Analysis of OpenFlow Networks.

Vikram Kulkarni

Jayesh Kawli

Introduction:

Enterprise data center networks are rapidly reaching a breaking point, because of the data center network scale and complexity is testing the limits of the networking equipment and IT operations. Networking pioneers and vendors are trying their best to handle the huge and exponentially growing networks, and with the use of VMware and cloud computing, there job is not getting any easier. The vendors are trying to fulfill the need of the ever greedy networking industry by implementing more network equipment and expanding there horizon by inventing techniques like aggressive data center consolidation, i.e. the existing organizations simply they will simply undergo massive scale as they house more devices, applications and network traffic. Also adding up to networking worries, about 25% of large organizations have deployed SOA or web based applications. Those web based application basically are applications which use X86 server tier and hence server to server communication has increased tremendously. This is leading to more problems like Network Segmentation and security, Traffic engineering, Load balancing, Congestion control and Network Provisioning and configuring those networks. The top three problems faced by Data center Networks were Network security, Network Performance and Network management.

While the networking community was dealing with all those problems, Martin Casado, a PhD student from Stanford was presenting a paper named Ethane which uses flow based network and a central administration. This paper further became the basis of Openflow. OpenFlow is an interface for remotely controlling the forwarding tables in network switches, routers, access points and any network components like firewall or Intrusion detection system. In simple words, Openflow network is software defined network which provides solutions which solves problems like security, mobility and scalability. Open Flow switches are basically a switch which consists of a centralized controller. This controller has information about the flow table of each switch and it learns the path to reach all networks and once it has learned it, it will store it. Open Flow switches provide a platform like NOX or Beacon through which anyone can write programs in java and python to control the packet flow. And decide which port should the packet parse from and which port should be safeguarded for congestion control and security against DOS attacks. The next figure explains the SDN network Architecture in detail.

