

	(19) 대한민국특허청(KR) (12) 공개특허공보(A)	(11) 공개번호 10-2014-0100348 (43) 공개일자 2014년08월14일
(51) 국제특허분류(Int. Cl.) H04L 12/741 (2013.01) (21) 출원번호 10-2013-0013548 (22) 출원일자 2013년02월06일 심사청구일자 2013년02월06일	(71) 출원인 한밭대학교 산학협력단 대전광역시 유성구 동서대로 125 (덕명동) (72) 발명자 김은기 대전광역시 유성구 동서대로 125 백민호 서울 성북구 장월로1길 28, 108동 1401호 (상월곡동, 동아에코빌아파트) (74) 대리인 원성수, 강세창, 박종경, 추혁 (73) 등록일자 2014년08월14일 (54) 발명의 명칭 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법	

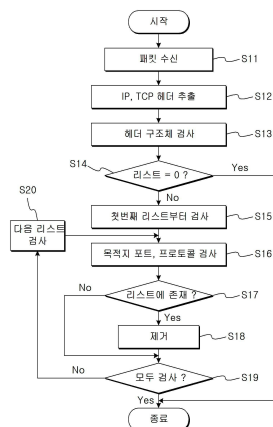
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법

(57) 요약

본 발명은 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법에 관한 것으로서, 수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, filter_list 구조체를 필터 테이블 리스트에 추가하고, 필터 테이블 리스트가 존재할 경우, 프로토콜 번호와 목적지 포트번호가 존재하는지 여부를 검사하고, 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 불필요한 패킷을 네트워크 드라이버 단에서 제거함으로써, 수신한 불필요한 패킷을 소켓 버퍼에 저장하고, 패킷의 프로토콜 타입에 맞는 계층에서 패킷을 제거하는 등의 비효율적인 동작을 수행하지 않으므로 스마트 단말기에서의 오버헤드를 줄일 수 있다.

대표도 - 도11



(72) 발명자

강정하

대전광역시 유성구 동서대로 125

김현수

대전 서구 관저로 51, 504동 1003호 (관저동, 구봉
마을5단지)

이 발명을 지원한 국가연구개발사업

과제고유번호 201301590001

부처명 교육부

연구사업명 2013년 지역혁신인력양성사업

연구과제명 IEEE p1609.2 Version 2 규격의 WAVE 보안 시스템 구현

기 여 율 1/1

주관기관 한밭대학교 산학협력단

연구기간 2013.05.01 ~ 2016.04.30

특허청구의 범위

청구항 1

수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 필터 테이블 작성부;

상기 필터 테이블 리스트가 존재할 경우, 상기 프로토콜 번호와 상기 목적지 포트번호가 존재하는지 여부를 검사하는 필터 테이블 검사부; 및

상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 상기 불필요한 패킷을 네트워크 드라이버 단에서 제거하는 패킷 제거부를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 장치.

청구항 2

제1항에 있어서,

상기 filter_list 구조체는,

상기 filter_list 구조체 간 링크를 연결해 주는 list 멤버;

상기 패킷의 목적지 포트 번호를 저장하고 있는 dport 멤버; 및

상기 패킷의 프로토콜 번호를 저장하고 있는 protocol 멤버를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 장치.

청구항 3

불필요한 패킷의 필터 테이블 리스트를 작성하여 소프트웨어적으로 접근 가능한 계층인 무선 LAN의 MAC 계층에서 상기 불필요한 패킷을 제거하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

청구항 4

수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계;

상기 필터 테이블 리스트가 존재할 경우, 상기 프로토콜 번호와 상기 목적지 포트번호가 존재하는지 여부를 검사하는 단계; 및

상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 상기 불필요한 패킷을 네트워크 드라이버 단에서 제거하는 단계를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

청구항 5

제4항에 있어서,

상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계는,

소켓 버퍼 구조체를 수신하는 단계;

소켓 버퍼로부터 IP 헤더 값과 TCP 헤더 값을 추출하는 단계;

상기 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하는 단계; 및

상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

청구항 6

제5항에 있어서,

상기 필터 테이블 리스트가 없을 경우에는 리스트를 추가하고, 첫 번째 리스트가 추가될 때 헤드 구조체를 생성하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

청구항 7

제4항에 있어서,

상기 불필요한 패킷을 제거하는 단계는,

수신된 패킷으로부터 IP 헤더와 TCP 헤더를 추출하는 단계;

상기 필터 테이블 리스트가 존재하면, 상기 수신한 패킷의 프로토콜 번호와 목적지 포트 번호가 상기 필터 테이블 리스트에 존재하는지 검사하는 단계; 및

상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재하면, 상기 불필요한 패킷을 제거하는 단계를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

청구항 8

제4항에 있어서,

상기 filter_list 구조체는,

상기 filter_list 구조체 간 링크를 연결해 주는 list 멤버;

상기 불필요한 패킷의 목적지 포트 번호를 저장하고 있는 dport 멤버; 및

상기 불필요한 패킷의 프로토콜 번호를 저장하고 있는 protocol 멤버를 포함하는 수신된 데이터에서의 불필요한 패킷 제거 방법.

명세서

기술분야

[0001] 본 발명은 수신된 데이터에서 불필요한 패킷 제거 기법에 관한 것으로, 더욱 상세하게는 불필요한 패킷이 제거되는 해당 계층에 전달하기 전에 수신된 데이터에서 불필요한 패킷을 제거하는 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법에 관한 것이다.

배경기술

[0002] 스마트 단말기의 보급이 증가하면서 무선 LAN(Local Area Network)을 이용한 무선 인터넷의 이용률이 점점 증가하고 있다. 스마트 단말기 사용자들은 이동통신망을 이용하여 인터넷을 사용할 수도 있지만, 비용적인 면을 고려하여 대부분 AP(Access Point)에 접속하여 데이터 통신 서비스를 이용한다. AP에 접속하여 무선 LAN을 사용하는 스마트 단말기는 AP와 수많은 패킷을 송수신한다. 스마트 단말기는 사용자가 이용하는 무선 LAN 서비스에 필요한 패킷뿐만 아니라, 필요 없는 패킷도 다수 수신한다.

[0003] 불필요한 패킷을 수신한 스마트 단말기는 패킷을 바로 제거하지 않고, 패킷의 프로토콜 타입에 맞는 계층에서 패킷을 제거한다. 이에, 불필요한 패킷을 처리하는 동작은 반복적으로 계속 수행된다. 이러한 동작은 스마트 단

말기에 있어서 비효율적인 작업에 해당한다. 스마트 단말기에 많은 양의 불필요한 패킷이 수신되면, 불필요한 패킷에 대한 처리 시간은 스마트 단말기의 오버헤드 원인이 될 수 있다.

