LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: December 20, 2018

S. Barkai
Fermi Serverless
D. Farinacci
lispers.net
D. Meyer
1-4-5.net
F. Maino
V. Ermagan
A. Rodriguez-Natal
Cisco Systems
A. Cabellos-Aparicio
Technical University of Catalonia

June 18, 2018

LISP Based FlowMapping for Scaling NFV
draft-barkai-lisp-nfv-12

Abstract

This draft describes an RFC 6830 Locator ID Separation Protocol (LISP) based distributed flow-mapping-fabric for dynamic scaling of virtualized network functions (NFV). Network functions such as subscriber-management, content-optimization, security and quality of service, are typically delivered using proprietary hardware appliances embedded into the network as turn-key service-nodes or service-blades within routers. Next generation network functions are being implemented as pure software instances running on standard servers - unbundled virtualized components of capacity and functionality. LISP-SDN based flow-mapping, dynamically assembles these components to whole solutions by steering the right traffic in the right sequence to the right virtual function instance.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Table of Contents

1. Introduction

This draft describes an RFC 6830 Locator ID Separation Protocol (LISP) based distributed flow-mapping-fabric for dynamic scaling of virtualized network functions (NFV).[RFC6830]Network functions such as subscriber-management, content-optimization, security and quality of service, are typically delivered using proprietary hardware

appliances embedded into the network as turn-key service-nodes or service-blades within routers.

This monolithic service delivery method increases the complexity of service roll-out and capacity planning, limits providers' choices, and slows down revenue generating service innovation. Next generation network functions are being implemented as pure software instances running on standard servers - unbundled ("googlized") virtualized components of capacity and functionality. Such a component based model opens up service provider networks to the savings of elasticity and open architecture driven innovation. However this model also presents the network with the new challenges of assembling components, developed by 3rd parties, into whole solutions, by forwarding the right traffic to the right function-block at the right sequence.

While this is possible, to some extent, by traditional virtual networking - virtual bridges(vBridges) and virtual-routing-forwarding (VRF) - these mechanisms are relatively static and require complex and intensive configuration of network interfaces, while elastic components are not network topology bound. Software-defined-networks, (SDN) flow based models are much more dynamically programmable but are also very centralized and hence have limited scale and resiliency. By enhancing SDN models with RFC6830 overlay model we offer a best fit to dynamic assembly of virtualized network functions in the service-providers data-centers and distribution-centers.

2. Terminology

The following terms are used to describe a LISP based implementation of Software-Defined Flow-Mapping-Fabric for NFV:

- LISP-SDN - is an enhancement to the basic SDN model of (1) hop-to-hop (2) push-down flow-commands (3) by concentrated-controller.. to a LISP based architecture of (1) distributed-overlay e.g. SDN over IP (2) based on a pull-publish-subscribe actions from xTR-edges up.. (3) to a global mapping service. A mapping service scaled by and connected over the IP underlay network. LISP-SDN lookup operation details are covered in [I-D.rodrigueznatal-lisp-multi-tuple-eids].

- Virtualized Network Function (VNF) - is a process instance with an EID and RLOC that performs a defined set of inline network functions. a VNF can be software on a virtual-machine (VM) performing a function like multimedia signaling, mobility management, content caching or streaming, security, filtering, optimization, etc. A VNF class type and VNF instance capacity,

load, and location are attributes that can be resolved by the LISP-SDN mapping service.

- Client-Flow - is a sequence of packets that corresponds to a specific communication thread or network conversation between a client application and a network service. Client-flows are typically processed by various in-network functions either as the end service side to the network conversation, or as middle-box functionality.

- SDN-xTR - is a LISP xTR that supports the lookup defined in [I-D.rodrigueznatal-lisp-multi-tuple-eids]. It classifies traffic into application flows, maps, encapsulates, and decapsulates flows in order to emerge a flow-mapping solution - along with a collection of the SDN-xTR elements, and the LISP-SDN mapping service.

- SDN-Overlay - is the network formed by the collection of inter-connected SDN-xTR

- SDN-Underlay - is the IPvN network connecting SDN-xTRs

- SDN-Outerlay (interim name)- is the collection of networks and interfaces aggregated by the various SDN-xTRs connecting VNFs and Client-flows coming from access networks or the Internet.

