

# PROJECT REPORT

# P4AUTH IN HULA

---

SANDEEP L

CS20BTECH11044

SHANTANU PANDEY

CS20BTECH11046

MENTOR RANJITHA K

# OVERVIEW OF RESEARCH PAPERS

## P4AUTH

In recent years, there has been increased use of Software-Defined Networking(SDN) & Programmable Data Planes(PDPs). This lead to a high risk of adversarial manipulation of the packets in the network.

The paper discusses the problem that an adversary can disturb the flow of packets in the network by attacking the data plane of the switch. There are many vulnerable places in the switch where the adversary can attack. At the high level, s/he can cause many problems, like changing the route of the packets by modifying the registers in the switch data plane, changing the entries in the table, etc.

P4Auth is introduced to stop these adversarial attacks. Here, we are using P4Auth in HULA Logic.

## HULA-Algorithm

Nowadays, Data-centers have invested a lot to have multi-rooted topologies like Fat-Tress and Leaf-Spine to provide large bisection bandwidth. Effectively balancing traffic load across multiple paths in the data plane is critical to fully utilizing the available bisection bandwidth

HULA, a data-plane load-balancing algorithm, solves the Load balancing issue of Data Centers which uses multi-rooted topologies by using a probe to track congestion on all paths to a destination with the help of neighboring switches and to adapt quickly in case of changes in the network (asymmetric topology). It is designed for programmable switches, so it does not need custom hardware and hence saves a lot of money.

# OVERVIEW OF DIRECTORIES

## ONLY HULA NO P4AUTH

In this directory, we have a topology where two hosts (namely h1, h2) are connected by six switches (namely S1, S2, S3, S4, S5, S6). Hula logic is implemented in this directory for load balancing in switches, but there is NO “P4Auth (for security)”. Except for switch S6, every switch has the same code, and switch S6 works as an adversary in this topology. But switch S6 is dormant at the beginning, so we have to give some commands, ‘Run-time commands, for table forwarding so it can start working.

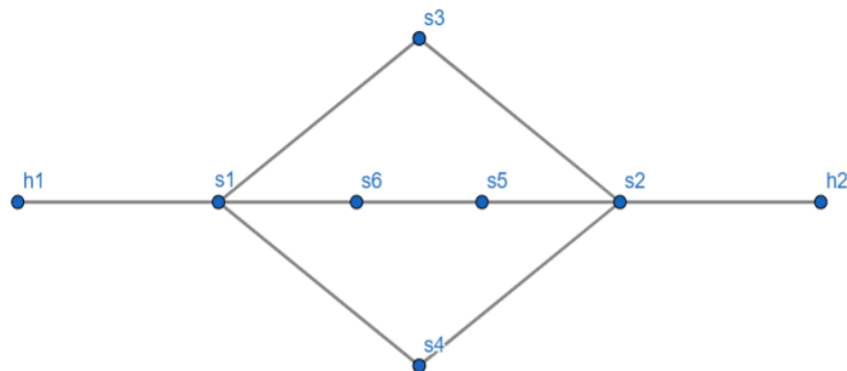


Figure 1: Topology of the network in Only HULA No P4Auth

## CRC32 MODEL WITHOUT ADVERSARY

In this directory, the topology is the same as the topology in (Only HULA No P4Auth), except for two changes. One is that there is no switch S6(hence no adversary). The other one is that there is P4Auth Authentication logic using CRC32 Algorithm installed in every switch for security purposes.

