

Name Singh, Avantika Username: \_\_\_\_\_ Date: 12/21/2020  
(last name, first name)  
Section: \_\_\_\_\_

**Assignment 8: Final Project**

\_\_\_\_\_ Part 3 functionality [Max 70 points]  
\_\_\_\_\_ Part 4 functionality [Max 25 points]  
\_\_\_\_\_ Style [Max 5 points]  
\_\_\_\_\_ Part 3 extra credit questions [Max 20 points (10 each)]  
  
\_\_\_\_\_ **Total [Max 100 points + Extra Credit points]**

**Total in points:** \_\_\_\_\_

**Total in extra credit points:** \_\_\_\_\_

**Professor's Comments:**

**Affirmation of my Independent Effort:** Avantika Singh  
(Sign here)

## **ACKNOWLEDGEMENT**

This project couldn't be completed without the advice and support of Professor Franchitti. I would also like to extend my gratitude to the TA Joanna for her guidance during the project and helping us whenever we got stuck on this project.

I am also grateful to University of Wisconsin and at Brown University who developed the part of code and instructions for this project.

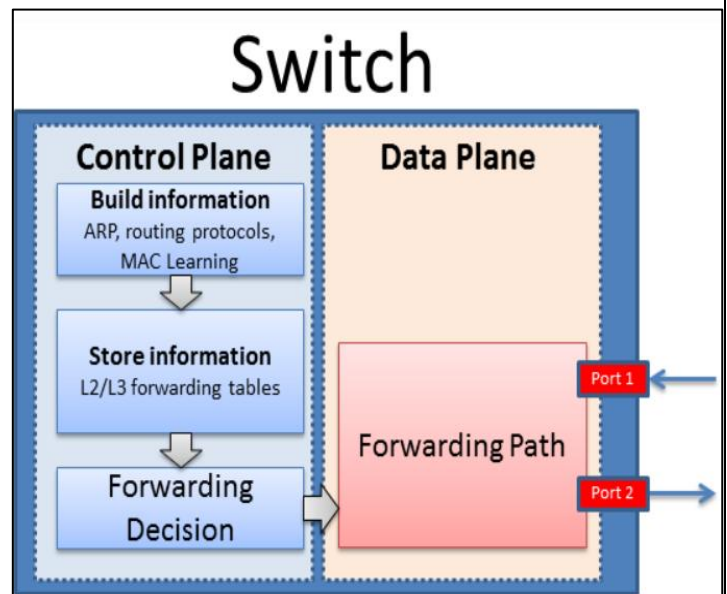
Thanking You,  
Avantika Singh

## Background:

Traditional networks use both a distributed data plane and a distributed control plane. In other words, each device has a data plane and a control plane, and the network distributes those functions into each individual device. Traditional networking is rooted in fixed-function network devices, such as a switch or router which are somewhat autonomous in their working. These devices each have certain functions that operate well together and support the network. For example : The switches/routers when connected to any topology in a computer network exchange certain messages which enables them to pass the information about their neighbors and the resulting topology to other nodes. Thus, every node has some idea (or complete picture of topology in case of link state algorithms) of the topology in which they are located.

Each router consists of two functioning planes: **Control Plane** and **Data Plane**.

1. **Control Plane:** Control plane contains the protocols and mechanisms that enable routers to efficiently learn how to forward packets towards their destination. It is used to determine how a packet is routed among routers along the path from source to destination. It is the part of router architecture that is concerned with drawing the network topological information using the information in routing tables that defines what to do with the incoming packets. Routing protocols specify how routers gather information for the routing tables and talk to their neighboring routers, disseminating information that enables them to select any two nodes on a computer network.



2. **Data Plane:** Data plane contains the protocols and mechanisms that allow hosts and routers to exchange packets carrying user data. Data plane is used for forwarding packets from one interface to another via forwarding table and decides what to do with packets arriving on an inbound interface. Mostly, it refers to a table in which the router looks up the destination address of the incoming packet and retrieves the information necessary to determine the path from the receiving element, through the internal forwarding fabric of the router, and to the proper outgoing interface(s).

These two parts work in combination to define the complete working of a router.

