A Carrier SDN Lessons Learned

1/14/2016

KDDI株式会社 IPネットワーク部
大垣健一

Agenda

1. SDNの定義
2. Motivation
3. WVS2概要
4. Lessons Learned
   - ネットワークアーキテクチャ
   - 制御システムアーキテクチャ
5. SDN revisit
6. Telecom DevOps?

SDNの定義

- **コントロールプレーン(経路制御機能)/データプレーン(転送機能)分離**
  - **データプレーンのプログラマビリティを提供**
    - 任意のネットワーク制御を実現

- **Dr. Nick McKeownのインタビュー記事**
  - **Kate Greene, “TR10: Software Defined Networking,” MIT Technology Review, Mar. 2009, http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/**
  - **新しいルーティング/スイッチングプロトコルを大規模ネットワークで検証したかったが、ルータやスイッチはベンダに縛られていた**
  - **OpenFlowはデータフローをソフトウェアで定義できる**
  - ***“OpenFlow (snip...) define data flows using software—a sort of “software-defined networking.””***

Motivation

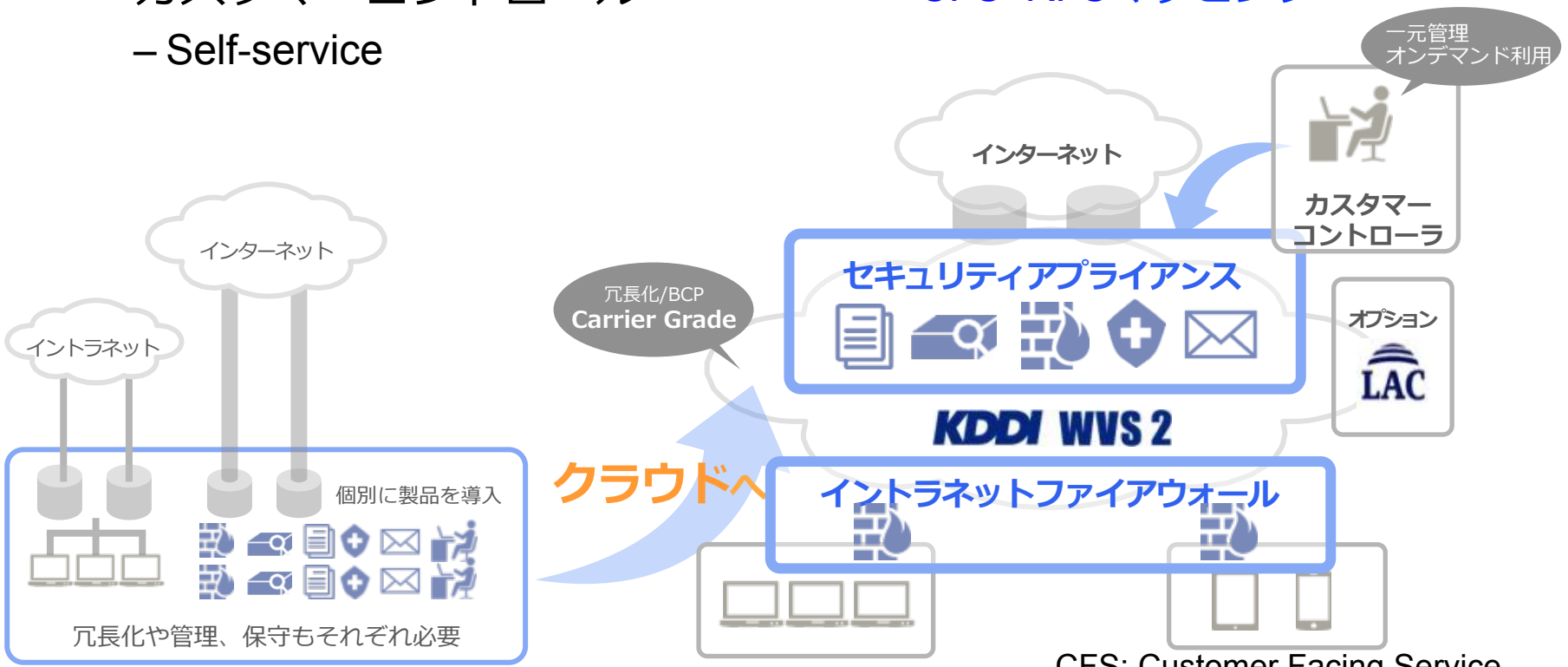
■ Wide Area Virtual Switch 2 (WVS2)