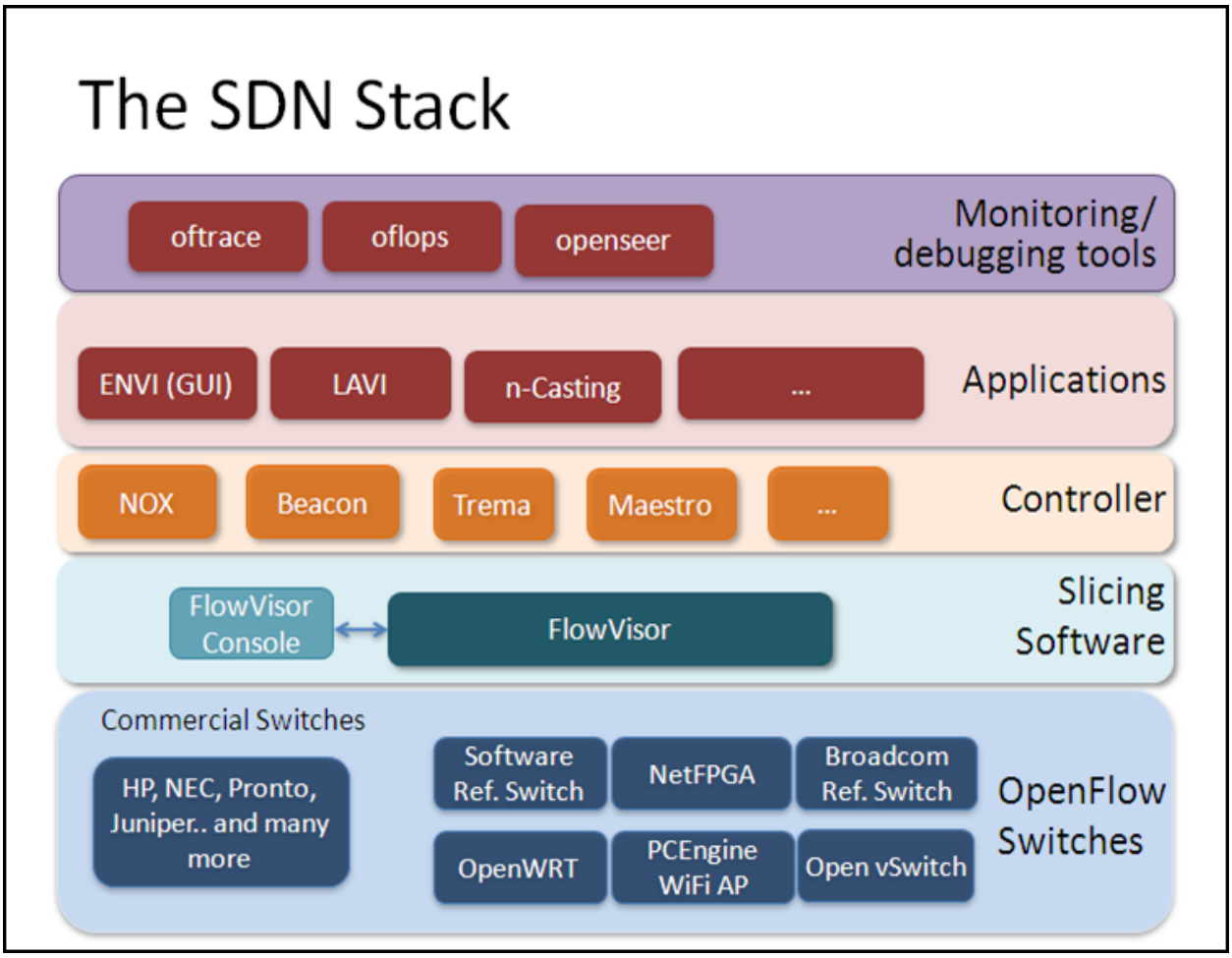


Fig 1.0

Openflow switches can be of two types Commercial and the local ones, while the commercial ones are costly, they can be implemented in large scale organizations. HP, NEC and more recently IBM are the major companies deploying Open Flow Switches. Above those switches is Slicing software like FlowVisor which is used to virtualize the Network. There is a controller above that which uses NOX or beacon platform to write programs to control the network which is the major use of a SDN. The most commonly used GUI is LAVI and ENVI. The most commonly debugging tool is oftrace.

Our aim is to study this OpenFlow network by deploying it practically and configuring a controller, providing it certain flows, understanding the network level concepts, and then analyzing the security behind OpenFlow networks. There have been various study papers before which speak about OpenFlow networks and there deployments in campus networks or which states different specific part like development of new language called frantic which will

benefit in development of OpenFlow network. Although an overall Risk analysis of OpenFlow networks was not conducted.

We have deployed the following architecture in a virtual machine. Firstly we analyzed a simple switch controller architecture shown in following figure.