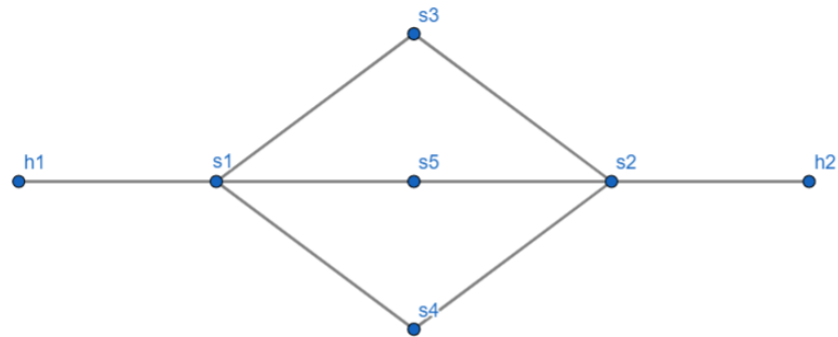


Figure 2: Topology of the network in CRC32 Model Without Adversary

## CRC32 MODEL WITH ADVERSARY

In this directory, the topology is the same as the topology in (Only HULA No P4Auth), except for just one change, i.e., there P4Auth Authentication logic using CRC32 Algorithm in every switch except S6(Adversary) for securing the network. This is a combination of the above two directories.

- In the first directory, an adversary is manipulating the packets.
- In the second directory, there is P4Auth which ensures the security of each switch's data planes.

Here we are testing the security of the network while the adversary manipulates the packets in the network.

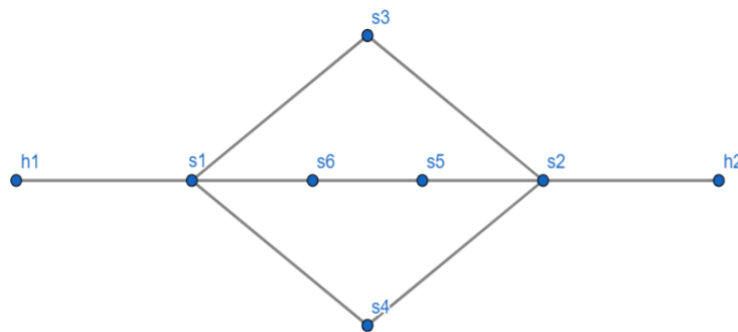


Figure 3: Topology of the network in CRC32 Model With Adversary

# OVERVIEW OF FILES

## IMPORTANT FILES

- **new\_controller.py:** this file is used to configure the data plane. It installs hula logic/P4Auth logic into the switches and calculates RTT time for different switches.
- **probe.py:** This file sends probe packets (of Hula logic) from the hosts' terminals.
- **send.py:** This file sends any message from one host to another host. The message is sent 1000 times to study packet flow and other observations.
- **receive.py:** This file receives packets/messages from another host. Please note that the receiver and the sender should run simultaneously on different hosts' terminals to send and receive packets/messages.
- **packet\_counter.py:** The file stores all the information (like checksum, dst\_ip,src\_ip) we receive from a packet/message into a file packets\_couts.txt
- **Rtt\_calculator.py:** This uses the statistics received from new\_controller.py to calculate the average RTT time.
- **test.py:** woThis
- **timer.py:** This file calculates the time taken to run a process like adding table entries, changing registers, etc.
- **topology.json:** This file contains network structure, i.e., how the switches are connected to each other and to the hosts.

- **keys.json:** This file gets generated at runtime and stores the port keys (secret keys) required for the secured communication (P4Auth) between the switches and to detect any manipulation of the packets.
- **switch1.p4:** This file has the P4 codes of switch logic with Hula logic, which gets installed in every switch from S1 to S5.
- **switch6.p4:** This file has the P4 codes without Hula logic and has code to make it work like an adversary.
- **Makefile:** It starts the mininet by generating the network's topology and activating the switches.

## DELETED FILES

- **benchmark.py:** To evaluate the port utilization of each switch under load, which snapshots the state of various registers on each switch at runtime. This file has codes from the experiment of Rachit Nigam, and we do not need them for our experiments.
- **cli\_commands/command:** This file has table entry commands for switches. We already use HULA logic for this, so we don't need it.
- **controller.py/controller1.py/orig\_controller.py:** The *new\_controller.py* file is the latest and updated version of the *controller*. So, we can use *new\_controller.py*, and there is no use for the other controllers which were not working as we need them to work for our experiment. Hence, deleted.
- **original\_switch1/p4auth\_hulaswitch/switch1\_old\_p4/switch\_temp\_halt/switch.p4.save1/switch2/switch6.p4.save/switch1\_hula\_alone:** These switches are the modifications of switches *switch1.p4* and *switch6.p4*, and we are not using them now. Hence, deleted.

