G-32

SRv6 を用いた低遅延トラフィックエンジニアリング手法の検討

Consideration about Low-latency Traffic Engineering Method using SRv6

中田 清登 †　　柏崎 礼生 §　　井口 信和 ‡ §
Kiyoto Nakata　　Hiroki Kashiwazaki　　Nobukazu Iguchi

1. 序論

自動運転，遠隔医療，映像配信，eSports といったリアルタイムな処理を必要とするアプリケーションやサービスが増加している．これらのアプリケーションでは，msec 単位のわずかな遅延がユーザ体験やシステム全体の処理時間に影響を与える可能性がある．具体例として，自動運転では，自動運転車の遅延が緊急ブレーキの反応時間に影響を与えてしまう．遠隔医療では，遠隔手術において医師の操作とロボットの反応に遅延が生じると，手術の精度が低下してしまう．映像配信では，ライブスポーツ配信において遅延が大きくなると映像と音声のずれなどにより視聴体験が低下してしまう．eSports では，競技中に遅延が発生するとプレイヤーの反応時間に影響を及ぼし，試合の公正性が損なわれてしまう．このような遅延に敏感なアプリケーションの通信要件を満たすために，より低遅延な経路を選択的に経由しながら，トラフィックを送受信する必要がある．

しかし，従来のインターネットで普及しているホップバイホップルーティングの方式だけでは，低遅延の経路や広帯域の経路といった，詳細な通信要件を満たす経路制御は困難である．ホップバイホップルーティングにおいて，代表的なプロトコルである Routing Information Protocol(RIP) や Border Gateway Protocol(BGP)，Open Shortest Path First(OSPF)といったプロトコルがあるが，いずれも最適経路を選択する時に，ホップ数や帯域幅などが考慮され，遅延は考慮されない．

通信要件を満たす経路制御のためにいくつかの技術が存在する．Policy-Based Routing (PBR) はルータの経路表ではなく，ルールマップを使用することで，送信元アドレスやポート番号といった条件に合致したパケットの送出インターフェースを制御できる．Multi-layer Protocol Label Switching (MPLS) はパケットに MPLS ヘッダを追加し，経路上のルータは MPLS ヘッダのラベル情報を基にラベルの追加，交換，除去などの処理を行うことで経路制御が可能となる．OpenFlow は Software Defined Networking (SDN) の実現手法の 1 つであり，フローテーブルと呼ばれるルールとアクションの組み合わせを示すテーブルを OpenFlow コントローラで一元管理することで，複雑な経路制御が可能となる．しかし，これらの手法は経路上の各ルータやコントローラがパケットの処理方法が定義された情報を保持しておく必要があり，パケットの処理方法を多く定義すれば負荷の増大によるスケーラビリティの問題や情報の複雑化といった問題が生じる．これらの課題を解決しながら柔軟な経路制御が可能になる手法として，送信元のルータが宛先に到達するまでに経由する経路を指定できるセグメントルーティングという技術が注目されている．

† 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.

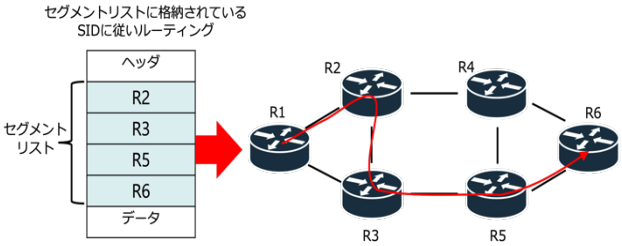

図 1:セグメントルーティングによる経路誘導

セグメントルーティングでは，転送手順を示す SR Policy をセグメントリストとして送信元のルータでパケットに付与する．セグメントリストに応じて経路上の各ルータはパケットの処理をするだけでいいので，経路制御のために情報を保持しておく必要がなくなる．そのため、各ルータの経路制御のための情報が削減され，柔軟でシンプルな経路制御が可能となる．セグメントルーティングを活用してトラフィック状況に応じた低遅延な経路制御を実現するためには，SR Policy を動的にネットワーク機器に付与することによる，動的な経路制御の仕組みが必要である．

