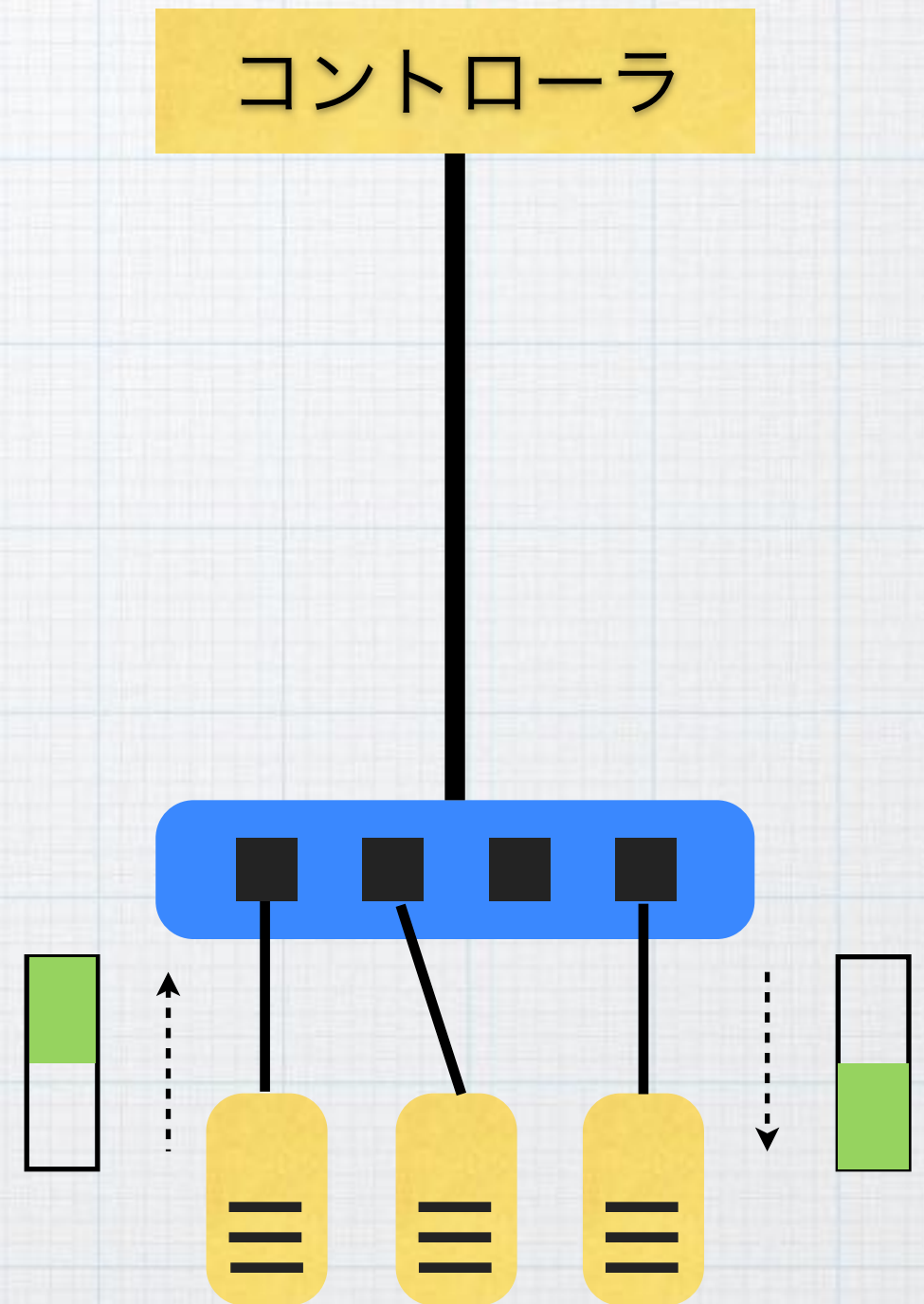