- **out.p4:** This is generated when we run test.py so we can delete it.
- **topology-orig.json:** This file is deleted since the network's topology is changed, and we are using the updated file topology.json.

## EXPERIMENTS & OBSERVATIONS

**NOTE:** The procedure to run the following experiments is provided in the documentation [here](#).

### ONLY HULA NO P4AUTH (WITH ADVERSARY)

#### EXPERIMENTS

- We used this directory as a learning ground to learn about how to run the code and learn about Hula logic and how it is implemented inside the switches and how it uses probes to calculate the keep track of best hops and of path utilizations.
- We send the packets from one host to another to see how the adversary is attacking.
- We changed the topology at run-time to see if Hula was able to adopt the change in network topology.

#### OBSERVATIONS

- ➔ All the information that we have collected about different files which are mentioned above are mainly due to experimenting through the files and codes of this directory.
- ➔ As we sent probes, the adversary was modifying the path utilization to divert the traffic to itself. It will set the path\_util=0 so that the path will look free of any congestion, but in reality, it will be flooded with traffic.

```

246     action packet_forward(egressSpec_t port){
247         if(hdr.hula.isValid()){
248             hdr.hula.path_util = 0;
249         }
250         standard_metadata.egress_spec = port;
251     }

```

In the above code snippet of Switch S6, we can see how it is changing the path utilization to zero to divert traffic to its route.

- Even on changing the network's topology on run time, Hula quickly adapted to the change and diverted the packets using other routes.

## CRC32 MODEL WITHOUT ADVERSARY

### EXPERIMENTS

- In this experiment, since there is no adversary, there is no manipulation of the packets; still, there is P4Auth to block any kind of attack from an adversary.
- We used this experiment to observe the RTT of the packets and compare these with the ones calculated in Only HULA without P4Auth experiment(Above experiment).

### OBSERVATIONS

- There are three paths between the hosts  $h1$  and  $h2$ , as shown in Figure 2. As all three paths are identical, hula logic chooses any of the three paths uniformly at random each time we run the *probe* packets. And when we send messages from one host to the other, the packet will travel with the chosen path. In the screenshot below, the graph inside the **RED** rectangle shows the probes from hosts  $h1$  &  $h2$ , and the graph in the **PURPLE** rectangle shows the flow of the packet first part  $h1 \rightarrow h2$  and second part  $h2 \rightarrow h1$  within the same rectangle. If we



compare it with the other **RED** and **PURPLE** rectangle pair, the paths chosen by the packets are different.

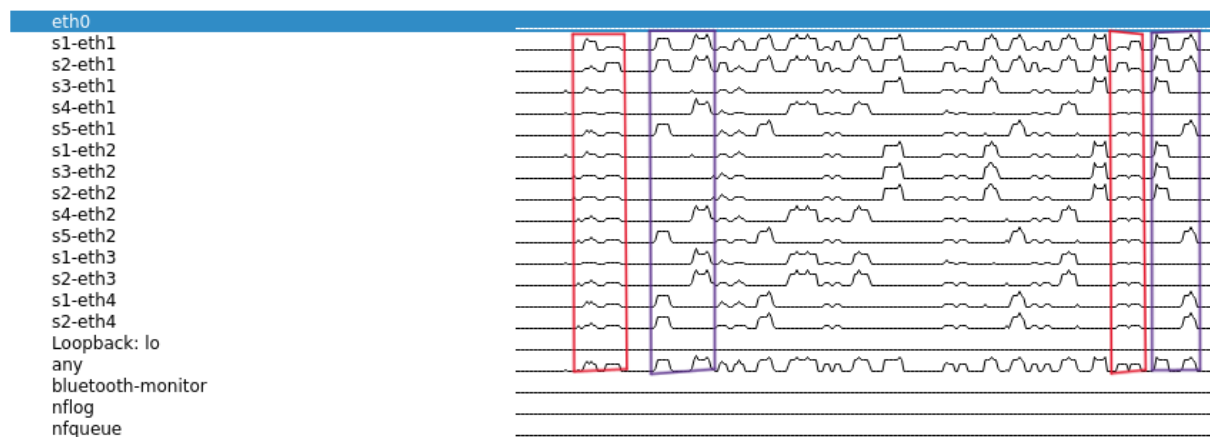
	From	Through	To
First <b>PURPLE</b> rectangle	h1	s5	h2
	h2	s4	h1
Second <b>PURPLE</b> rectangle	h1	s3	h2
	h2	s5	h1

**Note:** In each rectangle, the first peak(left part) is probe or message from h1, and the second peak(right peak) is probe or message from h2.

