LISP-Multicast

draft-farinacci-lisp-multicast-00.txt

Dino Farinacci, Dave Meyer, John Zwiebel, Stig Venaas

IETF Dublin - July 2008
Agenda

• Overview of LISP for Unicast Routing
• LISP-Multicast Design Goals
• LISP-Multicast Definitions
• Describe LISP-Multicast
• LISP-Multicast Interworking
• AF Case Studies
• Summary
LISP Internet Drafts

draft-farinacci-lisp-08.txt
draft-fuller-lisp-alt-02.txt
draft-lewis-lisp-interworking-01.txt
draft-farinacci-lisp-multicast-00.txt
draft-meyer-lisp-eid-block-01.txt
draft-mathy-lisp-dht-00.txt
draft-iannone-openlisp-implementation-01.txt
draft-brim-lisp-analysis-00.txt
draft-meyer-lisp-cons-04.txt
draft-lear-lisp-nerd-04.txt
draft-curran-lisp-emacs-00.txt
LISP Architecture

- Locator/ID Separation Protocol
  - Network-based solution
  - No changes to hosts whatsoever
  - No new addressing changes to site devices
  - Very few configuration file changes
  - Imperative to be incrementally deployable
  - Address family agnostic
LISP Architecture

- Locator/ID Separation Protocol
  - Defines architecture for 2 address spaces:
    - Endpoint IDs (EIDs)
    - Routing Locators (RLOCs)
  - Defines how Map-n-Encap routers operate and where they reside
  - Defines Variants for “EID routing”
  - Can use other designs for mapping EIDs to RLOCs
  - Defines encapsulation format for IPv4 and IPv6
  - Defines how RLOC reachability is performed
  - Defines how database mapping entries can be updated
  - Defines the Map-Request and Map-Reply format
Multi-Level Addressing

RLOCs used in the core

Mapping Database Entry:
1.0.0.0/8 -> (10.0.0.1, 11.0.0.1)

EIDs are inside of sites
Map-n-Encap

Mapping Entry:
- **EID-prefix**: 2.0.0.0/8
- **Locator-set (RLOCs)**:
  - 12.0.0.2, priority: 1, weight: 50
  - 13.0.0.2, priority: 1, weight: 50

Host Stack: supplies EIDs
LISP Router: supplies RLOCs by adding new header
Packet Forwarding

PI EID-prefix 1.0.0.0/8

Provider A
10.0.0.0/8

Provider B
11.0.0.0/8

Provider X
12.0.0.0/8

Provider Y
13.0.0.0/8

DNS entry:
D.abc.com  A  2.0.0.2

Legend:
EIDs -> Green
Locators -> Red

Policy controlled by destination site

LISP-Multicast

IETF Dublin - July 2008

Slide 8
LISP Multicast Design Goals

• Keep EID state out of core network
• No head-end replication at source site
• Packets only go to receiver sites
• No changes to hosts, site routers, core routers
• Use existing protocols
• Support PIM SSM, don’t preclude ASM & Bidir
• Have separate unicast and multicast policies
LISP Multicast Defs

- **(S-EID, G)**
  - S-EID is source host
  - G is group address receivers join to
  - State resides in source and receiver sites

- **(S-RLOC, G)**
  - S-RLOC is ITR on multicast tree
  - G is group address receivers join to
  - State resides in core

- **Group addresses have neither ID or Location semantics**
  - G is topologically opaque - can be used everywhere
LISP Multicast Defs

• Non-LISP Site
  - Does not support LISP at all
• uLISP Site
  - Support LISP-Unicast only
• LISP-Multicast Site
  - Supports unicast and multicast LISP
LISP Multicast Defs

- **Multicast ETRs at receiver sites**
  - Receives PIM JP (S-EID, G) messages from site routers
  - Obtains S-RLOC from mapping for S-EID
  - Sends unicast PIM JP (S-EID, G) to ITR S-RLOC address
  - Sends regular PIM JP (S-RLOC, G) through core
  - Decapsulates multicast (S-RLOC, G) into (S-EID, G)

- **Multicast ITRs at source sites**
  - Receive unicast PIM JP (S-EID, G) messages and forward into source site
  - Acts as root of (S-RLOC, G) tree for site
  - Encapsulates (S-EID, G) packets into (S-RLOC, G)

- **mPTRs**
  - Multicast Proxy Tunnel Routers (PTRs)
Multicast Packet Forwarding

