Vol.2014-HPC-145 No.1
2014/7/28

SDNによるマルチパスを用いたGridFTPによるデータ転送高速化手法の提案

黄 掣[†1,a)]　市川 昊平[†2,b)]

概要：大量のデータを分析・計算することによって科学技術上の問題の解決を図る計算科学の分野では，大量のデータをある拠点から別の拠点に高速に転送する技術が求められている．特に，グリッドコンピューティング分野では，拠点間データ転送技術に関して盛んに研究され，現在までに GridFTP などの技術が提案されている．GridFTP はある拠点間におけるデータ転送に複数の TCP ストリームを使用する並列データ転送方式を利用し，高速化を実現している．これは単一経路上における TCP のスロースタートやパケットロスなどによるスループットの低下を，複数の TCP ストリームを利用することで軽減することによる高速化技術である．本研究は，この GridFTP が用いる複数 TCP ストリームを単一の経路ではなく，複数の経路に分散させるトラフィックエンジニアリングを行うことにより，使用可能なネットワーク帯域を増加させる転送手法を提案する．具体的には Software-Defined Network (SDN) を実現する代表的実装である OpenFlow を用い，複数 TCP ストリームの経路割当を制御するシステムを構築する．本論文は構築したシステムの設計・実装について説明し，その動作および性能評価の結果を示す．

キーワード：SDN，OpenFlow，GridFTP，マルチパス，データ転送

A Method Using Multipath over SDN for Improving GridFTP Transfer

Huang Che[†1,a)]　Ichikawa Kohei[†2,b)]

Abstract: A large amount of scientific data needs to be transferred from one site to another as fast as possible in the computational science field. High-speed data transfer between sites is very important, especially in the Grid computing field; GridFTP has been widely used for bulk data transfer over wide area network. GridFTP achieves greater performance by supporting parallel TCP streams. Using parallel TCP streams improves the throughput of slow-start algorithm and lossy network even on a single path. This research proposes a traffic engineering technique that increases the data transfer performance by using multiple paths simultaneously for the parallel TCP streams. For the purpose, we use Software-Defined Network (SDN) technology and its implementation, OpenFlow. This paper presents the design and implementation of the proposed system and describes the results of the performance evaluation.

Keywords: SDN，OpenFlow，GridFTP，multipath，data transfer

1. はじめに

今日のビックデータ時代では，大量のデータの処理が必要となっている．科学研究では計算機の進展に伴い，取り扱うデータがペタバイト級に至るまで大容量化している．また，今日の国際的な共同研究においては，データの生成とそれを分析する計算機が一つの拠点に集約されることは

†1 現在，大阪教育大学大学院教育学研究科総合基礎科学専攻
Presently with Pure and Applied Sciences Graduate School of Education Osaka Kyoiku University
†2 現在，奈良先端科学技術大学院大学 情報科学研究科
Presently with Graduate School of Information Science, Nara Institute of Science and Technology
a) j139610@ex.osaka-kyoiku.ac.jp
b) ichikawa@is.naist.jp

なく，大規模なデータをある拠点から別な拠点に高速に転送することが基盤のサービスとして求められている [1].

特に，グリッドコンピューティングの分野では，拠点間の高速データ転送に関する研究が盛んであり，現在までに GridFTP という高速データ転送のためのシステムが開発されている．GridFTP はグリッドコンピューティングを実現するミドルウェアである Globus Toolkit [2] の中の一つのコンポーネントとして実装されており，広く使われている．現在では，Globus Online と呼ばれる SaaS 形態 (Software as a Service) のデータ転送サービスとしても GridFTP は用いられ，科学技術計算において重要な役割を果たしている [3].

GridFTP はある拠点間におけるデータ転送に複数の TCP ストリームを使用する並列データ転送方式を利用し，高速化を実現している．TCP はスロースタートにより輻輳が起こらないように制御しているため，通信開始直後は輻輳ウィンドウサイズが小さく，スループットがすぐには高くならない．また，パケットロスが発生した際にも，スループットが元の状態に戻るまでに時間を要する．GridFTP の複数 TCP ストリームによる並列データ転送は，同時に複数のコネクションを張ることで，一定時間内にスループットを高めることで転送速度の向上を図っている．

GridFTP が対象とする拠点間データ通信では，通常，拠点間を結ぶ複数のネットワーク経路（マルチパス）が存在する．ただし，GridFTP による複数 TCP ストリームは通常の IP 経路情報に従ってルーティングされるため，基本的にはその複数経路の中の一つの最短経路のみを用いて通信される．アプリケーションレイヤで実行される GridFTP からはこのようなネットワークレイヤの経路情報等を把握する方法はなく，複数経路を同時に利用するトラフィックエンジニアリングを実現することによる性能向上の余地を大きく残している．

