

# **Setting up and security evaluation of SoftwareDefined Networking (SDN) controllers ODL, Mininet, SFlow-RT, SNORT**

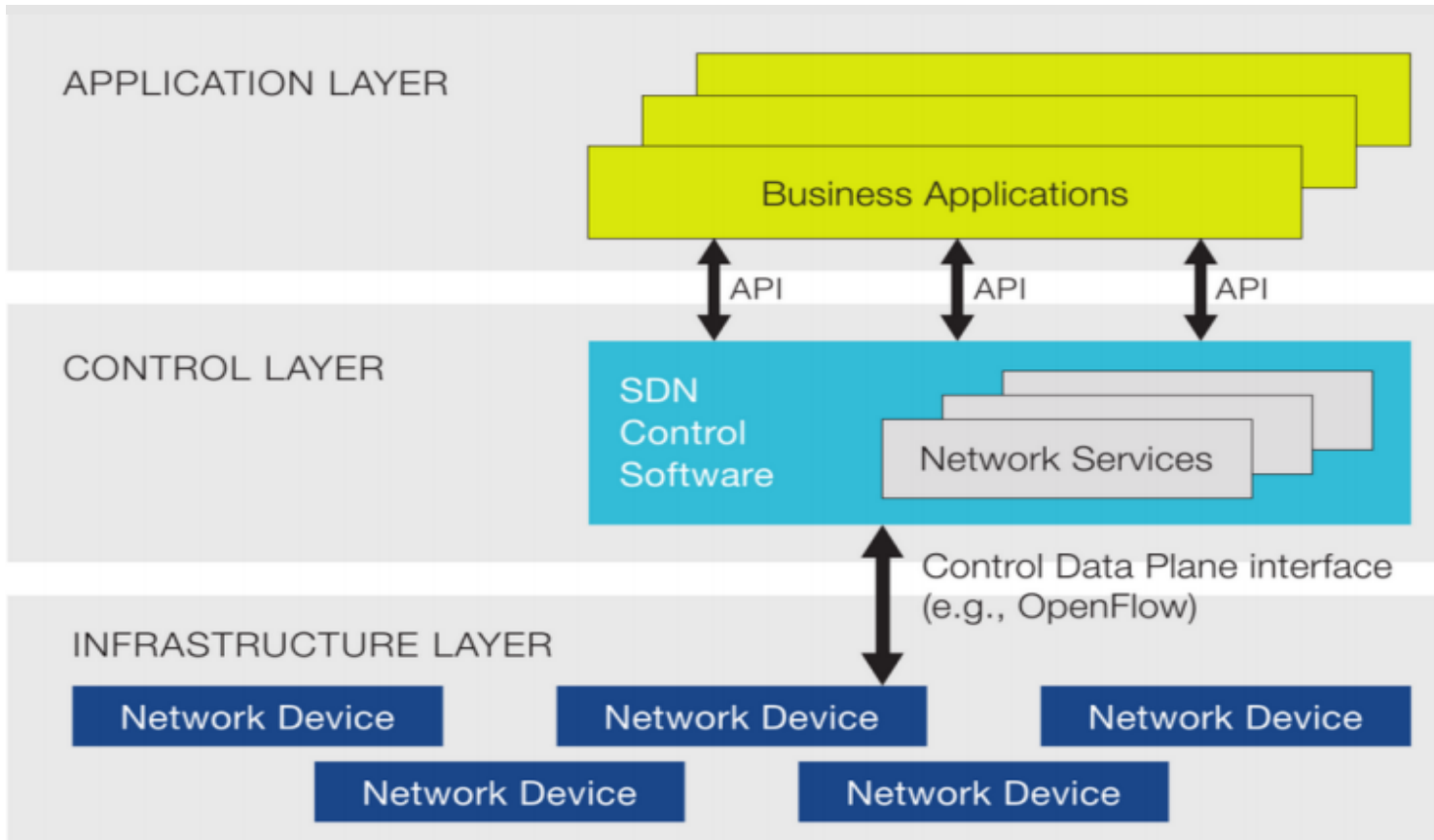
---

By EL HANAFI Maha

# Introduction

- This project analysis focuses on Distributed Denial of Service (DDoS) attacks on SDN networks, in particular, it focuses SYN attack and communications protocol flood attacks. two detection and mitigation methods are projected by adding a light-weight detection and mitigation mechanism at the controller to guard SDN switches and controller from DDoS attacks.
- The presentation has following main objectives
  - summary of SDN and its architecture, followed by introduction of OpenFlow protocol and OpenFlow Switch.
  - Setting up a Software Defined Networks using two virtual machines Mininet and Opendaylight controller.
  - Show the impact of SYN flood DDoS attacks using Hping3
  - DDoS Attack Detection and Mitigation using SFlow in SDN.
  - Propose a second DDoS detection using SNORT IDS in SDN

# SDN Definition



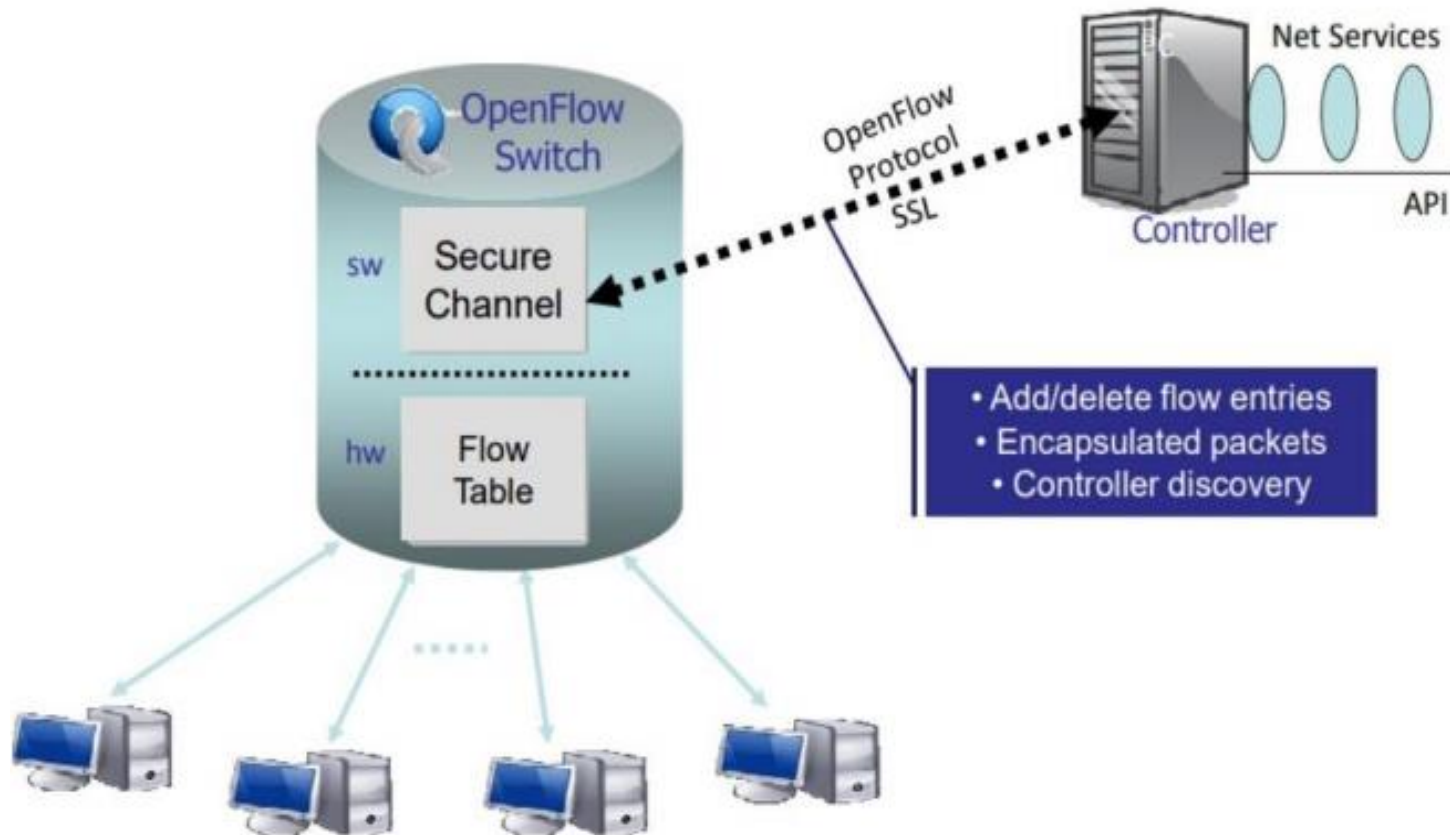
- Software-defined networking technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring, making it more like cloud computing than traditional network management.



