実ネットワークの再現を可能とする OpenFlow を用いたネットワーク運用管理支援システムの開発と評価方法の検討

堤啓彰[†1]　井口信和[†2]

クラウド環境やサーバ仮想化の普及により，SDN が注目を集めている．SDN ではコントロールプレーンとデータプレーンが分離されることにより，論理トポロジや経路制御の動的な変更が可能である．一方で，ネットワークの状況を正確に把握することが困難である．そこで本研究では，SDN の中核技術である OpenFlow を用いたネットワーク運用管理支援システムを開発した．本システムでは，実ネットワークを仮想環境上に再現することにより，SDN と従来型のネットワークの両方をテストできる．本稿では，従来型のネットワークを仮想環境に再現する機能に関する評価方法について検討する．

Consideration of Evaluation Method and Development of Network Management Support System Enabling Replicate Realistic Network using OpenFlow

HIROAKI TSUTSUMI[†1]　NOBUKAZU IGUCHI[†2]

SDN attracts attention along with the popularization of cloud environment and server virtualization. In SDN, it is possible to change the logical topology and route control dynamically by isolation of control plane and data plane. On the other hand, it is difficult to figure out the situation of network. In this study, we have developed network management support system using OpenFlow, which is core technology to realize SDN. This system replicates realistic network to virtual environment so users can test both SDN and current network. In this paper, we discuss evaluation method of the function that replicates current network to virtual environment.

1.　はじめに

クラウド環境やサーバ仮想化の普及に伴い，ネットワークに対する要件が変化してきている．例えばサーバ仮想化により，サーバの追加や移動が起こりやすくなっている．サーバの追加や移動に付随してネットワーク機器の設定を変更する必要がある．しかし，各ネットワーク機器に対して管理者が手動で設定する従来型のネットワークでは，ネットワークの設定に時間がかかり，ミスも発生しやすくなる．そこで，ネットワークの運用を自動化し，管理者の負担を軽減するための仕組みが求められている．

そうした背景から，SDN（Software-Defined Networking）[1]が注目を集めている．SDN のアーキテクチャを図 1 に示す．従来型のネットワークでは，各ネットワーク機器が経路制御といった複雑な計算を行うコントロールプレーンと，パケットの転送といった簡易な処理を行うデータプレーンの両方を持っている．一方，SDN では，従来型のネットワークでは一体化されていたコントロールプレーンとデータプレーンが分離される．従来型のネットワークでは，コントロールプレーンの実装はネットワーク機器ベンダに依存していたため，ベンダの提供する機能しか用いることができない．これに対して SDN では，ネットワーク機器からコントロールプレーンが分離されるため，管理者がコントロールプレーンを開発できる．これによりベンダ依存が解消され，自由なネットワーク制御が可能となる．さらに SDN では，コントロールプレーンによりネットワーク機器を一括して管理できるため，ネットワークの論理トポロジや経路制御を動的に変更できる．これにより管理者の負担が軽減され，運用コストの削減が期待される．しかし，動的な変更が可能になることにより，ネットワークの状況を正確に把握することが困難になる場合がある．この時，障害が発生した際の原因究明が困難となる．こうした理由から，SDN の運用にはネットワークをテストする環境が有用である．また，今後従来型のネットワークから SDN へと移行していく際に，コストの問題等により段階的に移行するこ

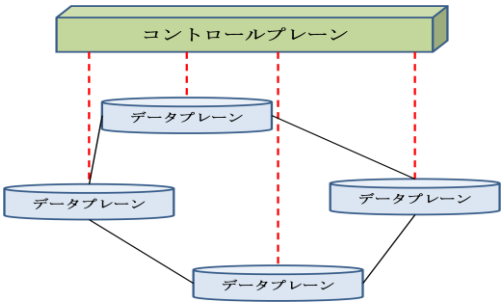

図 1　SDN のアーキテクチャ
Figure 1　Architecture of SDN.

