Vol.2017-OS-141 No.13
2017/7/27

フロースペース管理によるSDN仮想化基盤の提案

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，データセンタを利用するユーザネットワーク側では OpenFlow 技術による柔軟な制御という恩恵をテナント側でも利用したいという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が可能な 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フローエントリの衝突検証の手法による SDN 仮想化のコア技術を実現した．しかし，実際に制御可能なテナントネットワークを構築するためには仮想トポロジの定義・構築技術やトラフィックの分離性を保証するために必要なフローエントリ中に対するアクションの検証技術が必要である．そこで本論文では，各テナントの任意の制御に対してテナント間の通信の分離を実現する仮想トポロジのマッピング手法とフローエントリ中のアクション部分の書き換え手法を提案する．

Proposal for SDN Virtualized Infrastructure using Flowspace Management

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，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 VLAN 管理を 1 つのコントローラに集約することができ

[1] 法政大学
Hosei University
[a)] shun.higuchi.6j@stu.hosei.ac.jp
[b)] hirotsu@hosei.ac.jp

る．これによって，IaaS 提供者が SDN 技術を利用することで柔軟な仮想ネットワークの管理を実現している．またその一方で，SDN 技術によって可能となった「ソフトウェアによる柔軟なネットワーク制御機能」を，テナント側からテナントネットワークの制御にも用いたいという要求が存在している．このように SDN 技術を IaaS 基盤の管理・運用技術として用いるのではなく，各テナントが利用可能なネットワーク制御技術として提供する場合には，各テナントが発する OpenFlow などのプロトコルによる SDN 処理の要求を直接処理できる仕組みが必要となる．

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 では OpenFlow コントローラとスイッチ間にプロキシとして配置し，それらの間で制御メッセージの交換を管理する．これによって 1 つの OpenFlow ネットワーク上で複数のコントローラからそれぞれ個別に OpenFlow スイッチを制御することが可能となっている．FlowVisor では予めネットワーク空間をフロースペースという単位に分割し各テナントユーザが自身のコントローラに割り当てられたフロースペースに従って書き込むフローエントリなどの設定を行う．この仕組みによって複数の仮想 OpenFlow ネットワークを構築し，それぞれを異なるテナントコントローラから制御することが出来ている．しかしながら，FlowVisor では各フロースペース間の競合検証を行わずに衝突の回避を IaaS 管理者によるネットワーク設計に任せているため，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による仮想ネットワークの定義に対する検証をベースに置いたネットワーク仮想化技術 [6] を実現してきた．ここでは，OpenFlow ハイパーバイザ上で各テナントがネットワーク設計として定義したフロースペース間において重複部分を検証・管理し，コントローラが書き込むフローエントリに対して衝突検証と衝突を回避するための書き換えを行う手法について提案を行っている．この手法によって，マルチテナント環境において各テナント毎に自由なネットワーク設計とその独立性を保つような制御の両者を実現している．しかし，実際の運用を考慮すると各テナントのネットワーク空間を示すフロースペースの衝突検証だけでなく，仮想ネットワークを設計する上で必須な物理トポロジとの柔軟なマッピング手法も必要となる．また，OpenFlow においてパケットの処理内容を指定するアクションに対してもテナント間での分離性を保証する必要があるが，以前の研究ではこれについて明確に述べていなかった．

そこで，本研究では各テナントの任意の制御に対してテナントネットワーク間の通信の分離を実現する仮想トポロジの構築手法とフローエントリ中のアクション部分の書き換えについて提案を行う．具体的には Multi-Protocol Label Switching(MPLS)[3] のラベル情報を利用しながら，エッジスイッチに書き込んだ管理用フローエントリで制御を行い仮想トポロジを構築する手法について提案を行う．また，各テナントが設計したフロースペースと仮想トポロジ情報に基づいてフローエントリ中のアクション記述についても検証を行い，トラフィックの分離性を保証するのに必要となる制限事項について検討を行う．