そこで本研究では，トラフィック状況の変化に対応しながら，低遅延経路の利用を可能にすることを目的として，トラフィック状況に応じた動的な SR Policy の作成と付与を可能とするシステム(以下，本システム)を提案する．本システムでは，リンク間遅延値などのトラフィック状況を監視することで得られる測定データを基に SR Policy を動的に作成し付与する．本システムを用いることにより，トラフィック状況が変化しても低遅延の通信要件を満たす経路制御が可能となる．

2. SRv6

Segment Routing over IPv6 (SRv6) はセグメントルーティングを実現するためのデータプレーンに IPv6 を利用したものである[1-3].

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技術であり，送信元がパケットを転送する経路を決定できる．経路は最短経路だけではなく，送信元で指定する任意の要件に従って選択できる．送信元によって指定された各経路は，セグメントリストとして定義され，パケットに付与される．各セグメントは特定のノードを示しており，セグメントの値によって特定のリンクからパケットを転送するといった単純な処理から，より複雑に定義された処理まで，さまざまな処理が定義できる．各パケットの転送に必要な状態はセグメントリストに含まれており，セグメントは Segment ID(SID)として各ノードから広告される．パ

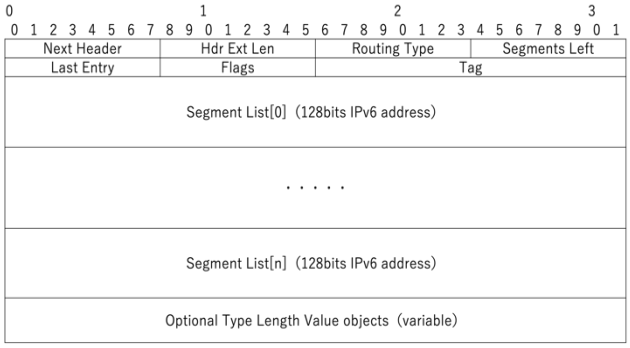

図 2：SRH のフォーマット

ケットがセグメントを広告しているノードに到着すると，そのノードはセグメントリストを更新し，次のセグメントを参照する．そして，セグメントに関連する処理を実行する．これらの処理を最終的な宛先に到達するまで繰り返す．セグメントルーティングによる経路制御の様子を図 1 に示す．セグメントリストには各ノードを一意に示す SID が格納されており，図 1 の場合，R2，R3，R5，R6 の順で各ノードを経由していき，セグメントリストに格納されている最後の SID である R6 に到達する．

セグメントルーティングでは，経路上の各ルータがパケットの中継経路を決定しながら宛先まで転送するルーティング方式であるホップバイホップルーティングでは対応できない，ユーザ定義の柔軟な経路制御が可能となる．セグメントルーティングのユースケースとして，トラフィックエンジニアリング(TE)やサービスファンクションチェイニング(SFC)などが挙げられる．本システムでは，セグメントルーティングのユースケースの 1 つである TE を用いることにより，低遅延な経路を選択して経路を制御する手法を提案する．

次に，Segment Routing Header (SRH)のフォーマットを図 2 に示す．SRH は，SRv6 による通信を実現するためにパケットに挿入されるヘッダである．SRH にはセグメントリストなどの経路制御に必要な情報が含まれており，IPv6 拡張ルーティングヘッダを利用する．Segment List フィールドに宛先に到達するまでに経由したいノードの SID を格納することで Segment List の値を順番に経由していく経路制御が可能となる．

3. 関連研究

Carmine らの研究では，Software Defined-Wide Area Network(SD-WAN)サービスにおいて，SRv6 ベースのエンドツーエンドの遅延監視手法を提案している[4].

