

SYN Cookies を活かす Locator/ID Separation Protocol 導入法の検討 Locator/ID Separation Protocol implementation considering SYN Cookies

渡邊 孝也[†]

Watanabe Takaya

今泉 貴史^{‡†}

Takashi Imaizumi

1. はじめに

現在, AS (Autonomous System, ISP や組織など) 間のルーティングに必要な経路情報が増大し, それを保持するルータの容量を超えてしまう問題が発生している. この対策として, IETF が **Locator/ID Separation Protocol (LISP)** を提案している.

AS 間と AS 内のルーティングは, 現状 IP アドレスを用いて一元管理している. しかし LISP では, AS 間と AS 内のアドレスを区別し, それぞれに異なる IP アドレス空間を用意することで現状の一元管理を分離する. これによって, 問題となっている経路情報の増大を防ぐことができる. ただし, AS 間を通過する通信において, このアドレス分離の境界となるルータが分離されたアドレスの対応関係を解決しなければならない. 対応関係の解決の際に, ルータはパケットを一度破棄するため, パケットを送信したエンドノードで再送が必要となる. したがって, エンドノードの再送ができない状態となった場合, 通信が困難になると考えられる.

エンドノードの再送ができないものとして, SYN Cookies がある. これは, DoS 攻撃の 1 つである TCP SYN Flood 攻撃の緩和策である. SYN Cookies は, 受信側エンドノードが TCP コネクション確立時に接続情報を保持しない. これにより攻撃に対応できるが, 影響として, 受信側エンドノードは TCP コネクション確立時に再送ができない. LISP 環境下で TCP SYN Flood 攻撃が行われ SYN Cookies を適用した場合, 現状のままではお互いの仕様から新たな問題が発生してしまう.

本研究では, SYN Cookies を活かす LISP 導入法を検討する. これによって, LISP 環境下で TCP SYN Flood 攻撃が行われた場合にも対策が可能となる.

2. LISP と SYN Cookies

LISP と SYN Cookies それぞれの概要と問題点を述べた後, LISP 環境下で SYN Cookies が適用された場合について説明する.

2.1. LISP

LISP は現在のルーティングに用いる IP アドレスを AS 間と AS 内で区別する. AS 間で用いる IP アドレスを **RLOC** (Routing LOCator), AS 内で用いる IP アドレスを **EID** (Endpoint IDentifier) と名付けている. RLOC と EID の対応関係は新しく用意される LISP インフラストラクチャ (後述) で管理される. これによって, 境界で RLOC と EID の橋渡しが行えるため, AS 間では RLOC のみ, AS 内では EID のみを用いてルーティングが行える.

現状の AS 間と AS 内のアドレスを一元管理している状況では, マルチホーミング (ある組織が複数の ISP を用いて Internet に接続する手法) などを行っている場合に, AS 内の経路情報の一部を AS 間ルーティングのための経路情報として広告する必要が生じていた. このような広告の増大が AS 間の経路情報を莫大なものにしていく. LISP でアドレス空間の分離を行うことによって, この広告を AS 間のルータは経路情報として保持しなくてもよい. これにより, ルータの容量を超える経路増大の問題を解決できる.

本節では, LISP を実現するためのデバイスについて触れ, 通信時の基本動作を示す. また, LISP 特有の動作となる, First Packet Drop についても述べる.

2.1.1. LISP デバイス

LISP では, 現状のアドレス空間を RLOC と EID へ分離するために, 次の機能が必要となる.

• LISP サイトエッジデバイス

AS 内 (EID 空間) と AS 間 (RLOC 空間) の境界に位置するデバイス (ルータ) であり, EID と RLOC それぞれのアドレス空間を橋渡しする. RLOC 空間への入口となる機能をもつデバイスを ITR, 出口となる機能をもつデバイスを ETR と呼ぶ.

– ITR (Ingress Tunnel Router)

EID 空間側からパケットを受け取り, 宛先が所属する AS の ETR へパケットをカプセル化して転送する. また, 宛先が所属する AS の RLOC を解決するリクエストを LISP インフラストラクチャへ送信する.

– ETR (Egress Tunnel Router)

RLOC 空間側からパケットを受け取り, パケットをデカプセル化し, エンドノードへ転送する. また, LISP インフラストラクチャ経由で ITR からリクエストを受け取ると, リプライを返信することで RLOC の解決を導く.

通常 ITR と ETR の機能は一つのデバイスで実現し, LISP サイトエッジデバイスとする. このデバイスを, xTR と呼ぶ. しかし, LISP では一つのデバイスで実現することが必須であるとは規定されていないため, 異なるデバイスとして ITR と ETR を用意してもよい.

LISP サイトエッジデバイスは既存のルータをアップグレードすることで実現できる. したがって, LISP は各 ISP や企業が導入しやすいという利点がある. ネットワークエンドポイントに影響を与えることなく段階的に導入できるため, 実装行程を考えた場合も現実的である.

[†]千葉大学融合科学研究科

[‡]千葉大学総合メディア基盤センター

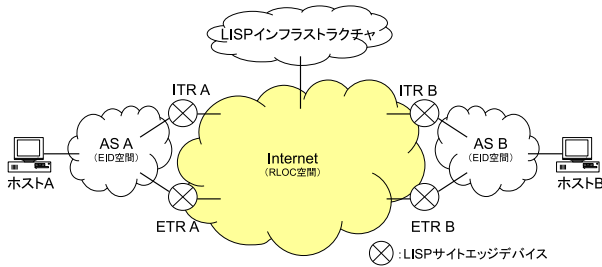


図 1: LISP 環境例

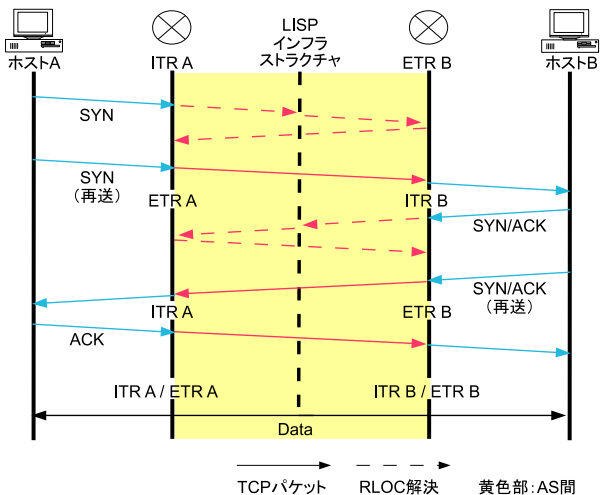


図 2: LISP 環境下の TCP コネクション

● LISP インフラストラクチャ

AS 内で用いる EID と、AS 間で用いる RLOC の対応関係を管理するデバイス群。ITR から受け取ったリクエストを、宛先 EID に対応する RLOC へ転送する。これによって、エンドノードは LISP を気にすることなくエンド・ツー・エンドのやり取りが可能となる。また、経路情報の増加に対してスケーラブルな対応ができる。

2.1.2. LISP 基本動作

TCP コネクションの確立を例に LISP の基本動作を示す。

図 1 に LISP 環境例を示す。それに対応して図 2 に、LISP 環境下においてホスト A からホスト B へ TCP コネクションを確立した場合のタイムラインを示す。ただし、ITR A, B の両者が送信先 EID に対応する RLOC のキャッシュを保持していない状態とする。

1. ホスト A が、ホスト B 宛てに SYN パケットを送信する。
2. SYN パケットが ITR A に到達する。ITR A は受け取ったパケットの宛先 EID に対応する RLOC を解決するため、LISP インフラストラクチャにリクエストパケットを送信する。この際、応答を待つ間に受け取った SYN パケットを破棄する。

3. LISP インフラストラクチャの情報によって、宛先ホスト B の EID に対応する RLOC が解決され、ETR B へリクエストパケットが転送される。
4. ETR B は受け取ったリクエストパケットの送信元である ITR A へ直接リプライパケットを送信し返答する。
5. ITR A はリプライパケットから、目的の宛先であるホスト B の EID に対応する RLOC を解決し、キャッシュとして保持する。
6. ホスト A は応答がないためタイムアウトして、SYN パケットを再送する。
7. ITR A にパケットが再到達する。ITR A は、先ほどキャッシュを作成しているため、ホスト B の EID に対応する RLOC へ SYN パケットを転送できる。この際、送信元を ITR A の RLOC、送信先を ETR B の RLOC としてカプセル化し、転送する。
8. ETR B にカプセル化された SYN パケットが到達する。ETR B はデカプセル化し、ホスト B へ SYN パケットを転送する。
9. TCP コネクションであるから、ホスト B は受け取った SYN パケットに SYN/ACK パケットを返答する。この場合も、1~8 と同様の行程を行う。
10. ホスト A は SYN/ACK パケットを受け取り、ACK パケットを送信する。このときは、ITR A が既にキャッシュを作成しているため、RLOC 解決は行わずそのままホスト B へ転送できる。

ACK パケットがホスト B に到達し、TCP コネクションの確立後は、RLOC 解決は双方向で不要である。RLOC 空間を通過する際に、ITR におけるカプセル化と ETR におけるデカプセル化を行うだけでデータ通信が可能となる。

2.1.3. First Packet Drop

前節の基本動作で述べた通り、RLOC 解決を行う際、ITR が初めに受け取ったパケットを破棄する。これを、First Packet Drop という。ここで、「初めに」というのは、宛先 EID に対応する RLOC のキャッシュを保持していない状態を指す。

RLOC 解決をするための時間、ITR がパケットを保持し続けると、ITR のメモリがすぐに埋まってしまう可能性がある。たとえば、TCP SYN Flood 攻撃の通過点となった場合に、パケットを保持してしまうとメモリが飽和状態となり、正常なパケットを受け入れることができない。これを考慮して、RLOC 解決を行う際は、パケットを破棄しメモリを守る。

First Packet Drop が行われた場合、パケットを失ってしまうので、エンドノードが再送する必要がある。つまり、LISP はエンドノードの再送に依存している。

2.2.SYN Cookies

SYN Cookies は、DoS 攻撃である TCP SYN Flood 攻撃の緩和手法のひとつである。TCP SYN Flood 攻撃は、TCP コネクションの確立行程に使用する SYN パケットを大量に送りつける攻撃である。このとき、攻撃者は送信元をランダムに偽装している場合が多い。攻撃により、受信側の接続情報を保持するメモリが飽和し、新しいコネクションが確立できない状態に陥る。SYN Cookies は、受信側が TCP コネクション確立時に接続情報を保持しないことで、攻撃を受けた場合でも新しいコネクションが確立でき、攻撃に対して耐性を持つ。

本節では、SYN Cookies の仕様を示し、その仕様から発生する問題点について述べる。

2.2.1. 仕様

通常 TCP では、受信側は受け取ったパケットの接続情報を保持するが、SYN Cookies は TCP コネクション確立過程において受信側がその情報を保持しない手法である。

接続情報を保持しない場合、受信側は応答パケットである ACK を受信しても、正常に通信できたか確認ができない。SYN Cookies では、返信する SYN/ACK パケットに接続情報を埋め込むことで、正常に通信ができたことを確認できる様になっている。具体的には SYN/ACK パケットのシーケンス番号 (32 ビット) に情報を埋め込む。その例を示す。

- 最初の 5 ビット： $t \bmod 32$ 。 t はカウンタで 64 秒ごとに増加する。
- 次の 3 ビット：MSS (最大セグメントサイズ) のエンコード値。
- 最後の 24 ビット：送信元/宛先の IP アドレスと TCP ポート番号。 および、カウンタ t の値を受信側の一方方向ハッシュ関数でハッシュ化したもの。

これらの接続情報が埋め込まれた SYN/ACK パケットを受信した接続要求側は、受け取ったパケットのシーケンス番号に 1 を足した値を確認応答番号として、ACK パケットを返信する。受信側は、ACK パケットのハッシュ値を計算して確認応答番号が正しいことを確認する。こうして TCP コネクションの確立が完了する。この行程を図 3 に示す。

2.2.2. 問題点

ACK パケットが受信側に到達する前に何らかのかたちで失われてしまった場合、通常であれば再送処理が行われる。しかし、SYN Cookies では再送のとき用いる接続情報にメモリを割り当てていないため、再送ができない。

SSH, FTP, または SMTP などの場合は、ACK パケットが失われてしまうと、要求側が待ち続けることになる。これは、SSH などが TCP コネクション確立後、受信側からデータを送るからである。本来 SYN/ACK パケットが再送されなければならない状況であるが、SYN Cookies では再送ができないため、要求側は TCP コネクションが確立されているものとして待ち続けること

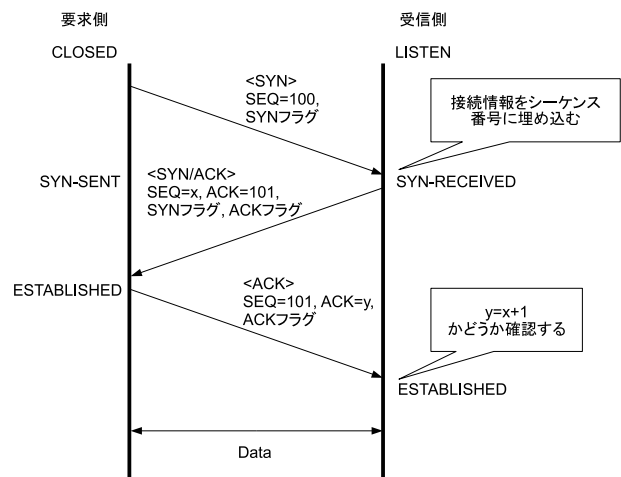


図 3: SYN Cookies の流れ

になる。これは要求側のメモリを無駄に使用することになる。したがって、通常時は SYN Cookies を適用せず、TCP SYN Flood 攻撃が行われた際に適用するといった方法がとられる。

2.3.LISP 環境下の SYN Cookies

LISP 環境下で TCP SYN Flood 攻撃が行われ、現状の SYN Cookies を用いた場合、お互いの仕様により発生する問題について述べる。

図 1 で示した環境において、SYN Cookies が適用されている場合、TCP コネクションの確立過程の変化を図 4 に示す。2.1.2 のときと同様に、ルータはキャッシュを保持していないものとする。SYN Cookies を適用しているため、受信側からの RLOC 解決において再送ができない。図 2 と比較して分かるように、SYN/ACK パケットの再送ができないため、要求側が再びタイムアウトして SYN パケットを再送する。

RLOC 解決の全体を通してみると、要求側が連続 3 回 SYN を送信し、再送回数は 2 回となる。SYN Cookies が適用されていない場合は、SYN と SYN/ACK が 2 回ずつ送信され、再送回数としては 2 回である。したがって、パケットの再送回数は同じである。しかし、TCP は「Truncated Binary Exponential Backoff」と呼ばれる方式を採用している。この方式は再送する度に再送間隔を増加させていく仕様である。同じパケットの再送回数が増加した場合、遅延時間が増加することになる。つまり、LISP 環境下で SYN Cookies が用いられた場合、同じパケットの再送が連続するため、全体としてコネクション確立に要する時間が長くなる。

前述した再送回数は RLOC 解決に要する時間が極めて小さい場合である。RLOC 解決に要する時間によっては、再送回数が増加し、コネクション確立中にタイムアウトになる可能性も考えられる。以下に RLOC 解決に要する時間について述べる。

RLOC 解決に要する時間として、バルセロナのカタルーニャ工科大学が現状の LISP ネットワークで計測を行った値を LISPmon (LISP monitoring platform)

表 2.1: RLOC 解決 Round-Trip-Time (RTT)

Map-Server	EIDs	RLOC	RTT (ms)
ijj-xtr (IIJ Internet Initiative Japan Inc.)	153.16.64.0/24	202.214.86.252	493
fns-c-xtr (ODN SOFTBANK TELECOM Corp.)	153.16.66.176/28	61.123.132.140	428
cisco-it-xtr-1 (CISCO-EU-109 Cisco Systems Global)	153.16.5.0/24	128.107.81.169	303
google-xtr (GOOGLEWIFI - Google Inc.)	153.16.30.0/28	64.9.224.225	299
unknown (INTERNLNET InterNLnet Autonomous System)	85.184.3.32/28	92.254.28.189	55

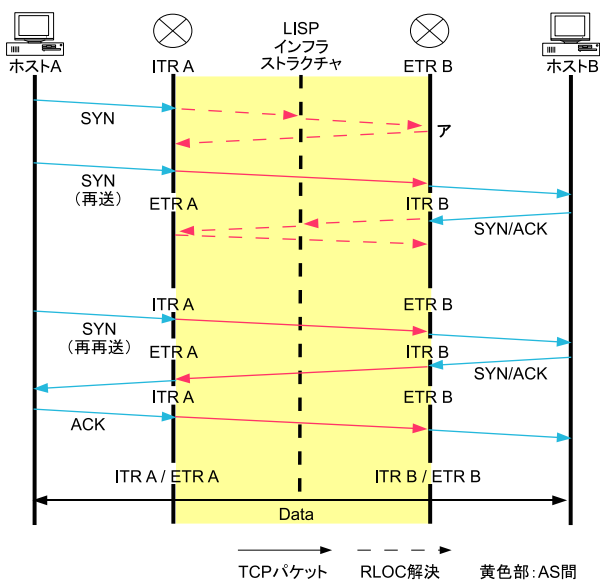


図 4: LISP 環境下の SYN Cookies 適用時における TCP コネクション

より抜粋して表 2.1 に示す。この値は、バルセロナのサーバーから 2012 年 2 月 2 日に観測されたものである。RLOC 解決 Round-Trip-Time (RTT) とは、ITR が RLOC 解決のリクエストを送信してから、ITR にリプライが返ってくるまでの時間である。つまり、RLOC 解決に要する時間である。EIDs は EID 群を表す。

ここで、RLOC 解決 RTT が最大であった iij-xtr の RTT が 493ms である。双方向でこの時間を要したとしても、約 1s である。たとえば、ほとんどのパクリレーシステムは、コネクションが要求されても確立できない場合に要求を諦める時間を、はじめに SYN が送信されてから 75s に設定する。したがって、SYN Cookies の適用によって発生する遅延は、タイムアウトが発生してしまうほどの遅延ではないということが分かる。しかし、再送間隔の増加による遅延が、初期の AS 間通信となる TCP コネクション確立時に、必ず発生してしまうことになるので問題は残る。

3. SYN Cookies を活かす LISP

LISP は、RLOC 解決の際に First Packet Drop を行いエンドノードの再送を必要とするため、この再送を待つ時間の遅延が発生する。通常の LISP が RLOC 解決が単方向であるため、それぞれの方向で RLOC 解決が必要となる。SYN Cookies 適用時の遅延の増加原因は、受信側の RLOC 解決である。したがって、要求側の RLOC 解決時において、一度に双方向に RLOC 解決を行うことを検討する。これにより SYN Cookies 適用時の遅延の増加を防止できる。

以下、双方向に RLOC 解決を行う手法について説明する。はじめに、IETF が提案する「piggybacked」について述べるが欠点があるため、次に、本稿で提案する新しい手法について説明する。

3.1. piggybacked

piggybacked はリクエストのオプションフラグである。このフラグが有効になっているリクエストを受信した場合、受信側ルータはリプライに加え、反対方向にリクエストを送信する。したがって、一度に双方向の RLOC 解決が可能となっている。ただし、このオプションは両側の LISP サイトエッジデバイスがこのオプションに対応している必要がある。これは、両側の LISP サイトエッジデバイスがリプライとリクエストの両方を送信できる xTR でなければならないことを意味する。

オプションの仕様上、piggybacked は、受信側の ITR と ETR が異なるデバイスで実装されている場合、実行できない。送信側がフラグを有効にしても、受信側が対応していなければ無視されてしまう。これでは LISP 環境下で TCP SYN Flood 攻撃が行われ、SYN Cookies が適用されたときに、遅延が避けられない場合がある。そのため、受信側のデバイスに関係なく双方向に RLOC 解決できる手法が必要である。

3.2. デバイスに依存しない手法の提案

RLOC 解決に用いるリクエストのフォーマットを図 5 に示す。このフォーマットに示される様に、リクエストパケットには Source EID Address が格納されている。これは、送信元エンドノードの EID である。また、

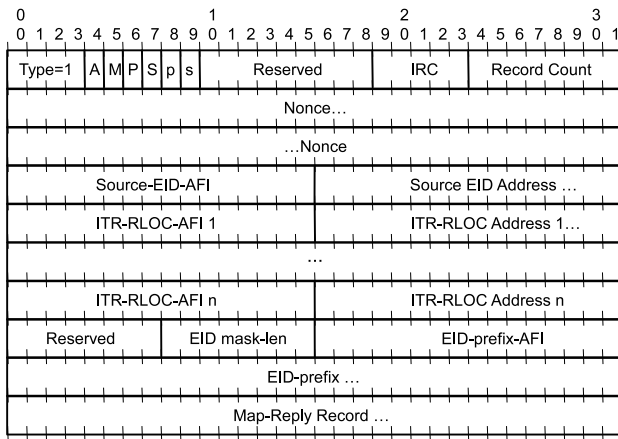


図 5: リクエストパケットフォーマット

ITR-RLOC Address として、送信元エンドノードに対応する ITR の RLOC も格納されている。したがって、受信側 LISP サイトエッジデバイスは、受信したリクエストから Source EID Address と ITR-RLOC Address を抽出することで逆方向の RLOC 解決を行うことができ、双方向 RLOC 解決を実現できる。しかし、実際にリクエストからの情報抽出によって RLOC 解決を行った場合、問題が発生する。

リクエストパケットから情報抽出を行うと、一方的に送られてきた情報を安易に受け入れることになる。例えば、リクエストパケットの送信元 RLOC または送信元エンドノードの EID のどちらかが偽装されている場合でも、LISP サイトエッジデバイスが誤ったキャッシュを安易に作成することになる。このキャッシュは、AS 単位のやりとりの情報であるから、大規模な通信障害を引き起こすことになる。したがって、受信したリクエストから RLOC 解決を行うことは推奨できない。そこで、送られてきたリクエストパケットの情報を確認する動作を加える。

図 4 を用いて、提案する受信側 LISP サイトエッジデバイスのリクエスト強制による、双方向の RLOC 解決を実現する手法を説明する。

通常通り ITR A から RLOC 解決のリクエストが、LISP インフラストラクチャ経由で、ETR B に送信される。ETR B はリプライを ITR A に送信する。このとき (図 4 ア)、ホスト B が所属する AS は単方向に RLOC 解決が行われることを認識する。そこで、ITR B からホスト A 宛てのパケットを送信するための RLOC 解決を強制させる。ITR B にリクエストを強制送信させることで、双方向の RLOC 解決を実現する。この段階で RLOC 解決をすることで、ITR B はホスト B からホスト A 宛てのパケットが送信されてくる前に RLOC 解決を済ませておくことができ、実際にパケットが送られてきた際には、破棄することなく転送できる。よって、問題としてあげた遅延が発生しない。

ITR B にリクエストを強制送信させる手法としては、リクエストを受信した ETR B にリプライに加え、ホスト A 宛ての ICMP Echo Reply を送信する。これに

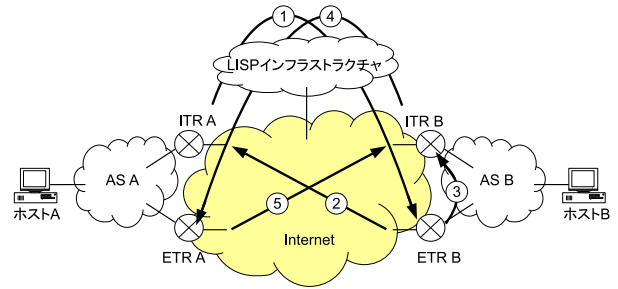


図 6: 双方向 RLOC 解決のためのリクエスト強制送信

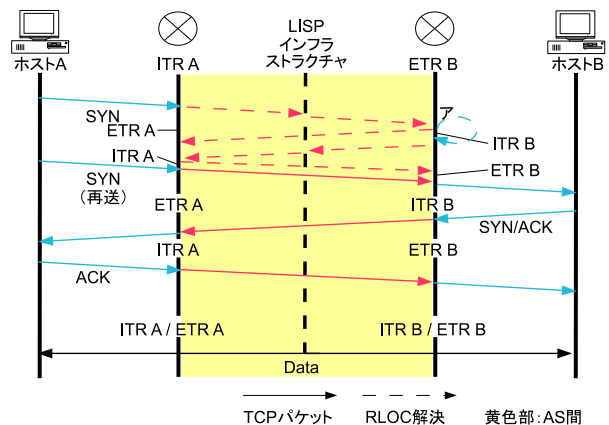


図 7: 提案手法の LISP 環境下の TCP コネクション

よって、ITR B から強制的にリクエストが送信されることになる。図 6 に示す。(3) が ICMP Echo Reply である。(1), (4) はリクエスト, (2), (5) はリプライである。

この手法を用いた LISP 環境下における TCP コネクションのタイムラインを図 7 に示す。この図 7 のアにおいて、ICMP Echo Reply を送信することで、双方向 RLOC 解決を実現している。

受信側 LISP サイトエッジデバイスに逆方向の RLOC 解決を行うリクエストを強制することで、前節では安易に受け入れてしまった情報をリプライが返ってきて初めて信頼する。受信したリクエストから抽出した情報が偽装されていた場合、リプライが返送されることはないため前節より信頼できる手法となった。

4. 考察

本稿では双方向 RLOC 解決を受信側 LISP サイトエッジデバイスに強制的にリクエストを送信させることで実現した。このリクエストの強制は、受信側 ETR が対応するエンドノードに向けて ICMP Echo Reply を送信することで実現していた。この ICMP Echo Reply パケットについて詳しく考察する。

また、piggybacked と提案手法の比較を行う。

4.1. 提案手法における ICMP Echo Reply

提案手法では、受信側 ITR に強制的にリクエストパケットを送信させるために、受信側 ETR が ICMP Echo Reply を送信する様にした。この ICMP Echo Reply パケットの詳細について述べる。

図 6 を用いて説明する。前述したが、図 6 (3) が ICMP Echo Reply である。宛先アドレスはホスト A の EID となっている。したがって、AS 内に宛先がないことが分かるため、AS 間通信の入口となる ITR B へ AS B 内を通して転送される。このとき、ITR B の保持しているキャッシュ情報にホスト A に対応する RLOC 情報が存在するかによって、ICMP Echo Reply の行方が異なる。

- RLOC 情報が存在しない場合
ICMP Echo Reply が到達すると、ITR B がリクエストパケットを送信すると共に、ICMP Echo Reply を破棄する (First Packet Drop)。
- RLOC 情報が存在する場合
ICMP Echo Reply の転送先 ETR を把握できているため、カプセル化して RLOC 空間へパケットを流す。

つまり、RLOC 情報が存在しない場合は、目的である ITR にリクエストの強制を行い、それと同時に ICMP Echo Reply は破棄される。一方、RLOC 情報が存在する場合には、ICMP Echo Reply は宛先エンドノードへ通常通り送信されるため、RLOC 空間へ流出することになる。これは提案手法の目的としては不必要なフローである。ただし、ICMP Echo Reply パケットが非常に小さいパケットであるため許容できると考えられる。

提案手法が ICMP Echo Request ではなく ICMP Echo Reply を用いている理由は、ICMP Echo Request の場合、それに対する ICMP Echo Reply が発生し、不必要なフローが増えてしまうからである。したがって、あえて ICMP Echo Reply を一方的に送りつける手法を提案する。

4.2. piggybacked

piggybacked は送信側 ITR が目的のパケットが双方向通信のパケットであるか判断ができるという利点がある。提案手法では、単方向通信の場合においても、双方向に RLOC 解決を強制する。これは、無駄な RLOC 解決を行うことであり、Internet に必要のないパケットを流すことになる。piggybacked は双方向のときのみフラグを有効にすることで、無駄な RLOC 解決をせずに済む。ただし、一般的に通信は双方向であることがほとんどである。双方向 RLOC 解決は、あらかじめ両側にキャッシュを作成できるため、SYN Cookies 適用時のみならず双方向通信全般に受信側の RLOC 解決による遅延を防ぐことができる。したがって、単方向を考慮するよりも、双方向通信時の遅延防止に利益があると考えられる。

導入を考えた場合においては、piggybacked は受信側の LISP サイトエッジデバイスが対応している必要

があることから、ITR と ETR を xTR として全体に導入しなければならない。しかし、提案手法は、受信側の LISP サイトエッジデバイスをアップグレードするのみで実現できる。したがって、適用したい AS が自身で LISP サイトエッジデバイスをアップグレードするのみでよい。

5. おわりに

本稿では、SYN Cookies を活かす LISP 導入法の検討を行った。現状の LISP 環境下で SYN Cookies を用いると、TCP コネクション確立に要する時間は、連続する SYN の再送により大きな遅延を要した。原因としては、現状の LISP が単方向に RLOC 解決をするため、受信側でも RLOC 解決の際に First Packet Drop を行い、受信側エンドノードにも再送が必要となるからである。そこで、送信側の単方向の RLOC 解決が行われる際に、受信側 ITR に逆方向の RLOC 解決も実行させるリクエストの送信を強制させた。これにより、双方向に RLOC 解決が行われ、SYN Cookies 適用時の大きな遅延の解消に加え、双方向通信全般の遅延短縮につながった。

LISP 環境下で TCP SYN Flood 攻撃が行われた際、エンドノードは提案手法を用いることで SYN Cookies による対策が可能となるが、LISP インフラストラクチャに関しての負荷は対策できていない。今後はそれについても考慮する必要がある。

参考文献

- [1] LISP monitoring platform
<http://lispmon.net/>
- [2] Locator/ID Separation Protocol
<http://tools.ietf.org/html/draft-ietf-lisp-22>
- [3] RFC 4987 : TCP SYN Flooding Attacks and Common Mitigations
<http://tools.ietf.org/html/rfc4987>
- [4] Motoyuki OHMORI, Koji Okamura, Kohei HAYAKAWA, and Fuminori TANIZAKI
Analyses on First Packet Drops of LISP in End-to-End Bidirectional Communications
Internet Conference 2011