Legend:
- EIDs → Green
- Locators → Red
- site or core
- regular PIM JP
- unicast PIM JP
Multicast Packet Forwarding

Aggregated-Trees
$(S\text{-EID}, G)$:
- $(1.0.0.1, G)$
- $(1.0.0.2, G)$

map to:
$(S\text{-RLOC}, G)$:
- $(11.0.0.1, G)$

Use SSM!
Join Policy - 1 Exit, 1 Entrance
Join Policy - 2 Exits, 1 Entrance
Join Policy - 1 Exit, 2 Entrances

Provider A 10.0.0.0/8
Provider B 11.0.0.0/8
Provider X 12.0.0.0/8
Provider Y 13.0.0.0/8

S1 10.0.0.1
S2 10.0.0.1
Provider A 10.0.0.0/8
Provider B 11.0.0.0/8
Provider X 12.0.0.0/8
Provider Y 13.0.0.0/8

S

D

S1 10.0.0.1
S2 10.0.0.1
R1
R2
R3

R11
R12
R21
R22

R1
R1
R1
R1
R1
R1
Join Policy - 2 Exits, 2 Entrances

Provider A
10.0.0.0/8

Provider B
11.0.0.0/8

Provider X
12.0.0.0/8

Provider Y
13.0.0.0/8

S

D

S1

S2

R11

R12

R1

R2

R3

R21

R22

ETR

ETR
6.1.4. Map-Reply Message Format

```
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|x|M|       Locator Reach Bits       |
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Nonce       |
+-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Type=2 |  Reserved      | Record Count |
+----> +-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      |                  Record TTL    |
|      |                            |
|      |                            |
|      | Locater Count | EID mask-len | A|  Reserved |
|      |                            |
|      |                            |
|      +-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      | ITR-AFI | EID-AFI |
|Re |                            |
|e |                            |
|c |                            |
|o |                            |
|r |                            |
|d |                            |
|d |                            |
|d |                            |
+----> +-----------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      |                  Mapping Protocol Data |
```

LISP-Multicast  IETF Dublin - July 2008  Slide 19
Locator Reachability

- Loc-reach-bits
  - Used for both unicast and multicast
  - Mapping entries contain all locators
  - You can turn locators on for unicast and off for multicast by setting priority to 255
  - Multicast and Unicast fate-share each other
    * A mapping obtained for a unicast packet can be used for a multicast join and vice-versa
LISP-Multicast Interworking

- Non-LISP sites
  - Unicast only
  - Unicast and multicast
- uLISP sites
  - Unicast-LISP
  - Multicast-Traditional
  - No Multicast at all
- Multicast-LISP sites
  - Both Unicast-LISP and Multicast-LISP
- Simplification
  - Treat Non-LISP-multicast and uLISP-multicast-traditional the same
Interworking Solutions

• **Unicast Interworking**
  - Translation at source site
  - Proxy Tunnel Routers (PTR)

• **Multicast Interworking**
  - Translation at receiver site
    • Would require another mapping table
    • Receiver sites must translate to same routable (S-EID)
    • Operationally a non-starter
  - **Multicast Proxy Tunnel Routers (mPTRs)**
    • Reduce where (S-EID) state resides in network
    • (S-EID) state exists between non-LISP site and mPTR
Interworking Case Studies

- Non-LISP multicast source site
  - All receiver sites are LISP
  - Mix of LISP and non-LISP receiver sites
- uLISP source site -- same as above
- LISP multicast source site
  - All receiver sites are non-LISP
  - All receiver sites are LISP
  - Mix of LISP and non-LISP
LISP-Multicast Interworking

1) EIDs must be routable
2) ITRs don’t encapsulate
3) ITRs never receive unicast-JP

Legend:
- LISP site
- non-LISP site
1) ITRs get two joins
   Options:
   - Head-end replicate
   - Use mPTR
2) Use mPTR so LISP source site can behave with 1 mechanism
3) mPTRs decapsulate and deliver to non-LISP receiver sites

Legend:
- LISP site
- non-LISP site
1) Receiver LISP sites know source site is non-LISP because there is no mapping
2) EID must be routable
3) No unicast-JPs sent just regular JPs

Legend:
- LISP site
- non-LISP site
LISP-Multicast Interworking

1) Receiver LISP sites know source site is non-LISP because there is no mapping
2) EID must be routable
3) No unicast-JPs sent just regular JPs