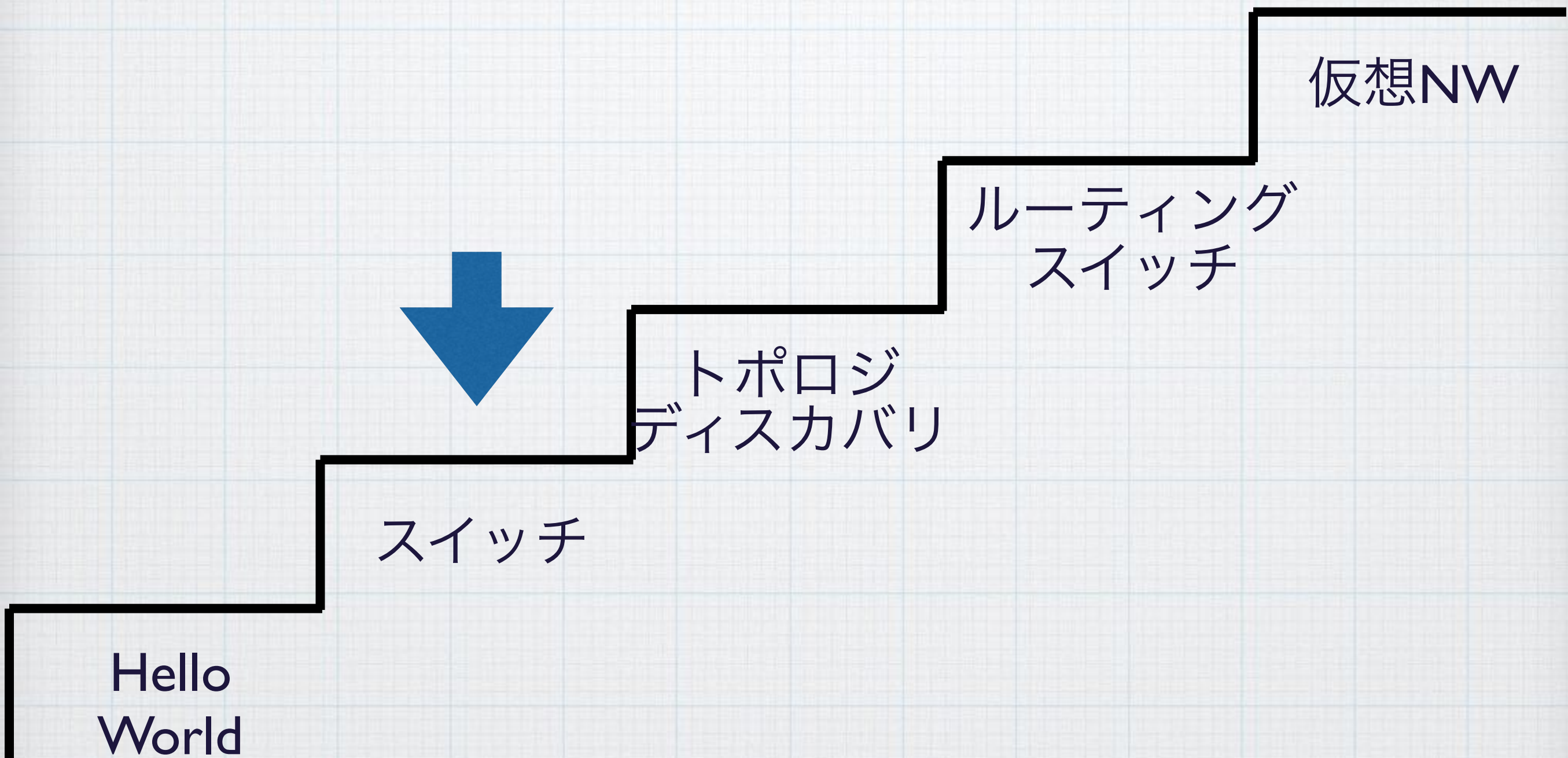


