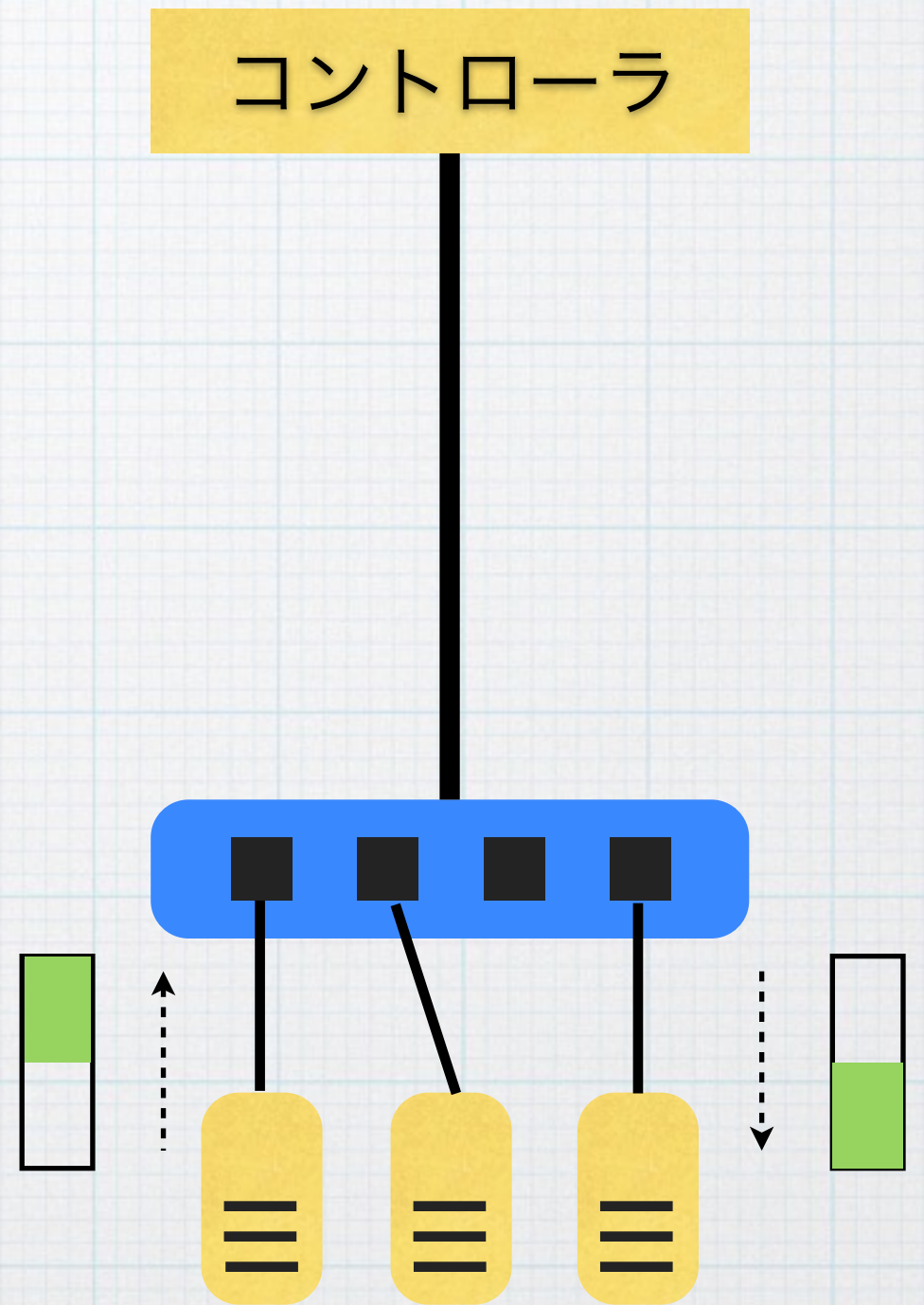
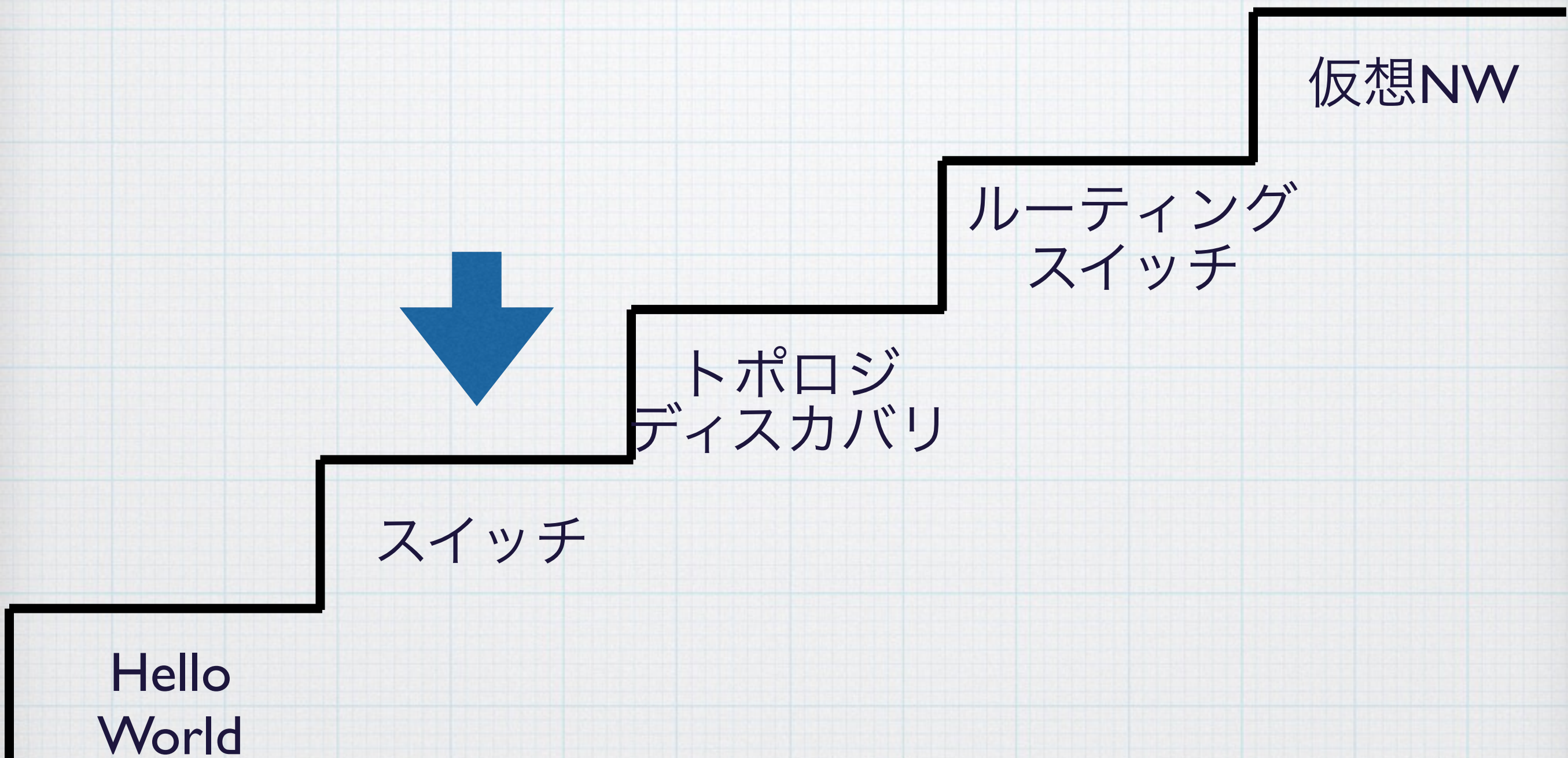


