K-ONE 기술 문서 #31

ONOS 상 LISP Mobility Management 개발 방안

Document No. K-ONE #31
Version 1.0
Date 2018-02-28
Author(s) 최준묵, 홍지범

K-ONE

ONK
OpenNetworkingKorea

■ 문서의 연혁

버전	날짜	작성자	내용
초안 - 0.1	2018. 01. 28	최준묵	초안 작성
0.5	2018. 02. 05	최준묵	시스템 디자인 추가
0.7	2018. 02. 07	홍지범	서론, 개요 추가
0.9	2018. 02. 10	홍지범	오탈자 수정
1.0	2018. 02. 28	최준묵	오탈자 수정

본 문서는 2017년도 정부(미래창조과학부)의 재원으로 정보통신 기술진흥센터의 지원을 받아 수행된 연구임 (No. B0190-15-2012, 글로벌 SDN/NFV 공개소프트웨어 핵심 모듈/기능 개발)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0190-15-2012, Global SDN/NFV OpenSource Software Core Module/Function Development)

기술문서 요약

현재 인터넷 구조가 직면하고 있는 가장 큰 문제는 라우팅 테이블의 크기가 급격하게 증가하는 것으로, 이러한 테이블 크기의 증가는 라우터 하드웨어 규모와 투자 비용의 증가를 초래한다. 이 외에도 도메인 간 트래픽 엔지니어링 등의 문제가 부각되고 있는데 이러한 문제들의 가장 큰 원인 중 하나는 IP주소만을 사용하는 현재 인터넷의 단일 주소 체계 때문이다. 단말 간 연결에서 식별자로 IP주소를 사용함과 동시에 네트워크 라우팅에 사용함으로써 IP주소에 부여되는 의미가 겹치게 되어 문제가 발생하게 된다.

이에 LISP(Locator Identifier Seperation Protocol)을 이용하여, 현재 IP주소를 식별 및 위치 주소로 병용함으로써 발생하는 확장성 문제를 해결하고 기존의 네트워크를 보다 낮은 비용으로 개발 및 운용하는 시도를 하고 있다. LISP은 기존 인터넷의 IP주소를 사용한 단일 주소 체계를 호스트를 구별하는 단말 식별자(EID: Endpoint Identifier)와 호스트의 위치를 나타내는 위치자(RLOC: Routing Locator)로 구분하여 터널링(Tunneling) 기법을 통해 확장성 문제를 해결하고자 하는 것으로서, LISP을 사용하면 새로운 서비스의 배포를 촉진할 수 있을 뿐만 아니라 지속적으로 증가할 것으로 예상되는 인터넷 트래픽을 비용대비 효율적으로 수용할 수 있을 것으로 기대된다.

그러나 LISP 표준안은 호스트가 이동하는 경우(handover) 이를 원활하게 지원할 수 있는 방법이 없다는 단점이 있다. 본 기술 문서에서는 LISP에서 이동성을 지원하기 위한 관련 연구를 소개하고 LISP mobility management를 위한 시스템 디자인을 제시하고자 한다.

Contents

K-ONE #31. ONOS 상 LISP Mobility Management 개발 방안

그림 목차

K-ONE #31. ONOS 상 LISP Mobility Management 개발 방안

1. 서론

최근 클라우드 기반 서비스, 스마트 디바이스의 보급, 사물 인터넷(IoT: Internet of Things) 기술의 등장으로 국내외 인터넷 트래픽이 급격하게 증가하고 있다. 게다가 이러한 기술을 활용하여 새로운 서비스들이 등장함에 따라 이 추세는 더욱 가속될 것으로 예상되기 때문에 현재 인터넷 구조의 확장성에 대한 문제가 지속적으로 제기되고 있다.

현재 인터넷 구조가 직면하고 있는 가장 큰 문제는 라우팅 테이블의 크기가 급격하게 증가하는 것으로, 이러한 테이블 크기의 증가는 라우터 하드웨어 규모와 투자 비용의 증가를 초래한다. 이 외에도 도메인 간 트래픽 엔지니어링 등의 문제가 부각되고 있는데 이러한 문제들의 가장 큰 원인 중 하나는 IP주소만을 사용하는 현재 인터넷의 단일 주소 체계 때문이다. 단말 간 연결에서 식별자로 IP주소를 사용함과 동시에 네트워크 라우팅에 사용함으로써 IP주소에 부여되는 의미가 겹치게 되어 문제가 발생하게 된다.

이에 LISP(Locator Identifier Seperation Protocol)을 이용하여, 현재 IP주소를 식별 및 위치 주소로 병용함으로써 발생하는 확장성 문제를 해결하고 기존의 네트워크를 보다 낮은 비용으로 개발 및 운용하는 시도를 하고 있다. LISP은 기존 인터넷의 IP주소를 사용한 단일 주소 체계를 호스트를 구별하는 단말 식별자(EID: Endpoint Identifier)와 호스트의 위치를 나타내는 위치자(RLOC: Routing Locator)로 구분하여 터널링(Tunneling) 기법을 통해 확장성 문제를 해결하고자 하는 것으로서, LISP을 사용하면 새로운 서비스의 배포를 촉진할 수 있을 뿐만 아니라 지속적으로 증가할 것으로 예상되는 인터넷 트래픽을 비용대비 효율적으로 수용할 수 있을 것으로 기대된다.

본 기술 문서에서는 LISP의 mobility management 지원을 위한 관련 연구들을 소개하고 ONOS-LISP subsystem을 사용하여 NAT 환경에서의 mobility를 지원하는 시나리오 구현 및 개선된 mobility management 시스템 디자인을 제안하고자 한다.

2. LISP 프로토콜

본 절에서는 LISP(Locator Identifier Separation Protocol)[1]에서 사용되는 용어, 구조와 동작 순서에 대해 설명한다.

2.1. LISP 개요

LISP은 IP 터널링에 기반하여 위치자-식별자 분리 패러다임을 구현한 프로토콜의 일종이다. 위치자-식별자 분리 패러다임은 기존의 라우팅 체계가 가지는 확장성 등과 같은 한계를 극복하기 위해 제안된 네트워크 패러다임이며, LISP은 IP 터널링을 통해 이러한 패러다임을 구현하였다. 위치자는 기존의 라우팅 체계가 IP 주소를 사용하여 패킷을 전달하듯이 호스트의 위치를 나타내는 주소이며, 식별자는 기존의 호스트가 IP 주소를 사용하여 목적지를 식별하듯이 호스트를 구별하는 주소이다. LISP은 주소 체계를 각각 역할에 따라 분리하여 사용하며 이에 대한 대응 정보를 사용함으로써 분리된 주소 체계를 연결 지어 패킷을 전달한다. LISP 장치들이 대응 정보와 관련된 처리를 하는 네트워크를 제어 평면(Control Plane)이라 하며 호스트들이 패킷을 주고 받는 네트워크는 데이터 평면(Data Plane)이라 한다. 따라서 LISP은 소프트웨어 정의 네트워크(Software-Defined Network)의 일종이라고 할 수 있다. LISP 제어 평면은 LISP 매핑 시스템과 LISP 라우터들에 의해 구성되며 LISP 데이터 평면은 LISP 라우터들과 호스트들에 의해 구성된다. LISP이 동작하는 방식, 네트워크를 제어하기 위해 사용되는 제어 패킷(Control Packet) 등은 RFC[1]를 통해 표준화 되어 있다.

