「SDN」の最新動向

SDNとその周辺の最近の動向、ならびに株式会社ストラトスフィアの製品Omnisphereについて解説します。

3.1 最近の動向

「SDN[*1]」という単語は、もはやバズワードと言われる程一般的に知られるようになり、関連分野のマーケット活動は非常に活発です。筆者の個人的視点から最近のトピックを表-1にまとめました。要素技術としてはOpenFlowやVXLANに関連しています。

表-1 最近のSDN関連ニュース

日付	トピック
2012年２月７日	Nicira社がステルスを解除する
２月29日	OpenvSwitch 1.4.0 Fedoraパッケージ化される
３月10日	自宅ラック勉強会 #2.5 秋葉原出張編 開催
３月21日	OpenvSwitch kernel module 取り込まれたLinux 3.3リリース
４月16日	CloudStackがApache incubationプログラムに入る
６月	GoogleがSDN-WANを発表
６月25日	OpenFlow 1.3発行
７月23日	VMWare社がNicira社を買収
９月６日	OpenFlow 1.3.1発行
９月27日	OpenStack Folsomリリース
９月	trema-edge forkされる
10月31日	SSP 1.0リリース
11月30日	SSP 1.0.2リリース
12月10日	VXLAN kernel module取り込まれたLinux 3.7リリース
12月19日	Omnisphereプロジェクト開始
2013年２月５日	JR東日本駅構内設備にOpenFlowを使うと発表。無線LANも対象
２月13日	SSP 1.0.3リリース
３月20日	Apache CloudStackがtop level projectになる
３月29日	SSP 1.1.1リリース
４月	OpenDayLightプロジェクト公開
４月３日	trema-edgeを使い始める
４月４日	OpenStack Grizzlyリリース
４月15日	RDO(Red Hat)公開
４月17日	trema-openwrt公開
４月25日	OpenFlow 1.3.2発行
５月28日	Apache CloudStack 4.1リリース
６月12日	Interop Tokyo 2013開催 - Omnisphere発表
６月25日	Indigo Virtual SwitchがOpen Sourceになる (Big Switch Networks)
７月５日	SSP 1.1.2リリース
８月５日	OpenFlow 1.4発行
９月17日	SDN Japan 2013開催 - Omnisphere 1.0リリース
10月１日	Apache CloudStack 4.2リリース
10月17日	OpenStack Havanaリリース

:ストラトスフィアの動き

仮想OpenStack、Cloudstackのニュースを挙げていますが、これらはオーケストレータと呼ばれるソフトウェアで、仮想マシンを管理するハイパーバイザとその周辺コンポーネントをまとめて制御します。マイグレーション機能を使った際には仮想マシン移動前後で同じネットワーク環境を構築しなければなりませんが、オーケストレータでは、その調整のためにOpenFlowを活用しています。OpenFlowの事実上の本家であるNicira社をVMWare社が買収したニュースはインパクトがありました。

また象徴的なものはGoogle SDN-WANの発表で、OpenFlowを用いたネットワーク制御を世界規模で実運用に使用しているというものでした。オーケストレータでの用途とは異なる側面で、ネットワークのトラフィック制御に重点が置かれています。従来からあるBGPなどで制御されたネットワークとOpenFlow制御のネットワークとを結合して運用していることも実証されており、また、普段あまり語られないGoogleのネットワーク運用を垣間見られるという側面もあり、業界では話題になりました。

3.2 オフィス環境のSDN

一方で私たちは、オフィス環境ネットワークの運用に問題を抱えていました。オフィスフロアにネットワーク回線を敷設する際に、電源系統と同様に情報コンセントを床下配線し、フロアラックにあるスイッチングハブで集中管理しています。部署ごとに異なるネットワークを割り当てて運用していますが、そこではスイッチの機能であるVLANを使用してネットワークを隔離しています。そのため、人事異動などでオフィス内の座席配置が変わるたびにフロア情報コンセントに対応するVLANの設定変更が必要となります。この運用負荷は大きく、更にループなどが原因で発生するストーム