# スイッチを作ろう







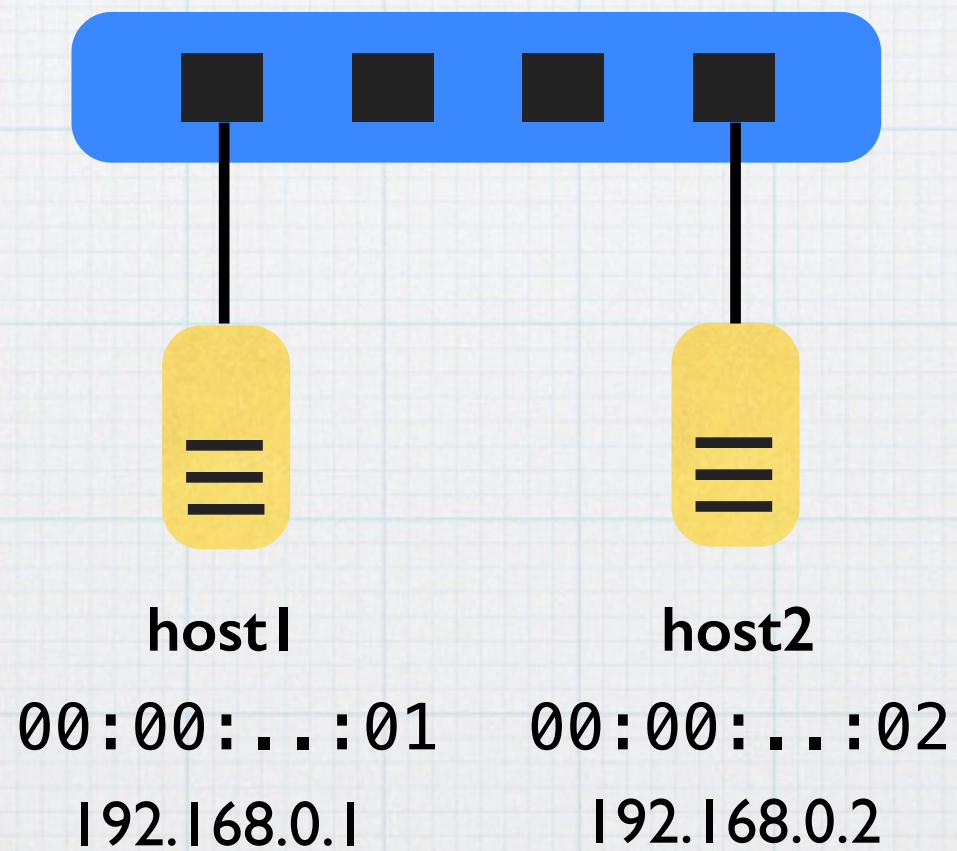
# 普通のスイッチの仕組み



## Forwarding DB

00:00:00:00:00:01 → 1

00:00:00:00:00:02 → 4



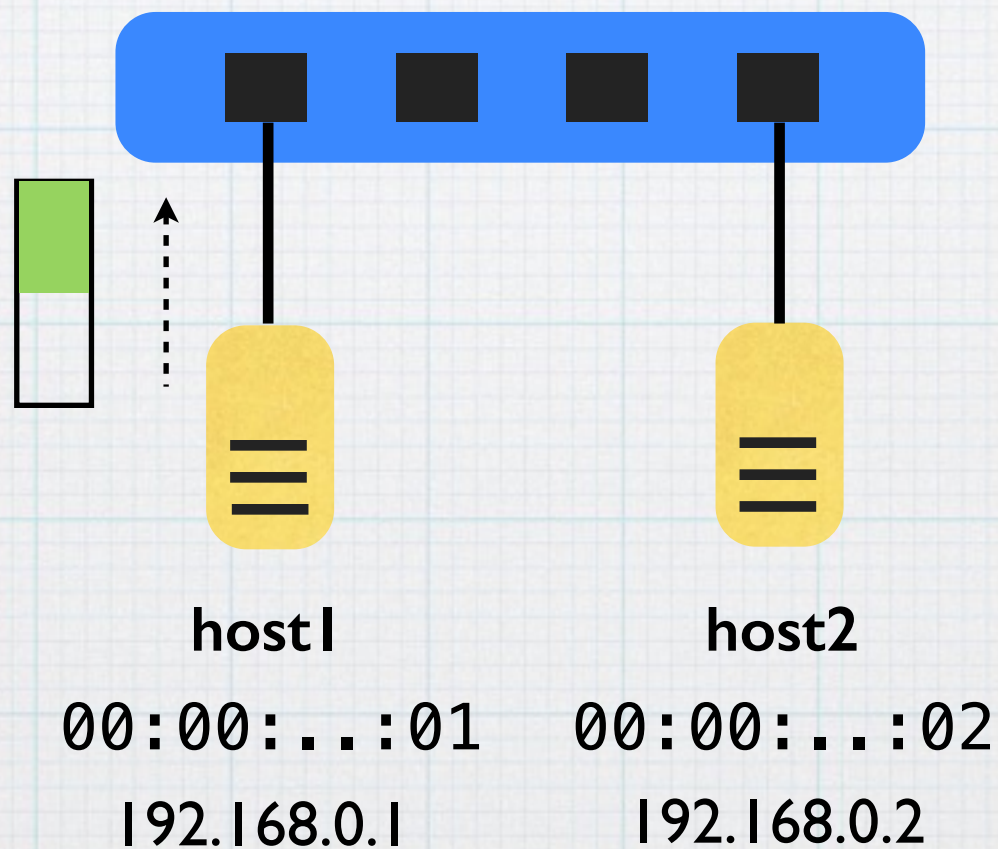


## Forwarding DB

00:00:00:00:00:01 → 1

00:00:00:00:00:02 → 4

宛先 = host2

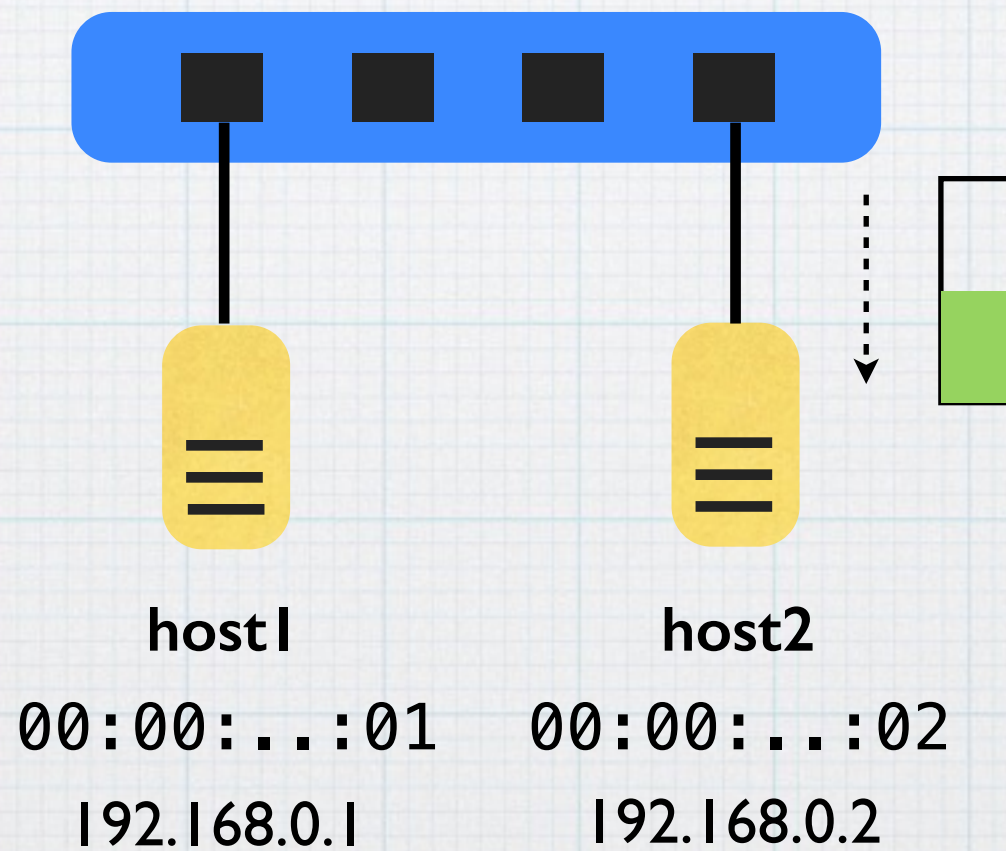




## Forwarding DB

00:00:00:00:00:01 → 1

00:00:00:00:00:02 → 4

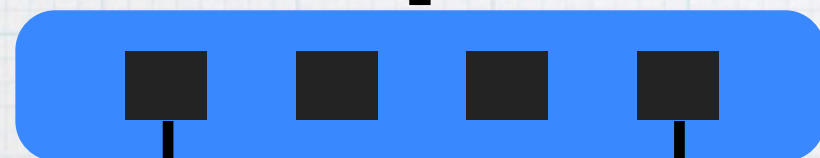


# OpenFlowによる スイッチの仕組み

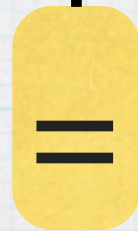
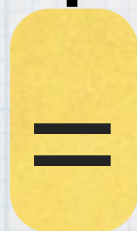


