# マルチテナントデータセンタのための OpenFlow検証手法

牛, 躍川 / Niu, Yuechuan

(出版者 / Publisher)
法政大学大学院情報科学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 情報科学研究科編 / 法政大学大学院紀要. 情報科学研究科編
(巻 / Volume)
11
(開始ページ / Start Page)
1
(終了ページ / End Page)
6
(発行年 / Year)
2016-03-24
(URL)
https://doi.org/10.15002/00012884

# マルチテナントデータセンタのための OpenFlow 検証手法
# A OpenFlow Verification Method for Multi-Tenant Datacenter Network

牛　躍川
Yuechuan Niu
法政大学情報科学研究科情報科学専攻
*E-mail: yuechuan.niu.u6@stu.hosei.ac.jp*

## Abstract

*OpenFlow is the state of the art technology for efficient network operation. Since OpenFlow enables central management of entire network, it is expected to widely use in a multi-tenant data center with its flexible routing control. In multi-tenant data center, a same physical resources will be shared with multiple tenants. It is significantly important to validate the configuration of the network flow control to satisfy both reachability within a tenant network and isolation among networks of different tenants. However, each tenant assigns arbitrary IP addresses to its owned networks, then it may conflicts each other. VLAN (Virtual LAN) technology is used to solve these conflicts, but it has limitation where only up to 4094 VLAN-ID can be used. It is difficult to assign a different VLAN-ID for each tenant's network segment. OpenFlow technology will be useful to dynamically assign and swap VLAN-IDs to keep the separation between tenants' network segments. In this paper, we propose a method for automatically verifying reachability within an OpenFlow virtual network using static analysis of flow table, and a method for automatically generating test cases to verify the correctness of the VLAN configuration. We also show examples of applying the proposed method in a typical data center network.*

## 1. はじめに

サーバの仮想化やクラウドの急速な進化の中で，1 つのデータセンタの中に複数の組織のサーバネットワークが重畳するマルチテナント型のデータセンタが一般的になってきた．マルチテナント型データセンタでは論理的に分割された複数のテナントネットワークが集約されており，それぞれのテナントネットワークは互いに通信ができないように分離されている必要がある．このためには，VLAN 技術を用いて各テナントの使うネットワークを異なる論理ネットワークとすることが一般的であった．しかし，Ethernet の VLAN には VLAN-ID のビット幅により最大 4094 個の VLAN しか構築できないという制約が存在する．そのため，各テナントの仮想ノードの配置やネットワークの構成には制約が大きかった．

この問題の解決のために VXLAN(Virtual Extensible LAN)[4]や Nested VLAN，TRLL(Transparent Interconnect a Lot of Links)[5] といったデータリンクレイヤの拡張技術が提案されている．しかし，これらの技術は専用のハードウェアを必要とし，機器に互換性がないため移行や導入が困難であった．そこで，最近注目されている SDN(Software-Defined Network)[1] 技術を適用することが注目されている．SDN では従来 1 つの機器筐体内に共存していたコントロールプレーンとデータプレーンを別の筐体に分離し，エッジノード上のソフトウェアの挙動などに適応してソフトウェア的にネットワーク全体の挙動を制御する．これを実現する代表的な技術が OpenFlow[2] である．OpenFlow ではパケットヘッダの組み合わせに対して処理を指示するフローエントリを各 OpenFlow スイッチに設定することでこまめな経路の制御が可能である．Ethernet VLAN についてはタグの認識だけでなく書き換え機能も持つため，アドレス空間に衝突のあるテナント間だけ異なる VLAN タグを付与するといった柔軟な制御が可能になる．

一方で，クラウドの利用が進むにつれて，従来は組織内で保有していた Firewall や IDS/IPS，VPN ゲートウェイなどといった専用ネットワーク機器により提供されていた機能も全てクラウド上でソフトウェア的に実現する NFV(Network Functions Virtualization)[3] の技術が広く使われるようになってきている．NFV ではネットワークで必要とされる機能を必要に応じてネットワーク上に展開し，その機能間を Chaning により接続することで，従来の組織内ネットワークに配置されていたネットワーク機能をクラウド上に移行することが可能になる．これにより，マルチテナント型データセンタ上には，テナント毎に複数のネットワークが仮想ノードや NFV の各機能ノードが配置されたサーバ間に複雑に配置されることになる．そのため，テナント間分離のために各 OpenFlow スイッチに設定されるフローエントリは複雑になり，テナント間での競合の検証が難しくなる．

