

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)(11) 공개번호 10-2022-0056689
(43) 공개일자 2022년05월06일

(51) 국제특허분류(Int. Cl.)

H04L 47/00 (2022.01) H04L 43/00 (2022.01)
H04L 45/24 (2022.01) H04L 45/28 (2022.01)
H04L 49/00 (2022.01) H04L 49/111 (2022.01)
H04L 9/40 (2022.01)

(52) CPC특허분류

H04L 47/2408 (2022.05)
H04L 43/0823 (2022.05)

(21) 출원번호 10-2020-0141500

(22) 출원일자 2020년10월28일
심사청구일자 2020년10월28일

(71) 출원인

고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자

이원준

서울특별시 강남구 선릉로 221, 104동 2005호(도곡동, 도곡테크아파트)

김규영

서울특별시 성북구 안암로 145, 고려대학교 우정정보통신관 507A (안암동5가)

(74) 대리인

윤귀상

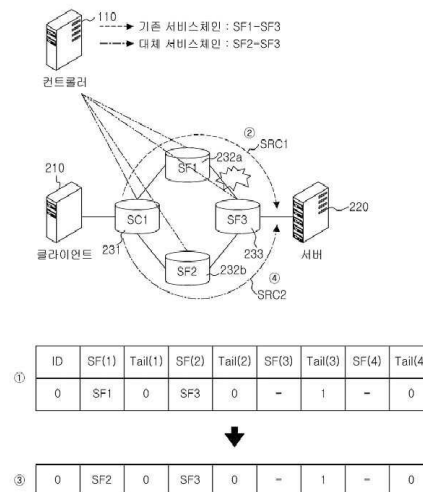
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 고가용성을 지원하는 프로그래머블 스위치 상에서의 서비스 기능 체이닝 시스템

(57) 요약

본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템은 클라이언트와 서버 사이에서 복수의 서비스 체인을 이루며, 컨트롤러에 의해 감시 및 제어되는 프로그래머블 스위치를 포함하는 서비스 기능 체이닝 시스템에 있어서, 상기 컨트롤러는, 일 서비스 체인을 통한 서비스 기능 체이닝 패킷 처리 과정에서 회선 장애 감지 시, 회선 장애가 없는 대체 서비스 체인으로 경로를 갱신하여 해당 대체 서비스 체인으로 서비스 기능 체이닝 패킷이 전달되도록 하는 것을 특징으로 하는 서비스 기능 체이닝 시스템에 관한 것이다.

대표도 - 도7



(52) CPC특허분류

H04L 45/22 (2022.05)
H04L 45/28 (2022.05)
H04L 47/2441 (2022.05)
H04L 49/3009 (2013.01)
H04L 49/35 (2022.05)
H04L 69/22 (2022.05)
H04L 2212/00 (2014.06)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711104878
과제번호	2017M3C4A7083676
부처명	과학기술정보통신부
과제관리(전문)기관명	한국연구재단
연구사업명	차세대정보·컴퓨팅기술개발(R&D)
연구과제명	Web 응용을 위한 Seamless 자율 전송 네트워킹 제어 프레임워크
기 여 율	1/2
과제수행기관명	고려대학교
연구기간	2020.01.01 ~ 2020.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호	1711102841
과제번호	2017-0-00195-004
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	방송통신산업기술개발(R&D)
연구과제명	멀티 서비스를 지원하는 프로그래머블 스위치 제어 기술 개발
기 여 율	1/2
과제수행기관명	고려대학교산학협력단
연구기간	2020.01.01 ~ 2020.12.31

명세서

청구범위

청구항 1

클라이언트와 서버 사이에서 복수의 서비스 체인을 이루며, 컨트롤러에 의해 감시 및 제어되는 프로그래머블 스위치를 포함하는 서비스 기능 체이닝 시스템에 있어서,

상기 컨트롤러는,

일 서비스 체인을 통한 서비스 기능 체이닝 패킷 처리 과정에서 회선 장애 감지 시, 회선 장애가 없는 대체 서비스 체인으로 경로를 갱신하여 해당 대체 서비스 체인으로 서비스 기능 체이닝 패킷이 전달되며 처리되도록 하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 2

제 1 항에 있어서,

상기 서비스 체인은,

클라이언트와 연결되며 캡슐화(Encapsulation)를 담당하는 서비스 기능 분류자 모듈;

상기 서비스 기능 분류자 모듈과 연결되며 서비스 기능(SF)을 담당하는 제1 서비스 기능 모듈 및

서비스 기능(SF)과 역캡슐화(Decapsulation)를 담당하는 제2 서비스 기능 모듈을 포함하는 서비스 기능 체이닝 시스템.

청구항 3

제 2 항에 있어서,

상기 컨트롤러는,

최초 구동 시 상기 서비스 기능 분류자 모듈과 각 서비스 기능 모듈에 초기 서비스 체인 정보를 기록하며, 각 서비스 기능 모듈의 포트 감지 시 회선 장애가 발견되면 대체 서비스 체인 정보를 상기 서비스 기능 분류자 모듈에 기록하여 패킷 전달 경로를 갱신하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 4

제 3 항에 있어서,

상기 컨트롤러는,

상기 각 서비스 기능 모듈의 포트 감지 시, 상기 프로그래머블 스위치가 제공하는 Barefoot Runtime API를 참조하여 프로그래머블 스위치 내 포트 색인번호에 해당하는 \$DEV_PORT 값을 읽고, \$DEV_PORT 값이 Down에 해당하는 값인지 아닌지에 따라 회선 장애를 판단하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 5

제 2 항에 있어서,

상기 서비스 기능 체이닝 패킷은,

이더넷(ETH) 헤더와 IP 헤더 사이에 서비스 기능 체이닝을 위한 서비스 체인 헤더(SCH) 및 서비스 기능 헤더

(SFH)를 포함하며,

상기 서비스 체인 헤더는,

서비스 체인 식별자인 ID 필드와, 현재까지 거친 서비스 기능 모듈의 수를 역순으로 셈하는 LEN 필드를 포함하고,

상기 서비스 기능 헤더는,

서비스 기능 체이닝 패킷이 방문해야 할 서비스 기능 모듈의 식별자를 기록하는 SF 필드와, 상기 SF 필드의 위치를 표시하는 TAIL 필드를 포함하며,

상기 프로그래머블 스위치의 인그레스 파서(Ingress parser)는, 상기 이더넷(ETH) 헤더부터 IP 헤더까지 파싱하는 과정에서 상기 TAIL 필드 값이 마지막 위치를 표시하는 경우까지 파싱을 반복하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 6