● セキュリティアプライアンスクラウド

- 所有から利用へ
  - クラウドモデル
  - Pay-as-you-go
- カスタマーコントロール
  - Self-service

- サービスチェイニング
  - D-planeプログラマビリティ
- 自動化&抽象化
  - CFS–RFSマッピング

SDN

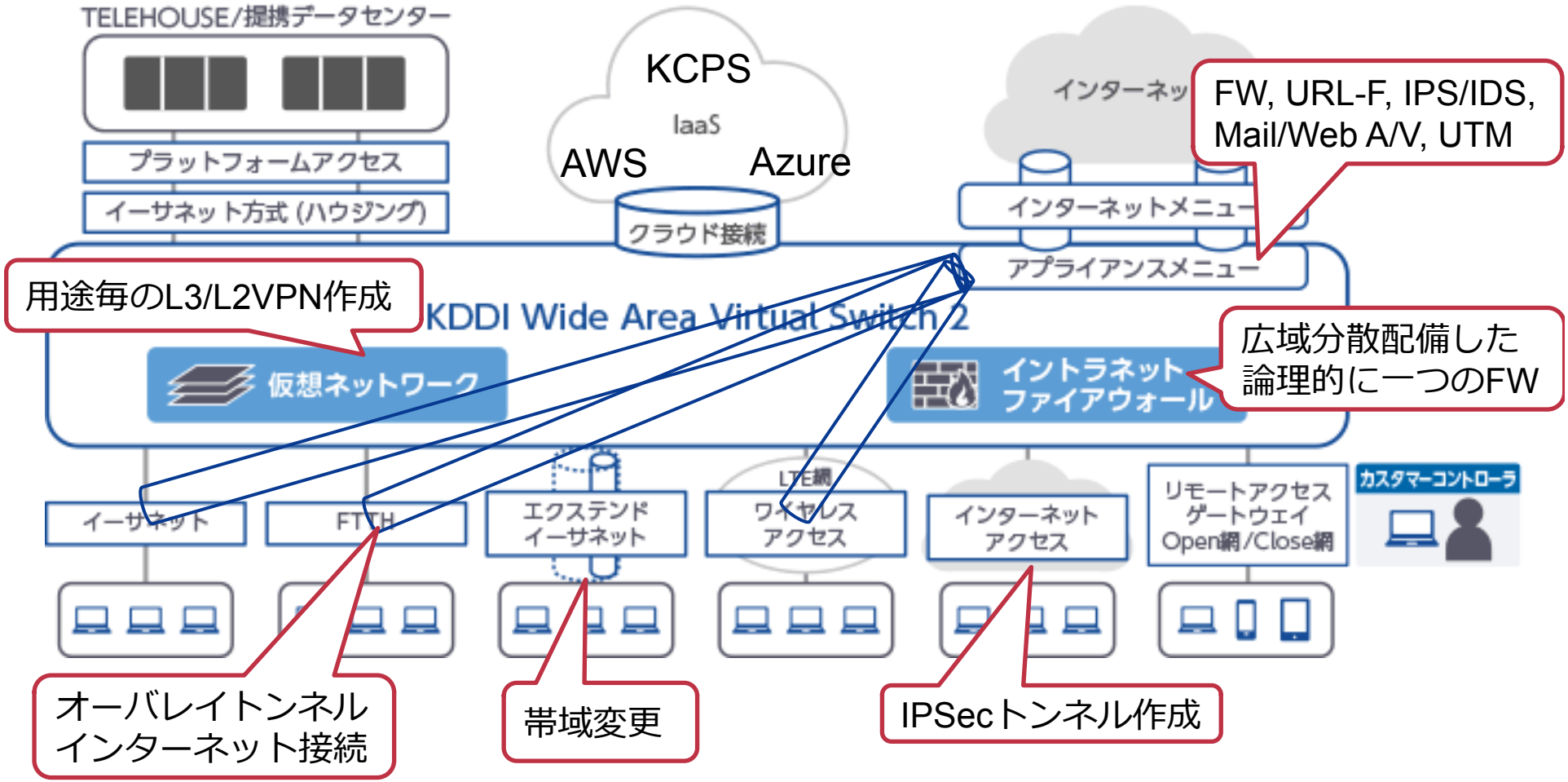


CFS: Customer Facing Service
RFS: Resource Facing Service

WVS2概要(1/2)

