自作OpenFlowコントローラlilyと Quantumについて

2012/3/23

宮下　一博
@kamakura_bakufu
miyakz1192@gmail.com

自己紹介

- 某電機メーカ勤務
- 入社以来、サーバのNIC冗長化ドライバ、クラウドインフラ管理ソフトウェア等々、ネット系のソフトウェア開発に従事
- #nwstudyに二回参加
  - 勉強会参加者の自宅ラボ同士をL2-VPNで接続するインフラを紹介
- #sdnstudy OpenFlow勉強会その1を開催
  - 2回目は近日

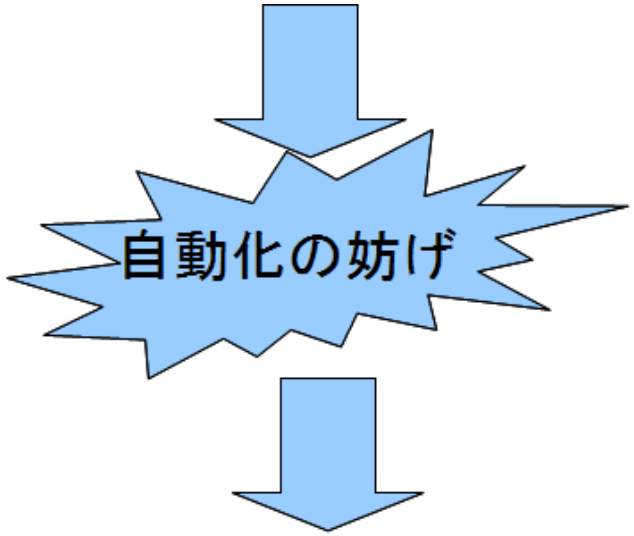
データセンタネットワークの課題

- 仮想化集約するとネットワークが複雑になる
- 短納期なのにネットワークの設計、構築が大変
- 様々な技術が絡むためトラブルシューティングが困難で時間がかかる
  - RSTP、802.1q、LACP、仮想スイッチ、VNICなど・・・
- 全自動化を目指すと、もっと多くの知識の習得が必要
  - AMPP、VXLAN、

これだけではない、もっとたくさん課題がある・・・

「全自動化」が技術の流れ

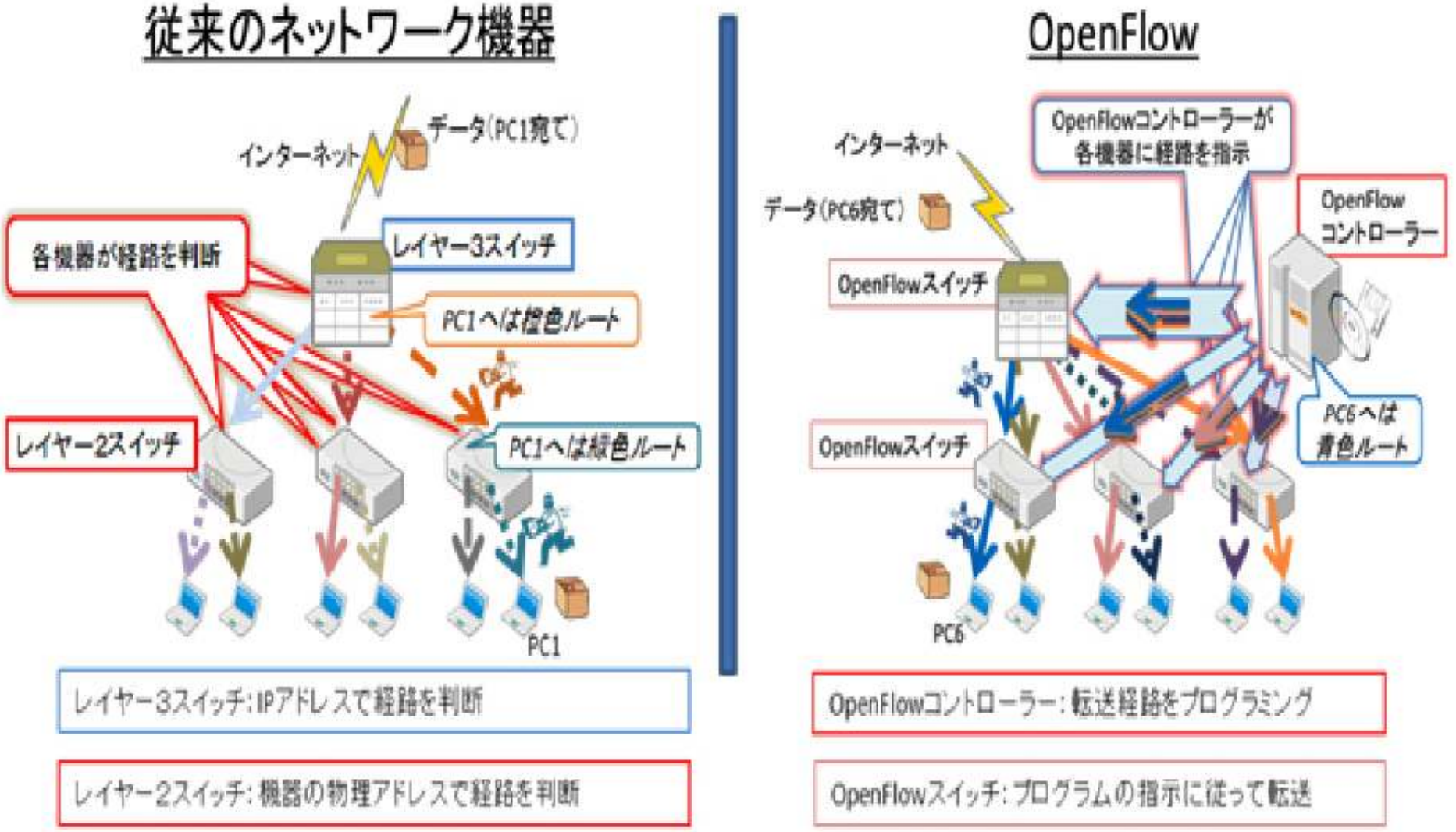
- 現在のネットワークは様々なプロトコルのジグソーパズル。スイッチ/ルータのインタフェースは統一されていない。



ソフトウェアによってネットワークの論理構成を
柔軟に変える、Software Defined Network(SDN)

