# 使用说明(-基于 Openflow 的感知网络系统)

## 一 目标:

设计一种基于 openflow 架构的**网络资源感知系统**,控制层能够获取底层网络的**拓扑信息**和**链路的实时状态信息**,同时控制层具备**一定的业务分析能力**;利用仿真软件实现该资源感知系统。

## 二:设计:

**1 实验环境:**

Ubuntu14.0.4;Ryu 控制器;Openflow1.3;Mininet 和 Openvswitch 仿真网络环境

**2 系统功能:**

1 可以实时获取网络 topo 信息,并且分析计算出包括交换机之间的连接情况,链路的连接端口情况。

2 可以实时获取、统计、计算端口信息:包括各个端口的容量,实时速率,收发包数,错误率,端口的状态(比如 down up)

3 链路的带宽,链路的状态(比如通断)

4 可以发现主机通信,并且掌控主机的连接状况

5 具有带环路 topo 下的主机通信与路由、避免环路风暴业务

6 可以完成最短路径分析学习与路由规划功能,让主机以最短路通信。

## 三:实验步骤:

1 先将文件夹四个资源文件存入 Linux 系统的某目录下:

```
root@sdnsy:/home/sdnsy/ryu/ryu/app/myapp/network_aware# ls
network_aware.py  network_monitor.py  shortest_route.py  tree.py
```

2 开启控制器 Ryu,加载写好的网络资源感知等代码文件:

ryu-manager shortest_route.py network_monitor.py --observe-links

```
root@sdnsy:/home/sdnsy/ryu/ryu/app/myapp/network_aware# ryu-manager shortest_route.py network_monitor.py --observe-links
loading app shortest_route.py
loading app network_monitor.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app None of Network_Aware
creating context Network_Aware
instantiating app shortest_route.py of Shortest_Route
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app network_monitor.py of network_minitor
----------Topo Link----------
    switch
----------Link Port----------
    switch
----------Access Host----------
    switch            Host
NO found host
```

3 启动 Mininet ,即启动 topo，连接控制器。

方法：通过写好的 topo 脚本启动，命令：python tree.py（需要设置版本 of 1.3）

```
root@sdnsy:/home/sdnsy/ryu/ryu/app/myapp/network_aware# python tree.py
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(10.00Mbit) (10.00Mbit) (s1, s2) (5.00Mbit) (5.00Mbit) (s1, s4) (10.00Mbit) (10.00Mbit) (s2,
 s3) (5.00Mbit) (5.00Mbit) (s2, s5) (10.00Mbit) (10.00Mbit) (s3, s1) (5.00Mbit 10ms delay) (
5.00Mbit 10ms delay) (s3, s6) (1.00Mbit) (1.00Mbit) (s4, h1) (1.00Mbit) (1.00Mbit) (s4, h2)
(1.00Mbit) (1.00Mbit) (s5, h3) (1.00Mbit) (1.00Mbit) (s5, h4) (1.00Mbit) (1.00Mbit) (s6, h5)
 (1.00Mbit) (1.00Mbit) (s6, h6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
controller
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...(10.00Mbit) (10.00Mbit) (5.00Mbit) (10.00Mbit) (10.00Mbit) (5.00Mbit) (
10.00Mbit) (10.00Mbit) (5.00Mbit 10ms delay) (5.00Mbit) (1.00Mbit) (1.00Mbit) (5.00Mbit) (1.
00Mbit) (1.00Mbit) (5.00Mbit 10ms delay) (1.00Mbit) (1.00Mbit)
*** Starting CLI:
mininet>
```

4 观察控制器的实时资源信息：

实时信息循环打印：

1 )topo 信息：

```
---------------------Topo Link----------------------
    switch          1          2          3          4          5          6
         1          0          1          1          1        inf        inf
         2          1          0          1        inf          1        inf
         3          1          1          0        inf        inf          1
         4          1        inf        inf          0        inf        inf
         5        inf          1        inf        inf          0        inf
         6        inf        inf          1        inf        inf          0
---------------------Link Port----------------------
    switch          1          2          3          4          5          6
         1    No_link     (1, 1)     (2, 2)     (3, 1)    No_link    No_link
         2     (1, 1)    No_link     (2, 1)    No_link     (3, 1)    No_link
         3     (2, 2)     (1, 2)    No_link    No_link    No_link     (3, 1)
         4     (1, 3)    No_link    No_link    No_link    No_link    No_link
         5    No_link     (1, 3)    No_link    No_link    No_link    No_link
         6    No_link    No_link     (1, 3)    No_link    No_link    No_link
```

