Vol.2018-OS-142 No.11
2018/2/28

マルチテナント型SDN仮想化基盤のための ネットワーク抽象化手法

樋口 俊[1,a)]　廣津 登志夫[1,b)]

概要：近年のサーバ仮想化技術の発展により，既存の IT 基盤をデータセンタ上で仮想化して提供するクラウドサービスの普及が進んでいる．こうしたサービスを提供しているマルチテナント型データセンタでは OpenFlow 技術が提供する仮想化や集中制御の機能を活用して，多数のテナント向けネットワークを提供している．これに対して，OpenFlow 技術による柔軟な制御をテナント側でも利用することに対する期待が高まっている．そこで，テナント毎に自由なネットワーク設計を可能とする OpenFlow ネットワークの仮想化技術の実現に向けて，各テナントのアドレス空間の衝突管理と仮想トポロジの構築手法による SDN 仮想化基盤を提案した．そこでは実際のサービスに向けて，物理ネットワークのトポロジを隠蔽した上でテナントが必要とする規模の仮想トポロジに抽象化して提示する手法が必要となる．そこで本論文では，各テナントの仮想ネットワーク設計を支援する物理トポロジの抽象化手法について提案する．また，新たにカプセル化技術を用いることで各テナントの OpenFlow 制御を制約しない仮想トポロジの構築技術についても述べる．

A Network Abstraction Method for Multi-Tenant SDN Hypervisor

Shun Higuchi[1,a)]　Toshio Hirotsu[1,b)]

1. はじめに

サーバ仮想化技術の発展によって，組織が必要とする IT インフラをデータセンタ上で仮想化しインターネット経由で提供する IaaS(Infrastructure as a Service) などのクラウドコンピューティングサービスが普及している．このようなサービスを提供するデータセンタでは，単一の物理リソースを複数企業で共有するマルチテナントへの対応が必要となる．その中でも，マルチテナントネットワークでは 1 つの物理ネットワークを複数のテナント用仮想ネットワークへと論理的に分割し，それぞれの仮想ネットワーク内で行われる通信が分離されている必要がある．これらを実現するネットワーク仮想化技術としては従来の VLAN 技術の利用が一般的であった．IaaS 提供者は各テナントネットワークに VLAN-ID を割り当てることで 1 つの物理ネットワークを複数のレイヤ 2 ネットワークに分割し，各テナントのネットワーク管理者は割り当てられた複数のレイヤ 2 ネットワークを組み合わせて自由にレイヤ 3 ネットワークを構築していた．VLAN を利用した手法ではネットワーク構成の変更の度に IaaS 提供者が必要な全ネットワーク機器に対して VLAN の設定を変更することが必要になる．従来に比べ動的に仮想ネットワークや仮想マシンの動的な増減が頻発するクラウド環境では，構成の変更に対応したより柔軟な仮想ネットワークの構築・管理手法が必要とされる．

この要求を満たす技術として，近年注目されている Software-Defined Network (SDN)[1] の代表的アーキテクチャである OpenFlow[2] が挙げられる．OpenFlow は経路制御を行うコントローラとデータ転送を行うスイッチを分離することで，柔軟な経路制御とネットワークの集中管理が可能となっている．OpenFlow ではコントローラからスイッチに書き込むフローエントリにおいて VLAN-ID の認識・書き換えなどが指定できるため，構成変更に伴う

[1] 法政大学
Hosei University
[a)] s.higuchi@dsl.k.hosei.ac.jp
[b)] hirotsu@hosei.ac.jp

VLAN 管理を 1 つのコントローラに集約することができる．これによって，IaaS 提供者は SDN 技術をの恩恵を受け柔軟な仮想ネットワークの管理を一元的に実現することが可能となっている．またその一方で，テナント側にすると SDN 技術を用いることで「ソフトウェアにより柔軟にネットワークを制御することが可能になる．しかし，SDN 技術を IaaS 基盤の管理・運用技術に加えて各テナントのネットワーク制御技術としても提供するには，複数のテナントが発する OpenFlow などのプロトコルによる SDN 処理の要求を競合を解消しつつ SDN 基盤で直接処理することが必要となる．

複数の OpenFlow コントローラの要求を処理する技術としては FlowVisor[5] が挙げられる．FlowVisor は予めネットワークアドレスをフロースペースという単位に分割し，各テナントは割り当てられたフロースペースの範囲内で書き込むフローエントリなどの設定を行う．しかし，各フロースペース間の競合検証を行っていないため衝突の回避は IaaS 管理者によるネットワーク設計に依存していた．そのため，設計ミスによる競合のリスクは免れず，またテナントユーザにとっては自由な設計が許されていないネットワークとなっていた．