一方で，近年，SDN (Software-Defined Network) と呼ばれる，ソフトウェアでネットワークの構成・動作を動的に制御する概念が提唱され始めている．SDN は，従来，ネットワークスイッチ間の分散アルゴリズムで解決されていた経路制御に対し，プログラマブルネットワークという概念を導入し，ソフトウェアから動的に経路選択を制御可能なネットワークの実現を目指している．したがって，SDN によってアプリケーションレイヤからの要求に応じて，ネットワークレイヤの制御を動的に行える基盤が整いつつある．

本研究では，このような現状を鑑み，GridFTP が用いる複数 TCP ストリームによる並列データ転送を，SDN 技術を用いたトラフィックエンジニアリングにより複数のネットワーク経路に分散し，データ転送速度の向上を実現するシステムを提案する．具体的には，Open Networking Foundation [4](ONF) によって，標準化が進められている OpenFlow [5] という SDN を実現する技術を用い，GridFTP と連携して動作するネットワーク制御システムを研究開発する．

本稿の構成は，以下の通りである．2 節では，GridFTP の高速化に関連する既存研究について説明する．3. 節では，我々が提案する OpenFlow を用いた SDN 技術によるマルチパスを用いた GridFTP のデータ転送高速化手法を説明する．4. 節では，提案システムの評価実験について述べ，本システムの有効性に関して述べる．5. 節では，本論文のまとめとこれからの課題について述べる．

2. 関連研究

本節では，現在までに取り組まれてきた GridFTP におけるデータ転送高速化に関連する研究について述べる．

Kissel らは Internet2 などの学術研究ネットワークが提供する動的ネットワーク (DCN: Dynamic Circuit Network) を活用するため，新たにセッションレイヤのプロトコルをベースとした Phoebus を開発し，GridFTP から利用可能にした [6]．Phoebus はセッションレイヤのプロトコルの背後に TCP や UDT など複数の異なるトランスポートレイヤのプロトコルを隠ぺいする Phoebus Gateways (PGs) を提供し，容易にデータ転送速度を改善可能なネットワークシステムを一般のアプリケーションに対し提供している．

また, Dan らは前述の Kissel らの研究を発展させ, OpenFlow による SDN 技術を利用して，従来の IP ベースの通信路，Phoebus の提供する動的ネットワーク回線や NDDI (Network Development and Deployment Initiative) の OS^3E が提供する経路上に GridFTP の複数 TCP ストリームの通信を割り当てることでデータ転送速度の向上を実現するシステムを提案した [7]．NDDI/OS^3E は NDDI が構築する OpenFlow 基幹回線上に動的にレイヤ 2 経路を確保するサービスを提供する．

Kissel らの研究は Internet2 の DCN が提供する高速な回線や，その上での複数のトランスポートプロトコルを利用可能にしている点は興味深いが，DCN が構築する Internet2 内の経路に関してはシステム上から細かに制御する方法はない．したがって，GridFTP の複数 TCP ストリームのそれぞれをどのような経路でルーティングするかを制御する方法は提供されていない．Dan らの研究では NDDI/OS^3E による OpenFlow ネットワーク上におけるレイヤ 2 ネットワークを利用しているが，本サービスも Internet2 の DCN と同様，動的に経路を確保するものの，その経路制御は NDDI/OS^3E 内に閉じている．そのため，Dan らのシステムは NDDI/OS^3E によって確保された回線や IP 通信路など利用可能な経路に GridFTP の複数 TCP ストリームを割り当てるのみで，拠点を結ぶ各スイッチ上の経路を細かに制御するものではない．

しかしながら，アプリケーションレイヤの要求とそれに

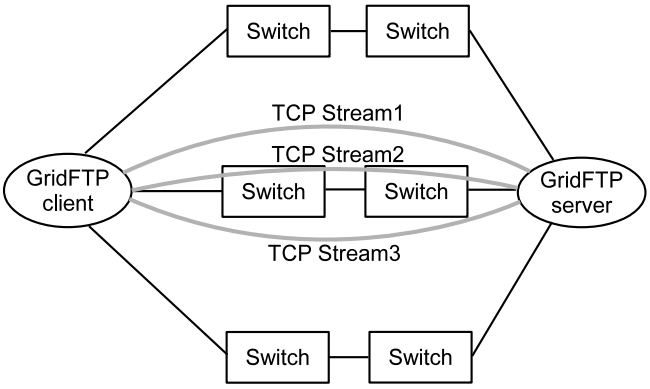

図 1 従来の GridFTP の並列転送

応じたネットワークレイヤの最適化に対する期待は高まっており，近年の OpenFlow ネットワークの普及に伴って，アプリケーションレイヤにも OpenFlow スイッチ自身の制御を仮想的に開放する JGN- X の RISE のようなサービスなどが出現してきている [8]．このような現状を鑑みると，エンド・ツー・エンドで OpenFlow ネットワークが利用可能な環境において，複数経路を制御・割り当てし，アプリケーションのパフォーマンスを最大化するような仕組みの構築が必要であると考えられる．

