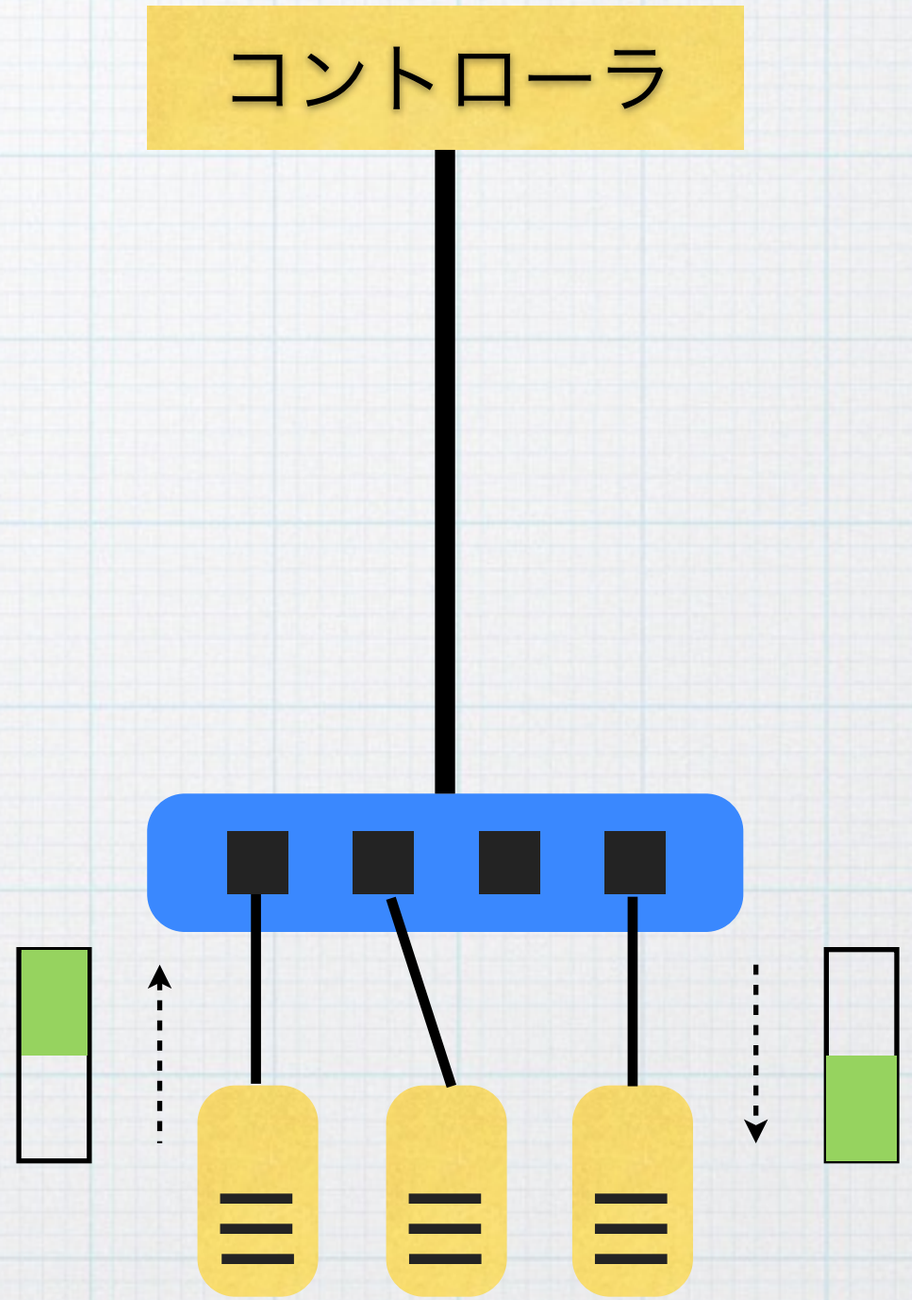
スイッチを作ろう
OpenFlow1.3編



仮想NW

ルーティング
スイッチ

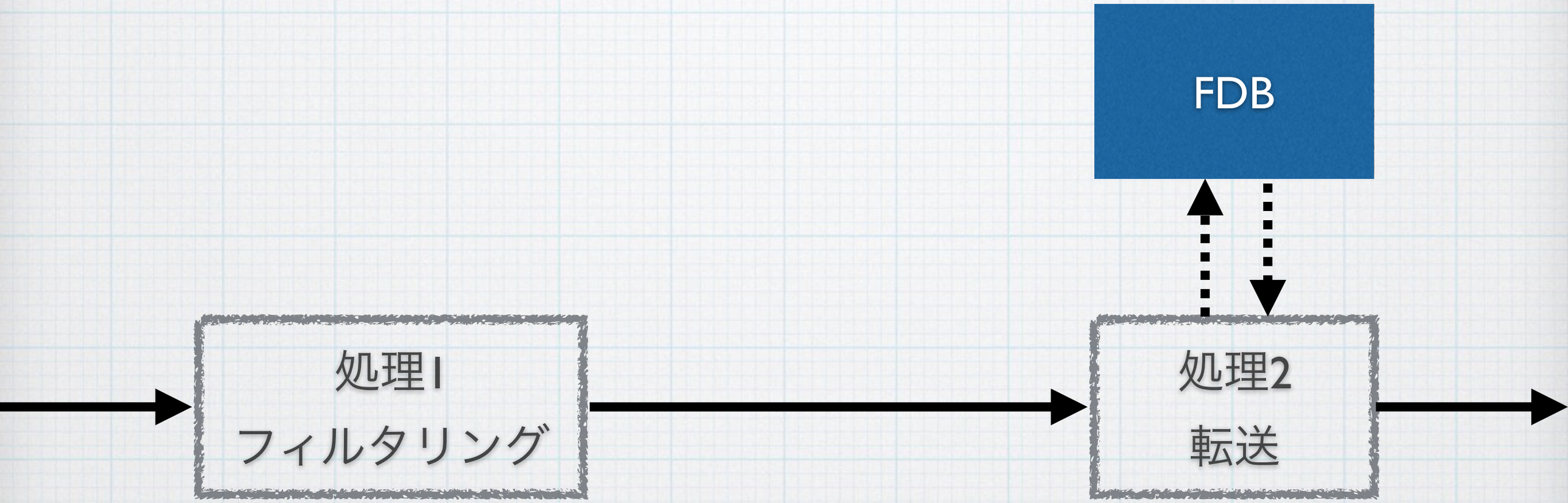
トポロジ
ディスカバリ

スイッチ

Hello
World

おさらい
OpenFlow1.0でのパケット処理

ラーニングスイッチ