＊1 https://www.opennetworking.org/ja/sdn-resources/sdn-definition

の原因特定やその問題解消も重荷になっていました。無線によるネットワーク運用でも問題が出てきています。無線機器が普及し、例えばDHCPでの割り当てアドレスが足りないなど、当初想定していた容量を超える問題が起きています。これらをSDNのアプローチで解決できないかと考えていました。

これまでOpenFlowというと、オーケストレータやSDN-WANが話題の中心となっていました。オーケストレータは主にマシン群が設置されるデータセンター環境を想定されていますし、SDN-WANはバックボーン回線などが行きかうNOCの環境です。OpenFlowの仕組み自体は分野を特定するものではありませんが、資金面・研究分野からの連続性といった理由から、SDNを扱うマーケットは大規模ネットワークが盛んでした。これに対して、Omnisphereは身近なオフィス環境にSDNを適用する試みの1つとなっています。

3.3 Inside Omnisphere

今後の実装では変更される可能性もありますが、現時点でのOmnisphereの実装内部について記述します。Omnisphereは図-1のような構成になっています。

3.4 OpenFlow Switch

オフィス環境にOpenFlowを導入する場合、まず課題となるのは、安価なOpenFlowスイッチが入手できるかどうかです。高価なOpenFlowスイッチを使ってデモンストレーションを行っても、実際に導入するイメージとの乖離が大きくては説得力がありません。

OpenFlowプロジェクト本体に、OpenFlow 1.0のリファレンス実装を用いた「Pantou for OpenWrt[*2]」というプロジェクトがありました。OpenWrt[*3]は、市販ルータのファームウェアを書き換えて、汎用的なLinuxとして使えるようにするプロジェクトです。専用のOpenFlowスイッチは現在でも簡単に手に入るものではありませんが、OpenFlowが作られはじめた当初から、OpenFlowスイッチの解の1つとして使われてきたようです。

日本のコミュニティに知れ渡ったのは「自宅ラック勉強会#2.5秋葉原出張編[*4]」でしょう。日曜大工的な感覚でOpenFlowハードウェアスイッチを作ることができるということで、ハッカー的な心をくすぐる課題となりました。自宅ラック勉強会#2.5は、安価に入手できるBuffalo AirStationをOpenFlow対応スイッチにして遊びましょう、という趣旨の会で、盛況だったそうです。余談ですがこの会の主催者の1人は、これを使って「Interop Tokyo 2012特別企画Open Router Competition」でNEC賞を受賞されています[*5]。そのような中、ストラトスフィアでもBuffalo AirStationを用いた実験を始めていました。この時点で有線・無線LANを同時に進めるのは自然な流れでした。

Omnisphereの開発にあたっては、いくつかの理由から総合的に判断し、OpenFlow 1.3をターゲットとしました。具体的には、1）オフィス環境に設置されるOpenFlowスイッチはNOC設置の想定よりは小さいリソースしかないものであり、その時点で最も省スペースに高性能な表現ができるプロトコルであること、つまり最新のプロトコルが望ましかったこと、また、2）OpenFlowを開発しているONF（Open Networking Foundation）においてOpenFlow 1.3で普及

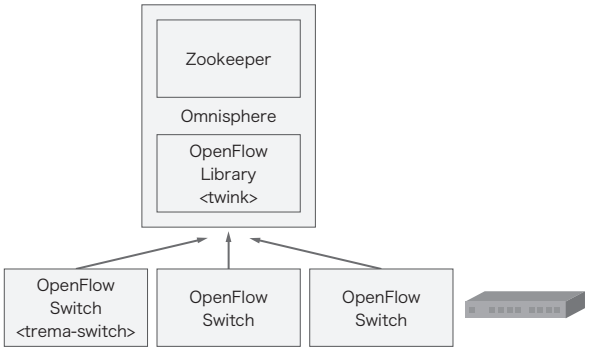

図-1 Omnisphereの構成

*2 http://archive.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT
*3 https://openwrt.org/
*4 http://atnd.org/events/26147
*5 http://www.interop.jp/2012/orc/