■ セキュリティアプライアンスクラウド
■ 仮想ネットワーク

} **オンデマンド提供**

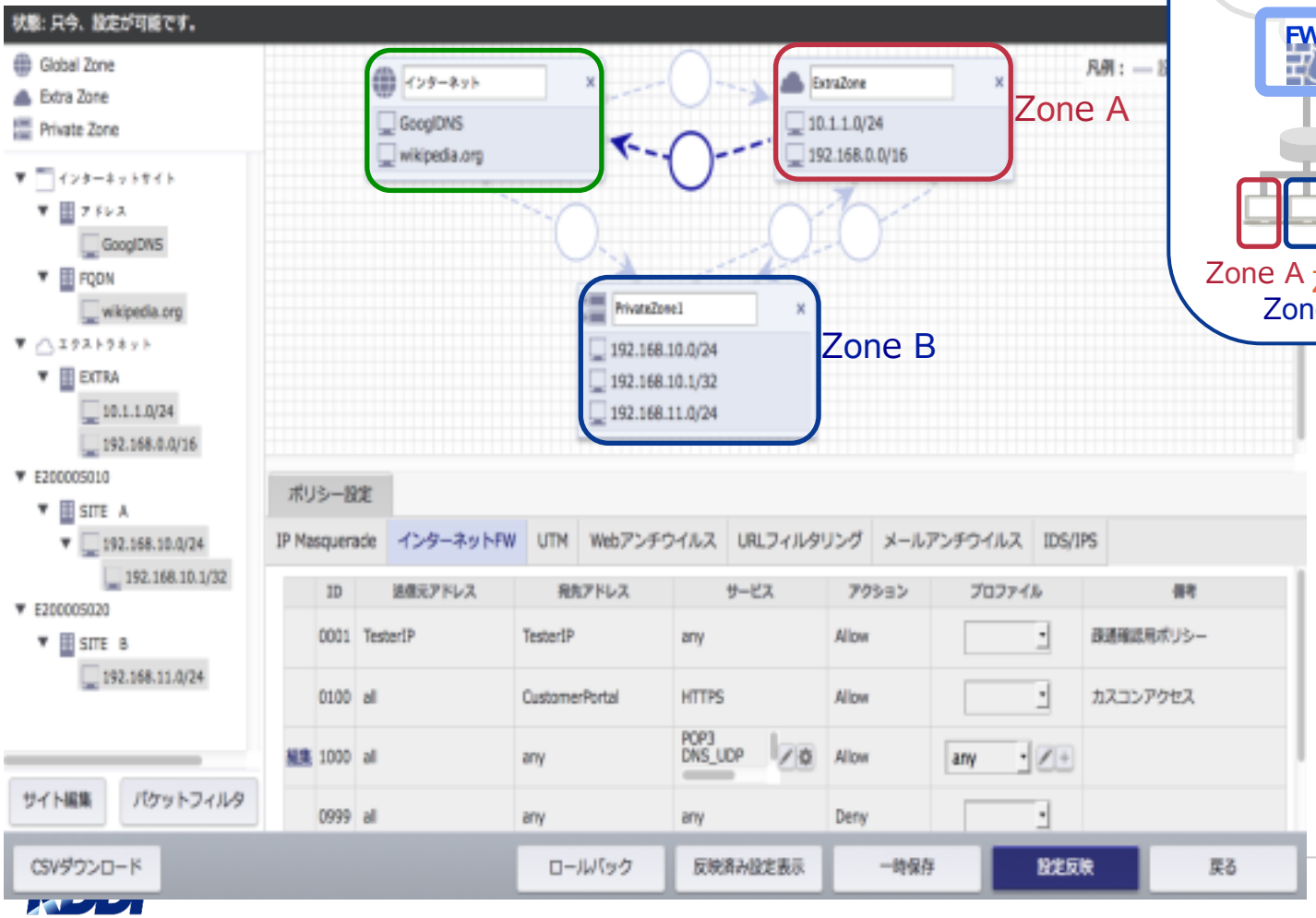


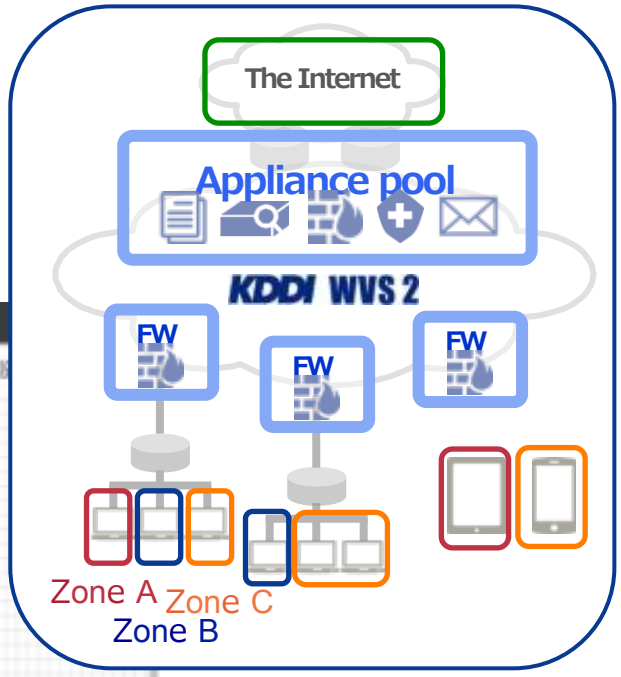
http://www.kddi.com/business/network/intranet/kddi-wvs2/