†1 近畿大学大学院 総合理工学研究科
Interdisciplinary Graduate School of Science and Technology Kinki University
†2 近畿大学 理工学部 情報学科
Department of Informatics School of Science and Engineering Kinki University

とが予測される．そのため，SDN と従来型のネットワークが混在する環境が一時的にしろ存在すると考えられる．この場合，SDN だけでなく，従来型のネットワークも同時にテスト可能であるシステムが必要となる．

そこで本研究は，SDN と従来型のネットワークの混在環境に対応し，実ネットワークのテスト環境を仮想的に提供することを目的とする．目的を達成するため，SDN の中核技術である OpenFlow[2]を用いて，ネットワーク運用管理支援システム（以下，本システム）を開発した．本システムでは, OpenFlow ネットワークと従来型のネットワークの両方を仮想環境に再現する．従来型のネットワークの再現には NETCONF（Network Configuration Protocol）[3]を使用する．これにより，OpenFlow ネットワークと従来型のネットワークの混在する環境であってもテストが可能となる．

本稿では，開発したシステムの詳細について述べた後，従来型のネットワークの再現の評価方法について検討する．

2. 関連技術

本章では，本システムの開発に使用した技術のうち，主要なものについて述べる．

2.1 OpenFlow

OpenFlow は，Stanford 大学において研究開発されたネットワークアーキテクチャである．OpenFlow の登場により，SDN が注目を集めるようになった．そのため，現在 OpenFlow に関する研究が盛んに行われている[4][5][6][7][8][9][10]．OpenFlow を用いたネットワークの構成を図 2 に示す．OpenFlow を用いたネットワークは，データプレーンに相当する OpenFlow スイッチと，コントロールプレーンに相当する OpenFlow コントローラ（以下，コントローラ）から構成される．OpenFlow スイッチとコントローラは, OpenFlow プロトコルにより規定されているメッセージを用いて通信する．OpenFlow スイッチは，フローテーブルと呼ばれるテーブルを持っている．フローテーブルには，コントローラから指示された，パケットの処理方法が記述されたエントリが格納されている．これをフローエントリと呼ぶ．フローエントリには，マッチフィールドという部分があり，そこにはパケットのヘッダ情報に相当する値が格納されている. OpenFlow スイッチがパケットを受信すると自身のフローテーブルを参照し，各フローエントリのマッチフィールドとパケットのヘッダ情報が一致するかどうかを調べる．一致するフローエントリが存在する場合，その内容に従ってパケットを処理する．存在しない場合，コントローラへ処理を問い合わせる．コントローラはパケットのヘッダ情報を基にどう処理するかを決定し，OpenFlow スイッチに知らせる．

このように OpenFlow では，コントローラの挙動によりネットワーク全体の動作が決定される．そのため，OpenFlow ではコントローラを作成することにより, 各ネットワークの要件に最適なネットワークの設定や制御ができる．また OpenFlow では，コントローラから OpenFlow スイッチを一括して制御できるため，ネットワークの運用負担を軽減できる．

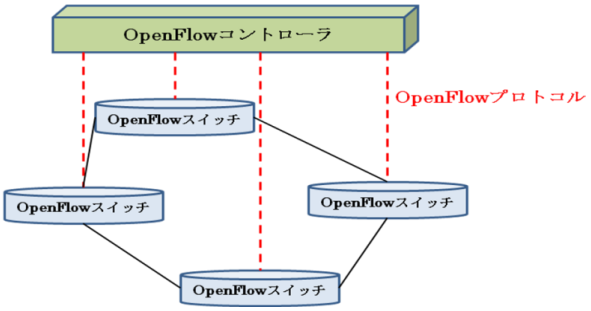

図 2 OpenFlow のアーキテクチャ
Figure 2 Architecture of OpenFlow.

2.2 NETCONF

NETCONF は，ネットワークを管理するためのプロトコルである．IETF により 2006 年に RFC4741 として公開された．NETCONF は，従来型のネットワーク機器の設定を取得･変更するための RPC に基づいたメカニズムを提供している． NETCONF はサーバとクライアントから構成される．サーバはネットワーク機器であり，クライアントはサーバを設定するためのアプリケーションである．クライアントは BEEP や SSH を用いてサーバと接続する．これにより，クライアントとサーバ間でのセキュリティが確保される．接続が確立されると，クライアントはサーバへリクエストを送信できるようになる．クライアントから送られるリクエストは XML で記述されている．設定データを受け取ったサーバは，その XML を解釈しレスポンスを返す．クライアントはレスポンスの内容を解釈することにより，ネットワーク機器の情報を取得できる．

