Vol.2021-CSEC-93 No.18
Vol.2021-IOT-53 No.18
2021/5/14

SRv6を用いたアプリケーションの特性を考慮した通信経路制御手法

杉浦 智基[1,a)]　高橋 慧智[1,b)]　市川 昊平[1,c)]　飯田 元[1,d)]

概要：ネットワーク通信において，最適な通信経路はアプリケーションの特性に応じて変化する．VoIP をはじめとする遅延時間に性能が大きく左右されるものは遅延が少ない経路が最適であり，ファイル通信を伴うものは高帯域幅を確保できる経路が最適であると言える．一方で既存のルーティングプロトコルはこれを考慮しない．本研究では SRv6 を用い，アプリケーションの特性とネットワークの通信状況に応じた動的な経路制御手法を提案した．また，仮想マシンを用いた実験環境に提案システムを適用し，その有効性を確認した．

キーワード：SRv6，Segment Routing，SDN，Traffic Engineering

1. 諸言

ネットワーク上の通信方法は，アプリケーションによってその性質が異なるため，アプリケーションごとに最適な経路は異なる．例えば，VoIP を用いるものは広い通信帯域を確保できる経路よりも遅延が少ない経路の方が最適であり，一方ファイル通信を伴うものはその逆である．このようにアプリケーションごとに最適となる経路が異なる一方，BGP[1] や OSPF[2] といった既存のルーティングプロトコルはこれらを考慮せず，送信元と宛先間で設定される経路コストの最小化のみを制御する．複数経路が存在するときの経路選択の手法として ECMP[3] が挙げられるが，これもアプリケーションの特性を考慮せず，複数経路への通信の機械的な分散のみを制御する．本研究ではこのような課題を解決すべく，アプリケーションの特性に応じた，動的な通信経路の決定と制御による通信効率の向上を目的とする．

今日では Software Defined Networking（SDN）という概念が普及している [4]．SDN ではネットワークをソフトウェア的に扱うことで，柔軟かつ詳細な制御を可能とする．これにより，個々のアプリケーションの要求を反映したネットワーク制御も可能となる．SDN を実現する技術の 1 つとして Segment Routing がある [5]．Segment Routing では，制御対象ネットワーク内のノード，またはリンクに対して SID と呼ばれる ID を割り当て，送信パケットにこの ID のリストを付与することで経路を制御する．本研究ではこの SID の仕組みに IPv6 を用いた SRv6 を活用して，動的な通信経路の決定と制御を実現する．

以下，2 章では関連研究について述べ，3 章で提案システムに用いる SRv6 の概要について述べる．4 章ではシステムの設計について解説し，5 章でシステムの実装について述べる．6 章では実装したシステムの評価し，7 章でシステムについての考察をした後，8 章を結言とする．

2. 関連研究

現在までに，アプリケーションの特性を考慮した通信経路の最適化を試みる研究がいくつか報告されている．

Pongsakorn らは OpenFlow を用いたアプリケーションの特性を考慮した経路制御を提案した [6]．この研究では，事前にユーザが各アプリケーションをその特性に応じて分類し，OpenFlow を用いてネットワークの通信状況や各リンクの特性から最適経路を決定し各フローを制御している．しかし，この手法は OpenFlow の特性から，アプリケーション数に比例してフローテーブルが肥大化する点や，中央集権的なコントローラが各スイッチの管理をするため経路変更のコストが高いといった問題点があげられる．そのため，制御対象のネットワーク規模の拡大に限界がある．SRv6 には中央集権的なコントローラがなく，アプリケーションの転送を制御するテーブルサイズも OpenFlow のも

[1] 奈良先端科学技術大学院大学
NAIST, Takayama, Ikoma, Nara, 8916-5, Japan
[a)] sugiura.tomoki.sr2@is.naist.jp
[b)] keichi@is.naist.jp
[c)] ichikawa@is.naist.jp
[d)] iida@itc.naist.jp

