



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년10월04일
(11) 등록번호 10-2028650
(24) 등록일자 2019년09월27일

- | | |
|---|---|
| <p>(51) 국제특허분류(Int. Cl.)
H04L 12/931 (2013.01) H04L 12/935 (2013.01)
H04L 29/06 (2006.01)</p> <p>(52) CPC특허분류
H04L 49/70 (2013.01)
H04L 47/2483 (2013.01)</p> <p>(21) 출원번호 10-2018-0146621</p> <p>(22) 출원일자 2018년11월23일
심사청구일자 2018년11월23일</p> <p>(56) 선행기술조사문헌
draft-ietf-sfc-control-plane-02
draft-fedyk-sfc-mac-chain-02
David Hancock. HyPer4: Using P4 to Virtualize the Programmable</p> | <p>(73) 특허권자
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)</p> <p>(72) 발명자
백상현
서울특별시 서초구 고무래로 35, 109동 1004호(반포동, 반포리체아파트)</p> <p>이재욱
서울특별시 용산구 청파로57길 28-7, 401호
(뒷면에 계속)</p> <p>(74) 대리인
김홍석</p> |
|---|---|

전체 청구항 수 : 총 5 항

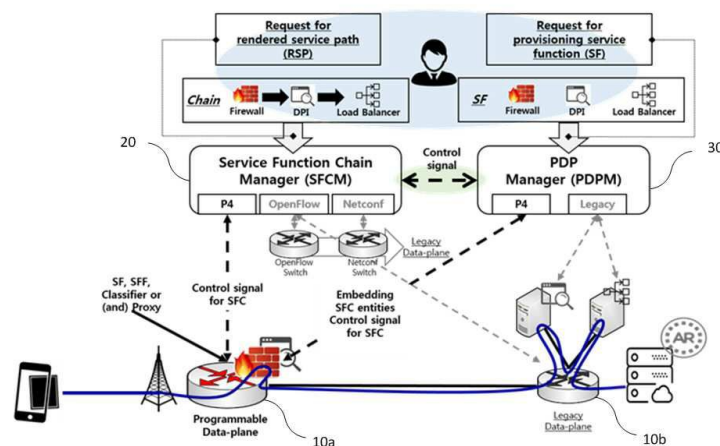
심사관 : 김진권

(54) 발명의 명칭 프로그래밍 가능한 스위치 기반의 서비스 기능 체이닝 시스템 및 방법

(57) 요약

서비스 기능 체인 시스템이 개시된다. 상기 서비스 기능 체인 시스템은 PDP(Programmable Data Plane) 상에 SFC(Service Function Chain) 순서를 고려하여 SFC 엔티티들을 구현하는 PDP 관리 모듈, 및 상기 PDP 상에 구현할 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 상기 PDP 관리 모듈에 전달하거나, 상기 PDP 상에 포함된 노드에 SFC 룰을 전달하는 SFC 관리 모듈을 포함하고, 상기 PDP 관리 모듈은 상기 제1 SFC 엔티티 정보를 수신하는 수신부, 상기 SFC 엔티티들이 상기 PDP 상에 구현되고 나서 SFC 순서에 따라 상기 패킷을 처리할 수 있도록 상기 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하는 가공부, 및 상기 제2 SFC 엔티티 정보를 이용하여 상기 SFC 엔티티들을 상기 PDP 상에 구현하는 구현부를 포함한다.

대표도 - 도4



(52) CPC특허분류

H04L 49/3063 (2013.01)

H04L 69/22 (2013.01)

H04L 2212/00 (2014.06)

(72) 발명자

고한열

경기도 고양시 덕양구 화신로 170, 2111동 1504호

이호찬

서울특별시 광진구 용마산로5길 55, 101호

이 발명을 지원한 국가연구개발사업

과제고유번호 20150005750042003

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정보통신방송연구개발사업

연구과제명 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발

기 여 율 1/2

주관기관 고려대학교 산학협력단

연구기간 2018.04.01 ~ 2019.01.31

이 발명을 지원한 국가연구개발사업

과제고유번호 20170001950021001

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정보통신방송연구개발사업

연구과제명 멀티 서비스를 지원하는 프로그래머블 스위치 제어 기술 개발

기 여 율 1/2

주관기관 고려대학교 산학협력단

연구기간 2018.01.01 ~ 2018.12.31

명세서

청구범위

청구항 1

PDP(Programmable Data Plane) 상에 SFC(Service Function Chain) 순서를 고려하여 SFC 엔티티들을 구현하는 PDP 관리 모듈; 및

상기 PDP 상에 구현할 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 상기 PDP 관리 모듈에 전달하거나, 상기 PDP 상에 포함된 노드에 SFC 룰을 전달하는 SFC 관리 모듈을 포함하고,

상기 PDP 관리 모듈은,

상기 제1 SFC 엔티티 정보를 수신하는 수신부;

상기 SFC 엔티티들이 상기 PDP 상에 구현되고 나서 SFC 순서에 따라 패킷을 처리할 수 있도록 상기 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하는 가공부; 및

상기 제2 SFC 엔티티 정보를 이용하여 상기 SFC 엔티티들을 상기 PDP 상에 구현하는 구현부를 포함하고,

상기 SFC 엔티티들은 SFC 식별자(SFC classifier), SF(service function), SFF(service function forwarder) 및 SFC 프록시(SFC proxy)를 포함하고,

상기 제1 SFC 엔티티 정보는 매치(match) 필드와 동작(action) 필드를 포함하고,

상기 가공부는,

상기 SF가 상기 SFC 순서에 따라 상기 패킷을 처리하도록 하기 위해 상기 SF에 대한 제1 SFC 엔티티 정보의 매치 필드에 SPI(service path identifier)와 SI(service index)를 포함하는 NSH(network service header)를 추가하고,

상기 SFF에 대한 제1 SFC 엔티티 정보의 동작 필드에 루프백(loopback) 동작을 추가하여 상기 제2 SFC 엔티티 정보를 생성하고,

상기 PDP 상에 상기 SF와 상기 SFC 프록시가 동시에 구현될 경우 상기 SF가 NSH가 추출된 패킷을 드롭(drop)시키지 않도록 상기 SF에 대한 제1 SFC 엔티티 정보의 매치 필드 및 동작 필드에 각각 SIP(source IP)/DIP(destination IP)와 통과(pass) 동작을 추가하고,

상기 PDP 상에 상기 SFC 식별자와 상기 SF가 동시에 구현될 경우 SFC 식별자가 상기 SFF의 루프백 동작에 의해 되돌아온 패킷을 드롭시키지 않도록 매치 필드에 SPI를 추가하고,

상기 PDP 상에 상기 SFC 식별자와 상기 SFC 프록시가 동시에 구현될 경우 상기 SFC 프록시에 의해 NSH가 추출된 패킷을 드롭시키지 않도록 상기 SFC 식별자에 대한 제1 SFC 엔티티 정보의 매치 필드 및 동작 필드에 각각 SIP/DIP와 통과 동작을 추가하여 상기 제2 SFC 엔티티 정보를 생성하는,

서비스 기능 체인 시스템.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

제1항에 있어서,

상기 구현부는 상기 패킷으로부터 헤더를 추출하기 위한 파서(parser)와, 상기 패킷으로부터 추출된 헤더를 조합하기 위한 디파서(deparser)를 정의하고, 상기 파서와 상기 디파서를 상기 PDP 상에 구현하는 서비스 기능 체인 시스템.

