10'00"00 インターネット10分講座

MIN SEC 1/100S INTERNET

OpenFlow

今回のインターネット10分講座では、ネットワークの制御方法を管理者自らが自由に設計することができる、OpenFlow技術について解説します。

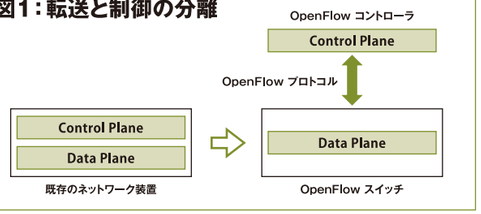
1 はじめに

現在広く使用されているネットワーク機器は、自律的に収集した情報を元にして、パケットを転送します。例えばL2スイッチは、パケット転送に、受信したパケットから学習したMACアドレスを用います。また、ルータはOSPFやBGPといった、経路制御プロトコルを用いて取得した経路情報を用いています。このような自律的動作のための制御機能が、ネットワーク機器にはそれぞれ搭載されています。ネットワーク管理者は、ベンダーによりあらかじめ用意されている範囲で、この制御機能を使用するしかありませんでした。

一方、ネットワークサービス事業者は、ネットワーク技術の複雑化に加え、事業環境の急激な変化や、ユーザーニーズの多様化といった課題に直面しています。この課題解決のために、ソフトウェアのように柔軟に制御可能なネットワーク技術（Software Defined Network）が求められています。

これらの動きの中で注目されている技術の一つが、OpenFlow技術です。制御機能と転送機能が共存する既存のネットワーク機器とは異なり、OpenFlowでは、制御部と転送部を分離したアーキテクチャを採用しています（**図1**）。ネットワーク管理者は、OpenFlowコントローラと呼ばれる制御部を、自ら設計・実装することで、必要な制御機能を自由に実現することができます。

図1：転送と制御の分離



本稿では、このOpenFlow技術を解説します。

2 OpenFlow概要

OpenFlow 1.0の仕様[※1]で規定されている内容を説明します。

2.1 OpenFlowスイッチ

OpenFlowスイッチは、Datapath IDと呼ばれる64bitの識別子を持ちます。コントローラ側から識別するために、ネットワーク中の各OpenFlowスイッチには、それぞれユニークなDatapath IDが割り当てられている必要があります。

OpenFlowスイッチは、条件（ヘッダフィールド）にマッチしたパケットに対して、アクションで指定された動作を行います。ヘッダフィールド、アクションおよび統計情報の組をフローと呼びます。このフローは、OpenFlowコントローラにより生成され、後述するOpenFlowプロトコルを使って、OpenFlowスイッチへと送られます。送られたフローはスイッチ中のフローテーブルに格納されます。

ヘッダフィールドとは、**表1**に示す12種類のフィールドで、受信したパケットを識別するための条件として用いられます。フロー識別に使用しないフィールドにはワイルドカードを指定することで、任意のフィールドのみを条件として用いることができます。例えば、“受信ポート番号が1であり、かつ宛先MACアドレスがFF:FF:FF:FF:FF:FF”や“IPパケットであり、その送信元IPアドレスが192.168.1.1”といった条件を指定することができます。

表1 ヘッダフィールド

フィールド	説明
Ingress port	受信ポート
Ethernet src address	送信元 MAC アドレス
Ethernet dst address	宛先 MAC アドレス
Ethernet type	プロトコル種別
VLAN id	VLAN ID
VLAN priority	VLAN PCP 値
IP src address	IP 送信元アドレス
IP dst address	IP 宛先アドレス
IP protocol number	プロトコル番号
IP ToS bits	ToS 値
Transport src port	送信元ポート番号
Transport dst port	宛先ポート番号

アクションは、ヘッダフィールドにマッチしたパケットの処理を指定します。仕様で規定されているアクションを**表2**に示します。一つのフローに対して、複数のアクションを指定することも可能です。例えば、“宛先IPアドレスを書き換えた上での指定のポートからの出力”や、“指定のポートから出力させた後、さらに別のポートからも出力”といった動作を指定できます。

表2 アクション

アクション	説明
Forward	パケットを転送する
Enqueue	指定のキューに入れる
Drop	パケットを破棄する
Modify-Field	指定のフィールドを書き換える

Forwardアクションには、出力先のポート番号が指定されます。OpenFlowスイッチが実際に持つポートの他に、特殊な用途に用いるための論理ポートが定義されています（**表3**）。

表3 論理ポート

論理ポート	説明
ALL	受信ポート以外のすべてのポートから出力
CONTROLLER	Packet In として、コントローラへと送信
TABLE	フローテーブルを参照してアクションを決定（Packet Out メッセージでのみ使用される）
IN_PORT	受信ポートから出力
NORMAL	従来の L2 スイッチ機能を使って転送
FLOOD	スパニングツリーに沿って転送