のより小さくなるため，大規模なネットワークに対しても適用可能であることが期待できる．

William らは Opera というサーキットスイッチを用いた手法によって通信を最適化している [7]．この研究ではサーキットスイッチを用いて，定期的にネットワークトポロジを変化させ，低遅延を要求するアプリケーションと高帯域幅を要求するアプリケーションそれぞれについて経路を設定している．このとき，低遅延の通信パケットは直ちに送出し，高帯域幅を消費する通信は最小ホップ数の経路を選択させることでネットワーク全体に対する消費帯域幅を減らし，通信を最適化している．ただし，この手法の適用にはサーキットスイッチの導入という物理的な変更が要求される．一方，本提案システムは制御対象ネットワークの各ノードが SRv6 に対応していれば，物理的な変更を必要としない．また，現在 SRv6 は Linux カーネルをはじめとするいくつかのソフトウェアで対応しており，システムの導入障壁は比較的低いと言える．加えて，Opera と本提案システムを組み合わせることは技術的に可能であり，これらを組み合わせた手法も考えられる．

帯域幅を多く消費するフローであるエレファントフローについて着目し，その制御を試みる研究も行われている．Andrew らはアプリケーションが実行されているホストごとに通信の様子を観察，エレファントフローを検知し，経路制御する手法を提案した [8]．しかし，この研究はエレファントフローの検出手法に焦点が当てられており，経路制御には OpenFlow が用いられている．そのためこの手法も OpenFlow の特性から生じる課題を克服できない．ただし，提案システムはアプリケーション特性の指定をユーザの入力に頼っているが，この手法と組み合わせることでその特性を自動的に決定でき，より効果を発揮できることが期待される．

3. SRv6

SRv6 とは IPv6 の機能を用いた Segment Routing を実現する技術の 1 つである [9]．OpenFlow に代表される中央集権コントローラによりネットワーク通信を制御する手法は，制御対象ネットワークの拡大や通信フローの増加に伴い，コントローラの負荷が増大する欠点が存在する．一方で，個々の計算機で通信制御処理が完結する分散型の手法はこのような課題を克服できる．SRv6 をはじめとする Segment Routing はこの分散型の制御手法をとっており，またこれらはソースルーティングと呼ばれる送信元で通信経路を指定し制御する手法を用いている．Segment Routing ではネットワーク中のノードやそれらを結ぶリンクを Segment として扱い，制御対象ネットワークの入口でパケットに対して，SID と呼ばれる各 Segment に対応する ID を付与することで経路を制御する．SRv6 ではこの SID の表現に IPv6 のアドレス形式を適用し，IPv6 の拡張ヘッダを用いてこれら SID のリストを Segment Routing Header（SRH）としてパケットに付与する．パケット転送時は SID のリストから任意の SID を宛先アドレスとして設定し，各中継ノードがその宛先アドレスを参照し，ルーティングテーブルに基づいて転送するか，またはその SID に対応する処理が登録されている場合はその処理を実行する．そのためアプリケーションごとの通信制御は Segment Routing が適用されるネットワークの入口ノードで完結し，各中継ノードはアプリケーションの数や状態が変化しても保持すべきデータ量は変化しない．また，アプリケーションの通信経路を変更する際は，対応する入口ノードの SRH を付与する設定を変更することで完了する．このような特性により，中央集権的に経路を制御する手法と比較して SRv6 を用いた手法は，スケーラビリティに優れ，また経路設定の変更コストを低く抑えることができる．

4. 設計

本章では提案システムの設計について述べる．初めに提案システムの概要について述べた後，経路の決定方法，提案システムのアーキテクチャについて述べる．

4.1 概要

本提案システムは帯域幅を大きく消費するアプリケーションに対して，制御対象ネットワーク内の通信状況に応じて最適な経路を算出，その経路を使用するように通信を制御する．一方で，低遅延要求型のアプリケーションは制御の対象としない．これは消費帯域幅を考慮しない場合，既存のルーティングプロトコルによって設定される経路の通信遅延は他の経路と比較して小さいと期待され，また SRv6 の処理のために発生するオーバーヘッドが低遅延要求型のアプリケーションにとって無視できないと考えられるためである．

提案システムでは実行環境として，ネットワーク内ノードの各ネットワークインタフェースに SID を付与することを前提としている．従って，各パケットは任意の SID を指定することで，その SID に対応するネットワークインタフェースへ転送が行われる．そのため，提案システムは最適経路を算出した後，それに基づいて通過すべきネットワークインタフェースに対応する SID をリスト化し，パケットに付与することで経路を制御する．

4.2 経路の算出方法

本研究では最適化後の経路として，アプリケーションの要求帯域幅を満たす経路の内，最もホップ数の少ない経路と定義した．アプリケーションの要求帯域幅は，アプリケーションごとに設定ファイルを用いてユーザが事前に定義する．高帯域消費型のアプリケーションの通信を最短経路以外に流すことは，冗長な経路の長さだけ余計に帯域を