청구항 6

제5항에 있어서,

상기 구현부는 상기 파서가 상기 패킷으로부터 헤더를 추출하기 위해 필요한 파싱 그래프(parsing graph)를 상기 제2 SFC 엔티티 정보를 이용하여 생성하고,

상기 파서는 상기 파싱 그래프를 참조하여 상기 패킷으로부터 헤더를 추출하는 서비스 기능 체인 시스템.

청구항 7

서비스 기능 체인 시스템에 의해 수행되는 서비스 기능 체인 방법으로서,

PDP(Programmable Data Plane) 상에 구현할 SFC(Service Function Chain) 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 수신하는 단계;

상기 SFC 엔티티들이 SFC 순서에 따라 패킷을 처리할 수 있도록 상기 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하는 단계;

상기 패킷으로부터 헤더를 추출하기 위한 파서(parser)와 상기 패킷으로부터 추출된 헤더를 조합하기 위한 디파서(deparser)를 정의하는 단계; 및

상기 파서, 상기 제2 SFC 엔티티 정보, 및 상기 디파서를 이용하여 상기 SFC 엔티티들을 상기 PDP 상에 구현하는 단계를 포함하고,

상기 SFC 엔티티들은 SFC 식별자(SFC classifier), SF(service function), SFF(service function forwarder) 및 SFC 프록시(SFC proxy)를 포함하고,

상기 제1 SFC 엔티티 정보는 매치(match) 필드와 동작(action) 필드를 포함하고,

상기 제2 SFC 엔티티 정보를 생성하는 단계는, 상기 SF가 상기 SFC 순서에 따라 상기 패킷을 처리하도록 하기 위해 상기 SF에 대한 제1 SFC 엔티티 정보의 매치 필드에 SPI(service path identifier)와 SI(service index)를 포함하는 NSH(network service header)를 추가하고, 상기 SFF에 대한 제1 SFC 엔티티 정보의 동작 필드에 루프백(loopback) 동작을 추가하여 상기 제2 SFC 엔티티 정보를 생성하고,

상기 제2 SFC 엔티티 정보를 생성하는 단계는,

상기 SF가 상기 SFC 순서에 따라 상기 패킷을 처리하도록 하기 위해 상기 SF에 대한 제1 SFC 엔티티 정보의 매치 필드에 SPI(service path identifier)와 SI(service index)를 포함하는 NSH(network service header)를 추가하는 단계,

상기 SFF에 대한 제1 SFC 엔티티 정보의 동작 필드에 루프백(loopback) 동작을 추가하여 상기 제2 SFC 엔티티 정보를 생성하는 단계,

상기 PDP 상에 상기 SF와 상기 SFC 프록시가 동시에 구현될 경우 상기 SF가 NSH가 추출된 패킷을 드롭(drop)시키지 않도록 상기 SF에 대한 제1 SFC 엔티티 정보의 매치 필드 및 동작 필드에 각각 SIP(source IP)/DIP(destination IP)와 통과(pass) 동작을 추가하는 단계,

상기 PDP 상에 상기 SFC 식별자와 상기 SF가 동시에 구현될 경우 SFC 식별자가 상기 SFF의 루프백 동작에 의해 되돌아온 패킷을 드롭시키지 않도록 매치 필드에 SPI를 추가하는 단계, 및

상기 PDP 상에 상기 SFC 식별자와 상기 SFC 프록시가 동시에 구현될 경우 상기 SFC 프록시에 의해 NSH가 추출된 패킷을 드롭시키지 않도록 상기 SFC 식별자에 대한 제1 SFC 엔티티 정보의 매치 필드 및 동작 필드에 각각 SIP/DIP와 통과 동작을 추가하여 상기 제2 SFC 엔티티 정보를 생성하는 단계를 포함하는, 서비스 기능 체인 방법.

청구항 8

삭제

청구항 9

제7항에 있어서,

상기 파서와 디파서를 정의하는 단계는

상기 파서가 상기 패킷으로부터 헤더를 추출하기 위해 필요한 파싱 그래프(parsing graph)를 상기 제2 SFC 엔티티 정보를 이용하여 생성하는 단계를 더 포함하는 서비스 기능 체인 방법.

발명의 설명

기술 분야

[0001] 본 발명은 서비스 기능 체이닝(Service Function Chaining, SFC) 시스템 및 방법에 대한 것으로서, 보다 구체적으로 SFC 특성을 고려하여 서비스 기능(SF)들을 프로그래밍 가능한 스위치에 구현 및 지원할 수 있는 서비스 기능 체이닝 시스템 및 방법에 대한 것이다.

배경 기술

[0002] 네트워크 사업자는 네트워크 기능(Network function 또는 Service function, 이하 NF 또는 SF)을 통해 네트워크 상의 트래픽들을 관리한다. 이때, 사업자의 목적에 따라 트래픽을 처리하는 SF들의 집합을 네트워크 기능 체인(Network Function Chain) 또는 서비스 기능 체인(Service Function Chain, 이하 SFC)라고 한다. 즉, 트래픽은 사업자가 정의한 체인에 따라 처리됨으로써, 사업자는 네트워크 상의 트래픽을 관리한다. SF는 범용서버에 구현되거나 특정 SF를 위한 장비를 통해 네트워크 내의 트래픽을 처리하게 된다. 즉, SF를 구현할 범용서버나 특정장비의 위치에 따라 네트워크 내의 트래픽 경로가 설정되므로, 네트워크 경로가 비효율적이며, 특히 저지연을 요구하는 트래픽의 요구사항 충족이 어렵다는 문제점이 있다.

[0003] 한편, 효율적인 트래픽 경로 설정을 위해 네트워크 사용자가 원하는 기능을 데이터 평면에 프로그래밍 가능케 하는 프로그래밍 가능한 데이터 평면(Programmable Data Plane, 이하 PDP)에 SF(예를 들면, Firewall, DPI, NAT 등)를 구현하는 연구가 진행되고 있다. 일 예로서, 데이터 센터 내의 소프트웨어 기반의 Layer-4 부하 분산(Load Balancing) 기능을 P4 언어를 통해 PDP에 구현하여, line-rate의 속도로 부하 분산과 사용자와 서버 간의 연결성 지속성을 보장해줄 수 있는 연구가 있다. 다른 일 예로서, PDP의 in-memory 저장소에 캐시 레이어를 구현함으로써, key-value 저장소 기반의 데이터 센터 내에 서비스 요청에 의한 부하들을 완화시키며, 스위치 내의 캐시 레이어를 통해 빠른 응답시간을 제공하는 연구가 있다. 그러나, 상술한 연구들은 SF가 구현된 PDP를 지나 는 모든 트래픽을 처리하는 문제가 존재하므로, SFC 특성을 고려하여 SF를 PDP에 구현할 필요성이 있다.