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．

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し，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可能としている．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では変更内容をコントローラ上で記述すること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 チャンネルと呼ばれる 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出来る．加えて，OpenFlow を用いたネットワークの仮想化では VLAN-ID 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．

これらの利点が存在する一方で従来の OpenFlow 技術では，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を行う，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する，などの OpenFlow ネットワークそのものを

仮想化し制御する仕組みが実装されてい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. 既存研究

3.1 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 の利用形態としては，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した後に，その情報を何らかの形でそれぞれのテナントユーザに提示する．各テナントユーザは，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し FlowVisor に接続することでテナントネットワークが制御可能となる．

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問題が存在していた．また，この問題を発生させないために FlowVisor 管理者にネットワーク設計を委ねるという方針は，テナントユーザにとっての自由なネットワーク設計を制限しており，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るものだった．

3.2 フローエントリ検証に基づく仮想化

FlowVisor のネットワーク設計の制約を解決するために，各テナントに割り振られるフロースペースにおいて重複管理と衝突検証を行った上で，各テナントコントローラが書き込むフローエントリの衝突検証と書き換えを行うことでテナント毎の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手法を提案した [6]．図 1 に示したように提案した検証機構を FlowVisor に追加実装する．この研究は 3 つの手法から成り立っており，まず FlowVisor から変更して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の概念を導入している．次に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・管理する機構を導入している．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，テナント間でフローエントリが衝突してしまうことを可能な限り回避することが出来る．加えて，定義内でアドレス空間が重複しているフロースペースにおいてフローエントリが書き込まれる場合には，フローエントリの衝突検証と書き換えを行うことで各テナント間のトラフィックの分離性を保証する．このように，フロースペースに対する検証・管理を予め行っておくことで，フローエントリに対する書き換え処理を最低限に抑えることが出来る．

3.2.1 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，Src/Dst TCP ポートという L2〜L4 の 5 種類のヘッダ情報を指定することができる．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, Src TCP = 80
- VLAN ID = 101, Src IP = 192.168.64.0/20, Dst IP = 192.168.64.0/20, Src TCP = 80

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

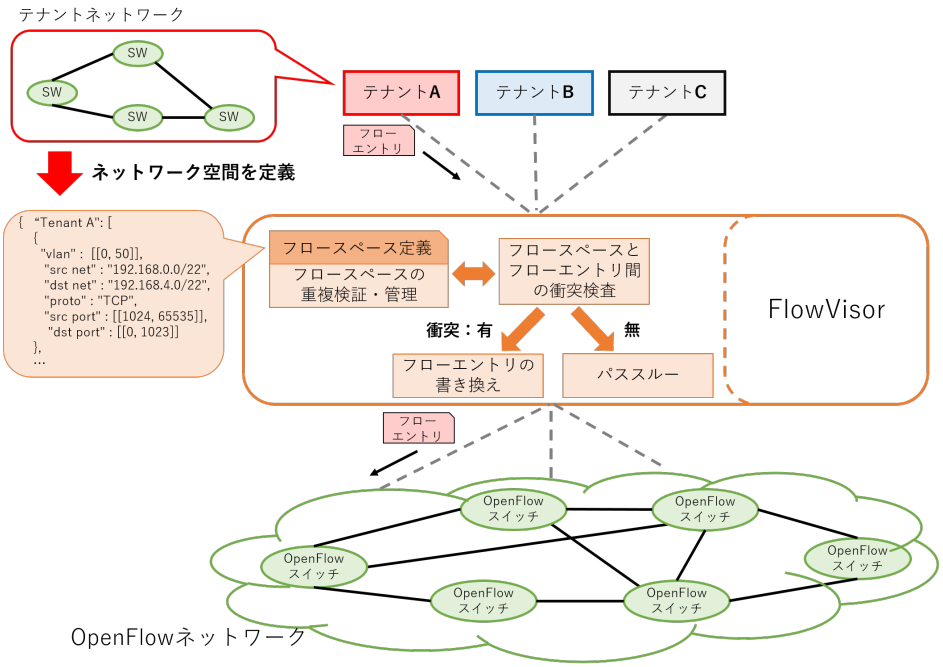

図 1　フローエントリ検証に基づく仮想化

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．

3.2.2　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, Src TCP = 80
- 定義例 2 : VLAN ID = 100, Src IP = 192.168.64.0/20, Dst IP = 192.168.64.0/20, Src TCP = 80