コントローラ

Forwarding DB



Flow Table



host1

host2

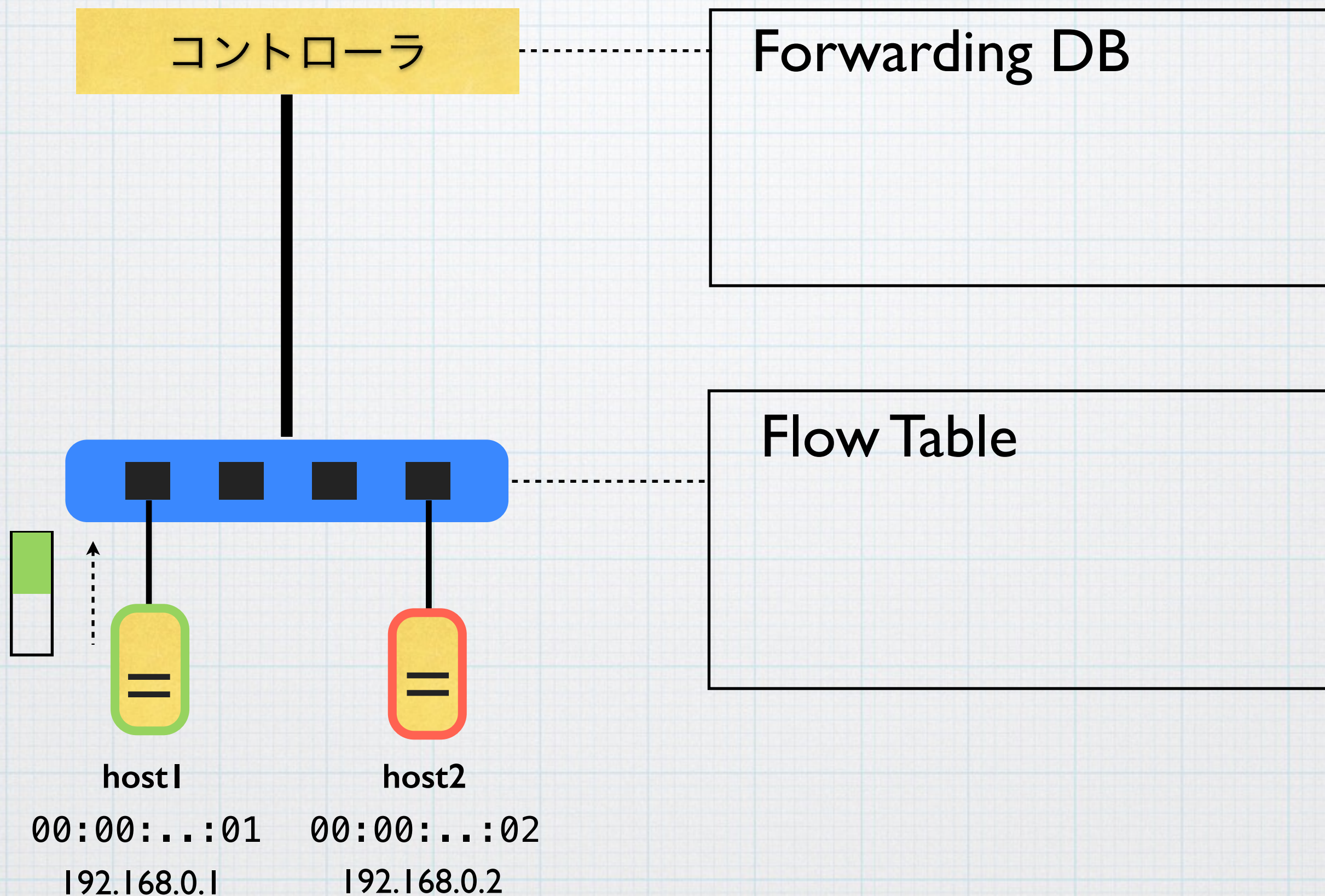
00:00:::01

00:00:::02

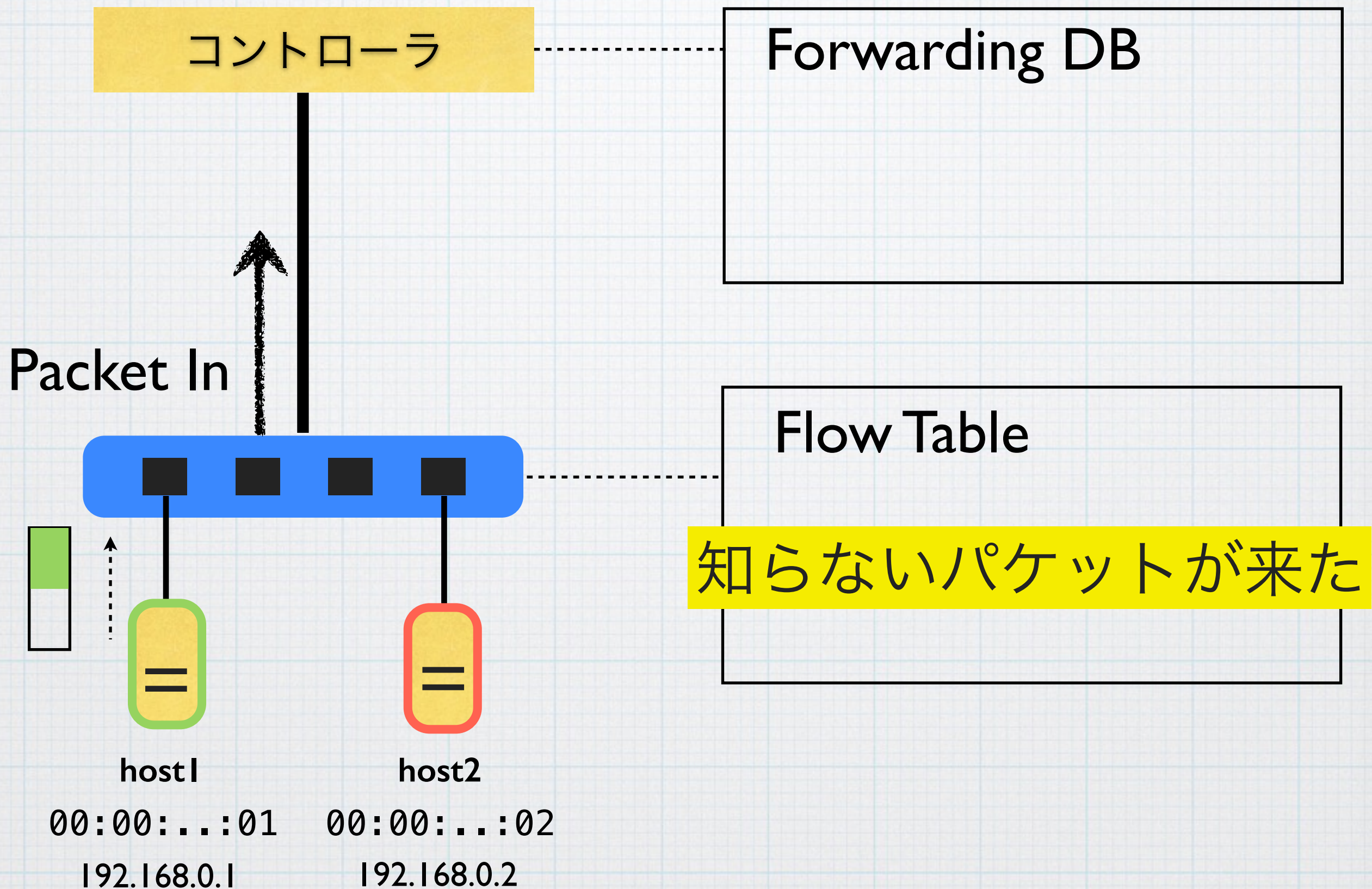
192.168.0.1

192.168.0.2

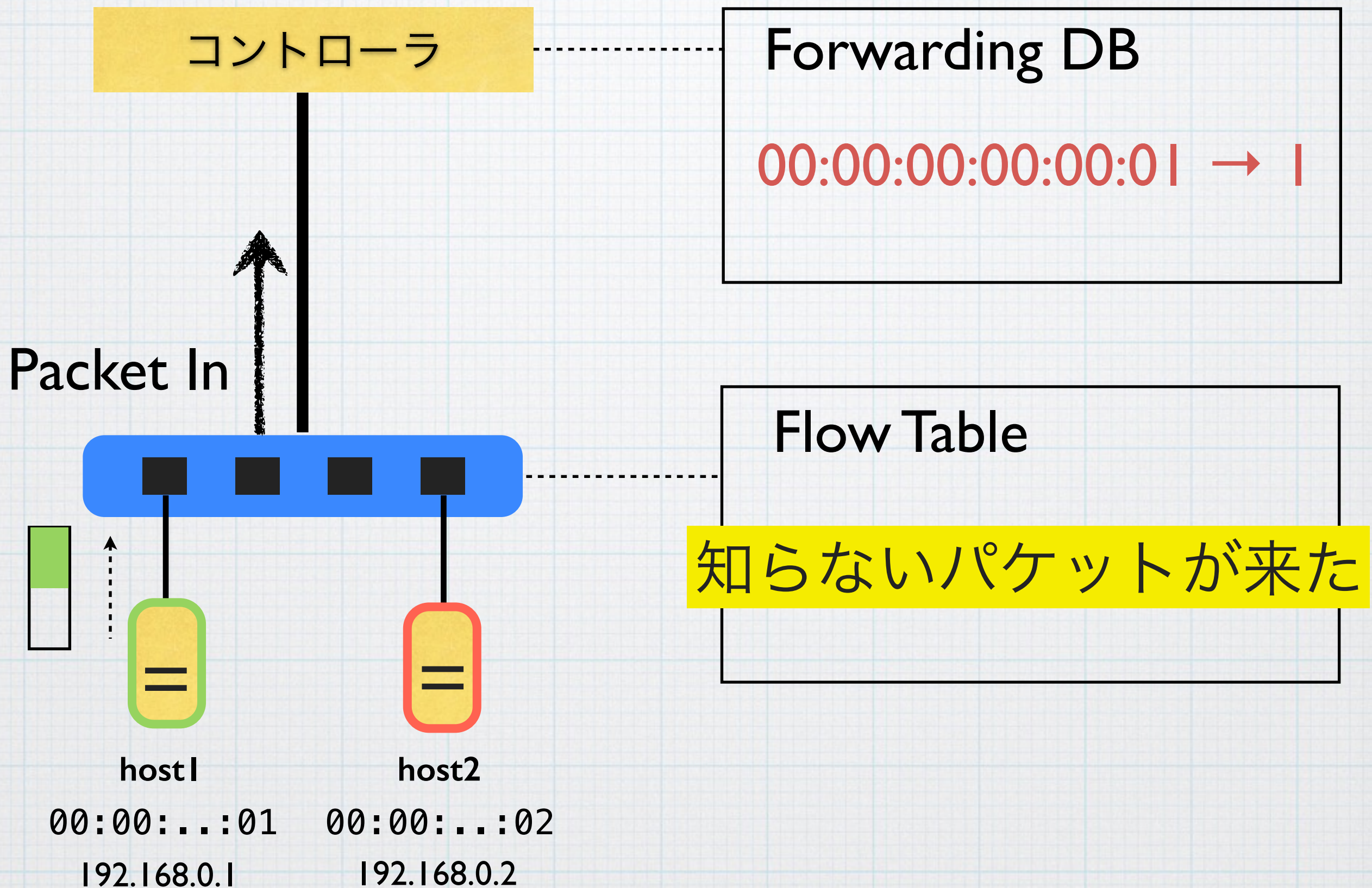














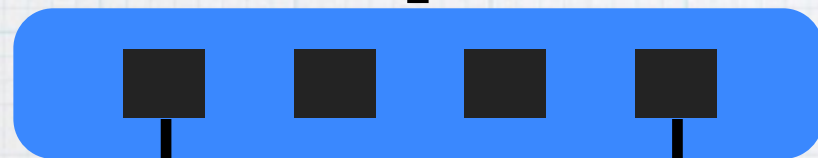
コントローラ

Forwarding DB

00:00:00:00:00:00:0 | → |

Packet Out  
(FLOOD)

Flow Table



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2



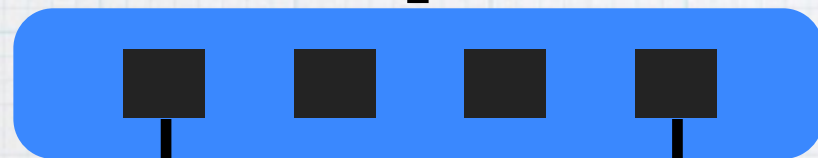
コントローラ

Forwarding DB