- Flow-Rule - is a set of pattern tuples that match any part of a packet header and is used to classify packets into flows as well as trigger forwarding actions such as encapsulation / decapsulation, network address translation (NAT), etc. We differentiate between exact-match rules (many) which include an exact set of tuple bits, and best match rules (fewer) which contain both tuple bits and wild-cards "*".

- Virtual IP (VIP) - is an IP address or EID that identifies a function rather then a specific destination. For example all the encapsulated client-flow traffic sent from a base-station eNodeBs over a transport network, can have as destination a VIP which represents in a given LISP-SDN solution, the function mobile-gateway or PGW, and not any specific destination.

- Flow-Affinity - is the association between a client-flow and a VNF instance. VNF logic will typically create long-lived (minutes) in memory states in order to perform its functions. Therefore once an affinity is established it is best to keep it for as long as possible in order not to stress or break the VNF application.

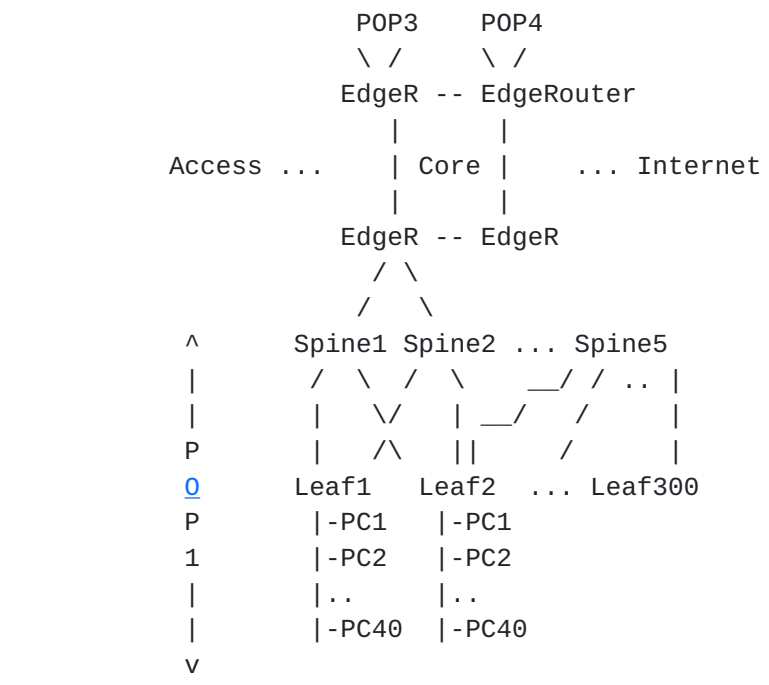
3. Connectivity Model

The basic connectivity model used to assemble VNFs into whole solution is the flow-mapping-fabric. Unlike topological forwarding which is based on source-subnet >> routed hop by hop >> destination-subnet, a flow-mapping-fabric maps, forwards and "patches" flows by identity directly to the end systems. The identities used for the flow-mapping-fabric are those associated with the client-flows e.g. Subscriber ID, phone number, TCP port, etc. and those associated with the VNF e.g. the type, location, physical address, etc. the flow-mapping-fabric is implemented as a LISP-SDN overlay, over in-place IP underlay, assembling outerlay flows into solutions. Bellow are basic assumptions regarding the Underlay, Outerlay, and Overlay in the solution:

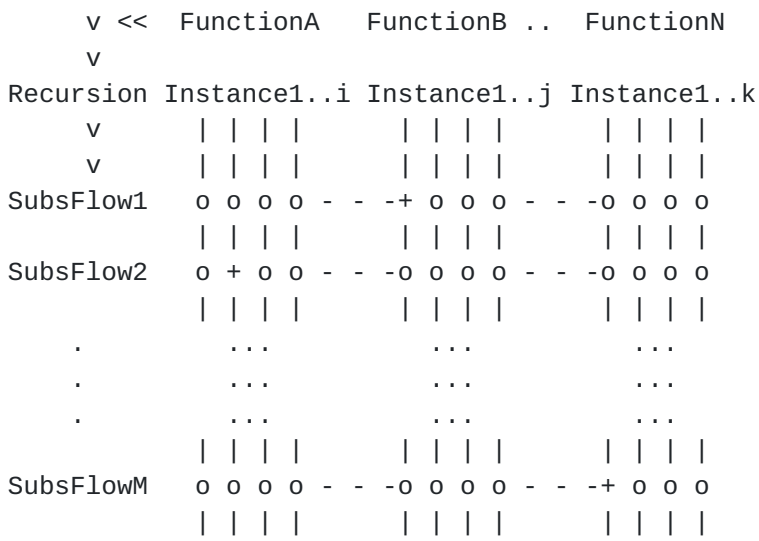
- The underlying physical network is assumed to be topology based and implemented using standard bridging and routing. Conventional design principles are applied in order to achieve both capacity and availability of connectivity. Typical examples of underlays include spine-leaf switching for clustering server racks, and, core-edge routing inter-connecting server clusters across points of presence. Edge networks are also used to connect to access networks and Internet.