Welcome to Wireshark

## Capture

...using this filter:



→ There is a significant difference in the average RTT calculated. We can see the increase in RTT due to P4Auth calculating the checksum/digest of CRC32 model

```
10 iterations done
Average running time : 2352776.7199999997
p4@p4:~/Downloads/P4-Auth-siphash-main/crc32_models/without_adversary_model_crc32$
```

```
10 iterations done
Average running time : 933231.18
p4@p4:~/Downloads/P4-Auth-siphhash-main/only_hula_no_p4_auth$
p4@p4:~/Downloads/P4-Auth-siphhash-main/only_hula_no_p4_auth$
```

## CRC32 MODEL WITH ADVERSARY

### EXPERIMENTS

- Here, we did the same experiments as the above one, but here there is an adversary in the switch S6.
- We sent probe packets from the hosts  $h1$  &  $h2$ , which calculates the best path for sending the packets. After that, we sent packets from both  $h1 \rightarrow h2$  and  $h2 \rightarrow h1$  and observed what paths were chosen by the packets.

### OBSERVATIONS

- In the below screenshot, the graph inside the **RED** rectangle is the graph of *probes*. We can see that the probe packet is not captured in S5 from S6 since P4Auth detects the manipulation of the packet from the adversary in S6. Likewise, it will not reach from S6 to S1 also.
- Since the probe packets are not reaching S1 or S5 from S6, HULA will not consider the path, including the switch S6, and calculates the best possible path from the remaining two paths. For the same reason, we can see that the messages from both  $h1 \rightarrow h2$  and  $h2 \rightarrow h1$  will not take the path including S6 and chooses a path uniformly at random from the other two paths since both the paths are identical.

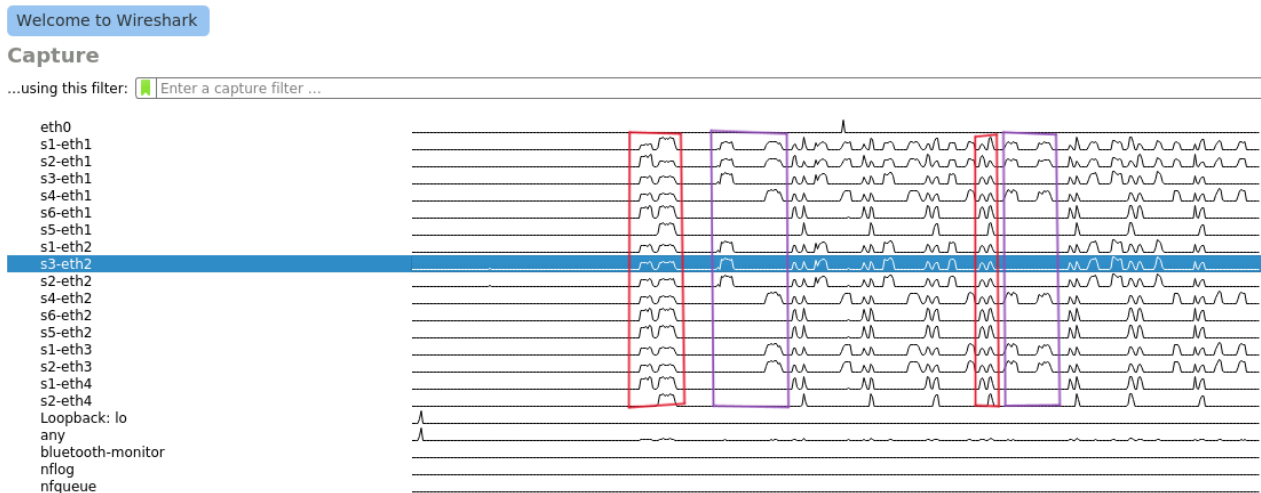
In the below table, we can see the two examples.

