Trusted Communication in Software Defined Distributed Systems

Under the guidance of:
Ms. Vaishnavi Moorthy
Assistant Professor
Department of Computer Science and Engineering

Presented By:
Naman Arora (RA1511003010235)
Nikhil Gupta (RA1511003010245)

Abstract

- Software Defined Networking (SDN) is presented as a paradigm shift in regard to the evolution of internet since its advent.
- 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.
- An effort has been to augment the idea of SDN to a more distributed framework. Using cleverly designed topologies like Spine leaf, interconnection of controllers using relay system designed from bottom up is demonstrated.
- The latter part of the project acknowledges the need to secure such translations and try to mitigate Denial of Service (DoS) attacks on the control plane.

Introduction

- An effort to distribute controller plane in Software Defined Networking
- Scalability and robustness primary focus
- Provide a topology
- Provide proof of scalability via benchmark testing
- Provide proof of robustness via launching and mitigating a DDoS stack
- Major Highlights:
  - Relay
  - Blacklisting
  - DoS
  - DDoS
  - WhiteListing

Limitations of Present System

- Less scalability factor.
- More energy consumption.
- Greater expense for physical cables.
- More vulnerable area to secure.
- $(n-1)^n$ connections within controllers due to Mesh topology.

Survey Paper Title	Author	Description
Proprietary Defense Systems in Software defined Networks	Radware	(Radware)[6],has expressed that the scene is evolving. It isn't just the IT framework which is making strides in intricacy, amount and expectation, but attackers are using latest accessible technology and the after effects of this are as of now being seen on the cyber battlefield. DefenseFlow permits the service providers to effectively automate the (Incident Response (IR)) activities in the most perplexing and profoundly distributed environments.
Threat categorization and identification in SDN	Krishnan and Najeem	Krishnan and Najeem (2017)[3], in their study found out that using (SDN) in today's networks supplies with the required spryness and transparency for the installation of network solutions. Be that is it may, from the security point of view in terms of threat and risk assessment, especially for layer 4 and layer 7 attacks such as (Distributed De- nial of Service (DDoS)), there are yet many difficulties to be pursued in (SDN) environ- ments. In their study, they have exhibited the categorization of threats, risks and attack vectors that can disrupt the (SDN) network and have presented various techniques to mitigate these issues, to deploy (SDN) securely in production environments.
Research in SDN and usage in cloud computing	Kannan Govindarajan	Kannan Govindarajan (2013)[2], expressed that a key developing pattern in Cloud com- puting is that the core frameworks to be shifting towards Software-Defined. Storage and networks would no longer be constrained by the availability of physical hardware rather will be able to customize according to the needs in a virtual environment. (SDN) as- sumes a significant role in distributing the resources in the network based on the demand and requirement. These experts reviewed the cutting edge Software-Defined Network- ing (SDN) in four regions: Network Quality of Service (Quality of Service (QoS)), Load Balancing, Scalability and Security.
DOS attacks mitigation strategies	Lobna Dridi	Lobna Dridi (2016)[4], expressed that regardless of the considerable number of focal points offered by Software Defined Networks, Denial of Service (DOS) attacks are viewed as a noteworthy risk to such systems as they can flood the network with huge amount of invalid packets that may cause overflow in the (Content Addressable Mem- ory (CAM)) tables ultimately resulting in the deterioration of the quality of network service. They proposed SDN-Guard, a novel plan to effectively ensure (SDN) sys- tems against (DOS) attacks by dynamic (1) rerouting of potentially malicious traffic, (2) adjusting flow timeouts and (3) customizing the flow rules.

Survey Paper Title	Author	Description
SDN, Cloud computing and vulnerabilities	Qiao Yan	Qiao Yan (2015)[5], have expressed that the abilities of (SDN), including traffic ex- amination on a software level, centralized control, worldwide perspective on the net- work, dynamic updation of sending rules, make it simpler to distinguish and respond to (DDoS) attacks but the vulnerability of SDN is still an issue to be addressed, and po- tential (DDoS) vulnerabilities exist crosswise over various (SDN) platforms. Qiao Yan (2015)[5] have talked about the new patterns and qualities of DDoS attacks in dis-
Implementation of SDN networks in a global perspective	Sakir Sezer	Sakir Sezer (2013)[7], have expressed that Software-Defined Networking has risen as an effective network technology fit for support of the dynamic idea of future network func- tions and smart applications while bringing down expenses through improved equip- ment, programming, and management. They have discussed about generating a fruitful and functional network with Software-Defined Networking. S Existing systems and current industry standards could help in resolution of a portion of these issues and various working groups are additionally examining potential arrangements. The goal of the model is to upgrade flow handling in SDN.
Behavioural Detection of malicious traffic in the SDN	Syed Akbar Mehdi	(Syed Akbar Mehdi)[8], have contended that coming of Software Defined Networking gives a remarkable chance to identify and isolate security issues. They have outlined how four conspicuous traffic inconsistency identification algorithms can be used in Soft- ware Defined Networks with NOX as a controller in the controller plane and Open flow switches in the data plane. One of the key advantages of this method- ology is that the compartmentalized and controlled programmability of SDN enables these algorithms to exist with regards to a more extensive structure.
Data forwarding policies in SDN	Takayuki Sasaki	Takayuki Sasaki (2016)[9] have examined that the service provider needs apparatuses to proactively guarantee that the policies will be abode or to reactively assess the behaviour of the network. Any updates in the data plan are in a distributed manner hence lead to inconsistent behaviour amid reconfiguration. Also, the substantial flow space makes the data plane powerless to state exhaustion attacks. These experts have presented SDNsec, a security extension which provides forwarding accountability for the SDN data plane. Forwarding rules are encoded in the packet, which makes sure that the network behaviour is consistent amid reconfiguration and constraints state exhaus- tion attacks due to table lookups.