解释:

topo link：以标准的邻接矩阵打印，说明各个交换机之间的连接情况。

1 为连接，0 为自己，inf 为无链路连接。

Link port:进一步探测各个链路在各个交换机上的的连接情况。如第一行第二列（1,1）说明，交换机 1 的端口 1 和交换机 2 的端口 1 以链路连接。

2 )端口、链路实时信息

| datapath | port | rx-pkts | rx-bytes | rx-error | tx-pkts | tx-bytes | tx-error | port-speed(B/s) | current-capacity(Kbps) | port-stat | link-stat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000000000000001 | 1 | 492 | 28367 | 0 | 491 | 28297 | 0 | 110.5 | 10000000 | up | up |
| 0000000000000001 | 2 | 491 | 28297 | 0 | 490 | 28246 | 0 | 110.5 | 10000000 | up | up |
| 0000000000000001 | 3 | 491 | 28297 | 0 | 491 | 28297 | 0 | 119.0 | 10000000 | up | up |
| 0000000000000002 | 1 | 491 | 28297 | 0 | 492 | 28367 | 0 | 110.4 | 10000000 | up | up |
| 0000000000000002 | 2 | 491 | 28316 | 0 | 490 | 28246 | 0 | 101.9 | 10000000 | up | up |
| 0000000000000002 | 3 | 492 | 28367 | 0 | 491 | 28297 | 0 | 110.4 | 10000000 | up | up |
| 0000000000000003 | 1 | 490 | 28246 | 0 | 491 | 28316 | 0 | 102.0 | 10000000 | up | up |
| 0000000000000003 | 2 | 490 | 28246 | 0 | 491 | 28297 | 0 | 110.5 | 10000000 | up | up |
| 0000000000000003 | 3 | 491 | 28297 | 0 | 491 | 28297 | 0 | 119.0 | 10000000 | up | up |
| 0000000000000004 | 1 | 491 | 28297 | 0 | 491 | 28297 | 0 | 118.9 | 10000000 | up | up |
| 0000000000000004 | 2 | 7 | 558 | 0 | 490 | 28246 | 0 | 51.0 | 10000000 | up | up |
| 0000000000000004 | 3 | 7 | 558 | 0 | 493 | 28615 | 0 | 51.0 | 10000000 | up | up |
| 0000000000000005 | 1 | 491 | 28297 | 0 | 492 | 28367 | 0 | 110.5 | 10000000 | up | up |
| 0000000000000005 | 2 | 7 | 558 | 0 | 491 | 28297 | 0 | 59.5 | 10000000 | up | up |
| 0000000000000005 | 3 | 7 | 558 | 0 | 492 | 28367 | 0 | 59.5 | 10000000 | up | up |
| 0000000000000006 | 1 | 491 | 28297 | 0 | 491 | 28297 | 0 | 118.9 | 10000000 | up | up |
| 0000000000000006 | 2 | 7 | 558 | 0 | 491 | 28297 | 0 | 59.5 | 10000000 | up | up |
| 0000000000000006 | 3 | 7 | 558 | 0 | 491 | 28297 | 0 | 59.5 | 10000000 | up | up |

解释：

从左到右依次为：交换机 id 号，端口号，收包数，接收 Byte 数，接收错误数，传输包数，传输 Byte 数，传输错误数，端口实时速率，端口容量，端口状态，端口所在链路的状态。

可以尝试模拟现实情况，down 掉一个端口，再查看这个实时信息：

在 mininet 交换界面输入：

```
mininet> s1 ifconfig s1-eth1 down
```

再次查看 Ryu 界面输出信息：

| datapath | port | rx-pkts | rx-bytes | rx-error | tx-pkts | tx-bytes | tx-error | port-speed(B/s) | current-capacity(Kbps) | port-stat | link-stat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000000000000001 | 1 | 756 | 41881 | 0 | 756 | 41862 | 0 | 0.0 | 10000000 | Down | Down |
| 0000000000000001 | 2 | 843 | 46299 | 0 | 843 | 46299 | 0 | 119.0 | 10000000 | up | up |
| 0000000000000001 | 3 | 843 | 46299 | 0 | 843 | 46299 | 0 | 119.0 | 10000000 | up | up |
| 0000000000000002 | 1 | 756 | 41862 | 0 | 756 | 41881 | 0 | 0.0 | 10000000 | up | Down |
| 0000000000000002 | 2 | 844 | 46369 | 0 | 842 | 46248 | 0 | 110.5 | 10000000 | up | up |
| 0000000000000002 | 3 | 844 | 46369 | 0 | 843 | 46299 | 0 | 102.0 | 10000000 | up | up |
| 0000000000000003 | 1 | 842 | 46248 | 0 | 844 | 46369 | 0 | 110.4 | 10000000 | up | up |
| 0000000000000003 | 2 | 843 | 46299 | 0 | 843 | 46299 | 0 | 118.9 | 10000000 | up | up |

可以发现相应端口速率为 0，且状态为 down，且相应链路为 down。当然在 1 的 topo 信息输出中也可以看到。

记得测试完，up 刚才 down 掉的端口。为了更好的后面实验^.^

3 )环路处理，最短路径路由

在 mininet 中输入 h1 ping h2：

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=9.80 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.205 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.072 ms
64 bytes from 10.0.0.2: icmp_req=6 ttl=64 time=0.068 ms
64 bytes from 10.0.0.2: icmp_req=7 ttl=64 time=0.062 ms
64 bytes from 10.0.0.2: icmp_req=8 ttl=64 time=0.068 ms
```

在 Ryu 中立即查看：

```
-----------------FOUND CONMUNAICATION-------------
 10.0.0.1 <--> 10.0.0.4 : the shortest path on switch:[4, 1, 2, 5]
-----------------FOUND CONMUNAICATION-------------
 10.0.0.1 <--> 10.0.0.4 : the shortest path on switch:[4, 1, 2, 5]
-----------------FOUND CONMUNAICATION-------------
 10.0.0.4 <--> 10.0.0.1 : the shortest path on switch:[5, 2, 1, 4]
-----------------FOUND CONMUNAICATION-------------
 10.0.0.4 <--> 10.0.0.1 : the shortest path on switch:[5, 2, 1, 4]
```

控制器获取到 h1 与 h2 通信，并且经过计算分析，获取到一条最短路径：4-1-2-5

4) 主机信息学习与获取

```
-------------------Access Host--------------------
    switch             Host
         4:     10.0.0.1
         4:     10.0.0.2
         5:     10.0.0.4
```

学习到 h2 和 h4，并且学习到其连接的 switch。

要想学习到更多主机，可以在 mininet 中输入 pingall，完成互相通信。

5) 流标信息获取：

```
datapath         in-port  ip-dst          out-port packets  bytes  flow-speed(B/s)
---------------- -------- ---------------- -------- -------- -------- ---------
0000000000000001        1          10.0.0.1        3       26     2548      98.0
0000000000000001        3          10.0.0.4        1       26     2548      98.0
0000000000000002        1          10.0.0.4        3       26     2548       0.0
0000000000000002        3          10.0.0.1        1       26     2548    8166.7
0000000000000004        1          10.0.0.1        2       25     2450      98.0
0000000000000004        2          10.0.0.4        1       25     2450      98.0
0000000000000005        1          10.0.0.4        3       25     2450       0.0
0000000000000005        3          10.0.0.1        1       25     2450    8166.7
```

可以实时获得交换机流表信息

**注:**

1 Ryu 打印信息的速率可以调节，选择一种适应的打印速率。

2 问题处理与建议：先打开 Ryu，后启动 mininet 的 topo 脚本。

如果 ping 不成功，由于 topo 学习冲突等，可以关闭重新打开。

3 祝你好运！