スイッチを作ろう

OpenFlow1.3編

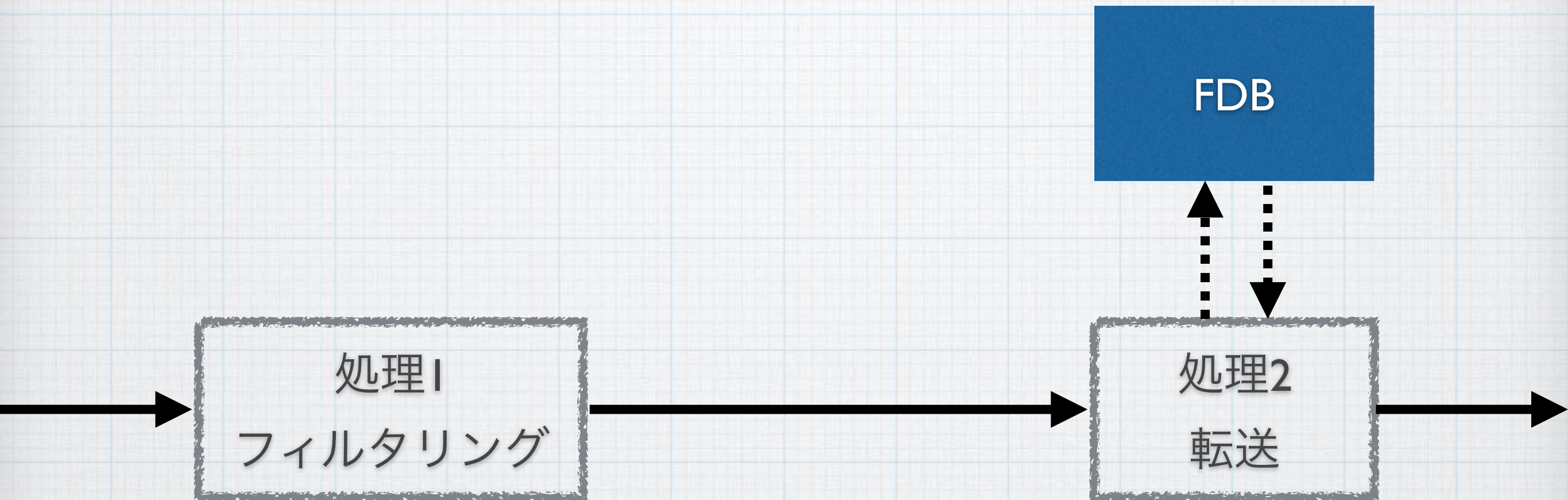




おさらい

OpenFlow1.0でのパケット処理

ラーニングスイッチ

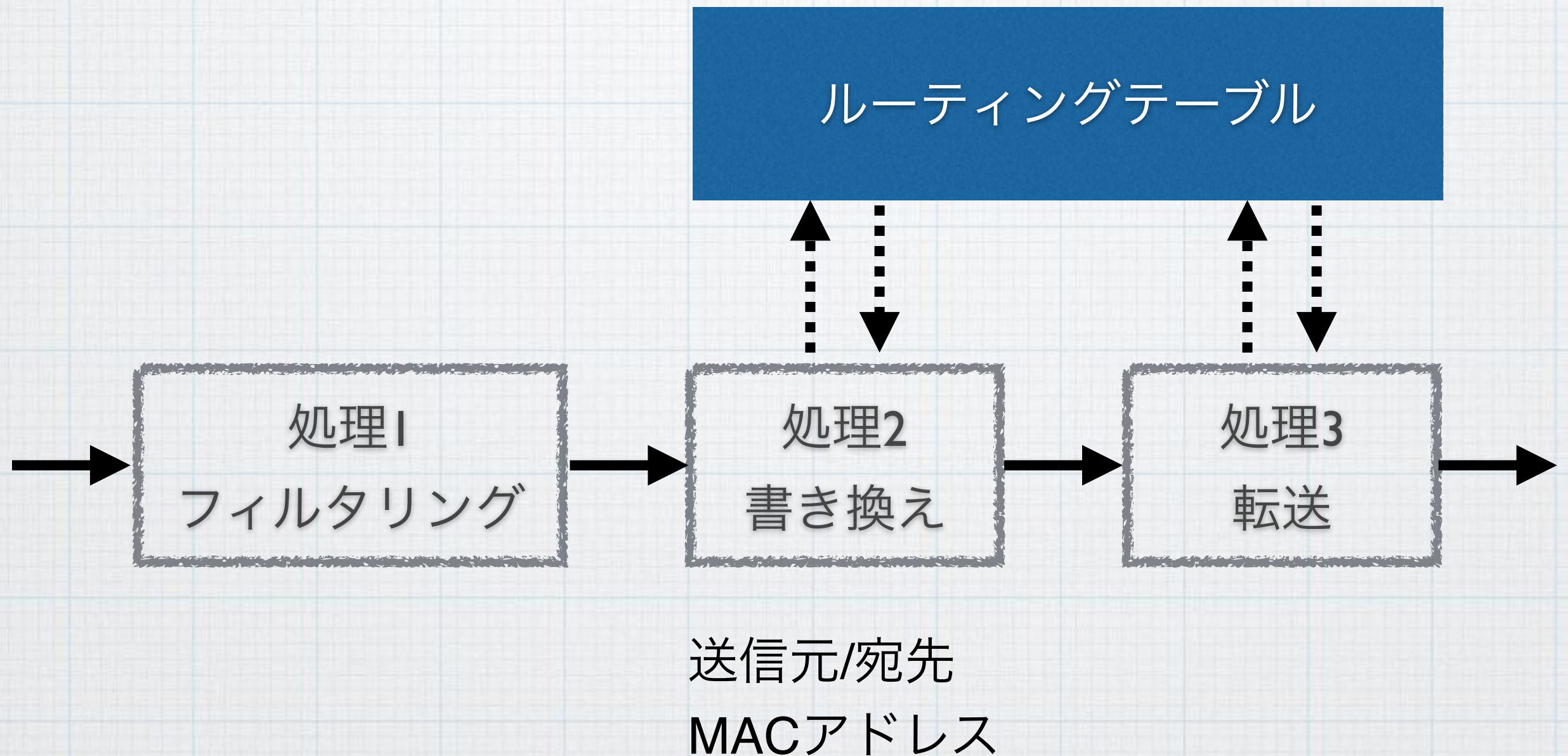


宛先 MAC =

- 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