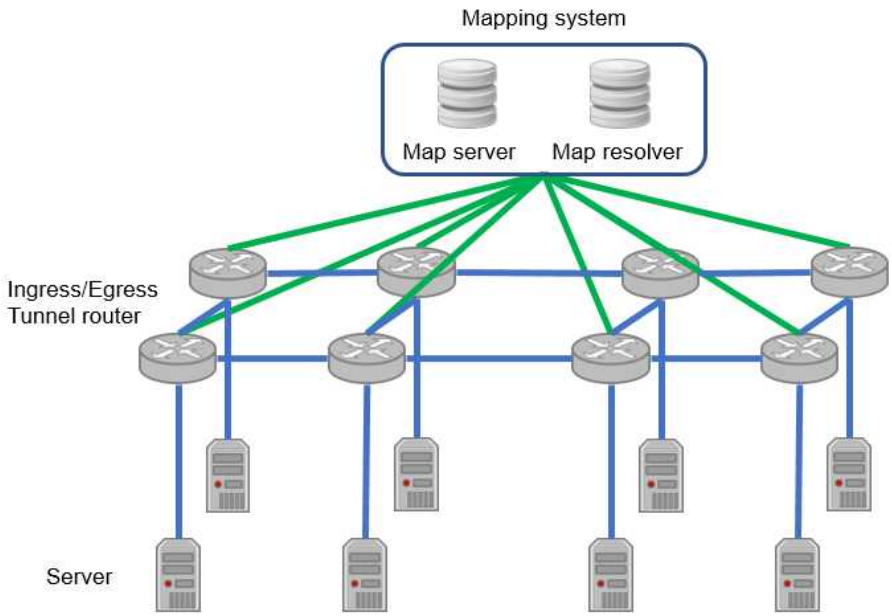

그림 1 LISP 구조도

LISP은 Layer 3 터널링을 통해 구현되며 이러한 터널링은 패킷을 IP 헤더로 전달할 패킷을 감싸서 구현한다. LISP 패킷은 외각 IP 헤더와 내부 IP 헤더를 가지게 된다. 내부 IP 헤더는 데이터를 주고 받는 호스트들의 식별자를 통해 구성되며 외곽 IP 헤더는 LISP 장치들이 데이터를 전달하기 위해 LISP 장치들의 위치자를 통해 구성된다. 또한 LISP은 UDP를 사용하여 구현되었다.

2.2. LISP 용어 정리

- **LISP site**

하나의 LISP site는 특정 EID 대역으로 관리되며 다수의 호스트와 하나 이상의 xTR로 구성된다. 같은 site 내에서는 EID로 라우팅이 가능하다.

- **EID(Endpoint ID)**

LISP site 내에서 DHCP 등과 같은 방법으로 네트워크 인터페이스에 할당 되는 주소거나 임의로 할당되는 주소이다. 이는 호스트를 식별하는데 사용되는 주소이며 어플리케이션은 이 주소를 사용하여 통신하게 된다.

- **RLOC(Routing Locator)**

LISP site 밖에서 패킷을 라우팅하는데 사용되는 주소이다. 이 주소는 xTR에 할당된 주소로 코어 네트워크(일반 네트워크) 내에서 라우팅하는 데에 사용된다.

- **MS(Map server)**

EID-RLOC 매핑 정보를 관리하는 서버로 특정 EID 대역을 관리하고 있는 xTR의 RLOC 정보를 관리한다. xTR은 주기적으로 MS로 자신의 EID 대역과 RLOC을 등록하는 메시지를 전달하며 ITR이 EID-RLOC 매핑 정보를 얻기 위한 메시지를 보낼 경우 데이터베이스에서 해당 EID 대역을 관리하는 ETR 정보를 찾아내어 해당 정보를 전달한다.

- **MR(Map resolver)**

ITR로부터 EID-RLOC 매핑 정보를 요청하는 메시지를 받으면 해당 정보를 가지고 있는 MS를 찾아 요청하는 메시지를 전달한다.

- **ITR(Ingress tunnel router)**

LISP site와 코어 네트워크의 경계에 위치하며 자신이 관리하는 LISP site 내부의 호스트로부터 패킷을 전달 받아 캡슐화 하여 목적지로 전달하는 기능을 수행한다. 만약 목적지에 대한 정보가 없을 경우 해당 패킷의 목적지를 EID로 간주하여 MR에 요청을 보내 해당 EID에 대한 RLOC을 요청한다.

- **ETR(Egress tunnel router)**

LISP site와 코어 네트워크의 경계에 위치하며 캡슐화 된 패킷을 전달 받아 외각 헤더를 제거하여 내부 패킷을 복원한 뒤 해당 목적지 EID를 가진 호스트로 라우팅 한다.

- **xTR**

ITR와 ETR의 기능은 패킷의 방향에 따라 달라지는데 이를 구분 지을 필요가 없을 때 하나로 통칭하는 용어이다. 일반적으로 ITR과 ETR은 하나의 장치로 구현된다.

- **RTR(Re-encapsulating tunnel router)**

직접 연결 될 수 없는 LISP site들을 연결하는 데에 사용된다. 예를 들어 방화벽이나 NAT 환경의 경우 LISP이 사용하는 포트들이 개방 되어 있지 않을 수 있기 때문에 이를 우회하는데 RTR이 사용된다. 즉 RTR은 LISP site들의 고정점 역할을 수행한다.