SD-WAN は，地理的に分散した場所を持つ企業に，柔軟かつ効率的な方法でサービスを提供するために使用される技術である．SD-WAN では，ネットワーク運用を最適化するために管理下の各ノードの性能劣化や障害を検出するためのネットワーク監視技術が必要である．そこで，Simple Two-Way Active Measurement Protocol(STAMP)[5]と呼ばれる，エンドツーエンドで専用のセッションを確立することで監視を実現するアクティブモニタリング技術を活用することにより，SRv6 ドメイン上のルータ間でエンドツーエンドの遅延監視を実現する手法を提案している．

それに対し本研究では，SRv6 ドメイン上で In-band Network Telemetry(INT)[6]を活用して，各ノードの情報を収集して監視する．INT は流れているトラフィックに対して INT 用のパケットヘッダを挿入する．そのヘッダに収集した情報を格納することで監視を可能にするパッシブモニタリング技術である．INT では監視用のセッションを確立する必要はなく，リンク間遅延値といった，エンドツーエンド以外の遅延値も計測可能である．また，本研究では測定した情報を基に最適な低遅延経路を算出し，SR Policy を作成することが可能である．

Pierpaolo らの研究では，SRv6 ネットワークのパフォーマンスモニタリング(PM)のためのクラウドネイティブアーキテクチャである SRv6-PM を提案している[7].

SRv6 を IP バックボーンやデータセンターで大規模に展開するためには，パケット損失や遅延を監視するパフォーマンスモニタリング手法が必要とされている．そこで，Alternate-Marking Method[8] や A Two-Way Active Measurement Protocol (TWAMP)[9]を拡張し，SDN コントローラと Big-Data ツールを組み合わせることで，統合的な SRv6 ネットワークのパフォーマンス監視，特にリアルタイムパケット損失監視に焦点を当てた手法を提案している．

それに対し本研究では，INT を用いて監視をしており，データプレーンの監視のみに特化している．また，遅延監視に焦点を当てており，詳細なリンク間遅延の算出による SR Policy の作成が可能である．

N. L. M. van Adrichem らの研究では，OpenFlow でコントロールされた SDN ネットワーク内のフローごとのスループット，遅延，パケットロスを監視するためのアプローチおよびオープンソースソフトウェア実装である OpenNetMon を提案している[10].

OpenNetMon は，エッジスイッチと OpenFlow コントローラ間のポーリングレートを最適化することで，フローデータの問い合わせを最小限に抑えることができる．この適応的なレートによって，ネットワークおよびエッジスイッチの CPU のオーバーヘッドを削減しながら，測定精度を最適化できる．

それに対して本研究では，INT を用いることで各ノードと SDN コントローラ間で直接測定データのやり取りをせずに，INT ヘッダに格納された測定データをまとめてサーバへ送信できる．

4. 提案手法

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 INT による測定データの収集，収集情報に基づくリンク間遅延値算出，パス計算，SR Policy の作成，遅延閾値超過時の SR Policy 付与について説明する．

4.1 システム構成

本システムのシステム構成を図 3 に示す．本システムはデータプレーンに Programming Protocol-independent Packet Processors (P4)[11]言語でデータプレーンの処理動作を定義した P4 Switch，コントロールプレーンに Open Network Operating System (ONOS)[12]と呼ばれるオープンソースの SDN コントローラを用いている．また，測定データを収集し，測定データを基に通信経路の遅延の監視や SR Policy の作成を行うサーバを配置する．パケットを転送する時に測定データを収集するための INT ヘッダを挿入し，INT ヘッダに格納されたタイムスタンプなどの測定データをサーバに送信する．サーバは受信したデータを整形し，リンク間遅延値を算出する．リンク間遅延値を基にパス計算，SR