- The flow-mapping-fabric maps outerlay client-flows to VNFs. This enables assembly, scaling, balanced high-utilization, massive concurrency, and hence, performance of NFVs. By mapping each client-flow to the correct functional instance the system engages as many VNF components as are available, scaled within and across data-centers. Applied recursively client-flow mapping can chain a sequence of VNF components to make up an end-to-end service.

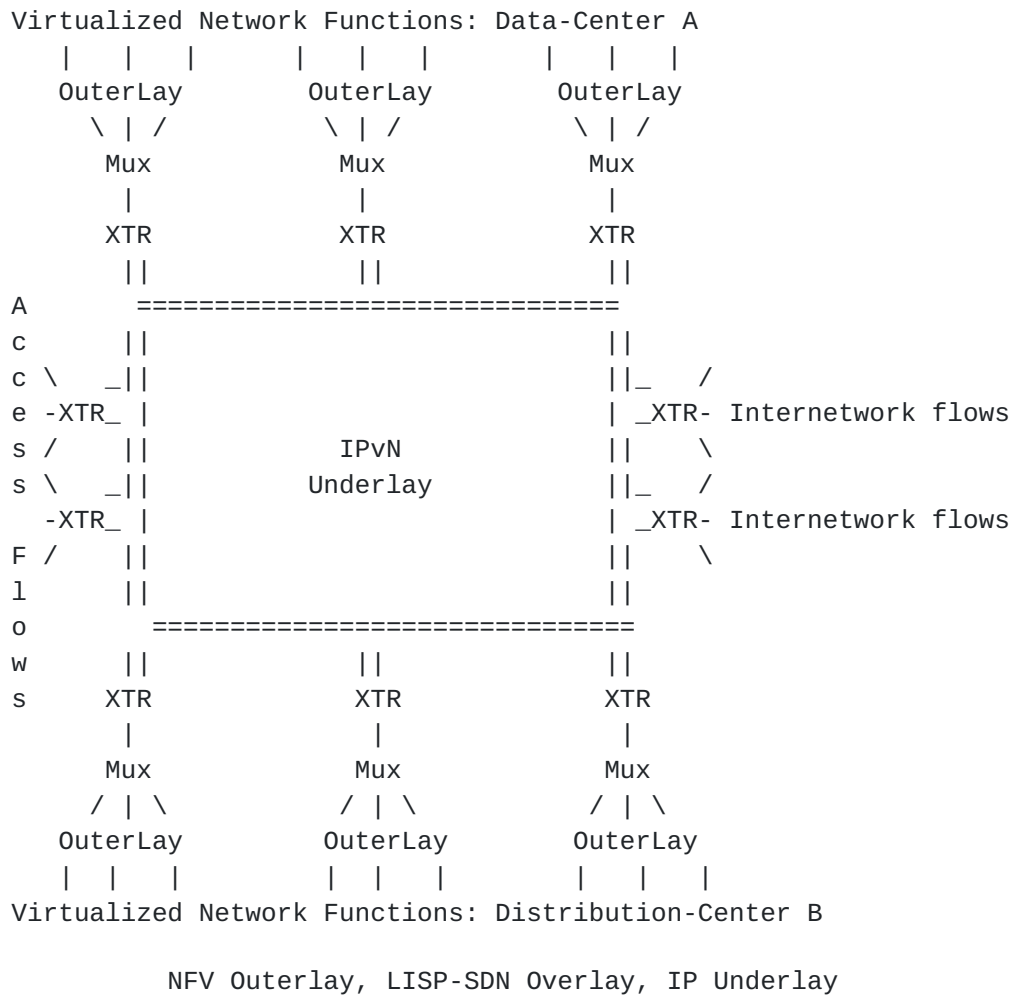
- The overlay network is based on location-identity-separation and forms a virtualization indirection ring around spines and cores. The overlay edges aggregate outerlay client-flows and VNFs. Outerlay flows are classified, mapped, and encapsulated over the edge through the underlay interfaces and are transported to the right identity's locations.



