情報処理学会論文誌 Vol.58 No.3 672–682 (Mar. 2017)

OpenFlowネットワーク移行支援システム

野村 圭太[1,a)] 谷口 義明[2] 井口 信和[2] 渡辺 健次[3]

受付日 2016年6月22日, 採録日 2016年12月1日

概要：今後，企業や大学などの既存ネットワークにおいてOpenFlowネットワークへの移行が進められると予測される．しかし，OpenFlowネットワークへの移行には，コントローラの設定やテスト環境の構築にコストを要すると考えられる．そこで本稿では，従来型のネットワークから，OpenFlowネットワークへの移行を支援するシステムを提案，設計，実装，評価する．提案システムを用いることにより，従来型のネットワーク上のルータから自動的に設定情報を取得し，その設定情報をOpenFlowネットワークで適用可能な形式に変換，その後，変換した設定情報をOpenFlowネットワークへ反映させることができる．これにより，従来型のネットワークと同等のパケット制御を行うOpenFlowネットワークを半自動的に構築できる．本稿では，実ルータを用いた評価の結果，最大10台のルータから構成される従来型のネットワークを6分以内でOpenFlowネットワークに移行できることを確認した．

キーワード：Software-Defined Networking，OpenFlow，NETCONF

A System for Supporting Migration to OpenFlow Network

Keita Nomura[1,a)] Yoshiaki Taniguchi[2] Nobukazu Iguchi[2] Kenzi Watanabe[3]

Received: June 22, 2016, Accepted: December 1, 2016

Abstract: OpenFlow networks have attracted attention along with the popularization of cloud environment and server virtualization since it enables flexible network configuration. It is expected that existing networks of various organizations and universities will migrate to OpenFlow networks in the future. However, it takes costs to migrate a traditional network to an OpenFlow network as well as costs for test. In this paper, we propose, design, implement and evaluate a system for supporting migration by acquiring settings from a traditional network and reflecting them to an OpenFlow network. Through experimental verification, we demonstrate that our system can migrate a traditional network composed of up to 10 routers to an OpenFlow network within 6 minutes.

Keywords: Software-Defined Networking, OpenFlow, NETCONF

1. はじめに

クラウド環境やサーバ仮想化の普及にともない，ネットワークに対する要件が変化してきている．たとえばサーバ仮想化技術により，仮想サーバの追加や移動が可能となった．このような環境の変化にともない，ネットワークの構成や，ルータ，スイッチなどのネットワーク機器の設定を変更する必要が生じる場合がある．しかし，従来型のネットワーク機器で構成されるネットワークでは，各ネットワーク機器を手動で設定する必要がある．そのため，ネットワークの構成や機器の設定の変更に手間を要し，サーバの追加や移動などの環境の変化に柔軟に対応することが困難となっている．そこで，このような環境の変化に柔軟に対応するための仕組みが求められている．

そうした背景から，Software-Defined Networking（SDN）

[1] 近畿大学大学院総合理工学研究科
Graduate School of Science and Technology, Kindai University, Higashiosaka, Osaka 577–8502, Japan
[2] 近畿大学理工学部情報学科
School of Science and Engineering, Kindai University, Higashiosaka, Osaka 577–8502, Japan
[3] 広島大学大学院教育学研究科
Graduate School of Education, Hiroshima University, Higashihiroshima, Hiroshima 734–8551, Japan
[a)] 1533340411s@kindai.ac.jp

というコンセプトが注目を集めている [1], [2], [3], [4], [5]．従来型のネットワークでは各ネットワーク機器上に，経路制御を行うソフトウェア・OS 部に相当する制御部と，フレームやパケットの転送を行うハードウェア部に相当する転送部の両方がある．これに対し SDN では，制御部と転送部が分離されている．従来型のネットワークでは，制御部の実装はネットワーク機器のベンダに依存しているため，ベンダが提供する機能しか用いることができない．一方，SDN では，管理者が自由に制御部を開発できるため，ベンダ依存が解消され，用途に応じた柔軟なネットワークを構築できる．

SDN を実現するための技術の 1 つに OpenFlow [6] がある．OpenFlow を用いることにより，ネットワークの運用コストや新規のネットワーク構築に要する時間を削減できると期待されている [7]．そのため，今後，企業や大学などの既存のネットワークの OpenFlow ネットワークへの移行が進められると予測されている [8], [9], [10]．しかし，OpenFlow ネットワークは従来型のネットワークとは異なるアーキテクチャである．そのため，OpenFlow に習熟していない管理者にとって，従来型のネットワークから OpenFlow ネットワークへの移行や OpenFlow コントローラの設定は困難である．また，OpenFlow のアーキテクチャに習熟している管理者でも OpenFlow ネットワークへの移行，OpenFlow コントローラの設定には時間を要する．さらに，OpenFlow ネットワークへ移行する前に，設定した OpenFlow コントローラが正しく OpenFlow ネットワーク上で動作するかの検証を行うことが望ましい．しかし，規模の大きな OpenFlow ネットワークの場合，動作検証を行う環境の構築に時間やコストを要する．こうした理由から，OpenFlow ネットワークへの移行には，OpenFlow コントローラの設定，テスト環境の構築に時間やコストを要すると考えられる．そのため，これらの時間やコストを削減可能なシステムが求められる．

そこで本稿では，従来型のネットワークから，OpenFlow ネットワークへの移行を支援するシステムを提案，設計，実装，評価する．提案システムを用いることにより，従来型のネットワーク上のルータから NETCONF [11] を用いて自動的に設定情報を取得し，その設定情報を OpenFlow ネットワークで適用可能な形式に変換，その後，変換した設定情報を OpenFlow ネットワークへ反映させることができる．これにより，従来型のネットワークと同等のパケット制御を行う OpenFlow ネットワークを半自動的に構築できる．このことにより，OpenFlow ネットワーク移行のための OpenFlow コントローラの設定コストを削減できる．また，提案システムでは，仮想 OpenFlow ネットワークを 1 台の PC 上で構築できる Mininet [12] を用いて，移行後の OpenFlow ネットワークの動作検証が可能であり，テスト環境の構築に要するコストを削減できる．本稿では，提案システムを実装し，実ルータを用いた評価を行う．

なお，本稿における提案システムでは，従来型のネットワークから OpenFlow ネットワークへの移行支援に関する基本的なシステムの提案，実装，評価を行うため，ネットワークの経路制御が安定して動作している従来型のネットワークを移行の対象とする．安定して動作しているネットワークでは，ネットワーク障害などによる経路切替えは起こりにくい．そのため，従来型のネットワークの経路情報のみを OpenFlow ネットワークへ移行すれば十分有効であると考えらえる．そのため，本稿では，経路制御機能の移行は対象としない．

本稿の構成は以下のとおりである．まず，2 章で関連研究について述べる．次に，3 章で本稿における前提環境について述べ，4 章で提案システムの詳細について述べる．5 章で提案システムの評価を行う．最後に，6 章で本稿のまとめを述べる．