これに対して，テナント毎の自由なネットワーク設計を目的としてフロースペースの競合検証と Multi-Protocol Label Switching(MPLS) を用いた仮想トポロジの構築手法による SDN 仮想化基盤を実現してきた [6]．ここでは，各テナントが定義したフロースペース間の重複部分を検証・管理し，MPLS ラベルを用いてフローエントリを変換することで各テナントの仮想トポロジ間で独立性を保証する手法について提案している．しかし，実際の運用を考慮すると物理トポロジ全体を単純にテナントに提示するのではなく，ネットワーク設計を支援するために冗長化や負荷分散が可能な仮想トポロジを抜き出して提示するトポロジ抽象化の機能が必要である．また，MPLS を利用した仮想トポロジの構築手法では各テナントが OpenFlow の機能として MPLS の制御を利用できない設計となっていた．

そこで，本研究では物理トポロジを抽象化して十分な信頼性を持つ仮想トポロジを構成する手法と OpenFlow 制御を制限してしまう MPLS の代わりに Virtual eXtensible Local Area Network (VXLAN)[3] を用いた仮想トポロジの構築手法について提案する．まず，物理トポロジから冗長性を持った複数の経路を抜き出して抽象化し仮想トポロジをテナントに提示する．この仮想トポロジを OpenFlow の拡張仕様と VXLAN を用いて実際の物理ネットワーク上に構築する．

2. OpenFlow/SDN

クラウド環境を中心とする次世代基盤技術として注目を集めているのが，Software-Defined Network の代表的アーキテクチャの 1 つとして標準化が進む OpenFlow である．OpenFlow ではネットワークの経路制御を担う OpenFlow コントローラと，その制御に従ってパケットの転送を行う OpenFlow スイッチによって，従来のネットワーク構成からコントローラプレーンとデータプレーンに分離したネットワークを一元管理可能とする中央制御型アーキテクチャとなっている．

OpenFlow ではソフトウェアであるコントローラにおいて MAC アドレスや IP アドレス，トランスポート番号，VLAN-ID などのパケットに対するマッチ条件とパケットへの処理の組をフローエントリとして定義する．各 OpenFlow スイッチがこのフローエントリに従ってパケットを処理することで柔軟な経路制御を実現している．また，ネットワーク構成の変更に伴いスイッチの再設定が必要な場合は変更内容をコントローラ上で記述するだけで全てのスイッチに変更が反映されるなど，高いネットワークの管理性を実現している．コントローラとスイッチ間は，データネットワークとは別に構築される OpenFlow チャンネルという TCP/IP を用いた制御ネットワークによって接続され，OpenFlow メッセージと呼ばれる制御情報のやりとりが行われる．コントローラはこの OpenFlow メッセージを通して，フローエントリの書込みなどのスイッチ制御を行う．OpenFlow ではコントローラが全てのスイッチを制御しトポロジ情報を把握しているため，ソースルーティングやマルチパス転送といった柔軟な経路制御を行うことができる．加えて，OpenFlow を用いたネットワークの仮想化では VLAN の管理性向上だけでなく，マッチ条件として指定できるレイヤ 1〜4 のヘッダ情報を用いたネットワークの論理的な分割が可能となっている．アドレス空間を予め分割して利用することで，通常の VLAN-ID の上限を超えた数の仮想ネットワークを作成することが出来る．

これらの利点の一方で，従来の OpenFlow 技術では，OpenFlow ネットワークそのものを仮想化し制御する仕組みが提供されておらず，1 つの OpenFlow ネットワークに対して複数のコントローラから個別にスイッチ制御したり，1 つの OpenFlow ネットワークを論理的に複数の仮想 OpenFlow ネットワークに分割したりといったことができないという課題があった．この問題から，IaaS を提供しているマルチテナント型データセンタなどにおいて各テナントがそれぞれのコントローラと OpenFlow ネットワークを構築・制御するといった利用形態が不可能だった．

3. FlowVisor

FlowVisor は，コントローラとスイッチ間を接続する OpenFlow チャンネル上に配置され，コントローラからスイッチを制御するのに必要な OpenFlow メッセージを転送する透過型プロキシとして動作する．まず，FlowVisor の管理者が予めそれぞれのテナントで利用可能なネット