そこで，本研究では OpenFlow ネットワーク上において構築されたマルチテナントネットワークの間の独立性を自動的に検証する枠組みの構築を目指す．ここでは，LLDP プロトコルを用いたトポロジ検出とフローテーブルの静的解析によってノード間の到達性を検証する手法と，その到達性検証エンジンの入力となるテストケースを自動的に生成する手法を提案し，提案手法を実 IP アドレスを付与した実験ネットワークに対して適用した際の検証例を示す．

---

Supervisor: Prof. Toshio Hirotsu

## 2. マルチテナント型データセンタ

従来のデータセンタサービスでは，ユーザ毎に物理サーバを貸し出し，その上に個別に構築されたシステムを運用していた．それがサーバコンピュータの価格性能比が飛躍的に向上したことによりデータセンタ使用コストが高まり，データセンタに設置されるサーバの効率化が求められるようになった．サーバやストレージなどの物理資源を有効活用するために，1 台の物理サーバ上に複数の仮想サーバを集約し，CPU，メモリ，I/O リソースを複数の仮想マシンで共有するサーバ仮想化が積極的に利用されるようになった．マルチテナントとは，このサーバ仮想化技術を用いて 1 つの物理サーバ上に複数の独立した環境を構築し，同一の物理リソースを複数のユーザ(テナント)で共有するための技術である．このような場合には，顧客ごとに提供するリソースは同じ物理ノード上に配置するのではなく，実際には冗長化などのために複数のノードに配置することになる．また，顧客の要望に即時に対応し，必要に応じてリソースを必要な分だけ確保することと，設備コスト削減のためにサーバの集約度を上げることを考えると，同一テナント内でも物理的には離れた異なるノード間にまたがった通信をすることが一般的である．さらに，テナント毎に複数のネットワークセグメントを使用する場合に加えて，NFV により VPN ゲートウェイの機能を実装した場合にはデータセンタ外部から接続された社内 LAN と直結しているように見えることもある．このような場合，同じ物理ネットワーク上で複数のテナントネットワークの通信が流れることになるが，テナント同士には論理的に相関関係はなく，それぞれのテナントはお互いに排他独立したシステムでなければならない．そのため，基盤としてのネットワークは多様な要求に応える柔軟性と十分な重畳度，そして確実なテナント間分離が要求される．

### 2.1 Ethernet VLAN による制御

テナントネットワークの分離を実現する古典的な方式としてテナントごとに別々の IP アドレスブロックを割り当てて，IP レベルでのブロードキャストセグメントを分割する方法がある．しかし，異なるテナントで同じプライベートアドレスを使う場合には，IP アドレスの重複が発生してしまうため対応できないという問題がある．テナント間における IP アドレスブロックの重複という課題に対して，従来では IEEE802.1Q VLAN を用いて VLAN-ID による分離を行う手法が用いられてきた．これにより各 VLAN グループはデータリンク層レベルで分離されているため，レイヤ 3 の IP アドレスに同じ空間を使用していたとしてもテナント間で影響することなく通信をすることができる．しかし，VLAN-ID フィールドは 12bit であり，理論上は最大で 4094 個のセグメントまでしか分離することはできない．それに対し，ハードウェアの性能向上及び仮想化技術の発展によって 1 つの物理サーバの中で稼働する仮想マシンの数が増大し，さらにユーザごとに複数の VLAN-ID が要求されることもある．したがって，このような状況では VLAN-ID が枯渇してしまうという問題がある．