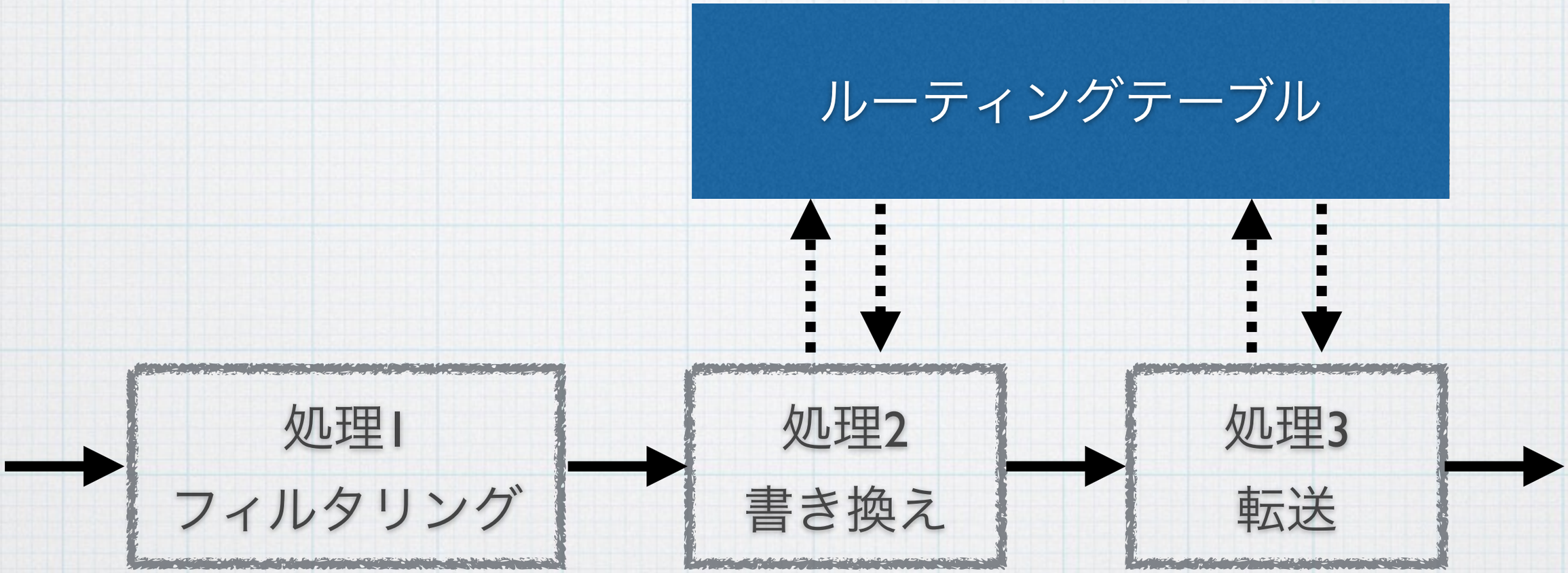
宛先 MAC =

- 802.1D/802.1Q reserved MAC
- マルチキャスト

をドロップ

ポートn番へ
or
FLOODING

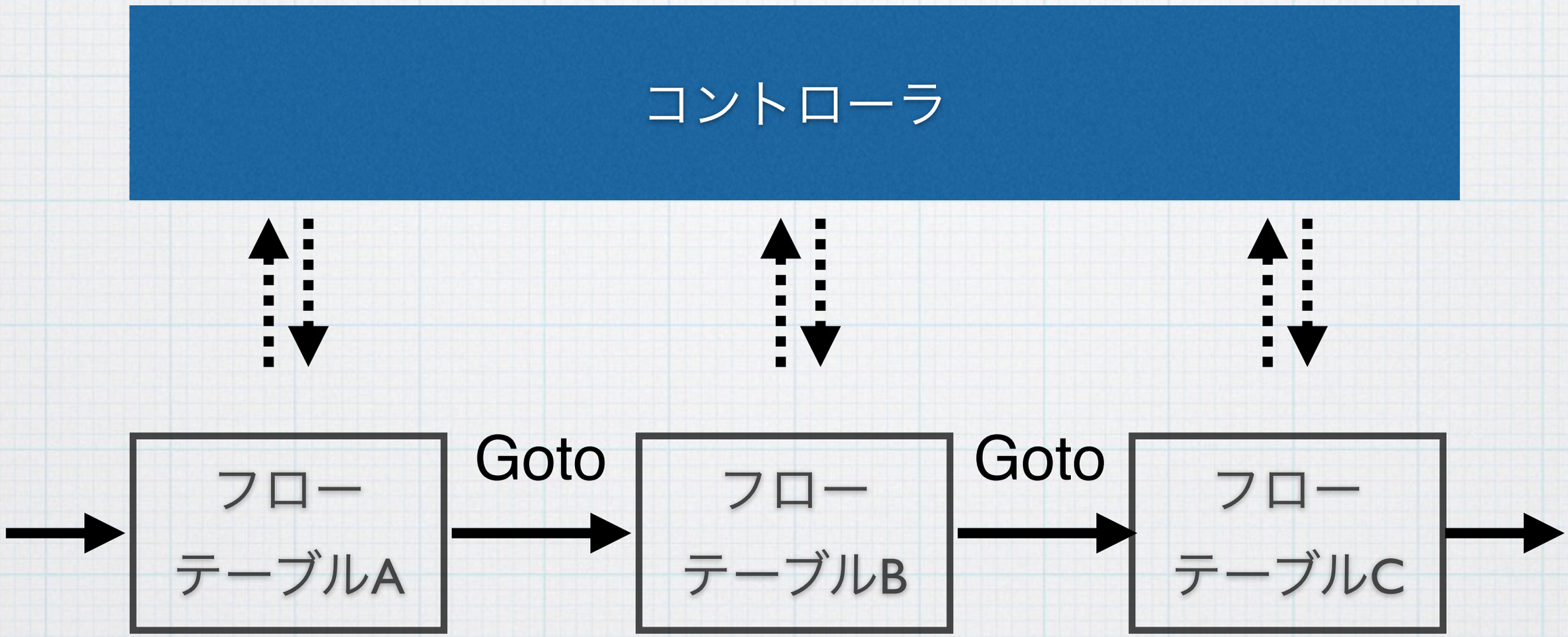
例: ルータ



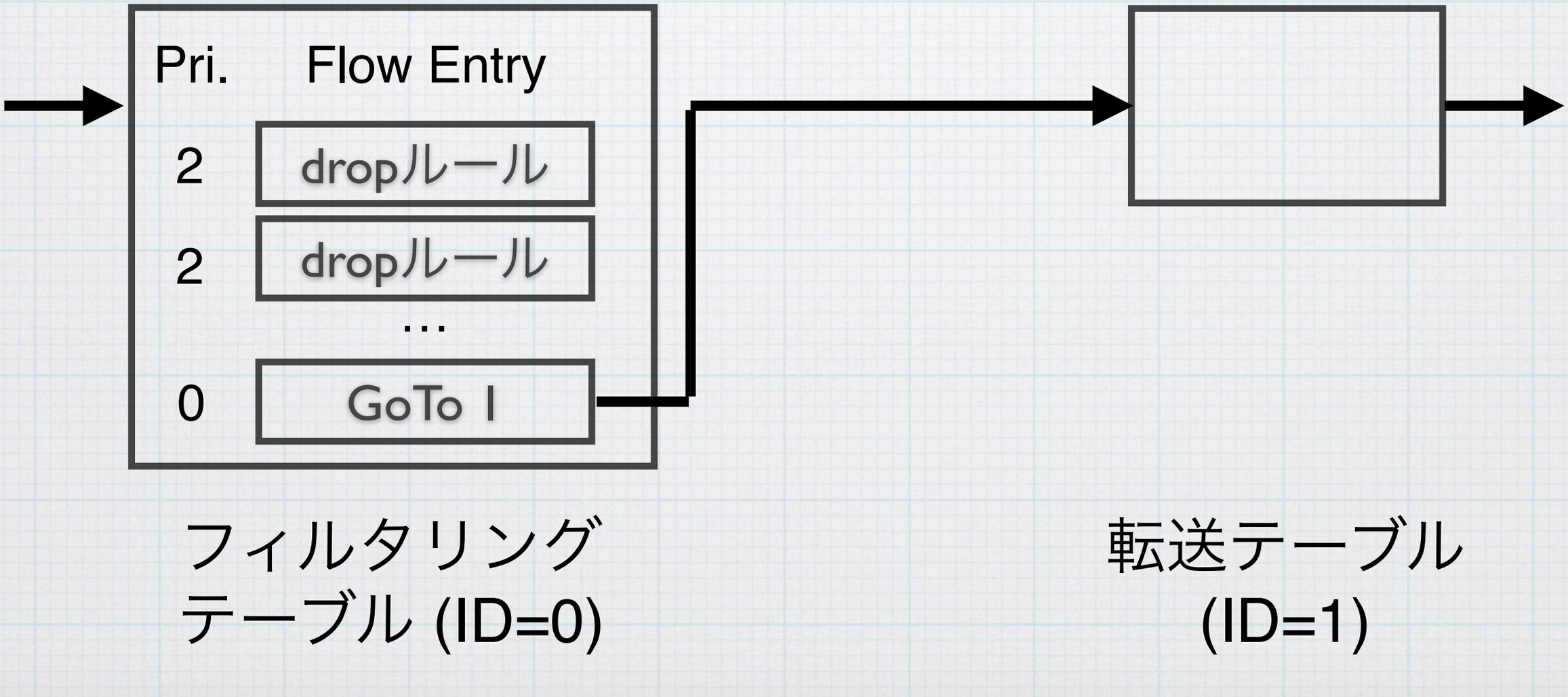