3. 提案手法とシステム設計

本研究では，エンド・ツー・エンドで OpenFlow ネットワークが利用可能な環境において，GridFTP が有する並列データ転送機能における複数の TCP ストリームに対し，複数の異なる経路を個別に割り当てることによって，データ転送の高速化を実現するシステムを提案する．

複数の拠点間でデータ転送を実施する場合，拠点間を接続する経路は複数あるが，通常の IP ベースの通信ではそのうちの最短路一つのみが使用されるのが普通である．従来の GridFTP の並列転送は図 1 に示すように，その一つの経路において，複数の TCP ストリームで通信することで，その経路の帯域を十分に活用し，データ転送の高速化を目指している．しかし，実際のネットワークでは，回線速度が制限されており，一つの経路において，複数の TCP ストリームを用いて通信しても速度の向上には限界がある．

提案する GridFTP の並列転送は図 2 に示すように，OpenFlow スイッチからなる OpenFlow ネットワークにおいて，複数の TCP ストリームをそれぞれ異なる経路で通信させることで，複数経路で利用可能な帯域を束ね，データ転送速度を向上させることを目指す．本研究では，OpenFlow ネットワークを制御する OpenFlow コントローラにエンド・ツー・エンド間で利用可能な経路を幅優先探索アルゴリズムに基づいて動的に計算させ，アプリケーションの要求に応じて順次，経路を割り当てるシステムを構築する．

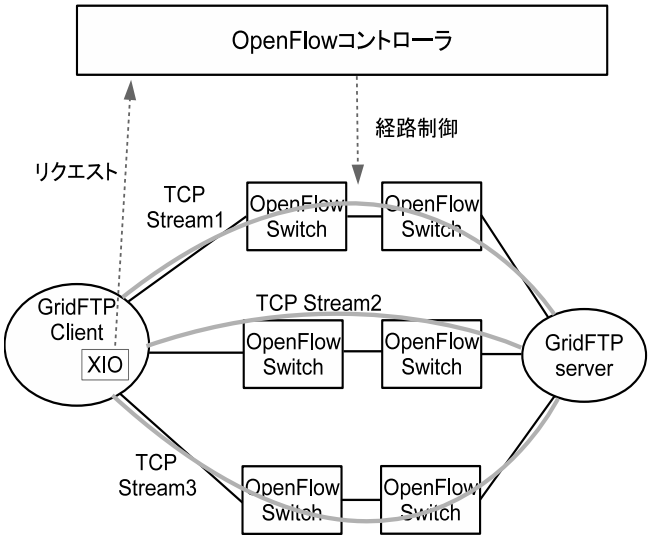

図 2 提案する GridFTP の並列転送

本研究の目的は GridFTP のデータ転送速度の向上であるが，システムの設計としては GridFTP に依存した方式にせず，以下の汎用的な 2 つの機能を提供するインタフェースを実装することで実現する．具体的には 1) ある拠点間において指定された本数分の経路を探索し，その経路を確保する機能，2) アプリケーションからの TCP コネクション開始の要求に応じて， 経路上の OpenFlow スイッチにフローエントリルールを配備する機能を提供する．あるホスト間で複数の TCP ストリームを確立する場合，宛先 IP アドレス，送信元 IP アドレスの組み合わせは TCP ストリーム間で同一である．そのため，複数の TCP ストリームを区別するプロパティは送信元の TCP ポート番号のみになる．TCP ポート番号は通常，TCP のコネクションを開始する段階にならないと確定しないため，上述のように複数の経路の確保と，実際に経路上のスイッチに対するフローエントリの配備機能を分けて提供する．

また，提案システムに GridFTP を対応させるにあたり，本研究では GridFTP の実装自体への修正を最小限に抑えるため，GridFTP が通信ライブラリとして用いている Globus XIO [9] の通信ドライバの一つとして，提案システムへの対応に必要な機能を実装する．Globus XIO は Globus Toolkit 内で実装されている I/O ライブラリのプラグインフレームワークであり，通信方式やファイル，ストレージアクセス方式ごとにプラグインを実装することで様々なプロトコルやファイル形式に対応可能とする．本研究では，Globus XIO に標準で組み込まれている XIO TCP ドライバをベースに，OpenFlow コントローラとやり取りを実施して TCP コネクションを確立する通信ドライバを作成する．これにより GridFTP 本体には手を加えることなく，OpenFlow ネットワークに対応した GridFTP を構築可能となる．

