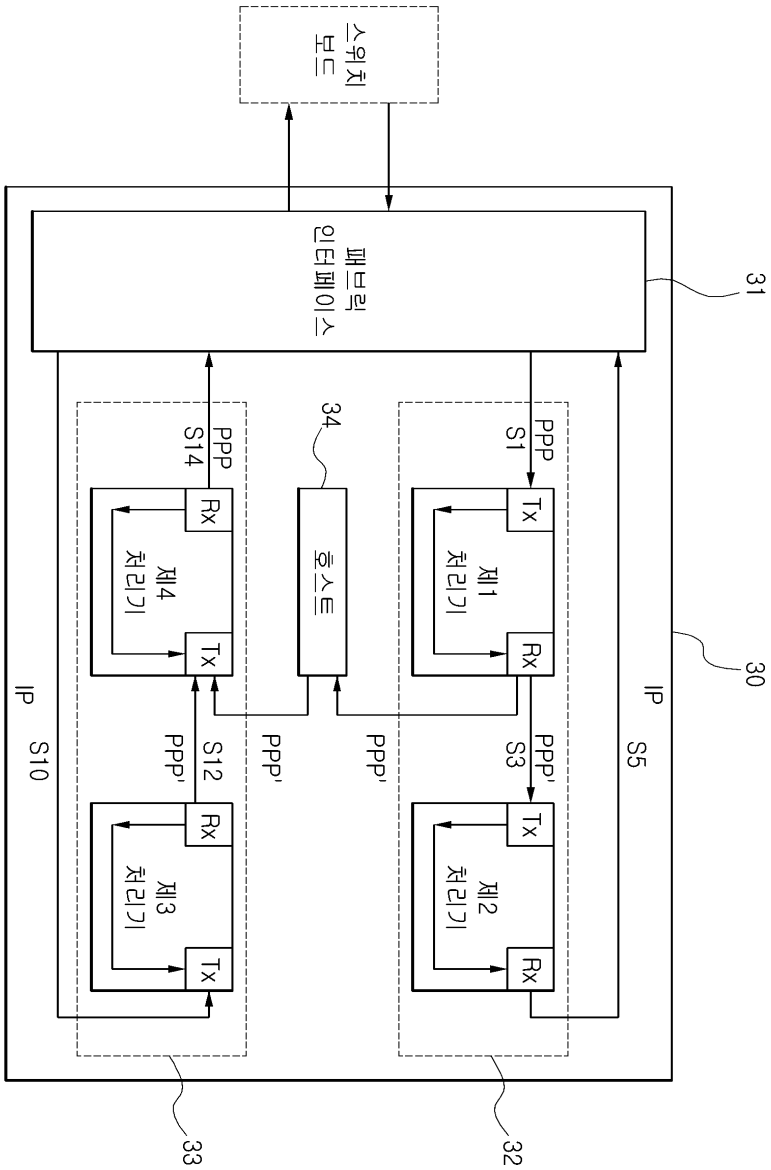
도면1



도면2


22
버퍼관리부
21
23
네트워크 프로세서
프로토콜 필드
룩업부
버퍼 1
IPv4 패킷 처리부
25
버퍼 2
IPv6 패킷 처리부
26
24

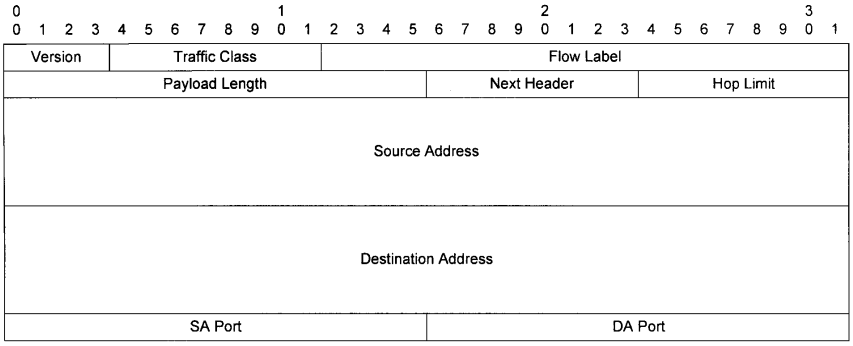
도면3