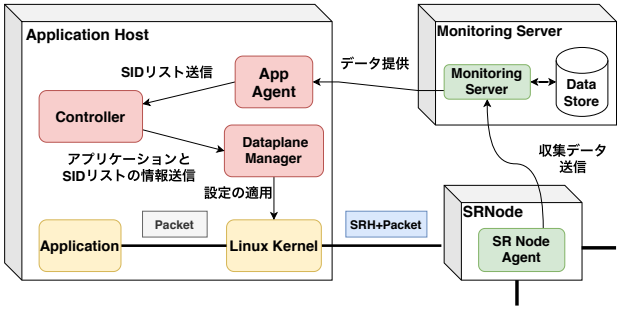

図 1 提案システムのアーキテクチャ

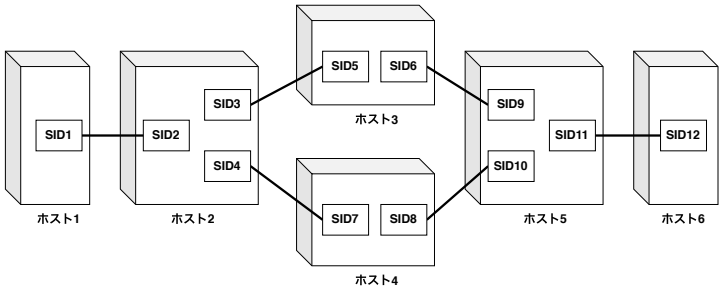

図 2 制御対象ネットワークの例

消費することになる．提案システムではこの余計な帯域幅消費を軽減させるため，最も空き帯域幅が大きい経路を選択するのではなく，要求する帯域幅を確保できる経路の中で最もホップ数が少ない経路を選択する手法をとった．

4.3 提案システムのアーキテクチャ

本提案システムのアーキテクチャを図 1 に示す．提案システムは「SRNodeAgent」,「MonitoringServer」,「Data Store」,「AppAgent」,「Controller」,「DataplaneManager」の 6 つの構成要素から成り立つマイクロサービスアーキテクチャの形態を採用した．各コンポーネント間の通信は RPC フレームワークの一種である gRPC を用いて実装した．

各ソフトウェアの役割は次の通りである．

- SRNodeAgent：制御対象ネットワーク内のパケット中継ノードから経路決定に必要な情報を収集する．
- MonitoringServer：SRNodeAgent から情報を受け取り Data Store に保存し，要求に応じてこれを提供する．
- Data Store：SRNodeAgent の収集した情報を保存する．
- AppAgent：MonitoringServer から情報を取得し，経路を算出する．
- Controller：AppAgent から送られた経路情報を DataplaneManager に転送する．
- DataplaneManager：Controller からの要求に応じて，Linux カーネルに対して設定を適用する．

5. 実装

本章では提案システムの実装について述べる．はじめに経路算出アルゴリズムについて述べた後，提案システムの構成要素の実装内容について述べる．

5.1 経路算出アルゴリズム

変更後の経路は SRNodeAgent が収集したデータを用いてグラフを作成し，ダイクストラ法を用いて算出される．制御対象ネットワーク内の各ノードをグラフのノードとして扱い，ネットワークのトポロジを反映するように各グラフのノードはリンクによって結ばれる．リンクのコストは，MonitoringServer から得られるネットワークリンクの空き帯域幅がアプリケーションの要求帯域幅を上回っていた場合 1 とし，そうでなければ十分に大きな値を設定する．また，経路算出の結果として SID のリストが生成される．例えば図 2 で，算出された経路がホスト 1，ホスト 2，ホスト 3，ホスト 5，ホスト 6 となった場合は，結果として出力される SID のリストは起点と終点を除き，SID3，SID6，SID11 となる．

5.2 SRNodeAgent

SRNodeAgent は制御対象ネットワーク内の各中継ノードに設置され，ノード内の各ネットワークインタフェースの 1 秒あたりの通信データ量の測定結果と，生成グラフにおける隣接する SID の情報を MonitoringServer へ送信する．通信データ量の測定には SNMP[10] を用いた．なお SNMP からは累計の通信データ量しか取得できず，単位時間あたりの通信量を取得できないため，本システムでは一定時間をおいて 2 回計測することで算出している．隣接ノードの SID 情報はユーザが事前に設定ファイルに記述し，SRNodeAgent がそれを読み取る方法をとった．

