Trusted Communication in Software Defined Distributed Systems

Abstract

The internet, since the advent of ARPANET, has come along a very long way. It has undoubtedly changed millions of lives and even now is in its infancy. Software Defined Networking (SDN) is presented as a paradigm shift in this regard. It strives to standardize the networking on all levels. This is an initiative to redesign the current networking stack and compartmentalize into three main planes, the data plane, the control plane and the management plane, respectively moving from bottom up. We, here, have put an effort to augment the idea of SDN to a more distributed framework. Using cleverly designed topologies like Spine leaf, we demonstrate the interconnection of controllers using relay system designed from bottom up as the first phase. The second phase, on other hand, acknowledges the need to secure such translations and we try to mitigate Denial of Service (DoS) attacks on the control plane.

What are Software Defined Networks?

·In a software-defined network, a network engineer or administrator can shape traffic from a centralized control console.

·The centralized SDN controller directs the switches to deliver network services wherever they're needed.

·A typical representation of SDN architecture comprises three layers: the application layer, the control layer and the infrastructure layer.

SDN Architecture

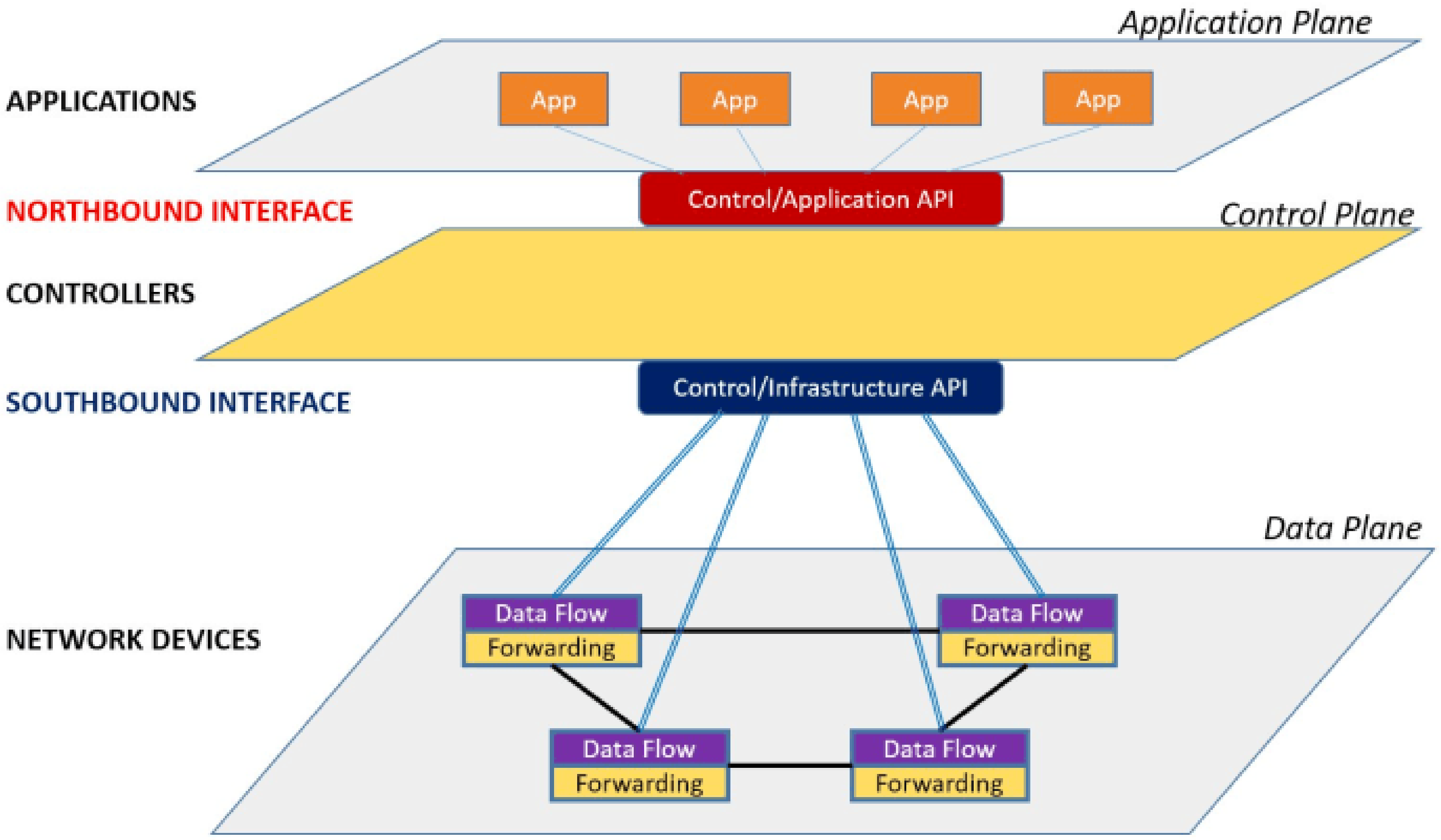


Figure 1 - Software-Defined Networking – A high level architecture

Introduction

- This is an initiative to redesign the current networking stack and compartmentalize into three main planes, the data plane, the control plane and the management plane.
- Using cleverly designed hybrid topologies, we demonstrate inter-controller connection in a distributed environment in first phase.
- The second phase acknowledges the need to secure such translations and we try to mitigate Denial of Service (DoS) attacks on the control plane.

Current Situation

1)Current SDN industry standard controller implementations do not inherently support distributed inter-controller communication.

2)Some of them who do, use infeasible and expensive mesh topologies.

3)SDN, thus, is limited to single controller and non-scalable networks.

4)Inter-controller communication of different types is also a problem.

Limitations of Current System

1)Less scalability factor.

2)More energy consumption.

3)Greater expense for physical cables.

4)More vulnerable area to secure.

5)$(n-1)^n$ connections within controllers due to Mesh topology.

Proposed System

1)We propose a relay to act as a bridge between the controllers in a distributed system.

2)The relays can be sub-relayed as per geographical requirements.

3)Controllers use relay as proxy to broadcast flow query in the network.

4)A duplex connection between each controller and relay facilitates simultaneous broadcast and reply.

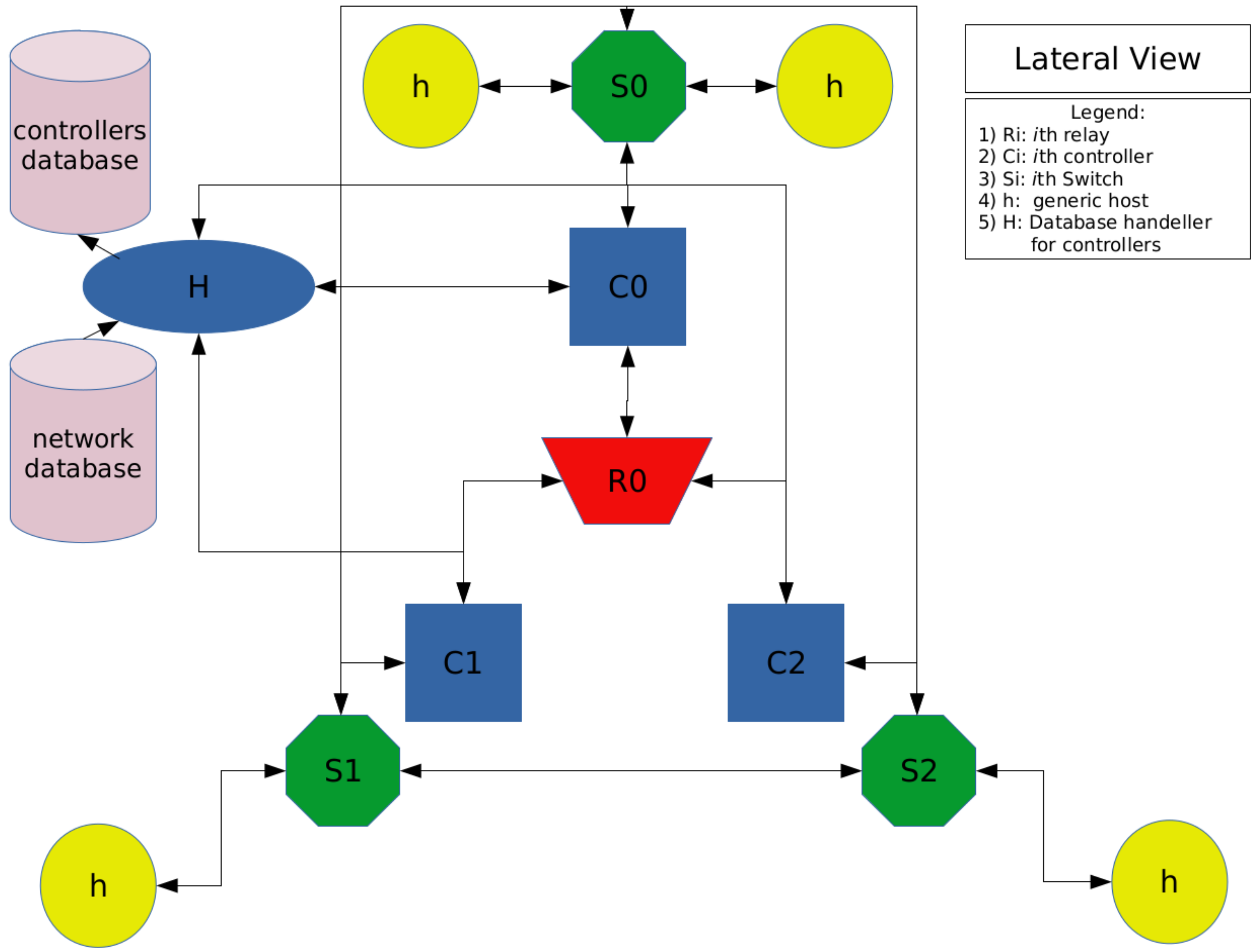
5)Bottlenecks are eliminated using frequent multi threaded constructs.