を目指すという合意が一度なされていること、3)無線LANに対応した商用OpenFlowスイッチが存在せず、ほぼ新規に対応を目指すことになること、などです。ちょうどtremaがOpenFlow 1.3対応を目指したtrema-edge[*6]をforkして、開発がかなり進んでいたところでもありました。

tremaはOpenFlowコントローラのためのフレームワークとして知られていますが、trema-edgeにはそれに付随してC言語で記述されたOpenFlow 1.3ソフトウェアスイッチがあります。OpenFlow 1.3に対応したスイッチ実装は、他にofsoftswitch[*7]やIVS[*8]などがあります。OpenFlow 1.3を比較的シンプルに実装しているという理由で、私たちはtrema-edgeのスイッチを開発に活用することにしました。trema-switchをOpenWrt上で動作させるためには、OpenWrtのフィードを使ってファームを作ります。フィードは、GitHubで公開しています[*9]。Interop 2013ではこのスイッチ実装を用いて実際にデモンストレーションを行いました。

3.5 OpenFlow Library

OpenFlowプロトコル部分のフレームワークについては、必要な機能を満たすフレームワークが存在しなかったため、新規に作成しました。OpenFlowアプリケーションのフレームワークとしてはフロールールを書くことが簡単に、また簡潔に記述できることが重要になります。フロールールが簡潔にプログラム中に記載されているということは、そのままプログラムのメンテナンス性に直結します。そこでOpen vSwitch ovs-ofctlでの記述形式も使えるようにしました。

よくあるOpenFlowライブラリでは、OpenFlowメッセージの解析・組立がTCPサーバの機能と一体になっています。OpenFlowメッセージを作成する際には、ライブラリ独自の流儀に合わせるための学習コストが高いのはもちろん、ライブラリが対応していない部分については手が出せなくなってしまいます。OpenFlow 1.3で導入されたAuxiliary connectionでは、UDPのサブ接続を持つことができるのですが、サーバ機能が一体になっているものは対応が困難です。作成したライブラリでは、I/Oイベントループとプロトコル解析部分を分離し、コントローラのコアからはプロトコルバイナリフォーマットを翻訳する機能を一切なくしています。

実際の開発や運用ではOpenFlowスイッチがコントローラと接続中であっても、OpenFlowスイッチに対して直接状態の問い合わせを行いたいので、このような問い合わせを中継できるコントローラをフレームワーク内に用意しました。UNIX domain socketからの接続を外部プログラムからの中継の入り口として使えるようにもしました。OpenFlowスイッチからの出力をすべてモニタリングする外部出力も作りました。

構成についてまとめると、図-2のようになっています。コントローラが複数の上位アプリケーションと通信する際は、OpenFlowのBARRIERメッセージで隔離しています。フレームワーク中でOpen vSwitch表記で記述したフロールールはovs-ofctlの引数となって、コントローラを経由してOpenFlowスイッチへと届きます。モニターポートを設置する点はOpen vSwitchを参考にしました。中継ポートを使うと、複数のOpenFlowアプリケーションをチェーンすることもできます。

このような構成のため、ライブラリのコア部分はOpenFlowプロトコルバイナリフォーマットをそのまま使っています。プロトコルバイナリフォーマットを扱うライブラリは、OpenFlowのメッセージを直接的に作成したいときに必要になるので、別途単体で作成しました。

*6 https://github.com/trema/trema-edge
*7 http://cpqd.github.io/ofsoftswitch13/
*8 http://www.projectfloodlight.org/indigo-virtual-switch/
*9 https://github.com/iHiroakiKawai/trema-openwrt