3. ネットワークテスト環境の要件

SDN のテスト環境の提供を目的とした研究として，Mininet[11]と Flowvisor[12]がある．

Mininet は Open vSwitch を用いて仮想的に OpenFlow ネットワークのテスト環境を提供する. Mininet では設定ファイルを定義することで，自由なネットワークのトポロジでテストが可能である．しかし Mininet で提供されるテスト環境には設定が施されていない．そのため，実ネットワークと同等の環境でテストを実施するには，テスト環境に対して新規に設定する必要がある．実ネットワークの規模が大きくなると，設定作業はより困難になる．

Flowvisor は物理 OpenFlow ネットワークのトポロジや各 OpenFlow スイッチの帯域，フローテーブル等を分割し，それらをコントローラへと割り当てる．これにより，各コントローラに独立した論理 OpenFlow ネットワークを提供する．分割された各論理 OpenFlow ネットワークのことをス

ライスと呼ぶ. 各スライスは他のスライスの影響を受けず，また他のスライスに影響を及ぼさない．そのため割り当てられたスライスをテストに用いることで，実稼働しているネットワーク上でのテストが可能である．しかし一方で，各スライスは完全に独立していることから，他のコントローラによりすでに施されている，あるいは新たに施された設定の影響について検証できない．

以上を踏まえて，本システムが提供するテスト環境の要件について述べる．本システムのテスト環境の要件を以下のように定義する．

・**要件 1：** 実ネットワークと同じトポロジで，同じ設定を施していること

ネットワークのテストが必要となった際に，新規にテスト用のネットワークを構築し，同じ設定を施すのはコストがかかる．そこで本システムのテスト環境では，実ネットワークと同じトポロジで，同じ設定が施されていることを要件とする．この要件を満たすことにより，テスト用のネットワークを構築するためのコストが削減できる．

・**要件 2：** OpenFlow ネットワークの設定変更がリアルタイムでテスト環境に反映されること

前述したとおり，SDN ではネットワークの論理トポロジや経路制御を動的に変更できる．ところが，動的に変更できるために，実ネットワークの状況を正確に把握するのが困難な場合がある．そこで本システムのテスト環境では，実ネットワークに施された設定が，リアルタイムで仮想環境にも反映されることを要件とする．この要件を満たすことにより，実ネットワークの状況に応じたテストが可能となる．

4. 既存のネットワーク運用管理支援システム

本章では，既存のネットワーク運用管理支援システムについて述べた後，それらのシステムと本システムとの違いについて述べる．

4.1 Zabbix

Zabbix[13]は，Zabbix SIA により開発・提供されているオープンソースのネットワーク運用管理システムである．Zabbix では，SNMP を用いることによりネットワーク機器を管理できる．また，Zabbix エージェントをインストールすることにより，サーバ等の CPU，ディスク容量，ネットワークの使用率を監視できる．さらに，事前にイベントを定義しておくことで，メール等のアラートによりそのイベントが発生したことを通知できる．

しかし，Zabbix は OpenFlow には対応していない．そのため，コントローラや OpenFlow スイッチがどのように動作しているかを確認できない．また Zabbix では，ネットワークのテスト環境を提供していない．

4.2 Hinemos

Hinemos[14]は，NTT データにより開発・提供されているオープンソースのネットワーク運用管理システムである．Hinemos は Zabbix と同様に，SNMP を用いることによりネットワーク機器を管理できる．また Hinemos は OpenFlow に対応しており，画面上でネットワークを設定するだけで仮想ネットワークを自動で構成できる. さらに, 従来 SNMP により実現されていた障害監視を OpenFlow で実現している．これにより，障害を検知した後自動的に経路を迂回させ，通信を継続できる．

しかし，Hinemos はネットワークのテスト環境を提供していない．また，Hinomos はコントローラとして動作するため，使用するコントローラを選択できないといったデメリットがある．

4.3 VOLT

VOLT（Versatile OpenFlow Validator）[15]は，NTT コミュニケーションズにより開発された，SDN によるネットワークを設計･テストするためのシステムである. VOLT では，OpenFlow ネットワークの構成や経路情報を複製したテスト環境をシステム上に構築する．これにより，実環境と同条件で新たなネットワークの設計・テストが可能となる．また，テスト環境で設定した内容を実ネットワークに反映できる．