2. 関連研究

本章では，本研究と関連する研究について述べ，提案システムとの比較を行う．

従来型のネットワークから設定情報を自動的に取得する手法としてさまざまな手法が提案されている．EDGE [13] は，ネットワークのルータの設定情報をハッシュテーブル形式に変換し，SQL データベースに格納する．そして，データベースに格納した情報を Web ベースの GUI で利用者に表示する．この情報を用いて利用者は，データ分析やネットワークプロトコルのモデリング，ネットワーク運用の練習を行う．これに対して，提案システムでは，取得したルータの設定情報を OpenFlow ネットワークに対応する形式に変換し，ルータと同様のパケット処理を実現する OpenFlow ネットワークを構築する．

Exodus [14] は，本稿における提案システムと同様に，従来型のネットワーク上のルータの設定情報を OpenFlow ネットワークで対応可能な形式に変換する．Exodus が変換可能な設定情報には，インタフェース，標準・拡張・再帰 ACL（Access Control List），NAT，VLAN に用いられるトランク・アクセスポート，OSPF を介した静的・動的ルーティング情報がある．ACL とは，IP アドレスやサブネット，プロトコル，アプリケーションのポート番号などに基づいて，パケット転送の可否を決定するルールを定めたリストのことである．しかしながら，Exodus では，独自の SDN コントローラである Flowlog [15] の使用を想定している．また，Exodus では，設定情報の自動取得については考慮されておらず，Cisco IOS の設定情報をあらかじめ Flowlog で保持しておく必要がある．さらに，現状，Exodus は，Cisco IOS に対応しているルータの設定情報のみ変換が可能であり，他のベンダのルータの設定情報を変換するためには，OS の情報を変換するモジュールを

追加で開発する必要がある．これに対し，提案システムでは，さまざまなベンダのネットワーク機器で採用されている NETCONF を用いてルータから設定情報を取得し，XML ファイルに変換，この XML ファイルに基づいて設定情報を OpenFlow に対応した形式へ変換する．そのため，Exodus に比べ，他のベンダに対応する際のモジュール開発を容易に行え，マルチベンダの環境に対応できる．

B4 [16] は，WAN の BGP ルータを，同等の機能を持つ独自にカスタムした OpenFlow スイッチに手動で置き換える手法を提案している．B4 では，BGP の実現のために Quagga を使用しており，BGP の設定情報を Quagga に移行する必要がある．一方，提案システムでは，標準の OpenFlow スイッチで構成された OpenFlow ネットワークへの移行が可能である．また，ルータからの設定情報の自動取得，OpenFlow ネットワークに対応した形式への設定情報の自動変換を行える．

3. 前提

本稿では，従来型のネットワークから OpenFlow ネットワークへの移行を支援するシステムを提案する．本章では，本稿における移行支援システムが前提とする環境を述べる．

3.1 従来型のネットワーク

従来型のネットワークは，すべてのネットワーク機器がルータで構成されており，任意のルータにホストが接続されているものとする．ネットワークのルーティングプロトコルとして RIP，ACL として，標準 ACL，標準名前付き ACL，拡張 ACL，拡張名前付き ACL を対象とする．なお，ACL には，インターネット層および上位層に関するパケット転送可否の設定が記述されているものとし，ネットワークインタフェース層に関する設定の記述は想定しない．

本稿では，ネットワークの経路制御が安定して動作している従来型のネットワークを移行の対象とする．安定して動作しているネットワークではネットワーク障害などによる経路切替えは起こりにくい．そのため，従来型のネットワークの経路情報のみを OpenFlow ネットワークへ移行すれば十分有効であると考えらえる．そのため，本稿では，従来型のネットワークにおける経路情報のみを移行の対象とし，経路制御機能は移行の対象としない．しかし，ネットワーク障害への対策や負荷分散などのためには経路制御機能の移行は重要であり，経路制御機能の移行機能の開発は今後の課題とする．

また，提案システムでは，RIP や ACL の設定が施されたルータで構成された従来型のネットワークを OpenFlow ネットワークへ移行することを移行ケースとして想定している．これは，従来型のネットワークのうち一部分を OpenFlow に移行しても十分な効果が得られるため [10]，企業や大学などのネットワークにおいて，ルータ部分のみを OpenFlow に移行することが考えられるためである．なお，本稿における提案システムを拡張することで，マルチエリア OSPF や IGRP などを用いた大規模なネットワークの移行支援にも対応可能であり，今後，これらの機能を順次開発する予定である．また，多くの企業や大学などでは L2 スイッチが用いられている．そこで，提案システムの適用範囲を広げるため，ルータと L2 スイッチが混在した従来型のネットワークから OpenFlow ネットワークへの移行を支援する機能を今後検討，実装する予定である．

3.2 OpenFlow ネットワーク

次に，移行後の OpenFlow ネットワークについて述べる．本稿では，あらかじめ実機で構成された OpenFlow ネットワークが物理的に結線，構築されているものとする．移行後の OpenFlow ネットワークは，従来型のネットワークと同一トポロジであり，従来型のネットワークにおけるルータと OpenFlow ネットワークにおける OpenFlow スイッチが 1 対 1 で対応する．また，OpenFlow ネットワークにおいて，従来型のネットワークと同等のパケット処理を実現するものとする．なお，本稿では，パケットの処理手順があらかじめ OpenFlow スイッチに書き込まれる，プロアクティブ型制御の OpenFlow ネットワークを移行の対象とする．

4. OpenFlow ネットワーク移行支援システム

4.1 システム概要

提案システムの概要を図 1 に，構成を図 2 に示す．提案システムは，従来型のネットワークの管理用セグメントおよび OpenFlow ネットワークにおける OpenFlow チャネルとの接続性が保たれる管理用セグメントの両方に配置される．提案システムは，OpenFlow ネットワークに対して OpenFlow コントローラとして振る舞うため，OpenFlow ネットワークに対しては接続性を確保する必要がある．一方，従来型のネットワークから取得した設定情報は設定ファイルとして保存されるため，設定情報取得後は従来型のネットワークとの接続性を保つ必要はない．

提案システムには，図 2 に示すように，NETCONF を用

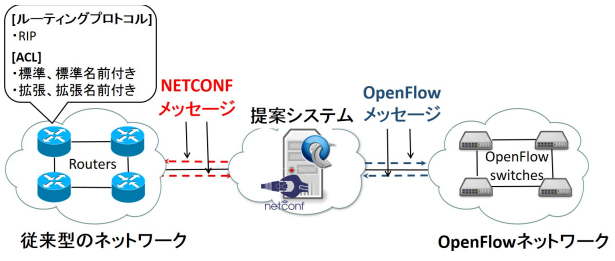

図 1 提案システムの概要
Fig. 1 Overview of proposed system.

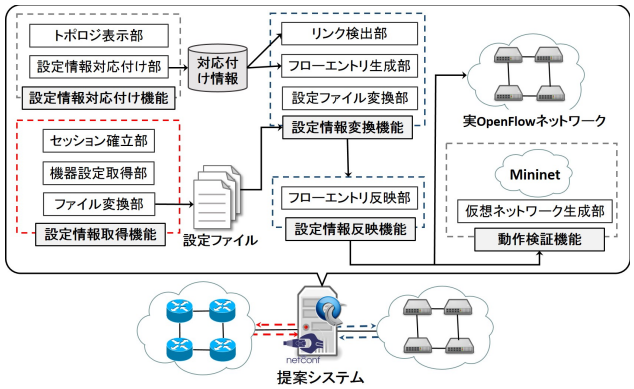

図 2 システムの構成
Fig. 2 System architecture.

