Empowered by Innovation NEC

Internet Week 2014
まだ間に合う！　NFVとSDNの基本から最新動向まで

SDN のこれまでとこれから
～活用事例と最新動向～

日本電気株式会社
情報・ナレッジ研究所
鈴木 一哉
2014/11/18

自己紹介

氏名

- 鈴木 一哉 (すずき かずや)

役職

- 日本電気株式会社 情報・ナレッジ研究所 主任研究員
- 電気通信大学大学院 情報システム学研究科 客員准教授

主な職歴

- OEM (Ascend, 古河電工, ヤマハ) 製品担当
- 経路制御高信頼化に関する研究開発
- ProgrammableFlow Controller 経路計算機能の開発
- 総務省委託研究「ネットワーク仮想化技術の研究開発」(O3 プロジェクト)

SDN パートの構成

1. なぜ SDN なのか？ (鈴木)

2. SDN 概論 (鈴木)

3. SDN 最新動向 (中島)
   - 標準化動向
   - SDN/OpenFlow 実装最新動向

4. SDN 活用事例 (鈴木、中島)
   - 企業向け導入事例
   - Google G-Scale

なぜ SDN なのか？

なぜ SDN なのか？

大規模なデータセンターを、低コストに運用したい

構成変更を迅速に行いたい

サービス要件からネットワークを設計したい

現状のネットワーク機器の課題

多様なユーザニーズを満たすために、多様なプロトコル標準が存在

- 装置仕様の肥大化
- 装置コストの高騰

使いたい機能は、一部だけなのに…

自律分散が故の動作の複雑さ

構成を変更したいけど、利用中のユーザに影響を与えないようにしないと…

装置の制限により、提供可能なサービスが限定

この制限がなければ、もっとよいサービスを提供できるのに…

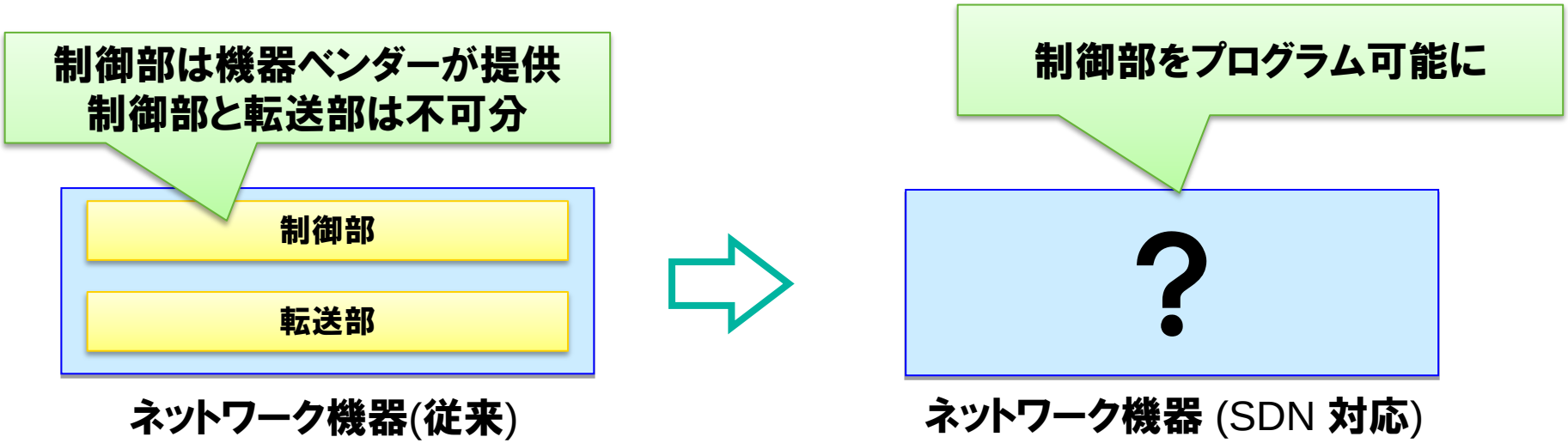
SDN **概論**

どのように？

どのように？

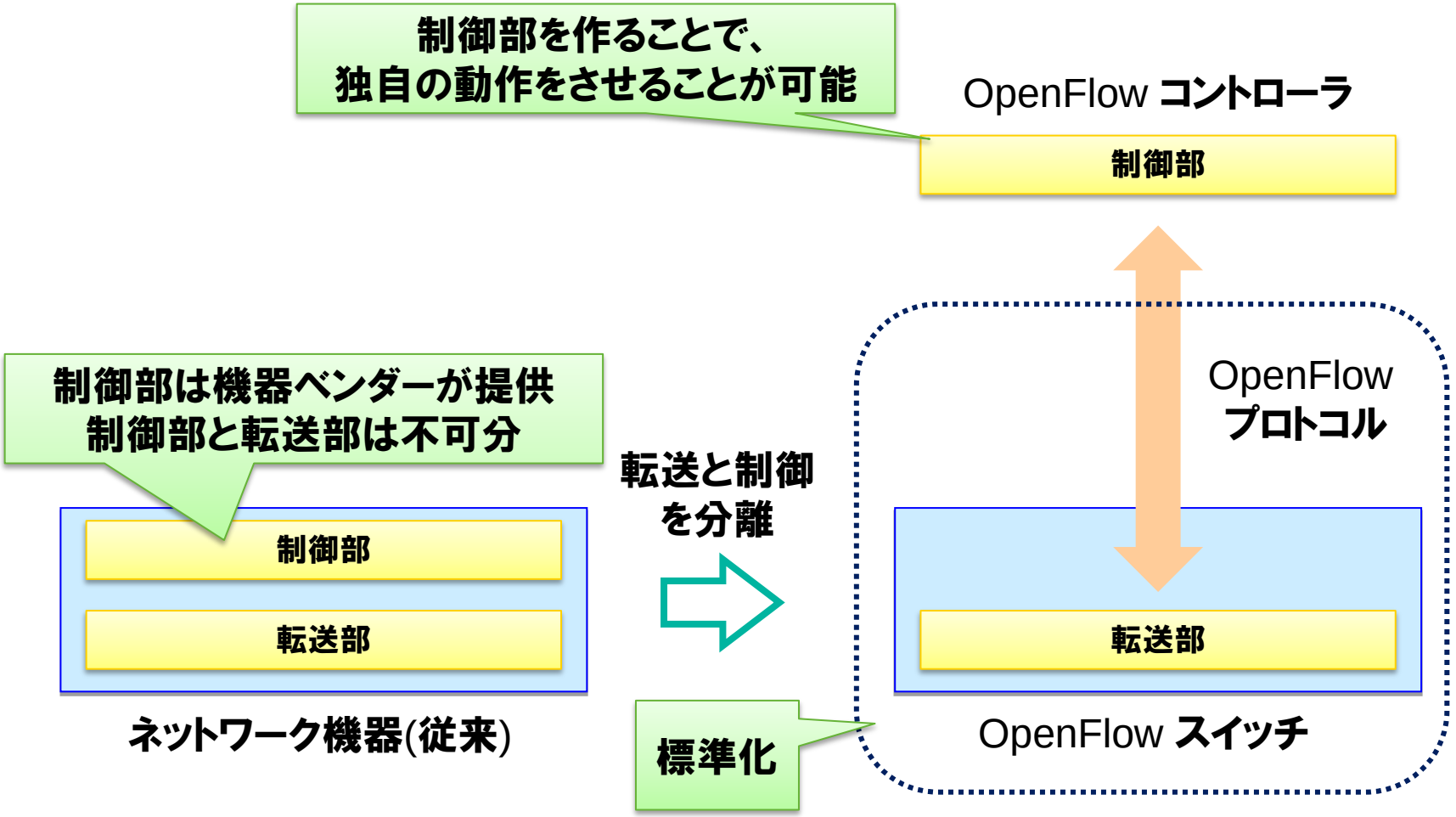
ネットワーク機器の制御部をプログラム可能に (Software-Defined Networking)

- 要件ベースでのネットワーク設計
- 自動化による運用コストの削減
- コモディティハードウェア利用による設備コストの削減



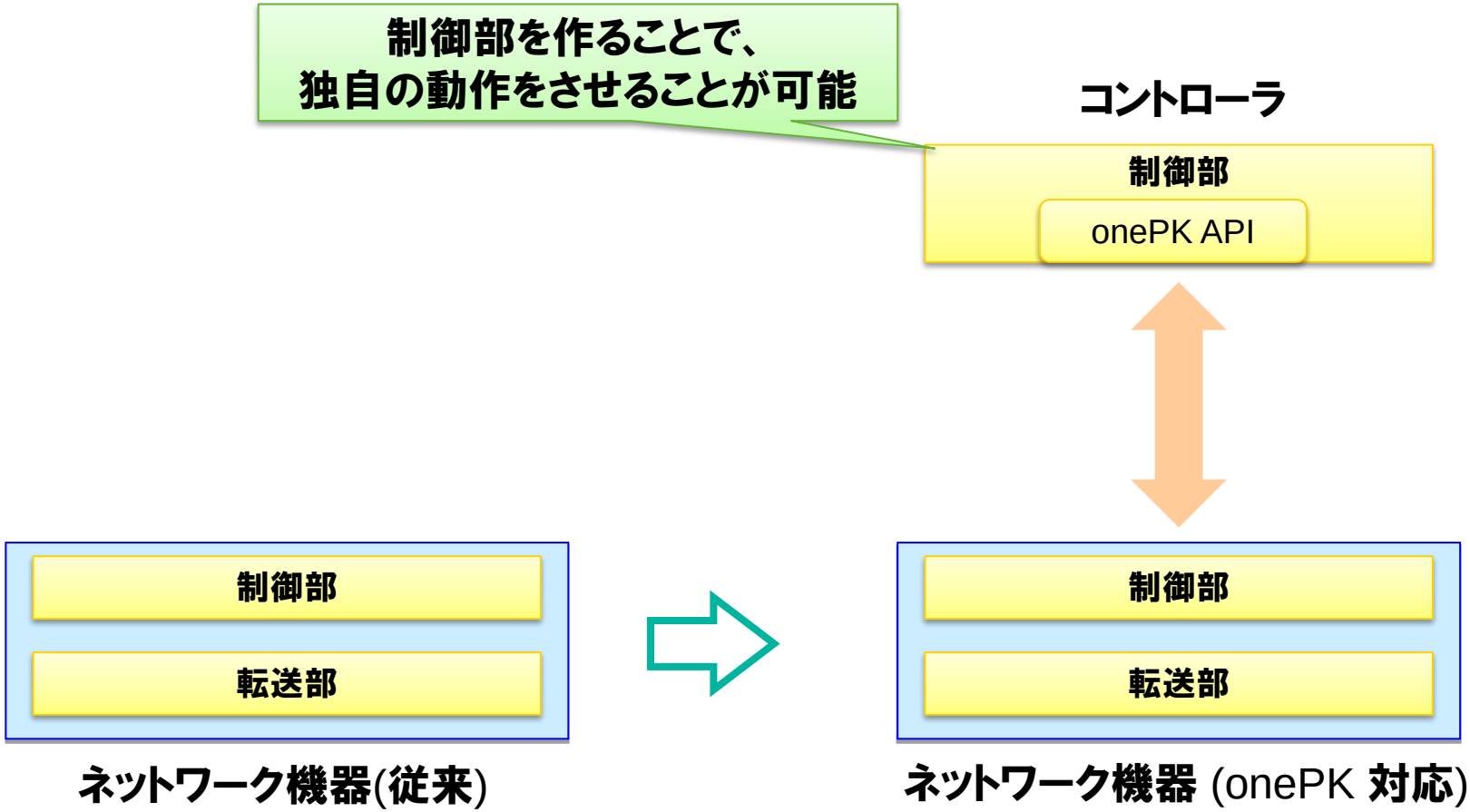
制御と転送を分離 (OpenFlow)

利点：プロトコルおよびスイッチ仕様の標準化による普及



外部からの制御を可能に (onePK)

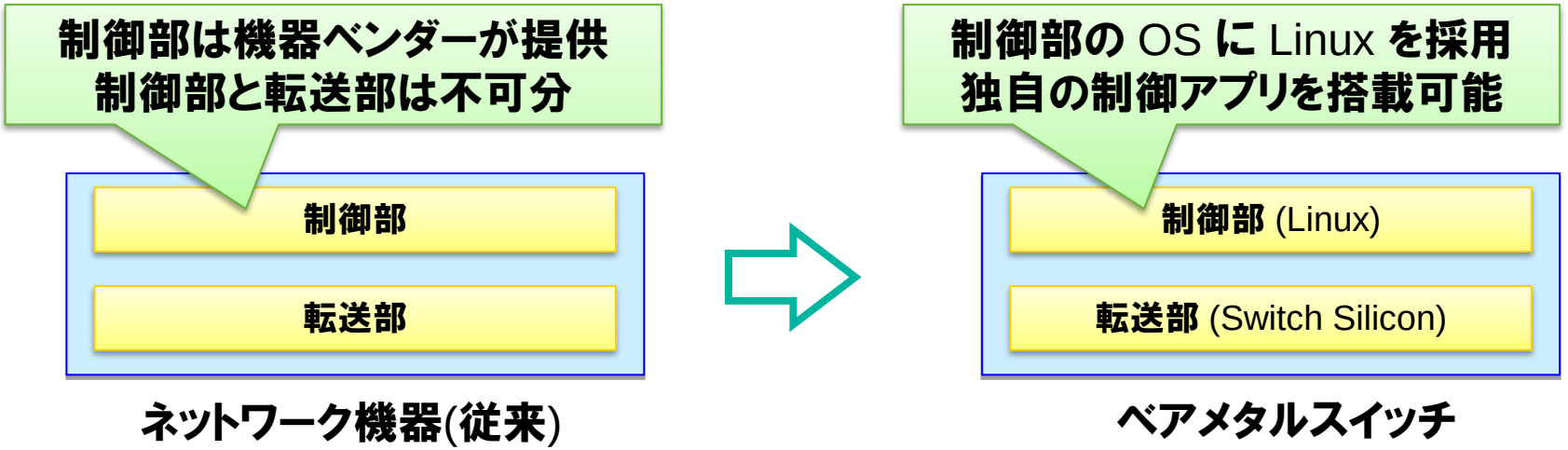
利点：従来のIOS機能による自律分散制御とonePKを用いた制御の併用が可能



スイッチ内への機能追加を可能に(ベアメタルスイッチ)

利点：コモディティハードウェアの低コスト化

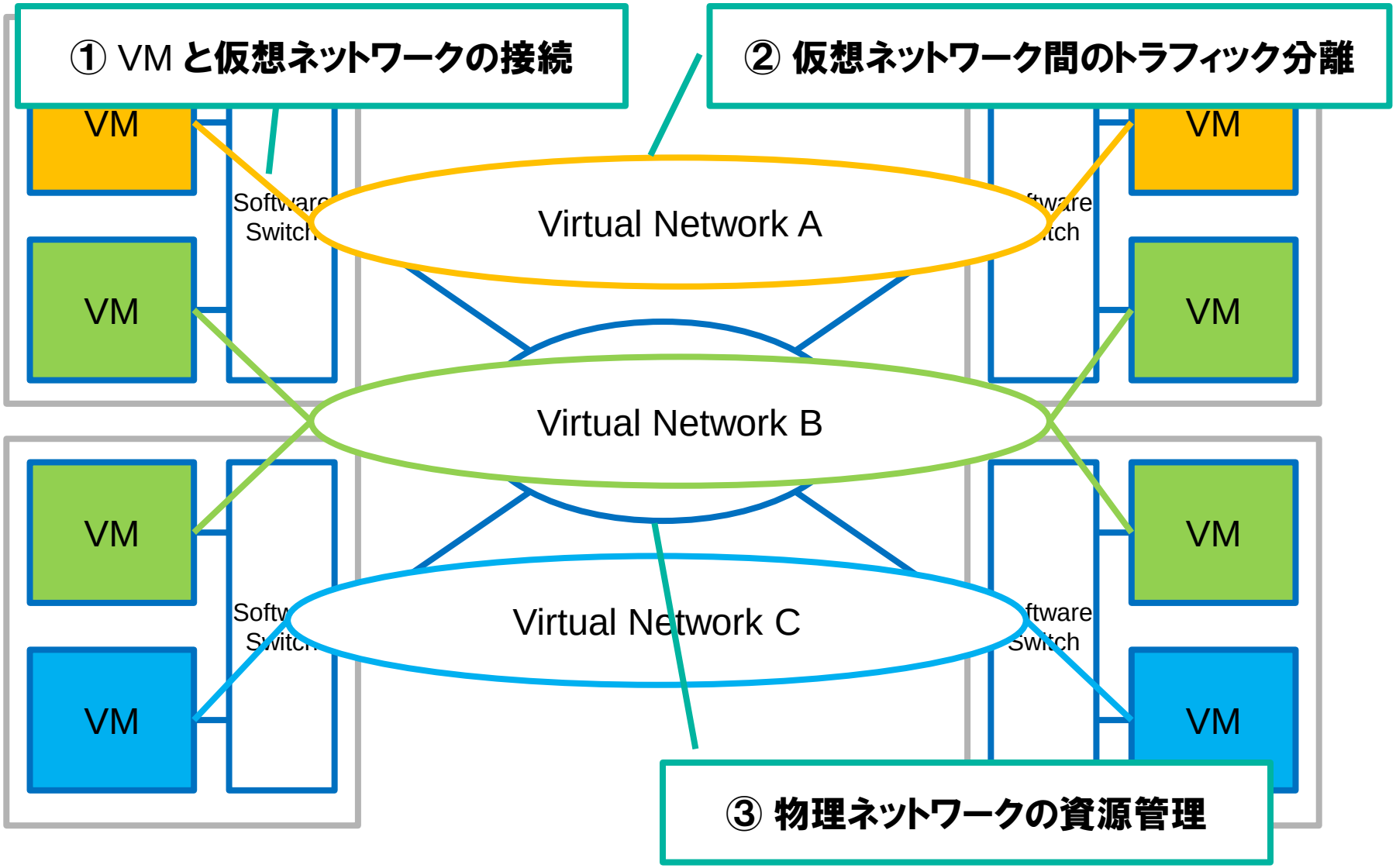
詳細については、中島さんパートで解説



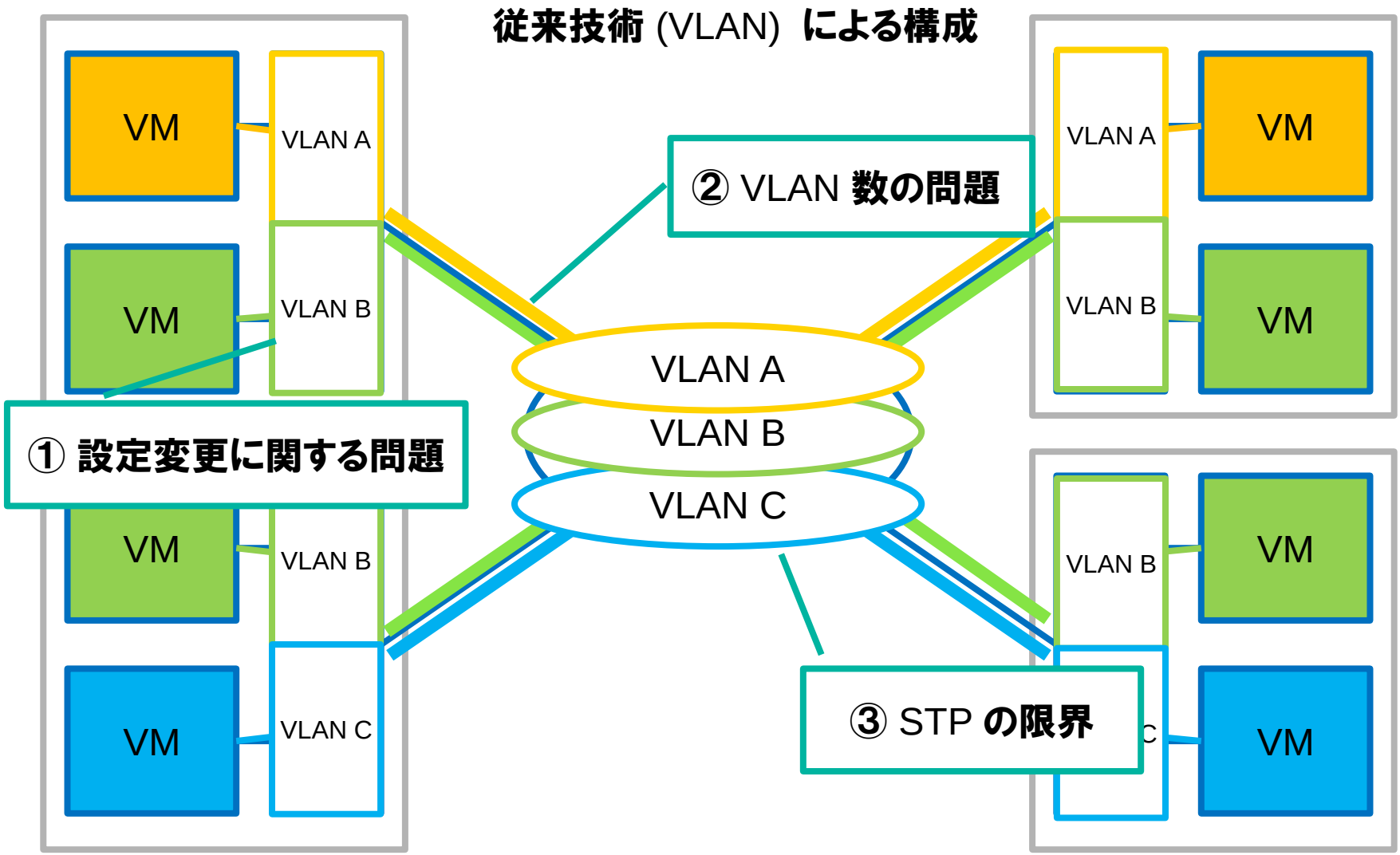
SDN **概論**

データセンターへの OpenFlow **適用例**

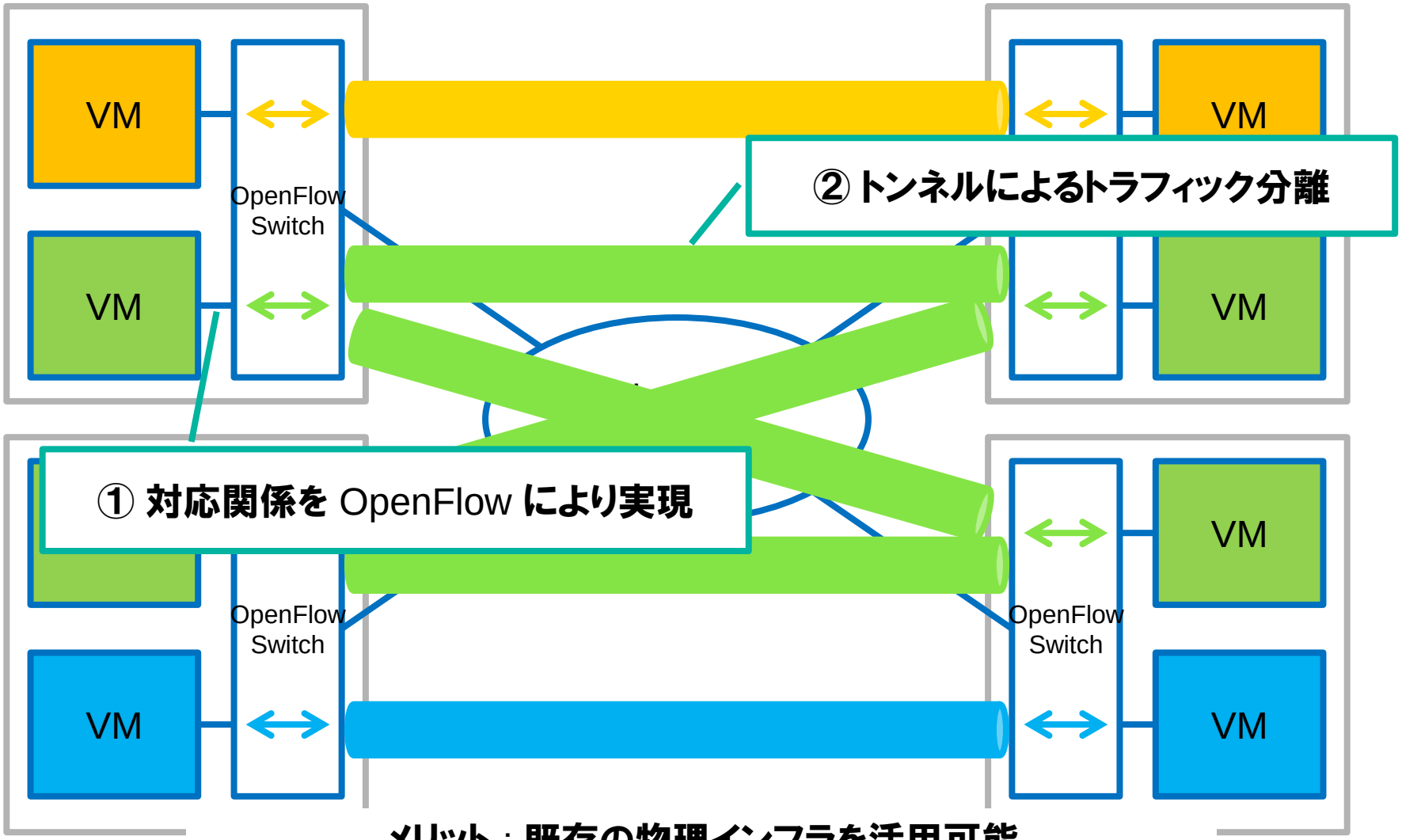
データセンターにおけるネットワーク仮想化の課題



従来技術を用いた場合の課題



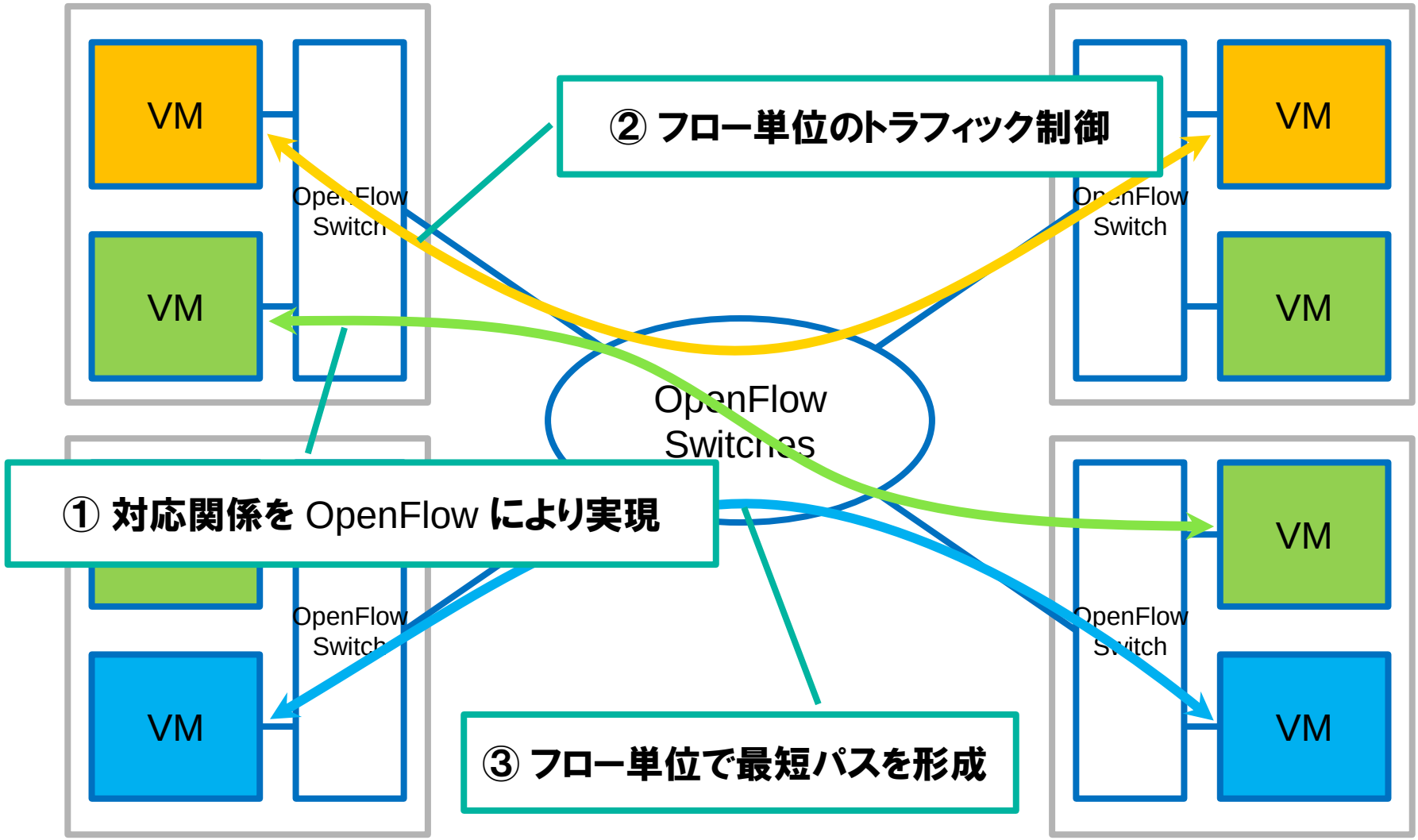
OpenFlow による構成 (Overlay アプローチ)



メリット：既存の物理インフラを活用可能

デメリット：トンネル化によるオーバーヘッド (MTU長、処理遅延)

OpenFlow による構成 (Hop-by-hop アプローチ)



Hop-by-hop 型での動作例

3. 入口、出口ポートが同一 slice に所属しているかを判定

slice

2. 宛先MACアドレスから
出口スイッチ、ポートを検索

4. 出口までの最短パスを計算

fdb

Sliceable Switch

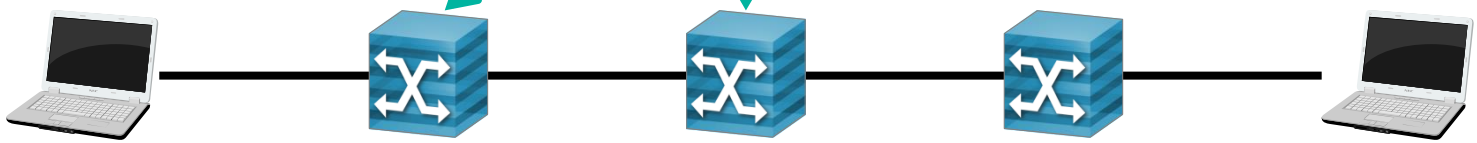
Trema

topology

1. packet_in

5. flow_mod

6. packet_out



フロー単位で、コントローラが判断を行い、パスを構成
→ 設定変更をスイッチに落とし込む必要がない

クラウド基盤ソフトウェアとの連携

クラウド基盤ソフトウェア
(OpenStack 等)

OpenFlow コントローラ

OpenFlow
プロトコル

サーバ

ネットワーク

SDN の活用事例

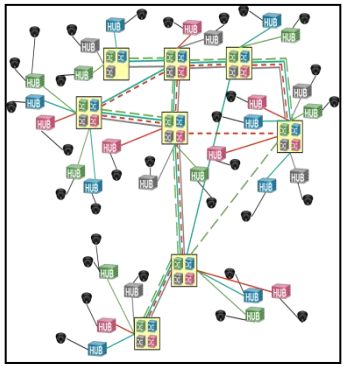
SDN の活用事例

企業向け導入事例

ネットワークを分けておきたい！ (ProgrammableFlow 導入事例)

「ネットワークを分けておきたい！」というお客様

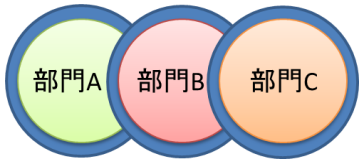
従来のNWの課題



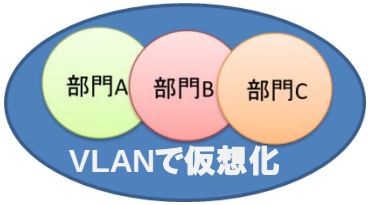
従来のNW技術では、それぞれの物理的な経路上に装置が必要。増設時の設定変更など、運用管理も煩雑になってきていました。

このような状況で、「ネットワークを分けておきたい」ワケをお持ちのお客様には様々な苦労が・・・

様々な*理由(ワケ)*からネットワークを分けたい、、



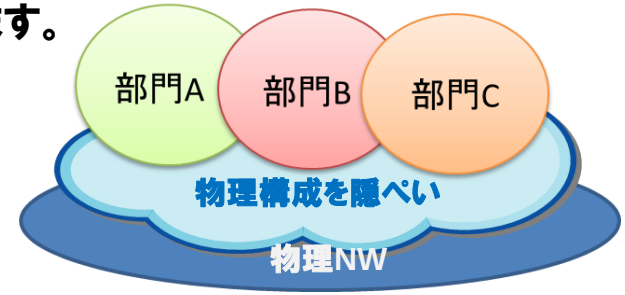
物理的にネットワークを分離して敷設している
⇒ LANだらけ！



個々の機器への設定でネットワークを分けている
⇒ 構成変更時、設定が大変！

ProgrammableFlowが解決！

ProgrammableFlowなら、1つの物理ネットワークの上に、複数の独立した仮想的なネットワークを作れます。

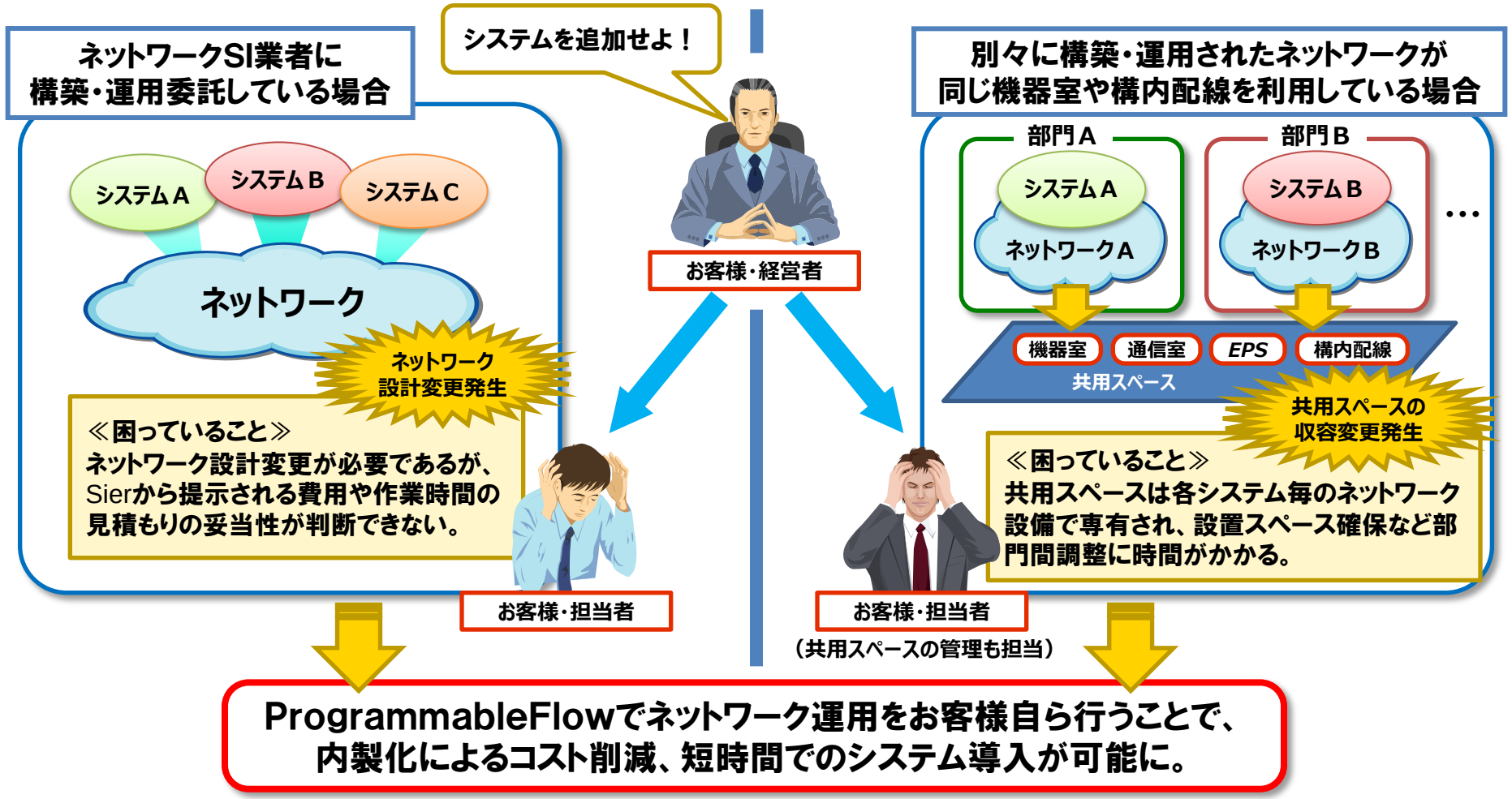


仮想ネットワークはGUIで作成！不要になったら簡単削除。
物理的な敷設コスト、運用管理コストの両面でメリット！仮想ネットワークは各々独立しているのでセキュリティ面でも安心です

サーバ仮想化と同じように、物理ネットワークを共有して利用し、NWを統合できます！

ネットワーク運用管理を取り戻したい!(ProgrammableFlow 導入事例)

「ネットワーク運用管理を取り戻したい！」というお客様



設備投資費、運用管理費を削減したい! (ProgrammableFlow 導入事例)

設備投資費(CAPEX)、運用管理費(OPEX)を削減したい!

設備担当者の悩み

スモールスタートから始めて、設備投資(CAPEX)を低く抑えたいのに、将来を見越して製品選定をすると、高くついてしまう・・・
もっとネットワーク機器が柔軟に対応できたらいいのに・・・

設備担当者

経営者

コスト削減!

せっかくサーバを仮想化して運用費(OPEX)を削減できたと思ったのに、ネットワークの設定変更には相変わらず時間とコストがかかっている・・・

運用管理担当者

運用担当者の悩み

コスト削減と言われても、運用管理費(OPEX)はこれ以上の削減は難しい・・・
切り替えは深夜帯や年末年始にやるしかないし、設定変更は外注するしかないし・・・

ProgrammableFlowでネットワーク基盤を構築。
⇒ 設備投資費や運用管理費の削減に貢献

安心・安全・安定した医療環境の実現！～ 金沢大学附属病院 様 ～

ITネットワークが主業務ではない、大学病院でも運用できるネットワークを手に入れた！

導入の目的

急速に進歩する医学および医療技術の発達に追従できる安全・安定したネットワーク基盤

ミッションクリティカルな業務を遂行するために24H 365日安全・安定して稼動できるインフラが必要！

導入前の課題

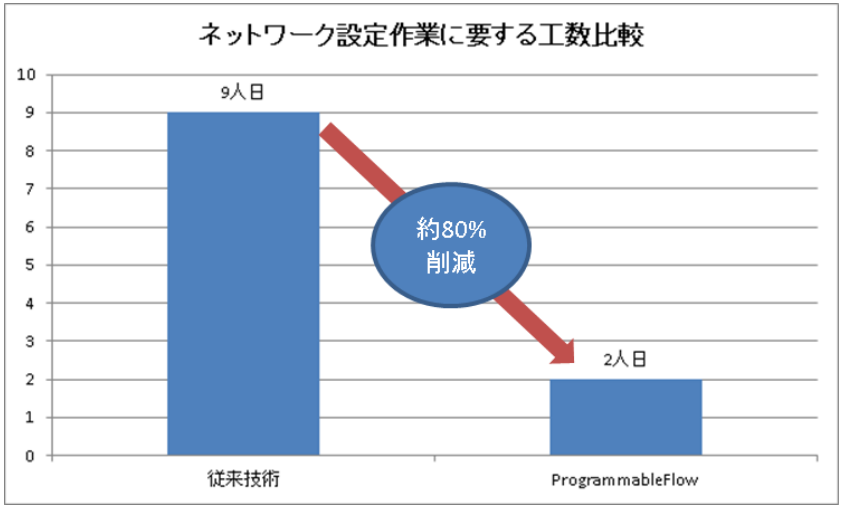
診療科毎に特定のベンダの医療機器が導入されているため、国内外からアクセスされる際のセキュリティを考慮して、**物理的に診療科毎にLANを分割。**

NWが複雑化し、機器の誤接続や誤設定などのリスク大。

LANの乱立への歯止めと、複雑化したネットワークの管理

導入の効果

- ネットワーク設定の人的コストを**約80％削減**
- ProgrammableFlowのネットワーク仮想化により、物理的な配線不要。**配線費用はゼロ**に。
- 回線障害発生時も迅速に自動回復（従来分単位で回復していたところを秒単位に短縮）



構築しては解体する研究開発用NWにSDNが響いた！
～ 新日鉄住金ソリューションズ　様～

様々な研究開発PJに合わせた環境を迅速に提供！開発効率も業務効率も向上！

導入の目的

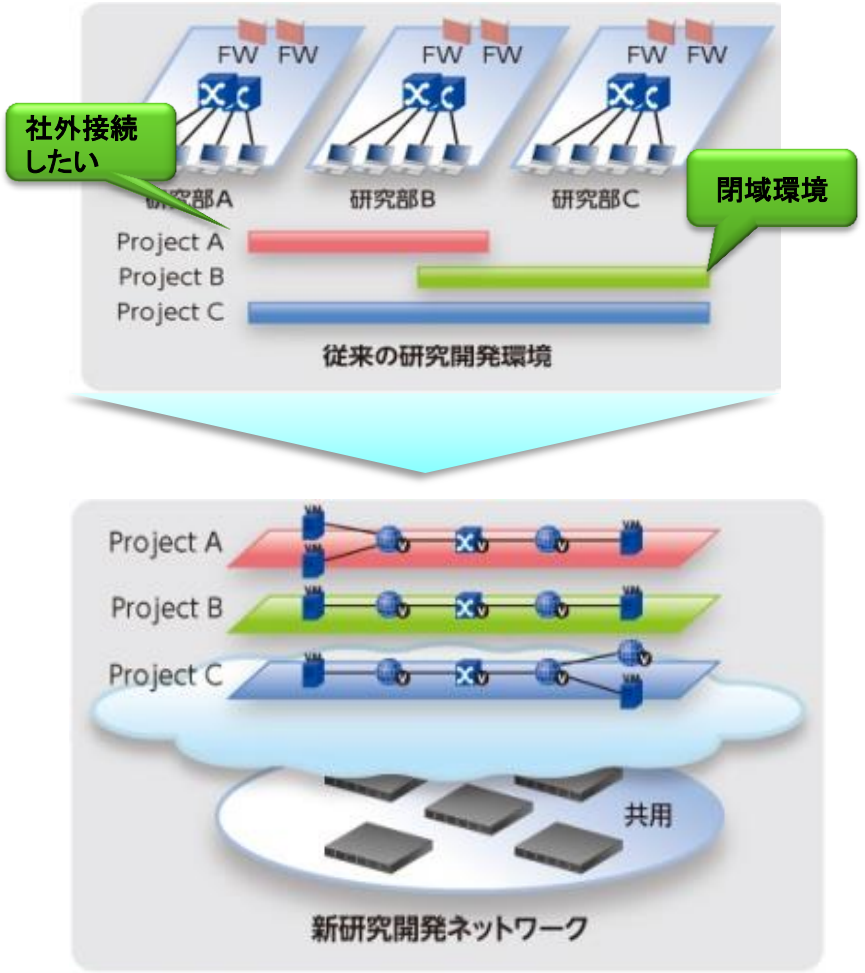
研究開発PJ発足・完了の都度、構築・解体する必要があったNWの運用負荷の軽減および開発環境の提供時間の短縮し、研究開発の効率を向上させたい

導入前の課題

- 研究開発PJが頻繁に発足し、そのPJ毎にNW機器の調達および開発用NWを構築が必要。
  部門をまたがるようなプロジェクトも増加し、従来技術のVLANでの対応では限界！
  その開発環境の提供作業がNW管理者に多大な負荷をかけていた。

導入の効果

- 物理NWはそのままで、研究開発PJ毎のネットワークは**論理的に柔軟に敷設・変更・削除が可能**となり作業負荷を大幅に軽減。
- 開発用環境を迅速に提供でき、**開発効率も向上**。



物理ネットワークは全プロジェクトで共有。
プロジェクトごとのネットワークは論理的／柔軟に敷設・変更・削除可能に

最適な開発環境とリソースの効率運用を実現
～ NEC ソフトウェアファクトリ（全社統合開発環境）～

導入の目的

- 新DCの立ち上げを契機にBCP/DR対策かつ、双方向で負荷分散可能な環境を実現したい。
- 無停止稼働が求められる全社統合開発環境を柔軟な共通基盤として整備したい。

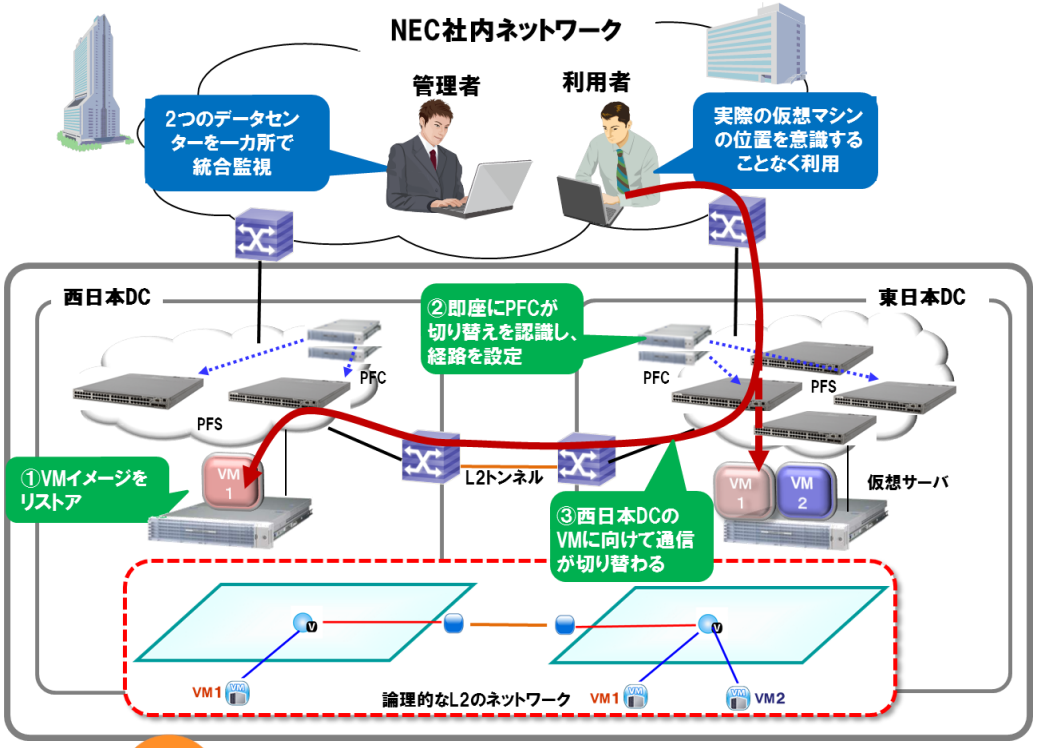
導入前の課題

- メンテナンス等のために仮想マシンの移動が必要な際には様々変更作業が必要。手間がかかる上に設定ミスのリスクがあった
- 開発検証環境を物理的に準備していたため、コストや時間が掛かっていた。
- 双方のDCで個別に余剰リソースを確保していたため、無駄が発生していた。

導入の効果

- バックアップ環境の実現に両DC間のネットワーク構成を同一にする必要がなく、**設備投資を抑えた相互バックアップDR環境を実現**。
- **両DCのICTリソースを共有しながら利用者に開発環境を迅速に提供**。
- 現場での**NW設定変更作業を大幅に削減**。

仮想サーバの増設・変更が容易なため
「開発環境の迅速な提供」
「余剰リソースの最小化」が可能



Point

メインDCと同規模の機器を
調達・構成する必要はありません。
BCP/DR対策コストを低減できます！

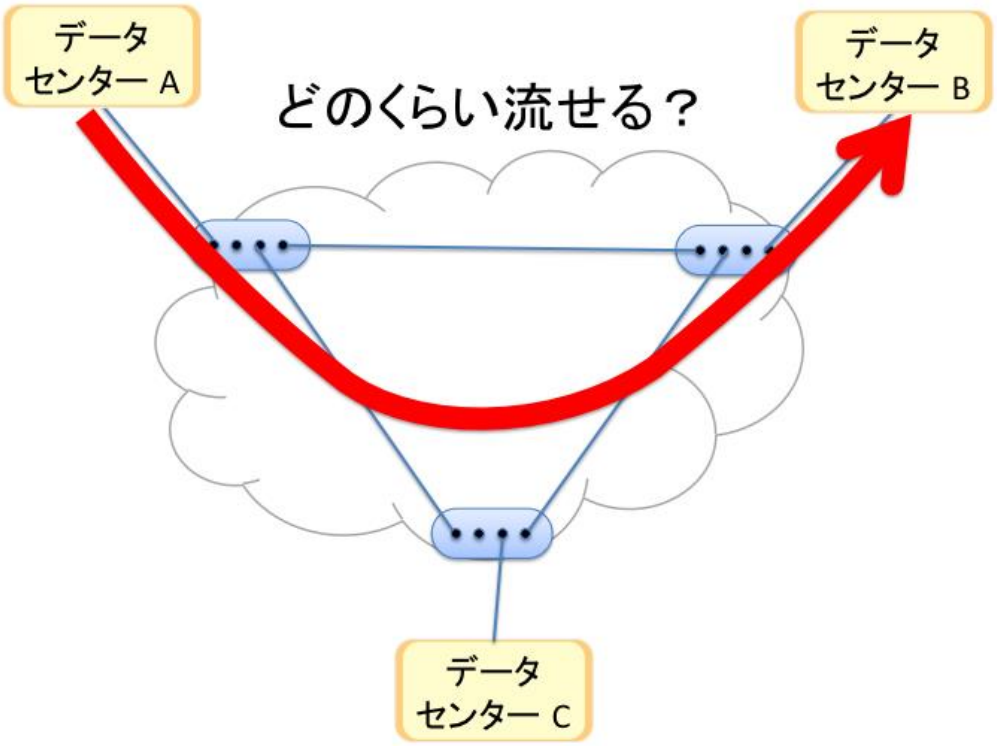
SDN **の活用事例**

Google G-Scale

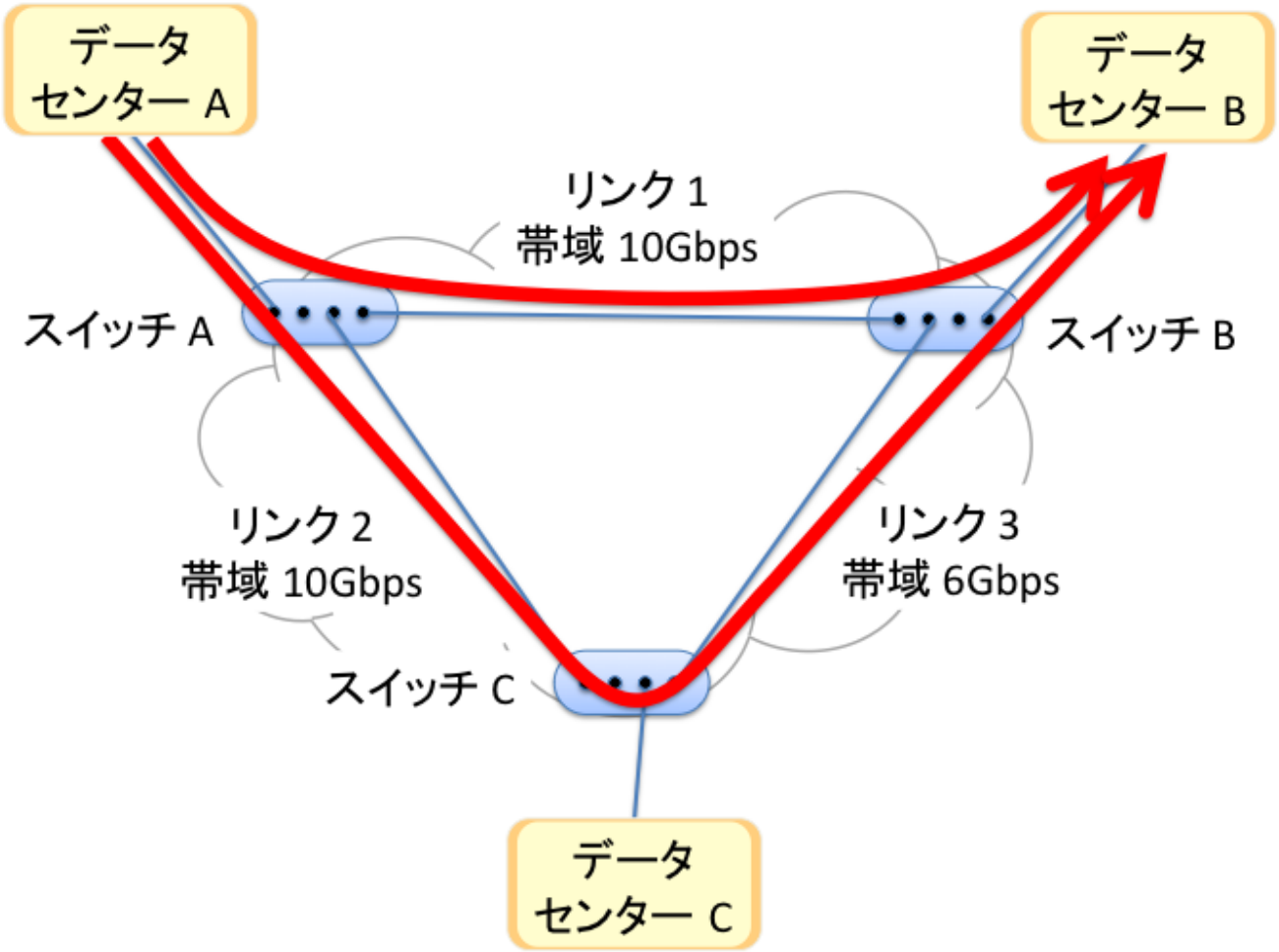
Google における OpenFlow の活用

データセンター間ネットワーク (G-Scale)

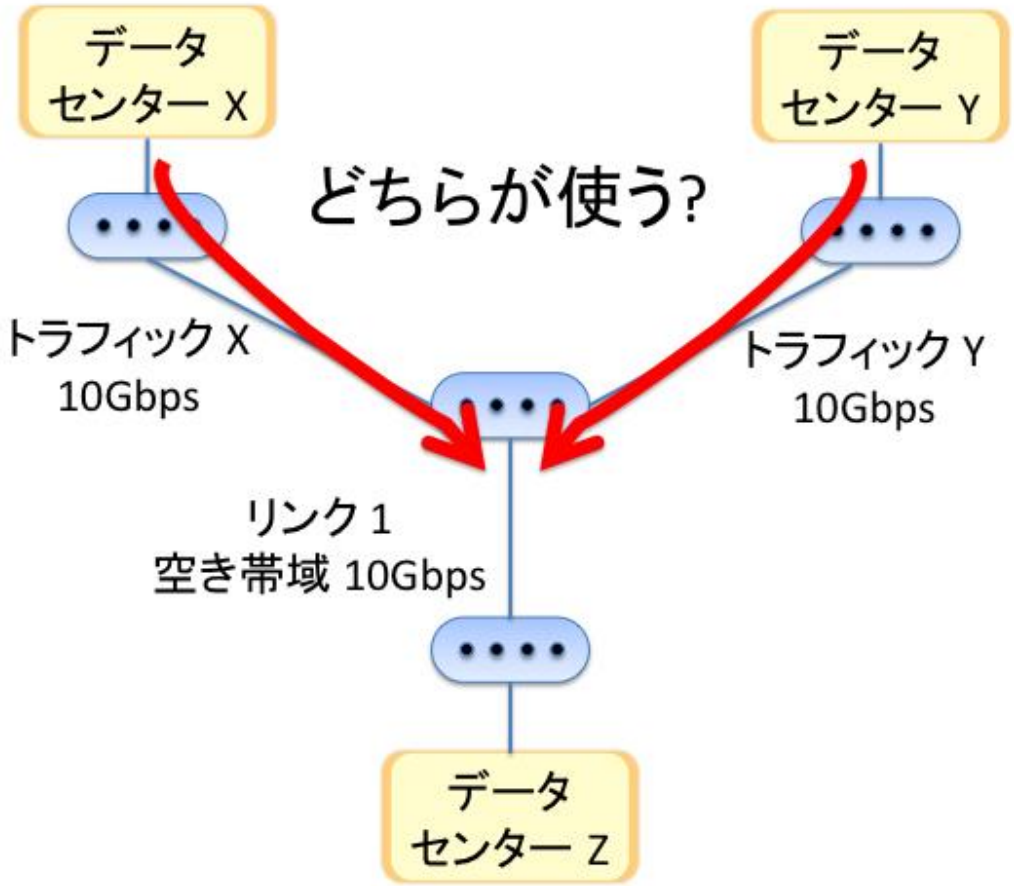
- 急増するトラフィック
- 徹底したコスト削減
- → これらの課題解決を両立させるために、OpenFlow を活用



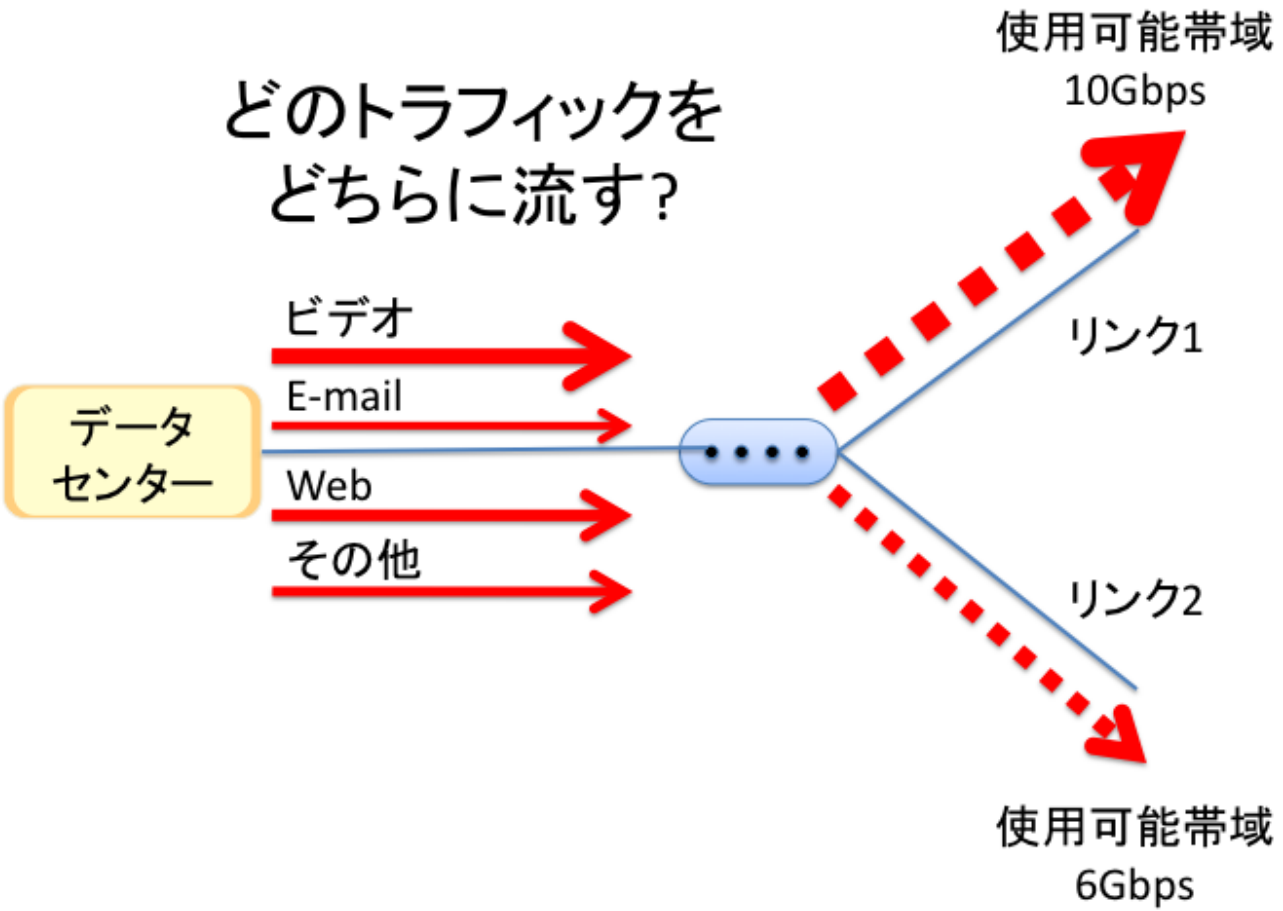
WAN 回線有効活用の課題 (空き帯域の把握)



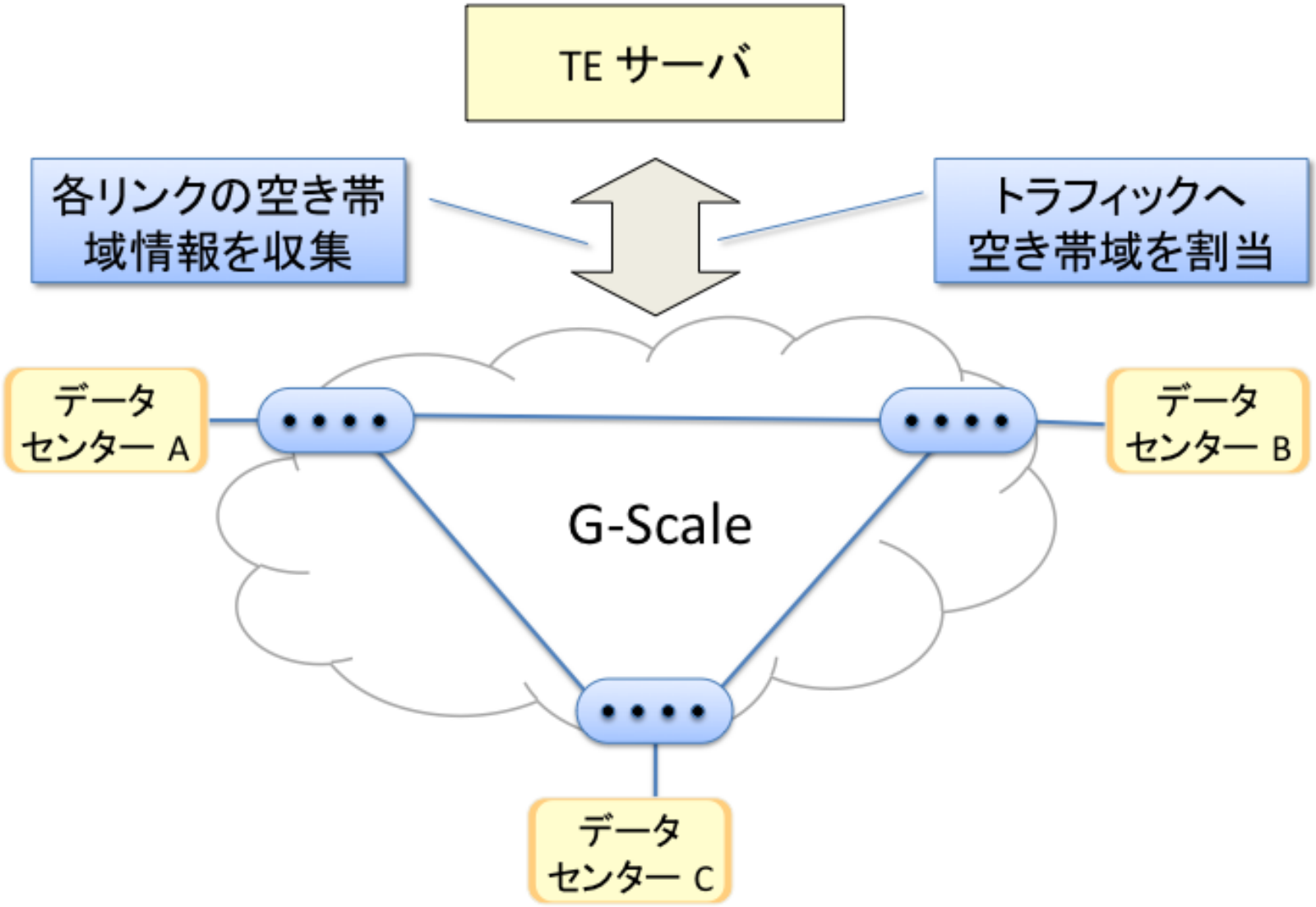
WAN 回線有効活用の課題 (どちらが使うかの調停)



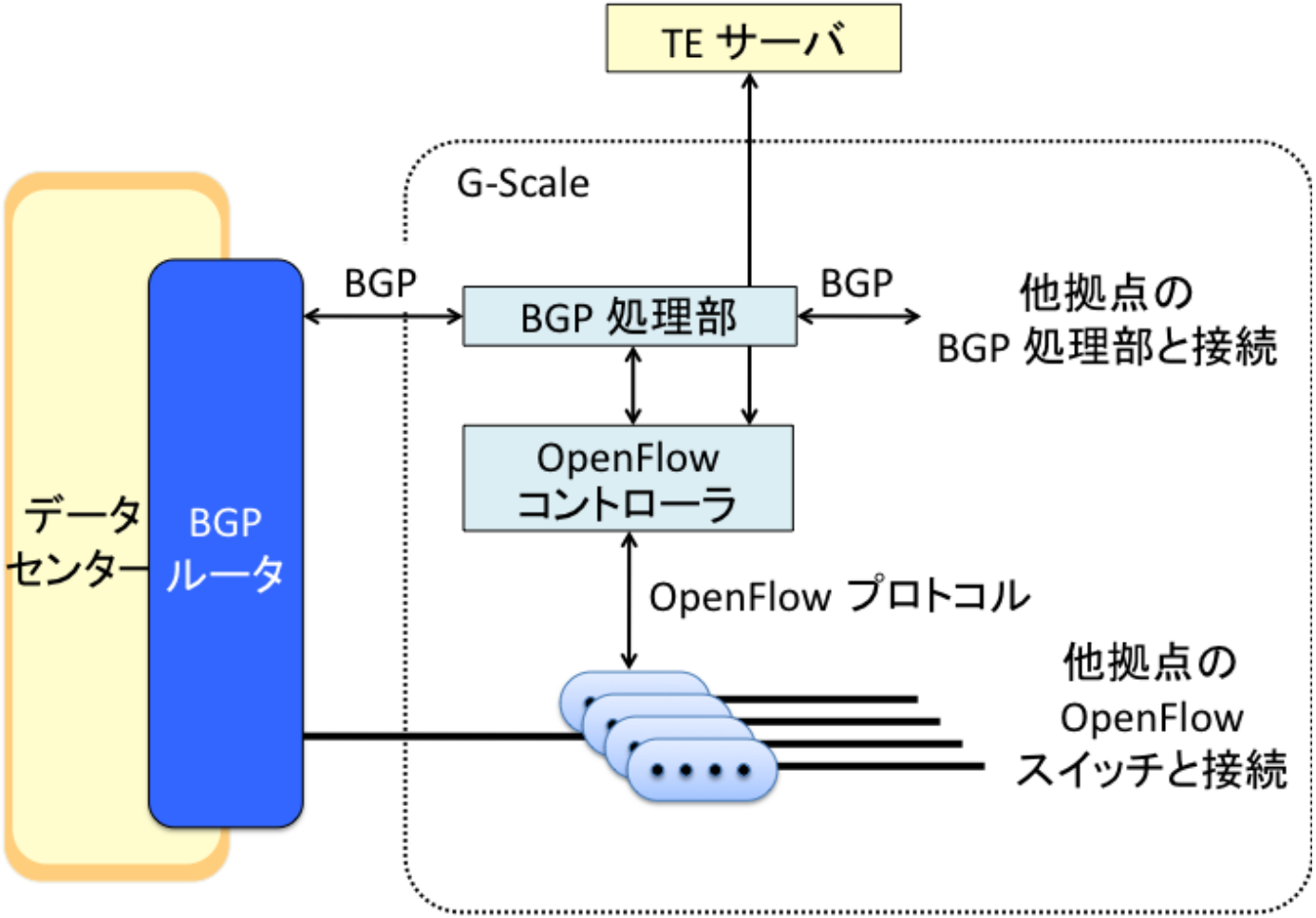
WAN 回線有効活用の課題 (トラフィック毎に帯域の割当)



課題の解決 (TE サーバの導入)



課題の解決 (OpenFlow によるトラフィック種別毎の制御)



Google G-Scale まとめ

WAN 回線の有効活用に OpenFlow を活用

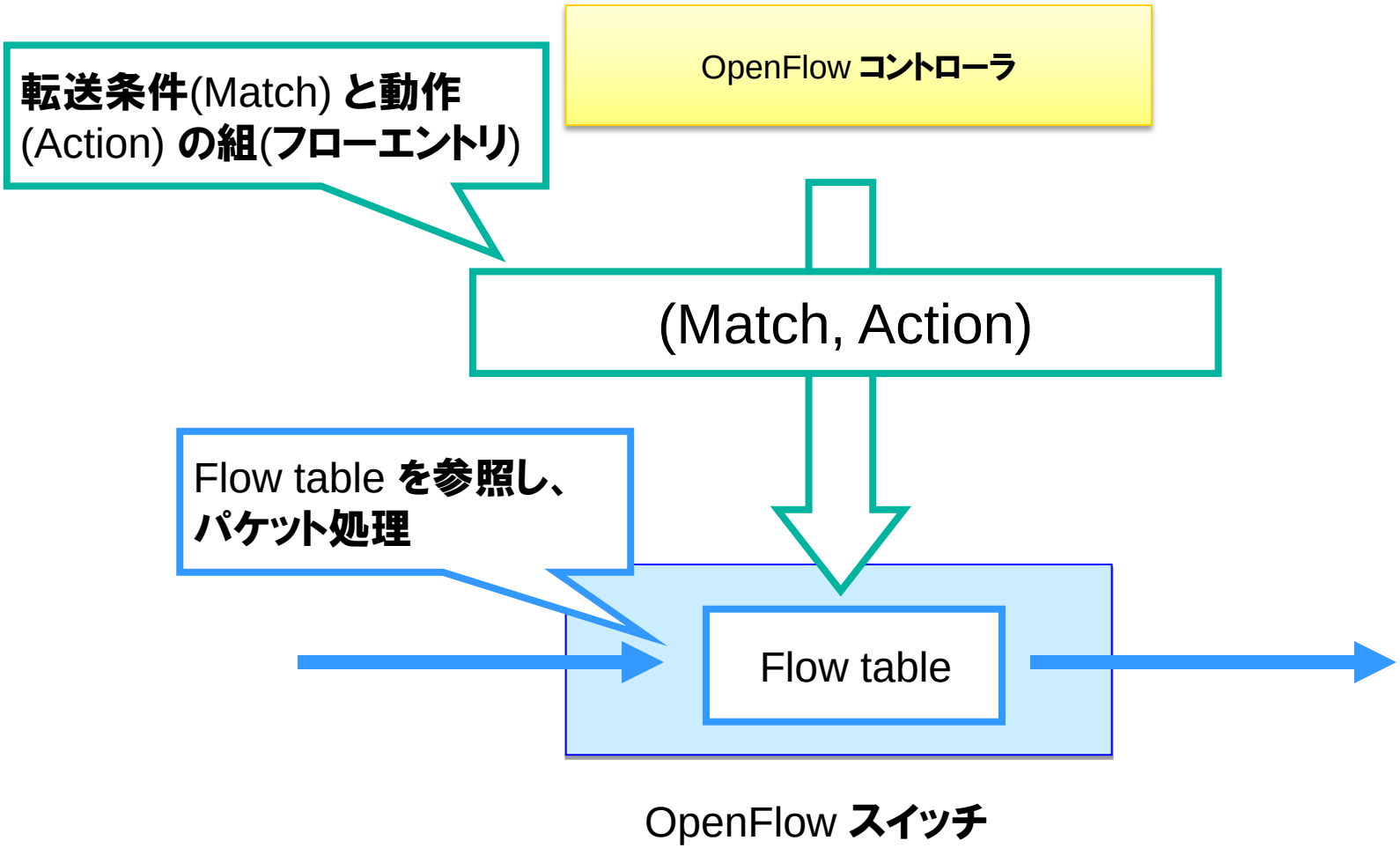
- TE サーバの導入 (Global view からの帯域割当)
- OpenFlow によるマルチパス転送

Google は、OpenFlow の実運用をすでに行なっている

参考資料

OpenFlow 概要

OpenFlow 概要 (スイッチの動作)



OpenFlow 概要 (Match)

- Ingress port
- Ethernet source/destination address
- Ethernet type
- VLAN ID
- VLAN priority
- IPv4 source/destination address
- IPv4 protocol number
- IPv4 type of service
- TCP/UDP source/destination port
- ICMP type/code

L1 から L4 まで 12 tuple のヘッダフィールドを利用可能

OpenFlow 概要 (Action)

- Forward
  - Physical ports (Required)
  - Virtual ports : All, Controller, Local, Table, IN_PORT (Required)
  - Virtual ports : Normal, Flood (Required)
- Enqueue (Optional)
- Drop (Required)
- Modify Field (Optional)
  - Set/Add VLAN ID
  - Set VLAN priority
  - Strip VLAN Header
  - Modify Ethernet source/destination address
  - Modify IPv4 source/destionation address
  - Modify IPv4 type of service bits
  - Modify IPv4 TCP/UDP source/destination port

さまざまな転送ルール

ヘッダが書き換えも可能

アクションを複数してすることも可能

OpenFlow 動作モデル

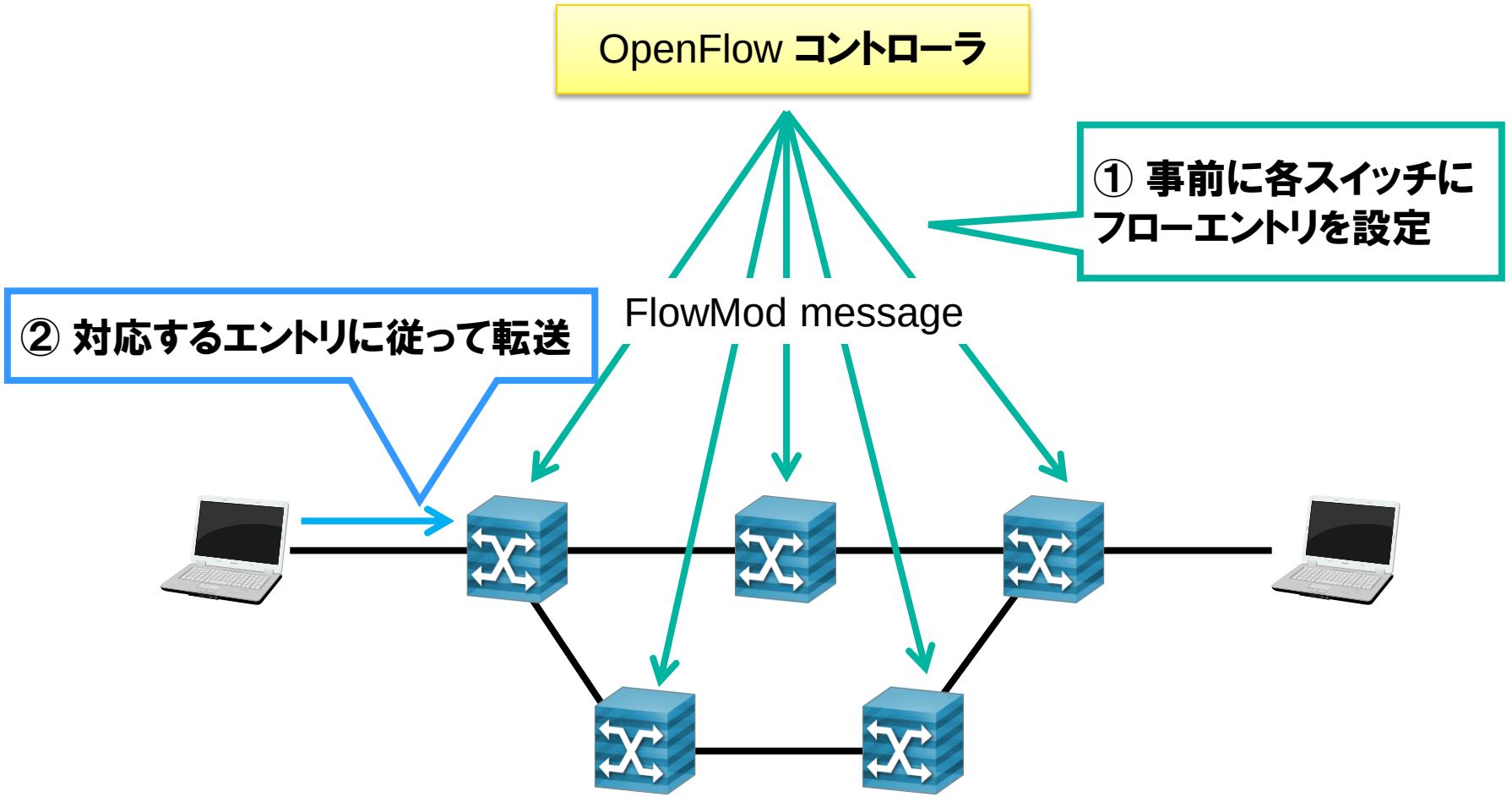
Proactive 型

- フローテーブルにエントリを、事前に設定しておく

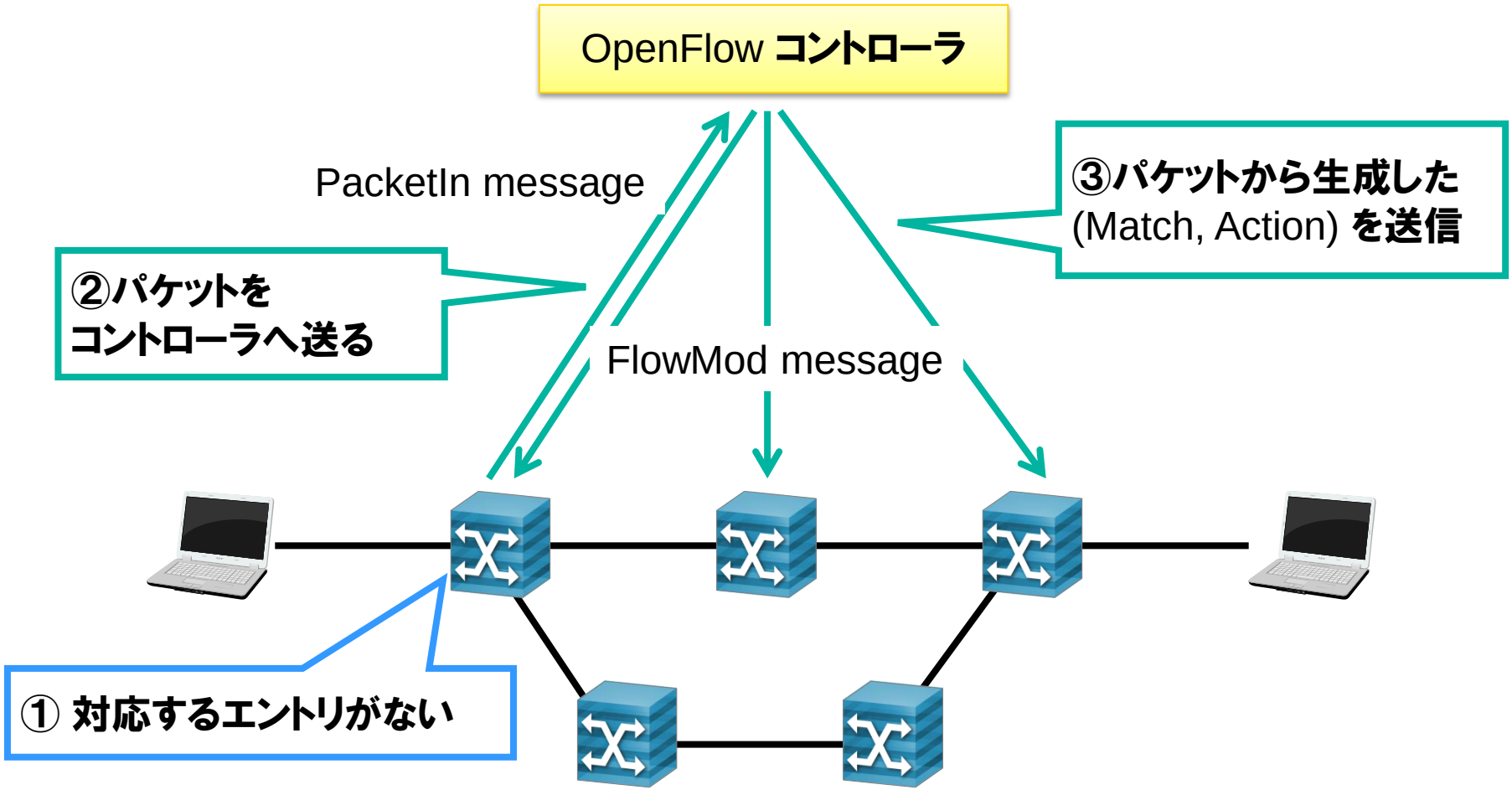
Reactive 型

- パケット受信時、対応するエントリがない場合、そのパケットをコントローラへ送る
- コントローラは、パケットからフローエントリを作成し、スイッチへと送る

OpenFlow **動作モデル** (Proactive **型動作**)



OpenFlow **動作モデル** (Reactive **型動作**)



OpenFlow **概要** (Messages)

パケット

- Packet in : **スイッチ ⇒ コントローラ**
- Packet out : **コントローラ ⇒ スイッチ**

フローエントリ

- Flow mod : **コントローラ ⇒ スイッチ**
- Flow removed : **スイッチ ⇒ コントローラ** (expire **などでエントリ消去時**)

マネージメント

- Port status : **スイッチ ⇒ コントローラ** (**ポートの状態変化時**)
- Echo request/reply
- Features request/reply
- …

OpenFlow Protocol Standard

OpenFlow Switch Specification

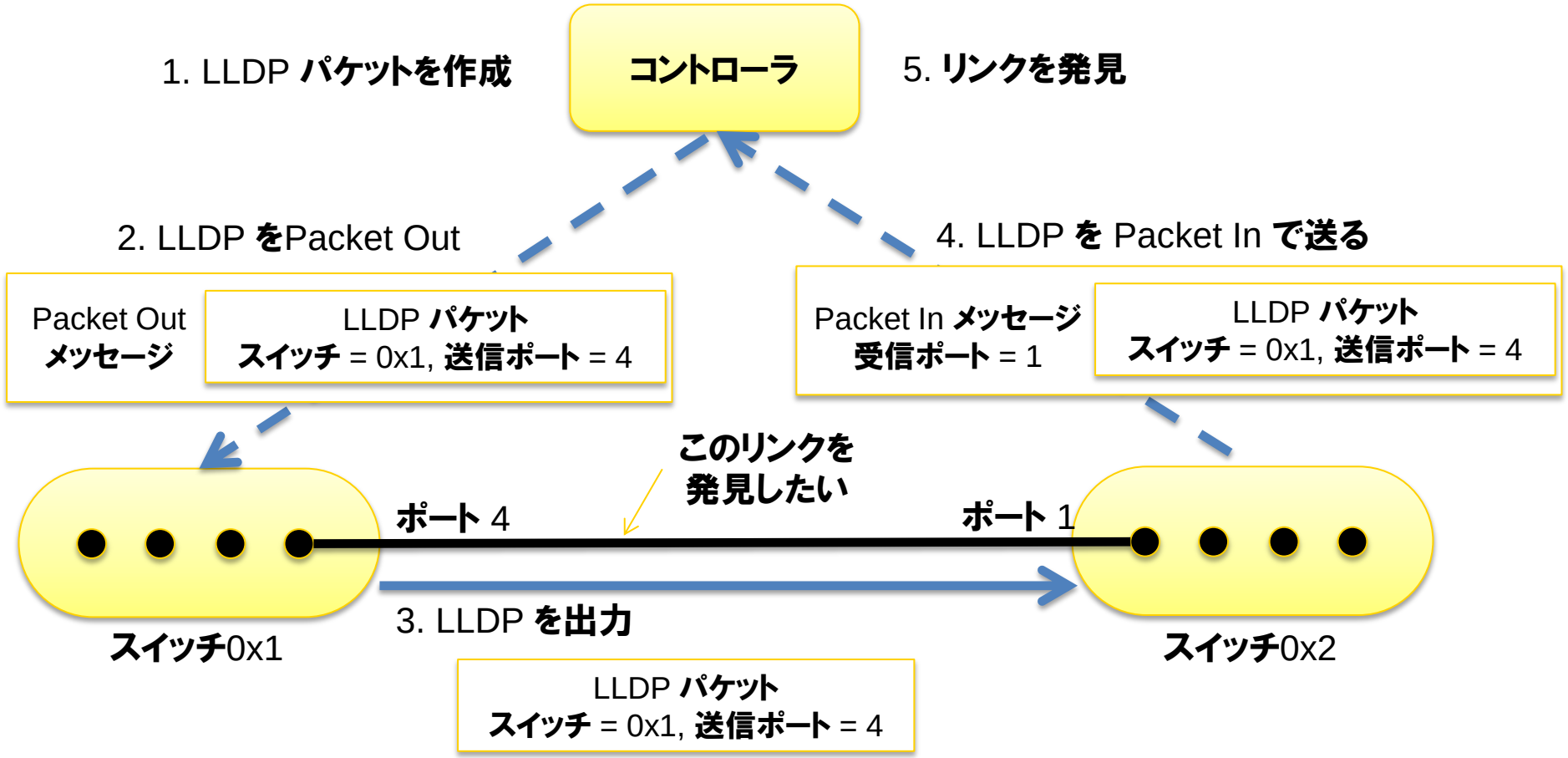
- 1.0 (2010/3)
  - **初期普及バージョン**
- 1.1 (2011/2)
  - MPLS shim header, multiple table, etc
- 1.2 (2011/12)
  - IPv6, etc
- 1.3 (2012/4)
  - PBB, etc
  - Long term support (**今後、利用拡大が予想される**)
- 1.4 (2013/10)
  - Optical port **のサポート**

参考資料

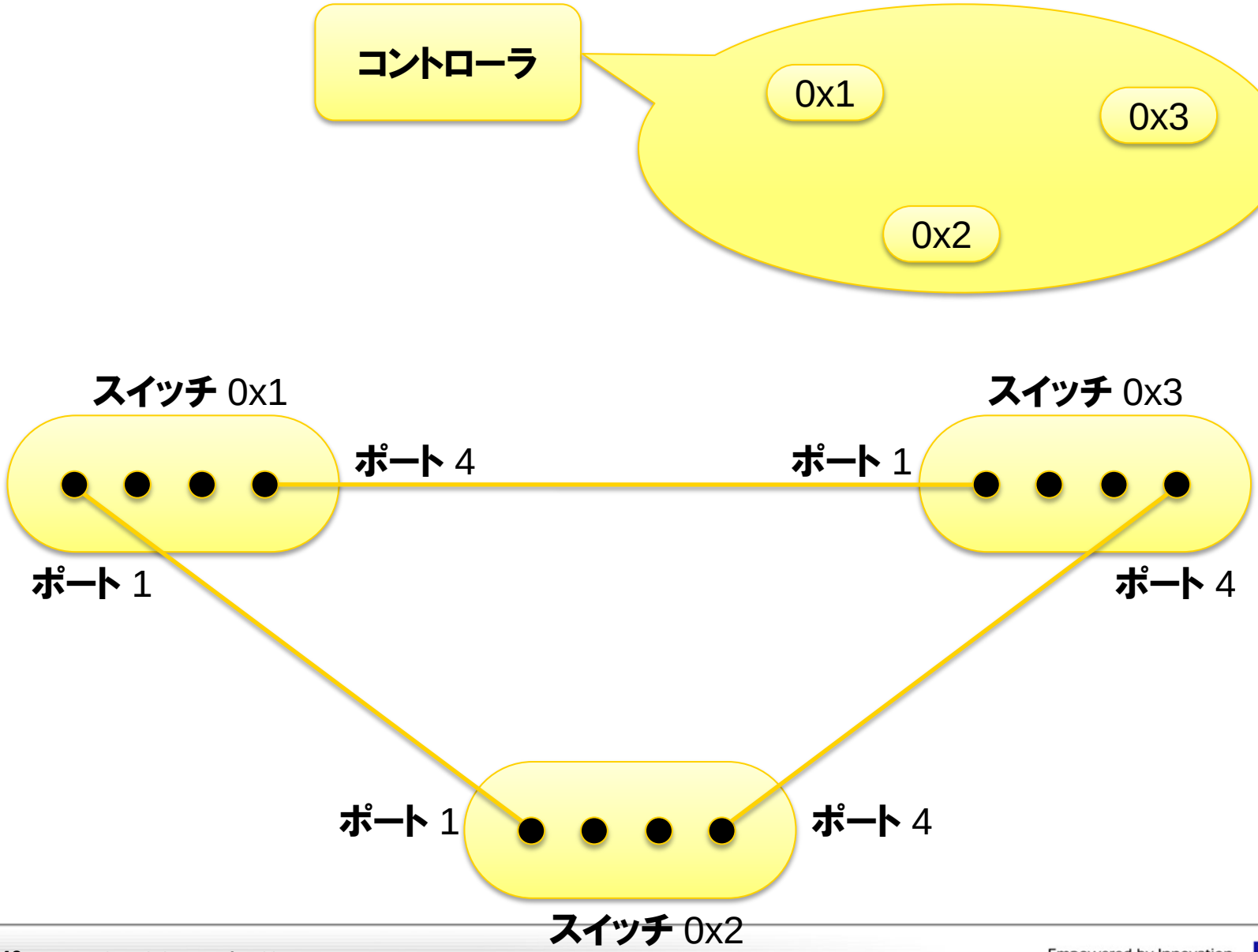
OpenFlow でのトポロジー探索

リンクの発見

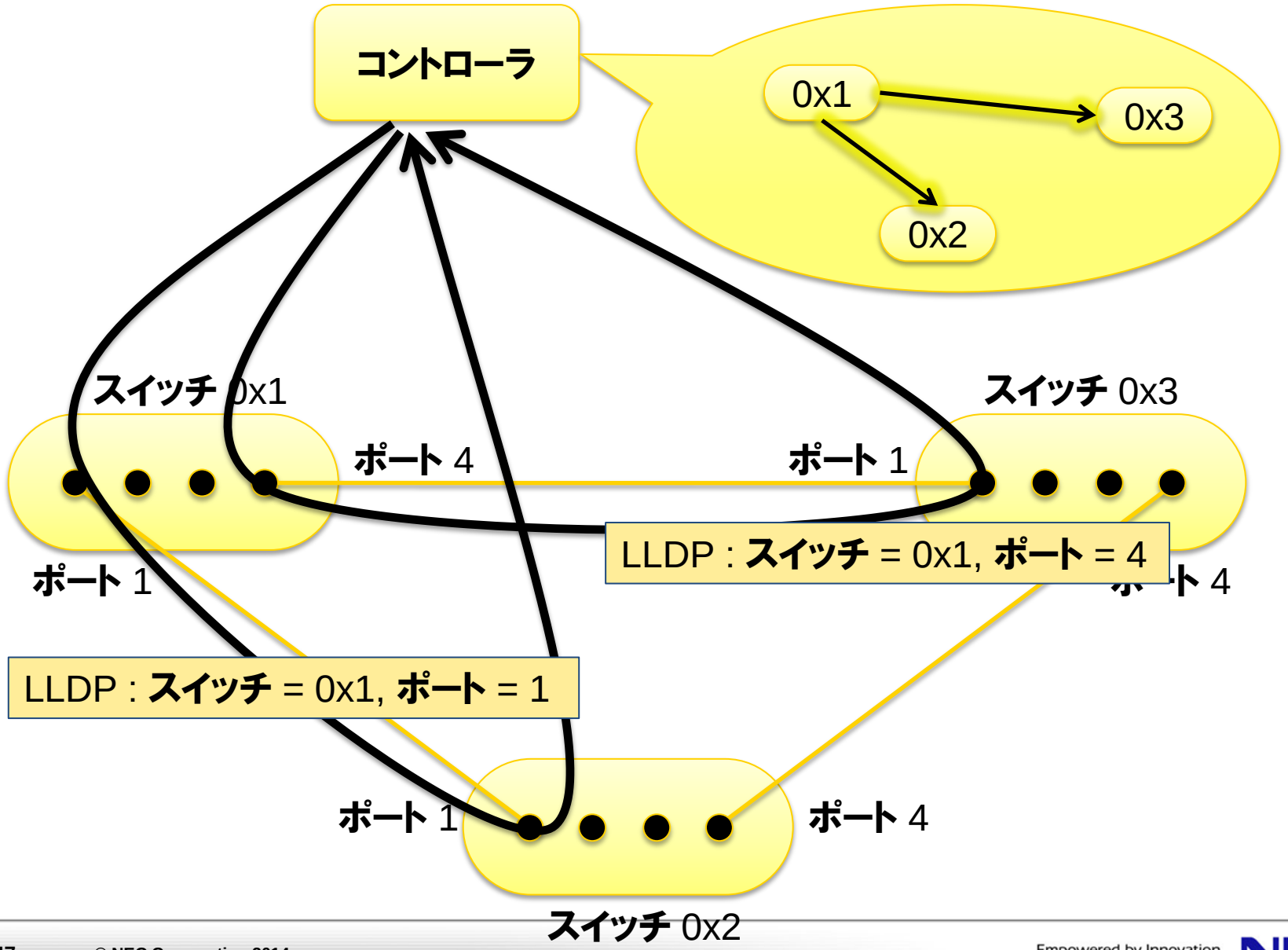
コントローラは、LLDP パケットを作り、Packet Out メッセージでスイッチに送信。
Packet In メッセージ中の LLDP パケットを参照することで、スイッチ間のリンクを発見。



トポロジーの探索 (1)



トポロジーの探索 (2)



トポロジーの探索 (3)