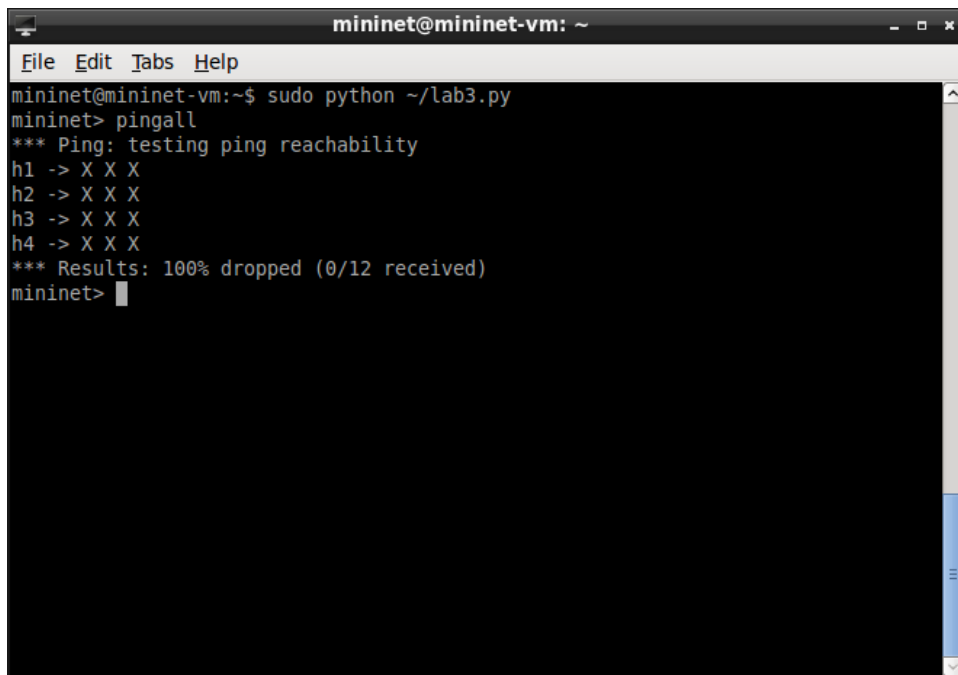


## Lab 3: Simple Firewall using OpenFlow

1. The output of the `pingall` command shows the connectivity between all pairs of hosts in my network by sending ICMP Echo Request packets (“pings”) from each host to every other host. `h1 -> X X X` means that host `h1` cannot reach any other host in my network. Each ‘X’ represents an unsuccessful ping to another host. The same interpretation applies for all other hosts `h2`, `h3`, and `h4`. The summary, `*** Results: 100% dropped (0/12 received)`, indicates that all ping packets sent were dropped and not a single one was successfully received. The number in parentheses, `(0/12)`, shows the ratio of successful ping replies to the total number of pings sent. Below is a screenshot of my output:



```
mininet@mininet-vm: ~  
File Edit Tabs Help  
mininet@mininet-vm:~$ sudo python ~/lab3.py  
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> X X X  
h2 -> X X X  
h3 -> X X X  
h4 -> X X X  
*** Results: 100% dropped (0/12 received)  
mininet>
```

2. After I generated some traffic using `pingall`, I immediately ran `dpctl dump-flows` to show the active flow entries installed in the switch `s1` before they expire due to the `idle_timeout` or `hard_timeout` specified in my `of_flow_mod` (`idle_timeout = 180`, `hard_timeout = 180`). Each entry in the output represents a flow rule that the switch has set up to handle packets in the network.
  - a. `cookie=0x0`: The cookie is an identifier that can be used by the controller to filter flow entries. In my case, it is set to zero.
  - b. `duration=XX.XXXs`: This shows how long the flow entry has been in the table. Each entry has its own duration in seconds.
  - c. `table=0`: The table field tells me which flow table the rule belongs to. In OpenFlow switches, I can have multiple tables. In my case, all flow rules are in table 0.
  - d. `n_packets=X`, `n_bytes=X`: These values tell me how many packets and total bytes have matched the rule since it was added.