→

	From	Through	To
First <b>PURPLE</b>	$h1$	$s3$	$h2$

rectangle	h2	s4	h1
Second <b>PURPLE</b> rectangle	h1	s4	h2
	h2	s4	h1

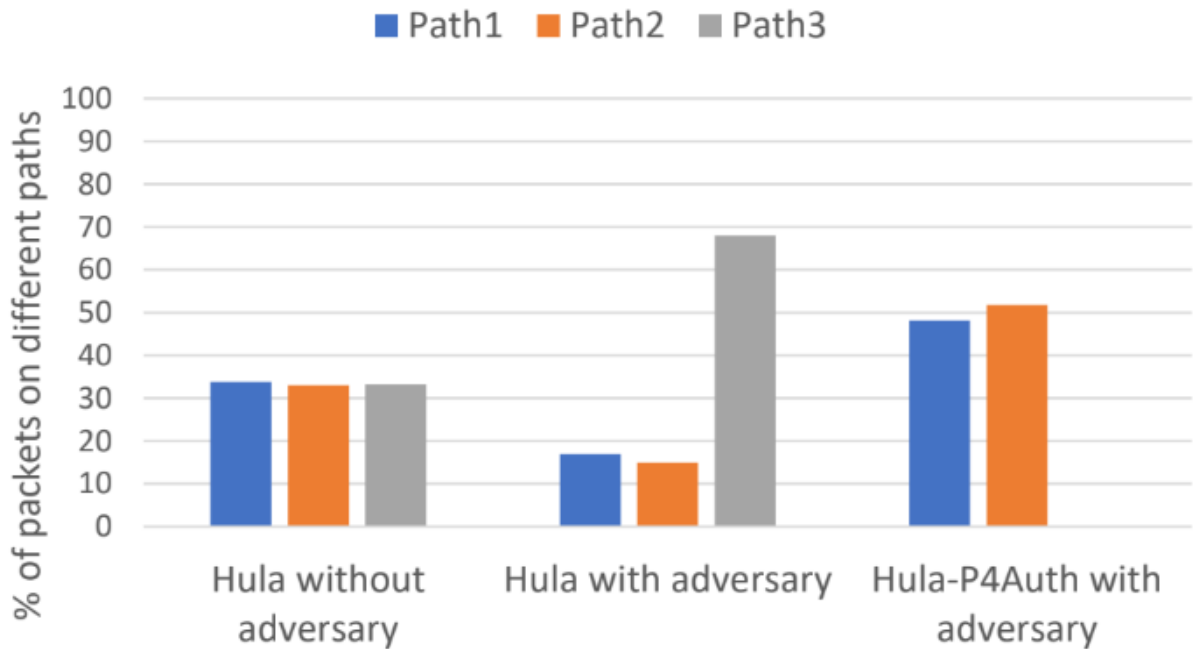
**Note:** In each rectangle, the first peak(left part) is probe or message from h1, and the second peak(right peak) is probe or message from h2.



## EXPERIMENTS & OBSERVATIONS

Sir asked us whether we did the below experiment (while giving the presentation).

So we did the below experiment and found similar observations.



