

問題

複数スイッチ対応版 (multi_learning_switch.rb) の動作解説

解答

コードの解説

multi_learning_switch.rb のメソッド毎にコメントとしてコードの解説を追記する.

また, コード中に追記された logger.info メソッドは動作確認のために挿入されたものである

```
1  $LOAD_PATH.unshift __dir__
2
3  require 'fdb'
4
5  class MultiLearningSwitch < Trema::Controller
6    timer_event :age_fdb, interval: 5.sec
7
8    # Trema が開始したときに全ての FDB (Forwarding DB)を管理する連想配列を作成
9    # 開始メッセージを表示
10   def start(_argv)
11     @fdb = {}
12     logger.info 'MultiLearningSwitch started.'
13   end
14
15   # 各スイッチ (lsw1-4) が接続されたらそれぞれについて FDB を作成
16   # 各 FDB の識別はスイッチの ID で行う
17   def switch_ready(datapath_id)
18     @fdb[datapath_id] = FDB.new
19   end
20
21   # Packet In が発生したら呼ばれるメソッド
22   # 宛先が 802.1D 予約済みMACアドレスである場合は何もしない
23   # Packet In が発生したスイッチの FDB を取得し,
24   # (宛先アドレス, 転送先ポート) = (Packet In の送信元アドレス, Packet In の受信ポート)を記録
25   # 最後に flow_mod_and_packet_out メソッドを実行
26   def packet_in(datapath_id, message)
27     return if message.destination_mac.reserved?
28     logger.info "スイッチ #{datapath_id} にて Packet_In が発生しました"
29     @fdb.fetch(datapath_id).learn(message.source_mac, message.in_port)
30     flow_mod_and_packet_out message
31   end
32 end
```

```

31 end
32
33 # 5 秒毎に実行されるメソッド
34 # 有効期限が切れた FDB のエントリがあれば削除する
35 def age_fdb
36   @fdb.each_value(&:age)
37 end
38
39 private
40
41 # Packet In の宛先 Mac アドレスから FDB を参照することで転送先ポート番号を得る
42 # 転送先ポートが見つければ flow_mod を実行
43 # さらに転送先ポートが見つければそのポート (port_no) からパケットを転送し、
44 # 転送先ポートが見つからなければフラッディングする (packet_out で処理)
45 def flow_mod_and_packet_out(message)
46   port_no = @fdb.fetch(message.dpid).lookup(message.destination_mac)
47   flow_mod(message, port_no) if port_no
48   packet_out(message, port_no || :flood)
49 end
50
51 # Packet In の転送先ポートが見つかった場合は
52 # 今後同じパケットをスイッチのみで同様に処理できるよう
53 # send_flow_mod_add を実行する
54 def flow_mod(message, port_no)
55   send_flow_mod_add(
56     message.datapath_id,
57     match: ExactMatch.new(message),
58     actions: SendOutPort.new(port_no)
59   )
60   logger.info "スイッチ #{message.datapath_id} の FlowTable にルールを追加しました"
61 end
62
63 # Packet_In を指定のポート (port_no) から送信するメソッド
64 # port_no に :flood が指定された場合はフラッディングする
65 def packet_out(message, port_no)
66   send_packet_out(
67     message.datapath_id,
68     packet_in: message,
69     actions: SendOutPort.new(port_no)
70   )
71   logger.info "スイッチ #{message.datapath_id} のポート #{port_no} からパケットを送信しました"
72 end
73 end

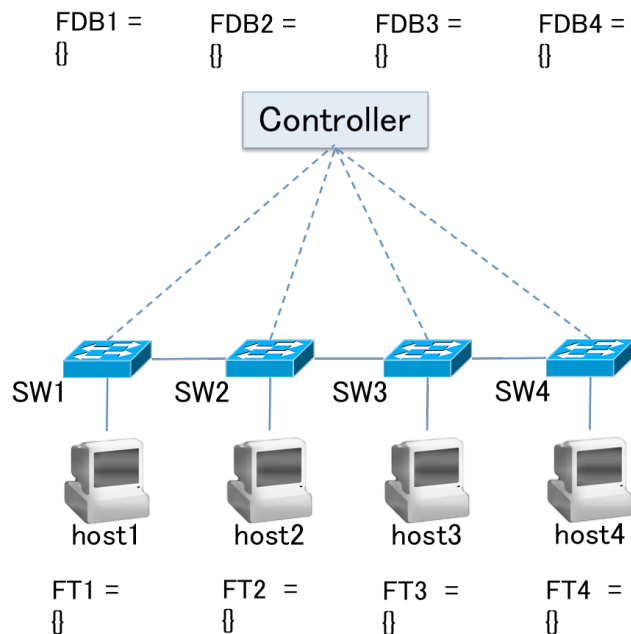
```