5.3 MonitoringServer

MonitoringServer は SRNodeAgent から送信された情報を Data Store に保存する．Data Store 内では各ネットワークインタフェースに割り当てられた SID をキー値として 1 秒あたりの通信量および隣接ノードの SID 情報を保存する．また，これらの情報を公開する WebAPI も実装した．

5.4 Data Store

提案システムでは Key/Value ストアの一種である Redis を使用した．Redis はデータの保存場所としてメモリを用いており，ストレージを用いるデータベースよりもデータの書き込みおよび参照が高速である．提案システムが対象としている環境ではデータ操作の所要時間に対して敏感であることから，このような特性をもつ Redis を採用した．

5.5 AppAgent

AppAgent は各制御対象のアプリケーションと一対一で動作する．つまり 1 つのアプリケーションにつき，1 つの

表 1　実験用仮想マシンのスペック

OS	Ubuntu 20.04
CPU	vSphere vCPU 2 コア
メモリ	8GB

表 2　実験用物理マシンのスペック

CPU	Intel(R) Xeon(R) Silver 4208 2 ソケット
メモリ	96GB
ネットワーク	10GbE

AppAgent が動作する．AppAgent は MonitoringServer に対して定期的に情報を要求し，その後 5.1 節で述べた手順に従い経路を算出する．経路を算出した後，その経路を通るよう SID のリストを生成し，Controller に送信する．

5.6 Controller

Controller は AppAgent から受信した SID リストを DataplaneManager に転送する．このとき，SID リストを受信した AppAgent と同一ホスト上で動作している DataplaneManager を転送先とする．

5.7 DataplaneManager

DataplaneManager は Controller からの要求に応じて，Dataplane に対して送信パケットに SRv6 のヘッダを付与する設定を適用する．本提案システムでは Dataplane として Linux カーネルを採用しており，Go 言語の netlink ライブラリである vishvananda/netlink[*1]を用いて SRv6 の設定適用部分を実装する．

6. 評価

仮想マシンを用いた実験環境で提案システムの有効性を評価する．ここでは提案システム導入前後のアプリケーションのレイテンシおよびスループットを比較することで，本手法の有効性を確認する．

6.1 実験環境

本実験では vShpere 上の 8 台の仮想マシンによって構成されたネットワークを構築した．ルータとする仮想マシンはそれぞれ別の物理マシンで実行し，アプリケーションが動作するホストは隣接するルータ仮想マシンと同一の物理マシン上で動作させる．図 3 にネットワークの構成を示す．また，表 1 に各仮想マシン，表 2 に各仮想マシンを実行する物理マシンの性能諸元を示す．

各仮想マシンを結ぶリンクは VXLAN を用いており 10Gbps の帯域幅をもつ．また実験のため，tc コマンドによりルータ 1，ルータ 2，ルータ 4，ルータ 5 間のリンクは 5Gbps に，ルータ 1，ルータ 3，ルータ 5 間のリンクは 1Gbps にそれぞれ帯域幅を制限した．各ルータではソフトウェアルータの一種である FRRouting[*2]を実行し，OSPF を用いて経路を広告させた．

6.2 実験内容

本実験ではホスト 1，ホスト 3 でアプリケーションが動作していると仮定し，それぞれホスト 2 に対して通信する．このときホスト 3 からの通信経路に対して提案システムを用いて経路を制御する．ホスト 3 のアプリケーション要求帯域幅を 500Mbps に設定し，ホスト 1 がこの要求帯域に影響がない通信を生成する場合と影響を生ずる通信を生成する場合で実験し，提案システムの挙動を評価する．そのため，ここではホスト 1 からホスト 2 へ 500Mbps で通信した際の提案システム導入前後の各ホストのレイテンシとスループット，ホスト 1 からホスト 2 へ 1Gbps で通信した際の提案システム導入前後の各ホストのレイテンシとスループットをそれぞれ計測する．また，レイテンシとスループットの計測はそれぞれ分けて行う．なお，ホスト 1 からホスト 2 への通信およびスループットの計測には iperf3 を用い，レイテンシの計測には ping を用いた．