2.3. LISP 동작 순서

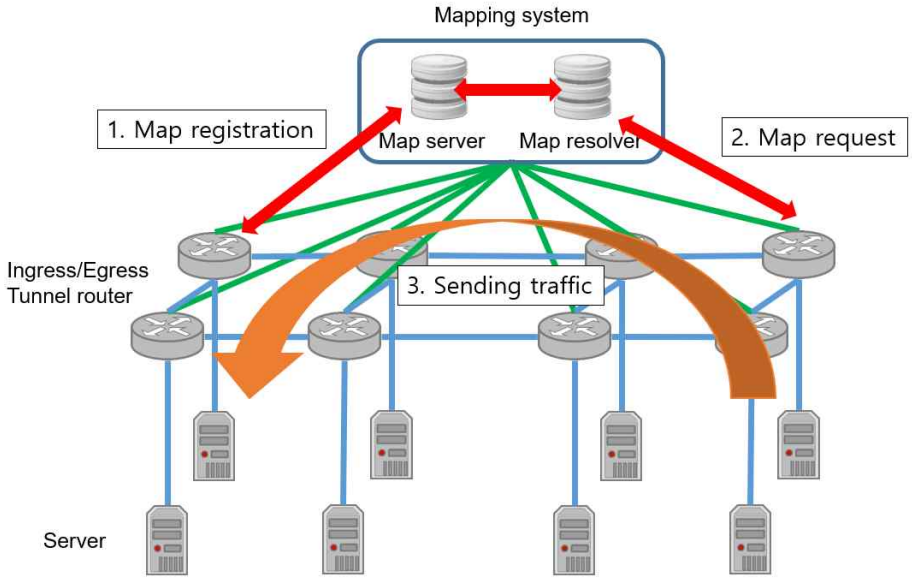

그림 2 LISP 동작 순서

- **Map registration**

xTR은 자신이 관리하는 LISP site의 EID 대역과 자신의 RLOC을 Map-register 메시지를 통해 LISP 매핑 시스템에 등록한다. 매핑 시스템은 Map-notify 메시지에 Map-register 메시지의 nonce를 포함시켜 xTR에게 응답하여 Map registration 과정이 끝났음을 알려준다.

- **Map request**

xTR은 자신에게 도착한 패킷의 목적지 주소를 EID로 간주하여 데이터베이스에 해당하는 RLOC이 없을 경우 LISP 매핑 시스템에 EID에 대한 RLOC 매핑 정보를 요청한다. 이는 Map-request 메시지를 통해 이루어지며 매핑 시스템은 Map-reply 메시지를 통해 매핑 정보를 전달한다. xTR은 이를 사용하여 패킷을 캡슐화하여 목적지 호스트가 있는 xTR까지 RLOC을 사용하여 전달한다.

- **Sending traffic**

LISP site 내에서는 캡슐화 없이 EID를 통해 라우팅 되며 코어 네트워크에서는 캡슐화를 통해 RLOC으로 라우팅 된다.

3. 관련 연구

본 절에서는 LISP에서 mobility management를 위한 관련 연구들을 소개한다.

3.1. LISP-MN

LISP-MN[2]은 호스트가 하나의 터널 라우터 기능을 수행하게 하는 MN(Mobile node)를 사용하여 mobility를 지원한다. 즉 호스트가 직접 LISP 매핑 시스템을 통해 매핑 정보를 등록하고 요청하여 자체적으로 패킷 캡슐화를 수행하여 LISP을 지원하게 된다. 이러한 방식은 비교적 구현이 간단하나 모든 호스트에 프로그램을 설치하는 등의 수정이 필요하며 아래의 그림과 같이 모바일 노드가 xTR이 관리하는 LISP site 내부에 위치하는 경우 xTR에 의한 이중 캡슐화가 일어난다는 문제가 발생한다. 또한 이 경우에는 모바일 노드가 LISP site 내에서 할당 받는 내부 아이피와 EID의 매핑, EID와 RLOC의 매핑으로 2가지 매핑 정보가 존재하기 때문에 2번의 탐색을 거쳐야 매핑 정보를 얻을 수 있다는 단점이 있다.

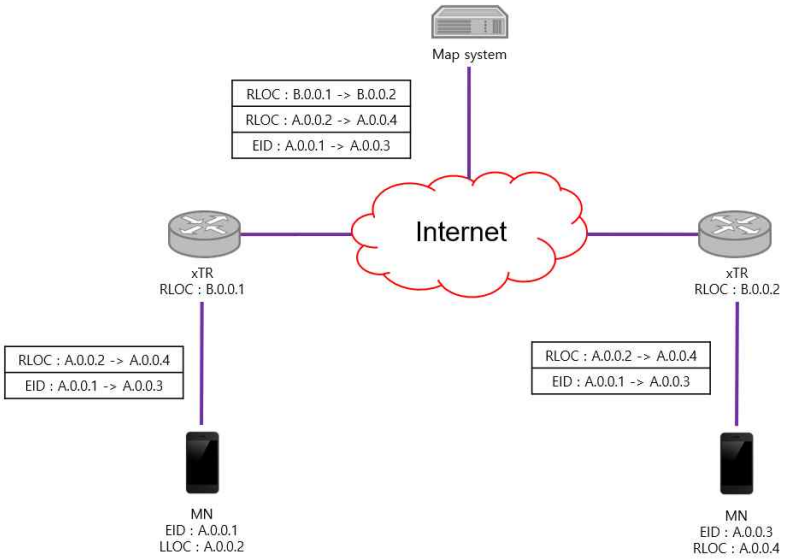

그림 3 LISP-MN의 이중 캡슐화

3.2. LISP-ROAM

LISP-ROAM[3]은 Network-based 방식의 mobility management를 지원한다. 즉 호스트에 대한 변경이 필요 없다. 이는 호스트의 아이피 주소를 항상 고정시켜 놓는 것으로 구현되는데, 가장 먼저 RADIUS 서버를 통해 유저는 자신을 인증하고 유저의 위치를 업데이트 할 수 있도록 LISP 매핑 시스템의 비밀키를 얻는다. 만약 handover가 발생하여 이미 등록한 매핑 정보에 대한 요청이 다시 오는 경우 새로운 xTR 뿐만 아니라 기존의 xTR에도 Map-notify 메시지를 보내 유저의 위치를 갱신하도록 한다.

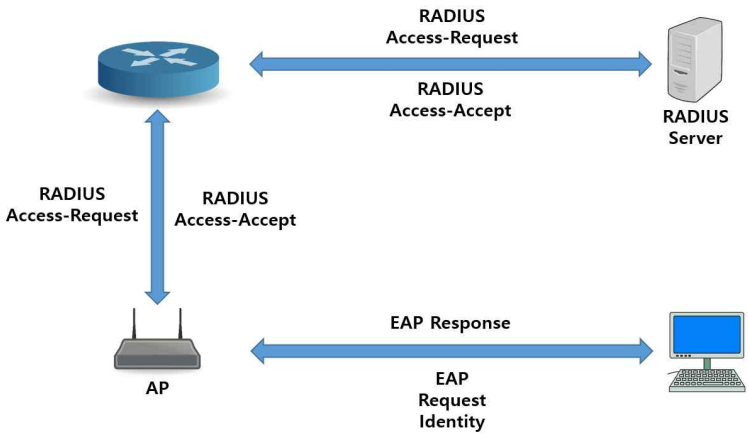

그림 4 LISP-ROAM 구조도

3.3. NEMO-LISP

NEMO-LISP[4]은 MR(Mobile router) 개념을 도입하였다. 모바일 노드는 MR을 통해 LLOC(Local locator)를 할당 받고 MR은 xTR에 연결 되어 있다. xTR은 EID-LLOC 매핑 정보와 LLOC-RLLOC 매핑 정보를 가지고 있다. 이러한 매핑 정보는 라우팅 가능 여부의 차이가 있으며 이는 매핑 시스템에서 플래그를 통해 표시된다. 모바일 노드가 패킷을 보내기 위해서는 먼저 EID를 통해 LLOC 매핑 정보를 찾아 오고 플래그를 통해 해당 주소가 라우팅이 불가능하다는 것을 알 수 있으므로 다시 LLOC을 통해 RLOC 매핑 정보를 찾아온다. 그 후 라우팅이 가능해진다.