선행기술문헌

특허문헌

[0004] (특허문헌 0001) 대한민국 등록특허 제1800145호

(특허문헌 0002) 대한민국 등록특허 제1823346호

발명의 내용

해결하려는 과제

[0005] 본 발명의 목적은 서비스 기능 체인(SFC) 특성을 고려하여 서비스 기능(SF)들을 프로그래밍 가능한 데이터 평면(PDP)에 구현함으로써 PDP에 SF가 구현될 경우 PDP를 지나는 모든 트래픽을 처리하는 문제를 해결할 수 있는 프로그래밍 가능한 스위치 기반의 서비스 기능 체이닝 시스템 및 방법을 제공하는데 있다.

과제의 해결 수단

[0006] 본 발명의 일 실시 예에 따른 서비스 기능 체이닝 시스템은 PDP(Programmable Data Plane) 상에 SFC(Service Function Chain) 순서를 고려하여 SFC 엔티티들을 구현하는 PDP 관리 모듈, 및 상기 PDP 상에 구현할 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 상기 PDP 관리 모듈에 전달하거나, 상기 PDP 상에 포함된 노드에 SFC 룰을 전달하는 SFC 관리 모듈을 포함하고, 상기 PDP 관리 모듈은 상기 제1 SFC 엔티티 정보를 수신하는 수신부, 상기 SFC 엔티티들이 상기 PDP 상에 구현되고 나서 SFC 순서에 따라 상기 패킷을 처리할 수 있도록 상기 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하는 가공부, 및 상기 제2 SFC 엔티티 정보를 이용하여 상기 SFC 엔티티들을 상기 PDP 상에 구현하는 구현부를 포함할 수 있다.

[0007] 본 발명의 일 실시 예에 따른 서비스 기능 체이닝 방법은 서비스 기능 체인 시스템에 의해 수행되는 서비스 기능 체인 방법으로서, PDP(Programmable Data Plane) 상에 구현할 SFC(Service Function Chain) 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 수신하는 단계, 상기 SFC 엔티티들이 SFC 순서에 따라 상기 패킷을 처리할 수 있도록 상기 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하는 단계, 상기 패킷으로부터 헤더를 추출하기 위한 파서(parser)와, 상기 패킷으로부터 추출된 헤더를 조합하기 위한 디파서(deparser)를 정의하는 단계, 및 상기 파서, 상기 제2 SFC 엔티티 정보 및 상기 디파서를 이용하여 상기 SFC 엔티티들을 상기 PDP 상에 구현하는 단계를 포함할 수 있다.

발명의 효과

[0008] 본 발명의 일 실시 예에 따르면, SFC 특성을 고려하여 SF들을 PDP에 구현함으로써 PDP에 SF가 구현될 경우 PDP를 지나는 모든 트래픽을 처리하는 문제를 해결하고 SFC의 요구 사항을 빠른 처리속도로 만족할 수 있다.

[0009] 또한, 스위치가 PDP의 스위치 내에 직접 매치 키들과 액션들을 정의하기 때문에 오픈 플로우(Openflow)에서는 지원하지 못하는 다양한 SF들을 스위치 내에 구현할 수 있다.

도면의 간단한 설명

[0010] 도 1은 기존의 SFC(Service Function Chaining) 환경을 도시한 것이다.
 도 2는 NSH(Network Service Header)의 구조를 도시한 것이다.
 도 3은 기존의 SFC 시스템을 도시한 것이다.
 도 4는 본 발명의 일 실시 예에 따른 SFC 시스템을 도시한 것이다.
 도 5는 본 발명의 일 실시 예에 따른 SFC 시스템에 포함된 PDP(Programmable Data Plane) 관리 모듈의 블록도이다.
 도 6은 SFC 엔티티 정보의 가공 과정을 설명하기 위한 도면이다.
 도 7은 파싱 그래프(parsing graph)를 생성하는 과정을 설명하기 위한 도면이다.
 도 8은 PDP 스위치 구조를 도시한 것이다.
 도 9는 본 발명의 일 실시 예에 따른 SFC 시스템이 적용된 PDP를 도시한 것이다.
 도 10은 본 발명의 일 실시 예에 따른 SFC 방법의 순서도이다.
 도 11은 본 발명과 기존의 SFC 간에 성능을 비교하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0011] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예

들은 다양한 형태로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

- [0012] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에 상세하게 설명하고자 한다. 그러나 이는 본 발명의 개념에 따른 실시예들을 특정한 개시형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 변경, 균등물, 또는 대체물을 포함한다.
- [0013] 제1 또는 제2 등의 용어를 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만, 예를 들어 본 발명의 개념에 따른 권리 범위로부터 이탈되지 않은 채, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.
- [0014] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성요소들 간의 관계를 설명하는 표현들, 예를 들어 "~사이에"와 "바로~사이에" 또는 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.
- [0015] 본 명세서에서 사용한 용어는 단지 특정한 실시예들을 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 실시된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함으로 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0016] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다. 이하, 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다.
- [0018] 도 1은 기존의 SFC(Service Function Chaining) 환경을 도시한 것이다.
- [0019] 도 1을 참조하면, 서비스 기능 체이닝(Service function chaining, 이하 SFC)은 IETF에서 표준화되고 있으며, 표준화된 SFC 환경은 제어 평면(control plane)과 데이터 평면(또는 사용자 평면, data plane)을 포함한다.
- [0020] 데이터 평면은 SFC 엔티티들로 구성된다. SFC 엔티티들은 각각 SFC 환경에서 패킷을 처리하는 논리적 객체를 의미하는 것으로서, SFC 식별자(SFC classifier), Service Function Forwarder(이하, SFF), SFC 프록시(SFC-Proxy) 및 Service Function(이하, SF)를 포함할 수 있다. 또한, 상기 SFC 엔티티들로 구성된 네트워크를 SFC-enabled domain(or network) 라고 명명할 수 있다. SFC 엔티티들은 Network Service Header(이하, NSH)가 캡슐화된 패킷을 처리하게 된다.
- [0021] NSH(Network Service Header)의 구조를 도시한 도 2를 참조하면, NSH에서 Service Path Identifier(이하, SPI)를 통해 플로우 별로 따라야 하는 체인을 구분하고, Service Index(SI)를 통해 플로우가 체인 내에서 몇 개의 SF 인스턴스가 처리됐는지를 인식한다. SFC 엔티티들은 NSH를 읽고, 각 엔티티 별로 서로 다른 역할을 수행한다. 각 엔티티 별로 수행하는 역할은 다음과 같다.
- [0022] 1) SFC 식별자 : SFC 식별자는 들어오는 패킷 헤더에 NSH를 삽입하여 SFC 캡슐화를 수행한다. 어떤 NSH를 삽입할 것인지는 SFC 환경의 제어 평면에서 결정하여 SFC 식별자에 SFC 캡슐화 물을 내려준다. 이때, 캡슐화 물은 패킷에 삽입될 NSH에 포함될 정보를 의미할 수 있다.
- [0023] 2) SFF : SFF는 패킷 헤더에 삽입된 NSH의 SPI와 SI를 읽은 후에 미리 정의된 포워딩 룰에 따라 패킷을 전달한다. 포워딩 룰은 SFC 환경의 제어 평면에서 정의하고 SFF에 내려준다.
- [0024] 3) SF : SF는 SFC-aware SF와 SFC-unaware SF를 포함할 수 있다. SFC-aware SF는 NSH가 삽입된 패킷을 자신의 네트워크 기능에 맞게 처리하고, 처리한 후에 SI를 수정한다. 한편, SFC-unaware SF의 경우 NSH 헤더 처리를 하지 못하며, NSH 헤더가 삽입된 패킷을 처리할 수 없다.