6)A TCP server listening for particular connections at controller handler, relay and controller as well for the duplex connection.

7)Python/java has been used as controller implementation language to interface with dynamic shared libraries coded in pure C, while relay engine remaining in pure C.

Review 1

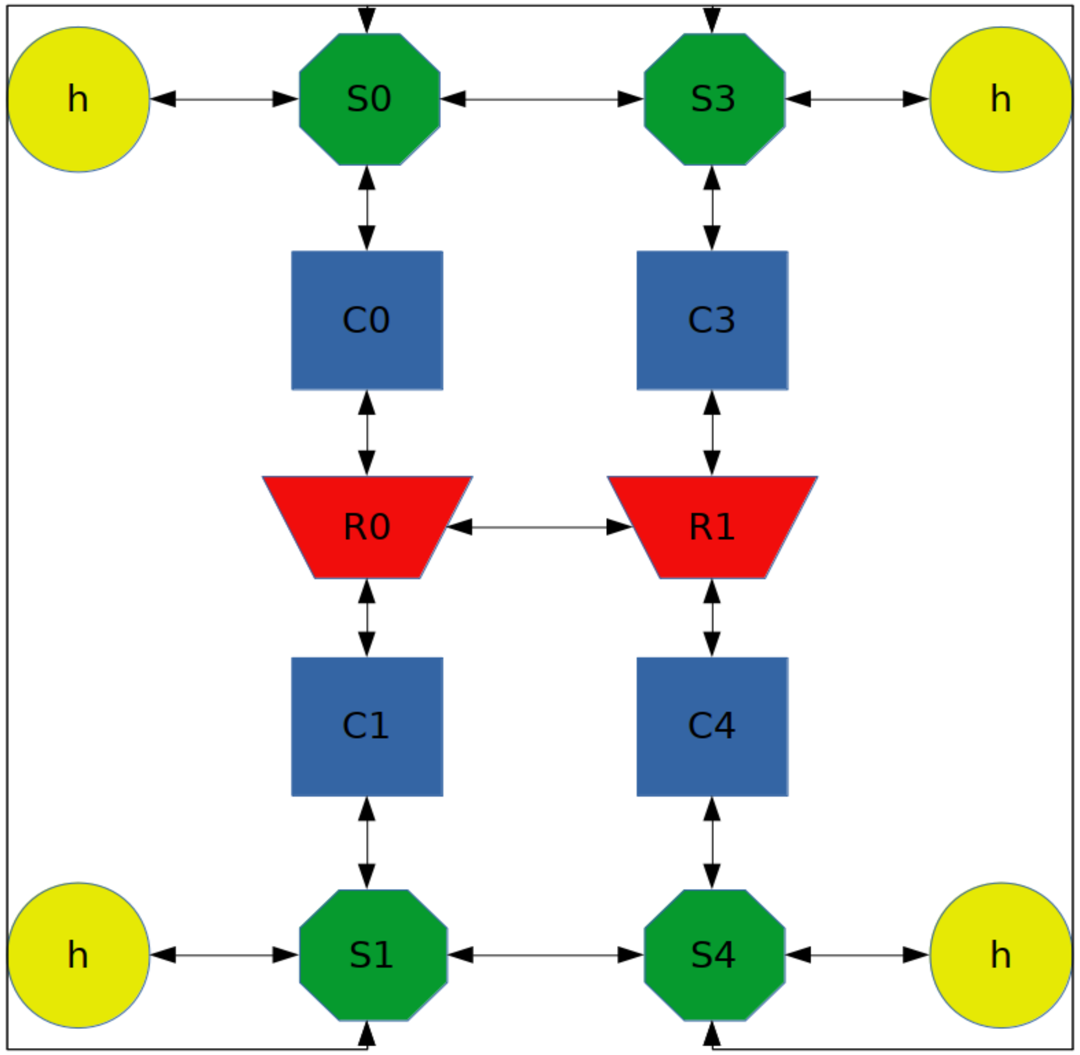
Designed Architecture



Lateral View

Legend:
1) Ri: *i*th relay
2) Ci: *i*th controller
3) Si: *i*th Switch
4) h: generic host
5) H: Database handeller
for controllers

Cross-Sectional View (Observer’s left)



Legend:

1) Ri: *ith* relay
2) Ci: *ith* controller
3) Si: *ith* Switch
4) h: generic host
5) H: Database handeller
   for controllers

How the Architecture works?

1)Database docker image starts

2)Controller handler start connecting to ‘controllers’ database.

3)Controllers boot sending information to controller handler to be registered.

4)Topology simulator fetches the Controller Database

5)User Input for the required number of hosts and switches.

6)Random pairing for switches and hosts using the libsodium library.

7)Random number of hosts connected to the switches.

8)Topology is described as a 3d array, where the entries in database are separate tables with each controller has a separate table with its IP as the name for each table.

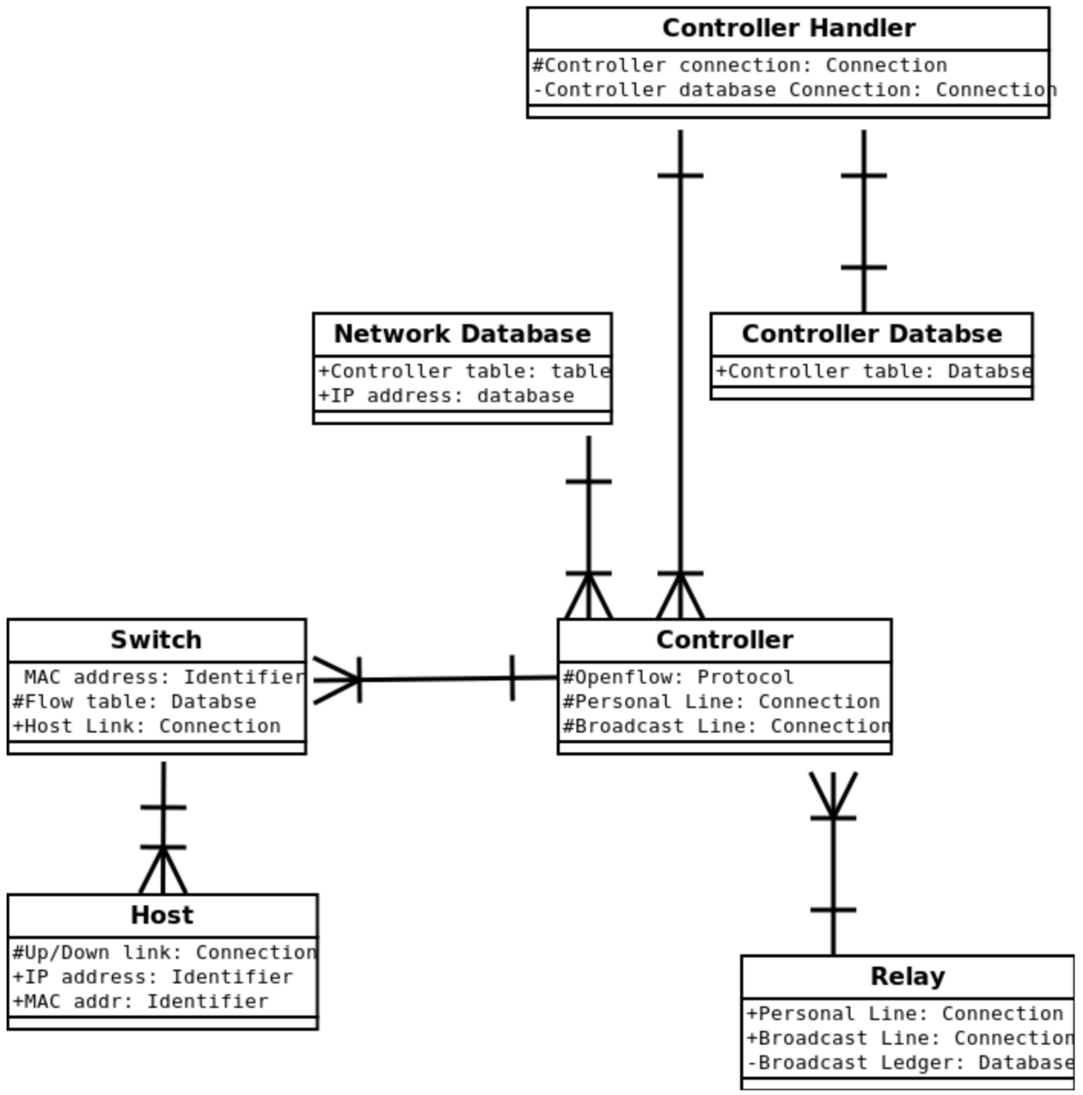
9)Class A Ips are automatically assigned by the Mininet.