Core-Edge Spine-Leaf Underlays



Flow-Mapping-Fabric



NFV Outerlay, LISP-SDN Overlay, IP Underlay

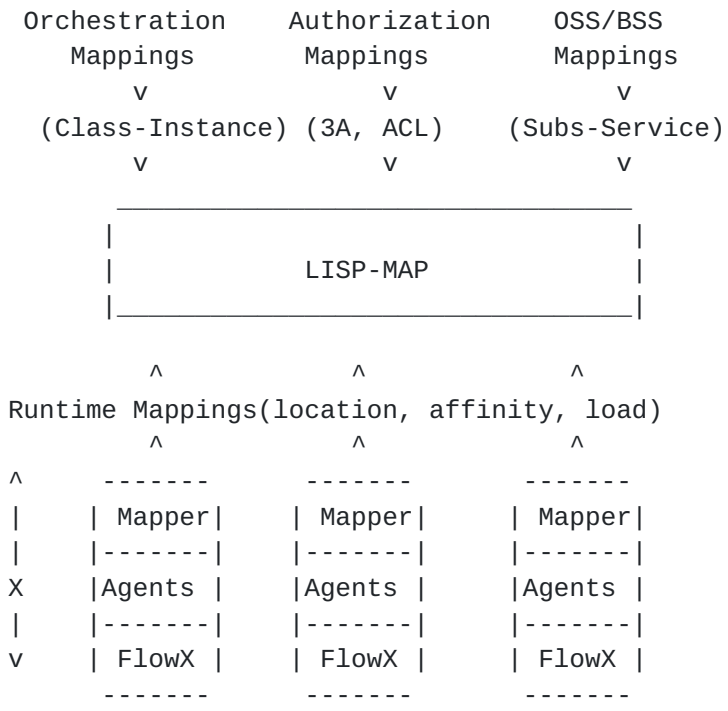
4. Flow-Mapping Elements

In order to implement NFV Flow-Mapping-Fabric using LISP-SDN We use the following components and capabilities:

1. Flow-Switching: is a component within an SDN-xTR and contains a set of n-tuple flow-rules matched against each packet in order to separate it to (LOCALLY defined) sequences representing flows. Flows are either Encapsulated into the Overlay, decapsulated to the Outerlay, or forwarded to SDN-xTR Control Agents.

2. Control-Agents: are software processes running in SDN-xTRs and are invoked for each flow where an exact match was not present in the Flow-Switching. The default "catch-all" Flow-Handler maps IP flows to locations and gateways based on RFC 6830. Protocol and application specific handlers can be loaded into the SDN-xTR for handling specific mapping and AFFINITY requirements of network functions. Examples of such protocols and applications can be SIP, GTP, S1X etc.

3. Global-Mapping: is how GLOBALLY significant key-value mappings is translated to LOCALLY defines flow masks and encapsulation actions. Examples of such mappings include: Map a functional instance ID to a function class ID; map subscriber-application ID to virtual function instance ID; map instance ID to location; instance to health, load, tenant; etc.



5. Day-in-life of a Mapped Flow

Let us walk through detailed steps of the use of RFC6830 and LISP architecture in order to perform resource virtualization and flow assignment to virtual function instances.

At a high level, when a client-flow packet first arrives at a SDN-xTR on the edge of the LISP overlay, the SDN-xTR must decide on a VNF instance that is best suited to service this flow, assign this flow to the selected VNF, and encapsulate this flow to the RLOC of the selected virtual function instance.

To select the best suited VNF instance, the SDN-xTR queries the Mapping System with the extracted identity parameters, both the client and the function EIDs, and receives the list of all VNF instances that represent that Function along with their RLOC and health-load attributes. The SDN-xTR runs local algorithms on the returned set to select the best suited virtual function instance.

