クラウドインフラにおける xFlowの応用について

さくらインターネット（株）
研究所 大久保修一

Agenda

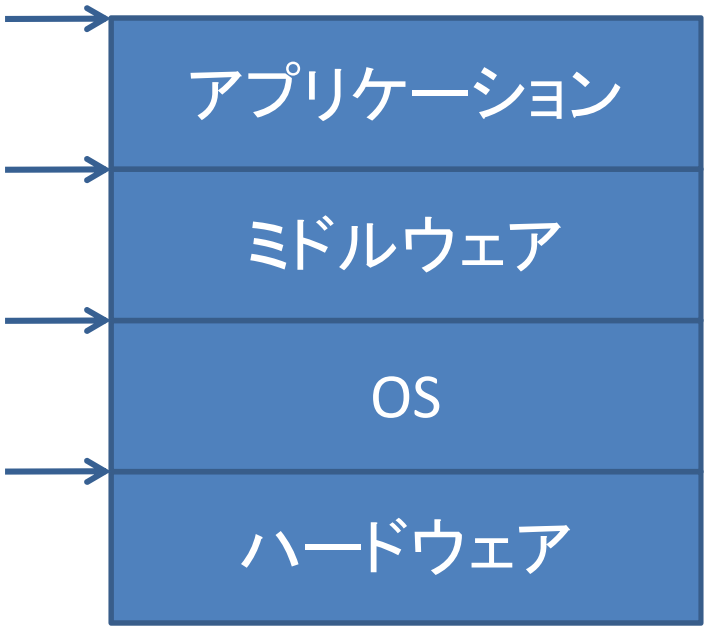
- 自己紹介
- クラウドとは？
- IaaSのネットワーク構成
- IaaSネットワークにおける通信フローの懸念点
- 仮想スイッチによるxFlow
- 実験環境と実験結果
- その他クラウドにおけるxFlowのメリット
- まとめ
- 議論

クラウドとは？

- NISTの定義がよく引用される
- 五つの特徴
  - On-demand self-service
    必要に応じてコンピュータリソースを利用可能
  - Resource pooling
    マルチテナントモデル、リソース配置場所の隠蔽
  - Rapid elasticity
    スケールアウト可能
  - Broad network access
    様々なプラットフォームから利用可能
  - Measured Service
    計測によるサービスの透明性

クラウドとは？

- 三つのサービスモデル
  - SaaS (Software as a Service)
  - PaaS(Platform as a Service)
  - IaaS(Infrastructure as a Service)
  - HaaS(Hardware as a Service)



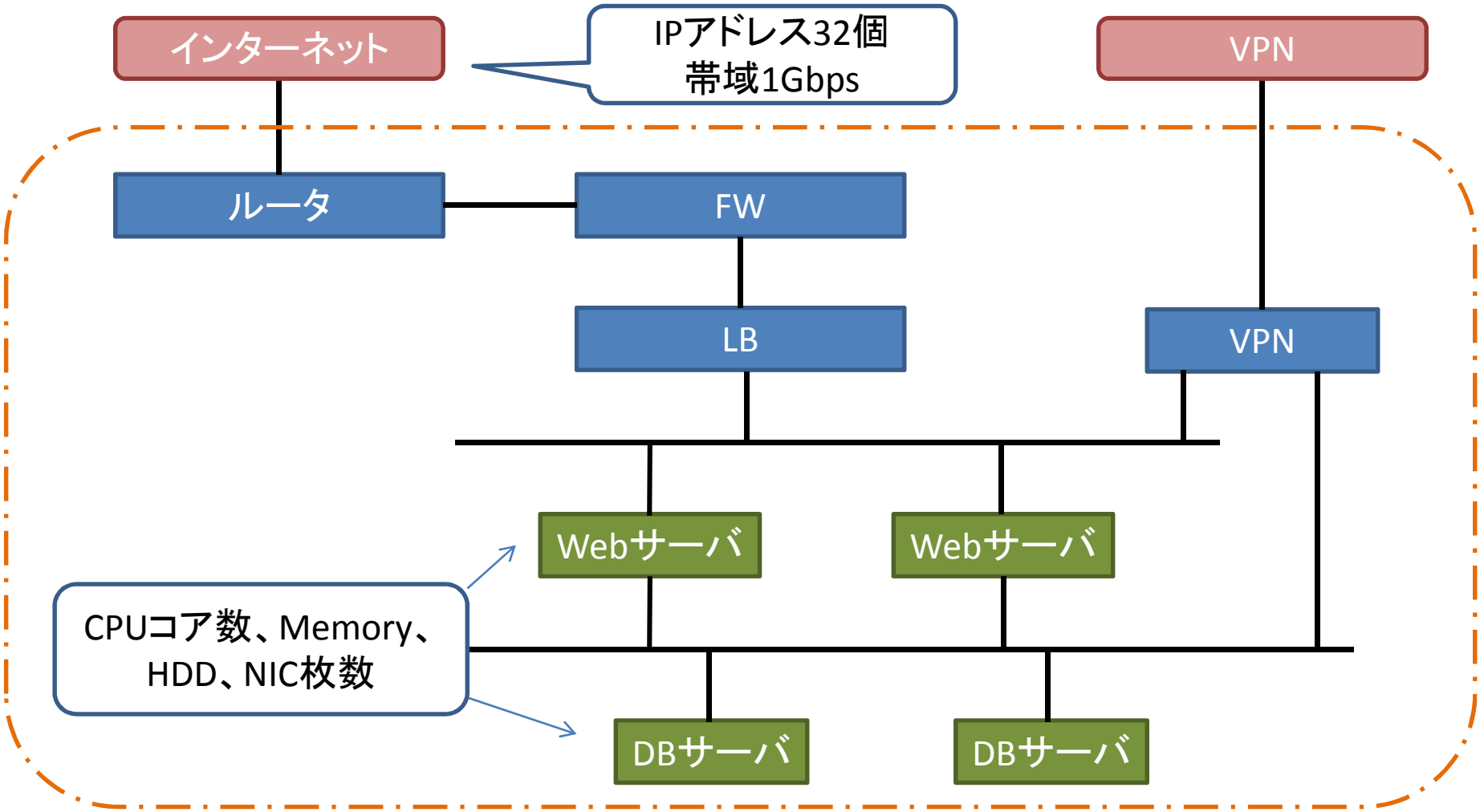
今回は、IaaSの話をします。

IaaSで実現できること

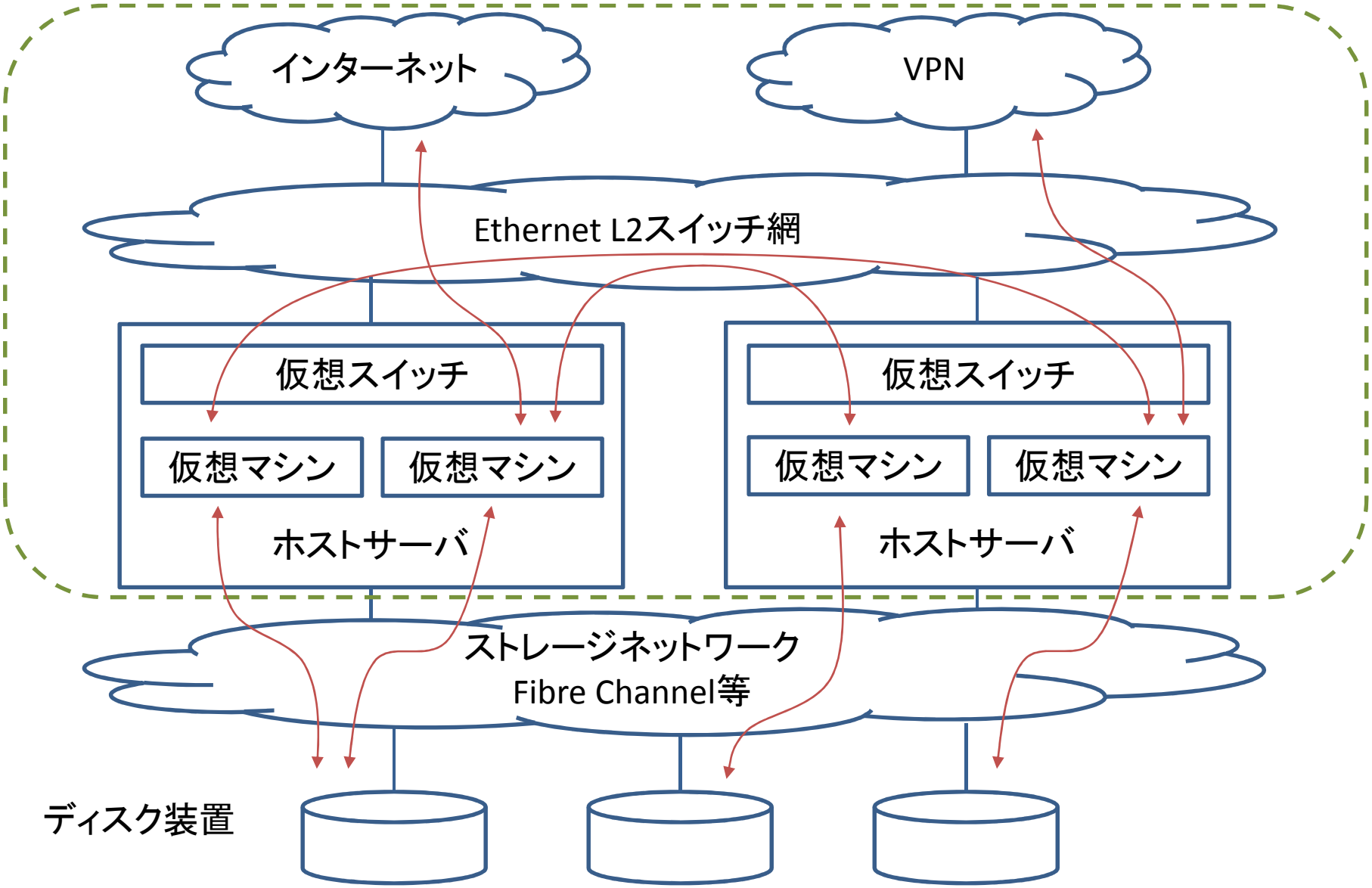
- 仮想マシンを自由に構成して作成できる
  （例）
  - CPUコア数: 13個
  - メモリ: 5.7GB
  - HDD: 987GB
  - NIC: 5枚
- 仮想ネットワークを自由に構成できる
- 様々な仮想アプライアンスを使える
  - 例：仮想ルータ、仮想FW、仮想LB
- 利用実績に応じた支払い