しかし，VOLT では使用できるコントローラが Ryu[16]に制限されている．そのため，異なるコントローラを使用しているOpenFlowネットワークでは使用できない. また，VOLT は従来型のネットワークの再現はできない．

4.4 既存のシステムと本システムとの比較

本節では，先述した既存のシステムと本システムとを比較し，その違いについて述べる．表 1 に，既存のシステムと本システムとの機能の違いについてまとめた．表 1 から分かる通り，従来型のネットワークと OpenFlow ネットワーク両方のテスト環境を提供できるのは本システムのみである．また本システムでは，使用するコントローラに制限がない．このことから，本システムは OpenFlow ネットワークと従来型のネットワークが混在している環境において有用であると言える．

表 1 既存のシステムとの比較
Table 1 Comparison of this system with existing system.

	Zabbix	Hinemos	VOLT	本システム
OpenFlow 対応	×	○	○	○
OpenFlow ネットワーク テスト環境の提供	×	×	○	○
コントローラの制限	-	あり	あり	なし
従来の ネットワーク テスト環境の提供	×	×	×	○

5. 開発したシステム

本システムは, OpenFlow ネットワークと従来型のネット

ワークが混在する環境での使用を想定した，ネットワークの運用管理を支援するシステムである．5.1 節では，まず本システムの構成について述べる．5.2 節では，ネットワークのテスト環境を提供するためのネットワークエミュレーション機能について述べる．

5.1 本システムの構成

本システムは，コントローラと OpenFlow スイッチの間で，プロキシのように動作する．本システムを用いた場合の構成を図 3 に示す．

本システムが動作するサーバは，ネットワーク上の OpenFlow スイッチに対してはコントローラのように振る舞い，コントローラに対しては OpenFlow スイッチのように振る舞う．本システムが OpenFlow スイッチから接続要求を受け取ると，コントローラへと接続を試みる．コントローラとの接続が確立されると，OpenFlow スイッチからのメッセージを受信する．実際に OpenFlow スイッチを制御するのは本システムと接続されているコントローラである．しかし，OpenFlow スイッチにとっての見かけ上のコントローラは本システムである．したがって，OpenFlow スイッチからのメッセージは本システム宛に送信される．これにより，本システムは OpenFlow スイッチからのメッセージを収集できる．

受け取ったメッセージは本システムを経由してコントローラへと送信される．コントローラにとって実際の制御対象となる OpenFlow スイッチはネットワーク上に存在している．しかしコントローラにとっての見かけ上の OpenFlow スイッチは本システムである．したがって，コントローラからのメッセージも本システム宛に送信される．そのため，本システムはコントローラからのメッセージを収集できる．このメッセージもまた，本システムを経由して OpenFlow スイッチへと送信される．

以上より，本システムは OpenFlow スイッチからのメッセージと，コントローラからのメッセージの両方を収集できる．収集したメッセージを解析することにより，ネットワーク上でどのようなイベントが起こり，それに対してコントローラがどのように対応したのかを確認できる．これにより，本システムを用いることで，ネットワークが実際にどのように動作しているかを把握できる．

また，本システムは従来型のネットワーク機器に対して，NETCONF クライアントとして動作する．このとき，ネットワーク機器は NETCONF サーバとして動作している必要がある．クライアントとして動作する本システムが，サーバであるネットワーク機器と接続される．接続が確立されると，NETCONF を用いてやり取りすることにより，ネットワーク機器の設定情報を取得する．取得した情報は，ネットワークエミュレーション機能で使用する．

5.2 ネットワークエミュレーション機能

本機能は，利用者にネットワークのテスト環境を提供する．本機能では，実ネットワーク上の機器を仮想環境上に再現する．再現されるネットワーク機器には，実際のネットワーク機器と同等の設定が反映されている．以下，OpenFlow ネットワークを再現する場合と従来型のネットワークを再現する場合の本システムの動作について述べる．

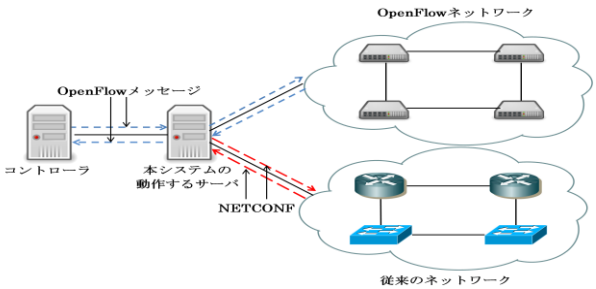

図 3　システム構成
Figure 3　System architecture.