送信元/宛先
MACアドレス

OpenFlow1.3でのパケット処理

マルチプルテーブル



ラーニングスイッチ



フィルタリング
テーブル (ID=0)

転送テーブル
(ID=1)

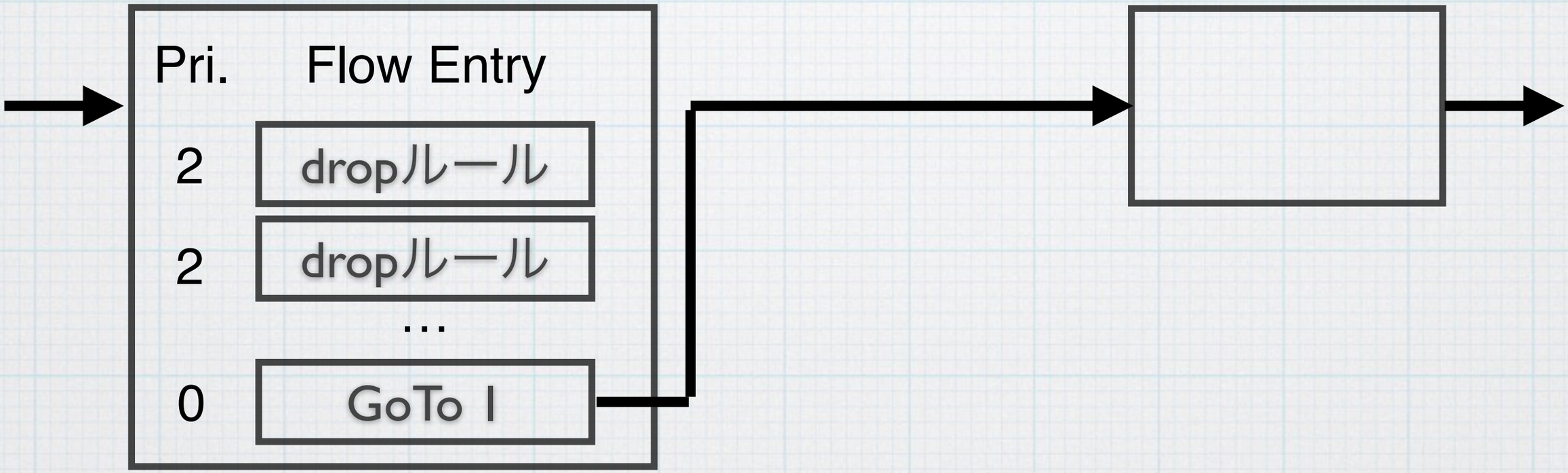
```
send_flow_mod_add(
      datapath_id,
      table_id: 0,
      idle_timeout: 0,
      priority: 2,
      match: Match.new(ether_destination_address: '01:00:5e:00:00:00',
                       ether_destination_address_mask: 'ff:ff:ff:00:00:00')
)
```

- マルチキャストを落とす
- 最初のテーブル (ID=0) で処理

```
send_flow_mod_add(
        datapath_id,
        table_id: 0,
        idle_timeout: 0,
        priority: 1,
        match: Match.new,
        instructions: GotoTable.new(1)
    )
```

- フィルタにかからなかったら、処理をテーブル1へ移行
- フィルタリング (pri=2) の後にこのルール (pri=1) にヒット

ラーニングスイッチ(再掲)



フィルタリング
テーブル (ID=0)

転送テーブル
(ID=1)

コントローラ

PacketIn

Pri.

1

FLOOD

フィルタリング
テーブル (ID=0)

転送テーブル
(ID=1)

```
send_flow_mod_add(
  datapath_id,
  table_id: 1,
  idle_timeout: 0,
  priority: 1,
  match: Match.new,
  instructions: Apply.new([SendOutPort.new(:controller),
                           SendOutPort.new(:flood)])
)
```

・パケットをフラッディング
・加えて、PacketInを起こす

コントローラ

FlowMod

Pri.

2 転送ルール

1 FLOOD

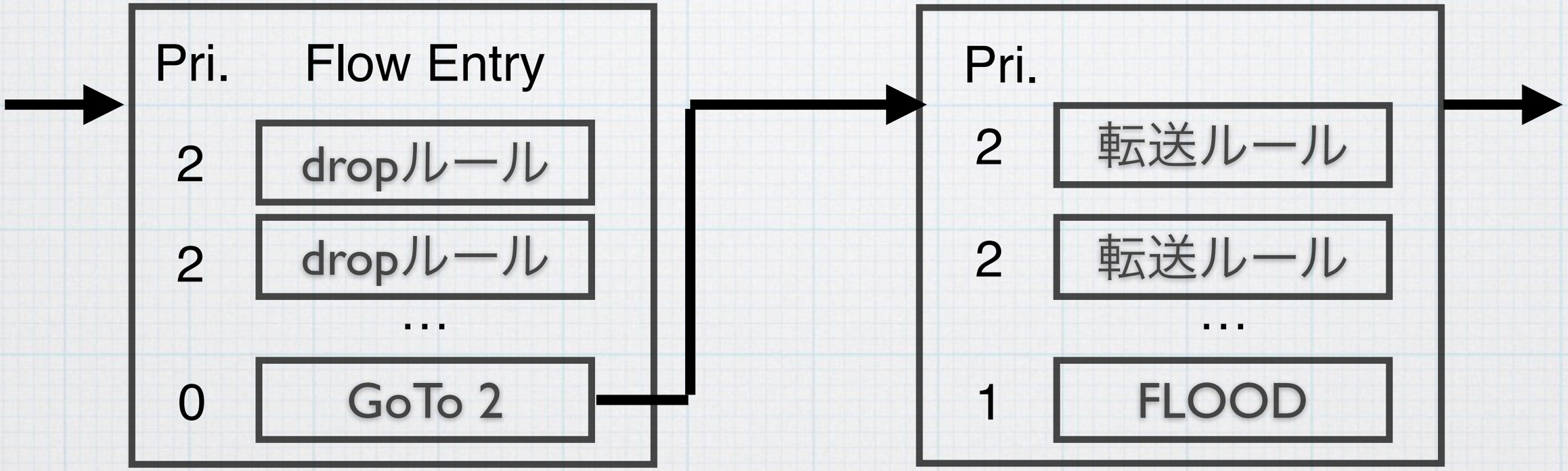
フィルタリング
テーブル (ID=0)

転送テーブル
(ID=1)

```
send_flow_mod_add(
      datapath_id,
      table_id: FORWARDING_TABLE_ID,
      idle_timeout: AGING_TIME,
      priority: 2,
      match: Match.new(ether_destination_address: packet_in.source_mac),
      instructions: Apply.new(SendOutPort.new(packet_in.in_port))
  )
```

- packet_inの送信元+ポートを
  フローエントリとして追加
- 優先度はFLOODより高くする

コントローラ



フィルタリング
テーブル (ID=0)

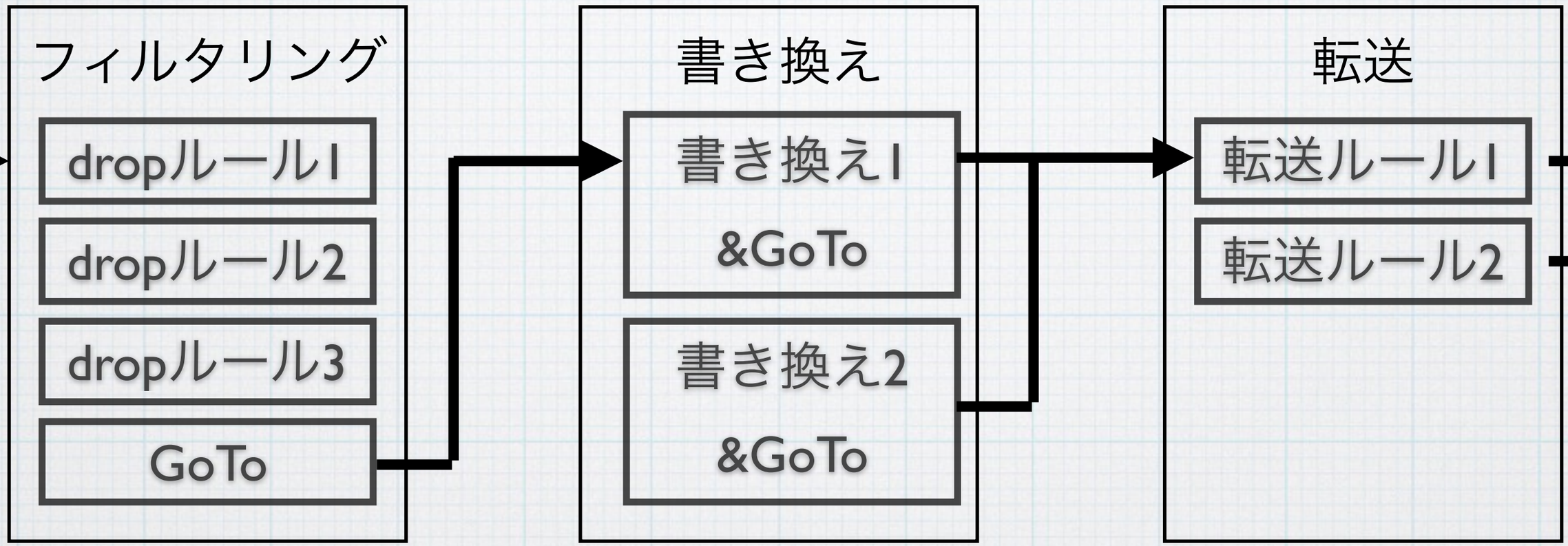
転送テーブル
(ID=1)

OpenFlow1.3の利点

OpenFlow1.0の問題点 I

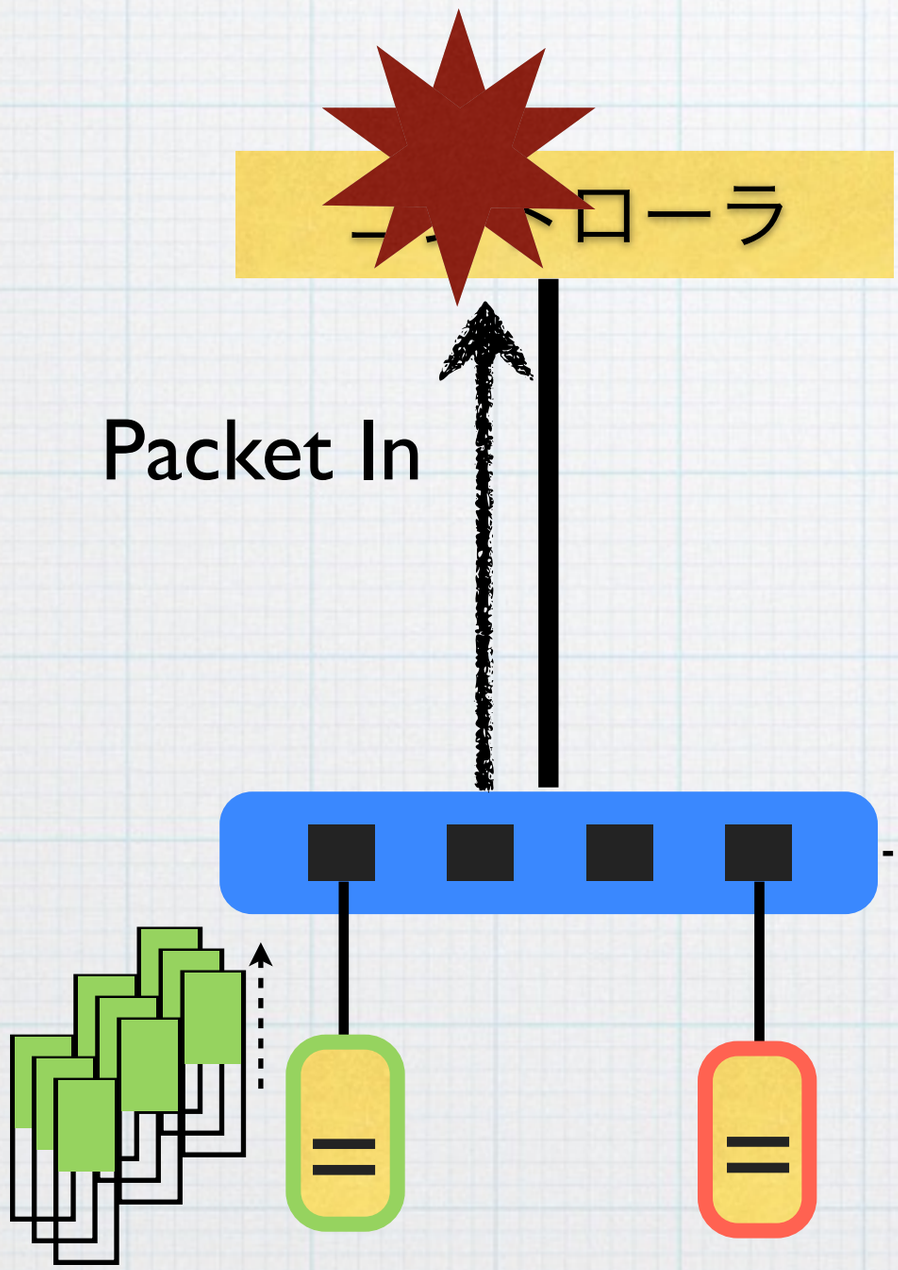
- OpenFlow1.0 では、
  フローテーブルは一つだけ
- 役割の違うエントリ
  が混在 & 混乱しがち