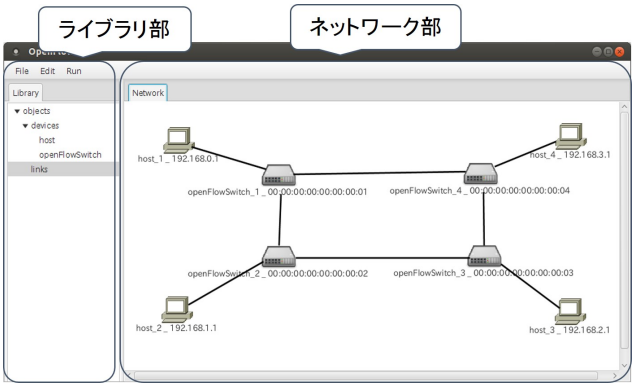

図 3 システム GUI
Fig. 3 System GUI.

いて従来型のネットワーク中のルータから設定情報を自動的に取得する設定情報取得機能，従来型のネットワークにおけるルータと OpenFlow スイッチの対応付け情報およびルータのインタフェースと各 OpenFlow スイッチのポートの対応付け情報を管理する設定情報対応付け機能がある．さらに，取得した設定情報を OpenFlow ネットワークにおけるフローエントリに変換する設定情報変換機能，フローエントリを OpenFlow スイッチに反映する設定情報反映機能，および，移行する OpenFlow ネットワークが正しく動作するかの確認を行う動作検証機能から構成される．

提案システムの利用方法は以下のとおりである．利用者は，初めに，設定情報取得機能を用いて従来型のネットワーク上のルータから設定情報を取得する．設定情報は設定ファイルとして保存される．次に，設定情報対応付け機能を用いて従来型のネットワークのルータと OpenFlow ネットワーク上の OpenFlow スイッチの設定情報，ルータのインタフェースと OpenFlow スイッチのポートの対応づけを行う．提案システムは，これらの機能により取得した設定情報と対応付け情報を用いて，ルータの設定情報をフローエントリへ変換する（設定情報変換機能）．その後，利用者は，設定情報反映機能を用いて，フローエントリを OpenFlow ネットワークへ反映させる．なお，これらの設定情報の取得や対応付け，反映は**図 3** に示すような GUI を用いて行う．さらに，利用者は動作検証機能を用いることにより，必要に応じて，移行後の OpenFlow ネットワークの動作検証を行うことが可能である．以下，それぞれの機能の詳細について述べる．

4.2 設定情報取得機能

設定情報取得機能は，従来型のネットワーク上のルータから設定情報を取得するために用いられる．利用者は，まず，提案システムの GUI（図 3）を用いて設定情報を取得したいルータの IP アドレスと SSH 接続に必要なパスワードを入力し，取得を行うボタンを押下する．このことにより，提案システムは，指定された IP アドレスを持つルータに対して SSH を用いて接続し，NETCONF により設定情報を自動的に取得する．NETCONF を使用することで複数ベンダの機器が混在する環境においても，ベンダごとの違いに影響されずルータの設定情報を取得できる．取得する設定情報は，経路表，標準 ACL，標準名前付き ACL，拡張 ACL，拡張名前付き ACL，およびインタフェースの設定情報である．経路表には，ルーティングプロトコルである RIP により学習した経路の情報とルータに直接接続された経路（直接接続経路）の情報が含まれる．なお，提案システムでは，ホストの情報は取得の対象としない．

なお，本稿では，3 章で述べたように，従来型のネットワークと OpenFlow ネットワークは同一のトポロジであることを想定しており，また，従来型のネットワークにおける経路情報と ACL のみを移行の対象とする．そのため，経路表と ACL の情報の取得が必要となる．加えて，後述するように，OpenFlow スイッチにおけるパケットの出力ポートの決定やフィルタリングの設定時に，従来型のネットワークにおけるインタフェース設定情報が必要となる．したがって，従来型のネットワークからこれらの設定項目を取得できれば，OpenFlow ネットワークへの移行が可能である．また，提案システムでは，NETCONF を使用しているため，今後，たとえば経路制御機能を移行する場合に取得する設定項目を増やす場合においても，取得する設定項目を容易に追加できる．

提案システムは，取得したそれらの情報を基に XML 形式の設定ファイルを生成，保存する．この XML ファイルは，4.4 節で説明する設定情報変換機能において，ルータの設定情報をフローエントリに変換する際に使用する．**図 4** に，XML 形式で保存されたルータの設定情報の例を示す．

4.3 設定情報対応付け機能

設定情報対応付け機能は，ルータと OpenFlow スイッチの対応付け情報およびルータのインタフェースと各 OpenFlow スイッチのポートの対応付け情報を管理するために用いられる．

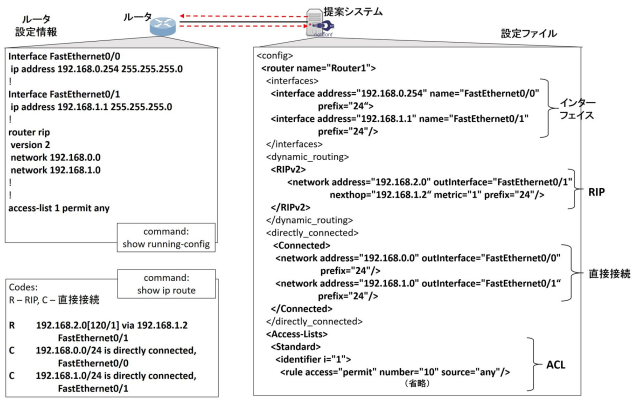

図 4 ルータの設定情報の一例

Fig. 4 An example of setting information of routers.

4.3.1 ルータと OpenFlow スイッチの対応付け

利用者は，まず，システム GUI（図 3）上に移行予定の OpenFlow ネットワークトポロジを作成する．システム GUI 上で OpenFlow スイッチを追加する場合，ライブラリ部からネットワーク部へドラッグアンドドロップする．また，GUI 上の機器どうしを接続することで結線が可能である．次に，利用者は，各 OpenFlow スイッチとルータの設定情報の対応付け設定を行う．設定には，システム GUI 上に表示される設定ウィンドウを用いて，OpenFlow スイッチに対応するルータを選択する．このことにより，システムに OpenFlow スイッチとルータの対応付け情報が設定される．

4.3.2 ルータに接続されているインタフェースと OpenFlow スイッチポートの対応付け

ルータにおいて他のルータとの接続に使用されているインタフェースと OpenFlow スイッチのポートの対応付けは以下のように行う．対応付けには，OpenFlow コントローラ起動時に自動で取得する OpenFlow スイッチ間のリンク情報，上述の操作で得たルータと OpenFlow スイッチの対応付け情報，およびルータに設定されている経路表の情報を用いる．これらを用いて，OpenFlow スイッチどうしの接続に利用されているポートそれぞれに対して，ルータのインタフェースと OpenFlow スイッチのポートの対応付け情報を得る．

4.3.3 ホストに接続されているインタフェースと OpenFlow スイッチポートの対応付け

一方，ホストとの接続に使用されているルータのインタフェースと OpenFlow スイッチのポートの対応付けは以下のように行う．まず，ルータの経路表において直接接続経路エントリが設定されているインタフェースのうち，ルータと接続されていないインタフェースはホストと接続していると判断する．