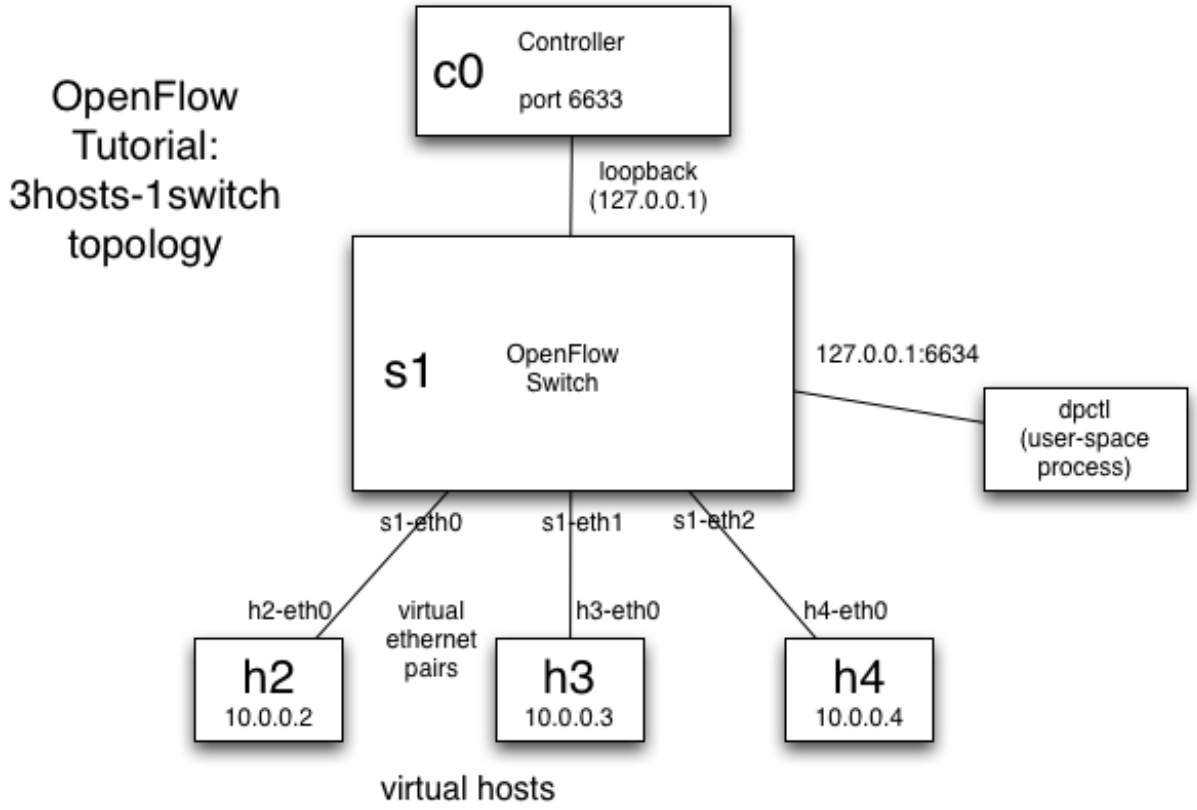


Fig 2.0

In the above figure we have configured three hosts to be connected to a switch and dpctl user space is used to debug the OpenFlow switch and see the flow in the switch. Controller c0 is where the programs like this are used for a simple data flow path.

```
def install_datapath_flow(self, dp_id, attrs, idle_timeout, hard_timeout,
                              actions, buffer_id=None,
                              priority=openflow.OFP_DEFAULT_PRIORITY,
                              inport=None, packet=None):
```

Similarly there is a simple function called send_openflow () for forward packets.

We have also deployed a router and analyzed the following network configuration to learn how the packets are routed between different subnets.