00:00:00:00:00:0 | → |

Packet Out  
(FLOOD)

Flow Table



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2



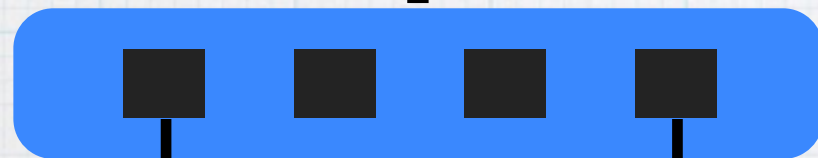


コントローラ

Forwarding DB

00:00:00:00:00:0 | → |

Flow Table



host1

host2

00:00:..:01

00:00:..:02

192.168.0.1

192.168.0.2

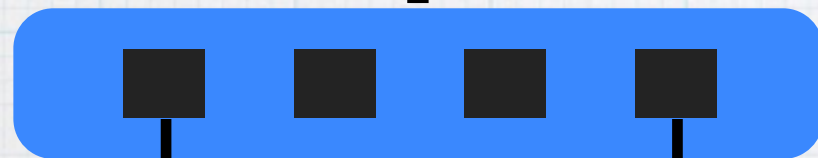


コントローラ

Forwarding DB

00:00:00:00:00:0 | → |

Flow Table



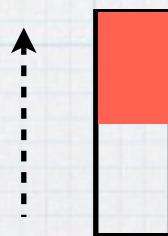
host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2