IaaS上でのシステム構築例

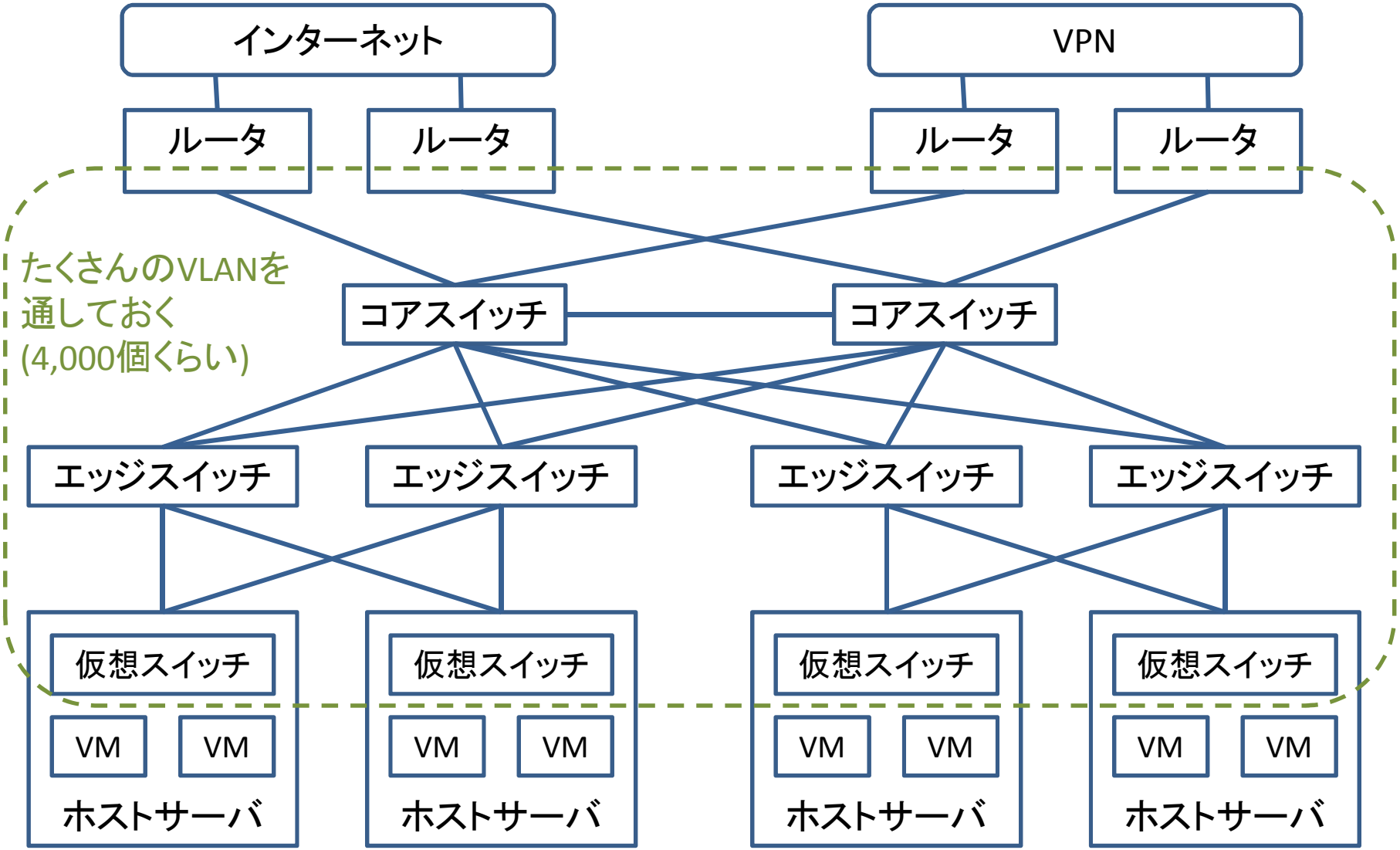
利用者がサーバ構成、ネットワーク構成を定義



IaaSインフラの構成

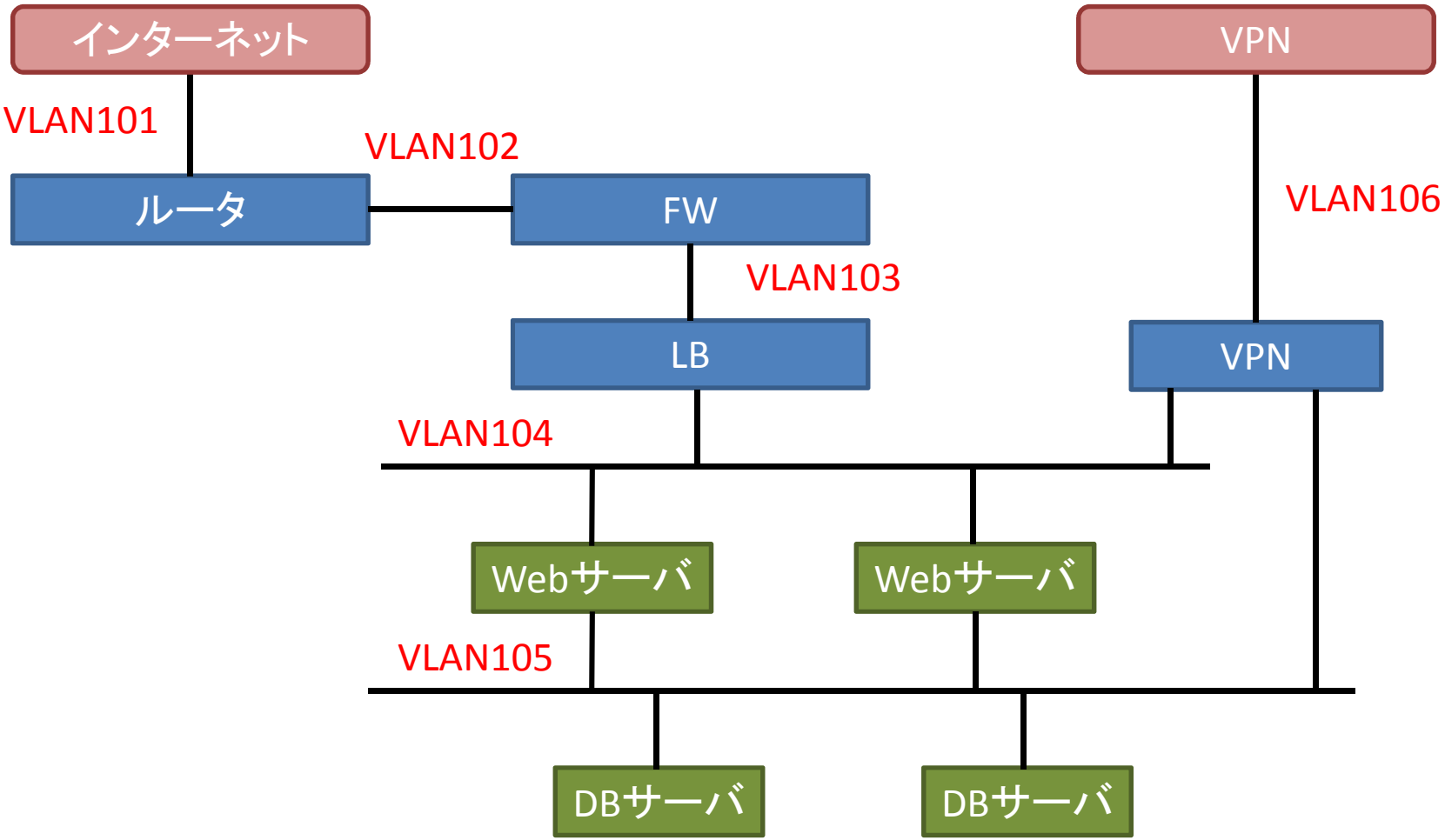


IaaSネットワークの物理構成

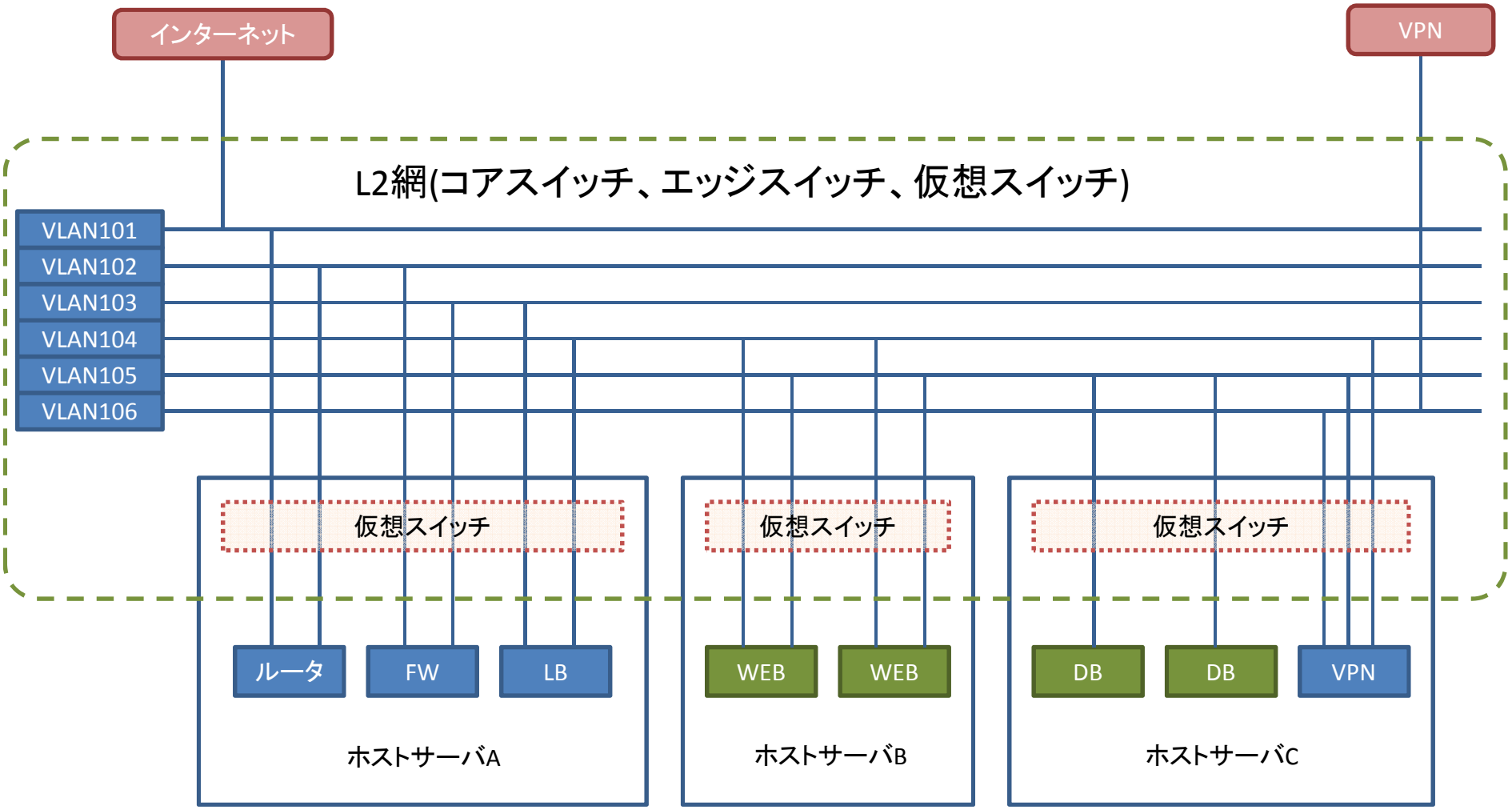