ワーク空間をフロースペースとして定義しておく必要がある．フロースペースでは，テナントネットワークの名前を示すスライス名と OpenFlow スイッチの DPID，そして MAC アドレスや IP アドレス，トランスポート番号など OpenFlow のフローエントリ中で利用可能なレイヤ 1 からレイヤ 4 までのマッチフィールドと優先度の空間を定義する．また，それぞれのフロースペースで定義されているネットワークの空間は重複せず独立していることを前提としており，管理者が注意深くフロースペースの定義を個々のスイッチに対して行う必要がある．

FlowVisor では，まず管理者が各テナントが利用できるネットワーク空間をフロースペースとして定義する．テナントは FlowVisor の管理者から提示された仮想 OpenFlow ネットワークのトポロジ情報とフロースペース情報に基づいて，フローエントリとそれを書き込むコントローラを作成する．テナントはこのコントローラを FlowVisor に接続してテナントネットワークを制御することができる．

FlowVisor では，各テナントに自由にネットワークを設計させることを想定してそれぞれが定義したフロースペースをそのまま利用すると，他のフロースペースと衝突するようなフローエントリが書き込まれて意図しないトラフィック制御が行われる恐れがある．FlowVisor は管理者にネットワーク設計を委ねることでテナント間で競合することを防いでいるが，テナントにとっては自由なネットワーク設計が制限されているため，IaaS 環境に求められている自由かつ柔軟なマルチテナントネットワークの提供とは異なっている．

4. マルチテナント型 SDN 仮想化基盤

FlowVisor のネットワーク設計の制約を解決するためには，各テナントが設計したフロースペースにおいて重複管理と衝突検証を行うシステムが必要である．これまでにこの検証を元にテナントネットワーク間の分離性を保証した仮想トポロジを構築することでテナント毎に自由なネットワーク設計を可能とする SDN 仮想化基盤を提案した [6][7]．図 1 に示したように提案した機構を OpenFlow ハイパーバイザ上に実装している．この研究は以下の 3 つの手法から成り立っている．

- 自由な設計と自動検証を支援する新たなフロースペース
- フロースペースのアドレス空間に対して重複部分を検証・管理する機構
- OpenFlow による MPLS のラベルスイッチングを利用した仮想トポロジの構築機能

これらの手法により，各テナントネットワークで利用するアドレス空間の組み合わせそのものをフロースペースとして定義し重複を検証・管理することで，テナントに自由な設計を許しながらフローエントリが衝突しうる組み合わせを予め管理しておくことが出来る．加えて，物理ネットワークと仮想テナントネットワークの柔軟なマッピングを行って，テナント間のフローエントリの衝突を効率的に回避することが出来る．

4.1 フロースペースの定義

FlowVisor のフロースペースでは各テナントが制御可能なスイッチ毎に定義を行っていたが，この手法のフロースペースでは 1 つのテナントネットワーク全体に対して利用可能なアドレス空間の組み合わせを定義する．1 つのフロースペースは複数のルールによって構成され，1 つのルールは表 1 に示されるようにルール ID，フロースペース名，テナントが利用可能なマッチングフィールドの並びから成っている．マッチングフィールドにはネットワークを定義する実運用上で必要な，VLAN ID，Src/Dst IP アドレスという L2・L3 の 3 種類のヘッダ情報を指定することができる．これらの内，Src/Dst IP アドレスについては管理性の面から最小を 27bit とするサブネット空間として設計を行う．1 つのフロースペースは，フロースペース名とそれぞれのフロー定義の集合で記述される．フロー定義はマッチフィールドの各要素毎に記述し，1 つのフロー定義は各フィールドの要素の AND として定義される．1 つのフロースペースは 1 つ以上のフロー定義で表すこととして，複数のフロー定義はいずれかに合致するフローエントリが許可され OR として機能する．これによって，各テナントではこのフロースペースで指定されたアドレス空間の組み合わせを利用することが出来る．定義例をまとめた表 1 において，最下段にある定義例 2 では以下のようなアドレス空間を形成している．

- VLAN ID = 100, Src IP = 192.168.64.0/20, Dst IP = 192.168.64.0/20
- VLAN ID = 101, Src IP = 192.168.64.0/20, Dst IP = 192.168.64.0/20

このフロースペースを割り当てられたテナントは，これら 2 種類の組み合わせをフローエントリのマッチフィールドに用いてネットワークを制御することが出来る．また，表 1 の上段にマッチフィールド中で利用可能なアドレス空間を示しているが，その内の VLAN ID の上限が本来の上限である 4096 個の半分となっている．これはフロースペースの重複管理において，衝突を解決するために必要な独立したアドレス空間を予め管理用空間として確保し，必要な場合に適時フロースペースに割り振るためである．

4.2 フロースペースの重複検証

各テナントのフロースペースを分解して作成されたフロー定義群について重複の検証と管理を行う．この重複検証では，単純に 2 つのフロー定義間でアドレス空間の包含関係について検証を行い，完全な包含関係にあるフロー定義をフローエントリの衝突が起こり得るフロー定義として

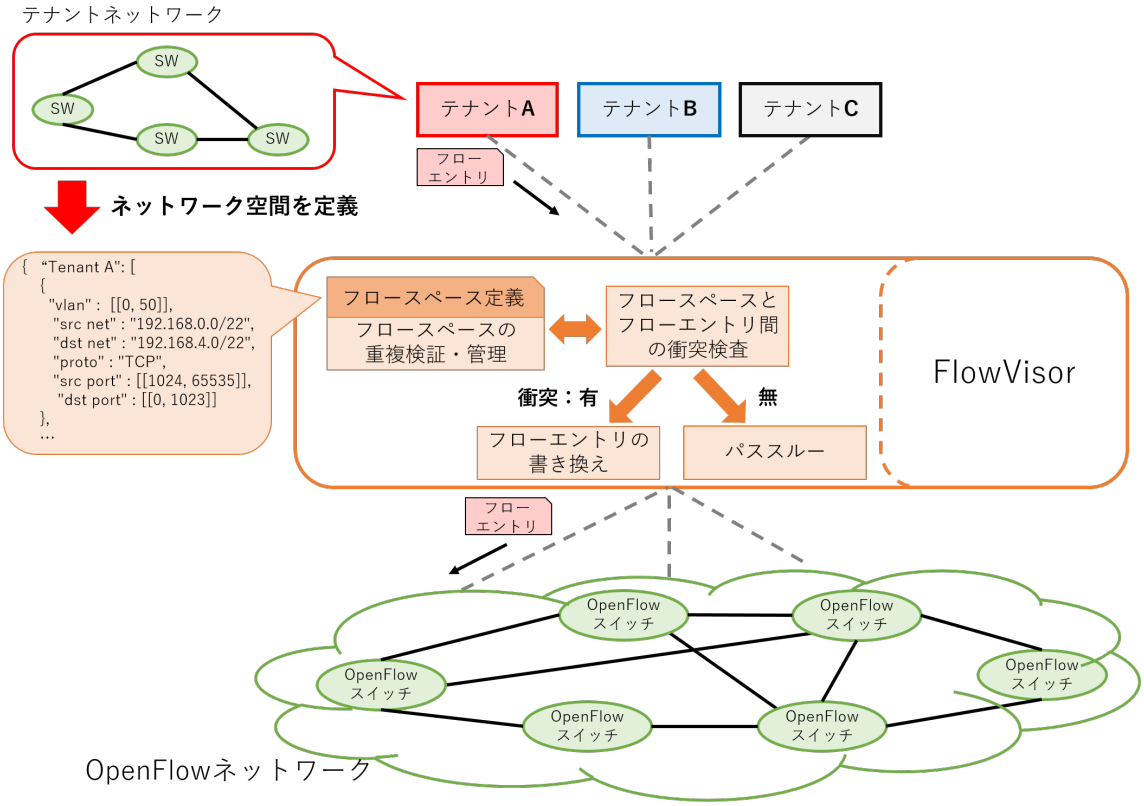

図 1　マルチテナント型 SDN 仮想化基盤

表 1　フロースペースの定義例

Rule ID	Space Name	VLAN	Src IP	Dst IP
1	定義例 1	0〜100	192.168.64.0/22	192.168.64.0/22
2	定義例 2	100, 101	192.168.64.0/20	192.168.64.0/20

重複関係を管理する．

重複の例として表 1 のフロースペースについて定義例 1 と定義例 2 を分解したフロー定義群の中には，以下の 2 つのフロー定義が存在している．

- 定義例 1 : VLAN ID = 100, Src IP = 192.168.64.0/22, Dst IP = 192.168.64.0/22
- 定義例 2 : VLAN ID = 100, Src IP = 192.168.64.0/20, Dst IP = 192.168.64.0/20