4. ONOS-LISP Mobility management 구현

본 절에서는 LISP-MN 프로토콜과 ONOS[5]를 사용하여 기본적인 mobility management를 구현하는 방법에 대해 설명하며 구현한 방법을 테스트하는 방법에 대해 서술한다.

4.1. 시스템 디자인

기본적인 mobility management는 LISP-MN 프로토콜을 사용해 구현 할 수 있다. 그러나 LISP-MN은 NAT 환경에서 적용하기가 어렵다는 단점이 있다. 따라서 RTR(Re-encapsulating tunnel router)을 사용해 이를 보완하고자 한다. 본 시스템은 다음 그림(그림)과 같이 구현 되었다. ONOS-LISP subsystem을 사용하여 LISP 매핑 시스템을 사용하였으며 호스트들은 LISP-MN을 사용하였다. LISP-MN을 사용한 호스트들은 모바일 노드(Mobile node)라 지칭한다. 모바일 노드들은 직접 인터넷에 연결 되어 있을 수 있으며 또 다른 xTR에 연결 되어 있을 수 있다. 모바일 노드는 프로그램 설정에서 지정된 EID를 사용하며 자신에게 할당된 주소를 RLOC으로 사용한다. 따라서 모바일 노드가 NAT 환경에 속해 있는 경우 RLOC로 내부 아이피를 사용하게 되므로 NAT 환경 밖에서는 정상적으로 통신 할 수 없게 된다.

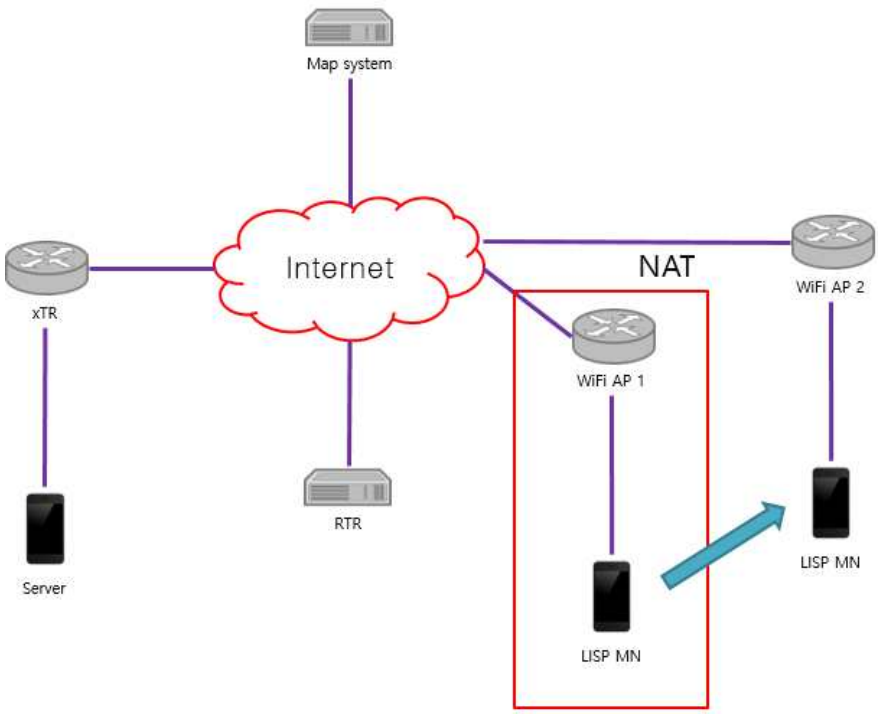

그림 5 시스템 디자인

이를 해결하기 위해서 모바일 노드는 Info-request 메시지[6]를 사용하게 된다. 프로그램 설정에서 NAT traversal 옵션을 사용하게 되면 모바일 노드는 LISP 매핑 시스템에게 Info-request 메시지를 보내게 된다. Info-request 메시지는 두 가지 목적을 가지고 있는데, 첫 번째는 자신이 실제로 NAT 환경에 있는지를 확인하기 위해서이며 두 번째는 그럴 경우 RTR을 통해 트래픽을 전송해야 하므로 RTR 주소를 찾기 위함이다. LISP 매핑 시스템은 Info-request 메시지를 받아 Info-reply 메시지로 응답한다. Info-reply 메시지는 Info-request 메시지 헤더에 있는 IP 주소를 포함한다. 모바일 노드는 이 정보를 사용하여 자신에게 할당 된 주소와 Info-reply 메시지에 포함된 IP 주소가 일치하지 않는다면 NAT에 의해 주소가 변환 되었음을 확인 할 수 있고 이를 통해 NAT 환경에 놓여 있음을 파악 할 수 있다. 또한 Info-reply 메시지에는 RTR 주소들이 포함 되어 있어 모바일 노드가 이 중 원하는 RTR로 메시지를 보낼 수 있게 된다.

RTR은 일반적인 터널 라우터와 달리 한정된 수의 제어 메시지를 처리하면 된다. 먼저 모바일 노드는 Info-reply 메시지로 RTR의 주소를 파악하고 Map registration을 시작한다. 이를 위해 Map-register 메시지를 RTR로 전달하게 되는데 Map-register 메시지의 R 비트를 셋업한다. 이 비트는 해당 메시지가 RTR에 의해 처리 되어야함을 명시하는 역할을 하게 된다. 또한 일반적인 Map-register 메시지와 달리 이 메시지는 캡슐화 되어져서 전달되게 된다. 내부 헤더에는 자신의 RLOC에서 LISP 매핑 시스템의 RLOC으로 전달하는 정보가 담기게 되지만 이는 NAT에 의해 변형되므로 이를 방지하기 위해 외각 헤더에 자신의 RLOC에서 RTR의 RLOC으로 전달하는 정보를 담는다. 그러면 이는 NAT를 거치면서 NAT의 RLOC(NAT 환경의 Global IP 주소)에서 RTR의 RLOC으로 전달하는 정보로 변형되게 된다. RTR은 이 메시지를 받아 모바일 노드가 NAT 내부에서 할당 받은 RLOC과 NAT의 RLOC을 모두 파악 할 수 있게 된다. 이러한 정보는 외부로부터 오는 제어 메시지나 데이터 트래픽을 NAT 내부의 모바일 노드로 전달하는데 사용 되게 된다. 이를 위해 RTR은 해당 메시지의 외곽과 내부 IP 헤더 정보를 저장한다. NAT를 거치게 되면 기존의 LISP이 사용하는 포트가 아니며 마찬가지로 외부에서 모바일 노드에 접근하기 위해서 사용하는 포트도 기존의 포트가 아니기 때문에 이를 기록해둔다. Map-register 메시지를 받은 RTR은 이를 다시 LISP 매핑 시스템에 전달한다. 그러나 해당 메시지를 모바일 노드가 아닌 자신이 보낸 것처럼 헤더를 고쳐서 전달한다. 이러한 과정을 거쳐 LISP 매핑 시스템은 모바일 노드가 등록하고자 하는 EID를 RTR의 RLOC으로 등록하게 되고, 모바일 노드의 EID로 접근하고자 하는 호스트들은 RTR로 트래픽을 보내게 된다.