仮想システムプロビジョニング例

これを、IaaSインフラ上に展開してみる



クラウド上に配置したシステム

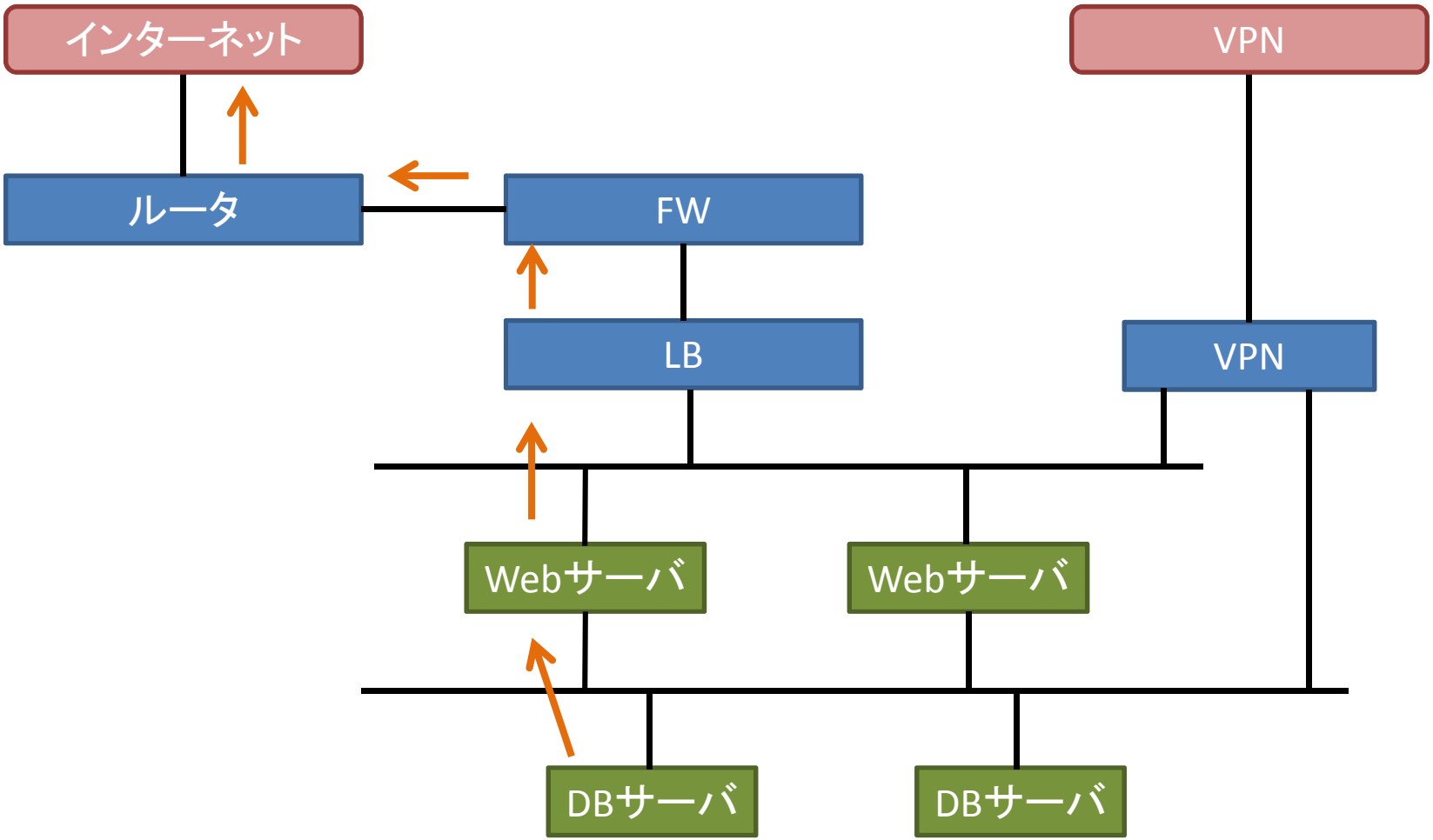


IaaSネットワークの懸念点

- 全てのホストサーバから全てのVLANに接続可能。
- VMはどこのホストサーバにあっても通信可能。
- それ故に、仮想マシンの配置方法によっては物理的に非効率な通信が発生する可能性がある。