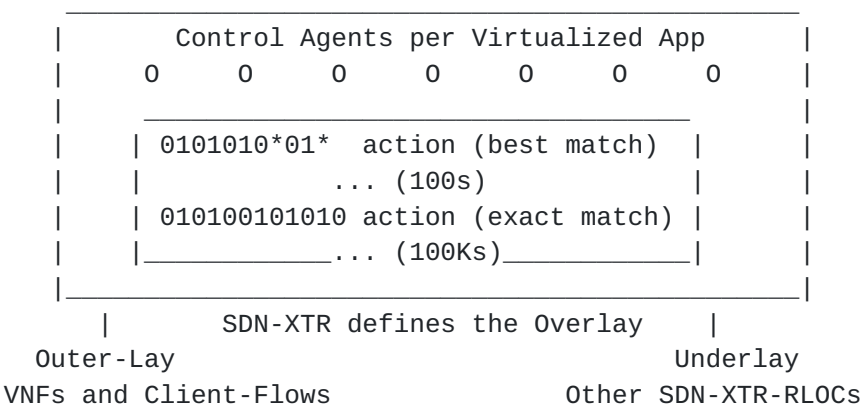
Once selected, the SDN-xTR stores (registers) the assignment of this flow to the associated VNF instance in the Mapping System. This assignment is referred to as the Affinity for this flow. The SDN-xTR also programs an exact match flow rule in its data-plane, so future packets from this flow will be mapped to the same EID-RLOC.

In the following subsections We describe this process in more detail.

5.1. XTR Flow Edge

SDN-xTR locations define the boundary of the virtual network. For the purpose of LISP-SDN flow-mapping-fabric We refer to the bellow SDN-XTR generic reference architecture. Actual vendor implementations may vary, but most likely will include similar components and structure. The SDN-XTR includes:

- Mux-DeMux: Interfaces to the Underlay and Outerlay
- Flow-Rules: Patterns-Actions, Exact / Best Match, Encap-Decap
- Control-Agents: Application specific flow-handlers registered in the Flow-Rules



SDN-XTR Reference Architecture

SDN-XTR Flow Switching works as follows:

1. For traffic from the Outerlay of THIS xTR that has an exact match of all the source-dest-tags.. n-tuples, the packets are processed by rule actions including encapsulation to the RLOC of the xTR which aggregates the relevant function instance to which this flow is mapped to.

2. For traffic from the Underlay that has an exact match of all the source-dest-tags.. n-tuples, the packets are processed by rule actions including decapsulation and forwarding to the Outerlay of THIS xTR.

3. Traffic from the Outer-Lay or Underlay that does NOT have an exact match of all the source-dest-tags.. tuples required for normal forwarding, packets are forwarded to the control agent registered in the best-matching rule.

SDN-XTR Control Agents work as follows:

1. Mapping agent type and application scope is defined by the best match entries that point to it. Control agents will typically self-register in the flow-switch. XTR control-agents can register to an existing best-match rule, or instantiate a new one.

2. Typical rule-patterns are pattern-scoped by an agent registration, and can include: protocol or service type header indications; specific virtual IP addresses (VIP) that represent a service and not a specific destination; a specific source and wild-card destination; or vice versa.

3. Mapping agents work with the LISP-SDN mapping service in order to establish a global context and local considerations for mapping decision. The goal of the agents' decision is ultimately to provision the correct exact-match rule and actions that will offload the flow-packets to flow-switching described above.

The SDN-xTR control agents query the LISP-SDN Mapping System with the flow attributes including the destination VIP, as followes:

Mapping System Lookup: Map-Request (Client identity, Function-EID)

Two outcomes are possible based on whether an affinity already exists for this flow (flow has already been assigned to a virtual function instance):

o Outcome A:

  * If an affinity already exists in the Mapping System, the Mapping System returns the locator address (RLOC) associated with the Function-Instance-EID that the (Client-EID, Function-EID) is mapped to.

  * Map-Reply: ((Client-EID, Function-EID) -> Function-Instance-RLOC)

  * In this case the Mapping System also subscribes the SDN-xTR to the Function-Instance-EID, and to the (Client-EID, Function-EID) flow in order to receive updates in case of changes on these entries. Examples of these changes are change of RLOC for the Function-Instance-EID (specially if this is a virtual application), or change of affinity for (Client-EID, Function-EID) to another Function-Instance-EID.

  * After receiving the Map-Reply form the Mapping System, the SDN-xTR programs an exact match for the flow in the xTR data-plane.

- Outcome B:

  * If there is no affinity previously stored, the Mapping System returns a list of Records, including one Record per each instance of the Function-EID, with their associated RLOCs and flags (weight, priority).