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，フローエントリの書き換えが必要となる．

3.2.3　フローエントリの書き換え

テナントネットワークのトポロジに重なりがある複数のテナントにおいて，前述のフロースペースの重複検証からフロー定義の重複が存在していると，トポロジの重なっている部分のスイッチでフローエントリの衝突が発生し得るため，テナントから書き込まれるフローエントリとパケットヘッダの書き換えを行うことで自動的な衝突部分の解消を行う．この書き換えではトポロジの重なり方によって 2 種類の処理を行う．

トポロジの一部分が重なっている場合には，利用されていないアドレス空間を用いた NAT による一時的な書き換えを行う．手順としては，重なっている部分の隣接スイッチに対しまだ利用されていない同じサイズのアドレス空間を用いてパケットのアドレスを変換する NAT のフローエントリを予め書き込む．その後に NAT フローエントリによって変換されているアドレスに合わせて，重複部分にテナントから書き込まれるフローエントリのマッチフィールドを書き換えてから転送を行う．

次に，トポロジの全体が重なっている場合には，3.2.1 節で述べた管理用アドレス空間の VLAN-ID を利用した書き換えを行う．まず，この VLAN-ID を新たに割り当て，もしくは現在使用している ID の変換を行うフローエントリを重なっている全スイッチに対して書き込む．続いて，テナントから書き込まれるフローエントリについてマッチフィールドの VLAN-ID 部分を書き換えてから転送を行う．

3.2.4　既存手法の課題

この研究ではフロースペースの競合管理手法を主眼にしているため，実運用のためには物理的な OpenFlow ネットワークを仮想化してテナントネットワークにマップする仮想トポロジの構築手法の実現が必要となる．この仮想トポロジの構築では新規のテナント追加時に単純なマッピングを行うだけでなく，物理ネットワークのリンク障害に伴う

表 1　フロースペースの上限と定義例

Rule ID	Space Name	VLAN	Src IP	Dst IP	Src TCP	Dst TCP
1	利用上限	0〜2047	0.0.0.0〜255.255.255.255	0.0.0.0〜255.255.255.255	0〜65535	0〜65535
2	定義例 1	0〜100	192.168.64.0/22	192.168.64.0/22	0〜1023	*
3	定義例 2	100, 101	192.168.64.0/20	192.168.64.0/20	80	*

再マッピングやテナントの制御下にないスイッチのトンネリングなどに柔軟に対応する必要がある．

また，3.2.3 節では各テナントの仮想トポロジの重なり方による 2 種類の書き換え手法として，空いているアドレス空間と管理用 VLAN-ID 空間を用いた書き換えを提案している．これに対して，実際の利用では各テナントの仮想マシンなどのホストと直接接続されているエッジスイッチやインターネットとの接続を行うゲートウェイのスイッチにおいて，テナント間の仮想トポロジが重なる可能性が高いと予測することができる．現在の手法ではフローエントリの書き換えに利用可能なアドレス空間の大きさが収容可能なテナント上限になるため，これらのスイッチ上で頻発するフローエントリの衝突を効率的に回避することでスケーラビリティを向上させることができる．

これらに加えて，既存研究のフロースペースの競合管理とフローエントリの書き換えという手法では，あるテナントが書き込んだフローエントリのマッチフィールドと他テナントのトラフィックがマッチしないことでそれぞれのテナントネットワークが分離されていると論じているが，フローエントリ中でパケットに対する処理を記述しているアクション部分については述べられていない．実際にはこのアクション部分で他テナントのトラフィックに影響を与えられないように管理を行う必要がある．例として，仮想トポロジ上で自身のテナントに割り当てられていない物理ポートを出力先に指定することや，ヘッダ情報の書き換えアクションで他テナントのアドレス空間を指定をした場合には，他テナントネットワーク上に自身のトラフィックが流れてしまう．

