LISP-CONS
A Mapping Database Service

NANOG 41

David Meyer, Dino Farinacci, Vince Fuller, Darrel Lewis, Scott Brim, Noel Chiappa

NANOG 41
October, 2007

http://www.1-4-5.net/~dmm/talks/NANOG41/cons
Agenda

• Brief Intro
• Design Considerations
• Brief Definitions
• How CONS Works
• What We’ve Learned
• Questions/Comments?
What is LISP?

- Locator/ID Separation Protocol (LISP)
  - draft-farinacci-lisp-03.txt
- Creates two namespaces: IDs and Locators
- Why do this?
  - Improve site multihoming
  - Improve ISP Traffic Engineering
  - Reduce site renumbering costs
  - Reduce size of core routing tables
  - PI for all?
  - Some form of mobility?
Locator/ID Split?

- The idea here is that the IP address is overloaded
  - It encodes both location in the topology (locator) and the identity of the user of the address
- The locator role is used by the routing system
- The identity role is used by upper layer protocols
  - e.g., TCP pseudo-header
- **Problem**: Since we want locators to aggregate topologically, and since identifiers are usually allocated on organizational boundaries, it is difficult (impossible?) to get one number space to efficiently serve both purposes.
  - There are other issues as well, including
    - The expected lifetime of a name (don’t want to reconfigure...)
    - Who has control over the name(s)?
    - ...
Locator/ID Split?

- One solution: split the functions -- This is at the heart of the Locator/ID split idea
  - So how might we achieve this?
- Architecturally, we might try to “Jack-up” the existing IP layer

![Diagram](image.png)
Implementing a Locator/ID Split

• There are two main ways to engineer a Loc/ID split

• **Rewriting**
  - If you have enough address space (e.g., IPv6), you could use the lower 64 bits as an identifier, and the upper 64 bits as a locator, and rewrite the locator at the border
  - This is the basis of O’Dell’s 8+8/GSE scheme
    - Credit to Bob Smart and Dave Clark on this one too

• **Map-n-Encap**
  - You could also put another header on the packet, and make the inner header carry the IDs and the outer header carry the locators
    - LISP is an instance of this approach
    - Credit to Bob Hinden & Steve Deering on map-n-encap...
Loc/ID Split in practice


Locator  ID

IPv4: 209.131.36.158.10.0.0.1

Locator  ID
LISP is a Jack-Up

- Uses IDs
- Host Stack
- Uses Locators
- Map-n-Encap

Layer Diagram:

- Application Layer
  Telnet, HTTP, FTP, SMTP
- Transport Layer
  TCP, UDP
- Network Layer
  IP
- Network Layer
  IP
- Physical Layer
  Ethernet, X.25, Token Ring
LISP Parts

• Data-plane
  - Design for encapsulation and tunnel router placement
  - Design for locator reachability
  - Data triggered mapping service

• Control-plane
  - Design for a scalable mapping service
  - This talk is about LISP Control-planes
LISP Variants

- LISP 1
  - Routable IDs over existing topology to probe for mapping reply
- LISP 1.5
  - Routable IDs over another topology to probe for mapping reply
- LISP 2
  - EIDs are not routable and mappings are in DNS
- LISP 3
  - EIDs are not routable, mappings obtained using new mechanisms (DHTs perhaps, LISP-CONS, NERD, APT)
Quick LISP Terms

• Endpoint Identifiers (EIDs)
  - IDs for host-use and only routeable in source and dest sites
  - Can be out of PA or PI address space
• Routing Locators (RLOCs)
  - Routable addresses out of PA address space
• Ingress Tunnel Router (ITR)
  - Device in source-site that prepends LISP header with RLOCs
• Egress Tunnel Router (ETR)
  - Device in destination-site that strips LISP header
LISP Control-Plane

- Build a large distributed mapping database service
- Scalability paramount to solution
- How to scale: \((\text{state} \times \text{rate})\)
- If both factors large, we have a problem
  - \text{state} will be “large” \((O(10^{10})\) hosts)
  - Aggregate EIDs into EID-prefixes to reduce state
  - \text{So rate} must be small
  - Make mappings have “subscription time” frequency
    - i.e., we expect such mappings to change with low frequency
  - And no reachibility information in the mapping database
Some Questions for a LISP Control-Plane