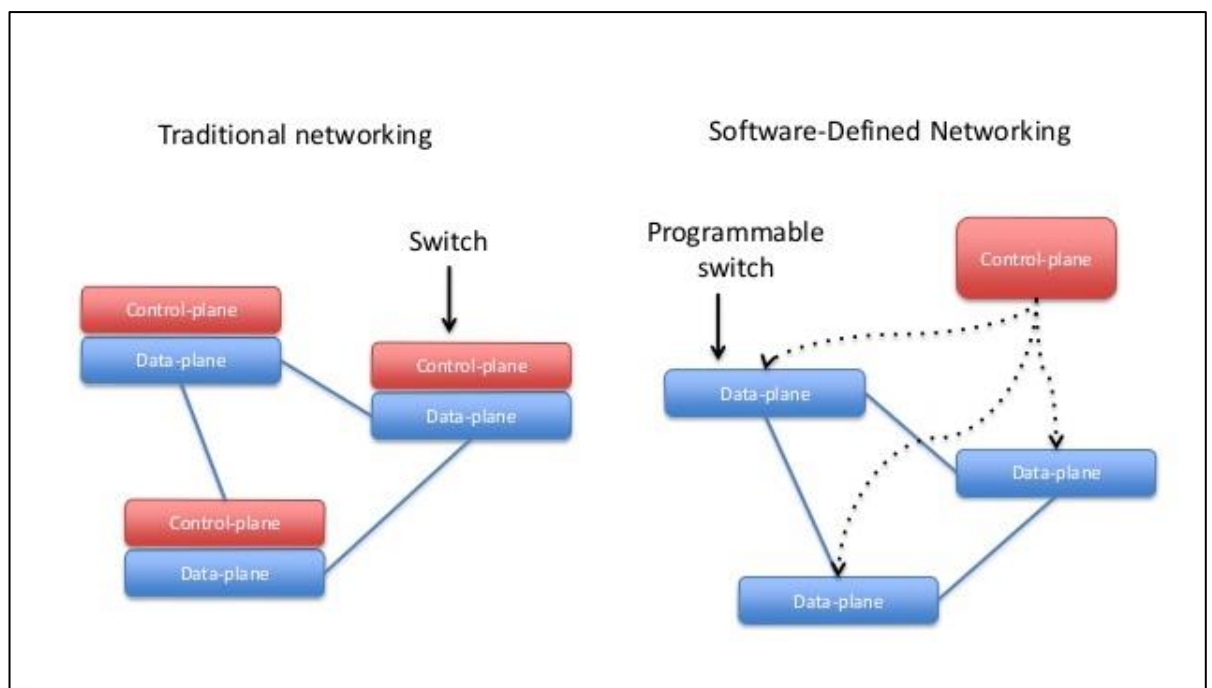
But with such complexity, this approach draws certain drawbacks which are as follows:

- Flexibility is a recurring hurdle for traditional networks. Few APIs are exposed for provisioning and most switching hardware and software is proprietary. Traditional

networks often work well with proprietary provisioning software, but this software can't be quickly modified as needed.

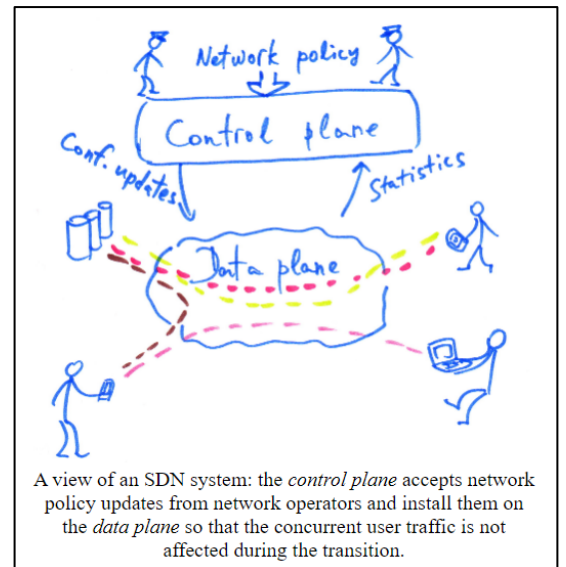
- Since the routers need to setup the topology when they are connected (determine neighboring nodes, spanning tree construction etc.), they need significant processing capability to do this which in return results in expensive routers/switches.
- With traditional networking significant amount of setup time is required to identify loop free topology and the neighboring nodes.
- With a traditional network the physical location of the control plane hinders an IT administrator's ability to control the traffic flow.

It is with these shortcomings; concept of Software Defined Networks was introduced.



## Introduction:

Software-Defined Networking(SDN) is a relatively new network architecture in which the control plane is separated from each individual network device and instead implemented in an external software entity. The external entity has complete knowledge of the topology of a network under its control, and programs the forwarding tables of each individual device in the network. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. Software-Defined Networking is an emerging architecture that is manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications.



