OpenFlow概要

2014-03-03

事業戦略グループ
OSS基盤技術センター
OSS技術第二課
山本高志

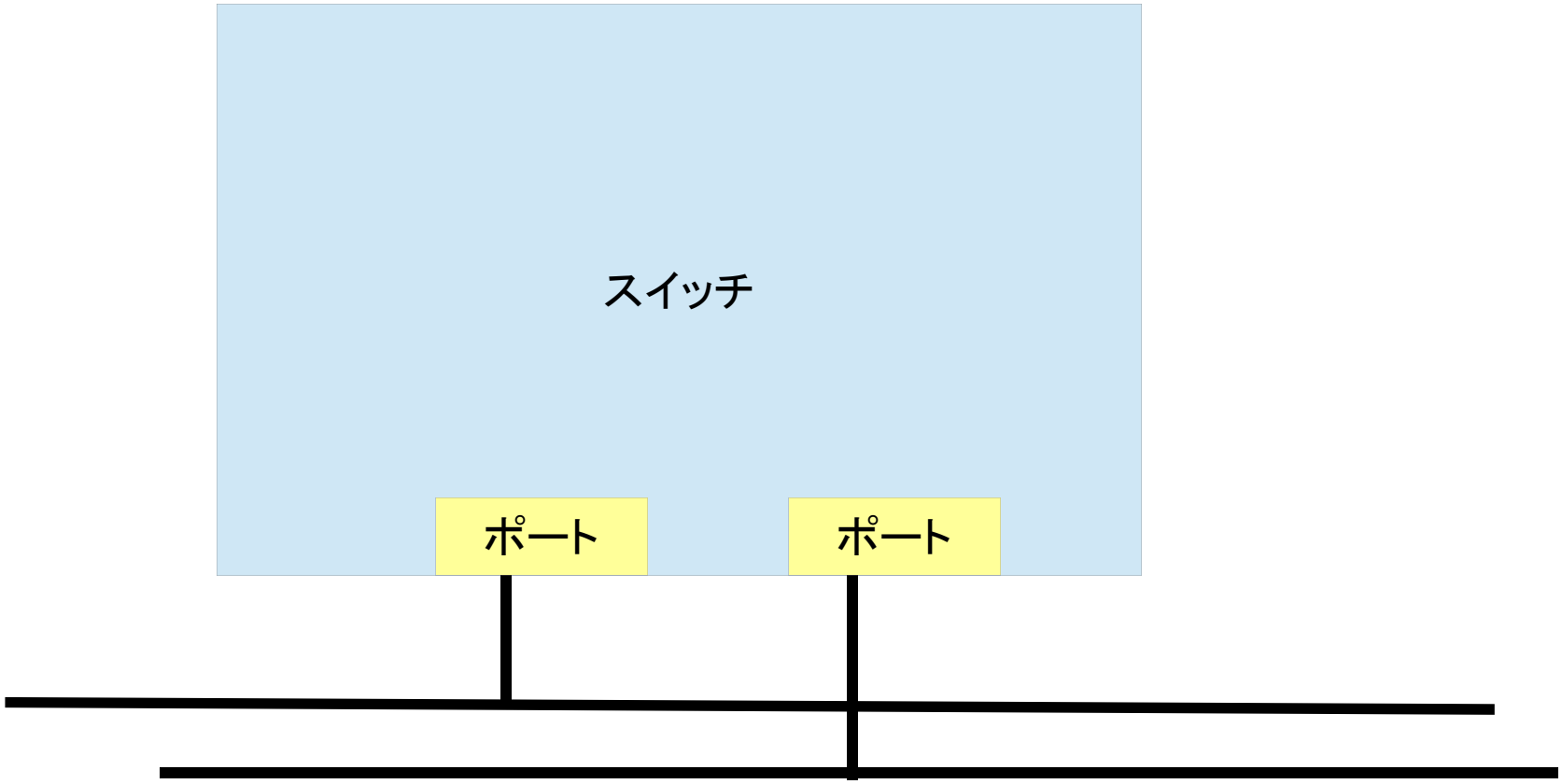
内容

- プロトコルの紹介
- OpenFlow 1.3.2 ベース
- 主にControllerを実装する視点で
- ハードウェア実装のことは知りません

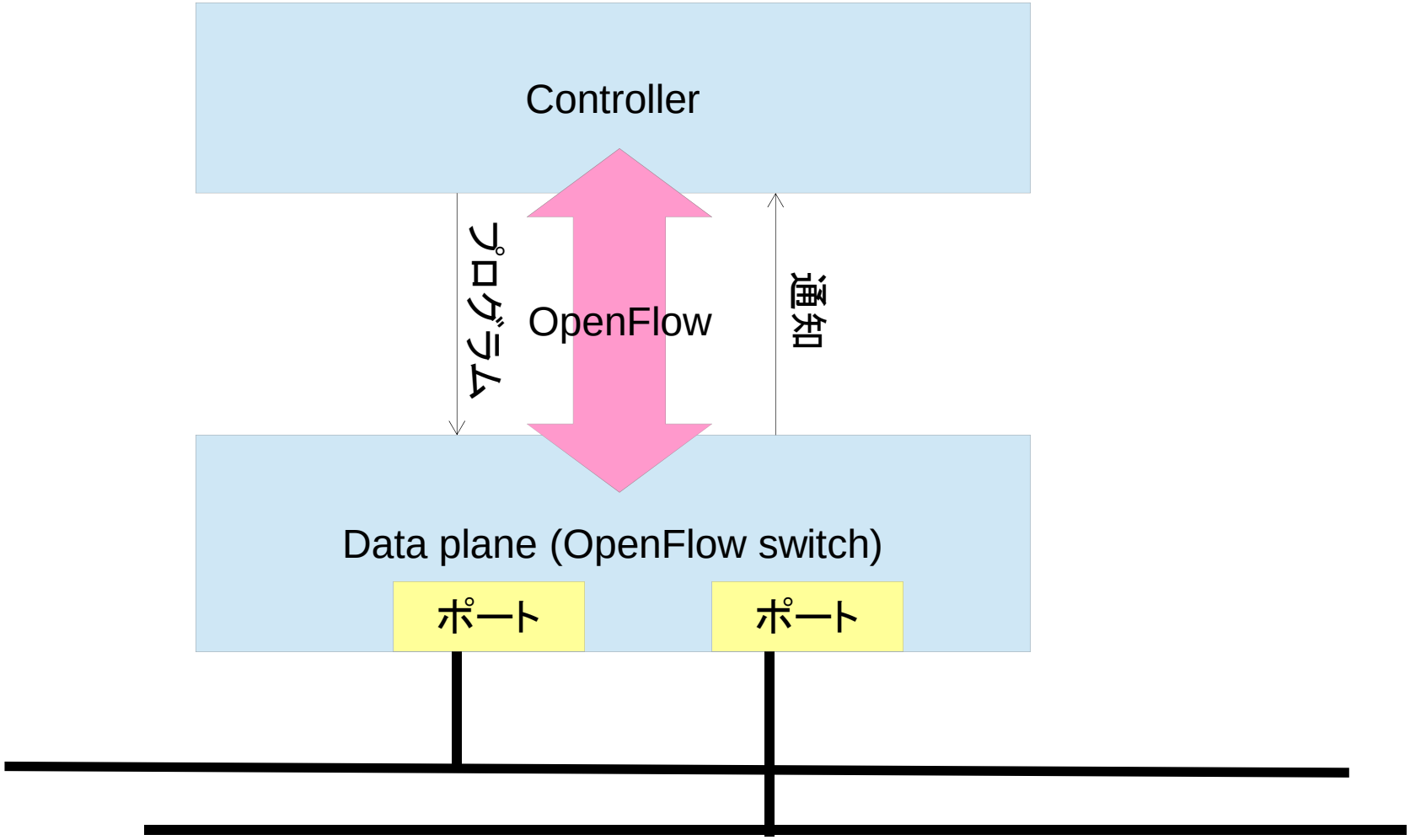
OpenFlow

- Switchを制御するためのプロトロル
- 規格の策定はメンバーのみの密室
  - 会費30,000USD/年/会社
  - 規格バグ報告もメンバーのみ
- 策定済み規格は無料で公開されている
- 最新版 1.4.0
- 良く使われるバージョン 1.0.x (次点1.3.x)

Traditional switch



OpenFlow switch



Controller

- Data planeをプログラムする (ofp_flow_mod等)
- Forwarding logicの実装

Switch

- パケット受信
  - Flow tableの内容に従って処理
- パケット送信
- 各種イベントをcontrollerに通知
- controllerによる設定に従って受信パケットを処理

OpenFlow channel

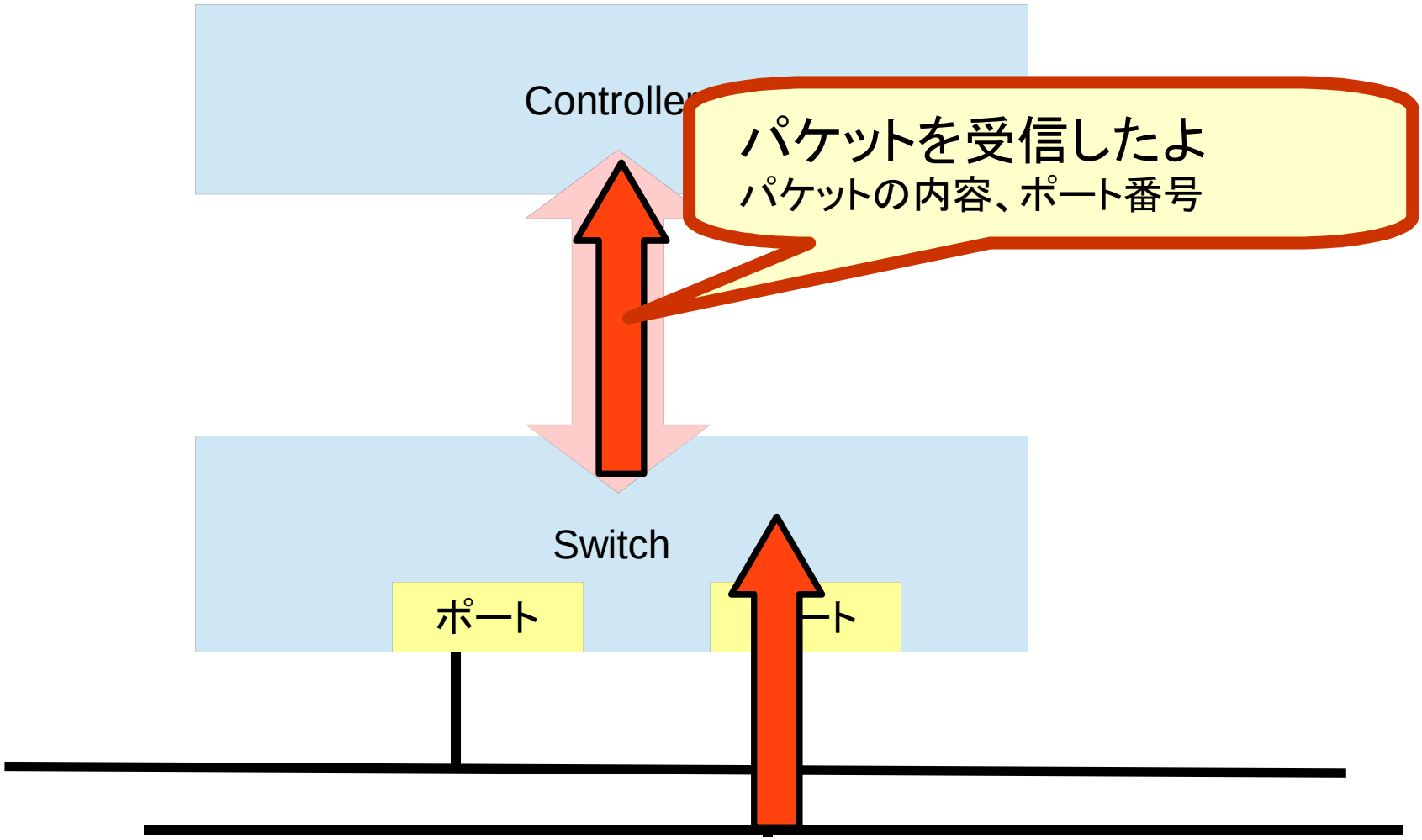
- ControllerとSwitchの間の通信路
- 実装依存
  - 普通はTCPまたはTLS
  - 普通はSwitchからControllerに接続
- この通信路上で、互いに非同期にメッセージを送り合う

OpenFlow message

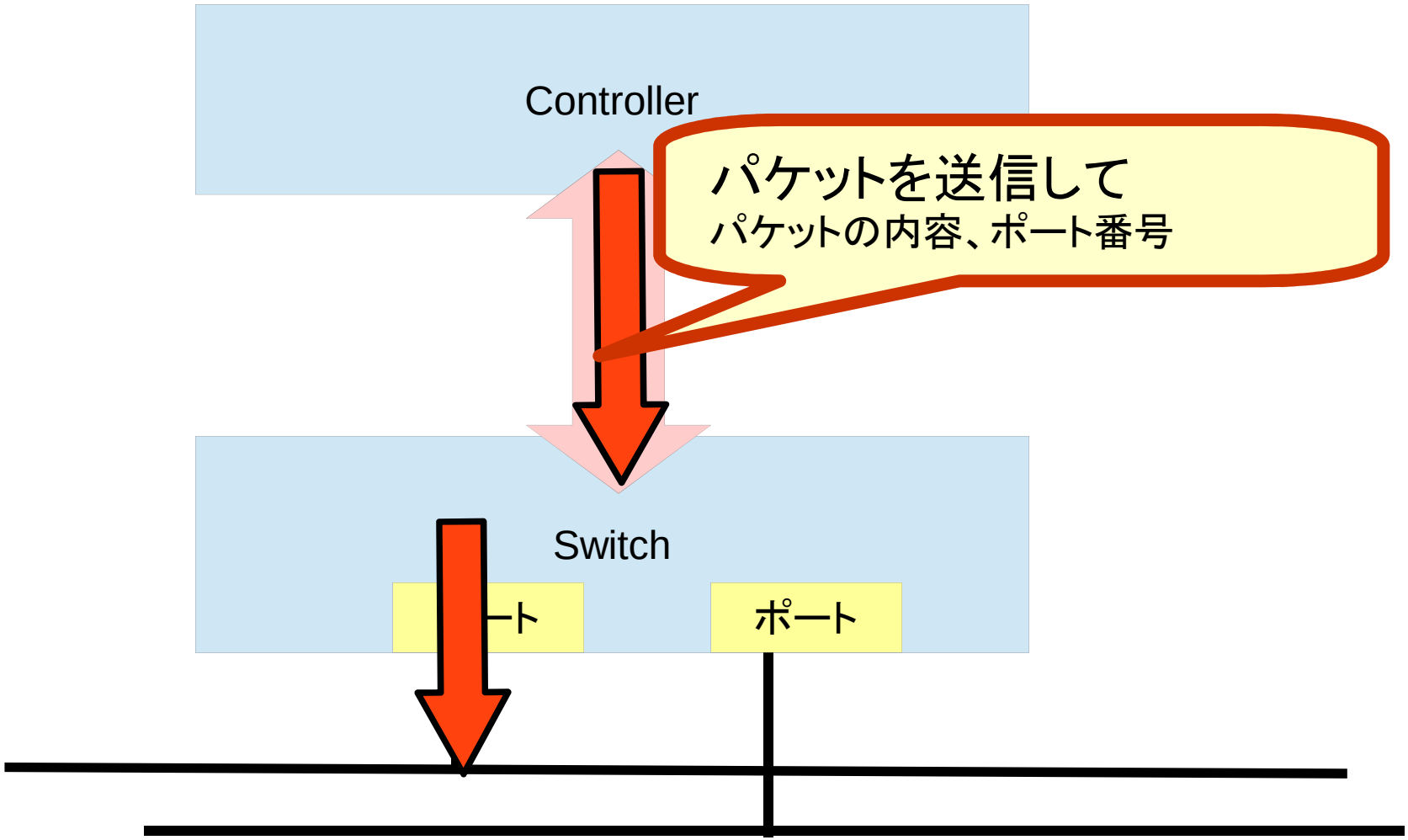
- xidでrequestとreplyを対応づけ
- 受信側は任意の順番で処理してよい

どういうことができるのか

Packet in

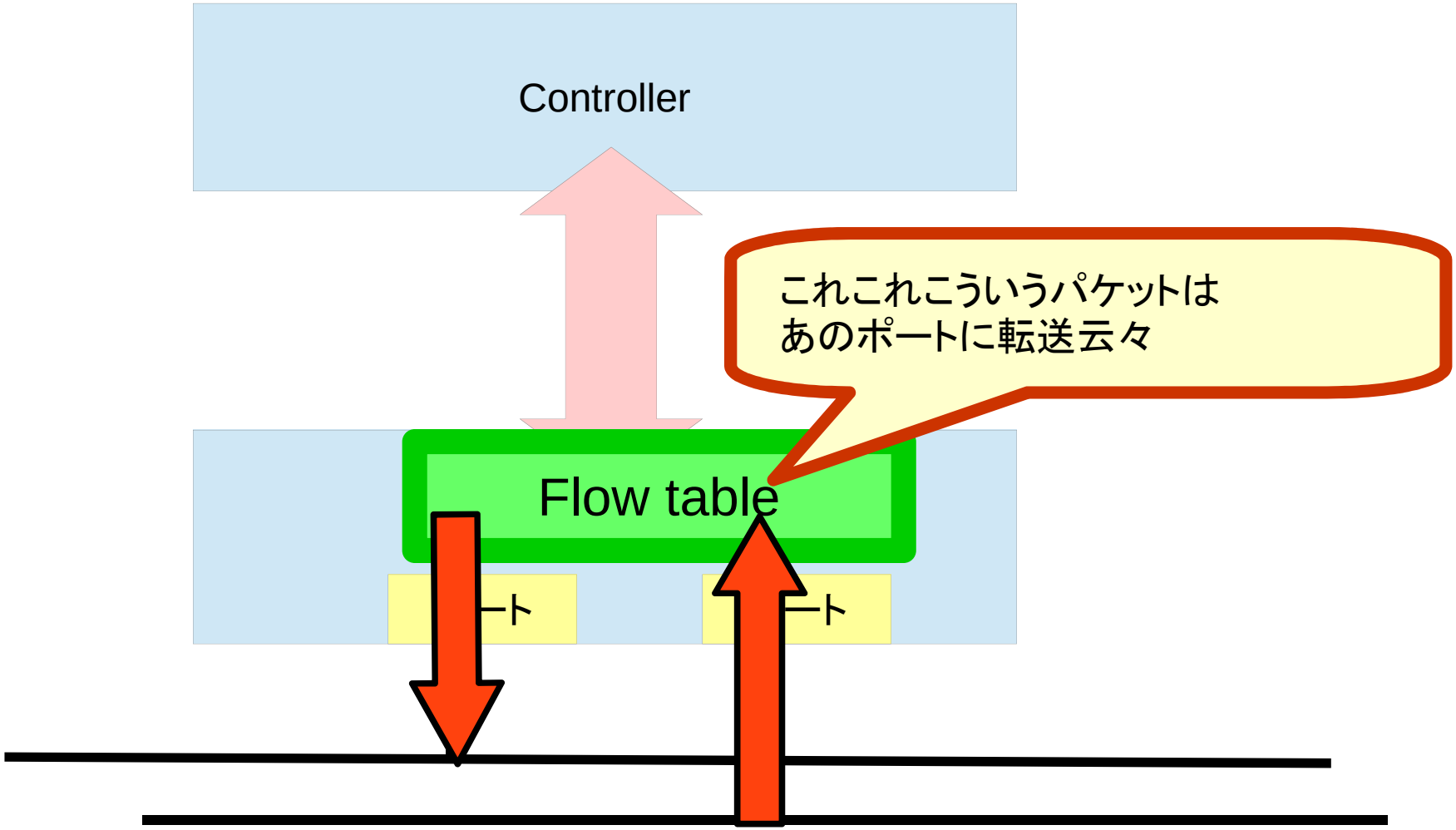


Packet out



Flow table

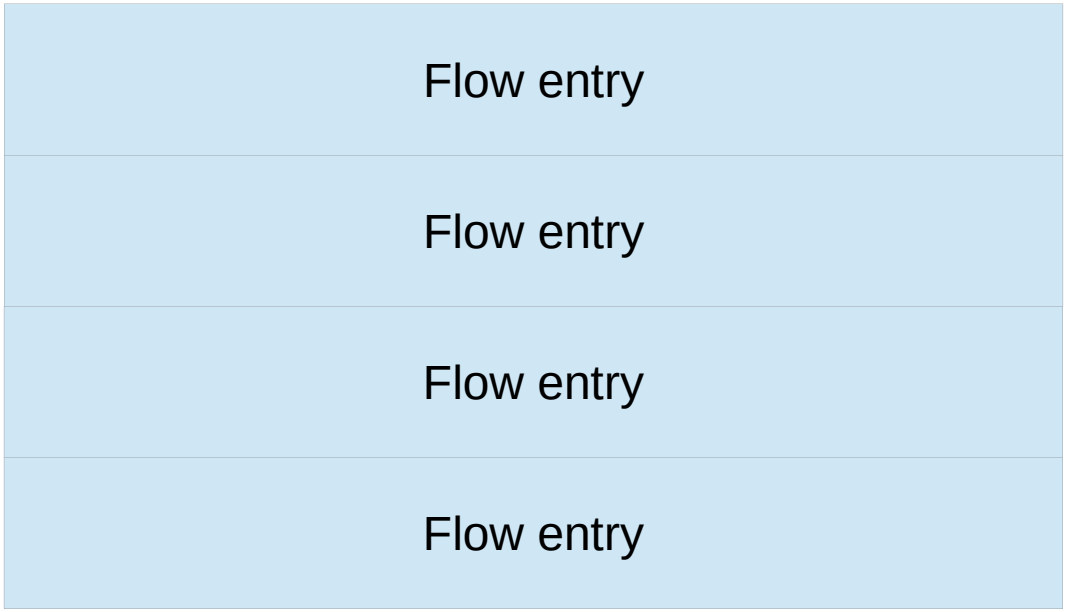
- 処理内容をあらかじめ設定しておく



Flow table

Flow table

- 受信したパケットの処理方法の記述
- 複数実装しても良い



Flow entry

- Match
  - 例. ipv4 src addressが10.0.0.1
- Instructions
  - 例. ポート1から送信

match	priority	instructions

Match

- Field
  - 疑似フィールド
    - in_port,
  - 通常のフィールド
    - eth_type, eth_src, ipv4_src, ….
- 値
  - 10.0.0.1など
- マスク
  - 255.255.0.0など

Instruction

- 処理内容の記述
  - actionの指定・実行
    - write_actions, apply_actions, …
  - 他のtableへのjump
    - goto_table
  - その他
    - meter

Action

- パケット内容の変更
  - Push/pop headers
  - Set field
- パケットの送信
  - Output
- その他
  - Group
  - Set-queue
  - Drop

Action list

- actionのリスト
- 記述した順番通り実行される
- 同種のものが複数含まれていてもよい
- 実装はoptional

Action set

- actionの集合
- 各action種別につき一個まで含むことができる
- 実行順序はaction種別により決まっている
- 実装必須
  - Open vSwitchではv2.1.0からサポート

Group

- bucket(Action set)のリスト
- group種別
  - All 全てのbucketを実行
  - Select 実装依存の方法でbucketを一つ選んで実行
  - Fast failover 対応するportがliveであるbucketを実行

OpenFlowメッセージ

- 主要なメッセージの紹介

ofp_hello

- OpenFlowバージョン情報の交換
- 1.3.xは4 (わかりにくい)

ofp_echo_request

- ofp_echo_replyの送信を要求
- 生死確認などに使う

ofp_switch_features_request

- Controller→Swtich
- Swtich情報(ofp_switch_features_reply)の要求
  - Datapath ID
  - Flow tableの数
  - 他
  - OpenFlow 1.2まではポートの一覧がここに含まれていたが、OpenFlow 1.3からは無くなった
    - cf. OFPMP_PORT_DESC

ofp_multipart_request/reply

- メッセージサイズの上限(64KB)を回避するため、要求や回答を複数のメッセージに分割して送信する
- 以下のような要求に使用される
  - OFPMP_PORT_DESC
    - ポート一覧の取得
  - OFPMP_FLOW
    - Flow tableの取得
  - 他

ofp_port_status

- Switch→Controller
- ポートの追加・削除・状態変更の通知

ofp_packet_in

- Switch→Controller
- Output actionで特殊なポート(controller)を指定することで、パケットをOpenFlow channel経由でControllerに転送させることができる
- 事前に設定(ofp_set_config)しておけば、パケットの先頭何バイトまで等の指定が可能

ofp_packet_out

- Controller→Switch
- パケットの内容とInstructionsを指定し、実行させる
- この内容のパケットをこのポートから送信する、など

ofp_flow_mod

- Controller→Switch
- Flow tableの内容変更

ofp_error_msg

- Switch→Controller
- 対応する要求がエラーになったことを通知

ofp_barrier_request

- Controller->Switch
- これ以前に受信した全てのメッセージの処理を完了させ、replyやerrorがあれば送信し、その後でofp_barrier_replyを返すことを要求

エラー処理

- 多くのメッセージは正常完了時は何も返さない
- そのため、真面目にエラーチェックをするなら、以下のような処理を行なう必要がある
  - 要求メッセージの後にofp_barrier_requestを送信
  - ofp_barrier_replyを待つ
  - その間に元の要求に対応するofp_error_msgを受信しなければ成功
- 真面目にエラーチェックをしている例
  - Open vSwitch (ovs-ofctl)
  - Ryu (ofctl application)

Port liveness

- 定義は実装依存、OpenFlowの外
- STPなど

queue

- port毎に複数の送信queueを持てる
- ofp_queue_get_config_requestで構成情報を取得できる
  - Min rate (guarantee)
  - Max rate
  - Length
- 構成方法はOpenFlowの外

meter

- Rate limitなどのQoS制御
- Band types
  - Drop
    - パケットを落とす
  - Dscp remark
    - IP DSCPを指定した値増やす
- rateがN以上のときはdscp remark、M以上ならdrop、というような記述ができる

関連プロトコル

- OVSDB
  - RFC 7047
  - 管理プロトコル
  - JSON-RPC (風?)
  - 事実上Open vSwitch専用
- OF-Config
  - ONFによる策定
  - NETCONFのスキーマ
  - 目的はOVSDBと同じ
  - queueの構成なども含まれる

まとめ

- Switchの実装を決めうちにするのがおすすめ
  - 機能差異を吸収するのは大変
  - 実装依存が多い