- [0025] 4) SF 프록시 : SF 프록시는 상술한 SFC-unaware SF가 패킷을 처리할 수 있도록 패킷 헤더에서 NSH를 제거(de-capsulation)한 후에 패킷을 SF에 전달하고, SF로부터 받은 패킷 헤더에 NSH를 삽입 후(encapsulation) SI를 수정한다.
- [0027] 도 3은 기존의 SFC 시스템을 도시한 것이다.
- [0028] 도 3을 참조하면, 기존의 SFC 시스템에서는 사용자가 서버로부터 특정 서비스를 받기 위하여 SFC 엔티티들을 각각 모두 지나가야하는 비효율적인 경로를 설정하는 문제가 발생한다. 예를 들어, SFC가 방화벽(firewall)과 DPI를 순차적으로 수행하도록 설정된 경우, SFF에서 SFC 식별자로 패킷을 포워딩하고, SFC 식별자에서 패킷에 NSH를 삽입하여 다시 SFF로 전달하고, 방화벽 SF는 패킷을 전달받아 해당 기능에 맞게 패킷을 처리하여 다시 SFF로 전달하고, DPI SF는 패킷을 전달받아 해당 기능에 맞게 패킷을 처리하여 서버로 전달한다. 또한, DPI SF는 SFC-unaware SF이므로 DPI SF에 전달되는 패킷은 SF 프록시를 거쳐야한다.
- [0029] 이러한 기존의 SFC 시스템의 문제점은 PDP(Programmable Data Plane) 상에 SF 엔티티들을 구현하여 해결할 수 있으나, 이 경우 PDP를 지나는 모든 트래픽을 처리하는 문제가 발생한다. 예를 들어, 어떤 트래픽은 NAT 기능만 처리되고, 어떤 트래픽은 방화벽 기능만 처리되도록 사용자가 정의하더라도, 경로상의 PDP에 NAT SF와 방화벽 SF가 구현되어 있으면 두 기능이 모두 처리된다. 이는 NAT만 필요로 하는 트래픽을 방화벽이 버릴(Drop) 수도 있으며, 방화벽만 필요로 하는 트래픽의 헤더를 NAT가 수정할 수도 있는 문제가 발생한다. 이러한 문제는 SFC 특성을 고려하지 않고, 단순히 SF를 PDP에 구현하기 때문에 발생하는 문제이다. 이에 따라, 본 발명에서는 SFC 특성을 고려하여 PDP 상에 SF 엔티티들을 구현할 수 있는 시스템 및 방법을 제안한다.
- [0031] 도 4는 본 발명의 일 실시 예에 따른 SFC 시스템을 도시한 것이다.
- [0032] 도 4를 참조하면, 본 발명의 일 실시 예에 따른 SFC 시스템(1)은 제어 평면과 데이터 평면을 포함한다. 여기서, 데이터 평면은 프로그래밍이 가능한 PDP를 의미할 수 있다. 또한, 데이터 평면에는 적어도 하나 이상의 노드(10a,10b)가 포함되고, 제어 평면에는 SFC 관리 모듈(20)과 PDP 관리 모듈(30)이 포함된다.
- [0033] 노드(10a,10b)란 네트워크 장치일 수 있으며, 스위치(switch) 또는 라우터(router)와 같이 트래픽 또는 패킷을 포워딩하거나 스위칭 또는 라우팅하는 기능 요소를 의미할 수 있다. 예컨대, 적어도 하나 이상의 네트워크 장치는 오픈플로(OpenFlow), IETF, ETSI 및/또는 ITU-T 등에서 정의하고 있는 스위치, 라우터, 스위칭 요소(Switching Element), 라우팅 요소(Routing Element), 또는 포워딩 요소(Forwarding Element) 등을 의미할 수 있다. 도 4에 도시된 적어도 하나 이상의 네트워크 장치들의 개수가 본 발명의 권리범위를 제한하는 것이 아님은 자명하다. 또한, 적어도 하나 이상의 네트워크 장치 각각은 PDP 관리 모듈(30)로부터 구현되는 프로그램을 저장 및 관리하기 위한 저장 공간(메모리 또는 저장부 등)을 포함할 수 있다. 노드(10a,10b)는 PDP 상에 포함되므로, 네트워크 사용자가 원하는 기능이 프로그래밍될 수 있다.
- [0034] 제어 평면에 포함되는 SFC 관리 모듈(20)은 SFC 관리를 위한 모듈이다. SFC 관리 모듈(20)은 캡슐화 물이나 포워딩 룰과 같은 SFC 룰을 설정하고, 해당 SFC 룰을 PDP 상에 포함된 노드(10a,10b)에 전달한다. 이에 따라, PDP 상에 포함된 노드(10a,10b)에 구현되는 SFC 엔티티들은 SFC 룰을 토대로 패킷을 처리한다. 또한, SFC 관리 모듈(20)은 후술할 PDP 관리 모듈(30)에 SFC 엔티티 정보를 전달한다. SFC 엔티티 정보는 각 SFC 엔티티들이 패킷에 대하여 수행하는 기능이 정의된 매치(match) 필드와 동작(action) 필드를 포함한다. SFC 엔티티들은 각 매치 필드에 정의된 정보와 일치하는 패킷에 대하여 해당 매치 필드와 쌍을 이루는 동작 필드에 정의된 기능을 패킷에 대하여 수행한다.
- [0035] SFC 관리 모듈(20)은 SFC 엔티티 정보 전달을 위해 PDP 관리 모듈(30)과 연결될 수 있으며, 또한 SFC 관리 기능을 수행하기 위해 PDP 상에 포함된 노드(10a,10b)들과 기존의 데이터 평면 상에 포함된 노드(10a,10b)들과도 연결될 수 있다.
- [0036] 제어 평면에 포함되는 PDP 관리 모듈(30)은 PDP 관리를 위한 모듈이다. PDP 관리 모듈(30)은 PDP 상에 구현할 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 수신하고, SFC 엔티티들이 PDP 상에 구현되고 나서 SFC 순서에 따라 패킷을 처리할 수 있도록 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성하고, 제2 SFC 엔티티 정보를 이용하여 SFC 엔티티들을 PDP 상에 구현(embedding)한다. PDP 관리 모듈(30)에 대한 상세한 설명은 도 5를 참조하여 후술하기로 한다.
- [0038] 도 5는 본 발명의 일 실시 예에 따른 SFC 시스템에 포함된 PDP 관리 모듈의 블록도이다.
- [0039] 도 5를 참조하면, PDP 관리 모듈(30)은 수신부(310), 가공부(320) 및 구현부(330)를 포함한다.

