トンネリング技術を用いた汎用機器による データセンターマルチパスの実現

中村 遼　石原 知洋　関谷 勇司　江崎 浩

データセンター内における通信量は日々増加しており，ネットワークの可用性とスループットの向上が大きな課題となっている．そこで，データセンターネットワークにおけるマルチパスを用いた効率的なデータ配送手法について多くの研究が行われてきた．しかし，既存研究の多くは，ネットワークスイッチのハードウェアに対して変更を加える必要があり実環境への適用性に欠けるという問題がある．そこで本研究では，IP トンネル技術を用いた特定パスへのトラフィック配送手法を拡張することで，コストや障害時のスループットにおいて優位であるランダムグラフトポロジのデータセンターネットワークを汎用機器のみを用いて構築する手法を提案する．本提案手法は，ネットワークスイッチには変更を加えず，サーバのネットワークスタックへの小さな変更のみで実現される．また，現在一般的に利用されている OSPF を用いた自動設定と障害復旧を行うシステムの設計と実装を行った．そして，シミュレータを用いた評価と実機を用いた実験を行い，提案手法がネットワーク全体の通信量の向上と高い可用性を汎用機器のみで実現可能であることを示した．

Improving availability and throughput is a significant challenge for data center networks. Recent studies have attempted to use a variety of routing and multipathing techniques. However, no method has yet managed to combine availability and throughput improvement with actual deployability, usually because of dedicated hardware requirements. In addition, recent approaches lack availability for topology changes because of requiring pre-configured topology. In this study, we propose a method to construct efficient commodity-based random graph topology network by using an IP tunneling technique. The proposed method only requires minimal modification for software network stack on end hosts. In addition, we designed and implemented a control plane system for proposed method using OSPF. Moreover, the evaluation shows that our proposed method enables throughput improvement of a network and high availability with commodity hardware switches.

1　はじめに

クラウドコンピューティングに代表されるような，データをネットワーク越しに保存し利用するアプリケーションの増加によって，データセンターにおける通信量は日々増加を続けている．Cisco Systems の報告によると，世界のデータセンターにおける IP トラフィックは 2019 年に 2014 年の 3 倍に達すると予測されている [5]．そこで現在，一般的なデータセンター内部のネットワークにおいて可用性とスループットの向上が大きな課題となっている．可用性とは，ある一定の期間内において，システムが要求通り動作できる状態にあると定義される．データセンターネットワークにおいては，障害の発生のしにくさや，障害発生時の修復速度が可用性に影響をおよぼす．そのためネットワークを構成する機器やリンクに障害が発生した際には，迅速に通信経路を復旧することが求められる．またコストの観点から，データセンターの仕様か

Multipathing in Commodity-based Data Center Networks with IP Tunneling.

Ryo Nakamura, Hiroshi Esaki, 東京大学情報理工学系研究科, Graduate School of Information Science and Technology, The University of Tokyo.

Tomohiro Ishihara, 東京大学総合文化研究科, Graduate School of Arts and Sciences, The University of Tokyo.

Yuji Sekiya, 東京大学情報基盤センター, Information Technology Center, The University of Tokyo.

コンピュータソフトウェア, Vol.33, No.3 (2016), pp.29–43.

[研究論文] 2015 年 8 月 14 日受付.

らある通信容量を要求された際に，トポロジや経路制御の工夫によって可能な限り少ない機器数およびリンク数により同要求を満たすことが求められる．

こうしたデータセンターネットワークそのものに対する課題に対して，様々な手法が提案されてきた．現在，実際に運用されているデータセンターネットワークでは，Border Gateway Protocol(BGP) を用いたレイヤー 3 の Clos トポロジを構築することで高い可用性を実現する方法 [3] [14] が用いられている．一方上記のような Clos トポトジ以外にも，Fat-tree [2] や B-Cube [11]，HyperX [1] といったトポロジの利用が提案されている．そして，これらのトポロジ上で効率的にトラフィックを配送するための様々な手法が提案されている [4]．また Singla らは，様々なトポロジの中でも，データセンターネットワークにランダムグラフトポロジを適用することを提案している [21]．Singla らは論文 [21] の中で，ランダムグラフトポロジが Fat-tree に対してコストや障害時のスループットの面で優位であるとしている．またランダムグラフトポロジでネットワーク全体のスループットを向上するため，k-shortest path [23] と Multipath TCP [7] を利用している．

このようにデータセンターネットワークの可用性とスループットを向上するために様々な手法が提案されてきたが，実際のデータセンターネットワークに適用されるには至っていない．その原因として，構築コストが挙げられる [9]．ネットワークを構築するためのコストを抑えるため，多くの場合，Commercial-Off-The-Shelf (COTS) スイッチと呼ばれる容易に入手可能な汎用ネットワーク機器の利用が前提とされる．しかし既存の手法は，ネットワークを構成するスイッチのハードウェアに対して特殊な変更を必要とする．特殊なハードウェアの利用はコストの増大を招くため，これらの技術を実環境に導入するための大きな障害となっている．一方 COTS スイッチを用いる手法では高い可用性を実現できてはいない．つまり，COTS スイッチのみを用いて，可用性とスループットの向上を同時に実現するには至っていない．

そこで本研究では，Singla らの提案したランダムグラフトポロジのデータセンターネットワークを COTS スイッチのみを用いて実現する手法を提案する．筆者らはこれまでに行った研究において，COTS スイッチのみを用いて構築した Fat-tree トポロジのレイヤー 3 データセンターネットワークで，IP トンネル技術を用いてトラフィックを効率的に分散する手法を提案した [18]．本研究では，この手法を拡張し，ランダムグラフトポロジにおける k-shortest path の経路制御と各パスへのトラフィックの分散を，COTS スイッチのみを用いたネットワーク上で実現する．その上で，OSPF を利用してパスの自動設定と復旧を行う機構を設計し，実装した．以降，本論文ではまず 2 章で関連研究を述べ，3 章で IP トンネル技術を用いたパス選択手法について述べる．そして 4 章でランダムグラフトポロジとその実現手法について述べ，5 章で評価実験を行う．最後に，6 章で結論と今後の課題について述べる．

2 既存研究

データセンターネットワークの可用性とスループットの向上を目的とした多くの既存研究がある．しかしこれらは，COTS スイッチを利用しながら，その両方を同時に実現できてはいない．既存研究を，想定するトポロジ，利用できるパス，COTS スイッチを使用できるか，そして可用性という 4 つの項目について比較した結果を表 1 に示す．可用性は，それらの手法が，障害などのトポロジの変化に対して迅速に経路を修復することができるか，である．この表 1 をもとに，可用性とスループットの向上，そして COTS スイッチの利用による実環境への適応性という観点から既存研究の問題点を整理する．

データセンターネットワーク用に提案されるトポロジの特徴は，あるスイッチ間に複数のパスが存在し，それらのパスにトラフィックを分散することによってネットワーク全体の転送量を増やすことである．このとき，トラフィックを複数のパスに分散する最も一般的な手法として，レイヤー 3 ネットワークにおける Equal Cost Multipath (ECMP) がある．ECMP は，ある宛先への最短経路上に同一コストの複数のリンクがあるとき，パケットのハッシュ値に基づいてトラフィックをそれらのリンクに分散する機能であ

