

Locator-ID Separation Protocol in Traditional & SDN Networks

Ragavi Swarnalatha Raman, Srikanth Gopalakrishnan, Aparajita Bagchi
Software Defined Networking, North Carolina State University

1. Introduction

1.1. Purpose

The purpose of the project is to demonstrate the LISP (Locator/ID separation protocol) in traditional and SDN networks. The idea behind the development of LISP is as follows: The Internet routing and addressing architecture uses a single namespace, the IP address, to simultaneously express two functions about a device: its identity, and its location within the network. One very visible and detrimental result of this single namespace has been manifested in the rapid growth of the Internet's DFZ (default-free zone) i.e. the group of Autonomous Systems(AS) that form the core network. Also, the increase in the routing table entries of routers in the core network is a concern to the entire internet. This happens as a consequence of multihoming, traffic engineering, non aggregatable address allocations, and business events such as mergers and acquisitions. Further, as devices such as laptops or smartphones move around, they connect to the nearest available network, and this causes TCP connection losses among those devices. We need a way for a Clients should be able to move around different network domains while still being able to receive traffic that originated before the client made a shift.

The aim of this project is to demonstrate how LISP overcomes this mobility issue by separating the single namespace i.e. IP addresses into addresses that are routable (RLOCs) and addresses that are private(EIDs). This project implements the LISP principles in a Traditional Network conforming to the RFCs 6830 and 6833. And in an SDN network using an SDN controller and OpenvSwitch.

1.2. Project URL:

<https://sites.google.com/ncsu.edu/lisp/>

1.3. Terms Used:

- **LISP** : Locator/ID Separation Protocol
- **EID**: Endpoint Identifier
- **RLOC**: Routing Locators
- **ITR**: Ingress Tunnel Router - Router present at incoming point that receives packets from source end systems, encapsulates them in LISP header, and sends them to RLOCs connected to destination end hosts.
- **ETR**: Egress Tunnel Router- Router present at outgoing point that receives the LISP encapsulated packets from ITRs, decapsulates them and sends them to the destination end points.
- **xTR**: ITR/ETR
- **Map Request**: Request message sent from ITR to Map Resolver for EID to RLOC mapping
- **Map Reply**: Response message sent by Map-Server/Map-Resolver or ETR to ITR providing EID to RLOC mapping.
- **Map Register**: Message sent by ETR to Map-Server specifying its EID-prefix to RLOC associations.
- **MS: Map Server** - Distributed database that maintains EID prefix to RLOC mapping.
- **MR: Map Resolver** - Resolves EID to RLOC map request of ITR.
- **OVS**: OpenvSwitch
- **OOR**: Open Overlay Router

2. Components of the Project

2.1.1. Platform:

- Two separate EoGeni slices , one dedicated for the Traditional approach and the other for the SDN topology were created.
- All ITRs/ETRs , Map Resolver/Map Server , core routers and end clients (EIDs) are each separate ExoGeni nodes in the traditional topology.
- All OVS switches, Controller with MS/MR , core routers and end clients (EIDs) are each separate ExoGeni nodes in the SDN topology.

2.1. Areas:

The project covers and involves work in these areas:

- Tunneling: OOR's open source library was used to enable LISP tunnel on Ingress and Egress routers.
- Networking: Established a core network with several routers with BIRD running OSPF v2. Manually added routes from xTR's to the EID and RLOC portions of the network. In OpenFlow ARP messages cannot be handled by the OVS trivially. The headers of ARP were modified to be forwarded within OVS from an EID to RLOC on the core network while still maintaining the integrity of the protocol. Even ICMP messages will have to be modified in a similar manner.
- Openflow: Openflow messages act as a southbound communication between the SDN controller and the OVS switches. They also act as a northbound interface between the SDN controller to the Map Server/Resolver Application.

1. Traditional Approach : The traditional network has the following nodes:

- Tunnel Routers: ITRs and ETRs. These are directly connected to the end hosts or clients on one side and the core network on the other. The OOR (an open source implementation to deploy overlay networks) is installed and configured on these routers.

- Core Routers: These routers form the network core. Bird routing daemon is installed on these routers to enable routing between them using OSPF that we had to manually configure. They are identified as RLOCs as per the LISP terminology.
- Clients: These are the endpoints of the network identified by EIDs.
- Map Server/Resolver: Maintains the mapping between EIDs and RLOCs, interacts with the ITRs to receive Map requests and to resolve the EID-RLOC mappings and with the ETRs to register the EID-RLOC mapping. The OOR (Open Overlay Router) has a Map Resolver/Server module that is used to set up the Map Server/Resolver. This MS/MR should be able to receive Map request/resolve messages from xTR's

2. SDN Approach : The core routers and clients remain as is, according to the traditional network topology. The additional components in the SDN topology are:

- SDN Controller (Ryu) and Map Server/Resolver : Ryu Controller , an open source, component based controller is used to manage the OVS switches. The Map Server/Resolver that takes care of resolving the EID-RLOC mapping and manages the mapping database is implemented as an application and is interfaced with the SDN controller. Ryu uses southbound APIs to interact/manage the switches and uses northbound APIs to interact with the controller applications.
- OpenvSwitch : The tunnel routers are replaced by OVS switches working in L3 mode. These OVS switches are connected to the EIDs , the core network and the SDN controller. The SDN controller manages the fl w table entries of these switches. The OVS will take the place of traditional ETRs and ITRs.

3. Design and Develepment Plan

3.1. HIGH LEVEL DESIGN

3.1.1. Locator ID Separation protocol

LISP is based on the concept of dynamic encapsulation and decapsulation of a packet header. It is a network layer protocol that enables separation of IP addresses into two new numbering spaces:

1. Endpoint Identifiers (EIDs) and
2. Routing Locators (RLOCs).

The EIDs and RLOCs are in practice the same as IPv4 or IPv6 addresses. This enables backward compatibility with existing internet infrastructure. It is only their semantics that differ. RLOCs can be thought of as the IP address space present only in the Core network side, while EIDs are a unique IP address on the sub-network which may have different prefix matching. New LISP capable routers will have to be deployed, in addition to a MAP resolver for this approach, but the core network can remain the same.

(i) LISP site devices:

ITR : An ingress tunnel router that receives packets from the client (source EID) in its LISP site, encapsulates the packets and sends them to a remote LISP site where the destination EID is attached to via the underlying core network. The ITR also sends Map Requests to the Map Server / Map Resolver querying for EID-RLOC mappings and processes the Map Reply messages.

ETR : An Egress tunnel router that receives packets from the core network, decapsulates them and sends it to the destination EID (client) at its LISP site. The ETR also registers with the Map Server to inform its EID-to-RLOC mappings and responds to Map Request messages from the Map Server.

(ii) LISP Infrastructure devices: Map Server : The Mapping System or Map Server (MS), is a distributed database that maintains the association between EIDs and respective RLOCs. The MS also receives Map Requests from the ITRs and forwards it to the registered ETRs that is authoritative for the EID prefix being queried.

Map Resolver : The Map Resolver receives Map Requests from ITRs and resolves it to the corresponding RLOC address sending the resolved address back to the ITR. The Map Resolver also sends negative map replies to ITRs that query for EIDs prefix that are not present in registered LISP sites handled by the Map Server.

(iii) Working: The clients / EIDs have private IP addresses and have no information about the core network and the interfaces facing the core network on the xTRs. Communication between the EIDs in different LISP sites happens with the help of LISP Tunneling where an additional RLOC header is added on top of the header containing EIDs with the help of the Map Server / Resolver. This Map Server/Resolver maps the EID-RLOC pairs. The movement of one EID to a different RLOC gets updated in the

Map Server and thereby gets updated in the respective ITRs when these ITRs needs to reach the relocated EID.

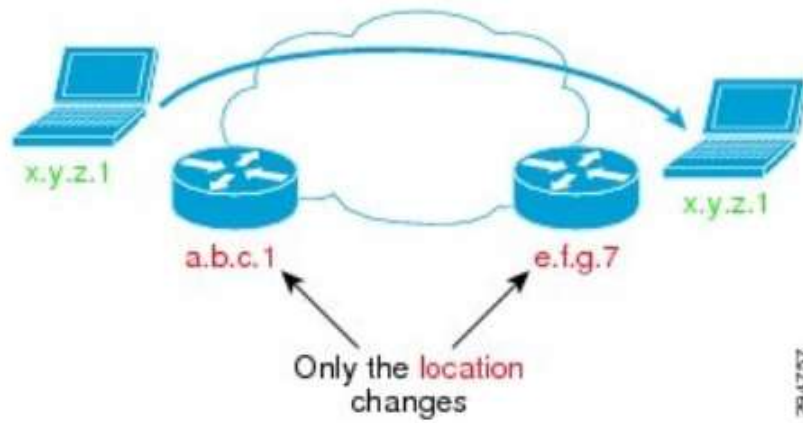


Figure 1: How EID mobility works, **source:** www.cisco.com

3.1.2. Network Topology

General Topology Showing LISP Sites AND Core Network:

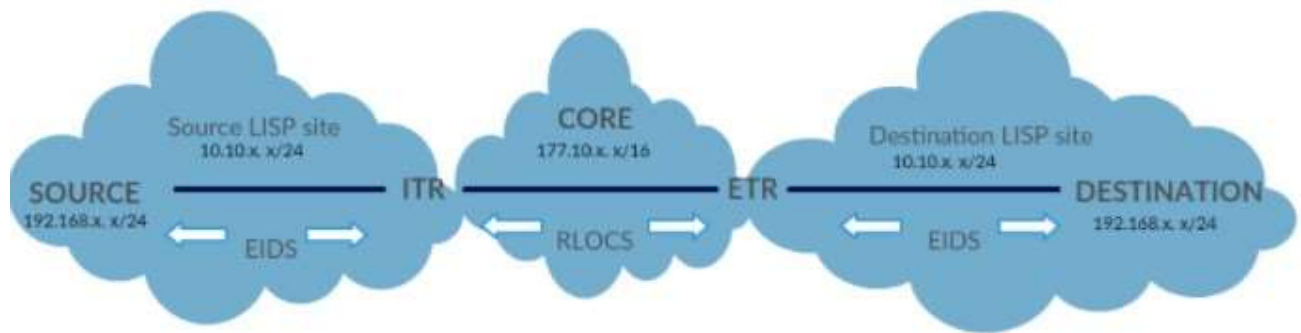


Figure 2: Subnet plan for Client to Client communication

Traditional Network Topology

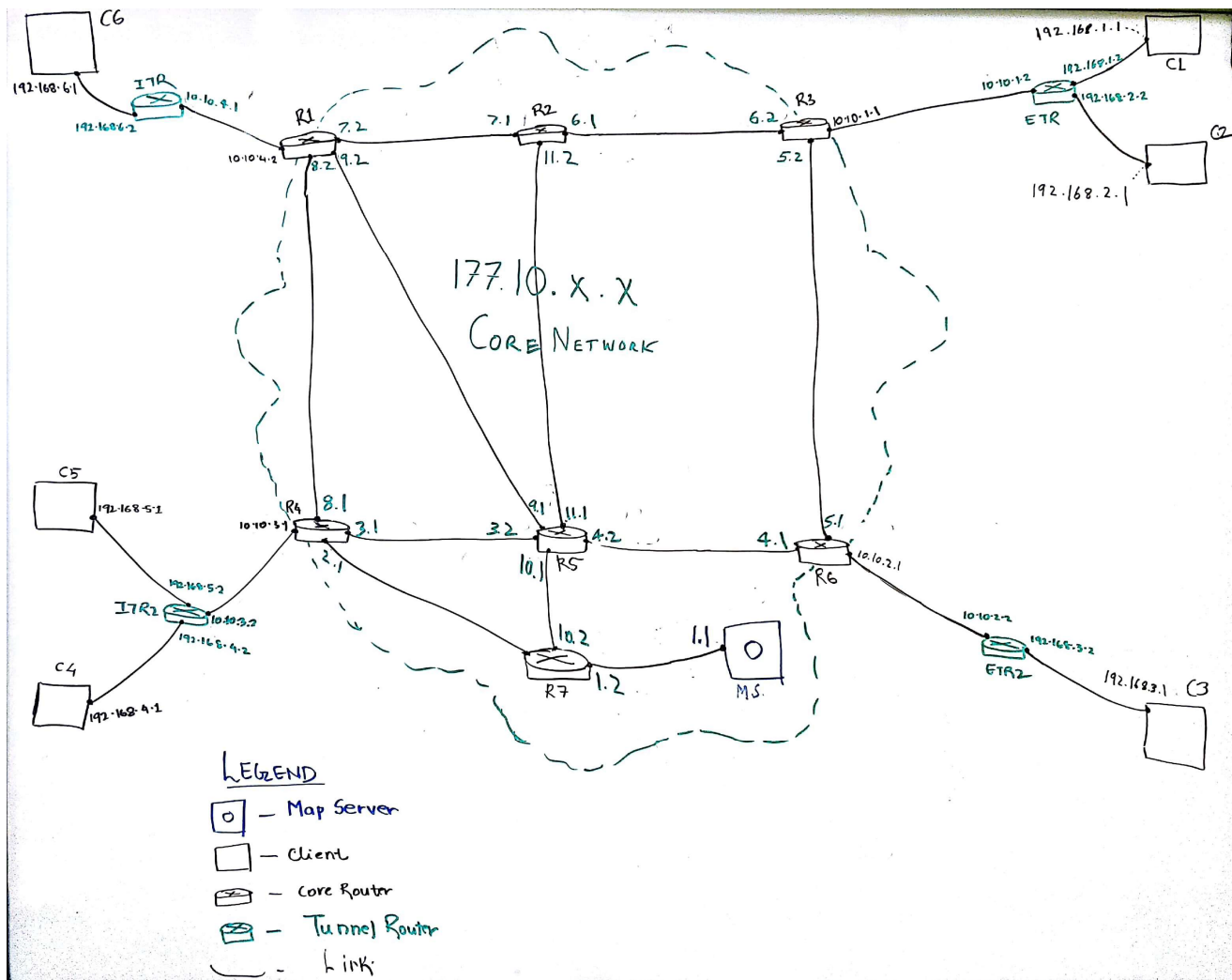


Figure 3: Subnet plan for Client to Client communication

- The traditional topology is created as a separate ExoGeni slice with each node having a specific functionality. Each node is a stand alone linux box or a linux box with a specific application installed and running in it to perform the role in the topology it has been assigned.
- End systems (EIDs) are configured with private addresses and are connected only to the Tunnel routers (ETRs and ITRs). The ETRs and ITRs are connected to the core network through public addresses represented as RLOCs.

- Map Resolver/Map Server ,ITRs and ETRs are linux boxes with OOR installed and running and they communicate to each other using the public addresses.
- The core network have a ospf routing daemon enabled between them and act as the underlying network to carry packets between the Tunnel routers.

Traditional Network Topology

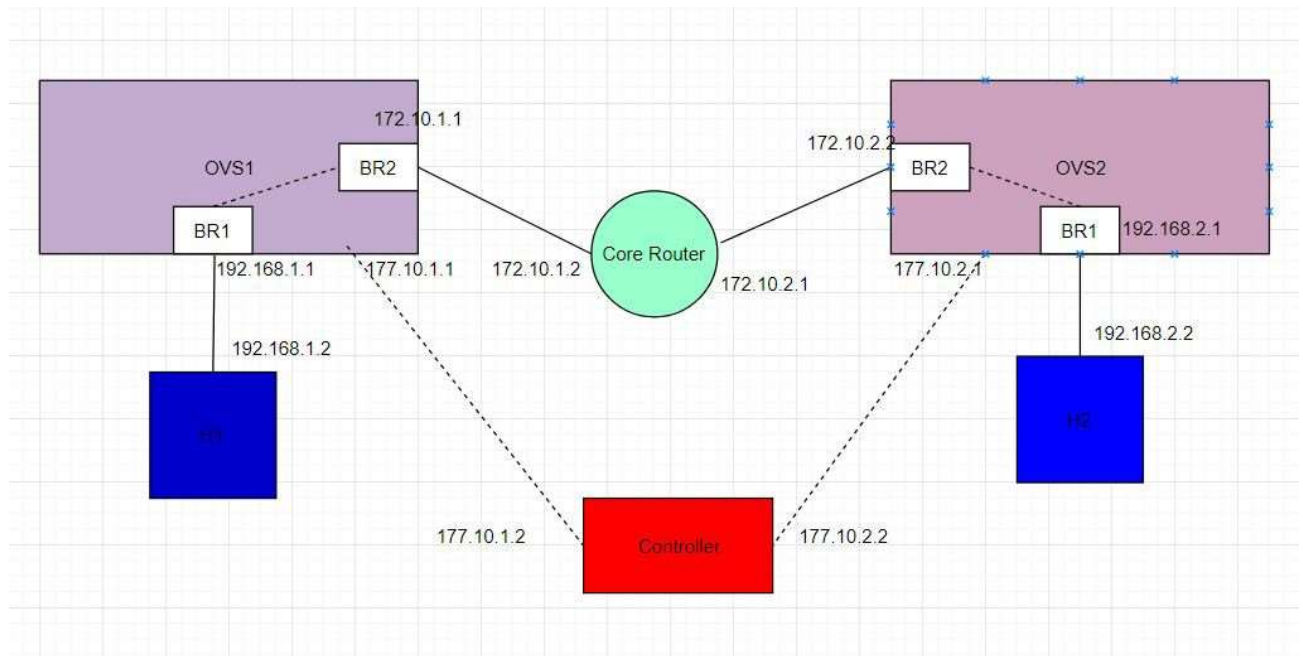


Figure 4: Subnet plan for Client to Client communication

- The setup for the SDN approach is as follows: (ITRs and ETRs) are replaced with OVS switches and the Map Server/Resolver replaced with a SDN controller and the function of the Map Server/Resolver written as a controller application.
- The SDN topology is created as a separate ExoGeni slice with each node having a specific functionality. Each node is a stand alone linux box or a linux box with a specific application installed and running in it to perform the role in the topology it has been assigned.

- The OVS switches perform as alias Tunnel routers and have ovs installed, 2 bridges added along with fl ws configured in them. They connect to the core network via public addresses and the clients (EIDs) via private addresses.
- End systems (EIDs) are configured with private addresses and are connected only to the Tunnel routers (ETRs and ITRs). The ETRs and ITRs are connected to the core network through public addresses represented as RLOCs.
- The Core router is a stand alone Linux box with a public subnet addressing in each of its interface. The single core router takes the role of the underlying core network that needs to be crossed by the traffic that fl ws between the LISP sites

3.1.3. Design Specification

Traditional Network

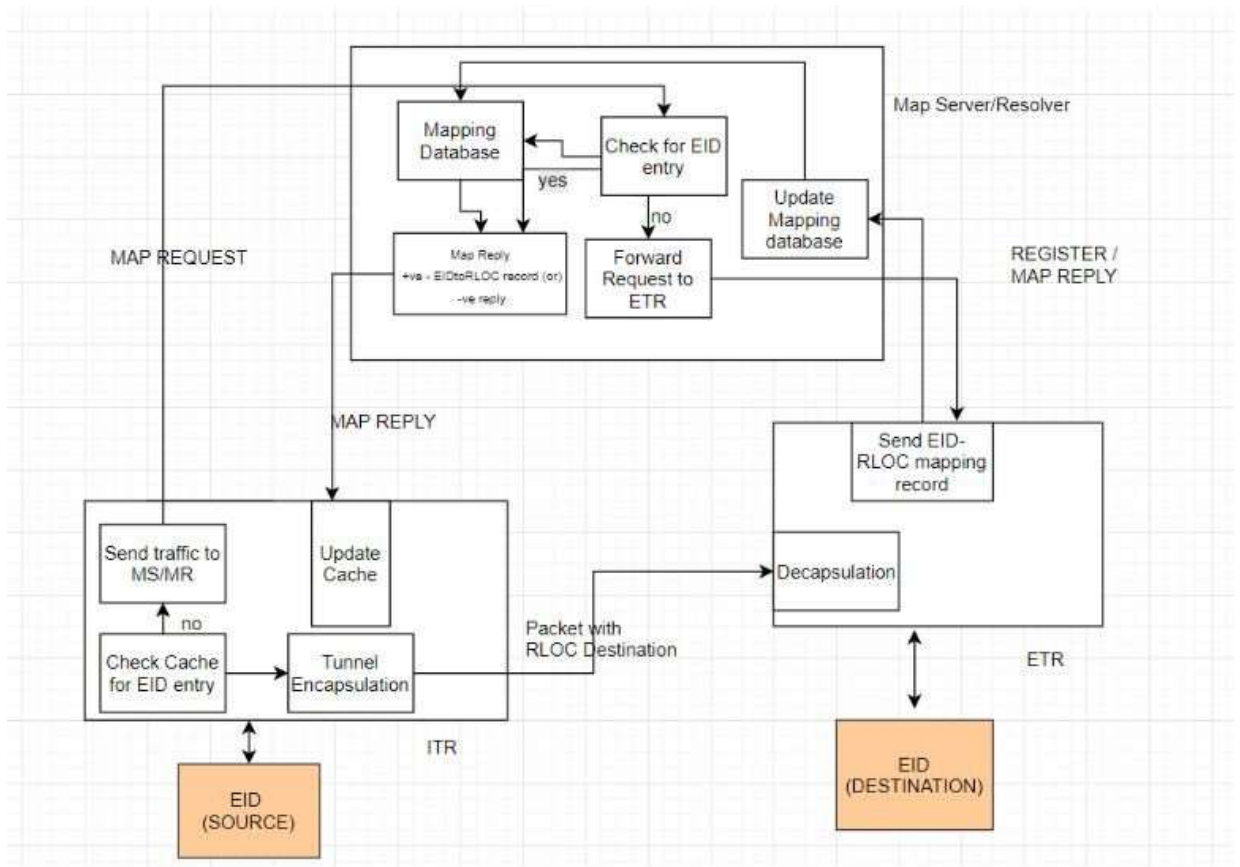


Figure 5: Traditional

Stage 1 - Map Server Registration and Learning :

On boot up, the Map Servers database is empty and listens for Registrations from ETRs. ETRs register with the Map Server and send their Map Records containing EID-RLOC mapping ie) the list of clients in the ETRs own LISP site. The Map Server adds these entries to its database and the ETR receives a successful registration message. An error in registering would be informed to the ETR via appropriate error messages. The ETRs re-register their Mapping records to the Map Server every 60 seconds throughout the time period when LISP is enabled.

Stage 2 - ITR sends Map Requests to Map Resolver:

When a source EID attached to an ITR in a LISP site pings an EID present in another LISP site, the ICMP request packets created at the source

EID reaches the ITR. The ITR realizes that it is a private address and checks its cache for EID-RLOC mapping record that helps the packet to reach the destination EID via a tunnel interface. If no cache entry for the destination EID prefix exists, the ITR sends a mapping request to the Map Resolver asking for the EID-RLOC mapping.

Stage 3 - Map Replies by Map Resolver / ETR :

When a Map Resolver/Server receives the Map Request, it checks its Mapping database to find the appropriate EID-RLOC mapping record. If no records are found, the Map Resolver forwards the request to its registered ETRs to check if they have an EID-RLOC mapping for the requested destination EID prefix. The ETR replies with its EID-RLOC mapping which in turn is sent to the ITR as a Map Reply. If the Map Resolver finds no matching EID-RLOC mapping record, it responds with a negative Map Reply.

Stage 4 - Packet Encapsulation and Tunneling :

On learning the RLOC address of the destination EID, the ITR adds the entry to its cache, does encapsulation of the original packet, adds a LISP header, and a tunnel header with the outer destination address as the RLOC address. The encapsulated packet is now sent to the ETR via a tunnel established between tunnel interfaces at the ITR and ETR.

Stage 5 - Packet Decapsulation and forward to destination EID:

The encapsulated packet on reaching the ETR is decapsulated, the outer header and the LISP headers are removed, the ETR now processes the inner header of the packet. The inner header of the packet's destination address is the EID that is present in the LISP site of the ETR and the ETR has routing entries to reach it. The packet is now forwarded to the appropriate destination EID.