次に，そのルータのホストとの接続に使用されているインタフェースに対応する OpenFlow スイッチのポートを特定する．提案システムはこの特定のために ARP パケット，および Packet-In メッセージを用いる．具体的には，対応付けが完了していない OpenFlow スイッチの全ポートから ARP Request をマルチキャストするよう，提案システムから OpenFlow スイッチに Packet-Out メッセージを送信する．ARP Request の送信元 MAC アドレスは任意の固定アドレス，送信元 IP アドレスをホストが従来型のネットワークにおいて接続していたルータのインタフェースの IP アドレス，宛先 MAC アドレスをブロードキャストアドレスとする．ここで送信元 MAC アドレスを任意の固定アドレスとしている理由は，ARP Reply を応答させるために必要な情報が，宛先 MAC アドレス，送信元・宛先 IP アドレスのみであるためである．ARP Request の宛先 IP アドレスはホストが属すると判断したネットワークアドレスの全 IP アドレスとする．そのため，たとえばネットワークアドレスのプレフィックスが/24 の場合には，254 回パケットをマルチキャストすることになる．

ARP Request を受信したホストは，自身の IP アドレスと宛先アドレスが一致すれば，OpenFlow スイッチに ARP Reply を送信する．OpenFlow スイッチはパケット情報とフローエントリを比較しパケットを処理するが，提案システムを用いる場合，この時点で OpenFlow スイッチにフローエントリがないため，Packet-In メッセージが OpenFlow コントローラである提案システムに必ず送信される．Packet-In メッセージの中にはパケットを受信した OpenFlow スイッチのポート情報が存在するため，この情報を用いて，ホストと OpenFlow スイッチとの接続ポートを取得する．以上によりホストとの接続に使用されているルータのインタフェースと OpenFlow スイッチのポートの対応付けを行う．

本章で述べた処理により，ルータと OpenFlow スイッチの対応付け情報およびルータのインタフェースと各 OpenFlow スイッチのポートの対応付け情報が得られる．

4.4 設定情報変換機能

設定情報変換機能は，4.2 節で述べた設定情報取得機能および 4.3 節で述べた設定情報対応付け機能により得た情報に基づき，設定情報を OpenFlow ネットワークで適用可能な形式に変換するために用いられる．具体的には，提案システムは，本機能を用いることにより，OpenFlow スイッチごとに 4 つのフローテーブルを生成する．それぞれのフローテーブルは，従来型のネットワークの対応するルータにおける，インバウンド方向の ACL，経路表のうちホストとの直接接続経路，経路表のうち RIP により学習した経路，アウトバウンド方向の ACL に対応したフローエントリを管理するために用いる．以降，個々のルータにおいて，経路表の情報（ホストとの直接接続経路の情報と RIP により学習した経路情報）および ACL の情報を，それぞれ，対応する OpenFlow スイッチ上のフローエントリに変

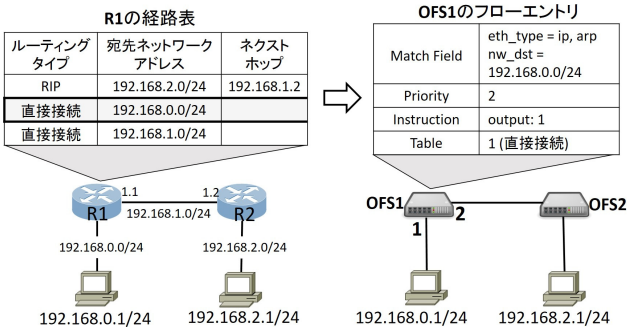

図 5 ホストとの直接接続経路の変換例

Fig. 5 Conversion of a directory connected entry to a flow entry.

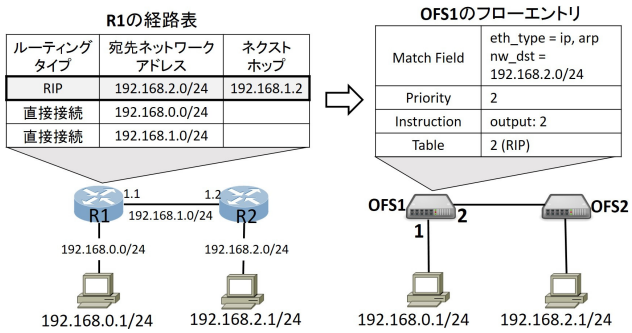

図 6 RIP 経路情報の変換例

Fig. 6 Conversion of a RIP routing entry to a flow entry.

換する手順を説明する．

4.4.1 経路表の情報の変換

設定情報取得機能により取得した経路表をフローエントリに変換する手順を述べる．なお，本稿における提案システムでは，経路表の情報をフローエントリに変換する際，経路表の各エントリに対して IP 用と ARP 用に 2 つのフローエントリを作成する．

まず，経路表の情報のうち，ホストとの直接接続経路の情報をフローエントリに変換する手順を述べる．**図 5** に，ルータ R1 と直接接続されたホストへの経路情報を OpenFlow スイッチ OFS1 のフローエントリに変換する動作例を示す．まず，ルータより取得した経路表の直接接続経路エントリのうちホストとの直接接続経路エントリが存在するかどうかを調べる．存在する場合，それぞれの経路エントリに対して，経路情報に含まれる宛先ネットワークアドレスをマッチフィールド，ホストに接続されている接続ポートを宛先の出力ポートとするフローエントリを作成する．図 5 の場合，経路表の上から 2 番目のエントリが該当する．なお，後述するように，フローテーブルにはプライオリティ 1 のフローエントリを別途作成するため，ここで作成するフローエントリのプライオリティは 2 に設定する．このようにして設定されたフローエントリの例を図 5 右に示す．

なお，これらホストとの直接接続経路に対応するフローエントリは，フローテーブル 1 として管理される．また，フローテーブル 1 のいずれのフローエントリにもマッチしなかった場合に，フローテーブル 2 のフローエントリとの比較を行う処理に遷移するよう，フローテーブル 1 中に，フローテーブル 2 に遷移するフローエントリをプライオリティ 1 で作成する．

次に，設定情報取得機能によりルータから取得した経路表の情報のうち，RIP により学習した経路情報をフローエントリに変換する手順を述べる．**図 6** に，ルータ R1 の RIP に関する経路情報を OpenFlow スイッチ OFS1 のフローエントリに変換する動作例を示す．この場合，経路表のうち，RIP により学習した経路エントリそれぞれに対して，経路情報に含まれる宛先ネットワークアドレスをマッチフィールド，経路情報のネクストホップ IP アドレスに対応する OpenFlow スイッチとの接続ポートを宛先の出力ポートとするフローエントリを作成する．図 6 の場合，経路表の上から 1 番目のエントリがこれに対応する．なお，後述のように，フローテーブル遷移用にプライオリティ 1 のフローエントリが作成されるため，ここで作成するフローエントリはプライオリティ 2 で作成する．このようにして設定されたフローエントリの例を図 6 右に示す．

なお，これら RIP により学習した経路に対応するフローエントリは，フローテーブル 2 として管理される．フローテーブル 1 と同様に，フローテーブル 2 においても，フローテーブル 3 に遷移するフローエントリをプライオリティ 1 で作成する．