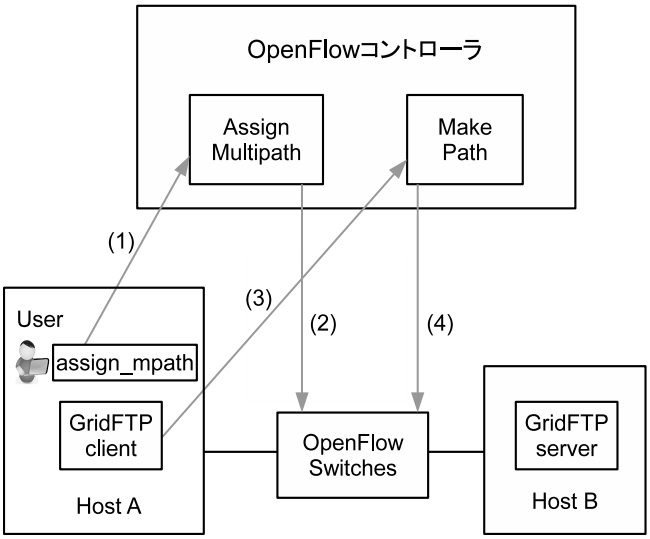

図 3 提案したシステムの概要図

4. 提案システムの実装

本節では，提案したシステムの実装および動作を説明する．図 3 に，今回の提案したシステムの概要図を示す．

4.1 システムの実装

本研究では，提案する OpenFlow コントローラを実装するにあたって，C と Ruby でコントローラを実装可能な Trema [10] を用いて開発した．具体的には Trema のサンプルコントローラの実装を集約した Trema Apps [11] に収録されている `routing_switch` [12] をベースに開発を行った．`routing_switch` は OpenFlow コントローラに接続されている OpenFlow スイッチとその接続関係のトポロジを常にモニタリングし，あるホスト間で通信を始めようとした際に，そのホスト間の最短ホップ経路をダイクストラ法によって算出し，経路設定を行うコントローラ実装である．本研究では，GridFTP がリクエストする複数経路の通信以外の通常の通信に関しては，ホスト間で最短ホップ経路にて通信させるため，この `routing_switch` をベースにした開発を行った．

本研究では，前節で説明した求められる 2 つの機能，1) ある拠点間において指定された本数分の経路を探索し，その経路を確保する機能，2) アプリケーションからの TCP コネクション開始の要求に応じて，経路上の OpenFlow スイッチにフローエントリルールを配備する機能をそれぞれ OpenFlow コントローラ上に XML-RPC インタフェースとして，実装した．汎用的な XML-RPC インタフェースとして実装することによって，特定の実装に依存することなく，任意のアプリケーションから提案コントローラを使用可能としている．具体的には前者の機能として，Assign Mulitipath というインタフェースを実装し，後者の機能のために MakePath というインタフェースを実装した．

したがって，我々のコントローラを使用するためには，Assign Mulitipath でまずは経路を確保し，MakePath によって実際のフローエントリの配備を行うという 2 段階の手順を踏む必要がある．1 段階目の経路の確保処理は GridFTP の通信を開始する前に実施する処理であるから，Assign Mulitipath インタフェースにアクセスするための軽量なクライアントプログラム，`assign_mpath` を GridFTP とは独立して作成した．2 段階目の MakePath にアクセスする処理は本研究で実装した XIO ドライバ内に実装し，GridFTP から呼び出されるように実装した．

具体的なプログラム実行手順は以下のとおりである．

```
> ./assign_mpath <controller_address> <port>
<src_ip> <src_mac> <dst_ip> <path_num>

> MPATH_ASSIGNMENT_ID=<id> globus-url-copy -p
<path_num> test.data gsiftp://HostB/test.data
```

`assign_mpath` は OpenFlow コントローラの IP アドレス (`controller_address`)，コントローラのポート番号 (`port`)，送信元の IP アドレス (`src_ip`)，送信元の MAC アドレス (`src_mac`)，宛先の IP アドレス (`dst_ip`)，確保したい経路数 (`path_num`) を指定して起動する． `path_num` で 0 で指定した場合，送信元と宛先間の考えられるすべての経路を算出する．指定された経路数以下の経路しか存在しない場合は，その存在する経路数分だけ確保する．`assign_mpath` は確保した経路数と割当 ID を結果として出力する．globus-url-copy は Globus Toolkit で提供される GridFTP client アプリケーションである．ここでは，割当 ID を環境変数として渡すために，`MPATH_ASSIGNMENT_ID` に割当 ID を指定して，起動する．

