情報ネットワーク学演習 II
第3回レポート課題

所属：大阪大学 大学院情報科学研究科 情報ネットワーク専攻
提出者: 33E16019 満越貴志
電子メールアドレス: t-mangoe@ist.osaka-u.ac.jp

提出年月日：平成 28 年 10 月 23 日

課題内容

OpenFlow1.3 版スイッチの動作の説明を行う。

スイッチの動作の説明

スイッチの動作の説明にあたり、ネットワークの構成は learning-switch-t-mangoe リポジトリの trema.conf ファイルに記述されたものを用いる。ネットワークの構成を図示すると、図 1 のようになる。まず、初期状態でマルチプルテーブルの内容を出力させると、以下のような結果が得られた。

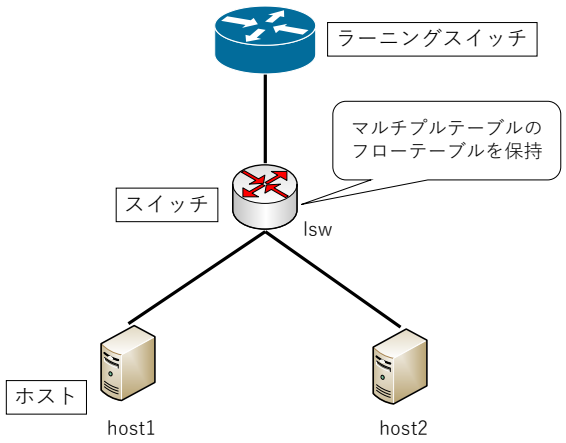

図 1: ネットワークの構成図

```
cookie=0x0, duration=11.834s, table=0, n_packets=0, n_bytes=0, priority=2,dl_dst
    =01:00:00:00:00:00/ff:00:00:00:00:00 actions=drop
cookie=0x0, duration=11.796s, table=0, n_packets=20, n_bytes=3302, priority=2,
    dl_dst=33:33:00:00:00:00/ff:ff:00:00:00:00 actions=drop
cookie=0x0, duration=11.796s, table=0, n_packets=5, n_bytes=1710, priority=1
    actions=goto_table:1
```

```
cookie=0x0, duration=11.796s, table=1, n_packets=5, n_bytes=1710, priority=3,
    dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
cookie=0x0, duration=11.796s, table=1, n_packets=0, n_bytes=0, priority=1 actions
    =CONTROLLER:65535
```

この結果を図示すると、図 2 のようになる。table 0 では、パケットのフィルタリングが行われる。

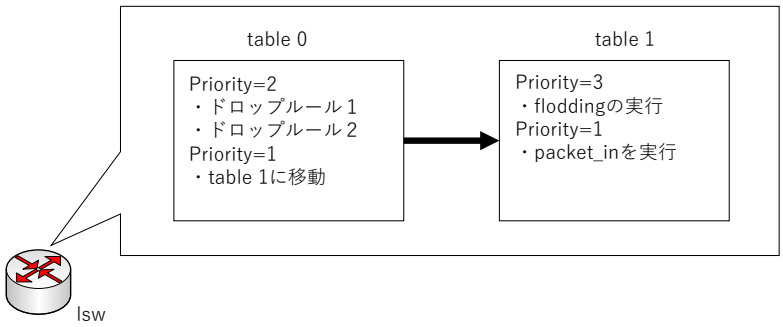

図 2: 初期状態におけるマルチプルテーブル

フィルタリングのルールは 2 つ存在し、

- 宛先が 01:00:00:00:00:00/ff:00:00:00:00:00 であるパケット
- 宛先が 33:33:00:00:00:00/ff:ff:00:00:00:00 であるパケット

の 2 種類のパケットが、table 0 のルールでドロップされることとなる。それ以外のパケットについては、table 1 を参照して処理が行われる。初期状態における table 1 のルールは以下の 2 つである。

1. パケットの宛先が ff:ff:ff:ff:ff:ff である場合、入力されたパケットを flooding で出力する。
2. それ以外の宛先のパケットである場合、packet_in を行う。

table 1 のルールは、コントローラからの flow_mod をスイッチが受け取ることで増えていく。

初期状態から、host1 が host2 に対してパケットを送った状況を考える。host1 からのパケットを受け取ったスイッチは、まず table 0 のルールを参照する。host1 からのパケットはフィルタリングのルールには該当しないため、スイッチは次に table 1 を参照する。host2 宛のパケットの宛先はフラッディングようのアドレス（ff:ff:ff:ff:ff:ff）ではないため、スイッチは packet_in を行う。この流れを図示すると、図 3 のようになる。コントローラは host2 のポート番号を知らないため、flow_mod は行わず、フラッディングで packet_out を行う。この結果、host2 にパケットが届く。packet_out 後のマルチプルテーブルを出力させると、以下のような結果が得られる。

```
cookie=0x0, duration=14.536s, table=0, n_packets=0, n_bytes=0, priority=2,dl_dst
    =01:00:00:00:00:00/ff:00:00:00:00:00 actions=drop
cookie=0x0, duration=14.498s, table=0, n_packets=22, n_bytes=3516, priority=2,
    dl_dst=33:33:00:00:00:00/ff:ff:00:00:00:00 actions=drop
cookie=0x0, duration=14.498s, table=0, n_packets=6, n_bytes=1752, priority=1
    actions=goto_table:1
cookie=0x0, duration=14.498s, table=1, n_packets=5, n_bytes=1710, priority=3,
    dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
cookie=0x0, duration=14.498s, table=1, n_packets=1, n_bytes=42, priority=1
    actions=CONTROLLER:65535
```