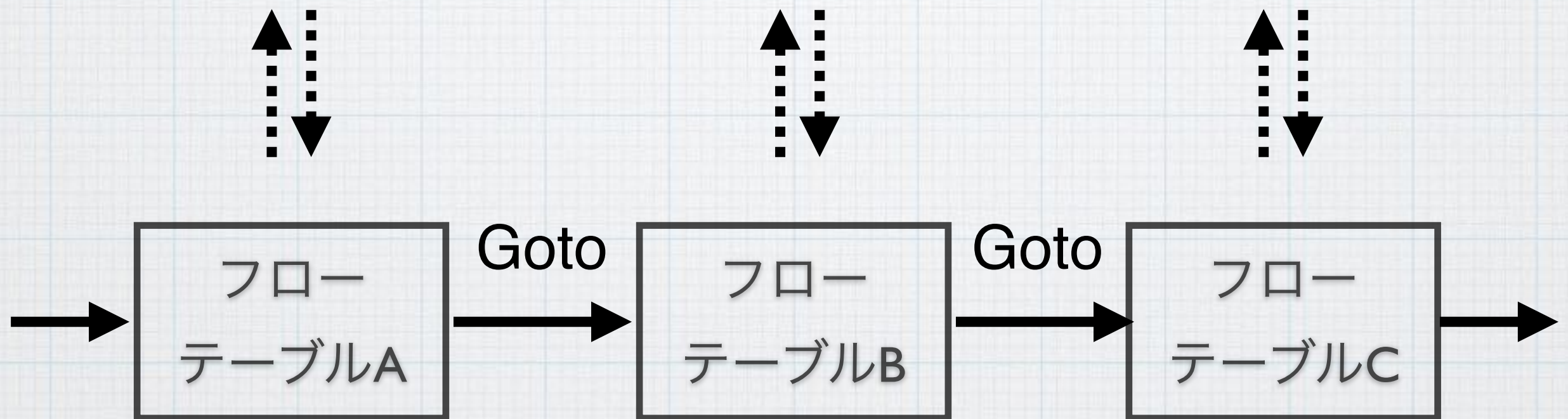
例: ルータ



OpenFlow1.3でのパケット処理

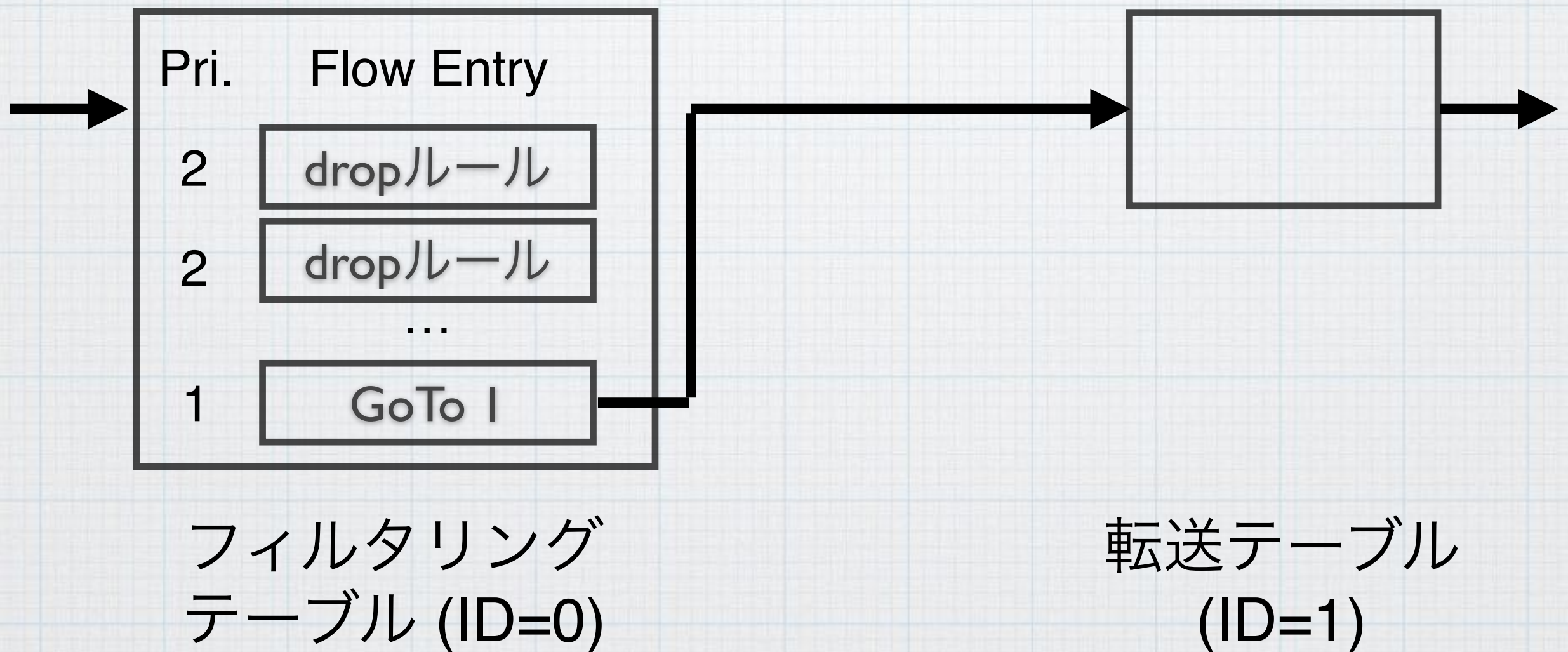
マルチプルテーブル

コントローラ



処理ごとにテーブルを分けるとデバッグしやすくなる

ラーニングスイッチ



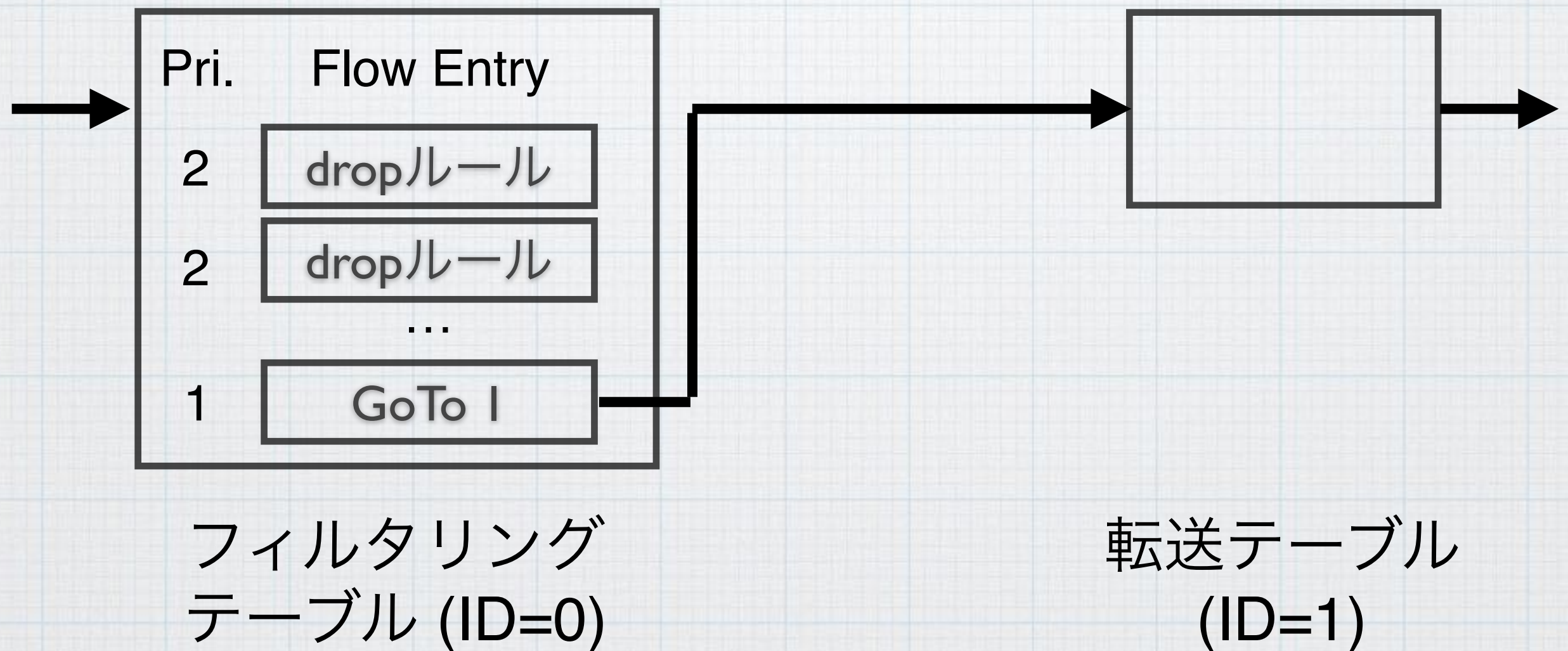

```
send_flow_mod_add(  
    datapath_id,  
    table_id: 0,  
    idle_timeout: 0,  
    priority: 2,  
    match: Match.new(ether_destination_address: '01:00:5e:00:00:00',  
                     ether_destination_address_mask: 'ff:ff:ff:00:00:00')  
)
```