表 1 SPAIN [17] にもとづく既存研究の比較

	トポロジ	利用するパス	COTS スイッチを利用	可用性
Facebook [3]	Multiple Tree	ECMP	YES	YES
TRILL [6]	Arbitrary	ECMP	NO	YES
SEATTLE [13]	Arbitrary	Single Path	NO	YES
PortLand [19]	Fat-tree	ECMP	NO	NO
SPAIN [17]	Arbitrary	Multiple Paths	YES	NO
VL2 [10]	Fat-tree	ECMP	YES	YES
Jellyfish [21]	Random Graph	k-shortest path	NO	YES

る．ECMP は既に多くのネットワーク機器において実装されており，VL2 [10] や Facebook [3] など，汎用機器を用いることを主眼に置いた手法で利用されている．またデータセンターネットワークをレイヤー 3 で構築することで，高い可用性を得ることができる [10] [3] [14]．しかし ECMP には 2 つの問題がある．ひとつは，パケットのハッシュ値に基づいてトラフィックを分散するため特定のパスにトラフィックが偏ってしまう点．もうひとつは，最短経路で同一コストのリンクしか利用できないためランダムグラフのようなトポロジでは利用できるリンクが限られてしまう点である [21]．

ECMP に対して，SPAIN [17] はマルチパスを利用するために VLAN を用いる．VLAN は大抵の COTS スイッチで利用可能である．その上で，SPAIN は複数のパスを指定できるように VLAN をセットアップするためのアルゴリズムを提案している．これによって，SPAIN は COTS スイッチの利用による実環境への適用性と，ECMP 以外のマルチパス利用手法による高いスループットを同時に実現している．しかし SPAIN では，トポロジの変更が起こった際に，まずネットワーク全体のトポロジを再検出し，各パスを指定するための VLAN のセットアップを再計算し，全スイッチに再度設定を投入する必要がある．そのため，SPAIN はレイヤー 3 の動的経路制御 [3] [10] を行う手法に比べて可用性の面で不足がある．

また，可用性を実現するレイヤー 3 のホップバイホップルーティングと動的経路制御のアーキテクチャをレイヤー 2 ネットワークに適用する手法として，TRILL [6] や SEATTLE [13] がある．これらの手法は，レイヤー 2 のデータリンク層に新たな識別子を導入し，ルーティングに利用する．しかし，こういった機能は既存の COTS スイッチには無いため，スイッチのハードウェアへの特別な変更を必要とする．また一方で，PortLand [19] はレイヤー 2 ネットワークに特定のアドレッシングルールを適用し，そのルールに基づいて ECMP とは異なるマルチパスの利用を行う．PortLand は，Al-Fares らによって提案されたレイヤー 3 ネットワークのためアドレッシングルール [2] をレイヤー 2 へと拡張したものである．この PortLand も SEATTLE などと同様スイッチのハードウェアに変更を必要とするため，COTS スイッチを用いて構築することができない．

最後に，Singla らの提案する Jellyfish [21] では，データセンターネットワークとしてランダムグラフトポロジを用いる．ランダムグラフトポロジは Fat-tree トポロジに対して構築にかかるコストや障害時のスループットの面で優位であるとしている．しかし，ランダムグラフトポロジにおいては，ある送信元から宛先まで同一コストで到達する経路が少ないため，ECMP ではトポロジ上の多くのリンクを利用することができない．そこで Jellyfish では，宛先ごとに k 番目まで短いパス，k-shortest path を利用する．この k 本のパスに MPTCP を用いて各サブフローを分散することで複数パスを利用する．しかし，k-shortestpath を実現する経路制御機構は COTS スイッチにはなく，ハードウェアに変更が必要なため実環境への適用性に欠ける．

3 IP トンネル技術を用いたパス決定手法

本研究では，Singla の提案するランダムグラフトポロジのデータセンターネットワークを，COTS スイッチのみを用いてレイヤー 3 で構築する手法を提案する．ランダムグラフトポロジにおいて k-shortestpath と MPTCP を利用することでスループットの向上を，COTS スイッチの利用によって実環境への適用性を，そしてレイヤー 3 で動的経路制御を使ってネットワークを構築することで可用性を実現する．このために，本章ではまず筆者らが先行研究 [18] で提案した，IP トンネル技術を用いてトラフィックを選択的に決定したパスへ転送する手法について述べる．そして，4 章にてランダムグラフへの拡張について述べる．また本論文では以後レイヤー 3 スイッチとしてスイッチという言葉を用いる．

3.1 概要

COTS スイッチは ECMP 以外にトラフィックを複数パスへ分散する機能を持たない．そこで，本手法は，トラフィックを複数のパスに分散する機能を，ネットワークを構成するスイッチから，ソフトウェアによる機能追加が容易なサーバへと移す．つまり，サーバのネットワークスタックに変更を加え，サーバが複数のパスへとトラフィックを分散することで，ネットワーク全体のスループットを向上させる．

図 1 に，本手法の概要を示す．ある送信元サーバからある宛先サーバへのパスは，そのパスに対応するスイッチをパケットの中継点として明示することで指定できる．図 1 中，サーバ S からサーバ D への最短経路がスイッチ B を経由する場合，サーバ S はスイッチ C を中継するようパケットに指定できれば，最短経路以外のパスへとパケットを通すことができる．また，パケットごとに中継点としてスイッチ B とスイッチ C を指定すれば，両方のパスへとトラフィックを分散できる．このように，中継点を明示するための何らかの識別子をパケットに埋め込むことができれば，サーバがパケットを通すべきパスを指定し，トラフィックを複数のパスに分散できる．

パケットに特定のスイッチを経由させるための手法

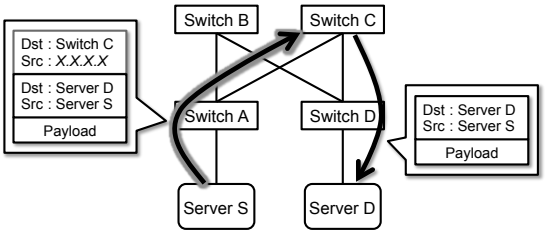


図 1 IP トンネル技術を用いたパス決定手法の概要

として，ソースルートオプションがある．IPv4 では Loose Source and Record Route Option，IPv6 では Routing Header として標準化されており，IP スタックの一部であるため COTS スイッチも対応している．しかしながら，ソースルートオプションはセキュリティ上の問題によって利用できない，また多くのスイッチにおいて CPU で処理されるためスループットが出ないなど，多くの問題がある．そのため，実環境において，ソースルートオプションを中継点の指定に用いることは現実的ではない．