Map-register 메시지를 받은 LISP 매핑 시스템은 Map-notify 메시지를 RTR에게 전달한다. RTR은 이 메시지를 받아 자신이 가지고 있는 정보를 통해 원래 해당 메시지를 받아야 할 모바일 노드의 주소를 찾는다. 그리고 해당 메시지를 모바일 노드가 받을 수 있도록 캡슐화 하여 전달한다. 해당 메시지의 외각 헤더에는 RTR의

RLOC에서 NAT의 RLOC으로 보내는 정보가 담기게 되고 내부 헤더에는 LISP 매핑 시스템이 모바일 호스트의 RLOC으로 보내는 정보가 담기게 된다. 해당 메시지는 NAT를 거쳐 NAT 내부의 모바일 노드로 전달 되게 되고 모바일 노드는 해당 메시지의 외각 헤더를 제거하여 Map-notify 메시지를 전달 받게 된다. 이로써 NAT 환경에서의 Map-registration 과정이 끝나게 되고 데이터 트래픽은 RTR을 거쳐 NAT 내부의 모바일 노드에게 전달되며 반대로 모바일 노드의 트래픽 또한 RTR을 거쳐 전달 되게 된다. 다음 그림(그림)은 해당 과정을 나타내는 순서도이다.

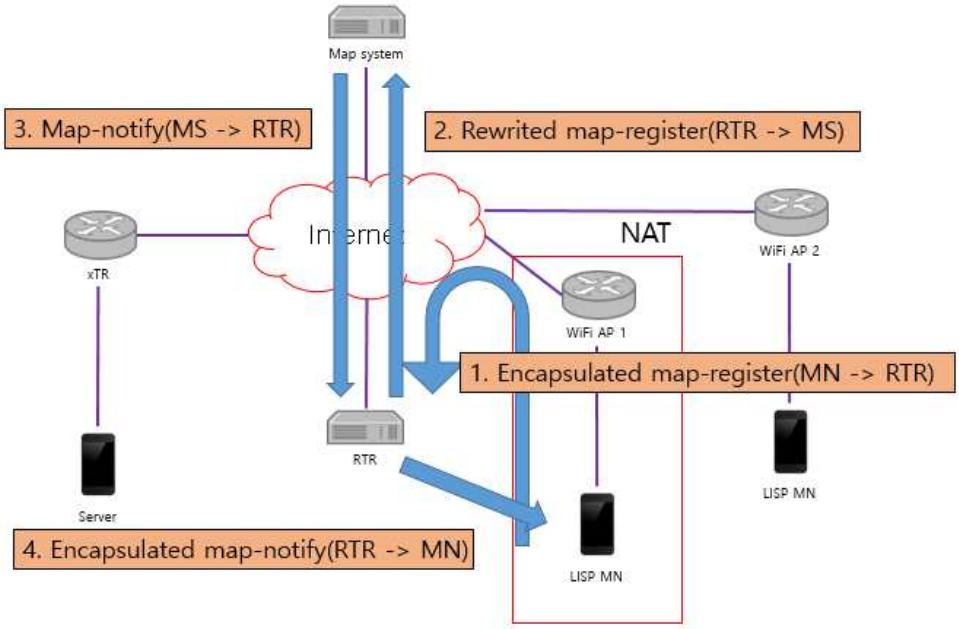

그림 6 RTR을 통한 Map registration 순서도

4.2. Re-encapsulating tunnel router 구현

Re-encapsulating tunnel router는 NAT 환경에서의 LISP을 지원하기 위해 중요한 장치이다. 본 절에서는 ONOS 어플리케이션 형태로 구현된 RTR의 구조에 대해 설명할 것이다.

RTR은 ONOS의 어플리케이션으로 동작하며 네트워크로부터 전달되는 패킷을 받기 위해 netty[7]를 사용한다. 어플리케이션이 활성화 되면 LISP의 4341, 4342 포트로부터 패킷을 받기 위해 UDP 채널을 생성하고 해당 채널의 파이프라인에 메시지 디코더와 채널인바운드 핸들러어댑터를 추가한다. 메시지 디코더는 채널에 UDP의 데이터그램 형태로 전달된 패킷을 LISP의 메시지 객체로 디코딩하는 역할을 한다.

```
public class LispPacketDecoder extends MessageToMessageDecoder<DatagramPacket> {

        private final Logger log = LoggerFactory.getLogger(getClass());

        @Override
        protected void decode(ChannelHandlerContext ctx, DatagramPacket msg, List<Object> list) throws Exception {
        if ( msg.recipient().getPort() == 4342 ) {
                // Control packet
                ByteBuf byteBuf = msg.content();
                LispMessageReader reader = LispMessageReaderFactory.getReader(byteBuf);
                LispMessage message = (LispMessage) reader.readFrom(byteBuf);
                message.configSender(msg.sender());
                list.add(message);
        }
        else {
                // Data packet
                ByteBuf content = msg.content().copy();
                LispDataPacket.DataPacketReader reader = new LispDataPacket.DataPacketReader();
                LispMessage message = reader.readFrom(msg.content(), content);
                list.add(message);
        }
    }
}
```

그림 7 LispPacketDecorder 클래스

```
public class MapcacheEntry {

        byte eidLen;
        InetAddress eidPrefix;
        InetSocketAddress sxTR_public_RLOC;
        InetAddress sxTR_private_RLOC;
        long[] sxTR_Id;
        long nonce;
        IP iph;
        UDP udh;
```

그림 8 맵캐시 항목

다음 그림(그림)은 메시지 디코더의 코드이며 ONOS-LISP subsystem의 메시지리더 팩토리 클래스를 활용하여 각 메시지에 맞는 리더를 호출하여 LISP 메시지 객체를 생성한다. 그러나 ONOS-LISP subsystem은 LISP 매핑 시스템을 기본으로 하여 제어 메시지 외에는 처리하는 루틴이 없다. 따라서 일관성을 위해 데이터 패킷들도 LISP 메시지를 상속 받아 LISP 메시지 객체의 일종으로 생성하여 디코딩한다. 이러한 과정을 거쳐 패킷을 LISP 메시지 객체로 만든 뒤 파이프라인에 메시지 리스트 형태로 출력한다.

