

# 南京大学本科生实验报告

181860097 计算机科学与技术系 王明骁

Email: 1164151483@qq.com

开始日期: 2020年3月11日 结束日期: 2020年3月13日

lab1

## 实验名称

Lab 1: Switchyard & Mininet

## 实验目的

- 学习python的基础语法
- 学习和掌握wireshark, mininet, switchyard的相关使用
- 理解网络拓扑的构建代码, 学会更改相关内容达到一些目的

## 实验内容

- step 1

step 1中选择了在网络拓扑中删去server2

由于分布式网络中一个结点的删除并不会直接影响到该网络本身, 因此只需要将和server2相关的代码删去即可

在start\_mininet.py中注释了设置server2的ip和mac信息的代码, 同时更改

for node in ['server1', 'server2', 'client']:

为: for node in ['server1', 'client']:

- step 2

首先是对于muhub.py中的hub相关逻辑的理解:

```
log_debug ("In {} received packet {} on {}".format(net.name, packet, dev))
eth = packet.get_header(Ethernet)
cnt_in += 1
if eth is None:
    log_info("Received a non-Ethernet packet?!")
    continue

if eth.dst in mymacs:
    log_info ("Received a packet intended for me")

else:
    for intf in my_interfaces:
        if dev != intf.name:
            log_info ("Flooding packet {} to {}".format(packet, intf.name))
            cnt_out += 1
            net.send_packet(intf, packet)
    log_info("in:{} out:{}".format(cnt_in, cnt_out))
net.shutdown()
```

从中可以知道hub每收到一个包时, 首先判断这个包是不是从以太网收到的, 如果不是则不处理, 否则判断一下包的目标mac是不是自己, 如果是自己的mac, 则收下后不转发, 否则对除了来源端口之外的端口转发, 因此可以知道在收到包时计数可得到收到的总数, 在转发时计数可得到转发的总数, 最后将其按照格式输出即可

- step 3

step 3中选择通过传递给mk\_pkt函数不同的参数来构造不同的测试样例

```
def mk_pkt(hwsrc, hwdst, ipsrc, ipdst, reply=False):
    ether = Ethernet(src=hwsrc, dst=hwdst, ethertype=EtherType.IP)
    ippkt = IPv4(src=ipsrc, dst=ipdst, protocol=IPProtocol.ICMP, ttl=32)
    icmppkt = ICMP()
    if reply:
        icmppkt.icmptype = ICMPType.EchoReply
    else:
        icmppkt.icmptype = ICMPType.EchoRequest
    return ether + ippkt + icmppkt
```

mk\_pkt函数中通过接收发送方物理地址，目标物理地址，发送方ip和目标ip四个参数来构造一个包，因此只需要准备这四个参数传递给mk\_pkt函数即可构造好一个包，在准备发送的包后，仿照已有代码的测试样例的发送方式，使用s.expcet来完成一个测试样例的构建

```
# test case 4: a frame created by myself
mytest = mk_pkt("20:00:00:00:00:01", "ff:ff:ff:ff:ff:ff", "192.168.1.100", '172.16.42.2')
s.expect(PacketInputEvent("eth1", mytest, display=Ethernet), "An Ethernet frame with a broadcast destination address")
s.expect(PacketOutputEvent("eth0", mytest, "eth2", mytest, display=Ethernet), "The Ethernet frame with a broadcast destination address")
```

具体构建了一个从20:00:00:00:00:01到ff:ff:ff:ff:ff:ff的包，该包将会将在网络上进行广播，而hub会进行转发

- step 4

根据视频教程的讲解可知道，首先运行虚拟环境，然后在虚拟环境下运行sudo python lab\_1/start\_mininet.py启动mininet，在mininet的cli中运行xterm hub启动xterm，在xterm中运行虚拟环境，然后运行swyard lab\_1/muhub.py即可运行我构建的网络拓扑，在网络拓扑中可以运行pingall构造一些流量来看自己的网络构建是否正确，也可以在虚拟环境下运行swyard -t lab\_1/hubtest.py lab\_1/myhub.py来查看由测试样例构造的流量

- step 5

根据视频教程中对wireshark的讲解可以知道抓包的具体方法，首先按照step 4中的步骤运行自己的网络拓扑，然后在mininet中运行server1 wireshark &可在server1上启动wireshark，此时运行指令pingall构造一些流量可以看到wireshark中出现了构造的流量的相关信息，抓取到的包的具体说明在实验结果中列出