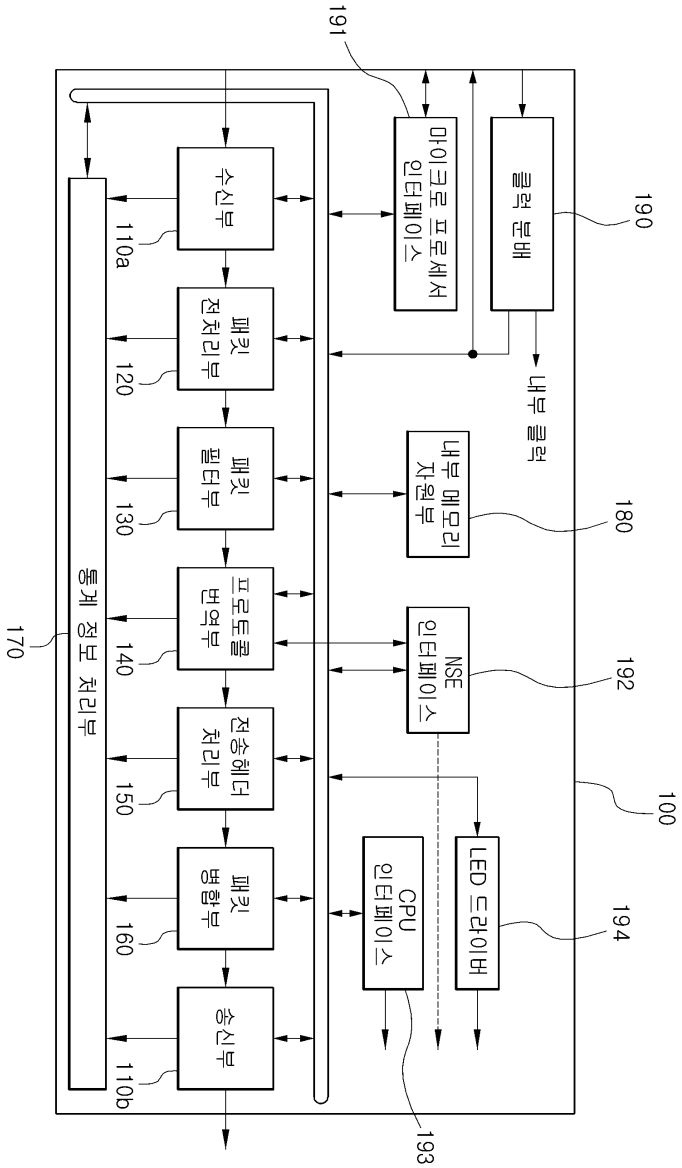
도면4

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live		Protocol	Header Checksum
Source Address			
Destination Address			
Options			Padding