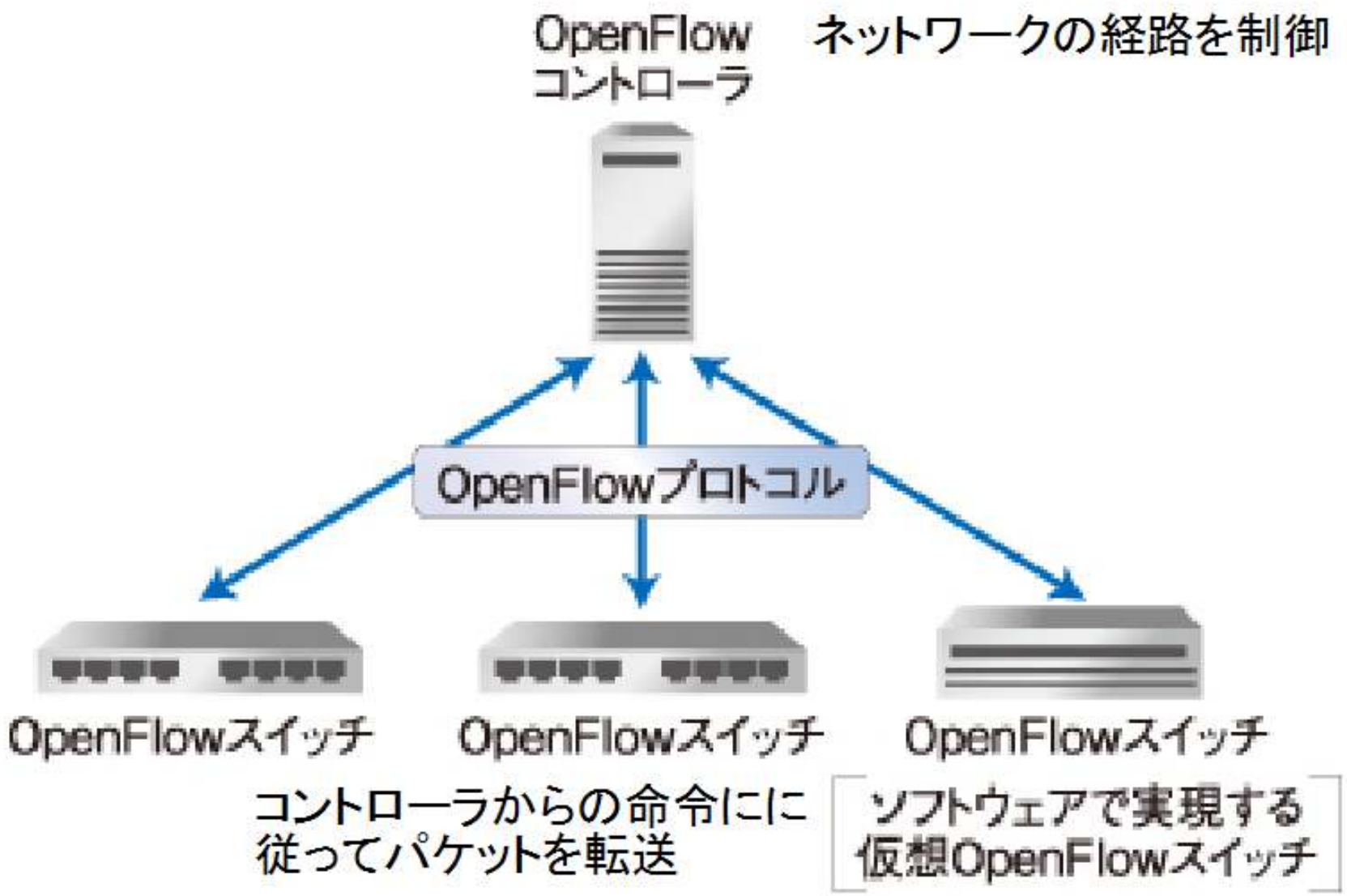
OpenFlowへの期待感

1つのプロトコル、1つのレイヤリング、ネットワークの構築・運用・保守をシンプルに

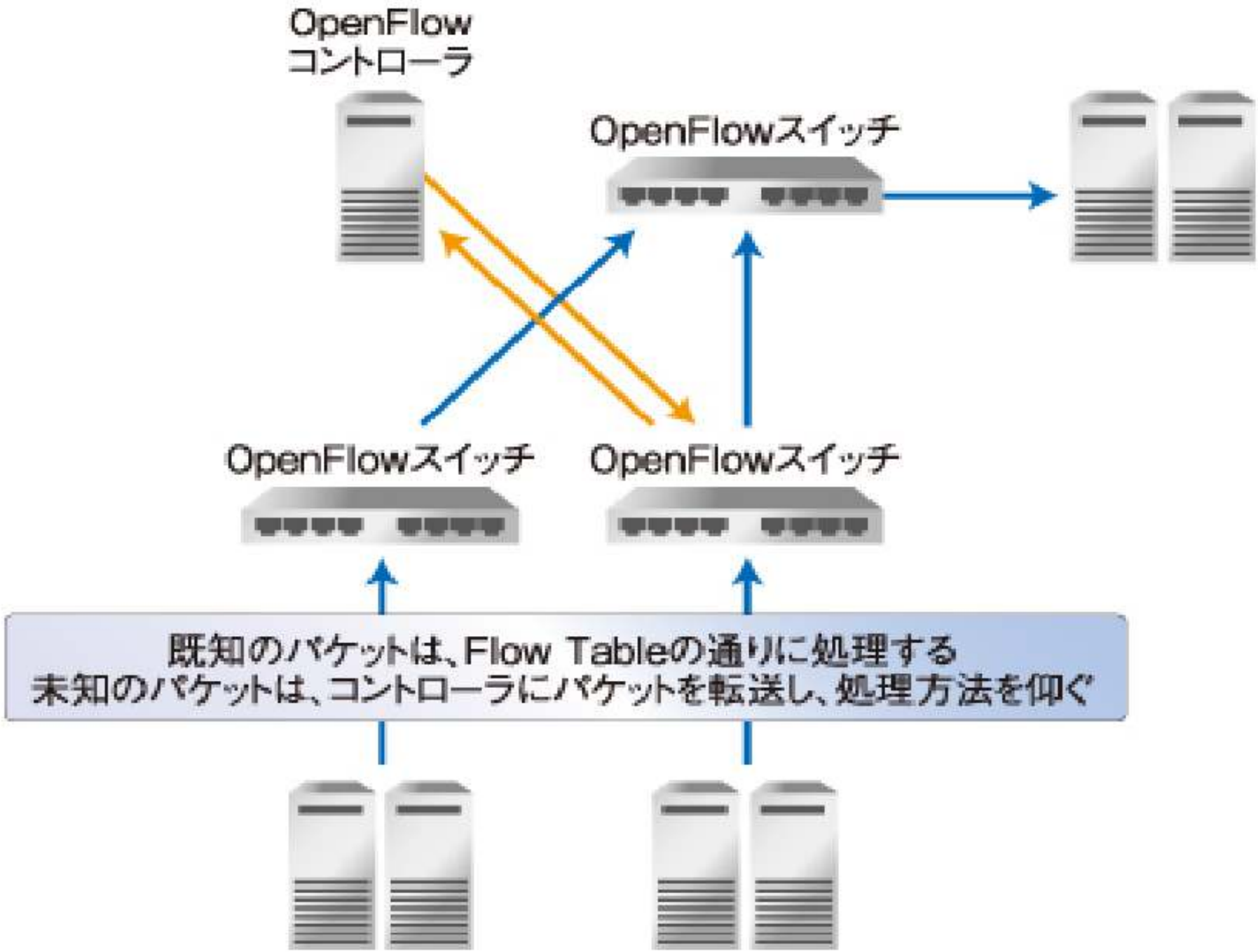


http://www.kobelcosys.co.jp/column/itwords/243/

OpenFlowのアーキテクチャ



出典、引用元:


OpenFlow
コントローラ
OpenFlowスイッチ
OpenFlowスイッチ
OpenFlowスイッチ
既知のパケットは、Flow Tableの通りに処理する
未知のパケットは、コントローラにパケットを転送し、処理方法を仰ぐ

パケットのL2～L4フィールドを見てスイッチング

- Ingress Port
- MPLS label
- Metadata
- MPLS traffic class
- Ethernet source address
- IPv4 source Address
- Ethernet destination adress
- IPv4 destination Address
- Ethernet type
- IPv4 protocol / ARP opcode
- VLAN id
- IPv4 ToS bits
- VLAN priority
- TCP/ UDP / SCTP src port / ICMP Type
- TCP/ UDP / SCTP dst port / ICMP Code

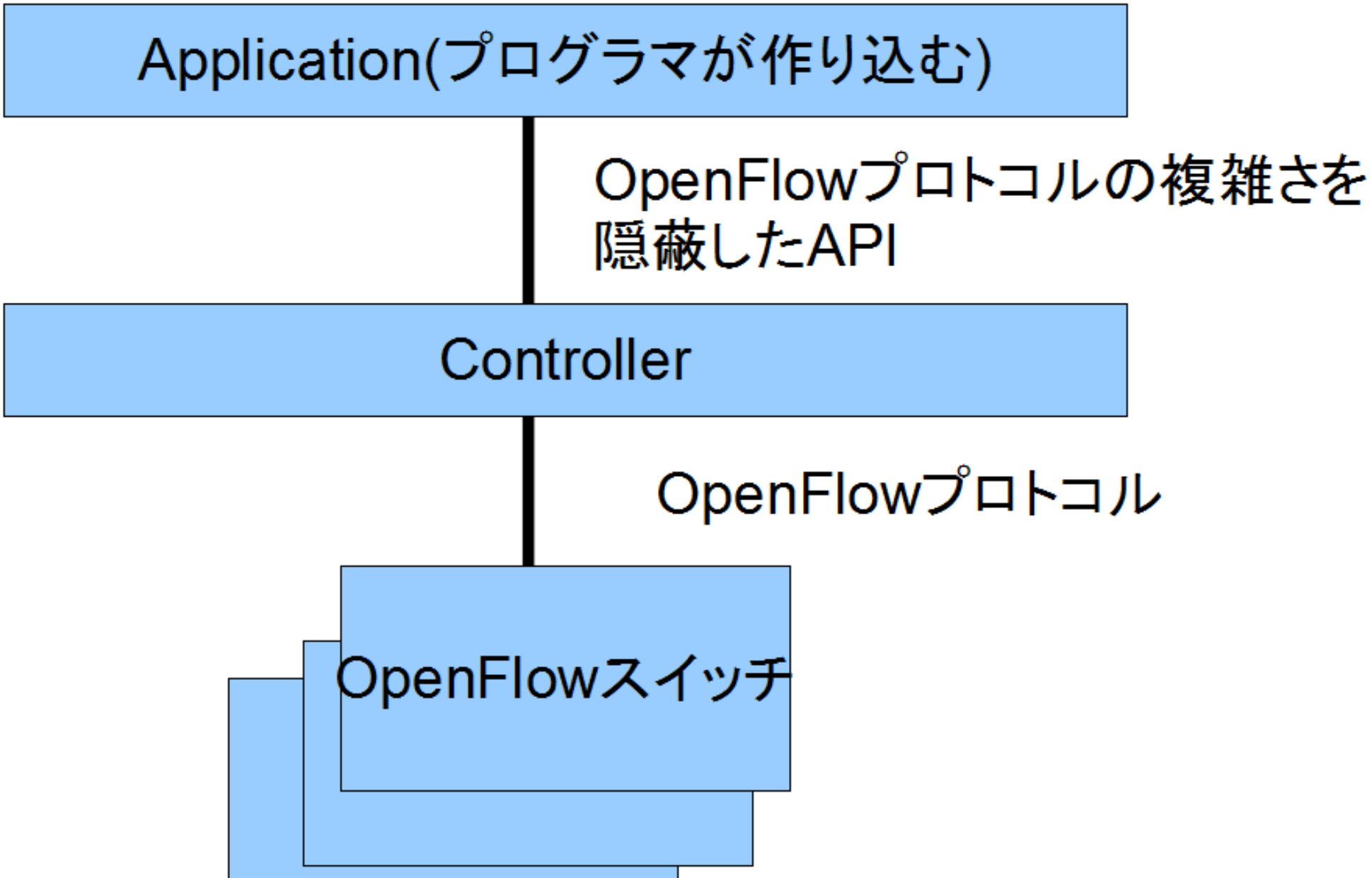
OpenFlowスイッチの状況

- Cisco Nexus 3000
- IBM BNT RackSwitch G8264
- NEC UNIVERGE PFシリーズ
- HP OpenFlow使用可能にするファームウェア
- Open Vswitch(Xen/KVM)
- NEC Hyper-V仮想スイッチ
- Juniper Junos OS上のSDKで対応
- Pronto
- BigSwitch