e. `idle_timeout=30,hard_timeout=180`: This is the timeout setting for the flow. If the flow hasn't been matched for a period of time equal to the idle timeout, it will be removed. If the flow has been in the table for a period of time equal to the hard timeout, it will also be removed.

f. `idle_age=XX`: This field indicates the time in seconds since the flow entry was last matched.

g. `priority=X`: This represents the priority of the flow rule. If a packet matches multiple flow rules, the one with the highest priority will be chosen.

h. `match fields`: The fields after the priority represent the match conditions of the flow entry. `icmp`, `arp`, `dl_src`, `dl_dst`, `nw_src`, `nw_dst`, `nw_tos`, `icmp_type`, `icmp_code`, `arp_spa`, `arp_tpa`, and `arp_op` are all match fields that specify conditions for when this rule should be applied.

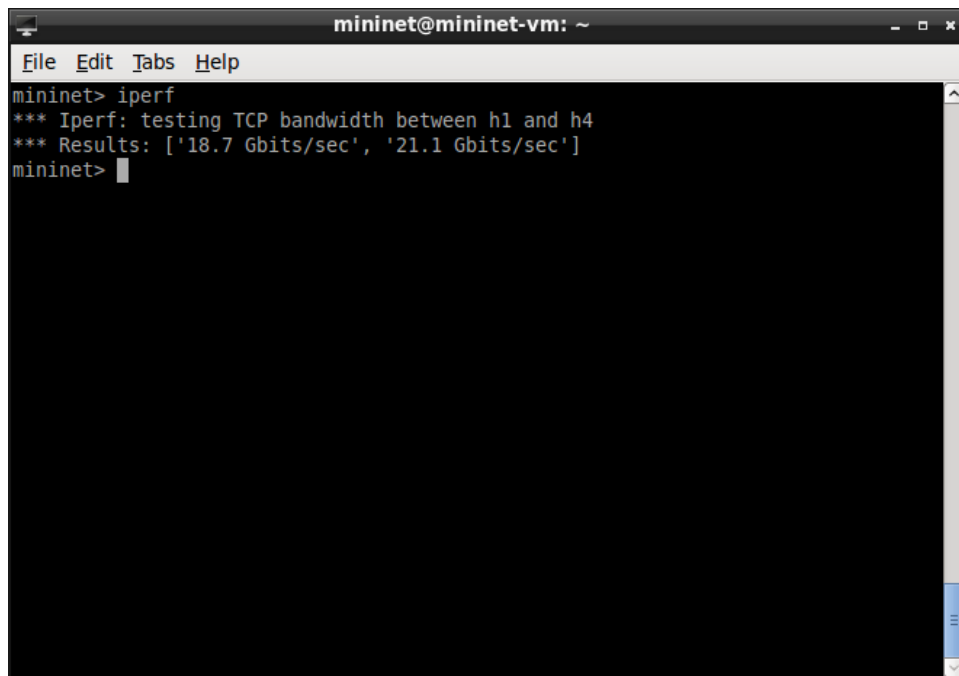
- `icmp`, `arp`: These are the types of the network protocol.
- `dl_src`, `dl_dst`: These are the source and destination MAC addresses.
- `nw_src`, `nw_dst`: These are the source and destination IP addresses.
- `nw_tos`: This is the Type of Service (ToS) in the IP header.
- `icmp_type`, `icmp_code`: These are the type and code of ICMP packet.
- `arp_spa`, `arp_tpa`: These are the source and target IP addresses in ARP packet.
- `arp_op`: This is the operation that the sender is performing (1 for request, 2 for reply).

i. `actions=ANY, actions=FLOOD`: This tells the switch what to do with the packet when it matches this rule. "ANY" action means that any action can be applied to this flow. "FLOOD" action means to send the packet out of all ports except the incoming port.