- [0004] 그러면 여기서, 스마트 단말기에서 필요로 하지 않는 패킷들의 대표적인 예를 간략하게 설명한다.
- [0005] 첫 번째 예로서, 멀티캐스트 패킷이나 브로드캐스트 패킷을 수신한 경우이다. 멀티캐스트 패킷은 특정 그룹을 대상으로 데이터가 전송된다. 멀티캐스트 패킷을 수신한 커널은 해당 패킷의 그룹에 해당하는지를 판단하여 패킷의 수신여부를 결정한다. 브로드캐스트는 네트워크에 연결된 모든 호스트에게 패킷을 전송하는 방식으로, 호스트의 맥 어드레스와 일치하지 않아도 브로드캐스트 패킷을 수신한다[1, 2]. 스마트 단말기는 수신한 멀티캐스트 패킷이나 브로드캐스트 패킷이 불필요하면 네트워크 계층과 전송 계층에서 제거한다.
- [0006] 두 번째 예로서, 스마트 단말기에서 동작하지 않는 어플리케이션에 대한 TCP(Transmission Control Protocol) 패킷을 수신한 경우이다. TCP 패킷을 수신한 스마트 단말기는 패킷에 대한 어플리케이션이 동작하는지를 검사한다. 커널은 TCP 패킷에 대한 어플리케이션이 동작하지 않는다면 패킷을 전송 계층에서 제거한다.[1, 2].
- [0007] 세 번째 예로서, 라우팅 프로토콜 패킷을 수신한 경우이다. 라우팅 프로토콜은 네트워크 전반에 걸쳐 출발지에서 목적지까지 패킷의 종단 간 경로를 결정해 주는 프로토콜이다. 스마트 단말기는 라우팅 프로토콜 패킷이 필요 없기 때문에, 수신한 라우팅 프로토콜 패킷은 네트워크 계층에서 제거된다[3, 4].
- [0008] 도 1은 AP로부터 스마트 단말기가 수신한 불필요한 패킷의 수를 나타낸 그래프이다.
- [0009] 도 1을 참조하면, 스마트 단말기가 1시간 동안 AP로부터 SSDP(Simple Service Discovery Protocol), HSRP(Hot Standby Router Protocol), OSPF(Open Shortest Path First), NBNS(Netbios Name Service) 패킷을 수신한 수를 보여주고 있다. HSRP 패킷과 OSPF 패킷은 라우팅 프로토콜 패킷이고, NBNS 패킷은 브로드캐스트 패킷이며, SSDP 패킷은 멀티캐스트 패킷이다. 이와 같은 패킷들은 안드로이드 기반의 스마트 단말기에서는 불필요한 패킷에 해당한다.
- [0010] 여기에서는, 4개의 패킷만을 예시하고 있지만, 이 패킷 이외에도 스마트 단말기에 불필요한 패킷이 다양하게 존재한다. 하지만 AP마다 각기 다른 환경 설정 값을 가질 수 있기 때문에, 도 1의 결과는 하나의 예에 불과하며, 다양한 환경에서 다양한 결과가 나올 수 있다.
- [0011] 이와 같이 스마트 단말기의 오버헤드 원인의 원인이 되는 불필요한 패킷을 제거하는 새로운 제거 알고리즘이 필요하다 할 것이다.

선행기술문헌

비특허문헌

- [0012] (비특허문헌 0001) [1] Lydia Parziale, David T.Britt, Chuck Davis, Jason Forrester and Dr.Wei Liu, TCP/IP Tutorial and Technical Overview, ibm.com/redbooks, December 2006
- (비특허문헌 0002) [2] Forouzan and Behrouz A, TCP/IP Protocol Suite 4th Edition, McGraw Hill, April 2009
- (비특허문헌 0003) [3] Klaus Wehrle and Frank Pöhlke, The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel, Prentice Hall, August 2004
- (비특허문헌 0004) [4] Christian Benvenuti, Understanding Linux Network Internals, O'Reilly, December 2005

발명의 내용

해결하려는 과제

[0013] 따라서, 본 발명은 상기한 종래 기술의 문제점을 해결하기 위해 이루어진 것으로서, 본 발명의 목적은 스마트 단말기의 비효율적인 작업을 개선하기 위해, 수신된 불필요한 패킷을 해당 계층에 전달하지 않고, S/W적으로 접근 가능한 하위 계층인 무선 LAN의 MAC 계층에서 불필요한 패킷을 제거해 주는 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법을 제공하는데 있다.

과제의 해결 수단

[0014] 상기와 같은 목적을 달성하기 위한 본 발명의 수신된 데이터에서의 불필요한 패킷 제거 장치는, 수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 필터 테이블 작성부; 상기 필터 테이블 리스트가 존재할 경우, 상기 프로토콜 번호와 상기 목적지 포트번호가 존재하는지 여부를 검사하는 필터 테이블 검사부; 및 상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 상기 불필요한 패킷을 네트워크 드라이버 단에서 제거하는 패킷 제거부를 포함하는 것을 특징으로 한다.

[0015] 한편, 본 발명의 수신된 데이터에서의 불필요한 패킷 제거 방법은, 불필요한 패킷의 필터 테이블 리스트를 작성하여 소프트웨어적으로 접근 가능한 계층인 무선 LAN의 MAC 계층에서 상기 불필요한 패킷을 제거하는 것을 특징으로 한다.

[0016] 좀 더 구체적으로, 본 발명의 수신된 데이터에서의 불필요한 패킷 제거 방법은, 수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계; 상기 필터 테이블 리스트가 존재할 경우, 상기 프로토콜 번호와 상기 목적지 포트번호가 존재하는지 여부를 검사하는 단계; 및 상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 상기 불필요한 패킷을 네트워크 드라이버 단에서 제거하는 단계를 포함하는 것을 특징으로 한다.

[0017] 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계는, 소켓 버퍼 구조체를 수신하는 단계; 소켓 버퍼로부터 IP 헤더 값과 TCP 헤더 값을 추출하는 단계; 상기 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하는 단계; 및 상기 filter_list 구조체를 필터 테이블 리스트에 추가하는 단계를 포함한다. 이때, 상기 필터 테이블 리스트가 없을 경우에는 리스트를 추가하고, 첫 번째 리스트가 추가될 때 헤드 구조체를 생성한다.

[0018] 상기 불필요한 패킷을 제거하는 단계는, 수신된 패킷으로부터 IP 헤더와 TCP 헤더를 추출하는 단계; 상기 필터 테이블 리스트가 존재하면, 상기 수신한 패킷의 프로토콜 번호와 목적지 포트 번호가 상기 필터 테이블 리스트에 존재하는지 검사하는 단계; 및 상기 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재하면, 상기 불필요한 패킷을 제거하는 단계를 포함한다.

발명의 효과

[0019] 상술한 바와 같이, 본 발명에 의한 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법에 따르면, 수신한 불필요한 패킷을 소켓 버퍼에 저장하고, 패킷의 프로토콜 타입에 맞는 계층에서 패킷을 제거하는 등의 비효율적인 동작을 수행하지 않으므로 스마트 단말기에서의 오버헤드를 줄일 수 있다.

도면의 간단한 설명

[0020] 도 1은 AP로부터 스마트 단말기가 수신한 불필요한 패킷의 수를 나타낸 그래프이다.

도 2는 데이터 링크 계층에서 패킷이 처리되는 함수의 호출 과정을 나타낸 흐름도이다.

도 3은 네트워크 계층의 패킷 전송 경로를 나타낸 흐름도이다.

도 4는 전송 계층에서의 패킷 전송 경로를 나타낸 흐름도이다.

도 5는 전송 계층에서의 UDP 패킷 전송 경로를 나타낸 흐름도이다.

도 6은 rt3070 칩셋의 드라이버 동작을 나타낸 흐름도이다.

도 7은 본 발명의 일실시예에 의한 불필요한 패킷의 자동 제거 알고리즘의 개념도이다.

도 8은 본 발명의 일실시예에 의한 필터 테이블의 구조와 저장 방식을 나타낸 구조도이다.

도 9는 본 발명의 일실시예에 의한 수신된 데이터에서의 불필요한 패킷 제거 장치의 제어회로블록도이다.

도 10은 본 발명의 일실시예에 의한 필터 테이블을 구성하는 과정을 나타낸 흐름도이다.

도 11은 본 발명의 일실시예에 의한 불필요한 패킷을 제거하는 과정을 나타낸 흐름도이다.

도 12는 제1AP에 연결하여 24시간 동안 수신한 불필요한 패킷의 리스트이다.

도 13은 제1AP로부터 시간별로 수신한 불필요한 패킷의 수를 나타낸 그래프이다.

도 14는 제2AP에 연결하여 24시간 동안 수신한 불필요한 패킷의 리스트이다.

도 15는 제2AP로부터 시간별로 수신한 불필요한 패킷의 수를 나타낸 그래프이다.

발명을 실시하기 위한 구체적인 내용

- [0021] 이하, 본 발명의 수신된 데이터에서의 불필요한 패킷 제거 장치 및 방법에 대하여 첨부된 도면을 참조하여 상세히 설명하기로 한다.
- [0022] 스마트 단말기의 운영체제로서, 안드로이드 OS(Android Operating System)가 많이 사용되고 있으며, 안드로이드 OS는 리눅스 커널 위에서 동작하므로, 본 발명에서는 일례로서 안드로이드 OS에서 불필요한 패킷을 제거하는 기법에 대해 설명하기로 한다. 물론, 본 발명이 다른 OS에도 적용될 수 있음은 당연할 것이다. 구체적인 예로서, 본 발명에서는 리눅스 커널 2.6.29 버전의 소스와 rt3070 칩셋을 사용하는 네트워크 디바이스 드라이버 소스에 적용하는 경우에 대해 설명하기로 한다.
- [0023] 먼저, 도 2 내지 도 6을 통해 패킷이 처리되는 기본적인 과정에 대해 설명하기로 한다.
- [0024] 도 2는 데이터 링크 계층에서 패킷이 처리되는 함수의 호출 과정을 나타낸 흐름도이다.
- [0025] 데이터 링크 계층은 NIC가 수신한 패킷을 소켓 버퍼에 저장하고, 패킷의 프로토콜에 맞게 상위 계층으로 전달해주는 역할을 한다.
- [0026] 도 2를 참조하면, NIC(Network Interface Card)가 패킷을 수신하면 인터럽트(Interrupt)가 발생하고, net_interrupt() 함수가 호출된다. 드라이버 인터럽트 핸들러는 수신된 패킷을 소켓 버퍼(Socket Buffer)에 저장하고, netif_rx() 함수를 호출한다. netif_rx() 함수는 상위 계층으로 전달하기 위해 do_softirq() 함수를 호출한다.
- [0027] 이어서, do_softirq() 함수는 큐(Queue)에 소켓 버퍼를 추가하고, 스케줄링 작업과 소프트웨어 인터럽트를 동작시키는 작업을 수행한다. 소프트웨어 인터럽트는 net_rx_action() 함수가 처리한다. net_rx_action() 함수는 네트워크 드라이버의 큐를 조사하고, 큐에 저장된 패킷들을 처리하는 netif_receive_skb() 함수를 호출한다. 마지막으로 netif_receive_skb() 함수는 패킷의 프로토콜에 맞는 상위 계층의 핸들러 함수를 호출한다.
- [0028] 그리고, 네트워크 계층은 데이터 링크 계층으로부터 전달 받은 소켓 버퍼와 라우팅 정보를 검사한다.
- [0029] 도 3은 네트워크 계층의 패킷 전송 경로를 나타낸 흐름도이다.
- [0030] 도 3을 참조하면, IPv4 패킷을 수신하는 경우에는 ip_rcv() 함수가 호출된다. ip_rcv() 함수는 패킷의 길이와 헤더, 체크섬 등의 패킷 검사를 수행한 후 ip_rcv_finish() 함수를 호출한다. ip_rcv_finish() 함수는 라우팅 정보를 검사한다. 라우팅 정보가 IP 포워딩인 경우 ip_forward() 함수를 호출하고, 멀티캐스트인 경우에는 ip_mr_input() 함수가 호출된다. 패킷이 내부로 수신되는 패킷이라면 ip_local_deliver() 함수가 호출된다. ip_local_deliver() 함수는 수신한 패킷이 조각(Fragment)으로 나누어서 수신되었다면, 패킷을 원래의 데이터로 재구성하는 작업을 수행한다. ip_local_deliver_finish() 함수는 전송 계층으로 패킷을 전달할 준비 작업을 수

행한다.

- [0031] 호스트의 그룹 정보와 관련이 없으면 멀티캐스트 패킷은 ip_rcv_finish() 함수에서 제거된다. 또한, 라우팅 프로토콜 관련 패킷들도 ip_rcv_finish() 함수에서 제거된다. 이러한 패킷은 스마트 단말기에 불필요한 패킷으로 분류되기 때문에, 본 발명에서 제안하는 필터 테이블 알고리즘 함수를 패킷이 제거되는 소스 부분에 추가해 줄 필요가 있다.
- [0032] 도 4는 전송 계층에서의 패킷 전송 경로를 나타낸 흐름도이다.
- [0033] 전송 계층은 TCP 패킷인 경우 소켓 버퍼의 에러를 검사하고 흐름 제어를 통해 어플리케이션 계층으로 전달해주는 역할을 한다.
- [0034] 도 4를 참조하면, ip_local_deliver_finish() 함수에서는 수신된 패킷의 프로토콜에 맞는 핸들러 함수를 호출한다. TCP 패킷은 tcp_v4_rcv() 함수가 호출된다. tcp_v4_rcv() 함수는 패킷의 타입과 헤더 길이, 체크섬(Checksum) 등을 검사한다. 다음으로 호출되는 tcp_v4_do_rcv() 함수는 TCP 세션이 성립된 상태(TCP_ESTABLISHED)이면 tcp_rcv_established() 함수를 호출한다. TCP_ESTABLISHED, TIME_WAIT를 제외한 TCP 상태 플래그에 따른 처리는 tcp_rcv_state_process() 함수에서 처리한다. tcp_rcv_established() 함수는 패킷의 처리 방법이 빠른 경로(Fast Path)와 느린 경로(Slow Path)로 나뉜다. 빠른 패스는 보통 정상적인 패킷이 처리되고, 재전송 요구를 해야 하거나 잘못된 TCP 옵션을 가진 비정상적인 패킷들은 느린 패스를 이용하게 된다.
- [0035] tcp_v4_rcv() 함수는 __inet_lookup_skb() 함수를 호출하여 소켓 검사를 수행한다. __inet_lookup_skb() 함수는 수신한 패킷의 해당하는 소켓을 찾지 못하면, 그 패킷은 제거한다. 이러한 패킷은 스마트 단말기에 불필요한 패킷으로 분류되기 때문에, 본 발명에서 제안하는 필터 테이블 알고리즘 함수를 패킷이 제거되는 소스 부분에 추가해줄 필요가 있다.
- [0036] 그리고, 스마트 단말기가 UDP 패킷을 수신하면 ip_local_deliver_finish() 함수는 udp_rcv() 함수를 호출한다.
- [0037] 도 5는 전송 계층에서의 UDP 패킷 전송 경로를 나타낸 흐름도이다.
- [0038] 도 5를 참조하면, udp_rcv() 함수는 바로 __udp4_lib_rcv() 함수를 호출한다. __udp4_lib_rcv() 함수는 UDP 패킷의 길이와 체크섬을 검사한다. 수신한 패킷이 멀티캐스트 패킷이나 브로드캐스트 패킷이면 __udp4_lib_rcv() 함수는 __udp4_lib_mcast_deliver() 함수를 호출하고, 멀티캐스트 패킷이나 브로드캐스트 패킷이 아닌 경우에는 소켓 검사를 수행하는 __udp4_lib_lookup_skb() 함수를 호출한다. __udp4_lib_lookup_skb() 함수는 수신한 패킷의 해당하는 소켓을 찾으면, 소켓의 수신 큐에 저장하는 udp_queue_rcv_skb() 함수를 호출한다. __udp4_lib_mcast_deliver() 함수에서도 소켓 검사를 통해 수신한 패킷의 해당하는 소켓을 찾으면, 소켓의 수신 큐에 저장하는 udp_queue_rcv_skb() 함수를 호출한다.
- [0039] __udp4_lib_rcv() 함수와 __udp4_lib_mcast_deliver() 함수는 수신한 패킷의 해당하는 소켓을 찾지 못하면, 그 패킷은 제거된다. 이러한 패킷은 스마트 단말기에 불필요한 패킷으로 분류되기 때문에, 본 발명에서 제안하는 필터 테이블 알고리즘 함수를 패킷이 제거되는 소스 부분에 추가해줄 필요가 있다.
- [0040] 도 6은 rt3070 칩셋의 드라이버 동작을 나타낸 흐름도이다.
- [0041] 도 6을 참조하면, 도 2에서 NIC가 패킷을 수신하면 인터럽트와 핸들러 함수를 실행시키는 것과 같이, 패킷을 수신한 rt3070 칩셋 기반의 무선 LAN NIC도 인터럽트를 발생시키고, 핸들러 함수인 STARxDoneInterruptHandle() 함수를 수행한다. STARxDoneInterruptHandle() 함수는 NIC가 수신한 패킷을 호스트 메모리 버퍼로 저장한다. 상위 계층으로 전달할 패킷이면 STAHandleRxDataFrame() 함수를 호출한다. STAHandleRxDataFrame() 함수는 802.11 패킷을 802.3 패킷으로 변환해주는 indicate_Legacy_Packet() 함수를 호출한다. 이 작업이 완료되면 패킷을 소켓 버퍼에 저장해주는 announce_802_3_packet() 함수가 호출된다. 소켓 버퍼에 저장된 패킷은 netif_rx() 함수로 전달된다. netif_rx() 함수 이후의 동작은 데이터 링크계층의 패킷 전송 경로에 대한 설명과 동일하다.

- [0042] 도 7은 본 발명의 일실시예에 의한 불필요한 패킷의 자동 제거 알고리즘의 개념도이다.
- [0043] 도 7을 참조하면, 먼저 불필요한 패킷을 구분하기 위한 필터 테이블을 생성한다. 이후, 네트워크 드라이버에서는 생성된 필터 테이블을 참조하고, 수신한 패킷이 필터 테이블에 존재하는지를 검사한다. 수신한 패킷이 필터 테이블에 존재한다면 패킷을 제거한다. 수신한 패킷이 필터 테이블에 존재하지 않는다면 기존의 커널이 패킷을 수신했을 때와 같은 동작을 하도록 한다.
- [0044] 필터 테이블 알고리즘은 불필요한 패킷이 실제 리눅스 커널 소스에서 제거되는 위치에 구현된다. 필터링 알고리즘은 네트워크 드라이버에서 패킷이 소켓 버퍼에 저장하기 전에 구현된다.
- [0045] 이와 같이, 불필요한 패킷의 자동 제거 알고리즘은 불필요한 패킷의 정보를 저장하고, 네트워크 드라이버 단에서 불필요한 패킷의 리스트를 참조할 수 있도록 한다.
- [0046] 도 8은 본 발명의 일실시예에 의한 필터 테이블의 구조와 저장 방식을 나타낸 구조도이다.
- [0047] 도 8을 참조하면, filter_list 구조체는 세 가지의 멤버로 구성되어 있다. list 멤버는 filter_list 구조체 간 링크를 연결해 주는 역할을 수행한다. dport 멤버와 protocol 멤버는 불필요한 패킷의 목적지 포트 번호와 프로토콜 번호가 저장된다. 본 발명에서는 불필요한 패킷의 필터 테이블을 만들기 위해 unpacket_filter_table() 함수를 구현한다.
- [0048] 도 9는 본 발명의 일실시예에 의한 수신된 데이터에서의 불필요한 패킷 제거 장치의 제어회로블록도이다.
- [0049] 도 9를 참조하면, 수신된 데이터에서의 불필요한 패킷 제거 장치는, 수신된 패킷의 IP 헤더와 TCP 헤더를 추출하여 불필요한 패킷의 프로토콜 번호와 목적지 포트번호를 filter_list 구조체에 저장하고, filter_list 구조체를 필터 테이블 리스트에 추가하는 필터 테이블 작성부(1)와, 필터 테이블 리스트가 존재할 경우, 프로토콜 번호와 목적지 포트번호가 존재하는지 여부를 검사하는 필터 테이블 검사부(2)와, 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재할 경우, 불필요한 패킷을 네트워크 드라이버 단에서 제거하는 패킷 제거부(3)를 포함한다.
- [0050] 도 10은 본 발명의 일실시예에 의한 필터 테이블을 구성하는 과정을 나타낸 흐름도이다.
- [0051] 도 10을 참조하면, unpacket_filter_table() 함수는 커널 소스의 불필요한 패킷이 제거되는 위치에서 호출된다. unpacket_filter_table() 함수는 불필요한 패킷이 수신되면 호출되고, 매개변수는 소켓 버퍼 구조체를 받는다(S1). 이 함수에서 맨 처음 하는 일은 수신한 패킷의 프로토콜 번호와 목적지 포트 번호를 참조하기 위해 소켓 버퍼로부터 IP 헤더와 TCP 헤더 값을 가져온다(S2 ~ S3). 다음으로 하는 동작은 필터 테이블에 대한 헤드 구조체를 검사한다(S4). 헤드 구조체는 필터 테이블에 리스트가 하나도 없는 경우 0 값을 가지고 있기 때문에, 첫 번째 리스트가 추가될 때 헤드 구조체를 생성한다(S5 ~ S6). 첫 번째 리스트가 추가된 이후부터는 헤드 구조체를 만드는 작업은 수행하지 않는다. 헤드 구조체의 조건 검사가 끝나면 수신한 불필요한 패킷의 프로토콜 번호와 목적지 포트번호는 filter_list 구조체에 저장한다(S7). 마지막으로 list_add_tail() 함수가 호출된다. 이 함수는 filter_list 구조체를 필터 테이블 리스트에 추가해 주는 작업을 수행한다(S8).
- [0052] 커널 영역의 필터 테이블은 네트워크 드라이버 단에서 바로 참조할 수 없기 때문에 EXPORT_SYMBOL() 매크로를 사용해야 한다. EXPORT_SYMBOL() 매크로는 네트워크 드라이버 단에서도 필터 테이블의 헤더 주소를 참조할 수 있도록 해준다.
- [0053] 한편, 네트워크 드라이버 단에서는 필터 테이블을 참조하여 수신한 패킷이 불필요한 패킷인지, 필요한 패킷인지를 판단한다. 불필요한 패킷인 경우에는 네트워크 드라이버 단에서 바로 제거한다.
- [0054] 본 발명에서는 네트워크 드라이버에서 불필요한 패킷을 필터링하는 unpacket_filter() 함수를 구현하기로 한다.
- [0055] 도 11은 본 발명의 일실시예에 의한 불필요한 패킷을 제거하는 과정을 나타낸 흐름도이다.

[0056] 도 11을 참조하면, 네트워크 드라이버에서 unpack_filter() 함수는 패킷을 소켓 버퍼에 저장하기 전에 호출된다. rt3070 칩셋을 사용하는 네트워크 드라이버는 수신한 패킷을 _RX_BLK 구조체에 저장한다. unpack_filter() 함수는 매개변수로 _RX_BLK 구조체를 받는다(S11). unpack_filter() 함수는 수신한 패킷의 프로토콜 번호와 목적지 포트번호를 가져오기 위해 _RX_BLK 구조체로부터 패킷의 IP 헤더와 TCP 헤더를 가져온다(S12). 다음으로 하는 동작은 필터 테이블의 헤드 구조체를 검사한다(S13). 만약, 헤드 구조체의 값이 0인 경우에는 필터 테이블에 리스트가 존재하지 않는 경우이므로 unpack_filter() 함수를 바로 종료한다. 헤드 구조체의 값이 존재하는 경우에는 필터 테이블의 리스트가 존재하기 때문에, 필터 테이블의 첫 번째 리스트부터 하나씩 검사한다(S14 ~ S15). 수신한 패킷의 프로토콜 번호와 목적지 포트 번호가 필터 테이블의 리스트에 존재하는지를 검사한다(S16). 필터 테이블 리스트에 해당 프로토콜 번호와 목적지 포트 번호가 존재한다면, 그 패킷은 불필요한 패킷으로 판단한다. 불필요한 패킷은 rt3070 칩셋의 네트워크 드라이버에서 사용하는 REALEASE_NDIS_PACKET() 매크로를 이용하여 패킷을 제거한다(S17 ~ S18). 테이블 리스트에서 수신한 패킷과 동일한 포트와 프로토콜 번호를 필터 찾지 못한다면, unpack_filter() 함수는 종료된다. 한편, 모든 필터 테이블의 모든 리스트의 검사가 완료될 때까지 상기한 단계 S16 내지 단계 S19를 반복한다(S19 ~ S20).

[0057] 여기서, 본 발명에서 구현한 알고리즘을 테스트한 결과에 대해 설명한다.

[0058] 테스트 환경은 다음과 같다.

[0059] 리눅스 커널 2.6.29 버전에서 필터 테이블 알고리즘을 적용하기 위해 커널 소스 코드를 수정한다. rt3070 칩셋을 사용하는 무선 LAN 네트워크 디바이스에 필터링 알고리즘을 적용하기 위해 rt3070 드라이버 소스 코드를 수정한다. 임베디드 보드는 EZ-S3C6410을 사용하고, 해당 보드에 수정한 리눅스 커널과 네트워크 드라이버를 포팅한다. 임베디드 보드는 AP와 무선으로 연결되도록 환경을 설정한다. 서로 다른 2개의 AP에 연결하여 결과를 비교해 본다. 측정 시간은 24시간으로 한다. 최종 결과 값은 시간별 수신한 불필요한 패킷 수와 패킷의 프로토콜 타입별 수신한 패킷의 수이다.

[0060] 도 12는 제1AP에 연결하여 24시간 동안 수신한 불필요한 패킷의 리스트이다.

[0061] 도 12를 참조하면, 커널 영역에서 불필요한 패킷은 프로토콜 타입과 포트 번호가 필터 테이블에 등록되는 것이다. 불필요한 패킷의 리스트가 등록된 필터 테이블은 무선 LAN 드라이버 단에서 참조되고, 수신된 패킷의 수신 여부를 검사한다.

[0062] [테이블 1]은 제1AP에 연결된 보드가 패킷 타입에 따라 필터링된 패킷의 수를 나타낸다.

[0063] [테이블 1]

Layer		Packet Type	The number of filtered packets
Transport Layer		TCP	No Scket
		UDP	No Scket
Network Layer	Broadcast	NBNS	
		SSDP	
	Multicast	MDNS	
		LLMNR	
		Plsrv-HTTPS	
		WD-Discovery	
		SRVLOC	
		TEREDO	
	Routing	HSRP	
		OSPF	

[0064]

[0065] [테이블 1]을 참조하면, 24시간 동안 많은 패킷들이 필터링된 것을 확인할 수 있다. 네트워크 계층에서는 멀티

캐스트 패킷과 라우팅 프로토콜 패킷의 수가 많이 필터링된 것을 확인할 수 있다. 전송 계층에서의 'No Socket'은 수신한 패킷에 해당하는 소켓을 찾지 못한 경우이다. 즉, 수신한 패킷을 필요로 하는 상위 어플리케이션이 존재하지 않는다는 것이다. 전송 계층에서는 브로드캐스트 패킷이 대부분 필터링된 것을 확인할 수 있다.

[0066] 도 13은 제1AP로부터 시간별로 수신한 불필요한 패킷의 수를 나타낸 그래프이다.

[0067] 도 13을 참조하면, 시간당 수신하는 불필요한 패킷의 수가 많이 존재한다는 것을 확인할 수 있다. 불필요한 패킷이 필터링되는 수가 시간당 최소 6000개 이상이고, 주로 트래픽이 많은 오후 시간에 필터링되는 패킷의 수가 많다는 것을 확인할 수 있다.

[0068] 도 14는 제2AP에 연결하여 24시간 동안 수신한 불필요한 패킷의 리스트이다.

[0069] 도 14를 참조하면, 도 12의 필터 테이블과 도 14의 필터 테이블을 비교하면, 제1AP보다 제2AP에 연결된 보드의 리스트 수가 적다는 것을 확인할 수 있다. 그 이유는 AP마다 환경설정이 다르기 때문이며, 이에 시험 결과는 다르게 나타날 수 있다.

[0070] [테이블 2]는 제2AP에 연결된 보드가 패킷 타입에 따라 필터링된 패킷의 수를 나타낸다.

[0071] [테이블 2]

Layer	Packet Type	The number of filtered packets
Transport Layer	Broadcast	NBNS
		7,138
Network Layer	Multicast	SSDP
		54,477
		LLMNR
		2,061
		Plysrv-HTTPS
		213
		MDNS
		125

[0072]

[0073] [테이블 2]를 참조하면, 24시간 동안 많은 패킷들이 필터링된 것을 확인할 수 있다. 제2AP에 연결된 보드는 주로 멀티캐스트 패킷과 브로드캐스트 패킷이 많이 필터링된 것을 확인할 수 있다.

[0074] 도 15는 제2AP로부터 시간별로 수신한 불필요한 패킷의 수를 나타낸 그래프이다.

[0075] 도 15를 참조하면, 이 그래프도 역시 트래픽이 많은 오후 시간에 좀 더 많은 패킷이 필터링되는 것을 확인할 수 있다.

[0076] 이와 같이, 보드를 2개의 AP에 각각 연결하여 분석한 결과, 무선 단말기는 많은 수의 불필요한 패킷을 수신하고 있으며, 수신된 불필요한 패킷은 필터 테이블에 해당 패킷의 프로토콜 번호와 포트 번호가 저장되는 것을 확인할 수 있다. 또한, 필터 테이블에 등록된 패킷은 무선 LAN 드라이버 단에서 필터링되는 것을 확인할 수 있다.

[0077] 이상에서 몇 가지 실시예를 들어 본 발명을 더욱 상세하게 설명하였으나, 본 발명은 반드시 이러한 실시예로 국한되는 것이 아니고 본 발명의 기술사상을 벗어나지 않는 범위 내에서 다양하게 변형실시될 수 있다.