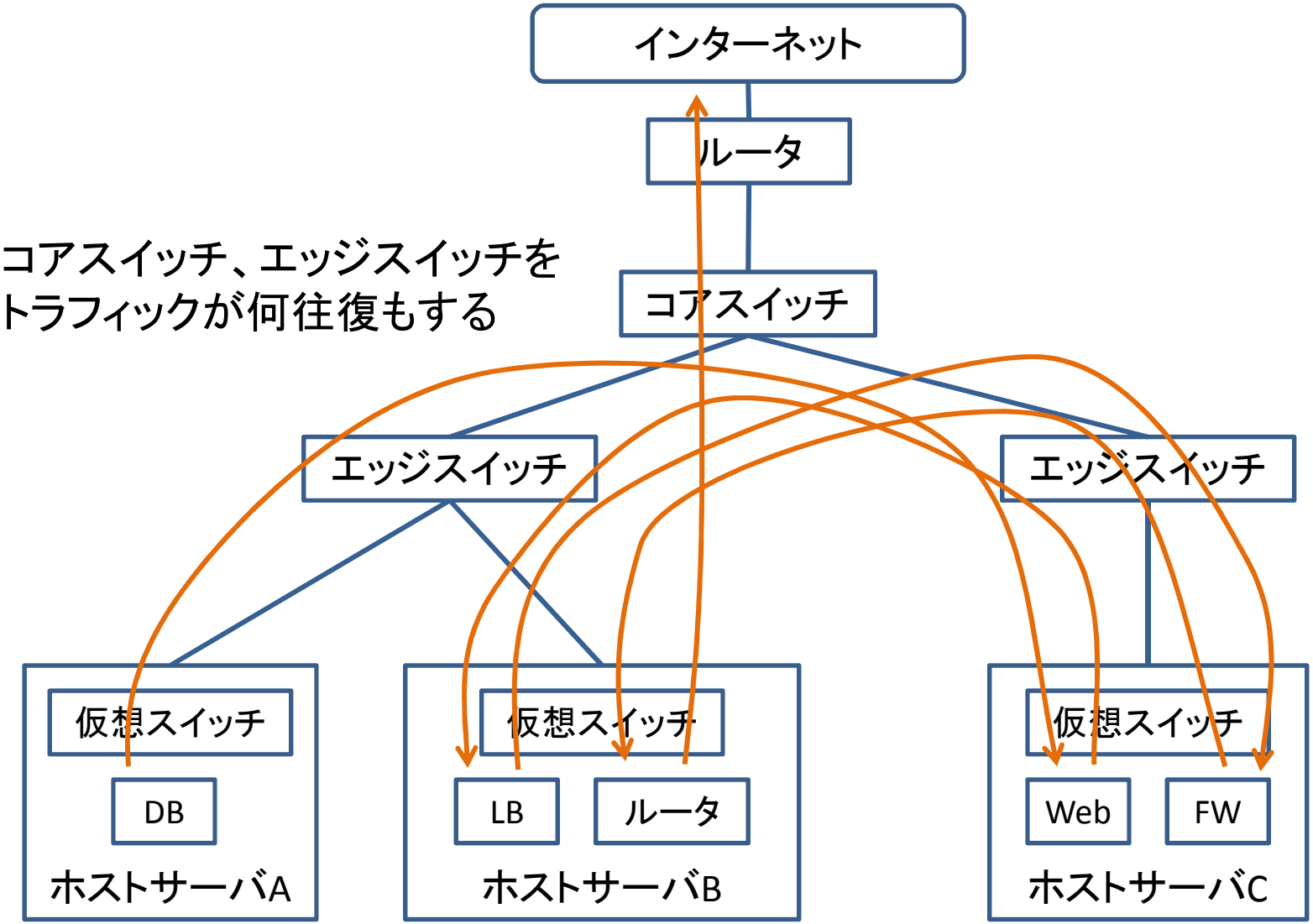
トラフィックフローの例

DBサーバに入っているデータが、各機器を経由してインターネットに抜ける

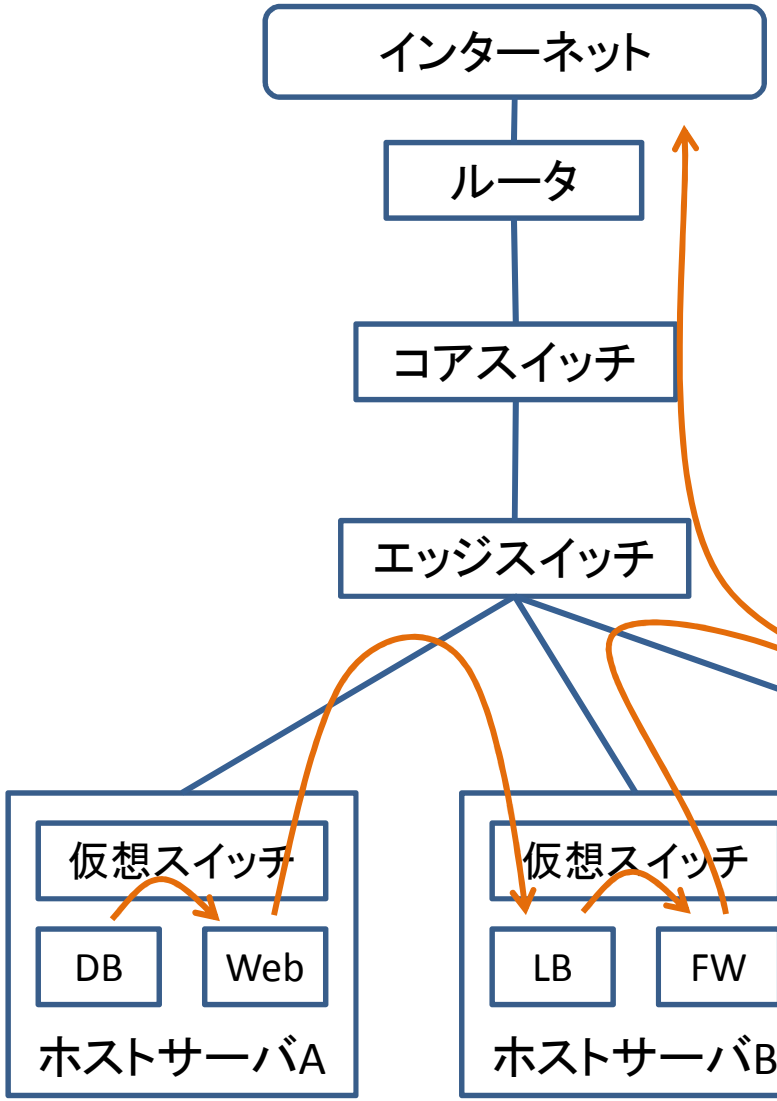


非効率な通信の例

コアスイッチ、エッジスイッチを
トラフィックが何往復もする



効率的な通信の例



通信が発生しやすい同一顧客のVMは、
・リソースが許すなら同じホスト上に
・無理なら、なるべく同じエッジスイッチ配下に
なるべく配置する。

が、そうはうまくいかない（＞＜）

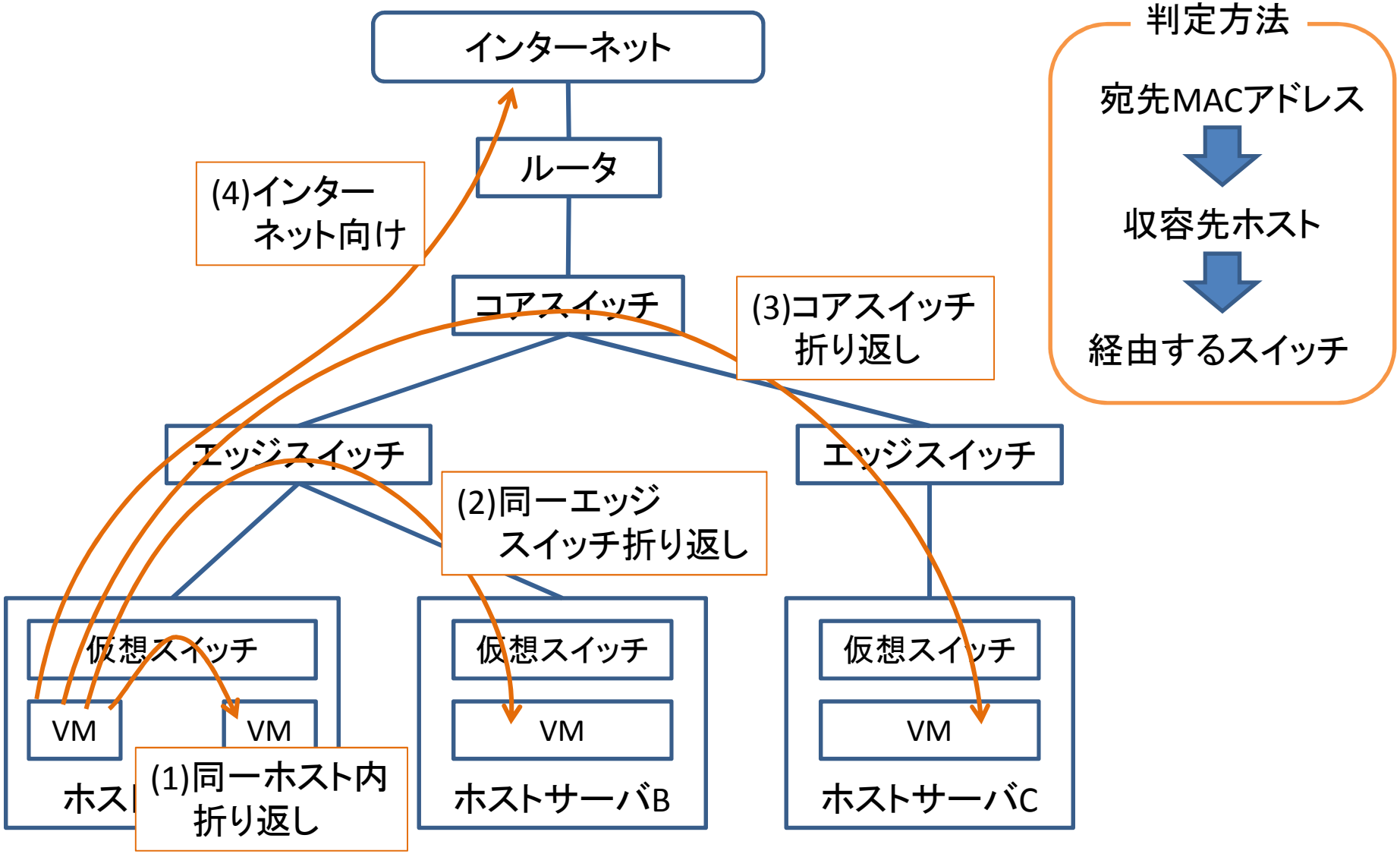
効率的に通信されているか
観測する必要がある。

そうだ！

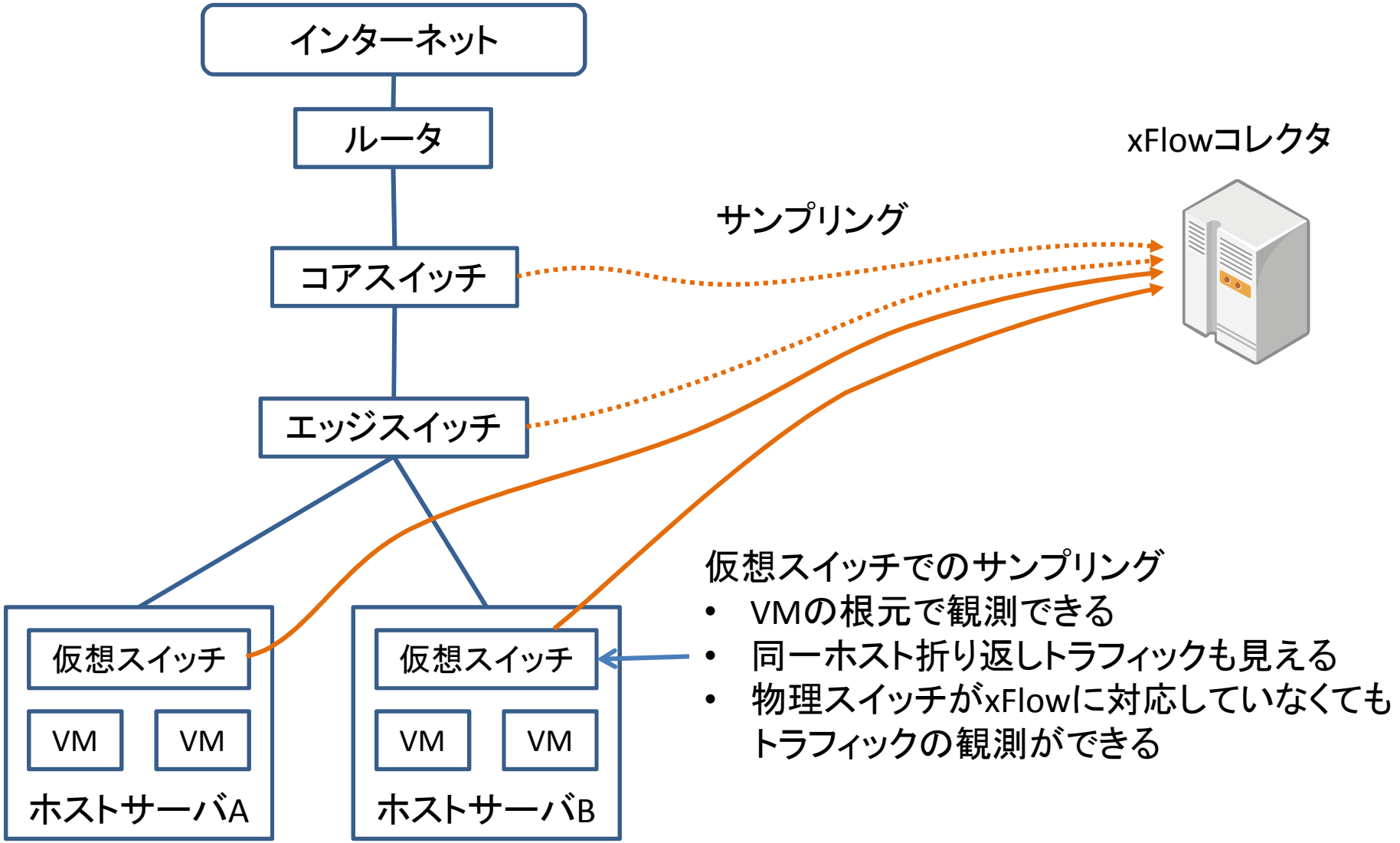
- クラウドネットワークのトラフィックフローを見える化しよう！
- 僕らにはxFlowがあるじゃないか！

- xFlowを用いて、トラフィックフローを分類
  1. 同一ホスト内折り返し　○　良い
  2. 同一エッジスイッチ内折り返し　△　まあ良い
  3. コアスイッチ折り返し　×　よくない
  4. インターネット向け　○　しかたない

トラフィックの分類


インターネット
ルータ
コアスイッチ
エッジスイッチ
エッジスイッチ
仮想スイッチ
VM
VM
ホスト
仮想スイッチ
VM
ホストサーバB
仮想スイッチ
VM
ホストサーバC
(1)同一ホスト内
折り返し
(2)同一エッジ
スイッチ折り返し
(3)コアスイッチ
折り返し
(4)インター
ネット向け
判定方法
宛先MACアドレス
収容先ホスト
経由するスイッチ

xFlowを用いたサンプリング



xFlowを実装している仮想スイッチ

- Cisco Nexus 1000V
  - http://www.cisco.com/web/JP/product/hs/switches/nexus1000/index.html
  - VMware ESX + vCenter Server + vDS(vNetwork Distributed Switch)の組み合わせで動作
- Open vSwitch
  - http://openvswitch.org/
  - Linux標準ブリッジの代替として利用可能
  - カーネルモジュールをロードして使用