本稿の提案システムでは，直接接続経路および RIP により学習した経路を移行の対象としているが，OSPF などの他のルーティングプロトコルに対応する場合の提案手法の拡張も容易である．その場合，新しいフローテーブルをアドミニストレーティブディスタンス値の昇順に参照することにより対応できる．たとえば，シングルエリア OSPF に対応する場合，RIP と同様の処理手順で新たなフローテーブルを作成する．直接接続経路，OSPF，RIP のアドミニストレーティブディスタンス値がそれぞれ，0，110，120 であるため，直接接続経路に関するフローテーブル，OSPF に関するフローテーブル，RIP に関するフローテーブルの順にフローテーブルを参照するようにフローエントリを作成する．このことにより，シングルエリア OSPF への対応が可能となる．マルチエリア OSPF に対応する場合は，エリア情報を考慮したフローエントリを作成することで対応できると考えられる．

4.4.2 ACL 設定情報の変換

ACL がルータに設定されていた場合には，その情報もフローエントリに変換する．

図 7 にインバウンド方向の ACL をフローエントリに変換する例を示す．インバウンド方向の ACL に対応するフローエントリはフローテーブル 0 で管理される．ルータの

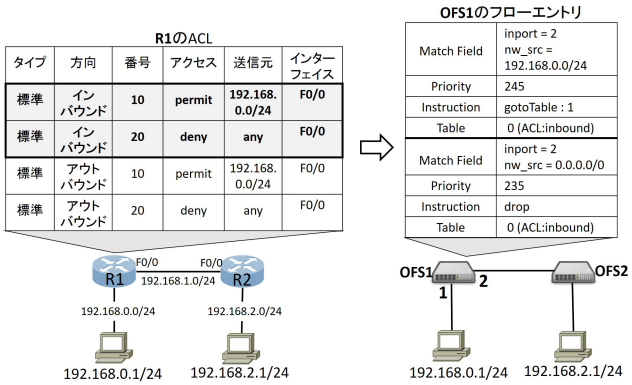

図 7 インバウンド方向 ACL の変換例

Fig. 7 Conversion of inbound ACL to a flow entry.

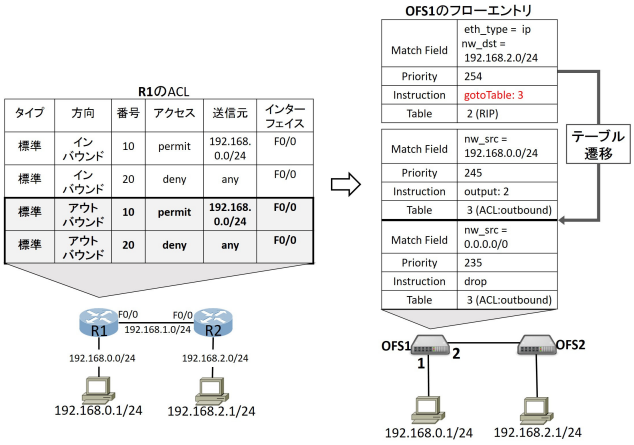

図 8 アウトバウンド方向 ACL の変換例

Fig. 8 Conversion of outbound ACL to a flow entry.

インタフェースにインバウンド方向の ACL が設定されている場合，それぞれの ACL に対して，該当インタフェースに対応する OpenFlow スイッチのポートを，フローエントリ中のマッチフィールドの inport に設定したフローエントリを作成する．ACL のアクセス条件が permit の場合にはフローテーブル 1 に遷移するようインストラクションを設定し，deny の場合にはパケットを破棄するようインストラクションを設定する．マッチフィールドには，ACL のルールを設定する．またプライオリティは，255 から ACL の処理順序を決定する番号の値を引いたものを設定する．このことにより，従来型のネットワークと同じ処理順序で ACL を適用できる．なお，フローテーブル 1，2 と同様に，フローテーブル 0 においても，フローテーブル 1 に遷移するフローエントリをプライオリティ 1 で作成する．このようにして設定されたフローエントリの例を図 7 右に示す．

図 8 にアウトバウンド方向の ACL をフローエントリに変換する例を示す．アウトバウンド方向の ACL に対応するフローエントリはフローテーブル 3 として管理される．ルータのインタフェースにアウトバウンド方向の ACL が設定されている場合，まず，経路表の情報から作成したフローエントリ（フローテーブル 1，フローテーブル 2 に含

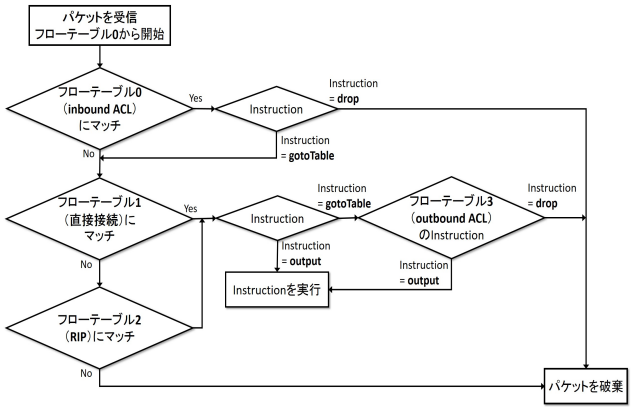

図 9 パケット処理のフローチャート

Fig. 9 Flowchart of packet processing.

まれるエントリ）の中から，該当するインタフェースに対応する OpenFlow スイッチのポートと同一のポートが出力ポートとして指定されているエントリを調べる．そして，それらのフローエントリのインストラクションを，フローテーブル 3 に遷移するよう変更する．ACL のアクセス条件が permit の場合には，もともとフローテーブル 1，2 で設定されていたインストラクションをフローテーブル 3 のフローエントリに設定する．ACL のアクセス条件が deny の場合は，パケットを破棄するようインストラクションを設定する．マッチフィールドとプライオリティはインバウンド方向の ACL と同様に設定する．このようにして設定されたフローエントリの例を図 8 右に示す．

4.5 設定情報反映機能

設定情報反映機能は 4.4 節で述べた設定情報変換機能で生成したフローテーブルを各 OpenFlow スイッチに反映させるために用いられる．設定情報変換機能を用いて 4 つのフローテーブルを生成することにより，各 OpenFlow スイッチにおいて図 9 に示すような手順でパケットが処理される．

従来型のネットワークにおけるルータは，パケットを受信すると，インバウンド方向の ACL に従った処理，経路表に従ったパケットのフォワーディング，アウトバウンド方向の ACL に従った処理の順に処理を行う．なお，ルータでは経路表に同じ宛先への経路が複数存在する場合，最小のアドミニストレーティブディスタンス値を持つ経路を選択するが，通常，直接接続経路のアドミニストレーティブディスタンス値が最も小さい．そのため，OpenFlow スイッチにおけるパケット処理においても，図 9 に示すように，インバウンド方向の ACL に対応するフローテーブル 0，ホストとの直接接続経路に対応するフローテーブル 1，RIP により学習した経路に対応するフローテーブル 2，アウトバウンド方向の ACL に対応するフローテーブル 3 の順に処理を行う．この手順でパケットが処理されること