제 5 항에 있어서,

상기 서비스 기능 체이닝 패킷 처리는,

상기 프로그래머블 스위치가 IP 헤더 내 Type of Service(ToS) 필드 값을 확인하여, 서비스 기능 체이닝 패킷임이 확인 될 경우 처리를 수행하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 7

제 6 항에 있어서,

상기 서비스 기능 체이닝 패킷 처리하는 과정에서,

상기 서비스 기능 체이닝 패킷임이 확인 될 경우, 서비스 체인 헤더의 존재 유무를 확인하며, 서비스 체인 헤더가 없을 경우에는 캡슐화를 진행하여 서비스 체인 헤더와 서비스 기능 헤더를 패킷에 생성한 후에 처리를 수행하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 8

제 7 항에 있어서,

상기 서비스 체인 헤더의 존재 유무는 isvalid() 함수를 통해 유효성을 검증하여 확인하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 9

제 6 항에 있어서,

상기 서비스 기능 체이닝 패킷 처리는,

상기 서비스 기능 모듈을 거쳐 서비스 체이닝 패킷 처리를 받을 때마다 상기 LEN 필드 값을 1씩 감소시키며 상기 서비스 기능 헤더의 가장 앞에 있는 SF 필드-TAIL 필드 쌍을 제거하는 과정을 LEN 필드 값이 0이 될 때까지 반복 수행하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

청구항 10

제 9 항에 있어서,

상기 컨트롤러는,

최초 구동 시 상기 서비스 기능 체이닝 패킷 내 저장된 서비스 체인 정보를 읽고 해당하는 출력 포트에 패킷을 포워딩하는 서비스 기능 체이닝 포워딩 룰을 저장하며,

상기 서비스 기능 체이닝 패킷 처리는,

상기 LEN 필드 값이 0이 아닐 경우, SF Forward로 넘어가 SFF 기능을 수행하며, 상기 프로그래머블 스위치가 저장하고 있는 서비스 기능 체이닝 포워딩 룰을 참조하여 현재 패킷의 출력 포트를 결정하는 것을 특징으로 하는 서비스 기능 체이닝 시스템.

발명의 설명

기술 분야

[0001] 본 발명은 서비스 기능 체이닝 시스템에 관한 것으로, 보다 상세하게는 프로그래머블 스위치 ASIC를 장착한 네트워크 스위치 장치에서 서비스 기능 체이닝을 통해 패킷을 처리할 시 네트워크 장애 발생에도 끊김 없이 패킷 송수신이 가능케 하는 고가용성 서비스 기능 체이닝 시스템에 관한 것이다.

배경 기술

[0003] 서비스 기능 체이닝(Service Function Chaining)은 패킷이 서비스 체인에 명시된 서비스 기능(Service Function, SF)들을 순차적으로 방문하여 처리를 받는 복잡한 인-네트워크 패킷 프로세싱(In-network packet processing)을 가능케 하여 많은 관심을 받고 있다.

[0004] 서비스 기능 체이닝 패킷은 서비스 기능 분류자(Service Classifier, SC)와 서비스 기능 전달자(Service Function Forwarder, SFF)와 같은 체이닝 기능(Chaining Function, CF)들에 의해 제어된다. 여기서, 서비스 기능 분류자(SC)는 네트워크 양 끝 단에서 패킷을 캡슐화 및 역캡슐화하여 패킷이 서비스 기능 체이닝을 시작하고 종료할 수 있게 해주며, 서비스 기능 전달자(SFF)는 도착한 패킷이 다음에 방문해야 하는 서비스 기능(SF)을 결정하고 전달하는 역할을 담당한다. 또한, 서비스 기능(SF)은 패킷에 대해 방화벽, NAT 등 다양한 기능을 수행한다.

[0005] 인-네트워크 패킷 프로세싱은 네트워크 집약적이므로 저지연과 고처리량이 제공되어야 한다. 그러나 체이닝 기능(CF)과 서비스 기능(SF)은 대개 범용 서버 상에서 동작하기 때문에 이러한 요구사항을 만족하기가 어렵다. 또한, 체이닝 기능(CF)과 서비스 기능(SF)들은 물리적으로 분리되어 있으므로 패킷이 처리되기 위해서는 수백마이크로초에 달하는 지연시간을 겪어야 한다. 오늘날의 스위치 하드웨어는 패킷 당 1마이크로초 이하의 프로세싱 지연만을 요구하므로 체이닝 기능(CF)과 서비스 기능(SF)을 경유하며 발생하는 지연은 서비스 기능 체이닝(SFC)의 전체 지연을 크게 증가시키는 요인으로 작용된다. 또한, 상용 서버는 스위치 하드웨어에 비해 낮은 초당 패킷 처리량을 제공한다.

[0006] 한편, 최근 등장한 Barefoot Tofino와 같은 프로그래머블 스위치 ASIC는 사용자가 패킷 프로세싱 파이프라인을 재구성하는 것을 허용한다. 이에 서비스 기능 체이닝의 구성요소인 체이닝 기능(CF)과 서비스 기능(SF)을 프로그래머블 스위치 상에 구현하여 저지연과 고처리량을 달성하는 접근이 제안된 바 있다.

발명의 내용

해결하려는 과제

[0008] 본 발명의 목적은 고가용성을 지원하는 프로그래머블 스위치 상에서의 네트워크 장애 발생 시에도 끊김 없이 서비스 기능 체이닝이 정상적으로 작동할 수 있도록 하는 것으로, 네트워크 상에 별개의 노드로 존재하였던 체이닝 기능(CF)과 서비스 기능(SF), 그리고 스위치가 하나의 노드로 합쳐지면서 회선 장애와 같은 네트워크 장애 발생 시 서비스 기능 체이닝 역시 정상적으로 처리되지 못하는 문제에 있어서, 서비스 체인 정보를 각 스위치가 아닌 패킷 헤더에 저장하고, 컨트롤러가 스위치 회선 상태를 감시하여 장애 발생시 서비스 기능 분류자(SC)의

서비스 체인 정보를 업데이트하여 대체 서비스 체인으로 서비스 기능 체이닝 패킷이 전송되도록 하여 상기의 문제점을 해결한 서비스 기능 체이닝 시스템을 제공하는데 목적이 있다.