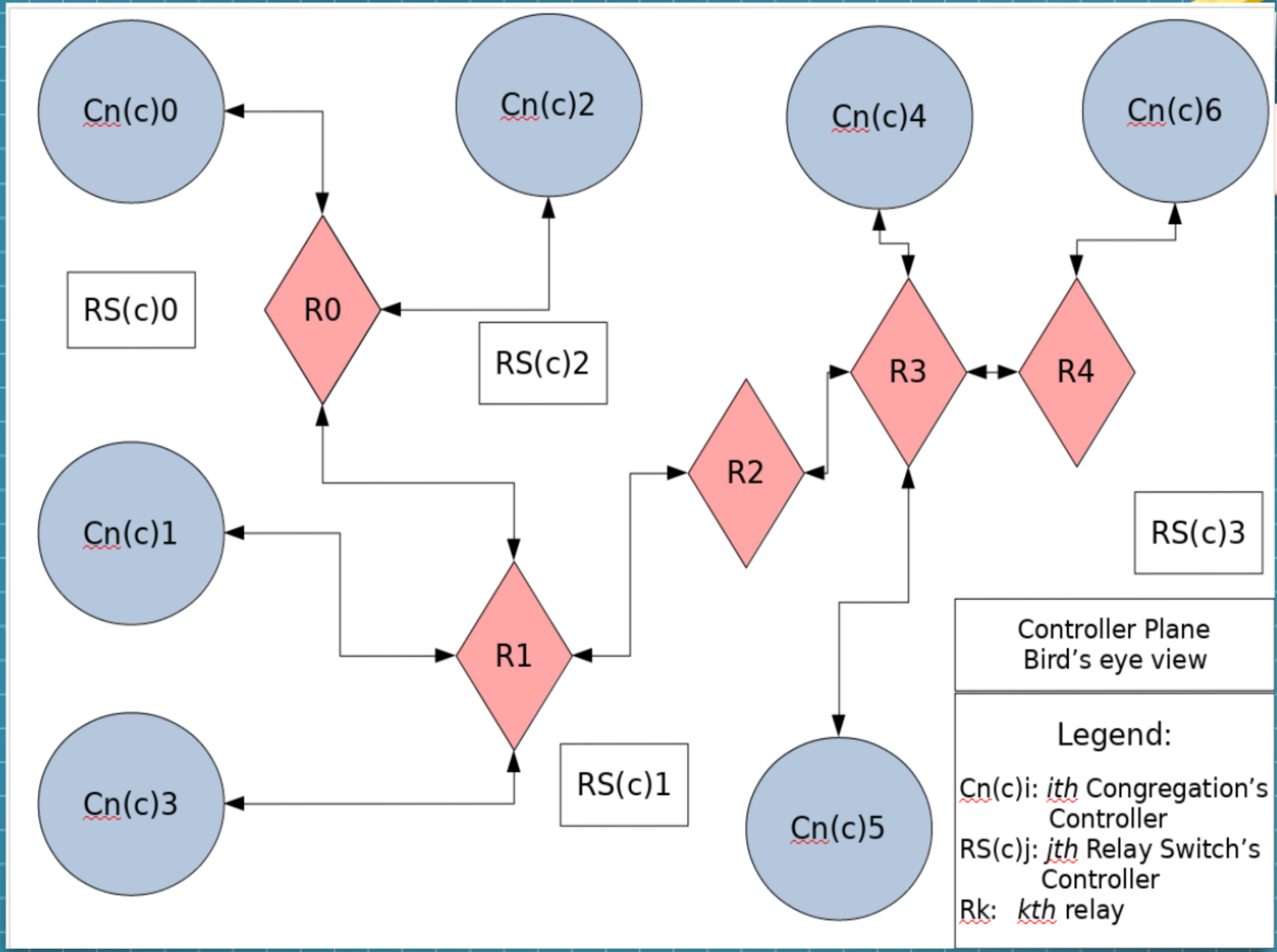
Inference from the survey

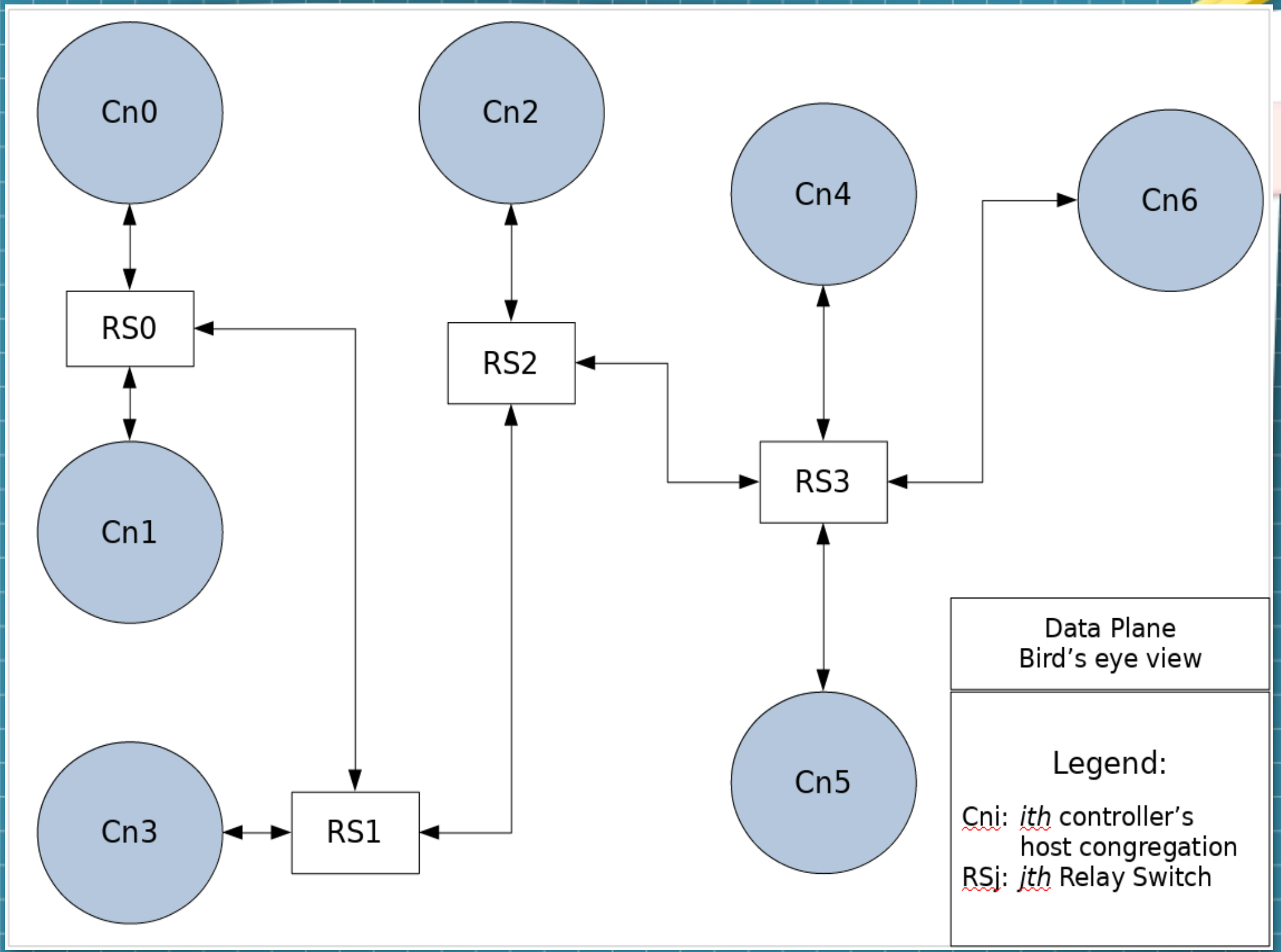
- It was found that the current topologies for Software Defined networks are not scalable to a large extent and inter-controller communication is still a big challenge when the number of controllers involved is huge as per Abubakar Siddique Muqaddas (2017)[1].

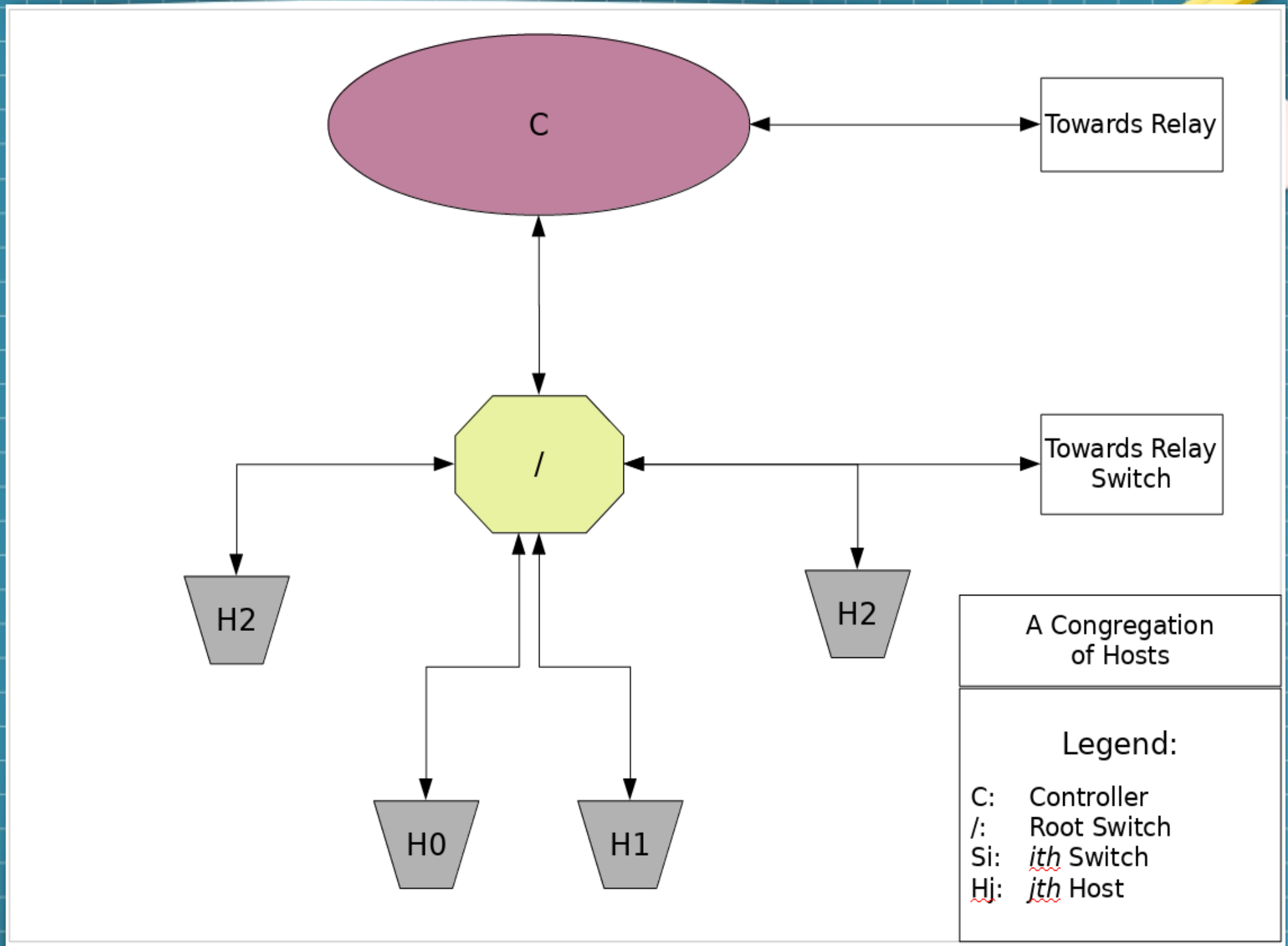
- When dealing with the cyberattacks such as Denial of service or Distributed Denial of service, the system requires a proper mechanism to stop the attack from affecting the whole network using some anomaly detection systems or detection algorithms and pre-defined parameters