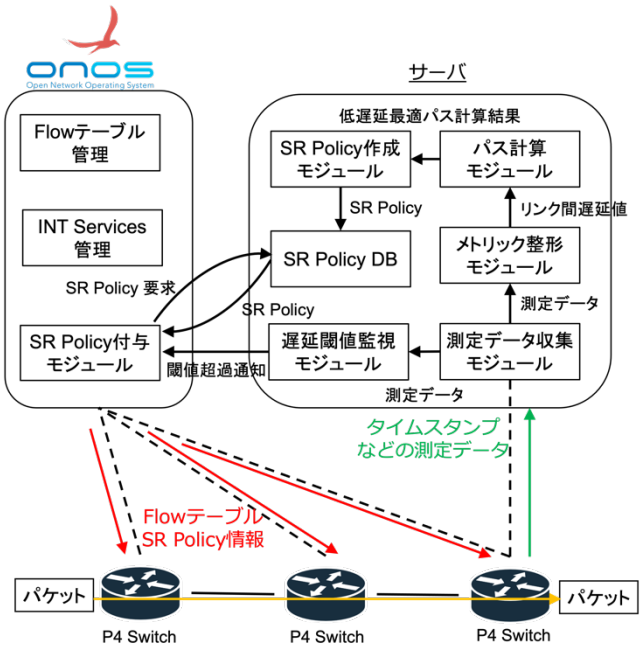

図 3:システム構成

Policy 作成を行い，ONOS コントローラを通じて各ノードに SR Policy を付与する．

4.2 INT による測定データの収集

経路上の P4 Switch で行われる INT による測定データの収集について説明する．INT はパケットの転送経路で経由するネットワーク機器のテレメトリメタデータをリアルタイムで収集する技術である．テレメトリメタデータとして，各ノードの ingress/egress タイムスタンプ，キュー専有率，パケットを受信してから送信するまでの遅延値，出力ポートのパケットカウント数，ノード ID などのデータが挙げられる．本システムでは，テレメトリメタデータを測定データとしてサーバに送信し，様々な処理に利用する．INT の動作概念を図 4 に示す．INT によるテレメトリメタデータの収集を開始するノードを Source ノード，宛先に到達するまでに経由するノードを Transit ノード，INT によるテレメトリメタデータの収集を終了するノードを Sink ノードと呼ぶ．Source ノードではパケットに INT ヘッダとテレメトリメタデータが挿入される．Transit ノードでは INT ヘッダが挿入されているパケットに対してテレメトリメタデータを挿入する．Sink ノードでは INT ヘッダとテレメトリメタデータを分析用のサーバへ送信するとともに，INT ヘッダとテレメトリメタデータを削除し，ネクストホップへ元のパケットが送信される．また，INT によるテレメトリメタデータは送信元 IP アドレス，宛先 IP アドレス，送信元ポート番号，宛先ポート番号，L4 プロトコルの情報に基づき，アプリケーションごとに測定できる．測定データ収集モジュールは，測定データを受け取るためのモジュールであり，InfluxDB を用いて時系列順にデータベースに測定データを格納する．本システムでは測定データ収集モジュールのデータベースから ingress タイムスタンプ，egress タイムスタンプ，ノード ID を利用して各ノードのリンク間遅延値を算出するために利用する．ingress タイムスタンプは P4 Switch のインターフェースからパケットを受信する瞬間の値を取得するものであり，egress タイムスタンプは P4 Switch のインターフェースからパケットを送信する瞬間の値を取得するものである．タイムスタンプは 48bit の μ sec 単位で取得される．