과제의 해결 수단

- [0010] 상기 과제를 해결하기 위한 본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템은, 클라이언트와 서버 사이에서 복수의 서비스 체인을 이루며, 컨트롤러에 의해 감시 및 제어되는 프로그래머블 스위치를 포함하는 서비스 기능 체이닝 시스템에 있어서, 상기 컨트롤러는, 일 서비스 체인을 통한 서비스 기능 체이닝 패킷 처리 과정에서 회선 장애 감지 시, 회선 장애가 없는 대체 서비스 체인으로 경로를 갱신하여 해당 대체 서비스 체인으로 서비스 기능 체이닝 패킷이 전달되며 처리되도록 할 수 있다.
- [0011] 여기서, 상기 서비스 체인은, 클라이언트와 연결되며 캡슐화(Encapsulation)를 담당하는 서비스 기능 분류자 모듈; 상기 서비스 기능 분류자 모듈과 연결되며 서비스 기능(SF)을 담당하는 제1 서비스 기능 모듈 및 서비스 기능(SF)과 역캡슐화(Decapsulation)를 담당하는 제2 서비스 기능 모듈을 포함할 수 있다.
- [0012] 또한, 상기 컨트롤러는, 최초 구동 시 상기 서비스 기능 분류자 모듈과 각 서비스 기능 모듈에 초기 서비스 체인 정보를 기록하며, 각 서비스 기능 모듈의 포트 감지 시 회선 장애가 발견되면 대체 서비스 체인 정보를 상기 서비스 기능 분류자 모듈에 기록하여 패킷 전달 경로를 갱신할 수 있다.
- [0013] 또한, 상기 컨트롤러는, 상기 각 서비스 기능 모듈의 포트 감지 시, 상기 프로그래머블 스위치가 제공하는 Barefoot Runtime API를 참조하여 프로그래머블 스위치 내 포트 색인번호에 해당하는 \$DEV_PORT 값을 읽고, \$DEV_PORT 값이 Down에 해당하는 값인지 아닌지에 따라 회선 장애를 판단할 수 있다.
- [0014] 또한, 상기 서비스 기능 체이닝 패킷은, 이더넷(ETH) 헤더와 IP 헤더 사이에 서비스 기능 체이닝을 위한 서비스 체인 헤더(SCH) 및 서비스 기능 헤더(SFH)를 포함하며, 상기 서비스 체인 헤더는, 서비스 체인 식별자인 ID 필드와, 현재까지 거친 서비스 기능 모듈의 수를 역순으로 셈하는 LEN 필드를 포함하고, 상기 서비스 기능 헤더는, 서비스 기능 체이닝 패킷이 방문해야 할 서비스 기능 모듈의 식별자를 기록하는 SF 필드와, 상기 SF 필드의 위치를 표식하는 TAIL 필드를 포함하며, 상기 프로그래머블 스위치의 인그레스 파서(Ingress parser)는, 상기 이더넷(ETH) 헤더부터 IP 헤더까지 파싱 하는 과정에서 상기 TAIL 필드 값이 마지막 위치를 표시하는 경우까지 파싱을 반복할 수 있다.
- [0015] 또한, 상기 서비스 기능 체이닝 패킷 처리는, 상기 프로그래머블 스위치가 IP 헤더 내 Type of Service(ToS) 필드 값을 확인하여, 서비스 기능 체이닝 패킷임이 확인 될 경우 처리를 수행할 수 있다.
- [0016] 또한, 상기 서비스 기능 체이닝 패킷 처리하는 과정에서, 상기 서비스 기능 체이닝 패킷임이 확인 될 경우, 서비스 체인 헤더의 존재 유무를 확인하며, 서비스 체인 헤더가 없을 경우에는 캡슐화를 진행하여 서비스 체인 헤더와 서비스 기능 헤더를 패킷에 생성한 후에 처리를 수행할 수 있다.
- [0017] 또한, 상기 서비스 체인 헤더의 존재 유무는 isvalid() 함수를 통해 유효성을 검증하여 확인할 수 있다.
- [0018] 또한, 상기 서비스 기능 체이닝 패킷 처리는, 상기 서비스 기능 모듈을 거쳐 서비스 체이닝 패킷 처리를 받을 때마다 상기 LEN 필드 값을 1씩 감소시키며 상기 서비스 기능 헤더의 가장 앞에 있는 SF 필드-TAIL 필드 쌍을 제거하는 과정을 LEN 필드 값이 0이 될 때까지 반복 수행할 수 있다.
- [0019] 또한, 상기 컨트롤러는, 최초 구동 시 상기 서비스 기능 체이닝 패킷 내 저장된 서비스 체인 정보를 읽고 해당하는 출력 포트에 패킷을 포워딩하는 서비스 기능 체이닝 포워딩 룰을 저장하며, 상기 서비스 기능 체이닝 패킷 처리는, 상기 LEN 필드 값이 0이 아닐 경우, SF Forward로 넘어가 SFF 기능을 수행하며, 상기 프로그래머블 스위치가 저장하고 있는 서비스 기능 체이닝 포워딩 룰을 참조하여 현재 패킷의 출력 포트를 결정할 수 있다.

발명의 효과

- [0021] 본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템은, 네트워크 상에 별개의 노드로 존재하였던 체이닝 기능(CF)과 서비스 기능(SF), 그리고 스위치가 하나의 노드로 합쳐지면서 회선 장애와 같은 네트워크 장애 발생 시 서비스 기능 체이닝 역시 정상적으로 처리되지 못하는 문제에 있어서, 서비스 체인 정보를 각 스위치가 아닌 패킷 헤더에 저장하고, 컨트롤러가 스위치 회선 상태를 감시하여 장애 발생 시 서비스 기능 분류자(SC)의 서비스

체인 정보를 업데이트하여 대체 서비스 체인으로 서비스 기능 체이닝 패킷이 전송되도록 하여 고가용성을 지원하는 프로그래머블 스위치 상에서의 네트워크 장애 발생 시에도 끊김 없이 서비스 기능 체이닝이 정상적으로 작동할 수 있도록 할 수 있다.

[0022] 또한, 위에서 언급된 본 발명의 실시예에 따른 효과는 기재된 내용에만 한정되지 않고, 명세서 및 도면으로부터 예측 가능한 모든 효과를 더 포함할 수 있다.

도면의 간단한 설명

[0024] 도 1은 본 발명의 실시 예에 따른 프로그래머블 스위치를 이용한 서비스 기능 체이닝을 사용하는 네트워크의 구성도이다.

도 2는 도 1에 도시된 네트워크 중 제어평면에 위치한 컨트롤러의 작동 과정을 나타낸 흐름도이다.