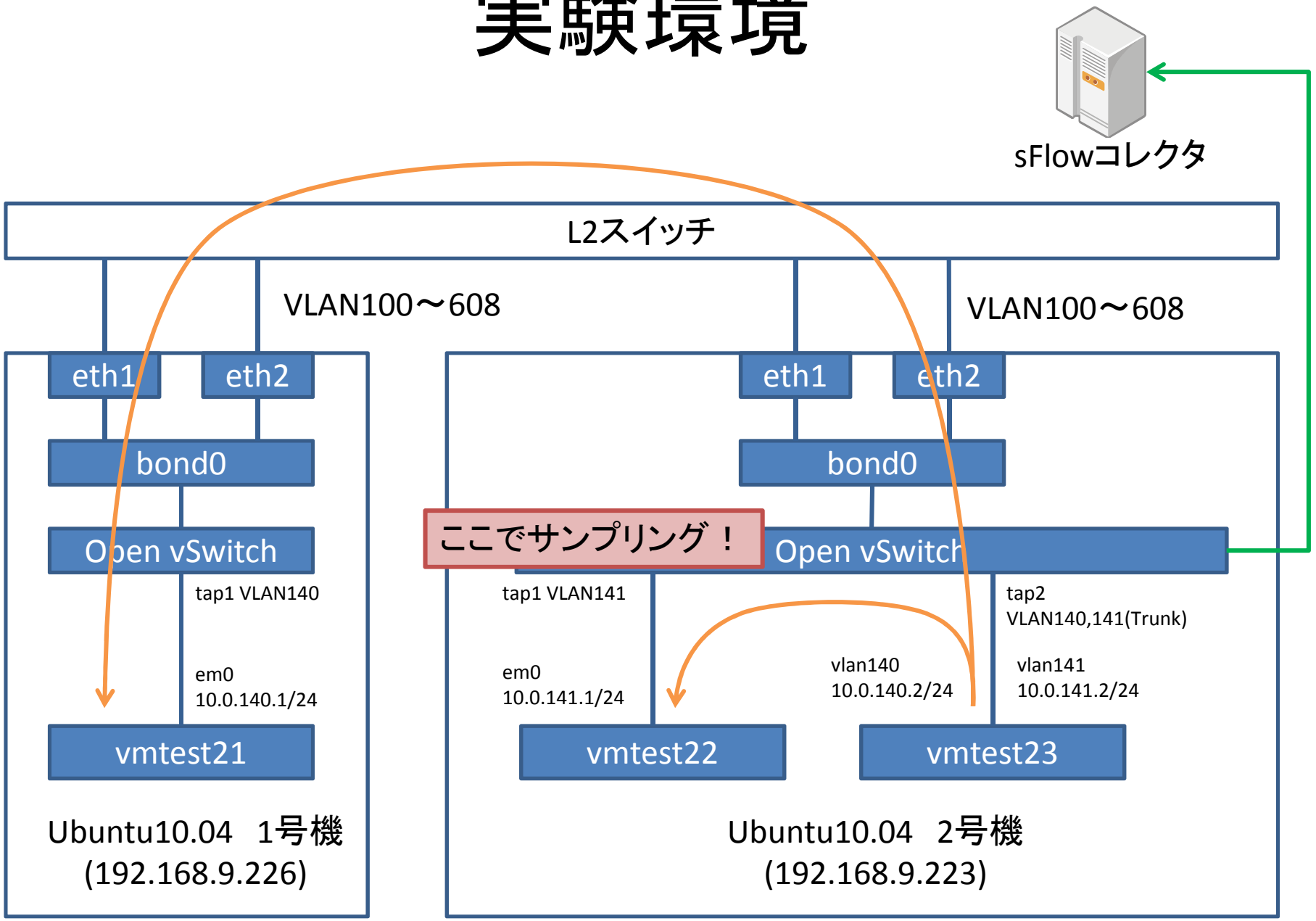
仮想スイッチ機能比較

機能	Linux標準ブリッジ	Open vSwitch	Nexus1000V
対応ハイパーバイザ	KVM, Xen	KVM, Xen	Vmware
VLAN制御	×	○	○
パケットフィルタ	×	×	○
ポリシング	×	○	○
sFlow	×	○	×
NetFlow	×	○	○
ポートミラー	×	○	○
分散スイッチ	×	OpenFlow	vDS
SNMP	×	×	○
CLI	×	×	○

Open vSwitch実験環境概要

- 2台のUbuntu10.04サーバを用意
- Open vSwitch1.1.0pre2をインストール
- KVMの仮想マシンを3台セットアップし、Open vSwitchに接続
- 1つは、同一ホスト折り返しのトラフィックを、
- もう1つは、スイッチ経由のトラフィックを流す。
- 上記トラフィックをsFlowで分離計測

実験環境


sFlowコレクタ
L2スイッチ
VLAN100～608
VLAN100～608
eth1
eth2
eth1
eth2
bond0
bond0
Open vSwitch
ここでサンプリング！
Open vSwitch
tap1 VLAN140
em0
10.0.140.1/24
tap1 VLAN141
em0
10.0.141.1/24
vlan140
10.0.140.2/24
tap2
VLAN140,141(Trunk)
vlan141
10.0.141.2/24
vmtest21
vmtest22
vmtest23
Ubuntu10.04　1号機
(192.168.9.226)
Ubuntu10.04　2号機
(192.168.9.223)

Open vSwitchのsFlowの設定(1)

ホストサーバ上でコマンド実行

```
# ovs-vsctl create sFlow agent=eth0 targets=61.211.224.207 sampling=1024
21c0cf5e-a2aa-45bd-afe5-9e7566d3f21f

# ovs-vsctl list sFlow
_uuid               : 21c0cf5e-a2aa-45bd-afe5-9e7566d3f21f
agent               : "eth0"
external_ids        : {}
header              : []
polling             : []
sampling            : 1024
targets             : ["61.211.224.207"]
```

マニュアル
http://openvswitch.org/ovs-vswitchd.conf.db.5.pdf

Open vSwitchのsFlowの設定(2)

ホストサーバ上でコマンド実行

```
# ovs-vsctl set Bridge br0 sflow=21c0cf5e-a2aa-45bd-afe5-9e7566d3f21f

# ovs-vsctl list Bridge
_uuid               : d339a52e-e9d6-41ea-8ab6-2dec4558b9f6
controller          : []
datapath_id         : "0000005056890010"
datapath_type       : ""
external_ids        : {}
fail_mode           : []
flood_vlans         : []
mirrors             : []
name                : "br0"
netflow             : []
other_config        : {}
ports               : [4832f6db-540d-4fb9-803d-f33fc26a2169,
                       7105d8dc-028f-4ad4-bd16-42ea32759983,
                       d0e70424-4710-4577-abf7-b02efb8fbede,
                       fe07e6bb-6211-4855-8b92-dcb1d7932958]
sflow               : 21c0cf5e-a2aa-45bd-afe5-9e7566d3f21f
```

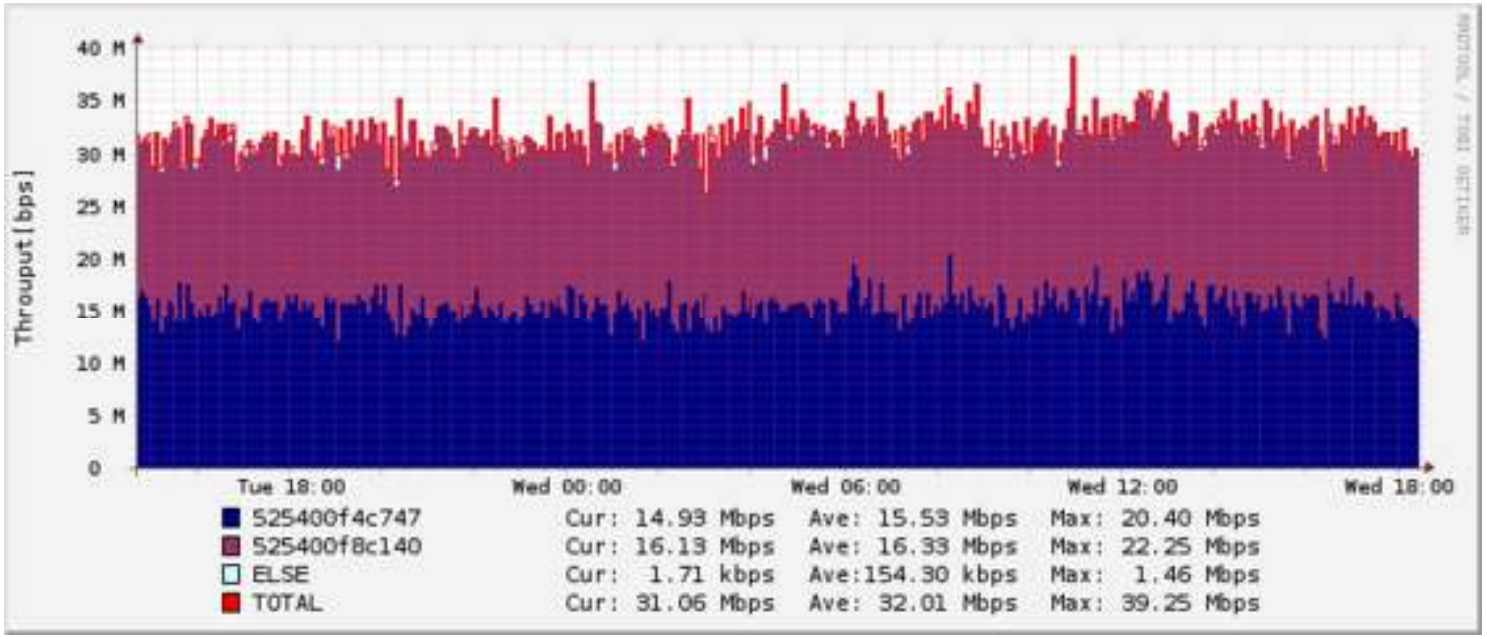
サンプリングしてみた

```
% sflowtool
startDatagram =================
datagramSourceIP 59.106.69.99
datagramSize 460
unixSecondsUTC 1301981403
datagramVersion 5
agent 192.168.9.223
packetSequenceNo 2573
sysUpTime 2812000
samplesInPacket 2
startSample ----------------
sampleType_tag 0:1
sampleType FLOWSAMPLE
sampleSequenceNo 7412
sourceId 0:12
meanSkipCount 1024
samplePool 7799792
inputPort 12
outputPort multiple 2
flowBlock_tag 0:1001
extendedType SWITCH
in_vlan 140
```

```
out_vlan 140
flowBlock_tag 0:1
flowSampleType HEADER
headerProtocol 1
sampledPacketSize 1522
strippedBytes 4
headerLen 128
headerBytes 52-54-00-F4-C7-47-<省略>
dstMAC 525400f4c747
srcMAC 52540044fc55
decodedVLAN 140
decodedPriority 0
IPSize 1500
ip.tot_len 1500
srcIP 10.0.140.2
dstIP 10.0.140.1
IPProtocol 17
IPTOS 0
IPTTL 64
UDPSrcPort 13584
UDPDstPort 57361
UDPBytes 1480
```