Modify-Fieldアクションでは、**表4**に示すフィールドの書き換えが定義されています。ルータ機能を実現する際に必要となるMACアドレスの書き換えや、NAT機能を実現する際のIPアドレス、TCP/UDPポート番号の書き換えなどを行うことができます。

表4 Modify-Field アクション

アクション	説明
SET_VLAN_VID	VLANヘッダ中のVLAN IDを書換（VLANヘッダがない場合、新たに付与）
SET_VLAN_PCP	VLANヘッダ中のPriorityを書換（VLANヘッダがない場合、新たに付与）
STRIP_VLAN	VLANヘッダを除去
SET_DL_SRC	送信元 MAC アドレスを書換
SET_DL_DST	宛先 MAC アドレスを書換
SET_NW_SRC	送信元 IPv4 アドレスを書換
SET_NW_DST	宛先 IPv4 アドレスを書換
SET_NW_TOS	IPv4 ToS フィールドを書換
SET_TP_SRC	送信元 TCP／UDP ポート番号を書換
SET_TP_DST	宛先 TCP／UDP ポート番号を書換

2.2 OpenFlow プロトコル

図1で示した通り、OpenFlowスイッチは、OpenFlowコントローラから制御されます。OpenFlowコントローラとスイッチとの間は、TCPまたはTLS（Transport Layer Security）を用いて接続されます。コントローラとスイッチ間の接続のために、別途レガシーネットワークを用意して、その上でTCP/TLSコネクションを張る構成が一般的です。どちらからコネクションの開設を行うかは仕様中に規定されていませんが、現状ではスイッチ側から行う実装が一般的です。

このTCP/TLSコネクションを用いて、コントローラとスイッチ間の通信に用いられるのがOpenFlowプロトコルです。OpenFlowプロトコルには、**表5**に示すメッセージが定義されています。この中で主要なメッセージについて、次に説明を行います。

- Packet Inメッセージ：フローテーブル中にマッチするフローがなかった場合、受信パケットをコントローラへと送るために用いられます。このメッセージで送られるパケットを元にフローを作成するなど、OpenFlow特有の処理を実現するために欠かせないメッセージです。

- Packet Outメッセージ：Packet Inでコントローラに送られてきたパケットを本来の宛先に送るために、スイッチ側へと送り返す際に用いられます。また、コントローラが独自に作ったパケットを出力させる際にも用いられます。

- Flow Modメッセージ：コントローラが作ったフローをスイッチへと送るために用いられます。その際に、フローを設定してから一定時間後に消すためのハードタイムアウトと、フローが参照されない時間が一定時間経過した後に消すためのアイドルタイムアウトの、2種類のタイムアウト値を設定できます。

- Flow Removedメッセージ：何らかの理由でフローテーブルからフローが消された時に、コントローラへの通知のために用いられます。その際に、そのフローの参照回数や生存時間などの統計情報が併せて送られます。

- Barrier Request/Replyメッセージ：スイッチは、Flow ModやPacket Out等のメッセージ受信後に、それぞれの処理完了をコントローラへ報告しません。コントローラ側で処理完了を知る必要がある場合、Barrier Requestメッセージを送る必要があります。スイッチは、このメッセージ受信以前のすべての処理完了後に、Barrier Replyメッセージをコントローラへと送ります。

表5 OpenFlow メッセージ

Message	Description
コントローラ側から送られるメッセージ	
Packet Out	スイッチから出力させるためのパケットを送るためのメッセージ
Flow Mod	生成したフローをスイッチへと送る際に用いられるメッセージ
Port Mod	ポートの状態を変更するためのメッセージ
Set Config	スイッチのコンフィグパラメータを設定するためのメッセージ
コントローラ側から要求が送られ、スイッチ側で応答するメッセージ	
Features Request／Reply	スイッチのケイパビリティ取得に用いられるメッセージ
Stats Request／Reply	統計情報取得に用いられるメッセージ
Get Config Request／Reply	スイッチのコンフィグパラメータの取得に用いられるメッセージ
Barrier Request／Reply	コントローラ側の要求に対する処理完了の確認に用いられるメッセージ
Queue Get Config Request／Reply	キューに関するコンフィグ取得に用いられるメッセージ
スイッチ側から送られるメッセージ	
Packet In	スイッチが受信したパケットを送るメッセージ
Flow Removed	スイッチ中のフローが消えたことを通知するためのメッセージ
Port Status	ポート状態変化をスイッチから通知するためのメッセージ
コントローラ、スイッチ双方から送られるメッセージ	
Hello	コントローラ、スイッチ間接続の開始時に用いられるメッセージ
Echo Request／Reply	コントローラ、スイッチ間接続の死活監視に用いられるメッセージ
Error	エラーを通知するためのメッセージ
Vendor	ベンダー独自定義のためのメッセージ