10) Static Ips assigned to each controller on connection.

11) Standalone controller connection through relay.

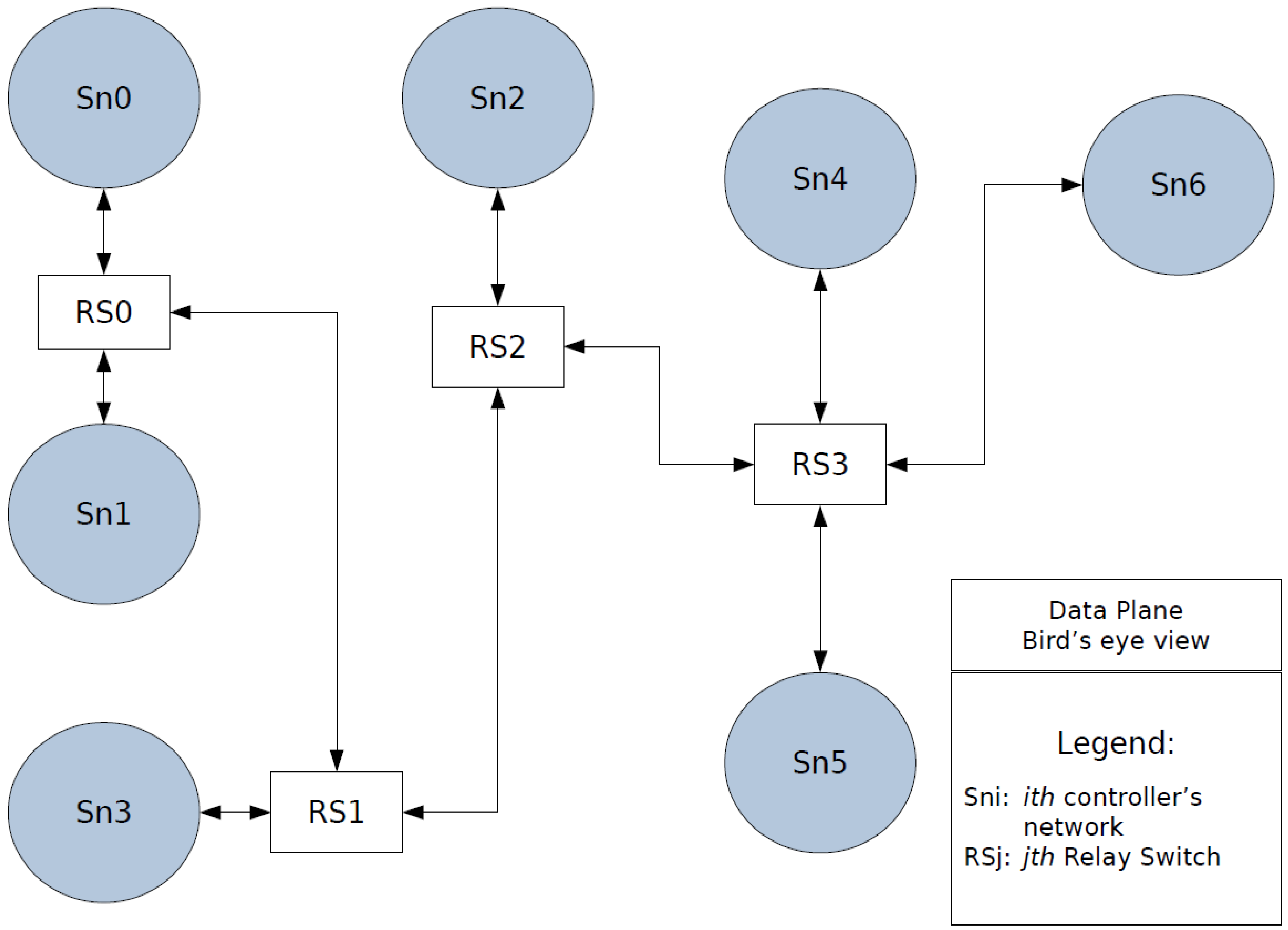
12) Network database accessed by controller upon the connection.

Workflow


Controller Handler
#Controller connection: Connection
-Controller database Connection: Connection
Network Database
+Controller table: table
+IP address: database
Controller Databse
+Controller table: Databse
Switch
MAC address: Identifier
#Flow table: Databse
+Host Link: Connection
Controller
#Openflow: Protocol
#Personal Line: Connection
#Broadcast Line: Connection
Host
#Up/Down link: Connection
+IP address: Identifier
+MAC addr: Identifier
Relay
+Personal Line: Connection
+Broadcast Line: Connection
-Broadcast Ledger: Database

Review 2

Some updations in Architecture


Sn0
Sn2
Sn4
Sn6
RS0
RS2
RS3
Sn1
Sn3
RS1
Sn5
Data Plane
Bird's eye view
Legend:
Sni: *ith* controller's network
RSj: *jth* Relay Switch

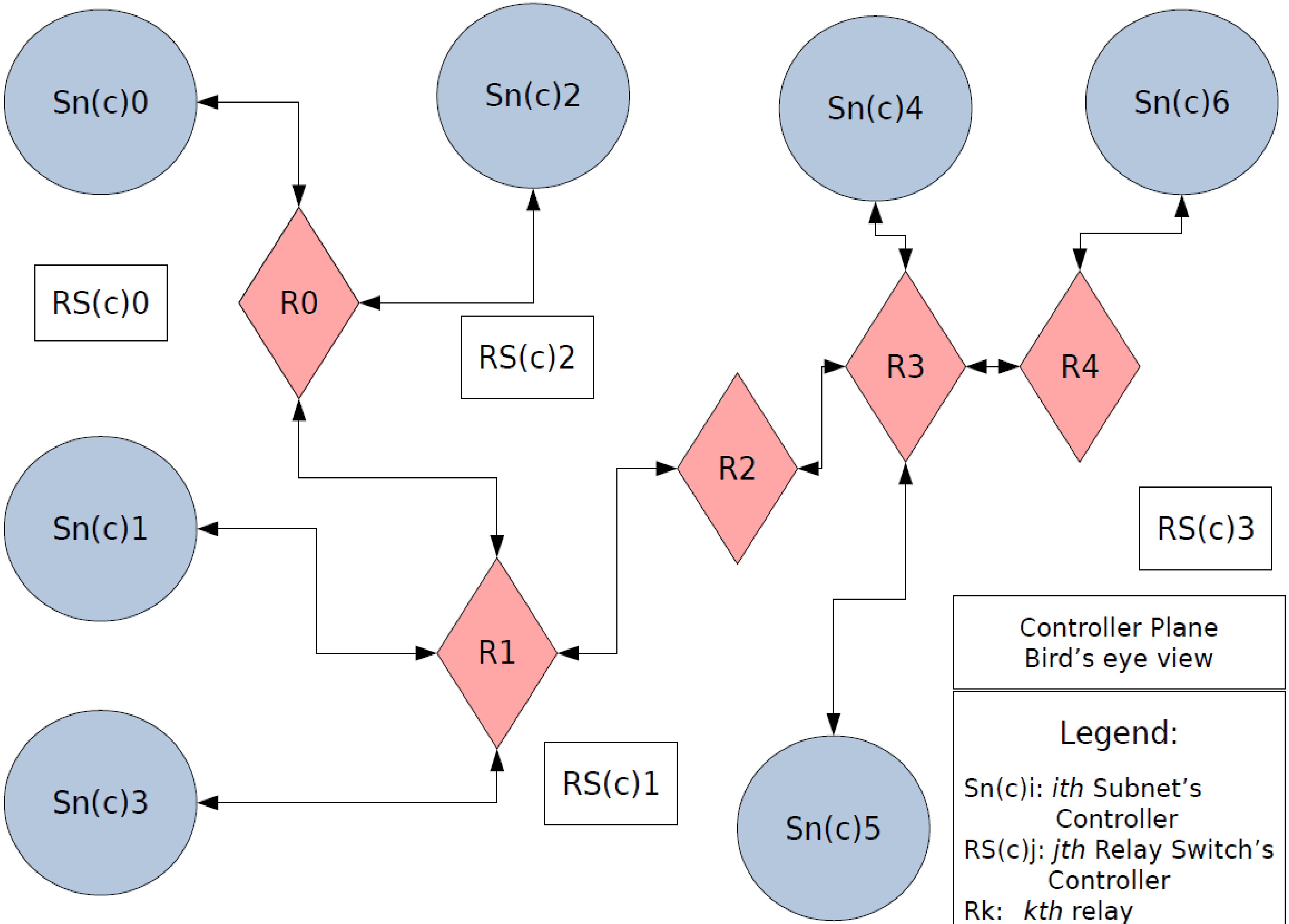
The New Data Plane

The data plane now consists of three separate components, namely:

– The Root Switch:
  - A unique and mandatory entity for every controller subnet.
  - Every host/switch within a subnet have a connection to it.