Mobility:

When an EID moves from one LISP site to another LISP site, the ETR of the new LISP site to which the EID has relocated updates the Map Server with the new EID-RLOC mapping. The ITR keeps sending its traffic to the EID to the old RLOC address until its cache gets cleared. A subsequent new Map Request updates the ITR with the new EID-RLOC mapping and the ITR now forwards the packet to the new ETR to whose LISP site the EID has relocated to.

SDN Network

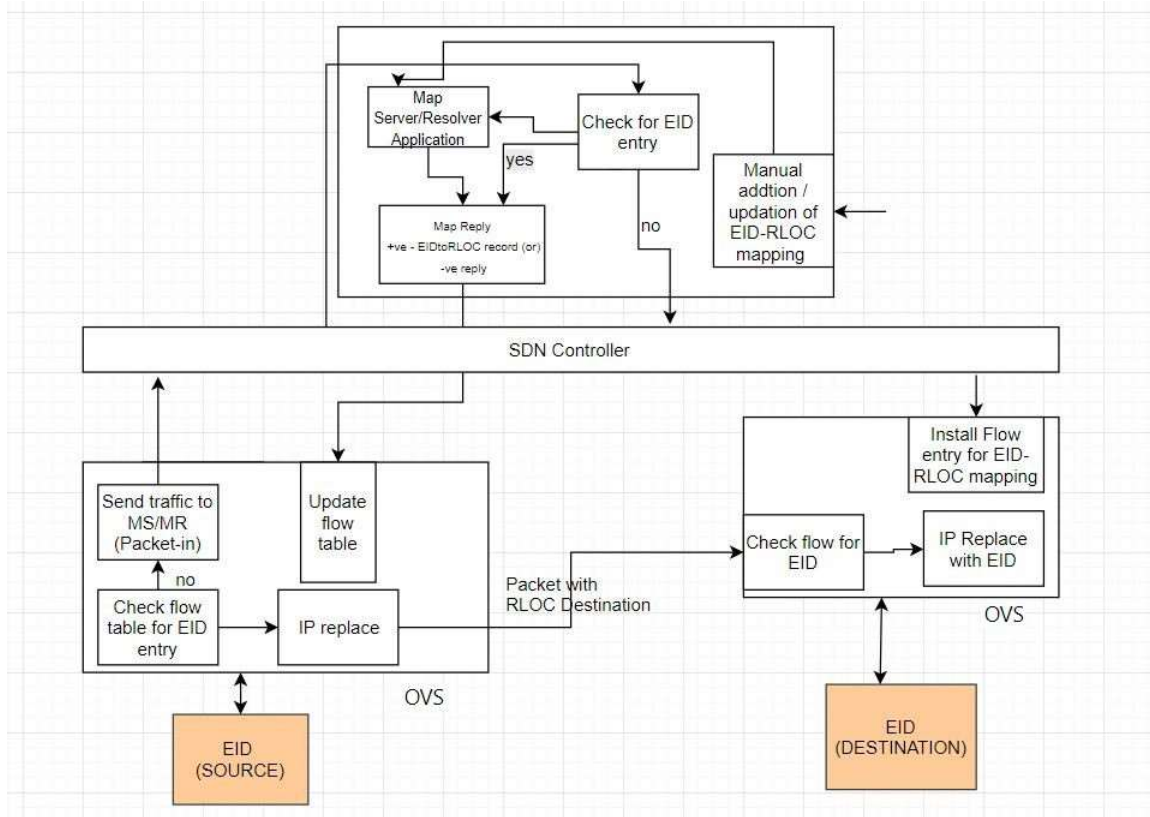


Figure 6: SDN

Stage 1 - Map Server Learning and Destination OVS Flow table installation:

On boot up, the Map Server applications database is empty and Map Records containing EID-RLOC mapping ie) the list of clients connected to the destination OVS switch LISP site are updated manually. The entries do not change until the records are re-updated. The EID-RLOC mappings are installed as flow table updates in the corresponding OVS switches functioning as ETRs.

Stage 2 - Source OVS sends Packet-In to SDN Controller:

When a source EID attached to a source OVS switch in a LISP site pings an EID present in another LISP site, the ICMP request packets created at the source EID reaches the OVS switch. The OVS realizes that it is a private address and checks its flow table for EID-RLOC mapping record that helps the packet to reach the destination EID. If no cache entry for the destination

EID prefix exists, the OVS sends a Packet-In to the Controller which in turn interfaces with the Map Resolver application asking for the EID-RLOC mapping. If the mapping is present the Controller installs a flow table update in the OVS switch indicating what to do with the subsequent flows (ie) change flow destination or drop.

Stage 3 - Flow table Update and IP Re-write:

The OVS switch on receiving the flow table update, updates its flow table entry and adds a match and action pair to it. The match and action are as follows: Match: The source and destination IP addresses of the flow are the EIDs. Action: Replace the source and destination EIDs with the RLOC (OVS switch) source and destination IP addresses. The traffic flows are now sent to the re-written RLOC addresses.

Stage 4 - IP Rewrite at destination and forward to EID:

The OVS switch at the destination receives the flow and finds a flow table entry that matches the source and destination RLOC addresses. The flow table entries are installed by the SDN controller at the destination OVS switches after startup. Action : The source and destination RLOC addresses are replaced with the source and destination EIDs from the flow table entries. The flow is now forwarded to the appropriate destination EID. If no flow table match is found, the traffic is dropped.

Mobility:

When an EID moves from one LISP site to another, the OVS switches do not self-realize the change that has happened. The updated EID-RLOC mapping needs to be added in the Map Resolver application and reinstalled in both the source and destination OVS switches as new flow table updates. The old table entries are removed after a certain timeout. Now when the source EID traffic tries to reach the relocated EID in a new LISP site, the updated flow table entry changes the flow destination address to the new LISP site's RLOC and sends the traffic.

3.2. Low Level Design

3.2.1. Traditional Network

Tunnel Routers: The traditional network takes the help of the Open Overlay Router (a library/software that enables packet encapsulation/decapsulation) and uses tunneling to transfer the packet from the source to destination. This software is installed in the ITRs and ETRs. In the current version, OOR uses the LISP protocol for the control-plane, NETCONF/YANG for the management-plane (e.g. overlay identifiers provisioning, etc) and use

LISP and headers for encapsulation. The ITRs and ETRs require the following for correct functioning : A non-routable private address that connects the Tunnel router to a single EID / several EIDs A public routable address (RLOC) through which it connects to the underlay network The public address (RLOC) of the Map Resolver/Map Server node that provides the EID-RLOC mapping The RLOC of proxy ETRs in the form of MAP REPLIES from the Map Resolver / Map Server or its own cache A LISP tunnel that performs packet encapsulation / decapsulation and extra headers that conform to the LISP specifications.

Ingress Tunnel Router Flow:

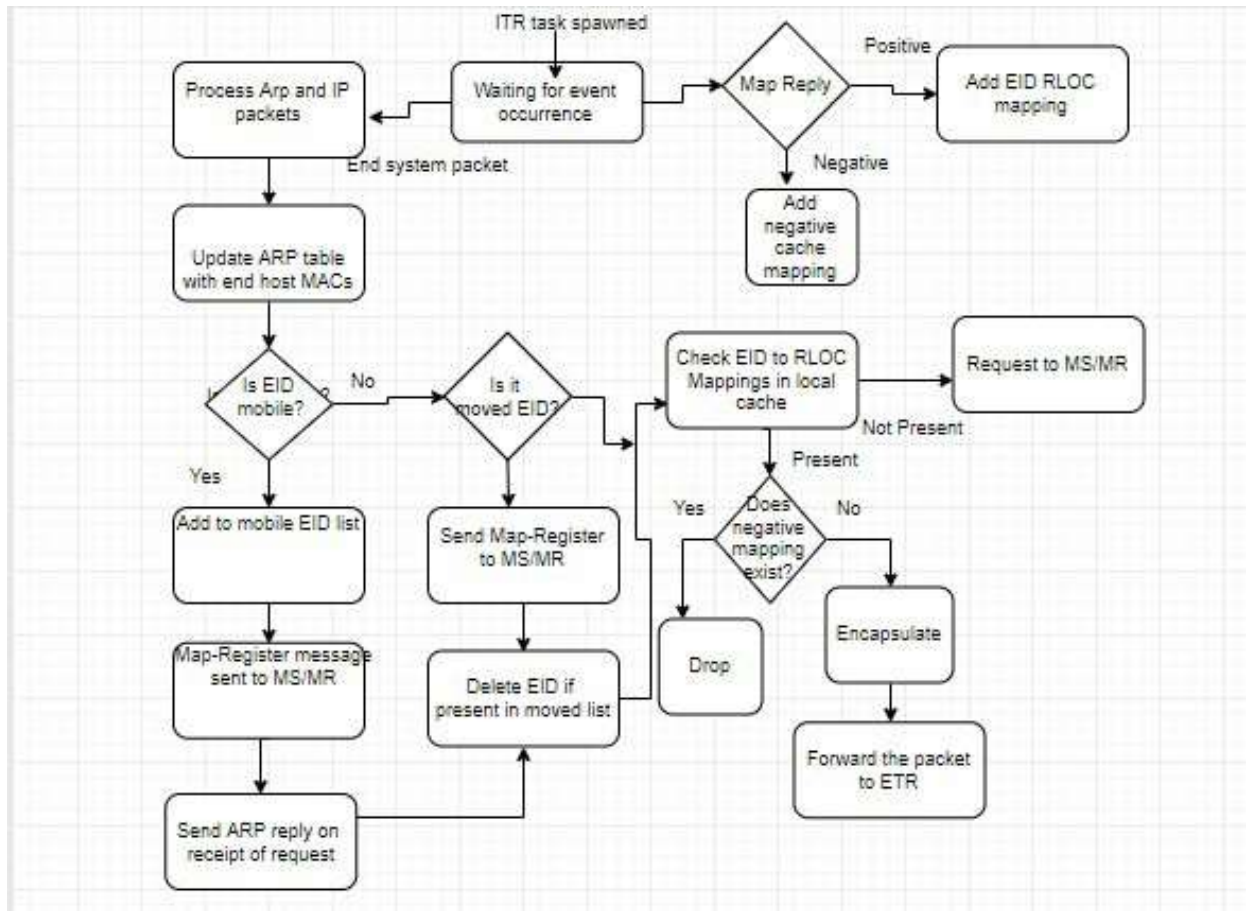


Figure 7: ITR Flow

Egress Tunnel Router Flow:

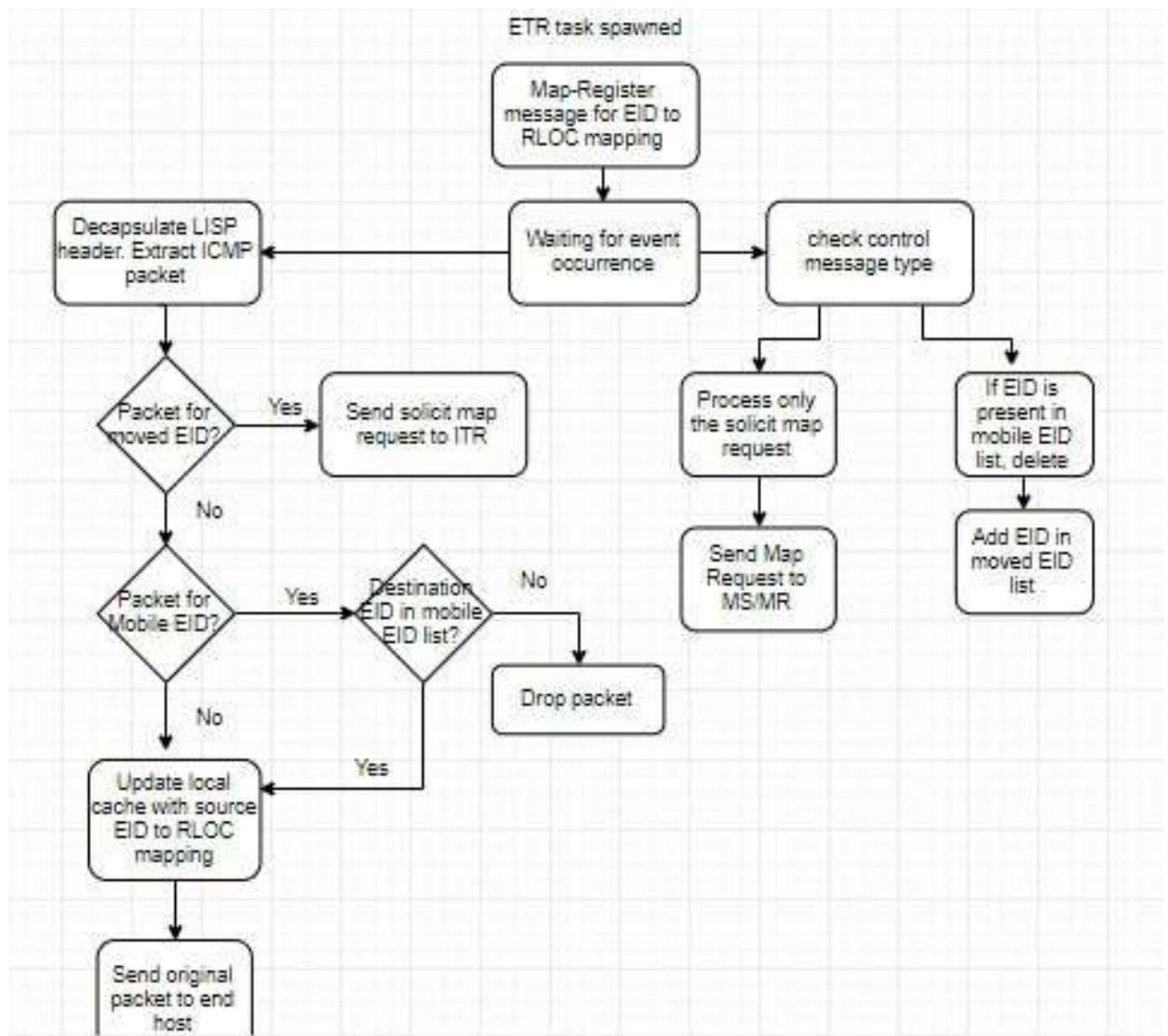


Figure 8: ETR Flow

Map Server / Resolver Flow: The Map Resolver processes registration requests from the tunnel routers and notify registration messages in the console as part of the registration process. It can accept EID-RLOC mapping for a subnet or even for specific prefix inside the subnet. The Map Resolver accepts mapping requests from ITRs and forwards them to ETRs during the search of a mapping.

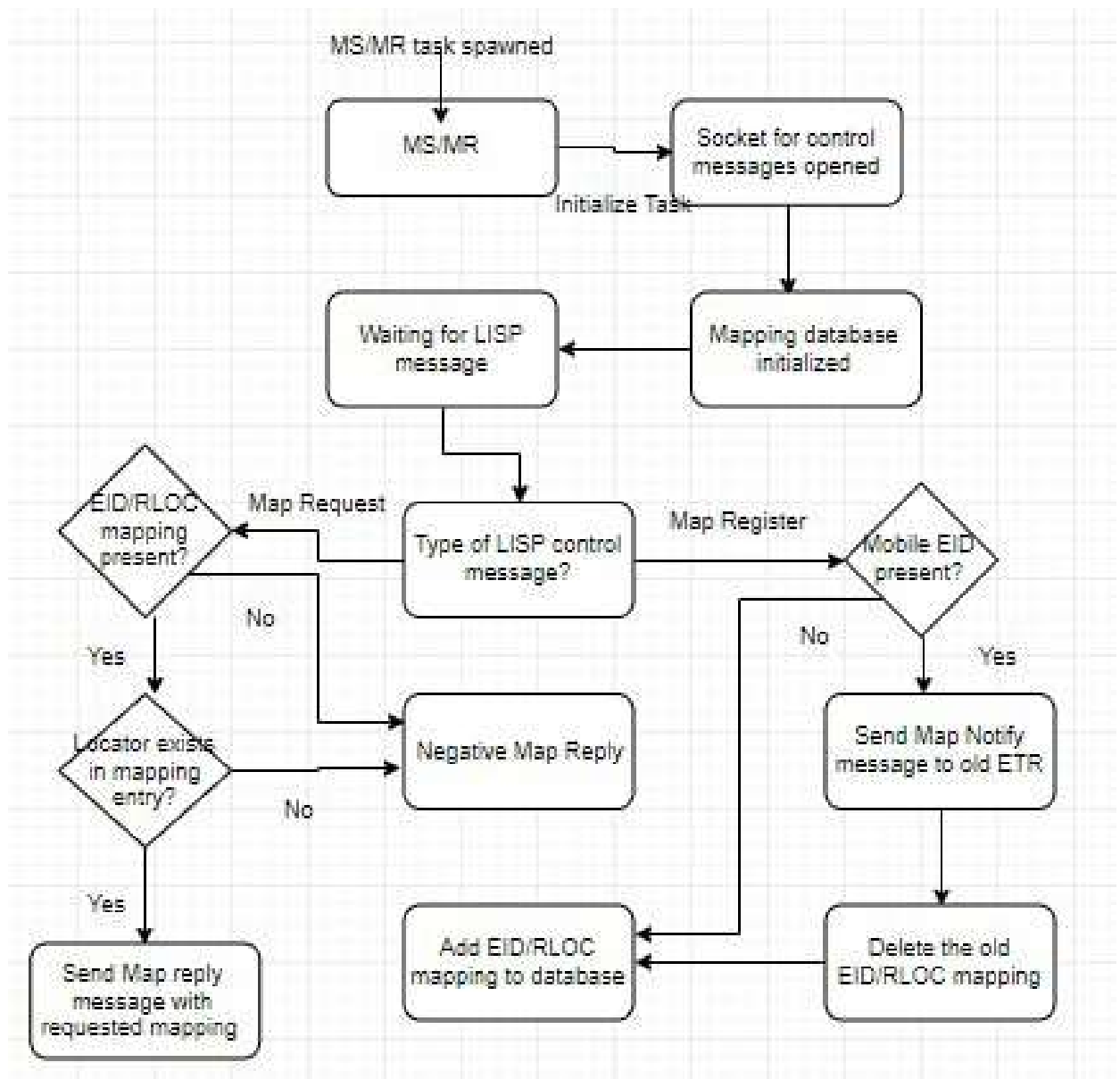


Figure 9: Map Resolver/ Server Flow

3.2.2. SDN Network

SDN Controller and Map Server/Map Resolver Application The SDN controller acts as a centralized control to the communication between

the private LISP sites and the public core network. Ryu Controller has been configured to suit the needs of the scenario. The controller to start with registers with the OVS switches and does a receives Packet-In Messages from them. All packet-in messages are sent to the Controller by the switchfeature-handler() module and based on the type of message (ARP/ICMP)packets the fl w proceeds to handle them