3　動作例

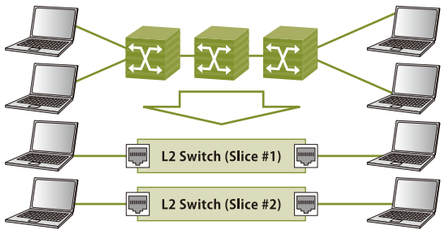
前章で説明したようにOpenFlow仕様では、スイッチの動作とその制御のためのメッセージが定義されています。しかし、これらを組み合わせて、ネットワークをどのように制御するかは、仕様中に規定されていません。つまり、コントローラを開発する際に、必要となる制御機能自体を設計し、それを実現することが可能です。そのため、コントローラの動作は、その目的や実装ごとに異なります。

ここでは、一例として、仮想ネットワーク機能を実現するためのOpenFlowコントローラであるSliceable Switch※2を取り上げ、その動作例を紹介します。さらに、多くのコントローラで採用されているトポロジー検出の仕組みを紹介します。

3.1 仮想ネットワークの実現

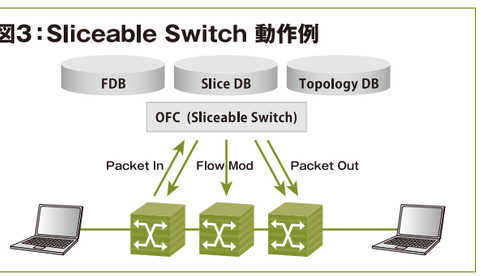
Sliceable Switchは、OpenFlowスイッチで構成されたネットワークを、複数のL2ドメイン（スライス）に分けて使用する際に用いるコントローラです（**図2**）。

図2：ネットワークの仮想化



OpenFlowスイッチは、新規のフロー到着時に、Packet Inメッセージを用いて受信パケットをSliceable Switchへと送ります。このときSliceable Switchは、次のように動作します（**図3**）。

図3：Sliceable Switch 動作例



1. パケットの受信スイッチ、ポートと送信元MACアドレスの組を、Forwarding DB（FDB）に登録します。

2. パケットの宛先MACアドレスをキーにFDBを検索し、出口スイッチとポートを決定します。

3. 2.で検索したポートと、パケットを受信したポートの所属スライスをSlice DBを用いて検索し、両者が同一スライスに所属している場合のみ次の処理に進みます。

4. Topology DBの情報を用いて、パケットを受信したスイッチから出口スイッチまでの最短パスを計算します。

5. 4.で計算した最短パス中に沿ってパケットが転送されるよう、Flow Modメッセージを用い、パス上の各スイッチにフローを設定します。

6. 受信パケットを2.で検索したポートから出力させるために、Packet Outメッセージを用いて出口スイッチへと送ります。

L2スイッチングの機能をコントローラ側で実現するために、ステップ1でMACアドレスを学習し、ステップ2で学習した結果を用いた出力ポートを決定しています。まだ学習していない宛先MACアドレスを持つパケットを受信した場合には、パケットを受信したポートと同一スライスに所属するポートから出力させるために、Packet Outメッセージで各スイッチへと送ります。

この実装の利点は、他のスライスに影響を与えることなく、スライスの追加や削除が容易にできることです。既存のVLAN技術を使ってスライスを作る場合、ネットワークを構成する各スイッチに対して設定変更を行う必要があります。しかし、この実装では、新規スライスの作成時、コントローラ側で保持するSlice DBに追加するだけで完了します。Slice DBの情報を使用するのは、実際に所属するスライス間の通信が発生する時なので、新規スライスの追加が他のスライスに影響を与えることはありません。

Sliceable Switchは、オープンソースのコントローラ開発フレームワークである、Trema※3を用いて実装されています。このコントローラのソースコード自体もGitHub上で公開されています※4。

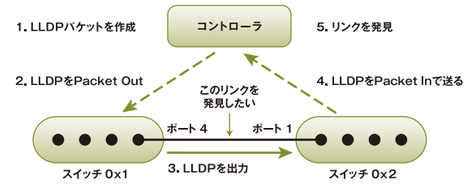
3.2 トポロジー検出

Sliceable Switchでは、最短パスを計算するために、トポロジー情報（スイッチ間の接続関係に関する情報）を用いていました。しかし、トポロジー検出の仕組みは、OpenFlow仕様中には規定されていません。ここでは、標準的なOpenFlowスイッチで動作するLink Layer Discovery Protocol（LLDP）の仕組みを流用した検出方法について紹介します。この検出方法は、OpenFlow仕様の範囲内で動作するため、多くのOpenFlowコントローラにて採用されています。