# Benefits of SDN

- In order for SDN to deliver on its full promise, it must be enabled by open networking standards that can be easily integrated with current infrastructures. Adopting an SDN methodology has a myriad of benefits including flexibility, scalability, redundancy, and performance. In a traditional network, there might be certain limited hardware and software pieces. When a network requires additional resources, there will be considerable cost in buying new hardware and licensing.

# OpenFlow Protocol and OpenFlow Switch



- OpenFlow is a southbound SDN management standard defined by the Open Networking Foundation (ONF). The OpenFlow protocol describes the communication between the SDN/OpenFlow controller and the OpenFlow compatible switch. The OpenFlow protocol allows the SDN/OpenFlow controller to set up the flow and collect traffic statistics from the switch/router

- An OpenFlow switch includes: (1) a flow table, (2) a secure, The OpenFlow protocol provides a standard form of communication between the controller and the switch



# the implementation of setting up a SDN controller

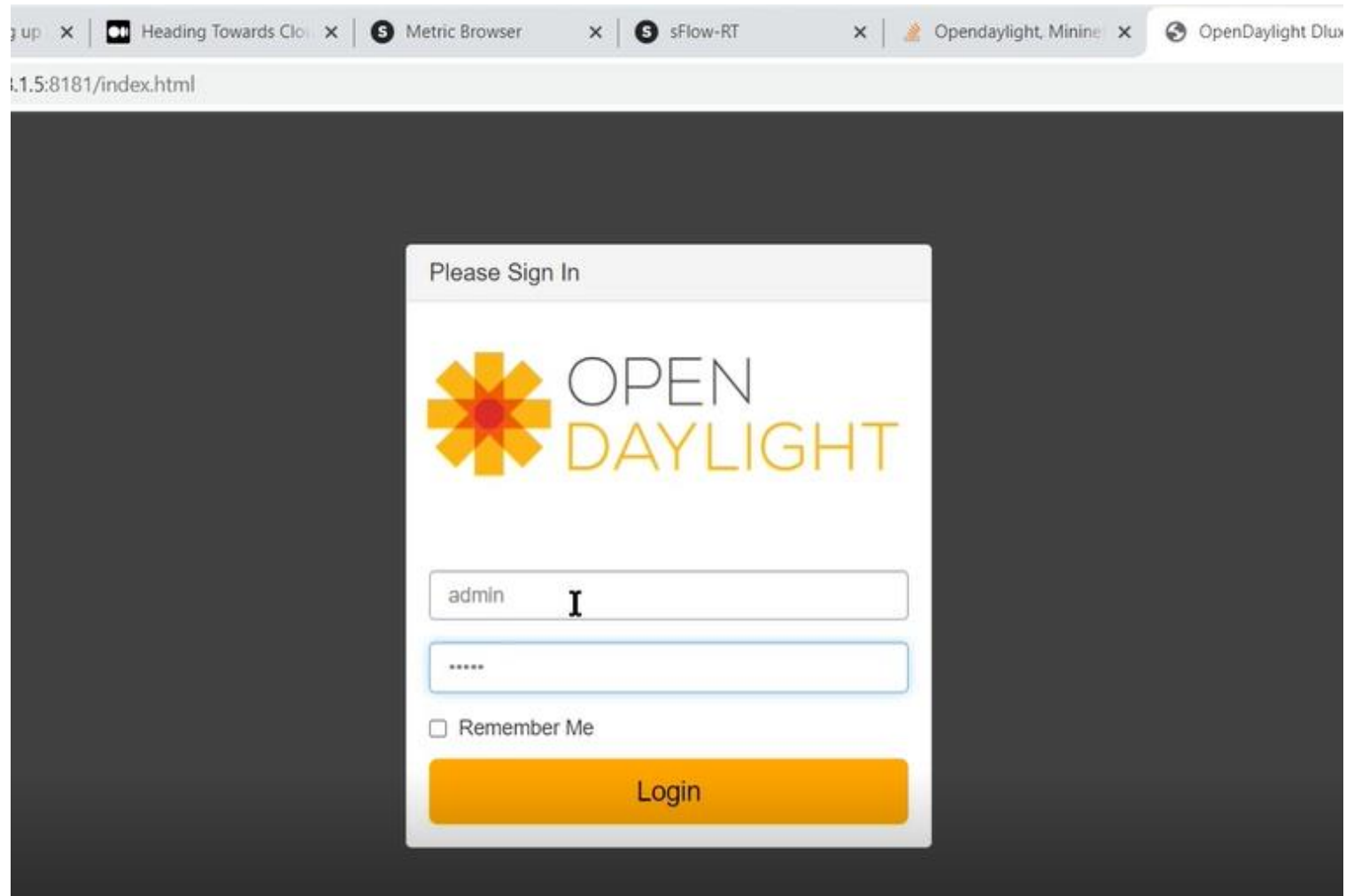
we will use an open-source SDN controller named OpenDaylight. <https://www.opendaylight.org> (first VM)

Network simulation will be done through the mininet tool <http://mininet.org/>. (Second VM)

The management of the SDN network will be done using the OpenDaylight OpenFlow Manager (OFM) tool App,

Simulation of DDos attack using Hping3

DDos attack detection and mitigation using Sflow and SNORT in SDN.