コントローラ

Forwarding DB

00:00:00:00:00:00:01 → 1  
00:00:00:00:00:00:02 → 4

Packet In

Flow Table

知らないパケットが来た



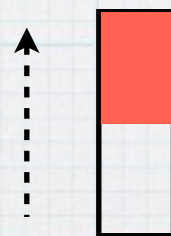
host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2





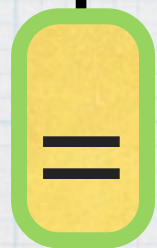
コントローラ

Forwarding DB

00:00:00:00:00:00:01 → |  
00:00:00:00:00:00:02 → 4

Packet In

Flow Table



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2





コントローラ

Flow Mod

Forwarding DB

00:00:00:00:00:00:01 → 1  
00:00:00:00:00:00:02 → 4

Flow Table

 → 1



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2



コントローラ

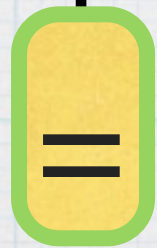
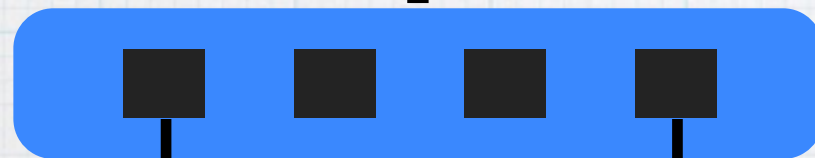
Forwarding DB

00:00:00:00:00:01 → 1  
00:00:00:00:00:02 → 4

Packet Out  
(port = 1)

Flow Table

 → 1



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2



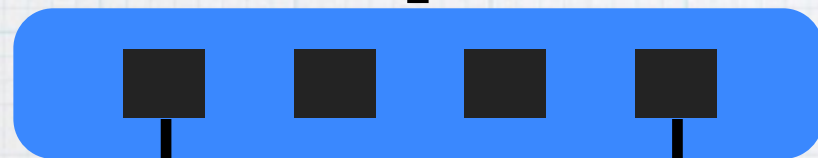
コントローラ

Forwarding DB

00:00:00:00:00:01 → 1  
00:00:00:00:00:02 → 4

Flow Table

 → 1



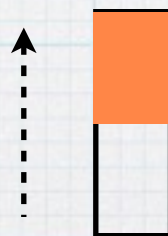
host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2





コントローラ

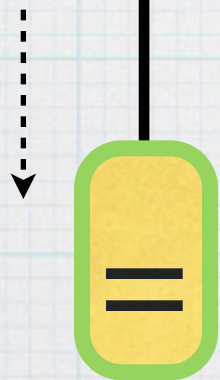
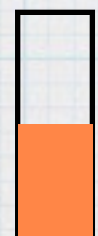
Forwarding DB

00:00:00:00:00:01 → 1  
00:00:00:00:00:02 → 4

Packet In  
しない

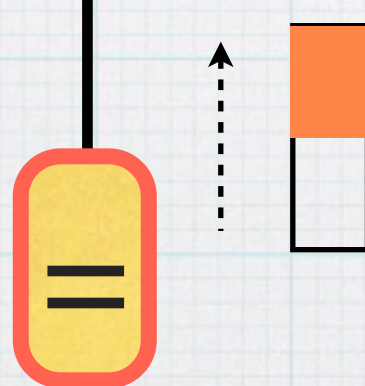
Flow Table

 → 1



host1

00:00:...:01  
192.168.0.1

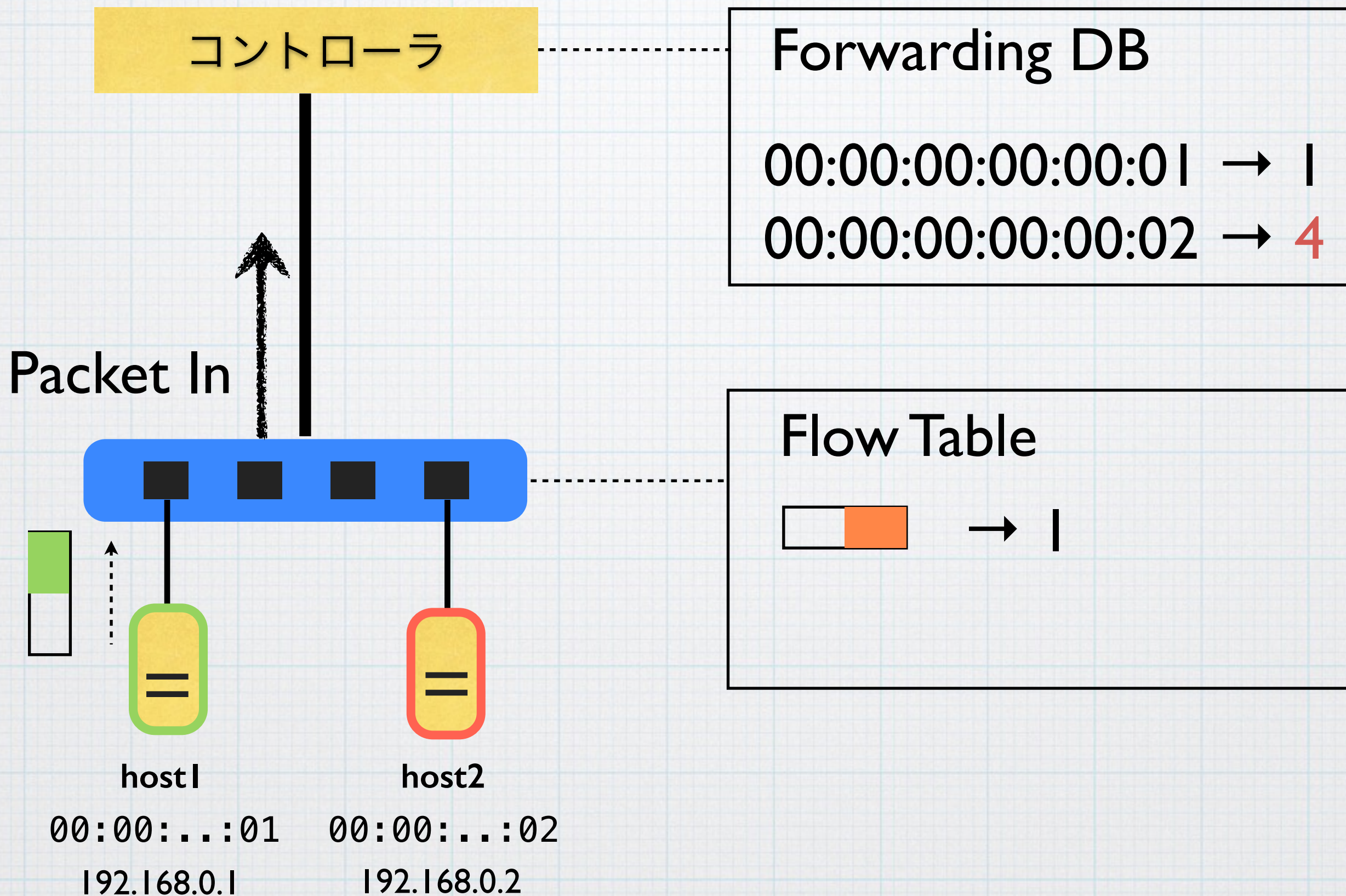


host2

00:00:...:02  
192.168.0.2









コントローラ

Forwarding DB

Flow Mod  
Packet Out

00:00:00:00:00:00:01 → 1  
00:00:00:00:00:00:02 → 4

Flow Table

 → 1



host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2





コントローラ

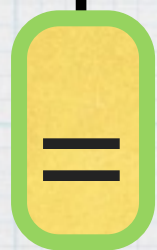
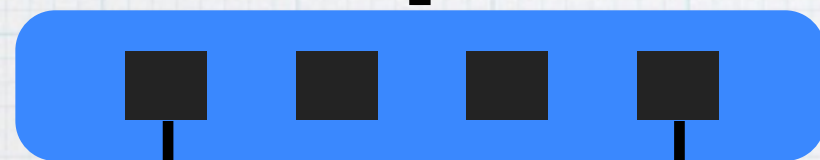
Forwarding DB