により，従来型のネットワークにおけるルータと同等のパケット処理を OpenFlow スイッチで再現できる．

4.6 動作検証機能

動作検証機能は，OpenFlow ネットワークの動作検証環境を提供し，環境の構築には，Mininet を利用する．

検証を行う場合，利用者は，まず，図 3 に示すシステム GUI 上にホストを追加し，追加したホストと OpenFlow スイッチを結線する．ホストには，IP アドレスとホストが接続する OpenFlow スイッチのポートを設定する．次に，利用者は GUI を用いて Mininet のトポロジファイルを出力する．なお，このファイルには，ホストの IP アドレス，OpenFlow スイッチの Datapath ID，ホストと OpenFlow スイッチの結線関係などが記述されている．その後，本トポロジファイルを用いて，Mininet 上に GUI 上で構築したネットワークと同一構成の仮想 OpenFlow ネットワークを構築する．これを使用し，利用者は，実際の OpenFlow ネットワークを用いた動作検証を行う前に，移行後の OpenFlow ネットワークを仮想的に検証できる．

5. 検証および考察

本章では，開発したシステムの動作検証およびフローエントリ数に関する評価を行う．

5.1 動作検証

本節では，開発したシステムの動作検証および評価を行う．まず，本システムを用いることにより，従来型のネットワークと同等の OpenFlow ネットワークを構築できるかについて動作検証を行った．**図 10** に示す 8 種類のトポロジを対象として，本システムを用いてそれぞれのトポロジに対し OpenFlow ネットワークへの移行を行った．なお，従来型のネットワークは Cisco 社製のルータ（型番：Cisco1921，OS：c1900）およびノート PC（CPU：1.86 GHz Intel Core 2 Duo, Mem.：4 GB, OS：OS X v10.9 Mavericks 64 bit）を用いて構築し，提案システムが動作するサーバとしては，PC（CPU：Core i5-4460 3.20 GHz*2，Mem.：4 GB，OS：Ubuntu14.04 64 bit）を用いた．OpenFlow ネットワークの動作検証には，本システムの動作検証機能を用い，Mininet 上で検証を行った．その結果，すべてのトポロジについて，従来型のネットワークと仮想 OpenFlow ネットワークの動作が同一であることを確認した．

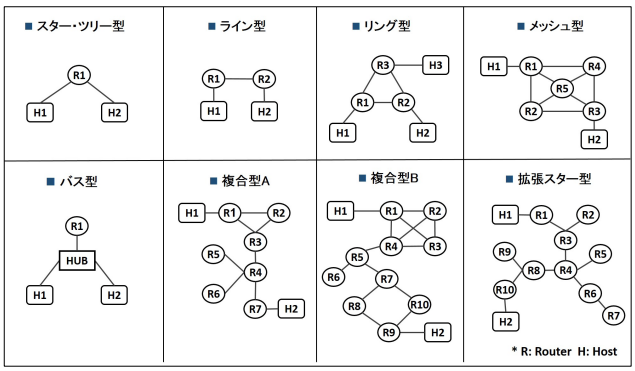

図 10 動作検証に使用したトポロジ
Fig. 10 Topologies for evaluation.

ここで，誌面の都合上，動作検証を行った 8 種類のトポロジのうち，最も複雑な複合型トポロジ B（**図 11**）について，実験で使用したルータの設定情報と，それを変換したフローエントリの比較検証結果の一部を示す．複合型トポロジ B は RIP と ACL を用いて構築している．実験で使用したルータ R1，R9 の設定情報の一部を**図 12**，**図 13** に示す．ルータ R1 には IP 通信を許可するが，ホスト 1 からホスト 2 に対する HTTP 通信は許可しない拡張 ACL がインバウンド方向に設定されている．一方，ルータ R9 には標準 ACL がアウトバウンド方向に設定されている．このような複合型トポロジ B の設定情報を提案システムにより変換したフローエントリのうち，OpenFlow スイッチ OFS1，OFS9 のフローエントリを**図 14**，**図 15** に示す．

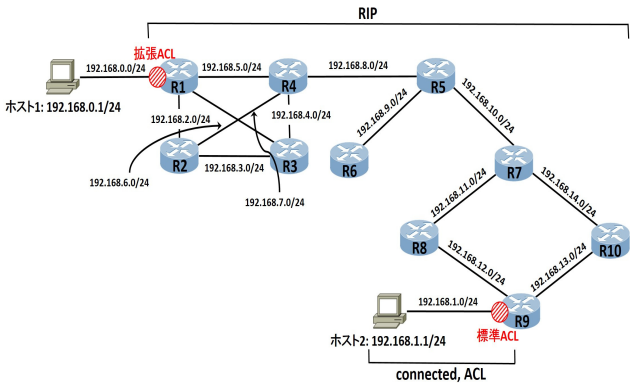

図 11 複合型トポロジ B の詳細図
Fig. 11 Details of complex network topology B.

```
<router name="R1">
 <interfaces>
  <interface address="192.168.0.254" name="GigabitEthernet0/0" prefix="24"
             direction="Inbound" acl="http"/>
  <interface address="192.168.2.1" name="GigabitEthernet0/1" prefix="24"/>
  <interface address="192.168.5.2" name="Serial0/1/0" prefix="24"/>
  <interface address="192.168.7.1" name="Serial0/1/1" prefix="24"/>
 </interfaces>
 <dynamic_routing>
  <RIPv2>
   <network address="192.168.1.0" outInterface="Serial0/1/0" nexthop="192.168.5.1"
            metric="5" prefix="24"/>
                              (省略)                                  RIP
   <network address="192.168.14.0" outInterface="Serial0/1/0" nexthop="192.168.5.1"
            metric="3" prefix="24"/>
  </RIPv2>
 </dynamic_routing>
 <directly_connected>
  <Connected>
   <network address="192.168.0.0" outInterface="GigabitEthernet0/0" prefix="24"/>
   <network address="192.168.2.0" outInterface="GigabitEthernet0/1" prefix="24"/>
   <network address="192.168.5.0" outInterface="Serial0/1/0" prefix="24"/>      直接接続
   <network address="192.168.7.0" outInterface="Serial0/1/1" prefix="24"/>
  </Connected>
 </directly_connected>
 <Access-Lists>
  <Extended-Named>
   <identifier i="http">
    <rule access="deny" number="10" format="tcp" src="192.168.0.0" src_wildcard="0.0.0.255"
          dst="192.168.1.0" dst_wildcard="0.0.0.255" option="eq" protocol="www"/>
    <rule access="permit" number="20" format="ip" src="any" dst="any/>          拡張ACL
   </identifier>
  </Extended-Named>
 </Access-Lists>
</router>
```

図 12 複合型トポロジ B のルータ R1 設定情報
Fig. 12 Router R1 settings of complex network topology B.

```
<router name="R9">
 <interfaces>
  <interface address="192.168.1.254" name="GigabitEthernet0/0" prefix="24"
             direction="Outgoing" acl="1"/>
  <interface address="192.168.12.2" name="Serial0/1/0" prefix="24"/>
  <interface address="192.168.13.1" name="Serial0/1/1" prefix="24"/>
 </interfaces>
 <dynamic_routing>
  <RIPv2>
   <network address="192.168.0.0" outInterface="Serial0/1/0" nexthop="192.168.12.1"
            metric="5" prefix="24"/>
                                  （省略）                                  RIP
   <network address="192.168.14.0" outInterface="GigabitEthernet0/0" nexthop=
            "192.168.13.2" metric="1" prefix="24"/>
  </RIPv2>
 </dynamic_routing>
 <directly_connected>
  <Connected>
   <network address="192.168.1.0" outInterface="GigabitEthernet0/0" prefix="24"/>
   <network address="192.168.12.0" outInterface="Serial0/1/0" prefix="24"/>   直接接続
   <network address="192.168.13.0" outInterface="Serial0/1/1" prefix="24"/>
  </Connected>
 </directly_connected>
 <Access-Lists>
  <Standard>
   <identifier i="1">
    <rule access="deny" number="10" source="192.168.2.0" wildcard="0.0.0.255"/>
    <rule access="permit" number="20" source="any"/>                        標準ACL
   </identifier>
  </Standard>
 </Access-Lists>
</router>
```