### 2.2 VLAN 技術の拡張

上記のテナント分離数の上限という問題を解決する代表的な技術として，VXLAN や TRILL などが開発されている．VXLAN は，Cisco や VMWare 等が中心となって開発された規格であり，既存の IP レイヤの上に仮想的なレイヤ 2 ネットワークをオーバーレイすることでレイヤ 2 セグメントの拡張や IEEE802.1Q の VLAN-ID 上限問題の解決などを実現している．VXLAN では，VXLAN トンネリングのエンドポイントを VTEP(Virtual Tunnel End Point)と呼んでおり，ハイパーバイザの仮想スイッチや物理サーバ単位でこの VTEP を設置してレイヤ 2 の通信をレイヤ 3 でトンネリングする．仮想マシンから送出されるパケットは VTEP でカプセル化され，8 バイトの VXLAN ヘッダが付加されてネットワーク上に転送される．それを対向する VTEP が VXLAN ヘッダを取り外してから宛先仮想マシンへと転送する．VXLAN ヘッダは，1bit の VXLAN フラグと，24bit の VNI(VXLAN Network Identifier), そして 39bit の未使用フィールドによって構成されている．このうちの VNI は IEEE801.Q の VLAN-ID と同じ意味を持ち，同一 ID を持つ仮想マシン同士だけが直接通信をすることができる．この 24bit の VNI によってネットワークを最大約 1600 万のネットワークセグメントに分離することが可能となるため，VLAN-ID 数が不足している問題を解決することができる．

TRILL は，IETF で RFC6325 として標準化されている Ethernet で経路を冗長化する技術である．TRILL におけるマルチホップルーティングは，MAC アドレスヘッダと TRILL ヘッダを仮想マシンから送出された Ethernet フレームに付加してカプセル化することによってレイヤ 2 レベルのルーティングを実現している．スイッチは仮想マシンから受け取った Ethernet フレームに，VLAN タグにも対応した MAC アドレスヘッダを付加してカプセル化する．これには，TRILL のルーティングテーブルに基づいてネクストホップとするスイッチを送信先として指定し，スイッチをホップするたびに送信元と送信先の MAC アドレスが書き換えられる．さらに，Ethernet フレームには TRILL ヘッダが付加され，送信元のスイッチと最終的な宛先スイッチがそれぞれ RBridge(Routing Bridge)として指定される．各 RBridge は 6~7 バイトの IS-IS ID を持ち，この ID を TRILL のネットワークにおける識別子として用いる．これによって VLAN-ID のスケーラビリティの問題を解決している．

これらの技術を用いることによりテナントを識別するための識別子の上限を増やし，ネットワークセグメント分離数の上限問題を解決することが可能となるが，いずれもその技術に対応した専用機器を使用しなければならず，互換性がないために移行，導入コストがかかってしまうということが課題となっている．