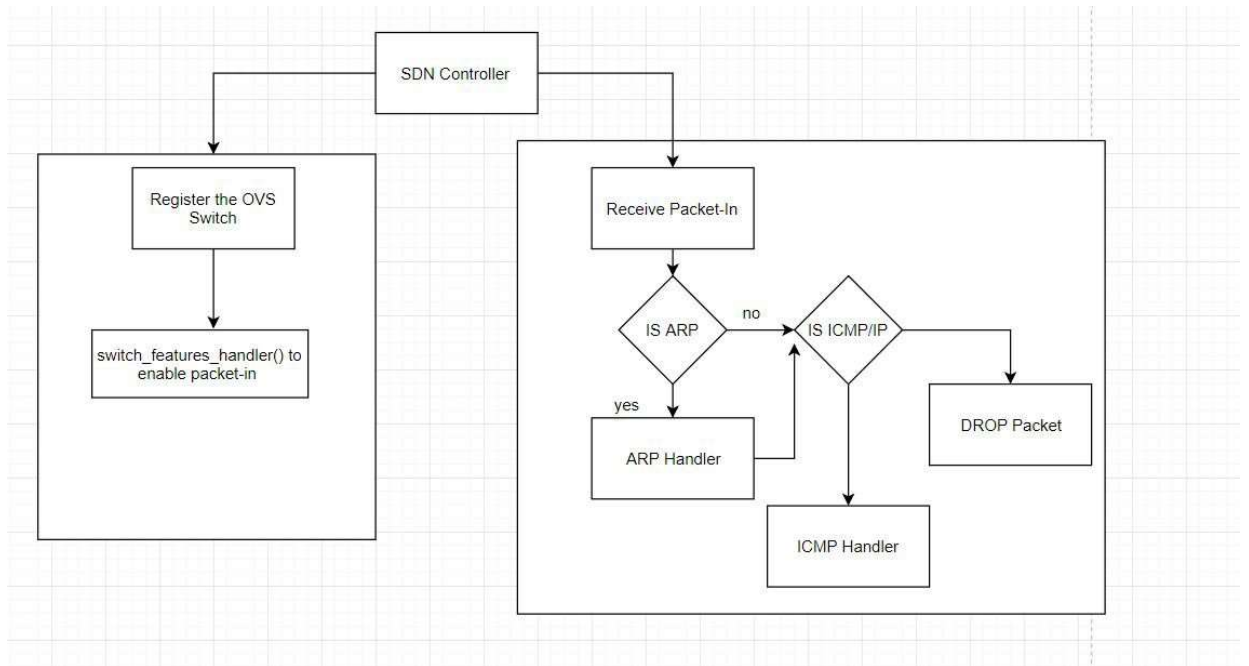


Figure 10: Controller Flow

Implementation of ARP and ICMP Handlers

ARP Handling

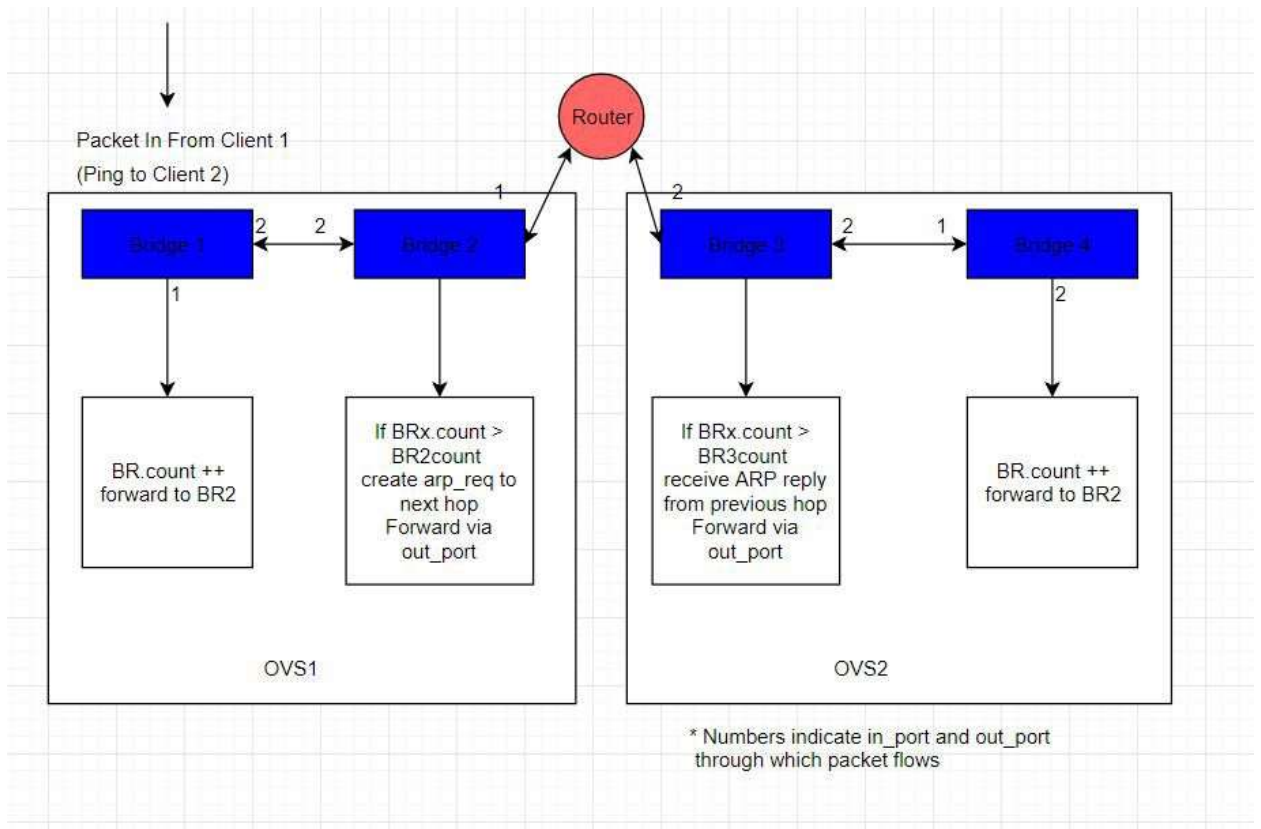


Figure 11: ARP Handler

ICMP Handling

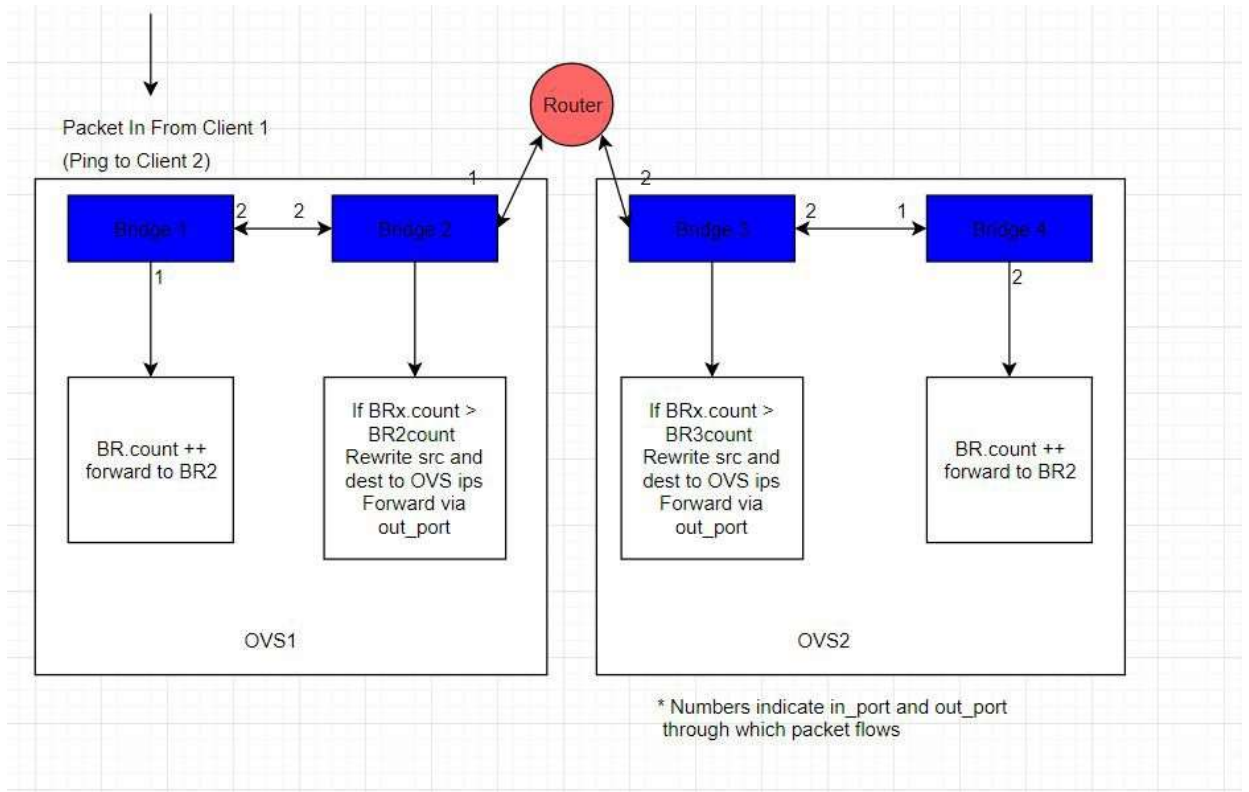


Figure 12: ICMP Handler

Tunnel Routers OVS switches are used to mimic the functionality of the Tunnel routers that function as ETRs and ITRs. The OVS switches by behavior they function as L2 bridges. Our implementation has tweaked the functionality of the OVS bridge to behave like a router and make it work like a router.

The OVS switches do not perform encapsulation and decapsulation of the packets as in the traditional approach. Instead, they rewrite the IPs of the source and the destination that dynamically changes the way in which the packet is transmitted.

The clients send packets with the private source and destination addresses which are received by the OVS switches, processed and packets it in to the controller where using the Map Resolver application that runs in the SDN controller, it rewrites the private ip addresses into the corresponding RLOC (public) addresses.

The OVS switches accepts flows sent by the controller and does a match and action on the flows.

3.3. *Per-Component Development Phases:*

3.3.1. *Traditional Topology*

- Setting up Geni Topology [Ragavi,Srikanth,Aparajita]
- Setting up Core Router routing daemon [Srikanth,Aparajita]
- Installation and Configuration of OOR [Srikanth]
- Setting up ITRs, ETRs and Map Resolver and Map Database [Srikanth,Aparajita]
- Configure/Test the Base LISP Case [Ragavi,Srikanth,Aparajitha]
- Configure/Test the Mobility node Case [Ragavi,Srikanth,Aparajitha]
- Automation of all Node configuration [Srikanth]

3.3.2. *SDN Topology*

- Setting up Geni Topology [Aparajita]
- Setting up OVS switches [Srikanth , Ragavi]
- Installation and Configuration of Ryu [Srikanth,Aparajita]
- Implement ARP and ICMP Handling application at SDN controller [Ragavi,Srikanth]
- Creating Map Resolver / Server Database at SDN Controller [Ragavi,Srikanth]
- Test and debug the ARP/ICMP Handler [Ragavi,Srikanth,Aparajitha]

3.4. *Timeline:*

3.4.1. *Traditional Topology*

- Setting up Geni Topology [4/11/2018]
- Setting up Core Router routing daemon [4/12/2018 - 4/13/2018]
- Installation and Configuration of OOR [4/12/2018 - 4/13/2018]

- Setting up ITRs, ETRs and Map Resolver and Map Database [4/13/2018 - 4/15/2018]
- Configure/Test the Base LISP Case [4/14/2018 - 4/18/2018]
- Configure/Test the Mobility node Case [4/17/2018 - 4/19/2018]

3.4.2. *SDN Topology*

- Setting up Geni Topology [4/20/2018]
- Setting up OVS switches [4/21/2018 - 4/23/2018]
- Installation and Configuration of Ryu [4/23/2018]
- Implement ARP and ICMP Handling application at SDN controller [4/24/2018 - 4/28/2018]
- Creating Map Resolver / Server Database at SDN Controller [4/28/2018]
- Test and debug the ARP/ICMP Handler [4/28/2018]
- Final Report [4/27/2018 - 4/29/2018]

4. Verification and Validation

4.1. Test Application

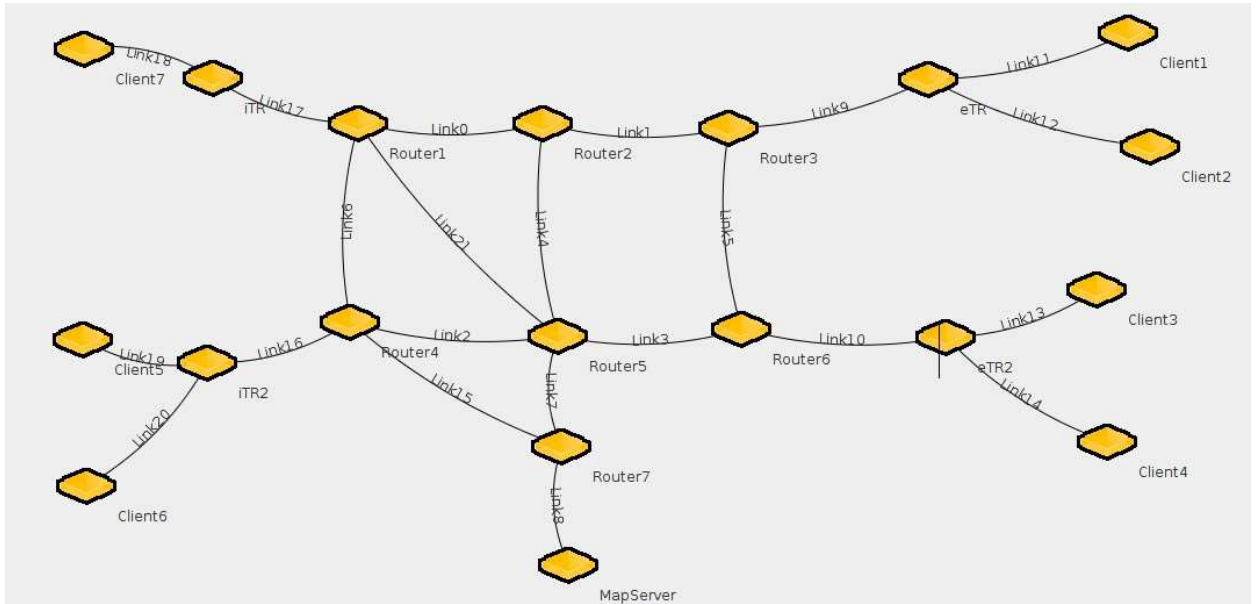


Figure 13: Traditional Topology

Table 1: Test Plan

Case	Setup	Action	Expected Observation	Recorded Observation
1	xTR's are running and MS/MR is running	Ping from source EID to destination EID	The ETR registers its EID-RLOC mapping to the Map Server. The ITR sends a map request to the Map Server asking for the destination EID-RLOC mapping. The Map server responds with this information through a map reply message. A LISP tunnel is established and now the source EID should be able to reach the destination EID.	Source EID is able to send ICMP packets to destination EID. On debugging, the messages from the ETR to the MS/MR, ITR's map request and MS map reply are visible in the respective xTR's terminal
2	MS/MR off at the beginning, xTR's are running	Ping from source EID to destination EID	The ETR is not able to register EID-RLOC mapping with the MS/MR. The ITR's map-request is not served and the LISP tunnel is not successfully established. Source EID should not be able to reach destination EID	ICMP packets do not reach the destination because the xTRs are not able to register the clients with the MS/MR.

