E-25

# 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

## Development of Network Management Support System Enabling Replicate Realistic Network using OpenFlow

堤 啓彰†　井口 信和‡
Hiroaki Tsutsumi　Nobukazu Iguchi

## 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 では，従来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．従来のネットワークと異なるアーキテクチャにより，ネットワークの柔軟な設定や制御が可能となる．しかし SDN は，運用実績が少ないこともあり，トラブルが発生した際の対処法やノウハウが確立されていない．そのため，原因の究明や解決が困難であるといった問題がある．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必要である．また，今後従来のネットワークから SDN へと移行していく際に，コストの問題により段階的に移行することが予測される．そのため，SDN と従来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．したがって，SDN だけでなく，従来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．

そこで本研究では，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．さらに，SDN と従来のネットワークの混在環境にも対応する．これら二つの目的を達成することで，SDN の運用を支援できる．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のネットワークの両方を仮想環境に再現する．従来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，まず 2 章で関連技術について述べる．3 章では既存のネットワーク運用管理支援システムと本システムの違いについて述べる．4 章では開発したシステムの構成とその機能について述べる．5 章では実施した動作検証とその考察について述べる．6 章ではまとめと今後の予定について述べる．

†近畿大学大学院 総合理工学研究科, Interdisciplinary Graduate School of Science and Technology Kinki University

‡ 近畿大学 理工学部 情報学科，Department of Informatics School of Science and Engineering Kinki University

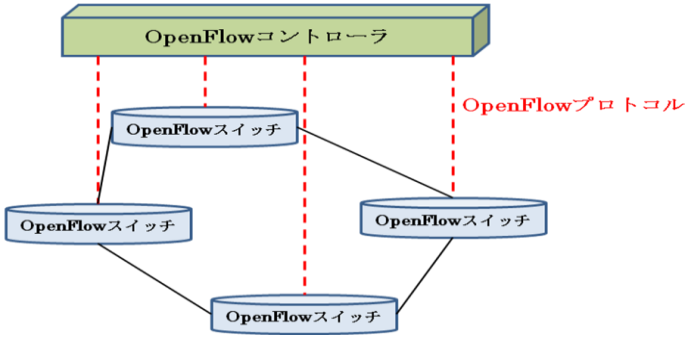


**図 1 ： OpenFlow のアーキテクチャ**

## 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)]．OpenFlow を用いたネットワークの構成を図 1 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と，その内容に従ってパケットを処理する．存在しない場合，コントローラへと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，

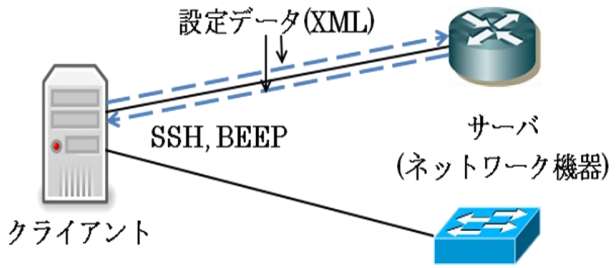


**図 2 ： NETCONF の構成**

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．

### 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．NETCONF を用いる際の構成を図 2 に示す．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. 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．

### 3.1 Zabbix

Zabbix[7]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．

**表 1 ： 既存のシステムとの比較**

| | 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 | × | × | × | ○ |

### 3.2 Hinemos

Hinemos[8]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 では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．

### 3.3 VOLT

VOLT(Versatile OpenFlow Validator)[9]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[10]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 は従来のネットワークの再現はできない．

### 3.4 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．前述したとおり，Zabbix は OpenFlow に対応していない．そのため，SDN の運用には使用できない．対して Hinemos は OpenFlow に対応している．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．そのため，設定の自動化により，障害発生時の原因究明が困難となる SDN の運用には不十分である．また Hinemos では，使用するコントローラを自由に選択できない．VOLT もまた OpenFlow に対応している．さらに VOLT は OpenFlow ネットワークのテスト環境を提供している．しかし VOLT は，使用するコントローラが Ryu に制限されるため，自由にコントローラを選択できない．そのため VOLT では，ネットワークの動作が Ryu の機能に依存してしまう．また，VOLT では従来のネットワークのテスト環境を提供していない．そのため，従来のネットワークと OpenFlow ネットワークが混在する環境においては VOLT だけでは不十分であると言える．一方，本システムは従来のネット