ソースルートオプションの代わりに，本手法ではパケットに中継点を明示するための識別子として，スイッチのハードウェアでサポートされた IP トンネル技術を用いる．本手法を適用したサーバは，パケットを送信する際，IP でカプセル化を行う．そのとき，内部 IP ヘッダの宛先は最終的な通信先サーバのアドレス，送信元は送信するサーバ自身のアドレスとする．そして，カプセル化した外部 IP ヘッダの宛先は中継するスイッチのアドレス，送信元には全サーバで同一の IP アドレスを用いる．図 1 は，サーバ S がサーバ D に対してパケットを送信する際，外部 IP ヘッダの宛先としてスイッチ C のアドレスを用いることで，最短経路ではないパスへとパケットを通す様子を示している．スイッチ C がカプセル化されたパケットを受信すると，外部 IP ヘッダとカプセル化ヘッダを取り外してサーバ D へと送信する．また複数のスイッチを中継させたい場合には，中継する順番に従ってパケットを複数回カプセル化することで任意のパスへとパケットを転送する．

本手法はこのように IP トンネル技術を利用することで，COTS スイッチで構築されたレイヤー 3 ネッ

トワークにおける明示的なパスへのパケット転送を実現する．IP トンネル技術は既に多くの COTS スイッチでサポートされており，ワイヤーレートでの転送も可能である．例えば VL2 は，IP-in-IP トンネリングを用い，汎用のスイッチチップである Broadcom の 56820 及び 56514 を使った実機の実験環境を作成している．これらのチップは，ワイヤーレートのレイヤー 2 及びレイヤー 3 パケット転送，そしてカプセル化とその解除の処理をサポートしている [10]．また Microsoft はこの VL2 を Azure のストレージネットワークにデプロイしている [15]．このように，IP トンネル技術を用いる本手法は，汎用機器を用いる実際のデータセンター環境で即座に利用可能である．

3.2 IP トンネルの規模性

端点から端点へとパケットをカプセル化して運ぶトンネリング技術を，そのまま中継するスイッチの指定のために用いた場合，トンネルの数とその規模性が問題になる．あるデータセンターネットワークにおいて，全てのサーバが全てのパスを指定できるようにするためには，全てのサーバと全てのスイッチの間で IP トンネルを構築する必要がある．しかし，一般的な COTS スイッチで作成できるトンネル数は数十から数百程度であり，数千台規模のサーバやスイッチを収容するデータセンターにおいては現実的でない．

そこで本手法ではこのトンネル数の規模性の問題を回避するために，全サーバで同一のアドレスをトンネルの送信元アドレスとして用いる．図 1 中の *X.X.X.X* がこのアドレスを示す．本手法において，パケットは常にサーバによってカプセル化され，スイッチによってカプセル化を解除される．つまり，スイッチがパケットをカプセル化してサーバへ送信することは無く，トンネル内のパケットの通信方向は常に一方向のみである．よって，全てのサーバはカプセル化した際の外側 IP ヘッダの送信元アドレスとして，同一の IP アドレスを用いることができる．同一の IP アドレスを外部ヘッダの送信元に用いることで，各スイッチに必要なトンネルは，スイッチ自身からその共通アドレス (*X.X.X.X*)，というただ 1 つとなる．また，*X.X.X.X* という存在しないアドレスを利用するため，ステートを管理するトンネリングプロトコルは本手法に用いることができない．しかし，ステートを管理しないプロトコルとして，GRE や IPIP，VXLAN など，多くのプロトコルが利用可能である．このようにして，本手法ではトンネル数の規模性を確保する．

3.3 サーバネットーワクスタックへの変更

本手法は，トラフィックを複数のパスに分散する機能を，スイッチからサーバのネットワークスタックへと移す．本節では，本手法を実現するために必要なサーバへの変更について述べる．

サーバのネットワークスタックへの変更は図 2 に示す 1 つのカーネルモジュール (`iplb`) によって実現される．`iplb` はカーネルスペースにロンゲストプレフィックスマッチに基づく経路表を保持する．この経路表には，宛先ネットワークごとに，その宛先ネットワークへと到達する複数のパスと，各パスを通すために中継しなければならないスイッチのリストを保持する．ユーザランドのアプリケーションがパケットを送信すると，`iplb` はまずパケットの宛先アドレスが先ほどの経路表にマッチするかを検証する．マッチした場合，そのパケットを通すべきパスを決定し，パスに通すために中継するスイッチを宛先とした IP ヘッダを付与していく．このとき，1 つのパスに通すために複数のスイッチを経由する場合，各中継点を宛先として複数回のカプセル化を行う．

ある宛先ネットワーク宛のパケットを通すべきパスは，フローごとに決定する．フローは 5 タプル (宛

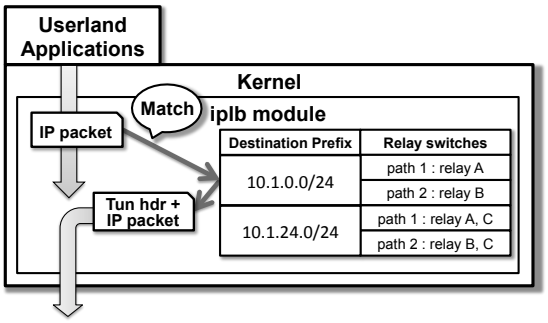


図 2 サーバへの変更の概要

先アドレス，送信元アドレス，プロトコル番号，宛先ポート番号，送信元ポート番号) によって識別される．これは，同一フローを構成するパケットが別のパスを通った場合，パケットの到着順序に入れ替わりが発生し，上位のトランスポート層において性能の低下が発生するのを防ぐためである．

4 ランダムグラフによるデータセンターネットワーク

本研究では，3 章で述べたトラフィックを特定パスに転送する手法を拡張し，ランダムグラフトポロジに適用する．本章ではそのための MPTCP におけるサブフローの各パスへの割り当て手法，中継点の選択手法，そして OSPF を利用して自動で `iplb` の設定を行う機構について述べる．

4.1 ランダムグラフトポロジの利用法

Jellyfish において，ランダムグラフは以下のように定義される．ネットワークを構成するスイッチ i は k_i 本のポートを持つ．そして r_i 本ポートを他のスイッチとの接続に用い，残った $k_i - r_i$ 本のポートを用いてサーバを収容する．スイッチ i が r_i 本のポートを用いて接続するスイッチは，一様な確率で他のスイッチから選択される．このようにして定義されるランダムグラフを，データセンターネットワークのトポロジとして用いる．本論文では Jellyfish と同様に，全てのスイッチが同じ k 本のポートを持ち，r 本のポートをスイッチ間の接続に用いるものとする．

k-shortest path とは，あるグラフにおいて，特定の始点から特定の宛先へ到達するパスのうち，k 番目までの短いパスのことである．これを見つけるために，Yen によって提案された k-shortest loopless path アルゴリズム [23] をはじめ，計算量の異なるいくつかのアルゴリズムがある．Yen のアルゴリズムによって作られる k-shortest path は，始点から，宛先への最短経路を外れた点 (分岐点) までの最短経路と，その分岐点から宛先までの最短経路をつなげたパスである [23]．この分岐点を変更していくことで，k 本のパスを作成する．2 章で述べたように，ECMP で利用できるパスは同一コストの最短経路に限られる．そこで各サーバ間の通信において k-shortest path アルゴリズムで発見した k 本のパスにトラフィックを分散することで，ECMP では使えなかったリンクも利用することができる．

k-shortest path アルゴリズムでネットワーク上の複数パスを見つけた上で，Jellyfish ではそれらのパスにトラフィックを分散するために Multipath TCP (MPTCP) [7] を用いる．MPTCP は，複数の TCP セッションを束ねて 1 つの TCP セッションとして扱うためのトランスポート層のプロトコルである．MPTCP は，1 つの MPTCP セッションを構成する複数の TCP セッションが異なる IP インターフェース経由で異なる経路を通って確立されることを想定するため，複数の TCP セッションをまとめて輻輳制御する機構 (Linked Increase アルゴリズム) を持つ．Jellyfish では，この MPTCP を用いて 1 つの TCP セッションを複数のサブフローとし，各サブフローを k-shortest path アルゴリズムで見つけた複数のパスに分散する．各パスへのトラフィックの分散は MPTCP の輻輳制御機構によって管理され，ECMP で発生する特定パスへのトラフィックの偏りを回避し，ネットワーク全体のスループットを向上する．