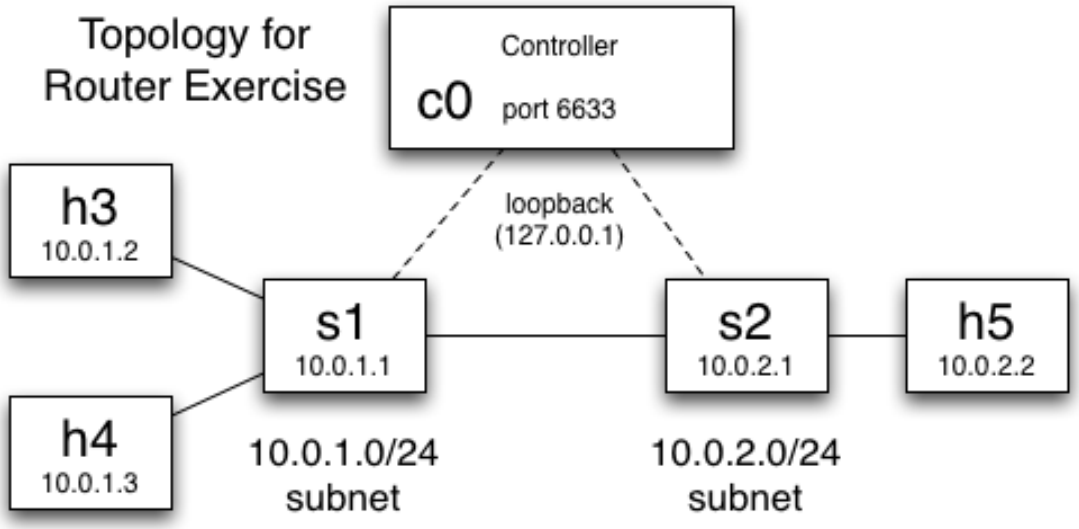


Fig 2.1

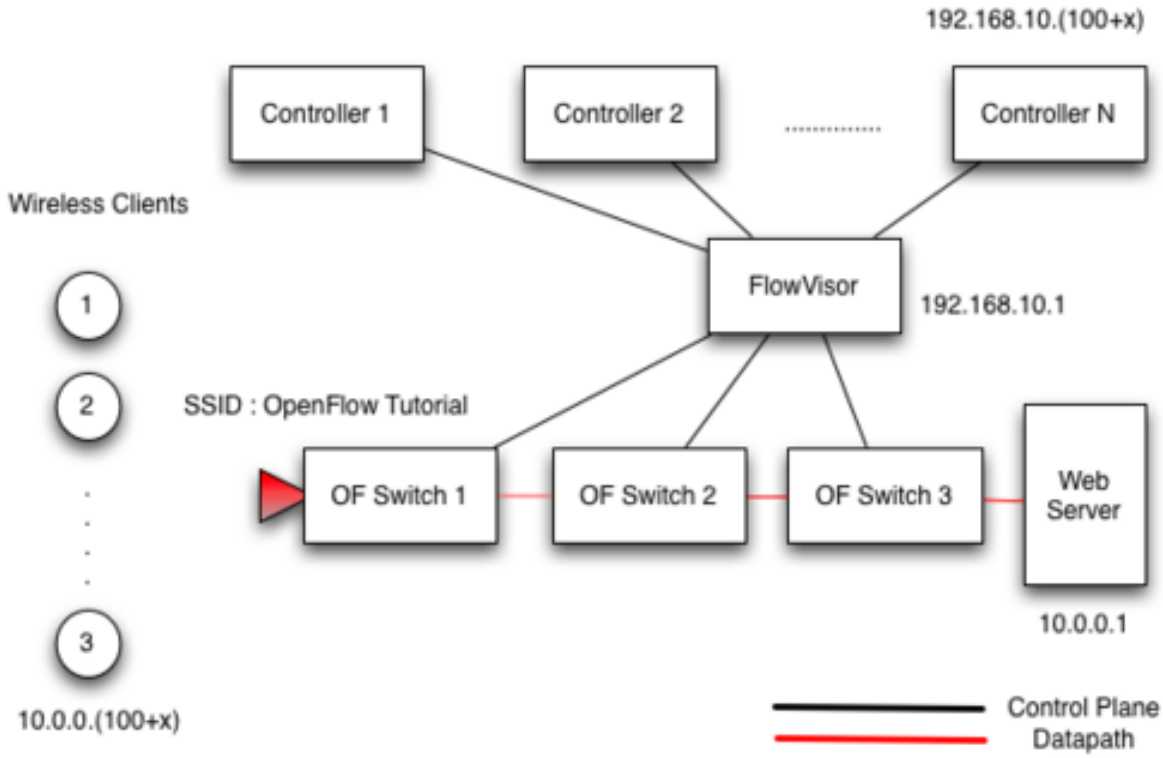


Fig 2.2

Risk Analysis:

STEP 1) Define use scenarios

- Client using webserver to access various services provided by Web site
- Host using open flow network to surf Internet
- Admin using Dpctl to debug controller