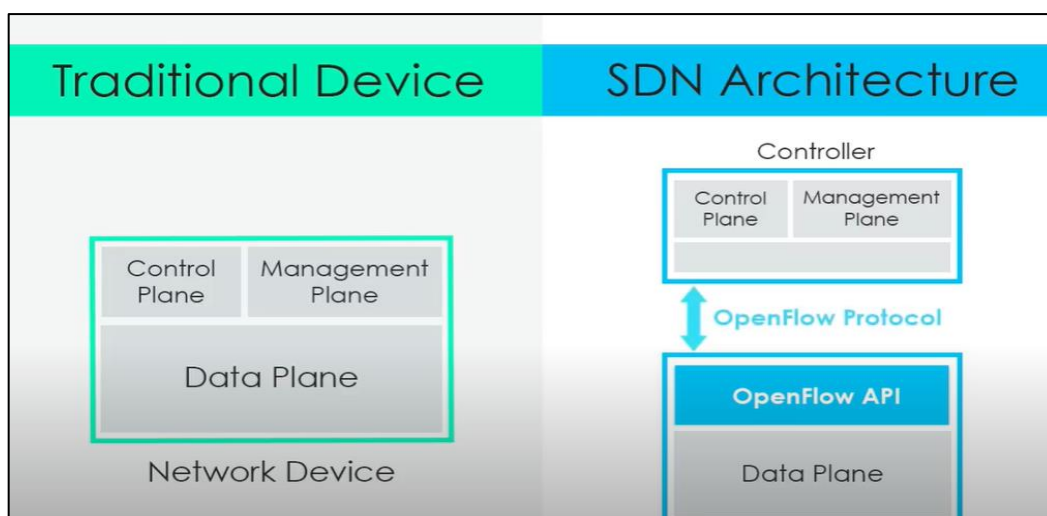
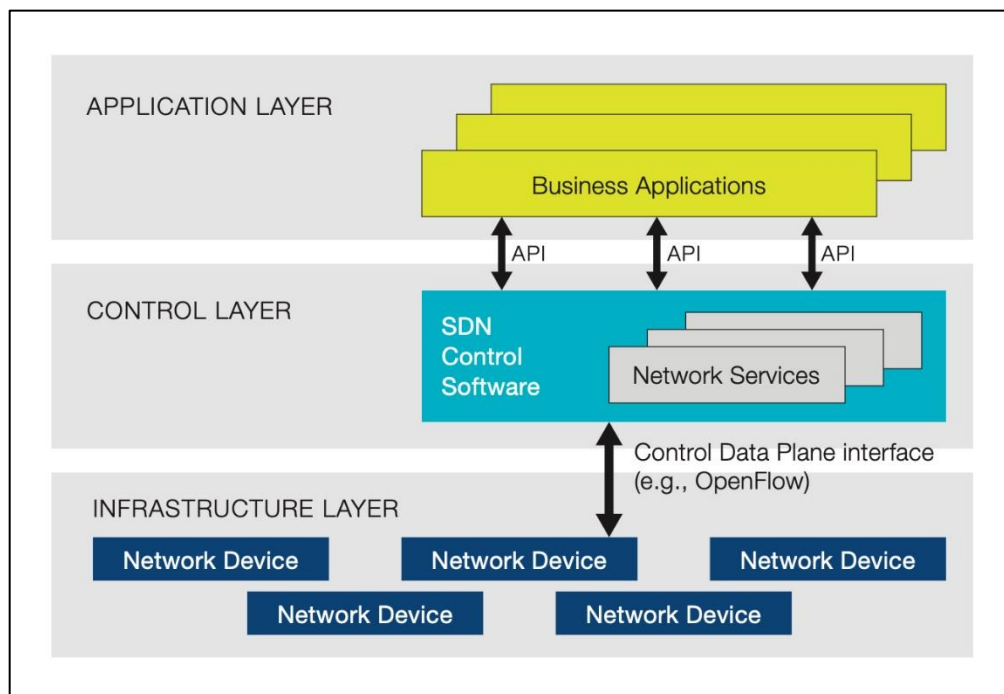
### Distinct properties of SDN Architecture:

1. **Control and data plane separation:** Removing the control plane from network devices and implementing it in an external SDN controller significantly reduces the complexity of network devices, making them simpler and cheaper than CN devices whose distributed control plane functionality is implemented across millions of lines of code, defined across hundreds of RFCs.
2. **Network programmability:** An SDN controller, with complete knowledge of a network's topology, controls a multitude of network devices within its administrative domain. By providing application programming interfaces (APIs), SDN makes it possible to develop networking applications, e.g., traffic engineering, thus enabling network innovation. In contrast, traditional networking devices are proprietary and closed, making it hard or impossible to develop innovative network applications

### Advantages of SDN:

- Control plane and Data plane technologies can evolve individually without one restricting the growth of other.
- It is possible to centralize the control of the network which offers many advantages including better support for traffic engineering, maintaining a consistent network-wide policy implementation, security.
- Virtualization epitomizes the primary difference between SDN and traditional networking. When SDN virtualizes your entire network, it generates an abstract copy of your physical network, and lets you provision resources from a centralized location.
- SDN lets users use software to provision new devices instead of using physical infrastructure, so IT administrators can direct network paths and proactively arrange network services. Unlike traditional switches, SDN can communicate better with devices using the network.

- The use of SDN helps revive older network devices and simplifies the process of optimizing commoditized hardware. By following the instructions from the SDN controller, older hardware can be repurposed as this process allows new devices to become “white box” switches that have intelligence focused at the SDN controller.
- Centralized network provisioning. SDN’s helps centralize enterprise management and provisioning by offering a unified perspective on the whole network. SDN can also speed up service delivery and boost agility in provisioning virtual and physical network devices in a central location.



## Project Overview:

SDN switches and routers do not run control plane protocols and mostly only forward packets based on matching of packet predicates to a set of forwarding rules. They export a simple API to configure these rules, as well as some feedback about current and past packets. The currently accepted standard for this API is the **OpenFlow protocol**, which has been implemented by dozens of switch vendors and has fostered a rich software ecosystem. The intelligence of the control plane is (logically) centralized in a **network controller**. The controller decides which rules to install based on its configuration, and on a global view of the network topology and flows.

For this project we will implement two control application for a software defined network (SDN).

- A layer-3 routing application will install rules in SDN switches to forward traffic to hosts using the shortest, valid path through the network. The application logic will manage the efficient switching of packets among hosts in a large LAN with multiple switches and potential loops. We create an SDN controller application that will compute and install shortest path routes among all the hosts in your network.
- A distributed load balancer application will redirect new TCP connections to hosts in a round-robin order.

## Implementation:

### 1. Layer-3 routing application

- The code for the layer-3 routing application resides in `L3Routing.java` in the `edu.wisc.cs.sdn.apps.l3routing` package.
- Bellman-Ford algorithm was used to compute the shortest paths to reach a host  $h$  from every other host  $h' \in H$ ,  $h \neq h'$  ( $H$  is the set of all hosts).
- There are two link objects between pairs of switches, one in each direction. Due to the way links are discovered, there may be a short period of time (tens of milliseconds) where the controller has a link object only in one direction.
- When a host joins the network, both the `deviceAdded(...)` and `linkDiscoveryUpdate(...)` event handlers will be called. There are no guarantees on which order these event handlers are called. Thus, a host may be added but we may not yet know which switch it is linked to.
- The `isAttachedToSwitch()` method in the `Host` class will return true if we know the switch to which a host is connected, otherwise it will return false.
- The following is assumed to hold true in the network:

The network is a connected graph. In other words, there will always be at least one possible path between every pair of switches.

