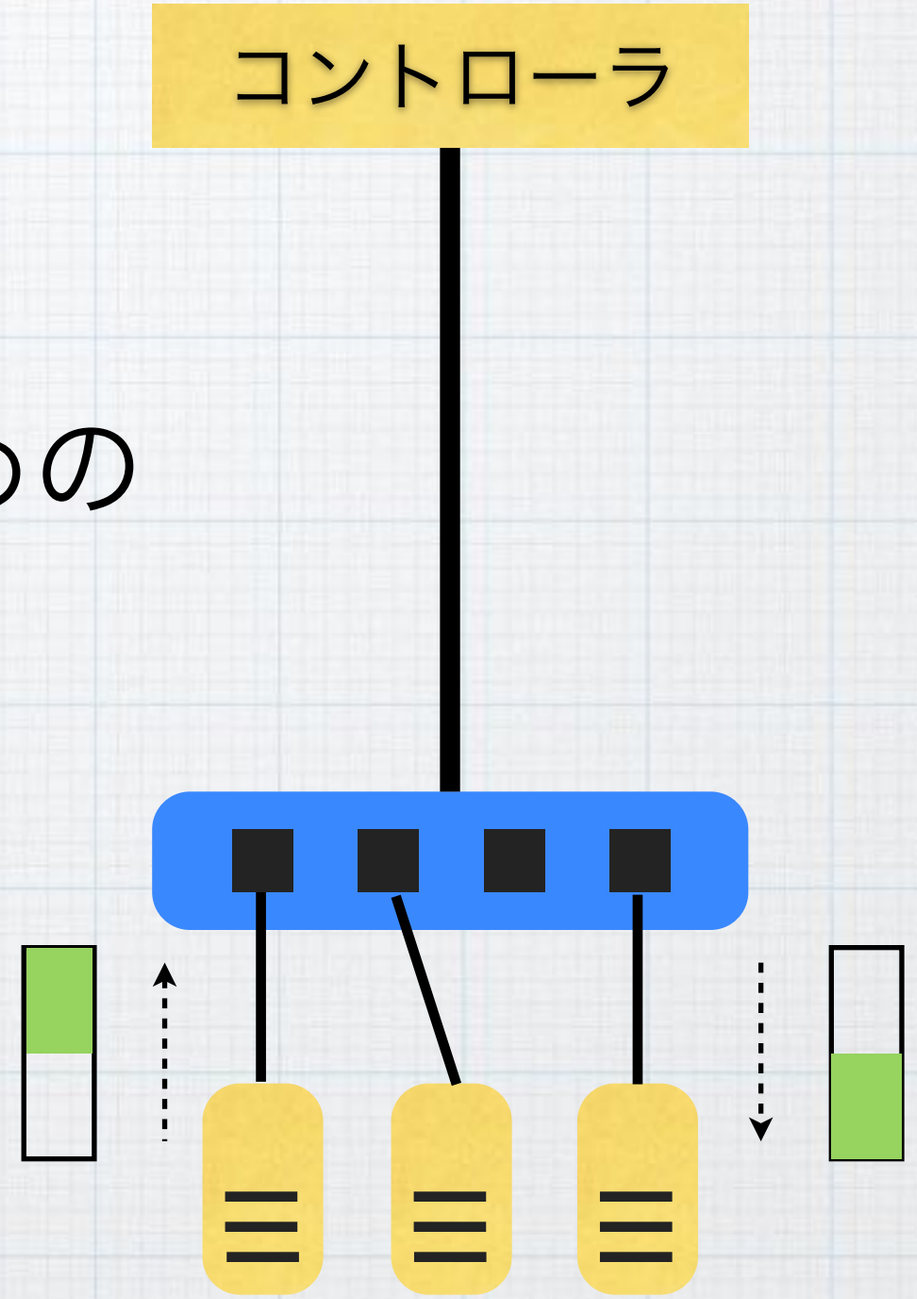
Cbench

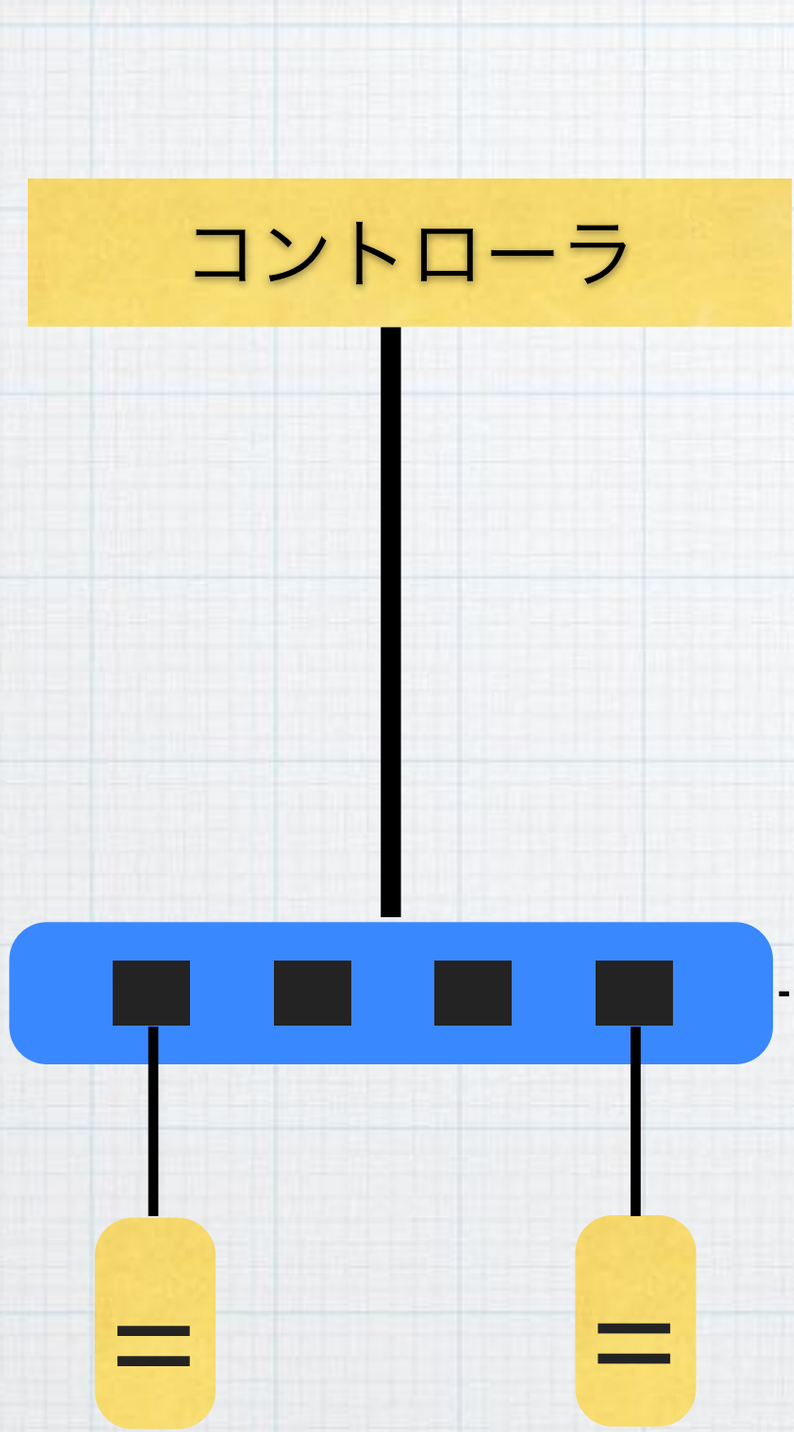
実用コントローラを作るための
基礎知識



OpenFlowの3大メッセージ

FlowMod · PacketIn · PacketOut

コントローラ



Flow Table

host1

00:00:..:01

192.168.0.1

host2

00:00:..:02

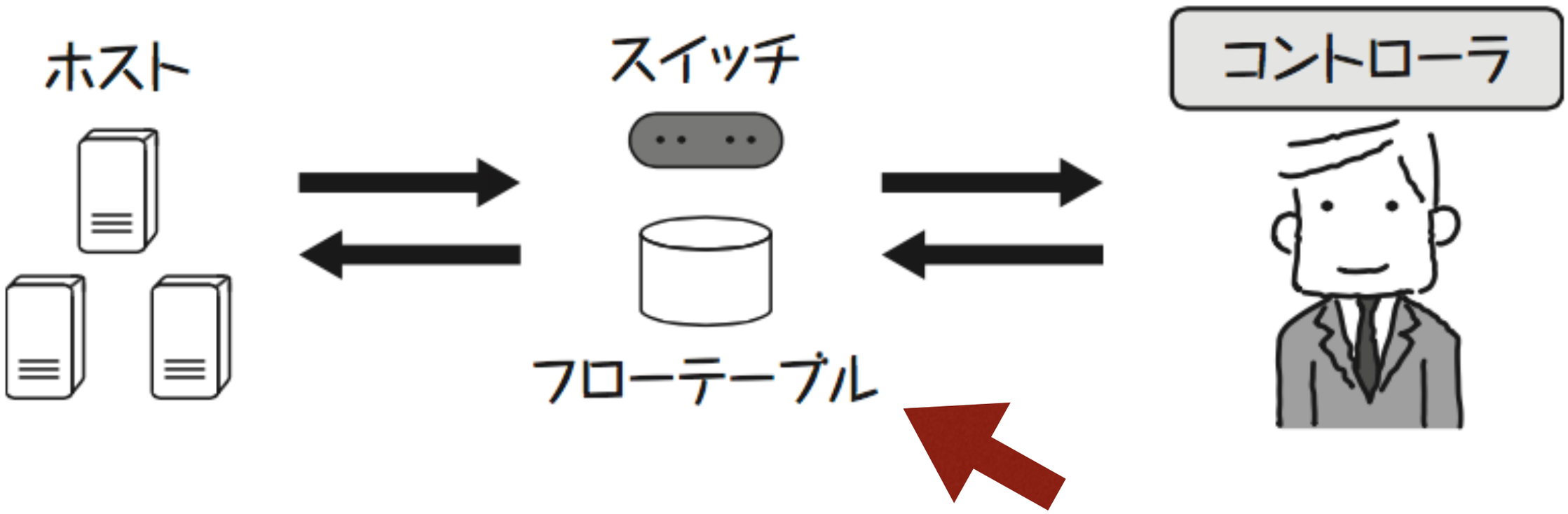
192.168.0.2

復習: Flow Table

パケットの処理ルールを管理する DB

速い
(ハードウェア)

すごく遅い
(ソフトウェア)



復習:フローエントリ

パケットの条件	処理方法	処理量
マッチフィールド	アクション	カウンタ
送信元 IP アドレス = 192.168.1.0	ポート 8 番に転送	80 パケット
VLAN ID = 10	ポート 10 番に転送	64 パケット
送信元 MAC アドレス = 00:50:56:c0:00:08	VLAN ID = 2 を付けてポート 8 番に転送	24 パケット
送信元 IP アドレス = 203.0.113.0/16	パケットを破棄	10 パケット

FlowMod

フローエントリの追加・書き換え・削除

```
# フローエントリを追加
send_flow_mod_add(
  dpid,  # スイッチのIĐ
  match: Match.new( … ) # マッチフィールド
  actions:  …  # アクション
)
```

マッチフィールド

パケットの条件 マッチフィールド	処理方法 アクション	処理量 カウンタ
送信元 IP アドレス = 192.168.1.0	ポート 8 番に転送	80 パケット
VLAN ID = 10	ポート 10 番に転送	64 パケット
送信元 MAC アドレス = 00:50:56:c0:00:08	VLAN ID = 2 を付けてポート 8 番に転送	24 パケット
送信元 IP アドレス = 203.0.113.0/16	パケットを破棄	10 パケット

```
send_flow_mod_add(
  datapath_id,
  match: Match.new(in_port: 1)
  # …
```

・ポート1番から入ってきたら

```
send_flow_mod_add(
  datapath_id,
  match: Match.new(
    ip_destination_address: '192.168.0.30',
    transport_destination_port: 80
  )
  # …
```

- 宛先が 192.168.0.30 で HTTP だったら

指定できる条件

- Ingress port
- Ether src
- Ether dst
- Ether type
- IP src
- IP dst
- IP proto
- IP ToS bits
- TCP/UCP src port
- TCP/UDP dst port
- VLAN id
- VLAN priority

```
send_flow_mod_add(
  datapath_id,
  match: ExactMatch.new(message),
  # …
)
```

- ExactMatch: 条件12個がすべて同じ

アクション

フローエントリにヒットしたパケットの処理方法

```
# フローエントリを追加
send_flow_mod_add(
  dpid,  # スイッチのIĐ
  match: Match.new( … ) # マッチフィールド
  actions:  …  # アクション
)
```

アクション

パケットの条件	処理方法	処理量
マッチフィールド	アクション	カウンタ
送信元 IP アドレス = 192.168.1.0	ポート 8 番に転送	80 パケット
VLAN ID = 10	ポート 10 番に転送	64 パケット
送信元 MAC アドレス = 00:50:56:c0:00:08	VLAN ID = 2 を付けてポート 8 番に転送	24 パケット
送信元 IP アドレス = 203.0.113.0/16	パケットを破棄	10 パケット

アクション

パケットの転送

- SendOutPort.new(ポート番号)

パケット書き換え (Match と同じ 12 種類)

- SetEtherDestinationAddress.new(新しい宛先MACアドレス)
- SetIpDestinationAddress.new(新しい宛先IPアドレス)
- Etc.

```
send_flow_mod_add(
  datapath_id,
  match: Match.new(in_port: 1),
  actions: SendOutPort.new(4),
  # …
```

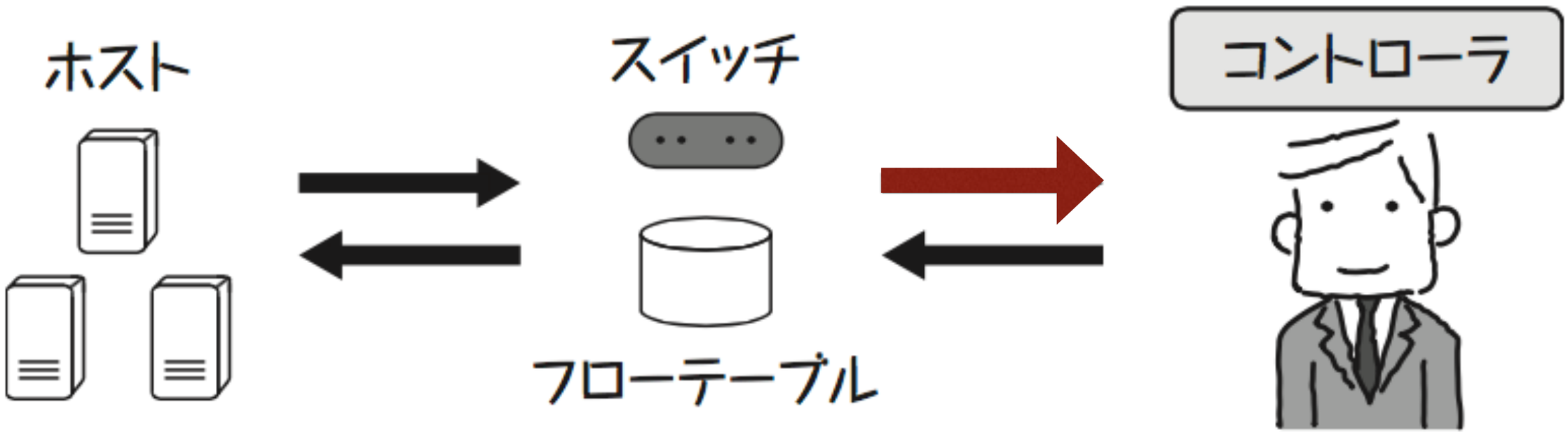
1. ポート1番から入ってきたら
2. ポート4番に出す

```
send_flow_mod_add(
  datapath_id,
  match: Match.new(in_port: 1),
  actions: [SetEtherDestinationAddress.new(…),
            SetIpDestinationAddress.new(…),
            SendOutPort.new(4)]
  # …
```

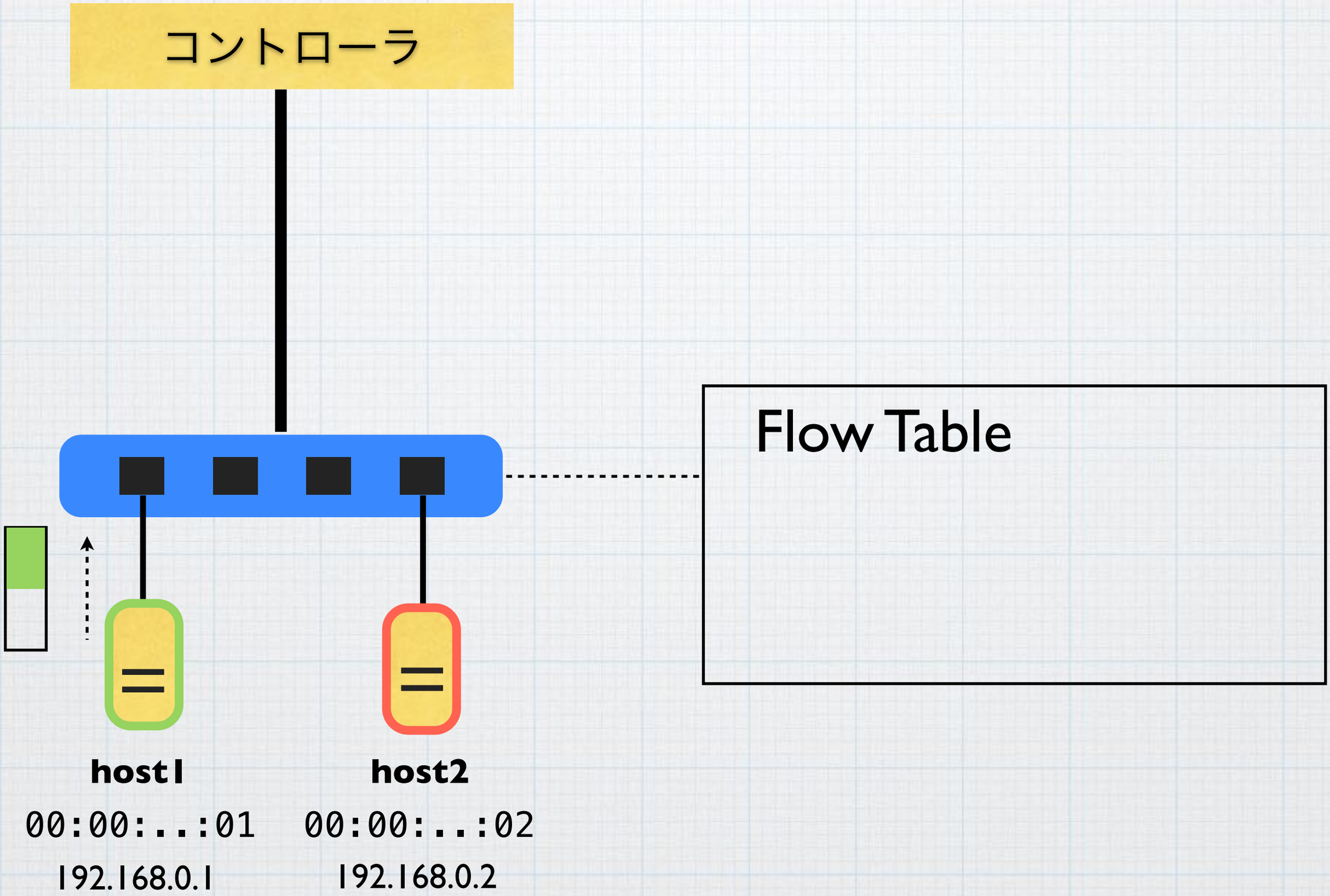
- 複数アクションが指定可能
- 書き換え2回→転送

PacketIn

フローテーブルで処理できない
パケット到着を知らせるメッセージ

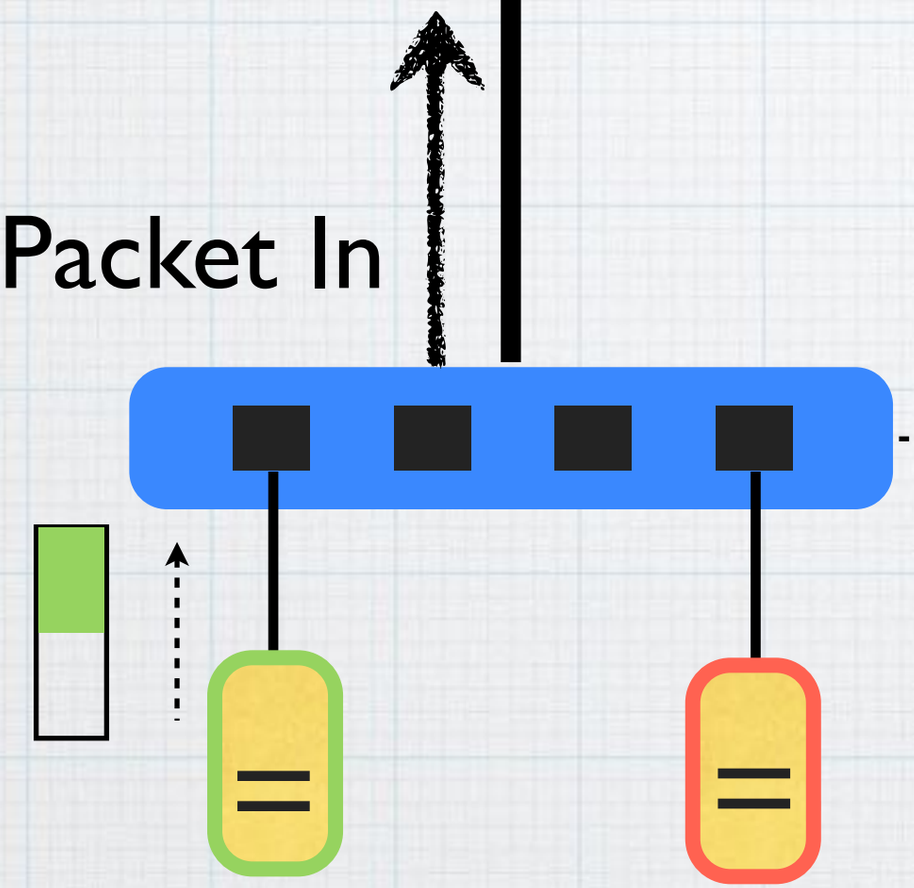


コントローラ



コントローラ

Packet In



host1

00:00:..:01

192.168.0.1

host2

00:00:..:02

192.168.0.2

Flow Table

知らないパケットが来た

```
def packet_in(dpid, message)
  # messageの中身を見て、

  # ここで何か処理をする

end
```

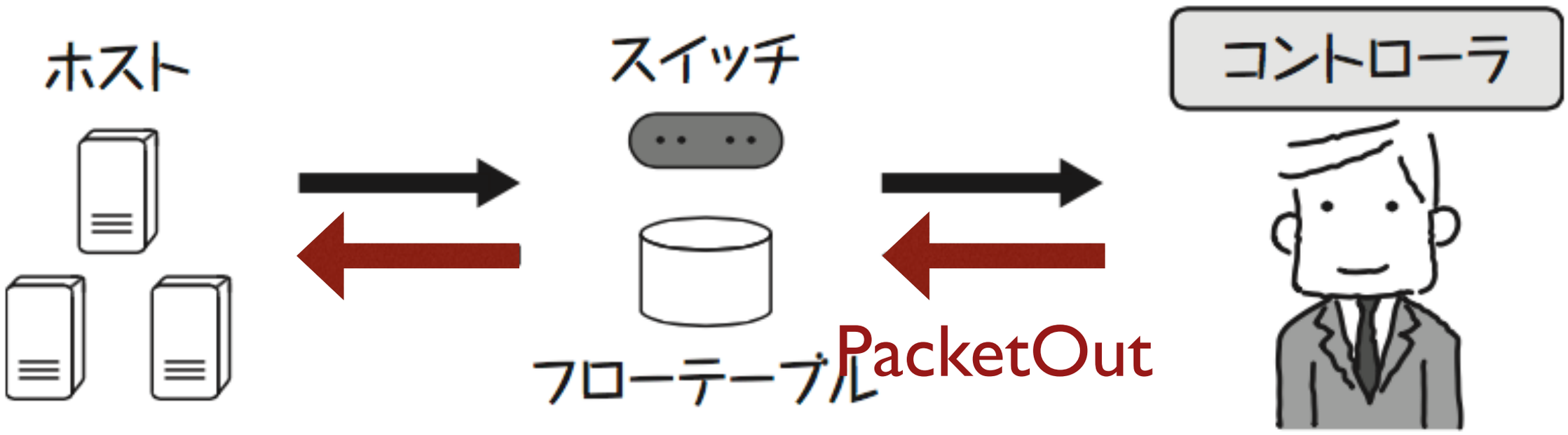
・ハンドラで PacketIn を拾う

・message = PacketIn オブジェクト

PacketOut

(PacketIn したパケットを)

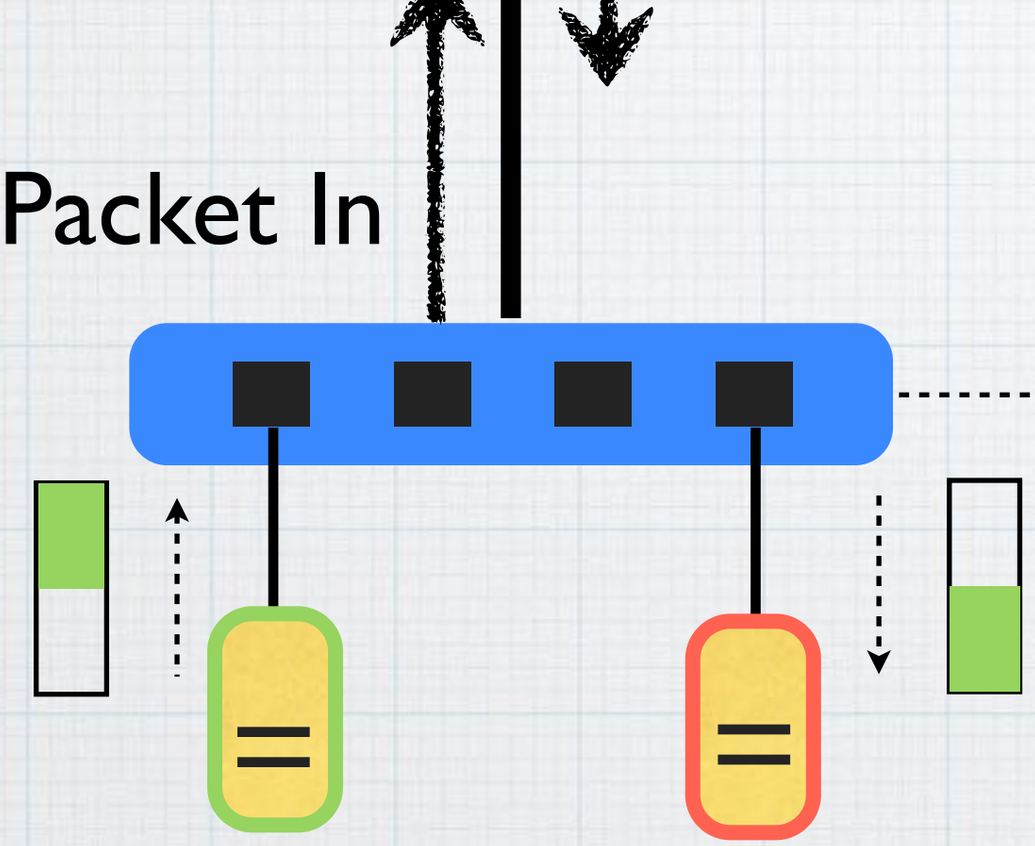
指定したポートから出す



コントローラ

Packet Out

Packet In



Flow Table

知らないパケットが来た

host1

00:00:..:01

192.168.0.1

host2

00:00:..:02

192.168.0.2

```
def packet_in(dpid, message)
  send_packet_out(
    datapath_id,
    in_port: message.in_port,
    raw_data: message.raw_data,
    actions: SendOutPort.new(1)
  )
```

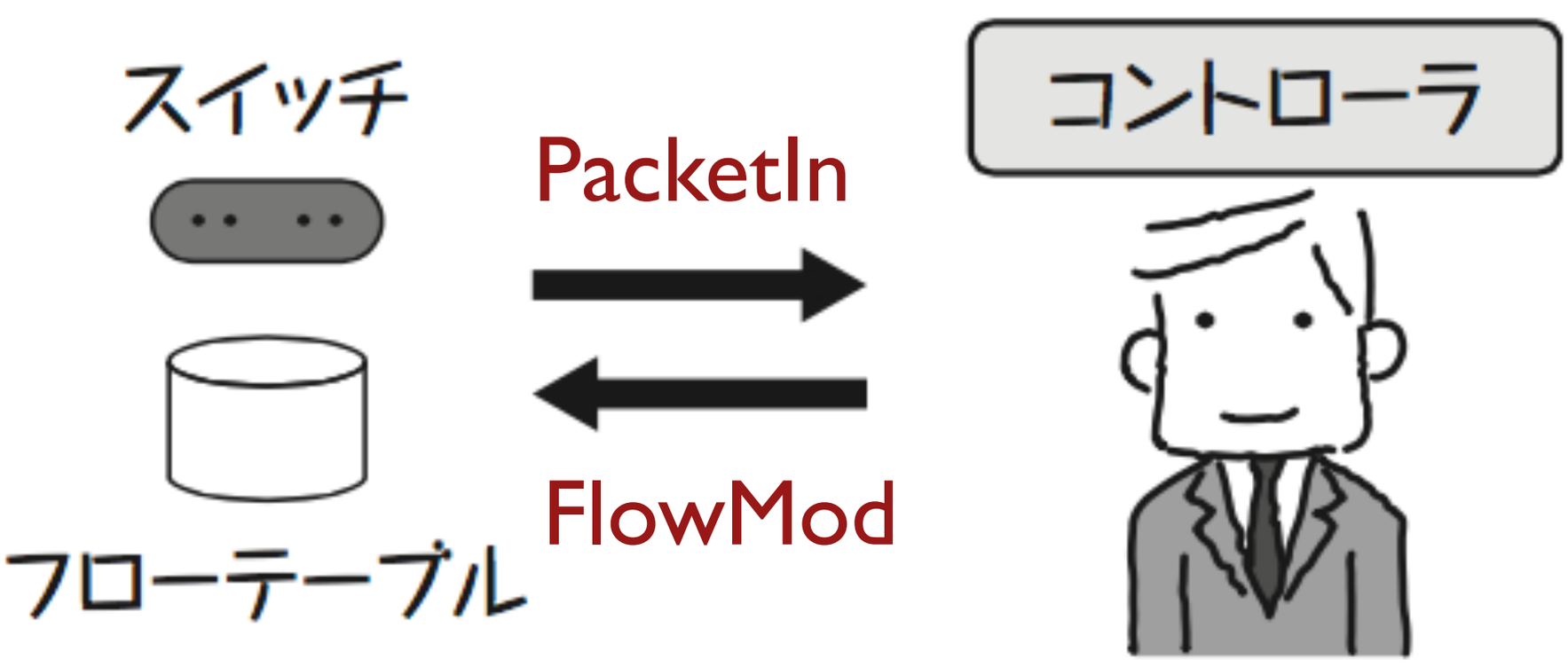
- PacketIn と同じパケットを、ポート1番から出す

```
def packet_in(dpid, message)
  send_packet_out(
    datapath_id,
    in_port: message.in_port,
    raw_data: message.raw_data,
    actions: SendOutPort.new(:flood)
  )
```

- PacketIn と同じパケットを、PacketInしたポート以外のすべてのポートから出す

cbenchとは

コントローラのマイクロベンチマーク



cbenchの動作

OpenFlow用のマイクロベンチマーク

- cbenchプロセスがスイッチのふりをしてTremaに接続し、Packet Inを送りまくる
- Tremaは決められたFlow Modを返す
- 一定時間内にたくさん返したほうが高スコア

```
send_flow_mod_add(
  datapath_id,
  match: ExactMatch.new(message),
  buffer_id: message.buffer_id,
  actions: SendOutPort.new(message.in_port + 1)
)
```

- cbenchに送るFlowMod

まとめ

- OpenFlowの基本。これが分かれば大丈夫
  - FlowMod
  - PacketIn
  - PacketOut
- Cbench はコントローラのベンチマーク
  - PacketInしてFlowModの練習