Netty를 통해 RTR로 전달된 패킷을 모두 LISP 메시지 객체로 바꾼 뒤, LispChannelHandler라는 채널인바운드 핸들러어댑터로 전달된다. 이 클래스에서 메시지의 타입에 따라 처리하게 된다. 다음 그림(그림)은 LispChannelHandler 클래스를 나타내며 해당 핸들러는 디코더가 생성한 메시지의 리스트를 전달 받아 처리하게 된다. 메시지의 타입을 보고 제어 메시지인지 데이터 메시지인지 확인 후 각각 타입에 맞는 핸들러 객체에 전달하여 결과물을 전달 받고 이를 netty를 통해 다른

디바이스에게 전달하는 형태로 동작한다. 현재 데이터 메시지는 ONOS-LISP subsystem에 포함 되어 있지 않기 때문에 해당 메시지 타입은 null로 지정하여 구현하였다.
RTR은 Map-reply 메시지를 받게 되는 경우가 있다. 해당 메시지는 모바일 노드가 아니라 RTR이 매핑 정보를 찾기 위해 Map-request 메시지를 보낸 것에 대한 응답으로, 해당 메시지로부터 EID-RLOC 매핑 정보를 추출하여 맵캐시에 추가한다. 그리고 어떤 모바일 노드로부터 데이터 패킷을 전달하기 위해 Map-request 메시지를 보냈으므로 RTR에는 아직 전달하지 못한 패킷이 버퍼링 되어 있다. 따라서 저장된 패킷들을 확인하여 추가된 매핑 정보로 전달 할 수 있는 패킷들을 모두 전달한다.

Map-notify 메시지를 받게 되는 경우, 이는 모바일 노드가 보낸 Map-register에 대한 notify 메시지이다. 따라서 이를 다시 모바일 노드에게 전달해주어야 하는데 이를 위해 ECM으로 캡슐화 하여 보낸다. Map-notify 메시지는 Map-register 메시지의 nonce를 담아 전달되므로 이를 사용하여 맵캐시에서 전달할 모바일 노드의 주소 정보를 찾아 ECM을 생성한다. 이를 간단하게 구현하기 위해 Map-register 메시지의 IP 헤더 정보를 저장하고 있다가 주소를 덮어 쓰고 체크섬을 새로 생성하여 ECM의 IP 헤더로 사용하였다.

데이터 메시지의 경우 비교적 처리가 간단하다. 데이터 메시지를 받았을 때 해당 메시지 목적지에 해당하는 주소 정보가 맵캐시에 있는지 확인한다. 만약 존재한다면 해당 주소 정보를 사용하여 데이터 메시지를 netty를 통해 바로 전달하면 된다. 그렇지 않은 경우 주소 정보를 LISP 매핑 시스템으로부터 얻어 와야 한다. 이는 RTR이 Map-request 메시지를 매핑 시스템에 보냄으로써 얻을 수 있다. 해당 매핑 정보가 매핑 시스템으로부터 도착 할 때까지는 RTR의 버퍼에 패킷을 저장해놓는다. 버퍼는 ArrayList로 구현 되었으며 Map-reply 메시지가 도착하게 되면 처리 할 수 있는 패킷들은 제어 메시지 핸들러 쪽에서 전달하게 된다.

4.3. 시나리오 구현

● 프로그램 설정

먼저 NAT 환경에서의 mobility management를 위해 ONOS-LISP subsystem을 수정하고 RTR 어플리케이션을 설치 해야 한다. ONOS-LISP subsystem은 현재 RTR의 주소를 명시하는 방법이 없으며 이는 하드 코딩으로 수정 해야 한다. 또한 LISP 제어 평면이나 데이터 평면을 어플리케이션이 직접 접근 할 수는 없다. 따라서 RTR은 netty를 직접 사용하여 LISP 제어 평면과 데이터 평면으로부터 패킷을 받아들인다. 그러나 ONOS-LISP subsystem 또한 해당 채널로부터 패킷을 받아 들이려고 하기 때문에 RTR이 구동되는 ONOS에서 LISP 채널을 생성하지 못 하도록 수정해야 한다.

```
    // TODO: need to specify RTR addresses
        IpAddress addrip = IpAddress.valueOf("192.168.36.137");
        LispIpAddress addrv4 = new LispIpv4Address(addrip);
        ArrayList<LispAfiAddress> addrlist = new ArrayList<LispAfiAddress>();
        addrlist.add(addrv4);
        natBuilder.withRtrRlocAddresses(addrlist);

} catch (UnknownHostException e) {
    log.warn(FAILED_TO_FORMULATE_NAT_MSG, e);
}
```

그림 9 Info-request 메시지 속의 RTR 주소 하드 코딩

- **프로그램 실행**

테스트 시나리오를 위해 ONOS를 구동하는 VM 2개, xTR을 구동하는 VM 1개, 호스트 역할을 하는 VM 1개, LISP-MN을 사용하는 안드로이드 머신 1개를 사용하였다. ONOS는 RTR 주소를 하드 코딩한 매핑 시스템으로 사용할 ONOS 1개와 LISP subsystem을 비활성화하여 RTR 어플리케이션을 동작시키는 ONOS 1개를 사용하였다. xTR은 Openoverlay router를 사용하였으며 해당 xTR에서 nat-traversal 옵션을 주어 동작시켰다.

- **테스트 결과**

테스트 환경은 다음과 같다. 안드로이드 폰을 LTE 네트워크를 사용하는 휴대폰의 테더링을 사용하여 NAT 환경을 구성하였다. 테더링을 할 경우 휴대폰을 AP로 172.x.x.x 대역을 할당 받아 동작하게 된다. 그리고 와이파이로 이동 할 때 공유기를 사용하여 NAT 환경에서 NAT 환경으로 Vertical handover가 일어나는 시나리오를 구성하였다. xTR과 호스트 VM 또한 공유기로 하나의 NAT 환경을 구성하였고 호스트 VM에서 안드로이드 폰으로 핑을 보내면서 안드로이드 폰을 테더링에서 공유기로 네트워크 연결을 변경하는 형태로 Vertical handover를 테스트 하였다. LISP은 UDP 연결을 사용하여 핑 테스트는 손실된 패킷에 대한 재전송을 하지 않으므로 handover이 일어나는 딜레이 동안 패킷이 잠깐 손실 된 후 동일한 EID에 대해 다시 핑 테스트가 성공하는 것을 확인 할 수 있다.