There is only one physical link between a pair of switches. Links are undirected.

```

1  function BellmanFord(list vertices, list edges, vertex source)
2      ::distance[],predecessor[]
3
4      // This implementation takes a graph as input
5      // A graph is represented as a list of vertices and edges
6      // The algorithm fills two arrays
7      // (distance and predecessor) with shortest-path
8      // (less cost/distance/metric) information
9
10     // Step 1: initialize graph
11     for each vertex v in vertices:
12         distance[v] := inf          // At the beginning: All vertices have a weight of infinity
13         predecessor[v] := null      // And a null predecessor
14
15     distance[source] := 0           // Initial distance for the source is NULL
16
17     // Step 2: Relax edges repeatedly
18     for i from 1 to size(vertices)-1:
19         for each edge (u, v) with weight w in edges:
20             if distance[u] + w < distance[v]:
21                 distance[v] := distance[u] + w
22                 predecessor[v] := u
23
24     // Step 3: Check for negative-weight cycles
25     for each edge (u, v) with weight w in edges:
26         if distance[u] + w < distance[v]:
27             error "Graph contains a negative-weight cycle"
28     return distance[], predecessor[]

```

The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.

Bellman–Ford is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. We implement Bellman-Ford using a queue to define which edges to explore next

## 2. Load balancer:

Networks employ load balancing to distribute client requests among a collection of hosts running a specific service (e.g., a web server). A hardware load balancer is placed in the network and configured with an IP address and a set of hosts among which it should distribute requests. Clients wanting to communicate with a service (e.g., a web server) running on those hosts are provided with the IP address of the load balancer, not the IP address of a specific host. Clients initiate a TCP connection to the IP address of the load balancer (10.0.100.1) and the TCP port associated with the service. For each new TCP connection, the load balancer selects one of the specified hosts (usually in round robin order). The load balancer maintains a mapping of active connections—identified by the client’s IP and TCP port—to the assigned hosts.

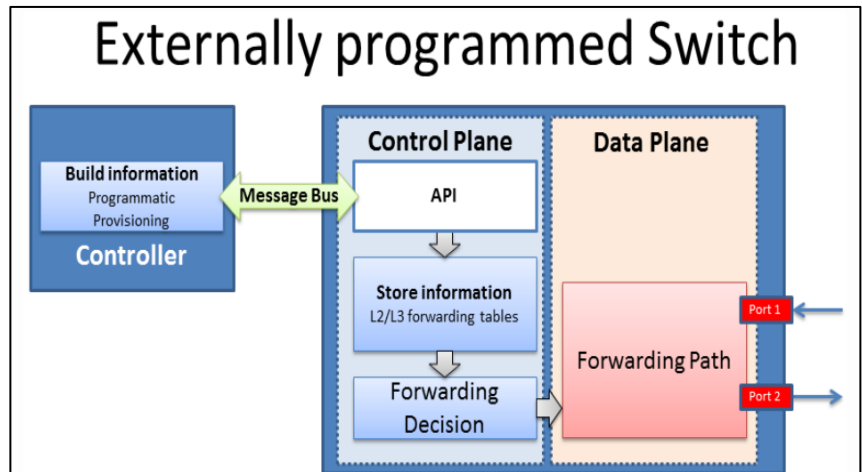
- The code for the load balancer application resides in LoadBalancer.java in the edu.wisc.cs.sdn.apps.loadbalancer package.
- The LoadBalancerInstance class represents a single distributed load balancer.
- Each load balancer instance has a virtual IP address, virtual MAC address, and set of hosts among which TCP connections should be distributed.
- The instances class variable in the LoadBalancer class maps a virtual IP address to a specific load balancer instance.



## Execution Environment:

We run the code for the project in an emulated network inside of a single Linux VM. We use the Mininet network emulator, which is designed to emulate arbitrary topologies of emulated OpenFlow switches and Linux hosts. It uses container-based virtualization for very light-weight emulated nodes.

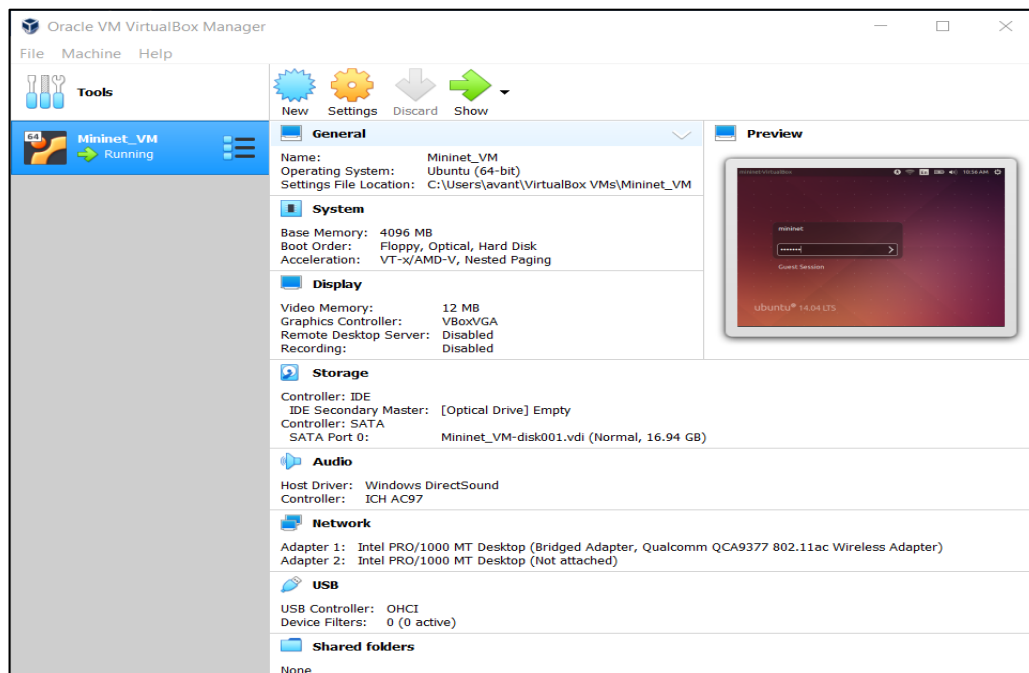
The switches in our network run the open source OpenvSwitch switch software, which implements the Openflow protocol. The switches connect to an Openflow network controller. We will use Floodlight, a relatively mature Java-based controller. We will use OpenFlow version 1.0 for this project. The SDN applications are written in Java and run atop the Floodlight OpenFlow controller.

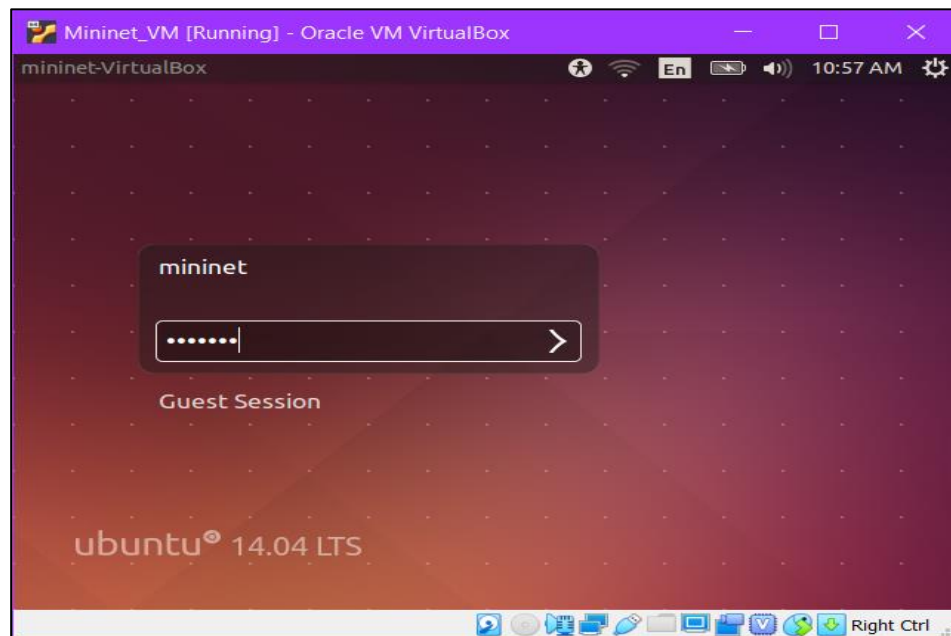


## Environment Setup :

The steps followed have been listed below:

1. Install the Oracle VirtualBox and download the Virtual Box Image with the necessary software required for development. To install the .ova file go to File and Import Appliance on VirtualBox. This VM uses “mininet” as username and password.





2. To ssh into the VM from your host computer we first log in first using the GUI, open a terminal, and type ifconfig. This shows us the IP addresses of the VM. We can now connect to the virtual machine from our host computer via ssh.

```
mininet@mininet-VirtualBox: ~  
mininet@mininet-VirtualBox:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 08:00:27:a7:ce:2f  
          inet addr:192.168.0.109  Bcast:192.168.0.255  Mask:255.255.255.0  
          inet6 addr: fe80::a00:27ff:fea7:ce2f/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:57 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:12892 (12.8 KB)  TX bytes:12523 (12.5 KB)  
  
eth1      Link encap:Ethernet  HWaddr 08:00:27:d7:fb:f8  
          inet6 addr: fe80::a00:27ff:fed7:fbf8/64 Scope:Link  
          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  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  Metric:1  
          RX packets:78 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:78 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:6882 (6.8 KB)  TX bytes:6882 (6.8 KB)  
  
mininet@mininet-VirtualBox:~$
```

We can now connect to the virtual machine from our host computer via ssh.  
**ssh mininet@192.168.0.109**

```

PS C:\WINDOWS\system32> ssh mininet@192.168.0.109
mininet@192.168.0.109's password:
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

765 packages can be updated.
519 updates are security updates.

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Dec 21 04:42:25 2020 from 192.168.0.110
mininet@mininet-VirtualBox:~$

```

3. We will additionally be using Putty and WinSCP to connect to the Virtual machine and transfer files.

4. **Refactor the code to resolve the any module loading errors.**

In order to resolve any issues during execution we need to refactor the code:

- Navigate to the openFlow directory and change the directory path from `src/brown/cs/sdn/apps/` to `/src/wics/cs/sdn/apps/`
- For all files in sub-directories of `/src/wics/cs/sdn/apps/` change the package names to `edu.wisc.cs.sdn.apps.`
- Navigate to the floodlight-plus directory in `/src/main/resources/META-INF/services/` and open the file called `net.floodlightcontroller.core.module.IFloodlightModule` and change the following from: `edu.brown.cs.sdn.apps.sps.ShortestPathSwitching` and `edu.brown.cs.sdn.apps.util.ArpsServer` to: `edu.wisc.cs.sdn.apps.sps.ShortestPathSwitching` and `edu.wisc.cs.sdn.apps.util.ArpsServer`
- Add `edu.wisc.cs.sdn.apps.l3routing.L3Routing` to the `net.floodlightcontroller.core.module.IFloodlightModule`

## Steps for Execution:

1. Navigate to the floodlight-plus directory and run "ant" to compile the code. The `floodlight.jar` file should get generated and placed inside the "target" directory.

```

mininet@mininet-VirtualBox: ~/floodlight-plus
mininet@mininet-VirtualBox:~$ cd floodlight-plus
mininet@mininet-VirtualBox:~/floodlight-plus$ ant
Buildfile: /home/mininet/floodlight-plus/build.xml

init:

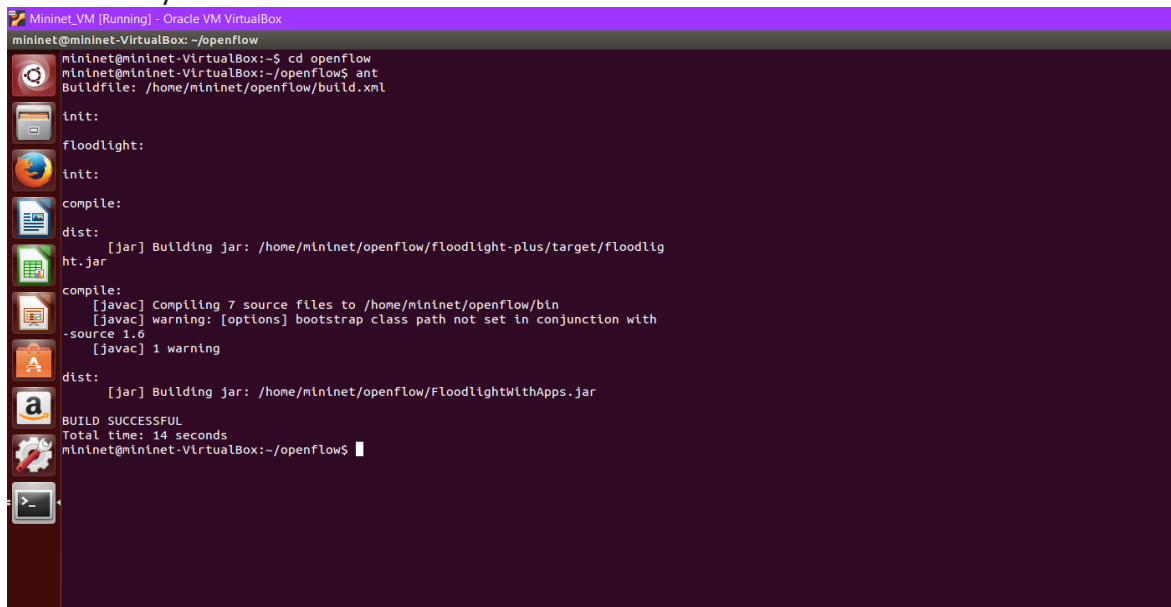
compile:
[copy] Copying 1 file to /home/mininet/floodlight-plus/target/bin

dist:
[jar] Building jar: /home/mininet/floodlight-plus/target/floodlight.jar

BUILD SUCCESSFUL
Total time: 11 seconds
mininet@mininet-VirtualBox:~/floodlight-plus$

```

2. Navigate to the openflow directory and open the build.xml file. Additionally, make sure the relative paths to the floodlight-plus/target directory are properly configured to the correct relative paths.
3. Run "ant" in the root level of the openflow directory. In the bin directory, you should see the shortestPathSwitching.class file with the new package and directory structure. The FloodlightWithApps.jar file gets generated in this openflow root directory.

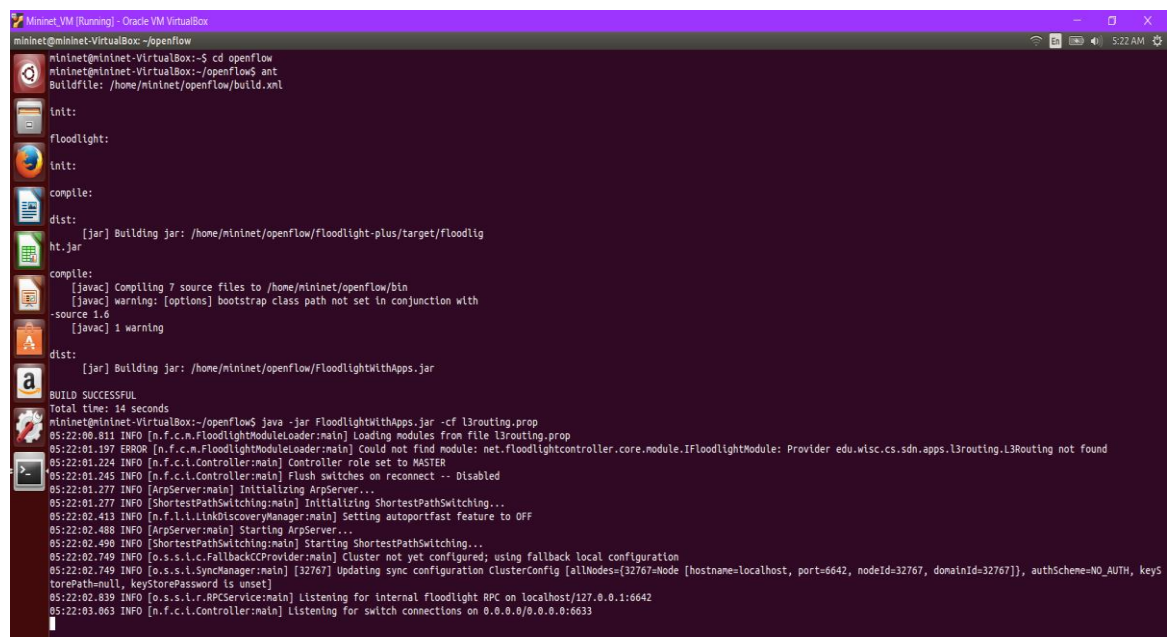


```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox: ~/openflow
mininet@mininet-VirtualBox:~$ cd openflow
mininet@mininet-VirtualBox:~/openflow$ ant
Buildfile: /home/mininet/openflow/build.xml

init:
floodlight:
init:
compile:
dist: [jar] Building jar: /home/mininet/openflow/floodlight-plus/target/floodlig
ht.jar
compile: [javac] Compiling 7 source files to /home/mininet/openflow/bin
[javac] warning: [options] bootstrap class path not set in conjunction with
-source 1.6
[javac] 1 warning
dist: [jar] Building jar: /home/mininet/openflow/FloodlightWithApps.jar

BUILD SUCCESSFUL
Total time: 14 seconds
mininet@mininet-VirtualBox:~/openflow$
```

4. Start Floodlight and your SDN applications. Run the program with the command: **java -jar FloodlightWithApps.jar -cf l3routing.prop**  
The above command will start Floodlight and only our layer-3 routing application.  
The .prop file configures our application.  
Output we get once Floodlight starts :



```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox: ~/openflow
mininet@mininet-VirtualBox:~$ cd openflow
mininet@mininet-VirtualBox:~/openflow$ ant
Buildfile: /home/mininet/openflow/build.xml

init:
floodlight:
init:
compile:
dist: [jar] Building jar: /home/mininet/openflow/floodlight-plus/target/floodlig
ht.jar
compile: [javac] Compiling 7 source files to /home/mininet/openflow/bin
[javac] warning: [options] bootstrap class path not set in conjunction with
-source 1.6
[javac] 1 warning
dist: [jar] Building jar: /home/mininet/openflow/FloodlightWithApps.jar

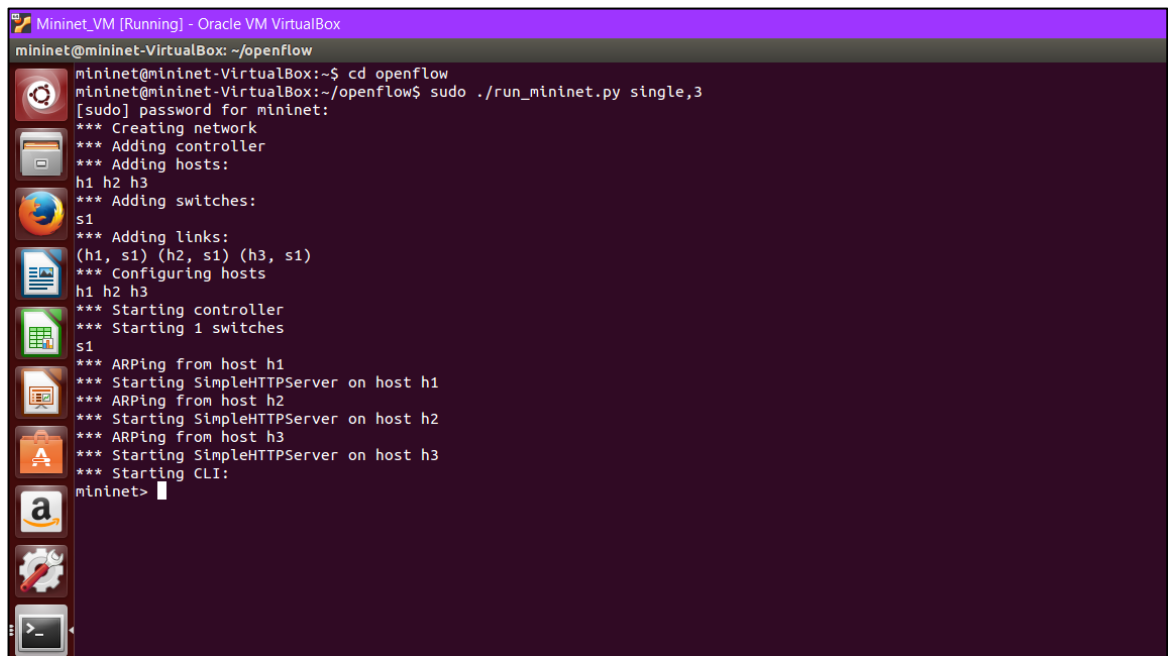
BUILD SUCCESSFUL
Total time: 14 seconds
mininet@mininet-VirtualBox:~/openflow$ java -jar FloodlightWithApps.jar -cf l3routing.prop
05:22:00.811 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from file l3routing.prop
05:22:01.197 ERROR [n.f.c.m.FloodlightModuleLoader:main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule: Provider edu.wisc.cs.sdn.apps.l3routing.L3Routing not found
05:22:01.224 INFO [n.f.c.i.Controller:main] Controller role set to MASTER
05:22:01.245 INFO [n.f.c.i.Controller:main] Flush switches on reconnect -- Disabled
05:22:01.277 INFO [ArpServer:main] Initializing ArpServer...
05:22:01.277 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
05:22:02.413 INFO [n.f.l.l.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
05:22:02.488 INFO [ArpServer:main] Starting ArpServer...
05:22:02.490 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
05:22:02.749 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured; using fallback local configuration
05:22:02.749 INFO [o.s.s.i.c.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes={32767=Node [hostname=localhost, port=6642, nodeId=32767, domainId=32767]}, authScheme=NO_AUTH, keys=
torrePath=null, keyStorePassword=to unset]
05:22:02.839 INFO [o.s.s.i.c.RPCService:main] Listening for internal Floodlight RPC on localhost/127.0.0.1:6642
05:22:03.063 INFO [n.f.c.i.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
```