Flow Mod  
Packet Out

00:00:00:00:00:01 → 1  
00:00:00:00:00:02 → 4

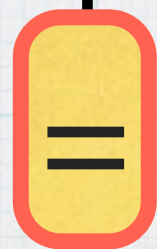
Flow Table

 → 1  
 → 4



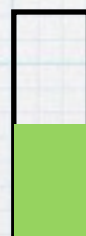
host1

00:00:...:01  
192.168.0.1



host2

00:00:...:02  
192.168.0.2







コントローラ

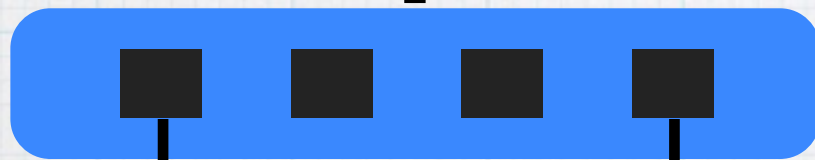
Forwarding DB

00:00:00:00:00:01 → 1  
00:00:00:00:00:02 → 4

Packet In  
しない

Flow Table

 → 1  
 → 4



host1

00:00:...:01  
192.168.0.1



host2

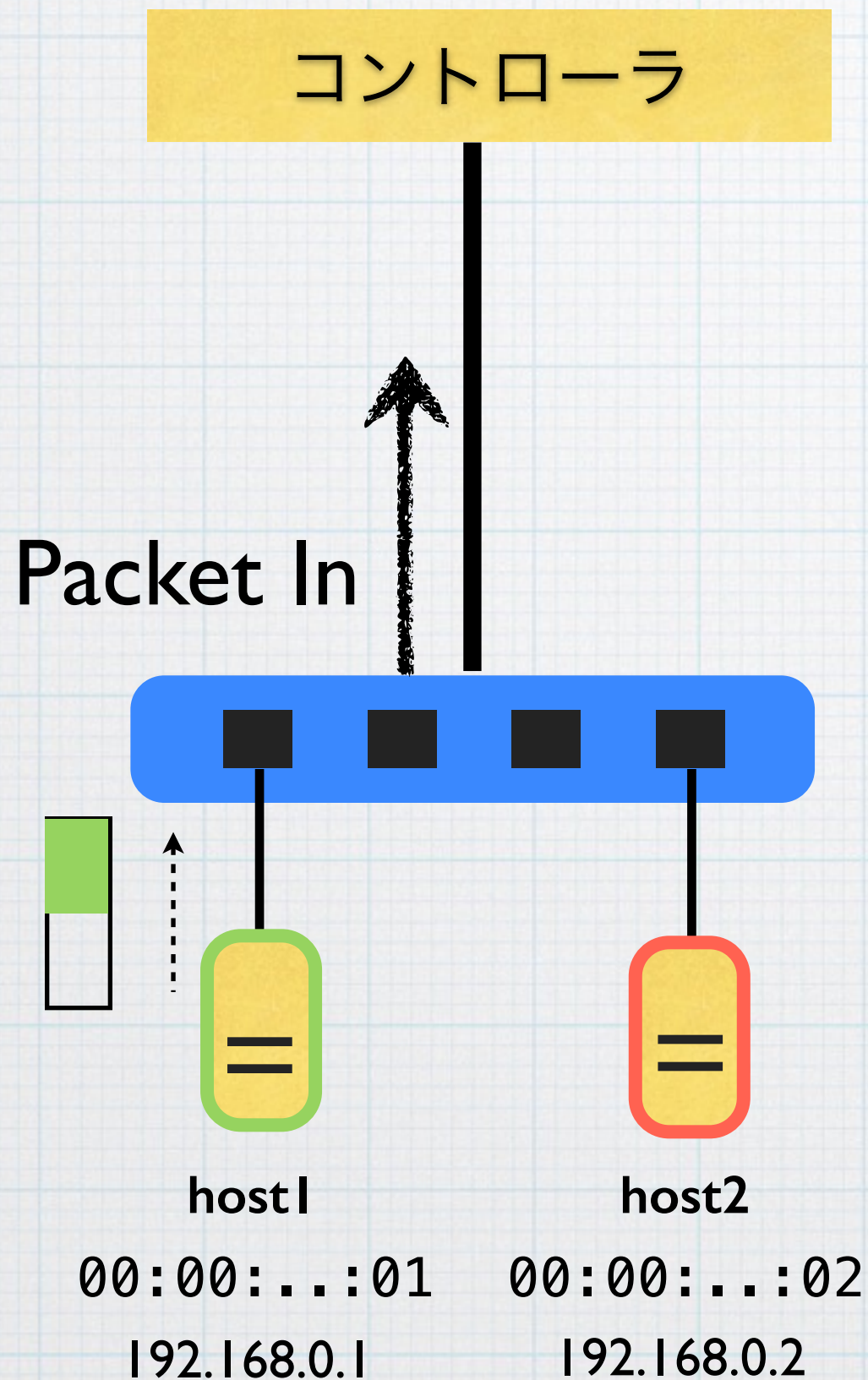
00:00:...:02  
192.168.0.2





# 課題

Packet In を  
起こしてみよう





```
$ trema send_packets \  
    --source host1 --dest host2
```

- ・ テストパケットの送信
- ・ host1 から host2 へ送る



```
$ trema show_stats host2
```

- host2の受信パケットの統計
- ちゃんと届くか? 確認



宛先ホストだけに  
パケットを届けるには？



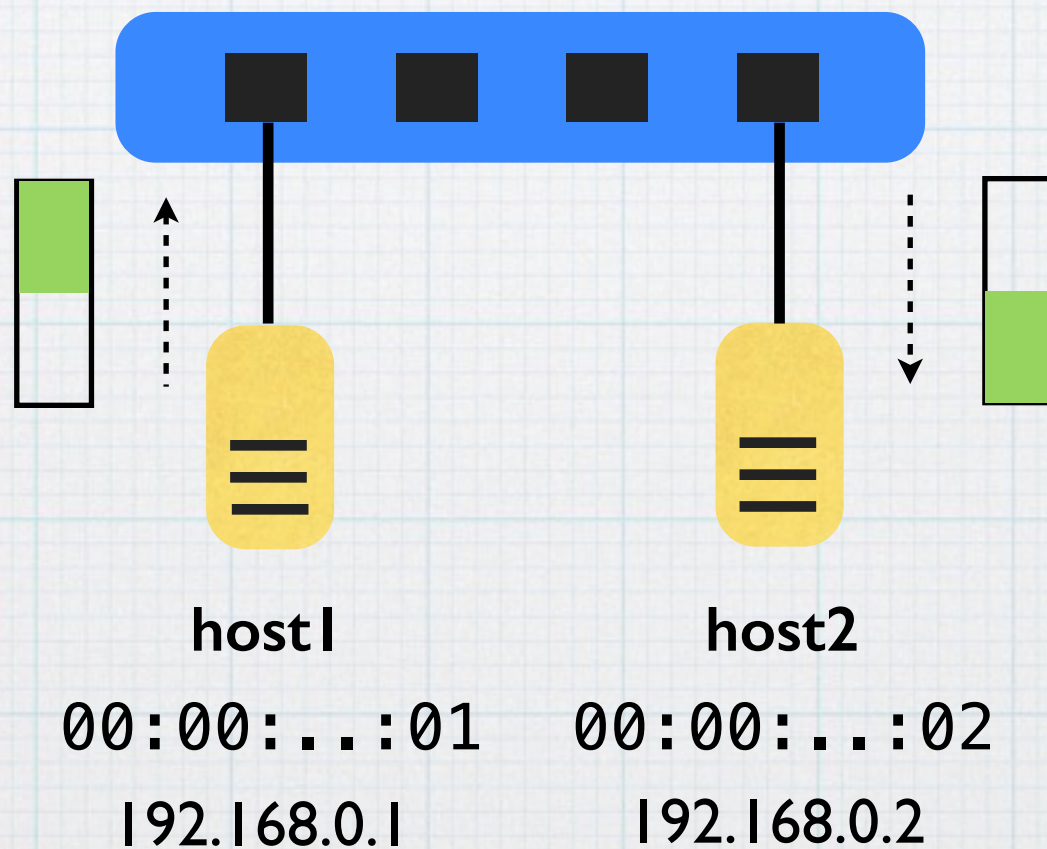
# ハッシュテーブル

## Forwarding DB

00:00:00:00:00:01 → 1

00:00:00:00:00:02 → 4

宛先 = host2





# MAC アドレス→ポート番号の学習

```
@fdb[message.macsa] = message.in_port
```

# MAC アドレスから転送先のポート番号を引く

```
port_no = @fdb[message.macda]
```

- 学習: `hash[キー] = 値`
- ルックアップ: `hash[キー]`



# 疑似 コード

```
def packet_in(dpid, message)
```

```
    「MAC → ポート番号」を学習
```

```
    port ← message の macda からポート番号を検索
```

```
    if port がみつかった
```

```
        フロー（message → port）をスイッチに書き込む
```

```
        message を port に出力
```

```
    else
```

```
        message を FLOOD
```

```
    end
```

```
end
```



```
$ trema dump_flows 0xabc
```

- ・ フローのダンプ
- ・ 0xabcのフローテーブルを表示



# 前半まとめ

- OpenFlow スイッチの仕組み  
Packet In/Packet Out/Flow Mod
- テストパケットの送りがた
- フローテーブルのダンプ



# レポート課題

## I. 複数スイッチ対応版

(multi\_learning\_switch.rb) の動作解説

※説明 (コードの解説、動作確認例)

※レポートに図を入れる場合は .pdf を  
コミットすること



# 複数スイッチ対応