The ODL controller was opened through the web browser using the following link as shown in figure 11.

(<http://192.168.1.15:8181/index.html>)

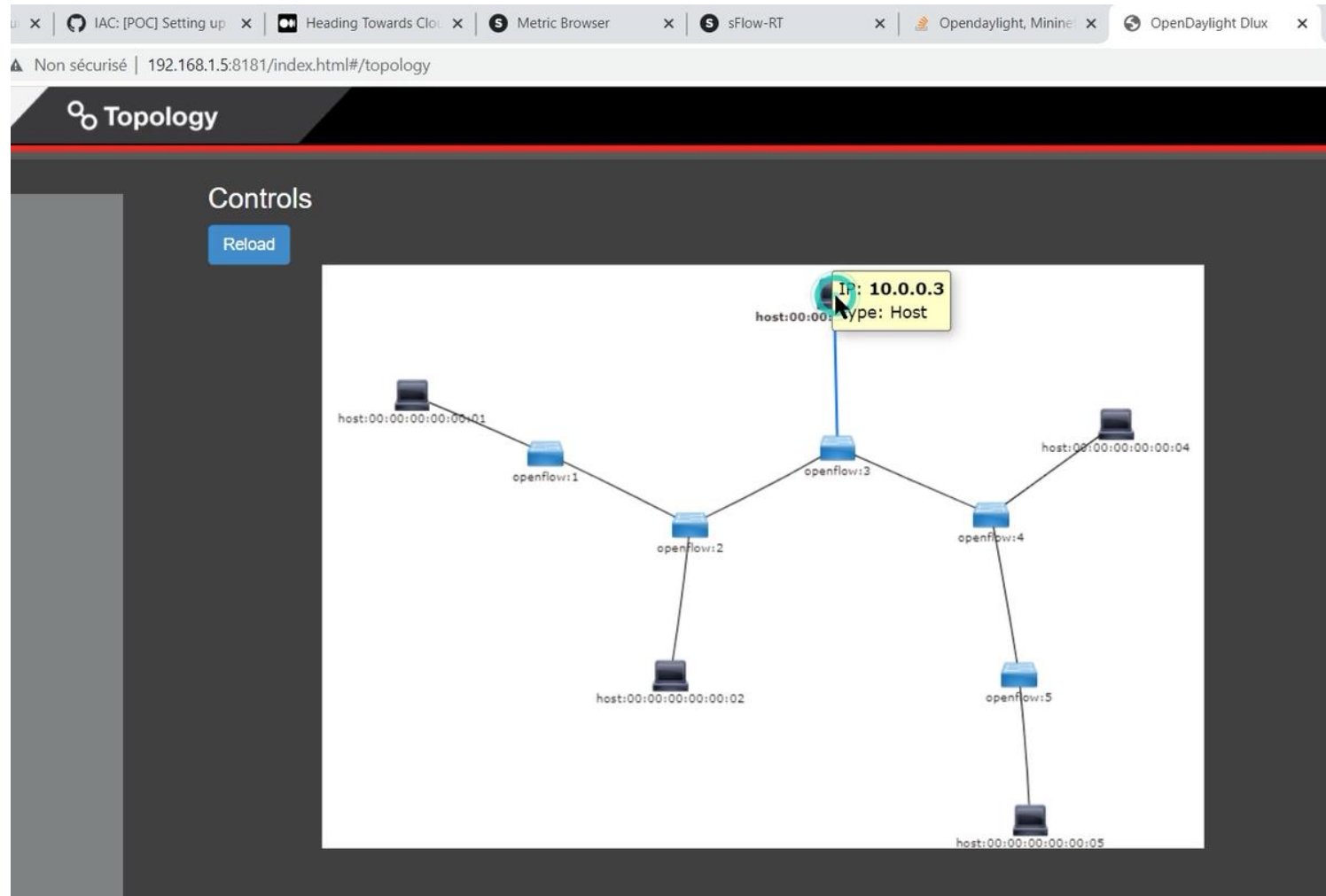
# Installing and testing Mininet

```
mininet@mininet-vm: ~  
odl@ubuntu:~$ ssh -X mininet@192.168.1.6  
mininet@192.168.1.6's password:  
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 4.2.0-27-generic x86_64)  
  
 * Documentation:  https://help.ubuntu.com/  
New release '16.04.7 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Sat Jun 19 04:44:12 2021 from 192.168.1.5  
mininet@mininet-vm:~$ sudo mn --topo linear,5 --mac --controller=remote  
rt=6633 --switch ovs,protocols=OpenFlow13  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2 h3 h4 h5  
*** Adding switches:  
s1 s2 s3 s4 s5  
*** Adding links:  
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5)
```

- To start mininet, you need to define a base topology for it as well as information on the controller:

**sudo mn --topo=linear,5 --controller=remote,ip=192.168.1.5:6633 --switch=ovs,protocols=OpenFlow13 --mac**, this command create a straightforward SDN network with a controller (c0), a 5 switches, and 5 hosts (h1, h2, h3, h4, h5),





Now, if the ODL controller interface are going to be refreshed, the topology and nodes will be shown there as illustrated in this figure

```
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 5% dropped (19/20 received)
mininet> █
```