4.2 iplb の適用手法

4.1 節で述べたように，k-shortest path と MPTCP を用いることでランダムグラフにおいて ECMP 以上のスループットを実現できる．しかし，Jellyfish の論文中でも述べられているように，トラフィックを選択的に k-shortest path に分散して配送することは既存の COTS スイッチではできない．そこで本研究では，サーバが MPTCP を構成するサブフローを k-shortest path の各パスに通すために 3 章で述べた `iplb` を用いることで，COTS スイッチによるランダムグラフトポロジのネットワークを実現する．

提案手法を k-shortest path と MPTCP に組み合わせて用いる際の概要を図 3 に示す．ユーザランドのアプリケーションがある宛先にデータを送信すると，MPTCP がそのデータを複数の TCP セッションに分散して送信する．`iplb` は，各 TCP サブフローを事前に設定された k 本の各パスへとカプセル化を

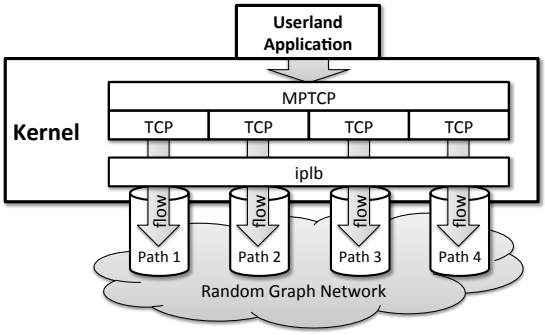


図 3 提案手法による MPTCP サブフローの各パスへの割り当て

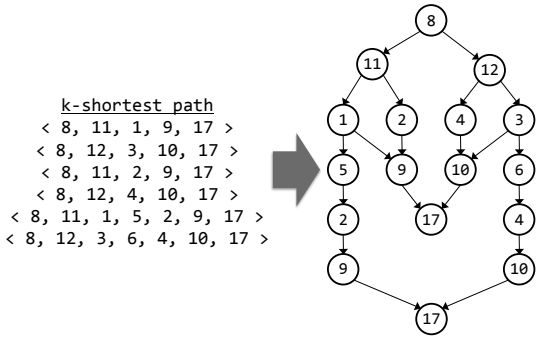


図 4 k 本のパスから各パスの中継点の探索 ($k = 6$)

用いて転送する．このとき，ある宛先ネットワーク宛てのパケットを通すべきパスは，フローごとに，ラウンドロビンで決定する．異なるパスを通る複数フロー間の輻輳制御や，パケットの到着順が入れ変わった際の再構成などは MPTCP によって行われる．また，`iplb` はサーバの同一インターフェースから送信される複数のフローも異なるパスに通すことができるため，MPTCP を利用するためにひとつのサーバに複数のインターフェースを用意する必要は無い．このようにサーバの送信するパケットを k 本のパスへ分散して送信することで，Jellyfish で提案されたランダムグラフトポロジにおけるスループットの向上を，COTS スイッチのみを用いて実現する．

4.3 中継点の選択

`iplb` におけるパスは，中継するスイッチのリストで構成される．そのため，k-shortest path の特定パスへパケットを通すためには，そのパスにおいて中継すべきスイッチを決定する必要がある．本節では，この中継スイッチを決定する手法について述べる．

ある送信元からある宛先への k-shortest path の各パスは，経路が途中で分岐することによって形成される．そこで，各パスを始点から比較していき，分岐したすぐ先のスイッチを中継点として記録していくことで，パスごとの中継点のリストを作成することができる．まず，k-shortest path で発見した k 本のパスから木構造をつくる．例として，$k = 6$ のとき，スイッチ 8 からスイッチ 17 へと向かうパスから木構造をつくる様子を図 4 に示す．各パスを木構造に追加する際，同一のスイッチであっても異なるホップ数に出てきたスイッチは別のノードとして扱う．これは，木構造の中でループが発生するのを防ぐためである．

Algorithm 1 中継点リストの作成

```
function DFsearch(V, relaylist, parent)
    if V.leaflist = 0 then
        V is bottom node. dump relaylist.
        return.
    end if
    if parent.leaflist > 1 then
        push V to relaylist.
    end if
    for v in V.leaflist do
        DFsearch(v, relaylist, V)
    end for
end function
```

各パスを指定するには，この木構造において宛先に到るまでに分岐した箇所を中継点として指定していけばよい．そこで，Algorithm 1 に示す深さ優先探索を行う．始点から，自ノードの親のノードが 2 個以上の子ノードを持つ場合，つまり自ノードが分岐した先のノードである場合は，自身を中継点のリストに追加する，という処理を行いながら探索する．例えば図 4 において，ノード 8，11，1 と辿った際，ノード 11 から見て親となるノード 8 は 2 つの子ノードを持つ．そこでノード 11 を中継点リストに追加する．次にノード 1 もノード 11 が複数の子ノードを持つため自身を

中継点リストに追加する．こうして宛先であるノード 17 まで辿り，$\langle 8, 11, 1, 5, 2, 9, 17\rangle$ のパスを通すための中継点は $\langle 11, 1, 5\rangle$ であると分かる．同様にして，$\langle 8, 11, 1, 9, 17\rangle$ のパスは $\langle 11, 1, 9\rangle$，$\langle 8, 11, 2, 9, 17\rangle$ のパスは $\langle 11, 2\rangle$ となる．

このようにして，COTS スイッチで構築したデータセンターネットワークにおいてサーバが明示的にパスを指定するための手法をランダムグラフトポロジに適用する．サーバが送信するトラフィックは MPTCP によって複数の TCP サブフローに分割され，各フローが提案手法を用いて k-shortest path の各パスへ分散して送信される．また各パスを指定するための中継スイッチは本節で述べた手法によって決定される．

4.4 OSPF を用いた制御システム

実際のデータセンターネットワークでは，スイッチの追加や障害などによって頻繁にトポロジ変更が発生する．高可用性を実現するためには，こういったトポロジの変更時に迅速にパスを復旧しなければならない．そこで，自動でトポロジとその変更を検知し，各サーバが各宛先へのパスを再計算して `iplb` の設定を更新する仕組みとして，`iplbd` を設計し，実装した．

トポロジとその変更の検知には，Open Shortest Path First (OSPF) [16] を用いる．OSPF は，現在最も広く利用されているルーティングプロトコルの 1 つであり，多くの COTS スイッチにも実装されている．OSPF に参加するノードは，Link State Advertisement (LSA) というリンクの状態を記したパケットをネットワーク内に伝搬する．この LSA をもとに，全参加ノードが Link State Data Base (LSDB) というそのネットワークの完全なトポロジ情報を共有する．また，リンクが切断した場合には新たに LSA を伝搬し，トポロジの変更に追従して LSDB を更新する．OSPF では各ノードが個別に宛先までの経路を計算するため，各ノードが持つ LSDB の内容，つまりトポロジの情報は同一になることが保証されている．提案手法では，データセンターネットワークのルーティングプロトコルとして OSPF を用い，その上で `iplbd` がこの OSPF に参加することによってトポロジ情報やその変更を検知し，パスの設定を更新する．

`iplbd` は，ユーザランドソフトウェアとして `iplb` を用いる各サーバ上で動作する．この `iplbd` は 2 つのコンポーネントから構成される．1 つ目は OSPF プロトコルを処理する部分である．これはサーバ自身を収容するスイッチと OSPF の隣接関係を結び，LSA を受け取ってトポロジ情報である LSDB を常に最新の状態に保つ．2 つ目は，カーネル内の `iplb` のパス情報を更新する部分である．`iplbd` の起動時に LSDB の構築が完了すると，このコンポーネントが LSDB の情報をもとに 4.3 節で述べたアルゴリズムを用いて各サーバへのパスと中継点を決定し，`iplb` に設定する．また新たな LSA を受信した場合，LSDB の更新が完了すると，起動時と同様に再度パスの計算を行い，パス情報を更新する．このように各サーバ上で動作する `iplbd` が OSPF を用いてトポロジ情報の取得と障害時のパスの更新を行うことで，トポロジ変更に対して迅速に対応し，可用性を実現する．

5 評価

本研究では，IP トンネル技術を用いた特定パスへのパケット転送手法を応用することで，Jellyfish で提案されたランダムグラフトポロジのデータセンターネットワークを COTS スイッチのみを用いて実現する．また，OSPF を利用した制御機構によって，レイヤー 3 ネットワークと同様の高い可用性を実現する．本論文ではこれらの評価として，1) 提案手法によるスループットの向上度合い，2) カプセル化によるオーバーヘッド，3) 障害時復旧に要する時間，の 3 点について評価実験を行った．

5.1 スループットの向上

本節では，ランダムグラフネットワークに `iplb` を適用することで，Jellyfish で示された通りにスルー

表 2 実験トポロジサイズ

スイッチのポート数	スイッチ数	サーバ数
4 ポート	20	16
6 ポート	45	54
8 ポート	80	128

表 3 シミュレーションパラメータ

パラメータ	値
リンク設定	タイプ:ポイントツーポイント，速度:10Mbps，MTU:9000 バイト
シミュレーション時間	開始 10 秒間がノード設定，その後 30 秒間テストトラフィックの送受信
テストトラフィック	TCP 1 フロー，MPTCP によって送信元ポートの異なる 8 サブフローに分割
MPTCP パスマネージャー	ndiffports (8 サブフロー)
TCP 輻輳制御機構	CUBIC (Linux デフォルト)
ネットワークスタック	liblinux [22] (Linux Kernel 3.14.33, MPTCP version 0.89)

表 4 シミュレーション結果

		k-shortest path (k)						
トポロジサイズ	ECMP	2	3	4	5	6	7	8
4 ポート	1.12	1.18	1.22	1.23	1.22	1.23	1.19	1.17
6 ポート	1.21	1.19	1.28	1.30	1.28	1.30	1.30	1.31
8 ポート	1.25	1.16	1.28	1.31	1.31	1.31	1.30	1.30

プットを向上できたかについて評価を行う．本手法によって，COTS スイッチで構築したネットワークにおいても，最短経路のみを用いた場合と ECMP を用いた場合に対してより多くのリンクを利用してトラフィックを転送できる．これを確認するため，シミュレータ上でランダムグラフネットワークを構築し，最短経路のみを用いた場合，ECMP を用いた場合，そして提案手法を用いた場合に転送できたトラフィック量を計測し，比較した．

実験環境として，ns-3 [20] とその拡張である ns-3 Direct Code Execution (ns-3-dce) [22] を用いた．ns-3 はネットワークシミュレータであり，ns-3-dce は，Linux カーネルをそのまま ns-3 上で実行することができる拡張である．ns-3-dce を用いることで，実機用に実装した `iplb` のコードを変更を加えることなくそのままシミュレーションに用いた．このシミュレータ上で，表 2 に示すサイズの異なる 3 つのトポロジを構築した．ポート数に対するスイッチの数とサーバの数は，3-level k-ary Fat-tree に準ずる [2]．このシミュレータ上でネットワークに接続されたサーバには，Linux カーネルモジュールとして実装した `iplb` を用いて他のサーバへの k-shortest path を通る中継点の設定を行った．カプセル化には，Generic Routing Encapsulation (GRE) を用いた．スイッチには同様に Linux カーネルを用い，基本的な IP ルーティングの設定と，GRE カプセル化を解く設定のみを行った．また，全てのリンクの MTU は 9000 バイト，リンク速度はシミュレーション時間の短縮のため 10Mbps とした．その他のシミュレーションパラメータを表 3 に示す．また `iplb` の実装及び本シミュレーションコードを筆者らの Github レポジトリにて公開している †1．実験では，それぞれのトポロジにおいて，各サーバがランダムに選択した宛先サーバに対して 8 つの TCP サブフローから成る MPTCP トラフィックを同時に送信し，各サーバが受信できたバイト数の合計を計測した．

各トポロジにおいて，テストトラフィックを送受信するサーバのペアをランダムに変更しながら 15 回実験を行った結果の平均を表 4 に示す．表 4 の値は，最短経路のみを用いた場合に各サーバが受信できたバイト数の合計を 1 として正規化したものである．この実験結果から，4 ポートの際は $k = 3$ から $k = 6$ までの k-shortest path を利用した場合，最短経路のみの場合に対して約 1.2 倍スループットが向上することがわかった．$k = 6$ までなのは，各サーバから宛先までのパスに対してネットワーク全体のリンクの数が少

†1 https://github.com/upa/iplb

ないためである．そして 6 ポート以上のサイズのランダムグラフでは，$k = 6$ 以上の k-shortest path を利用することで最短経路のみを用いた場合に対して 1.3 倍ほどのスループットを実現できることがわかった．また全てのサイズにおいて，ECMP よりも高いスループットを実現できた．

$k = 8$ までの k-shortest path と MPTCP の併用によるスループットの向上は Jellyfish の論文中において述べられている．本実験では，シミュレータ上のスイッチは一般的な IP ルーティングと GRE の機能のみを持つ．ここに `iplb` を実装した Linux カーネルをホストとして接続して利用することで，Jellyfish で示された通り ECMP よりもスループットが向上することを確認した．このことから，提案手法は Jellyfish で提案されるランダムグラフトポロジにおける k-shortest path と MPTCP によるスループットの向上を，COTS スイッチのみを用いたデータセンターネットワークで実現できるといえる．

5.2 カプセル化によるオーバーヘッド

提案手法では k-shortest path の特定パスにトラフィックを転送するためにパケットをカプセル化する．この手法は，パスの中継点ごとにカプセル化ヘッダが必要なため中継点の数が多いほどカプセル化の回数も増え，MTU サイズの減少による転送量の低下につながる．また，カーネルモジュールである `iplb` 内での経路検索やカプセル化処理自体によってサーバの送信性能が低下することが考えられる．そこでまず，k-shortest path を実現するために各パスで必要になる中継点の数を調べ，MTU サイズの減少によるスループットの低下率を検証した．その上で，`iplb` を用いた場合と用いない場合についてサーバの送信性能を比較し，カーネルモジュールの処理による送信性能の低下について評価を行った．

5.2.1 複数の中継点による MTU サイズの減少

ランダムグラフトポロジにおいて必要な中継点の数を求めるため，簡単なシミュレータを実装した．シミュレータは，指定されたサイズのランダムグラフトポロジを Jellyfish にもとづいて生成し，各サーバから他のサーバまでの k 本のパスとその中継点を算出する．トポロジのサイズは 5.1 節での実験と同様 k-ary Fat-tree トポロジに従い，スイッチのポート数によって決まる．スイッチのポート数を p とすると，サーバ数は $p^3/4$，スイッチ数は $p^2 + p^2/4$ となる．

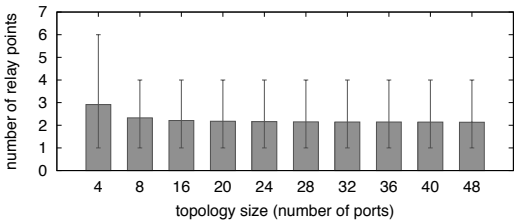


図 5　各トポロジサイズにおける中継点数

図 5 に，トポロジサイズに対して，$k = 8$ の k-shortest path の各パスを指定するために必要な中継点の数を示す．X 軸はトポロジサイズを，Y 軸は，そのサイズのランダムグラフトポロジで全サーバ間に設定したパスの中継点数の平均および最大最小値を示している．図 5 から，最も小さいサイズである 4 ポートスイッチ (16 サーバ，20 スイッチ) の場合に，平均 3 つ，最大 6 つと最も中継点が必要となっている．これは，8 本の k-shortest path を作成するためにはトポロジサイズが小さく，多くの迂回が必要なためである．8 ポート以上のトポロジサイズでは，トポロジサイズによらず 8 本のパスを作成するための中継点数は，平均で 2 つ，最大で 4 つとなった．

カプセル化のフォーマットとして GRE を用いた場合，1 回のカプセル化で GRE ヘッダと外側 IP ヘッダの追加によって，パケットサイズが 24 バイト減少する．つまり，1 つのパスを指定するために 2 つの中継点が必要な場合，48 バイトのパケットサイズの減少が起こる．MTU サイズが 1500 バイトの場合，48 バイトのパケットサイズの減少は，3.2%の性能低下となる．しかし，データセンターネットワークは 1 つの管理者によって管理されるネットワークであり，MTU が 9000 バイト以上のジャンボフレームを利用できる．MTU を 9000 バイトとした場合，48 バイトのカプセル化オーバーヘッドは，0.5%の性能低下にしかならない．また図 5 から，8 ポート以上のトポロジサイズで最も多く中継点を必要とするのは 4 つで

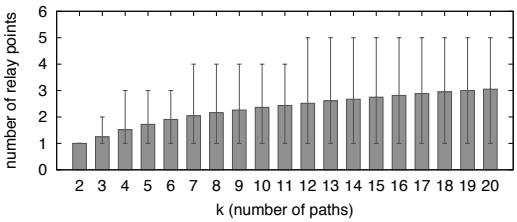


図 6 16 ポートスイッチのトポロジにおける中継点数

あり，4 回のカプセル化でも低下率は 1%である．このことから，ジャムボフレームの利用可能なデータセンターネットワークにおいては，カプセル化によるパケットサイズの減少は問題にならないといえる．

また，図 6 に 16 ポートスイッチ (1024 サーバ，320 スイッチ) のランダムグラフトポロジにおける，k-shortest path のパス数に対する中継点数の変化を示す．図 5 同様，Y 軸はパス数 k に対して，全サーバ間に設定したパスのカプセル化回数の平均値と最大最小値を示している．図 6 から，パス数 k の増加に応じて，必要な中継点の数も増えることがわかる．しかし，その増加分は，$k = 20$ でも $k = 8$ に対して平均および最大値が 1 回増える程度である．Jellyfish で示され，5.1 節での追試の通り，ランダムグラフトポロジにおける MPTCP と k-shortest path の利用では，$k = 8$ でスループットを向上できることがわかっている．このことから，トポロジサイズに対するカプセル化回数同様，k の数に対するカプセル化回数の増加も問題にならないといえる．

5.2.2 iplb によるオーバーヘッド

次に，サーバに `iplb` を適用した場合と適用しなかった場合について送信性能の計測を行った．実験では，2 台のサーバを用意し，その間に 1 台のレイヤー 3 スイッチを挟んで接続し，一方のサーバからもう一方へとパケットを送信した．`iplb` を用いる場合には間のレイヤー 3 スイッチがカプセル化を解除するよう設定した．リンク MTU は全て Juniper Networks の機器における最大 MTU である 9216 バイト，カプセル化のフォーマットには GRE を用い，1Gbps での計測時には CPU Intel Xeon L3426 1.87GHz，メモリ 4GB，NIC Intel e1000 (82574L) のサーバと，スイッチとして Juniper Networks EX4200 を，10Gbps での計測時には CPU Intel Xeon CPU E5-2650 2.60GHz，メモリ 32GB，NIC Intel X520 (82599ES) のサーバと，スイッチとして Juniper Networks QFX5100 を用いた．トラフィックの送信と計測には iperf を用い，10 秒間 TCP のテストトラフィックを送信する試験を各組み合わせについて 20 回行った．

`iplb` を用いる場合と用いない場合，そしてリンク速度を 1Gbps と 10Gbps で計測した結果を表 5 に示す．表 5 中の低下率とは，`iplb` を用いた結果が用いなかった結果に対してどの程度性能が低下したかを示している．GRE による 1 度のカプセル化はパケットサイズを 24 バイト減少させる．これは，MTU 9216 バイトでは 0.26%の性能低下となる．リンク速度 1Gbps の際，`iplb` を用いない場合は 990Mbps に対して用いた場合は 987Mbps であり，実測値では平均 0.3%の性能低下であった．またリンク速度が 10Gbps の際は，`iplb` を用いない場合の 8929Mbps に対して 8925Mbps と，平均 0.04%の性能低下となった．しかしこの性能低下は，実験の計測誤差の中に収まっている．以上の結果から，両方のリンク速度において，`iplb` による性能低下は理論値の近くに収まっており，`iplb` 内での経路検索やカプセル化自体の処理は大きなボトルネックではないといえる．

5.3 トポロジ変更への追従性

提案手法は，ランダムグラフネットワークの経路制御プロトコルとして OSPF を用いる．各サーバ上で動作する `iplbd` は OSPF の LSDB の情報をもとにトポロジの把握と k-shortest path の計算を行う．OSPF は堅牢性の高い経路制御プロトコルとして一般に広く利用されており，大規模環境への適用のための最適化手法も多く提案されている [8]．また既存研究のひとつである VL2 [10] も，データセンターネットワークの経路制御に OSPF を用いている．

OSPF によるレイヤー 3 ネットワークで `iplb` を用いた際，経路変更時の k-shortest path の再計算時間が問題となる．`iplb` において，k-shortest path は，宛先となるサーバごとに設定される．しかし実際にはひとつのスイッチの下に複数のサーバが収容されるた

表 5 iplb によるサーバ送信性能の低下

リンク速度	iplb 無し	標準偏差	iplb 有り	標準偏差	低下率
1Gbps	989.6Mbps	0.10Mbps	987Mbps	1.38Mbps	0.30%
10Gbps	8929Mbps	38.5Mbps	8925Mbps	104.1Mbps	0.04%

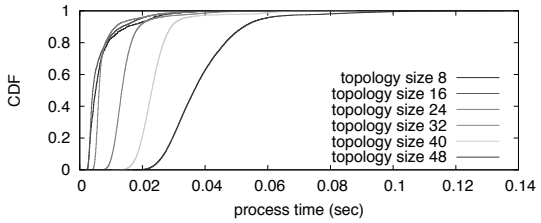


図 7 k-shortest path の計算時間

め，各サーバは他のサーバを収容する各スイッチまでの k-shortest path を求めるだけでよい．Yen のアルゴリズムを用いて k 本の shortest path を求める計算量は，$O(kn(e+n\log n))$，e はリンク数，n はスイッチ数，である．トポロジ変更が発生した際，各サーバはこの計算をスイッチごとに行う必要がある．この計算が現実的な時間で完了するかを確認するため，iplbd が k-shortest path を計算するのに要する時間について評価を行った．その上で，実機を用いた実験環境を構築し，小規模な環境で iplbd が実際に動作することを確認した．

iplbd の計算時間に関する実験では，実際に数千台規模のネットワークを用意することは難しいため，Python で実装した iplbd に事前に作成したランダムグラフのトポロジ情報を与え，k-shortest path の計算にかかる時間を計測した．実験には，CPU が Intel i7-3770K 3.50GHz，32GB メモリのマシンを用い，処理系には PyPy 2.0.2 (Python 2.7.3) を用いた．実験では，iplbd に異なるサイズのトポロジ情報を与え，ひとつのサーバから他のサーバを収容する各スイッチまでの 8 本の shortest path を計算するのにかかる時間を計測した．また各サイズにおいて，トポロジをランダムに変更しながら 10 回実験を行った．このトポロジのサイズはこれまでの実験と同様 3-level k-ary Fat-tree トポロジのサイズに従う．

図 7 に，各トポロジサイズにおいて，ひとつのサーバから各スイッチまでの k-shortest path の計算に要した時間を累積密度分布に並べたものを示す．図 7 から，トポロジサイズが 24 ポートスイッチの場合まで，ひとつの宛先への k-shortest path を計算する時間は，9 割までが約 0.01 秒ほどであることがわかる．16 ポートスイッチの場合，サーバ数は 1024 台，スイッチ数は 320 台のため，約 3.2 秒で全スイッチへの k-shortest path の計算が完了する．24 ポートスイッチの場合，サーバ数は 3456 台，スイッチ数は 720 台のため，全スイッチへの 8 本のパスを計算するのに要する時間は約 7.2 秒となる．32 ポート以上のトポロジサイズでは，9 割が 0.02 秒以内に完了するものの，スイッチ数が 1620 台のため，全スイッチまでの計算には約 32.4 秒要する．そのため，現在の実装で全スイッチへの k-shortest path を再計算する場合には，16 ポートスイッチまでが現実的な時間に計算が終わるサイズであることがわかった．

今後の課題として，k-shortest path の計算時間をより短くするために，より高速なアルゴリズムと実装を用いる必要がある．ループの無い k-shortest path を求めるアルゴリズムとして，Yen のアルゴリズムよりも計算量の少ないアルゴリズムが Hershberger らによって提案されている [12]．また本実装では Python を用いたが，C 言語などを用いてより最適化した実装を行うことで計算時間を短縮できる可能性がある．

次に，実機を用いて実際に iplbd によってパスを再計算し，通信を復旧することができるか実験を行った．本実験のトポロジを図 8 に示す．2 台のサーバを用意し，その間に 2 本のパスができるように 4 台のスイッチを接続した．その上で，全てのスイッチで OSPF を動作させ，サーバでは iplbd を動作させた．実験では，iplbd による最初のパス設定が終わった後，サーバ S からサーバ D へ iperf を用いて TCP トラフィックを流しつづけた．iperf の生成したトラフィックは MPTCP によって 8 本のサブフローに分

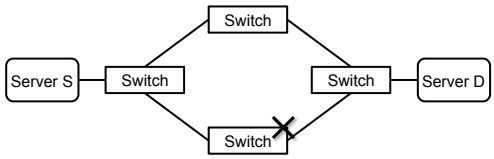


図 8 復旧時間の実験トポロジ

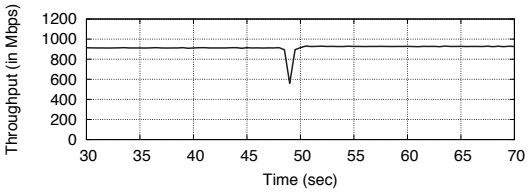


図 9 リンク断発生時の受信トラフィック量の推移

割され，各パスへと分散されてサーバ D へと送信された．そして図 8 に示す下側スイッチの持つ 2 本のリンクを切断し，中継点のひとつであるこのスイッチをネットワークから取り除いた．このとき，OSPF と `iplbd` によってパスの復旧が完了するまでサーバ D で受信できたトラフィック量を計測した．本実験の結果を図 9 に示す．図 9 の X 軸はトラフィックを送信し始めてからの時間，Y 軸はサーバ D で受信できたトラフィック量を示している．このトラフィック量は，iperf サーバ (受信側) において 0.5 秒の頻度で取得したものである．

図 9 から，48 秒の時点で上記のリンクを切断し，49 秒に受信トラフィック量が 600Mbps 以下に，そして 49.5 秒には約 900Mbps に戻ったことがわかる．これは，リンクが切断された瞬間，物理層のリンク断を検知した OSPF が新しい LSA を送信し，それを受信した `iplbd` がパスの再計算を行い，障害が発生したパスを `iplb` から削除したためである．パスが削除されるまでの間，上側スイッチを経由するパスを通るサブフローによって通信が行われていた．`iplbd` によって不通となったパスの削除が行われた後，該当パスが無くなったサブフローは別のパスへと再割り当てされる．そして残った 1 本のパスを経由して全てのサブフローが送信され，TCP が切断されることなく復旧された．このことから，提案手法は OSPF を利用した制御機構によって，実機環境においてレイヤー 3 ネットワークの可用性を活かした障害の復旧が可能であることがわかった．