動作確認

以下の順にパケットを転送し、その際のログメッセージと動作概要を示す。

trema.multi.conf に記述されているトポロジは以下の通りである。
FDBn, FTn は スイッチ n における FDB, FlowTable を意味する。
(FDB についてはコントローラで実装されている)

Example のトポロジ



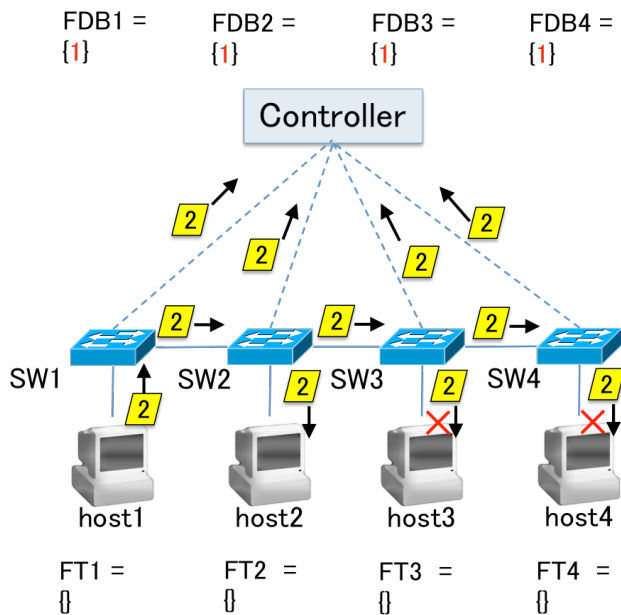
(1) host1 から host2 へパケットを送る

ログメッセージ

- 1 | スイッチ 1 にて Packet_In が発生しました
- 2 | スイッチ 1 のポート flood からパケットを送信しました
- 3 | スイッチ 2 にて Packet_In が発生しました
- 4 | スイッチ 2 のポート flood からパケットを送信しました
- 5 | スイッチ 3 にて Packet_In が発生しました
- 6 | スイッチ 3 のポート flood からパケットを送信しました
- 7 | スイッチ 4 にて Packet_In が発生しました
- 8 | スイッチ 4 のポート flood からパケットを送信しました

動作概要

host1 から host2 宛てのパケットが送られると, SW1 にて Packet In が生じる。
コントローラはパケットの送信元情報から SW1 の FDB に host1 宛てのエントリを生成し,
そして host2 (の MAC アドレス) 宛てのエントリは SW1 の FDB に登録されていないため,
SW1 に当該パケットをフラッディングさせる。
その結果, SW2, SW3, SW4 についても同様の処理が生じる。
host2 以外のホストは当該パケットの宛先が自分宛てでないため当該パケットを破棄する。