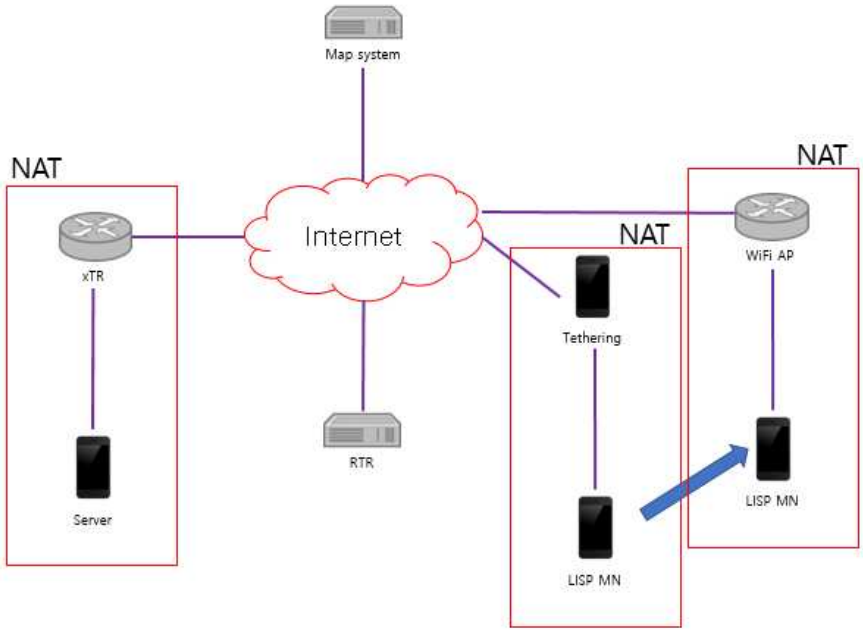

그림 10 테스트 구성도

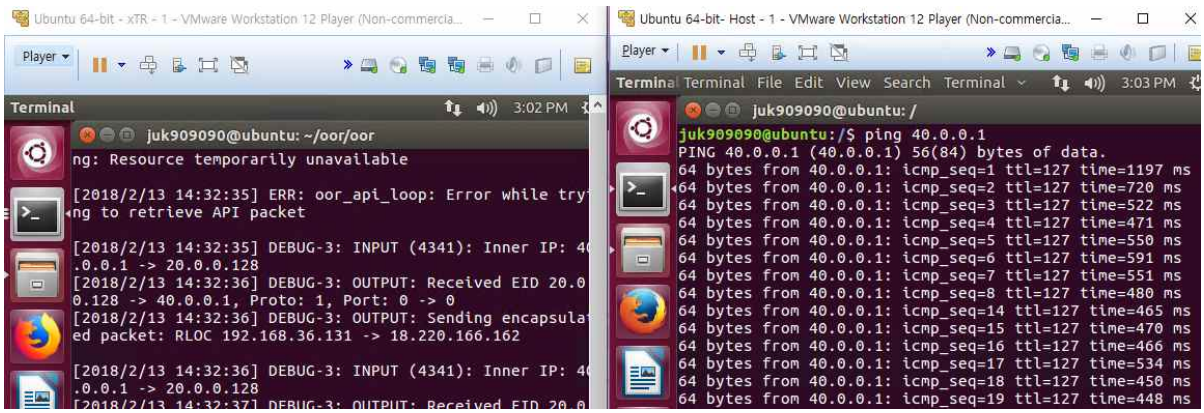

그림 11 핑 테스트 도중 handover 발생 결과

5. 개선된 ONOS-LISP Mobility management 제안

본 절에서는 4절에서 구현한 Mobility management 방식을 좀 더 개선하기 위해 몇 가지의 메시지를 추가하고 처리 절차를 추가하여 기존의 방식보다 좀 더 효율적인 방법을 제안하고자 한다.

5.1. 시스템 디자인

다음 그림은 제안하는 시스템의 디자인을 나타낸다. 기존의 방식에선 RTR의 위치를 변경 할 수 없기 때문에 handover가 일어났을 때 RTR의 위치가 고정되어 triangle 라우팅 문제가 발생 할 수 있다는 단점이 있다. 또한 RTR을 선택하는 기준이 없었다. 이를 해결하기 위해 RTR의 주소를 선택하는 기준을 추가하였고 handover이 발생했을 때 매핑 시스템이 이를 파악하여 RTR의 위치를 변경 할 수 있도록 하는 메시지를 추가하였다. 또한 기존의 LISP-MN은 터널 라우터의 기능을 수행하여 때에 따라서는 이중 캡슐화가 일어난다는 문제가 있어 LISP-MN의 기능 전부가 아니라 EID를 고정시켜 handover이 일어나더라도 어플리케이션의 EID는 변경 되지 않고 RLOC만 변경 될 수 있도록 호스트의 커널을 수정한다고 가정하였다.