- Where to put the mappings?
- How to find the mappings?
- Is it a push model?
- Is it a pull model?
- Do you use secondary storage?
- Do you use a cache?
- What about securing the mapping entries?
- What about protecting infrastructure from DOS-attacks?
- What about controlling packet loss and latency?
LISP Control-Plane

• “Push doesn’t scale, caching doesn’t scale, pick one”
LISP-CONS

- LISP-CONS is a hybrid approach
- Push EID-prefixes (but not mappings) at upper levels of hierarchy
- Pull from lower levels of hierarchy
- Mappings stored at lower-levels
  - Requests get to where the mappings are
  - Replies are returned
  - This is a crucial point as we’ll see in a bit
- Getting to the lower-levels via pushing of EID-prefixes
- LISP-CONS is a mapping system for LISP 3.0
We can get good EID-prefix aggregation
  - If hierarchy based on EID-prefix allocation and not topology
  - Then build a logical topology based on the EID-prefix allocation

Map-Requests routed through logical hierarchy
  - Key is the EID

Map-Reply returned to originator
  - With mapping record \{EID-prefix, RLOC-set\}
LISP-CONS Network Elements

- **Content Access Routers (CARs)**
  - **Querying-CARs**
    - Generate Map-Requests on behalf of ITRs
    - Returns answers to ITRs
  - **Answering-CARs**
    - Hold authoritative mappings at level-0 of hierarchy
    - Aggregate only EID-prefix upwards
    - Respond with Map-Replies

- **Content Distribution Routers (CDRs)**
  - Push around EID-prefixes with level-1 to n of hierarchy
  - Aggregate EID-prefix upwards
  - Advertise EID-prefixes in a mesh topology within level
  - Forward Map-Requests and Map-Replies
LISP-CONS -- Peering

All peering on TCP HMAC protected connections

Within a CDR-mesh, EID-prefixes get seq num pushed with PV lists

Parent Peer

Child Peer

Sibling Peer

[ EID-prefix agg ]
Here's how it works

Legend:
{}: mapping entry
[]: EID aggregate
:

CDR Mesh

Level-0

CDR

qCAR

ITR

LISP-CONS

CDR Mesh

Level-1

CDR

qCAR

ETR

NANOG 41

CDR Mesh

Level-n

CDR

CDR

Has more-specific entry downward

Map-Request 1.1.1.1

Take shortest path to 1.0.0.0/8

No EID-Prefix within mesh, forward to parent peer

No mapping cached, forward to parent peer

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

{ 1.1.0.0/24: L1, L2 }

{ 1.1.0.0/16 }

{ 1.0.0.0/8 }

Map-Request 1.1.1.1

Map-Request 1.1.1.1

CAR has mapping, returns Map-Reply to orig CAR EID address

{ 1.1.0.0/24: L1, L2 }

{ 1.1.2.0/24: L11, L22 }

{ 1.1.1.0/24: L1, L2 }

{ 1.1.2.0/24: L11, L22 }

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

{ 1.1.1.0/24: L1, L2 }

{ 1.1.2.0/24: L11, L22 }

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

{ 1.1.1.0/24: L1, L2 }

{ 1.1.2.0/24: L11, L22 }

Map-Request 1.1.1.1

Map-Request 1.1.1.1

Map-Request 1.1.1.1

CAR has mapping, returns Map-Reply to orig CAR EID address

no EID-Prefix within mesh, forward to parent peer

No mapping cached, forward to parent peer
What We’ve Learned

• We wanted to optimize aggregatability of EID prefixes
  - That led to the design in which only EID prefixes were pushed around at the higher levels (but not the mappings themselves)
  - We were concerned about the rate*state product
• However, some SPs articulated another dimension
  - Latency
  - So you have to tradeoff rate, state, and latency
  - If you push, you wind up with the whole database in network elements (state)
  - If you pull, you incur latency
  - If you try to do mobility, you get lots of updates (rate)
What We’ve Learned

• Current thinking is that a different hybrid approach might be most feasible
• Push the whole mapping table around in the “CDR” level
• ITRs pull mappings from the “CAR” level
• This has a few nice properties:
  – You can get the whole mapping table
    • If you happen to want it
  – Latency is reduced because you don’t have to traverse the whole hierarchy to retrieve the mappings
Drafts

• LISP
  • draft-farinacci-lisp-03.txt

• CONS
  • draft-meyer-lisp-cons-02.txt
Questions/Comments?

Thanks!