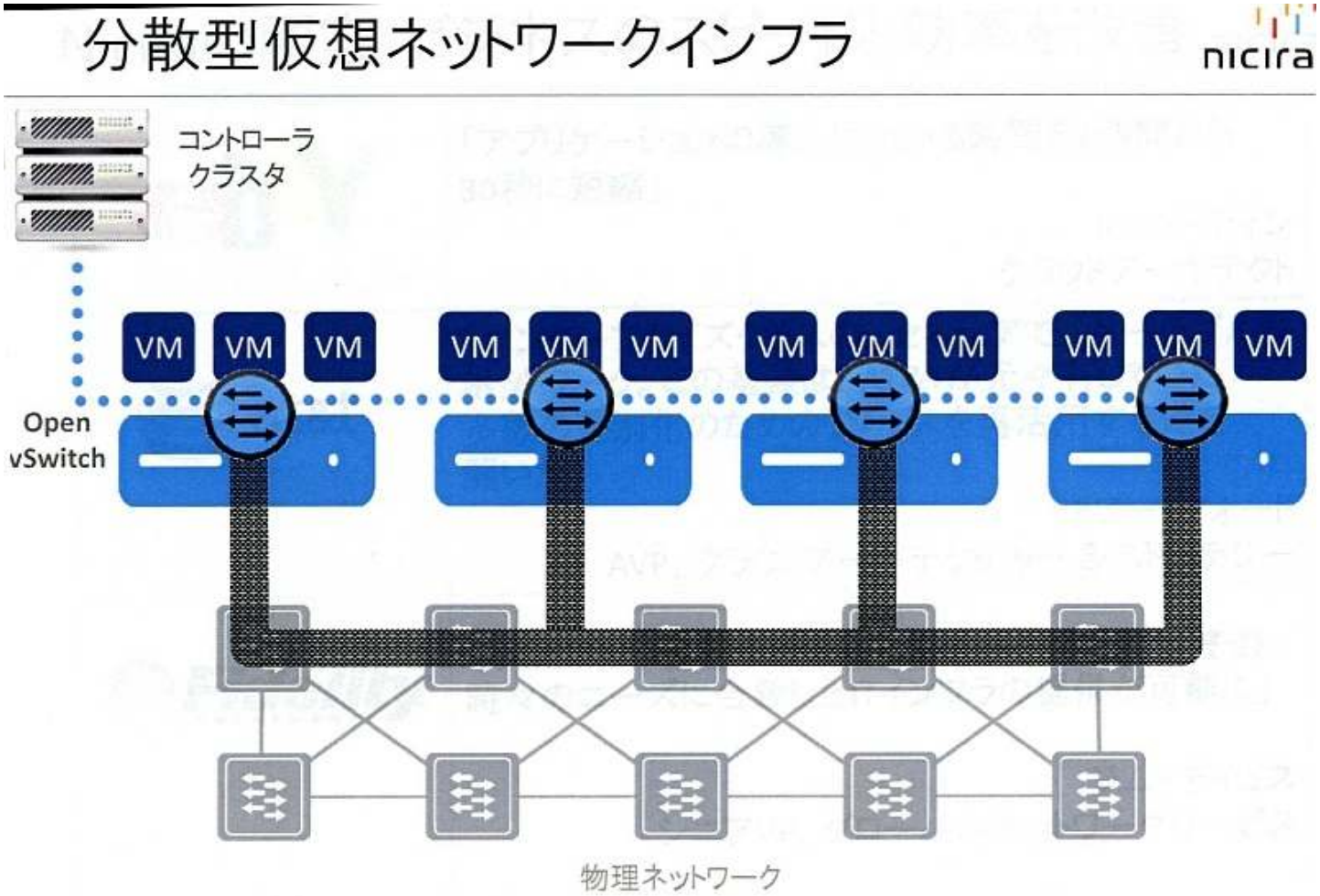
OpenFlowコントローラの状況

- アプリケーションフレームワーク型(単体ではネット制御機能を持たない、OSS)
  - Beacon(Java)
  - NOX(python)
  - Trema(C/ruby)
  - などなど・・・
- 製品(単体でネット制御機能を持つ)
  - Nicira Networks(NVP)
  - NEC プログラマブルフロー・コントローラ「UNIVERGE PF6800」
  - ミドクラ(Midonet)

コントローラの主流アーキ
アプリケーションフレームワーク型



Nicira Virtulization Platform(NVP)



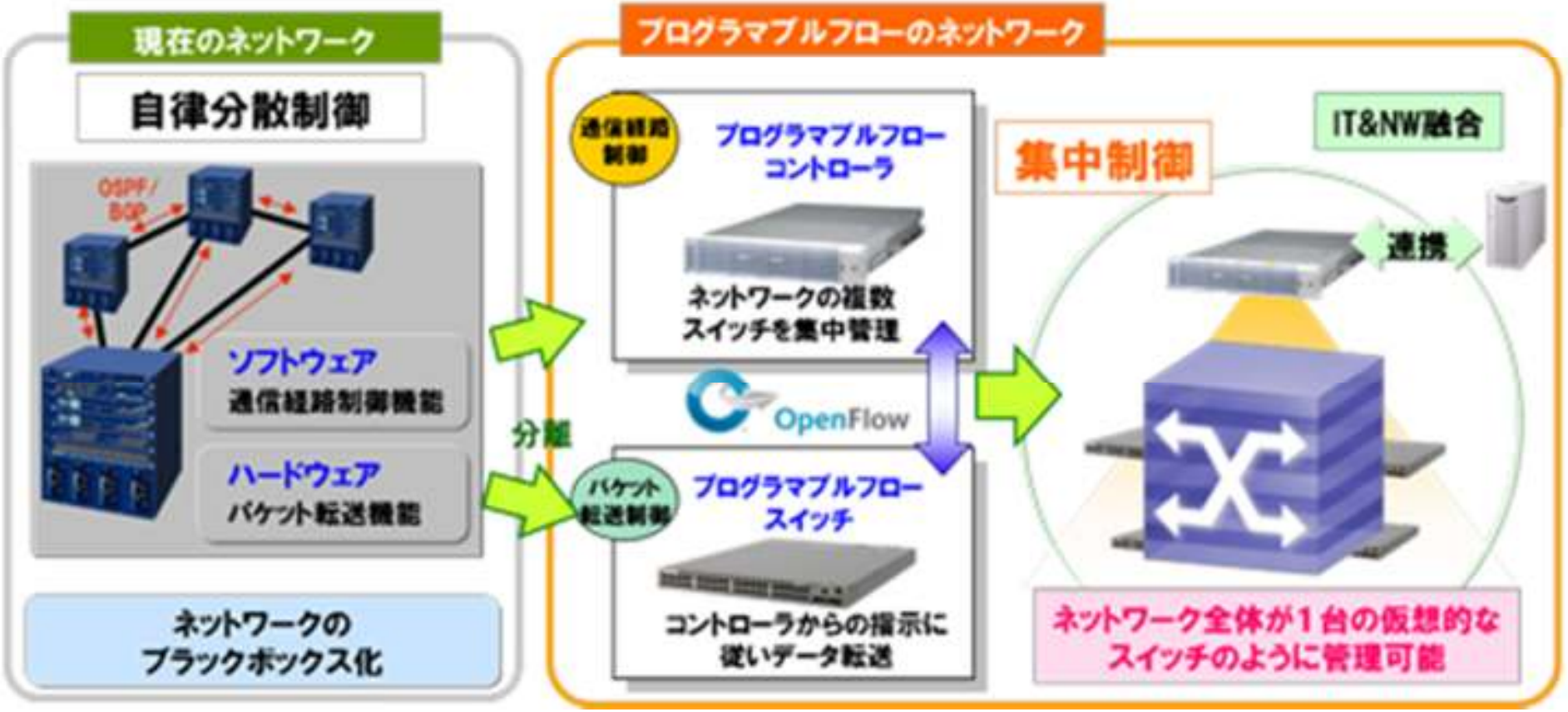
- VM間のL2セグメントをL3のトンネリングで実現（既存の物理スイッチを使用可能、VMホストの業務LAN間がIP到達可能であればNVPを使用可）
- VMマイグレーションに対応