3.6 Zookeeper

SDNはネットワーク制御部分とデータ通信部分を分離して、制御部分をよりプログラマブルに扱えるようにするアーキテクチャを指します。より具体的には、GoogleのOnix[*10]の発表にも登場します。制御部分を分離するということは、ますます高性能化する汎用サーバ機を使えるということで、一体化していた場合と比べて、制御部分のロジック拡張が飛躍的に自由になります。汎用サーバ機を計算資源として使うことは、暗に分散コンピューティングを行うことを示しています。SDNを成立させている背景には、そのような分散コンピューティングの環境が整ってきたことも要因だと考えています。

OmnisphereではApache Zookeeperを使っています。ZookeeperはHadoopから派生したプロジェクトで、Hadoop 2.0の基本コンポーネントです。合意アルゴリズムとして有名なPAXOSはGoogleのOnixで使われていましたが、ZookeeperではZabと呼ばれる同等のアルゴリズムを実装しています。

Zookeeperは分散したコンピュータ上で矛盾なく扱えるようにメンテナンスされたツリー構造を提供します。ツリー構造は直接使うこともできますが、そのままでは使いづらいので、ZookeeperにはRecipeと呼ばれるクラスが付属しています。Recipeはこのツリー構造を内部で使用して、キューやロックに始まり、リーダー選出アルゴリズムなど、使いやすいインタフェースを提供しています。アプリケーションから利用する際は、ツリー構造の一部をこのような機能に割り当てて使います。

3.7 まとめ

Omnisphereでは、スイッチ側にはフリーで利用できる実装を活用しつつ、コントローラ側には現代的なソフトウェア基盤を使って構築されております。

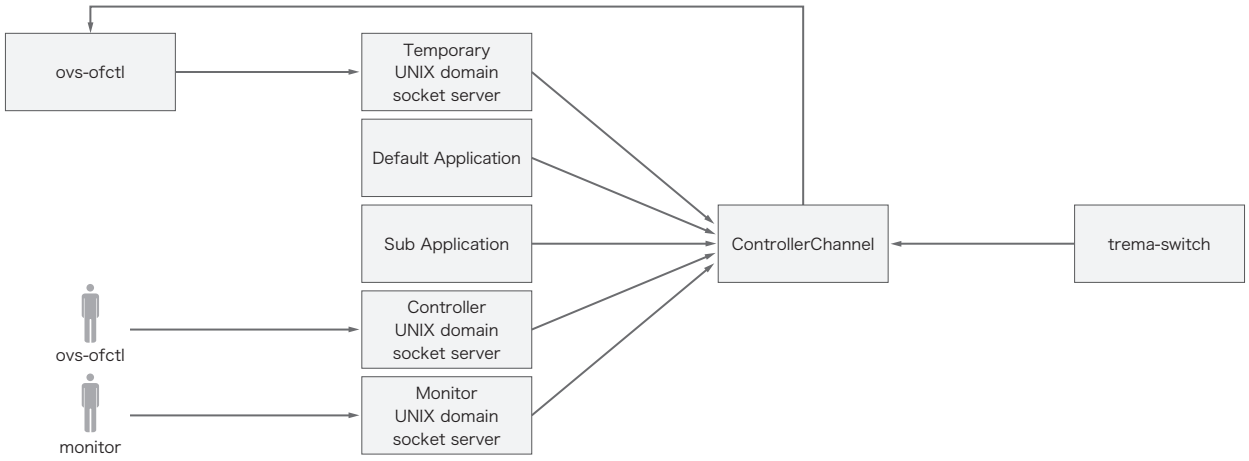

図-2 フレームワーク中でovs表記を使った場合

執筆者:
川井 浩陽(かわい ひろあき)
株式会社ストラトスフィア。2003年IIJ入社。2006年i-revo立ち上げに参加。関西在住でApache好きのオープンソース畑育ち。

*10 https://www.usenix.org/conference/osdi10/onix-distributed-control-platform-large-scale-production-networks