この 2 つのフロー定義は完全な包含関係にあるため，重複があるフロー定義として管理される．このような重複しているフロー定義において，テナントネットワークのトポロジに重なっている部分がある場合，そのスイッチ上ではテナント間でフローエントリの衝突が発生する可能性があるため，2 つのテナント間で仮想ネットワークが分離されている必要がある．

4.3　MPLS ラベルを用いた仮想トポロジの構築

OpenFlow では，フローエントリ中でマッチフィールドに MPLS ラベルを指定できるだけでなく，アクションとして MPLS ヘッダの付与と削除を行うことができる．MPLS ヘッダ中のラベルでは 20bit の値を表現できるため，この値を各テナントに対して予め一意に割り振りを行った上で，MPLS 制御用のフローエントリを書き込むことで仮想トポロジを構築する手法を提案する．

ここでは，MPLS ラベルと管理用のフローエントリを用いた 3 段階の制御によって仮想トポロジを構築する．まず，テナントのエッジスイッチそれぞれに対して，表 2 のようなホストからのパケットに MPLS ラベルを付与するフローエントリと，ホストに向けて送出されるパケットの MPLS ラベルを削除するフローエントリの 2 種類を設定する．この 2 つのフローエントリについては，ラベルの付与を行うフローエントリは最も早いフローテーブルでマッチし，ラベルの削除は最後のフローテーブルでマッチするというように指定して書き込む必要がある．続けて，図 2 のようにテナントの制御下にないスイッチを中継した経路が必要な場合には，表 2 中の MPLS ラベルをマッチフィールドとしたトンネリング用フローエントリを中継するスイッチに設定する．最後に，テナントから書き込まれるフローエントリに対して，マッチフィールドに MPLS ラベルを追加するように書き換えを行う．

表 2　MPLS を用いた仮想トポロジ構築フローエントリ

Entry	Table ID	Match Field	Action
ラベル付与	0	in_port = ホストの接続ポート	Push-MPLS
ラベル削除	255	mpls_label = テナントのラベル	Pop-MPLS
トンネリング	0	in_port = トンネルポート 1, mpls_label = テナントのラベル	Output:トンネルポート 2
トンネリング	0	in_port = トンネルポート 2 , mpls_label = テナントのラベル	Output:トンネルポート 1

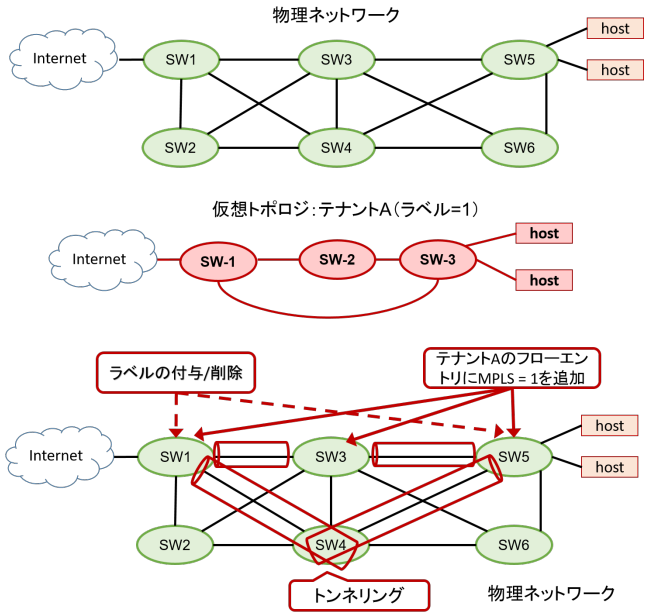

図 2　MPLS による仮想トポロジの構築

4.4　既存手法の課題

この研究では，マルチテナント型 SDN 仮想化基盤の実現を目的にフロースペースの競合管理手法に加えてテナントネットワーク間の分離性を保証する仮想トポロジの構築手法を提案している．これに対して IaaS 環境における実運用のためには，実際の物理的なネットワークトポロジを抽象化した上で各テナントに提供する手法が必要となる．この抽象化では，物理トポロジの全体像を隠蔽しながらテナントが必要とするエッジスイッチやゲートウェイ間の接続性を保証した仮想トポロジを提示することが重要である．