– The Relay Switch:
  - The communicator between the subnets.
  - *n* Relay Switches in whole network are connected via (*n-1)* connections
  - Any number of subnets can be managed by a Relay Switch.
  - They form a straight chain within themselves.

– A generic host:
  - Represents a host which is a generic node.



Controller Plane
Bird's eye view

Legend:

Sn(c)i: *ith* Subnet's Controller
RS(c)j: *jth* Relay Switch's Controller
Rk: *kth* relay

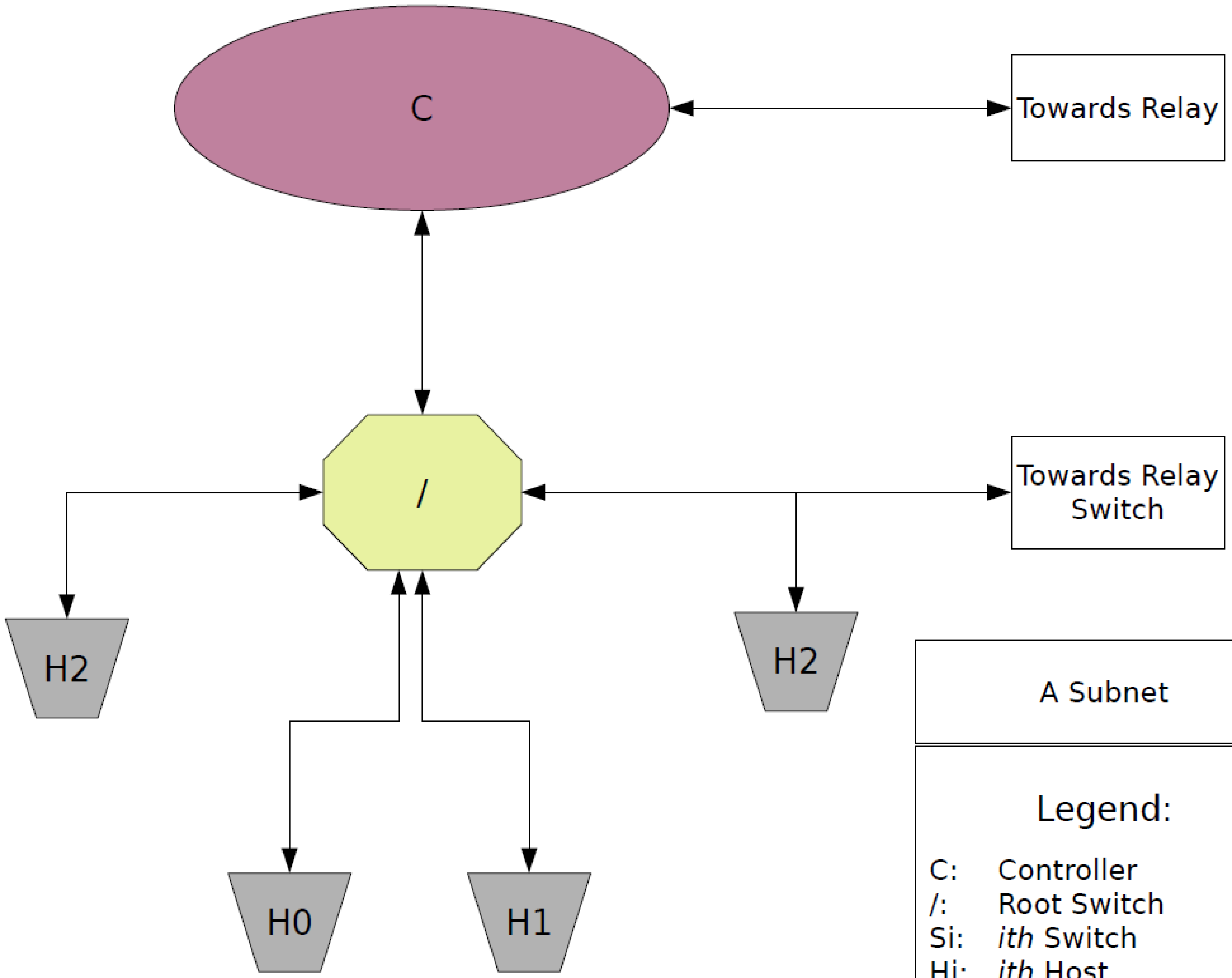
The New Controller Plane

The redesigned controller plane has three separate entities, namely:

– The Root Switch Controller:
  - It serves as the OpenFlow controller for every controller subnet.
  - Are interconnected via Relay for real time controller communication.

– The Relay:
  - This is a standalone multi-threaded TCP server.
  - Helps in real time connection between the Root Switch Controllers.
  - Forms duplex connections to every Root Switch Controller.

– The Relay Switch Controller:
  - A generic L2 learning switch controller template.
  - Enforces OpenFlow protocol on the Relay Switches.



A Subnet

Legend:

C:	Controller
/:	Root Switch
Si:	*ith* Switch
Hj:	*jth* Host

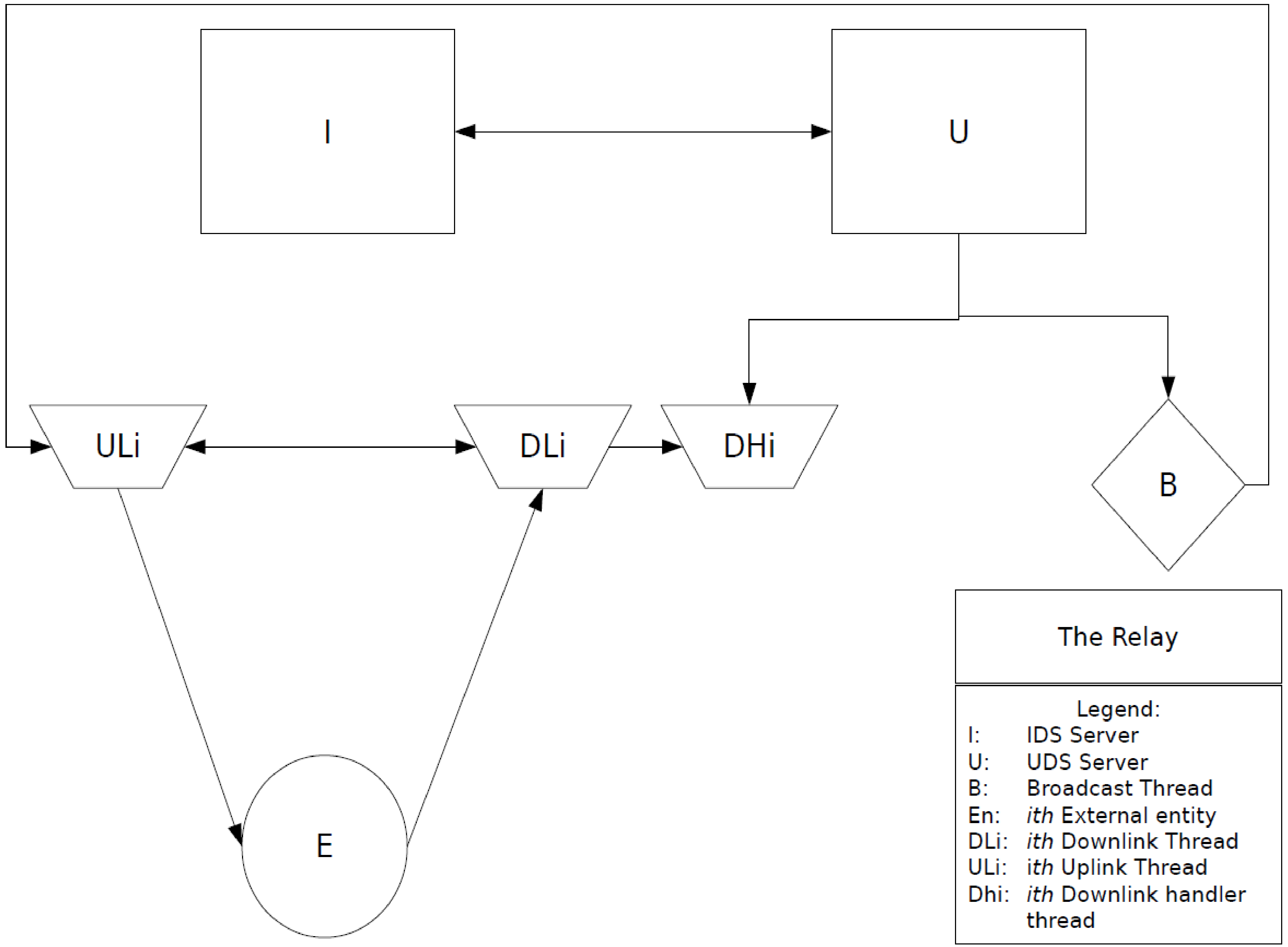
So what is new?