Below is a screenshot of my output:

```
mininet@mininet-vm: ~  
File Edit Tabs Help  
mininet> dpctl dump-flows  
*** s1 -----  
NXST FLOW reply (xid=0x4):  
cookie=0x0, duration=22.376s, table=0, n_packets=0, n_bytes=0, idle_timeout=30,  
hard_timeout=180, idle_age=22, priority=1, icmp, vlan_tci=0x0000, dl_src=00:00:00:  
00:00:04, dl_dst=00:00:00:00:00:02, nw_src=10.0.1.40, nw_dst=10.0.1.20, nw_tos=0, icm  
p_type=8, icmp_code=0 actions=ANY  
cookie=0x0, duration=12.364s, table=0, n_packets=0, n_bytes=0, idle_timeout=30,  
hard_timeout=180, idle_age=12, priority=1, icmp, vlan_tci=0x0000, dl_src=00:00:00:  
00:00:04, dl_dst=00:00:00:00:00:03, nw_src=10.0.1.40, nw_dst=10.0.1.30, nw_tos=0, icm  
p_type=8, icmp_code=0 actions=ANY  
cookie=0x0, duration=27.356s, table=0, n_packets=2, n_bytes=84, idle_timeout=30  
, hard_timeout=180, idle_age=25, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:04, dl  
dst=00:00:00:00:00:01, arp_spa=10.0.1.40, arp_tpa=10.0.1.10, arp_op=1 actions=FL00  
D  
cookie=0x0, duration=7.348s, table=0, n_packets=2, n_bytes=84, idle_timeout=30,  
hard_timeout=180, idle_age=5, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:04, dl_d  
st=00:00:00:00:00:03, arp_spa=10.0.1.40, arp_tpa=10.0.1.30, arp_op=1 actions=FL00D  
cookie=0x0, duration=26.392s, table=0, n_packets=1, n_bytes=42, idle_timeout=30  
, hard_timeout=180, idle_age=25, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:01, dl  
dst=00:00:00:00:00:04, arp_spa=10.0.1.10, arp_tpa=10.0.1.40, arp_op=2 actions=FL00  
D  
cookie=0x0, duration=6.368s, table=0, n_packets=1, n_bytes=42, idle_timeout=30,  
hard_timeout=180, idle_age=5, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:03, dl_d  
st=00:00:00:00:00:04, arp_spa=10.0.1.30, arp_tpa=10.0.1.40, arp_op=2 actions=FL00D  
cookie=0x0, duration=16.372s, table=0, n_packets=1, n_bytes=42, idle_timeout=30  
, hard_timeout=180, idle_age=15, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:02, dl  
dst=00:00:00:00:00:04, arp_spa=10.0.1.20, arp_tpa=10.0.1.40, arp_op=2 actions=FL00  
D  
cookie=0x0, duration=17.336s, table=0, n_packets=2, n_bytes=84, idle_timeout=30  
, hard_timeout=180, idle_age=15, arp, vlan_tci=0x0000, dl_src=00:00:00:00:00:04, dl  
dst=00:00:00:00:00:02, arp_spa=10.0.1.40, arp_tpa=10.0.1.20, arp_op=1 actions=FL00  
D  
mininet> 
```

3. The output of the `iperf` command measures the bandwidth between two hosts in my network, h1 and h4 by performing a test to send data from one host to another and measuring the amount of data that can be sent in a certain amount of time. The data is sent by TCP, which is a reliable transport protocol, so this measurement also reflects the reliability and performance of my network. The output '18.7 Gbits/sec' and '21.1 Gbits/sec' are the results of my bandwidth test, which indicates that my firewall is allowing TCP traffic. It's telling me that the TCP throughput from h1 to h4 is approximately 18.7 gigabits per second in one test and approximately 21.1 gigabits per second in another test. These values reflect the maximum amount of data that can be transmitted from h1 to h4 under the current network conditions. Below is a screenshot of my output:

A terminal window titled "mininet@mininet-vm: ~" with a menu bar containing "File", "Edit", "Tabs", and "Help". The terminal output shows the command "iperf" being executed, followed by two lines of status information: "\*\*\* Iperf: testing TCP bandwidth between h1 and h4" and "\*\*\* Results: ['18.7 Gbits/sec', '21.1 Gbits/sec']". The prompt "mininet>" is followed by a cursor.

```
mininet@mininet-vm: ~
File Edit Tabs Help
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['18.7 Gbits/sec', '21.1 Gbits/sec']
mininet> 
```