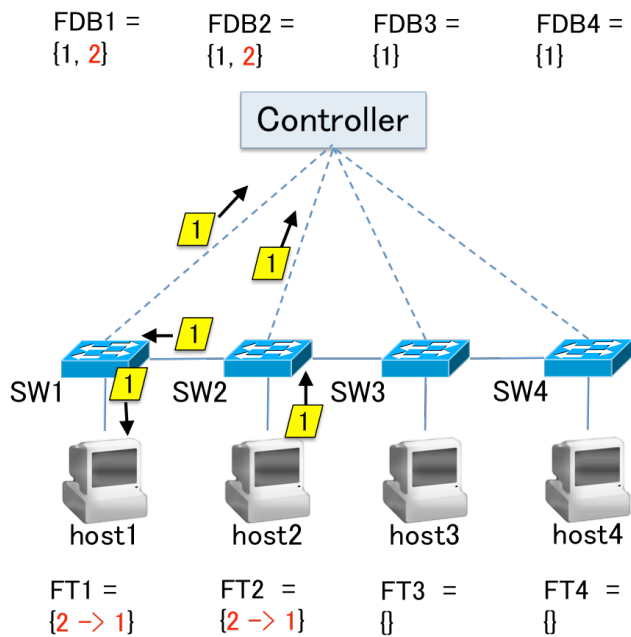
(2) host2 から host1 へパケットを送る

ログメッセージ

- 1 | スイッチ 2 にて Packet_In が発生しました
- 2 | スイッチ 2 の FlowTable にルールを追加しました
- 3 | スイッチ 2 のポート 2 からパケットを送信しました
- 4 | スイッチ 1 にて Packet_In が発生しました
- 5 | スイッチ 1 の FlowTable にルールを追加しました
- 6 | スイッチ 1 のポート 1 からパケットを送信しました

動作概要

host2 から host1 宛てのパケットが送られると、SW2 にて Packet In が生じる。
 コントローラはパケットの送信元情報から SW2 の FDB に host2 宛てのエントリを生成し、
 そして host2 宛てのエントリは SW1 の FDB に登録されているため、
 SW2 に当該パケットを適切な転送先（SW1）に転送させる。
 このとき、同じパケットを今後スイッチで転送処理するため FlowTable に転送ルールを書き込む。
 SW1 についても同様であり、結果的にパケットは適切な宛先（host1）に到達する。



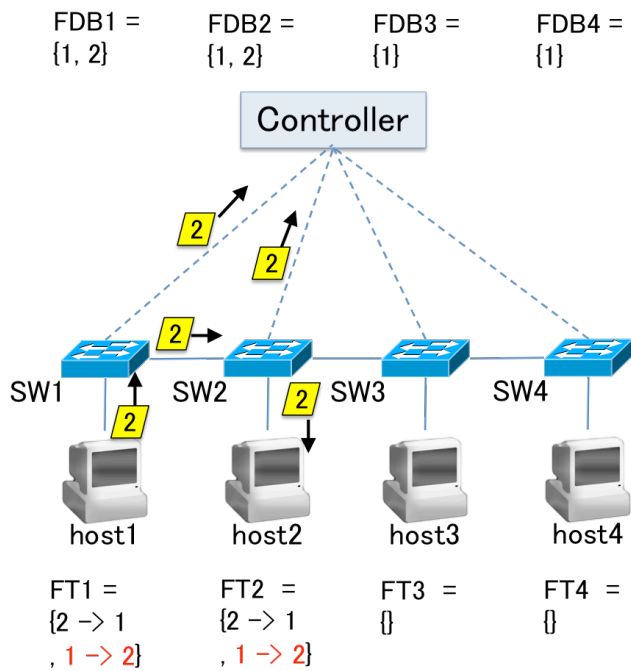
(3) host1 から host2 へパケットを送る

ログメッセージ

- 1 | スイッチ 1 にて Packet_In が発生しました
- 2 | スイッチ 1 の FlowTable にルールを追加しました
- 3 | スイッチ 1 のポート 2 からパケットを送信しました
- 4 | スイッチ 2 にて Packet_In が発生しました
- 5 | スイッチ 2 の FlowTable にルールを追加しました
- 6 | スイッチ 2 のポート 1 からパケットを送信しました

動作概要

(2) と同様である.