OpenFlow1.3では



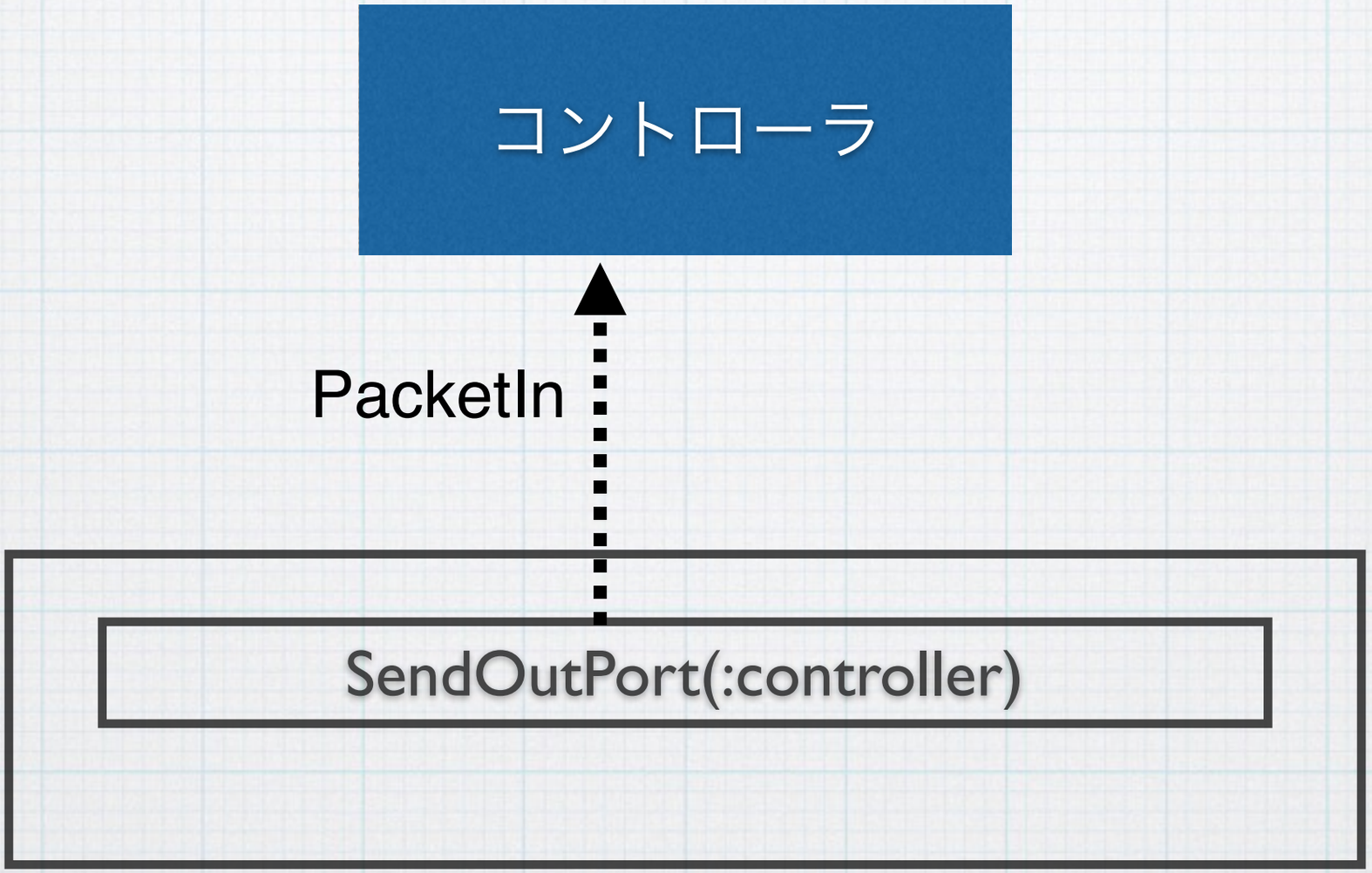
- 分けてスッキリ!
- 処理をパイプライン化できるので性能が上がる

OpenFlow1.0の問題点 2



PacktInが大量に到着すると
コントローラがパンク

OpenFlow1.3では



- 明示的に指定しない限り、PacketInは起こらない

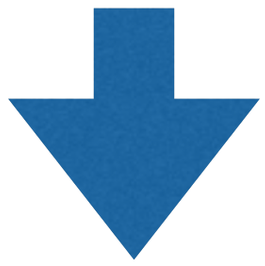
アクションと
インストラクション

- アクション
  - パケットの書き換えと転送 (OF1.0と同じ)
  - OF1.3で種類が増えました
- インストラクション
  - テーブルの移動 (GoTo)
  - アクションの実行方法の指定

```
send_flow_mod_add(
      datapath_id,
      table_id: 1,
      idle_timeout: 0,
      priority: 1,
      match: Match.new,
        instructions: GotoTable.new(2)
)
```

- 処理をテーブル2へ移行

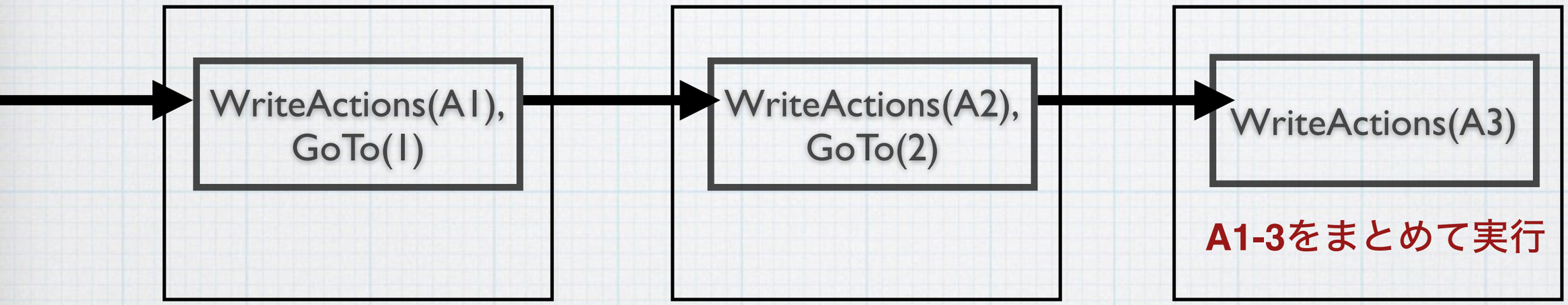
```
send_flow_mod_add(
  datapath_id,
    ...
  actions: SendOutPort.new(1)
)
```



```
send_flow_mod_add(
  datapath_id,
    ...
  instructions: Apply.new(SendOutPort.new(1))
)
```

アクションセット

[] [A1] [A1,A2]



- パケットに関連付いたアクション集合
- WriteActionsでアクションを追加
- GoToを含まないエントリにマッチしたら実行

まとめ

- OpenFlow 1.3 版スイッチの仕組み
- マルチプルテーブルの使い方
- アクションとインストラクション

レポート課題

- OpenFlow1.3 版スイッチの動作について、trema dump_flows の出力を混じえながら各ステップごとに説明しよう

- 実行のしかた

```
$ trema run lib/learning_switch13.rb
    --openflow13 -c trema.conf
```