## 3. OpenFlow

SDN は，従来個々のネットワーク機器が一台ずつで行ってきたネットワーク制御とデータ転送処理を分離し，ネットワーク制御機能をサーバ側のソフトウェアで一元

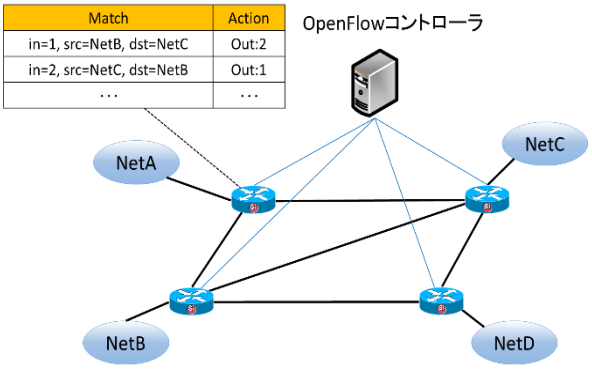


図 1．OpenFlow コントローラと OpenFlow スイッチ

管理することによってネットワーク全体を動的に制御することを可能とするネットワークアーキテクチャである．

OpenFlow は SDN の概念を実現する代表的な技術の 1 つであり，ONF(Open Networking Foundation)が標準化を行う集中制御型のネットワーク制御技術である．OpenFlow は図 1 のようにパケット転送処理のみを行う OpenFlow スイッチと，トポロジ管理やルーティングなどの経路制御機能を担う OpenFlow コントローラによって構成される．コントローラとスイッチ間の通信は OpenFlow プロトコルで定義しており，OpenFlow コントローラは複数の OpenFlow スイッチを一元管理し，経路計算や受信したパケットの振る舞いの指示などを行う．従来の TCP/IP ネットワークにおけるフォワーディングはデータリンク，ネットワーク，トランスポートなど各プロトコルレイヤ別に独立してフォワーディングルールが適用されていたが，OpenFlow では複数のレイヤやプロトコルを組合せたマッチング条件と，マッチ時に実施すべき処理の定義をまとめたフローエントリに従ってパケットを処理する．フローエントリには，パケットを受信した際にフローテーブルを検索するためのキーであるマッチフィールド，マッチフィールドに合致した場合の処理方法を指定するインストラクション，同じフローエントリに一致したパケットの情報を記録しているカウンタ，マッチフィールドに合致した際にフローエントリが複数存在した場合の優先順位であるプライオリティ，フローエントリが消滅するまでの時間であるタイムアウト，コントローラがフローエントリを管理する際に使える任意の 64 ビットの整数であるクッキーの 6 つの情報が含まれており，これらの情報を組み合わせて受信したパケットの処理方法を決定する．OpenFlow スイッチはパケットを受信した際に，ヘッダ情報を確認してどのフローエントリのマッチフィールドに一致するかを判断し，一致した場合にそのフローエントリのインストラクションが指定する処理を実行する．

## 4. マルチテナント間の独立性検査

前述のように，マルチテナント型のデータセンタネットワークにおいては，1 つの物理ネットワーク上に複数の論理ネットワークが構築されており，物理的な接続に対してそれぞれの論理ネットワークの通信設定が正しくなされていることと，論理ネットワーク間の通信が正しく分離されていることを検証しなければならない．またリンクダウンなどの物理故障が起こった際には，物理故障個所の特定とそれに対する論理ネットワークの影響を調べることが必要となる．文献[6]では，マルチテナント型のデータセンタネットワークにおけるネットワーク機器設定の事前検証を自動化する手法が提案されている．機器設定の適用前と適用後における設定項目間の依存関係を検証ルールに基づいてグラフ形式で抽象化し，それによってテナント間および各テナント内のノードの接続ミスがないことの検証を行っている．この研究では，グラフ理論を用いてネットワーク全体を抽象化し，グラフ構造で表現されたネットワーク上のノードが保持しているルーティングエントリの宛先 IP アドレスとそれに対応するネクストホップを辿ることで，目的ネットワークへの到達性を検証している．この手法では，IP 層であるレイヤ 3 の中継関係を，宛先アドレスをみて追跡することで分離性の検証を行っていたが，OpenFlow においてはアドレスだけでなくレイヤ 1 の情報にあたる物理ポート番号も含めた，レイヤ 3 からレイヤ 4 までの情報を統合した検証を行わなければならない．またエントリが複数マッチする場合やそのときの各エントリの優先度も考慮することが必要となるため，単純な追跡のみでは十分な検証を行うことができない．分離性の検証は，フローエントリの追跡からなる到達性の検証技術を確立した上で，データセンタ内のスイッチの全てのフローエントリから，検査の対象となるアドレス等の通信仕様のテストケースを生成しなければならない．以下，この節に到達性検証技術を，5 節にテストケース抽出技術を説明する．

### 4.1 物理ネットワークのトポロジ管理

物理ネットワークトポロジの検出は，OpenFlow で一般的に使われる LLDP(link layer discovery protocol)を用いて行う．LLDP では，接続関係を調べたいスイッチのスイッチ ID と物理ポート番号の情報を埋め込んだ LLDP パケットをコントローラから送出し，隣接するスイッチからコントローラへと返ってくる LLDP パケットと Packet-In メッセージによってスイッチ間のリンクを検出する．これを，コントローラと接続する全てのスイッチに対して繰り返すことによって，ネットワーク全体のトポロジを検出する．これによって取得したトポロジ情報は，ネットワーク上のノードとリンクを頂点と辺として抽象化してグラフ構造で表現し，管理する．ここで用いるグラフは$G$=($V$, $E$)として定義し，$V$=($v$1, …, $v$N)はグラフ中の頂点の集合であり，$E$=($e$1, …, $e$M)は辺の集合である．グラフ化したネットワークの接続関係に基づいて接続行列を生成し，これを辿ることで各ノード間の接続性を確認できるようにする．接続行列上でのポート番号の管理においては，列のノードの接続ポート番号を行列で保持するようにしている． また，OpenFlow の制御においては，ノードへ接続されている物理ポート番号の情報が必要となるため，辺$e$に双方の接続ポート番号の情報を付与する．