# Testing communication |

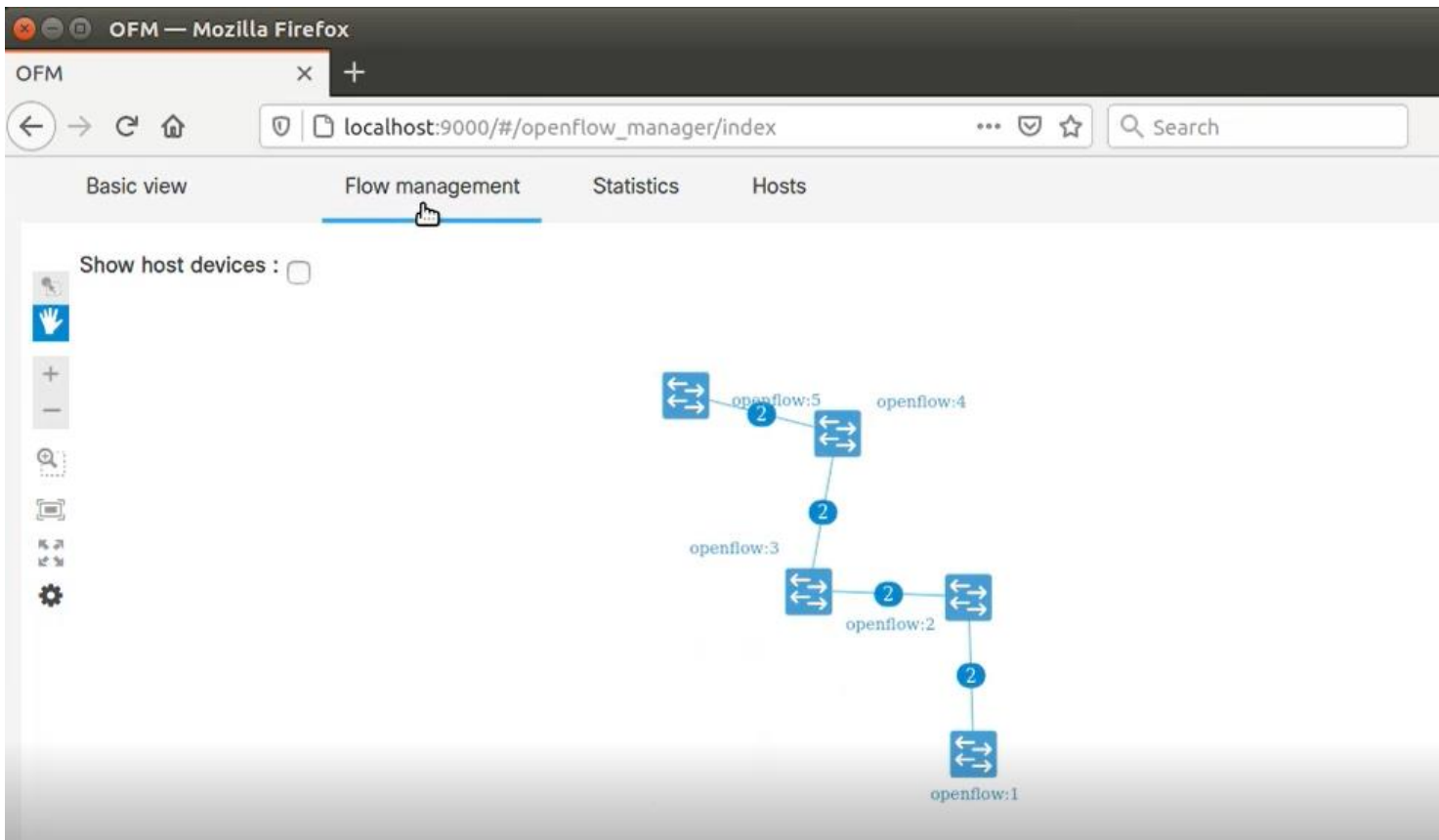
```

mininet> sh ovs-ofctl -O OpenFlow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
  cookie=0x2b000000000000001e, duration=1556.681s, table=0, n_packets=0, n_bytes=0, pr
3 actions=output:2,output:1
  cookie=0x2b000000000000001d, duration=1556.684s, table=0, n_packets=0, n_bytes=0, pr
1 actions=output:2,output:3,CONTROLLER:65535
  cookie=0x2b000000000000001c, duration=1556.687s, table=0, n_packets=0, n_bytes=0, pr
2 actions=output:1,output:3,CONTROLLER:65535
  cookie=0x2b0000000000000006, duration=1562.186s, table=0, n_packets=314, n_bytes=266
dl_type=0x88cc actions=CONTROLLER:65535
  cookie=0x2b0000000000000006, duration=1562.186s, table=0, n_packets=0, n_bytes=0, pr
drop
mininet> _

```

To check that the table is correctly presented on the switch, all you have to do is type the following command: `sh ovs -ofctl -O OpenFlow13 dump-flows s1`

# OpenDaylight OpenFlow Manager (OFM) App



- -Once the OpenFlow Manager has been launched you can launch it through the browser, you just have to type:

[http://192.168.x.x:9000/#/openflow\\_manager/index](http://192.168.x.x:9000/#/openflow_manager/index)

We go to the OFM configuration interface, under the Flow Management tab, we can add a new flow specifying that we want the traffic not to go through the link between the node2 and node1.

- This figure shows the general properties of our "abc" stream, we can see that the port we want apply this flow is the link between OF switches 1 and 2, and the action to be taken is packet DROP. This rule will generate the following stream entry on OF switch 2

The screenshot shows the OpenFlow Manager (OFM) configuration interface. The 'Flow management' tab is active. On the left, there are sections for 'Match' and 'Actions'. The 'Match' section has 'In port' selected. The 'General properties' section shows 'Table' as 0, 'ID' as 'abc', 'Priority' as 300, and 'In port' as 'openflow:2:1'. The 'Actions' section is empty.

This screenshot is a zoomed-in view of the 'Flow management' tab. It shows the 'Match' section with 'In port' selected. The 'General properties' section shows 'Table' as 0, 'ID' as 'abc', 'Priority' as 300, and 'In port' as 'openflow:2:1'. The 'Actions' section has 'Drop' selected.

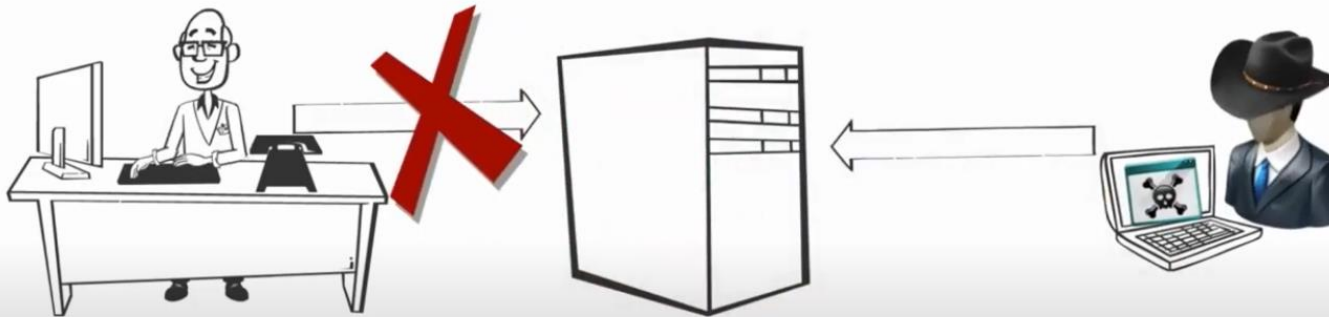


After applying the rule, we try a new accessibility test to verify that our rule takes effect

```
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)
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4
h2 -> X X X
h3 -> h1 X h4
h4 -> h1 X h3
*** Results: 50% dropped (6/12 received)
mininet>
```