부호의 설명

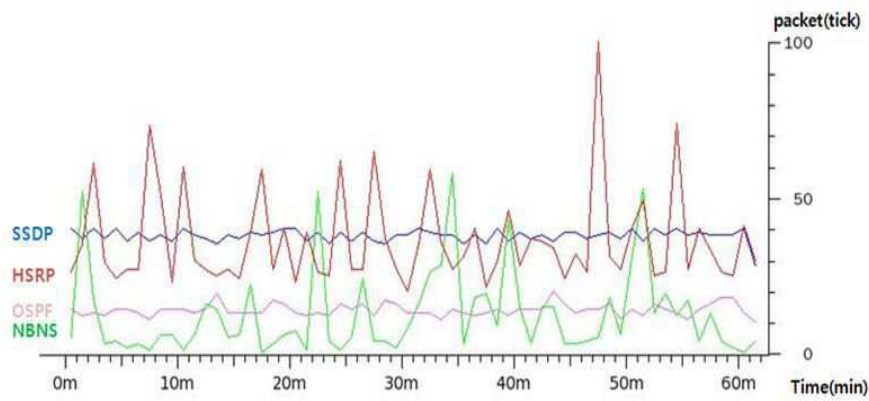
[0078] 1 : 필터 테이블 작성부

2 : 필터 테이블 검사부

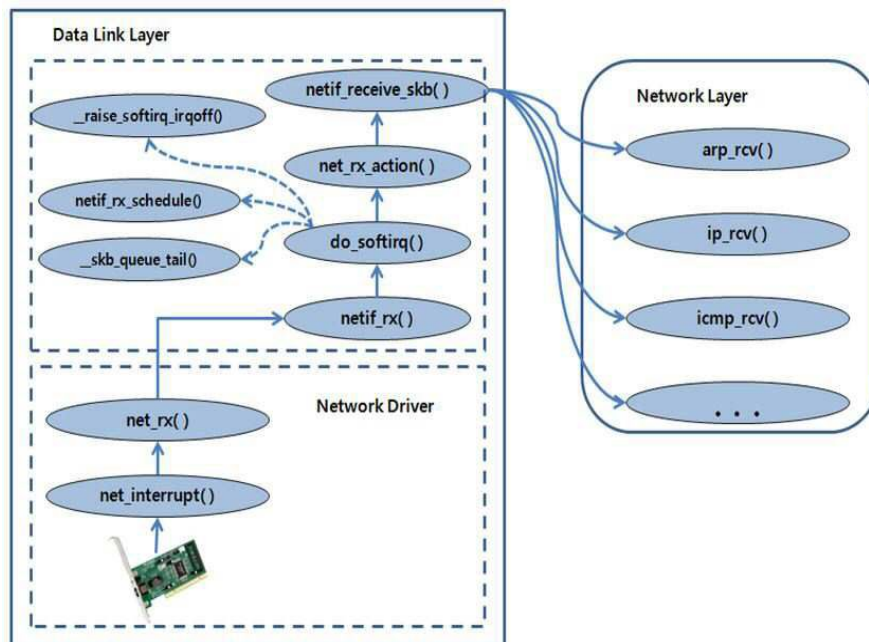
3 : 패킷 제거부

도면

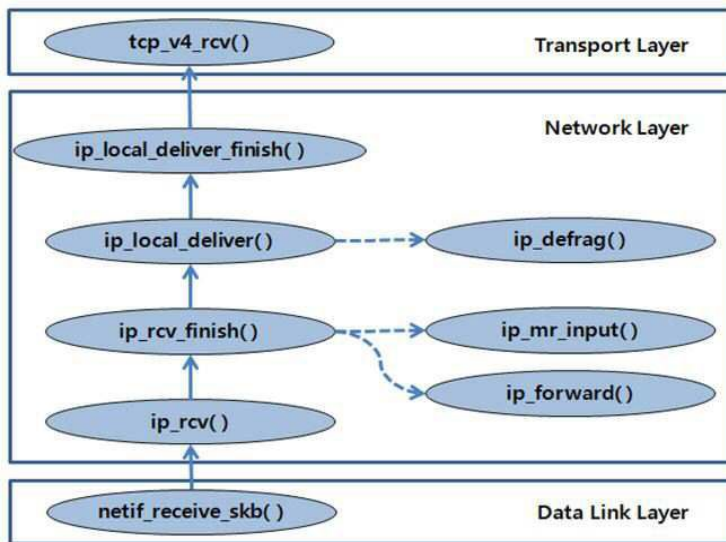
도면1



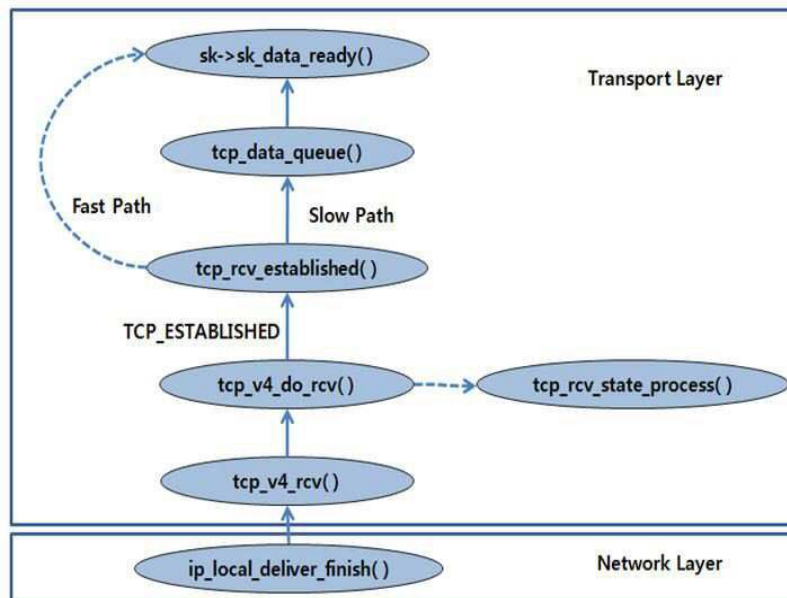
도면2



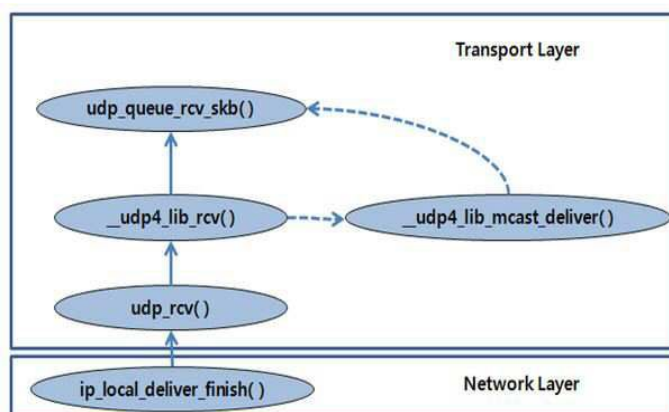
도면3



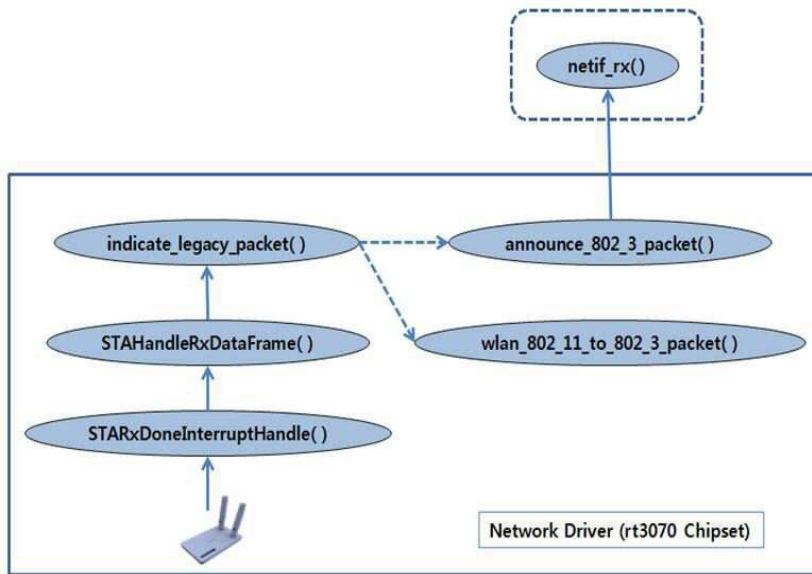
도면4



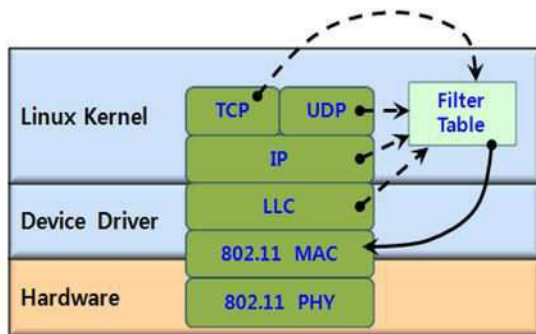
도면5



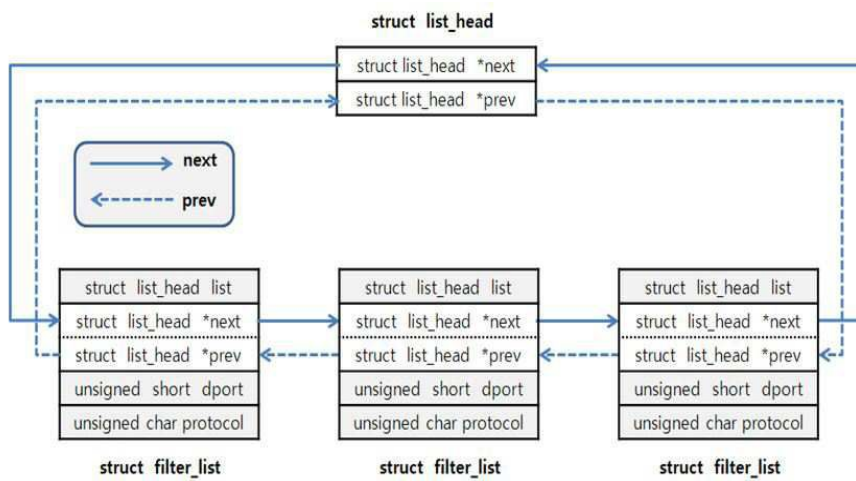
도면6



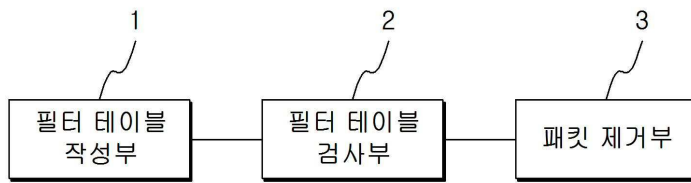
도면7



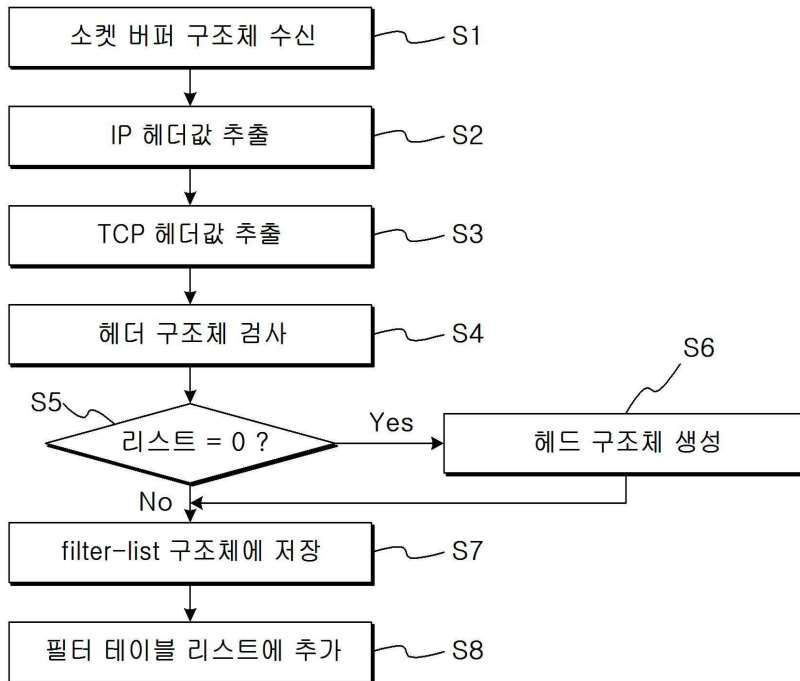
도면8



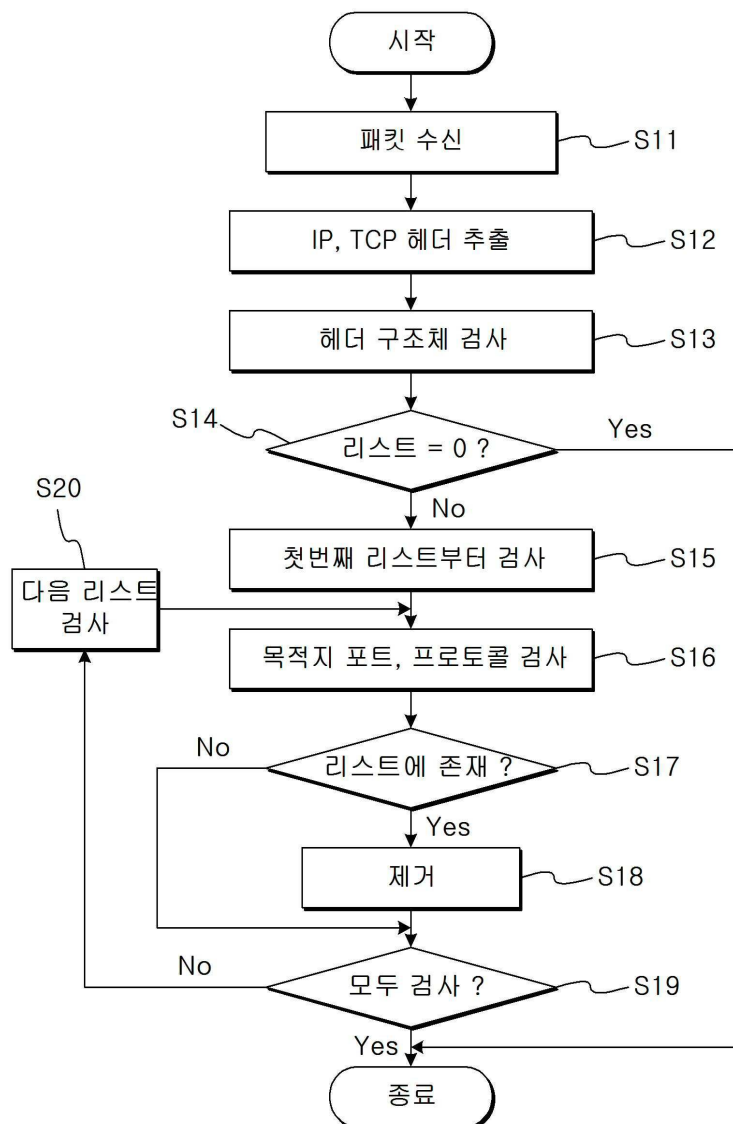
도면9



도면10



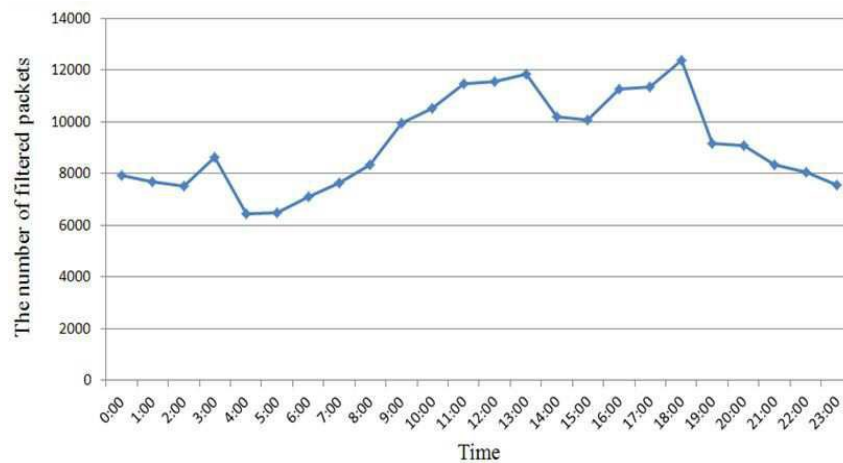
도면11



도면12

[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [137]
[FILTER TABLE]	Added Packet : Protocol = [OSPF]	Port = [-]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [1985]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [1900]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [17500]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [5353]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [1211]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [5355]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [138]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [427]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [6771]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [3702]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [3544]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55545]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [6666]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55604]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55608]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55636]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55637]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55667]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55669]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55696]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55703]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55765]
[FILTER TABLE]	Added Packet : Protocol = [TCP]	Port = [55770]

도면13



도면14

[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [1900]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [138]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [137]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [5353]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [5355]
[FILTER TABLE]	Added Packet : Protocol = [UDP]	Port = [6771]

도면15