- [0040] 수신부(310)는 사용자나 SFC 관리 모듈로부터 제1 SFC 엔티티 정보를 수신한다. 제1 엔티티 정보는 SFC 환경의 데이터 평면을 구성하는 SFC 엔티티들 중에서 사용자가 PDP 상에 구현하고자 하는 SFC 엔티티들에 대한 정보를 의미한다. 수신부(310)는 제1 SFC 엔티티 정보 중에서 SF에 대한 정보는 사용자로부터 수신할 수 있으며, SFC 식별자, SFF 및 SFC 프록시에 대한 정보는 SFC 관리 모듈로부터 수신할 수 있으나, 이에 한정되는 것은 아니다.
- [0041] 가공부(320)는 SFC 엔티티들이 PDP 상에 구현되고 나서 SFC 순서에 따라 패킷을 처리할 수 있도록 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성한다.
- [0042] SFC 엔티티 정보의 가공 과정을 설명하기 위한 도면인 도 6을 참조하면, 가공부(320)는 각 SFC 엔티티들의 고유한 매치/동작 필드를 유지하면서도, SFC 특성을 고려할 수 있도록 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보를 생성한다. 이를 위하여, 가공부(320)는 아래와 같이 각 SFC 엔티티들에 대한 정보인 SFC 엔티티 정보를 가공한다.
- [0044] 1. SF에 대한 SFC 엔티티 정보의 가공
- [0045] 기존의 SF는 SFC 특성을 고려하지 않으므로, PDP 상에 그대로 구현될 경우 구현된 노드에 들어오는 모든 패킷을 SF가 구현된 순서에 따라 처리하는 문제가 있다. 따라서, SFC 순서를 만족하지 못하는 문제점이 있다. 이를 해결하기 위하여, 가공부(320)는 SF가 기 설정된 SFC 순서에 따라 패킷을 처리하도록 하기 위해 SF에 대한 제1 엔티티 정보의 매치 필드에 SPI와 SI를 포함하는 NSH를 추가한다. 따라서, 가공된 엔티티 정보에 따라 구현되는 SF는 SF의 특성뿐만 아니라, SFC 순서에 대한 정보인 SPI가 포함된 NSH를 고려하여 패킷을 처리할 수 있다.
- [0046] 또한, PDP 상에 SF와 SFC 프록시가 동시에 구현될 경우 SF 프록시에 의해 NSH가 추출된 패킷을 SF가 드롭(drop)시키지 않도록 하기 위해 SF에 대한 제1 엔티티 정보의 매치 필드 및 동작 필드에 각각 SIP(source IP)/DIP(destination IP)와 통과(pass) 동작을 추가할 수도 있다.
- [0048] 2. SFF에 대한 SFC 엔티티 정보의 가공
- [0049] 상술한 바와 같이 SF가 구현된 순서에 따라 패킷이 처리되는 문제가 있으므로, 같은 노드에 SF 엔티티들이 구현되더라도 구현된 순서와 SFC 순서가 맞지 않으면 SFC 순서대로 패킷을 처리하기 어렵다. 이를 해결하기 위해, 가공부(320)는 SFF에 대한 제1 엔티티 정보의 동작 필드에 루프백(loopback) 동작을 추가한다. 따라서, 패킷은 말단의 SFC 엔티티를 통과해도 다시 첫단의 SFC 엔티티로 되돌아갈 수 있으므로 SFC 순서에 따라 패킷을 전달할 수 있다.
- [0051] 3. SFC 식별자에 대한 SFC 엔티티 정보의 가공
- [0052] PDP 상에 SFC 식별자가 SF와 동시에 구현될 경우, 가공부(320)는 SFF의 루프백 동작에 의해 되돌아온 패킷을 SFC 식별자가 드롭시키지 않도록 매치 필드에 SPI를 추가한다.
- [0053] 또한, PDP 상에 SFC 식별자가 SFC 프록시와 동시에 구현될 경우, 가공부(320)는 SFC 프록시에서 NSH가 추출된 패킷을 SFC 식별자가 드롭시키지 않도록 매치 필드 및 동작 필드에 각각 SIP/DIP와 통과(pass) 동작을 추가한다.
- [0055] 4. SFC 프록시에 대한 SFC 엔티티 정보의 가공
- [0056] 가공부(320)는 SFC aware SF를 위한 패킷을 SFC 프록시가 드롭시키지 않도록 NSH가 정의된 매치 필드와 쌍을 이루는 동작 필드에 통과 동작을 추가한다.
- [0058] 상술한 바에 따라 각 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 가공하여 제2 SFC 엔티티 정보(SF_SFC, Classifier_SFC, SFF_SFC, Proxy_SFC)를 도 6과 같이 생성한다.
- [0059] 구현부(330)는 패킷으로부터 헤더를 추출하기 위한 파서(parser)와, 패킷으로부터 추출된 헤더를 조합하기 위한 디파서(deparser)를 정의하고, 파서와 디파서를 SFC 엔티티들과 함께 PDP 상에 구현할 수 있다. 또한, 구현부(330)는 파서를 정의할 때 파서가 패킷으로부터 헤더를 추출하기 위해 필요한 파싱 그래프(parsing graph)를 생성할 수 있다.
- [0060] 파싱 그래프를 생성하는 과정을 설명하기 위한 도면인 도 7을 참조하면, 구현부(330)는 제2 SFC 엔티티 정보를 이용하여 파싱 그래프를 생성한다. 보다 구체적으로, 구현부(330)는 SFC 엔티티들 중에서 PDP 상에 어떠한 SFC 엔티티들이 구현되는지를 확인하고, PDP 상에 포함된 노드에 SFC 엔티티들이 구현되었을 때 어떠한 헤더들이 노드에 의해 사용되는지를 확인하고, PDP 상에 구현될 SFC 엔티티들에 대한 정보와 노드에 의해 사용될 헤더들에

대한 정보를 토대로 파싱 그래프를 생성한다. 생성된 파싱 그래프에서, ◇는 SFC 엔티티들을 의미하고, □는 헤더를 의미하고, ***은 각 SFC 엔티티들의 특성에 따라 정의되는 헤더를 의미한다. 예를 들어, SFC 엔티티들 중에서 SFC 식별자와 SFF만 PDP 상에 구현될 경우 SFC 식별자와 SFF에서 필요한 헤더와 NSH가 추출될 수 있도록 파싱 그래프를 생성한다. 여기서, 각 SFC 엔티티들에서 필요로 하는 헤더는 매치 필드와 동작 필드에 사용되는 헤더를 의미한다. 파싱 그래프가 생성되면, 파서는 파싱 그래프를 참조하여 패킷으로부터 헤더를 추출할 수 있다.