Graph 1

## EXPERIMENTS

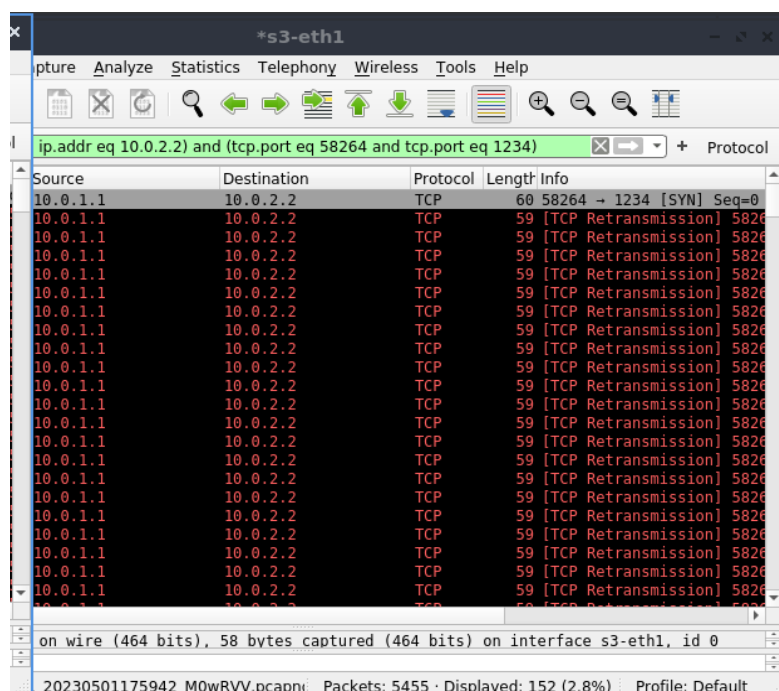
We conducted the above-mentioned three experiments as it is. But this time, we ran the probes continuously by opening the h1 and h2 terminals again(only for probes) while sending the packets(messages) between hosts. Since the *probe* is calculating the best possible path dynamically at run time, the path of the packets(messages) was changing continuously, and the observations made among the three directories are shown below.

**NOTE:** The number of packets(messages excluding probes) is shown in the lower right corner of the further images in the field “Displayed”.

## ONLY HULA NO P4AUTH (WITH ADVERSARY)

- This was the second experiment in Graph 1.
- In the below table, we can see that since there is an adversary and the adversary will try to manipulate the *probes* that there is very less traffic going through the adversary, so choose that path (here path containing S6). So, in the table below, we can see that most of the packets(messages) were sent from *h1* -> *h2* through the path containing S6, and the remaining packets(messages) were sent through S3 and S4 almost equally since they both are identical.

	s3	s4	s6
Percentage of packets transmitted	$152/811 \cdot 100 = 18.74 \%$	$172/811 \cdot 100 = 21.20 \%$	$487/811 \cdot 100 = 60.04 \%$



**\*s4-eth1**

View Go Capture Analyze Statistics Telephony Wireless Tools Help

10.0.2.2

Time	Source	Destination	Protocol	Length	Info
12.578607547	10.0.1.1	10.0.2.2	TCP	59	58264 → 1234 [SYN] Seq
14.861860702	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
15.304143330	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
15.455083721	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
16.515572064	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
18.407196491	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
19.713057619	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
20.513297382	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
22.629666957	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
22.857806389	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
23.131413004	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
24.674077482	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
25.493287378	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
26.050585604	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
26.652752850	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
28.226401295	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
33.928965361	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
36.118431057	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
36.676067844	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
37.193989982	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
38.801854707	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]
39.068782702	10.0.1.1	10.0.2.2	TCP	59	[TCP Retransmission]

1231: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface s4-eth1, id 0

wireshark\_s4-eth1\_202...1175949\_LtVyi0.pcapng Packets: 5475 · Displayed: 172 (3.1%) Profile: Default

**\*s6-eth1**

dit View Go Capture Analyze Statistics Telephony Wireless Tools Help

1.1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)

Time	Source	Destination	Protocol	Length	Info
5 249.910619874	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
6 250.436301206	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
1 250.727773211	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
8 251.020016167	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
2 251.844233168	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
8 252.094866582	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
3 252.335097934	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
9 252.572096017	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 252.816644440	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
1 253.062761523	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
8 253.309583313	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 253.595450194	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
0 253.823260325	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 254.092592209	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
1 254.378491131	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
3 255.386433364	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
8 255.646865439	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 256.840486044	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
8 257.351491630	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 257.618466951	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
2 257.899236706	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
4 258.394089787	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra
5 258.887514505	10.0.1.1	10.0.2.2	TCP	58	[TCP Retra

wireshark\_s6-eth...57\_1rgjc3.pcapn Packets: 5792 · Displayed: 487 (8.4%) Profile: Default

## CRC32 MODEL WITHOUT ADVERSARY

→ This was the first experiment in Graph 1.