図 13 複合型トポロジ B のルータ R9 設定情報
Fig. 13 Router R9 settings of complex network topology B.

```
root@nomura:~# ovs-ofctl dump-flows s1 table=2 -O OpenFlow13
table=2, priority=2, arp, arp_tpa=192.168.1.0/24 actions=output:1
table=2, priority=2, ip, nw_dst=192.168.1.0/24 actions=output:1
                （省略：経路表のRIPエントリに対応するフローエントリ）                RIP
table=2, priority=2, arp, arp_tpa=192.168.14.0/24 actions=output:1
table=2, priority=2, ip, nw_dst=192.168.14.0/24 actions=output:1
table=2, priority=1 actions=goto_table:3
table=2,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s1 table=1 -O OpenFlow13
table=1, priority=2, arp, arp_tpa=192.168.0.0/24 actions=output:2
table=1, priority=2, ip, nw_dst=192.168.0.0/24 actions=output:2
              （省略：経路表の直接接続経路に対応するフローエントリ）              直接接続
table=1, priority=2, arp, arp_tpa=192.168.6.0/24 actions=output:3
table=1, priority=2, ip, nw_dst=192.168.6.0/24 actions=output:3
table=1, priority=1 actions=goto_table:2
table=1,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s1 table=0 -O OpenFlow13
table=0, priority=245, tcp, in_port=2, nw_src=192.168.0.0/24,
         nw_dst=192.168.1.0/24, tp_dst=80 actions=drop                      拡張ACL
table=0, priority=235, icmp, in_port=2 actions=goto_table:1
table=0, priority=1 actions=goto_table:1
table=0,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s1 table=3 -O OpenFlow13
table=3, priority=0 actions=CONTROLLER: 65535
```

図 14 ルータ R1 の設定情報を変換した OFS1 のフローエントリ
Fig. 14 Flow entries of OFS1 (corresponding to R1).

ルータの設定情報が OpenFlow スイッチのフローエントリに正しく変換されていることを確認するために，従来型のネットワーク上の全ホスト間および Mininet 上の全ホスト間で ping によるパケットの到達性，ACL の適用状況を確認した．ACL の適用状況は，ホストにおいて HTTP サーバの 80 番ポートでの起動および wget による HTTP アクセスを行うことにより確認した．今回，従来型のネットワークにおいてホスト 1 からホスト 2 に対する HTTP アクセスを拒否する ACL が設定されている．ホスト 1 からホスト 2 に対して wget による HTTP アクセスを行ったところ，従来型のネットワーク，Mininet 上の OpenFlow ネットワークいずれにおいても ICMP による通信は行えたが，HTTP による通信は行えなかった．一方，ホスト 2 からホスト 1 に対しては，ICMP による通信，HTTP による通信を行えた．以上より，本システムを用いることで，従

```
root@nomura:~# ovs-ofctl dump-flows s9 table=2 -O OpenFlow13
table=2, priority=2, ip, nw_dst=192.168.0.0/24 actions=output:1
table=2, priority=2, arp, arp_tpa=192.168.0.0/24 actions=output:1
                （省略：経路表のRIPエントリに対応するフローエントリ）                RIP
table=2, priority=2, arp, arp_tpa=192.168.14.0/24 actions=output:1
table=2, priority=2, ip, nw_dst=192.168.14.0/24 actions=output:1
table=2, priority=1 actions=goto_table:3
table=2,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s9 table=1 -O OpenFlow13
table=1, priority=2, arp, arp_tpa=192.168.1.0/24 actions=output:2
table=1, priority=2, ip, nw_dst=192.168.1.0/24 actions=goto_table:3
              （省略：経路表の直接接続経路に対応するフローエントリ）              直接接続
table=1, priority=2, arp, arp_tpa=192.168.13.0/24 actions=output:1
table=1, priority=2, ip, nw_dst=192.168.13.0/24 actions=output:1
table=1, priority=1 actions=goto_table:2
table=1,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s9 table=3 -O OpenFlow13
table=3, priority=245, ip, nw_src=192.168.2.0/24 actions=drop
table=3, priority=235, ip actions=output:2                              標準ACL
table=3,  priority=0 actions=CONTROLLER: 65535

root@nomura:~# ovs-ofctl dump-flows s9 table=0 -O OpenFlow13
table=0, priority=1 actions=goto_table:1
table=0, priority=0 actions=CONTROLLER: 65535
```

図 15 ルータ R9 の設定情報を変換した OFS9 のフローエントリ
Fig. 15 Flow entries of OFS9 (corresponding to R9).

来型のネットワークと同等の OpenFlow ネットワークを構築できることを確認した．なお，他のすべてのトポロジについても同様の確認を行った．

次に，設定情報取得機能を用いてルータの設定情報取得を開始してから，動作検証機能の仮想 OpenFlow ネットワークにおける全ホスト間で通信が可能になるまでの時間（移行時間）を計測した．その結果，いずれのトポロジの場合においても移行時間は 6 分以内であった．これらの結果から，本システムを用いることにより短時間で従来型のネットワークから OpenFlow ネットワークへ移行できるといえる．

5.2 フローエントリ数に関する評価と考察

次に，開発したシステムを用いて変換したフローエントリ数に関する考察を行う．今，あるルータの経路表のエントリ数を N，ACL のルール数を L とし，このルータを OpenFlow スイッチに移行する場合を考える．この場合，まず，4.4.1 項で述べたように経路表から IP と ARP パケット処理のために 2N 個のフローエントリが作成される．また，4.4.2 項で述べたように ACL の設定情報から L 個のフローエントリが作成される．さらに，4.4 節で述べたようにフローテーブルを順に遷移するために，フローテーブル 0 からフローテーブル 2 にそれぞれ遷移用のフローエントリが作成される（合計 3 個）．加えて，OpenFlow 1.3 では，Packet-In メッセージを OpenFlow コントローラに送信するフローエントリがあらかじめプライオリティ 0 ですべてのフローテーブルに書き込まれている（合計 4 個）．したがって，移行後の OpenFlow スイッチにおけるフローエントリ数は合計で 2N＋L＋7 となる．すなわち，経路表のエントリ数と ACL のルール数に比例してフローエントリ数

が増える．

フローエントリ数に関する検証を行うため，図 10 のすべてのトポロジに対して，移行後の OpenFlow スイッチのフローエントリ数を計測した．その結果，2N＋L＋7 で計算される数のフローエントリが作成されていることを確認した．たとえば，経路表のエントリ数および ACL のルール数が最も多い複合型トポロジ B のルータ R1 は，経路表のエントリ数 N が 15，ACL のルール数 L が 2 であり，対応する OFS1 のフローエントリ数は 39 であった．なお，今回の検証でフローエントリ数の最も少ない OpenFlow スイッチのフローエントリ数は 13 であった．