STEP 2) Gather a list of Dependencies

- A Controller using iOS like NOX to write programs using python or java beacon
- Routers to route traffic between subnet by using rules and path from controller
- Switches to perform switching within a network
- Dpctl user space for debugging and entering paths in controller
- Host computers which use OpenFlow network to access internet
- Web server, which acts like a front end for clients
- DNS server, for IP to domain name mapping

STEP 3) Define Security Assumption

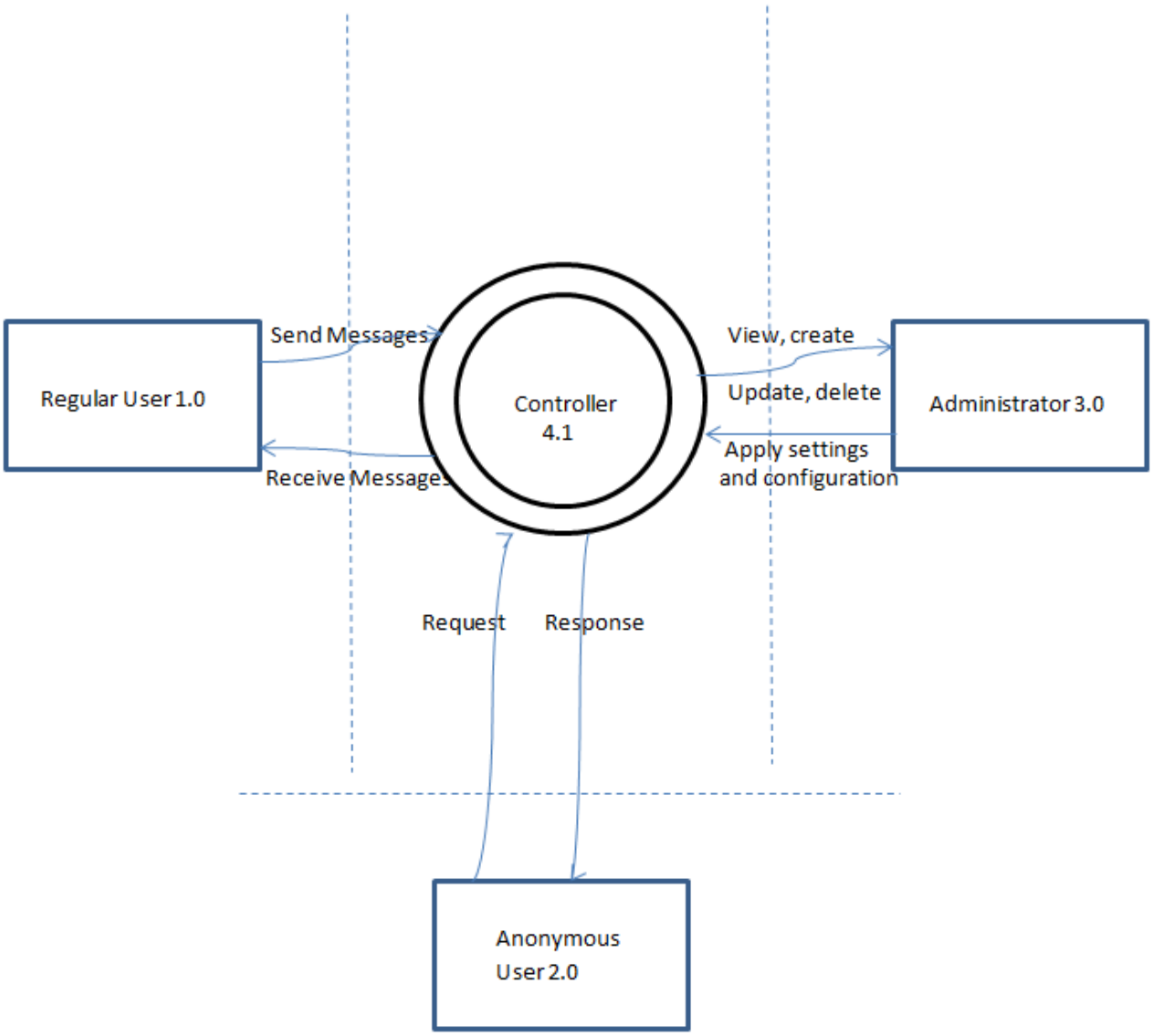
- Encrypted channel and protocol like SSL is used for Safe communication between server and client
- Anything above network layer is supposed to be secured by using usual security practices
- Application code for Web server is supposed to be secure
- The Controller is not accessible to outside world apart from the admin and users
- Anonymity of user is supposed to be implemented to protect user privacy