4.2 システムの動作説明

図 3 に示すとおり，提案したシステムは以下のように動作する．

- まず (1) では，ユーザから起動された `assign_mpath` は，OpenFlow コントローラ上に実装している Assign-Mulitipath インタフェースにアクセスし，HostA から HostB までの間で利用可能な複数の経路を要求する．
- 次に (2) では，AssignMulitipath がネットワーク全体のトポロジにおいて HostA から HostB までを結ぶ経路を，幅優先探索アルゴリズムに基づいて探索し，経路長が短いものから順にユーザから指定された数だけ経路を確保し，ユーザに確保ができた経路数とその割当 ID を返す．ここで返された割当 ID は次項以降で実際に経路を作成する際に，今回確保された経路のセットを参照するために用いられる．
- 続いて，ユーザは上記で取得された割当 ID を環境変

数を経由して渡す形で GridFTP client を起動する．

- (3) では，我々が構築した GridFTP 内の XIO ドライバが渡された割当 ID を読み取る．そして，GridFTP client が TCP ストリームを開始するごとに OpenFlow コントローラ上の MakePath インタフェースにアクセスし，開始しようとする TCP ストリーム用の経路の作成を OpenFlow コントローラに要求する．
- (4) では，MakePath は XIO ドライバから渡された割当 ID に基づき，(2) で確保された経路のセットの中から順番に経路を取り出し，その経路に従って TCP ストリームが確立されるように経路上の OpenFlow スイッチにフローエントリを書き込む．具体的には，開始しようとする TCP ストリームの宛先・送信元 IP アドレスおよび送信元 TCP ポート番号を条件に，フローエントリを生成し，各スイッチに経路を設定する．
- 最後に，TCP ストリームが上記で設定された経路に従って通信を開始する．以降，これに続く TCP ストリームは同様に我々の XIO ドライバによって (3), (4) の手順を踏んで確立され，各 TCP ストリームごとに異なる経路を使用して通信することになる．

5. 評価実験

提案手法の有効性を検証するために，OpenFlow によるマルチパスを用いる GridFTP と従来の GridFTP のパフォーマンスを比較するため，仮想環境上で複数の実験環境を構築し，評価実験を行った．本節では，その実験結果について述べる．

5.1 実験環境

実験環境の概要を図 4 に示す．HostA と HostB の 2 台のマシンに GridFTP をインストールし，この 2 台をサーバとクライアントとして使用した．本研究では，十分な数のハードウェア実装の OpenFlow スイッチを使用できなかったため，ソフトウェア実装ベースの Open vSwitch [13] をインストールした複数の Switch Host を HostA と HostB 間に配備することで，模擬的にマルチパスを有する OpenFlow ネットワークを構築した．また，GridFTP の通信が OpenFlow ネットワークを通過するように，HostA，HostB にも Open vSwitch をインストールし，tap による仮想ネットワークデバイスを追加した．HostA，HostB および各 Switch Host 上の Open vSwitch 間は GRE リンクによって接続した．この GRE リンクを付け替えることによって，様々なトポロジを模擬的に構築することができる．

今回の実験環境は具体的には 6 台の VMware ESX 上にそれぞれ 1 台ずつ配備した 6 台の仮想マシンによって構築した．各仮想マシンには CentOS 6.5 をインストールし，仮想 CPU として Intel Xeon E5649 を 2 プロセッサ分，メモリを 2GB 割り当てたものを使用した．各仮想マシンは