5.2.1 OpenFlow ネットワークの再現

本機能で OpenFlow ネットワークを再現する場合，仮想 OpenFlow スイッチには Open vSwitch[17]を使用する．Open vSwitch は Linux 上で動作する仮想スイッチで，OpenFlow に対応している．

本システムが OpenFlow スイッチと接続されると，その OpenFlow スイッチと同じ Datapath-id とポートを持った仮想 OpenFlow スイッチを，Open vSwitch を用いて作成する．作成された仮想 OpenFlow スイッチを相互に接続することにより，OpenFlow ネットワークのトポロジを再現する．

本システムでは OpenFlow ネットワークのトポロジを推定するため，LLDP（Link Layer Discovery Protocol）を用いて OpenFlow スイッチ間のリンクを検出している．具体的には，まずコントローラは OpenFlow スイッチの Datapath-id とポート番号を含んだ LLDP パケットを生成する．そして生成した LLDP パケットを，LLDP パケット内に記述された Datapath-id を持つ OpenFlow スイッチのポートから出力させる．

その LLDP パケットを接続先の OpenFlow スイッチが受信すると，そのパケットの処理をコントローラへ問い合わせる．問い合わせを受けたコントローラは，その内容を解析することにより，OpenFlow スイッチがどのポートでその LLDP パケットを受信したのかを把握できる．これにより OpenFlow スイッチ間のリンクを検出できる．リンクが検出されると，そのリンクで接続されている OpenFlow スイッチに相当する，仮想 OpenFlow スイッチ間を接続する．これにより，OpenFlow ネットワークのトポロジを仮想的に再現できる．再現した仮想 OpenFlow ネットワークに対してリアルタイムで設定を反映するために，本システムは収集したメッセージを実ネットワークの OpenFlow スイッチだけでなく，仮想 OpenFlow スイッチにも送信する．これにより，仮想環境の OpenFlow スイッチに実環境の OpenFlow スイッチと同じ設定を反映できる．したがって，本機能を

用いることで OpenFlow ネットワークを仮想環境に再現できる．

5.2.2 従来型のネットワークの再現

従来型のネットワークを再現する場合，仮想ネットワーク機器には Linux network namespace と Open vSwitch を用いる．Linux network namespace は Linux が持つネットワーク仮想化機能で，固有のルーティングテーブルやポート等を持つ独立したネットワーク環境をホスト内に作成できる．現在本システムで再現可能なネットワーク機器はルータと L2 スイッチである．ルータを再現する場合は Linux network namespace を，L2 スイッチを再現する場合は Open vSwitch を用いる．

利用者は本システムに対して再現したい機器の IP アドレスとユーザ名，パスワードを指定する．本システムはそれらの情報を基に，NETCONF クライアントとして NETCONF サーバであるネットワーク機器と SSH を用いて接続する．接続が確立されると，本システムはネットワーク機器の情報を収集する．ルータの場合はインターフェース，ルーティングテーブル，アクセスリスト等の情報を収集し，L2 スイッチの場合は，インターフェース，VLAN，STP，LLDP 等の情報を収集する．本システムではそれらの情報を基にネットワークのトポロジを再現する．

本システムでネットワークのトポロジを再現する際，ルータ間のリンクを検出するために，ルーティングテーブルの情報を利用する．ルーティングテーブルから直接接続されているネットワークの IP アドレスを取得する．取得した直接接続されているネットワークの IP アドレスと，他のルータのインターフェースの IP アドレスを比較することにより，ルータ間のリンクを検出する．L2 スイッチ間，あるいは L2 スイッチとルータ間のリンクを検出するには，LLDP の情報を利用する．LLDP の情報を利用することで，その L2 スイッチに接続されている機器の情報を把握できる．また，ルータ，L2 スイッチともに，CDP（Cisco Discovery Protocol）等のベンダ独自の近隣探索プロトコルの情報も利用する．これにより，同一ベンダ機器間であればリンクを検出できる．