도 3은 본 발명의 실시 예에 따른 서비스 기능 체이닝 패킷 헤더의 구조를 도식화한 것이다.

도 4는 본 발명의 실시 예에 따른 프로그래머블 스위치에서 패킷 헤더를 파싱하는 과정을 도식화한 것이다.

도 5는 본 발명의 실시 예에 따른 프로그래머블 스위치에서 패킷을 처리하는 인그레스 파이프라인(Ingress pipeline)을 도식화한 것이다.

도 6은 본 발명의 실시 예에 따른 서비스 기능 체이닝 시 패킷 헤더의 변화를 나타낸 도면이다.

도 7은 본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템을 이용한 서비스 기능 체이닝 시의 네트워크 장애에 컨트롤러가 대응하는 과정을 보여주는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하, 도면을 참조한 본 발명의 설명은 특정한 실시 형태에 대해 한정되지 않으며, 다양한 변환을 가할 수 있고 여러 가지 실시예를 가질 수 있다. 또한, 이하에서 설명하는 내용은 본 발명의 사상 및 기술 범위에 포함되는 모든 변환, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0026] 이하의 설명에서 제1, 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용되는 용어로서, 그 자체에 의미가 한정되지 아니하며, 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다.

[0027] 본 명세서 전체에 걸쳐 사용되는 동일한 참조번호는 동일한 구성요소를 나타낸다.

[0028] 본 발명에서 사용되는 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 또한, 이하에서 기재되는 "포함하다", "구비하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것으로 해석되어야 하며, 하나 또는 그 이상의 다른 특징들이나, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0029] 또한, 명세서에 기재된 "편부", "편기", "모듈" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어나 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다.

[0030] 이하, 첨부된 도면을 참조하여 본 발명의 실시 예에 따른 고가용성을 지원하는 프로그래머블 스위치 상에서의 서비스 기능 체이닝 시스템을 상세히 설명하기로 한다.

[0031] 도 1은 본 발명의 실시 예에 따른 프로그래머블 스위치를 이용한 서비스 기능 체이닝을 사용하는 네트워크의 구성도이다. 도 1을 참조하면, 본 발명의 서비스 기능 체이닝 시스템은 제어평면(100)과 데이터평면(200)으로 구성되는 네트워크(10)에 사용될 수 있다. 또한, 제어평면(100)은 컨트롤러(110)를 포함하여 구성될 수 있다. 이때, 컨트롤러(110)는 데이터평면(200)에 존재하는 프로그래머블 스위치(230)들을 감시하고 제어할 수 있다.

[0032] 데이터평면(200)은 클라이언트(210), 서버(220), 프로그래머블 스위치(230)를 포함하여 구성될 수 있다. 이때, 프로그래머블 스위치(230)는 서비스 기능 체이닝에 필요한 서비스 기능 분류자(Service Classifier, SC), 서비스 기능 전달자(Service Function Forwarder, SFF), 서비스 기능(Service Function, SF)을 모두 담당하도록 형성될 수 있다.

[0033] 이를 위해, 본 발명의 실시 예에 따른 네트워크(10)에서는 일례로 프로그래머블 스위치(230)가 총 4개가 존재할

수 있으며, 4개의 프로그래머블 스위치(230) 중 서비스 기능 분류자(SC) 역할을 하며 캡슐화를 담당하는 서비스 기능 분류자 모듈(231)이 클라이언트(210)에 연결될 수 있다.

[0034] 또한, 클라이언트(210)와 서버(220) 사이에는 복수의 제1 서비스 기능 모듈(232)과, 제2 서비스 기능 모듈(233)이 존재할 수 있으며, 서비스 기능 분류자 모듈(231)과 제1 및 제2 서비스 기능 모듈(233)들은 모두 서비스 기능 전달자(SFF)의 역할을 수행할 수 있으므로, 서비스 기능 전달자 모듈은 따로 구비되지 않는다.

[0035] 여기서, 중간에 위치한 복수의 제1 서비스 기능 모듈(232)은 서비스 기능(SF)만을 담당하며, 마지막에 위치한 제2 서비스 기능 모듈(233)은 서비스 기능(SF)과 함께 서비스 기능 분류자(SC)의 역할 중 역캡슐화의 기능을 수행할 수도 있다.

[0036] 이때, 복수의 제1 서비스 기능 모듈(232)은 각기 다른 노드를 갖는 것이 바람직하다. 이는 회선 장애에 대처하기 위함으로, 하나의 제1 서비스 기능 모듈(232)을 통하는 회선이 장애가 발생 할 시 다른 제1 서비스 기능 모듈(232)로 연결 노드를 변경하기 위함이다.

[0037] 즉, 클라이언트(210)와 서버(220) 사이의 프로그래머블 스위치(230)를 거쳐 서비스 기능 체이닝 패킷(이하 '패킷'이라 함)이 전달되며 처리되는 과정 속에서 본 발명의 실시 예에서는 서비스 기능 분류자 모듈(231)과, 제1 서비스 기능 모듈(232), 제2 서비스 기능 모듈(233)을 순차적으로 통과하게 되는데, 제1 서비스 기능 모듈(232)의 경우 복수로 마련되어 각기 다른 노드를 형성하므로, 회선 장애가 발생하더라도 경로의 변경이 가능할 수 있다.

[0038] 여기서, 서비스 기능 분류자 모듈(231), 제1 서비스 기능 모듈(232), 제2 서비스 기능 모듈(233)이 상술한 서비스 기능 분류자(Service Classifier, SC), 서비스 기능 전달자(Service Function Forwarder, SFF), 서비스 기능(Service Function, SF)을 모두 담당할 수 있는 프로그래머블 스위치(230)를 형성하므로, 이하에서는 이들의 합을 서비스 체인이라 지칭하기로 한다.

[0039] 즉, 본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템은, 서비스 기능 분류자 모듈(231), 제1 서비스 기능 모듈(232), 제2 서비스 기능 모듈(233)의 합으로 구성된 서비스 체인(SRC)이 클라이언트(210)와 서버(220) 사이에서 복수로 마련되는 것이다.

[0040] 한편, 상기와 같은 프로그래머블 스위치(230)는 본 발명의 이해를 돕기 위해 도면과 같이 4개로 한정하여 설명하였으나 이는 예시적인 것일 뿐, 한정되는 것은 아니며 프로그래머블 스위치(230)의 개수는 그 이상으로 달리 형성될 수 있다.

[0041] 예컨대, 제1 서비스 기능 모듈(232)과 제2 서비스 기능 모듈(233) 사이에 다른 서비스 기능 모듈이 마련될 수 있는 것이다.