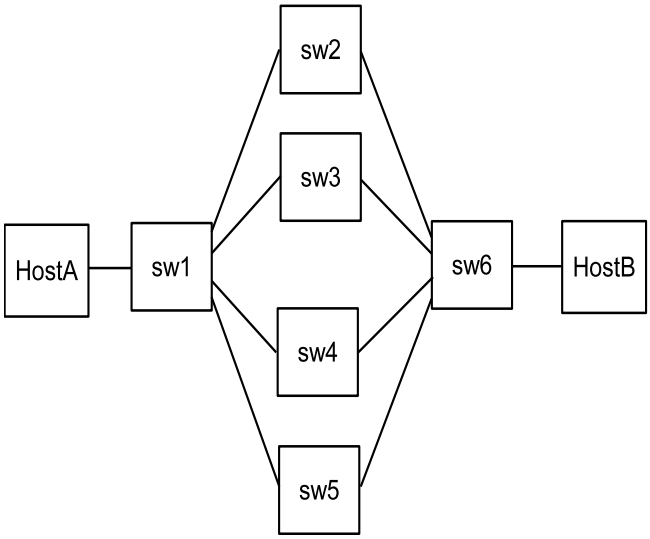

図 5 トポロジ A

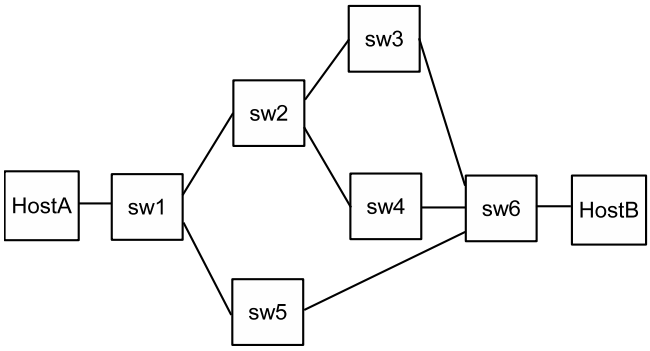

図 6 トポロジ B

物理的には 1Gbps のネットワークスイッチ 1 つを共有しており，iperf による実測では約 941Mbps の帯域が利用可能であった．本研究では，マルチパスの活用による効果を評価するため，この物理的な限界を超えないように，各 Open vSwitch 間を接続する GRE リンクは Open vSwitch の機能により 100Mbps に制限して用いた．

本研究では，以下の 2 つのネットワークトポロジにおいて評価実験を行った．図 5 に示すトポロジ A は各通信経路が完全に独立している単純なトポロジで，複数経路を使うと使った分だけ，使用可能なネットワーク帯域が向上するようなトポロジである．図 6 に示すトポロジ B はもう少し現実的なトポロジを模して，一部の経路においては重複が存在するようなトポロジである。今回はこの 2 つのトポロジを用いて，HostA，HostB 間において 1Gbyte のデータの転送実験を行い，その転送に要した時間，使用された帯域幅を測定した．

5.2 実験結果

転送時間は globus-url-copy コマンドの開始から終了までに要した時間として測定した．本研究の提案手法では，実際には globus-url-copy に先立って，`assign_mpath` を実

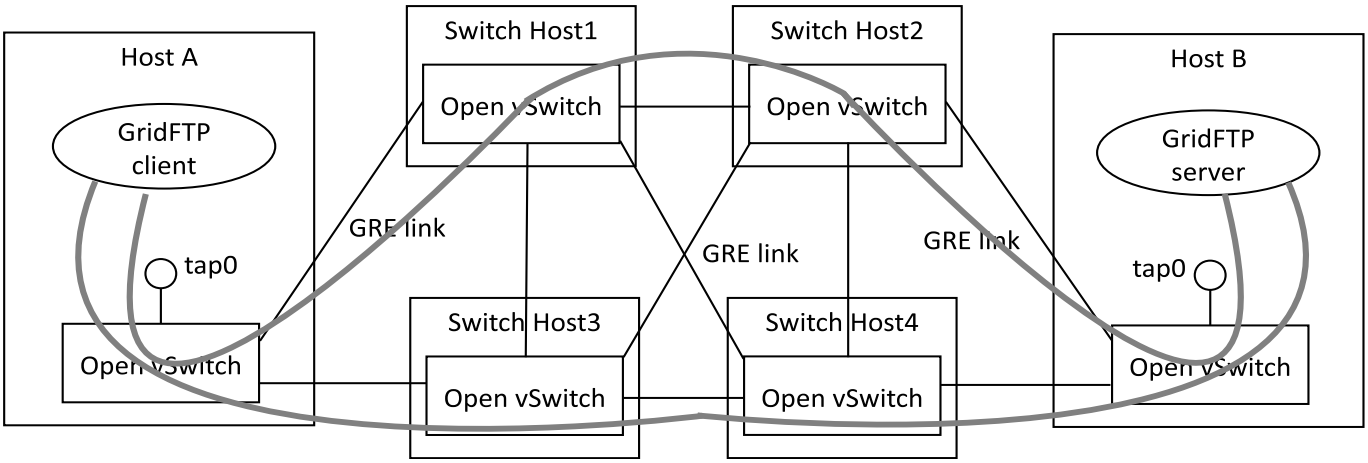

図 4 実験環境の概要図

行して経路確保をする必要があるが，今回の実験トポロジは非常に小さく，`assign_mpath` に要する時間も非常に短いため，測定対象にしていない．`assign_mpath` に要する時間の測定は，今後，より複雑なトポロジでの実験の際の課題とする．測定結果は表 1，表 2 に示す．

使用された帯域幅に関しては，GridFTP の各 TCP ストリームごとに個別に測定するため，今回の実験では OpenFlow スイッチに内蔵されているパケットカウント機能を用いて測定した．OpenFlow スイッチは各フローエントリごとにそのエントリにマッチした通過したパケット数と転送データバイト数を逐次記録しているため，その情報を一定時間間隔ごとに取得することで測定した．測定はデータ転送の宛先である HostB 上の Open vSwtich 上で実施した．ただし，今回の実験環境は仮想マシン上にインストールされた Open vSwitch を用いたものであり，頻繁にこの値を参照するとカウントの増加が時々不安定な動きをするため，値の参照は 2 秒ごとに 1 回実施することにし，図 7，図 8，図 9 にはその平均から算出した 1 秒あたりの転送バイト数を積み上げ面グラフとしてまとめることで使用帯域幅を示した．