Legend:
- LISP site
- non-LISP site
Multicast PTRs

- **Unicast PTRs**
  - They proxy for ITR functionality
  - They advertise coarse prefixes to attract and encapsulate packets

- **Multicast PTRs**
  - They proxy for ETR functionality
  - They advertise coarse prefixes to attract Joins and therefore attract and decapsulate packets

- No unicast EID state in core from PTRs to ETRs
- No multicast (S-EID, G) state in core from ITRs to mPTRs
AF Case Studies

1. LISP-Multicast IPv4 Site
2. LISP-Multicast IPv6 Site
3. LISP-Multicast Dual-Stack Site
4. uLISP IPv4 Site
5. uLISP IPv6 Site
6. uLISP Dual-Stack Site
7. non-LISP IPv4 Site
8. non-LISP IPv6 Site
9. non-LISP Dual-Stack Site
Case Studies

• A Combinatorial Explosion
  - $(9 \ 1) + (9 \ 2) + \ldots + (9 \ 9)$
  - 502 combinations!

• Rationalize by forcing multicast tree be AF homogenous
  - Avoid head-end replication and translation
  - Source site mapping entry priority decides which AF
  - When source site is non-LISP, use application-level directory to determine AF

• Further simplification
  - Make all sites that deploy LISP be unicast & multicast capable from the day-one
LISP Protocol Mechanisms

• Changes only to ITRs, ETRs, & PTRs
• Simple change to PIM
  – Sending/receiving unicast JPs
  – Translation of JP from EID to RLOC
• Simple change to MSDP
  – Translation for RPF-peering for RP
• Simplification
  – Only support SSM for inter-domain
• Use mPTRs to reduce (S-EID, G) state
  – Using (S-RLOC, G) reduces multicast routing table size we have today!
Best Possible Outcome

- Provider A (10.0.0.0/8)
- Provider B (11.0.0.0/8)
- Provider X (12.0.0.0/8)
- Provider Y (13.0.0.0/8)

Nodes:
- S
- S1
- S2
- D1
- R11
- R12
- R21
- R22
- R4
- R3

State only:

(S-RLOC, G)
(defun changer-one (failures database)
  (cond ((null failures) database))
  (setf first-part (process-one (car failures) database))
  (setf second-part (process-two (car failures) first-part))
  (cond ((null (cdr failures)) second-part)
        ((changer-one (cdr failures) second-part))
     )
)

(defun cbr-top (source destination plan path prev-plan)
  (setf chosen-path (cbr-evaluate source destination plan path prev-plan))
  (if (cbr-exact-match (list source destination) chosen-path)
    (cbr-new-plan chosen-path plan))
  (; we've reached the conclusion of CBR - return the path
   (cond ((cbr-exact-match (list source destination) plan)
      (append path (cbr-get-path list source destination prev path)))
     )
    )
)

(defun cbr-get-path list source destination
  (cond ((cbr-exact-match list source destination)
     (append path (cbr-get-path list source destination prev path)))
     )
)

(defun cbr-new-plan chosen-path plan
  (cond ((null chosen-path) nil)
     (if (cbr-exact-match (list source destination) chosen-path)
       (append path (cbr-get-path list source destination prev path)))
     )
)

(defun reverse (l)
  (cond ((null l) nil)
     (if (cbr-exact-match (list source destination) chosen-path)
       (append path (cbr-get-path list source destination prev path)))
     )
)

(defun process-two (pair database)
  (cond ((null database) nil)
     (or (equal (cdr pair) database))
     (equal (reverse (cdr pair)) database))
)

(defun convert-output-helper (path)
  (cond ((null path) nil)
     (t (append (list (cons (first path) (list (list (first path)))))
      (convert-output-helper (rest path))))
     )
)

(defvar database
  '((10th Tech-Parkway) (10th Curran) (Tech-Parkway North-Avenue))
  '((10th Curran) 10th McMillan) (10th Curran) (10th hemphill) (mcmillan 8th-1))
  ((10th hemphill) (10th mcmillan (10th dalney) (hemphill 8th-1))
  )
)

(defun lisp-interest@lists.civil-tongue.net
  (process-two pair (cbr-get-plan lisp-interest@lists.civil-tongue.net))
  )

(defun mLISP
  (process-two pair (cbr-get-plan mLISP))
  )

(defun rules
  (process-two pair (cbr-get-plan rules))
  )

(defun lisp-interest@lists.civil-tongue.net
  (process-two pair (cbr-get-plan lisp-interest@lists.civil-tongue.net))
  )