プロジェクト研究 課題

1. 内容物

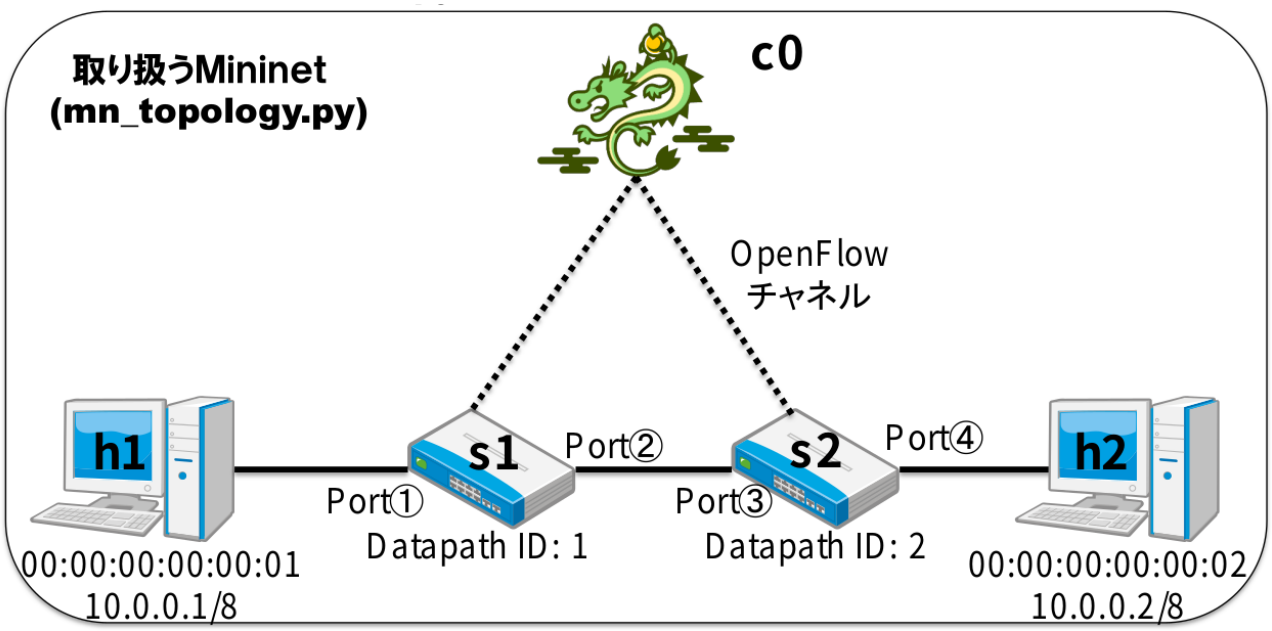
- `mn_topology.py`

  Mininetを起動するためのプログラム

  - 実行方法

```
$ sudo python mn_topology.py
```

  - ビルドされるトポロジ
    ホスト h1とh2、OFS s1とs2、OFC c0によるOpenFlowネットワーク



- `mn_topology_loss.py`

  mn_topology.pyとビルドされるトポロジは同じであるが、
  s1とs2間の双方向リンクのパケットロス率が10%に設定されている

- `L2monitor.py`

  本講義や課題で利用するRyuプログラムであり、5秒毎に各OFSに設置されてある優先度10のフローエントリの情報を表示する

  - 実行方法

```
$ sudo ryu-manager L2monitor.py
```

2. 課題

今回の講義で利用した`L2monitor.py`に追記をしてもらう
※追記箇所は「#課題1を以下に追記」「#課題2を以下に追記」と`L2monitor.py`中に書いている

2-1.特定のパケットのフィルタリング

内容

s2からのSwitch Features ReplyをOFCが受け取った後、
s2**のみに**宛先ポート番号3000を持つUDPパケットを廃棄する**優先度30**のフローエントリを設置する