[0061] 상술한 구성들을 포함하는 PDP 관리 모듈에 의해 PDP 상에 포함된 노드에 SFC 엔티티들이 구현되면, PDP 관리 모듈은 SFC 엔티티들을 제어할 수 있는 API(Application Programming Interface)를 생성할 수 있다. PDP 관리 모듈은 생성된 API를 통해 PDP 상에 구현된 SF를 관리할 수 있으며, SFC 관리 모듈은 생성된 API를 통해 PDP 상에 SFC 룰을 삽입할 수 있다.

[0063] 도 8은 PDP 스위치 구조를 도시한 것이다.

[0064] 도 8을 참조하면, 본 발명의 일 실시 예에 따른 PDP 관리 모듈에 의해 SFC 엔티티들이 PDP 상에 포함된 노드인 스위치에 구현될 경우, 해당 스위치의 구조는 크게 패킷으로부터 헤더를 추출하는 파서, SFC 엔티티들을 포함하여 추출된 헤더에 따라 동작하는 파이프라인(pipeline) 및 추출된 헤더를 조합하는 디파서로 구성된다. 여기서, 파이프라인의 각 SFC 엔티티들의 매치 필드와 동작 필드의 타입은 PDP 관리 모듈에서 정의하고, 매치 필드의 값(key)과 동작 필드의 동작은 SFC 관리 모듈과 PDP 관리 모듈에서 정의한다. 예를 들어, 매치 필드가 NSH고 동작이 드롭(drop)이라는 것은 PDP 관리 모듈에서 정의하고, 특정 NSH 값을 갖는 패킷을 드롭해야되는지를 결정하는 매치-동작 필드 쌍은 SFC 관리 모듈과 PDP 관리 모듈에서 API를 통해 정의할 수 있다.

[0065] 상술한 구조를 갖는 스위치에 들어온 패킷은 우선 파서를 통해 필요한 헤더들이 추출되고, 파이프라인에 의한 구현된 순서에 따라 처리된다. 예를 들어, 도 8의 경우 패킷은 SFC 식별자 → SFC 프록시 → SF → SFF 순으로 처리된다. 한편, 처리되는 순서는 이에 한정되지 아니하고 가상화 기법마다 상이할 수 있다.

[0067] 실시예

[0068] 도 9는 본 발명의 일 실시 예에 따른 SFC 시스템이 적용된 PDP를 도시한 것이다.

[0069] 도 9를 참조하면, 본 실시예에서는 방화벽과 로드 밸런서(Load Balancer)를 순차적으로 지나야 하는 파란색 패킷과, SF1(DPI 인스턴스)을 지나고 SFF2를 거쳐 다른 SF들을 순차적으로 지나가는 빨간 패킷이 존재한다. 본 실시예의 PDP에는 SFC 식별자 기능, 방화벽 기능, 로드 밸런서 기능 및 SFF 기능이 구현되어 있으며, SFC 관리 모듈과 PDP 관리 모듈로부터 매치와 동작 테이블의 필드 값들이 정의된다. 파란 패킷이 스위치에 들어오면 파서를 통해 헤더를 추출하고, SFC 식별자를 통해 NSH(SPI 1, SI 255)를 추가한다. 그 후에 구현된 방화벽 SF를 지나게 된다. 이때, SPI가 1이고, SI가 255이기 때문에 별다른 조치 없이 구현된 로드 밸런서 SF로 전달된다. 로드 밸런서 SF에서는 SPI/SI/port 조합을 통해 해당 패킷의 드롭 여부를 결정한다. 만일 port 번호가 80번(HTTP packet)이라면 SI를 1 줄인다. 그 다음, SFF에서는 SPI가 1이고, SI가 2이기 때문에 루프백 동작을 수행한다. 패킷은 다시 SF 식별자를 통과하고, 방화벽 SF에서 처리를 받은 후에 SI를 1 줄인다. 그런 다음, 로드 밸런서 SF를 통과하고, SFF에서는 SPI와 SI에 맞게 NSH를 제거하고 전달한다 (모든 SFC가 처리된 경우를 의미).

[0070] 한편, 빨간 패킷은 SFC 식별자를 통해 패킷에 NSH를 추가하고 SFF를 통과한 후, SF1으로 전송된다. SF1에 의해 패킷이 처리된 후, SI 값이 1이 줄여진 상태로 PDP에 다시 전송된다. PDP에 다시 들어온 빨간색 패킷은 SFF까지 통과되어 SFF2로 포워딩 된다.

[0072] 도 10은 본 발명의 일 실시 예에 따른 SFC 방법의 순서도이다. 이하에서는 앞서 설명한 부분과 중복되는 부분에 대한 상세한 설명은 생략하기로 한다.

[0073] 도 10을 참조하면, 본 발명의 일 실시 예에 따른 서비스 기능 체인 시스템에 의해 수행되는 서비스 기능 체인 방법은 SFC 엔티티 정보 수신 단계(S100), SFC 엔티티 정보 가공 단계(S200), 파서, 디파서 정의 단계(S300) 및 SFC 엔티티 구현 단계(S400)를 포함한다.

[0074] S100 단계는 PDP 상에 구현할 SFC 엔티티들에 대한 정보인 제1 SFC 엔티티 정보를 사용자나 SFC 관리 모듈로부터 수신하는 단계이다.

[0075] S200 단계는 S100 단계를 통해 수신한 제1 SFC 엔티티 정보를 가공한다. 보다 구체적으로, SFC 엔티티들이 SFC 순서에 따라 패킷을 처리할 수 있도록 제1 SFC 엔티티 정보에 포함된 각각의 SFC 엔티티들에 대한 매치/동작 필

드 데이터를 가공하여 제2 SFC 엔티티 정보를 생성한다. 제1 SFC 엔티티 정보의 가공 방법은 상술한 바와 같다.

[0076] S300 단계는 패킷으로부터 헤더를 추출하기 위한 파서와, 패킷으로부터 추출된 헤더를 조합하기 위한 디파서를 정의한다. 또한, S300 단계는 S200 단계를 통해 생성된 제2 SFC 엔티티 정보를 이용하여 파서가 패킷으로부터 헤더를 추출하기 위해 필요한 파싱 그래프를 생성하는 단계를 더 포함할 수 있다.