これに加えて，仮想トポロジ構築のために OpenFlow の機能として MPLS ラベルやヘッダの付与・削除を行っているが，この仕組みから各テナントは OpenFlow による MPLS の制御を利用することができない設計となっている．これに対して，MPLS 技術は拠点間の広域イーサネット構築を目的としておりテナントネットワークの内部で利用したいという要望は大きくないと論じているが，テナントがオンプレミスで運用している IT インフラと IaaS 上で構築した計算資源を相互接続する場合には MPLS を用いた VPN に需要が存在している．これらに対し，MPLS も含めて OpenFlow の標準機能を利用可能とするためには代替となるカプセリング技術を用いた仮想トポロジの構築手法が必要となる．

5.　提案手法

本研究では，テナント毎に自由なネットワーク設計が可能な OpenFlow ネットワークの仮想化基盤を実現するために，IaaS における実運用上で必要なテナントネットワークの抽象化手法と MPLS の代替となるカプセリング技術を用いた仮想トポロジの構築手法について提案する．まず，物理的なトポロジを抽象化し十分な信頼性を確保した仮想トポロジをテナントに対して提供する手法を導入する．この仮想トポロジは新たなテナントが追加され，初めに自身のネットワークを設計する際に実行される．この手法によって，テナントは物理トポロジ全体を不要に知ることなく，負荷分散や冗長化に必要な規模の仮想トポロジを利用することが出来る．加えて，VXLAN を用いた仮想トポロジの構築手法を提案する．この手法では，OpenFlow の拡張仕様を利用して VXLAN によるカプセリングと復号化を行うことで，仮想トポロジを物理ネットワーク上にマッピングした上でフローエントリの衝突を効率的に回避することができる．

6.　テナントネットワークの抽象化手法

テナントが新たに追加された際にはそのネットワーク設計を支援するために，テナントに利用可能なネットワークトポロジが提示される．この時，テナントに対して物理トポロジ全体を表示するのではなく，そのテナント用に抽象化された仮想トポロジが提示される．ここでは，ホストが接続されているエッジスイッチと外部に通信するゲートウェイスイッチ間の接続関係をベースとして，物理トポロジから複数の経路を抜き出すことで抽象化し仮想トポロジを構成する手法について提案する．この抽象化手法では，テナントが負荷分散や冗長化に利用する冗長経路を常に確保するために，サイクル構造となる仮想トポロジを構成する．

仮想トポロジの構成手順は以下の通りである．ここではスイッチ間のリンクを枝 $E = \{e_1, e_2, ...e_n\}$ として，トポロジのグラフを枝集合 $G = (E)$ で示す．

(1) グラフ G においてゲートウェイスイッチから各エッジスイッチに対して幅優先探索による最短経路を探索し，木 T_s を構成する．図 3 ではゲートウェイ SW1 とエッジスイッチ SW5・SW6 間の経路として $T_s = \{e_1, e_7, e_8\}$ が作成される．

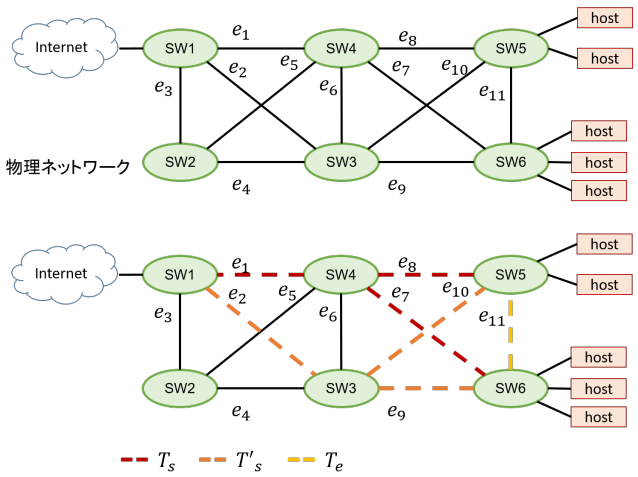

図 3　MPLS による仮想トポロジの構築

(2) 木 T_s とグラフ G の排他的論理和をとることで T_s を除外したグラフ G' を構成する．このグラフ G' においてもゲートウェイスイッチから各エッジスイッチ間の最短経路を探索し木 T'_s とする．この時, 経路が存在しない場合にはそのエッジスイッチに対する探索を打ち切る．図 3 では，T_s を用いて $G' = \{e_2, e_3, e_4, e_5, e_6, e_9, e_{10}, e_{11}\}$ が構成され，探索すると $T'_s = \{e_2, e_9, e_{10}\}$ となる．

(3) グラフ G においてエッジスイッチ同士を接続する最短経路群を幅優先探索で求めて枝集合 T_e とする．図 3 では，$G = \{e_1, e_2, ...e_{11}\}$ を用いて $T'_s = \{e_{11}\}$ となる．

(4) T と T，e の論理和をとって仮想トポロジのグラフ G とする図 3 では，$T_s = \{e_1, e_7, e_8\}$, $T'_s = \{e_2, e_9, e_{10}\}$, $T'_s = \{e_{11}\}$ の論理和から $G = \{e_1, e_2, e_7, e_8, e_9, e_{10}, e_{11}\}$ となる．

ゲートウェイスイッチとエッジスイッチ間において最短経路とその冗長経路を確保した上で，エッジスイッチ同士は相互に接続することで冗長性に優れるリングと部分的なメッシュトポロジが構成される．また，ゲートウェイスイッチが 2 つ以上存在する場合にはそれぞれに対して構成手順を適用して木を作成し，最後に論理和をとって合成した木 Tu を仮想トポロジとする．

7. VXLAN を用いた仮想トポロジの構築

OpenFlow では，標準の機能として VXLAN の ID である VNI をマッチフィールドで指定できる一方で，パケットのカプセリング/復号化自体は行うことはできない．これに対して，OpenFlow の拡張仕様である Nicira Extentions ではパケットに対する VNI の付与など VXLAN によるカプセリングと復号化をフローエントリを用いて行うことが出来る．本研究では，この機能を利用し VXLAN を用いた仮想トポロジの構築手法を提案する．OpenFlow の各バージョン間では互換性が無いこととは異なり，この Nicira Extentions によるフローエントリと通常のフローエントリは同時に利用可能なため，テナントコントローラでは標準

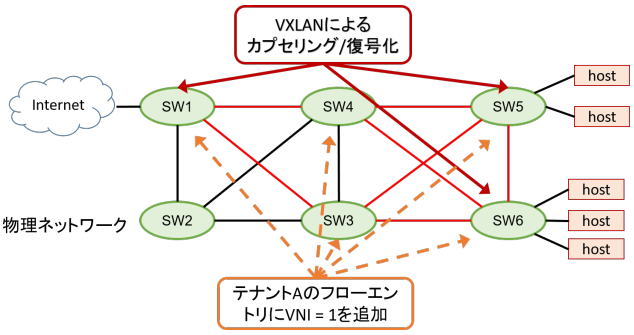

図 4　VXLAN による仮想トポロジの構築

仕様のフローエントリを書き込ませながら OpenFlow ハイパーバイザからは Nicira Extentions 仕様のフローエントリによって制御を行うことが可能になる．VNI には 20bit の値を表現できるため，この値をフロースペースが競合しているテナントに対して一意に割り振ることで約 1600 万以上のテナントを収容することができる．

ここでは，VNI と管理用のフローエントリを用いた 2 段階の制御によって仮想トポロジを構築する．まず，テナントのエッジスイッチそれぞれに対して，表 3 のようなホストからのパケットをカプセリングするフローエントリと，ホストに向けて送出されるパケットを復号化するフローエントリの 2 種類を設定する．カプセリング用のフローエントリでは，パケットとのマッチ条件にホストが接続している物理ポートを指定し，3 段階のアクションによってカプセリングを行う．3 段階のアクションでは，まず元のパケットヘッダをコピーし保持しておき，続いてパケットにテナントの VNI を付与する．最後にカプセル化ヘッダとしてコピーしておいた元のヘッダ情報を付与する．カプセル化ヘッダに元のヘッダと同じ情報を利用することで，後述のフローエントリ変換の際に VNI のみの書き換えで済む．復号化用のフローエントリでは，パケットとのマッチ条件に VNI を指定してアクションで VNI を 0 に指定することで復号化を行う．これらについて，カプセリングを行うフローエントリは最も早いフローテーブルでマッチし，復号化は最後のフローテーブルでマッチするというように指定して書き込む必要がある．最後に，テナントから書き込まれるフローエントリに対してマッチフィールドに VNI を追加するように書き換えを行う．

8. 考察

本研究では，テナント毎に自由なネットワーク設計が可能な SDN 仮想化基盤の実現を目的として，実際の運用で必要となる物理トポロジを抽象化して仮想トポロジを提示する手法と OpenFlow の拡張仕様と VXLAN を用いて仮想トポロジを構築する手法を提案した．提案手法では，複数の冗長経路を確保した仮想トポロジを提示することで，各テナントはトラフィックの振り分けによる経路負荷の分