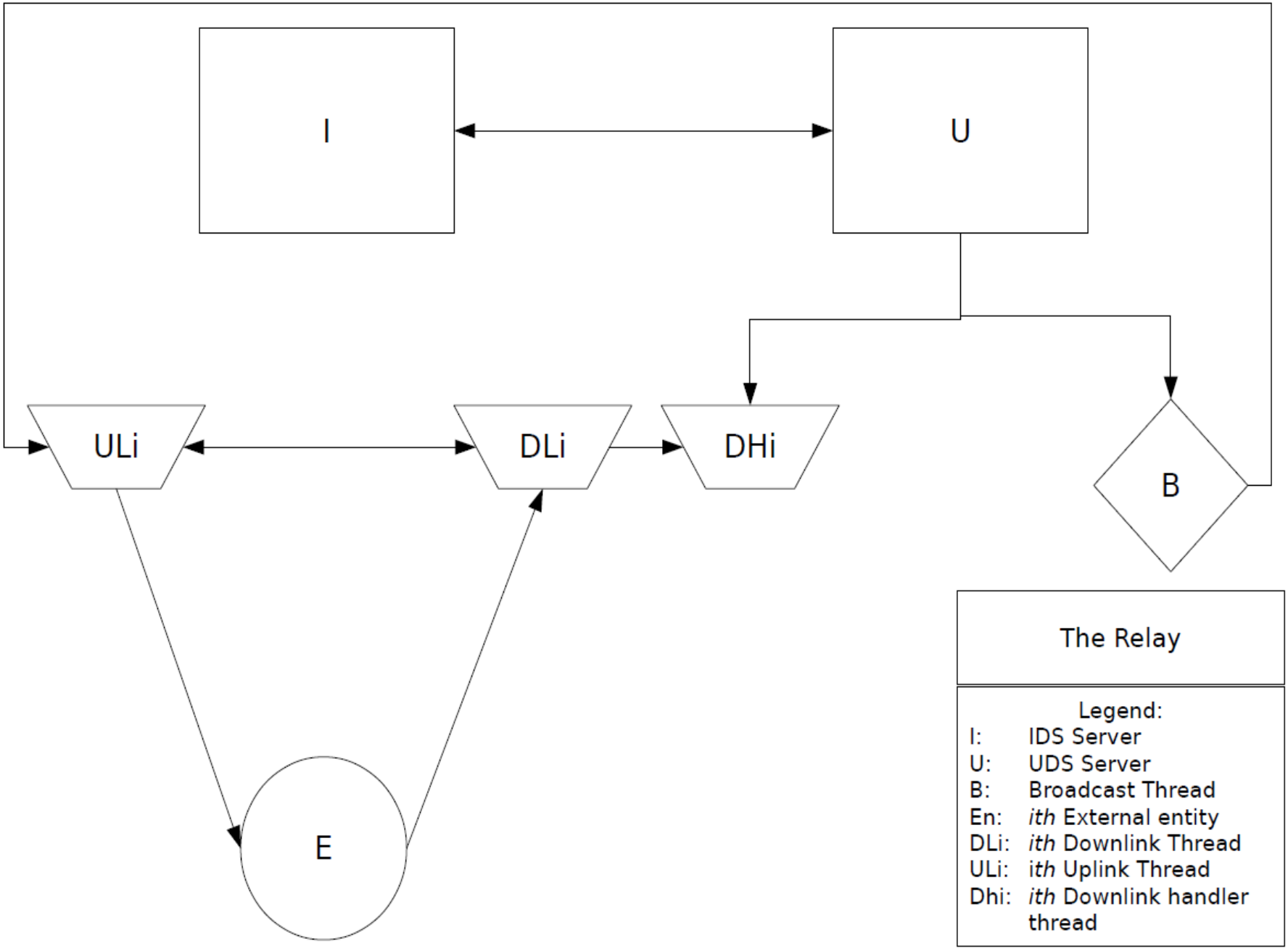
Proposed System

- Relay acts as a real time communication link between distributed controllers.
- Sub relaying to surpass geographical barriers.
- Discrete host congregations under one controller, numerous congregations possible.
- Relay switch to interconnect separate congregations in data plane.
- Unique root switches to handle separate congregations.
- Duplex connectivity.


Cn(c)0
Cn(c)2
Cn(c)4
Cn(c)6
RS(c)0
R0
RS(c)2
R3
R4
R2
Cn(c)1
RS(c)3
R1
Controller Plane
Bird's eye view
Cn(c)3
RS(c)1
Cn(c)5
Legend:
Cn(c)i: ith Congregation's Controller
RS(c)j: jth Relay Switch's Controller
Rk: kth relay


Cn0
Cn2
Cn4
Cn6
RS0
RS2
RS3
Cn1
Cn3
RS1
Cn5
Data Plane
Bird's eye view
Legend:
Cni: ith controller's host congregation
RSj: jth Relay Switch


C
Towards Relay
/
Towards Relay
Switch
H2
H2
H0
H1
A Congregation
of Hosts
Legend:
C: Controller
/: Root Switch
Si: ith Switch
Hj: jth Host



The Relay

Legend:

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

Addition of sub-relay in the concept

- Each relay can be optionally modded into a sub-relay
- A duplex connection between super and sub relays
- The sub relay creates a TCP server and listens on port 12346
- The super relay, connects back to the sub-relay
- Now the super relay forwards information to this sub-relay too
- Both the super relay and subrelay are backwards compatible
- Heterogenity within controllers is supported

The DDoS Attack Mechanism

- A host is randomly selected for being a bad server
- 10% hosts of the remaining pool of hosts are selected as zombies
- These hosts then also form a connection to the bad acting server.
- When the topology boots up, the attack can be triggered via echoing 'trigger' in a named pipe on the simulation system
- All the hosts then start firing up spoofed raw ethernet frames with generated source and destination MAC addresses.

Black listing of a node

- The controller checks the dst and src address of il-legitimate packet
- It adds a flow to drop all packets from that in port
- Adds a timeout to such a flow