[0077] S400 단계는 S200 단계 내지 S400 단계를 통해 정의 및 생성된 파서, 제2 SFC 엔티티 정보 및 디파서를 이용하여 SFC 엔티티들을 PDP 상에 구현한다.

[0078] 상술한 바와 같은 본 발명의 일 실시 예에 따른 SFC 시스템 및 방법은 기존의 SFC 시스템과 비교할 때 보다 낮은 SFC 처리시간을 보장할 수 있다. 기존 SFC와 SFC 처리시간을 비교하기 위한 수학적식은 아래의 수학적식 1,2와 같다.

수학적식 1

$$D_{compared} = \sum_{j \in J} c_{i,j} (P_0 + T_1 + \sum_{j' \in J} c_{1,j'} T_{h_{j,j'}}) + \sum_{i \in I} \left[P_i + 2T_1 + \sum_{j \in J} (c_{i,j} S_j + \sum_{j' \in J, j' \neq j} c_{i+1,j'} (S_j + T_{h_{j,j'}})) \right]$$

[0079]

수학적식 2

$$D_{proposed} = \sum_{j \in J} \sum_{i=0, i \in I} \left[e_{i,j} (1 - e_{i+1,j}) (S_j E(Loop) + \sum_{j' \in J, j' \neq j} e_{i+1,j'} T_{h_{j,j'}}) \right]$$

[0080]

[0081] 각 수학적식에서 사용되는 파라미터와 변수는 다음과 같다. SF들의 집합은 I로, 각 SF는 원소 i로 정의한다. 단, 수식을 간단히 하기 위해 i가 0인 경우는 SFC 식별자 기능으로 정의한다. SFF 들의 집합은 J로, 각 SFF는 원소 j로 정의한다. $c_{i,j}$ 는 식별자(즉, i가 0인 경우) 또는 SF i가 SFF j에 일 홉으로 연결되어 있는지 아닌지를 나타내는 변수이다. 해당 변수가 1이면 연결이 되어있고, 0이면 연결되어 있지 않다. SFC 식별자나 SF들은 하나의 SFF에만 1홉으로 연결되고, SFF는 다수의 SF와 SFC 식별자가 연결될 수 있다. 이와 유사하게, $e_{i,j}$ 는 SFC 식별자(즉, i가 0인 경우) 또는 SF i가 SFF j에 구현이 되었는지 아닌지를 나타내는 변수이다. 해당 변수가 1이면 구현된 상태이고, 0이면 구현이 되어 있지 않다. SFC 식별자나 SF들은 하나의 SFF에만 구현이 되고, SFF는 다수의 SF와 SFC 식별자를 구현할 수 있다. $h_{i,j}$ 는 SFF j와 SFF j' 사이의 홉 수를 나타내는 변수이다. P_i , S_j , T_n 는 각각 SF i 패킷 처리시간과 SFF j 스위칭 시간 그리고 n홉의 전송시간을 의미한다. P_0 는 SFC 식별자의 패킷 처리시간을 나타낸다.

[0082] 기존 연구와 본 발명의 성능 비교를 위해, SFF는 두 개만 존재하고, SFC는 집합 I의 원소들이 순차적으로 정의되고, 그 외에 나머지 변수들은 랜덤으로 정의된 환경에서, SF수를 늘려가면서(즉, FC의 정의된 SF수를 늘려가면서) 10000번 동안의 SFC 최소, 평균, 그리고 최대 처리시간을 비교하였다.

[0084] 도 11은 본 발명과 기존의 SFC 간에 성능을 비교하기 위한 도면이다.

[0085] 도 11을 참조하면, 시뮬레이션 결과를 통해 SF가 늘어날수록(SFC내의 SF가 늘어날수록) SFC 처리되는 시간이 증가하는 것을 알 수 있다. 이는 SF가 늘어날수록 패킷이 처리되는 홉수와 전송하는 홉수가 늘어나기 때문이다. 하지만, 본 발명의 경우 비교 기법들보다 낮은 SFC 처리 시간의 증가 폭을 나타낸다. 이는 SF가 늘어나더라도, SFF의 구현이 SFC의 순서에 맞게 잘 되어 있으면 스위처에서 패킷을 처리할 때 다수의 SF를 한 번에 처리할 수 있기 때문이다. 이러한 이유로 본 발명의 최소 SFC 처리시간은 항상 일정하다. 반면에, 기존의 SFC 방법은 사용자가 요구하는 체인에 따라 트래픽 경로가 결정되기 때문에, SF가 증가하게 되면 최소, 최대, 평균 처리시간이 모두 증가하게 된다. 따라서, 본 발명을 통해 낮은 처리시간을 보장 할 수 있다.

[0086] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨

터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0087] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

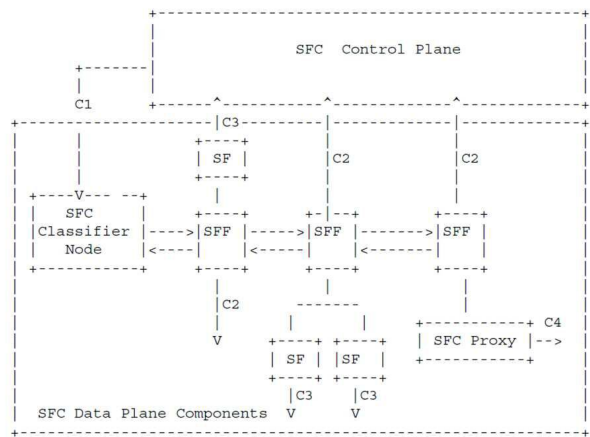
[0088] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0089] 이상과 같이 실시예들이 비록 한정된 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

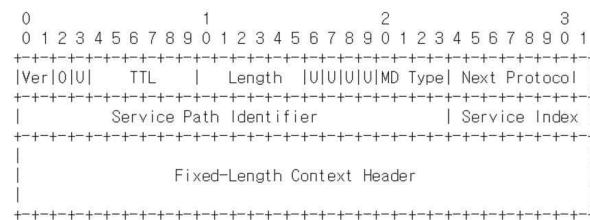
[0090] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