STEP 4) External Notes

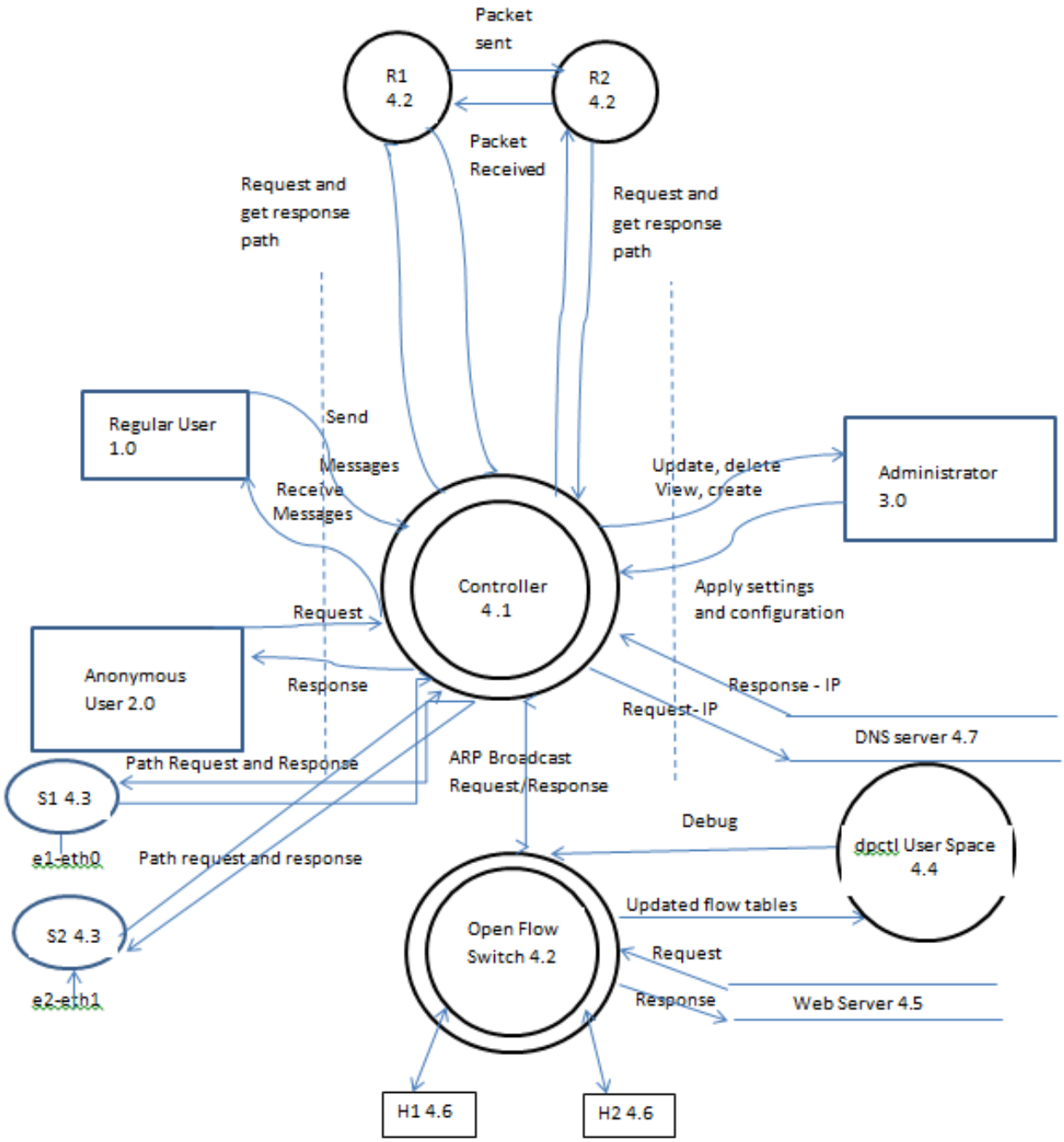
- Admin can change any flow from the flow tables in the Controller
- User should not be able to distinguish whether it is an OpenFlow or normal Network

STEP 5) Context Diagram and DFD

Context Level Diagram for OpenFlow Network:



DFD Level 0 for Open Flow Network



STEP 6) Identify Threats to the system

- **External Entities**
    1. Normal User (1.0)
    2. Anonymous User (2.0)
    3. Admin (3.0)

- **Process**
    1. Controller server. (4.1)
    2. Router R1, R2. (4.2)
    3. Switches S1,S2 (4.3)
    4. Dpctl user space. (4.4)
    5. Web server. (4.5)
    6. Hosts H1,H2 (4.6)

- **Data store.**
    1. DNS server. (4.7)

- **Processes**
    1. Openflow switch requests path to controller. (4.3 → 4.1)
    2. Openflow responds with path. (4.1 →4.3)
    3. R1 or R2 requests path. (4.2 → 4.1)
    4. Openflow Responds with path. (4.1 → 4.2)
    5. Dpctl configures Static new path. (4.4.1 → 4.1)
    6. Dpctl Configures dynamic Protocol to Controller to configure paths.(4.4.2 → 4.1)
    7. Openflow controller responds with Updated flow table. (4.1 → 4.4)
    8. S1 sends packet to S2 in regards to rules and path by controller. (4.3.1 → 4.3.2)
    9. R1 routes packets to R2 (4.2.1 → 4.2.2)
    10. Open Flow requests an ARP broadcast. (4.1 → 4.3,4.2)
    11. Web client Sends packet to web server. (4.6 →4.5)
    12. Web server replies back to Web client. (4.5→4.6)
    13. Web clients ask DNS for IP address for domain name (4.6 → 4.7)
    14. DNS replies with IP address (4.7 → 4.6)

STEP 7) Determine and classify Threat Types.

DFD element Type	Threat Types	DFD Item number
External entities	**STR**	(1.0) (2.0) (3.0)
Processes	**STRIDE**	(4.1)(4.2)(4.3)(4.4)(4.5)(4.6)
Data Store	**TID**	(4.7)
Data Flow	**TID**	(4.3 → 4.1 → 4.3), (4.2 → 4.1 → 4,2), (4.4.1 → 4.1), .(4.4.2 → 4.1), (4.1 → 4.4) , (4.3.1 → 4.3.2), (4.2.1 → 4.2.2), (4.1 → 4.3,4.2), (4.6 →4.5 → 4.6), (4.6 → 4.7 → 4.6).

STEP 8) Determine Risk

- **Denial of service**
  Denial of service attack is an important threat to Open flow network; It is a trivial problem harassing the Networking industry since a long time. DOS threat in our implementation of OpenFlow network will be
  - **a)** Host h1 or h2 is a botnet attack the network by DOS
  - **b)** S1 or s2 is configured incorrectly which leads to ARP broadcast attack
  - **c)** DDOS on DNS server
  - **d)** dpctl user space server is hacked to insert wrong path in controller, which leads to DoS attack on the entire network (due to congestion)
  - **e)** DDoS on webserver via reflected attack
- **Arp Poisoning**
  - **a)** ARP poisoning attack on flow table of the switches is also a major type of attack, bogus entry for IP to Mac conversion can be added in the flow table of switches
  - **b)** Basically Controller contains a centralized forwarding flow table for deciding paths in a network. This is a major threat to the system

