ソフトウェアLISPルータの設計と実装

上野 幸杜 † Yukito Ueno　堀場 勝広 ‡ Katsuhiro Horiba　片岡 広太郎 ‡ Kotaro Kataoka

1. はじめに

1.1 概要

近年, インターネットのルーティングアーキテクチャは, スケーラビリティ, マルチホーミング, トラフィックエンジニアリング等, 様々な問題に直面している. これらの問題に答えるべく設計されたプロトコルとして,Locator/ID Separation Protocol(LISP) が注目されている.LISP は商用のルータ (Cisco IOS,NX-OS) に既に実装されており, 実際のネットワークでの運用が可能である. これはIETF で議論されている [1] ような LISP 以外の比較的新しいルーティングプロトコルに対して,LISP が持つ利点と言える.

1.2 LISP とは

LISP は,IP アドレスが持つ Locator と Identifier という二つの側面を分離して管理するプロトコルである.Locator とは,IP ネットワーク上でノードが実際にどのルータに属しているかという情報であり,Identifier とは, ノードを一意に区別するという IP アドレスの性質を指す.LISP において Locator は Rloc と呼ばれ,Identifier は EID と呼ばれる. ある EID 宛のパケットは,ITR(Ingress Tunnel Router) によってカプセル化され, その EID を配下に持つ ETR(Egress Tunnel Router) まで転送される (ETR と ITR を総称して XTR と呼ぶ. 以後本文では,LISP ルータと XTR を同一のものとして扱う).ETR のもつグローバルアドレスが Rloc である. このとき,LISP のパケットをカプセル化/デカプセル化する機能を data-plane と呼び,Rloc と EID のマッピングを解決する機能を control-plane と呼ぶ.LISP の詳細なプロトコルスペックについては, 文献 [2] 及び [3] を参照されたい.

以下に LISP を導入することで得られる主なメリットを挙げる.

1. 任意のプレフィクスを経路表に影響を与えずにインターネット上の任意の位置に配置できる
2. マルチホームを PI アドレスやパンチングホールルーティングなしで行える
3. 比較的用意にトラフィックエンジニアリングをすることができる
4. IPv6 over IPv4 または IPv4 over IPv6 を行える

1.3 問題点

現時点ではオープンソースかつ control-plane の機能を持った LISP 実装は存在しない.control-plane は map-request/reply や map-regist など,LISP の運用を自動化するための種々の機能を含むが, これが存在しない場合, ネットワークの規模がそのまま運用コストにつながってしまう. また,IPv6 に対応した XTR の実装が少ないため, 途中経路のネットワークスタックに依存せず IPv6 を展開できるという LISP のメリットを活かすことができない.

LISP はクラウド技術と相性が良く, 特定のプレフィクスを第三者の運用するクラウド内に配置するというユースケースが予想されるが, 第三者の運用するクラウド内に物理的にルータを設置することは不可能であるため,VM 上に LISP XTR を構築することになる. このとき, 汎用 OS 上で動くオープンソースかつ control-plane の機能を持った LISP 実装が必要になるため, それらを兼ね備えた実装に対する需要は高いと言える.

1.4 目的

以上で述べた問題点を解決するため, 本研究で実装する LISP XTR の機能要件を以下に挙げる.

1. 汎用 OS 上で動作する
2. data-plane の機能を持ち, パケットのカプセル化/デカプセル化ができる
3. control-plane の機能を持ち,Map-Regist/Map-Request/Map-Reply 等の機能によって EID-Rloc マッピング解決や登録の自動化が出来る
4. IPv4/IPv6 に対応している

2. LISP XTR の設計

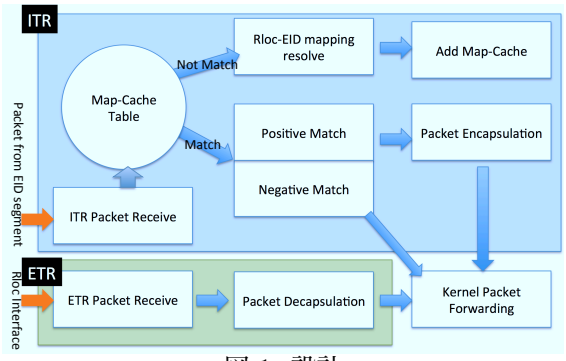

図 1: 設計

ITR は,EID セグメントに直接接続しているインターフェースに着信したパケットをカーネルからフックする. 次にフックしたパケットの宛先を Map-Cache テーブルからロンゲストマッチに基づいて検索し, マッチする経路が無い場合,Map-Request/Reply による EID-Rloc マッピングの解決を行い,Map-Cache テーブルに登録を行った後, 再び宛先を検索する.Map-Request がタイムアウトするか,Map-Server からネガティブキャッシュの応答があれば, その経路はネガティブキャッシュとして Map-Cache テーブルに登録され, ポジティブキャッシュの応答があればその経路は Rloc のアドレスと共に Map-Cache テーブルに登録される.

Map-Cache テーブルを検索し, マッチする経路があった場合,Rloc の有無によってネガティブキャッシュかポジティブキャッシュかを判断する. ネガティブキャッシュであれば, そのままカーネルのフォワーディングエンジンにパケットをインジェクトし, ポジティブキャッシュであればパケットをカプセル化した上でカーネルのフォワーディングエンジンにパケットをインジェクトする.