[0042] 도 2는 도 1에 도시된 네트워크 중 제어평면에 위치한 컨트롤러의 작동 과정을 나타낸 흐름도이다.

[0043] 도 2를 참조하면, 컨트롤러(110)는 최초 구동 시 네트워크(10)에 존재하는 각 프로그래머블 스위치(230)인 서비스 기능 분류자 모듈(231)과 각 서비스 기능 모듈(232, 233)에 초기 서비스 체인 정보를 기록하여 서비스 기능 체이닝을 통해 패킷 송수신을 가능케 할 수 있다.

[0044] 이때, 컨트롤러(110)는 캡슐화를 담당하는 서비스 기능 분류자 모듈(231)에 각 서비스 체인(SRC)별로 거쳐야 하는 서비스 기능 모듈과 그 순서를 저장할 수 있다.

[0045] 또한, 컨트롤러(110)는 서비스 기능 모듈 자신이 처리한 패킷이 다음 서비스 기능 모듈을 방문할 수 있게 하는 서비스 기능 전달자(SFF) 기능을 수행할 수 있도록, 패킷 헤더 내 저장된 서비스 체인 정보를 읽고 해당하는 출력 포트로 패킷을 포워딩하는 서비스 기능 체이닝 포워딩 룰을 저장할 수 있다.

[0046] 상기와 같이 초기 서비스 체인 정보 쓰기가 끝나면 컨트롤러(110)는 1초간 대기하는데, 이는 컨트롤러(110)가 매초마다 프로그래머블 스위치(230)의 포트 정보를 감시하기 위함이다. 이때, 1초는 네트워크 사용자가 지정하는 바람직한 수치로서 반드시 이에 한정되는 것은 아니며, 1초 미만 혹은 1초를 초과하는 임의의 수치로 변경될 수도 있다.

[0047] 또한, 컨트롤러(110)는 1초 대기가 끝나면 각 서비스 기능 모듈(232, 233)의 포트 상태를 읽어올 수 있다. 이 과정은, 프로그래머블 스위치(230)가 제공하는 Barefoot Runtime API를 참조하여 프로그래머블 스위치(230) 내 포트 색인번호에 해당하는 \$DEV-PORT값을 읽음으로써 이루어질 수 있다.

- [0048] 여기서, 컨트롤러(110)가 읽은 \$DEV_PORT 값이 Down에 해당하는 값이 아니라면 “No”의 과정을 거쳐 컨트롤러(110)는 다시 1초간 대기 후 각 서비스 기능 모듈(232, 233)의 포트 상태를 읽는 동작을 반복하여 수행할 수 있다.
- [0049] 반면, \$DEV_PORT의 값이 Down에 해당하는 값이라면 “YES”의 과정을 거쳐 컨트롤러(110)는 해당 서비스 기능 모듈을 경유하는 서비스 체인(SRC)들의 대체 서비스 체인 정보를 서비스 기능 분류자 모듈(231)에 기록하여 새로 네트워크(10)에 진입하는 패킷이 더 이상 장애가 발생한 서비스 기능 모듈을 경유하지 않도록 할 수 있다.
- [0050] 이후, 컨트롤러(110)는 다시 1초를 대기하고 사용자에게 의해 강제종료 될 때까지 그 다음 과정들을 반복 수행할 수 있다.
- [0051] 도 3은 본 발명의 실시 예에 따른 서비스 기능 체이닝 패킷 헤더의 구조를 도식화한 것이다.
- [0052] 도 3을 참조하면, 본 발명의 실시 예에 따른 서비스 기능 체이닝 패킷 헤더는 이더넷(ETH) 헤더와 IP 헤더 사이에 고가용성을 지원하는 서비스 기능 체이닝을 위한 두 개의 헤더를 포함할 수 있다.
- [0053] 첫째는 서비스 체인 헤더(Service Chain Header, SCH)로, 서비스 체인(SRC) 식별자인 ID 필드와 현재까지 거친 서비스 기능(SF)의 수를 역순으로 셈하는 LEN 필드를 포함하여 구성될 수 있다.
- [0054] 둘째는 서비스 기능 헤더(Service Function Header, SFH)로, 서비스 기능 헤더는 패킷이 방문해야 할 서비스 기능 모듈의 식별자를 기록하는 SF 필드와, 상기 SF 필드의 위치를 표식하는 TAIL 필드를 포함할 수 있다. 즉, SF 필드는 해당 서비스 기능 모듈이 해당 서비스 체인(SRC)의 마지막임을 나타낼 수 있다. 여기서, 서비스 체인(SRC)의 최대길이 n은 사용자의 설정에 의해 조정될 수 있으며, 서비스 기능 헤더는 n개의 SF 필드와 TAIL 필드의 쌍으로 구성될 수 있다.
- [0055] 도 4는 본 발명의 실시 예에 따른 프로그래머블 스위치에서 패킷 헤더를 파싱하는 과정을 도식화한 것이다.
- [0056] 도 4를 참조하면, 프로그래머블 스위치(230)의 인그레스 파서(Ingress parser)는 입력 포트에 패킷이 수신되면 패킷 헤더 내의 서비스 체인 정보를 이더넷(ETH) 헤더부터 읽어오기 시작할 수 있다.
- [0057] 여기서, 인그레스 파서는 서비스 체인 헤더(SCH)를 파싱 한 후, 서비스 기능 헤더(SFH)를 파싱하게 되는데, 이때 파싱 과정은 서비스 기능 헤더(SFH)의 서브헤더 중 TAIL 필드 값을 확인하여 해당 값이 1일 경우 IP 헤더를 파싱 한 후, 전체 파싱 과정을 종료할 수 있다.
- [0058] 반면, TAIL 필드 값이 0이라면 아직 파싱해야하는 서브헤더가 더 남아있다는 뜻이므로, TAIL 필드 값이 1인 서브헤더를 만날 때까지 서비스 기능 헤더 파싱을 반복할 수 있다.
- [0059] 도 5는 본 발명의 실시 예에 따른 프로그래머블 스위치에서 패킷을 처리하는 인그레스 파이프라인(Ingress pipeline)을 도식화한 것이다.
- [0060] 도 5를 참조하면, 패킷 처리가 시작되면 프로그래머블 스위치(230)는 IP 헤더 내 Type of Service (ToS) 필드 값을 읽어 0 이상인지 확인할 수 있다. 여기서, ToS 필드의 기본 값은 0으로써, 서비스 기능 체이닝 패킷이 아님을 의미하며, ToS 필드가 1 이상인 경우 서비스 기능 체이닝 패킷임을 뜻한다. 또한, 이 경우 ToS 필드는 서비스 체인의 색인 번호 역할을 할 수 있다.
- [0061] 따라서, ToS 필드 값이 0이라면 바로 이더넷 혹은 IP 패킷 포워딩을 실시하고 패킷 처리 과정을 종료할 수 있다. 반면, ToS 필드 값이 1 이상이라면 다음 단계로 넘어갈 수 있다.
- [0062] ToS 필드 값이 1 이상인 패킷에 대하여 프로그래머블 스위치(230)는 서비스 기능 헤더(SCH)가 존재하는지 여부를 isvalid() 함수를 통해 그 유효성을 검증할 수 있다. 여기서, 서비스 기능 헤더(SCH)가 존재한다는 것은 현재 이 패킷에 대한 캡슐화가 완료되었다는 의미이다.
- [0063] 즉, isvalid() 함수에 대한 답이 “No”인 경우 캡슐화(Encapsulation)을 진행하여 서비스 체인 헤더(SCH)와 서비스 기능 헤더(SFH)를 패킷에 추가한 후 처리를 받을 수 있다.
- [0064] 서비스 체인 헤더(SCH)가 존재하거나 캡슐화가 완료되었다면, 프로그래머블 스위치(230)는 해당 패킷에 대한 서비스 기능 처리(SF Processing)을 진행할 수 있다. 이때, 서비스 기능(SF)은 NAT와 방화벽 등 사용자가 해당 프로그래머블 스위치(230)에 설정한 과정을 따라 진행될 수 있다.
- [0065] 만약, 사용자가 해당 프로그래머블 스위치(230)에 서비스 기능(SF)을 설정하지 않은 경우, 패킷은 서비스 기능