http://www.atmarkit.co.jp/news/201202/23/nicira.html

NEC PF6800コントローラ

プログラマブルフローは、次世代ネットワーク技術である「OpenFlow」を利用することにより、新たなネットワークソリューションを提供します。

ProgrammableFlow = OpenFlow + NWシンプル化 + NW仮想化 + NW可視化



NECはスイッチが250万円、コントローラが1000万円から

http://www.nec.co.jp/datanet/pflow/index.html

NEC PF6800コントローラ

- 1台のコントローラで管理できるスイッチの個数は25台まで
- フロー数は50万フローまで
- コントローラの冗長化はサポート。アクティブスタンバイ構成
- 大規模なネットワークを構成できるわけではないとのこと

midonet

- 物理ネットワーク構成から分離された仮想化ネットワークが、フレキシビリティーを実現します。
- 現状お使いのIPネットワークの上で機能します。
- 実装されたネットワークサービスにより、高価なネットワーク機器を不要にします。
- 完全に分散されたアーキテクチャーになっており、単一障害点(Single Point of Failure)がありません。
- 必要な時に必要なだけコモディティー機器でネットワークサービスをスケールアウトで拡充する事が可能です。
- 集中管理で運用コストを削減する事ができます。
- マルチテナントシステムにより、効率的なシステムの拡張、管理が可能です。

- 技術的な詳細は不明・・・

http://www.midokura.com/J/midonet_j.html

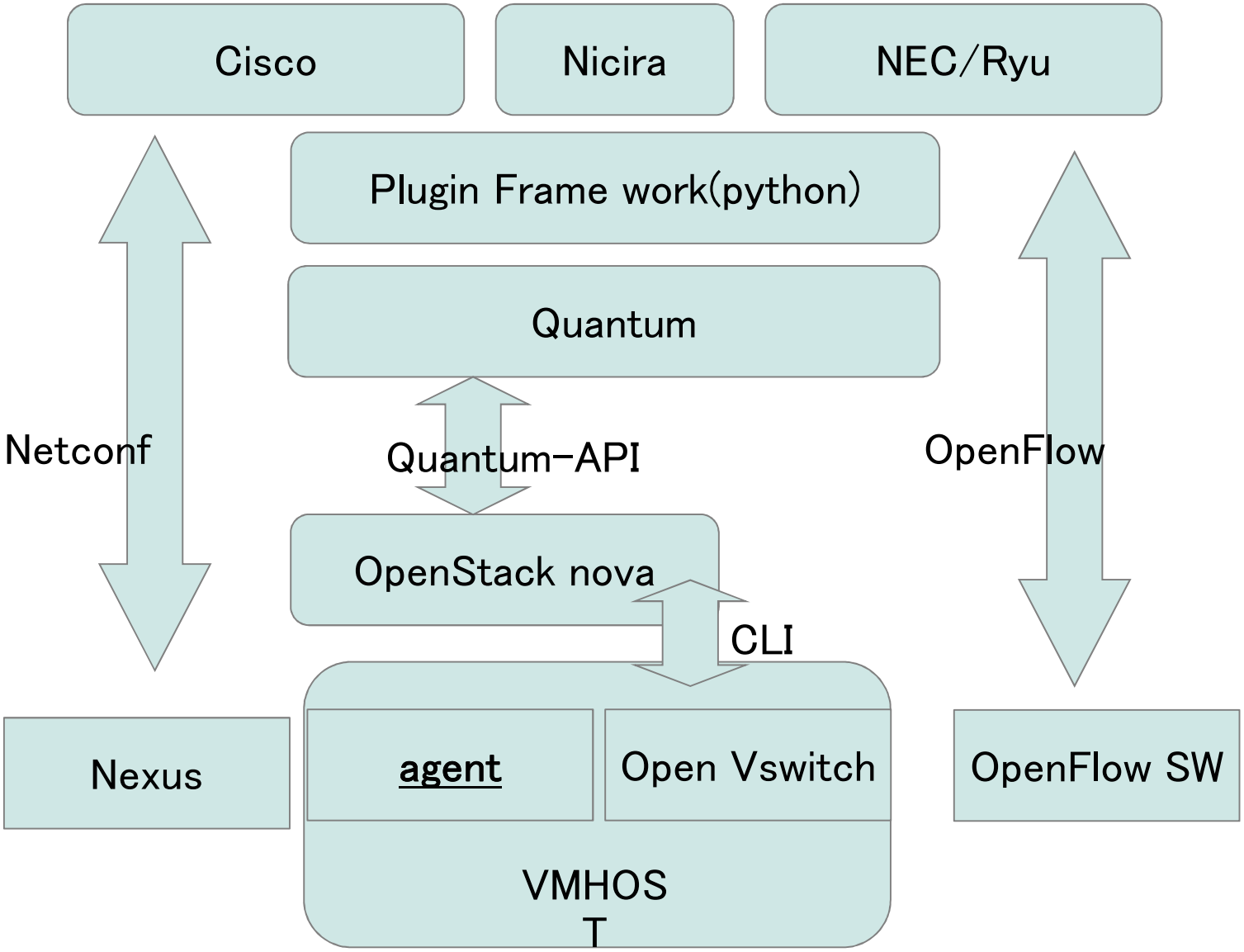
PF6800/NVP/Midonetの違い

- PF6800【NEC】
  - OpenFlowスイッチを使用して物理的なL2ネット全体を仮想化
  - 既存機器は使用できない(OpenFlowスイッチで置き換え）
- NVP【Nicira】
  - OpenFlowはコントローラとOpenVswitch間の連携のみ使用
  - L2をL3でカプセリング
  - 既存機器をそのまま使用可能
- Midonet【ミドクラ】
  - OpenFlowを使用 (OpenFlow勉強会から)
  - L2をL3でカプセリングしている様子(予測)
  - 既存機器をそのまま使用可能(予測)

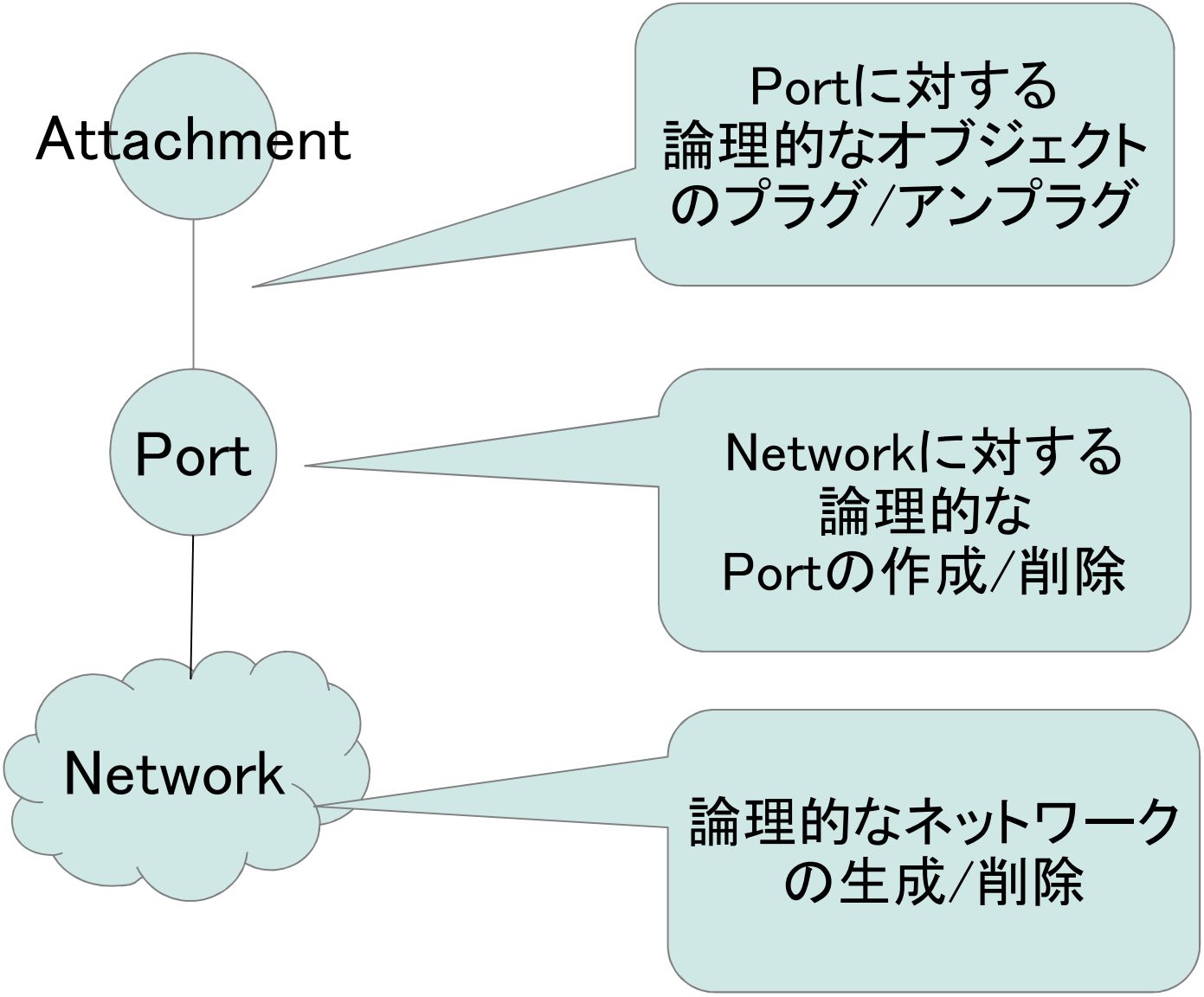
OpenStack Quantum

- OpenStackのネットワーク制御を担当するコンポーネント
- ネットワークの管理を抽象化している
- プラグイン構造をとっており、様々なネットワークの制御方式に対応
- APIも備えており汎用性が高い