以上の動作によって，本システムがネットワークのトポロジを推定した後，Linux network namespace と Open vSwitch を用いて仮想環境上にトポロジを再現する．そして再現したトポロジの仮想ネットワーク機器に対して，NETCONF で収集した情報を基に設定することで，ネットワーク機器の設定を仮想ネットワーク機器に反映する．ルータを再現する場合，ルーティングには Linux のルーティング機構を使用し，アクセスリストには iptables を用いる．L2 スイッチを再現する場合は Open vSwitch を用いる．したがって，本機能により従来型のネットワークを仮想環境に再現できる．

以上より，本機能は OpenFlow ネットワークと従来型のネットワークが混在するネットワークを仮想的に再現できる．利用者は本機能により提供される仮想環境をテストに用いることで，実際のネットワークに影響を及ぼすことなく，ネットワークをテストできる．そのため，ネットワークの急な設定変更にも柔軟に対応できる．また，ネットワークに障害が発生した際の原因の究明が容易になる．

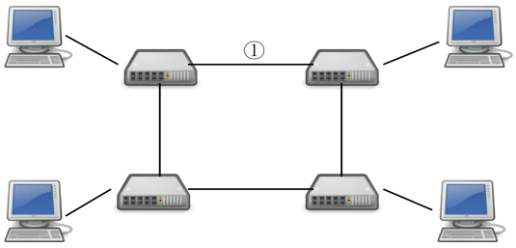

図 4 検証に用いたトポロジ
Figure 4　Network topology used for verification.

表 2　ネットワークエミュレーション機能の検証結果
Table 2　Verification result of network emulation function.

	Ryu	POX	Floodlight	Trema	Beacon
Datapath-id	○	○	○	○	○
トポロジ	○	○	○	○	○
フローエントリ	○	○	○	○	○

6. 動作検証

6.1 実施済みの検証

本節では，実施した動作検証について述べる．動作検証では，主要なコントローラのフレームワークである Ryu，POX[18]，Floodlight[19]，Trema[20]，Beacon[21]のそれぞれが，本システムを用いた際でも正常に動作するかを確認した．検証には，Mininet を用いて構築した図 4 に示す OpenFlow ネットワークを使用した．検証では，図 4 のネットワークにおいて，本システムを使用した場合と使用しなかった場合の各 OpenFlow スイッチのフローエントリの比較と各ホストからの ping による接続性確認を実施した．その後，図 4 の①に示す箇所のリンクを意図的にダウンさせた．その状態で，先ほどと同じ検証を実施した．検証の結果，どちらの場合においても相違点が見られなかったことから，本システムは主要なコントローラフレームワークにおいて正常に動作することが確認された．このことから，本システムは検証に使用したコントローラが動作する OpenFlow ネットワークで使用可能であることが分かった．

また，それぞれのコントローラを用いた際に，ネットワークエミュレーション機能が正常に動作しているかを確認した．具体的には，仮想環境で再現された OpenFlow ネットワークと実際の OpenFlow ネットワークにおいて，ネットワークのトポロジ，OpenFlow スイッチの datapath-id，OpenFlow スイッチ内のフローエントリを比較した．そして，それらが一致している場合は正常に動作していると判断し

表 3　実験用ネットワークの設定項目
Table 3　Configuration items of experimental network.