Old Topology	New Topology
Relay served as a **host lookup**.	**Relay** serves as **communication entity**.
Controller **Plane congestion was high**.	Controller plane **congestion is low.**
Absence of Relay Switch might result in controller flow loops (infinite).	**Presence of Relay Switches** mitigates controller flow loops.
Subnet **planning was difficult and asymmetrical.**	Due to mandated **Root Switch**, **subnet planning is symmetric and symmetrical.**

Lets Talk Numbers!

n(Subnets)/ n(Hosts/ Subnet)	Throughput (Bytes/sec)	Bandwidth (Bytes/Sec)	Delay (Hz)	Packet Loss (%)	Packet Delivery Ratio	Flow request Rate (Hz)
1/300	91.744	868.198	22.114	0.552	1	(22.579)
2/50	95.232	898.652	7.264	0.525	1	(25.115) (21.685)
2/150	95.168	846.332	22.837	0.526	1	(23.605) (20.311)
3/50	94.484	893.565	10.958	0.559	1	(22.466) (24.053)
4/75	95.744	845.227	23.018	0.586	1	(19.425) (23.220)

Review 3



The Relay

Legend:

I:	IDS Server
U:	UDS Server
B:	Broadcast Thread
En:	*ith* External entity
DLi:	*ith* Downlink Thread
ULi:	i*th* Uplink Thread
Dhi:	*ith* Downlink handler thread

Addition of sub-relay in the concept

- Each relay can be optionally modded into a sub-relay by supplying a file with the addresses of all the super relays it need to connect to.

- The sub relay connects to the super relay and this first connection becomes the downlink connection for the super relay while being uplink for the sub relay

- The sub relay creates a TCP server and listens on port 12346 which is generally on which all the root switch controllers listen on for getting a connection back from any relay.

- The super relay, as in the normal code, connects back to the sub-relay, assuming it to be just another controller, hence no new exception handling code has to be written.

- Now the super relay forwards information to this sub-relay too, just like it would for any other controller.

- The sub-relay and super relay concept is oblivious to both OpenFlow version and controller. Thus backward compatibility and heterogeneity are introduced.

- Thus the scalability is increased with addition of each sub-relay-super relay connection surpassing multiple geographic domains.

The DDoS Attack Mechanism

- The topology script randomly selects a host for posing like a bad actor, on which it runs a HTTP server on port 8000 and also a TCP server on 6666 port.
- A random number of hosts from the topology are then selected which fetch the vec.py (the attack vector file) from the bad acting server.
- They then also form a connection to the bad acting server, which the bad actor is listening for on it port 6666.
- When the topology boots up, the attack can be triggered via echoing 'trigger' in a named pipe on the bad actor system which in turn broadcasts the 'trigger' command to all its connected clients (the zombie hosts).
- All the hosts then start firing up about a 1,000,000 raw ethernet frames with spoofed and randomly generated source and destination MAC addresses.

- Whenever a spoofed raw ethernet frame hits a openVswitch, the query to route it is sent to the corresponding root switch controller, due to the non availability of the open flow entry for that particular route with the switch.

- The root switch controller acts upon the receipt of such a packet query by checking the destination address and if it is found to be invalid, the controller adds a flow in the root switch controller to drop all packets that come in from this particular port, thus mitigating the attack all together.

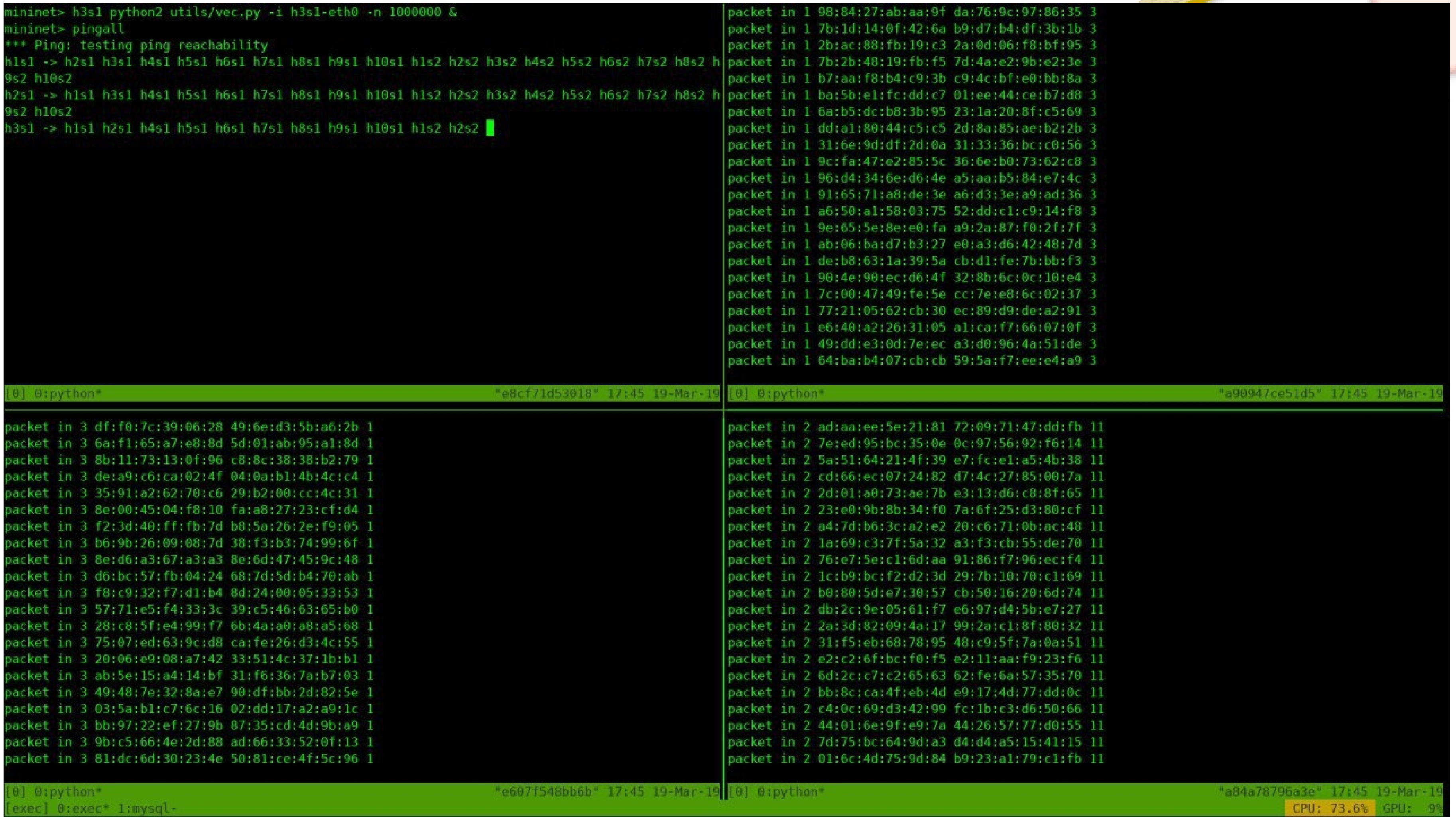
Test values to benchmark topology

n(Subnets)/ n(Hosts/ Subnet)	Throughput (Bytes/sec)	Bandwidth (Bytes/Sec)	Delay (Hz)	Packet Loss (%)	Packet Delivery Ratio	Flow request Rate (Hz)
1/75	40.755	18370.830	0.261	0	1	457.667
1/100	34.991	17356.51	0.368	0	1	421.218
1/125	33.385	16059.575	0.498	0	1	421.218
2/75	(39.800) (131.147)	8474.048	1.147	0	1	(211.058) (178.142)
2/100	(33.654) (33.092)	788.602	1.604	0	1	(206.138) (175.460)
2/125	(36.090) (144.687)	7664.663	2.104	0	1	(204.827) (175.027)
3/75	(22.492) (65.106)	7741.935	1.884	0	1	(184.219) (197.909)
3/100	(13.245) (114.217)	7305.647	2.654	0	1	(176.873) (189.724)
3/125	(19.464) (56.255)	6920.175	3.495	0	1	(168.550) (180.656)
4/75	(22.525) (35.294)	6363.316	3.059	0	1	(143.111) (174.360)
4/100	(16.619) (41.622)	6011.837	4.300	0	1	(135.744) (165.223)
4.125	(15.133) (43.412)	5862.210	5.502	0	1	(125.774) (161.112)

Test Values to Benchmark Attack Detection and Mitigation

(Subnet/Host)/ (pkt_sent/sec in each Condition)	No Attack	Attack with no Mitigation	Attack with proposed Mitigation stratergy
2 Subnets/ 75 Hosts	160.113	2203.116	65.666
2 Subnets/ 100 Hosts	178.533	1868.416	84.416
2 Subnets/ 125 Hosts	168.716	2137.166	78.51
3 Subnets/ 75 Hosts	292.916	2077.650	89.766
3 Subnets/ 100 Hosts	271.460	1746.650	80.083
3 Subnets/ 125 Hosts	250.016	2596.266	74.233
4 Subnets/ 75 Hosts	247.883	2257.012	86.866
4 Subnets/ 100 Hosts	219.916	2122.336	89.663
4 Subnets/ 125 Hosts	250.278	2399.616	83.116