WVS2概要(2/2)

■ カスタマーコントローラ

- 概念/直感的なユーザインタフェース





Lessons Learned

■ 課題

- **サービスチェイニング**
  - D-Planeプログラマビリティ
- **自動化&抽象化**
  - CFS – RFSマッピング

＋

- **サービス拡張性**
  - 任意のサービスを、導入したい時に
- **既存WVSとの相互接続(+オーバレイ)**
  - 非グリーンフィールド構築
- **信頼性&スケーラビリティ**
  - 千オーダのユーザ数、万オーダの回線数

■ どんな選択肢があったのか? (as of 2013)

- **OpenFlow**
  - スケールしない(少なくとも当時は)
  - 車輪の再発明
    - 既存網との相互接続性に懸念
- **NFV**
  - CAPEX/OPEX NG†
  - まだ早かった。
- **SFC(NSH)**
  - まだなかった。
- **EVPN**
  - 要件に合わない。
    - L2トランスペアレントなアプライアンス

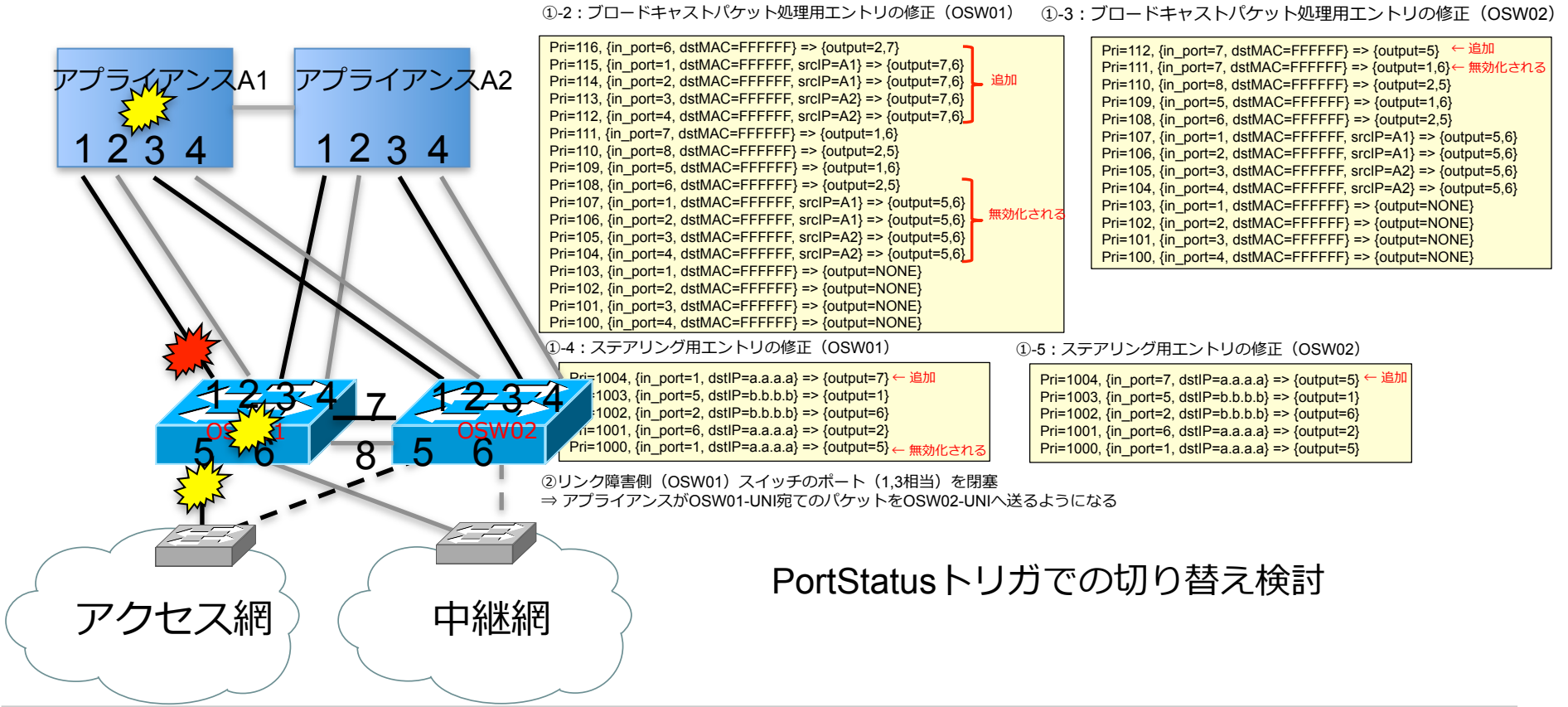
⇨ **ポリシーベースドフォワーディング　w/ コントローラ**

† http://www.slideshare.net/miyakohno/mk-epn-seminarpanelforpublic

OpenFlow

- トラヒックステアリング　→　フローエントリ数がスケールしない
- 既存網との相互接続
  - L2/L3/MPLS
- HA機能

車輪の再発明

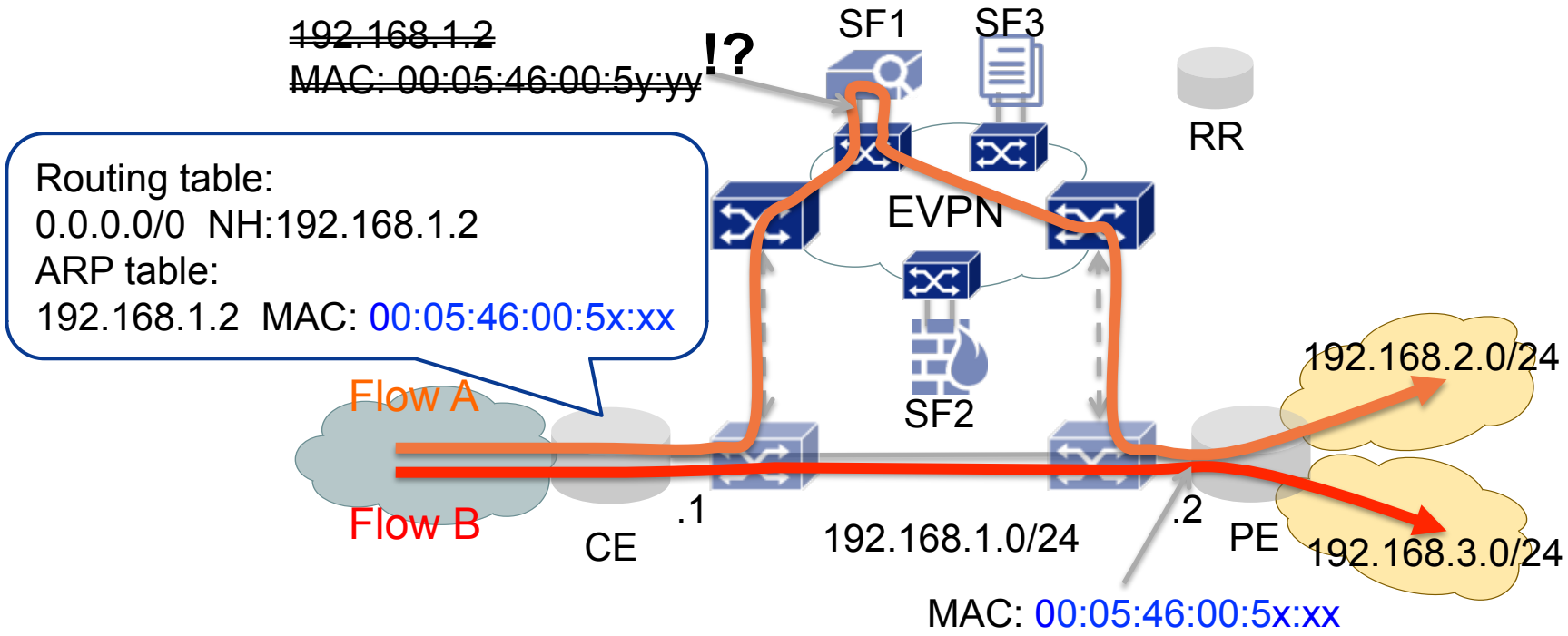


PortStatusトリガでの切り替え検討

EVPN

- **BGP MPLS-based Ethernet VPN (RFC7432)**
- **MACアドレスを持たないアプライアンスにどうやってフォワーディング?**
  - **動的なサービス挿抜では、アプライアンスにL3終端されると困る。**
    - L2トランスペアレントモード使う。
    - (SFC) Architecture Principle in RFC7665