# Simulation Attack: DDOS Attacks using Hping3

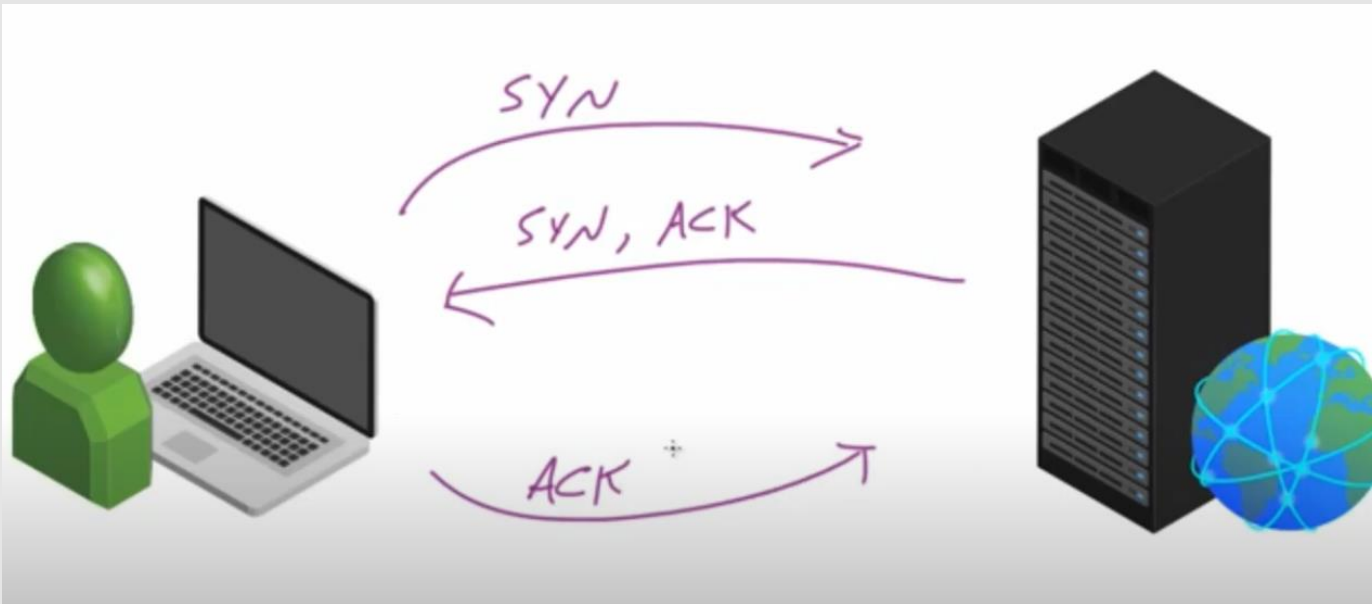
Denial of Service (DoS)



- What is DDoS attack ?

Attacker makes a system or data unavailable to someone who need its,

# Simulation Attack: DDOS Attacks using Hping3



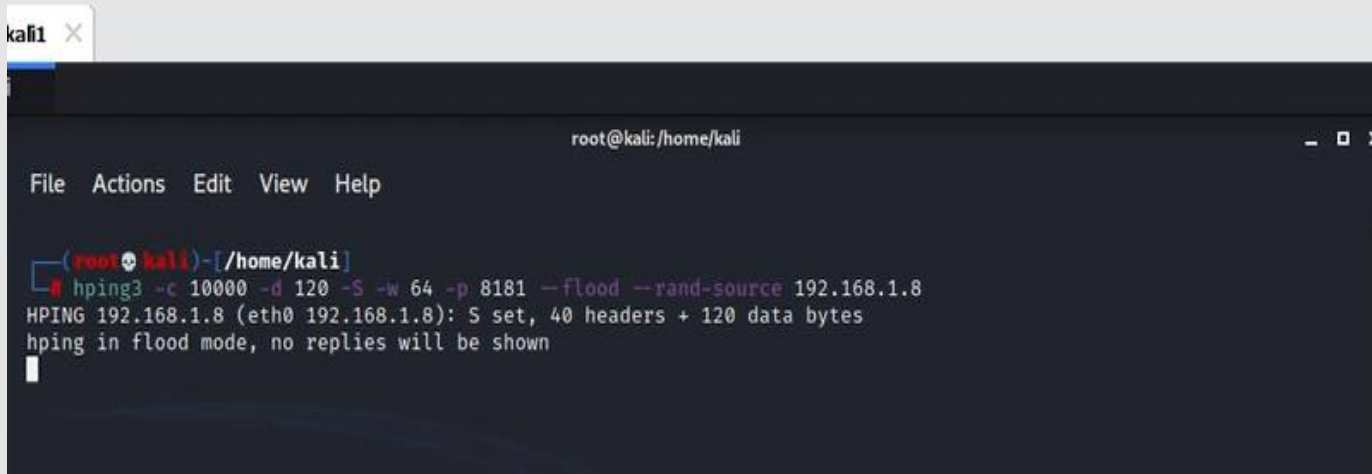
- SYN Concept

Exchange a series of messages to start A TCP connection between client and server,

# Simulation Attack: DDOS Attacks using Hping3

- SYN FLOOD:
- Attacker sends a succession of SYN REQUESTS to target's system in attempt to consume enough server resources to make the system unresponsive
- Hping:
- A network tool able to send custom TCP/IP PACKETS and to DISPLAY target Replies