(SF)이 설정되지 않은 서비스 기능 모듈은 처리 과정을 거치지 않고 그대로 통과할 수 있다.

- [0066] 서비스 기능(SF) 처리를 받게 된 후의 프로그래머블 스위치(230)는 서비스 체인 헤더(SCH)의 LEN 필드 값을 1만큼 감소시키며, 서비스 기능 헤더(SFH)의 가장 앞 서브헤더인 SF-TAIL 쌍을 Pop-front() 함수를 통해 제거할 수 있다. 여기서 LEN 필드의 초기값은 서비스 체인의 길이이다. 상기와 같은 과정은 현재 패킷이 서비스 체인(SRC)이 명시한 서비스 기능 모듈을 경유하여 정상적인 서비스 기능(SF) 처리를 받았음을 나타낸다.
- [0067] 이후, 프로그래머블 스위치(230)는 서비스 체인 헤더(SCH)의 LEN 필드 값이 0인지를 확인할 수 있다. LEN 필드 값이 0이라는 것은 현재 패킷이 서비스 체인에서 명시한 서비스 기능(SF)들을 모두 방문하였다는 것을 의미한다. 따라서, 이 경우 “YES”의 과정을 거쳐 프로그래머블 스위치(230)는 해당 패킷에 대한 역캡슐화(Decapsulation)을 진행함으로써 서비스 체인 헤더(SCH)와 서비스 기능 헤더(SFH)를 제거한 후, ETH/IP 패킷 포워딩을 실시하고 패킷 처리를 종료할 수 있다. 만약, LEN 필드의 값이 0이 아니라면 다음 단계인 “NO”의 과정을 거쳐 SF Forward로 넘어간다.
- [0068] SF Forward는 서비스 기능 전달자(SFF)의 기능을 수행할 수 있다. 구체적으로, 프로그래머블 스위치(230)가 저장하고 있는 서비스 기능 체이닝 포워딩 룰을 참조하여 현재 패킷의 출력 포트를 결정할 수 있다. 여기서, 서비스 기능 체이닝 포워딩 룰의 키 값은 서비스 기능 모듈의 ID이며 반환되는 값은 출력 포트 번호이다. 이러한 SF Forward에 의해 처리된 후 해당 패킷에 대한 처리는 완료될 수 있다.
- [0070] 이하, 하기 실시 예를 통하여 본 발명의 서비스 기능 체이닝 시스템을 이용한 서비스 기능 체이닝 시의 패킷 헤더의 변화와, 네트워크 장애에 컨트롤러가 대응하는 과정을 살펴보기로 한다.
- [0071] 하기 실시 예는 본 발명을 보다 상세히 설명하기 위한 과정일 뿐, 본 발명의 내용이 하기 실시 예에 반드시 한정되는 것은 아니다.
- [0073] **[실시예 1] 본 발명의 서비스 기능 체이닝 시스템을 이용한 서비스 기능 체이닝 시의 패킷 헤더의 변화**
- [0074] 도 6은 본 발명의 실시 예에 따른 서비스 기능 체이닝 시 패킷 헤더의 변화를 나타낸 도면이다.
- [0075] 본 실시예에서는 도 1을 참조하여 설명한 네트워크 환경과는 무관하게 4개의 서비스 기능 모듈을 거치는 경우만을 고려하였다.
- [0076] 처음 서비스 기능 체이닝 패킷이 네트워크(10)에 진입하면 서비스 기능 분류자 모듈(231)가 캡슐화를 진행하여 서비스 체인 헤더(SCH)와 서비스 기능 헤더(SFH)를 기존 패킷 헤더에 삽입한다. 이후, 서비스 기능 모듈을 거칠 때마다 서비스 체인 헤더(SCH)의 LEN 필드 값은 1씩 감소하며, 서비스 기능 헤더(SFH)의 가장 앞에 위치한 SF-Tail 필드 쌍이 차례로 제거된다.
- [0077] 제2 서비스 기능 모듈(233)에서 마지막 서비스 기능(SF) 처리를 받은 직후 패킷에는 더 이상 서비스 기능 헤더(SFH)가 존재하지 않으며, LEN 필드 값이 0인 서비스 체인 헤더(SCH)만 남아있게 된다. 이때, 프로그래머블 스위치(230)는 역캡슐화를 진행하여 서비스 체인 헤더(SCH) 역시 제거 후 패킷을 내보내게 된다.
- [0079] **[실시예 2] 본 발명의 서비스 기능 체이닝 시스템을 이용한 서비스 기능 체이닝 시의 네트워크 장애에 컨트롤러가 대응하는 과정**
- [0080] 도 7은 본 발명의 실시 예에 따른 서비스 기능 체이닝 시스템을 이용한 서비스 기능 체이닝 시의 네트워크 장애에 컨트롤러가 대응하는 과정을 보여주는 도면이다.
- [0081] 본 실시예에서는 기존 서비스 체인(SRC1)을 SF1의 제1 서비스 기능 모듈(232a)-SF3의 제2 서비스 기능 모듈(233), 대체 서비스 체인(SRC2)을 SF2의 제1 서비스 기능 모듈(232a)-SF3의 제2 서비스 기능 모듈(233)로 정하였다. 또한, 서비스 체인(SRC)의 최대 길이는 4인 경우를 고려하였다. 또한, 클라이언트(210)가 서버(220)로 보내는 서비스 기능 체이닝 패킷들의 ToS 필드 값은 1로 사용한다. 즉, 클라이언트(210)는 서비스 체인 ID 1을 사용한다.
- [0082] SC1의 서비스 기능 분류자 모듈(231)에 패킷이 도착하였을 때, SC1의 서비스 기능 분류자 모듈(231)은 도면에 도시된 ①과 같이 자신이 저장하고 있는 서비스 체인 정보를 참조하여 캡슐화를 진행한다. 이후, SF Forward를

통해 패킷은 SF1의 제1 서비스 기능 모듈(232a)으로 전송된다.

[0083] SF1의 제1 서비스 기능 모듈(232a)은 서비스 기능(SF) 처리 후 패킷을 SF3의 제2 서비스 기능 모듈(233)로 전달한다. 그러나, ②와 같이 SF1의 제1 서비스 기능 모듈(232a)-SF3의 제2 서비스 기능 모듈(233)간 회선에 장애가 발생하여 SF1의 제1 서비스 기능 모듈(232a)의 패킷이 SF3의 제2 서비스 기능 모듈(233)로 전달되지 못한다.

[0084] 이때, 본 실시예의 컨트롤러(110)는 이러한 네트워크 장애를 감지하여 ③과 같이 SC1의 서비스 기능 분류자 모듈(231)에 저장되어 있는 서비스 체인 ID 1의 기본 서비스 체인(SRC1) 정보를 대체 서비스 체인(SRC2) 정보로 갱신한다.

[0085] 이에 따라, 이후 들어오는 서비스 체인 ID 1을 이용하는 패킷들은 더 이상 SF1의 제1 서비스 기능 모듈(232)-SF3의 제2 서비스 기능 모듈(233)이 아니라 ④와 같이 SF2의 제1 서비스 기능 모듈(232b)-SF3의 제2 서비스 기능 모듈(233)의 경로를 경유하여 서버(220)로 전달된다.

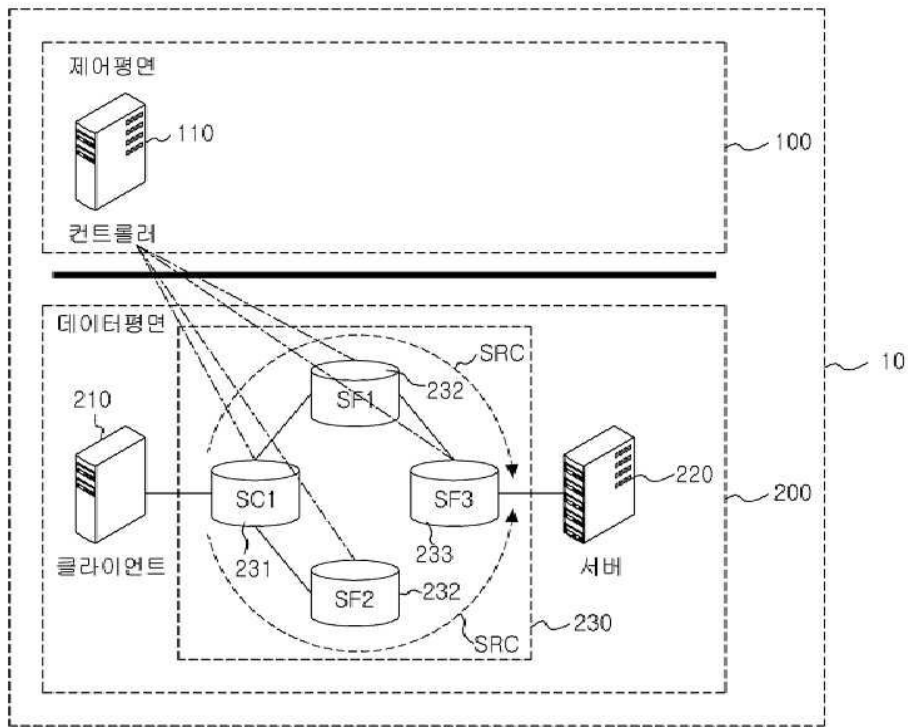
[0087] 이상으로 첨부된 도면을 참조하여 본 발명의 실시예를 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고 다른 구체적인 형태로 실시할 수 있다는 것을 이해할 수 있을 것이다. 따라서 이상에서 기술한 실시예는 모든 면에서 예시적인 것이며 한정적이 아닌 것이다.

부호의 설명

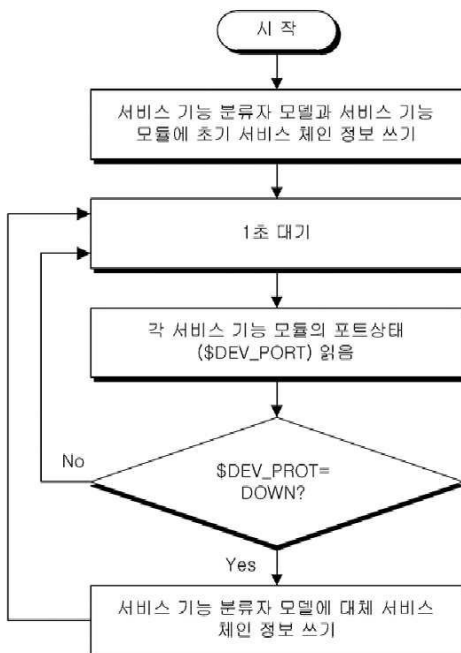
[0089] 10 : 네트워크
100 : 제어평면
110 : 컨트롤러
200 : 데이터평면
210 : 클라이언트
220 : 서버
230 : 프로그래머블 스위치
231 : 서비스 기능 분류자 모듈
232 : 제1 서비스 기능 모듈
233 : 제2 서비스 기능 모듈
SRC : 서비스 체인