1) トポロジ A

トポロジ A においては，並列転送数を 2 本と 4 本とする場合の 2 回の実験を行った．このトポロジで確保できる経路はそれぞれ重複する部分がなく，各スイッチを 1）sw1-sw2-sw6，2）sw1-sw3-sw6，3）sw1-sw4-sw6 と 4）sw1-sw5-sw6 という順でそれぞれ経由する 4 つの経路がある．sw1-sw2-sw6 は sw1, sw2, sw6 をそれぞれこの順にたどる経路を意味する．

表 1 トポロジ A の実験結果

並列転送数	提案手法	従来手法
2	47.486s	1m32.108s
4	23.404s	1m30.533s

実験結果は表 1 に示すように，本提案により GridFTP の複数 TCP ストリームを個別の経路に分散させることで，2 本並列転送の場合では，従来手法に比べて所要時間が約 1/2 になっている．また，4 本並列転送の場合では，従来

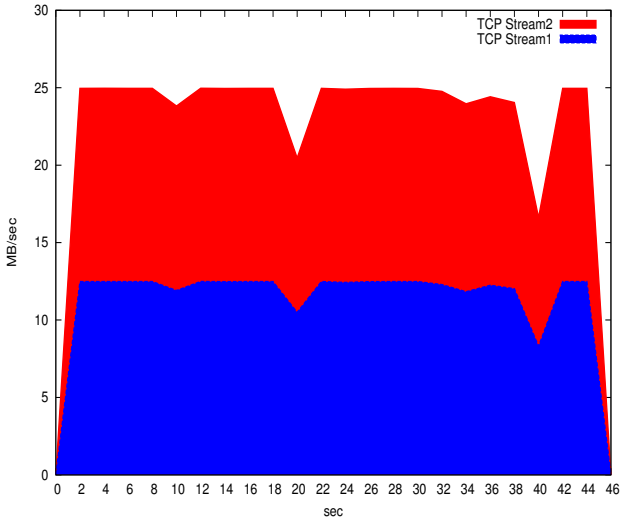

図 7 トポロジ A での 2 本並列転送実験

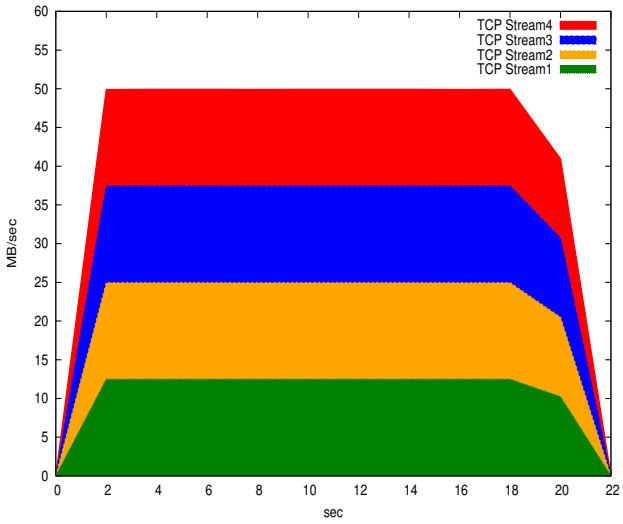

図 8 トポロジ A での 4 本並列転送実験

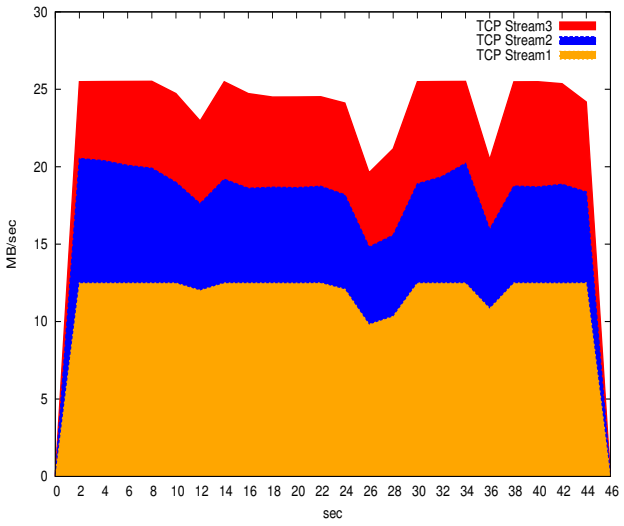

図 9 トポロジ B での 3 本並列転送実験

手法に比べて所要時間が約 1/4 になっている．図 7，図 8 は各 TCP ストリームそれぞれで使用された帯域を示しているが，それぞれ非常に安定したパフォーマンスを示していることが分かった．