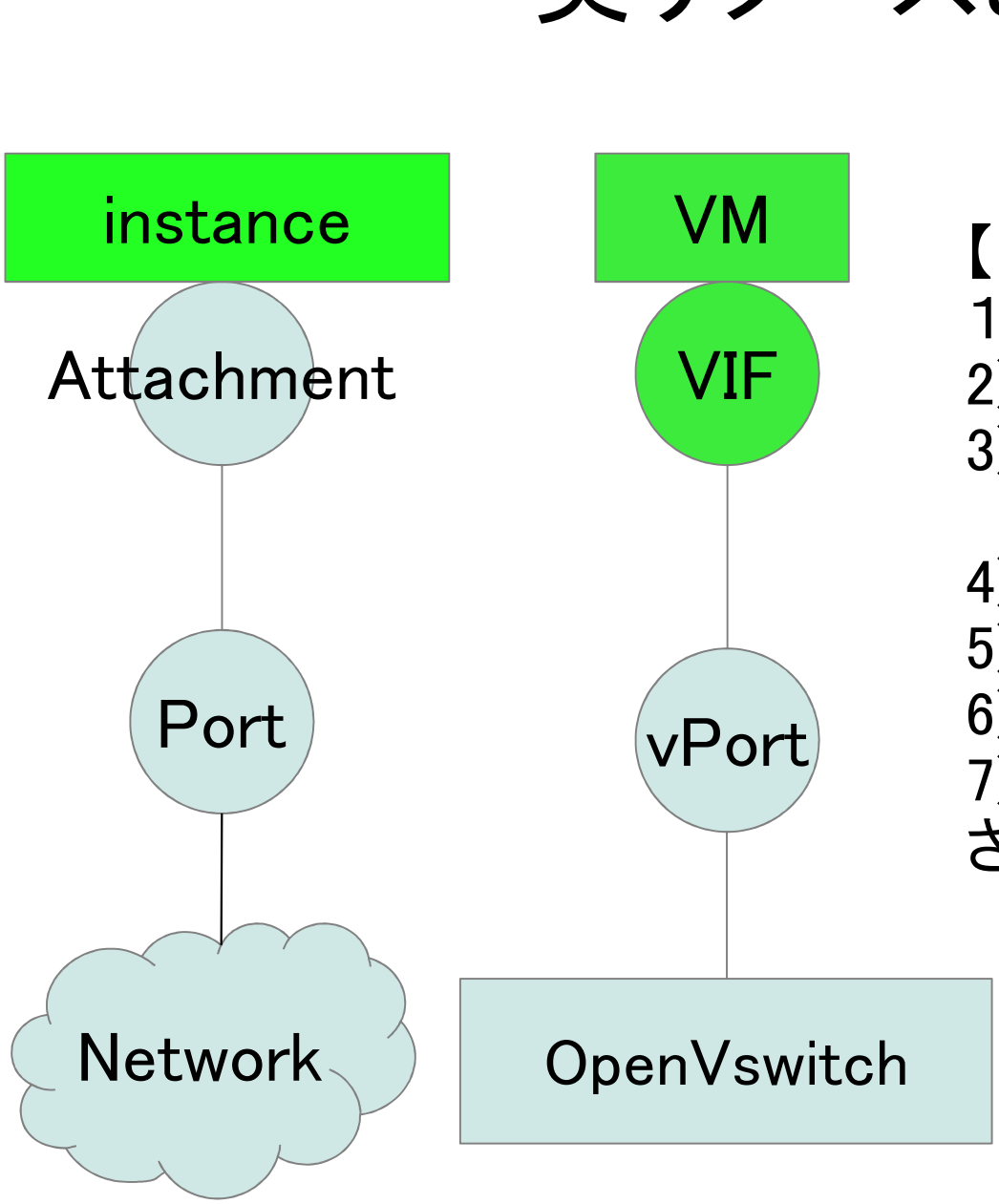
OpenStack Quantumアーキテクチャ



Quantumのネットワーク抽象化モデル



QuantumとNovaの管理分担および実リソースとの関連



【OpenVswitchプラグイン例】
1）Networkを作る(Quantum)
2)instanceを起動(Nova)
3)Networkに対応する
　スイッチを決定（決め打ち）(Nova)
4)OVSのvPort作成(Nova)
5)AttachmentとVIFを関連付け(Nova)
6)VM作成(Nova)
7)Attachment(=VIF)をどのVLANに属させるのかの設定(Quantum)

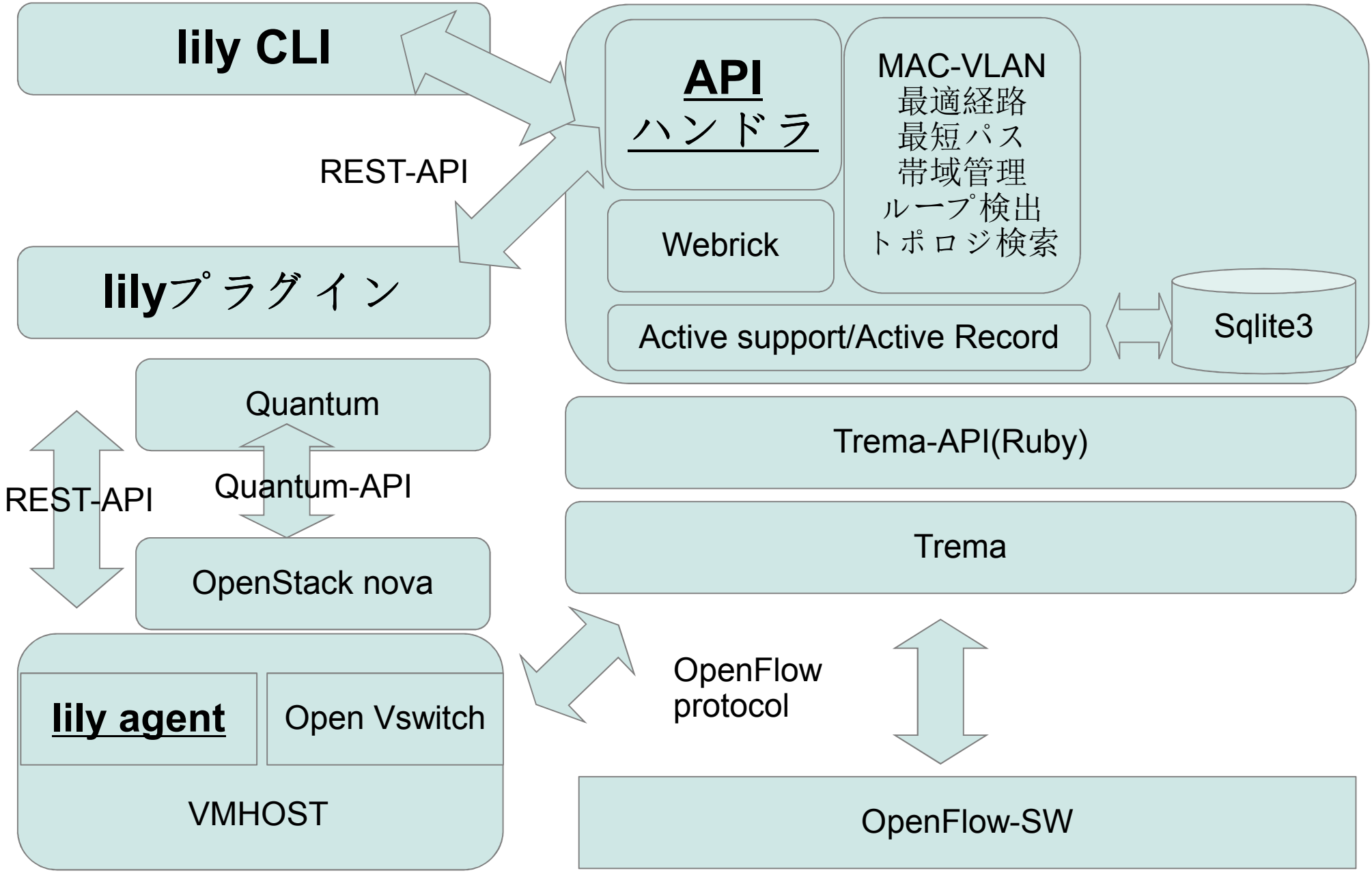
自作OpenFlowコントローラのゴール

- Openで機能拡張ができる
- トラブルシューティングが簡単
- 最適経路選択、VMマイグレーション追跡、障害迂回、論理ネット分割などのリッチな機能
- 導入が簡単
- 適当に配線しても動作し、既存ネットとの統合も容易
- 故障時のスイッチ交換も簡単

lilyの特徴

- MACアドレスベースVLAN(完)
- 物理ネットワークトポロジの自動検出(完)
- ループの自動検出(完)
- 最短パス検索(完)
- 障害箇所の迂回(完)
- 帯域制御(OpenVswitch)(完)
- 帯域プール(完)
- OpenStack Quantum API 1.0対応(50%)
- CLIによる操作(90%)