4. 提案手法

本研究では，テナント毎の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手法を実現するために，実運用上で必要なテナントの仮想トポロジの構築手法と各テナントが書き込むフローエントリ中のアクション部分の書き換え手法について提案する．本手法では，まず 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構築手法を導入する．この手法では，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うだけでなく，エッジスイッチの部分で頻発に発生が予測される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．加えて，各テナントが書き込むフローエントリに対して仮想トポロジの情報とフロースペースに基づき，アクション中で指定されているパラメータの書き換えを行う．また，以前のフロースペースの定義に対してもより実用的な設計となるように小規模な変更を加える．

4.1 フロースペースの定義変更

3.2 節で述べていたフロースペースでは，テナントユーザに対して実運用上で必要となるマッチフィールドとして VLAN ID，Src/Dst IP アドレス，Src/Dst TCP ポートという L2〜L4 の 5 種類のヘッダ情報の指定を求めていた．しかしながら，これらのマッチフィールドのうち Src/Dst TCP ポート番号については実際の利用方法を想定すると，テナントネットワークの設計時に自身が利用する TCP ポート番号の範囲を正確に予測することが難しい．このため，フロースペースの設計時にはワイルドカードを利用してすべてのポート番号を確保しておき，テナントネットワークの制御の中でフローエントリを用いて特定の TCP ポート番号のトラフィックを遮断する，もしくはトラフィックを許可するというような利用方法になると考えられる．こうした利用方法では TCP ポート番号をフロースペース上で設計させることの意義が薄くなっている．

これらのことから，実用性の観点から VLAN ID, Src/Dst IP アドレスをテナントが必ず設計を行うマッチフィールドとして，以前のフロースペースの定義に変更を加える．また，管理性の面からテナントはフロースペース上で記述される Src/Dst IP アドレスを最小を 27bit とするサブネット空間として設計を行う．一方で，設計されたフロースペースが複数のフロー定義に分解され，1 つのフロー定義が各フィールドの要素の AND として定義されること，複数のフロー定義がいずれかに合致するフローエントリが許可される OR として機能すること，というフロースペースの基本的な仕組みについては変更しない．

4.2 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．まず，各テナントは自身が利用したい仮想トポロジについて，図のようにテナントのホストが接続しているエッジスイッチから先のコア

ネットワークの接続関係のみを設計して本システムの管理者に要求する．続いて，そのテナントに割り振られているMPLS ラベルを用いて OpenFlow ハイパーバイザがフローエントリを書き込むことでテナント用の仮想トポロジを構築する．

前述のようにテナントが設計した仮想トポロジ情報は，実際の物理トポロジとのマッピング関係ではなくコアネットワーク部分の接続関係のみを記述している．このため，実際に物理トポロジのどの部分にマッピングして仮想トポロジを構築するかは本システムの管理者が決定する．この時，図 2 のようにテナント A の仮想トポロジを物理とマッピングする際には，本来存在しない SW-1 から SW-3 間を直接接続するリンクを仮想的に構築する必要がある．

これらのことから，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．

4.3 インストラクション/アクションの検証と書き換え

OpenFlow 1.3[4] ではパケットに対する処理として，指定した物理ポートからの送出，各ヘッダ情報の書き換え，VLAN ヘッダの付与/削除などの 17 種類のアクションが定義されている．コントローラ書き込むフローエントリ中では，これらから実行したいアクションの集合と複数のフローテーブル間の遷移処理をインストラクションとして記述することで柔軟な通信制御を実現している．この機構で利用しているパラメータについて以前の研究では明確に考慮していなかった．しかしながらトラフィックの分離性を保証するために，各テナントが書き込むフローエントリに対してマッチフィールドだけでなくアクションで利用している物理ポート番号やヘッダ情報についても検証を行い，フロースペース上の記述や仮想トポロジの情報に違反していた場合には書き換えを行う．