- **DNS poisoning**
  - **a)** DNS poisoning is a transport level attack, although the controller can control the flow to DNS server and port for DNS packets
  - **b)** Our inference is that a well-planned attack on controller can hamper DNS entries

STEP 9) Mitigation

a) **Spoofing**, it can be avoided by proper authentication, a strong password should be provided on dpctl user space to avoid anyone other than admin to access controller. The password can be hashed and encrypted
b) **Tampering**: - Tampering of services like MITM (Man in the Middle) attack can be avoided by implementing network encryption and a program for decryption can be configured on controller. In this way the adversary cannot track or manipulate the flow or the flow table respectively
c) **Information Disclosure: -** Information from web server and web clients should be encrypted and proper certificate exchange should happen for verifying the Integrity
d) **Repudiation**: activity like change of flow table or deletion or updating of flow table should be logged and the log should again be encrypted and saved at secure location
e) **DoS**: - Flow should be manipulated if DoS on one of the ports happens, botnets and macs performing dos should be fixed, and special program can be written to identify the Macs
f) **Elevation of Permissions**: - Major accessibility permissions should be given to administrator only

Attacks performed by us:-

Although studying open flow networks in details and implementing it on a home network was our major task, after risk analysis we were tempted to perform attacks on this network. As we had implemented an image of open flow network on a Virtual drive, we did not have any legal problems on attacking it, We have used Back track 5 By definition it is a tool for network penetration testing but I would say it is an operating system for Hackers, we referred online tutorial and performed two types of attack. Man in middle attack by performing ARP poisoning and DOS. Our finding where that those attacks do work. When you try to attack the mac table of switches. But one point to be considered is The forwarding program is important, our thinking is if the forwarding program logic can be strong enough then DOS and ARP broadcast can be prevented, We are not sure how yet. Further reading and research has to be performed.

Suggestions:

Software defined networks are supposed to be future generation of Networking, The idea of writing programs over a controller to control the network and control problems like congestion and DOS is quite novel. One ground people suggest that the networking will be quite developed and the admin will have almost full control over the open network. Although one can insert some virus into the controller which can delete or manipulate the program written for forwarding. One thing to be noted is a centralized controller will make the job of administrating simple but certain aspects like network security still need to be studied further

and deep analysis need to be performed to secure the open flow networks before full deployment over the globe.

Existing techniques similar to Open flow/Implementation of Open flow:

DIFANE **:** rule partitioning used for controller less flow insertion.

UCSD FAT TREE Series:- They scale out data centers which use open flow.

Tesseract: It is a centralized wan.

ONIX :-it is a fault tolerant controller platform like NEC.

Conclusion:

After studying and implementing an open flow network we conclude that configuring open flow networks is easy, debugging is easier than that. Writing a simple program for a switch to learn the paths takes less than 10 minutes. It is an awesome idea for implementation and the problems of ARP poisoning and DOS which we performed are trivial and those attacks were successful because of Particularly a controller was used is incorrect. Proper configuration of firewall and detecting software to detect that problem can be performed.

After Performing Risk analysis we conclude that, If a proper firewall and encryption as well as other good security practices are performed while setting up a network, it will be difficult to penetrate.

After performing attack we conclude that backtrack 5 is an awesome software which I and my partner will be using more frequently to penetrate Open flow and other networks which I configure in future.

Reference:

http://www.openflow.org/wk/index.php/OpenFlow_Tutorial

http://sbrc2010.inf.ufrgs.br/anais/data/pdf/wpeif/st02_05_wpeif.pdf

http://www.bladenetwork.net/userfiles/file/OpenFlow-WP.pdf

Figure (1.0), (2.0), (2.1), (2.2) are taken from Reference number 1.

Note: - Our finding and our conclusions are objective to our opinion which is formed after reading research papers and Implementation of Open flow network.