Screenshots



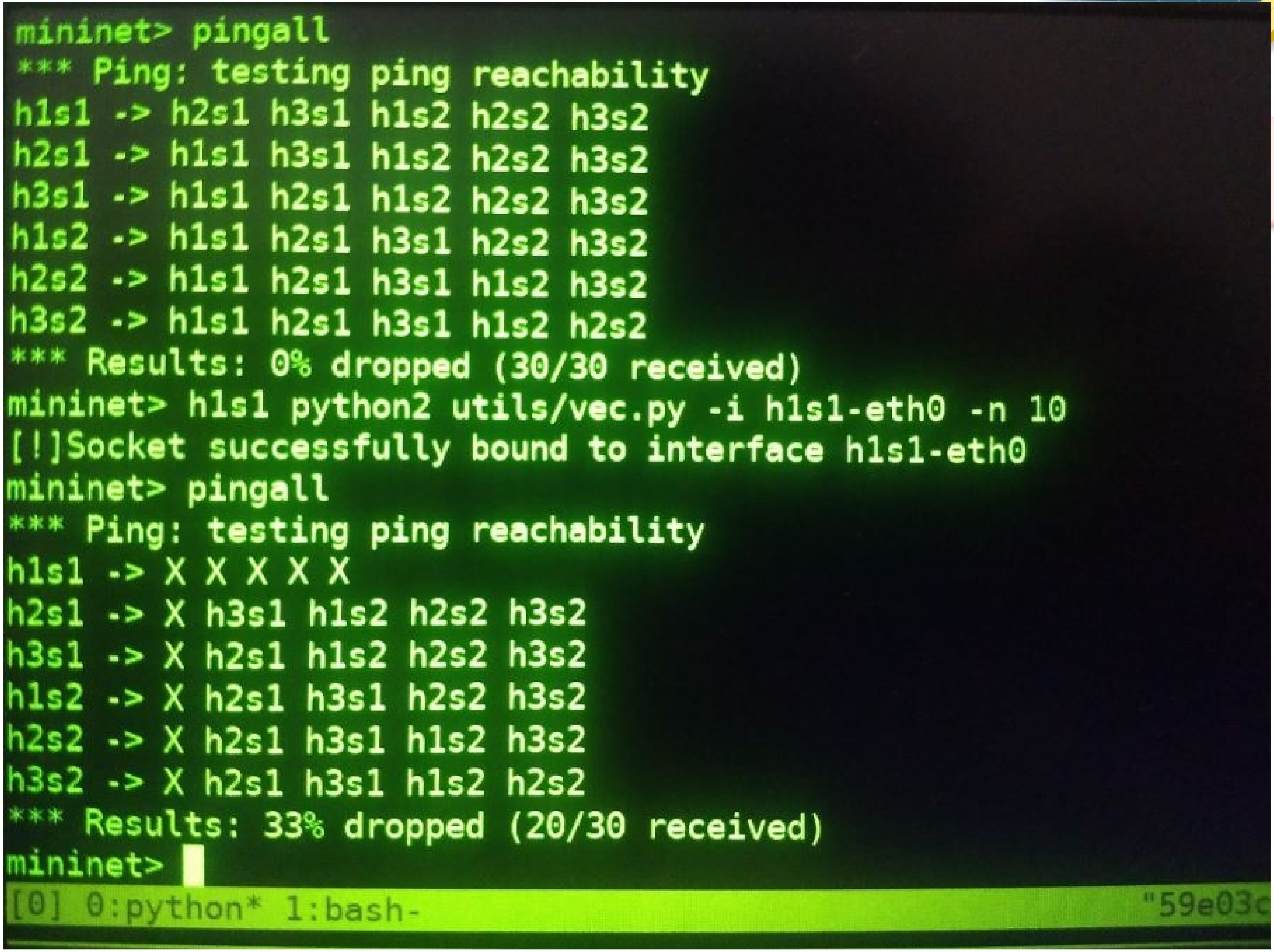
```
mininet> h3s1 python2 utils/vec.py -i h3s1-eth0 -n 1000000 &
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> h2s1 h3s1 h4s1 h5s1 h6s1 h7s1 h8s1 h9s1 h10s1 h1s2 h2s2 h3s2 h4s2 h5s2 h6s2 h7s2 h8s2 h
9s2 h10s2
h2s1 -> h1s1 h3s1 h4s1 h5s1 h6s1 h7s1 h8s1 h9s1 h10s1 h1s2 h2s2 h3s2 h4s2 h5s2 h6s2 h7s2 h8s2 h
9s2 h10s2
h3s1 -> h1s1 h2s1 h4s1 h5s1 h6s1 h7s1 h8s1 h9s1 h10s1 h1s2 h2s2 █







[0] 0:python*                                               "e8cf71d53018" 17:45 19-Mar-19
```

```
packet in 1 98:84:27:ab:aa:9f da:76:9c:97:86:35 3
packet in 1 7b:1d:14:0f:42:6a b9:d7:b4:df:3b:1b 3
packet in 1 2b:ac:88:fb:19:c3 2a:0d:06:f8:bf:95 3
packet in 1 7b:2b:48:19:fb:f5 7d:4a:e2:9b:e2:3e 3
packet in 1 b7:aa:f8:b4:c9:3b c9:4c:bf:e0:bb:8a 3
packet in 1 ba:5b:e1:fc:dd:c7 01:ee:44:ce:b7:d8 3
packet in 1 6a:b5:dc:b8:3b:95 23:1a:20:8f:c5:69 3
packet in 1 dd:a1:80:44:c5:c5 2d:8a:85:ae:b2:2b 3
packet in 1 31:6e:9d:df:2d:0a 31:33:36:bc:c0:56 3
packet in 1 9c:fa:47:e2:85:5c 36:6e:b0:73:62:c8 3
packet in 1 96:d4:34:6e:d6:4e a5:aa:b5:84:e7:4c 3
packet in 1 91:65:71:a8:de:3e a6:d3:3e:a9:ad:36 3
packet in 1 a6:50:a1:58:03:75 52:dd:c1:c9:14:f8 3
packet in 1 9e:65:5e:8e:e0:fa a9:2a:87:f0:2f:7f 3
packet in 1 ab:06:ba:d7:b3:27 e0:a3:d6:42:48:7d 3
packet in 1 de:b8:63:1a:39:5a cb:d1:fe:7b:bb:f3 3
packet in 1 90:4e:90:ec:d6:4f 32:8b:6c:0c:10:e4 3
packet in 1 7c:00:47:49:fe:5e cc:7e:e8:6c:02:37 3
packet in 1 77:21:05:62:cb:30 ec:89:d9:de:a2:91 3
packet in 1 e6:40:a2:26:31:05 a1:ca:f7:66:07:0f 3
packet in 1 49:dd:e3:0d:7e:ec a3:d0:96:4a:51:de 3
packet in 1 64:ba:b4:07:cb:cb 59:5a:f7:ee:e4:a9 3

[0] 0:python*                                               "a90947ce51d5" 17:45 19-Mar-19
```

```
packet in 3 df:f0:7c:39:06:28 49:6e:d3:5b:a6:2b 1
packet in 3 6a:f1:65:a7:e8:8d 5d:01:ab:95:a1:8d 1
packet in 3 8b:11:73:13:0f:96 c8:8c:38:38:b2:79 1
packet in 3 de:a9:c6:ca:02:4f 04:0a:b1:4b:4c:c4 1
packet in 3 35:91:a2:62:70:c6 29:b2:00:cc:4c:31 1
packet in 3 8e:00:45:04:f8:10 fa:a8:27:23:cf:d4 1
packet in 3 f2:3d:40:ff:fb:7d b8:5a:26:2e:f9:05 1
packet in 3 b6:9b:26:09:08:7d 38:f3:b3:74:99:6f 1
packet in 3 8e:d6:a3:67:a3:a3 8e:6d:47:45:9c:48 1
packet in 3 d6:bc:57:fb:04:24 68:7d:5d:b4:70:ab 1
packet in 3 f8:c9:32:f7:d1:b4 8d:24:00:05:33:53 1
packet in 3 57:71:e5:f4:33:3c 39:c5:46:63:65:b0 1
packet in 3 28:c8:5f:e4:99:f7 6b:4a:a0:a8:a5:68 1
packet in 3 75:07:ed:63:9c:d8 ca:fe:26:d3:4c:55 1
packet in 3 20:06:e9:08:a7:42 33:51:4c:37:1b:b1 1
packet in 3 ab:5e:15:a4:14:bf 31:f6:36:7a:b7:03 1
packet in 3 49:48:7e:32:8a:e7 90:df:bb:2d:82:5e 1
packet in 3 03:5a:b1:c7:6c:16 02:dd:17:a2:a9:1c 1
packet in 3 bb:97:22:ef:27:9b 87:35:cd:4d:9b:a9 1
packet in 3 9b:c5:66:4e:2d:88 ad:66:33:52:0f:13 1
packet in 3 81:dc:6d:30:23:4e 50:81:ce:4f:5c:96 1

[0] 0:python*                                               "e607f548bb6b" 17:45 19-Mar-19
[exec] 0:exec* 1:mysql-
```