また，スイッチに書き込まれるフローエントリは各テナントのコントローラからのものだけでなく，提案する OpenFlow ハイパーバイザによるものも存在している．

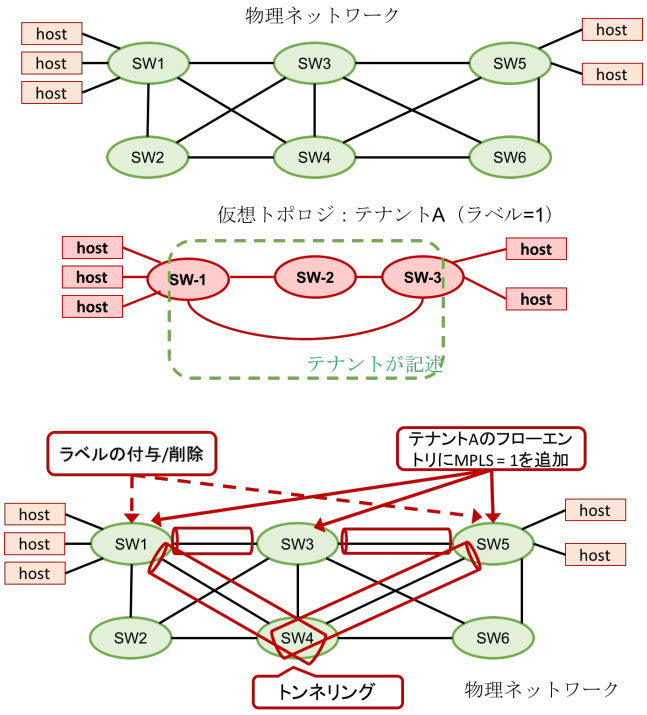

図 2　仮想トポロジの構築

OpenFlow ハイパーバイザによるフローエントリは，4.2 節のように衝突回避を行うためにテナントのフローエントリより前にマッチする，全ての処理の最後にマッチするなどの順番を考慮する必要がある．このため，必ずマッチさせる必要がある管理用フローエントリのためにいくつかのフローテーブルを管理用として確保する．その上で，テナントがフローエントリを書き込むフローテーブルについても検証を行うことで，管理用フローエントリに影響を及ぼすような記述を制限する．

4.3.1 アクションの検証と書き換え

OpenFlow 1.3 のアクションの中でも以下の 2 種類については，アクション中で指定されているパラメータを検証してフロースペースや仮想トポロジとして設計されている値を外れないように制限や書き換えを行う必要がある．

- OFPAT_OUTPUT：指定したポートからパケットを送出
- OFPAT_SET_FIELD：指定したヘッダ情報を書き換え

この中で OFPAT_SET_FIELD アクションについては，設計されたフロースペースのアドレス空間と仮想トポロジ用に割り当てられた MPLS ヘッダ以外を指定したフローエントリをテナントが書き込もうとした場合に，OpenFlow メッセージごと破棄してテナントのコントローラに通知を行う．

この一方で，OFPAT_OUTPUT アクションではスイッチの物理ポート番号以外にも表 3 に示した特殊な論理ポートがパラメータとして指定できるため，これらについて制限と書き換えを行う必要がある．表の論理ポートから ALL

表 2　仮想トポロジ用フローエントリ

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

表 3　特殊な論理ポート

Entry	Table ID
IN_PORT	パケットの入力ポート
ALL	入力以外の全てのポート
CONTROLLER	コントローラに明示的に送信

を指定した場合には入力以外の全ポートからフラッディングが行われるが，テナントがこのポートをアクションで指定した場合には，そのテナントが制御可能なポート番号の組み合わせに応じて 1 つのフローエントリを複数のフローエントリに展開してから書き込みを行う必要がある．例として，ある 8 ポートの物理スイッチ上でテナント A は 1,2,3 の物理ポートを利用可能な場合には，以下の 3 通りの入力/出力ポートの組み合わせが存在する．