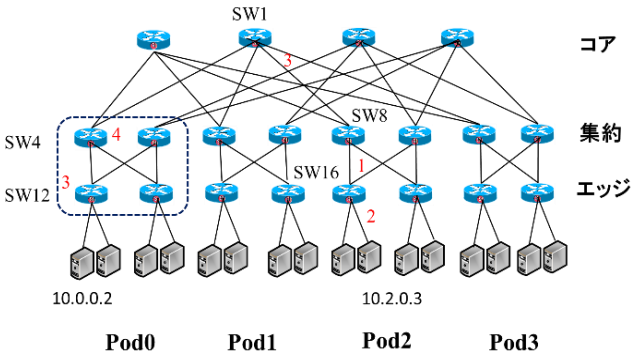


図 2. 到達性検証に用いる Fat-tree トポロジ

### 4.2 フローの検証

OpenFlow ネットワークでは，フローの経路制御に関する情報はコントローラがフローエントリをスイッチに追加することによって設定される．そのためコントローラから各スイッチに設定されるフローエントリ群を静的に解析することによって，フローの経路を確認することができる．まずコントローラが持つエントリ情報を集約して 1 つのテーブルにし，物理ネットワークのトポロジを元に各エントリの出力ポートにつながっているスイッチを取得する．マッチ条件に対する転送先スイッチとの接続関係を調べ，確認対象区間で探査することによって設定したフローの経路と到達性を確認することができる．OpenFlow ではフローエントリの Priority フィールドに優先度を設定することができ，複数のフローエントリにマッチした際は優先度の最も高いフローエントリを選択して処理を行う．したがってフローテーブルを探索する際には，マッチしたエントリのうち最もプライオリティが高いエントリを参照させるようにする．マッチするエントリが複数あるにも関わらず優先度の指定がなく，重複するものがあった場合は到達性がないとしてエラーを返す．また，経路を探査する際のヘッダフィールド情報は更新可能なものとして保持しておき，マッチしたエントリのアクションフィールドに応じて確認条件を更新する．これによって，Modify-Field アクションでパケットヘッダの書き換えや VLAN タグの挿入などの処理がある場合でも正しく転送経路を辿ることができる．

### 4.3. Fat-tree における到達性検証例

ここまでに述べた提案手法を用いて Fat-tree トポロジを用いた到達性検証の例を示す．まず，トポロジの設定と投入されるフローエントリについて述べ，更に検証例を示す．

#### 4.3.1 ネットワークトポロジ

本節では，ノード数 16，スイッチ数 20 で構成された Fat-tree トポロジ(図 2)を対象として到達性検証を行う際の例を示す．ここでは，各サーバに 10.0.0.0/8 のプライベート IP アドレスを以下のように割り当てる．Pod に 0 から 3 の Pod 番号 p，Pod 内のスイッチに同様に 0 から 3 のスイッチ番号 s を割り当て，エッジスイッチごとに左側のノードを 2，右側のノードを 3 とノード番号 n を割り当てたとき，10.p.s.n をサーバの IP アドレスとして割り当てる．また，スイッチにはコア層の左から順番に 0 から 19 までのスイッチ ID をつける．

表 1. フローエントリ

| SwitchID | Match Field | Action |
|---|---|---|
| 12 | ethType=0x0806, 0x0800<br>dstIP=10.2.0.3 | output:3 |
| 4 | ethType=0x0806, 0x0800<br>dstIP=10.2.0.3 | output:4 |
| 1 | ethType=0x0806, 0x0800<br>dstIP=10.2.0.3 | output:3 |
| 8 | ethType=0x0806, 0x0800<br>dstIP=10.2.0.3 | output:1 |
| 16 | ethType=0x0806, 0x0800<br>dstIP=10.2.0.3 | output:2 |

表 2. 接続行列

| | SW1 | SW4 | SW8 | SW12 | SW16 |
|---|---|---|---|---|---|
| SW1 | × | 1 | 3 | 0 | 0 |
| SW4 | 4 | × | 0 | 1 | 0 |
| SW8 | 4 | 0 | × | 0 | 1 |
| SW12 | 0 | 3 | 0 | × | 0 |
| SW16 | 0 | 0 | 1 | 0 | × |