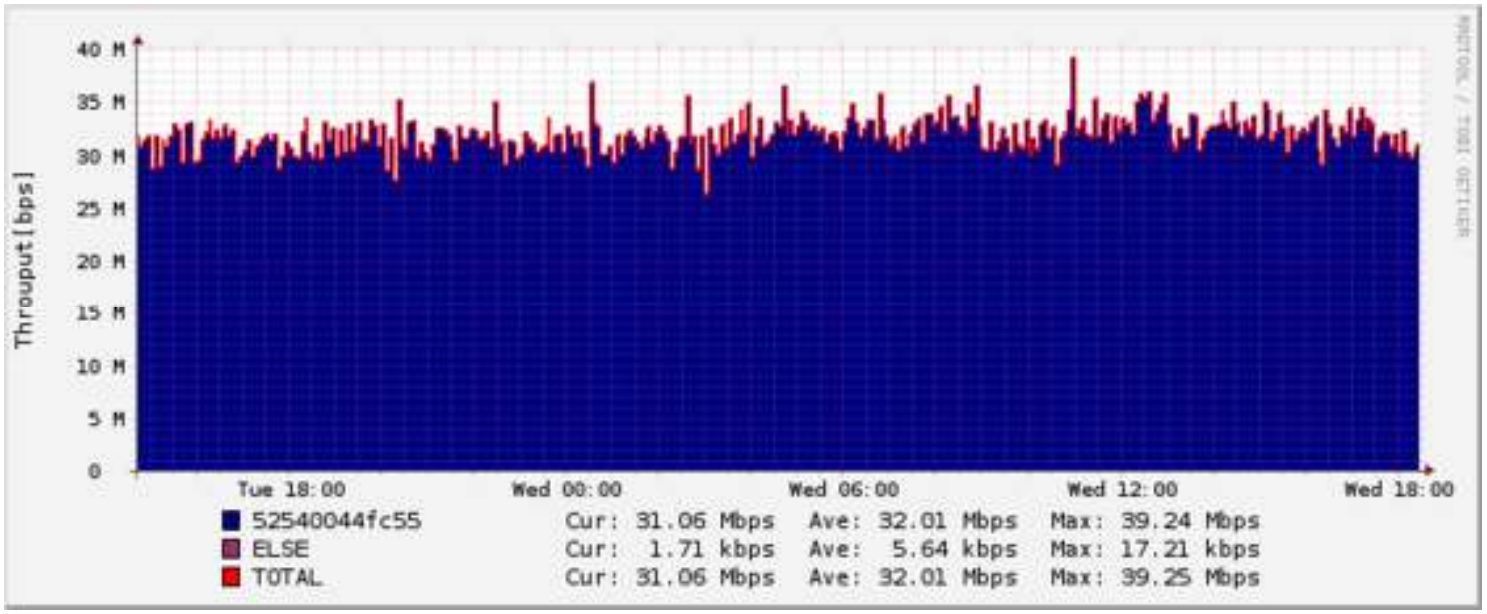
グラフ化してみた

宛先MACアドレス別



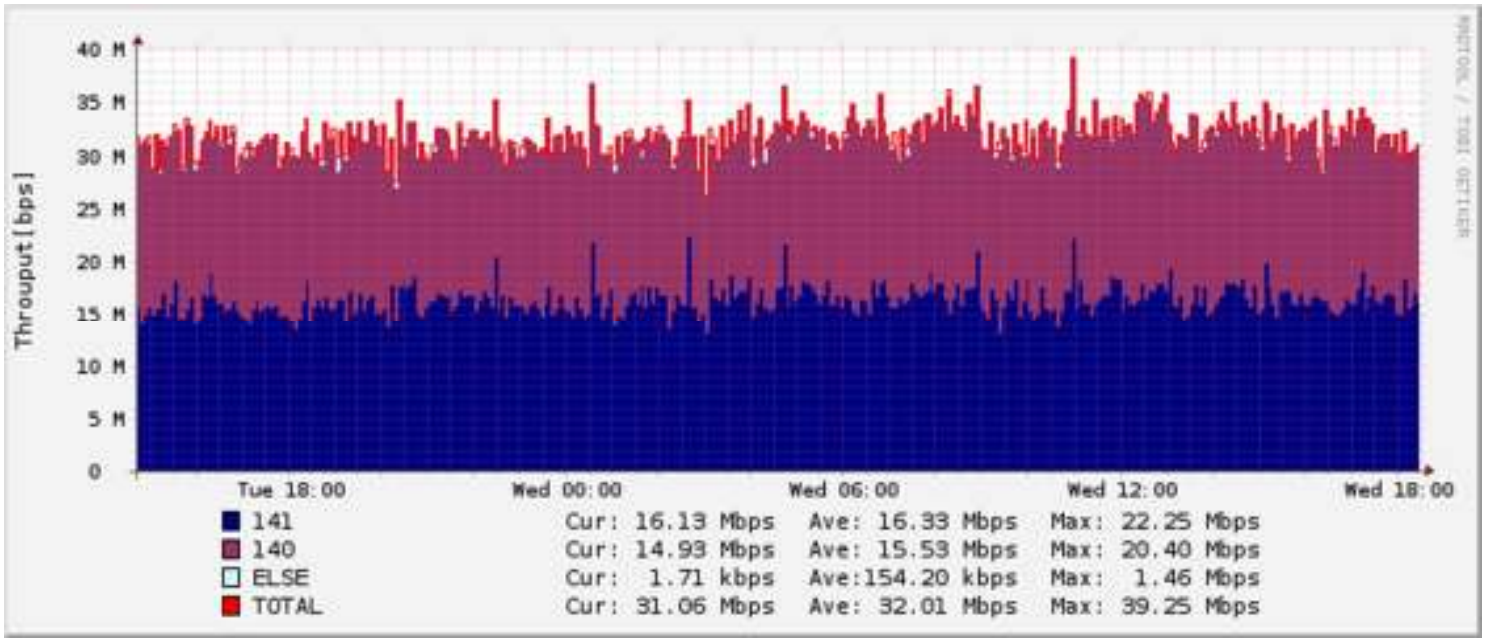
グラフ化してみた

送信元MACアドレス別



グラフ化してみた

VLAN別



Cisco Nexus1000Vの場合

- Nexus1000VのNetFlowエクスポート機能は、MACアドレスの収集に対応していない。
- クラウド網内では、プライベートアドレス等、同じIPアドレスが別VLANで使用されることもあるため、MACアドレスの情報が必要。

Cisco Nexus1000Vの設定

```
flow exporter test-exporter
  destination 61.211.224.207
  source mgmt0
  version 9

flow monitor test-monitor
  record netflow-original
  exporter test-exporter
  timeout active 60
  cache size 4096

interface Vethernet1
  ip flow monitor test-monitor output

interface Vethernet2
  ip flow monitor test-monitor output
```

Cisco Nexus1000Vの設定

```
TEST-VSM# conf t
TEST-VSM(config)# flow record test-record
TEST-VSM(config-flow-record)# ?
  collect      Specify a non-key field
  description  Provide a description for this Flow Record
  exit         Exit from command interpreter
  match        Specify a key field
  no           Negate a command or set its defaults

TEST-VSM(config-flow-record)# collect ?
  counter    Counters to collect
  timestamp  Timestamp fields
  transport  Transport layer fields
```

datalinkがない。。。

フローレコードのフィールド

```
TEST-VSM# show flow record netflow-original
Flow record netflow-original:
    Description: Traditional IPv4 input NetFlow with origin ASs
    No. of users: 1
    Template ID: 259
    Fields:
        match ipv4 source address
        match ipv4 destination address
        match ip protocol
        match ip tos
        match transport source-port
        match transport destination-port
        match interface input
        match interface output
        match flow direction
        collect routing source as
        collect routing destination as
        collect routing next-hop address ipv4
        collect transport tcp flags
        collect counter bytes
        collect counter packets
        collect timestamp sys-uptime first
        collect timestamp sys-uptime last
```