## 实验结果

- step 1

```
47 for node in ['server1','client']:  
48     # all links are 10Mb/s, 100 millisecond prop delay  
49     self.addLink(node, 'hub', bw=10, delay='100ms')  
50  
51 def set_ip(net, node1, node2, ip):  
52     node1 = net.get(node1)  
53     ilist = node1.connectionsTo(net.get(node2)) # returns list of tuples  
54     intf = ilist[0]  
55     intf[0].setIP(ip)  
56  
57 def reset_macs(net, node, macbase):  
58     ifnum = 1  
59     node_object = net.get(node)  
60     for intf in node_object.intfList():  
61         node_object.setMAC(macbase.format(ifnum), intf)  
62         ifnum += 1  
63  
64     for intf in node_object.intfList():  
65         print node,intf,node_object.MAC(intf)  
66  
67 def set_route(net, fromnode, prefix, nextnode):  
68     node_object = net.get(fromnode)  
69     ilist = node_object.connectionsTo(net.get(nextnode))  
70     node_object.setDefaultRoute(ilist[0][0])  
71  
72 def setup_addressing(net):  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

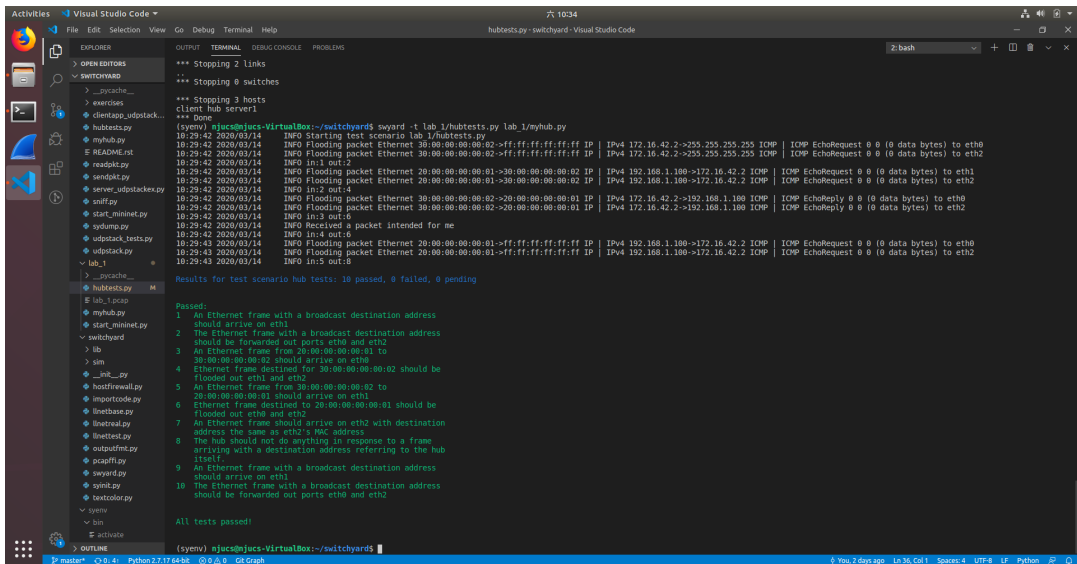
在mininet中运行nodes指令可以看到server2确实已经被删除

- step 2

```
09:49:15 2020/03/14 INFO Saving iptables state and installing switchyard rules  
09:49:15 2020/03/14 INFO Using network devices: hub-eth0 hub-eth1  
09:49:24 2020/03/14 INFO Flooding packet Ethernet 30:00:00:00:00:01->ff:ff:ff:ff:ff:ff ARP | Arp 30:00:00:00:00:01:192.168.100.3 00:00:00:00:00:00:192.168.100.1 to hub-eth0  
09:49:24 2020/03/14 INFO in:1 out:1  
09:49:25 2020/03/14 INFO Flooding packet Ethernet 10:00:00:00:00:01->30:00:00:00:00:01 ARP | Arp 10:00:00:00:00:01:192.168.100.1 30:00:00:00:00:01:192.168.100.3 to hub-eth1  
09:49:25 2020/03/14 INFO in:2 out:2  
09:49:25 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 1 (56 data bytes) to hub-eth0  
09:49:25 2020/03/14 INFO in:3 out:3  
09:49:25 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 1 (56 data bytes) to hub-eth1  
09:49:25 2020/03/14 INFO in:4 out:4  
09:49:25 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 1 (56 data bytes) to hub-eth1  
09:49:25 2020/03/14 INFO in:5 out:5  
09:49:25 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 1 (56 data bytes) to hub-eth0  
09:49:25 2020/03/14 INFO in:6 out:6  
09:49:30 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.3->192.168.100.1 1 (56 data bytes) to hub-eth0  
09:49:30 2020/03/14 INFO in:7 out:7  
09:49:30 2020/03/14 INFO Flooding packet Ethernet 0:00:00:01 IP | IPv4 192.168.100.1->192.168.100.3 1 (56 data bytes) to hub-eth1  
09:49:30 2020/03/14 INFO in:8 out:8
```

在mininet中运行pingall指令构造一些流量，在xterm中可以看到输出信息（运行测试样例的流量在step 3 的实验结果中展示，同样可以看到输出信息）

- step 3



运行 `swyard -t lab_1/hubtest.py lab_1/myhub.py` 可以看到测试样例成功构建，且能被正确执行

- step 4

```
(syenv) njucs@njucs-VirtualBox:~/Switchyard$ sudo python lab_1/start_mininet.py
[sudo] password for njucs:
*** Creating network
*** Adding hosts:
client hub server1
*** Adding switches:

*** Adding links:
(10.00Mbit 100ms delay) (10.00Mbit 100ms delay) (client, hub) (10.00Mbit 100ms delay) (10.00Mbit 100ms delay) (server1, hub)
*** Configuring hosts
client hub server1
server1 server1-eth0 10:00:00:00:00:01
client client-eth0 30:00:00:00:00:01
hub hub-eth0 40:00:00:00:00:01
hub hub-eth1 40:00:00:00:00:02
*** client : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** client : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** hub : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** hub : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.all.disable_ipv6=1',)
net.ipv6.conf.all.disable_ipv6 = 1
*** server1 : ('sysctl -w net.ipv6.conf.default.disable_ipv6=1',)
net.ipv6.conf.default.disable_ipv6 = 1
*** Starting controller

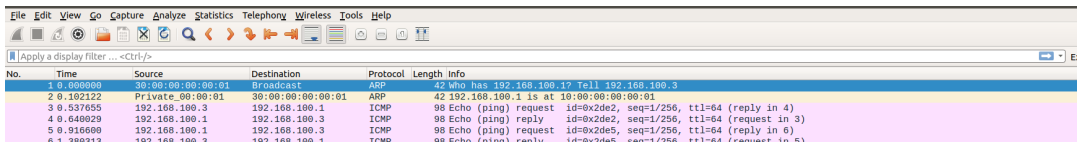
*** Starting 0 switches

*** Starting CLI:
mininet> xterm hub
mininet> pingall
*** Ping: testing ping reachability
client -> X server1
hub -> X X
server1 -> client X
*** Results: 66% dropped (2/6 received)
mininet>
```

在mininet中运行该网络，并构造一些流量，可以看到该网络正常

- step 5

运行pingall得到相关的流量被wireshark抓取



在wireshark中可以看到抓取到的包，首先是ARP request 请求192.168.100.1的MAC地址，然后是ARP reply 回复物理地址为10:00:00:00:00:01

然后pingall所构造的流量，从192.168.100.3发送到192.168.100.1，然后回复，然后从192.168.100.1发送到192.168.100.3，然后回复

## 核心代码

- step 1

注释了对server2的相关设置

```
def setup_addressing(net):
    reset_macs(net, 'server1', '10:00:00:00:00:{:02x}')
    # reset_macs(net, 'server2', '20:00:00:00:00:{:02x}')
    reset_macs(net, 'client', '30:00:00:00:00:{:02x}')
    reset_macs(net, 'hub', '40:00:00:00:00:{:02x}')
    set_ip(net, 'server1', 'hub', '192.168.100.1/24')
    # set_ip(net, 'server2', 'hub', '192.168.100.2/24')
    set_ip(net, 'client', 'hub', '192.168.100.3/24')
```

同时更改for

```
nodeconfig = {'cpu':-1}
self.addHost('server1', **nodeconfig)
# self.addHost('server2', **nodeconfig)
self.addHost('hub', **nodeconfig)
self.addHost('client', **nodeconfig)

#for node in ['server1','server2','client']:
for node in ['server1','client']:
    # all links are 10Mb/s, 100 millisecond prop delay
    self.addLink(node, 'hub', bw=10, delay='100ms')
```

- step 2

```
log_debug ("In {} received packet {} on {}".format(net.name, packet, dev))
eth = packet.get_header(Ethernet)
cnt_in += 1
if eth is None:
    log_info("Received a non-Ethernet packet?!")
    continue

if eth.dst in mymacs:
    log_info ("Received a packet intended for me")

else:
    for intf in my_interfaces:
        if dev != intf.name:
            log_info ("Flooding packet {} to {}".format(packet, intf.name))
            cnt_out += 1
            net.send_packet(intf, packet)
    log_info("in:{} out:{}".format(cnt_in,cnt_out))
net.shutdown()
```

在hub收到包时对收到的包进行计数，在hub发出包后对发出的包进行计数

- step 3

```
# test case 4: a frame created by myself
mytest = mk_pkt("20:00:00:00:00:01", "ff:ff:ff:ff:ff:ff", "192.168.1.100", '172.16.42.2')
s.expect(PacketInputEvent("eth1", mytest, display=Ethernet), "An Ethernet frame with a broadcast destination address")
s.expect(PacketOutputEvent("eth0", mytest, "eth2", mytest, display=Ethernet), "The Ethernet frame with a broadcast destination address")
```

提供给mk\_pkt函数相关参数构造包，并发送来构造测试样例

- step 4和step 5未涉及代码相关部分

## 总结与感想

本次实验是从0到1的突破，我学会了网络实验的相关工具的使用，同时也对网络相关的代码有了初步的了解，虽然完成了本次实验，但是框架代码中的一些细节的地方还不是太清楚，相信通过之后的学习可以更深入的理解网络相关的知识和网络相关的代码的编写