本稿では，提案システムを用いることにより移行が行えることを示すため，比較的小規模かつ単純なネットワークにおける動作検証を行った．今後，具体的な移行シナリオをいくつか想定し，提案システムを用いて移行を行う場合の移行時間，フローエントリ数，OpenFlow スイッチやコントローラの負荷などの詳細な性能を評価する予定である．

6. おわりに

本稿では，従来型のネットワークから OpenFlow ネットワークへの移行を支援することを目的とし，OpenFlow ネットワーク移行支援システムを開発した．提案システムは，従来型のネットワーク上のルータから設定情報を取得し，その設定情報を OpenFlow ネットワークへ反映させる．また，仮想 OpenFlow ネットワークを用いて，構築した OpenFlow ネットワークの検証を行える．本稿では，提案システムの動作検証およびフローエントリ数に関する評価を行った．その結果，最大 10 台のルータから構成される従来型のネットワークと同じパケット処理を行う OpenFlow ネットワークを 6 分以内で構築できた．また，本稿で行った実験のトポロジの場合，OpneFlow スイッチ 1 台あたりのフローエントリ数は最大で 39 個であった．

今後の課題として，従来型のネットワークの一部分を OpenFlow ネットワークへ移行する場合など，さまざまな移行シナリオにおける提案システムの動作検証や，OpenFlow スイッチ実機を用いて構築した実 OpenFlow ネットワーク環境での提案システムの動作検証があげられる．また，L2 スイッチや経路制御機能の移行への対応などの提案システムの拡張があげられる．

参考文献

[1] Nunes, B.A.A., Mendonca, M., Nguyen, X., Obraczka, K. and Turletti, T.: A Survery of Software-Defined Networking: Past, Present, and Future of Programmable Networks, *IEEE Communications Surveys & Turtorials*, Vol.16, No.3, pp.1617–1634 (2014).

[2] Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S. and Uhlig, S.: Software-Defined Networking: A Comprehensive Survey, *Proc. IEEE*, Vol.103, No.1, pp.14–76 (2015).

[3] Blenk, A., Basta, A., Reisslein, M. and Kellerer, W.: Survey on Network Virtualization Hypervisors for Software Defined Networking, *IEEE Communications Surveys & Tutorials*, Vol.18, No.1, pp.655–685 (2016).

[4] Li, Y. and Chen, M.: Software-Defined Network Function Virtualization: A Survey, *IEEE Access*, Vol.3, pp.2542–2553 (2015).

[5] Xia, W., Wen, Y., Foh, C.H., Niyato, D. and Xie, H.: A Survey on Software-Defined Networking, *IEEE Communications Surveys & Tutorials*, Vol.17, No.1, pp.27–51 (2015).

[6] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. and Turner, J.: OpenFlow: Enabling Innovation in Campus Networks, *ACM SIGCOMM Computer Communication Review*, Vol.38, No.2, pp.69–74 (2008).

[7] 事例にみる SDN を用いたネットワーク仮想化のメリット (2014), 入手先 ⟨http://jpn.nec.com/univerge/pflow/pdf/iexpokansai_2014_seminar2f_benefits_of_network_virtualization.pdf⟩（参照 2016-09-12）.

[8] OPEN NETWORKING FOUNDATION: Migration Use Cases and Methods (2014), available from ⟨https://www.opennetworking.org/images/stories/downloads/sdn-resources/use-cases/Migration-WG-Use-Cases.pdf⟩ (accessed 2016-06-17).

[9] Shenker, S., Casado, M., Koponen, T., McKeown, N., et al.: The Future of Networking, and the Past of Protocols (2011), available from ⟨http://opennetsummit.org/archives/oct11/shenker-tue.pdf⟩ (accessed 2016-06-17).

[10] Levin, D., Canini, M., Schmid, S., Schaffert, F. and Feldmann, A.: Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks, *Proc. 2014 USENIX Annual Technical Conference*, pp.333–345 (2014).

[11] Network Configuration Protocol, available from ⟨http://tools.ietf.org/html/rfc6241⟩ (accessed 2016-06-17).

[12] Mininet, available from ⟨http://mininet.org/⟩ (accessed 2016-06-17).

[13] Caldwell, D., Gilbert, A., Gottlieb, J., Greenberg, A., Hjalmtysson, G. and Rexford, J.: The Cutting EDGE of IP Router Configuration, *ACM SIGCOMM Computer Communication Review*, Vol.34, No.1, pp.21–26 (2004).

[14] Nelson, T., Ferguson, A.D., Fonseca, D.Y.R. and Krishnamurthi, S.: Exodus: Toward Automatic Migration of Enterprise Netwrok Configurations to SDNs, *Proc. 1st ACM SICOMM Symposium on Software Defined Netwroking Researh*, No.13, pp.1–7 (2015).

[15] Nelson, T., Ferguson, A.D., Scheer, M.J.G. and Krishnamurthi, S.: Tierless Programming and Reasoning for Software-Defined Networks, *Proc. 11th USENIX Conference on Networked Systems Design and Impelementation*, pp.519–531 (2014).

[16] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S. and Vahdat, A.: B4: Experience with a Globally-Deployed Software Defined WAN, *Proc. ACM SIGCOMM 2013*, pp.3–14 (2013).

野村 圭太 （学生会員）

2015 年近畿大学理工学部卒業．同年同大学大学院総合理工学研究科博士前期課程入学，現在に至る．OpenFlow ネットワークへの移行支援に関する研究に従事．

谷口 義明 （正会員）

2008 年大阪大学大学院情報科学研究科博士後期課程修了，博士（情報科学）．2014 年より近畿大学理工学部講師となり，現在に至る．センサネットワーク，無線ネットワークに関する研究に従事．電子情報通信学会，画像電子学会，IEEE 各会員．本会シニア会員．

井口 信和 （正会員）

1988 年三重大学大学院修士課程修了．同年（株）豊田自動織機製作所入社．1992 年和歌山県工業技術センター研究員．2001 年大阪大学大学院基礎工学研究科博士後期課程修了，博士（工学）．2002 年近畿大学理工学部情報学科助教授．2008 年同大学教授となり，現在に至る．近畿大学総合情報基盤センター長を兼務．インターネット応用，学習支援システム，農業 ICT に関する研究に従事．電子情報通信学会，農業情報学会，教育システム情報学会，AACE，IEEE 各会員．

渡辺 健次 （正会員）

1987 年佐賀大学理工学部物理学科卒業．1989 年同大学大学院理工学研究科物理学専攻修士課程修了．同年同大学情報処理センター助手．1993 年和歌山大学経済学部産業工学科講師．1996 年同大学システム工学部情報通信システム学科講師．1998 年同助教授．1999 年佐賀大学理工学部知能情報システム学科助教授．2006 年同教授．2010 年同大学大学院工学系研究科知能情報システム学専攻教授．2012 年広島大学大学院教育学研究科技術・情報教育学講座教授，現在に至る．広島大学附属福山中学校・高等学校校長を兼務．学習支援システム，インターネット応用，分散システム運用技術の研究に従事．博士（工学）．電子情報通信学会，人工知能学会，教育システム情報学会，日本教育工学会，日本産業技術教育学会，IEEE 各会員．