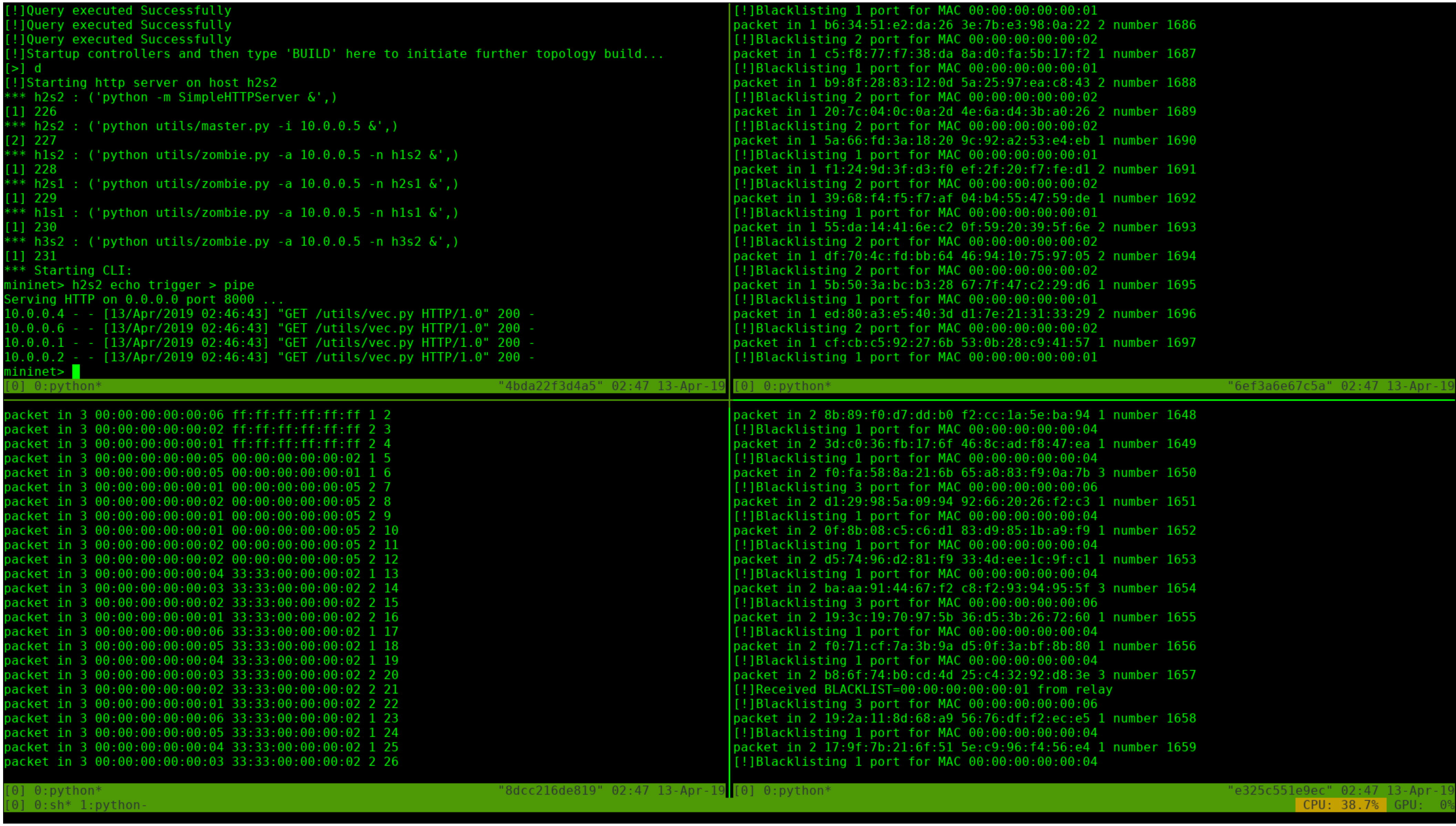
```
packet in 2 ad:aa:ee:5e:21:81 72:09:71:47:dd:fb 11
packet in 2 7e:ed:95:bc:35:0e 0c:97:56:92:f6:14 11
packet in 2 5a:51:64:21:4f:39 e7:fc:e1:a5:4b:38 11
packet in 2 cd:66:ec:07:24:82 d7:4c:27:85:00:7a 11
packet in 2 2d:01:a0:73:ae:7b e3:13:d6:c8:8f:65 11
packet in 2 23:e0:9b:8b:34:f0 7a:6f:25:d3:80:cf 11
packet in 2 a4:7d:b6:3c:a2:e2 20:c6:71:0b:ac:48 11
packet in 2 1a:69:c3:7f:5a:32 a3:f3:cb:55:de:70 11
packet in 2 76:e7:5e:c1:6d:aa 91:86:f7:96:ec:f4 11
packet in 2 1c:b9:bc:f2:d2:3d 29:7b:10:70:c1:69 11
packet in 2 b0:80:5d:e7:30:57 cb:50:16:20:6d:74 11
packet in 2 db:2c:9e:05:61:f7 e6:97:d4:5b:e7:27 11
packet in 2 2a:3d:82:09:4a:17 99:2a:c1:8f:80:32 11
packet in 2 31:f5:eb:68:78:95 48:c9:5f:7a:0a:51 11
packet in 2 e2:c2:6f:bc:f0:f5 e2:11:aa:f9:23:f6 11
packet in 2 6d:2c:c7:c2:65:63 62:fe:6a:57:35:70 11
packet in 2 bb:8c:ca:4f:eb:4d e9:17:4d:77:dd:0c 11
packet in 2 c4:0c:69:d3:42:99 fc:1b:c3:d6:50:66 11
packet in 2 44:01:6e:9f:e9:7a 44:26:57:77:d0:55 11
packet in 2 7d:75:bc:64:9d:a3 d4:d4:a5:15:41:15 11
packet in 2 01:6c:4d:75:9d:84 b9:23:a1:79:c1:fb 11

[0] 0:python*                                               "a84a78796a3e" 17:45 19-Mar-19
                                                                CPU: 73.6%  GPU:  9%
```

DoS attack



```
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> h2s1 h3s1 h1s2 h2s2 h3s2
h2s1 -> h1s1 h3s1 h1s2 h2s2 h3s2
h3s1 -> h1s1 h2s1 h1s2 h2s2 h3s2
h1s2 -> h1s1 h2s1 h3s1 h2s2 h3s2
h2s2 -> h1s1 h2s1 h3s1 h1s2 h3s2
h3s2 -> h1s1 h2s1 h3s1 h1s2 h2s2
*** Results: 0% dropped (30/30 received)
mininet> h1s1 python2 utils/vec.py -i h1s1-eth0 -n 10
[!]Socket successfully bound to interface h1s1-eth0
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> X X X X X
h2s1 -> X h3s1 h1s2 h2s2 h3s2
h3s1 -> X h2s1 h1s2 h2s2 h3s2
h1s2 -> X h2s1 h3s1 h2s2 h3s2
h2s2 -> X h2s1 h3s1 h1s2 h3s2
h3s2 -> X h2s1 h3s1 h1s2 h2s2
*** Results: 33% dropped (20/30 received)
mininet> 
[0] 0:python* 1:bash-                                          "59e03c
```