도면

도면1



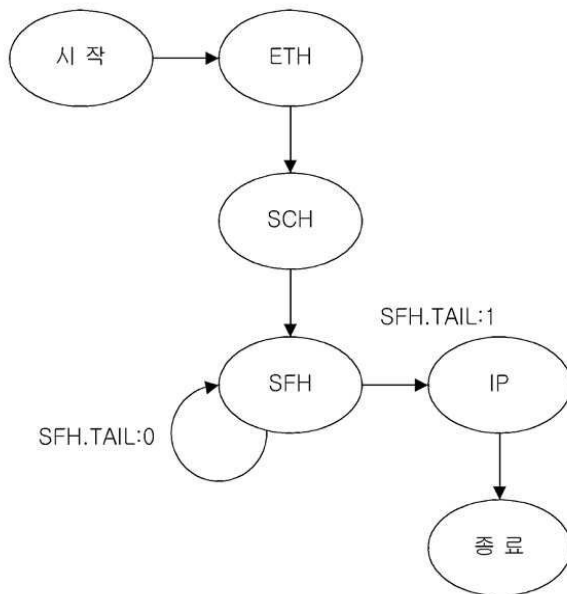
도면2



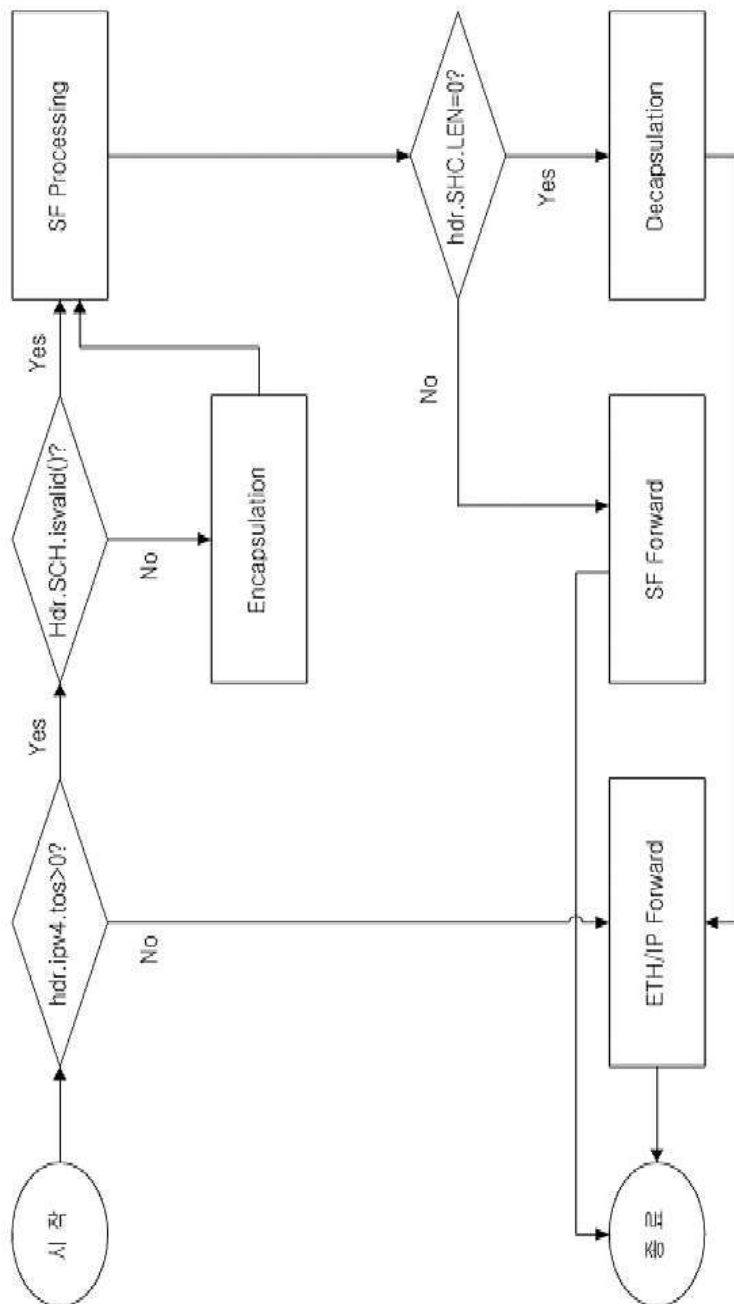
도면3



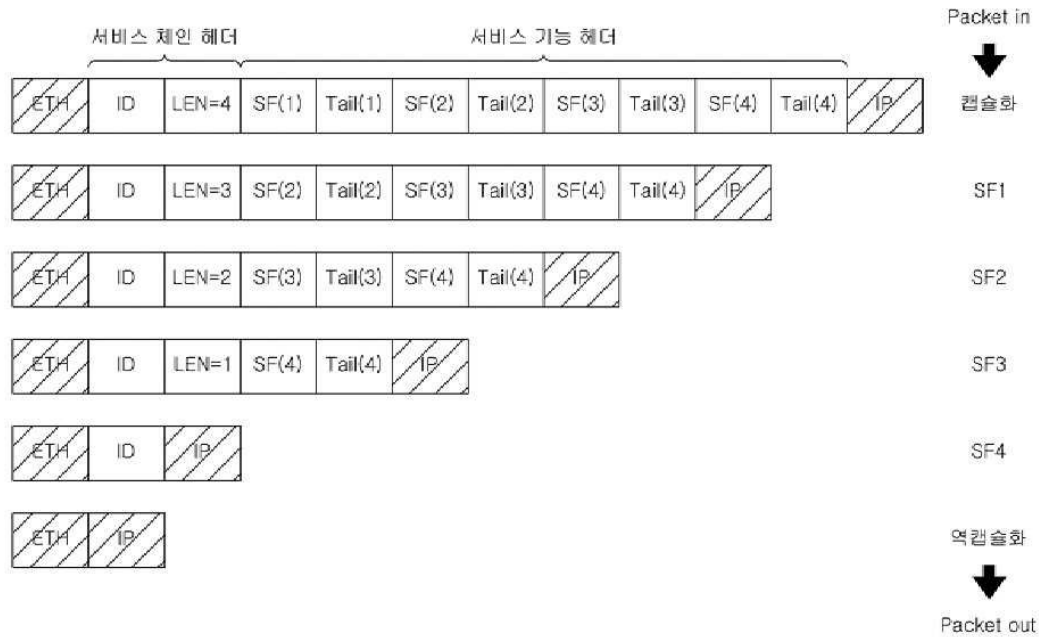
도면4



도면5



도면6



도면7