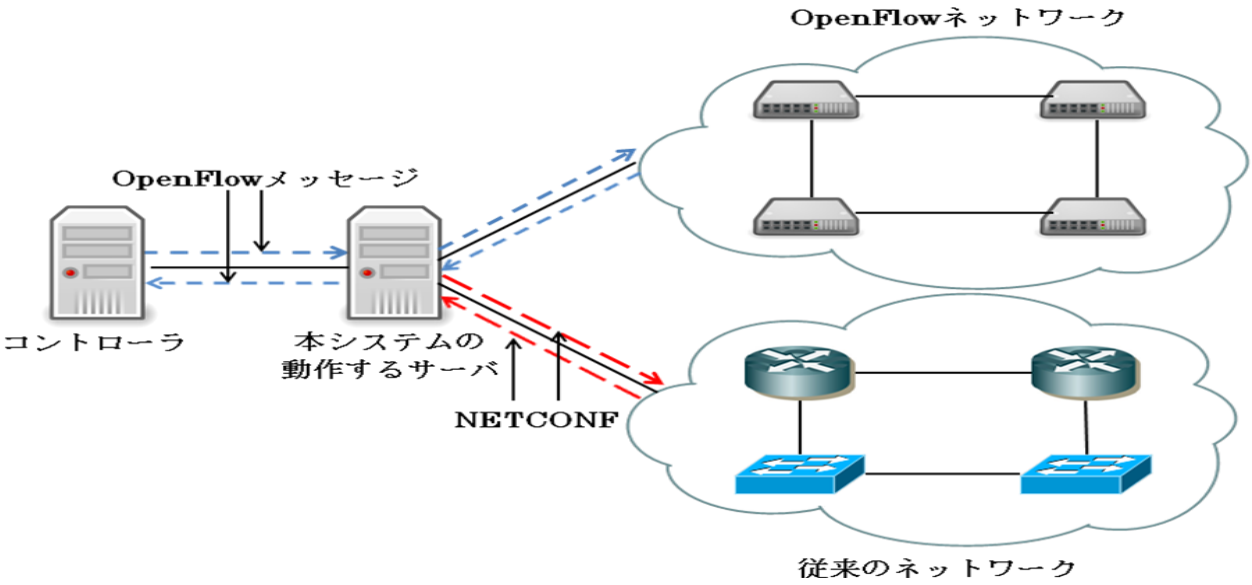


**図 3 ： 本システムの構成**

ワークと OpenFlow ネットワーク両方のテスト環境を提供している．また本システムでは，使用するコントローラの制限がない．このことから，本システムは OpenFlow ネットワークと従来のネットワークが混在している環境において有効であると言える．

## 4. 開発したシステム

### 4.1 概要

本システムは，OpenFlow ネットワークと従来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．4.2 節では，まず本システムの構成について述べる．4.3 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．

### 4.2 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．そのため，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．そのため，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，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．そのため，本システムを用いることにより，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．

### 4.3 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．

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[11]を，仮想ホストには Linux network namespace を使用する．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，本機能により OpenFlow ネットワークを仮想環境に再現できる．

従来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 を用いる． NETCONF で収集した情報を基に仮想ネットワーク機器を設定することで，ネットワーク機器の再現を可能とする．

**表 2 ： ネットワークエミュレーション機能の検証結果**

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

したがって，本機能により従来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．

## 5. 動作検証

本章では，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[12]，Floodlight[13]，Trema[14]，Beacon[15]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[16]を用いて構築した OpenFlow ネットワークを使用した．検証の結果，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と正常に動作していると判断した．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．今後は，従来のネットワークの再現について検証する予定である．

## 6. 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問題なく使用できることが分かった．

今後は，ネットワークを運用管理する上で必要となる機能を実装し，システムの評価実験をしていく予定である．

## 参考文献


1) Software-Defined Networking, https://www.opennetworking.org/
2) 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 Communications Review Vol.38 Issue2 pp.69-74(2008).
3) Network Configuration Protocol, http://tools.ietf.org/html/rfc6241
4) R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. M. Parulkar : Can the Production Network Be the Testbed?, In Proceedings of OSDI, 2010, pp.365-378.
5) T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Yuichiro Iwata, H. Inoue, T. Hama, S. Shenker : Onix: a distributed control platform for large-scale production networks, In Proceedings of OSDI, 2010, pp.1-6.
6) C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, A. W. Moore, OFLOPS: an open framework for openflow switch evaluation, Passive and Active Measurement pp.85-95(2012).
7) Zabbix, http://www.zabbix.com/jp/
8) Hinemos, http://www.hinemos.info/
9) VOLT, http://www.ntt.com/release/monthNEWS/detail/20130611.html
10) Ryu, http://osrg.github.io/ryu/
11) Open vSwitch, http://openvswitch.org/
12) POX, http://www.noxrepo.org/
13) Floodlight, http://www.projectfloodlight.org/
14) Trema, http://trema.github.io/trema/
15) Beacon, https://openflow.stanford.edu/display/Beacon/Home
16) Mininet, http://mininet.org/