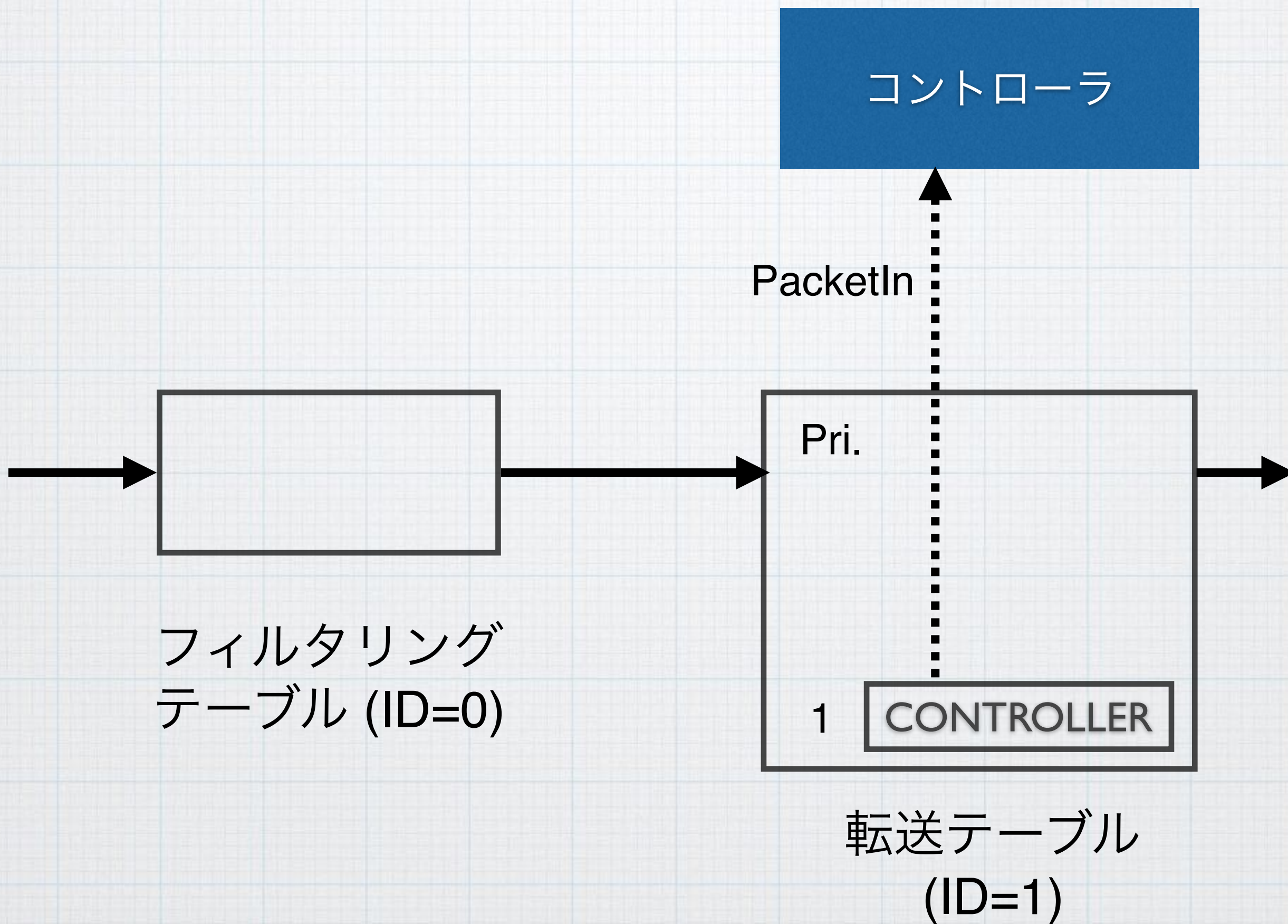
- マルチキャストを落とす
- 最初のテーブル (ID=0) で処理

```