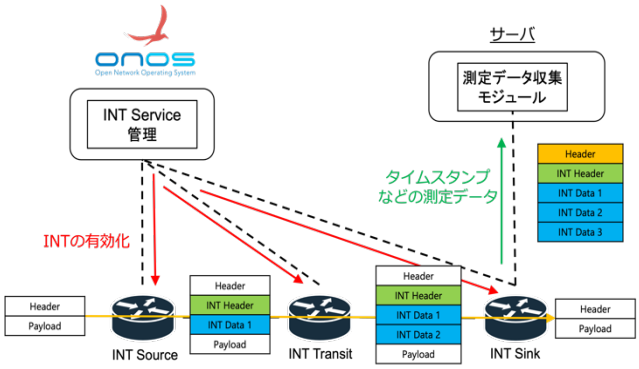

図 4:INT の動作概念

4.3 収集情報に基づくリンク間遅延値算出

収集した測定データに基づくリンク間遅延値算出について説明する．例として，ノード ID が S1，S2 の P4 Switch があると仮定する．パケットが S1 から S2 に転送される場合，リンク間遅延値は S1 の egress タイムスタンプの値と S2 の ingress タイムスタンプの値の差分を取ることで取得できる．メトリック整形モジュールでは，経路上の前後のノードの ingress/egress タイムスタンプの値を測定データ収集モジュールの時系列 DB から参照し，リンク間遅延値を算出する．この時，ingress/egress タイムスタンプは 48bit の μ sec 単位で取得されているため，計算が煩雑になってしまう．そこで，計算する前にタイムスタンプの値を 1,000 で割り，有効数字 3 桁の msec 単位に整形する．計算結果は低遅延最適パスを求める時のメトリックとして利用するため，パス計算モジュールに送信する．

4.4 パス計算

各ノードのリンク間遅延値をメトリックとしたパス計算の方法について説明する．計算方法として，4.3 節で求めたリンク間遅延値を重みとして SRv6 の開始ノードから終了ノードまでの最短距離をダイクストラ法で算出する．算出結果が低遅延最適パスとなる．パス計算のイメージを図 5 に示す．パス計算モジュールでは python を用いたダイクストラ法の実装と SR Policy 作成モジュールへの計算結果送信部分を実装する．

4.5 SR Policy の作成

SR Policy の作成について説明する．SR Policy の作成には 4.4 節で求めたパス計算の結果を活用する．低遅延最適パスで経由するノードのノード ID に対応する SID をセグメントリストに順に格納していく．作成した SR Policy はアプリケーションとアプリケーションごとの要件を満たすように設計された SR Policy との対応関係を示す SR Policy DB に格納する．この時，作成前から SR Policy DB に SR Policy が格納されていた場合，その SR Policy を上書きする形で新しい SR Policy を格納する．この方法により，低遅延最適パスを通る SR Policy を動的に最適化できる．SR Policy DB のイメージを図 6 に示す．SR Policy の用途とは，SR Policy による経路制御がどのアプリケーションに使用されているかを示すものである．図 6 を見ると，各アプリケーションに対応する SR Policy のセグメントリストが格納されている．

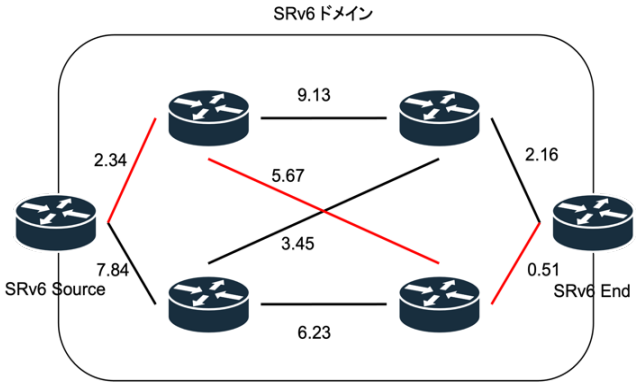

図 5：遅延値をメトリックとしたダイクストラ法

SR Policyの用途	SR Policyのセグメントリスト
app1	2001:abcd::1，2001:abcd::3
app2	2001:abcd::2，2001:abcd::4

作成したSR Policyで上書き

SR Policyの用途	SR Policyのセグメントリスト
app1	2001:beef::1，2001:beef::3
app2	2001:beef::2，2001:beef::4