6.3 実験結果

ホスト 1 からホスト 2 へ 500Mbps の通信を発生させた際のホスト 1 およびホスト 2 間，ホスト 3 およびホスト 2 間のレイテンシを図 4，スループットを図 5 に示す．図 4 を見ると提案システムの導入前後におけるレイテンシの値はおよそ 1 ミリ秒前後である．ホスト 1 の一部の計測時間では，比較的大きなレイテンシをとっているが，これは提案システム導入前後どちらにもみられるため，本システムの影響によるものとは考えにくい．図 5 を見るとはじめの数秒はホスト 3 からの通信のスループットがホスト 1 からのものと比較して大きいが，この傾向は提案システムの導入前後において変わらない．また，どのケースにおいても一定時間経過後は 500Mbps 前後のスループットに収束している．この実験ではホスト 3 からホスト 2 への通信経路において，要求帯域幅である 500Mbps の帯域が確保できているため，提案システム導入後も通信経路に変化がない．そのため導入前後でレイテンシ，スループット共に結果に大きな違いが生まれなかったと考えられる．

ホスト 1 からホスト 2 へ 1Gbps の通信を発生させた際のホスト 1 およびホスト 2 間，ホスト 3 およびホスト 2 間のレイテンシを図 6，スループットを図 7 に示す．図 6 を見ると，提案システム導入前は時間経過後もホスト 1 からの通信，ホスト 3 からの通信共にレイテンシの値に大きな変化はない．しかし提案システム導入後は，ホスト 1 からの通信のレイテンシに大きな変化はないが，ホスト 3 からの通信のレイテンシはおよそ 26 秒経過した後から大きく