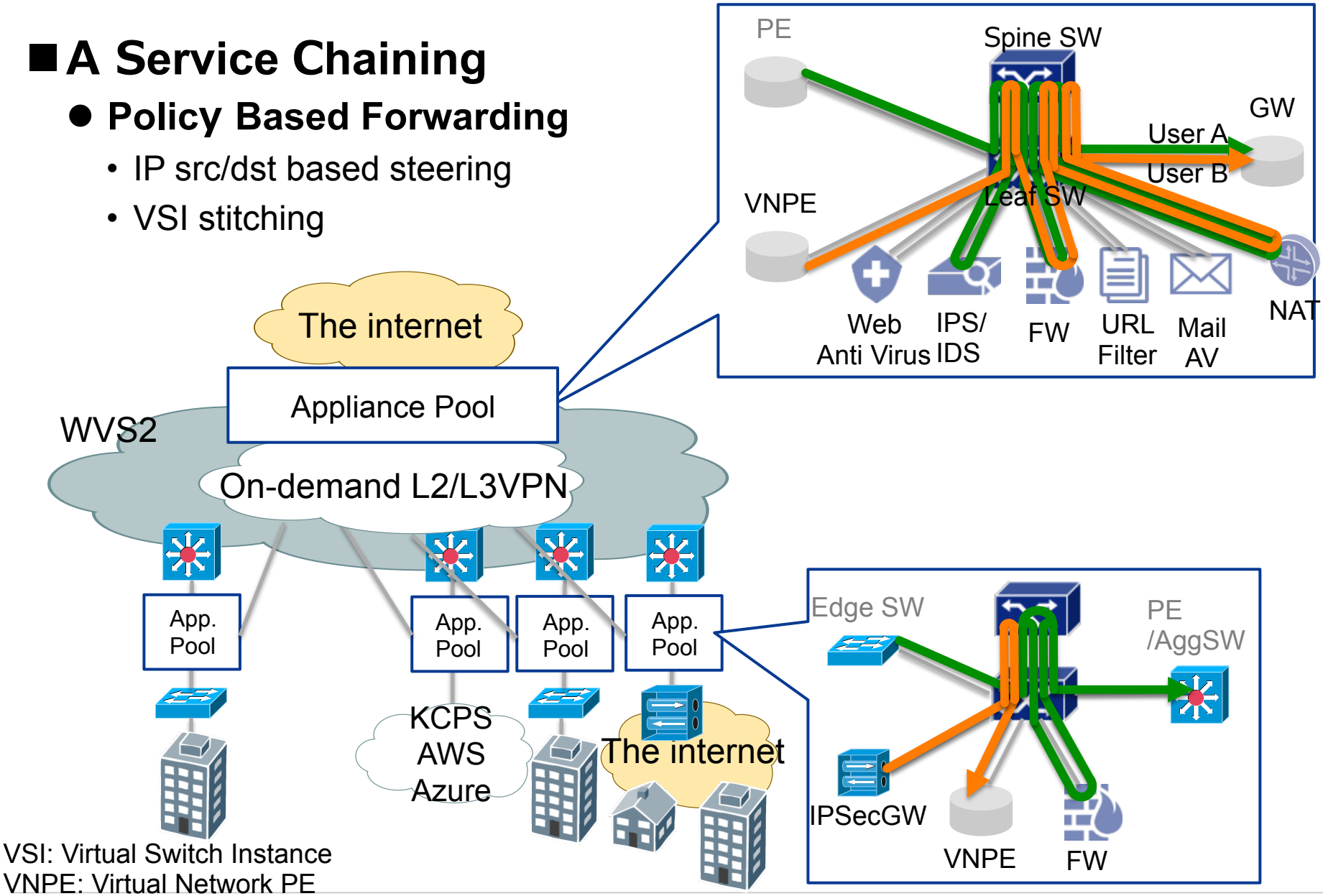
      1. Topological independence: no change to the underlay network forwarding topology - implicit, or explicit - are needed to deploy and invoke SFs or SFCs



ネットワークアーキテクチャ

■A Service Chaining

- **Policy Based Forwarding**
  - IP src/dst based steering
  - VSI stitching

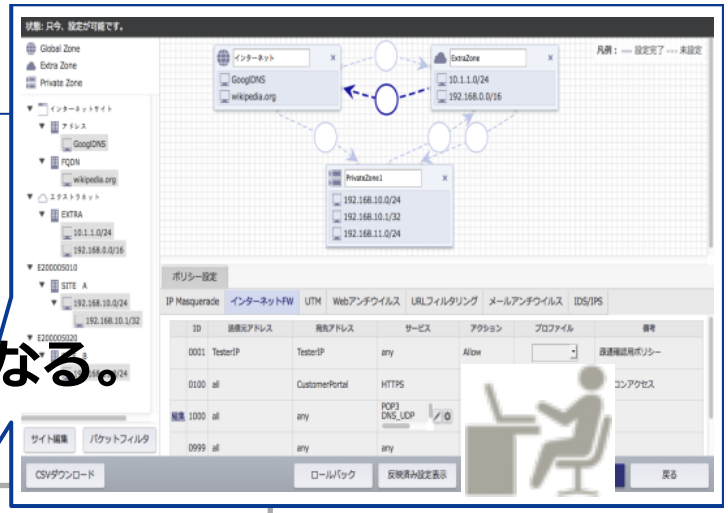


VSI: Virtual Switch Instance
VNPE: Virtual Network PE

制御システムアーキテクチャ

■Self-serviceの実現

- **抽象化と自動化**
- **お客さま向けと運用者向けでは難易度が異なる。**
  - 予想外の使い方　vs　手順書通り



Self-service Portal

お客様

抽象化されたサービス制御要求

Customer Facing Service(CFS)