White listing of a node

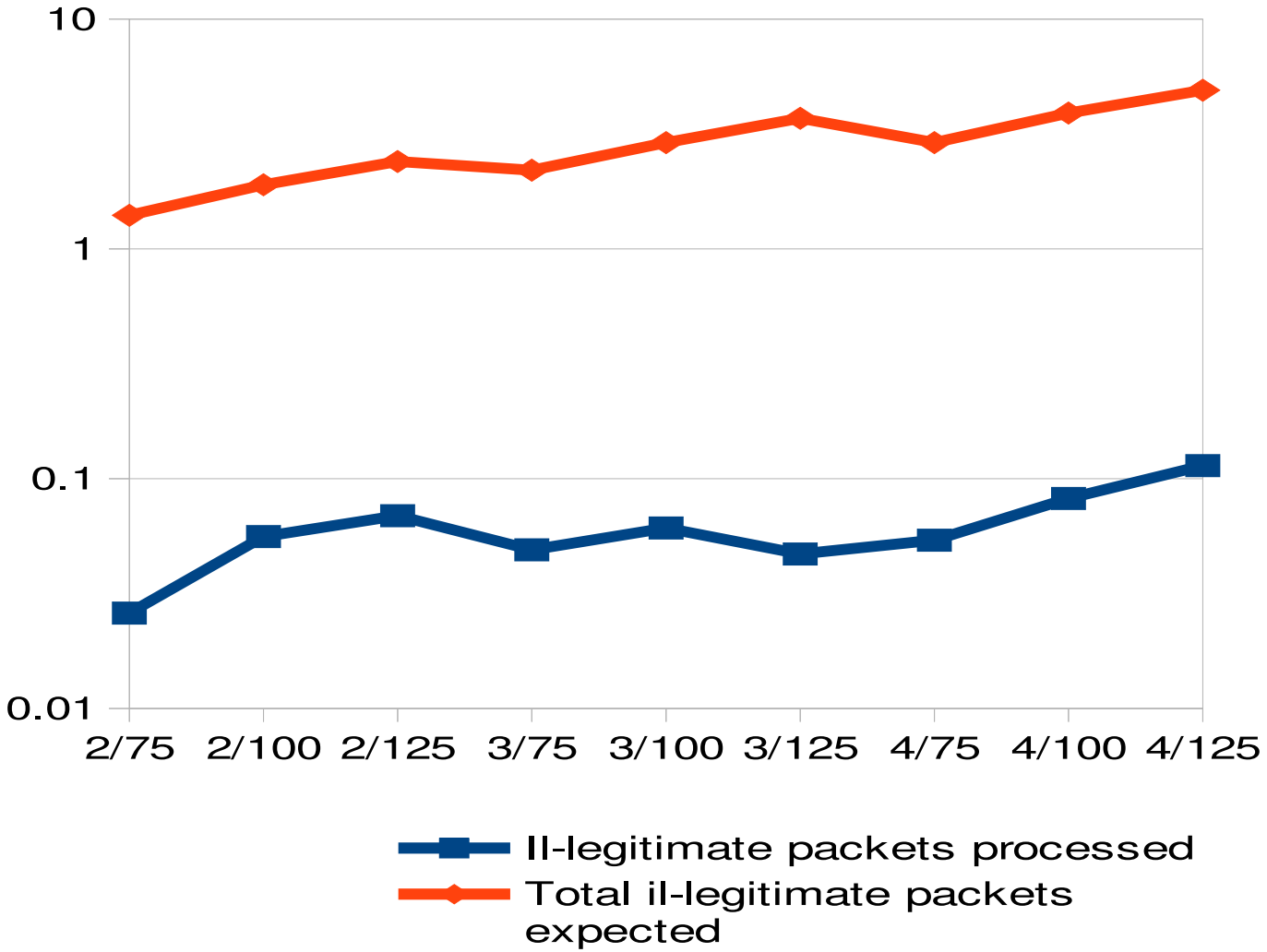
- Every 20 packets blackhosts is updated from the controller database
- Then legitimacy of each packet is checked
- On check pass, the reverse lookup of MAC is done in ARP cache
- MAC address sent on uplink to relay to inform others that this is a good MAC.
- Another controller receives information that this is bad MAC, controller database is updated through downlink server loop.

Test values to benchmark topology

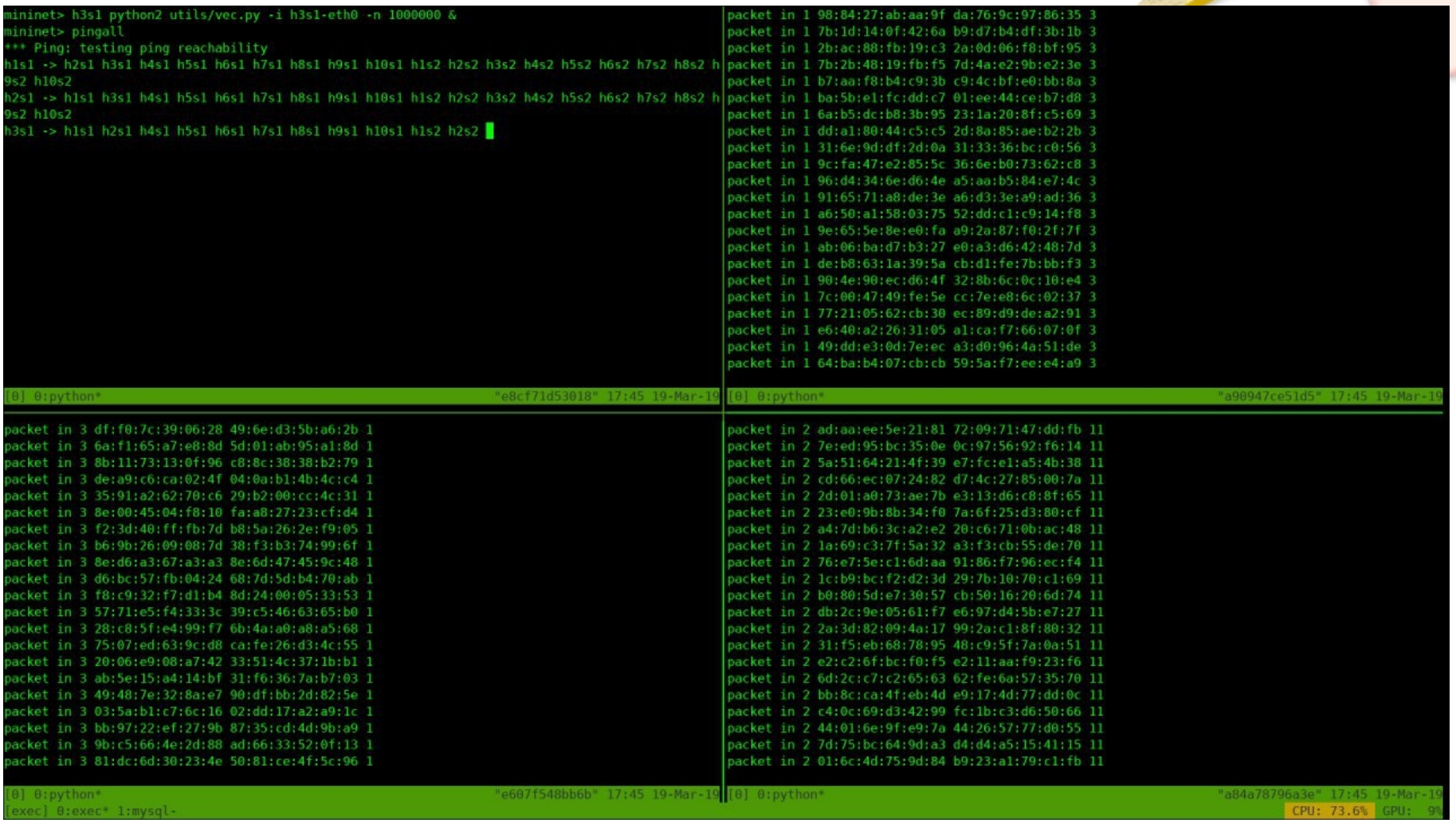
n(Subnets)/ n(Hosts/ Subnet)	Throughput (Bytes/sec) (Intra-subnet) (inter-subnet)	Bandwidth (Bytes/Sec)	Delay (Hz)	Packet Loss (%)	Flow request Rate (Hz) (Rootsw_ctrlr) (Relsw_ctrlr)
1/75	40.755	18370.830	0.261	0	457.667
1/100	34.991	17356.51	0.368	0	421.218
1/125	33.385	16059.575	0.498	0	421.218
2/75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

