ENOG84 Meeting

2024/11/22 @ 白玉の湯泉慶

会場諸注意

- 現地会場参加者は Zoom でマイクとスピーカーを有効にしないでください
- Zoom ではホストが勝手にミュートにする場合があります
  - 質問等をされる時はミュートされていないかご確認ください
- 現地の様子を活動記録のため写真に撮らせていただく予定です
  - もし映ると問題があるという方がおられましたらご相談ください
- もしなんかあったら ENOG Slack の #general で教えてください

- 会場 Wi-Fi は SSID: Hotel_Senkei PSK: hotelsenkei です

プログラム

- 14:00~14:20 “導入”
  - 浅間 正和 (有限会社銀座堂)
- 14:20~15:00 “SDNという名のデータプレーンプログラミングの歴史”
  - 海老澤 健太郎 (Arrcus, Inc.) さん
- 15:00~15:40 “SDNのHype Cycleを一通り経験してみて思うこと”
  - 進藤 資訓 (ヴイエムウェア(株) / Broadcom) さん
- 15:40~16:20 “5XCの分散コントローラとデータプレーンアーキテクチャ
  - 中嶋 大輔 (F5) さん　について〜SDNを利用したNaaSサービスの展開〜”
- 16:20~17:00 “テレコムキャリア網を前提に考えていたSDNのあれこれ”
  - 堀場 勝広 (アマゾンウェブサービスジャパン合同会社) さん

ところでみなさん

- SDNってご存知ですか？
- OpenFlowってご存知ですか？

ところでみなさん

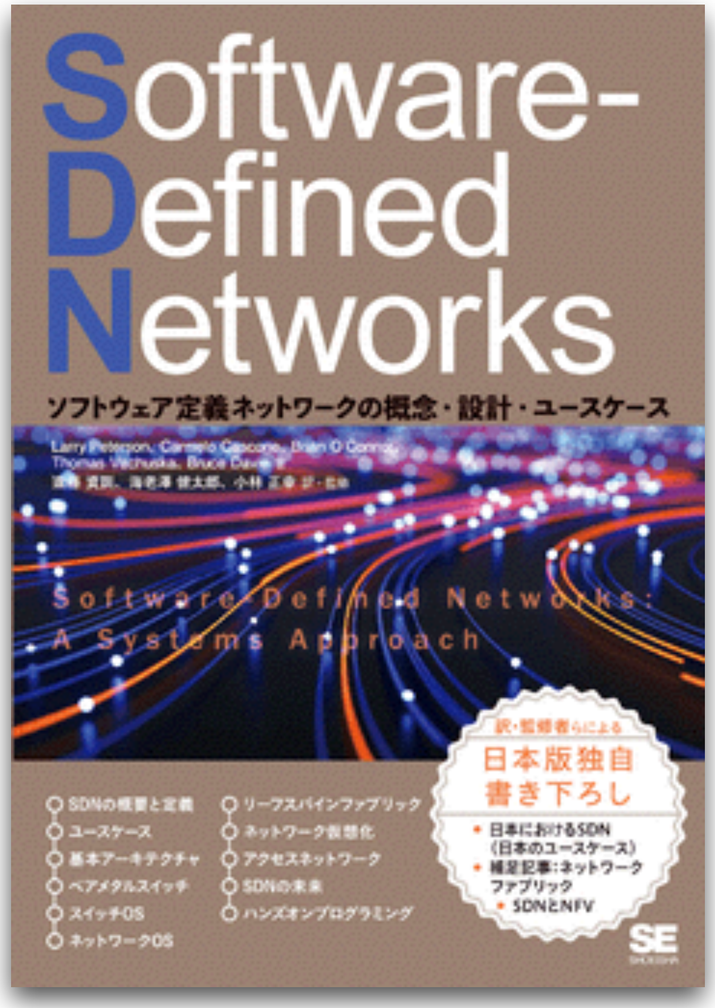
高嶋 隆一 (ブロケード コミュニケーションズ システムズ株式会社).
"で、実際OpenFlowで何ができるの？". JANOG29 Meeting in WAKAYAMA. 2012-01-19.
https://www.janog.gr.jp/meeting/janog29/_downloads/janog29-openflow-after-takashima-01.pdf, p.5.

ところでみなさん

- SDNってご存知ですか？
- OpenFlowってご存知ですか？

プログラムを最大限楽しむ為に
軽くおさらいしときましょう！

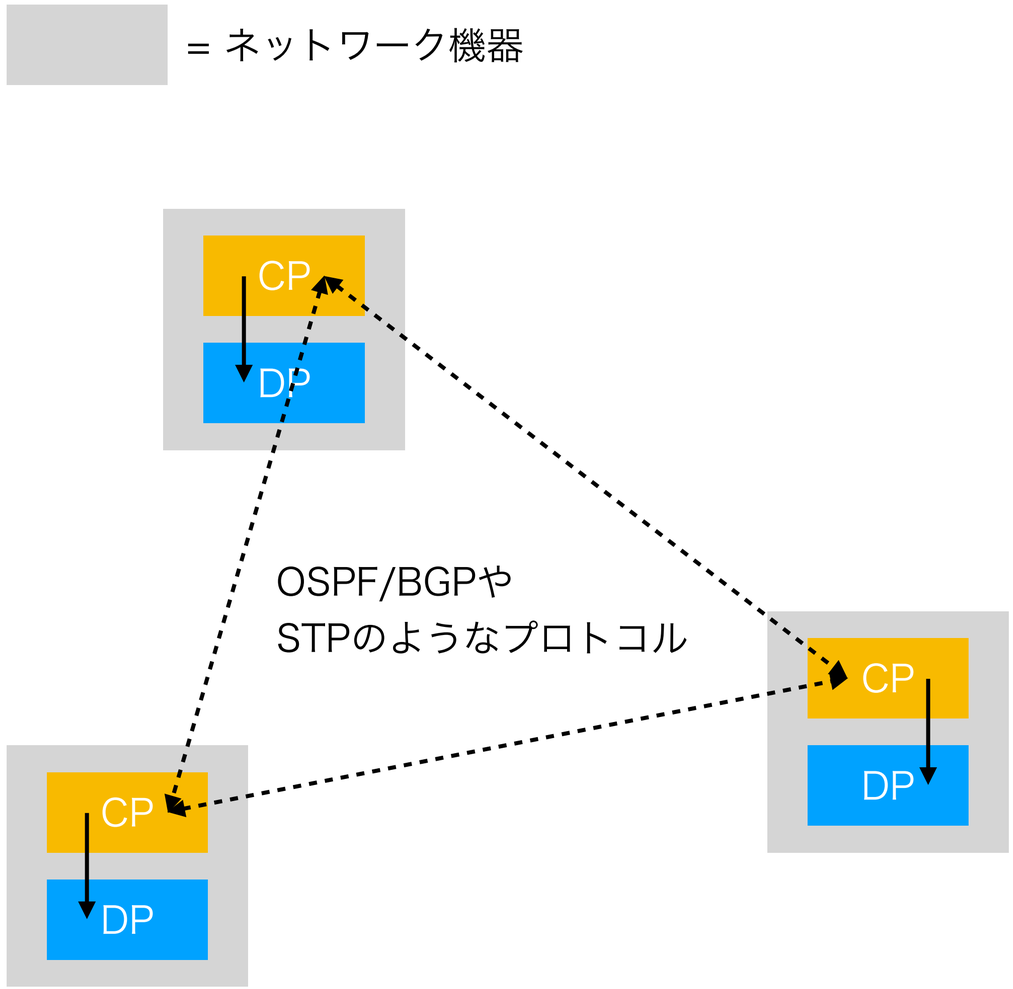
こちらを参考にしました



- Software-Defined Networks ソフトウェア定義ネットワークの概念・設計・ユースケース
- Larry Peterson、Carmelo Cascone、Brian O’Connor、Thomas Vachuska、Bruce Davie 著
- 進藤 資訓、海老澤 健太郎、小林 正幸 翻訳
  進藤 資訓、海老澤 健太郎、小林 正幸 監修
- 発売日 2022年06月22日
- https://www.shoeisha.co.jp/book/detail/9784798172040

- 以降[PCOVD]として引用

自律分散型のネットワーク

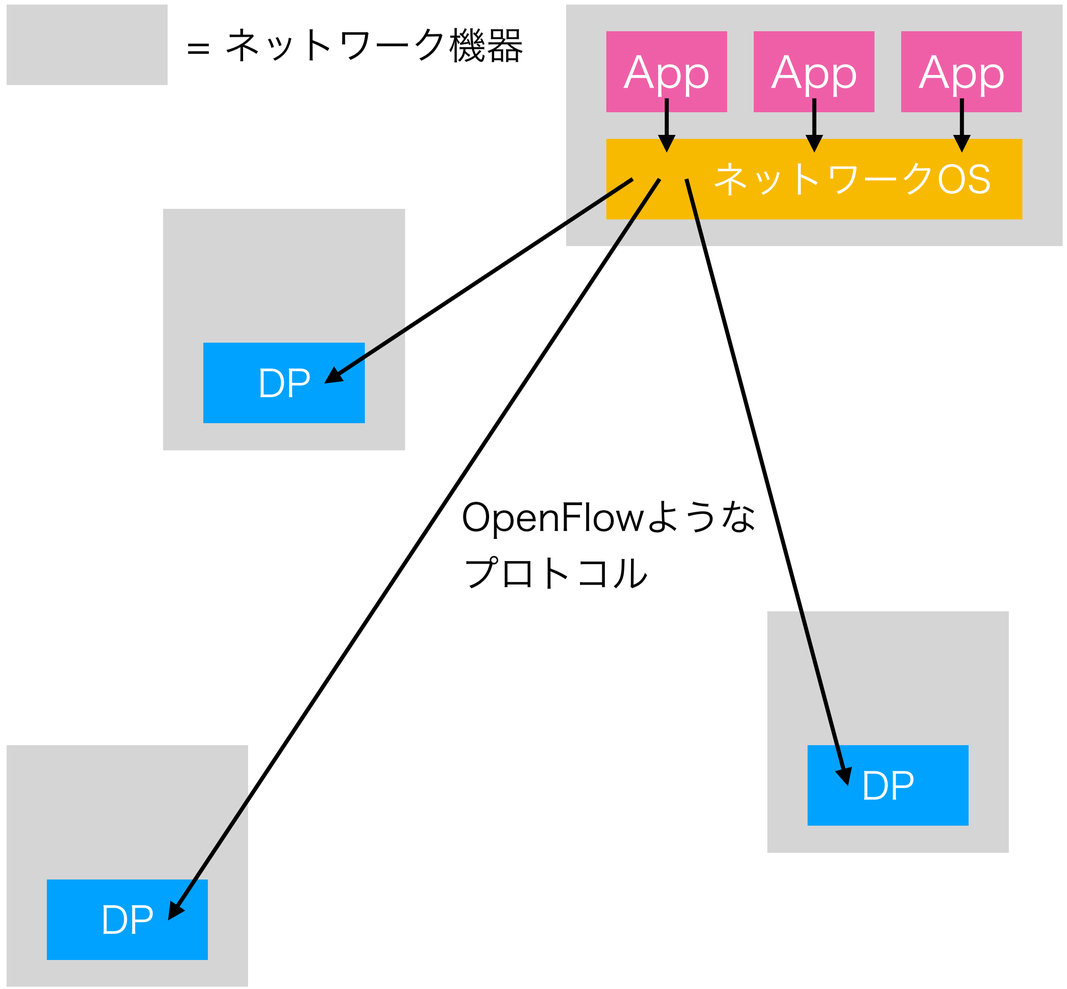


- コントロールプレーン(CP)は他のネットワーク機器と情報をやりとりし転送情報を決定しデータプレーン(DP)に設定する
- データプレーン(DP)はCPによって設定された転送情報に基づきパケットやフレームを転送する
- 誰かがネットワークを集中管理するのではなく各々のネットワーク機器が自分自身で転送情報を決定し管理する

[PCOVD]の主張

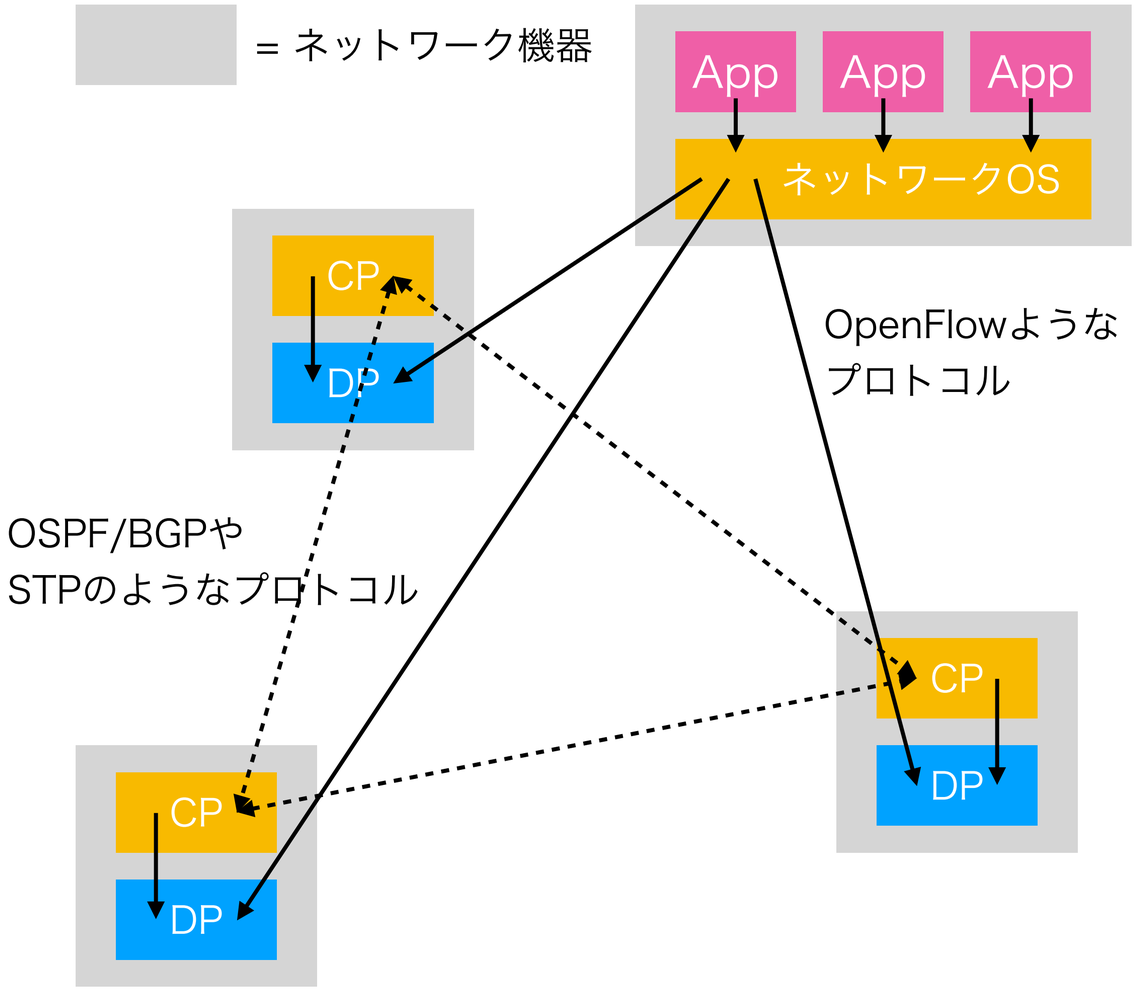
- 自律分散型のネットワークには以下の問題がある
  - ネットワーク機器は硬直化しておりイノベーションを阻害している
    - コンピュータがメインフレームからオープンなPCへ移行することで多様なハード・OS・アプリが生まれコストダウンをもたらした様にネットワーク機器もオープン化することでイノベーションが生まれる
  - プロトコルの標準化は非常に時間がかかりオペレータのアイデアを即座に実現することが困難となっている
    - データプレーンがオープン化されることでそれらを集中管理することが可能となる
    - オープンソースコミュニティによるイノベーションも加速するはず

純粋SDNのネットワーク



- ネットワークOSはネットワーク機器の隣接情報や統計情報などを収集しそれらをAppに提供
- ネットワークOSは以下の方法でネットワーク機器のDPに転送情報を設定する
  - Appから直接指定された転送情報をそのまま設定
  - Appから指定された要件を元に転送情報を計算しそれを設定
- ネットワークOSがネットワーク機器の転送情報を集中管理する

SDNライトのネットワーク



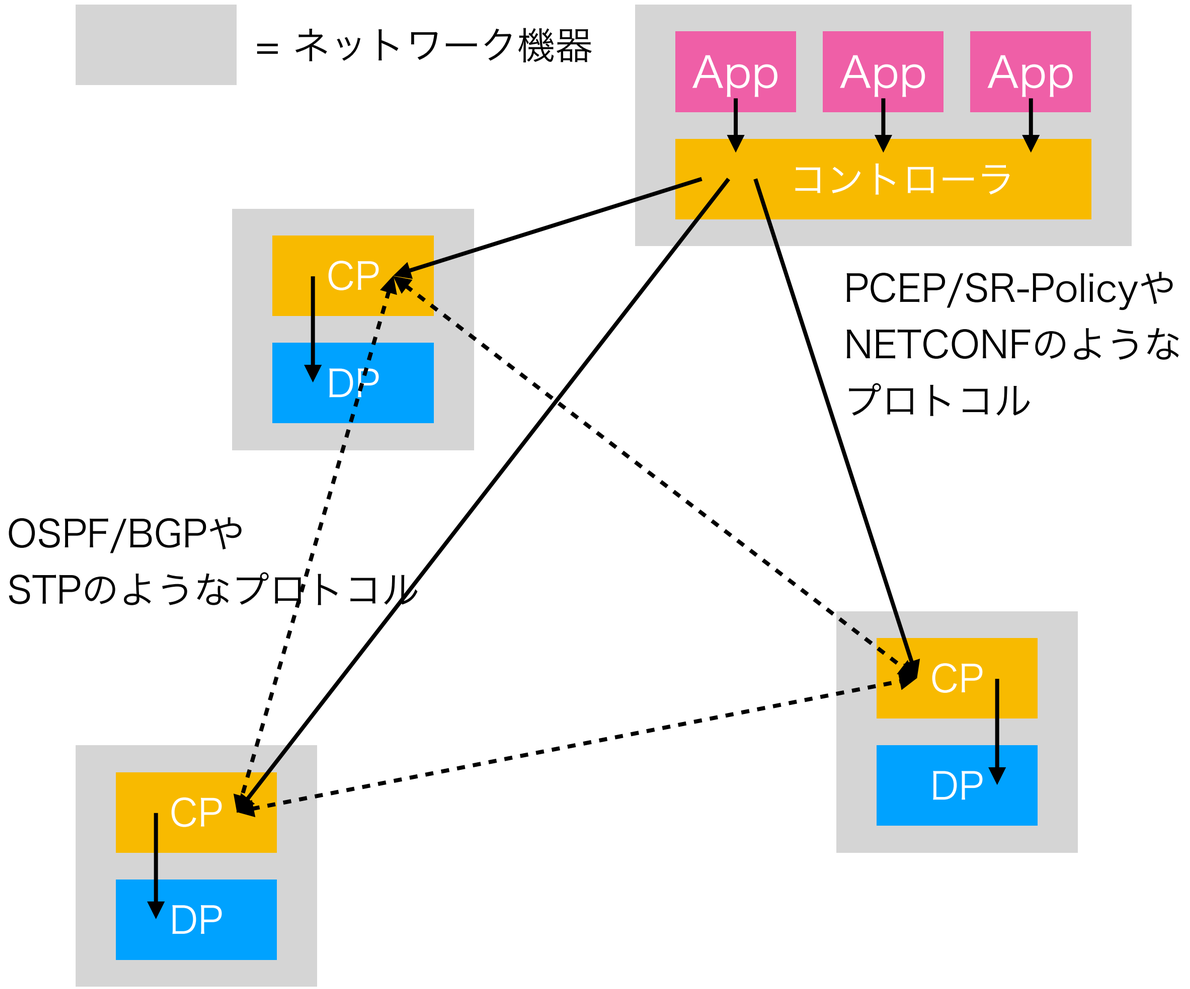
- 従来の自律分散型のネットワークからいきなり純粋SDNへ移行するのは現実的ではないため[PCOVD]ではこれらのハイブリッドな方式をSDNライトと呼んでいる
- 基本的には自律分散型のプロトコルで計算した転送情報を用いそれからはずれるものだけをネットワークOSから制御するといった使い方ができる

[PCOVD]から外れて

- [PCOVD]ではSDNを以下のように定義
  - “コントロールプレーンとフォワーディングプレーンが物理的に分離しておりひとつのコントロールプレーンで複数のフォワーディングデバイスを制御するネットワークのこと”
- [PCOVD]ではNETCONFなどを用いてプログラムで設定を制御する方式は以下の理由からSDNとは呼べないと言及
  - 理由(a): CP/DP間のプログラマビリティを完全にサポートできない
  - 理由(b): リアルタイムなコントロールループをサポートできない
- [PCOVD]ではPCEPやSR-Policyのような方式については言及がない

本プログラムではNETCONF/PCEP/SR-PolicyなどもSDNとします

広義のSDNのネットワーク



- コントローラはネットワーク機器の隣接情報や統計情報をBGP-LS(BGP Link State)などで収集する
- コントローラは以下の方法でネットワーク機器のDPに転送情報を設定する
  - 例1) NETCONFのようなプログラムから設定を変更する仕組みを用いてネットワーク機器の設定を変更
  - 例2) PCEP(Path Computation Element Protocol)やSR-Policy(Segment Routing Policy)のような自律分散型ルーティングプロトコルで転送情報を注入する仕組みを用いる

本プログラムでのSDNの分類

- 純粋SDN
  - 従来の自律分散型プロトコルによらず全てのデータプレーンを集中管理されたコントロールプレーンから直接制御するネットワーク
- SDNライト
  - 従来の自律分散型プロトコルも用いた上で差分を制御するネットワーク

狭義のSDN
([PCOVD]の定義によるSDN)

- 広義のSDN
  - 機器の設定自動化や自律分散型プロトコル経由で制御するネットワーク

ざっくりOpenFlow

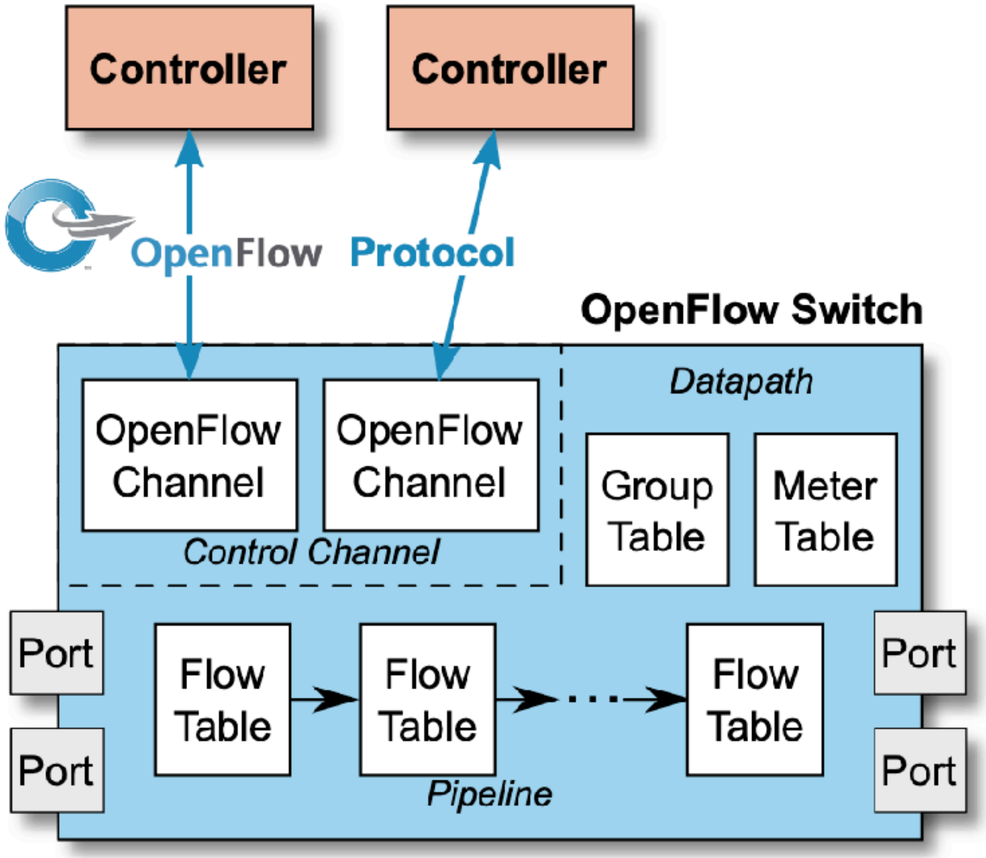


Figure 1: Main components of an OpenFlow switch.

OpenFlow Switch Specification Version 1.5.1 より引用

ざっくりOpenFlow

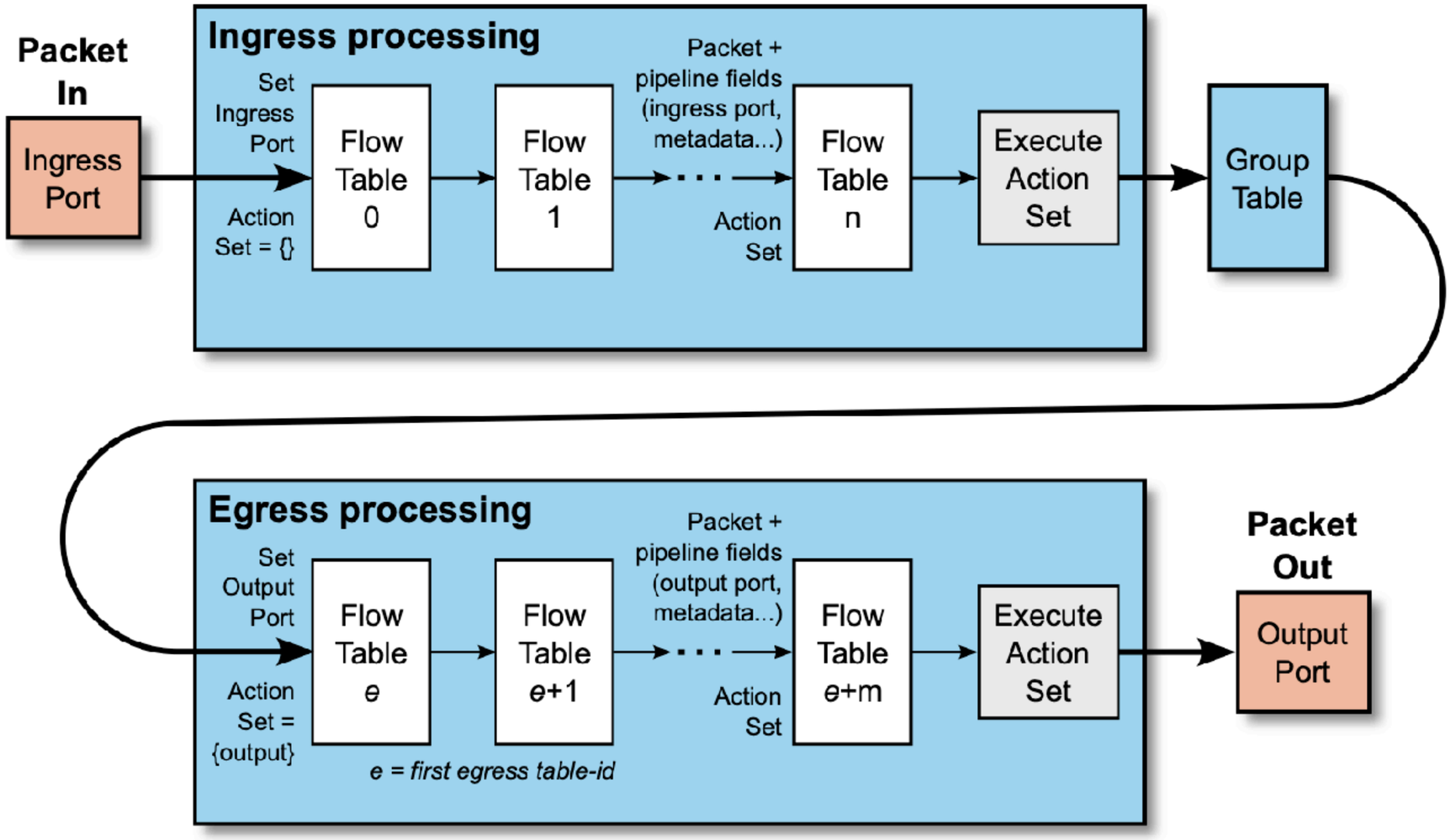


Figure 2: Packet flow through the processing pipeline.

Field		Description
OXM_OF_IN_PORT	*Required in ingress*	Ingress port. This may be a physical or logical port.
OXM_OF_ACTSET_OUTPUT	*Required in egress*	Egress port from action set.
OXM_OF_ETH_DST	*Required*	Ethernet destination address. Can use arbitrary bitmask
OXM_OF_ETH_SRC	*Required*	Ethernet source address. Can use arbitrary bitmask
OXM_OF_ETH_TYPE	*Required*	Ethernet type of the OpenFlow packet payload, after VLAN tags.
OXM_OF_IP_PROTO	*Required*	IPv4 or IPv6 protocol number
OXM_OF_IPV4_SRC	*Required*	IPv4 source address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV4_DST	*Required*	IPv4 destination address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV6_SRC	*Required*	IPv6 source address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV6_DST	*Required*	IPv6 destination address. Can use subnet mask or arbitrary bitmask
OXM_OF_TCP_SRC	*Required*	TCP source port
OXM_OF_TCP_DST	*Required*	TCP destination port
OXM_OF_UDP_SRC	*Required*	UDP source port
OXM_OF_UDP_DST	*Required*	UDP destination port

Table 12: Required match fields.

ざっくりOpenFlow

5.8 Actions

A switch is not required to support all action types, just those marked *"Required Action"* below. Required actions must be supported in all flow tables. The controller can also query the switch about which of the *"Optional Action"* it supports (see 7.3.5.18).

Required Action: **Output *port_no***. The Output action forwards a packet to a specified OpenFlow port (see 4.1) where it starts egress processing. OpenFlow switches must support forwarding to physical ports, switch-defined logical ports and the required reserved ports (see 4.5).

Required Action: **Group *group_id***. Process the packet through the specified group (see 5.10). The exact interpretation depends on group type.

Required Action: **Drop**. There is no explicit action to represent drops. Instead, packets whose action sets have no output action and no group action must be dropped. This result could come from empty instruction sets or empty action buckets in the processing pipeline (see 5.6), or after executing a *Clear-Actions* instruction (see 5.5).

ざっくりOpenFlow

Optional Action: **Set-Queue *queue_id*.** The set-queue action sets the queue id for a packet. When the packet is forwarded to a port using the output action, the queue id determines which queue attached to this port is used for scheduling and forwarding the packet. Forwarding behavior is dictated by the configuration of the queue and is used to provide basic Quality-of-Service (QoS) support (see section 7.3.5.8).

Optional Action: **Meter *meter_id*.** Direct packet to the specified meter (see 5.11). As the result of the metering, the packet may be dropped (depending on meter configuration and state). If the switch supports meters, it must support this action as the first action in a *list of actions*, for backward compatibility with earlier versions of the specification. Optionally, a switch supporting meters may also support the meter action in other positions in a *list of actions*, multiple meter actions in a *list of actions* and the meter action in the *action set* (see section 7.3.5.14).

Optional Action: **Push-Tag/Pop-Tag *ethertype*.** Switches may support the ability to push/pop tags as shown in Table 2. To aid integration with existing networks, we suggest that the ability to push/pop VLAN tags be supported.