nfdumpで表示してみた

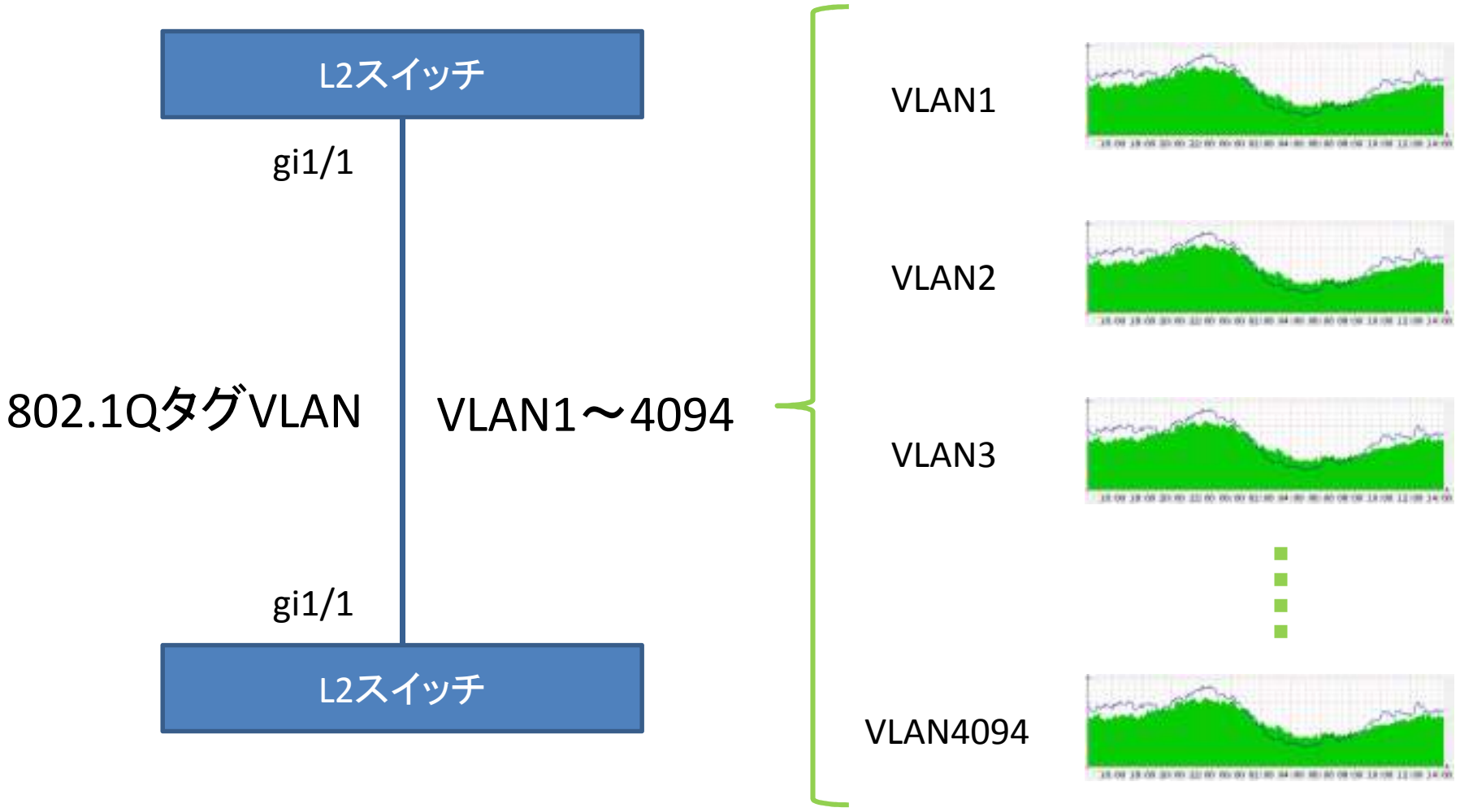
```
% nfdump -r nftest/nfcapd.201006251145
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2010-06-24 07:16:04.570     0.057 UDP           10.0.0.1:0     ->         10.0.0.2:0          251    62289     1
2010-06-24 07:16:04.747     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0           91    47856     1
2010-06-24 07:16:04.742     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0          196    45314     1
2010-06-24 07:16:04.724     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0          250    59428     1
2010-06-24 07:16:04.726     0.035 UDP           10.0.0.1:0     ->         10.0.0.2:0          225     5481     1
2010-06-24 07:16:04.628     0.060 UDP           10.0.0.1:0     ->         10.0.0.2:0          234    45852     1
2010-06-24 07:16:04.712     0.052 UDP           10.0.0.1:0     ->         10.0.0.2:0          103    59938     1
2010-06-24 07:16:04.699     0.060 UDP           10.0.0.1:0     ->         10.0.0.2:0          210    22803     1
2010-06-24 07:16:04.731     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0           79    39320     1
2010-06-24 07:16:04.713     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0          181    30755     1
2010-06-24 07:16:04.695     0.059 UDP           10.0.0.1:0     ->         10.0.0.2:0           26    56224     1
Summary: total flows: 11, total bytes: 475260, total packets: 1846, avg bps: 16.1 M, avg pps: 7822, avg bpp: 257
Time window: 2010-06-24 07:16:04 - 2010-06-24 07:16:04
Total flows processed: 11, Blocks skipped: 0, Bytes read: 596
Sys: 0.001s flows/second: 5774.3     Wall: 0.000s flows/second: 35947.7
```

IPアドレス、ポート番号の情報は取れる

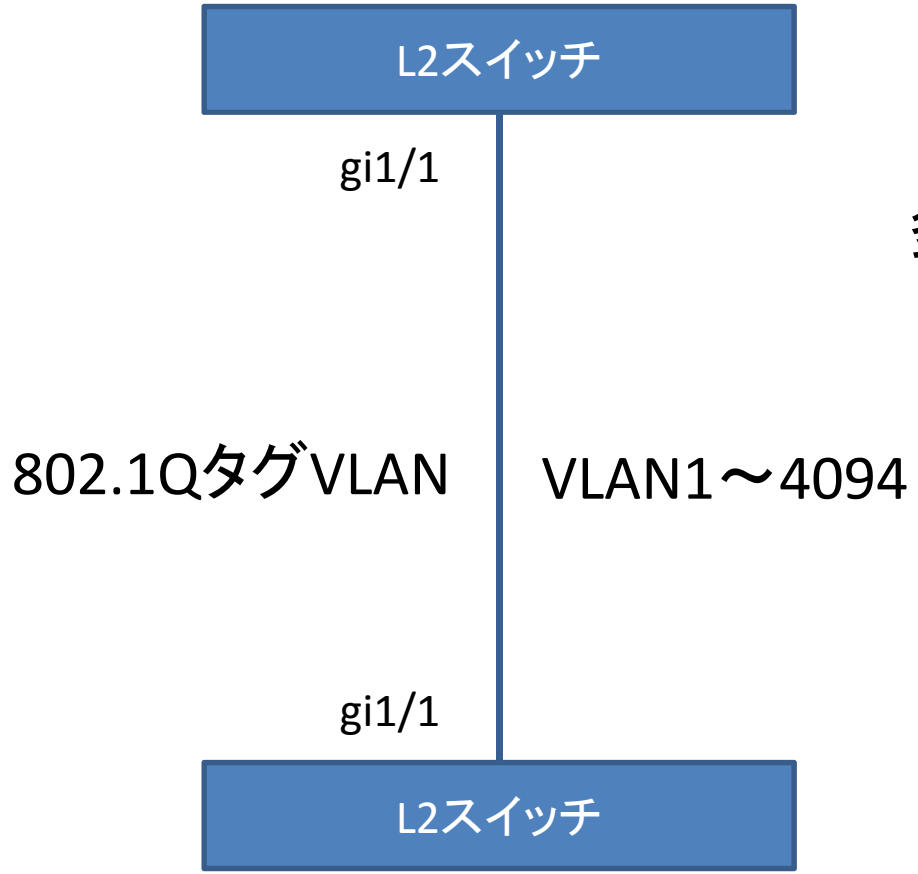
その他のxFlowのメリット

- 物理インターフェイスを流れるトラフィックのうち、どのVLAN(≒利用者)のものが多いのか？
- SNMPで取得すると、1物理インターフェイスあたり、4,000個のグラフが必要。
- 40ポートのL2スイッチだと、1台あたり16万個のグラフ。
- →現実的ではない
- xFlowでサンプリングして、トラフィックの多いVLANを積み上げてはどうか？

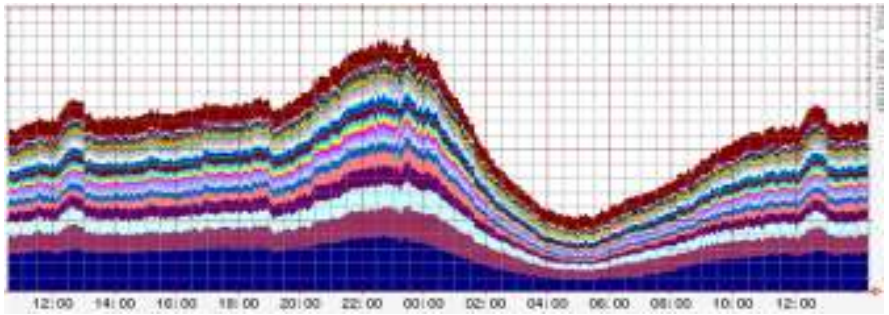
SNMPを用いたVLAN毎のトラフィック



xFlowを用いたVLAN毎のトラフィック



多いVLANを積み上げグラフで表示



まとめ

- IaaSネットワークインフラにおいて、仮想マシンの配置は通信フローの効率性に影響するため、重要になる。
- 通信フローが最適であるかを判定するために、xFlowによるトラフィック計測が有効である。
- xFlowエクスポート機能が実装された仮想スイッチもある。結構ちゃんと動く。