#### 4.3.2 フローエントリの設定

ここでは，通信制御の例として以下のようなルーティングを想定したフローエントリを設定する．このネットワークでは，宛先 IP アドレスに従ってルーティングを行うことによってマルチパスの活用を行う．上向きのルーティングにおいて，エッジ層のスイッチから集約層のスイッチにパケットを送出する際には，宛先 IP アドレスの 3 オクテッド目によって転送先のスイッチを決定し，集約層のスイッチからコア層のスイッチへの送出においては，宛先 IP アドレスの 4 オクテッド目によって転送先を決定する．

#### 4.3.3 検証例

例として， 10.0.0.2 の IP アドレスを持つサーバから 10.2.0.3 の IP アドレスを持つサーバへのフローを検証する際の手順を以下で説明する．最初に 10.0.0.2 のサーバに接続されているスイッチ 12 のフローエントリ(表 1)を参照し，宛先 IP アドレス 10.2.0.3 へのパケットは 3 番ポートに出力するという情報を得る．次に，物理ネットワークの接続行列(表 2)でスイッチ 12 に関する接続関係を参照することによって 3 番ポートの先にスイッチ 4 が接続されていることがわかる．さらにスイッチ 4 のフローエントリに関しても同じように参照し，スイッチ 4 の出力ポート 4 にスイッチ 1 が接続されているという情報を

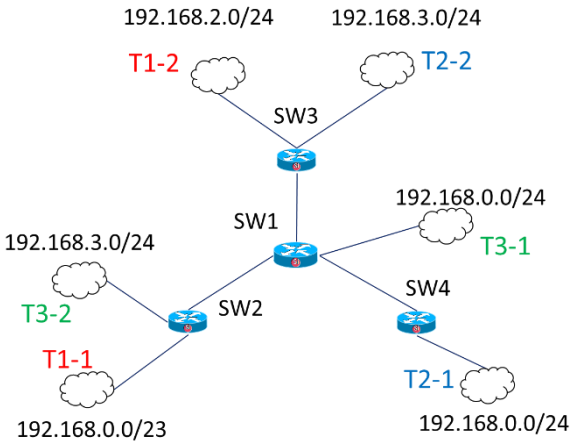


図 3. マルチテナントネットワークの例

表 3. スイッチ 1 のフローエントリ

| サブネット | Match Field | Action |
|---|---|---|
| T1-1 | srcIP=192.168.0.0/23<br>dstIP=192.168.2.0/24 | output:2 |
| T1-2 | srcIP=192.168.2.0/24<br>dstIP=192.168.0.0/23 | output:1 |
| T2-1 | srcIP=192.168.0.0/24<br>dstIP=192.168.3.0/24 | output:2 |
| T2-2 | srcIP=192.168.3.0/24<br>dstIP=192.168.0.0/24 | output:4 |
| T3-1 | srcIP=192.168.0.0/24<br>dstIP=192.168.3.0/24 | output:1 |
| T3-2 | srcIP=192.168.3.0/24<br>dstIP=192.168.0.0/24 | output:3 |

得る．このような接続行列とフローエントリを参照する手順を宛先のノードが接続されているエッジスイッチが見つかるまで繰り返す．

## 5. テストケースの自動生成

VLAN 間の分離の検証のためには，それぞれのネットワークのフローが経路上で混ざることなく，通信が正しく分離できることが保証されていなければならない．しかし，大規模なデータセンタネットワークにおいて全テナントネットワークのアドレス空間の重なりや複数レイヤにまたがったマッチフィールドの衝突を検証することは，データセンタ管理者にとって大きな負担となる．そのためこれらのことを確認するためのテストケースを自動的に生成する方法について説明する．本論文で取り上げるテストケースとは，前述の検証エンジンの入力となる送信元と送信先 IP アドレスの組合せであるテストパターンの集合のことであり，本来ならばフローエントリに出現するアドレス空間外のものでも他のルールに衝突するものが存在する可能性があるため，全アドレスの組合せを検査する必要がある．しかし，そのような莫大なテストケースでは検証に時間がかかりすぎてしまい，実用的な時間に収めることができない．そのため，本研究では各アドレス空間の代表値を抽出することによってテストケースの縮小を目指す．

### 5.1 テストパターンの抽出

全てのフローエントリから得られる送信元アドレス空間，宛先アドレス空間のそれぞれ毎にアドレス空間を代表するアドレスを決めて，それらを保有するスイッチポート間で総当たりして検査する．本節では，その代表値を抽出する方法について説明する．

