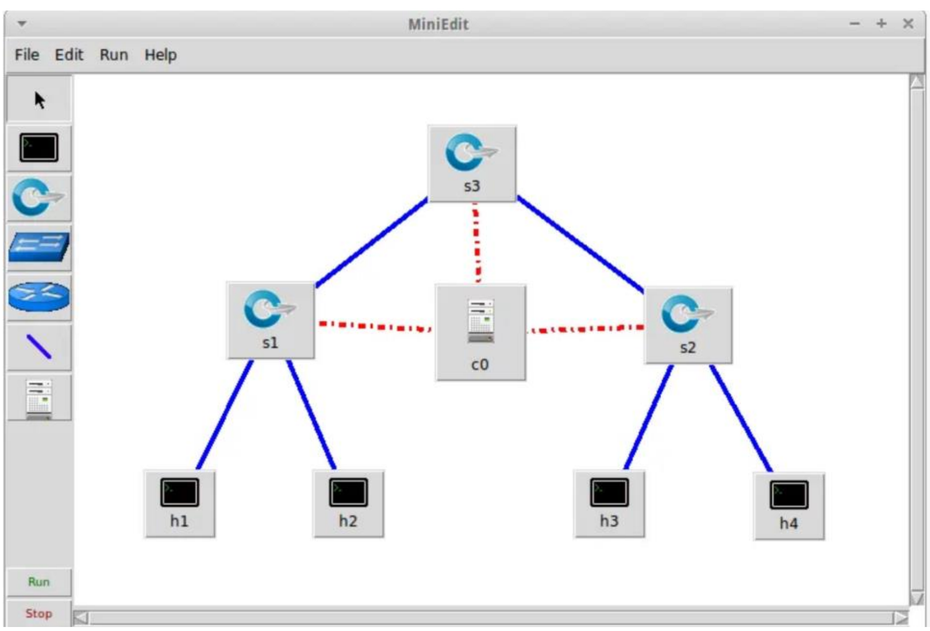
Environment Setup:

First and foremost, SDN has its wide applications in the data centers. So, I have designed a simple data center like fat-tree topology using the mininet emulator. The designed topology is as shown in the figure below.

```
ubuntu@sdnhubvm:~[12:20]$ sudo mn --topo tree,2,2 --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```



As seen in the figure above, we have a control plane, three switches and four hosts. We will be using this topology to perform several types of DoS attacks. But before that, we need to start the SDN controller. The choice of controller I made is a POX controller as it is light weight, flexible and supports python programming.

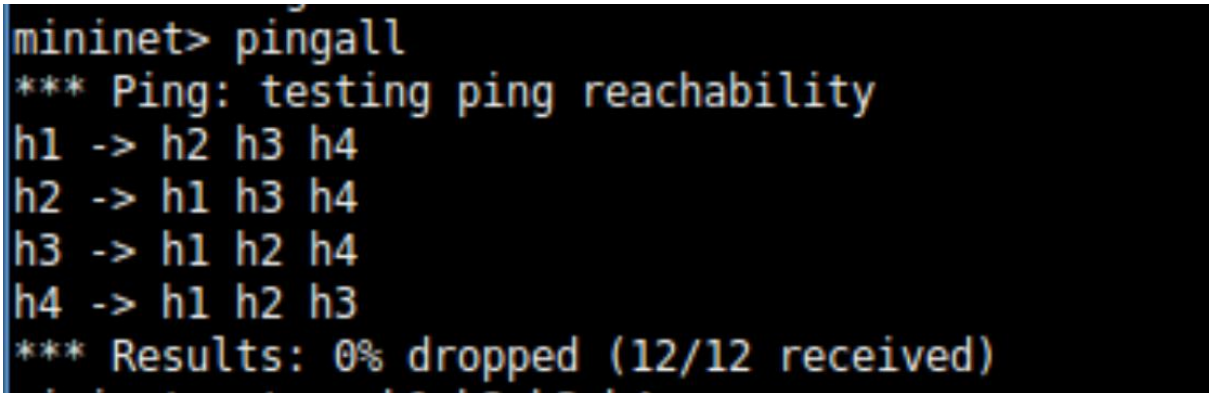
After starting the POX controller, we can see the log that shows the controller starts and connects to the switches previously set up by the Mininet network simulator:

```
Terminal
File  Edit  View  Terminal  Tabs  Help
ubuntu@sdnhubvm:~[12:40]$ sudo ~/pox/pox.py forwarding.l2_pairs \
>
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:forwarding.l2_pairs:Pair-Learning switch running.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-07 1] connected
INFO:openflow.of_01:[00-00-00-00-00-04 4] connected
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
INFO:openflow.of_01:[00-00-00-00-00-06 2] connected
INFO:openflow.of_01:[00-00-00-00-00-02 6] connected
INFO:openflow.of_01:[00-00-00-00-00-03 5] connected
INFO:openflow.of_01:[00-00-00-00-00-05 7] connected
```

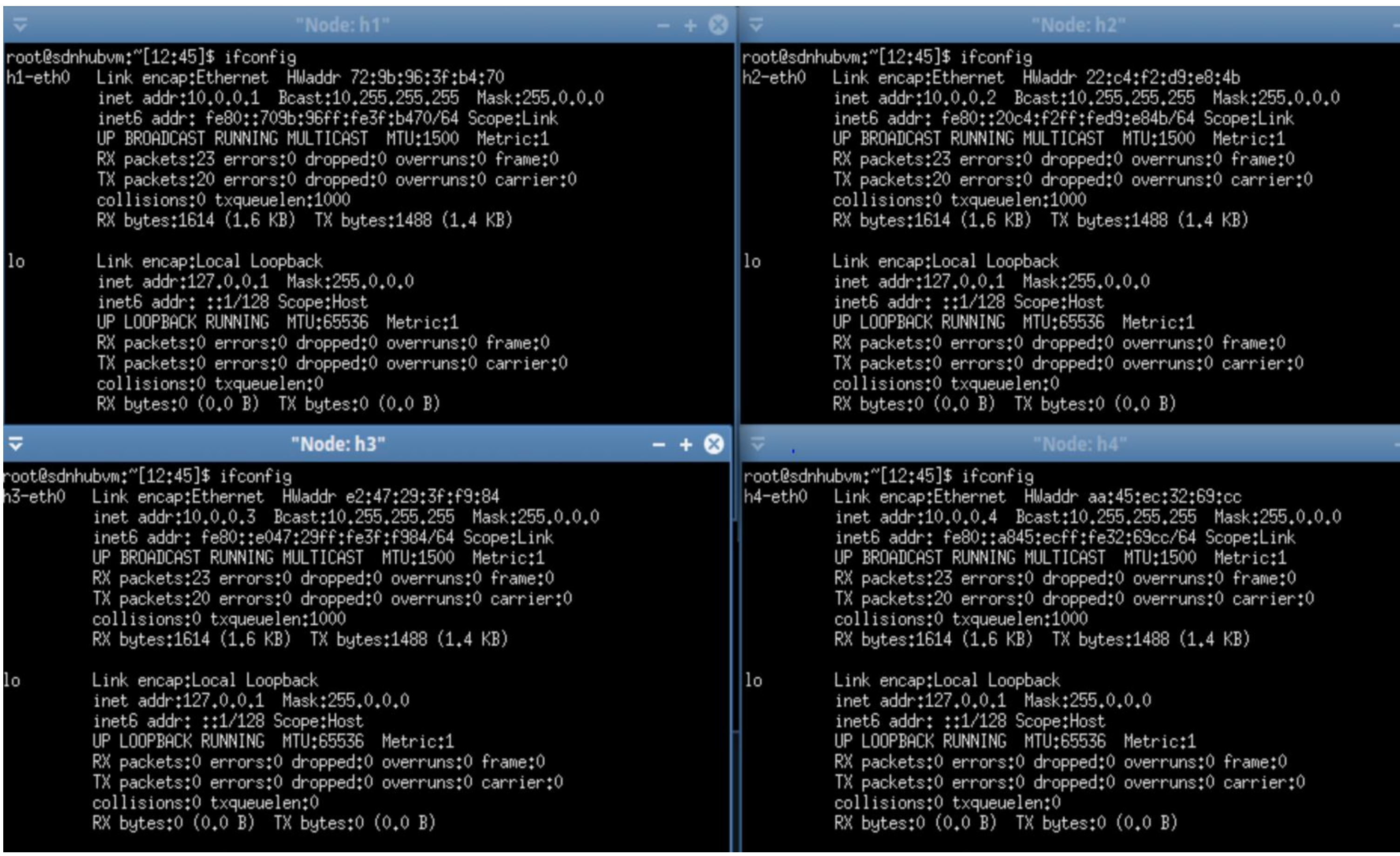
The successful reachability of the network can now be tested inside the mininet:

```
mininet> h2 ping -c5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.507 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.156 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.159 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.157 ms

--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.052/0.206/0.507/0.156 ms
mininet> h4 ping -c5 h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.548 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.078 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.360 ms

--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4032ms
rtt min/avg/max/mdev = 0.078/0.233/0.548/0.190 ms
mininet> h2 ping -c5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.931 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.003 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.155 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.213 ms
```

We can also use the ping all command to test the ping reachability inside the network:


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

The IP addresses of the host machines are:


```
"Node: h1"
root@sdnhubvm:~[12:45]$ ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 72:9b:96:3f:b4:70
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::709b:96ff:fe3f:b470/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1614 (1.6 KB)  TX bytes:1488 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
"Node: h2"
root@sdnhubvm:~[12:45]$ ifconfig
h2-eth0   Link encap:Ethernet  HWaddr 22:c4:f2:d9:e8:4b
          inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::20c4:f2ff:fed9:e84b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1614 (1.6 KB)  TX bytes:1488 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
"Node: h3"
root@sdnhubvm:~[12:45]$ ifconfig
h3-eth0   Link encap:Ethernet  HWaddr e2:47:29:3f:f9:84
          inet addr:10.0.0.3  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::e047:29ff:fe3f:f984/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1614 (1.6 KB)  TX bytes:1488 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

```
"Node: h4"
root@sdnhubvm:~[12:45]$ ifconfig
h4-eth0   Link encap:Ethernet  HWaddr aa:45:ec:32:69:cc
          inet addr:10.0.0.4  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::a845:ecff:fe32:69cc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1614 (1.6 KB)  TX bytes:1488 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Host 1, 2, 3, 4 have the IP addresses 10.0.0.1, 10.0.0.2, 10.0.0.3, 10.0.0.4 respectively. These end systems can be used as targets for DoS attacks. Hence, I will be focusing on using the tool hping3 to perform ten different types of attacks in DoS category and also provide with the analysis part by using Wireshark.