# Simulation Attack, DDOS Attacks using Hping3: Demonstration



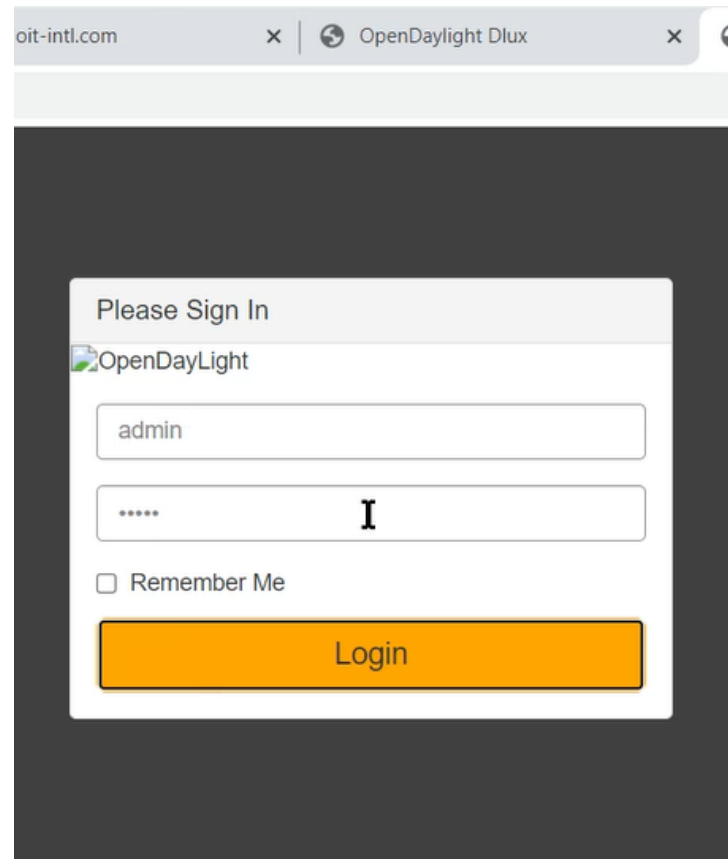
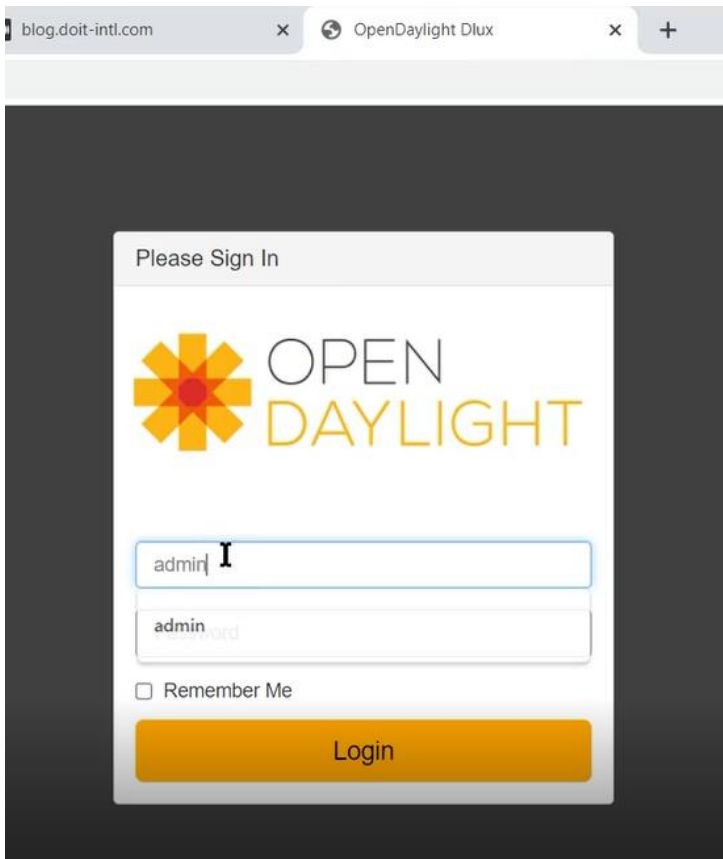
The screenshot shows a terminal window titled 'kali1' with a menu bar (File, Actions, Edit, View, Help) and a title bar (root@kali: /home/kali). The prompt is '(root@kali)~[/home/kali]'. The command executed is 'hping3 -c 10000 -d 120 -S -w 64 -p 8181 --flood --rand-source 192.168.1.8'. The output is 'HPING 192.168.1.8 (eth0 192.168.1.8): S set, 40 headers + 120 data bytes' followed by 'hping in flood mode, no replies will be shown'.

```
File Actions Edit View Help
root@kali: /home/kali
(root@kali)~[/home/kali]
# hping3 -c 10000 -d 120 -S -w 64 -p 8181 --flood --rand-source 192.168.1.8
HPING 192.168.1.8 (eth0 192.168.1.8): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

- Above Hping3 command includes following parameters:
  - -c 10000 - Number of packets that must be sent
  - -d 120 - packet size
  - -S - TCP SYN packet
  - -w 64 - TCP Window Size
  - -p 8181 - Destination Port
  - --flood - To send packets in burst mode
  - --rand-source - source address will be randomly generated
  - 192.168.1.8 - ODL Ip address



# Result: ODL controller before and after performing DDoS testing,



- the impact of DDoS attack as access to odl is lost after performing DDoS on ODL controller

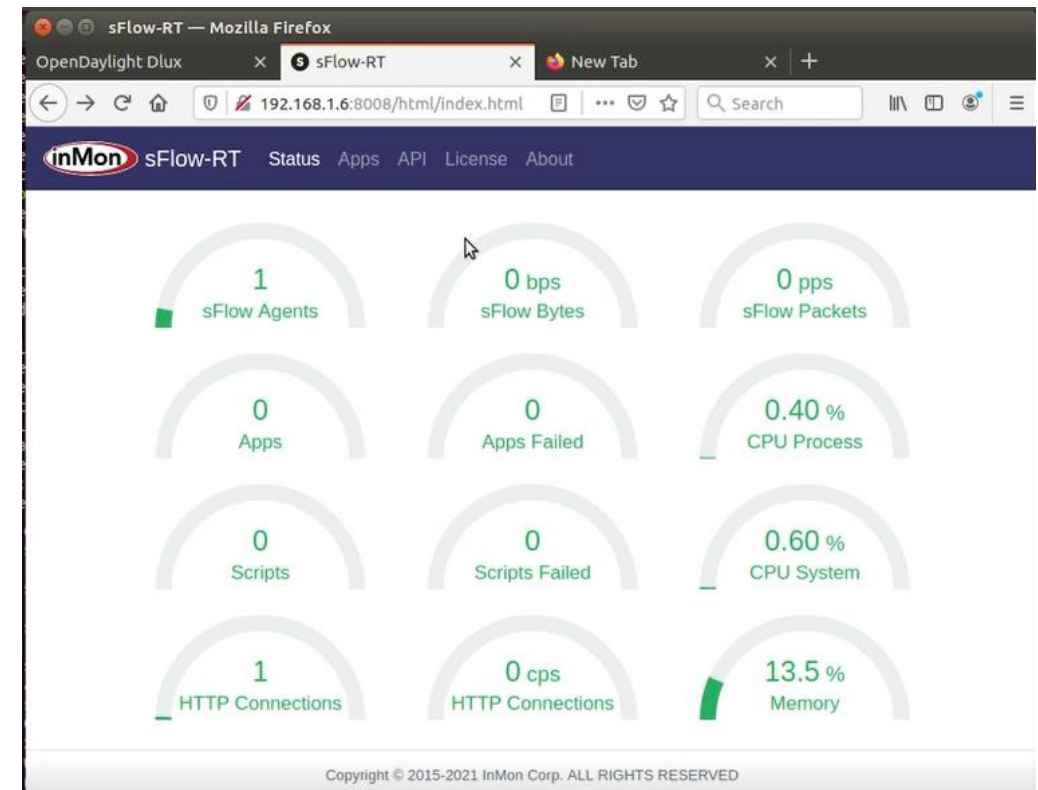
# DDoS Detection and mitigation using SFlow

- Enable sflow in s1: `sudo ovs-vsctl --id=@sflow create sflow agent=eth0 target=\"192.168.1.6:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow`, Then running the SFlow app by `./start.sh`

```
mininet@mininet-vm: ~
odl@ubuntu:~$ ssh -X mininet@192.168.1.6
mininet@192.168.1.6's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 4.2.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Jun 19 04:37:05 2021
mininet@mininet-vm:~$ sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 targ
et=\"192.168.1.5:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow
3b197c26-9132-4a98-9422-86d2e2dc8b85
mininet@mininet-vm:~$
```



# DDoS Detection and mitigation using SFlow