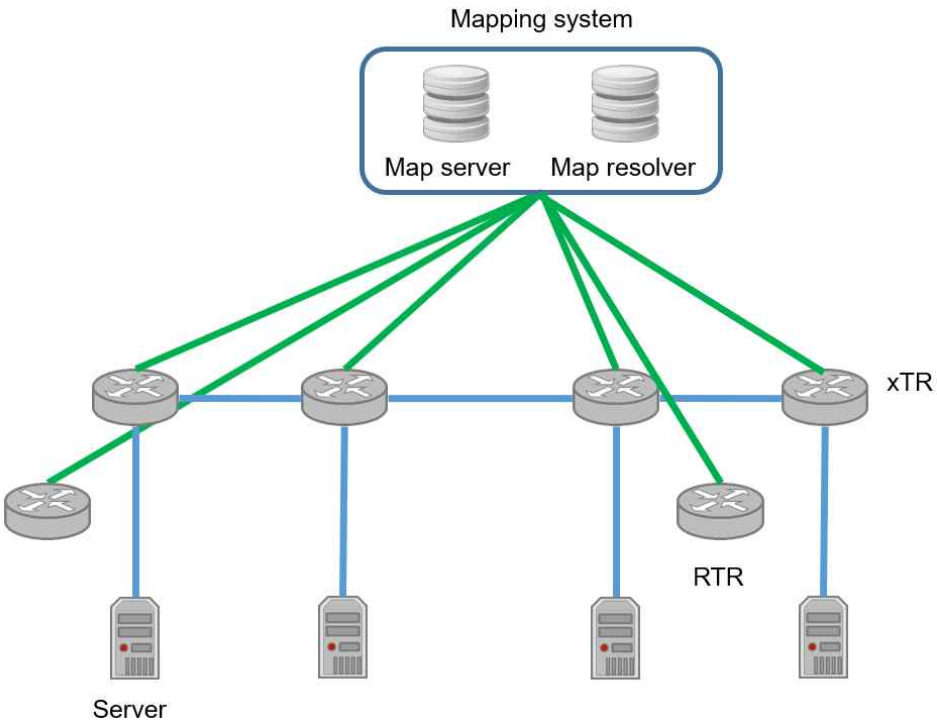

그림 12 시스템 구조도

5.2. 추가된 메시지

- Map-change

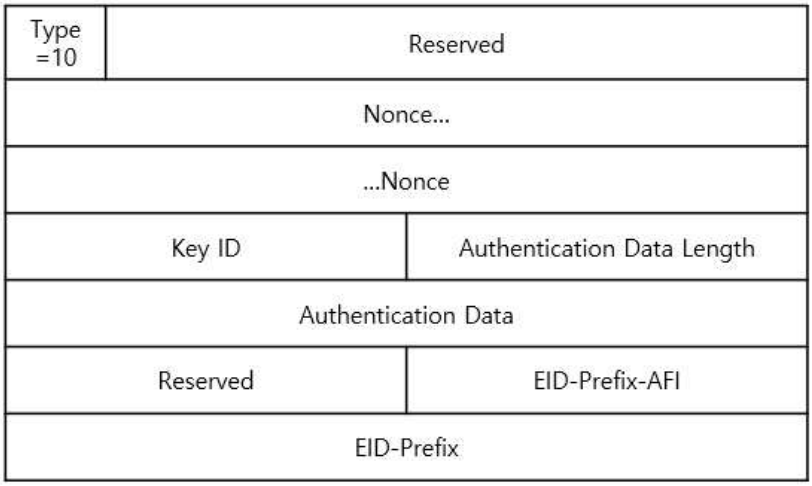

그림 13 Map-change

Handover가 일어난 경우 EID를 유지하면서 다른 xTR에 연결 될 경우 이는 기존의 EID 대역을 관리하고 있는 xTR의 RLOC과 다르므로 이를 LISP 매핑 시스템에 알려주기 위해 사용한다. 메시지에는 자신의 EID를 담아 매핑 시스템에 전달한다.

- Info-change

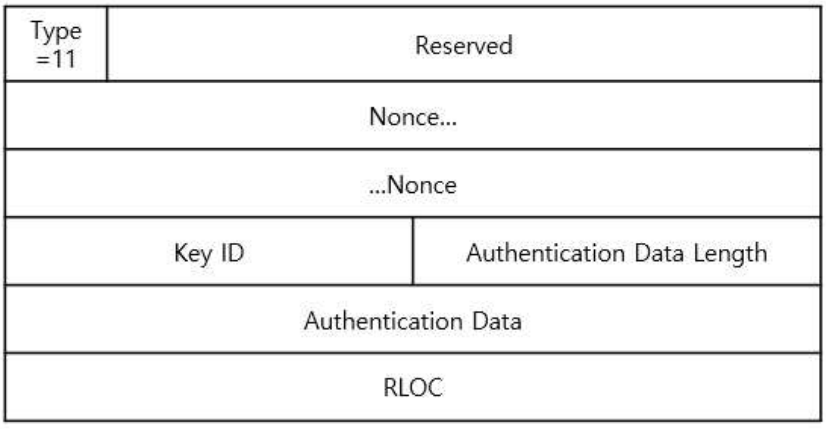

그림 14 Info-change

NAT 환경에서 handover가 일어난 경우 RTR을 통해 정상적으로 통신 할 수 있지만 RTR을 네트워크 상황에 맞춰 변경 할 수가 없기 때문에 매핑 시스템이 메시지를 보내 RTR을 변경 할 수 있도록 한다.

5.3. 추가된 프로세스

- **Handover이 일어난 경우**

VM이 새로운 xTR에 연결되는 경우 DHCP 등의 방법으로 RLOC을 새로 할당 받는다. 그러나 EID는 그대로 유지한다. xTR은 자신이 관리하지 않는 대역의 EID가 들어오면 이를 handover가 일어난 호스트로 간주하고 LISP 매핑 시스템으로 Map-change 메시지를 보낸다. 매핑 시스템은 해당 EID에 대응하는 RLOC을 새로운 xTR의 RLOC으로 변경하고 기존의 EID 대역을 관리하는 xTR에 Map-change 메시지를 보내 해당 EID에 대한 트래픽이 들어오면 이를 새로운 xTR로 포워딩하도록 한다. 이를 통해 다른 xTR의 매핑 정보가 갱신 될 때까지의 패킷 손실을 줄일 수 있다.

- **RTR을 변경 해야 하는 경우**

RTR이 변경 될 수 있는 원인에는 두 가지가 있다. 너무 많은 호스트들이 하나의 RTR을 경유하여 RTR이 과부화 되는 경우 이를 분산 시켜 줄 필요가 있으며 handover이 발생했을 때 기존의 RTR보다 네트워크 토폴로지에서 좀 더 지연 시간이 짧은 경로의 RTR을 발견 할 수 있다. 이를 위해 LISP 매핑 시스템은 Info-reply 메시지를 보낼 때 특정 RTR에 대한 응답 횟수를 체크하여 많이 응답한 RTR 주소는 우선 순위를 낮추는 형태로 부하를 분산 시킨다. 또한 Map-change를 보낸 주소가 RTR인 경우 RTR로부터 가장 네트워크 경로가 짧은 RTR을 찾아 Info-change를 보내 RTR을 바꿀 수 있도록 한다.

References

[1] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, “The locator/id separation protocol (lisp),” Internet Requests for Comments, RFC Editor, RFC 6830, January 2013

[2] A.R. Natal, L. Jakab, M. Portols, V. Ermagan, P. Natarajan, F. Maino, D. Mey er, A.C. Aparicio., 2013. “LISP-MN: mobile networking through LISP”. Wirel. Personal. Commun. 70 (1), 253–266.

[3] A. Galvani, A. Rodriguez-Natal, A. Cabellos-Aparicio., “LISP-ROAM: network-based host mobility with LISP”. 2014.

[4] Y. Wu, K. Chen, K. Xue, and D. Ni, “Nemo-based mobility management in lisp network,” in 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), Oct 2014, pp. 1–6.

[5] ONOS, https://onosproject.org/

[6] V. Ermagan, D. Farinacci, D. Lewis, J. Skriver, F. Maino, and C. White, “NAT traversal for LISP,” Internet Engineering Task Force, Internet-Draft draft-ermagan-lisp-nat-traversal-13, Sep. 2017, work in Progress.

[7] Netty, https://netty.io/

K-ONE 기술 문서

- 이 문서의 기술적인 내용은 프로젝트의 진행과 함께 별도의 예고 없이 변경될 수 있습니다.

- 본 문서와 관련된 문의 사항은 아래의 정보를 참조하시길 바랍니다.
  (Homepage: http://opennetworking.kr/projects/k-one-collaboration-project/wiki, E-mail: k1@opennetworking.kr)

작성기관: K-ONE Consortium
작성년월: 2018/02