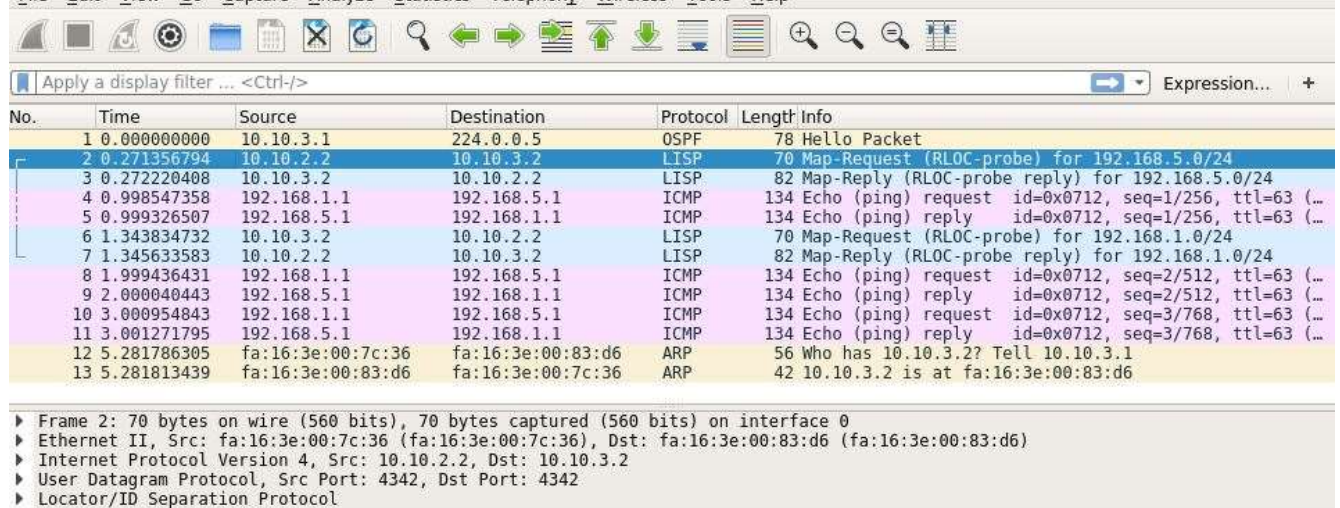
Table 2: Test Plan - Contd1

Case	Setup	Action	Expected Observation	Recorded Observation
3	MS/MR turned off after ETR's registration, xTR's are running	Ping from source EID to destination EID	Mapping information messages were never sent by the MS/MR, hence without local cache ping would fail	ICMP packets do not reach the destination because mapping information has not been exchanged between ITR and MS/MR
4	MS/MR turned off after one EID pings at least one other EID connected to an ETR	Ping from source EID to destination EID	ETR registers the destination EID-RLOC mapping. ITR sends a map-request for destination EID-RLOC mapping. MS/MR responds with mapping information, which is stored in the ITR's cache. LISP tunnel is established and ping from source EID should reach destination EID regardless of MS/MR	After the MS/MR is turned off, ping still works to the original EID that was first pinged. Different EIDs pings to the same RLOC is yet to be tested
5	MS/MR turned off after one EID pings all other EIDs connected to the ETRs	Ping from source EID to destination EID	Local cache obtained from MS/MR would still allow pings from the source EID to still work with all other EIDs till timeout	ICMP packets reach all EIDs as the ITR has stored all the EID prefix to RLOC mapping in its cache before the MS/MR was turned off.

Table 3: Test Plan

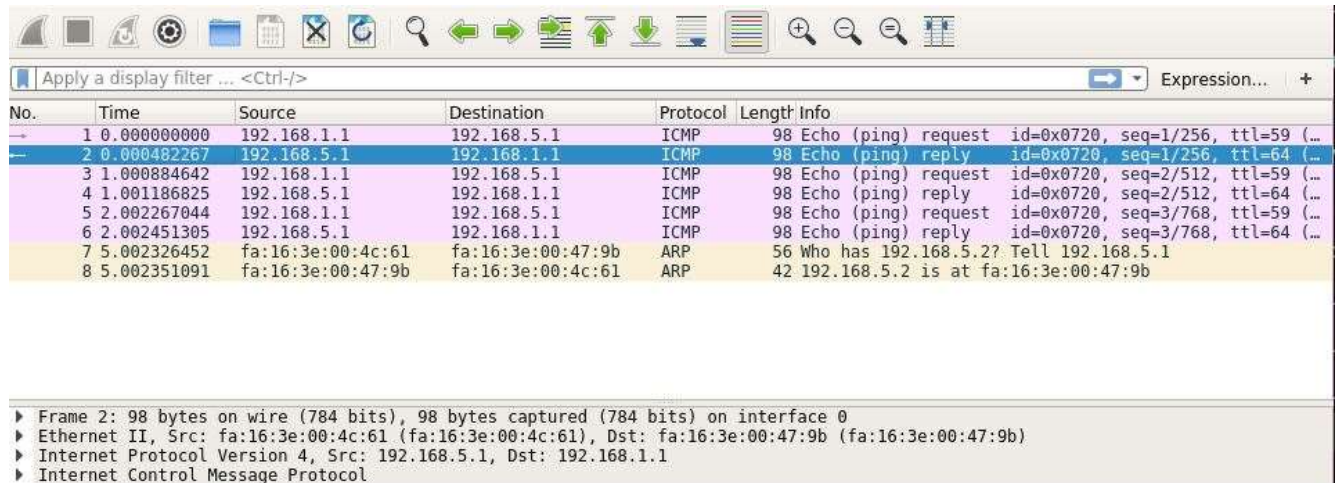
Case	Setup	Action	Expected Observation	Recorded Observation
6	xTRs turned off, but MS/MR is running	Ping from source EID to destination EID	No routing information is available from the private address space to the public address space, ping will fail.	ICMP packets do not reach destination because the tunnel routers that connect the clients in the private network to the public network are down. There is no mapping information associated with this ping.
7	Simulate Client Mobility by changing EID of destination while pinging.	Continuously Ping from source EID to destination EID. Ping should be close to the local mapping timeout	ICMP packets are received(after packet loss) at the xTR responsible for the moved destination EID. Indicating successful RLOC-EID re-mapping. A	After waiting for cache to timeout. Ping messages were seen after significant delay.

Successful Ping From Client1 to Client 2 : Traditional Topology :
Wireshark at ITR:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.3.1	224.0.0.5	OSPF	78	Hello Packet
2	0.271356794	10.10.2.2	10.10.3.2	LISP	70	Map-Request (RLOC-probe) for 192.168.5.0/24
3	0.272220408	10.10.3.2	10.10.2.2	LISP	82	Map-Reply (RLOC-probe reply) for 192.168.5.0/24
4	0.998547358	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x0712, seq=1/256, ttl=63 (...)
5	0.999326507	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x0712, seq=1/256, ttl=63 (...)
6	1.343834732	10.10.3.2	10.10.2.2	LISP	70	Map-Request (RLOC-probe) for 192.168.1.0/24
7	1.345633583	10.10.2.2	10.10.3.2	LISP	82	Map-Reply (RLOC-probe reply) for 192.168.1.0/24
8	1.999436431	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x0712, seq=2/512, ttl=63 (...)
9	2.000040443	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x0712, seq=2/512, ttl=63 (...)
10	3.000954843	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x0712, seq=3/768, ttl=63 (...)
11	3.001271795	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x0712, seq=3/768, ttl=63 (...)
12	5.281786305	fa:16:3e:00:7c:36	fa:16:3e:00:83:d6	ARP	56	Who has 10.10.3.2? Tell 10.10.3.1
13	5.281813439	fa:16:3e:00:83:d6	fa:16:3e:00:7c:36	ARP	42	10.10.3.2 is at fa:16:3e:00:83:d6

▶ Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
 ▶ Ethernet II, Src: fa:16:3e:00:7c:36 (fa:16:3e:00:7c:36), Dst: fa:16:3e:00:83:d6 (fa:16:3e:00:83:d6)
 ▶ Internet Protocol Version 4, Src: 10.10.2.2, Dst: 10.10.3.2
 ▶ User Datagram Protocol, Src Port: 4342, Dst Port: 4342
 ▶ Locator/ID Separation Protocol



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x0720, seq=1/256, ttl=59 (...)
2	0.000482267	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x0720, seq=1/256, ttl=64 (...)
3	1.000884642	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x0720, seq=2/512, ttl=59 (...)
4	1.001186825	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x0720, seq=2/512, ttl=64 (...)
5	2.002267044	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x0720, seq=3/768, ttl=59 (...)
6	2.002451305	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x0720, seq=3/768, ttl=64 (...)
7	5.002326452	fa:16:3e:00:4c:61	fa:16:3e:00:47:9b	ARP	56	Who has 192.168.5.2? Tell 192.168.5.1
8	5.002351091	fa:16:3e:00:47:9b	fa:16:3e:00:4c:61	ARP	42	192.168.5.2 is at fa:16:3e:00:47:9b

▶ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: fa:16:3e:00:4c:61 (fa:16:3e:00:4c:61), Dst: fa:16:3e:00:47:9b (fa:16:3e:00:47:9b)
 ▶ Internet Protocol Version 4, Src: 192.168.5.1, Dst: 192.168.1.1
 ▶ Internet Control Message Protocol

Wireshark at ETRs:

Io.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x77b0, seq=1/256, ttl=64 (...)
2	0.002538626	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x77b0, seq=1/256, ttl=59 (...)
3	1.001206569	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x77b0, seq=2/512, ttl=64 (...)
4	1.002640372	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x77b0, seq=2/512, ttl=59 (...)
5	2.003154415	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x77b0, seq=3/768, ttl=64 (...)
6	2.004788321	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x77b0, seq=3/768, ttl=59 (...)
7	3.004446258	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x77b0, seq=4/1024, ttl=64 (...)
8	3.005797599	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x77b0, seq=4/1024, ttl=59 (...)

▶ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: fa:16:3e:00:2d:84 (fa:16:3e:00:2d:84), Dst: fa:16:3e:00:57:71 (fa:16:3e:00:57:71)
 ▶ Internet Protocol Version 4, Src: 192.168.5.1, Dst: 192.168.1.1
 ▶ Internet Control Message Protocol

o.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.2.1	224.0.0.5	OSPF	78	Hello Packet
2	0.828812674	192.168.1.1	192.168.5.1	LISP	106	Encapsulated Map-Request for 192.168.5.1/32
3	0.833140620	10.10.3.2	10.10.2.2	LISP	82	Map-Reply for 192.168.5.0/24
4	1.834505597	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x7685, seq=2/512, tt...
5	1.838566254	192.168.5.1	192.168.1.1	LISP	106	Encapsulated Map-Request for 192.168.1.1/32
6	1.839203472	10.10.2.2	10.10.3.2	LISP	82	Map-Reply for 192.168.1.0/24
7	2.842394064	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x7685, seq=3/768, tt...
8	2.844012463	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x7685, seq=3/768, tt...
9	3.843839033	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x7685, seq=4/1024, t...
10	3.846048427	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x7685, seq=4/1024, t...
11	4.845608317	192.168.1.1	192.168.5.1	ICMP	134	Echo (ping) request id=0x7685, seq=5/1280, t...
12	4.847148070	192.168.5.1	192.168.1.1	ICMP	134	Echo (ping) reply id=0x7685, seq=5/1280, t...
13	5.846175167	fa:16:3e:00:68:9e	fa:16:3e:00:47:4d	ARP	56	Who has 10.10.2.2? Tell 10.10.2.1
14	5.846195602	fa:16:3e:00:47:4d	fa:16:3e:00:68:9e	ARP	42	10.10.2.2 is at fa:16:3e:00:47:4d

• Frame 2: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
 • Ethernet II, Src: fa:16:3e:00:47:4d (fa:16:3e:00:47:4d), Dst: fa:16:3e:00:68:9e (fa:16:3e:00:68:9e)
 • Internet Protocol Version 4, Src: 10.10.2.2, Dst: 177.10.1.1
 • User Datagram Protocol, Src Port: 4342, Dst Port: 4342
 • Locator/ID Separation Protocol
 • Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.5.1
 • User Datagram Protocol, Src Port: 4342, Dst Port: 4342
 • Locator/ID Separation Protocol

Wireshark at the Clients:

Io.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x05d7, seq=1/256, ttl=64 (...)
2	0.003155286	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x05d7, seq=1/256, ttl=59 (...)
3	1.001421406	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x05d7, seq=2/512, ttl=64 (...)
4	1.003416258	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x05d7, seq=2/512, ttl=59 (...)
5	2.002722865	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x05d7, seq=3/768, ttl=64 (...)
6	2.004505801	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x05d7, seq=3/768, ttl=59 (...)
7	5.018008878	fa:16:3e:00:2d:84	fa:16:3e:00:57:71	ARP	56	Who has 192.168.1.1? Tell 192.168.1.2
8	5.018026695	fa:16:3e:00:57:71	fa:16:3e:00:2d:84	ARP	42	192.168.1.1 is at fa:16:3e:00:57:71

▶ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: fa:16:3e:00:2d:84 (fa:16:3e:00:2d:84), Dst: fa:16:3e:00:57:71 (fa:16:3e:00:57:71)
 ▶ Internet Protocol Version 4, Src: 192.168.5.1, Dst: 192.168.1.1
 ▶ Internet Control Message Protocol