- 利用するMatch条件
    - s2の受信ポートのポート番号: {s2のh1側のポート番号を入れる}
    - イーサタイプ: {IPv4を意味する16進数を入れる}
    - プロトコル番号: {UDPのプロトコル番号を入れる}
    - 宛先ポート番号: 3000
- Action
    - パケット廃棄　　※Actionに空の配列を代入することで実現できる

利用するMatch条件の変数は以下のURLから確認できる
https://osrg.github.io/ryu-book/ja/html/openflow_protocol.html#id1
また、変数の型は以下のURLから確認できる
https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html#flow-match-structure

確認方法

- h1とh2でiperfを実行する

`-p`で宛先ポート番号を3000番に変更できる

```
<xterm h1>
# iperf -c 10.0.0.2 -u -i 1 -p 3000
```

```
<xterm h2>
# iperf -s -u -i 1 -p 3000
```

書き込んだフローエントリの設定がうまくいっていれば`<xterm h1>`上で、
「[15] WARNING: did not receive ack of last datagram after 10 tries.」と表示される
また`<xterm h2>`上ではパケットが届かないため何も表示されない

- フローエントリを確認する

```
<ipef実行後のフローエントリを確認>
mininet> dpctl dump-flows
                              ⋮
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=14.759s, table=0, n_packets=668, n_bytes=1010016,
priority=30,udp,in_port="s2-eth3",tp_dst=3000 actions=drop
                              ⋮
```

うまくいっていればactionsが廃棄を意味するdropになっていることが確認できる

※注意
iperfで確認できるパケット不通はend-to-end。パケットキャプチャツール（wireshark）を用いてどこでパケットロスしているのかを確認する。 3000番以外のポート番号で通信ができることをたしかめること。

2-2. パケットロス率の定期的測定

mn_topology_loss.pyを利用してMininetをビルドする

内容

s1からs2へのリンクのパケットロス率の測定を5秒間隔で行えるようにする
またh1からh2へのパケットの通信を終えると、「パケットが流れていません」と表示するようにする

※h1とh2で送受信可能となる優先度10のフローエントリが投入された時点を0秒とする

```
<実行結果の例>
$ sudo ryu-manager L2monitor.py
                  ⋮
**************************************
5秒毎のOFS間のパケットロス率: 9.64%
**************************************

**************************************
5秒毎のOFS間のパケットロス率: 7.38%
**************************************

**************************************
5秒毎のOFS間のパケットロス率: 9.88%
**************************************

**************************************
パケットが流れていません
**************************************

**************************************
パケットが流れていません
```

```
************************************
                ⋮
```

パケットロス率の測定に利用するフローエントリ

以下2種類のフローエントリを利用してパケットロス率の導出を行う

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
cookie=0x0, duration=22.932s, table=0, n_packets=1784, n_bytes=2697408,
priority=10,in_port="s1-eth1",dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02
actions=output:"s1-eth2"
                            ⋮
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=22.937s, table=0, n_packets=1616, n_bytes=2443392,
priority=10,in_port="s2-eth3",dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02
actions=output:"s2-eth4"
                            ⋮
```

※s1-eth1: Port①、s1-eth2: Port②、s2-eth3: Port③、s2-eth4: Port④

パケットロス率の導出方法

Aをs1のPort②から**5秒間**送信されたパケット数、
Bをs2のPort③で**その5秒間に**受信できたパケット数 とするとき、
パケットロス率は **(A-B)/A** となる

※注意
フローエントリの統計情報フィールドにある累積パケット数をそのまま使うのではなく、
ある5秒間に送受信したパケット数をパケットロス率の計算に用いる

確認方法

- h1とh2でiperfを実行する

`-t`でiperf実行時間を変更することができる
※下記例では30秒

```
<xterm h1>
# iperf -c 10.0.0.2 -u -t 30
```

`-i`でパケットロス率やジッターの結果を出力できる
※下記例では2秒毎に出力される

```
<xterm h2>
# iperf -s -u -i 5
```

3. Tips

Mininet上に設置されたフローエントリを削除する方法

Mininetプログラムの再起動をせずにフローエントリの削除を行うことができる

```
mininet> dpctl del-flows
```