[*1] https://github.com/vishvananda/netlink

[*2] バージョン 7.5.1 を使用

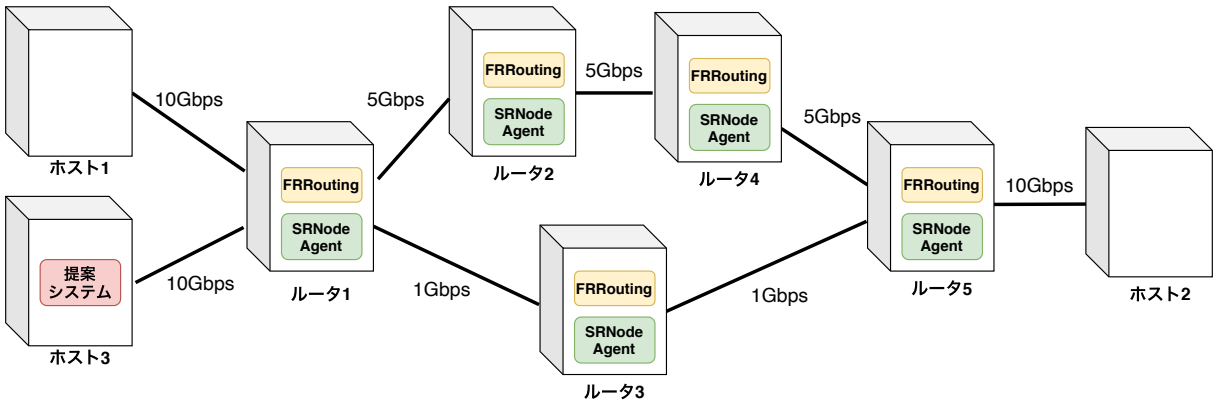

図 3 実験用ネットワークの構成図

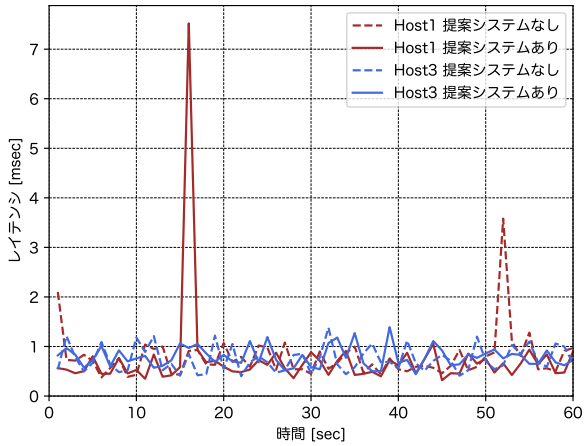

図 4 ホスト 1 からホスト 2 へ 500Mbps の通信を発生させたときのレイテンシ

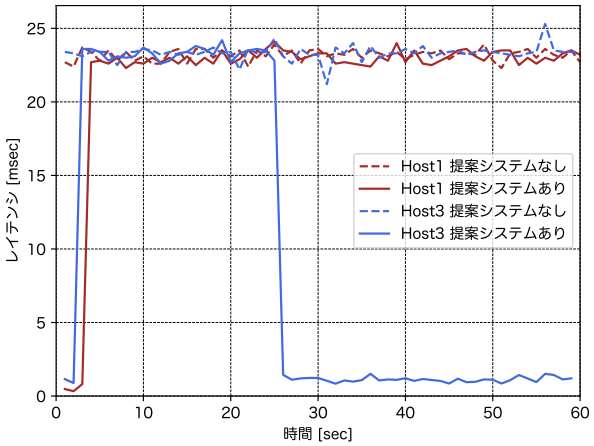

図 6 ホスト 1 からホスト 2 へ 1Gbps の通信を発生させたときのレイテンシ

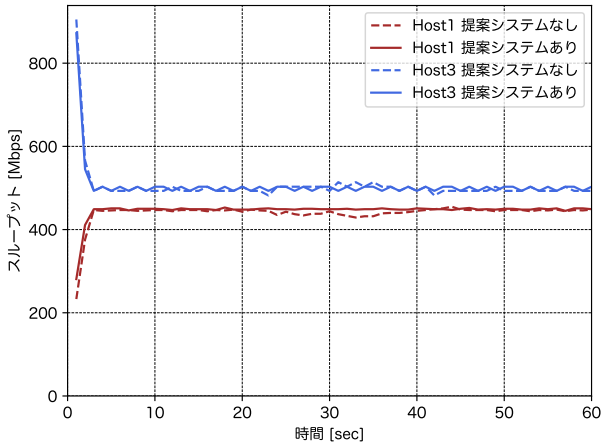

図 5 ホスト 1 からホスト 2 へ 500Mbps の通信を発生させたときのスループット

下がっている．図 7 を見ると，提案システム導入前はレイテンシのときと同様，ホスト 1 からの通信，ホスト 3 からの通信共にスループットの値に大きな変化はない．しかし，提案システム導入後はおよそ 24 秒経過した後からホスト 1 からの通信のスループット，ホスト 3 からの通信のスループット共に上昇している．この実験ではホスト 3 の要求帯域幅が満たされないため，ホスト 3 からホスト 2 への通信経路はルータ 2 経由の経路に切り替わる．そのためホスト 3 からの通信はレイテンシは低く，スループットは高くなり，ホスト 1 からの通信はスループットが高くなったと考えられる．ホスト 1 からの通信のレイテンシはホスト 1 からホスト 2 の経路における最小帯域幅である 1Gbps がすでに埋まっているために改善しなかったと推測される．また，ホスト 3 からの通信のスループットは本来であればルータ 2 経由の経路における最小帯域幅である 5Gbps となることが予想されるが実際はそれのおよそ半分程度の値となっている．これは，SRv6 適用時にパケットにヘッダが付与されることや，各ルータにおける SRH の ID の照合とそれに基づく SRH の操作のオーバーヘッドが存在するためだと考えられる．

7. 考察

7.1 収集メトリクスのリアルタイム性

提案システムでは，SNMP から単位時間あたりの通信データ量を直接取得できないために，10 秒程度の時間をおいて 2 回計測することで，メトリクスを算出している．そのため AppAgent で算出される経路は，その時点の制御対象ネットワークの状態を反映しているとは言えない．これ

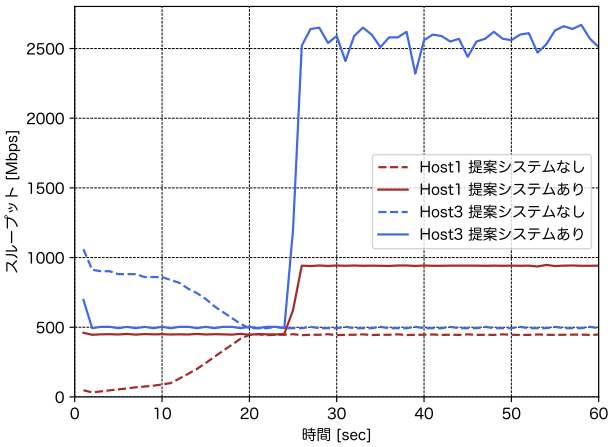

図 7 ホスト 1 からホスト 2 へ 1Gbps の通信を発生させたときのスループット

に対し，SRNodeAgent のメトリクス収集方法を前回のメトリクスも活用するように変更することで，2 回目以降は一度の計測でメトリクスの算出が可能となる．ただし既存の SNMP は通信データ量の反映に数秒の時間を要するため，経路計算時に用いられるメトリクスと実際の通信状況に数秒単位のずれが起きることは避けられない．一方で，本研究ではリアルタイムに実世界のネットワークの状態を反映できずとも，システムに対して致命的な影響は発生しない．また本システムは帯域幅を多く消費するアプリケーションを対象としており，これらのアプリケーションは通信完了までに通常，数秒から数分程度の時間を要すると考えられる．そのため，先に述べた経路反映までの遅延は通信完了時間と比較して無視してもよいと考えられる．