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)



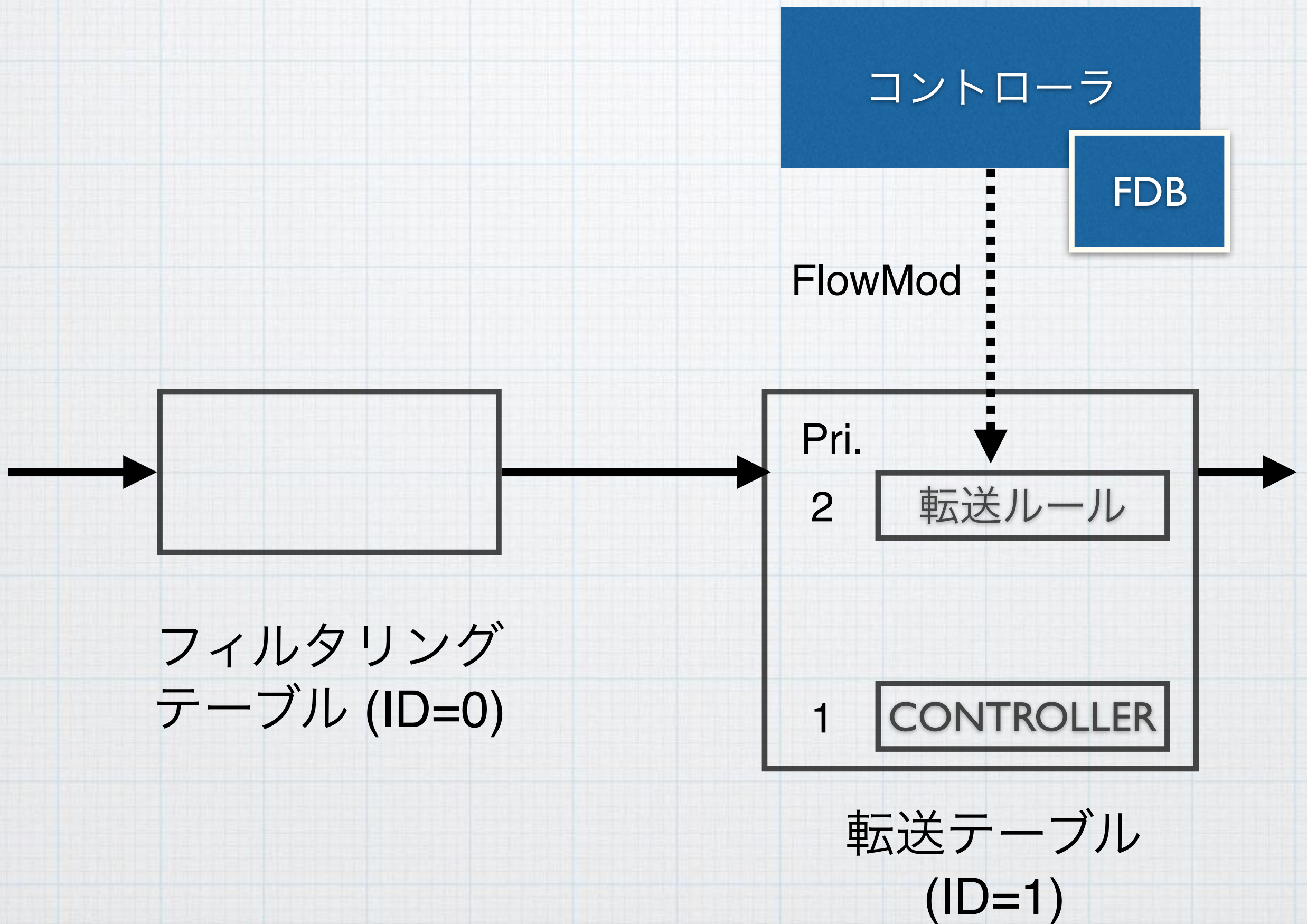