†慶應義塾大学 環境情報学部
‡慶應義塾大学 政策・メディア研究科

ETR は, 自分の持つ IP アドレス宛に着信した UDP パケットのうち,LISP data-plane ポートとして定義されている 4341 番ポート宛のものをカーネルからフックする. フックしたパケットから IP ヘッダ,UDP ヘッダ,LISP ヘッダを取り除き, カーネルにパケットをインジェクトする.

3. Linux における LISP XTR の実装

実装は,Linux kernel 2.6.26 及び C 言語 (gcc version 4.3.2) を使用して行った. 本実装の基本的な仕組みを図 [2] に示す.

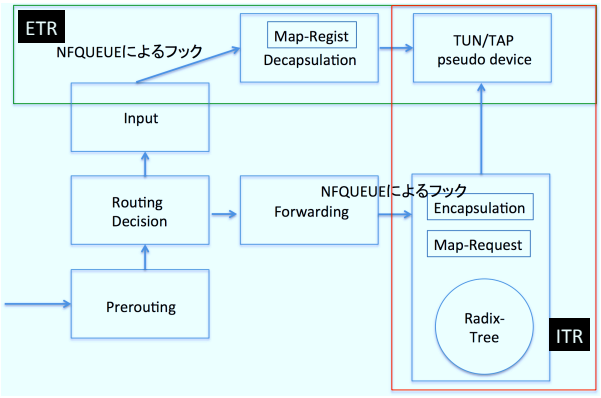

図 2: 実装

ITR としての動作をするために,EID 側からのパケットを全てフックする必要があるが, これは nfqueue を用いてあらかじめ EID 側のインターフェースに着信したパケットを全てフックするように設定し, アプリケーション側のハンドラでパケットを受け取ることで実現した. また, ポジティブキャッシュとネガティブキャッシュの管理及びロンゲストマッチは Radix-Tree 方式で実装した. ヘッダ付加後, カーネルにパケットをインジェクトする際には,TAN/TAP driver を使用している.

ETR としての動作をするためには, 自ホストの UDP port 4341 番宛のパケットをフックする必要があるが,LISP の data-plane パケットには UDP checksum を含めないことが推奨されているため,UDP ソケットによってパケットの中身を受け取る前にカーネルによってパケットが破棄されてしまう. よって, これも nfqueue を用いてカーネルのパケットフォワーディングエンジンの内,"INPUT" に到達したパケットを全てフックするように設定し, アプリケーション側のハンドラで受け取ることで実現した.Decapsulation 後の元のパケットは,TAN/TAP driver によってカーネルに再びインジェクトされ,ARP/ND の解決を行ったのち,EID 側のホストに到達する.

4. 評価

4.1 Cisco IOS との interoperability

実際に Cisco IOS との interoperability を確認した機能を以下に挙げる.

1. LISP パケットのカプセル化/デカプセル化
2. Map-Request/Reply/Regist

4.2 他の実装との比較

図 [3] に, 既存の LISP XTR 実装の現状を示す. 青色で示したものがオープンソース実装であり, 赤色で示したものはオープンソースではない. 本実装は data-plane と control-plane に対応し,IPv6 かつ Linux 上で動作する.

	control-plane	data-plane	UI	IPv6	動作環境	言語
OpenLISP	×	○	×	○	BSD kernel	C言語
Cisco LISP	○	○	○	○	Cisco IOS/ NX-OS	
Zlisp	○	○	×	○	Linux/ BSD/ MacOSX	Portable C++
Aless	×	○	×	×	Linux kernel	C言語
LISP-Click	×	○	×	×	Java VM	Java
現在	○	○	×	○	Linux Userland	C言語
今後	○	○	○	○	Linux kernel	C言語

図 3: LISP XTR 実装の現状

オープンソースで control-plane と IPv6 に対応し, 汎用 OS 上で動作する LISP XTR は存在しなかったため, 本実装が新規に達成したと言える.

4.3 パフォーマンス計測

本実装と Cisco LISP が実装された Cisco 1812j ルータ, 及び Map-Server/Resolver 機能を持った Cisco 2800 ルータを用意し, 各 XTR 毎に End-Point を 1 台ずつ配置した. この End-Point ともう一方の End-Point の間で iperf を用いた帯域計測を行い, 本実装の性能を評価した. また対照実験として Cisco1812j 同士での計測も行った. 計測は TCP,UDP のどちらについても MTU1000byte で行い,TCP の場合は 1Gbyte のデータ転送,UDP の場合は 10 秒間 100Mbps のデータ転送を行った. 結果を以下に示す.

	TCP	UDP
Cisco 1812 ↔ Cisco 1812	84.3Mbps	77.5 Mbps
this implementation ↔ Cisco 1812	86.3Mbps	86.9 Mbps

5. むすび

Linux で動作し, 機能要件のうち TTL 管理機能以外を満たす LISP XTR を実装した. これにより, オープンソースで control-plane 及び data-plane の機能を持ち,IPv6 に対応する LISP XTR を Linux 上で構築することが可能になった.

今後の予定として,TTL 管理などの未実装な部分を作成する. 次に,Cisco IOS 等との interoperability test を行い, 特に control-plane のバグフィックスする. 実装終了後は, 実際にクラウド環境に展開し, ライブマイグレーションをする際のダウンタイムなどを評価する. また, 学術系イベントのネットワークで使用し, 実地での使用に問題がないか検証する.

参考文献

[1] Li, T., Ed., "Recommendation for a Routing Architecture", RFC 6115, February 2011.

[2] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis,"Locator/ID Separation Protocol (LISP)", work in progress, July 2011.

[3] Farinacci, D., Fuller, V.,"LISP Map Server", work in progress, August 2011.