2) トポロジ B

トポロジ B において確保できる経路は，一部重複している部分があり，全部で 1）sw1-sw5-sw6，2）sw1-sw2-sw3-sw6，3）sw1-sw2-sw4-sw6 の 3 つがあった．経路 2) と 3) において，sw1-sw2 間の経路が共有されている形になっている．トポロジ B における実験は，この 3 つの経路を利用する並列転送数が 3 本の実験を行った．

表 2 トポロジ B の実験結果

並列転送数	提案手法	従来手法
3	46.603s	1m30.480s

実験結果は表 2 に示すように，本提案を提供することで，従来手法に比べて所要時間が約 1/2 になっている．並列転送数は 3 本であるが，そのうち 2 本は 1 つの経路を共有しているため，全体としては 2 倍のパフォーマンスしか出ていないことが分かる．図 9 は各 TCP ストリームの使用帯域をまとめたものであるが，TCP Stream1，TCP Stream2 と TCP Stream3 がそれぞれ， 上述の 1），2）と 3）の 3 つの経路の使用に対応している．TCP Stream1 の通信経路は他の 2 つとは独立しており，通信量が 12MB/sec ぐらいで非常に安定していることが分かる．TCP Stream2 と TCP Stream3 の通信経路は sw1-sw2 の部分で重複しており，それぞれの通信量が TCP Stream1 通信量の約半分になっている．また，安定性も TCP Stream1 に比べて悪いことが分かる．

6. おわりに

本稿では，GridFTP によるデータ転送を高速化する手法として，GridFTP の並列転送に OpenFlow を用いた SDN 技術による制御を行うことで，複数経路を使用したデータ転送を可能とする手法を提案した．提案手法の有効性を示すために，仮想環境上で複数の実験環境を構築して実験を行った．実験結果により，提案手法はネットワークの使用帯域を向上させ，GridFTP の並列データ転送時間が短縮できることを示した．

本研究はエンド・ツー・エンドで OpenFlow による複数経路確保を実施するシステムを提案したが，今回の実験では均質なリンクからなる実験環境を専有していることを前提に評価を行った．しかし，実運用上のネットワークでは，各リンクの質は一定しておらず，また他のアプリケーションの通信の影響を受けることになる．したがって，そのような状況の中で，最適なマルチパスの組み合わせを探索し確保することが重要である．今後の課題としては，実際のネットワークに存在する遅延やパケットロスなどの状況を考慮した状況下での最適な経路探索を実現するアルゴリズムを実装する予定である．

参考文献

[1] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S.: The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, *Journal of Network and Computer Applications*, Vol. 23 (2000).

[2] Foster, I., Kesselman, C. and Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations, *International journal of high performance computing applications*, Vol. 15, No. 3, pp. 200–222 (2001).

[3] Foster, I.: Globus Online: Accelerating and Democratizing Science through Cloud-Based Services, *IEEE Internet Computing*, Vol. 15, No. 3, pp. 70–73 (2011).

[4] Open Networking Foundation: https://www.opennetworking.org/ja/.

[5] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow: enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 2, pp. 69–74 (2008).

[6] Kissel, E., Swany, M. and Brown, A.: Improving GridFTP performance using the Phoebus session layer, *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ACM, p. 34 (2009).

[7] Gunter, D., Kettimuthu, R., Kissel, E., Swany, M., Yi, J. and Zurawski, J.: Exploiting Network Parallelism for Improving Data Transfer Performance, *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, IEEE, pp. 1600–1606 (2012).

[8] Kanaumi, Y., Saito, S.-i., Kawai, E., Ishii, S., Kobayashi, K. and Shimojo, S.: RISE: A Wide-Area Hybrid OpenFlow Network Testbed, *Ieice Transactions on Communications*, Vol. 96, No. 1, pp. 108–118 (2013).

[9] Allcock, W., Bresnahan, J., Kettimuthu, R. and Link, J.: The globus extensible input/output system (xio): A protocol independent io system for the grid, *Joint Workshop on HighPerformance Grid Computing and High-Level Paral-lel Programming Models in conjunction with In-*

ternational Parallel and Distributed Processing Symposium., IEEE (2005).
[10] Shimonishi, H., Takamiya, Y., Chiba, Y., Sugyo, K., Hatano, Y., Sonoda, K., Suzuki, K., Kotani, D. and Akiyoshi, I.: Programmable network using OpenFlow for network researches and experiments, *Proc. 6th International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2012)*, pp. 164–171 (2012).
[11] Trema App: https://github.com/trema/apps.
[12] `routing_switch`: https://github.com/trema/apps/tree/develop/`routing_switch`.
[13] Open vSwitch: http://openvsw-itch.org/.