表 3　VXLAN を用いた仮想トポロジ構築フローエントリ

Entry	Table ID	Match Field	Action
カプセル化	0	in_port = ホストの接続ポート	Copy-Field Set-Tunnel: テナントの VNI Set-Field: カプセル化前と同じヘッダ
復号化	255	tunnel_id_nxm = テナントの VNI	Set-Tunnel: 0

散やパケットの到達性を維持する冗長経路の構築などが可能になっている．また，この仮想トポロジを OpenFlow の拡張仕様と VXLAN を用いて構築することで，テナントは OpenFlow の機能を自由に用いてテナントネットワークを制御出来るとともに，テナント間の分離性を保証することが可能になっている．

SDN 仮想化基盤に関連している研究として，我が国で NICT が主導している VNode[8]，米国の GENI[9]，欧州の OFELIA[10] など SDN/OpenFlow テストベッドのプロジェクトが挙げられる．これらの内，VNode と GENI ではより独立したネットワークの割り当てを目指し，OpenFlow スイッチをベースに専用ハードウェアとマネジメントモデルを実装したネットワーク仮想化ノードを用いてデータプレーンを構成している．OFELIA プロジェクトは FlowVisor をベースとしたアーキテクチャとなっており，各利用者に対して VLAN-ID を指定したフロースペースを予め割り当てることで実験用ネットワークを分割している．これら 3 つのプロジェクトでは，研究開発を支援する SDN テストベッドとしての性質から物理ネットワーク上に作成したフルメッシュの仮想トポロジをそのまま提示しており，冗長性を考慮したテナントネットワークの抽象化は考慮されていない．

本論文中で提案した手法では，OpenFlow の拡張仕様である Nicira Extentions と VXLAN を利用することでテナント間のフローエントリの衝突を回避しながら仮想トポロジを構築している．これによって，テナントがオンプレミスで既に運用している IT インフラと IaaS 上で新しく構築した計算資源を広域ネットワーク越しに接続したい場合には，MPLS を用いて論理的なクローズドネットワークを構築することができる．

9. まとめ

本研究では，以前提案したフロースペースによる仮想ネットワーク定義の検証と MPLS を用いた仮想トポロジのマッピングによる SDN 仮想化基盤をベースとして，実際の運用上必要となる物理トポロジの抽象化手法と VXLAN を用いることで OpenFlow の機能を制限しない仮想トポロジの構築手法を提案した．これによって，実際の IaaS 環境において OpenFlow 技術を自由に用いることができる柔軟なテナントネットワークを提供するという目標の実現に近づいた．本研究の提案手法によって，物理トポロジを抽象化し負荷分散や冗長化が可能な仮想トポロジをテナントに提供することに加えて，OpenFlow を用いた任意の制御に対してもテナントネットワーク間の分離性を保証することが可能になっている．

謝辞　本研究は JSPS 科研費 JP15K00138 の助成を受けたものです．

参考文献

[1] N. McKeown, ”Software-defined networking,” INFOCOM keynote talk, vol. 17, no. 2, pp. 30-32, 2009.
[2] N. McKeown et al., ”OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, Issue 2, pp. 69-74, April 2008.
[3] M. Mahalingam, et al., ”Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.” Internet Engineering Task Force, RFC 7348, August 2014.
[4] ”OpenFlow Switch Specification Version 1.3.4 Implemented (Protocol 0x04)” March 27, 2014.
[5] R. Sherwood et al., ”FlowVisor: A Network Virtualization Layer,” Tech. Rep. OPENFLOW-TR-2009-01, OpenFlow Consortium, October 2009.
[6] 樋口俊, 廣津 登志夫. “フロースペース管理による SDN 仮想化基盤の提案” , 研究報告システムソフトウェアとオペレーティング・システム, pp. 1-8, July 2017.
[7] S. Higuchi, T. Hirotsu, ”A Verification Based Flow Space Management Scheme for Multi-Tenant Virtualized Network,” The Eleventh International Conference on Digital Society and eGovernments (ICDS), pp. 24-29, March 2017.
[8] Y. Kanada, K. Shiraishi and A. Nakao, ”Network-virtualization nodes that support mutually independent development and evolution of node components,” 2012 IEEE International Conference on Communication Systems (ICCS), pp. 363-367, November 2012.
[9] M. Berman et al., ”GENI: A federated testbed for innovative network experiments,” Computer Networks, vol. 61, pp. 5-23, March 2014.
[10] M. Suñé et al., ”Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed,” Computer Networks, vol. 61, pp. 132-150, March 2014.