LLDPとは、Ethernetスイッチ同士の自律的な情報交換により、スイッチ間の接続関係を知るための仕組みで、IEEEにて標準化されています※5。LLDPに対応したスイッチは、自身に関する情報を埋め込んだLLDPパケットを生成し、隣接するスイッチへと通知します。しかし、OpenFlow仕様では規定されていないため、OpenFlowスイッチはこの機能を持っていない可能性があります。そのため、ここで紹介する検出方法では、スイッチの代わりにコントローラがLLDPパケットの生成や、受け取ったLLDPパケットの解析を行います。

この検出方法の具体的な動作を、**図4**を用いて説明します。例えば、スイッチ1のポート4に接続するリンクを調べる場合について考えます。

図4：トポロジーの検出



1. まずコントローラは、接続関係を調べたいスイッチのDatapath ID 0x1とポート番号4を埋め込んだLLDPパケットを作ります。

2. ポート4から出力するというアクションを含むPacket Outメッセージを作り、先ほど作ったLLDPパケットをスイッチ0x1へと送ります。

3. Packet Outを受け取ったスイッチはアクションに従い、LLDPパケットを指定されたポート4から出力します。その結果、LLDPパケットは、ポート4の先につながるスイッチ0x2へと到着します。

4. LLDPパケットを受け取ったスイッチ0x2は、自身のフローテーブルを参照し、パケットの処理方法を調べます。このときLLDPに対するフローはあえて設定していないため、今回受信したLLDPパケットは、Packet Inとしてコントローラまで戻されます。

5. コントローラは、受け取ったPacket Inメッセージを解析することで、スイッチ0x1とスイッチ0x2の間のリンクを検出します。

この一連の手続きを、コントローラと接続するスイッチに対して繰り返すことで、ネットワーク全体のトポロジーを把握することができます。

4　標準化について

OpenFlowの標準化は、Stanford大学を中心とするOpenFlowスイッチコンソーシアムにて開始され、2009年にバージョン1.0の仕様※1が策定されました。現在商用化されているOpenFlowスイッチやコントローラの多くは、このバージョン1.0に対応しています。

その後、2011年にSoftware Defined Network（SDN）技術の推進を行う非営利団体Open Networking Foundation（ONF）が設立され、OpenFlowの標準化は現在このONFにて行われています。SDN技術の推進を利用する側からの視点で標準化作業を行っていくために、ONFのボードはGoogle社、Facebook社やNTTコミュニケーションズ社といった、サービス事業者や通信事業者で構成されています。

ONFでは、現在最新バージョンである1.3の仕様※6が策定されています。このバージョンでは、MPLS、IPv6やPBB（Provider Backbone Bridge）への対応など、キャリア網への適用を想定した機能拡張が行われています。今後登場するスイッチやコントローラの多くは、このバージョンに対応したものになると考えられます。

5　おわりに

本稿では、OpenFlow技術を解説しました。OpenFlowではネットワークの制御方法を、ネットワーク管理者自らが自由に設計し、それを実現することができます。本稿ではOpenFlowコントローラの動作例をいくつか紹介しましたが、これらはあくまで一例に過ぎません。目的が同じでも、いくつかの異なる制御方式が考えられます。

新たな制御方法の設計・実装は、従来のように機器ごとに用意された機能を組み合わせて使うことと比べると、少し敷居が高いかもしれません。しかし、従来の常識にとらわれることなく、ネットワーク制御のあるべき姿を再考してみることは、ネットワーク技術の今後の発展には不可欠であると、筆者は考えています。本稿がその一助となれば幸いです。

（日本電気株式会社クラウドシステム研究所 鈴木一哉）

※1 OpenFlow Switch Specification Version 1.0.0（Wire Protocol 0x01）December 31, 2009
http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

※2 L. Sun, K. Suzuki, Y. Chiba, Y. Hatano and H. Shimon-ishi : A network management solution based on Open-Flow towards new challenges of multitenant data center,in Proceedings of APSITT 2012, November 2012.

※3 Trema : Full-Stack OpenFlow Framework in Ruby and C
http://trema.github.com/trema/

※4 Trema Application repository
https://github.com/trema/apps/

※5 IEEE 802.1ab-2009 : IEEE Standards for Local and Metropolitan Area Networks, Station and Media Access Control Connectivity Discovery, September 2009.

※6 OpenFlow Switch Specication Version 1.3.0（Wire Protocol 0x04）June 25, 2012
https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf