

ネットワークコンピューティング 第12回

中山 雅哉 (m.nakayama@cnl.t.u-tokyo.ac.jp)
関谷 勇司 (sekiya@nc.u-tokyo.ac.jp)

本日のトピック

- SDN の詳細
 - OpenFlow を使ったプログラミング
- OpenFlow の演習
 - Mininet + Ryu (SDN Controller) を利用した演習環境

最終課題

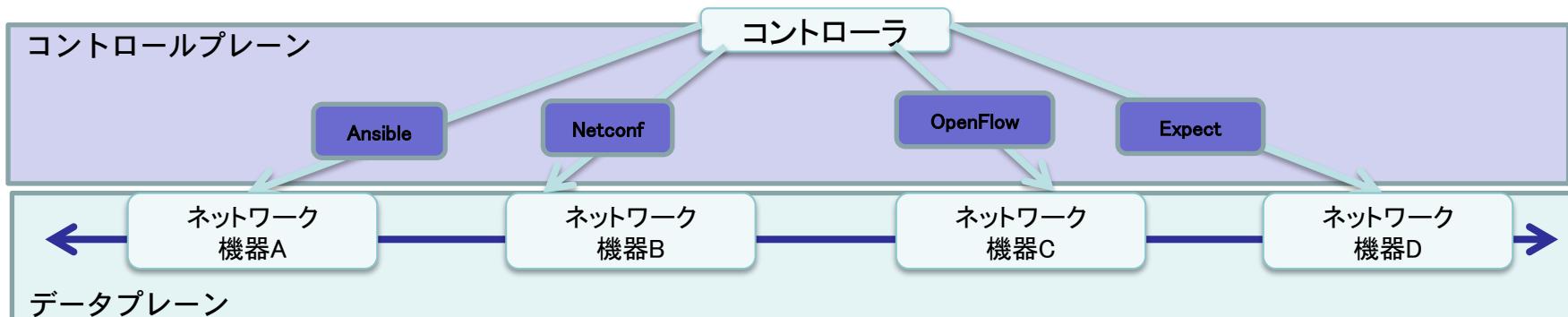
- 締め切り
 - 7/31(火) 23:59
 - lecture@sekiya-lab.info まで
- テーマは前々回授業（第10回）のスライドを見てください
 - 氏名 / 学生番号
 - 何を作ったのか
 - 作ったモノのソースコード
 - 動作を確認できるログもしくは画面ダンプ
 - 授業への感想、要望等

SDNって何？

- Software Defined Networking
 - ソフトウェアによるネットワーク機能の制御
 - 柔軟にトラフィックを制御
 - コントロールプレーンとデータプレーンの分離
 - ネットワーク機能の抽象化
 - 例
 - OpenFlow, NETCONF, BGP FlowSpec, PCEP

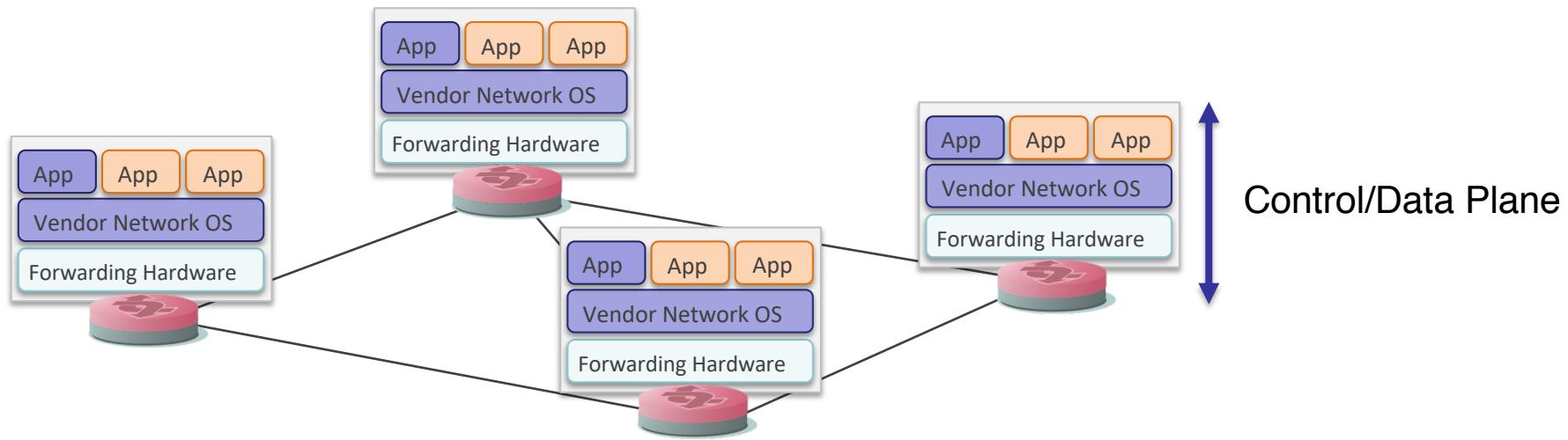
様々なSouthbound APIとSDN

- ・ 単純なconfig落としみは広義の意味でSDNという場合もあるが、、、
- ・ コントロールプレーンとデータプレーンの分離
 - ・ 経路計算、トポロジ把握、データプレーン状態の把握、物理統合仮想分離、、
- ・ それぞれに長所と短所があり使いどころがある
 - ・ 目的や環境、機器、スキルに合わせて最適なAPI、ソフトウェアを利用



従来のネットワーク

- ハードウェアとソフトウェアが各社により開発
 - ルータ、スイッチ、ファイアウォール、IDS/IPS...
- 自律分散
 - OSPF, RIP, BGP,...



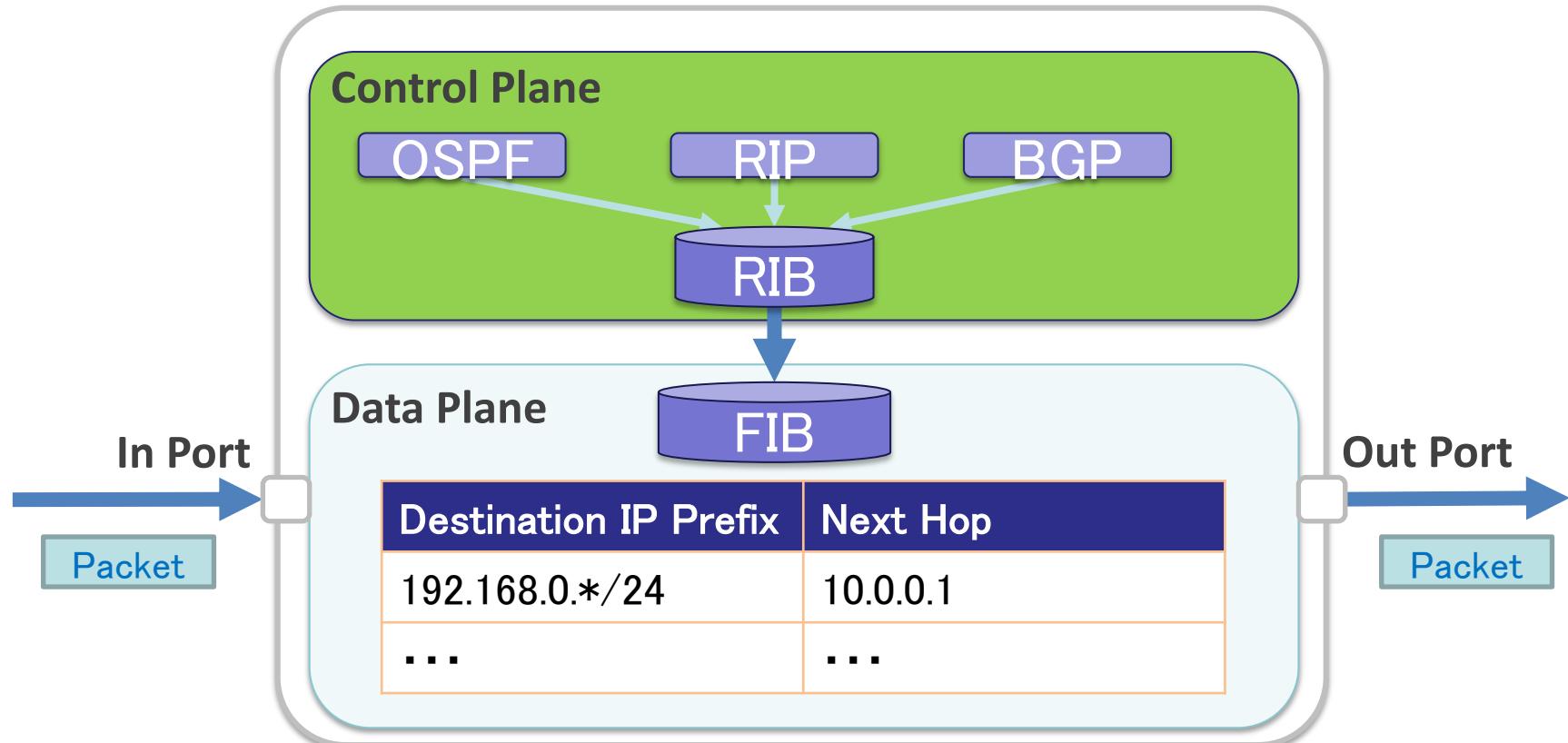
従来ネットワークの問題

- ベンダロックイン
 - HWとSWの一体化
 - 固有ベンダに依存した機能
- 限定された機能
 - ベンダ等の限られた開発者のみが実装
- ネットワークの大規模化と複雑化
 - 自律分散による個々機器設定管理
 - 複雑なネットワーク+仮想化

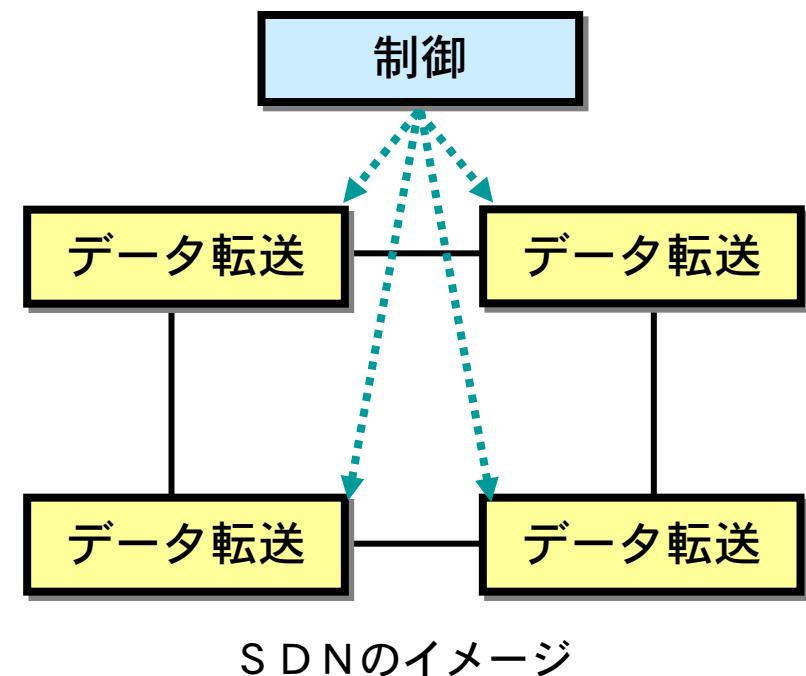
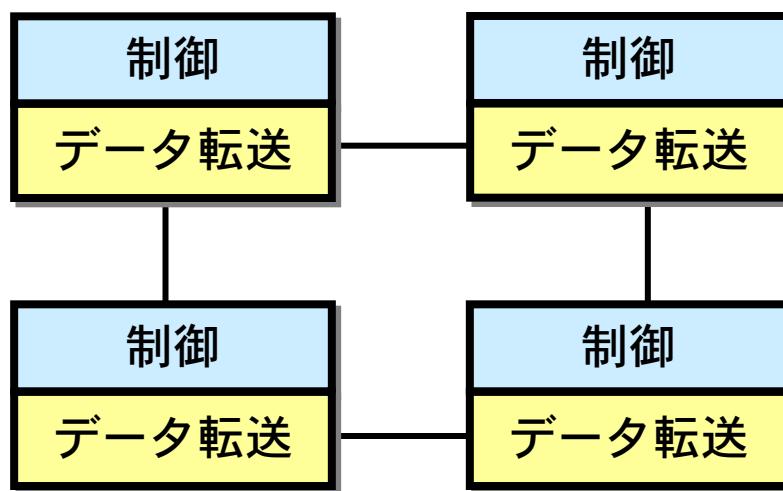
コントロールプレーン？ データプレーン？

- コントロールプレーン
 - 自律分散アルゴリズム
 - 経路計算
 - 他機器との制御情報交換(経路など)
- データプレーン
 - パケット転送制御(処理)
 - パケットの転送先決定

例えばルータの場合

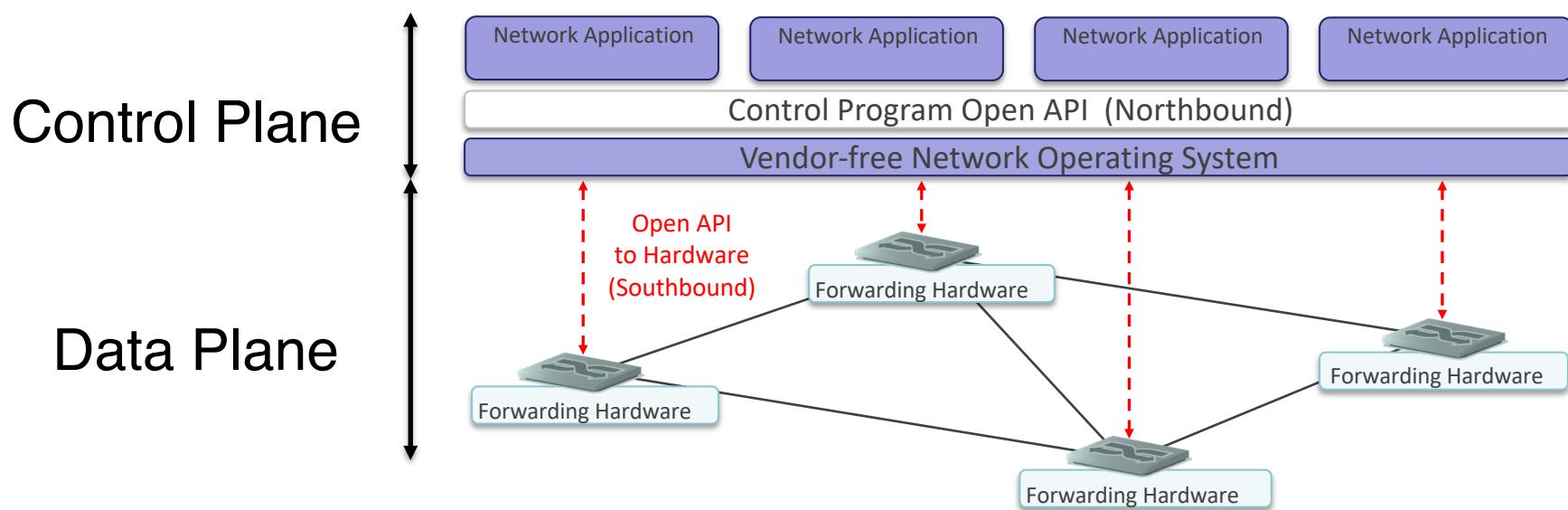


制御とデータ転送イメージ



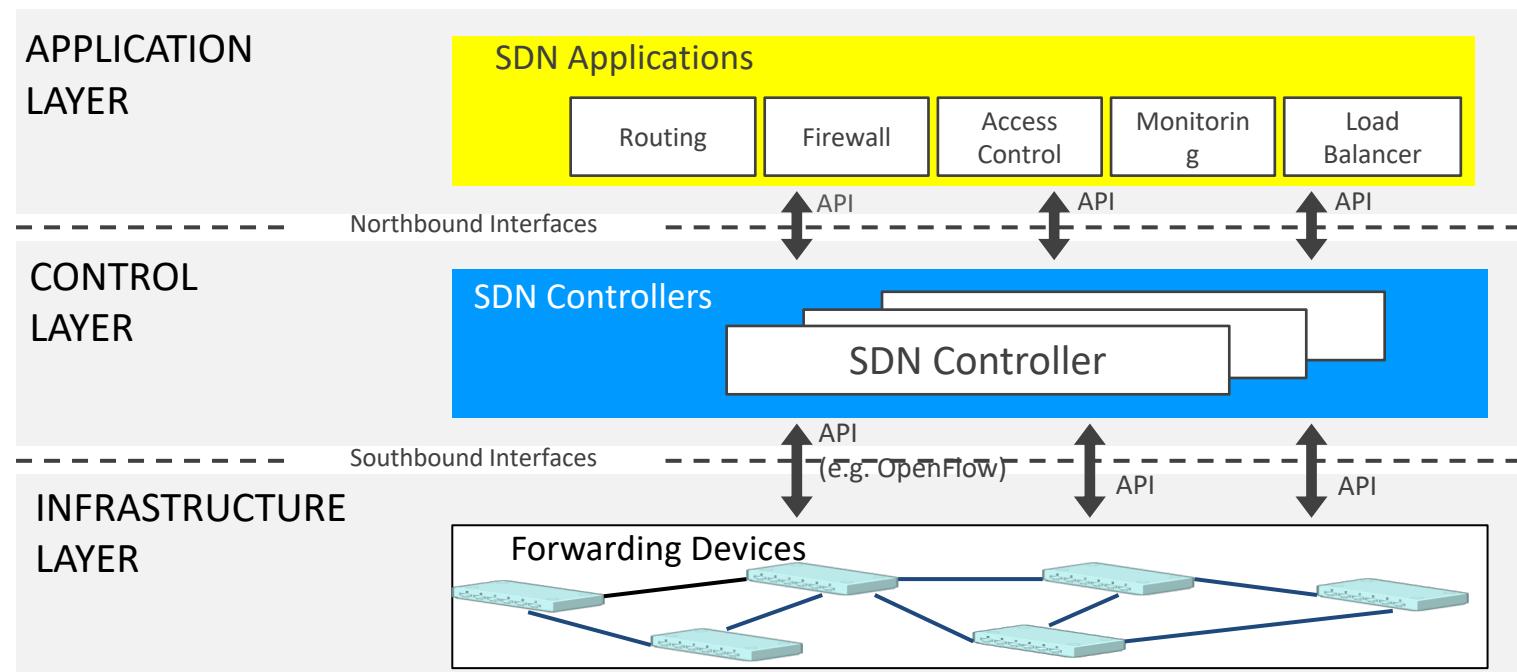
SDNに対応したネットワーク

- コントロールプレーンとデータプレーンを分離
 - 汎用的なネットワーク用OSを利用
 - APIを利用して新しいネットワーク機能を実装可能



SDNの階層構造

- 3層構造
 - アプリケーション、制御、基盤



Open Networking Foundation, <https://www.opennetworking.org>

SDNの用途

- 様々なアプリケーション
 - ロードバランサ
 - VPN網構築
 - トラフィックモニタ・filtrリング
 - 攻撃緩和装置へのトラフィック誘導
 - 仮想スライス/マルチスライス
 - エンドツーエンドでのトラフィック制御

SDNの利点

- ソフトウェア開発・実装の容易化
 - 共通化されたAPIによる機器制御
 - 新たな機能を迅速に実装し導入可能
- 機器の利用者自身が機能を実装可能
 - 自身の用途に合わせた機能の実装・変更が可能
- 人的ミスによる事故削減
 - 人はミスを繰り返す
 - コントローラはミス(バグ)が出て直せば同じミスは削減できる
- 作りこみよってネットワーク全体の最適化や自動化が可能になる

OPENFLOWとは？

OpenFlowとは？

- SDNの一つの形
 - ONFにより仕様が策定
- 簡素なプロトコル
 - Southbound API
 - Control PlaneとData Planeの間のインターフェイスを定義

※OpenFlowを通してSDNの根本的な考え方を理解できる。参考実装

| | |
|---------|-----------------------------------|
| 2008 | OpenFlowコンソーシアム立ち上げ |
| 2009/12 | OpenFlowプロトコル Version1.0 |
| 2011/2 | OpenFlowプロトコル Version1.1 |
| 2011/3 | ONF(Open Networking Foundation)設立 |
| 2011/12 | OpenFlowプロトコル Version1.2 |
| 2012/4 | OpenFlowプロトコル Version1.3 |



OPEN NETWORKING
FOUNDATION

Ethane: Taking Control of the Enterprise, Martín Casado et al., Sigcomm 2007..

OpenFlow: Enabling Innovation in Campus Networks, Nick McKeown et al., CCR 2008.

(参考)ONFの組織構成

SDNの標準化とプロモーションを目的に発足



各仕様の特徴

- 1.1
 - Action -> Instruction, Multi Table, Mask Expression (*),
MPLS Label, VLAN Label, TTL field..
- 1.2
 - IPv6, Multiple Controller..
- 1.3
 - Meter Table, PBB, Tunnel ID, IPv6 Extensible Header, ...
- 1.4
 - Optical Port, Change Controller Port,...

| Version | Date | Header Fields |
|---------|----------|--|
| OF 1.0 | Dec 2009 | 12 fields (Ethernet, TCP/IPv4) |
| OF 1.1 | Feb 2011 | 15 fields (MPLS, inter-table metadata) |
| OF 1.2 | Dec 2011 | 36 fields (ARP, ICMP, IPv6, etc.) |
| OF 1.3 | Jun 2012 | 40 fields |
| OF 1.4 | Oct 2013 | 41 fields |

From “P4: Programming Protocol-Independent Packet Processors”,
P. Bosshart et.al, ACM SIGCOMM CCR, July 2014

Ver1.1 の主な追加機能

| 追加機能 | 説明 |
|---------------|---------------------------|
| MPLS/VLANタグ対応 | MPLSおよびVLANタグの追加・修正・削除 |
| マスク表記対応 | MACアドレスおよびIPアドレスのマスク表記 |
| 複数テーブル対応 | OFSで複数のテーブルをサポート |
| グループ化対応 | OFSの複数のポートをグループとして処理可能とする |
| TTL処理対応 | MPLS、IPヘッダのTTL変更・複製・減算 |
| SCTP対応 | SCTPヘッダによるマッチング、書き換え処理 |

Ver1.2 の主な追加機能

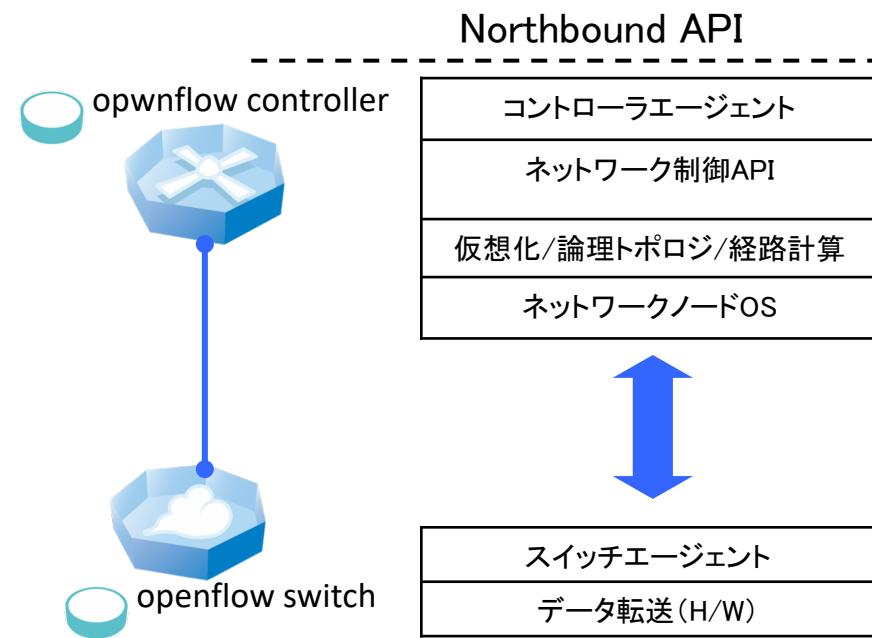
| 追加機能 | 説明 |
|-------------|-------------------------|
| IPv6対応 | IPv6をサポート |
| マッチングルールの拡張 | 新たなプロトコルへの柔軟な対応を目的として拡張 |

Ver1.3 の主な追加機能

| 追加機能 | 説明 |
|-----------------|------------------|
| トンネリングと論理ポート抽象化 | VPN等への利用を測定 |
| PBB対応 | データセンタ間の相互接続を想定 |
| フロー毎のQoS測定 | フロー毎にQoSを測定する |
| フロー毎のトラフィック測定 | フロー毎にトラフィックを測定する |

OpenFlowの構成

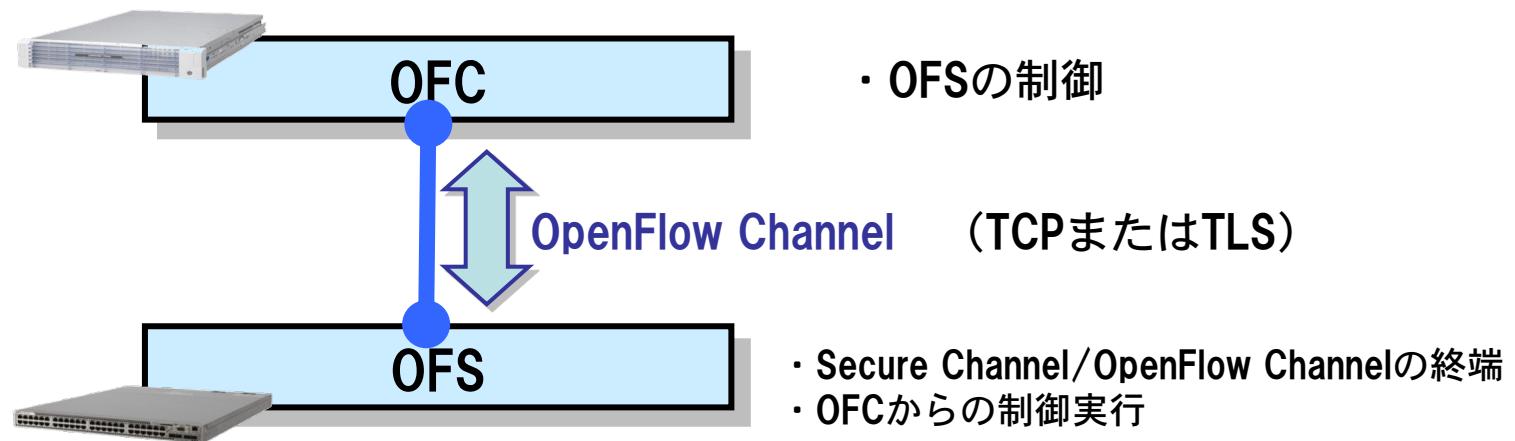
- OpenFlowコントローラ (OFC)
 - コントロールプレーン
- OpenFlowスイッチ (OFS)
 - データプレーン
- プロトコルで定めている主な仕様
 - OFCによるOFSの制御
 - OFC/OFSが参照するパケットヘッダ
 - OFSのフローテーブルの構成
 - フローに対する処理
 - OFC/OFS間のメッセージフォーマット



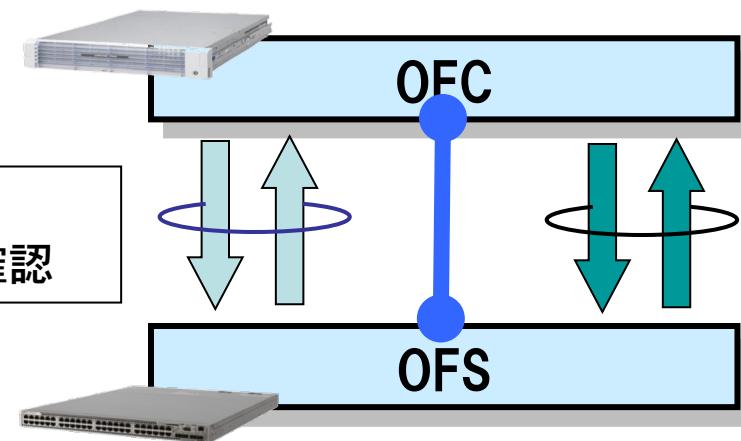
OpenFlowの基本動作

- OFCとOFSの接続(コントロールプレーン)
 - バージョン、機能確認
 - Secure Channel/OpenFlow Channel確立
 - トポロジ把握
 - Flow設定
- “FLOW”に対して”ACTION”を適応
 - FLOW
 - 複数の識別子により定義される通信の塊
 - ACTION
 - packet-in : コントローラへパケットを転送
 - output : 指定ポートからパケットを出力
 - modify : パケットを書き換え (e.g. vlan id, mac addr, ip addr)
 - drop : パケットを破棄

基本動作(コントロールプレーン)



基本的な通信：バージョンと機能のネゴシエーション



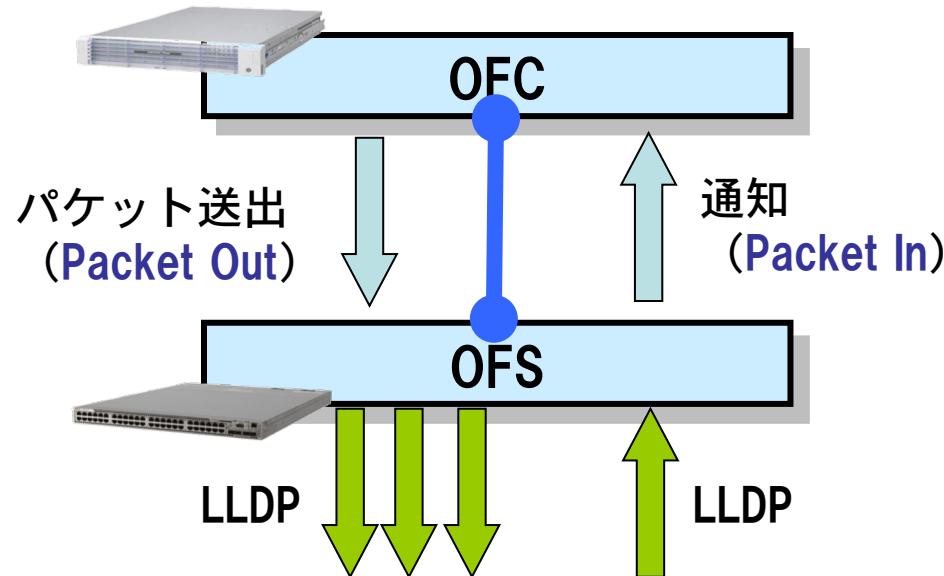
- **Hello**

バージョンの確認

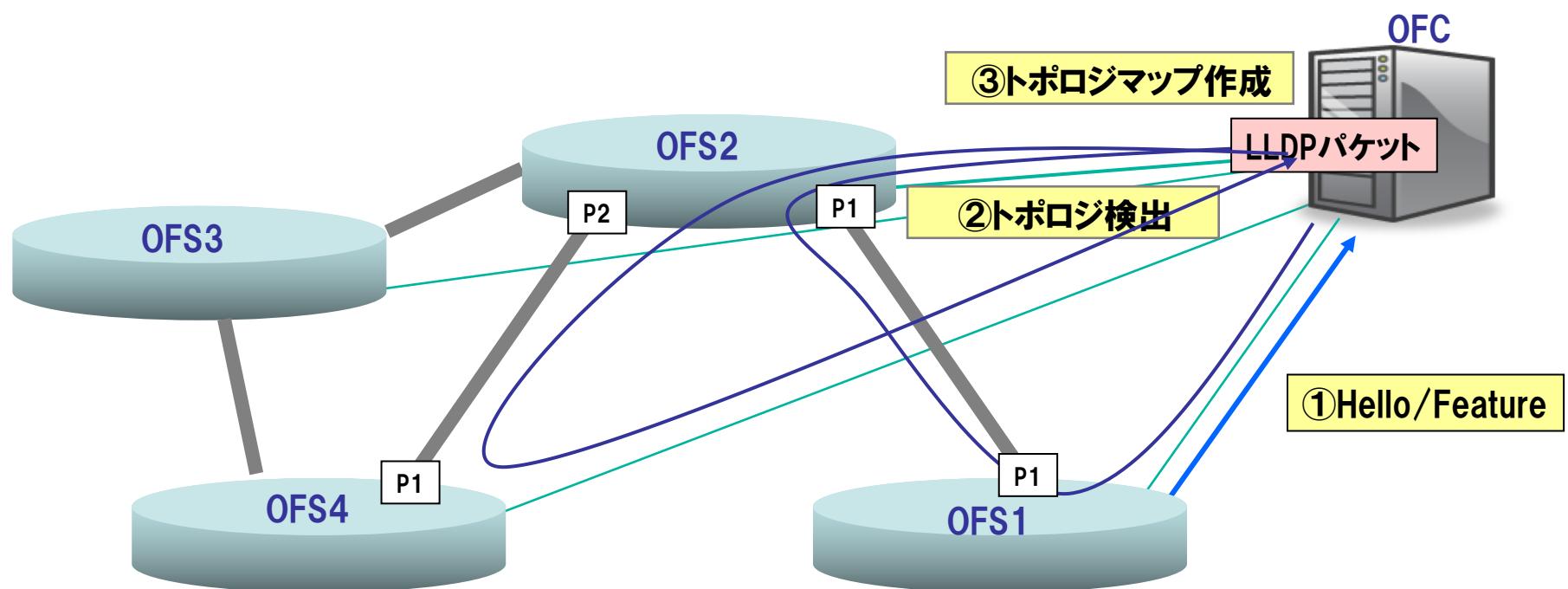
- **Feature**

スイッチ ID、物理ポート、
サポート機能等の確認

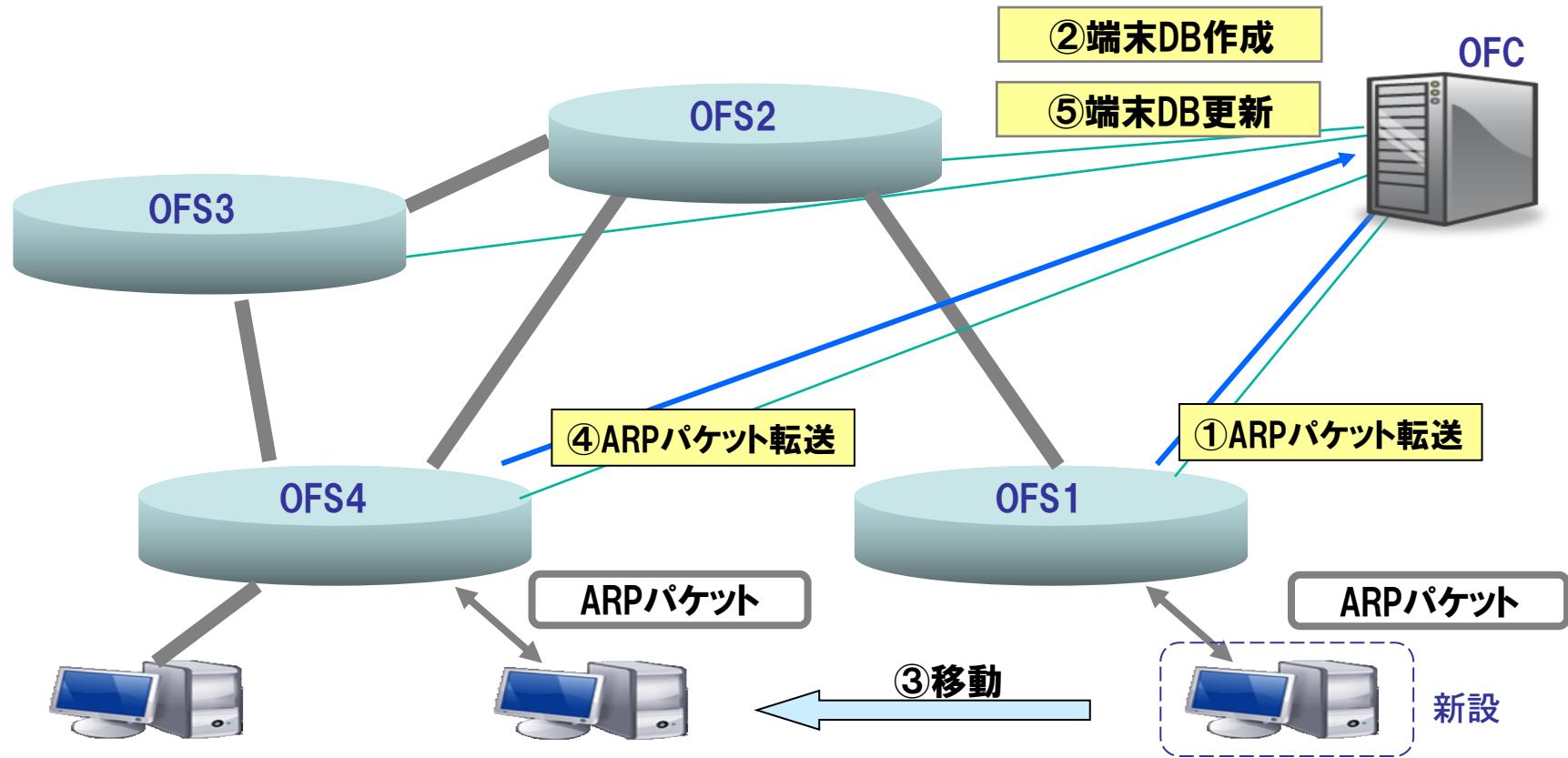
基本的な通信(トポロジマップの作成)



コントローラがトポジマップを作成(OFS部分)

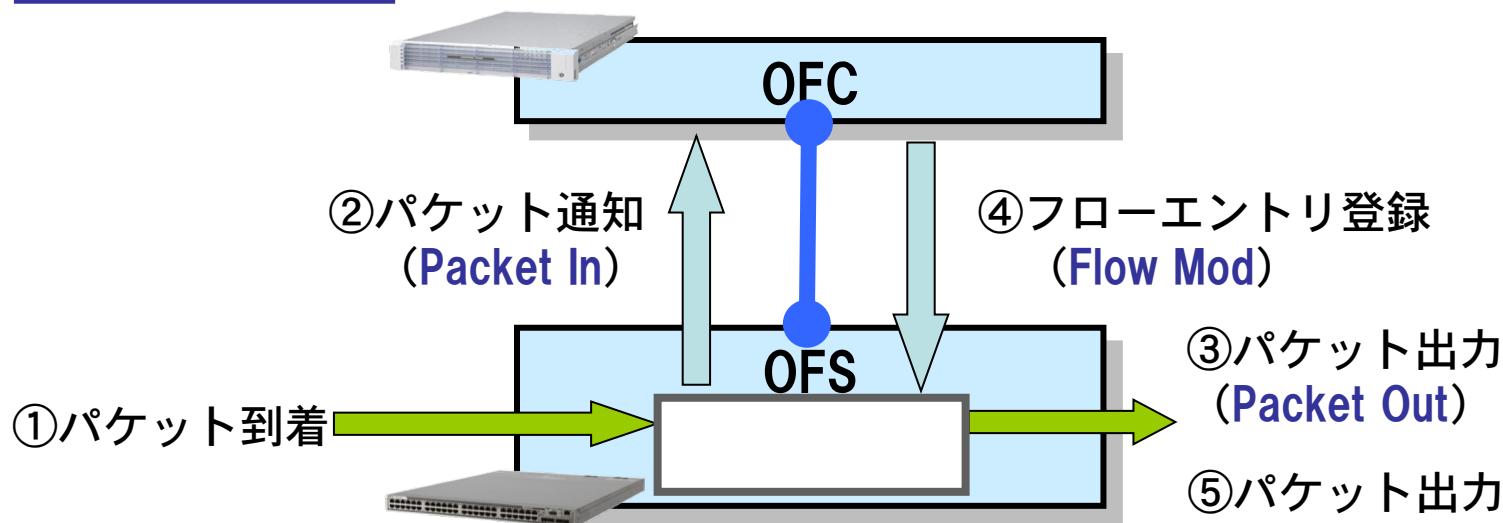


コントローラがノードを登録



データ転送(リアクティブ型制御)

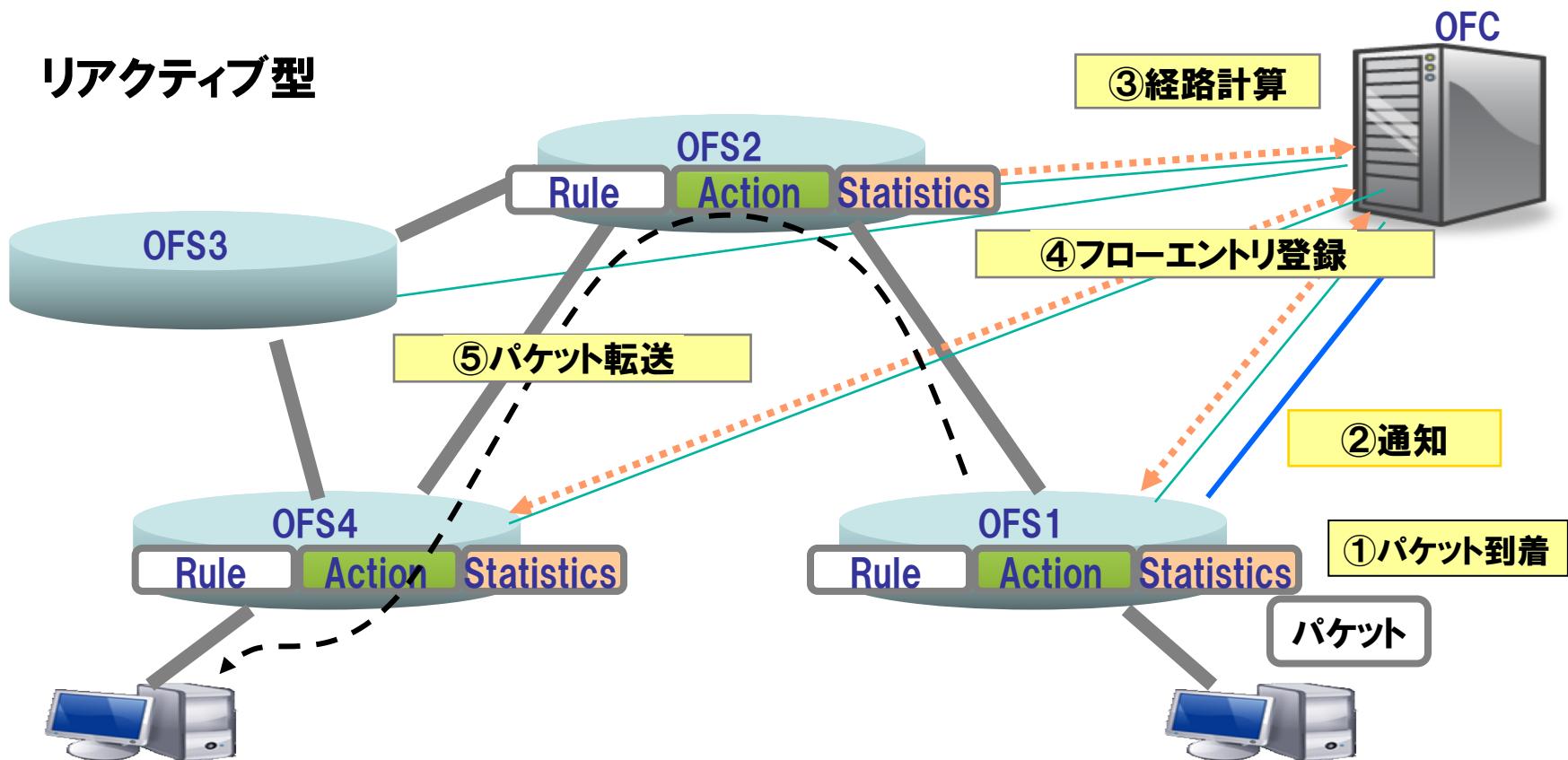
Reactive型



新規にフローを検出し対応するフローエントリを登録

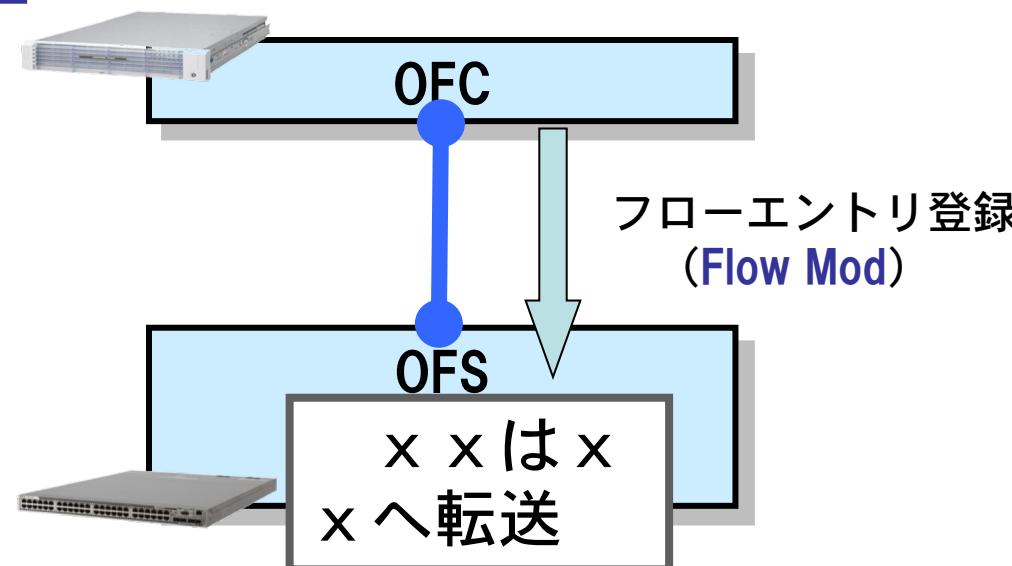
コントローラがスイッチへフローテーブルを設定するまで

リアクティブ型



データ転送(プロアクティブ型制御)

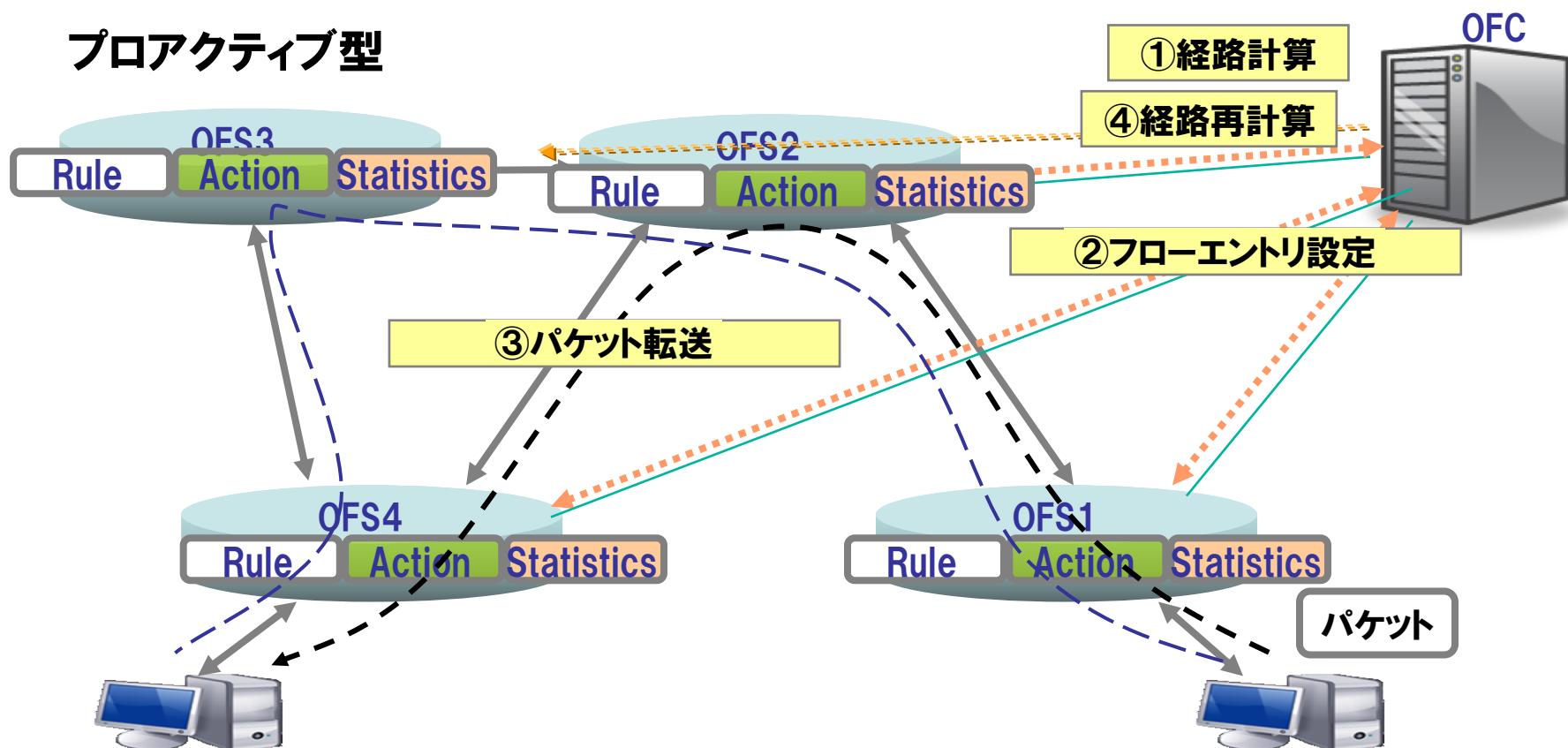
Proactive型



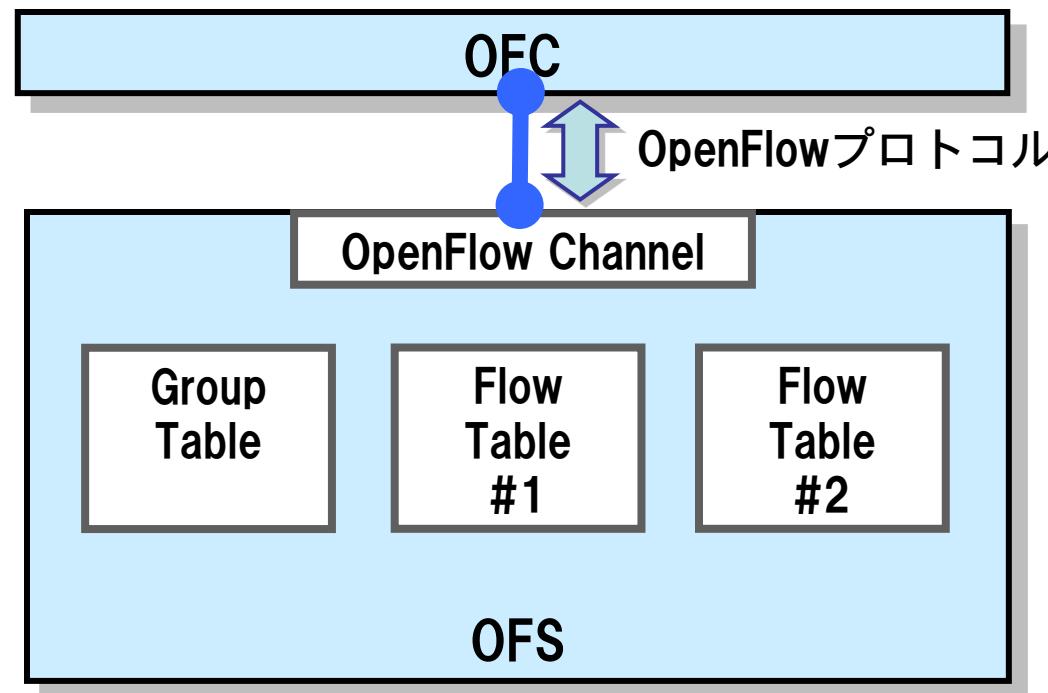
あらかじめOFCがフローエントリ登録

コントローラがスイッチへフローテーブルを設定するまで

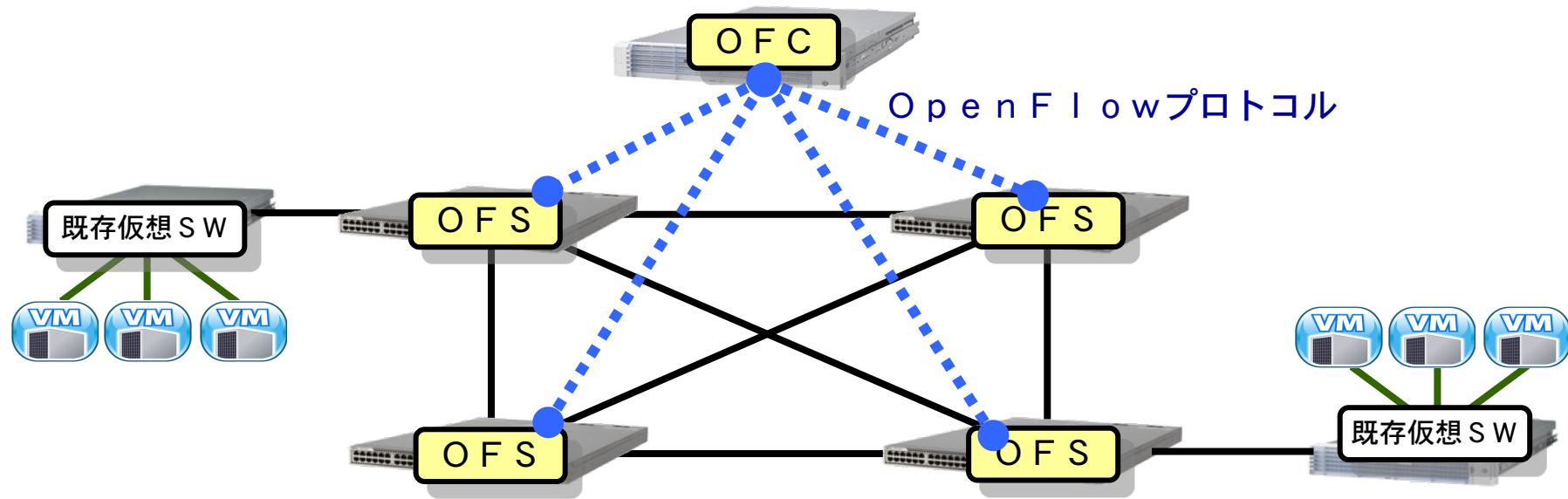
プロアクティブ型



OpenFlowスイッチの構造 (cont'd)

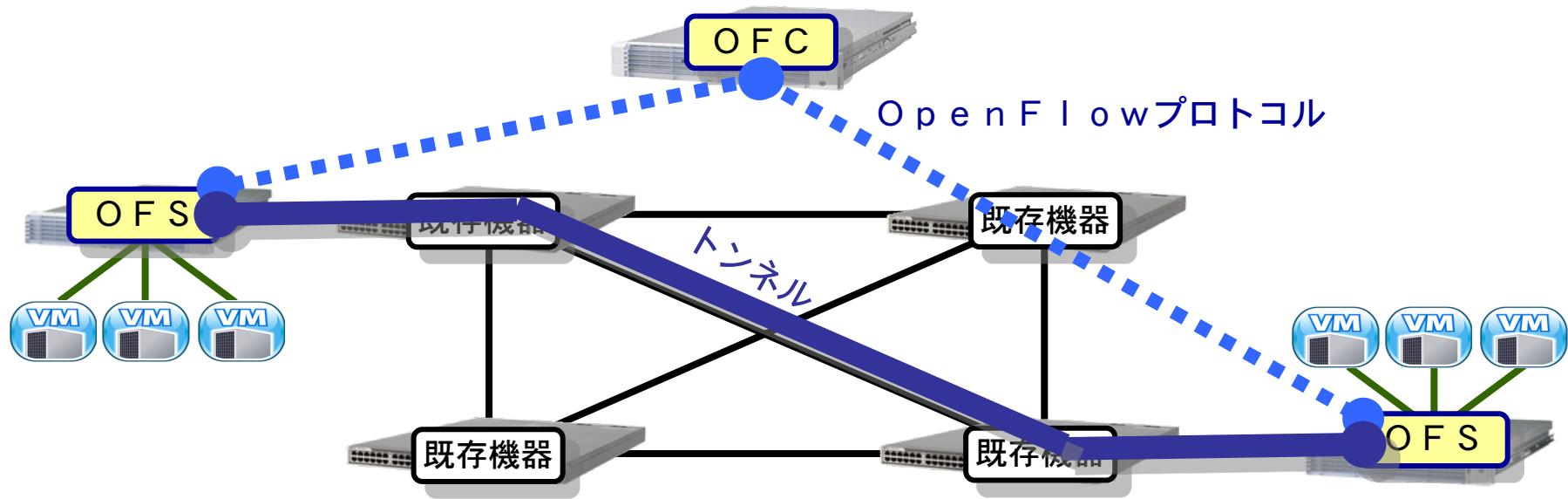


ホップバイホップモデル



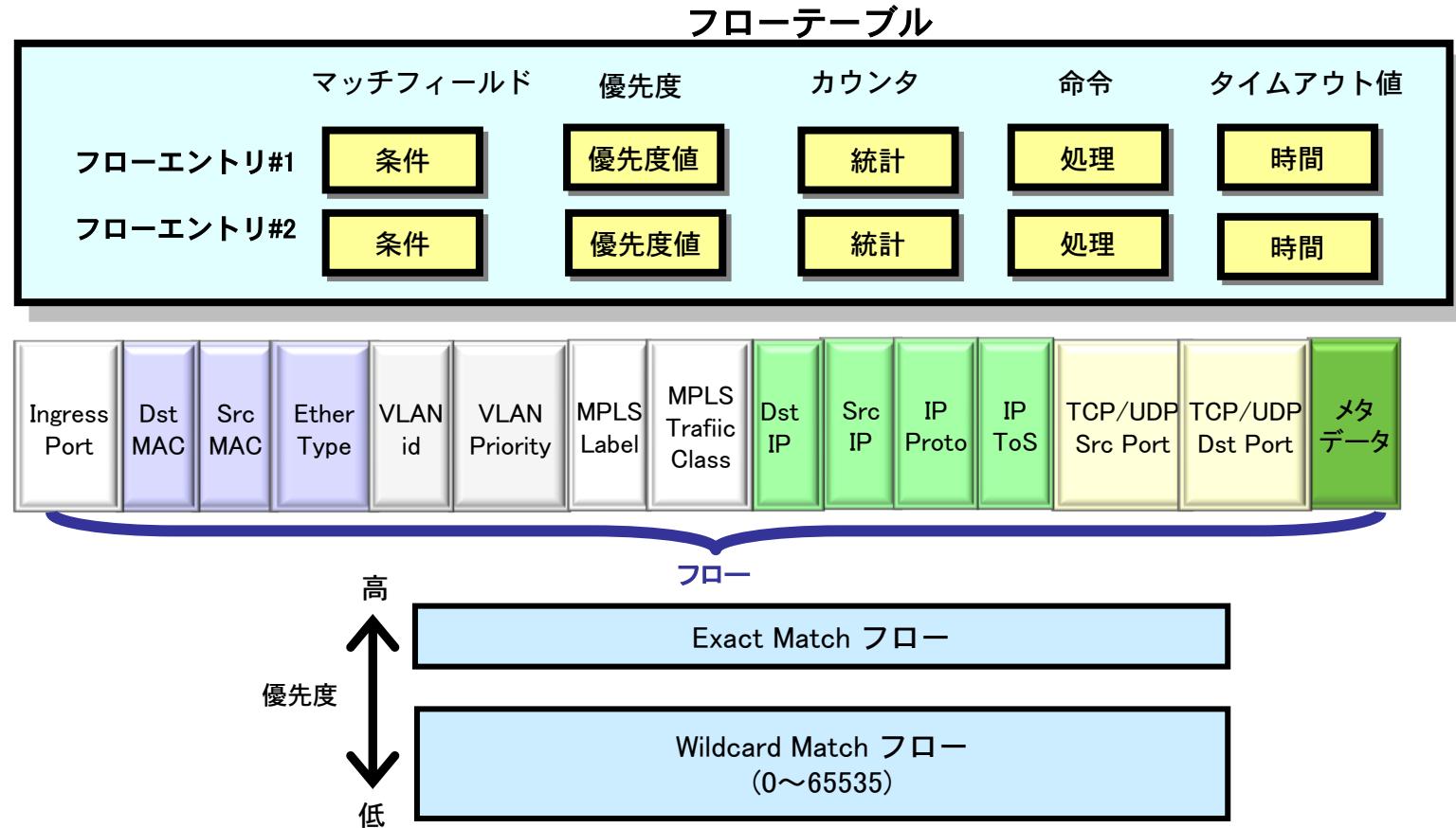
- OpenFlowスイッチを用いてネットワークを構成
- 各スイッチのフローテーブルをコントローラが制御し通信を実現

オーバレイモデル



- OpenFlowスイッチを端点にのみ設置
- トンネルを用いてOFS間を結んでトラフィックを通す(オーバレイ)

フローの概念



フローの定義 (cont 'd)

- 識別子の組み合わせによる通信の塊

<src ip, dst ip>

S

tcp, udp, src port*, dst port*

D



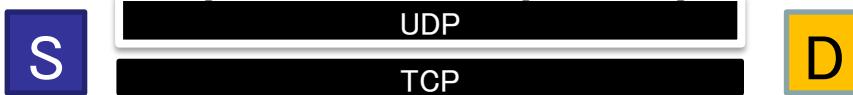
フローの定義 (cont 'd)

- 識別子の組み合わせによる通信の塊

<src ip, dst ip>



<protocol, src ip, dst ip>



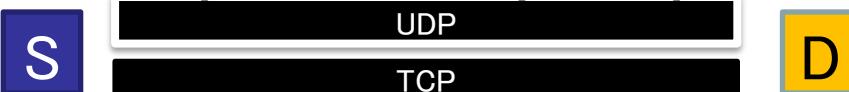
フローの定義 (cont 'd)

- 識別子の組み合わせによる通信の塊

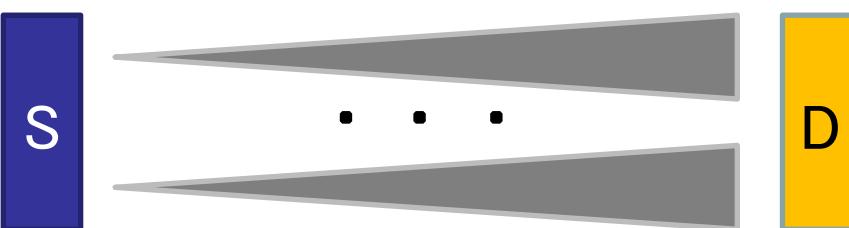
<src ip, dst ip>



<protocol, src ip, dst ip>



<protocol, src port, src ip, dst ip>



フローの定義 (cont 'd)

- 識別子の組み合わせによる通信の塊

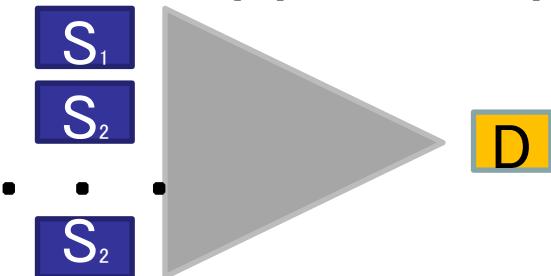
<src ip, dst ip>



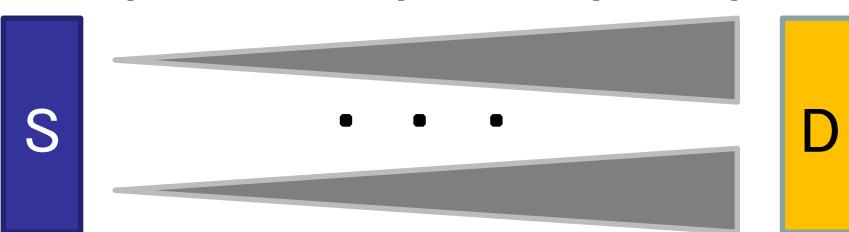
<protocol, src ip, dst ip>



<src ip prefix, dst ip>



<protocol, src port, src ip, dst ip>



フローの定義 (cont 'd)

- 識別子の組み合わせによる通信の塊

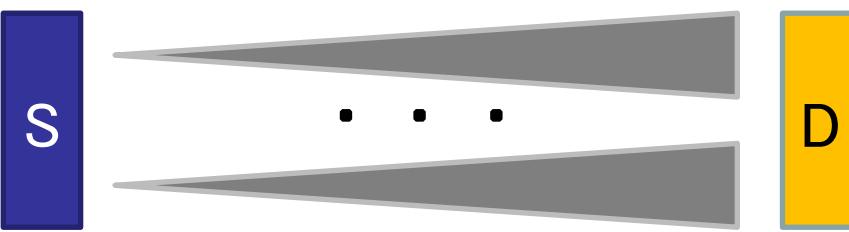
<src ip, dst ip>



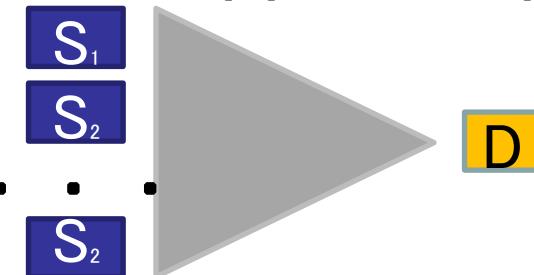
<protocol, src ip, dst ip>



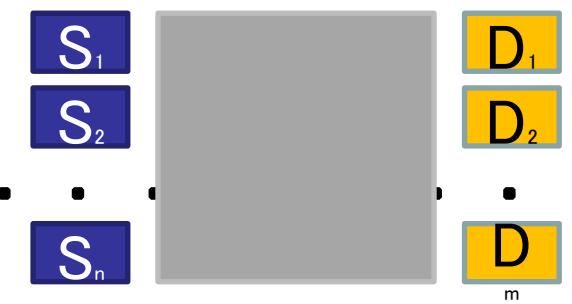
<protocol, src port, src ip, dst ip>



<src ip prefix, dst ip>



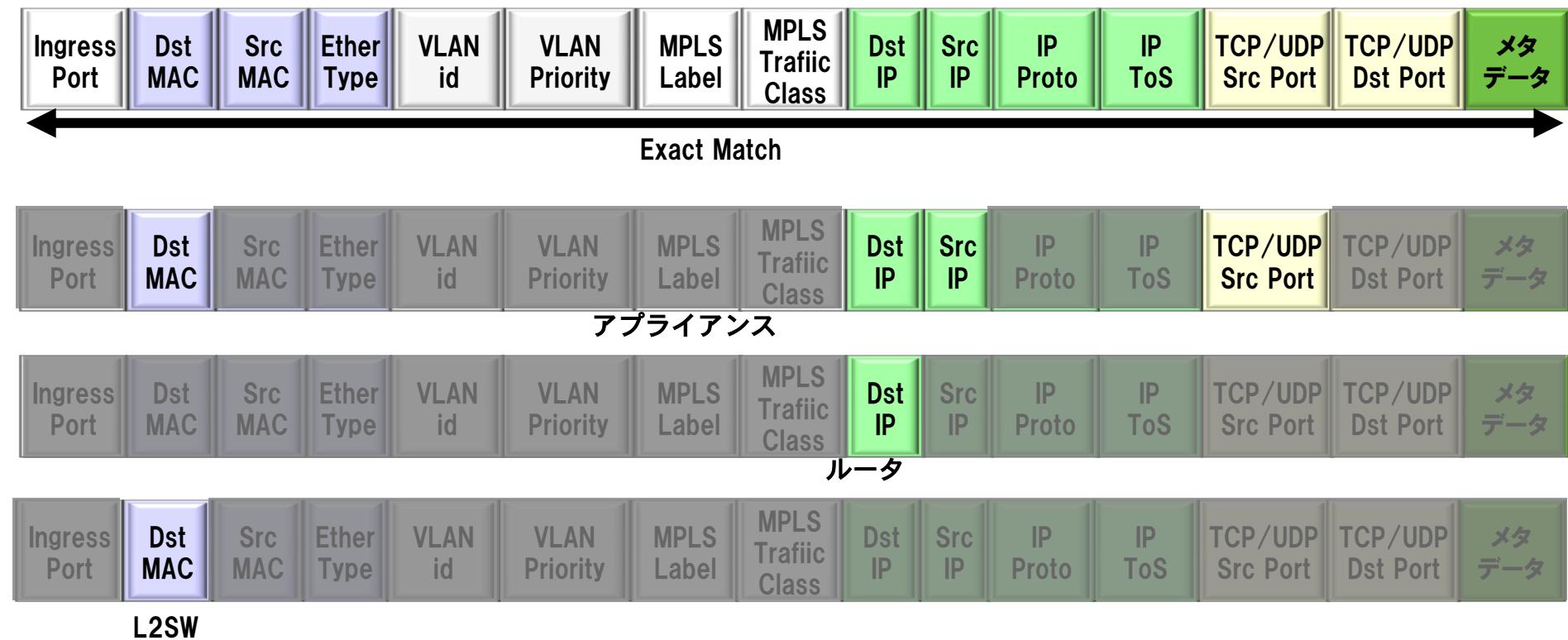
<src ip prefix, dst ip prefix>



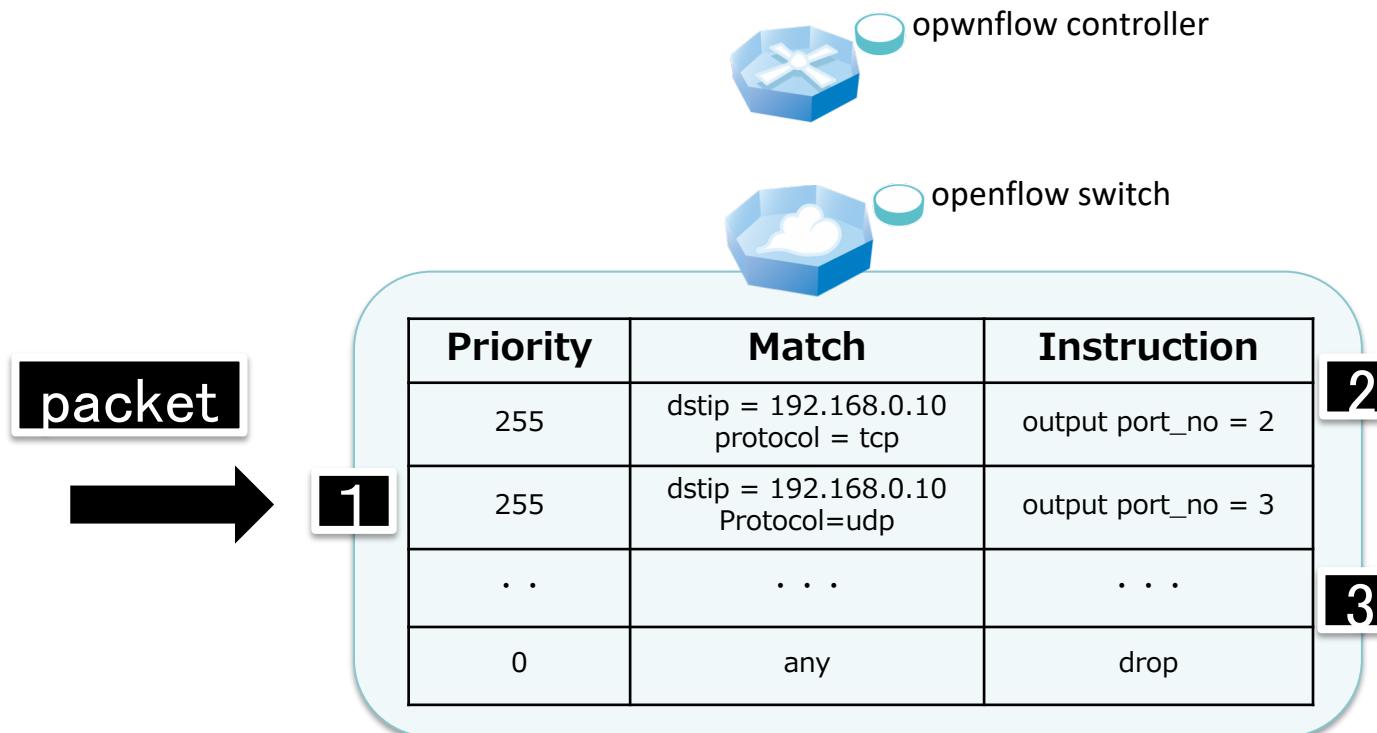
フローの定義

- 5-tuple (ファイブ タプル)
 - 既存ネットワークでもよく使われるフロー
 - プロトコル、送信元IPアドレス、送信元ポート番号、宛先IPアドレス、宛先ポート番号
 - 例: <TCP, 192.168.1.10, 11111, 10.0.1.2, 80>
- OpenFlowではより多くの識別子によりFLOWを定義可能
 - 送信元・宛先MACアドレス、VLAN ID、Ethernet番号、MPLS ラベル....
 - 利用可能な識別子はOFのバージョンに依存

フローの定義



OpenFlowの動作



FlowTableの構成(cont'd)

- Match Field
 - “FLOW”の条件
- Priority
 - Flowエントリ の優先度
- Counters
 - エントリに一致したパケットの数・数量
- Instructions
 - マッチしたパケットに対する処理

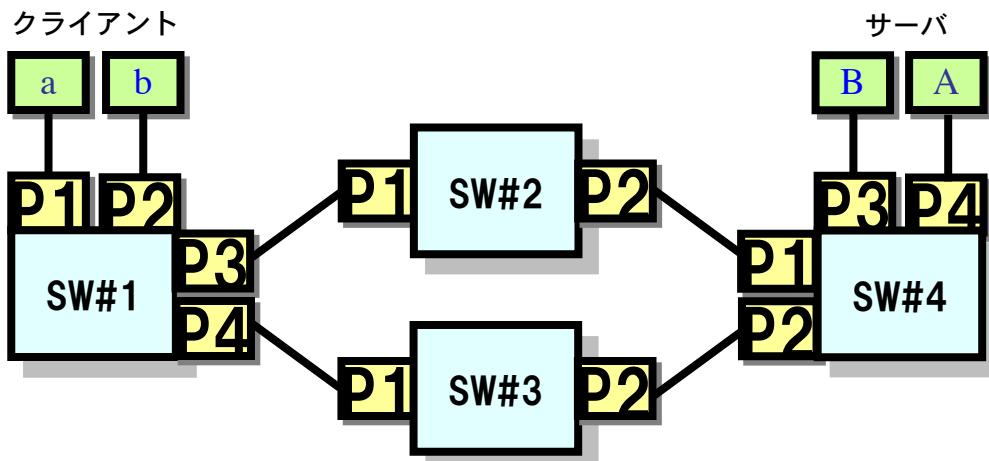
FlowTableの構成

- Timeout
 - Flowエントリのタイマー値
- Cookie
 - Flowエントリの識別子
- Flags
 - 制御用フラグ

OpenFlowネットワークの例1 (cont'd)

<要件>

- ・ クライアント **a** はサーバ **A** とのみ通信する
- ・ クライアント **b** はサーバ **B** とのみ通信する



各SWのフローテーブル

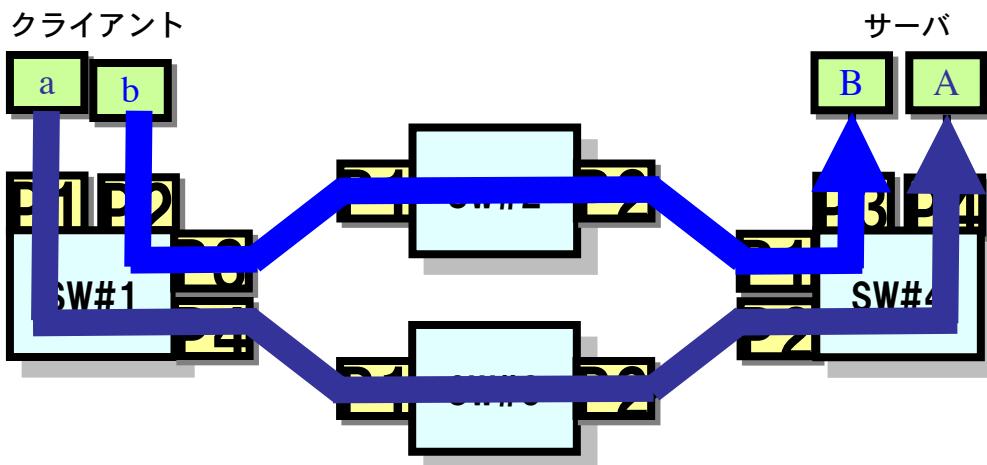
| | マッチフィールド | 命令 |
|------|----------|----------|
| SW#1 | 発信元MAC=a | Port4に出力 |
| | 発信元MAC=b | Port3に出力 |
| SW#2 | マッチフィールド | 命令 |
| | 発信元MAC=b | Port2に出力 |
| SW#3 | マッチフィールド | 命令 |
| | 発信元MAC=a | Port2に出力 |
| SW#4 | マッチフィールド | 命令 |
| | 発信元MAC=a | Port4に出力 |
| | 発信元MAC=b | Port3に出力 |

※クライアント、サーバのa,b,A,BはそれぞれMACアドレスをあらわす

OpenFlowネットワークの例1

<通信経路>

クライアント a ⇒ SW#1 ⇒ SW#3 ⇒ SW#4 ⇒ サーバ A
 クライアント b ⇒ SW#1 ⇒ SW#2 ⇒ SW#4 ⇒ サーバ B



各SWのフローテーブル

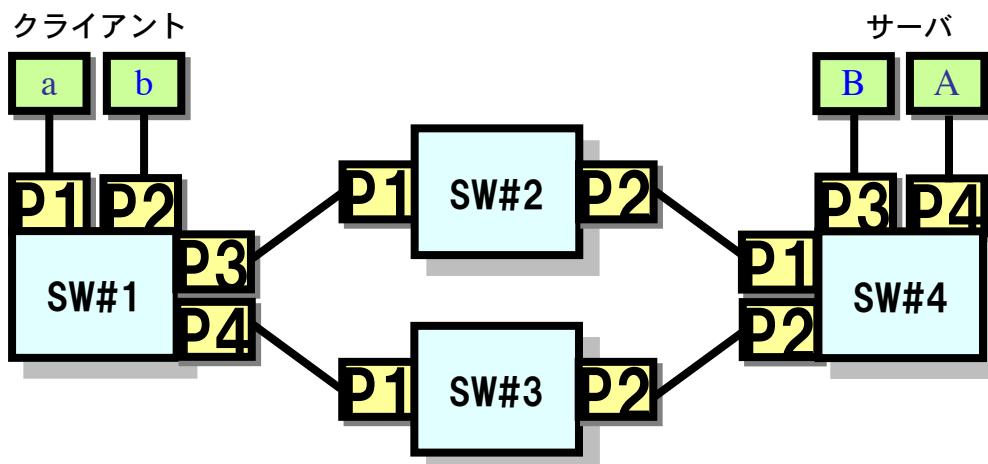
| | マッチフィールド | 命令 |
|------|----------|----------|
| SW#1 | 発信元MAC=a | Port4に出力 |
| | 発信元MAC=b | Port3に出力 |
| SW#2 | マッチフィールド | 命令 |
| | 発信元MAC=b | Port2に出力 |
| SW#3 | マッチフィールド | 命令 |
| | 発信元MAC=a | Port2に出力 |
| SW#4 | マッチフィールド | 命令 |
| | 発信元MAC=a | Port4に出力 |
| | 発信元MAC=b | Port3に出力 |

※クライアント、サーバのa,b,A,BはそれぞれMACアドレスをあらわす

OpenFlowネットワークの例2 (cont'd)

〈要件〉

- ・クライアント **a** はサーバ A とのみ通信する
- ・クライアント **b** はサーバ B とのみ通信する
- ・SW#1から送出するパケットをSW#2とSW#3に負荷分散



SW#1 フローテーブル

| マッチフィールド | 命令 |
|----------|-------|
| 発信元MAC=a | グループ1 |
| 発信元MAC=b | グループ1 |

グループテーブル

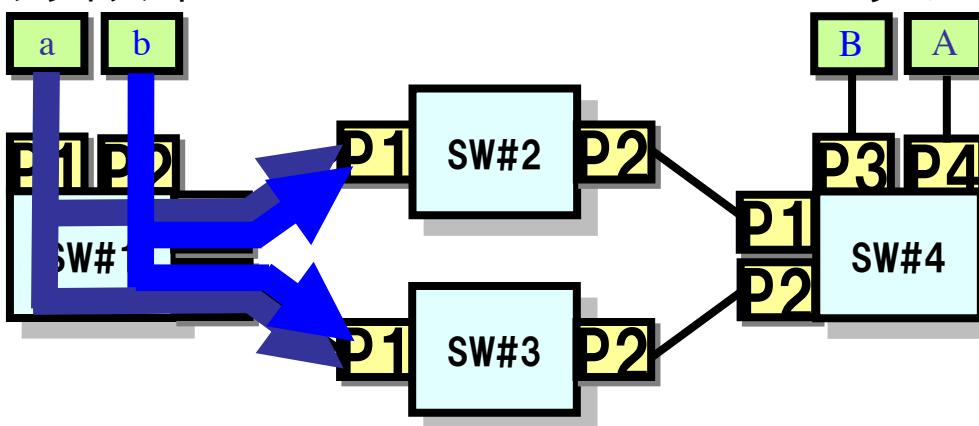
| グループID | タイプ | アクション |
|--------|--------|------------|
| 1 | select | Port3、4に出力 |

OpenFlowネットワークの例2

〈要件〉

- ・ クライアント a はサーバ A とのみ通信する
- ・ クライアント b はサーバ B とのみ通信する
- ・ SW#1 から送出するパケットを SW#2 と SW#3 に負荷分散

クライアント



サーバ

SW#1

| フローテーブル | |
|----------|-------|
| マッチフィールド | 命令 |
| 発信元MAC=a | グループ1 |
| 発信元MAC=b | グループ1 |

| グループテーブル | | |
|----------|--------|------------|
| グループID | タイプ | アクション |
| 1 | select | Port3、4に出力 |

OpenFlowスイッチ

- スイッチ・ルータベンダにて実装

| Vendor | Switch | Spec |
|-----------------------|----------------------------|----------|
| NEC | PF Series | 1.1, 1.3 |
| PICA8 | P Series | 1.* |
| Extreme (Brocade) | VDX, ICX, CES, MLX, CER... | 1.3 |
| IBM | RackSwitch Series | 1.0, 1.3 |
| Alcatel Lucent | OmniSwitch Series | 1.0, 1.3 |
| Open vSwitch | Open vSwtich | 1.* |
| NTT R&D (open source) | Lagopus switch | 1.3 |

OpenFlowコントローラ

- ベンダやオープンソースで幅広く開発
 - 様々な言語で実装
 - ただし、対応状況やスイッチとの相性はコントローラにより様々…
自身の環境に合わせて確認が必要

| Name | Language | Misc |
|--------------|----------|------------------|
| Trema | Ruby/C | NEC |
| Trema Edge | C | NEC |
| Ryu | Python | NTT R&D |
| POX | Python | VMware |
| Floodlight | Java | Big Switch |
| OpenDaylight | Java | Linux Foundation |

OpenFlowの長所・短所

- **長所**
 - ベンダ・キャリアによる統一された仕様
 - オープンソースでの実装が豊富
 - 見える化が容易
 - きめ細かい経路制御が可能
- **短所**
 - 必ずしもONFの仕様に応じた機能を実装していない
 - 汎用スイッチをOFS化している場合、テーブル、match条件など不足している場合もある

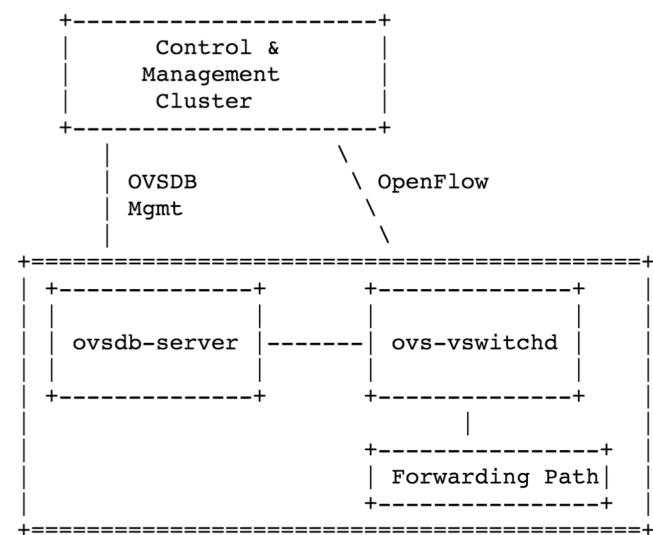
様々なSDN

BGP Flowspec

- BGPを用いてフローに対するアクション情報を伝播するプロトコル
 - RFC 5575
 - アクションの種類
 - サンプリング/VRFへの転送/帯域制限/マーキング (DSCP)
 - 例えばACLの設定ができる(帯域制限)
 - 192.168.0.0/24宛のパケットは破棄など
 - DoS/DDoSのミティゲーションにも使われる
- 実装
 - ルータ: Juniper JUNOS, Cisco ASR, Alcatel-Lucent SR OS
 - ソフトウェア: GoBGP, ExaBGP

OVSDB

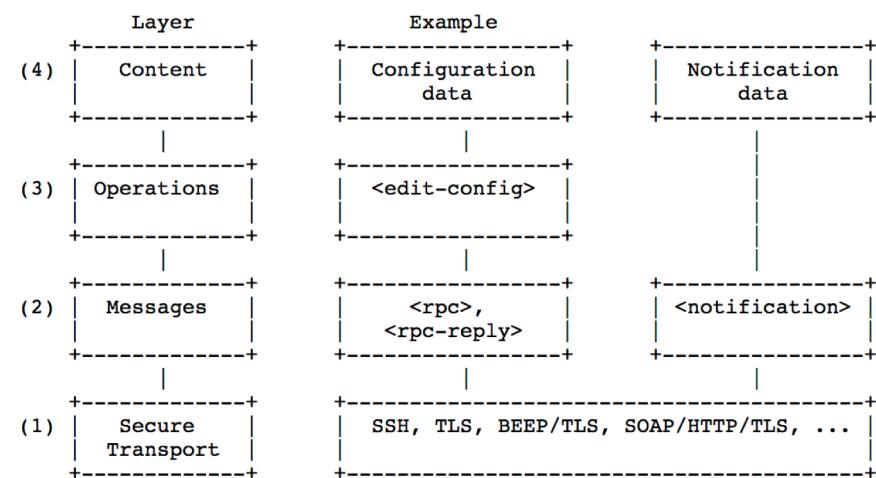
- Open vSwitchの設定と管理を行うためのプロトコル
 - RFC7404
 - OpenFlowルールの設定/キューの設定/QoSの設定など
 - JSON形式でやり取りが行われる
 - Open vSwitch databaseを制御する仕組み



<https://tools.ietf.org/html/rfc7047>

NETCONF (cont'd)

- ネットワーク機器の設定を統一されたフォーマットで管理・更新するためのプロトコル
 - RFC6241で定義
 - 同じネットワークの設定であってもベンダによって設定(コンフィグ)の書き方は様々
 - 多量の機器の設定変更・管理の効率化



4層のプロトコル

NETCONF

- データモデルとしてYANGが定義されている (XMLフォーマット)
RFC6020
 - プロトコル毎にデータモデルが提案されている
 - 例: OSPF → draft-ietf-ospf-yang-09
- NETCONFを利用するOS例
 - Juniper JUNOS, CISCO IOS-XEなど

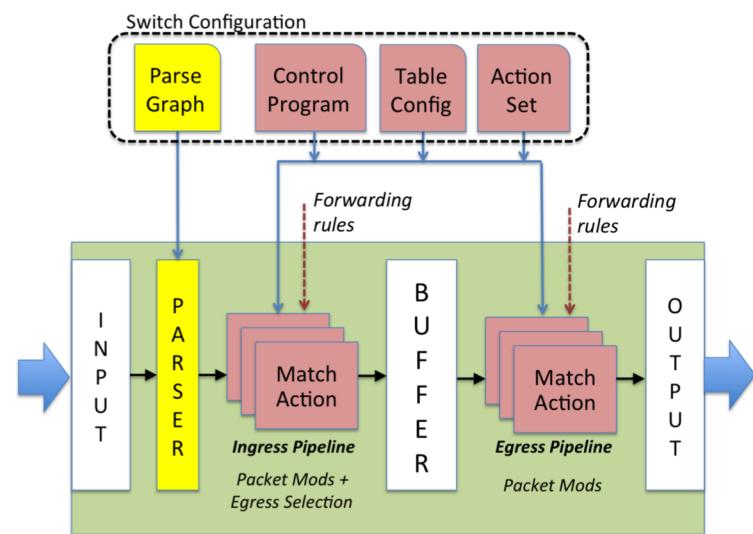
```
+--rw instance* [af]
  +--rw areas
    +--rw area* [area-id]
      +--rw area-id
      +--rw area-type?
      +--rw summary?
      +--rw default-cost?
      +--rw ranges
        +--rw range* [prefix]
          +--rw prefix      inet:ip-prefix
          +--rw advertise? boolean
          +--rw cost?       uint24
            . .
            . .
            . .
```

area-id-type
identityref
boolean
uint32
inet:ip-prefix
boolean
uint24

OSPF YANGのフィールド例

P4 (cont'd)

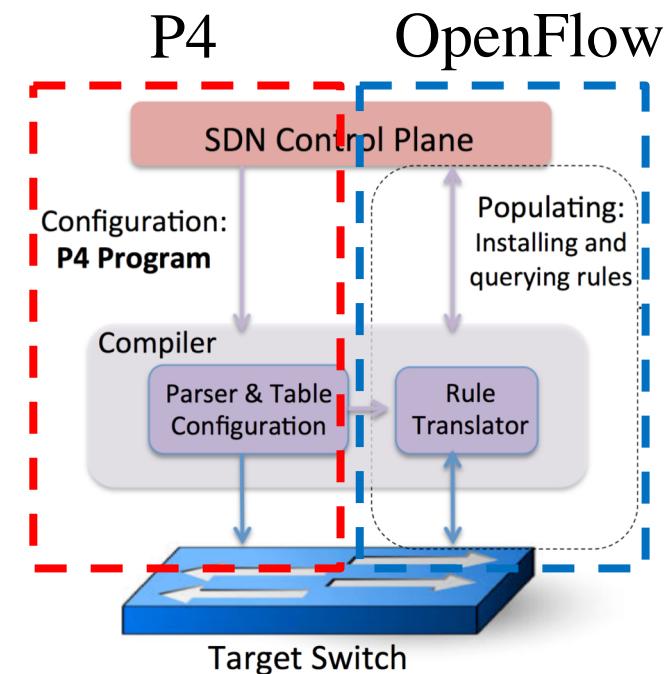
- P4 is a programming language designed to allow programming of packet forwarding planes.
 - データプレーンを設計するためのプログラミング言語
 - プログラマブル・データプレーン
 - P4 Language Consortiumにて言語仕様などが策定されている



P4 Official Site <https://p4.org/>

P4 (cont'd)

- 以前のSDNとは何が違うのか？
 - OpenFlowはデータプレーンが固定
 - マッチに利用できるフィールドはOpenFlowの仕様で定義され実装されているもののみ
 - 既存のプロトコルで利用されているフィールドのみ
 - 例：ペイロードの5ビット目から10ビット目がXXXだったらドロップ/書き換えるといったことが可能になる
 - 論理演算やビットシフトも可能
 - スイッチを経由するだけでデータの集約などができるかも



P4

- ハードウェア
 - Barefoot Networks Tofino
 - P4以外のプログラマブル・データプレーン
 - Cavium Xpliant, Netronome NFP, NetFPGA (Xilinx)
- ユースケース
 - データセンタスイッチ, キヤリア向けスイッチ, L4ロードバランサ, ファイアウォール, DDoS防御, パケットブローカーなどなど
 - プログラムをどう作るかによって様々な機能を実装可能

SDN利用時のポイントと注意点

- 適用領域によって適材適所がある
 - 全体管理、物理統合仮想分離、細かい制御
 - バックボーンルーティング？？？
- 実装
 - 商用レベルで考えると考慮しないといけないポイント
 - 冗長、コントローラ切断時動作、リソース管理、スケール、経路投入順序、デバッグ機能、BUMの取り扱い、経路制御計算、、

まとめ

- コントロールプレーンとデータプレーンの分離
- 既存プロトコルに依存しないソフトウェアによる柔軟な制御が実現可能
- 様々なSDN実装が存在するため、実現したい機能や機器の制約条件に合わせて選択する必要がある
- 一方でコントローラの実装に依存するためソフトウェアの理解と習熟が必要不可欠