	設定項目	
L2 スイッチ	VLAN	
	STP	
ルータ	ルーティング	RIP
		OSPF
		static ルーティング
	アクセスリスト	

た．その結果を表 2 に示す．表 2 より，本システムのネットワークエミュレーション機能は，主要なコントローラフレームワークにおいて正常に動作することが分かる．これにより本システムは，検証に使用したコントローラが動作する OpenFlow ネットワークのテスト環境を提供できる．

6.2 実施予定の検証

6.1 節の動作検証により，OpenFlow ネットワークを仮想環境に再現し，テスト環境を提供できることが分かった．しかし，従来型のネットワークの再現に関しての検証はできていない．ここでは，従来型のネットワークの再現に関して実施する予定の検証と，それにより期待される結果について述べる．

検証には, L2 スイッチとルータを使用して構築したネットワークを使用する．その際，Cisco やアラクサラ等，複数のベンダの機器を使用する．これにより，本システムがベンダに依存せず従来型のネットワークを再現可能であることを実証できる．検証では，L2 スイッチとルータに表 3 に示す項目を設定する．ルーティングの設定には RIP，OSPF，static ルーティングの 3 パターンを使用し，各パターンについて検証する．

検証では，ping や traceroute 等のコマンドを用いて，実ネットワーク機器の設定を仮想ネットワーク機器に反映できているかを確認する．コマンドの出力や実行結果が実ネットワーク機器と仮想ネットワーク機器で一致していれば，本システムが従来型のネットワークを再現可能であることを実証できる．さらに，システム稼働中のメインメモリと CPU の使用量を計測する．これにより，本システムがどの程度の規模のネットワークまで対応できるかを把握できる．

7. おわりに

本研究では，SDN のテスト環境を提供するため，OpenFlow を用いてネットワークの運用管理を支援するシステムを開発した．本システムは，コントローラと OpenFlow スイッチの間でやり取りされるメッセージを収集することにより，実際の OpenFlow ネットワークを仮想環境で再現できる．

動作検証の結果，本システムは今回テストしたコントローラが動作する OpenFlow ネットワークであれば，OpenFlow ネットワークを問題なく再現できることが分かった．しかし，従来型のネットワークの再現に関しては検証できていない．そのため本稿では，従来型のネットワークの再現に関して実施予定の検証と，それによって期待される結果について述べた．

今後は，本稿で述べた検証を実施し，その結果を基にシステムの実装を進めていく予定である．

参考文献

1) Software-Defined Networking，https://www.opennetworking.org/
2) McKeown, N. et al.: OpenFlow: Enabling Innovation in Campus Networks, ACM SIGCOMM Computer Communications Review Vol.38 Issue2 pp.69-74(2008).
3) Network Configuration Protocol，http://tools.ietf.org/html/rfc6241
4) Khurshid, A. et al: Veriflow: verifying network-wide invariants in real time, In Proceedings of NSDI'13, 2013.
5) Koponen , T. et al.: Onix: a distributed control platform for large-scale production networks, In Proceedings of OSDI, 2010, pp.1-6.
6) Rotsos, C. et al: OFLOPS: an open framework for openflow switch evaluation, Passive and Active Measurement, 2012, pp.85-95.
7) 田島伸一，他: OpenFlow を用いた未使用 IP アドレスへの通信をハニーポットへ集約する方法の検討，情報処理学会全国大会講演文集，vol.75，no.1，pp.541-543，March 2013.
8) 太田悟，他: OpenFlow を用いた DDoS 攻撃検知システムの検討，情報処理学会全国大会講演文集，vol.75，no.1，pp.543-545，March 2013.
9) 熊谷友来，他: OpenFlow をベースとした災害時における End-to-End 通信路の選択方法の実現，情報処理学会全国大会講演文集，vol.75，no.1，pp.337-339，March 2013.
10) 中村遼，他:SDN を用いたクラウドサービスネットワークの実現, 電子情報通信学会技術研究報告, vol. 113, no. 200, IA2013-17, pp. 5-10，September 2013.
11) Lantz, B. et al.: A network in a laptop: Rapid prototyping for software-defined networks, Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp.1-6.
12) Sherwood, R. et al.: Can the Production Network Be the Testbed?, In Proceedings of OSDI，2010，pp.365-378.
13) Zabbix，http://www.zabbix.com/jp/
14) Hinemos，http://www.hinemos.info/
15) VOLT，http://www.ntt.com/release/monthNEWS/detail/20130611.html
16) Ryu，http://osrg.github.io/ryu/
17) Open vSwitch，http://openvswitch.org/
18) POX，http://www.noxrepo.org/
19) Floodlight，http://www.projectfloodlight.org/
20) Trema，http://trema.github.io/trema/
21) Beacon，https://openflow.stanford.edu/display/Beacon/Home