Io.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x086d, seq=2/512, ttl=59 (...)
2	0.000026027	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x086d, seq=2/512, ttl=64 (...)
3	0.999520296	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x086d, seq=3/768, ttl=59 (...)
4	0.999534836	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x086d, seq=3/768, ttl=64 (...)
5	2.001005474	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x086d, seq=4/1024, ttl=59 (...)
6	2.001024086	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x086d, seq=4/1024, ttl=64 (...)
7	3.002155866	192.168.1.1	192.168.5.1	ICMP	98	Echo (ping) request id=0x086d, seq=5/1280, ttl=59 (...)
8	3.002197429	192.168.5.1	192.168.1.1	ICMP	98	Echo (ping) reply id=0x086d, seq=5/1280, ttl=64 (...)
9	5.003728294	fa:16:3e:00:4c:61	fa:16:3e:00:47:9b	ARP	42	Who has 192.168.5.2? Tell 192.168.5.1
10	5.004351083	fa:16:3e:00:47:9b	fa:16:3e:00:4c:61	ARP	56	192.168.5.2 is at fa:16:3e:00:47:9b

▶ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
 ▶ Ethernet II, Src: fa:16:3e:00:4c:61 (fa:16:3e:00:4c:61), Dst: fa:16:3e:00:47:9b (fa:16:3e:00:47:9b)
 ▶ Internet Protocol Version 4, Src: 192.168.5.1, Dst: 192.168.1.1
 ▶ Internet Control Message Protocol

Control Messages at the Map Resolver:

```

2018/4/29 5:11:53] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.1.0/24
2018/4/29 5:11:53] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.2.2
2018/4/29 5:11:53] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce DB1CF7505FF755DB, IP: 177.10.1.1 -> 10.10.2.2, UDP: 4342 -> 4342
2018/4/29 5:11:53] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.2.2 UDP: 4342 -> 4342
2018/4/29 5:11:59] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 46a389df5afdf55f, IP: 10.10.4.1 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:11:59] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.6.0/24
2018/4/29 5:11:59] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 3/10 255/0, addr: 10.10.4.1
2018/4/29 5:11:59] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 46a389df5afdf55f, IP: 177.10.1.1 -> 10.10.4.1, UDP: 4342 -> 4342
2018/4/29 5:11:59] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.4.1 UDP: 4342 -> 4342
2018/4/29 5:12:10] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 864884787bfd5eaf, IP: 10.10.1.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:12:10] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.2.0/24
2018/4/29 5:12:10] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.1.2
2018/4/29 5:12:10] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 864884787bfd5eaf, IP: 177.10.1.1 -> 10.10.1.2, UDP: 4342 -> 4342
2018/4/29 5:12:10] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.1.2 UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce e84e8a15ff5d63e, IP: 10.10.3.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.5.0/24
2018/4/29 5:12:15] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.3.2
2018/4/29 5:12:15] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce E84E8A15FF5D63E, IP: 177.10.1.1 -> 10.10.3.2, UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.3.2 UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce fbade1a15ffdf7ff, IP: 10.10.3.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.4.0/24
2018/4/29 5:12:15] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.3.2
2018/4/29 5:12:15] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce FBADE1A15FFDF7FF, IP: 177.10.1.1 -> 10.10.3.2, UDP: 4342 -> 4342
2018/4/29 5:12:15] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.3.2 UDP: 4342 -> 4342
2018/4/29 5:12:53] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 62dzf6945ff7df77, IP: 10.10.2.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:12:53] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.1.0/24
2018/4/29 5:12:53] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.2.2
2018/4/29 5:12:53] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 62dzf6945ff7df77, IP: 177.10.1.1 -> 10.10.2.2, UDP: 4342 -> 4342
2018/4/29 5:12:53] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.2.2 UDP: 4342 -> 4342
2018/4/29 5:12:59] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 20618a1b5afd777b, IP: 10.10.4.1 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:12:59] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.6.0/24
2018/4/29 5:12:59] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 3/10 255/0, addr: 10.10.4.1
2018/4/29 5:12:59] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 20618A1B5AFD777B, IP: 177.10.1.1 -> 10.10.4.1, UDP: 4342 -> 4342
2018/4/29 5:12:59] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.4.1 UDP: 4342 -> 4342
2018/4/29 5:13:10] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce b74784b47bfd55f7, IP: 10.10.1.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:13:10] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.2.0/24
2018/4/29 5:13:10] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.1.2
2018/4/29 5:13:10] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce B74784B47BFD55F7, IP: 177.10.1.1 -> 10.10.1.2, UDP: 4342 -> 4342
2018/4/29 5:13:10] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.1.2 UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 63a9e8e55ff5ff6f, IP: 10.10.3.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.5.0/24
2018/4/29 5:13:15] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.3.2
2018/4/29 5:13:15] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 63A9E8E55FF5FF6F, IP: 177.10.1.1 -> 10.10.3.2, UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.3.2 UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce 97e8e1e55ffdf7ff, IP: 10.10.3.2 -> 177.10.1.1, UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Mapping-record -> ttl: 10 loc-count: 1 action: no-action auth: 1 map-version: 0 eld: 192.168.4.0/24
2018/4/29 5:13:15] DEBUG: Locator-record -> flags: L=1,p=0,R=1, p/w: 2/10 255/0, addr: 10.10.3.2
2018/4/29 5:13:15] DEBUG: Map-Notify-> flags:ir, record-count: 1, nonce 97E8E1E55FFDF7FF, IP: 177.10.1.1 -> 10.10.3.2, UDP: 4342 -> 4342
2018/4/29 5:13:15] DEBUG: Sent control message IP: 177.10.1.1 -> 10.10.3.2 UDP: 4342 -> 4342
2018/4/29 5:13:53] DEBUG: Received Map-Register -> flags:pirM record-count: 1 nonce f638f7d85ff775f7, IP: 10.10.2.2 -> 177.10.1.1, UDP: 4342 -> 4342

```

5. SDN Test Plan

5.1. Test Application

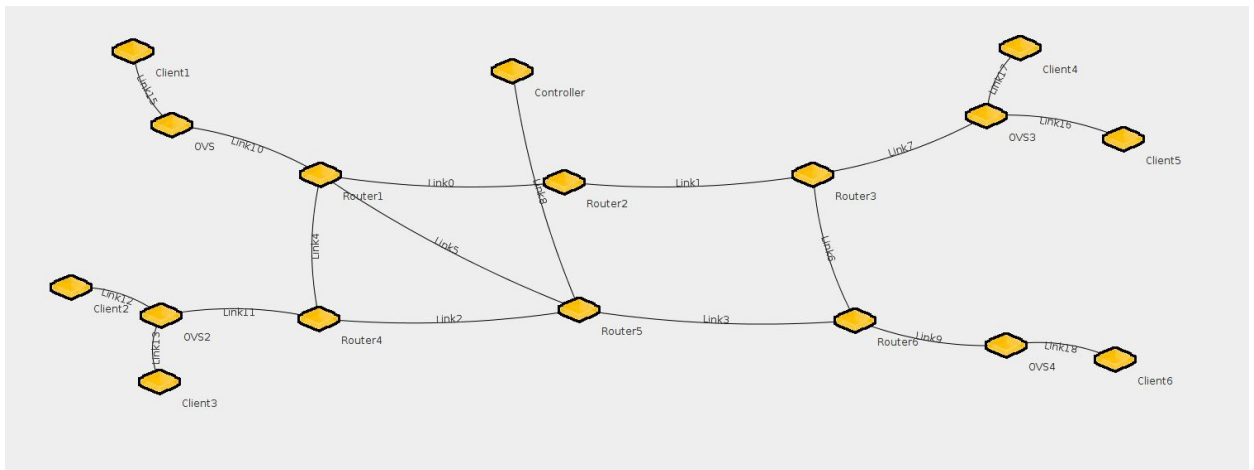


Figure 14: SDN topology

Match: The source and destination IP addresses of the flow are the EIDs.
 Action: Replace the source and destination EIDs with the RLOC (OVS switch) source and destination IP addresses. The test plan below refers to

the match and action definitions above.

Table 4: SDN Test Plan

Case	Setup	Action	Expected Observation	Recorded Observation
1	OVS switches are running, MS/MR application is running in the controller	Ping from source EID to destination EID	No mapping exists, and no route to core network exists, ICMP and ARP messages are sent to the controller	Ping fails
2	MS/MR application in the controller is not running at the beginning, source and destination OVS switches are running	Ping from source EID to destination EID	No flows are added into the Map server application. The source OVS switch's packet in message to the map server via the controller is not served and there is neither a match nor an action. Source EID should not be able to reach destination EID	Ping fails.
3	MS/MR turned off after one EID pings at least one other EID connected to an OVS switch after mappings are manually added	Ping from source EID to destination EID	Local cache obtained from MS/MR would still allow pings from the source EID to that particular EID, other pings would likely fail	Pings to the original source destination pair will work. Pings to other hosts will fail
4	MS/MR turned off after one EID pings all other EIDs connected to the ETRs	Ping from source EID to destination EID	Local cache obtained from MS/MR would still allow pings from the source EID to still work with all other EIDs till timeout	All pings work

Table 5: SDN Test Plan - Contd2

Case	Setup	Action	Expected Observation	Recorded Observation
5	Simulate Client Mobility by changing EID of destination while pinging	Continuously Ping from source EID to destination EID after adding manually flows to both the datapath before moving and the map server application is restarted with mapping of the changed RLOC	ICMP packets are received(after packet loss) at the OVS switch responsible for the moved destination EID. Indicating successful RLOC-EID re-mapping.	Incomplete

FAILED PING DUE TO CONTROLLER APPLICATION DOWN:

```

root@Client1: ~
root@Client1:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
^C
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3999ms
root@Client1:~#

root@Client2: ~
root@Client2:~# tcpdump -i ens6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens6, link-type EN10MB (Ethernet), capture size 262144 bytes

```



```
root@OV51:~# tcpdump -i 7
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens8, link-type EN10MB (Ethernet), capture size 262144 bytes
02:38:16.013497 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 1, length 64
02:38:16.014211 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 1, length 64
02:38:17.012856 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 2, length 64
02:38:17.013114 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 2, length 64
02:38:18.012798 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 3, length 64
02:38:18.013694 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 3, length 64
02:38:19.012759 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 4, length 64
02:38:19.013881 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 4, length 64
02:38:20.012789 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 5, length 64
02:38:20.013118 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 5, length 64
02:38:21.020073 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:21.021891 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42
02:38:22.019870 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:22.028664 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42
02:38:23.019903 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:23.020768 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42

root@OV52:~# tcpdump -i ens6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens6, link-type EN10MB (Ethernet), capture size 262144 bytes
02:38:16.013492 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 1, length 64
02:38:16.014210 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 1, length 64
02:38:17.012847 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 2, length 64
02:38:17.013110 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 2, length 64
02:38:18.012784 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 3, length 64
02:38:18.013691 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 3, length 64
02:38:19.012749 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 4, length 64
02:38:19.013077 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 4, length 64
02:38:20.012779 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 5, length 64
02:38:20.013115 IP 192.168.1.2 > 192.168.1.1: ICMP echo request, id 2556, seq 5, length 64
02:38:21.020068 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:21.021891 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42
02:38:22.019860 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:22.028662 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42
02:38:23.019892 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:38:23.020759 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 42
```