DoS Attack Mitigation



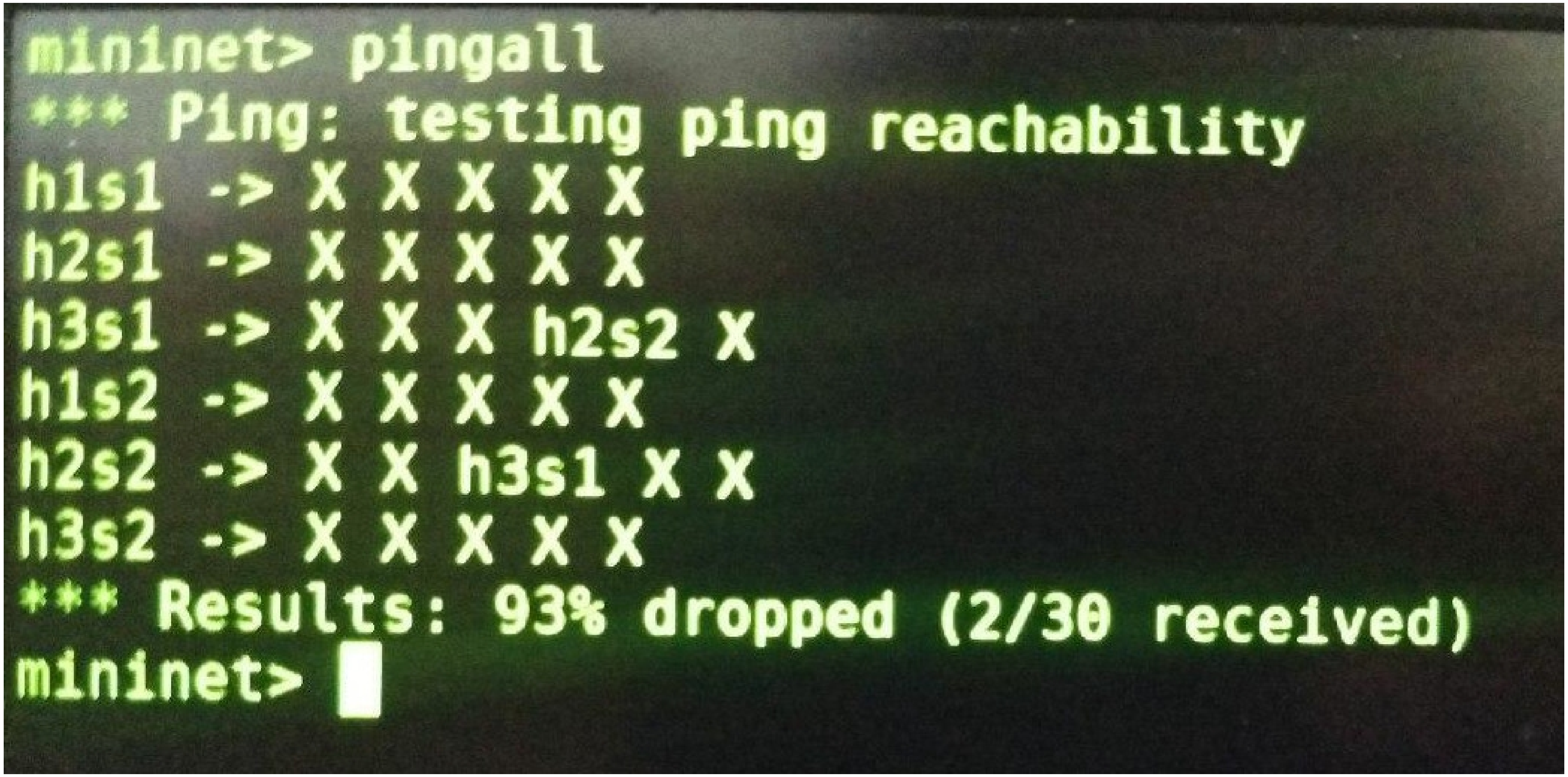
```
[!]Query executed Successfully
[!]Query executed Successfully
[!]Query executed Successfully
[!]Startup controllers and then type 'BUILD' here to initiate further topology build...
[>] d
[!]Starting http server on host h2s2
*** h2s2 : ('python -m SimpleHTTPServer &',)
[1] 226
*** h2s2 : ('python utils/master.py -i 10.0.0.5 &',)
[2] 227
*** h1s2 : ('python utils/zombie.py -a 10.0.0.5 -n h1s2 &',)
[1] 228
*** h2s1 : ('python utils/zombie.py -a 10.0.0.5 -n h2s1 &',)
[1] 229
*** h1s1 : ('python utils/zombie.py -a 10.0.0.5 -n h1s1 &',)
[1] 230
*** h3s2 : ('python utils/zombie.py -a 10.0.0.5 -n h3s2 &',)
[1] 231
*** Starting CLI:
mininet> h2s2 echo trigger > pipe
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.0.4 - - [13/Apr/2019 02:46:43] "GET /utils/vec.py HTTP/1.0" 200 -
10.0.0.6 - - [13/Apr/2019 02:46:43] "GET /utils/vec.py HTTP/1.0" 200 -
10.0.0.1 - - [13/Apr/2019 02:46:43] "GET /utils/vec.py HTTP/1.0" 200 -
10.0.0.2 - - [13/Apr/2019 02:46:43] "GET /utils/vec.py HTTP/1.0" 200 -
mininet>
[0] 0:python*                                      "4bda22f3d4a5" 02:47 13-Apr-19
```

```
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01
packet in 1 b6:34:51:e2:da:26 3e:7b:e3:98:0a:22 2 number 1686
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 c5:f8:77:f7:38:da 8a:d0:fa:5b:17:f2 1 number 1687
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01
packet in 1 b9:8f:28:83:12:0d 5a:25:97:ea:c8:43 2 number 1688
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 20:7c:04:0c:0a:2d 4e:6a:d4:3b:a0:26 2 number 1689
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 5a:66:fd:3a:18:20 9c:92:a2:53:e4:eb 1 number 1690
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01
packet in 1 f1:24:9d:3f:d3:f0 ef:2f:20:f7:fe:d1 2 number 1691
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 39:68:f4:f5:f7:af 04:b4:55:47:59:de 1 number 1692
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01
packet in 1 55:da:14:41:6e:c2 0f:59:20:39:5f:6e 2 number 1693
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 df:70:4c:fd:bb:64 46:94:10:75:97:05 2 number 1694
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 5b:50:3a:bc:b3:28 67:7f:47:c2:29:d6 1 number 1695
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01
packet in 1 ed:80:a3:e5:40:3d d1:7e:21:31:33:29 2 number 1696
[!]Blacklisting 2 port for MAC 00:00:00:00:00:02
packet in 1 cf:cb:c5:92:27:6b 53:0b:28:c9:41:57 1 number 1697
[!]Blacklisting 1 port for MAC 00:00:00:00:00:01

[0] 0:python*                                      "6ef3a6e67c5a" 02:47 13-Apr-19
```

```
packet in 3 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 1 2
packet in 3 00:00:00:00:00:02 ff:ff:ff:ff:ff:ff 2 3
packet in 3 00:00:00:00:00:01 ff:ff:ff:ff:ff:ff 2 4
packet in 3 00:00:00:00:00:05 00:00:00:00:00:02 1 5
packet in 3 00:00:00:00:00:05 00:00:00:00:00:01 1 6
packet in 3 00:00:00:00:00:01 00:00:00:00:00:05 2 7
packet in 3 00:00:00:00:00:02 00:00:00:00:00:05 2 8
packet in 3 00:00:00:00:00:01 00:00:00:00:00:05 2 9
packet in 3 00:00:00:00:00:01 00:00:00:00:00:05 2 10
packet in 3 00:00:00:00:00:02 00:00:00:00:00:05 2 11
packet in 3 00:00:00:00:00:02 00:00:00:00:00:05 2 12
packet in 3 00:00:00:00:00:04 33:33:00:00:00:02 1 13
packet in 3 00:00:00:00:00:03 33:33:00:00:00:02 2 14
packet in 3 00:00:00:00:00:02 33:33:00:00:00:02 2 15
packet in 3 00:00:00:00:00:01 33:33:00:00:00:02 2 16
packet in 3 00:00:00:00:00:06 33:33:00:00:00:02 1 17
packet in 3 00:00:00:00:00:05 33:33:00:00:00:02 1 18
packet in 3 00:00:00:00:00:04 33:33:00:00:00:02 1 19
packet in 3 00:00:00:00:00:03 33:33:00:00:00:02 2 20
packet in 3 00:00:00:00:00:02 33:33:00:00:00:02 2 21
packet in 3 00:00:00:00:00:01 33:33:00:00:00:02 2 22
packet in 3 00:00:00:00:00:06 33:33:00:00:00:02 1 23
packet in 3 00:00:00:00:00:05 33:33:00:00:00:02 1 24
packet in 3 00:00:00:00:00:04 33:33:00:00:00:02 1 25
packet in 3 00:00:00:00:00:03 33:33:00:00:00:02 2 26

[0] 0:python*                                      "8dcc216de819" 02:47 13-Apr-19
[0] 0:sh* 1:python-
```

```
packet in 2 8b:89:f0:d7:dd:b0 f2:cc:1a:5e:ba:94 1 number 1648
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 3d:c0:36:fb:17:6f 46:8c:ad:f8:47:ea 1 number 1649
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 f0:fa:58:8a:21:6b 65:a8:83:f9:0a:7b 3 number 1650
[!]Blacklisting 3 port for MAC 00:00:00:00:00:06
packet in 2 d1:29:98:5a:09:94 92:66:20:26:f2:c3 1 number 1651
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 0f:8b:08:c5:c6:d1 83:d9:85:1b:a9:f9 1 number 1652
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 d5:74:96:d2:81:f9 33:4d:ee:1c:9f:c1 1 number 1653
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 ba:aa:91:44:67:f2 c8:f2:93:94:95:5f 3 number 1654
[!]Blacklisting 3 port for MAC 00:00:00:00:00:06
packet in 2 19:3c:19:70:97:5b 36:d5:3b:26:72:60 1 number 1655
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 f0:71:cf:7a:3b:9a d5:0f:3a:bf:8b:80 1 number 1656
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 b8:6f:74:b0:cd:4d 25:c4:32:92:d8:3e 3 number 1657
[!]Received BLACKLIST=00:00:00:00:00:01 from relay
[!]Blacklisting 3 port for MAC 00:00:00:00:00:06
packet in 2 19:2a:11:8d:68:a9 56:76:df:f2:ec:e5 1 number 1658
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04
packet in 2 17:9f:7b:21:6f:51 5e:c9:96:f4:56:e4 1 number 1659
[!]Blacklisting 1 port for MAC 00:00:00:00:00:04

[0] 0:python*                                      "e325c551e9ec" 02:47 13-Apr-19
                                                           CPU: 38.7%  GPU:   0%
```

DDoS Attack



```
mininet> pingall
*** Ping: testing ping reachability
h1s1 -> X X X X X
h2s1 -> X X X X X
h3s1 -> X X X h2s2 X
h1s2 -> X X X X X
h2s2 -> X X h3s1 X X
h3s2 -> X X X X X
*** Results: 93% dropped (2/30 received)
mininet> 
```

DDoS Attack Mitigation

Thank You!

Prepared and Presented by:
Naman Arora
RA1511003010235
Nikhil Gupta
RA1511003010245