→ In the below table, we can see that since there was no adversary and all the paths(here switches S3, S4, S5) are identical, there will almost be equal no of packets through each path.

	s3	s4	s5
Percentage of packets transmitted	$121/343 \times 100$ = 35.27 %	$119/343 \times 100$ = 34.69 %	$103/343 \times 100$ = 30.02 %

**Capturing from s3-eth1**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

10.0.1.1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)

No.	Time	Source	Destination	Protocol	Length	Info
69	97.525136024	10.0.1.1	10.0.2.2	TCP	60	58264 → 1234
70	97.785567592	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
71	98.020817006	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
72	98.267935978	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
75	99.500743591	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
76	99.745102477	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
77	99.988911342	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
78	100.232273463	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
82	102.727080733	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
83	102.967781733	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84	103.207041988	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85	103.444193433	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
87	103.681208118	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
88	103.918478821	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
89	104.163269365	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
90	104.412141119	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
94	106.827836400	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
95	107.073212709	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
96	107.316120639	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
97	107.563518689	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
100	109.005251030	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
101	109.247740907	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra

0000 ff ff ff ff ff ff 00 00 00 00 01 01 08 00 45 00 .....E.  
0010 00 2e 00 01 00 00 40 06 63 c7 0a 00 01 01 0a 00 .....@.c.....  
0020 02 02 e3 98 04 d2 00 00 00 00 00 00 00 50 02 .....P.

s3-eth1: <live capture in progress> Packets: 310 · Displayed: 121 (39.0%) Profile: Default

**Capturing from s4-eth1**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

10.0.1.1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)

No.	Time	Source	Destination	Protocol	Length	Info
82	107.780325933	10.0.1.1	10.0.2.2	TCP	59	58264 → 1234
83	108.013036094	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84	108.252889621	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85	108.493675753	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
86	108.747950477	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
90	110.931541405	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
91	111.163772444	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
92	111.396410774	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
93	111.635676673	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
94	111.892105925	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
102	118.344261570	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
103	118.583726422	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
104	118.836507815	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
105	119.072739470	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
109	120.296677123	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
110	120.543316406	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
111	120.804898214	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
112	121.056410040	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
115	122.496275890	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
116	122.731979889	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
117	122.980506688	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
118	123.227780768	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra

0000 ff ff ff ff ff ff 00 00 00 00 01 01 08 00 45 00 .....E.  
0010 00 2d 00 01 00 00 40 06 63 c8 0a 00 01 01 0a 00 .....@.c.....  
0020 02 02 e3 98 04 d2 00 00 00 00 00 00 00 50 02 .....P.

s4-eth1: <live capture in progress> Packets: 315 · Displayed: 119 (37.8%) Profile: Default



Capturing from s5-eth1

View Go Capture Analyze Statistics Telephony Wireless Tools Help

1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234) X → + Protocol

Time	Source	Destination	Protocol	Length	Info
80.999077376	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
81.229422978	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
81.461210513	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
81.703341021	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
81.934147313	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
83.137675029	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
83.382333585	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
83.638660332	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
83.882768359	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84.122516372	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84.377387115	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84.629534644	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
84.873714133	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85.120375310	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85.362278552	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85.614214421	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
85.886056830	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
86.134123353	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
88.365736888	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
88.621613282	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
88.869783839	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
89.113838261	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra

5-eth1: <live capture in progress> Packets: 233 · Displayed: 103 (44.2%) Profile: Default

## CRC32 MODEL WITH ADVERSARY

- This was the third experiment in Graph 1.
- In the below table, we can see that there is an adversary, and also we have P4Auth. The adversary tries to manipulate the *probe* packets from switch S6 and the P4Auth installed in other switches detects this manipulation and doesn't include those probe packets while calculating the best path using HULA logic. As a result, there will not be any flow of packets(messages) through the adversary(here, S6). And the packets will be sent through any of the other two paths. Since they are identical, almost equal number of packets(messages) will pass through switches S3 and S4.

	s3	s4	s6
Percentage of packets transmitted	$492/1000 \cdot 100 = 49.2 \%$	$508/1000 \cdot 100 = 50.8 \%$	$0/1000 \cdot 100 = 0 \%$

The screenshot shows a Wireshark packet capture window titled '\*s3-eth1'. The filter bar contains the expression '10.0.1.1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)'. The packet list shows a series of TCP packets from source 10.0.1.1 to destination 10.0.2.2, all with length 59 and status '[TCP Retra]'. The status bar at the bottom indicates 'Packets: 1599 · Displayed: 492 (30.8%)'.