6 結論と今後の課題

本研究では，IP トンネル技術を用いて，ランダムグラフトポロジのレイヤー 3 データセンターネットワークを COTS スイッチのみを用いて構築する手法を提案した．また，OSPF を用いた制御システムの設計と実装を行った．そして，シミュレータを用いた評価を行い，Jellyfish で提案されたランダムグラフトポロジにおける k-shortest path と MPTCP を用いたスループットの向上を，COTS スイッチの持つ機能のみを用いて実現できることを示した．さらにランダムグラフトポロジにおけるカプセル化回数に関する評価と実機を用いた実験を行い，カプセル化やサーバへの変更が性能低下の大きな要因とはならないこと，また OSPF を用いた制御機構によりトポロジ変更時に自動的な復旧が可能であることを示した．

今後の課題として，他のトポロジへの適用がある．事前研究において提案手法を Fat-tree トポロジに適用し，本論文ではランダムグラフトポロジへの適用と復旧の自動化を行った．しかし，他にもデータセンタネットワークのために多くのトポロジが提案されている [1] [11]．今後，これらのトポロジへも適用可能な形に提案手法を拡張していく予定である．

参 考 文 献

[1] Ahn, J. H., Binkert, N., Davis, A., McLaren, M. and Schreiber, R. S.: HyperX: Topology, Routing, and Packaging of Efficient Large-scale Networks, in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, ACM, 2009.

[2] Al-Fares, M., Loukissas, A. and Vahdat, A.: A Scalable, Commodity Data Center Network Architecture, in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, ACM, 2008.

[3] Andreyev, A.: Introducing data center fabric, the next-generation Facebook data center network, https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/, November 2014. Facebook, Inc.

[4] Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q. and Zhani, M.: Data Center Network Virtualization: A Survey, *Communications Surveys & Tutorials, IEEE*, Vol. 15, No. 2(2013), pp. 909–928.

[5] Cisco Systems, Inc: Cisco Global Cloud Index : Forecast and Methodology, 20142019, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html.

[6] Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt, D. and Banerjee, A.: Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS, RFC 7176, IETF, May 2014.

[7] Ford, A., Raiciu, C., Handley, M. and Bonaventure, O.: TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824, IETF, January 2013.

[8] Goyal, M., Soperi, M., Baccelli, E., Choudhury, G., Shaikh, A., Hosseini, H. and Trivedi, K.: Improving Convergence Speed and Scalability in OSPF: A Survey, *Communications Surveys & Tutorials, IEEE*, Vol. 14, No. 2(2012), pp. 443–463.

[9] Greenberg, A., Hamilton, J., Maltz, D. A. and Patel, P.: The Cost of a Cloud: Research Problems in Data Center Networks, *SIGCOMM Comput. Commun. Rev.*, Vol. 39, No. 1(2008), pp. 68–73.

[10] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P. and Sengupta, S.: VL2: A Scalable and Flexible Data Center Network, in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, ACM, 2009.

[11] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y. and Lu, S.: BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers, in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, ACM, 2009.

[12] Hershberger, J., Maxel, M. and Suri, S.: Finding the K Shortest Simple Paths: A New Algorithm and Its Implementation, *ACM Trans. Algorithms*, Vol. 3, No. 4(2007), Article 45.

[13] Kim, C., Caesar, M. and Rexford, J.: Floodless in Seattle: A Scalable Ethernet Architecture for Large Enterprises, in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, ACM, 2008.

[14] Lapukhov, P.: Building Scalable Data Centers: BGP is the Better IGP, https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf, June 2010. NANOG 55.

[15] Microsoft: Windows Azure's Flat Network Storage to Enable Higher Scalability Targets, http://blogs.msdn.com/b/hanuk/archive/2012/11/04/windows-azure-s-flat-network-storage-to-enable-higher-scalability-targets.aspx.

[16] Moy, J.: OSPF Version 2, RFC 2328, IETF, April 1998.

[17] Mudigonda, J., Yalagandula, P., Al-Fares, M. and Mogul, J. C.: SPAIN: COTS Data-center Ethernet for Multipathing over Arbitrary Topologies, in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, USENIX Association, 2010.

[18] Nakamura, R., Sekiya, Y. and Esaki, H.: Layer-3 Multipathing in Commodity-based Data Center Networks, in *Global Internet Symposium 2015, IEEE*, GI'15, 2015.

[19] Niranjan Mysore, R., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V. and Vahdat, A.: PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric, in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, ACM, 2009.

[20] ns-3 project: http://www.nsnam.org/.

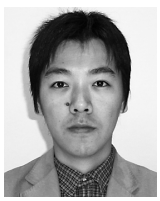
[21] Singla, A., Hong, C.-Y., Popa, L. and Godfrey, P. B.: Jellyfish: Networking Data Centers Randomly, in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, USENIX Association, 2012.

[22] Tazaki, H., Uarbani, F., Mancini, E., Lacage, M., Camara, D., Turletti, T. and Dabbous, W.: Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments, in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, ACM, 2013.

[23] Yen., J.: Finding the k shortest loopless paths in a network, in *Management Science, 1971*, 1971.

中 村 遼

2012 年慶應義塾大学環境情報学部卒業，2014 年東京大学情報理工学系研究科修士課程了，2014 年同後期博士課程入学．オーバーレイネットワークとルーティングの研究に従事．



石 原 知 洋

2001 年日本大学理工学部物理学科卒，2003 年慶應義塾大学政策・メディア研究科修士課程卒業．2009 年慶応義塾大学政策・メディア研究科後期博士課程修了，2010 年博士 (政策・メディア)．2009 年より東京大学総合文化研究科特任助教に就任．2015 年より同助教．ドメインネームシステムおよびイン

ターネットの運用技術に関する研究・開発に従事.

関谷勇司

1997 年京都大学総合人間学部卒. 2005 年慶應義塾大学政策・メディア研究科後期博士課程修了. 博士 (政策・メディア). 2002 年より東京大学情報基盤センター助手. 2011 年同センター准教授. クラウドを構成する要素技術と SDN, NFV, サイバーセキュリティに関する研究に従事.

江崎浩

1987 年九州大学大学院・工・電子修士課程了. 同年 (株) 東芝入社. 1990 年米国ニュージャージ州ベルコア社. 1994 年コロンビア大学・客員研究員. 1998 年東京大学大型計算機センター・助教授. 2001 年同大学大学院・情報理工学系研究科・助教授. 2005 年同大学大学院・同研究科・教授, 現在に至る. 博士 (工学, 東京大学). MPLS-JAPAN 代表, IPv6 普及・高度化推進協議会専務理事, WIDE プロジェクト代表, JPNIC 副理事長.