- IN_Port = 1　Output: 2, 3
- IN_Port = 2　Output: 1, 3
- IN_Port = 3　Output: 1, 2

このため，元のアクションに ALL を指定したフローエントリを破棄する代わりに，上記の 3 パターン分のフローエントリに展開して書き込みを行う．また，仮想トポロジ上で制御可能な物理ポート以外が指定された場合には，再びフローエントリを書き込む OpenFlow メッセージごと破棄してテナントのコントローラに通知を行う．

4.3.2　管理用フローテーブル

OpenFlow 1.3 におけるフローテーブルの ID は 0 255 となっており，4.2 節で述べているような必ず最初にマッチするフローエントリや最後にマッチするフローエントリをシステム側で書き込む必要がある場合には，これらのために Table ID=0 と 255 のフローテーブルを管理用に予約しておく必要がある．このため，テナントが実際に利用可能な Table ID の値は 1 254 に制限する．テナントが書き込んだフローエントリにおいて Table ID=0 もしくは 255 の場合には，そのフローエントリを破棄する．

5.　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 OpenFlow ネットワーク仮想化技術の実現を目的として，実際の運用で必要となるテナントの仮想トポロジの定義・構築手法と各テナントが書き込むフローエントリについてアクション部分を含めたより詳細な検証手法を提案した．提案手法では，物理トポロジ上にテナントネットワークを柔軟にマッピングできるだけでなく，エッジスイッチの部分で頻発に発生が予測されるフローエントリの衝突を効率的に回避することが可能となっている．また，実際にテナントが書き込むフローエントリについても，トラフィックの分離性を保証するために必要となる検証事項や要素についてより詳細に検討がされている．

本論文中で提案した手法では，OpenFlow が標準的に制御可能な MPLS を利用することでテナント間のフローエントリ衝突を回避しながら仮想トポロジを構築している．しかしながら，この提案手法のために各テナントでは OpenFlow の機能として MPLS の制御を利用することができない設計となっている．これは自由なネットワーク設計が可能な仮想化技術という考えと異なっているように見えるが，本来想定しているのは IaaS 提供のためにマルチテナント環境が必要となるデータセンタネットワークについてであり，大規模通信事業向けに拠点間の広域イーサネット構築を目的として利用されている MPLS 技術をテナントネットワーク中で利用したいという要望は大きくないと考えているからである．MPLS に代わって仮想トポロジの構築に利用できる技術としては，従来の VLAN を拡張した Virtual eXtensible Local Area Network(VXLAN) というパケットのカプセリング技術が存在するが，現在の OpenFlow の仕様では VXLAN パケットのカプセル/復号化ができないため，実際の利用では OpenFlow スイッチ上でこれらを可能にする拡張が必要となっている．

6.　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を用いたネットワーク仮想化技術をベースとして，実際の運用上必要となる物理トポロジと仮想ネットワークの柔軟なマッピング手法とパケットに対する処理内容を含めたフローエントリの検証手法について提案した．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，各テナントの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仮想トポロジを構築できることに加えて，OpenFlow のアクションも含めてテナント間でのトラフィック分離性をより明確に保証することが可能となっている．

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] McKeown, Nick. "Software-defined networking." INFOCOM keynote talk 17.2 (2009): 30-32.
[2] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Turner, J.: *OpenFlow: enabling innovation in campus networks* ACM SIGCOMM Computer Communication Review Volume 38 Issue 2, (April 2008): 69-74.
[3] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS Label Stack Encoding." Internet Engineering Task Force, RFC 3032, January 2001.
[4] "OpenFlow Switch Specification Version 1.3.4 Implemented (Protocol 0x04)" March 27, 2014.
[5] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G.: *FlowVisor: A Network Virtualization Layer*, Tech. Rep. OPENFLOW-TR-2009-01, OpenFlow Consortium, (October 2009)
[6] 樋口俊, 廣津 登志夫. "マルチテナント向け OpenFlow ハイパーバイザのためのフローエントリ検証手法の検討" , コンピュータシステム・シンポジウム論文集. 2016: 129-136.