```
"Node: h2"
h2-eth0  Link encap:Ethernet  HWaddr 00:00:00:00:00:02
         inet addr:10.0.0.2  Bcast:10.255.255.255  Mask:255.0.0.0
         UP BROADCAST RUNNING MULTICAST  MTU:1500  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:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         UP LOOPBACK RUNNING  MTU:65536  Metric:1
         RX packets:951 errors:0 dropped:0 overruns:0 frame:0
         TX packets:951 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:3286760 (3.2 MB)  TX bytes:3286760 (3.2 MB)

root@mininet-vm:~# sudo hping3 --faster --rand-source 10.10.10.2
HPING 10.10.10.2 (h2-eth0 10.10.10.2): NO FLAGS are set, 40 headers + 0 data byt
es
^C
--- 10.10.10.2 hping statistic ---
35363838 packets transmitted, 0 packets received, 100% packet loss
```

- Next step we are going to simulate a DDoS attack using hping3 from h2(10.10.10.2) towards h1(10.10.10.1). The packet rate is going to be about hundred packets per second and all of the packets are going to send from random IPs, we type in the shell of h2: **sudo hping3 --faster --rand-source 10.10.10.1**

Open wireshark and capture the  
wireshark traces, and we open the  
sflow-rt dashboard  
<http://192.168.1.6:8008/index.html>,

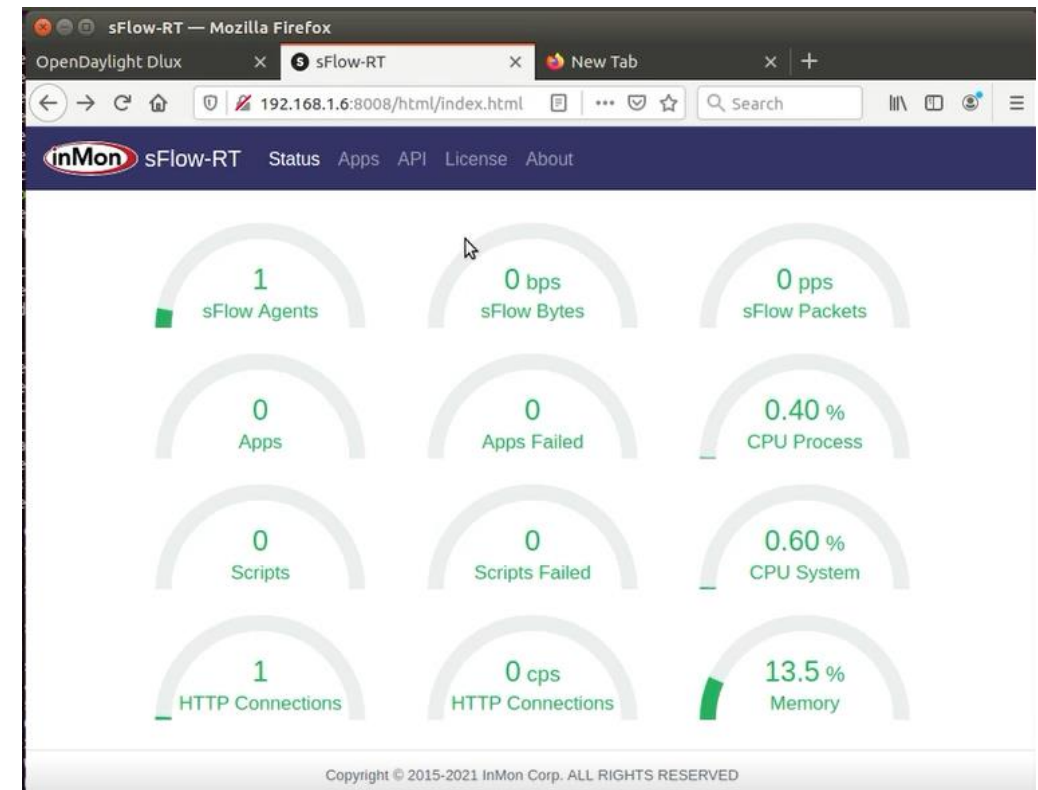
- As the figure(sflow-rt) below shows there is one yes flow agent collector, we configure the switch s1

Capturing from eth0 [Wireshark 1.10.6 (v1.10.6 from master-1.10)]

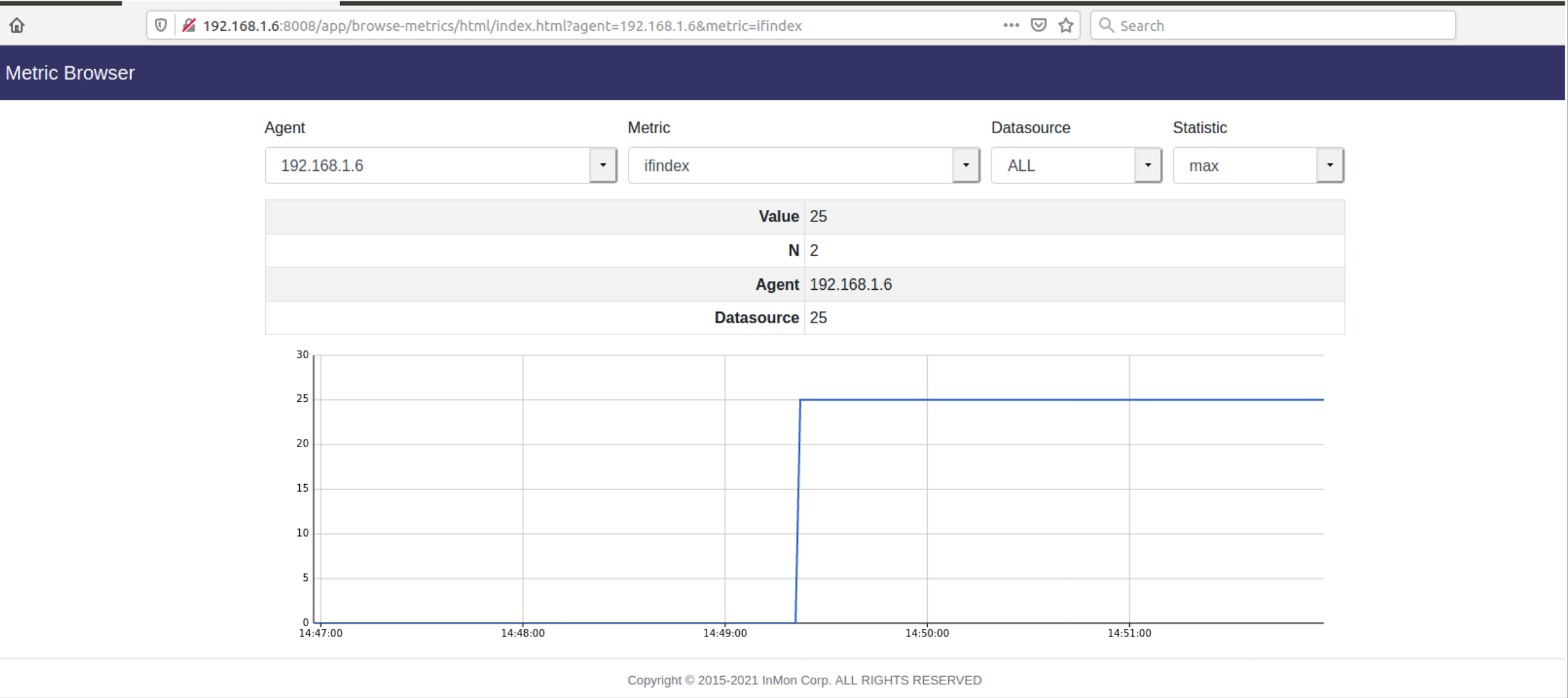
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
84	0.477431000	192.168.1.5	192.168.1.6	TCP	66	6633 > 39684 [ACK] Seq=57 Ack=489 Win=0 Len=0
85	0.512459000	192.168.1.5	192.168.1.6	OF 1.3	74	of_barrier_request
86	0.512605000	192.168.1.6	192.168.1.5	OF 1.3	74	of_barrier_reply
87	0.512965000	192.168.1.5	192.168.1.6	TCP	66	6633 > 39692 [ACK] Seq=129 Ack=7017 Win=0 Len=0
88	0.663439000	192.168.1.5	192.168.1.6	OF 1.3	82	of_table_stats_request
89	0.664108000	192.168.1.6	192.168.1.5	TCP	2962	[TCP segment of a reassembled PDU]
90	0.664220000	192.168.1.6	192.168.1.5	TCP	2962	[TCP segment of a reassembled PDU]
91	0.664260000	192.168.1.6	192.168.1.5	OF 1.3	386	of_table_stats_reply
92	0.664545000	192.168.1.5	192.168.1.6	TCP	66	6633 > 39686 [ACK] Seq=17 Ack=6113 Win=0 Len=0
93	0.669414000	192.168.1.5	192.168.1.6	OF 1.3	122	of_flow_stats_request
94	0.669728000	192.168.1.6	192.168.1.5	OF 1.3	418	of_flow_stats_reply
95	0.677730000	192.168.1.5	192.168.1.6	OF 1.3	90	of_port_stats_request
96	0.677911000	192.168.1.6	192.168.1.5	OF 1.3	418	of_port_stats_reply
97	0.678704000	192.168.1.5	192.168.1.6	OF 1.3	90	of_queue_stats_request
98	0.678784000	192.168.1.6	192.168.1.5	OF 1.3	82	of_queue_stats_reply
99	0.683521000	192.168.1.5	192.168.1.6	OF 1.3	82	of_table_stats_request



After performing the Ddos Attack, we can see the network graphics. As shown in the figure below, how many bits per second it's coming from the interfaces