アーキテクチャ

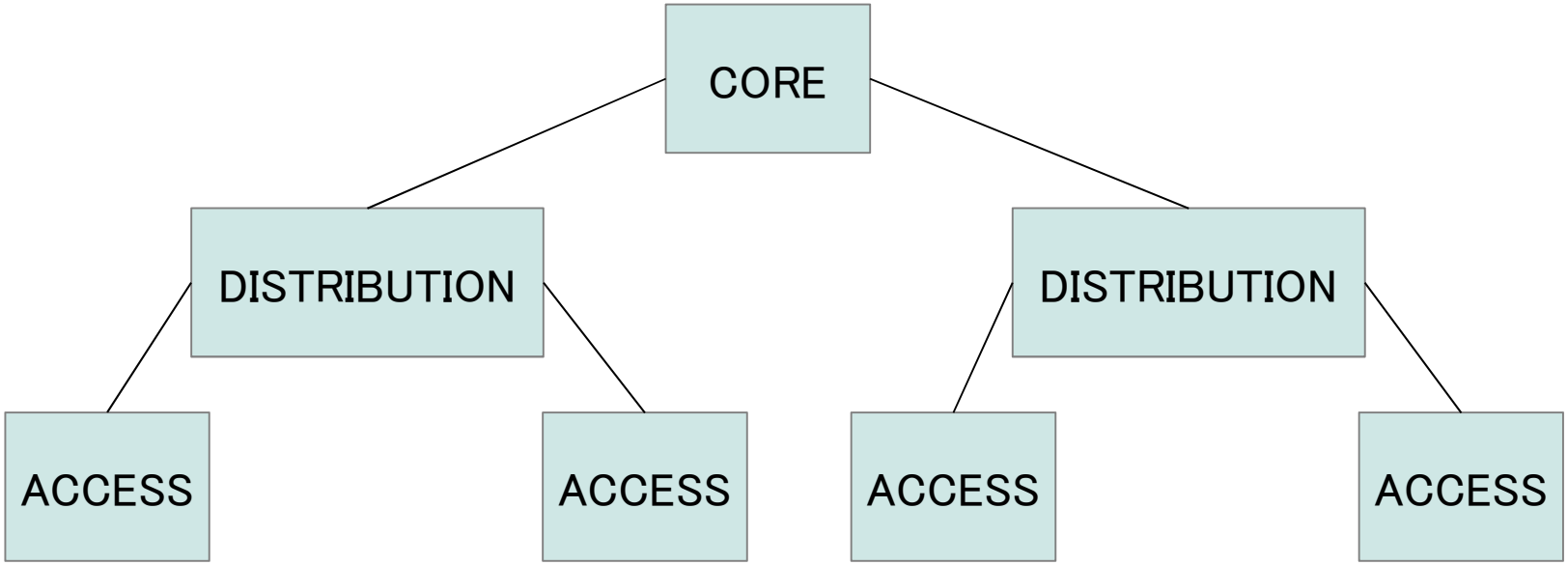

lily CLI
REST-API
lilyプラグイン
API
ハンドラ
MAC-VLAN
最適経路
最短パス
帯域管理
ループ検出
トポロジ検索
Webrick
Active support/Active Record
Sqlite3
Quantum
REST-API
Quantum-API
OpenStack nova
Trema-API(Ruby)
Trema
lily agent
Open Vswitch
VMHOST
OpenFlow
protocol
OpenFlow-SW

Quantum-API対応状況

API	APIハンドラ(本体)	プラグイン
List Networks	○	
List Networks Detail	○	
Show Network	○	
Show Network Detail	○	
Create Network	○	○
Update Network	○	
Delete Network	○	
List Ports	○	
List Ports Detail	○	
Show Port	○	
Show Port Detail	○	
Create Port	○	
Update Port	○	
Delete Port	○	
Show Attachment for Port	○	
Plug Attachment into Port	○	
Unplug Attachment from Port	○	

スイッチ一覧表示

```
[root@lily cli]# ./lily.rb sw
          name|                dpid|                role|  ip address(port no)
 -------------+ -------------------+ -------------------| --------------------
            a1|                 161|              ACCESS|      127.0.0.1/59022
            a2|                 162|              ACCESS|      127.0.0.1/59023
            a4|                 164|              ACCESS|      127.0.0.1/59025
            a3|                 163|              ACCESS|      127.0.0.1/59024
            d1|                 209|        DISTRIBUTION|      127.0.0.1/59026
            c1|                 193|                CORE|      127.0.0.1/59028
            d2|                 210|        DISTRIBUTION|      127.0.0.1/59027
[root@lily cli]#
```



トポロジ表示

```
[root@lily cli]# ./lily.rb topo
a1:3 <-> d1:3
a1:2 <-> d1:2
a2:1 <-> d1:4
a4:2 <-> d2:3
a3:2 <-> d2:4
a3:1 <-> d2:2
d2:1 <-> c1:2
d1:1 <-> c1:1
ACCESS LINK=>[a1:1]
ACCESS LINK=>[a4:1]
[root@lily cli]#
```

帯域プール

```
[root@lily cli]# ./lily bw
          name|band width(Mbps)
 ---------+ ----------------
             a1|          20000
             a2|          10000
             a4|          10000
             a3|          20000
band width pool(system total) = 60000Mbps
[root@lily cli]#
```

論理ネット作成・表示・変更・削除

【作成】
root@lily:/home/miyakz/trema/src/examples/lsw/cli# ./lily network create --name net1
<network id="03cde507-518e-885f-2ace-10f645c0f961"/>

【表示】
root@lily:/home/miyakz/trema/src/examples/lsw/cli# ./lily network list --detail
<networks>
  <network name="net1" id="03cde507-518e-885f-2ace-10f645c0f961"/>
</networks>

【変更】
root@lily:/home/miyakz/trema/src/examples/lsw/cli# ./lily network modify --name net1 --new_name new_net1

【削除】
root@lily:/home/miyakz/trema/src/examples/lsw/cli# ./lily network delete --name new_net1
INFO: deleting new_net1, detail(id=03cde507-518e-885f-2ace-10f645c0f961)
root@lily:/home/miyakz/trema/src/examples/lsw/cli#

lilyの課題

- MACアドレスベースVLANであり、VM間の通信フローを1つ1つ定義するため、フロー数が爆発（OpenFlowコントローラ共通の問題）
  - MPLS/802.1qを使用する
  - OpenFlowコンソーシアムへの働きかけ
- ライブマイグレーション未対応(Quantum対応で解決?)
- 性能(ruby)/スケールしない(作りの問題)
- GUI
- 信頼性
- 帯域プール+OpenStack(or ROR)
- Quantum APIの残りへの対応

最後に・・・
SDN/OpenFlowを
一緒に研究しませんか

- SDN勉強会(主催)
  - https://groups.google.com/forum/?fromgroups#!forum/sdnstudy
- クラウドネットワーク研究会
  - http://kokucheese.com/event/index/27170/

皆さんのお知恵やアドバイスを結集して、
OpenFlow界のLinuxみたいなコントローラを作ってみませんか。