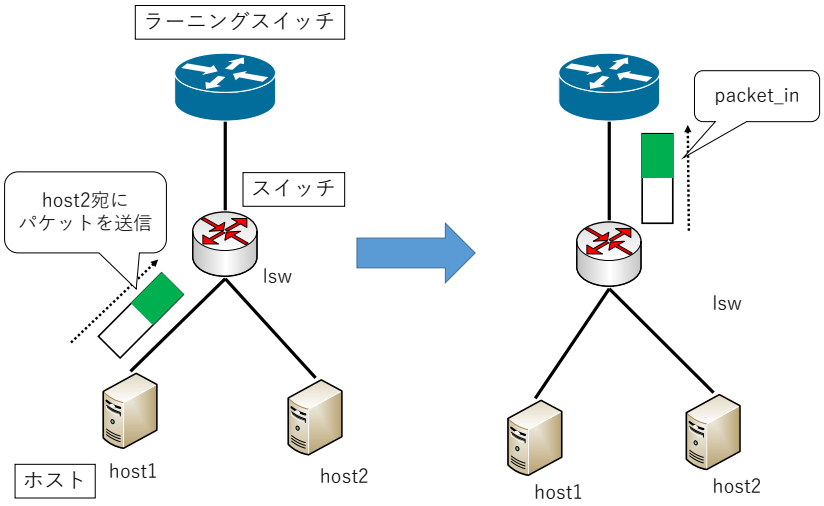

図 3: host1 のパケットの送信

このときの内容は、初期状態のときのものと同じである。

次に、host2 が host1 に対してパケットを送った状況を考える。スイッチが host2 からのパケットを受け取ると、マルチプルテーブルを参照し、コントローラに packet_in を行う。ここまでの流れは、host1 からのパケットを受け取ったときのものと同じであり、図示すると、図 4 のようになる。host2 からのパケットを受け取ったコントローラは、host1 のポートの情報を知っている。そのた

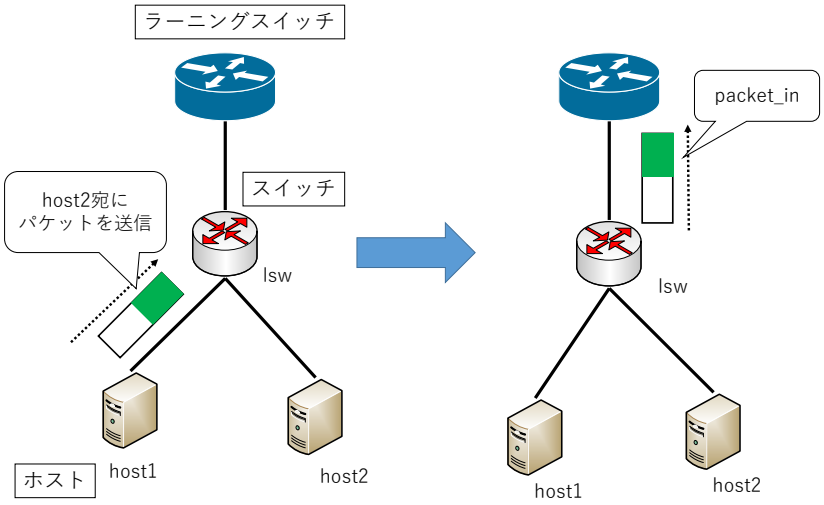

図 4: host2 のパケットの送信

め、host1 からのパケットを受け取った時とは異なり、コントローラはスイッチに対して flow_mod を行う。そして、受け取ったパケットを host1 のポート宛に packet_out する。

host1 へパケットが届いた後、スイッチのマルチプルテーブルを出力させると、以下のような結果が得られる。

```
cookie=0x0, duration=17.224s, table=0, n_packets=0, n_bytes=0, priority=2,dl_dst
    =01:00:00:00:00:00/ff:00:00:00:00:00 actions=drop
```

```
cookie=0x0, duration=17.186s, table=0, n_packets=22, n_bytes=3516, priority=2,
    dl_dst=33:33:00:00:00:00/ff:ff:00:00:00:00 actions=drop
cookie=0x0, duration=17.186s, table=0, n_packets=7, n_bytes=1794, priority=1
    actions=goto_table:1
cookie=0x0, duration=17.186s, table=1, n_packets=5, n_bytes=1710, priority=3,
    dl_dst=ff:ff:ff:ff:ff:ff actions=FLOOD
cookie=0x0, duration=1.27s, table=1, n_packets=0, n_bytes=0, idle_timeout=180,
    priority=2,in_port=2,dl_src=d8:51:88:53:aa:fb,dl_dst=6c:2a:f5:01:b4:1f actions
    =output:1
cookie=0x0, duration=17.186s, table=1, n_packets=2, n_bytes=84, priority=1
    actions=CONTROLLER:65535
```

この結果より、図 5 のように、table 1 にルールが追加される。

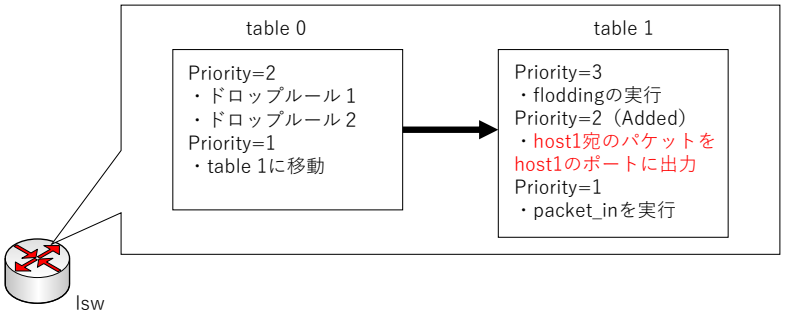

図 5: host1 のパケット受信後のマルチプルテーブル