  * Map-Reply: (client EID, Function-Instance-Record 1, Function-Instance-Record 2...)

  * the SDN-xTR then selects the best suited Function-Instance-EID for this flow based on local algorithms, and registers the affinity in the Mapping System. The Mapping System stores the affinity and subscribes the SDN-xTR to the affinity and to the Function-Instance-EID in the affinity, so that SDN-xTR would receive updates if any of these changes.

  * Map-Register ((Client-EID, Function-EID) -> Function-Instance-EID)

- Note: An SDN-xTR must be able to query for the list of App-Instance-Records even if an affinity already exists. For this purpose a flag is required in the Map-Request to indicate whether xTR wants this info or not. We can overload the M bit in Map-Request, or allocate a new bit for this.

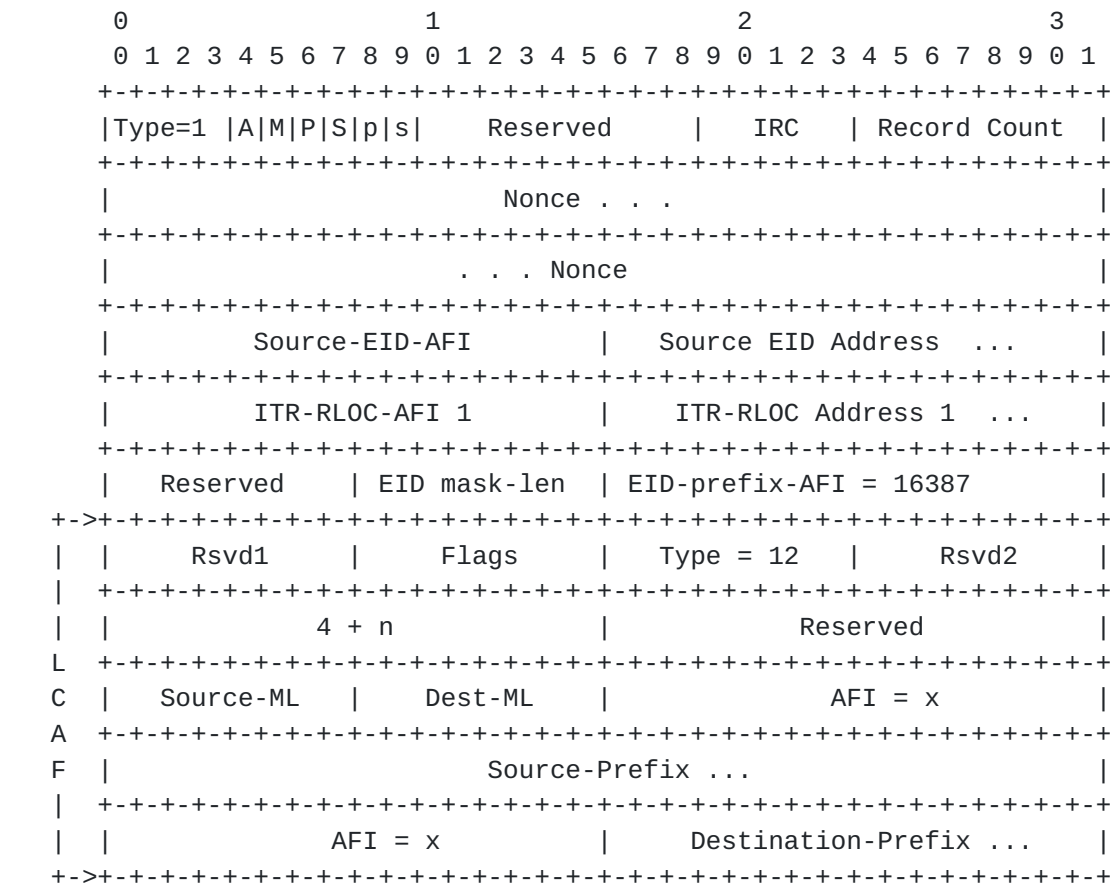
5.2. Map Resolvers-Servers

5.3. XTRs-Mappers Scaling

6. Message Formats

This section specifies the packet formats used throughout the flow-mapping process explained above. The lookup is based on what is described in [I-D.rodrigueznatal-lisp-multi-tuple-eids].