テナント間の分離の確認のためには，同一の論理ネットワークの到達性の確認と，異なる論理ネットワーク間での独立の両方を確認することが必要である．一方で，フローエントリに出現しないアドレスについては，マッチ条件に合致せずそもそも転送されないことから，出現する送信元と送信先のアドレス空間につき 1 つずつ代表アドレスを決めれば良い．但し，各テナントがそれぞれ自由にプライベートアドレスを決め，テナント間で使用しているアドレス空間が重なっている場合はフローの競合が発生し得るため，その重なり方が異なる全てのアドレスアドレス空間のそれぞれに代表アドレスを決定する必要がある．そのため，2 つのフローエントリの送信元アドレス空間を$A$，$B$としたときの送信元の代表アドレスは以下のような条件で選ぶ．

i. $A$が$B$に包含される場合
   $A \cap B$の空間から 1 つ
   $\bar{A} \cap B$の空間から 1 つ
ii. $A$と$B$の一部が重なる場合
   $A \cap B$の空間から 1 つ
   $A \cap \bar{B}$の空間から 1 つ
   $\bar{A} \cap B$の空間から 1 つ
iii. $A$と$B$が重ならない場合
   $A$と$B$の空間から 1 つずつ

送信先アドレスも送信元アドレスのときと同様にして決定し，スイッチ中の各ポートについて，入力ポートに代表アドレスを含むアドレス空間があればそれを全て割り当て，割り当てられた代表アドレス間の通信を総当たり的に検査する．テストパターンのアサーションとして，送信元と送信先 IP アドレスが両方とも同一論理ネットワークのアドレス空間内であるものは到達性のあるべきものとして TRUE とし，それ以外のものは到達性があってはならないものであるため FALSE とする．これによって論理ネットワーク間でフローの競合による影響があるものとないものを明らかにすることができ，影響の出ないもの同士に同じ VLAN-ID を割り当てるなどといった最適な VLAN 設定が可能となる．

### 5.2 テストケース生成の例

ここでは図 3 に示すマルチテナントネットワークにおいて，テストケースを生成する例を示す．例に用いるネットワークとして， 4 つの OpenFlow スイッチによって 6 つのサブネットが接続された物理ネットワークにおいて，T1，T2，T3 の 3 つのテナントネットワークが混在している環境を想定する．テナントネットワークはそれぞれ，

表 4. 送信元アドレスの代表値の例

| テスト SW | アドレス空間 | 代表アドレス |
|---|---|---|
| 1, 2, 4 | T1-1 ∩ T2-1 | 192.168.0.1 |
| 2 | T1-1 ∩ $\overline{\text{T2-1}}$ | 192.168.1.255 |
| 1, 2, 4 | T1-1 ∩ T3-1 | 192.168.0.1 |
| 2 | T1-1 ∩ $\overline{\text{T3-1}}$ | 192.168.1.255 |
| 1,2,4 | T2-1 ∩ T3-1 | 192.168.0.1 |
| 2, 3 | T2-2 ∩ T3-2 | 192.168.3.1 |
| 3 | T1-2 | 192.168.2.1 |

表 5. テストパターンの例

| Tenant | SW | SrcIP | DstIP | Assertion |
|---|---|---|---|---|
| T1 | 2 | 192.168.0.1 | 192.168.2.1 | TRUE |
| T1 | 2 | 192.168.1.255 | 192.168.3.1 | FALSE |
| T2 | 4 | 192.168.0.1 | 192.168.3.1 | TRUE |
| T2 | 4 | 192.168.0.1 | 192.168.2.255 | FALSE |

T1 が 192.168.0/0/23 と 192.168.2.0/24，T2 が 192.168.0/0/23 と 192.168.3.0/24，そして T3 も 192.168.0/0/23 と 192.168.3.0/24 という 2 つのサブネットを持っている．それぞれのテナント内のサブネット同士は互いに通信できるとして，3 つのテナントネットワークは全て送信元 IP アドレスと宛先 IP アドレスに基づいた転送ルールでパケットを転送する．各テナントネットワークの転送ルールはユーザの入力によって与えられるものとし，このときのスイッチ1におけるフローエントリの例を表3に示す．