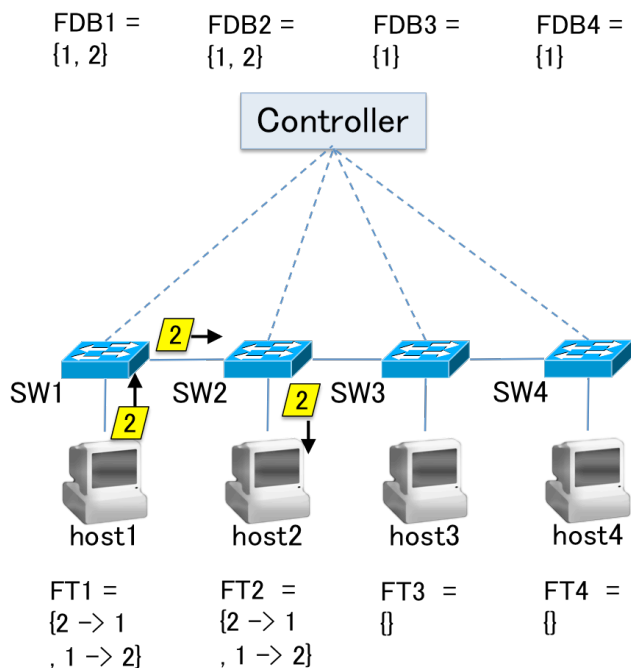


(4) host1 から host2 へパケットを送る

ログメッセージ

動作概要

SW1, SW2 の FlowTable に転送ルールが存在するため Packet In は生じない。
したがってログメッセージは表示されず、スイッチによって当該パケットは適切に転送される。



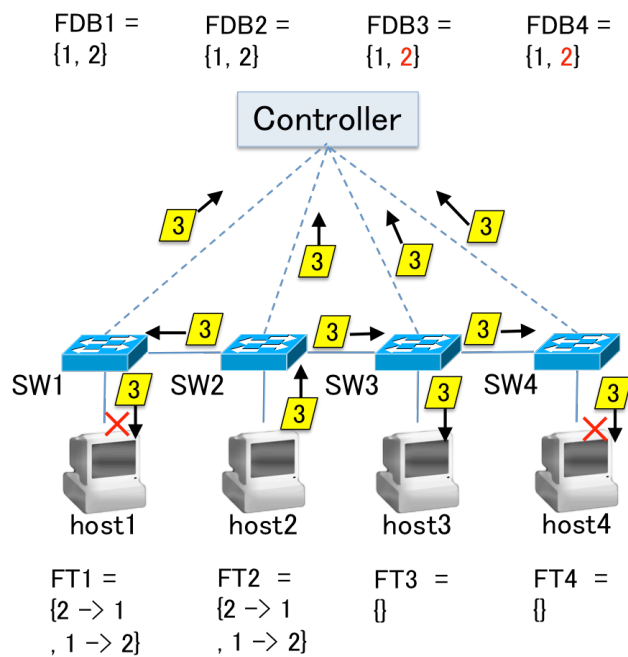
(5) host2 から host3 へパケットを送る

ログメッセージ

- 1 | スイッチ 2 にて Packet_In が発生しました
- 2 | スイッチ 2 のポート flood からパケットを送信しました
- 3 | スイッチ 3 にて Packet_In が発生しました
- 4 | スイッチ 3 のポート flood からパケットを送信しました
- 5 | スイッチ 1 にて Packet_In が発生しました
- 6 | スイッチ 1 のポート flood からパケットを送信しました
- 7 | スイッチ 4 にて Packet_In が発生しました
- 8 | スイッチ 4 のポート flood からパケットを送信しました

動作概要

SW2, SW3 において FDB のエントリに宛先アドレスが存在しないため (1) と同様に動作する.



(6) host4 から host2 へパケットを送る

ログメッセージ

- 1 | スイッチ 4 にて Packet_In が発生しました
- 2 | スイッチ 4 の FlowTable にルールを追加しました
- 3 | スイッチ 4 のポート 2 からパケットを送信しました
- 4 | スイッチ 3 にて Packet_In が発生しました
- 5 | スイッチ 3 の FlowTable にルールを追加しました
- 6 | スイッチ 3 のポート 2 からパケットを送信しました
- 7 | スイッチ 2 にて Packet_In が発生しました
- 8 | スイッチ 2 の FlowTable にルールを追加しました
- 9 | スイッチ 2 のポート 1 からパケットを送信しました

動作概要

SW4, SW3, SW2 において FDB のエントリに宛先アドレスが存在するため (2) と同様に動作する.