図 6：SR Policy DB の最適化

4.6　遅延閾値超過時の SR Policy 付与

4.5 節で作成した SR Policy を P4 Switch に付与する方法について説明する．動的な SR Policy 付与のためにアプリケーションごとに遅延閾値を設定することで，閾値を超えた場合に SR Policy 付与モジュールへ通知するという仕組みを実装する．通知を受信した SR Policy 付与モジュールは SR Policy DB から新しい SR Policy を取得する．SR Policy の付与は ONOS コントローラを拡張することにより実現する．ONOS コントローラから P4 Switch に現在付与されている SR Policy の削除と新しい SR Policy を付与するための情報を通知する．

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．実験として，本システムが正常に動作していることを確かめるための動作確認を予定している．また，本システムを用いた場合に従来のルーティング手法と比較して遅延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，本システムが閾値を超える遅延を計測した場合に，新しい SR Policy が正しく付与されることを確認する．実験環境として，Mininet を用いた仮想ネットワーク環境を作成する予定である．Mininet では，遅延の変更をシミュレーションでき，特定のリンク間遅延を制御可能である．

5.2　従来のルーティング手法との比較

本システムの性能評価として，従来のルーティング手法と比較して遅延に与える影響を確認する．比較対象として，現在のネットワーク環境で多く利用されている OSPF と BGP を予定している．これらのプロトコルを用いた実験環境を作成し，本システムの有無が遅延に与える影響を評価する．実験環境として，Mininet を用いた仮想ネットワーク環境を作成する予定である．

6. 結論

本稿では，トラフィック状況の変化に対応しながら，低遅延経路の利用を可能にすることを目的として，トラフィック状況に応じた動的な SR Policy の作成と付与を可能とするシステムについて説明した．本システムでは，リンク間遅延値などのトラフィック状況を監視することで得られる測定データを基に SR Policy を動的に作成し付与する．本システムを用いることにより，トラフィック状況が変化しても低遅延の通信要件を満たす経路制御が可能となる．

今後，SR Policy 作成・付与のためのモジュールを実装する予定である．また，予定している実験を実施することにより，本システムが従来の経路制御技術と比較して，より低遅延経路の継続的利用が可能であることを実証する．

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] Carmine Scarpitta , Giulio Sidoretti , Andrea Mayer , et al . : High Performance Delay Monitoring for SRv6-Based SD-WANs , IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 21, NO. 1, FEBRUARY 2024 , pp.1067—1081
[5] G. Mirsky , G. Jun , H. Nydell , et al . : RFC 8762 - Simple Two-Way Active Measurement Protocol , (2020)
[6] In-band Network Telemetry (INT) Dataplane Specification <https://p4.org/p4-spec/docs/INT_v2_1.pdf> (参照日:2024/07/26)
[7] Pierpaolo Loreti , Andrea Mayer, Paolo Lungaroni , et al . : SRv6-PM: A Cloud-Native Architecture for Performance Monitoring of SRv6 Networks , IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, VOL. 18, NO. 1, MARCH 2021 , pp.611—626
[8] A. Capello , M. Chen , G. Mirsky , et al . : RFC 8321 - Alternate-Marking Method for Passive and Hybrid Performance Monitoring, (2018)
[9] K. Hedayat , R. Krzanowski , A. Morton , et al . : RFC5357 - A Two-Way Active Measurement Protocol (TWAMP) , (2008)
[10] Niels L. M. van Adrichem, Christian Doerr and Fernando A. Kuipers , OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks , IEEE Network Operations and Management Symposium (NOMS) , pp.1—8 (2014)
[11] P4 – Language Consortium , <https://p4.org/ > , (参照日:2024/07/26)
[12] Open Network Operating System (ONOS) SDN Controller < https://opennetworking.org/onos/> , (参照日:2024/07/26)