No.	Time	Source	Destination	Protocol	Length	Info
886	52.038389612	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
888	52.095465915	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
890	52.162722681	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
891	52.230750590	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
893	52.287235617	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
895	52.346990510	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
896	52.396454408	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
898	52.447660088	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
900	52.498825747	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
901	52.556732856	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
903	52.607526499	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
905	52.662548664	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
907	52.739932518	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
908	52.807497595	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
910	52.874182479	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
912	52.942592136	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
913	52.994996923	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
915	53.054401587	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
917	53.110558057	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
921	53.331887547	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
922	53.404099826	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]
924	53.462818705	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra]

0000 ff ff ff ff ff 00 00 00 01 01 08 00 45 00 .....E-  
0010 00 2d 00 01 00 00 40 06 63 c8 0a 00 01 01 0a 00 .....@. C.....  
0020 02 02 e3 98 04 d2 00 00 00 00 00 00 00 50 02 .....P.

wireshark\_s3-et...9\_eTKfD0.pcapng Packets: 1599 · Displayed: 492 (30.8%) Profile: Default

\*s4-eth1

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

10.0.1.1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)

No.	Time	Source	Destination	Protocol	Length	Info
478	29.827273256	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
479	29.888953416	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
481	29.951783523	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
485	30.187477137	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
486	30.247483047	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
488	30.303724211	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
490	30.361655237	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
492	30.432059307	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
493	30.494963794	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
495	30.550997338	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
498	30.734358740	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
500	30.814369334	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
502	30.872894441	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
503	30.931071195	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
509	31.329686329	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
511	31.410512102	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
513	31.555697930	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
516	31.724549004	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
517	31.775050433	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
519	31.843185501	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
521	31.895307032	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra
527	32.340027057	10.0.1.1	10.0.2.2	TCP	59	[TCP Retra

0000 ff ff ff ff ff ff 00 00 00 01 01 08 00 45 00 .....E.

0010 00 2d 00 01 00 00 40 06 63 c8 0a 00 01 01 0a 00 .....@.C.

0020 02 02 e3 98 04 d2 00 00 00 00 00 00 00 00 50 02 .....P.

wireshark\_s4-et...2\_5uX113.pcapn Packets: 1616 · Displayed: 508 (31.4%) Profile: Default

\*s6-eth1

View Go Capture Analyze Statistics Telephony Wireless Tools Help

1 and ip.addr eq 10.0.2.2) and (tcp.port eq 58264 and tcp.port eq 1234)

Time	Source	Destination	Protocol	Length	Info
------	--------	-------------	----------	--------	------

wireshark\_s6-eth...17\_YKZ1y8.pcapng Packets: 1114 · Displayed: 0 (0.0%) Profile: Default



## CONCLUSION

- The Hula Logic works very efficiently with the help of a continuous stream of probes. It never fails to find the best path for every end-to-end communication. It handles changes in topology, asymmetry, and failure of switches with ease. Hence it has the potential to be used extensively in Data centers.
- Although HULA is a very effective Load Balancer, it also opens up a weak point in the system that can be targeted by an adversary, and that can lead to improper load balancing, which results are not in favor of data centers in many ways.
- P4Auth has a lot of potentials to provide security in the data plane of a network. One of its potentials is to combine it with HULA logic to secure its weak point by using a CRC32 model.
- P4Auth combined with Hula is able to keep away the adversarial attacks that can disrupt the load balancing.
- P4Auth is adding an additional time overhead in RTTs, which is there due to the verification of digest (CRC32).



## Pending experiment:

There is an experiment that we had to perform where we had to calculate the difference between the time is taken to read or write table entries and register entries with Hula only and with P4Auth & Hula

To do this experiment, we found out that we have to use “extern” and even if we implement the “extern” to calculate the time take we also have to modify the P4 compiler code to print the time on the console that is calculated inside the switch which is being run using P4 code. We tried to learn about the extern and P4 compiler, but we were not able to manage the time needed to go deep into it.