Test Values to Benchmark Attack Detection and Mitigation

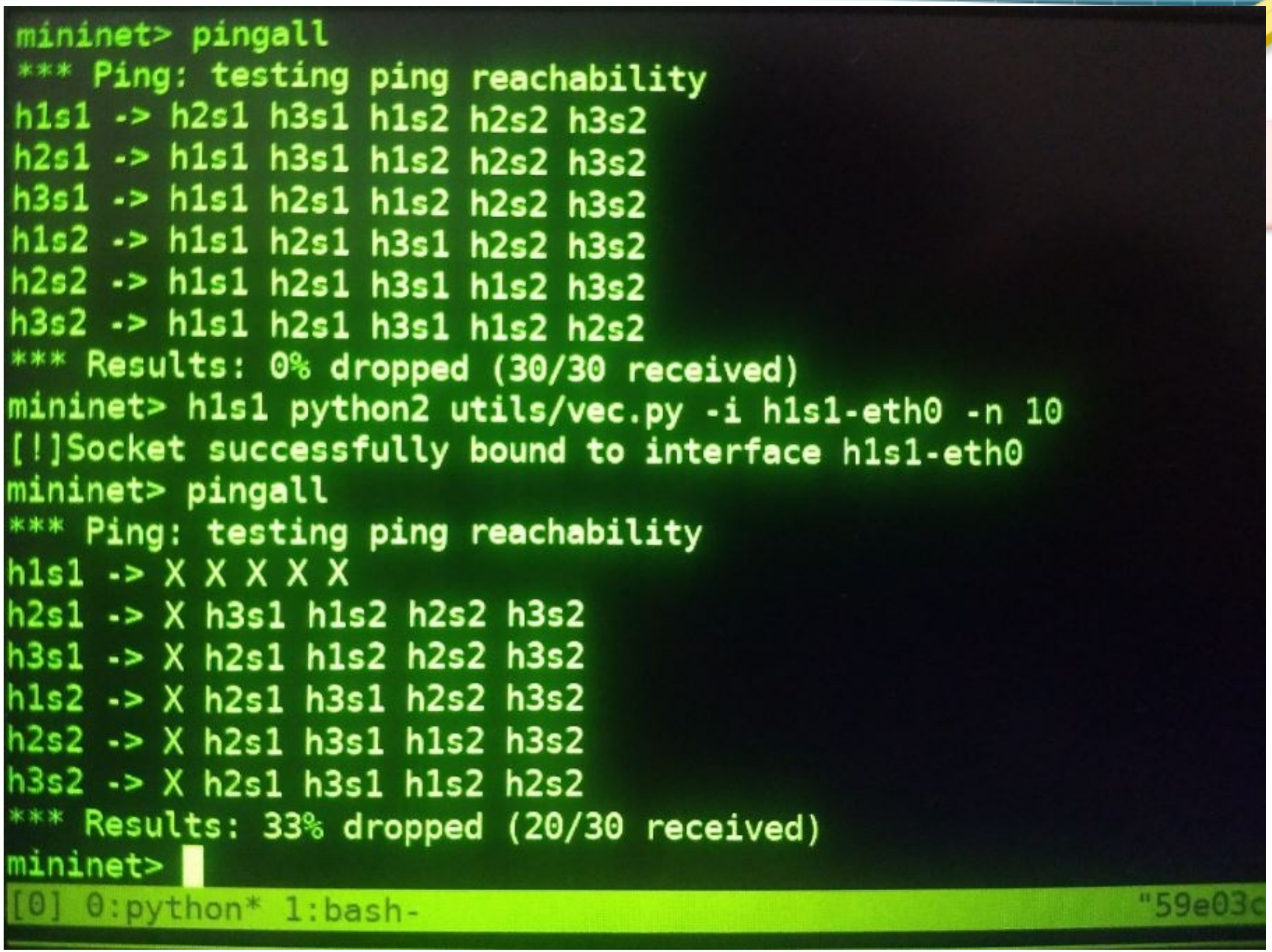
Subnets/Hosts per subnet	Il-legitimate packets processed	Total il-legitimate packets expected	% processed packets
2 subnets, 75 hosts each	25,657	1,400,000(14)	1.832
2 subnets, 100 hosts each	56,284	1,900,000(19)	2.962
2 subnets, 125 hosts each	69,142	2,400,000(24)	2.880
3 subnets, 75 hosts each	49,112	2,200,000(22)	2.232
3 subnets, 100 hosts each	61,091	2,900,000(29)	2.104
3 subnets, 125 hosts each	46,563	3,700,000(37)	1.258
4 subnets, 75 hosts each	54,311	2,900,000(29)	1.872
4 subnets, 100 hosts each	82,407	3,900,000(39)	2.113
4 subnets, 125 hosts each	114,170	4,900,000(49)	2.330


10
1
0.1
0.01
2/75
2/100
2/125
3/75
3/100
3/125
4/75
4/100
4/125
Il-legitimate packets processed
Total il-legitimate packets expected