まず送信元アドレスは，各スイッチのフローエントリに出現する送信元アドレスブロックを取得し，各アドレスブロックの重なりを見ながら代表値を選び出す．例として T1-1 と T2-1 のアドレス空間を見ると，T2-1 は T1-1 の空間に包含されており，192.168.0.1~192.168.0.255 までのアドレス空間が重なっている．そのような場合は，アドレス空間が重なっている 192.168.0.1~192.168.0.255 の中から 1 つと，重なっていない 192.168.1.1~192.168.1.255 の中から 1 つの計 2 つの代表アドレスを選出する．次に，T2-1 と T3-1 の場合を考えると，T2-1 と T3-1 は 192.168.0.1~192.168.0.255 で完全にアドレス空間が重なっていることがわかる．このような場合は，2 ブロック間で境界が存在しないため，双方共通のアドレス空間である 192.168.0.1~192.168.0.255 の中から 1 つだけ選ぶ．また，T1-2 の場合は，192.168.2.1~192.168.2.255 でいずれのアドレスブロックとも重なりがない．そのため，この場合も空間内からアドレスを 1 つだけ選ぶ．このようにして選びだした代表値はまとめて管理しておき，各スイッチは接続ポートの先にこれらの代表値が含まれるネットワークセグメントがあるか探す．そして，いずれかの接続ネットワークに含まれるようなアドレスを，そのスイッチから検査を開始する検査データの送信元アドレスとする．送信先アドレスに関しても同様にフローエントリの送信先アドレスブロックから代表アドレスを選び，今度はこれら代表アドレスの全てを送信先アドレスとする．こうして得られた送信元と送信先アドレス候補を総当たり的に組み合わせて，テストケースを生成する．このとき生成されたテストケースの一部を表 5 に示す．生成されたテストケースに従って検証エンジンによる到達性検証を行うと，T1 は T2 と T3 のフローと送信先 IP アドレスのアドレス空間が重なっていないため，フローが競合することなくいずれのテストパターンでも正常な通信ができることがわかる．一方で，T2 と T3 は送信元と送信先 IP アドレスが共に重なってしまっているため，T2-1 ネットワークから送信元 IP アドレスが 192.168.0.1 で送信先 IP アドレスが 192.168.3.1 のようなパケットを送出した場合，経路が重なるスイッチ 1 において T3-1 のフローエントリと競合してしまうことがわかる．これにより，T1 は T2 と T3 のどちらでも同じ VLAN-ID を割り当てることができるが，T2 と T3 は異なる VLAN-ID を用いて明確にフローを区別しなくてはならない結果を得ることができる．

## 6. まとめ

本研究では，OpenFlow スイッチによってマルチテナントネットワークを構築した際の論理ネットワークの通信の正常性を自動的に検証するための枠組みを提案した．本稿で提案した LLDP プロトコルを用いた動的トポロジ検出とフローテーブルを静的解析によるフロー到達性検証手法は，実環境のネットワークに設定を変更することなく事前検証できることが利点となっている．また，全フローエントリに設定されている送信元と送信先のアドレスブロックから，アドレス空間の重なり方に応じてそれぞれのネットワークセグメントの代表値を抽出することによって，最小限のテストデータで十分な検証をすることができることを実際のマルチテナントネットワークを例として示した．

今後の課題として，本研究では IP アドレスベースの転送ルールを対象とした手法について提案したが，OpenFlow はレイヤ 1 からレイヤ 4 に相当するフィールドを組み合わせることができるため，より複雑なルールが設定される場合の検証へ発展させることが必要である．

## 文　献


[1] OPEN NETWORKING FOUNDATION "Software-Defined Networking: The New Norm for Networks White Paper" April 13, 2012.

[2] NFV White Paper, http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow”: Enabling Innovation in Campus Networks” ACM SIGCOMM Computer Communication Review, vol. 38, no 2. p. 69, 2008.

[4] IETF “Virtual eXtensible Local Area Network: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks” August 2014, RFC7348.

[5] IETF “Routing Bridges: Base Protocol Specification” July 2011, RFC6325.

[6] Yosuke Himura, and Yoshiko Yasuda, “Static Validation of Network Device Configurations in Virtualized Multi-tenant Datacenters” IFIP/IEEEIM 2013, May 2013.