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

English version follows Japanese version.

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)

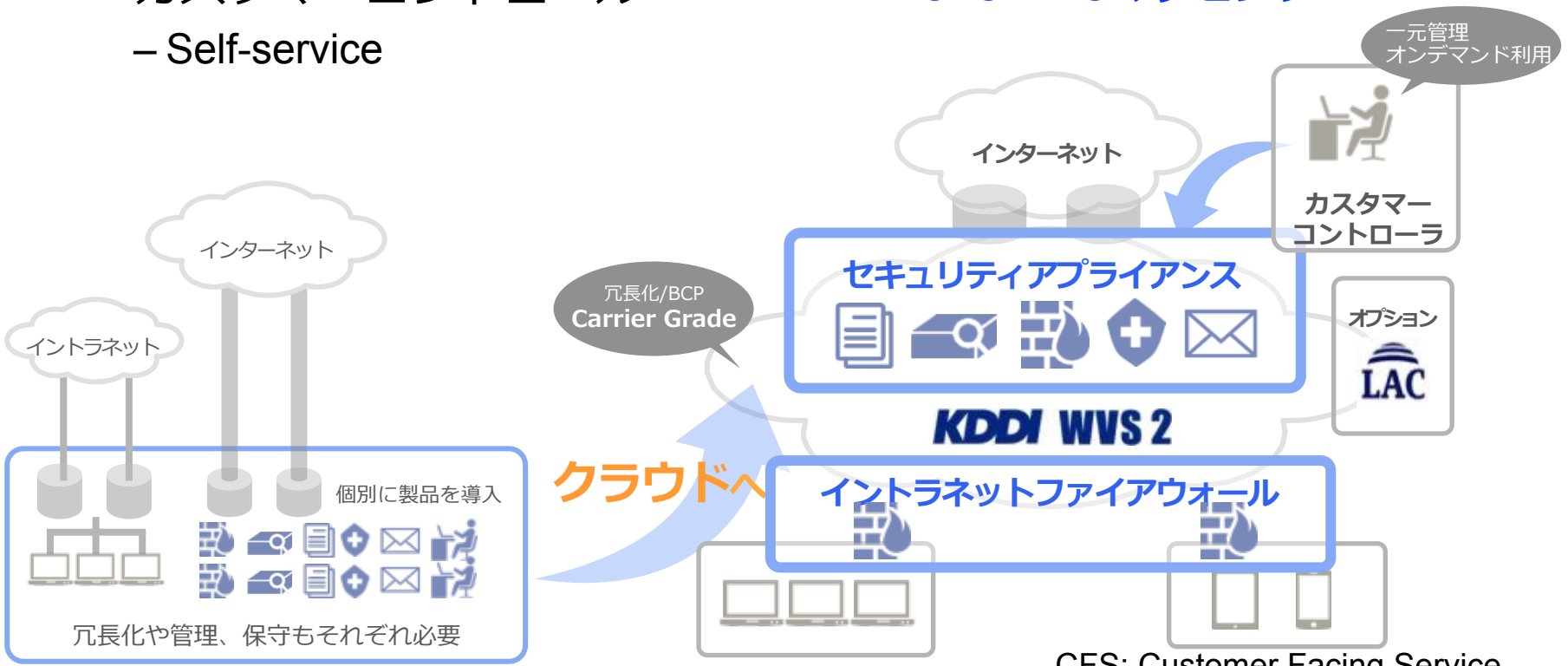
● セキュリティアプライアンスクラウド

- 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***

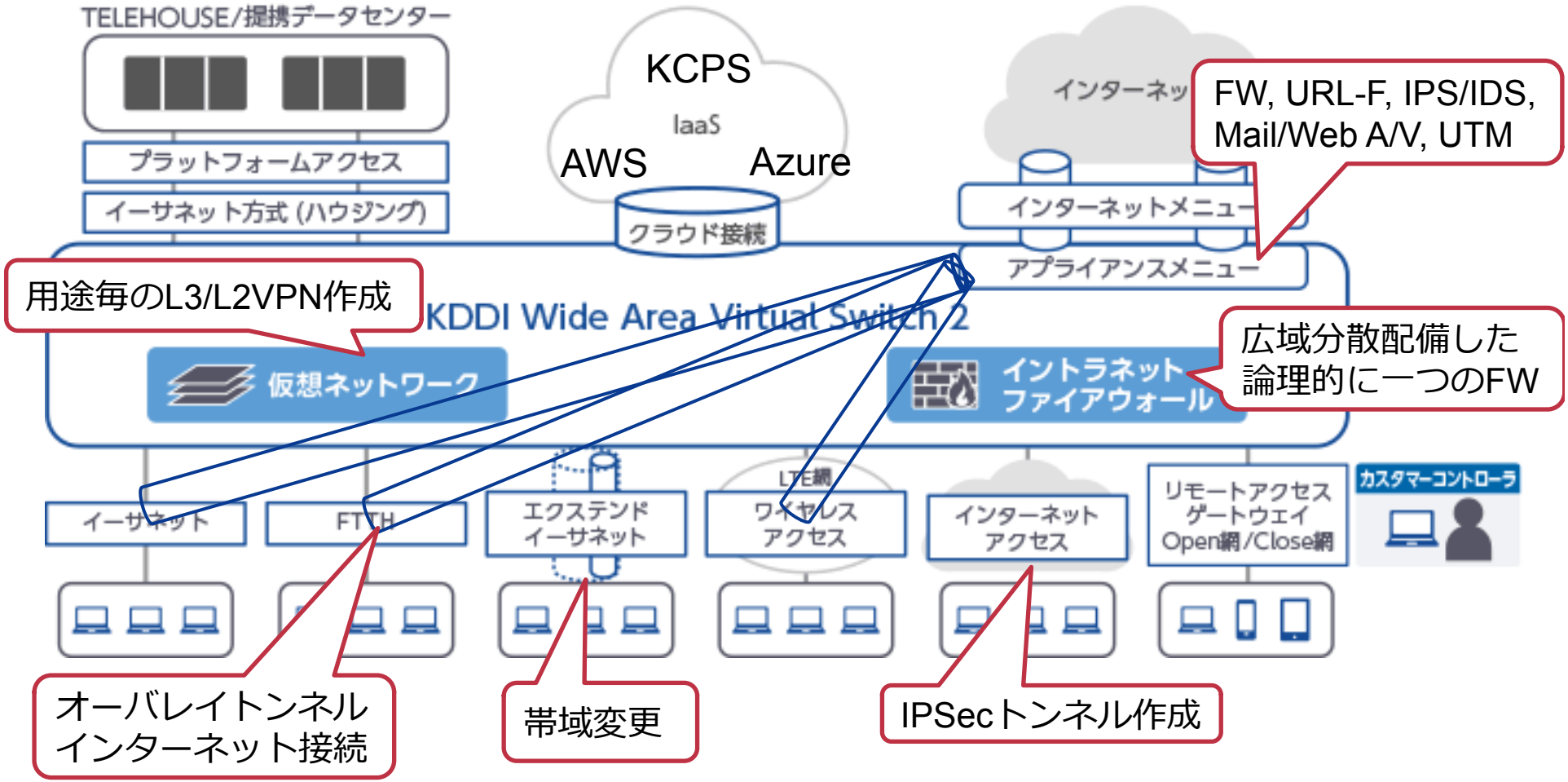


CFS: Customer Facing Service
RFS: Resource Facing Service

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**

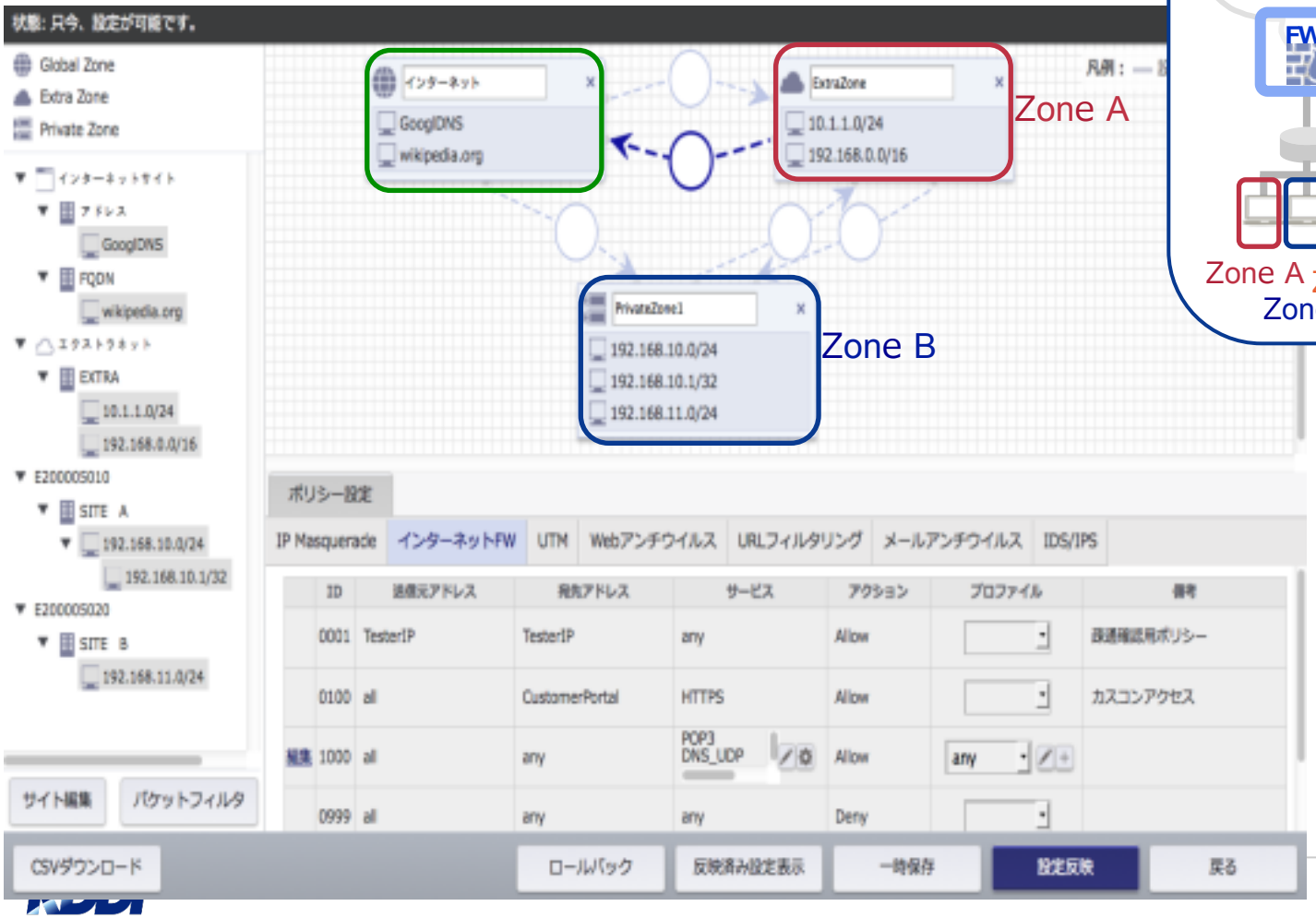


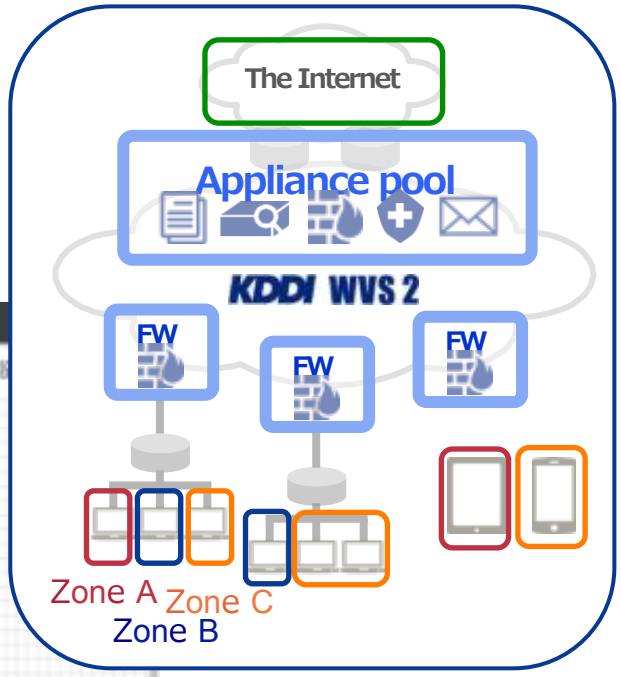
http://www.kddi.com/business/network/intranet/kddi-wvs2/

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

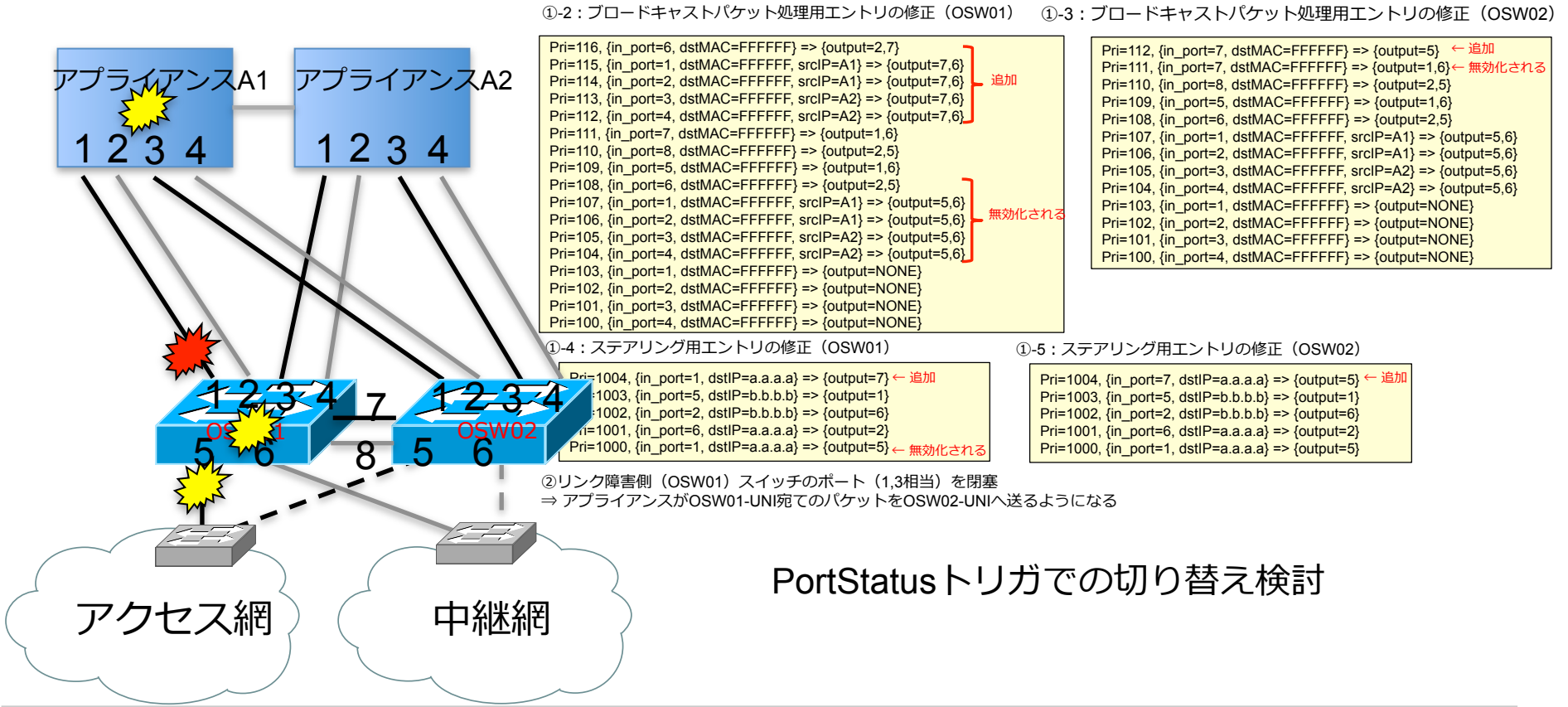
⇨ **ポリシーベースドフォワーディング　w/ コントローラ**

† 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

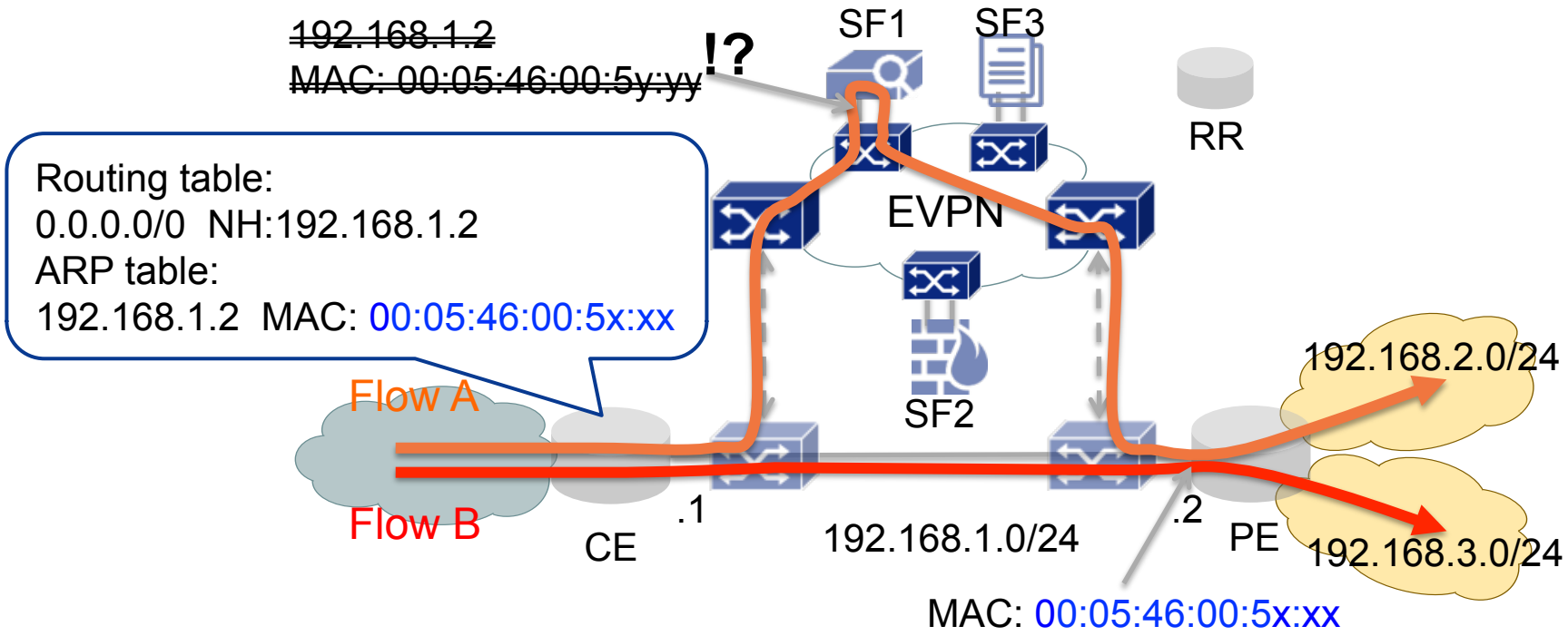


PortStatusトリガでの切り替え検討

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

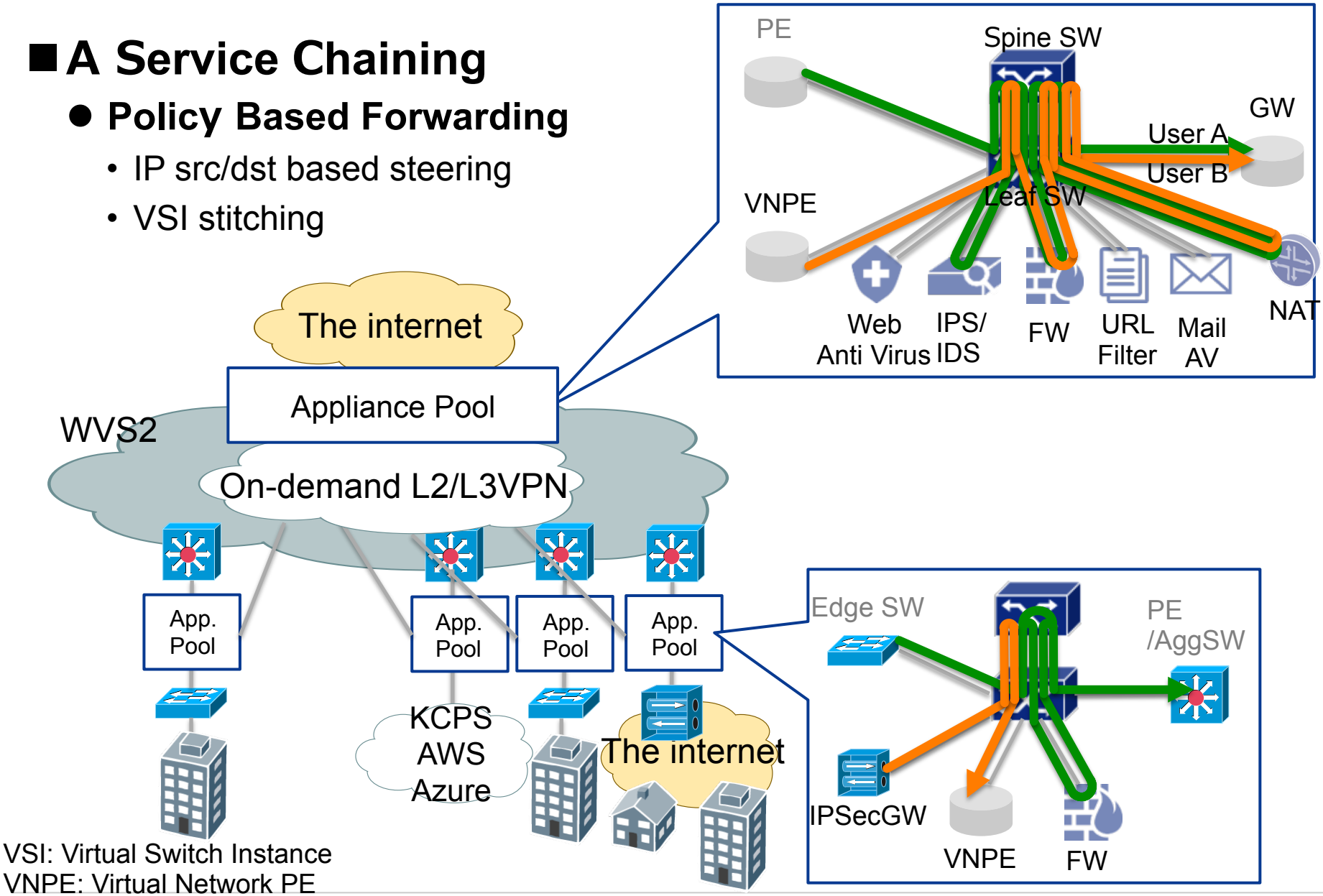
      1. Topological independence: no change to the underlay network forwarding topology - implicit, or explicit - are needed to deploy and invoke SFs or SFCs



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

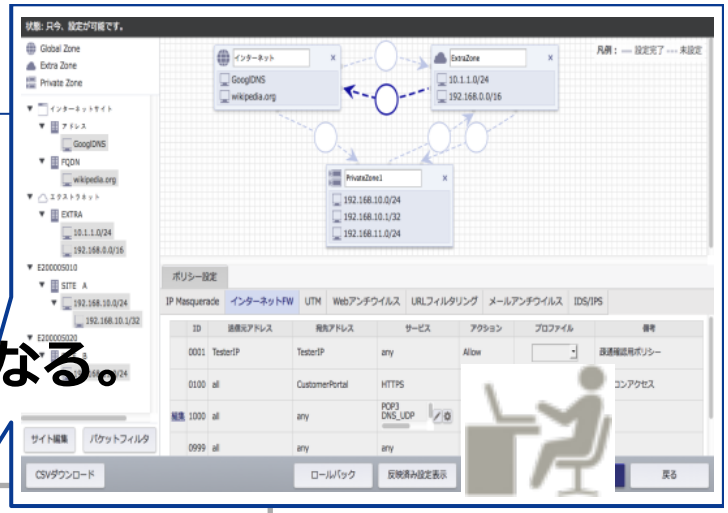


VSI: Virtual Switch Instance
VNPE: Virtual Network PE

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。

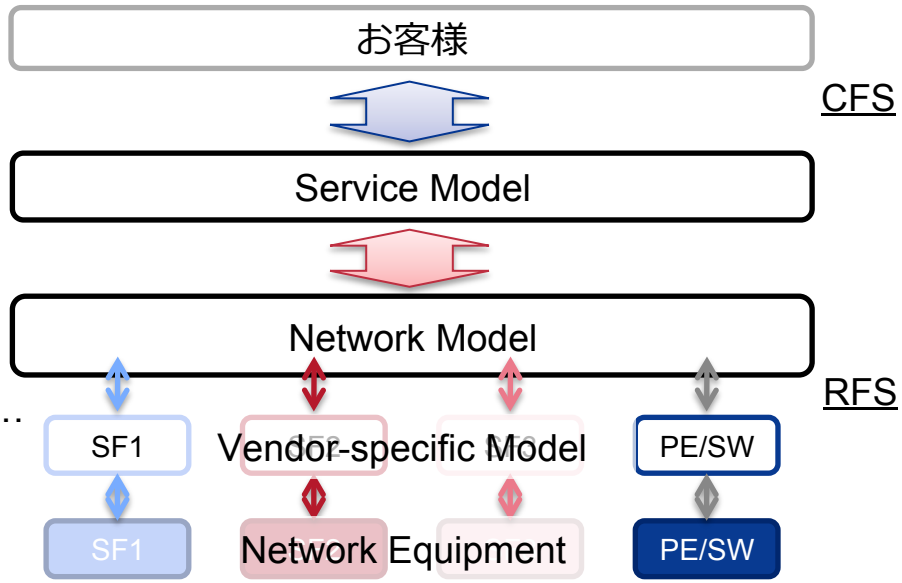
- **C-planeプログラマビリティを如何に簡単に実現するか?**
  - 制御システムデザイン論

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

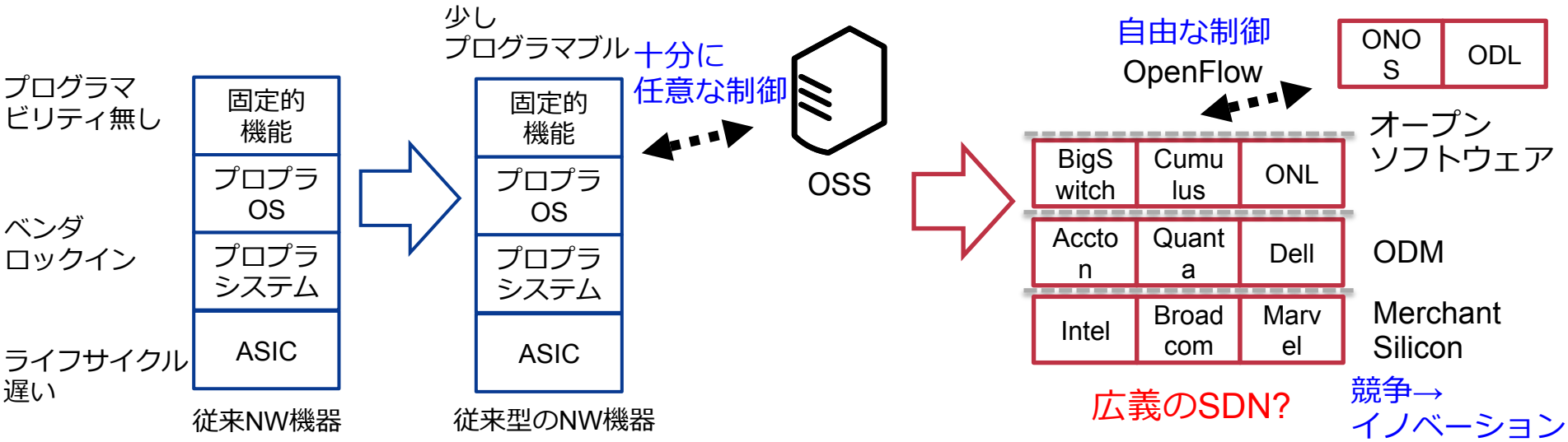
■ CFS-RFSマッピングは個別の課題
- サービス依存
  - サービスモデルも標準化
    - L3SM, I2NSF, OpenStack GBP, MEF Legato…
- ネットワークアーキテクチャ依存
  - 複数機器間のコンフィグ手順へ
- 既存システム依存



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?

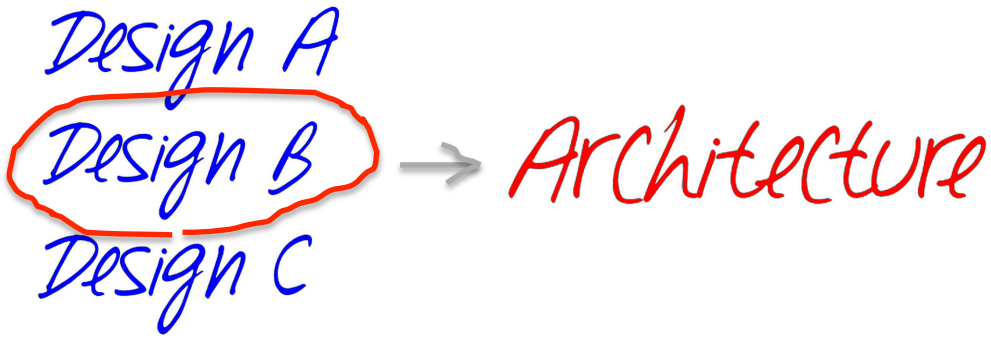
■ **振り返ると**

- **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☺

A Carrier SDN Lessons Learned

1/14/2016

Kenichi Ogaki
IP Network Dept., KDDI Corp.

Agenda

1. What's SDN
2. Motivation
3. WVS2 Basics
4. Lessons Learned
   - Network Architecture
   - Control system Architecture
5. SDN revisit
6. Telecom DevOps?

What's SDN

- **Control plane(Routing/Signaling) / Data plane(Forwarding) Separation**
  - **Provide Data plane programmability**
    - Achieve arbitrary forwarding

- **Interview to Dr. Nick McKeown**
  - **Kate Greene, "TR10: Software Defined Networking," MIT Technology Review, Mar. 2009,** http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/
  - ***"computer scientists have dreamed up ways to improve networks' speed reliability, energy efficiency, and security. … the routers and switches at the core of the Internet are locked down,…"***
  - ***"OpenFlow … define data flows using software—a sort of "software-defined networking.""***

Motivation

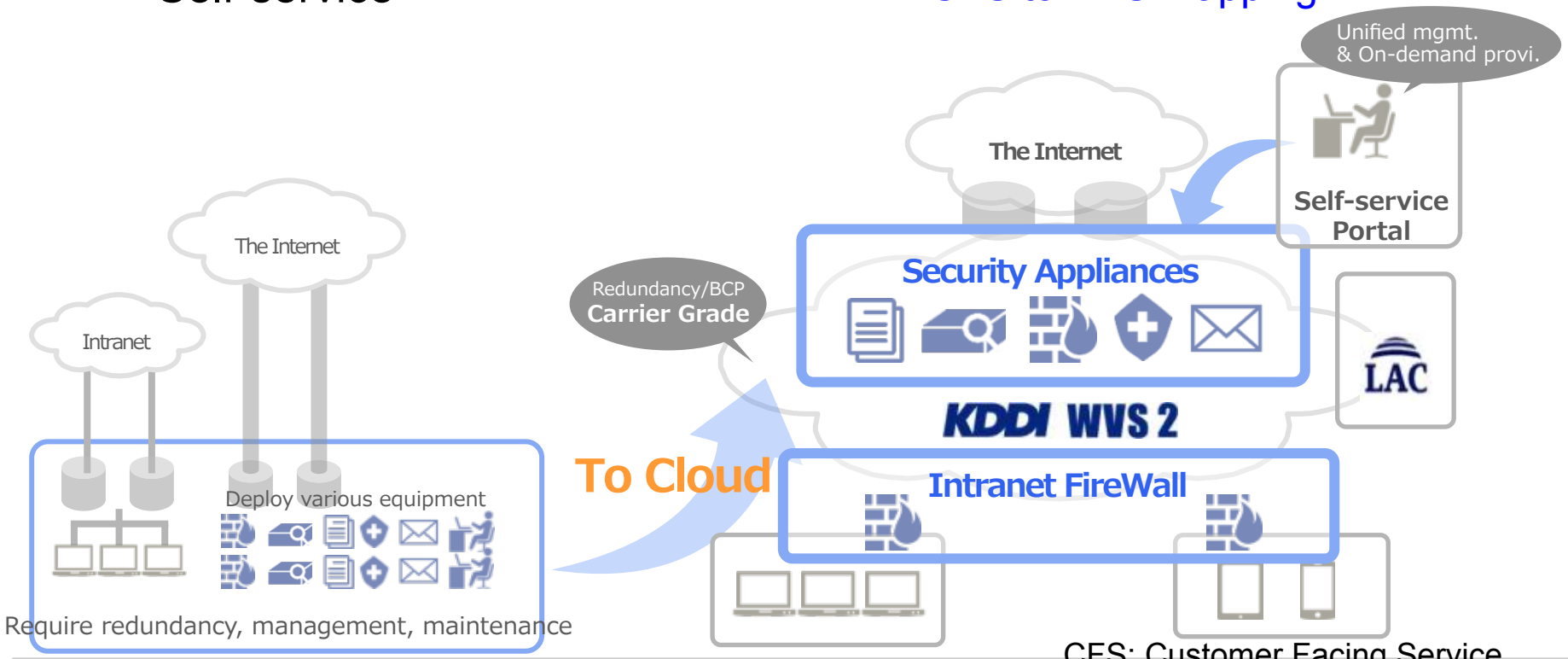
■ Wide Area Virtual Switch 2 (WVS2)

● Security Appliance Cloud

- Ownership to Usership
  - – Cloud model
  - – Pay-as-you-go
- Self-service

- Service Chaining
  - – D-plane programmability
- Automation & Abstraction
  - – CFS to RFS mapping

SDN

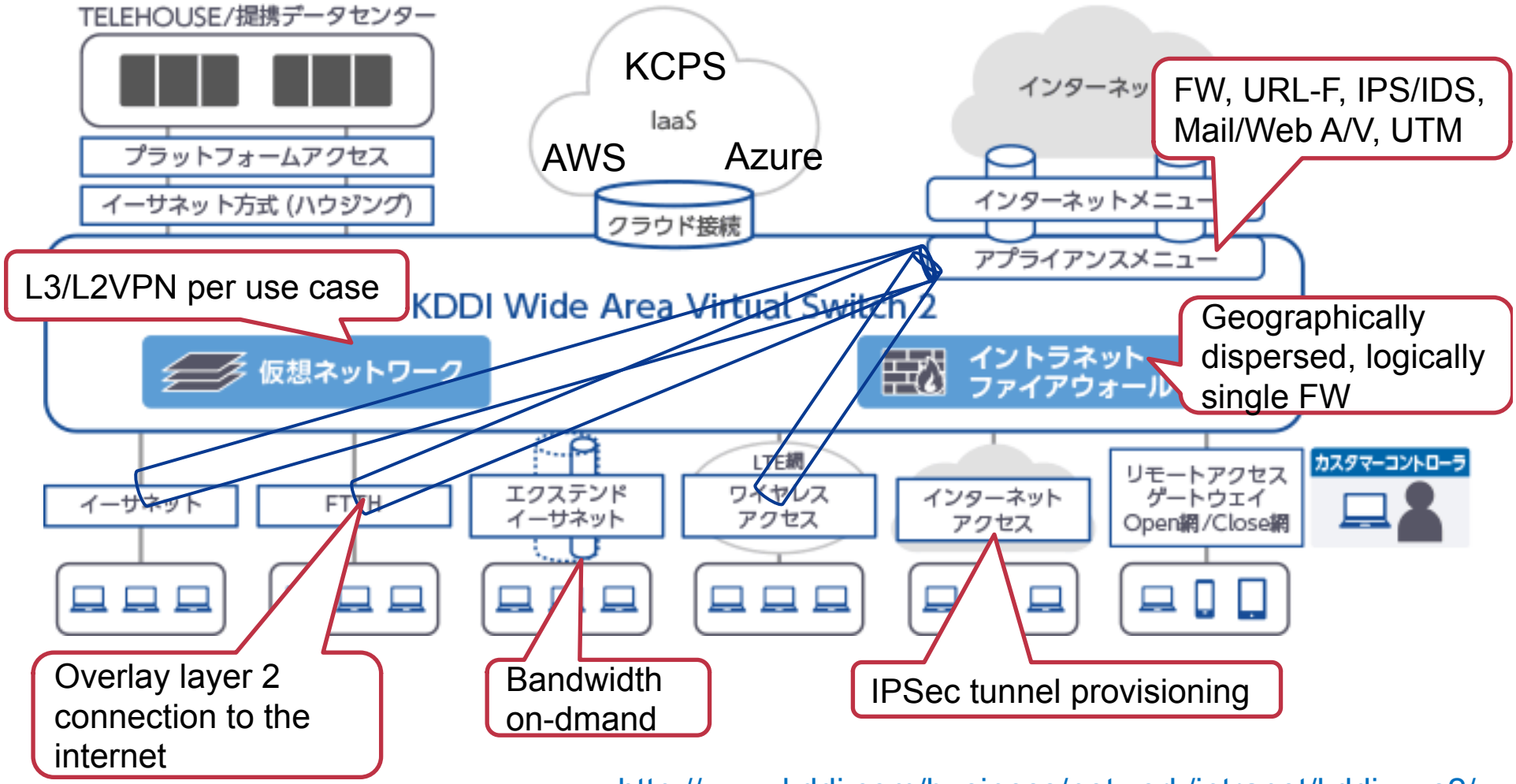


CFS: Customer Facing Service
RFS: Resource Facing Service

WVS2 Basics(1/2)

- **Security Appliance Cloud**
- **Virtual Network**

On-demand provisioning

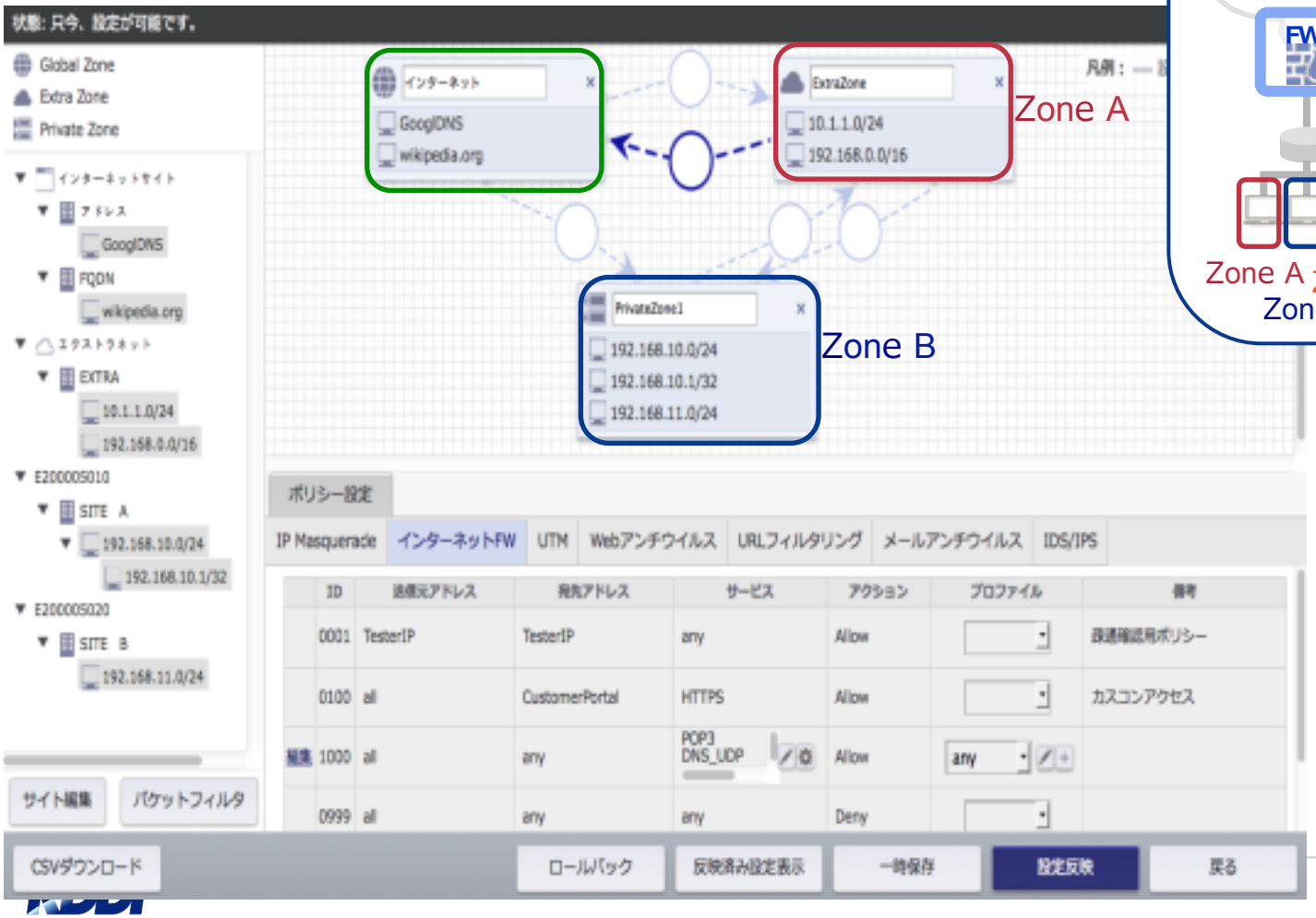


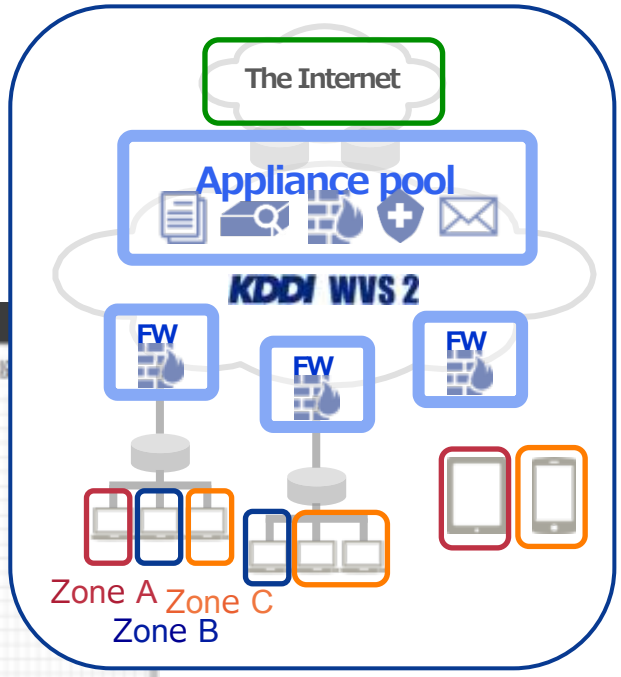
http://www.kddi.com/business/network/intranet/kddi-wvs2/

WVS2 Basics(2/2)

■ Self-service portal

- Intuitive/Intent-based User Interface





Lessons Learned

■ Challenge

- **Service Chaining**
  D-Plane programmability
- **Automation & Abstraction**
  - CFS to RFS mapping

+

- **Service sustainability**
  - Faster time-to-market
- **Interop(+overlay) w/ existing NWs**
  - Non green-field deployment
- **Reliability & Scalability**
  - Thousands of customers, tens of thousands of leased lines.

■ What options there were? (as of 2013)

- **OpenFlow**
  - No scalability (at that time)
  - Reinventing the wheel
    – Concern about the interoperability with existing NWs
- **NFV**
  - CAPEX/OPEX NG†
  - Too early
- **SFC(NSH)**
  - Not existed
- **EVPN**
  - Not meet reqs.
    – L2 transparent appliance

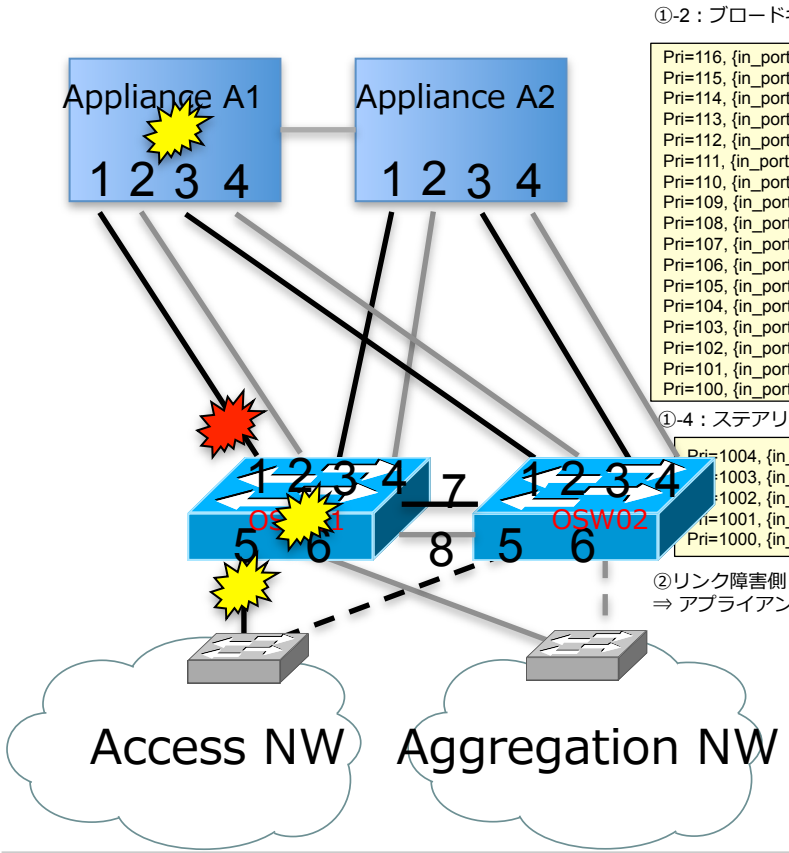
⇨ **Policy-based Forwarding w/ Controller**

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

OpenFlow

- Traffic steering → No scalability for # of flow entries
- Interoperability with existing NWs
  - L2/L3/MPLS
- High Availability functionality

} Reinventing the wheel



①-2：ブロードキャストパケット処理用エントリの修正（OSW01）

Pri=116, {in_port=6, dstMAC=FFFFFF} => {output=2,7}
Pri=115, {in_port=1, dstMAC=FFFFFF, srcIP=A1} => {output=7,6}
Pri=114, {in_port=2, dstMAC=FFFFFF, srcIP=A1} => {output=7,6}
Pri=113, {in_port=3, dstMAC=FFFFFF, srcIP=A2} => {output=7,6}
Pri=112, {in_port=4, dstMAC=FFFFFF, srcIP=A2} => {output=7,6}
（追加）
Pri=111, {in_port=7, dstMAC=FFFFFF} => {output=1,6}
Pri=110, {in_port=8, dstMAC=FFFFFF} => {output=2,5}
Pri=109, {in_port=5, dstMAC=FFFFFF} => {output=1,6}
Pri=108, {in_port=6, dstMAC=FFFFFF} => {output=2,5}
Pri=107, {in_port=1, dstMAC=FFFFFF, srcIP=A1} => {output=5,6}
Pri=106, {in_port=2, dstMAC=FFFFFF, srcIP=A1} => {output=5,6}
Pri=105, {in_port=3, dstMAC=FFFFFF, srcIP=A2} => {output=5,6}
Pri=104, {in_port=4, dstMAC=FFFFFF, srcIP=A2} => {output=5,6}
（無効化される）
Pri=103, {in_port=1, dstMAC=FFFFFF} => {output=NONE}
Pri=102, {in_port=2, dstMAC=FFFFFF} => {output=NONE}
Pri=101, {in_port=3, dstMAC=FFFFFF} => {output=NONE}
Pri=100, {in_port=4, dstMAC=FFFFFF} => {output=NONE}

①-3：ブロードキャストパケット処理用エントリの修正（OSW02）

Pri=112, {in_port=7, dstMAC=FFFFFF} => {output=5} ← 追加
Pri=111, {in_port=7, dstMAC=FFFFFF} => {output=1,6} ← 無効化される
Pri=110, {in_port=8, dstMAC=FFFFFF} => {output=2,5}
Pri=109, {in_port=5, dstMAC=FFFFFF} => {output=1,6}
Pri=108, {in_port=6, dstMAC=FFFFFF} => {output=2,5}
Pri=107, {in_port=1, dstMAC=FFFFFF, srcIP=A1} => {output=5,6}
Pri=106, {in_port=2, dstMAC=FFFFFF, srcIP=A1} => {output=5,6}
Pri=105, {in_port=3, dstMAC=FFFFFF, srcIP=A2} => {output=5,6}
Pri=104, {in_port=4, dstMAC=FFFFFF, srcIP=A2} => {output=5,6}
Pri=103, {in_port=1, dstMAC=FFFFFF} => {output=NONE}
Pri=102, {in_port=2, dstMAC=FFFFFF} => {output=NONE}
Pri=101, {in_port=3, dstMAC=FFFFFF} => {output=NONE}
Pri=100, {in_port=4, dstMAC=FFFFFF} => {output=NONE}

①-4：ステアリング用エントリの修正（OSW01）

Pri=1004, {in_port=1, dstIP=a.a.a.a} => {output=7} ← 追加
Pri=1003, {in_port=5, dstIP=b.b.b.b} => {output=1}
Pri=1002, {in_port=2, dstIP=b.b.b.b} => {output=6}
Pri=1001, {in_port=6, dstIP=a.a.a.a} => {output=2}
Pri=1000, {in_port=1, dstIP=a.a.a.a} => {output=5} ← 無効化される

①-5：ステアリング用エントリの修正（OSW02）

Pri=1004, {in_port=7, dstIP=a.a.a.a} => {output=5} ← 追加
Pri=1003, {in_port=5, dstIP=b.b.b.b} => {output=1}
Pri=1002, {in_port=2, dstIP=b.b.b.b} => {output=6}
Pri=1001, {in_port=6, dstIP=a.a.a.a} => {output=2}
Pri=1000, {in_port=1, dstIP=a.a.a.a} => {output=5}

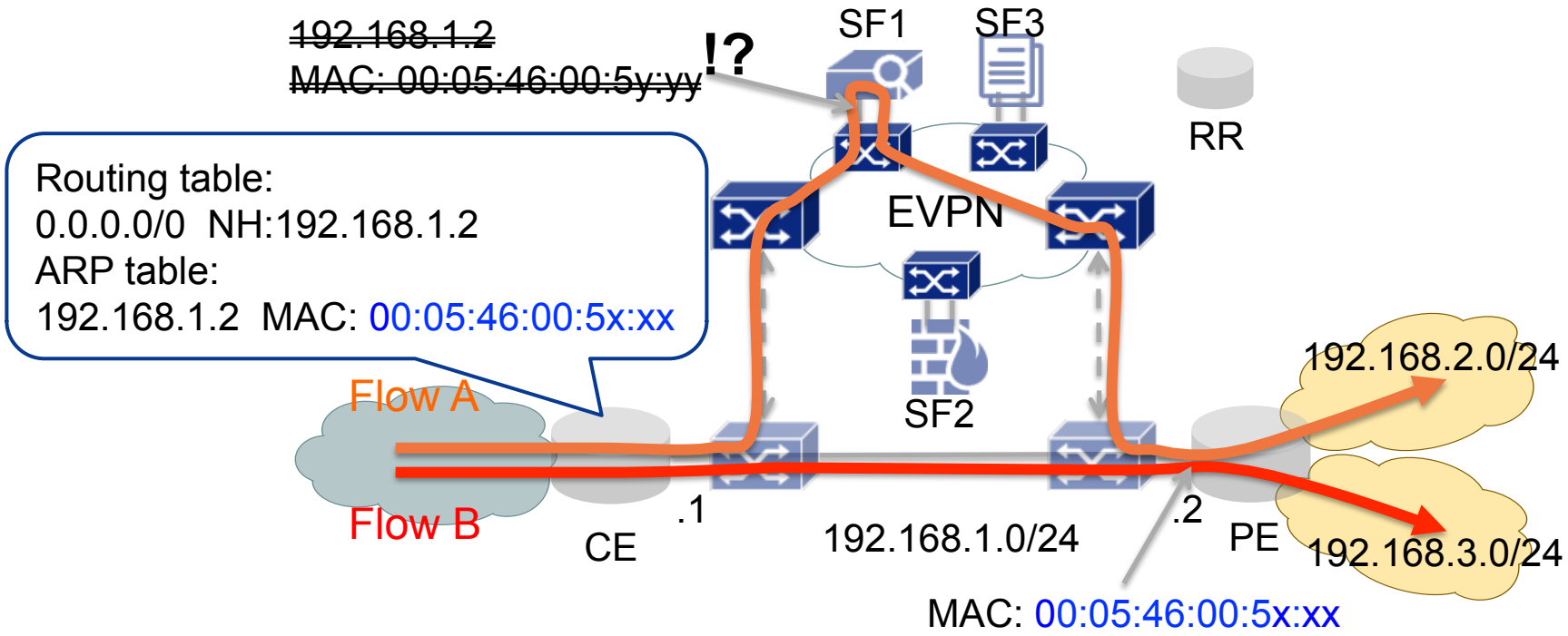
②リンク障害側（OSW01）スイッチのポート（1,3相当）を閉塞
⇒ アプライアンスがOSW01-UNI宛てのパケットをOSW02-UNIへ送るようになる

A redundancy mechanism triggered by PortStatus

EVPN

- **BGP MPLS-based Ethernet VPN (RFC7432)**
- **How to forward a flow to an appliance with no MAC address?**
  - **For a dynamic service insertion, an appliance must not terminate layer 3.**
    - Use L2 transparent mode
    - (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

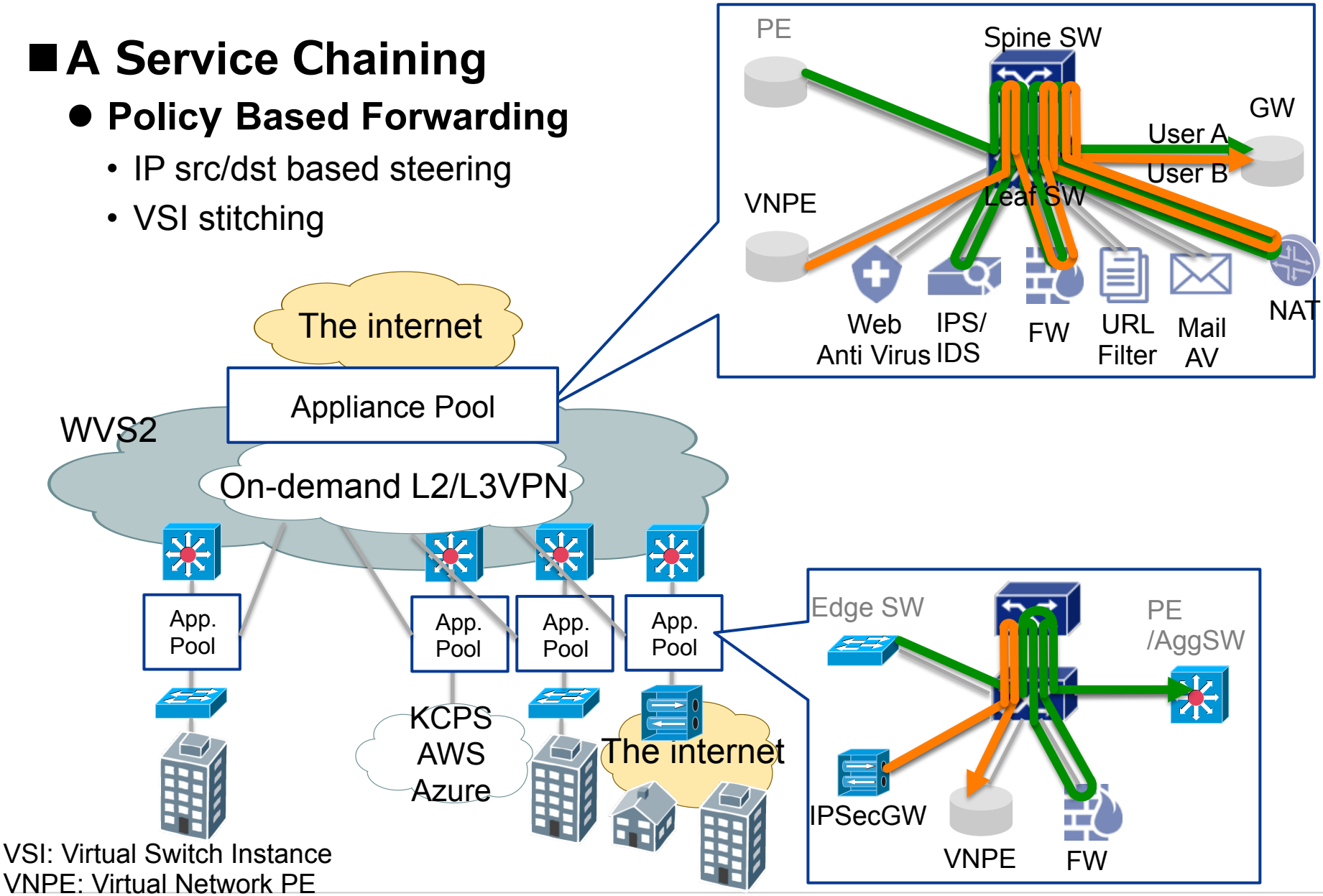


Network Architecture

■ A Service Chaining

● Policy Based Forwarding

- IP src/dst based steering
- VSI stitching

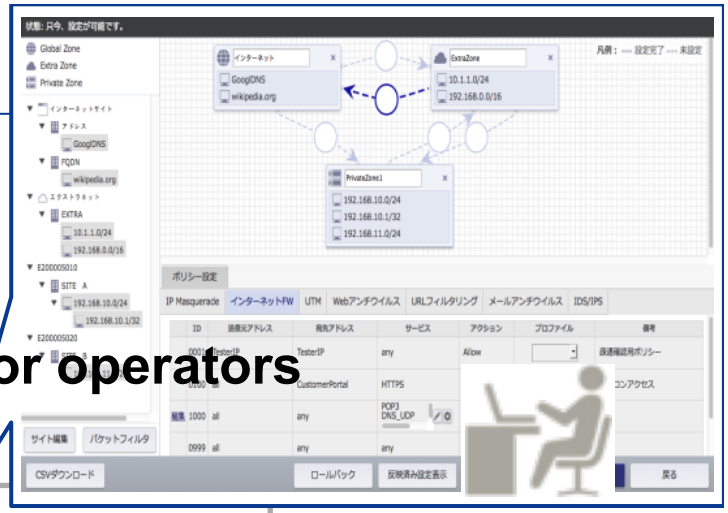


VSI: Virtual Switch Instance
VNPE: Virtual Network PE

Control system Architecture

■ How to achieve Self-service?

- **Abstraction and Automation**
- **Different difficulties for customers and for operators**
  - Unimagined operation vs Instruction-strict



Self-service Portal

Customer

Require abstracted service
Combination free, order free

Customer Facing Service(CFS)

Service Modeling

Security func.、on-demand VPN、BW on-demand、etc…

Route calc./Resource Alloc.
Order control

Too complex…

Operator

Network Modeling

SF1 SF2 SF3 PE/SW

NE level mapping

Resource Facing Service(RFS)

Vendor-Specific Modeling

SF1 SF2 SF3 PE/SW

Network Equipment

SF1 SF2 SF3 PE/SW

If redevelop, now

■ No change on D-plane architecture?

- **NSH needs more time**
  - L3 overlay is questionable from the resource utilization and the operationability trade-off.
    - – ECMP may be used for the higher utilization.
    - – Cannot locate which customer is affected when a link failure
      - » draft-amante-oam-ng-requirements
  - Need a transport mechanism to locate/specify the route
    - – Segment Routing?
      - » draft-ietf-spring-segment-routing-msdc

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

■ It's time to NFV?

- **3 years passed…**
  - https://portal.etsi.org/NFV/NFV_White_Paper.pdf
- **Apply where no forwarding performance needed**
- **License model to be improved**

■ Improve Control system architecture

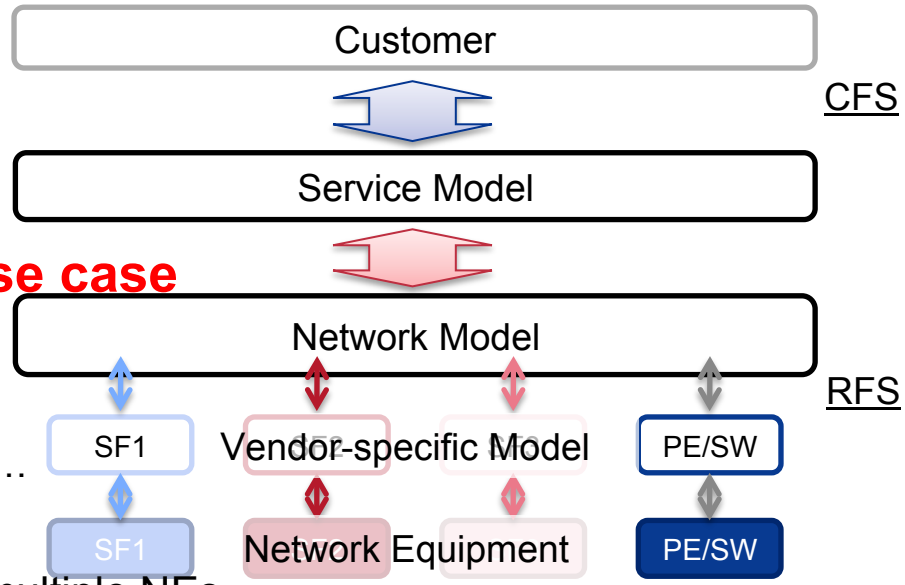
- **How to ease C-plane programmability**
  - Control system design

How to design Control system?

- **Output should be always same, not depending on input order.**
  - **Customer vs Operator**
  - **Un-imagined operation vs Instruction-strict**
  - **Model-driven, Declarative, Functional vs Workflow-based, Imperative, Procedural**

- **Commercial products available for RFS**
  - **Uniqueness of the configuration order is guaranteed for a single NE.**
  - **Support Multi-vendors**
  - **But, NE should solve by Netconf**
    - Network model standardization
      - NETMOD, Routing Area…

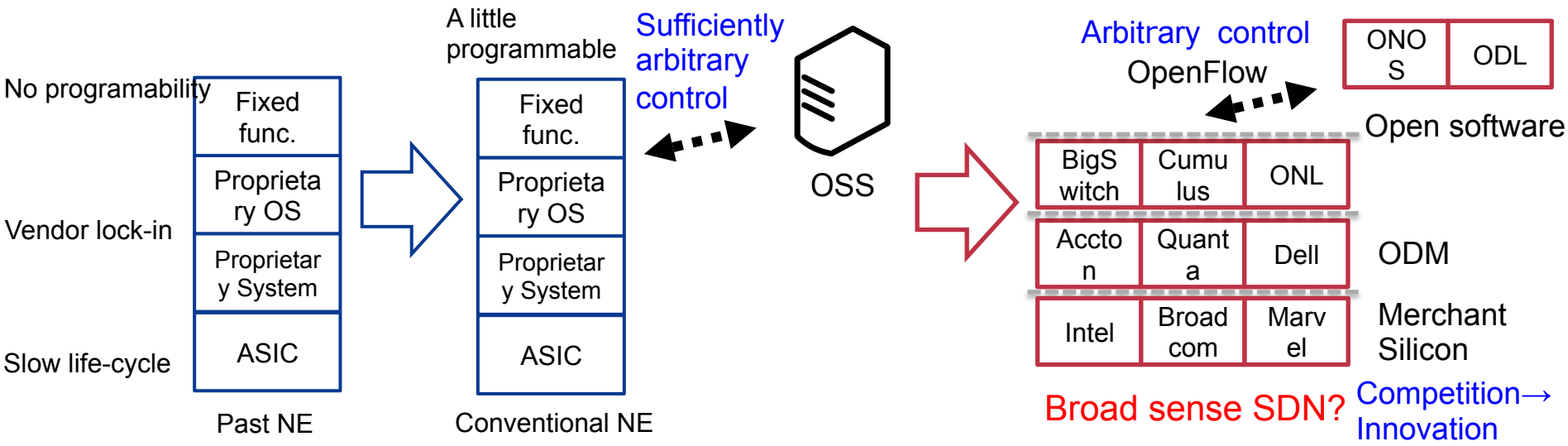
- **CFS to RFS mapping up to each use case**
  - **Depend on Service**
    - Service model standardization?
      - L3SM, I2NSF, OpenStack GBP, MEF Legato…
  - **Depend on NW architecture**
    - Optimize the configuration order among multiple NEs
  - **Depend on other systems, OSS/BSS**



SDN revisit

- **Service Sustainability**
  - **Expand service whenever operators want**
  - **Speed up service life-cycle**

- **C/D-plane separation**
  - **Free from vendor lock-in**

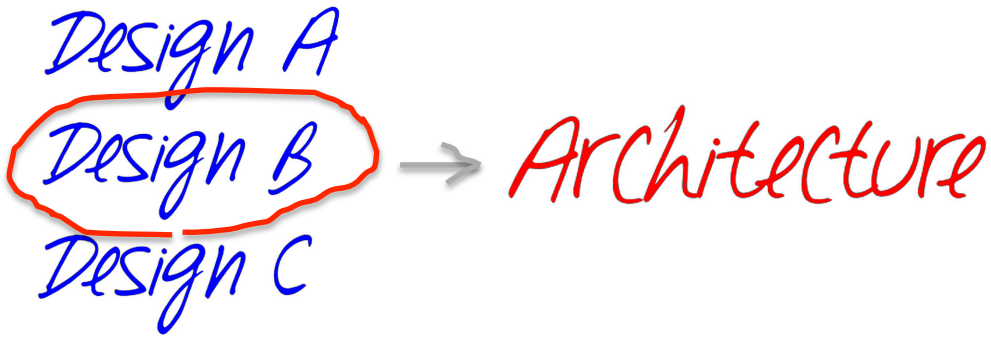


*"… very important to reduce ideas to practice. … the solutions I invent need to be "sufficient" to solve the problem; they should be as simple as possible, but the system has to really run, and it has to run with **good enough** performance.", Barbara Liskov*

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

Telecom DevOps?

- **Looking back…**
  - **A kind of DevOps doesn't meet SLA, *yet*.**
  - **Fashionable development methods don't suit?**
    - The risk of reworking is acceptable, or not. ⇨ Initial design/choice is important.
- **Telecom DevOps† = Development for Operations**
  - **Develop easily operationable network system**
    - Running-cost effective
  - **Architecture is important, rather than development method.**
    - Consider control system architecture as important as network architecture
    - Seek modularity
  - **"Find out unsuitable domain" by Mr. Nishi, SoftBank†**



Summary

- WVS2 was looking for Service Chaining and, Automation & Abstraction.

- SDN *was* not mature.
- System design is an eternal challenge.

- C/D-plane separation can also leverage service life-cycle control.
- Initial architectural design is important, but it should be ”good enough”.
- Attention to Control system

Welcome to SDN world through WVS2!

Thank you for your kind attention☺