```
send_flow_mod_add(  
    datapath_id,  
    table_id: 1,  
    idle_timeout: 0,  
    priority: 1,  
    match: Match.new,  
    instructions: Apply.new(SendOutPort.new(:controller))  
)
```

- 明示的にPacketInを起こす
- OpenFlow1.3ではデフォルトドロップ

デフォルトドロップ (OpenFlow 1.3)

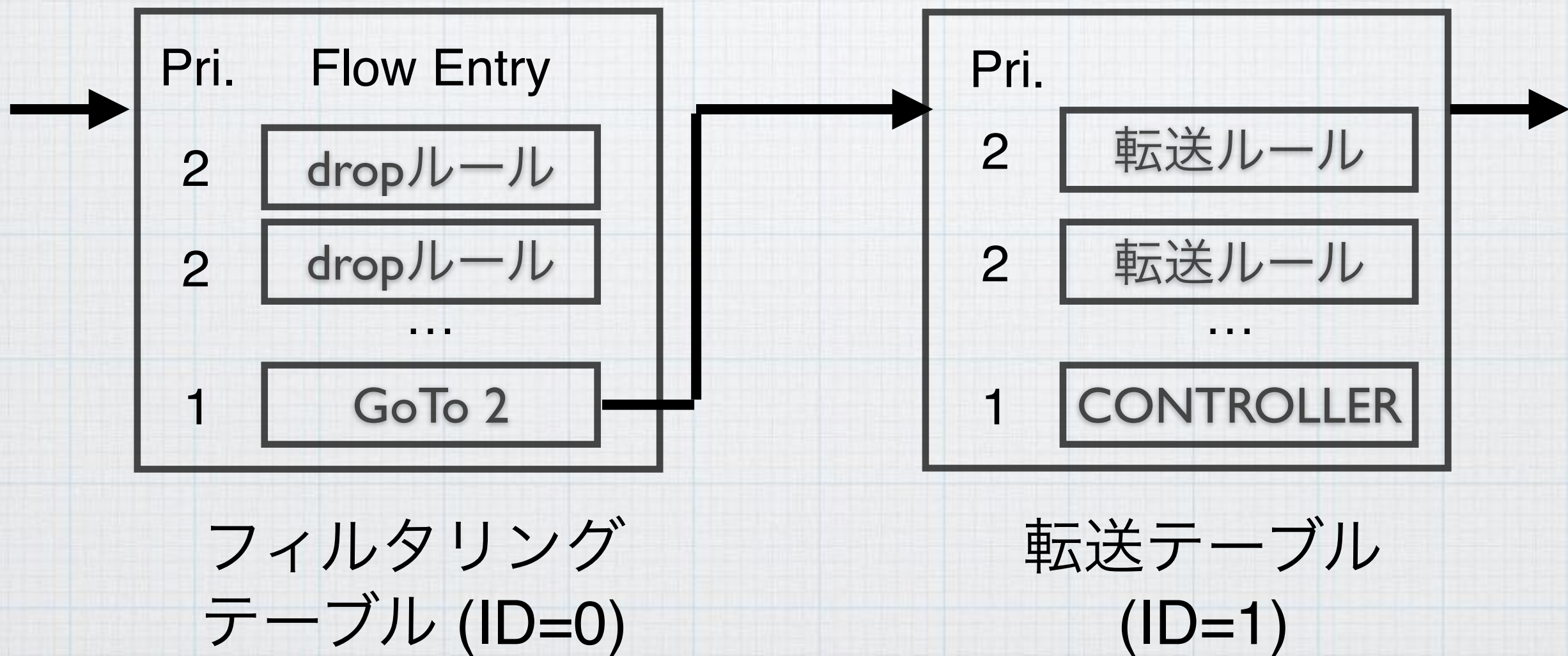
- OpenFlow 1.0はデフォルト PacketIn
- フィルタ用フローエントリを打ち込むより前にパケットが大量に到着すると、最悪コントローラがパンク
- そこでOpenFlow 1.3ではPacketInを明示的に起こすように修正された



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

- 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PacketInより高い

コントローラ



まとめ

ラーニングスイッチをOpenFlow1.3化

- マルチプルテーブルでデバッグしやすく
- GotoTableインストラクションで
テーブル (処理ステージ) を推移
- デフォルトでパケットをドロップ
万が一の大量のPacketInが防げる