```
mininet> sh ovs-ofctl -O OpenFlow13 dump-ports s1
OFPST_PORT reply (OF1.3) (xid=0x2): 4 ports
  port 3: rx pkts=326, bytes=27710, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=326, bytes=27710, drop=0, errs=0, coll=0
          duration=1622.606s
  port 1: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=326, bytes=27710, drop=0, errs=0, coll=0
          duration=1622.61s
  port 2: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=326, bytes=27710, drop=0, errs=0, coll=0
          duration=1622.604s
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
              tx pkts=326, bytes=31296, drop=0, errs=0, coll=0
              duration=1622.572s
mininet> _
```

To check that the table is correctly presented on the switch and how many packets send before and after performing DDoS attack, all you must do is type the following command `sh ovs -ofctl -O OpenFlow13 dump-flows s1,`

# DDoS Detection and mitigation using snort

- SNORT IDS/IPS:

Snort could be a network intrusion detection system that is capable of capturing the real time traffic analysis and packet pursuit on the networks also as carrying out some of the functions such as protocol analysis, content matching, port scans, cgi attacks and also detect form of attacks.

# DDoS Detection and mitigation using snort

```
"Node: h1"
root@mininet-vm:~# sudo ping -s 1 -f 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 1(29) bytes of data.
.....E.....E.....

--- 10.10.10.3 ping statistics ---
36 packets transmitted, 0 received, +24 errors, 100% packet loss, time 8363ms
pipe 12
root@mininet-vm:~# S
```

- In this part, we simulate ICMP flooding, For the ICMP flood implementation, a random host is selected as an attacker on a network who tries to flood other hosts on the network. The command mentioned below is used to launch an ICMP flood attack, Command:

**sudo ping -s 100 -f  
"IPaddress to flood"**

The above illustration shows that host h1 is flooding host h3, with host h3 receiving a high rate of packet flow from host h3, which slows host h3's performance or causes the host to refuse the services it can provide. Snort command to monitor network interface:

**sudo snort -q -A console -u snort -g  
snort -c /etc/snort/snort.conf -i  
"interfaae to monitor"**

# DDoS Detection and mitigation using snort

```
/27-22:42:52.655964 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656006 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656045 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656098 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656141 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656180 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656235 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656276 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656315 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656370 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656412 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656451 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656505 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656547 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656586 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656641 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656684 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.656723 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.657879 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.657975 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.658039 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.658101 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.658148 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.658471 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
/27-22:42:52.658629 [**] [1:1000001:1] FTP connection attempt [**] [Classification ID: 0] [Priority ID: 0] {ICMP
```

- Snort rule to detect ICMP flooding:  
**alert ICMP any any -> any any(flags: s; msg: “ICMP flood detected”; flow: stateless; threshold: track by\_src, count 1000, seconds 2; sid:10002; rev: 2;)**

**The Snort interface monitors the switch interface, which detects that the flood is occurring, as shown in this Figure**

# DDoS Detection and mitigation using snort

```
mininet> sh ovs-ofctl -O OpenFlow13 dump-ports s1
OFPST_PORT reply (OF1.3) (xid=0x2): 3 ports
  port 1: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=0, bytes=0, drop=0, errs=0, coll=0
          duration=484.296s
  port 2: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
          tx pkts=0, bytes=0, drop=0, errs=0, coll=0
          duration=484.296s
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
              tx pkts=0, bytes=0, drop=0, errs=0, coll=0
              duration=484.269s
```

- To check whether the rule has been inserted into a device in the network, use the following command:  
**sudo ovs-ofctl dump-flows s1** This command displays the new flow rule inserted into the switch,

➤ Once the rule is successfully sent to the switch, the data packet from the specific host will be discarded, preventing the attacking host from flooding other hosts on the network.



# Conclusion

- In this project we present a lightweight DDoS defense mechanism based on SDN. The aim of this thesis is to show how SDN technology can help defend against different types of DDoS attacks. We have proposed two detection and mitigation methods to protect SYN and ICMP floods. We implement these methods in three sets of modules, SFlow-rt, which collects flow statistics from OpenFlow switches, and SNORT, which detects anomalies in network traffic.



- THANK YOU FOR YOUR ATTENTION