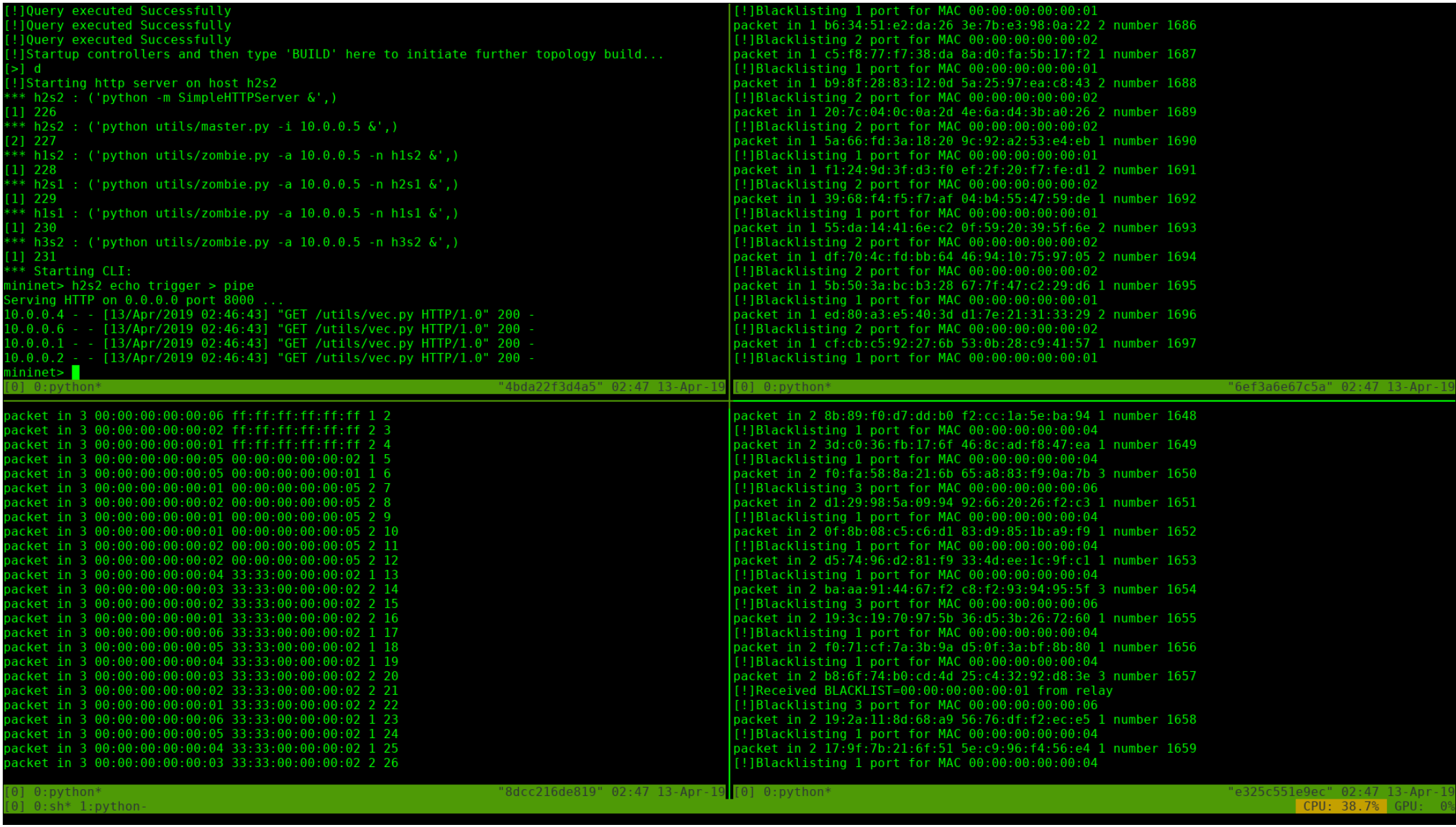
Screenshots



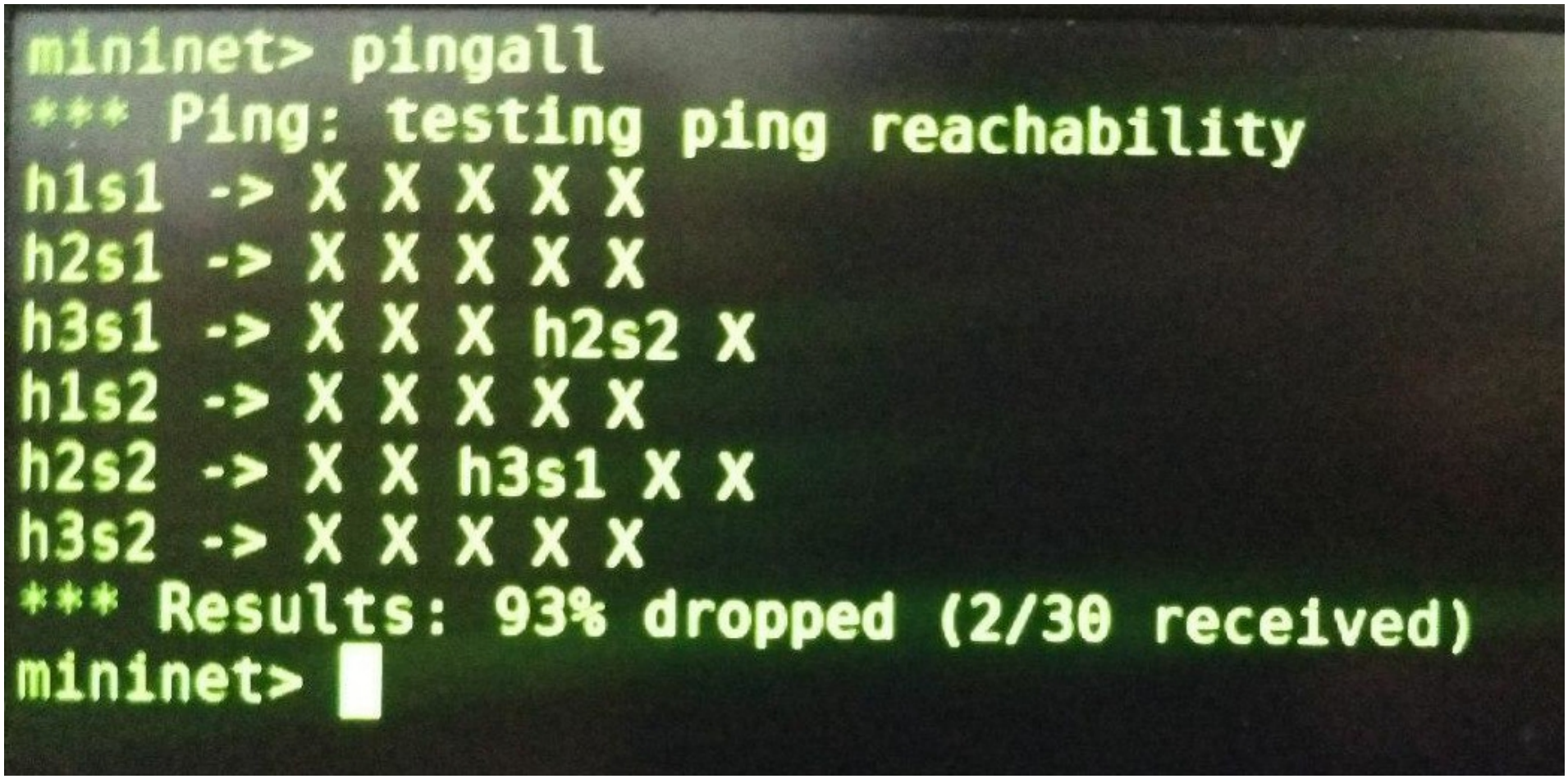
DoS attack



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



DDoS Attack Mitigation

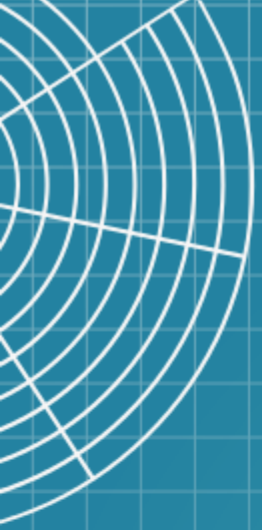
Publication details

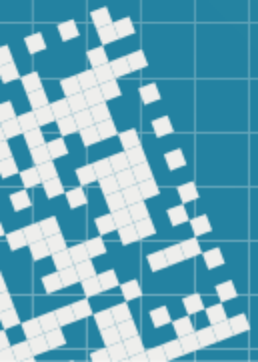
- Submitted patent application for Indian Patent journal, under the Indian Patent Act, 1970.

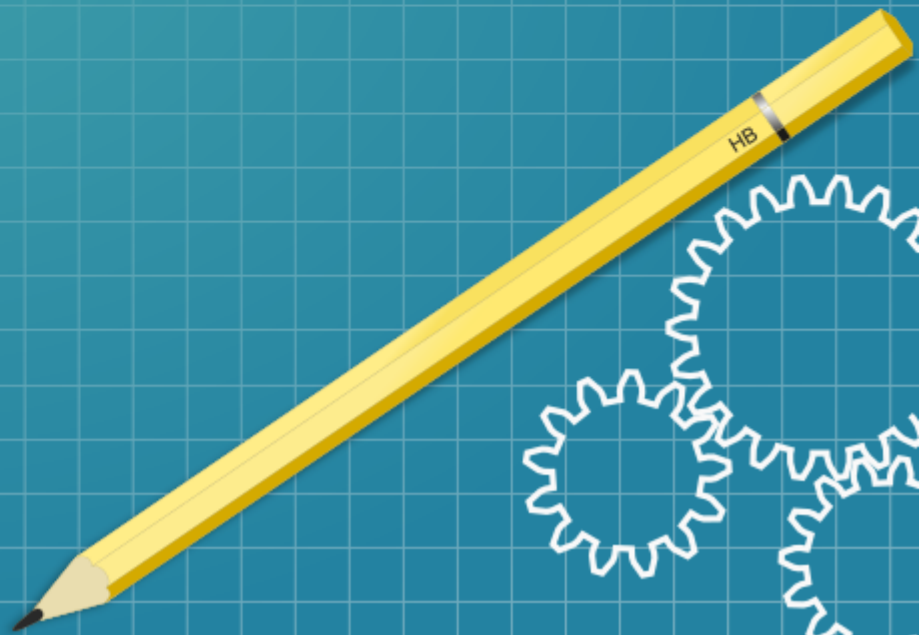
Future works

- Discover new relay applications.
- Implement a flow table overflow attack.
- Test with new topologies and compare.

Thank You!







Content Distribution

Naman	Nikhil
Introduction	Abstract
Data Plane	Proposed System
Controller Plane	Congregation of host architecture
Sub-Relay	Relay
DDoS	Blacklisting, whitelisting
Benchmark topology Test values	Benchmark attack mitigation stratergy

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

Subnets/Hosts per subnet	Total number of hosts	Number of zombies[1]	Total il-legitimate packets expected[2]
2 subnets, 75 hosts each	150	14	1,400,000
2 subnets, 100 hosts each	200	19	1,900,000
2 subnets, 125 hosts each	250	24	2,400,000
3 subnets, 75 hosts each	225	22	2,200,000
3 subnets, 100 hosts each	300	29	2,900,000
3 subnets, 125 hosts each	375	37	3,700,000
4 subnets, 75 hosts each	300	29	2,900,000
4 subnets, 100 hosts each	400	39	3,900,000
4 subnets, 125 hosts each	500	49	4,900,000

1 The zombies are calculated as following:

- Assume the topology has ‘n’ number of total hosts.
- Out of ‘n’ hosts, one is selected for the purpose of acting like a bad HTTP server, remaing unselected hosts (n-1).
- Now out of (n-1) remaining hosts, 10% are randomly selected for acting like zombies, i.e., if 150 hosts are total, 149 are left after bad server selection, 10% of 149 is 14.9. After integer roundoff, it is 14, hence the attack on such a topology will be carried out by 14 zombies simultaniously.

2 Each zombie is programmed to flood 100,000 il-legitimate packets. So number of expected il-legitimate packets are:

100000*(int_round(10%*(n-1)))