WORKING PING AFTER CONTROLLER APPLICATION RUNS:

```
root@Client1:~# Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

44 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Sun Apr 29 17:21:23 2018 from 152.7.255.194
root@Client1:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=63 time=27.1 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=63 time=24.9 ms
64 bytes from 192.168.2.2: icmp_seq=3 ttl=63 time=24.2 ms
64 bytes from 192.168.2.2: icmp_seq=4 ttl=63 time=22.6 ms
^C
-- 192.168.1.1 ping statistics --
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/ndev = 22.694/24.786/27.173/1.609 ms
root@Client1:~#

root@Client2:~# tcpdump -i ens6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens6, link-type EN10MB (Ethernet), capture size 262144 bytes
02:22:00.385643 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 1, length 64
02:22:00.385686 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 1, length 64
02:22:01.395494 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 2, length 64
02:22:01.395587 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 2, length 64
02:22:02.385806 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 3, length 64
02:22:02.385821 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 3, length 64
02:22:03.386482 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 4, length 64
02:22:03.386505 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 4, length 64
02:22:05.397760 ARP, Request who-has 192.168.2.2 tell 192.168.2.1, length 42
02:22:05.397788 ARP, Reply 192.168.2.2 is-at fa:16:3e:00:5d:09 (oui Unknown), length 28

root@Controller:~/our-attempt#
Switch Packet in
Handling ARP
This is an ARP reply
In arp rewrite function
New ARP next hop is: 172.10.1.2
ARP from 195794875627329 with in-port: 1 DST Changed to ----> 172.10.1.2
Sending packet out through port ----> 2
ARP was handled
Switch Packet in
Handling ARP
This is an ARP reply
Sending packet out through port ----> 1
```

```
root@OV51:~# tcpdump -i ens8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens8, link-type EN10MB (Ethernet), capture size 262144 bytes
02:22:00.326986 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 1, length 64
02:22:01.327045 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 2, length 64
02:22:01.340030 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 2, length 64
02:22:02.328628 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 3, length 64
02:22:02.340517 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 3, length 64
02:22:03.329856 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 4, length 64
02:22:03.340901 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 4, length 64
02:22:05.356088 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:22:05.368444 ARP, Reply adsl-172-10-1-1.dsl.sndg02.sbcglobal.net is-at fa:16:3e:00:71:db (oui Unknown), length 42

root@OV51:~# tcpdump -i ens8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens8, link-type EN10MB (Ethernet), capture size 262144 bytes
02:22:00.327003 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 1, length 64
02:22:00.340473 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 1, length 64
02:22:01.327050 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 2, length 64
02:22:01.340034 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 2, length 64
02:22:02.328633 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 3, length 64
02:22:02.340517 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 3, length 64
02:22:03.329862 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 4, length 64
02:22:03.340905 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 4, length 64
02:22:05.356093 ARP, Request who-has adsl-172-10-1-1.dsl.sndg02.sbcglobal.net tell adsl-172-10-1-2.dsl.sndg02.sbcglobal.net, length 42
02:22:05.368445 ARP, Reply adsl-172-10-1-1.dsl.sndg02.sbcglobal.net is-at fa:16:3e:00:71:db (oui Unknown), length 42

*** System restart required ***
Last login: Mon Apr 30 02:00:08 2018 from 207.244.78.75
root@OV52:~# tcpdump -i ens6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens6, link-type EN10MB (Ethernet), capture size 262144 bytes
02:22:00.524092 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 1, length 64
02:22:00.536245 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 1, length 64
02:22:01.523752 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 2, length 64
02:22:01.536148 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 2, length 64
02:22:01.535342 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 3, length 64
02:22:01.536614 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 3, length 64
02:22:01.526524 IP adsl-172-10-1-1.dsl.sndg02.sbcglobal.net > adsl-172-10-2-2.dsl.sndg02.sbcglobal.net: ICMP echo request, id 2487, seq 4, length 64
02:22:01.536896 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 4, length 64
02:22:05.536540 ARP, Request who-has adsl-172-10-2-2.dsl.sndg02.sbcglobal.net tell adsl-172-10-2-1.dsl.sndg02.sbcglobal.net, length 42
02:22:05.549129 ARP, Reply adsl-172-10-2-2.dsl.sndg02.sbcglobal.net is-at fa:16:3e:00:5d:09 (oui Unknown), length 42

*** System restart required ***
Last login: Sun Apr 20 17:25:20 2018 from 152.7.255.194
root@OV52:~# tcpdump -i ens8
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens8, link-type EN10MB (Ethernet), capture size 262144 bytes
02:22:00.530593 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 1, length 64
02:22:00.531196 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 1, length 64
02:22:01.530590 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 2, length 64
02:22:01.531232 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 2, length 64
02:22:02.531037 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 3, length 64
02:22:02.531532 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 3, length 64
02:22:03.531686 IP 192.168.1.2 > 192.168.2.2: ICMP echo request, id 2487, seq 4, length 64
02:22:03.532300 IP 192.168.2.2 > 192.168.1.2: ICMP echo reply, id 2487, seq 4, length 64
02:22:05.542776 ARP, Request who-has 192.168.2.2 tell 192.168.2.1, length 42
02:22:05.543473 ARP, Reply 192.168.2.2 is-at fa:16:3e:00:5d:09 (oui Unknown), length 42
```

6. Demo Plan

6.1. Demo Plan Topology

The traditional topology consists of: 6 clients C1, C2, C3, C4, C5, C6, four tunnel routers ITR, ITR2, ETR, ETR2, and a core network. The SDN topology consists of: A controller with the Map Server/Resolver application, 6 clients C1, C2, C3, C4, C5, C6, four OVS switches OVS, OVS2, OVS3, OVS4, and a core network.

Table 6: Demo Plan: Traditional implementation

Case	Scenario	Expected Observation	Conclusion
1.	MS/MR and xTRs are in running state. ICMP packets are sent from source EID to destination EID	ETR should send registration messages to MS/MR. ITR should send map-request messages to MS/MR. MS/MR must respond to ITR with mapping information in the map-reply message. LISP tunneling is established. Debug messages can also be observed in the xTRs. Unencapsulated and LISP encapsulated messages can be observed on the xTR interfaces. The map-request and reply messages can be observed on ITR interface and the outgoing interface of MS/MR respectively. Tshark can be used to observe the captured packets. The ICMP packets should successfully reach from source EID to destination EID.	ITR updates its local cache with the mapping information received from MS/MR. ITR sends LISP encapsulated messages with the RLOCs in the header. Packet is decapsulated at ETR and forwarded to destination EID.

Table 7: Demo Plan: traditional implementation

Case	Scenario	Expected Observation	Conclusion
2.	Client Mobility by changing EID	Client moves from old RLOC to new RLOC. New ETR registers updated EID-RLOC mapping with the MS/MR. ITR sends map-request to MS/MR after cache expiry. MS/MR responds to ITR with updated mapping information. Map-request/reply messages can be observed at ITR's and MS/MR's interface respectively. LISP tunnel is established. LISP encapsulated messages can be observed at the ITR's outgoing interface.	The LISP encapsulated messages consist of the new RLOC in the header. ETR decapsulates the LISP packets and forwards the unencapsulated packets to the destination EID. Ping is successful between the source EID and the moved client.
3.	MS/MR and xTRs are in running state. ICMP packets are sent from source EID to destination EID by pinging. Then MS/MR is killed	Since ITR's and ETR's are now aware of their mappings regardless of whether the map resolver is running or not. The path the packet takes should still be with the same tunnel capabilities and the Ping works	xTR's have local cache data that they will maintain for 10 mins

Table 8: Demo Plan: SDN implementation

Case	Scenario	Expected Observation	Conclusion
1.	Controller is off but OVS's are running Pinging from source EID to destination EID	Ping fails	Since there are no flows packets do not get re-written to RLAC address space and ping fails .
2.	MS/MR application in the controller and OVS switches are in running state. Flow entries are added manually in the MS/MR application. ICMP packets are sent from source EID to destination EID	Source OVS switch sends packet-in messages to MS/MR application via the controller. MS/MR application checks for EID entry. SDN controller responds to OVS with updated flow table entry. OVS switch performs match and action.	IP rewrite of RLOC address takes place. Flow reaches destination OVS. IP rewrite to EID takes place. EID receives the packets. Ping is successful from source EID to destination EID.
3.	Client Mobility by changing EID	Client moves from old RLOC to new RLOC. Manual updation of EID-RLOC mapping at the MS/MR application updated flow table entries installed at destination OVS switch. Packet-in message from source OVS to controller returns updated EID-RLOC mapping as new flow table entry.	IP rewrite of new RLOC address takes place. Flow reaches new destination OVS. IP rewrite to EID takes place. EID receives the packets. Ping is successful from source EID to destination EID.

7. Self-Study

7.1. Base Case:

- Clients are connected to the xTR's, but LISP tunnel feature is turned off, i.e they act as regular routers.

- No Map resolver is present in the system.

7.2. *Characteristics to observe*

- Direct end-to-end client pings will be observed.
- Clients and Routers(xTR's with tunnel off) will require additional routing table entries for packets to travel end-to-end. Pinging end-to-end is observed for this scenario
- xTR's have their tunnel featured turned on. And the additional routing table entries are removed. Ping will observed for such a scenario. Be reminded that a route does not exist between the private address space of the clients to the public address space of the network.
- xTR's are activated in tunnel mode, map resolver is not active, and a ping is observed.
- MS/MR is activated along with xTR's being in LISP tunnel mode. Ping between clients is observed.
- Client with a an EID is moved across to an xTR with a diff t RLOC. A 10 minute wait is introduced to trigger xTR cache timeouts and a ping is observed from a source client to a client with a diff t RLOC.

7.3. *Range of scenarios to investigate*

- The separation of private and public address space, before and after LISP tunneling should be investigated.
- EID mobility, i.e. an EID associated with a certain RLOC detaches itself and moves to a diff t RLOC should be able to retain its connections after possible packet loss.

8. Per Member Learning Experiences

Learning experiences of Ragavi Swarnalatha Raman:

- LISP working.
- Better understanding of how routers and OVS switches work.
- Hands on experience with OOR libraries and configuration.

- Understanding how to use OVS as a layer 3 device.
- Configuration and writing applications for Ryu controller
- Deep understanding of how ARP and ICMP gets handled in a SDN controller application

Learning experiences of Srikanth Gopalakrishnan:

- LISP implementation knowledge, and how it's tunnel features work.
- Hands on experience with OOR libraries and configuration.
- Synchronize packet in's on differentt OVS bridges across the network using the RYU controller.
- Configuring and observing the working of OSPF using the BIRD routing daemon.
- How packets from say 'hello' messages from OSPF could break an SDN's packet in logic
- Deep understanding of how ARP and packets in general work in an SDN setting, and how to manipulate them.

Learning experiences of Aparajita Bagchi:

- Hands on experience and in-depth knowledge of how LISP works.
- Better understanding of how routers and OVS switches work.
- Hands on experience of creating a large network and using linux to facilitate this network.
- Understanding how to use OVS as a layer 3 device.
- Experience with Ryu controller and OpenFlow
- Using open-source tools such as BIRD routing daemon and Open Overlay Router for encapsulation and decapsulation.

9. What went as expected

9.1. Traditional implementation

- Able to successfully create the topology on ExoGENI
- Configuration of core routers using BIRD routing daemon and observing the correct routes added in each router of the core network
- Correctly using the OOR encapsulation/decapsulation tool in the xTRs and MS/MR.
- Successful configuration of xTRs and MS/MR in the xTR and MS mode of OOR configuration respectively
- Sending ICMP packets from a host in one private subnet to a host in the private subnet across the core network. The end-hosts did not have any other routing entries other than to the private 192.168.0.0/16 subnet. This proved that the packets were successfully encapsulated and decapsulated at the tunnel routers using LISP.
- Node Mobility from one subnet to another. One EID is able to reach another EID even if it is moved to another network. Communication does not break.

9.2. SDN Implementation

- Successfully created a test topology on ExoGENI
- Configuration of the OVS switches to behave like routers by using two bridges inside the switch.
- Programming the Ryu controller such that ICMP packets are sent from one EID to another while the controller is running

10. What did not go as expected

- Could not scale the working test topology to the actual large topology.
- Node mobility did not work in the SDN test topology.

11. Project Package

This drive link consists of the codes and scripts used in the project, reports and demo slides:

<https://drive.google.com/drive/folders/1lH7xhThXSdrr8jOSyoYp9b46gpvCZnvy>

12. References

1. RFC 6830: The Locator/ID Separation Protocol (LISP)
2. Open Overlay Router : <https://openoverlayrouter.org/>
3. Bird Internet Routing Daemon : <http://bird.network.cz/>
4. LispMob: <https://github.com/LISPmob/lispmob>