組み合わせ自由、順不同

Service Modeling

セキュリティ機能、オンデマンドVPN、帯域変更、etc…

経路計算/リソース割当

順序制御

とっても大変

運用者

Network Modeling

SF1　SF2　SF3　PE/SW

機器レベルマッピング

Resource Facing Service(RFS)

Vendor-Specific Modeling

SF1　SF2　SF3　PE/SW

Network Equipment

SF1　SF2　SF3　PE/SW

今、一から作るなら

■ D-plane周りは、変わらない?

- **NSHは数年かかる**
  - 単純なL3 overlayはリソース利用効率と運用面から?
    - 高効率化するにはECMPとか
    - リンク障害時に、どのお客さまに影響を与えたか特定できない。
      - » draft-amante-oam-ng-requirements
  - 経路が分かる(指定できる)トランスポートが必要
    - Segment Routing?
      - » draft-ietf-spring-segment-routing-msdc

■ NFVも本気で考えるか

- **あれから3年。。。**
  - https://portal.etsi.org/NFV/NFV_White_Paper.pdf
- **転送性能がネックにならないところ**
- **ライセンスモデルなんとかなりませんか?**

Day one of NFV, 10/23/2012@Darmstadt

■ 制御システムアーキテクチャをもっと洗練させる。

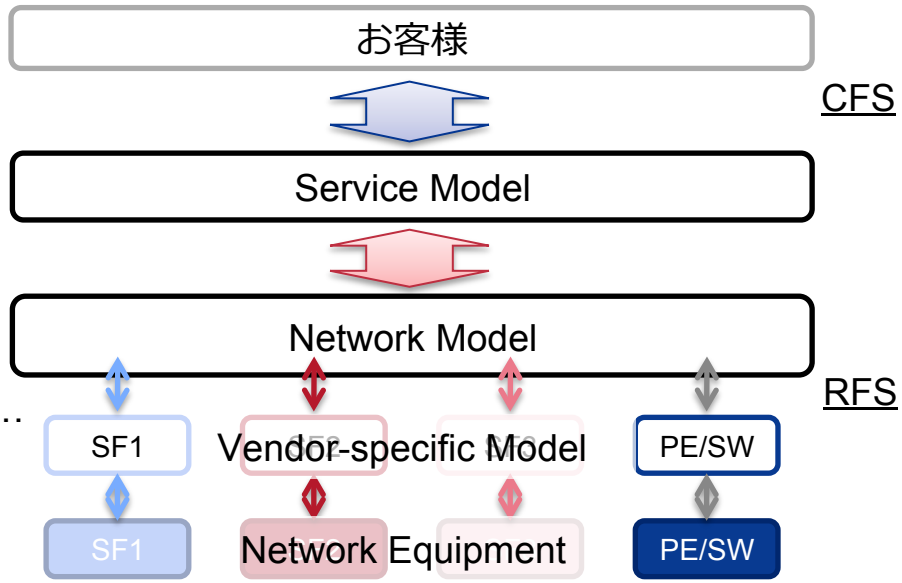
- **C-planeプログラマビリティを如何に簡単に実現するか?**
  - 制御システムデザイン論

制御システムデザイン論

■ 入力順によらず、出力はいつも同じであるべき
- お客様　vs　運用者
- 予想外の使い方　vs　手順書通り
- モデル駆動型、宣言型、関数型　vs　ワークフローベース、命令型、手続き型

■ RFSは対応製品を利用可
- 単一ネットワーク機器へのコンフィグ手順は一意
- マルチベンダ対応
- NE側がNetconfでよしなに計らうべき
  - 標準ネットワークモデルも
    - NETMOD, Routing Area他

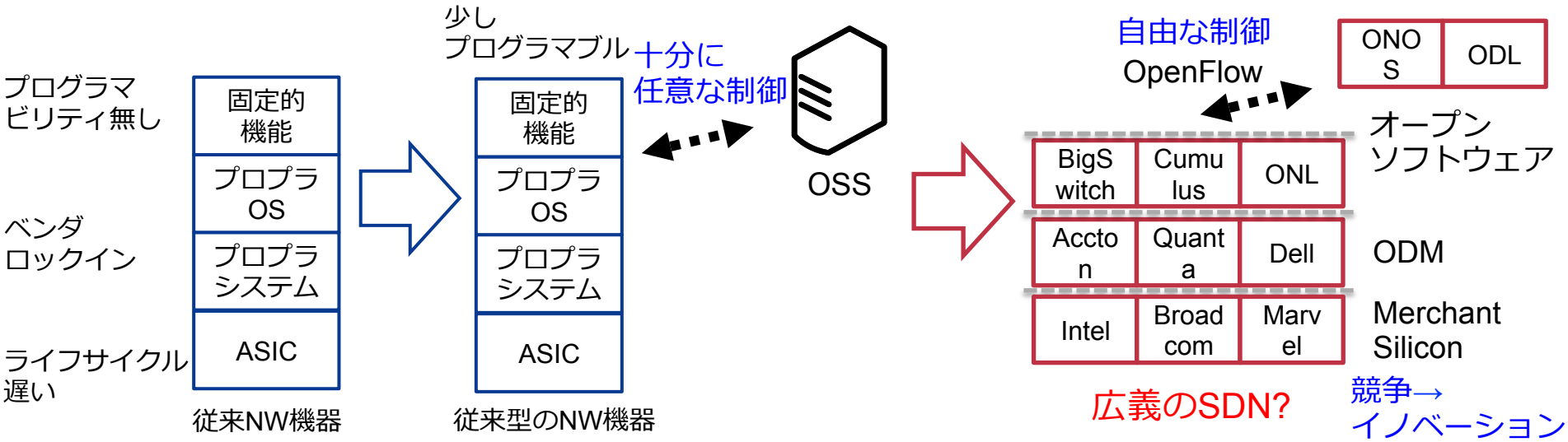
■ CFS-RFSマッピングは個別の課題
- サービス依存
  - サービスモデルも標準化
    - L3SM, I2NSF, OpenStack GBP, MEF Legato…
- ネットワークアーキテクチャ依存
  - 複数機器間のコンフィグ手順へ
- 既存システム依存



SDN revisit

■ **持続成長性**
- **追加サービスを任意のタイミングで投入したい。**
- **サービスライフサイクルのスピードアップ**

■ **コントロールプレーン/データプレーン分離**
- **機能のベンダロックインから解放**



*"… very important to reduce ideas to practice. … the solutions I invent need to be "sufficient" to solve the problem; they should be as simple as possible, but the system has to really run, and it has to run with **good enough** performance.", Barbara Liskov*

出典: https://www.computer.org/csdl/mags/ds/2005/02/o2002.pdf

Telecom DevOps?

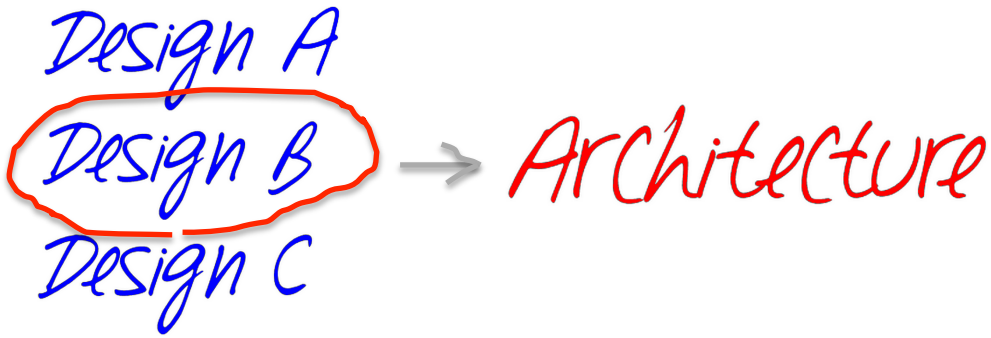
■ **振り返ると**

- **今は、SLAがDevOps的なものを許さない(と思う)。**
- **流行りの開発手法は向かない?**
  - 手戻りリスクの程度問題

⇨ 最初の設計/選択が肝心

■ **Telecom DevOps† = Development for Operations**

- **運用しやすい*ネットワークシステム*を開発する。**
  - ランニングコストに優しい
- **開発手法よりも、アーキテクチャが肝心**
  - ネットワークアーキテクチャと同じくらい、制御システムアーキテクチャに関心を
  - モジュール(コンポーネントモデル)化の追求
- **「向いてない領域の見極め」by SoftBank 西さん†**



まとめ

- WVS2はサービスチェイニングと自動化/抽象化が欲しかった。

- SDNの技術はまだまだ(だった。)
- システムデザインは永遠の?課題

- C/D-plane分離は、サービスライフサイクルのコントロールも可能に
- 最初の設計が肝心だが、good enoughで良い。
- 制御システムも頑張らないと

是非、*WVS2からSDNの世界へ*

Designing The Future

KDDI

ご静聴ありがとうございました☺