A Map-Request is used with a 2-Tuple Src/Dst LCAF to query the Mapping System for the affinity or list of virtual function instance records for this flow.

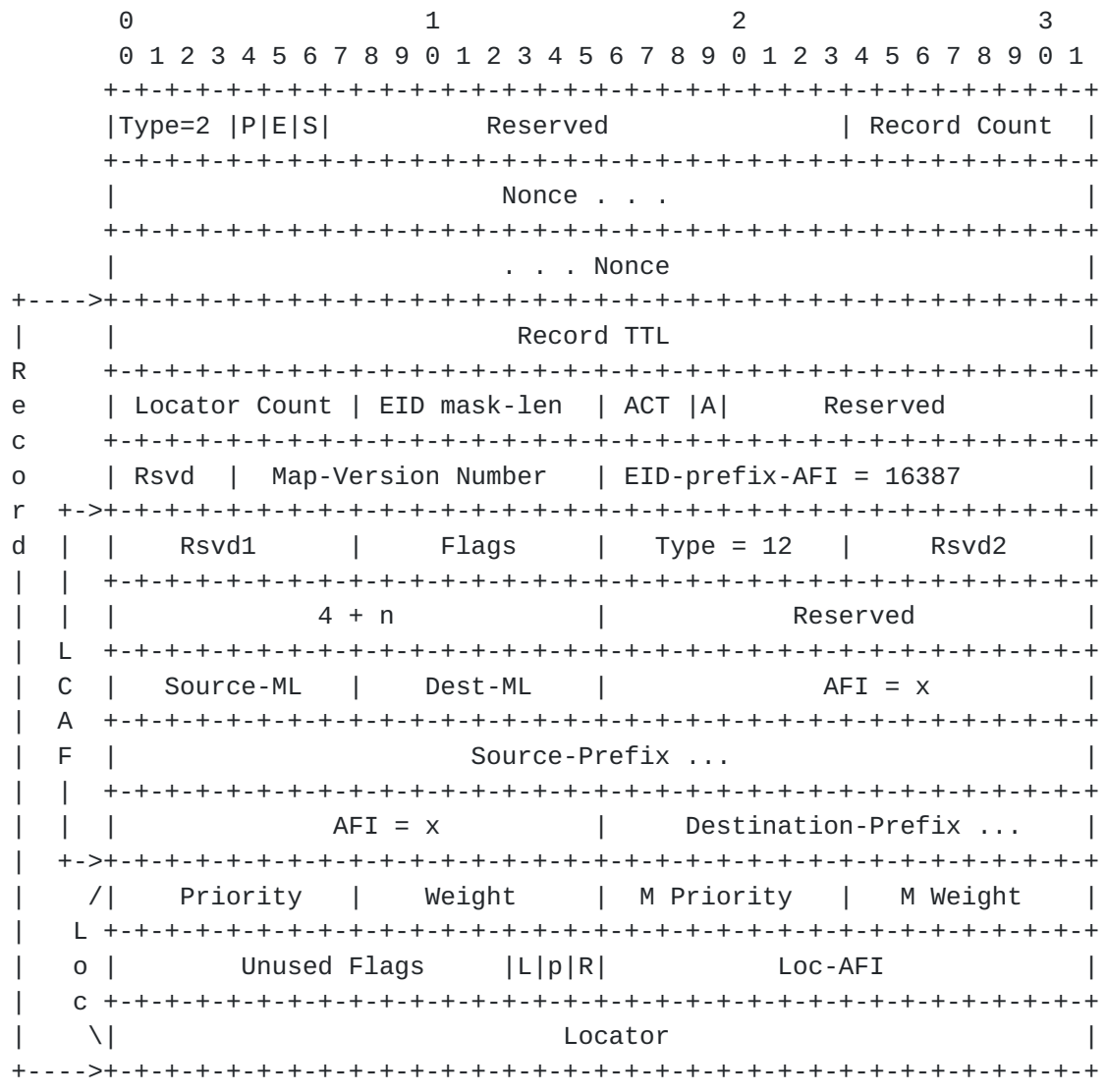


Where:
Source-Prefix = Client-EID
Destination-Prefix = App-EID

LISP Map-Request with 2-Tuple Src/Dst LCAF

In order to specify a 5 tuple flow, rather than just a two tuple source and destination, the combination of LCAF type 12 and LCAF type 4 must be used.