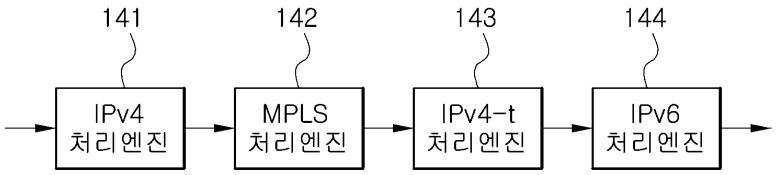
도면5


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
SA Port
DA Port

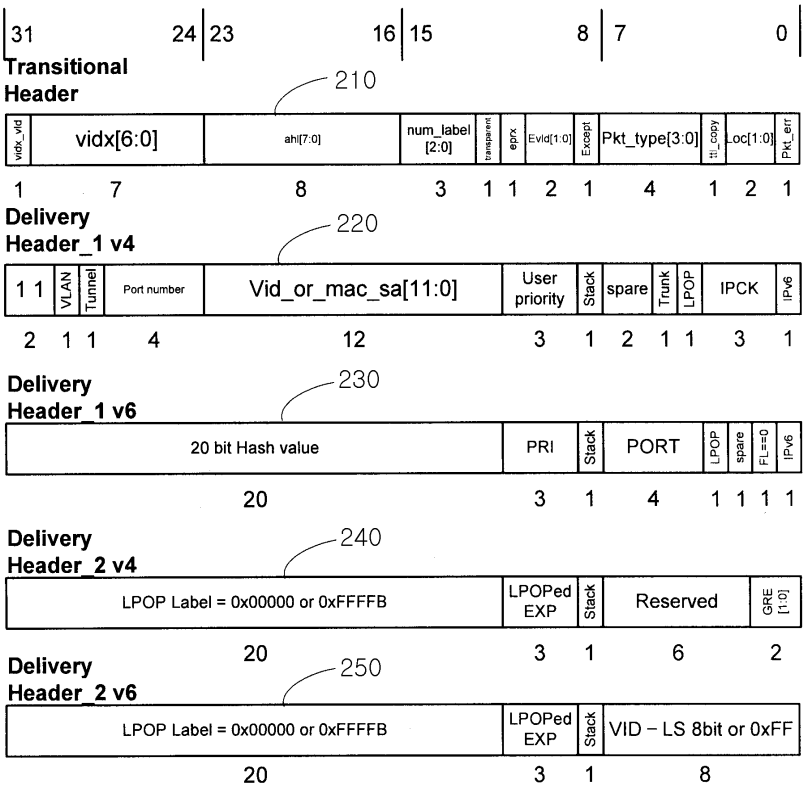
도면6


191
마이크로 프로세서 인터페이스
클럭 분배
190
수신부
110a
패킷 전처리부
120
패킷 필터부
130
프로토콜 번역부
140
전송헤더 처리부
150
패킷 병합부
160
송신부
110b
통계 정보 처리부
170
내부 클럭
내부 메모리 자원부
180
NSE 인터페이스
192
100
CPU 인터페이스
193
LED 드라이버
194

도면7


141
142
143
144
IPv4 처리엔진
MPLS 처리엔진
IPv4-t 처리엔진
IPv6 처리엔진

도면8


31
24
23
16
15
8
7
0
Transitional Header
210
vidx_vld
vidx[6:0]
ahl[7:0]
num_label [2:0]
transparent
eprx
Evid[1:0]
Except
Pkt_type[3:0]
ttl_copy
Loc[1:0]
Pkt_err
1 7 8 3 1 1 2 1 4 1 2 1
Delivery Header_1 v4
220
1 1
VLAN
Tunnel
Port number
Vid_or_mac_sa[11:0]
User priority
Stack
spare
Trunk
LPOP
IPCK
IPv6
2 1 1 4 12 3 1 2 1 1 3 1
Delivery Header_1 v6
230
20 bit Hash value
PRI
Stack
PORT
LPOP
spare
FL=0
IPv6
20 3 1 4 1 1 1 1
Delivery Header_2 v4
240
LPOP Label = 0x00000 or 0xFFFFB
LPOPed EXP
Stack
Reserved
GRE [1:0]
20 3 1 6 2
Delivery Header_2 v6
250
LPOP Label = 0x00000 or 0xFFFFB
LPOPed EXP
Stack
VID – LS 8bit or 0xFF
20 3 1 8

도면9

IPv6/MPLS/Ethernet – 802.3 (Ethernet version 2)

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

SFD	0x55	0x55	0x55	0x55	0x55	0x55	0xD5	0
DA_0	DA_1	DA_2	DA_3	DA_4	DA_5	SA_0	SA_1	1
SA_2	SA_3	SA_4	SA_5	TYPE_0	TYPE_1	MPLS_0	MPLS_1	2
MPLS_2	MPLS_3	V/TC_0	TC_1/FL0	FL_1	FL_2	LEN_0	LEN_1	3
NHDR	HLIM	SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	4
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	5
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	6
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	7
DA6_14	DA6_15							8

Packet Format A — 310

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

Transitional Header				Delivery Header_1 v4				
DA_0	DA_1	DA_2	DA_3	DA_4	DA_5	SA_0	SA_1	1
SA_2	SA_3	SA_4	SA_5	TYPE_0	TYPE_1	MPLS_0	MPLS_1	2
MPLS_2	MPLS_3	V/TC_0	TC_1/FL0	FL_1	FL_2	LEN_0	LEN_1	3
NHDR	HLIM	SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	4
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	5
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	6
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	7
DA6_14	DA6_15							8

Packet Format B — 320

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

Transitional Header				Delivery Header_2 v4				
Delivery Header_1 v4				DA_4	DA_5	SA_0	SA_1	1
SA_2	SA_3	SA_4	SA_5	TYPE_0	TYPE_1	MPLS_0	MPLS_1	2
MPLS_2	MPLS_3	V/TC_0	TC_1/FL0	FL_1	FL_2	LEN_0	LEN_1	3
NHDR	HLIM	SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	4
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	5
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	6
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	7
DA6_14	DA6_15							8

Packet Format C — 330

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

Transitional Header				Delivery Header_2 v6				
Delivery Header_1 v6				MPLS_0	MPLS_1	MPLS_2	MPLS_3	1
				NHDR	TC_1/FL0	FL_1	FL_2	LEN_0	LEN_1	3
		HLIM	SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	4
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	5
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	6
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	7
DA6_14	DA6_15							8

Packet Format D — 340

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

Transitional Header					Delivery Header_2 v6				
Delivery Header_1 v6					MPLS_0	MPLS_1	MPLS_2	MPLS_3	1
R	IP	Route ID		DOS Index	QOS Index	Sport[15:4]		TC	Sport [3:0]	2
					FL_1	FL_2				3
NHDR	HLIM									4
		SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	5
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	6
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	7
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	8
DA6_14	DA6_15							9

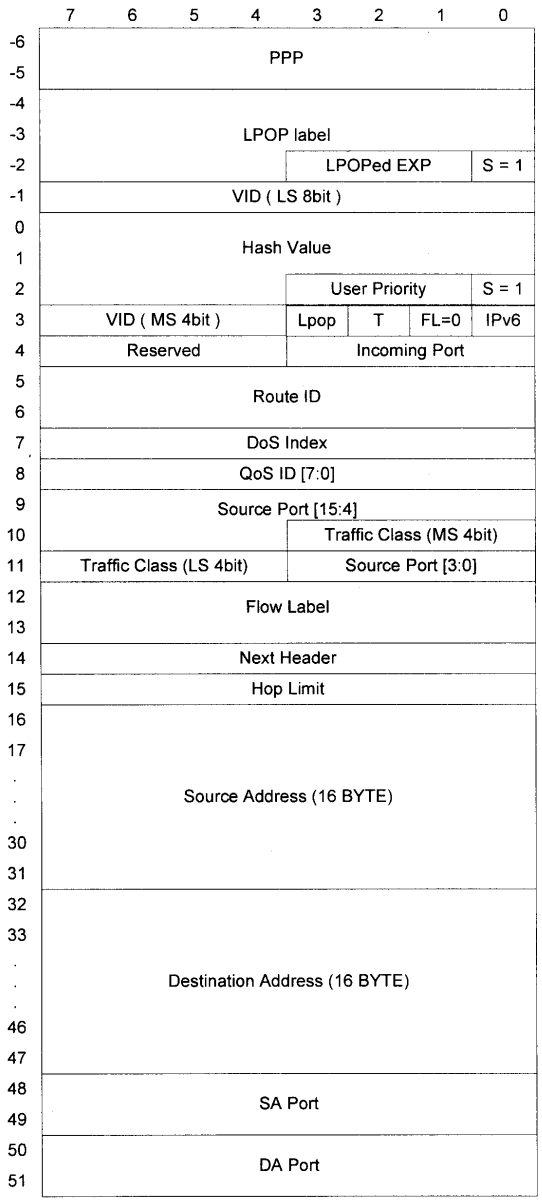
Packet Format E — 350

BYTE LANE 7 6 5 4 3 2 1 0 — [63:56] [55:48] [47:40] [39:32] [31:24] [23:16] [15:8] [7:0] — Clock Cycle

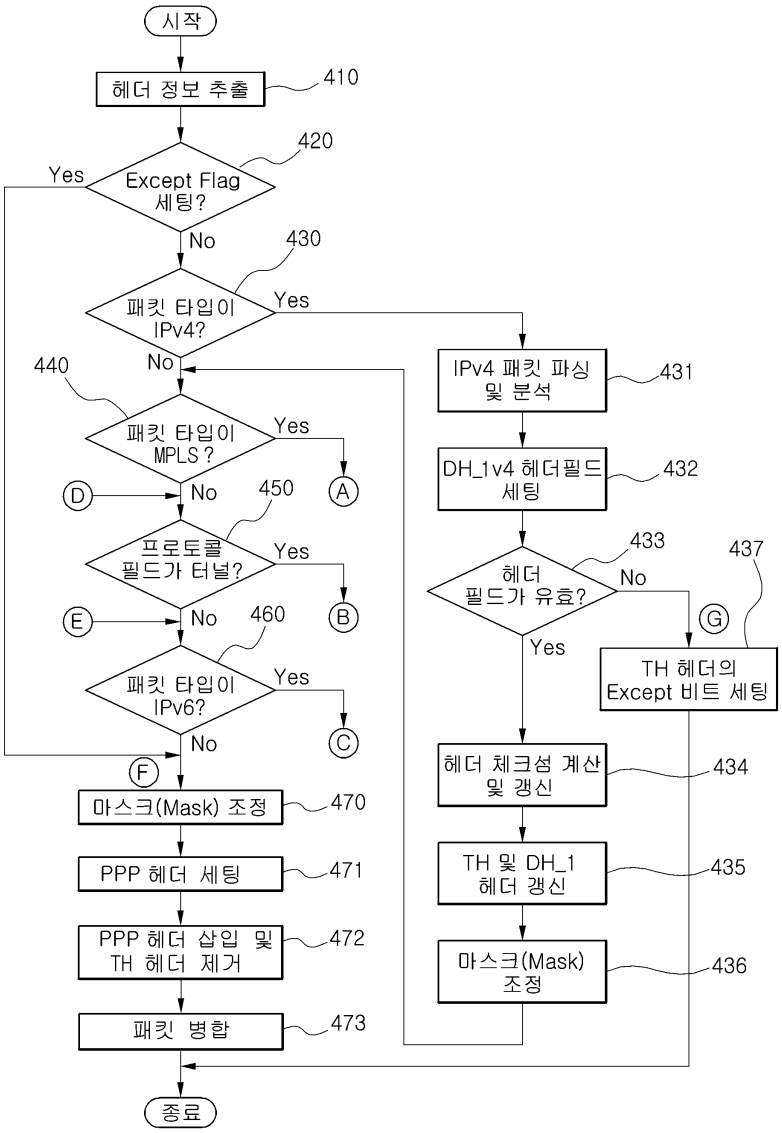
PPP Header		DH_2 v6				DH_1 v6			
DH_1 v6		MPLS_0	MPLS_1	MPLS_2	MPLS_3	R	IP	Route ID	1
Route ID	DOS Index	QOS Index	Sport[15:4]		TC	Sport [3:0]	FL_1	FL_2	2
NHDR	HLIM	SA6_0	SA6_1	SA6_2	SA6_3	SA6_4	SA6_5	3
SA6_6	SA6_7	SA6_8	SA6_9	SA6_10	SA6_11	SA6_12	SA6_13	4
SA6_14	SA6_15	DA6_0	DA6_1	DA6_2	DA6_3	DA6_4	DA6_5	5
DA6_6	DA6_7	DA6_8	DA6_9	DA6_10	DA6_11	DA6_12	DA6_13	6
DA6_14	DA6_15	SA Port		DA Port				7

Packet Format F — 360

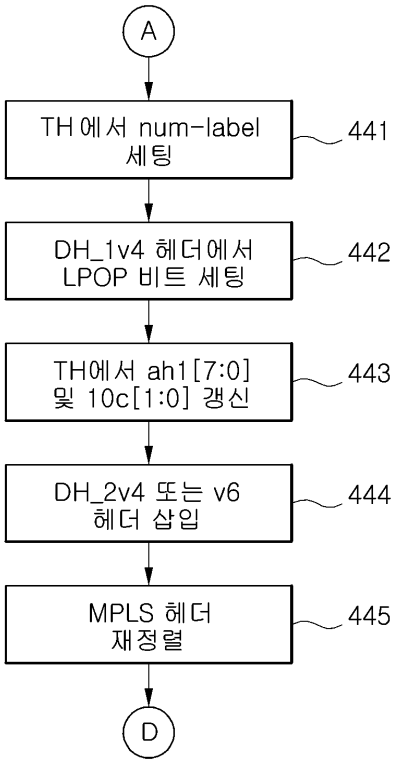
도면10


7 6 5 4 3 2 1 0
-6
-5
PPP
-4
-3
LPOP label
-2
LPOPed EXP
S = 1
-1
VID (LS 8bit)
0
1
Hash Value
2
User Priority
S = 1
3
VID (MS 4bit)
Lpop
T
FL=0
IPv6
4
Reserved
Incoming Port
5
6
Route ID
7
DoS Index
8
QoS ID [7:0]
9
Source Port [15:4]
10
Traffic Class (MS 4bit)
11
Traffic Class (LS 4bit)
Source Port [3:0]
12
13
Flow Label
14
Next Header
15
Hop Limit
16
17
Source Address (16 BYTE)
30
31
32
33
Destination Address (16 BYTE)
46
47
48
49
SA Port
50
51
DA Port

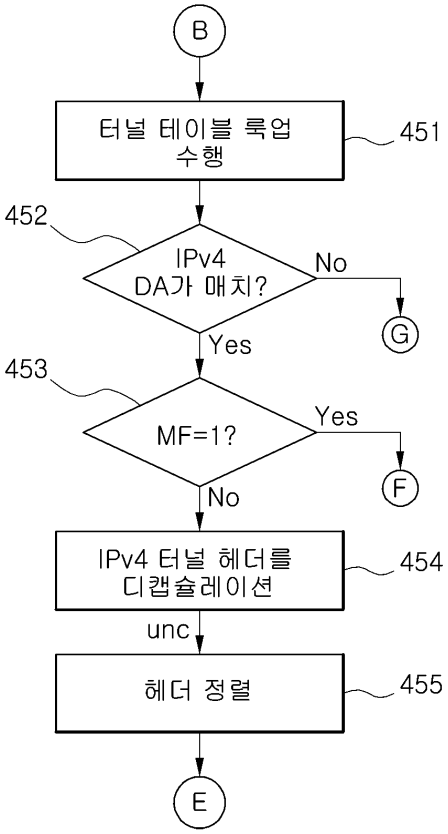
도면11a


시작
헤더 정보 추출
410
Except Flag 세팅?
420
Yes
No
패킷 타입이 IPv4?
430
Yes
No
IPv4 패킷 파싱 및 분석
431
DH_1v4 헤더필드 세팅
432
헤더 필드가 유효?
433
No
Yes
437
TH 헤더의 Except 비트 세팅
헤더 체크섬 계산 및 갱신
434
TH 및 DH_1 헤더 갱신
435
마스크(Mask) 조정
436
440
패킷 타입이 MPLS?
Yes
No
A
D
450
프로토콜 필드가 터널?
Yes
No
B
E
460
패킷 타입이 IPv6?
Yes
No
C
F
G
마스크(Mask) 조정
470
PPP 헤더 세팅
471
PPP 헤더 삽입 및 TH 헤더 제거
472
패킷 병합
473
종료

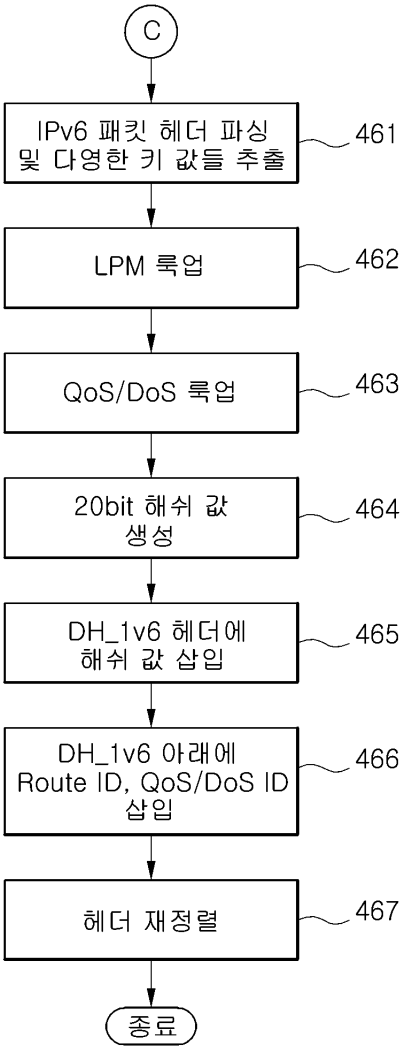
도면11b


A
TH 에서 num-label 세팅
441
DH_1v4 헤더에서 LPOP 비트 세팅
442
TH에서 ah1[7:0] 및 10c[1:0] 갱신
443
DH_2v4 또는 v6 헤더 삽입
444
MPLS 헤더 재정렬
445
D

도면11c


B
터널 테이블 룩업 수행
451
452
IPv4 DA가 매치?
No
G
Yes
453
MF=1?
Yes
F
No
IPv4 터널 헤더를 디캡슐레이션
454
unc
헤더 정렬
455
E

도면11d


C
IPv6 패킷 헤더 파싱 및 다영한 키 값들 추출
461
LPM 룩업
462
QoS/DoS 룩업
463
20bit 해쉬 값 생성
464
DH_1v6 헤더에 해쉬 값 삽입
465
DH_1v6 아래에 Route ID, QoS/DoS ID 삽입
466
헤더 재정렬
467
종료