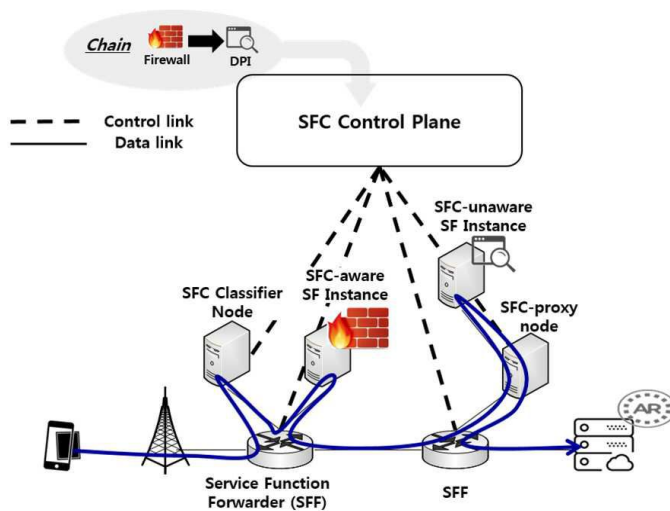
도면1



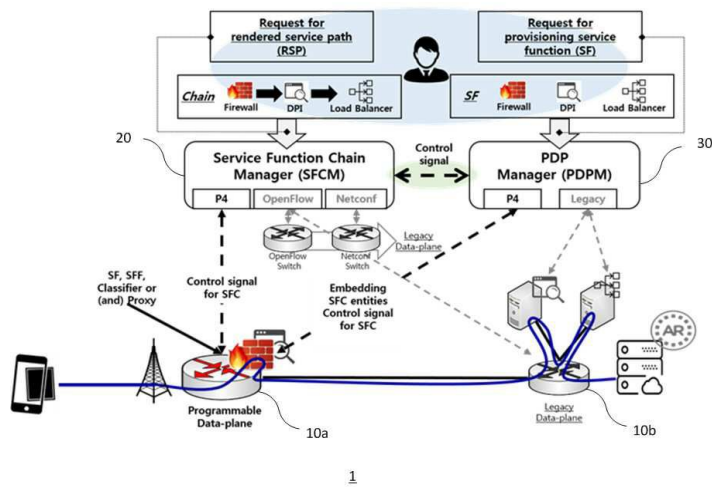
도면2



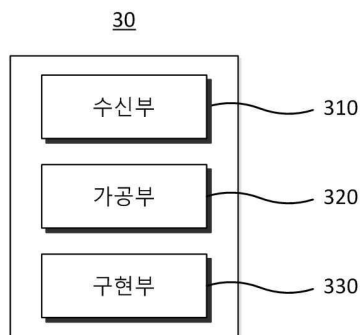
도면3



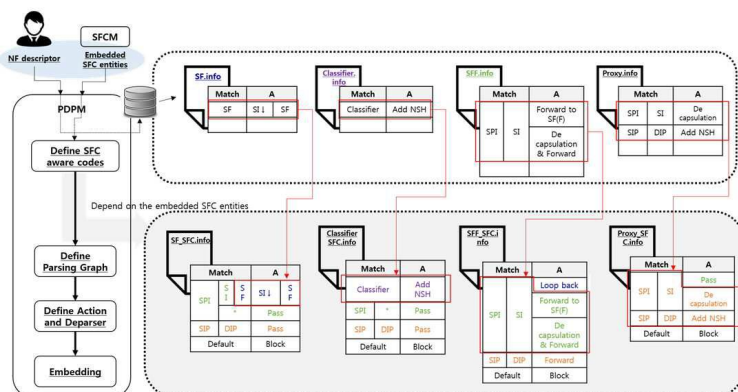
도면4



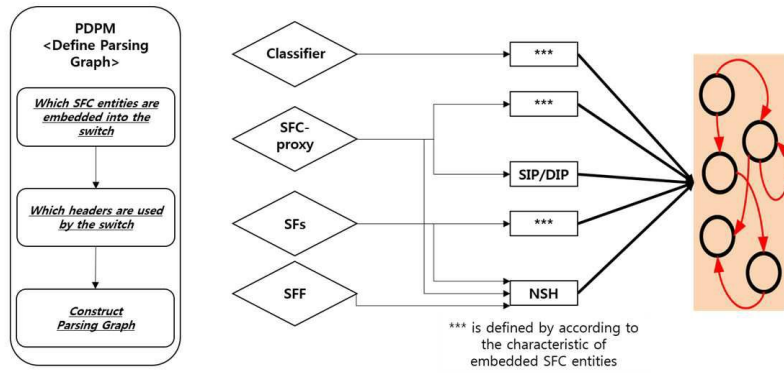
도면5



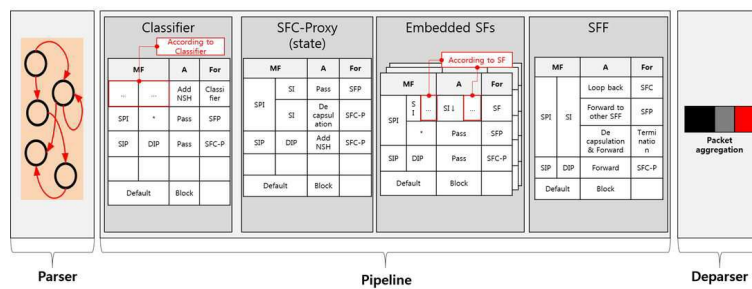
도면6



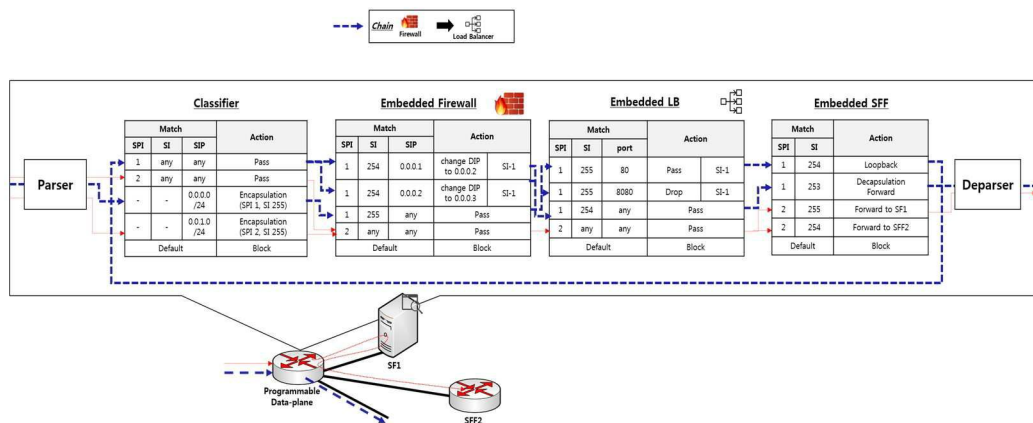
도면7



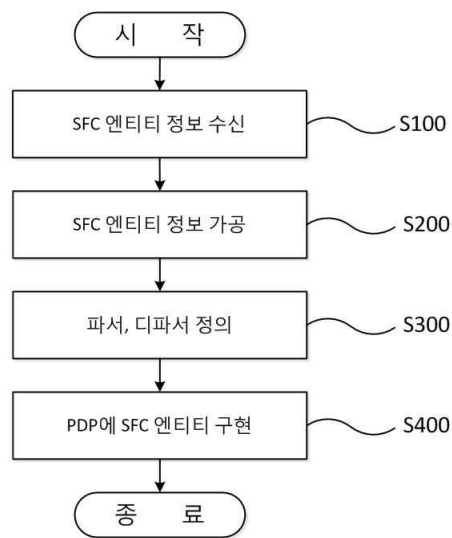
도면8



도면9



도면10



도면11