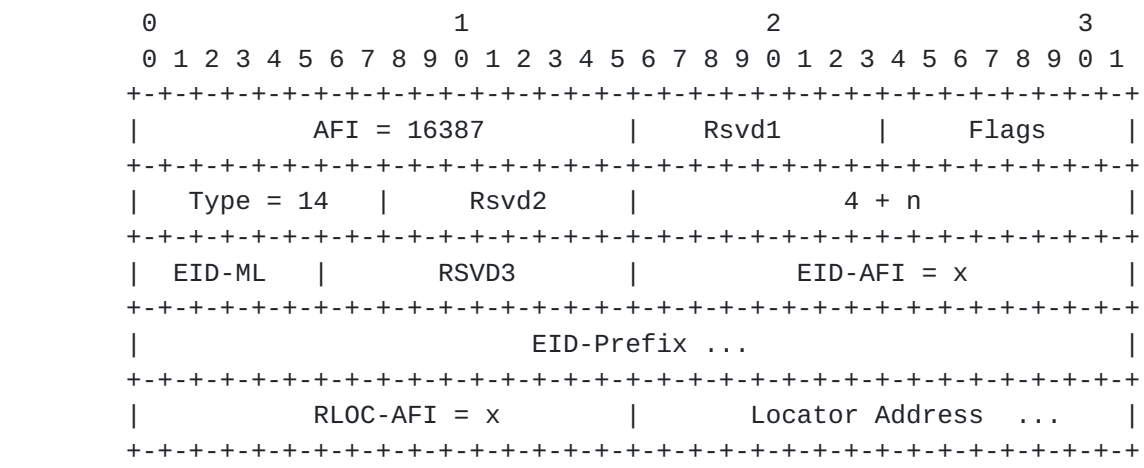
If an affinity exists in the Mapping System, meaning that the flow is already assigned to a virtual function instance, then the RLOC of that Function-Instance must be returned by the Mapping System. A Map-Reply with a 2-Tuple Src/Dst Lcaf can be used for this.



Map-Reply with 2-Tuple LCAF and Associated Function-Instance-RLOC

If no affinity exists, the Mapping System returns a list of records, including one record per each Function-Instance for the flow's Function-EID. A LISP Map-Reply can be used for this purpose with a 2-Tuple Src/Dst LCAF as the EID prefix in each Record.

If it is desired to return tuples of (Function-Instance-EID -> RLOC) per each record, a new LCAF, introduced as below, could be used.



EID-RLOC LCAF

In which, for the purpose of NFV, EID prefix will be used to specify Function-Instance-EID, and Locator address is the RLOC associated with that Funstion-Instance-EID. This LCAF can be used in place of the Loc-AFI in the Map-Reply Message above to include a list of (Function-Instance-EID,RLOC) for every (Client-EID, Function-EID) in the Map-Reply.

Finally to store the affinity of the flow in the Mapping System a Map-Register can be used where EID AFI is filled with a LCAF type 12 (2-Tuple Src/Dst LCAF), and Loc-AFI is filled with the AFI of the Function-Instance-EID, and the Locator is filled with the Function-Instance-EID. This way, a query on the flow 2-Tuple returns the Function-Instance-EID that the flow is assigned to.

7. QOS and Echo Measurements

8. Security Considerations

there are no security considerations related with this memo.

9. IANA Considerations

there are no IANA considerations related with this memo.

10. Acknowledgements

11. Normative References

[I-D.rodrigueznatal-lisp-multi-tuple-eids]
Rodriguez-Natal, A., Cabellos-Aparicio, A., Barkai, S., Ermagan, V., Lewis, D., Maino, F., and D. Farinacci, "LISP support for Multi-Tuple EIDs", draft-rodrigueznatal-lisp-multi-tuple-eids-05 (work in progress), March 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

Authors' Addresses

Sharon Barkai
Fermi Serverless
CA
USA

Email: sharon@fermicloud.io

Dino Farinacci
lispers.net
California
USA

Email: farinacci@gmail.com

David Meyer
1-4-5.net
USA

Email: dmm@1-4-5.net

Fabio Maino
Cisco Systems
California
USA

Email: fmaino@cisco.com

Vina Ermagan
Cisco Systems
California
USA

Email: vermagan@cisco.com

Alberto Rodriguez-Natal
Cisco Systems
California
USA

Email: natal@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu