G-31

SRv6 パケットの経路可視化を目的とする SRH への識別子付与に関する研究

A Study of Assigning Identifiers to SRH for SRv6 Packet Path Visualization

嶋谷 修一朗†　柏崎 礼生‡　井口 信和‡§
Shuichiro Shimatani　Hiroki Kashiwazaki　Nobukazu Iguchi

1. 序論

5G ネットワークの制御やデータセンターネットワークのスライシングに Segment Routing over IPv6 (SRv6) が利用されている[1-3]．SRv6 は，セグメントのリストによりシンプルで柔軟なネットワーク制御を可能にする．ネットワークのデバッグにおいて，パケットのドロップ箇所特定にはパケットの追跡が必要な場合がある[4]．しかし，SRv6 ネットワークでは，パケットを追跡することが難しい．これは，SRv6 のトンネリングの機能や ECMP の負荷分散によって，traceroute といった従来のツールでは対応できないことがあるためである．このため，パケットをドロップしている場所の特定はそれぞれのノードでパケットをキャプチャする必要があり，ネットワークにおけるデバッグの負荷が高いといえる．

そこで本研究では，SRv6 ネットワークのデバッグ支援を目的に，Segment Routing Header (SRH) に識別子を付与することでパケットを追跡しパケットの経路を可視化するシステム (以下，本システム) を開発する．本システムではパケットの識別子によりパケットを関連付けるため，正確なパケットの経路を得ることが出来る．本システムにより，SRv6 ネットワークの検証作業における負荷の削減やネットワークの動作を容易に把握できることが期待される．

2. SRv6

SRv6 は，IPv6 ネットワークにセグメントルーティングを適用したものである．セグメントルーティングは，ソースルーティングを活用したパケット制御技術である．セグメントルーティングでは，セグメントと呼ばれる命令のリストによりパケットを制御する．セグメントの識別子は Segment ID (SID) と呼ばれ，SRv6 での SID は IPv6 アドレス形式で表される．SRv6 の SID は，ノードにルーティングするための情報である Locator と SID が表すノードでの処理内容を示す Function，Function で使用する情報である Argument からなる．

SRv6 では，SRv6 の SID リストを IPv6 拡張ヘッダである SRH に含める．SRH のフォーマットを図 1 に示す．図 1 のように SRH には，Type Length Value (TLV) 形式のオブジェクトを配置できるフィールドが存在する．この TLV のフィールドは，Segment List の整合性を検証するための情報やパケットのタイムスタンプといった情報を格納する場合に使用される．本研究では，この SRH の TLV を用いてパケットへ識別子を付与する．

† 近畿大学大学院 総合理工学研究科,
Graduate School of Science and Engineering, Kindai University.
‡ 近畿大学 情報学部,
Faculty of Informatics, Kindai University.
§ 近畿大学 情報学研究所,
Cyber Informatics Research Institute, Kindai University.

Next Header	Hdr Ext Len	Routing Type	Segments Left
Last Entry	Flags	Tag	
Segment List [0] (128bits IPv6 address)			
...			
Segment List [n] (128bits IPv6 address)			
Optional Type Length Value object (variable)			

図 1 SRH のフォーマット

3. 関連技術・関連研究

Path Tracing は，SRv6 ネットワークにおいてパケットの転送経路や各ホップ間の遅延といった情報を取得するための技術であり，IETF の Internet Draft で提案されている[5]．Path Tracing では，経由したインターフェース ID やタイムスタンプの情報を IPv6 拡張ヘッダである Hop-by-Hop オプション内のスタックに追加する．また，タイムスタンプ情報を SRH に TLV として配置する．その情報を読み取ることで，パケットの経路や各ホップ間の遅延の情報を得る．Path Tracing では，40 バイトの Hop-by-Hop オプションヘッダで最大 14 つのホップまでトレースできる．パケットの経路やタイムスタンプの情報は，Path Tracing のパケットを受信するノードであるシンクノードから得る．そのため，経路の途中でパケットがドロップした場合，ドロップした場所を特定することは難しい．

それに対し本研究では，IPv6 パケットを処理するスイッチ・ルータが，経路を計算するサーバに情報を提供する方式を採用している．そのため，ホップ数が 14 以上であってもトレース出来る．また，新たな IPv6 拡張ヘッダを追加する必要がなく，それぞれのノードでパケットを取得するため，パケットがドロップした場所の特定が容易である．

NetSight は，ネットワーク障害を診断するためにパケットの履歴やスイッチの状態といった情報を得ることができるプラットフォームである[6]．NetSight では，パケットがスイッチを通過するたびに，パケットのヘッダやマッチしたフローエントリが含まれたポストカードと呼ばれる情報をサーバに送信する．サーバでは，正規表現のような言語である Packet History Filter を使用して，パケットの履歴を検索出来る．パケットの経路を得るために，IPv4 ヘッダの Identification と fragment offset，TCP シーケンス番号などの不変なフィールドの値を用いてパケットを識別している．この識別方法では，SRv6 パケットに対応しているとはいえない．

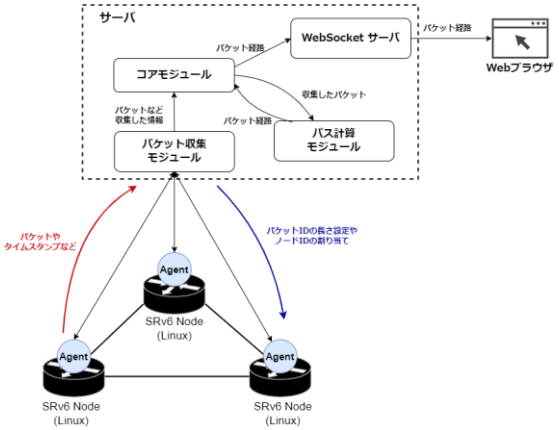

図 2 システム構成

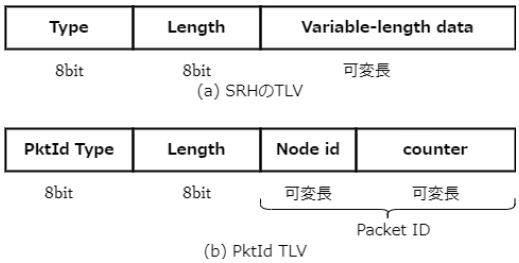

図 3 SRH の TLV と PktId TLV

それに対し本研究では，SRv6 パケットに識別子を付与するため全ての SRv6 パケットを識別できる．また，識別子のみでパケットを識別するので，パケットを関連付ける実装が容易である．

4. 研究内容

本章では，まず本システムのシステム構成について述べ，次にパケットへの識別子付与，パケットの収集と分析，パケットの経路可視化，トポロジ情報の入力について説明する．

4.1 システム構成

システム構成を図 2 に示す．エージェントは，パケットへ識別子を付与しパケットをサーバに送信するためのプログラムであり，SRv6 ノードで動作させる．サーバは，エージェントからパケットを受け取った後，パケットの経路を分析し，Web ブラウザで可視化するための WebSocket API を提供する．また，サーバはエージェントに対して，パケット識別子の長さやノード ID を設定する．エージェントとサーバのやり取りは gRPC[1]を用いる．

4.2 パケットへの識別子付与

各エージェントで行われる識別子付与について説明する．パケットの識別子は，識別子を含む TLV オブジェクトを SRH に配置することで付与する．このために，SRH の TLV オブジェクトとしてパケット識別子用の TLV (以下，PktId TLV) を新たに定義した．SRH で使用される TLV 形式と PktId TLV の形式を図 3 に示す．図のように SRH の TLV は，Type と Length，可変長の data からなる．PktId TLV において，可変長の data はパケットの識別子であり，Node ID と counter から構成される．Node ID はノードの識別子であり，サーバから割り当てられる．counter は，パケットに識別子を付与するたびに 1 が加算される．パケットを識別できれば良いため，Node ID と counter はそれぞれ可変長であり，ネットワークの規模に応じて利用者が設定できる．ただし SRH の長さのバイトが 8 オクテットである必要があり，そうでなければ パディング TLV を利用する必要がある．そのため本システムでは，PktId TLV のデフォルト長さを 64bit としている．PktId Type は，PktId TLV を識別するためのもので，本システムでは IANA で予約された実験用の Type である 124 を使用している[2]．

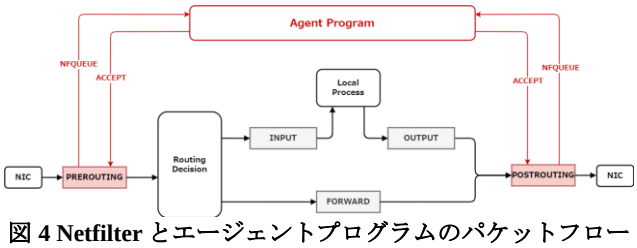

図 4 Netfilter とエージェントプログラムのパケットフロー

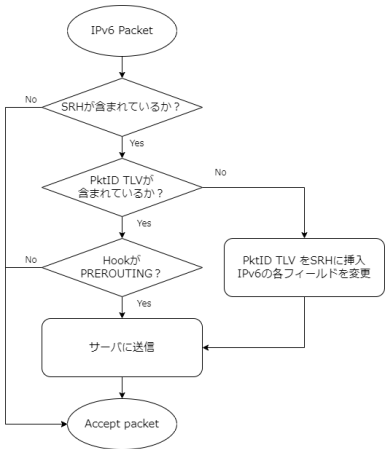

図 5 パケット処理のフロー

次に，識別子付与方法について説明する．本システムでは，Linux を用いて識別子を付与する．パケットを取得し変更するために，Linux の iptables のルールに一致したパケットをユーザスペースで処理することが可能なライブラリである libnetfilter_queue[3]の Python モジュールを使用した．SRH は Local Process で処理される[7]．なので，パケットを取得するためのルールを配置するチェーンは PREROUTING と POSTROUTING とする．Netfilter とエージェントでパケットをやり取りする様子を図 4 に示す．対象のチェーンでルールに一致したパケットは，キューを介してエージェントプログラムに送信される．エージェントプログラムはパケット取得後，図 5 のフローに従ってパケットを処理する．本システムでは，IPv6 パケットをエージェントプログラムに送信するようなルールをデフォルトで追加しているため，エージェントプログラムは IPv6 パケットをキューから取得する．IPv6 パケットを受け取ったエージェントプログラムは，SRH が含まれていて PktId TLV が存在しない場合に PktId TLV を追加する．PktId TLV を追加するとパケットのサイズが大きくなるため，IPv6 ヘッダの Payload Length と SRH の Hdr Ext Len をそれぞれ書き換える．その後，取得したパケットとそのタイムスタンプ，ノード ID をサーバに送信すると同時にパケットを Netfilter に再投入する．

[1] gRPC – An RPC library and framework
https://github.com/grpc/grpc

[2] Internet Protocol Version 6 (IPv6) Parameters
https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml

[3] The netfilter.org "libnetfilter_queue" project
https://netfilter.org/projects/libnetfilter_queue/

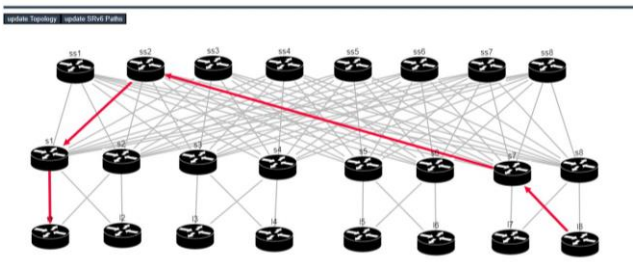

図 6 経路可視化の様子

本システムでは，iptables にルールを追加してパケットを取得する．このため，特定のパケットがマッチするようなルールを使用することで，経路を取得するパケットを選択することができる．

4.3 パケットの収集と分析

サーバに対して送信されたパケットやタイムスタンプの情報は，リストとして保持する．同じパケット ID のパケットを関連付けるため，パケットのリストをパケット ID でソートする．パケットの経路を取得するときには，パケット ID で関連付けられたパケットをタイムスタンプで並び替える．このようにすることで，ノードを通過した順にパケットを取得できる．また，パケットをキャプチャしたノードの情報を取得することで，パケットが経由した順のノードを経路として取得できる．

4.4 パケット経路の可視化

パケットの経路は，Web ブラウザでトポロジ上に可視化する． Web ブラウザでは，収集したパケットの一覧が表示され，クリックするとそのパケットの経路が表示される．可視化の様子を図 6 に示す．図のように，パケットの経路が赤い矢印で表示される．

4.5 トポロジ情報の入力

本システムでは，サーバがエージェントに対して接続するための IP アドレスとポート番号をそれぞれ入力する必要がある．また，トポロジを GUI で表示するためにノード間の接続情報やノードを識別するための名前の設定も必要である．これらの入力は，yaml ファイルを用いて設定することができる．

5. 実験

本章では，今後予定している実験について説明する．実験は，システムの動作確認とパケットへの識別子付与が遅延やスループットに与える影響の評価を予定している．

5.1 動作確認

動作確認は，本システムで正確にパケットの経路が得られるかを確認する．パケットが通過するノードを全て指定した SRv6 のルートをルーティングテーブルに設定し，システムで得られた経路と比較する．実験で使用するネットワークは，40 のノードを持つ Clos トポロジのネットワークで Mininet[4]を用いて構築する予定である[8].

5.2 識別子付与による遅延とスループットへの影響の評価

パケットへの識別子付与がネットワークに与える影響を評価するために，識別子付与による遅延とスループットの変化を測定する．評価するための構成を図 7 に示す．この

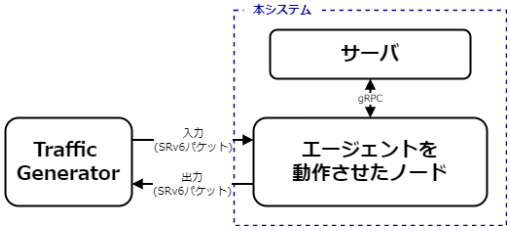

図 7 評価の構成

図のように，エージェントを動作させたノードにトラフィック生成器を接続して，パケットの遅延とスループットを測定する．この評価では，iptables にルールを追加しない SRv6 パケットの処理の場合とパケットに識別子を付加しサーバにパケットを送信する場合について測定し，結果を比較する．

6. 結論

本稿では，SRv6 ネットワークのデバッグ支援を目的に，SRH に識別子を付与することでパケットを追跡しパケットの経路を可視化するシステムについて説明した．本システムでは識別子によりパケットを関連付けるため，正確なパケットの経路を得ることが出来る．本システムにより，SRv6 ネットワークの検証作業における負荷の削減やネットワークの動作を容易に把握できることが期待される．

今後，予定している実験を実施するとともに，ルーティングテーブルとパケットのマッチや SRv6 の動作などの可視化できる情報を増やすことを予定している．また，キャプチャしたタイムスタンプを用いた遅延測定や得られたパケットの経路をテストするためのインターフェースを作成することを検討している．

参考文献

[1] Filsfils, C., Previdi, S., Ginsberg, L., et al.: RFC8402 Segment Routing Architecture, (2018)
[2] Filsfils, C., Dukes, D., Previdi, S., et al.: RFC8754 IPv6 Segment Routing Header (SRH), (2020)
[3] Filsfils, C., Camarillo, P., Leddy, J., et al.: RFC 8986 Segment Routing over IPv6 (SRv6) Network Programming, (2021)
[4] Zhu, Y., Kang, N., Cao, J., et al.: Packet-Level Telemetry in Large Datacenter Networks, In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15), pp. 479--491 (2015)
[5] Filsfils, C., Abdelsalam, A., Camarillo, P., et al.: draft-filsfils-spring-path-tracing-01 - Path Tracing in SRv6 networks, (2022)
[6] Handigol, N., Heller, B., Jeyakumar, V., et al.: I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks, 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pp. 71--85, (2014)
[7] Lebrun, D., and Bonaventure, O.: Implementing IPv6 Segment Routing in the Linux Kernel, Proceedings of the Applied Networking Research Workshop (ANRW'17), pp.35—41 (2017)
[8] Clos, C.: A study of non-blocking switching networks, The Bell System Technical Journal, vol. 32, no. 2, pp. 406--424,(1953)

[4] Mininet: An Instant Virtual Network on Your Laptop (or Other PC) http://mininet.org/