2Y-7

SDNを構成するネットワークサービス間の連携を実現する方式に関する研究

高橋基† 中島潤‡ 広奥暢§

北海道情報大学大学院 経営情報学研究科†

北海道情報大学 経営情報学部‡ 北海道情報大学 情報メディア学部§

1 はじめに

OpenFlow を筆頭に SDN（Software-Defined Networking）のコンセプトが注目されているが，ONF(Open Networking Foundation)が提唱する SDN アーキテクチャ[1]には大きく２つの API が設けられ，SDN コントローラが物理スイッチ及び仮想スイッチと盛業情報を交換するための API を，Southbound API，SDN コントローラプラットフォームが SDN 上に存在しているアプリケーションにネットワークの制御情報を提供するための API を，Northbound API と名付けている．Southbound API には，ONF が標準化を行う OpenFlow がある一方で，Northbound API は標準化の目処が立っておらず，各ベンダや各団体が独自の Northbound API を設けている現状がある．

既存の Northbound API の多くは，REST などの Web 技術を用いた API で，コントロールプレーンがアプリケーションにネットワークの制御情報を提供する目的で設計されている[2]．しかし，IDS のようにデータプレーンで発生したイベントをアプリケーションに通知するといったサービスでは，SDN コントローラからアプリケーションへリアルタイムに通知を行うべきであると考えるが，このようなリアルタイム性を必要とするサービスを構築するためには，アプリケーションレイヤとコントロールレイヤ間で双方向通信する必要があり，既存の SDN フレームワークでは容易に実現できない．

本研究では，双方向通信を容易に実現するための SDN フレームワークについて検討し，その実現方式を提案する．また，提案手法を用いて試作システムの開発を行ない，提案の有効性を確認した．

Studies on the method to realize the cooperation between network services and control plane of SDN
† Motoi Takahashi
‡ Jun Nakajima
§ Hirohku Tohru
Graduate School of Business Administration and Information Science, Hokkaido Information University（†）
Faculty of Business Administration and Information Science, Hokkaido Information University（‡）
Faculty of Information Media, Hokkaido Information University（§）

2 SDNフレームワークの要件と提案

SDN 上で動作するネットワークサービス間の連携を想定し，SDN フレームワークの要件を整理し，本研究の SDN フレームワークを提案する．

2．1 双方向通信の確保

既存の Northbound API で利用されている HTTP プロトコルはコネクションレス型プロトコルである．コネクションレス型プロトコルでは，クライアントからのリクエストに対してレスポンスを返すタイミングは限られており，サーバ側から任意のタイミングでクライアントへリクエストを送信することは容易ではない．

そこで本研究では，双方向通信を実現するために，コネクション型プロトコルを利用して，常にアプリケーション間でコネクションを維持し，お互いに任意のタイミングで情報交換することを提案する．

2．2 通信プロトコルの氾濫抑制

新たなネットワークサービスの導入を考えたとき，多数のベンダや開発者がアプライアンス、又はソフトウェア製品を提供しており，これを利用するネットワーク管理者は組織のポリシーに従って，最も適したアプリケーションを選択する．つまり，同様の機能を提供するサービスであっても，複数のアプリケーションが存在している．しかし，これらを１つのネットワークサービスとして捉え，抽象化することができると考えた．また，同様のサービスを提供するアプリケーション同士であっても，異種のアプリケーション同士が情報交換するためには，同一のプロトコルを利用する必要がある．

そこで本研究では，ネットワークサービスごとにプロトコルを定義することで，アプリケーションを抽象化することを提案する．ネットワークサービスごとに抽象化したプロトコルを定義することで，新たなアプリケーションの導入やネットワークサービスの機能に変更が生じても，他のネットワークサービスのプロトコルに影響を及ぼすことなく，プロトコルの追加・拡

張が可能であると考えた.

3　提案の有効性の確認

3．1　SDN フレームワークの実装

提案の有効性を確認するために，２章で述べた要件を満たす SDN フレームワークを設計し，動作検証のために試作システムを構築した．システム構成として，情報交換を行うためのメッセージ配信サーバと，これを利用するクライアントを配置する．連携のためのアプリケーション間の情報交換はメッセージ配信サーバを中継して行ない，連携を取るアプリケーションにはクライアント実装を組み込む必要がある.

メッセージ配信サーバの実装は，ネットワーク上で連携を行うアプリケーションの情報を一箇所に集約するためにクライアント・サーバモデルで実装を行なった．サーバの役割として，SDN 上に存在しているアプリケーションの情報を集約し，アプリケーション同士がお互いを発見しあう空間の提供と，アプリケーション間で情報交換するためのメッセージ配信を行う，連携のためのネットワーク中心ノードとなる.

また，メッセージ配信サーバと通信し，他のアプリケーションと連携を取るためにメッセージを送信するクライアント部の実装を行なった．メッセージ配信サーバ上に送信するメッセージとして，アプリケーションを抽象化したプロトコルを用いて情報交換する.

3．2　試作システムの構築

提案手法を用いた SDN フレームワークを利用した試作システムとして，仮想ネットワークタップ装置を開発し，提案の有効性の確認を行なった（図 1）．従来のネットワークタップ装置ではポートミラーリングによってトラフィックのタッピングを行うため，監視対象となるポートを流れる全てのトラフィックが複製されるものであった．試作システムで構築する仮想ネットワークタップ装置は，フィルタリングルールにしたがって特定のデータのみを対象にトラフィックをタッピングすることができる.

仮想ネットワークタップ装置の機能は OpenFlow コントローラとして実装し，Trema を用いて開発した．OpenFlow スイッチには Hewlett-Packard 社の HP 3500-24 Switch を利用し，実機環境で動作検証を行なった．また，連携を行うクライアント，メッセージ配信サーバは Ruby 言語で開発した.

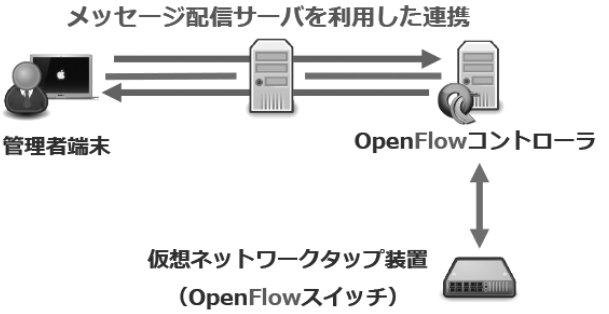

図 1．試作システムの構成

3．3　検証結果

アプリケーション間が双方向に通信可能であることを確認するため，最低限２つの検証シナリオが必要である．システムの利用シナリオとして，フィルタリングルールの設定変更を行うための，管理者端末からタップ装置への通信と，タップ装置内で発生した障害を管理者端末へ通知するための逆方向の通信を，シナリオとして設定し，動作検証を行なった.

検証を行なった結果，提案した SDN フレームワークは２つの利用シナリオに耐え，アプリケーション間で双方向に連携を行うための手法として有効であることを確認した.

4　まとめ

本研究では，既存の SDN フレームワークでは実現が容易でなかった双方向の連携を実現する SDN フレームワークを検討し，提案手法を用いた SDN フレームワークを利用し試作システムを構築することで，提案の有効性を確認した．提案では，既存の SDN アーキテクチャでは実現が用意ではない双方向通信を，コネクション型プロトコルを用いることで実現した．また，ネットワークサービスごとにアプリケーションを抽象化することで，ネットワークサービス間の連携を実現可能であることを確認した.

参考文献

[1] Software-Defined Networking: The New Norm for Networks: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[2] Technical Overview ― OpenDaylight: http://www.opendaylight.org/project/technical-overview