7.2 提案システムのスケーラビリティ

提案システムでは，各アプリケーションが実行されているホストは，そのアプリケーションの経路制御だけに集中すればよい．また，各中継ノードは自身のもつ SID とそれに対応するハンドラの組み合わせを管理すればよく，これはアプリケーションの通信内容には依存しない．ゆえに経路算出から各中継ノードでの通信制御までのプロセスにおいては，アプリケーションの数や中継ノードの数が増加することによって，システムの許容範囲を超えた負荷は発生しないと考えられる．

一方，アプリケーションや中継ノードの数が増えることによって，Data Store 内のデータ量やそれに伴う AppAgent，MonitoringServer，SRNodeAgent 間の通信量も増加することが予想される．ただしこれらは，シャーディングによるデータの分割保管や Data Store の読み取り専用レプリカを AppAgent が動作するホストに設置することなどの対策をとることで回避できると考えられる．また SRNodeAgent のメトリクス収集は算出メトリクスに変更がない場合は MonitoringServer への送信をしないといった動作を導入することで，負荷を下げることができると考えられる．

8. 結言

本研究では通信遅延と消費帯域幅に注目したアプリケーションの特性に応じて，SRv6 を用いた通信経路制御の手法を提案した．提案システムは既存研究で用いられた OpenFlow に対して，テーブルサイズ肥大化の回避，経路変更にかかるコストの削減という利点をもつ．また提案システムの有効性を仮想マシンを用いた実験で確認した．

提案システムは分散型の制御手法を採用しているため，制御対象のアプリケーションやネットワークの拡大に対して柔軟に対応できることが期待される．そのため，今日普及しているマイクロサービスアーキテクチャ型のアプリケーションへの適用や Kubernetes をはじめとするコンテナオーケストレーションシステムとの組み合わせ等の応用が期待される．

参考文献

[1] Rekhter, Y., Hares, S. and Li, T.: A Border Gateway Protocol 4 (BGP-4), RFC 4271 (2006).
[2] Moy, J.: OSPF Version 2, RFC 2328 (1998).
[3] Hopps, C.: Analysis of an Equal-Cost Multi-Path Algorithm, RFC 2992 (2000).
[4] Yassine, A., Rahimi, H. and Shirmohammadi, S.: Software defined network traffic measurement: Current trends and challenges, *IEEE Instrumentation Measurement Magazine*, Vol. 18, No. 2, pp. 42–50 (online), DOI: 10.1109/MIM.2015.7066685 (2015).
[5] Filsfils, C., Nainar, N. K., Pignataro, C., Cardona, J. C. and Francois, P.: The Segment Routing Architecture, *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6 (online), DOI: 10.1109/GLOCOM.2015.7417124 (2015).
[6] U-chupala, P., Watashiba, Y., Ichikawa, K., Date, S. and Iida, H.: Application-aware network: network route management using SDN based on application characteristics, *CSI Transactions on ICT*, Vol. 5, No. 4, pp. 375–385 (online), DOI: 10.1007/s40012-017-0171-y (2017).
[7] Mellette, W. M., Das, R., Guo, Y., McGuinness, R., Snoeren, A. C. and Porter, G.: Expanding across time to deliver bandwidth efficiency and low latency, *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA, USENIX Association, pp. 1–18 (2020).
[8] Curtis, A. R., Kim, W. and Yalagandula, P.: Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection, *2011 Proceedings IEEE INFOCOM*, pp. 1629–1637 (online), DOI: 10.1109/INFCOM.2011.5934956 (2011).
[9] Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S. and Voyer, D.: IPv6 Segment Routing Header (SRH), RFC 8754 (2020).
[10] Fedor, M., Schoffstall, M. L., Davin, J. R. and Case, D. J. D.: Simple Network Management Protocol (SNMP), RFC 1157 (1990).