Keep the terminal with Floodlight open, as we will need to see the output for debugging.

5. Start Mininet: Mininet is a system for rapidly prototyping large networks on the constrained resources of a single laptop. The lightweight approach of using OS-level virtualization features, including processes and network namespaces, allows it to scale to hundreds of nodes.

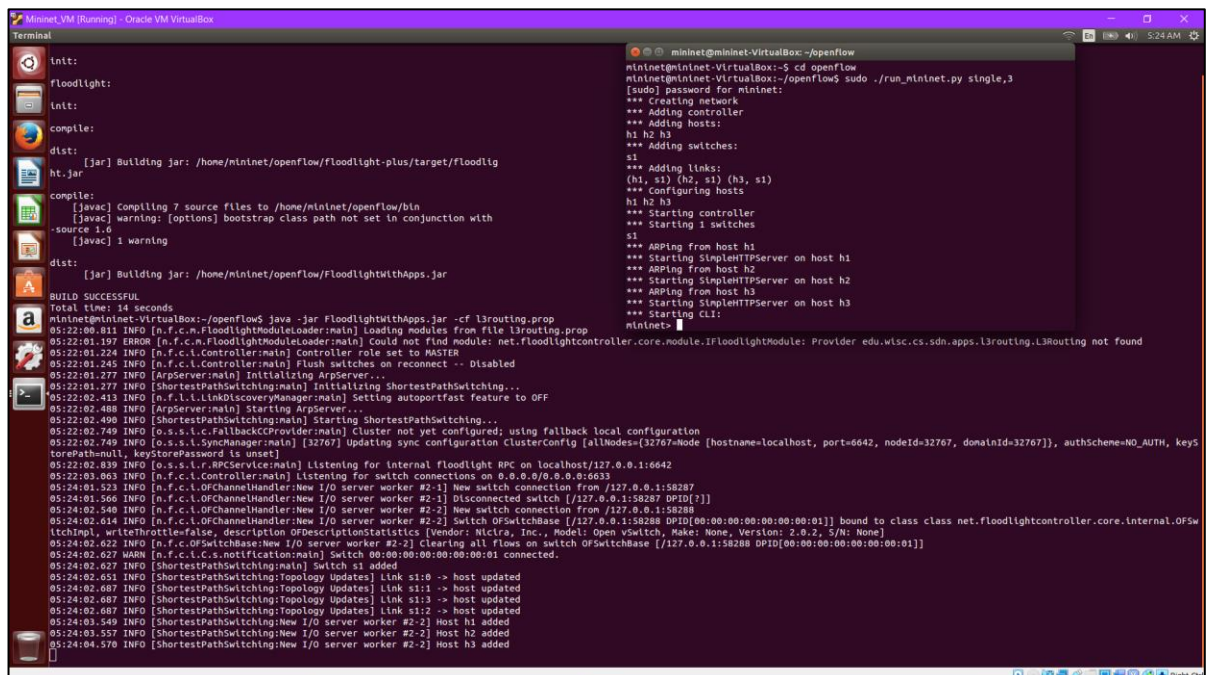
**\$ sudo ./run\_mininet.py single,3**

The above command will create a topology with a single SDN switch (s1) and three hosts (h1 - h3) directly connected to the switch.



```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox: ~/openflow
mininet@mininet-VirtualBox:~$ cd openflow
mininet@mininet-VirtualBox:~/openflow$ sudo ./run_mininet.py single,3
[sudo] password for mininet:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** ARPing from host h1
*** Starting SimpleHTTPServer on host h1
*** ARPing from host h2
*** Starting SimpleHTTPServer on host h2
*** ARPing from host h3
*** Starting SimpleHTTPServer on host h3
*** Starting CLI:
mininet>
```

We can see Floodlight produce the following output :



```
Mininet_VM [Running] - Oracle VM VirtualBox
Terminal
Floodlight:
Init:
Floodlight:
Init:
compile:
dist: [jar] Building jar: /home/mininet/openflow/Floodlight-plus/target/Floodli
ht.jar
compile:
[java] Compiling 7 source files to /home/mininet/openflow/bin
[javac] warning: [options] bootstrap class path not set in conjunction with
-source 1.6
[javac] 1 warning
dist:
[jar] Building jar: /home/mininet/openflow/FloodlightwithApps.jar
BUILD SUCCESSFUL
Total time: 14 seconds
mininet@mininet-VirtualBox:~/openflow$ java -jar FloodlightwithApps.jar -cf l3routing.prop
05:22:00.811 INFO [n.f.c.n.FloodlightModuleLoader:main] Loading modules from file l3routing.prop
05:22:01.197 ERROR [n.f.c.n.FloodlightModuleLoader:main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule: Provider edu.wisc.cs.sdn.apps.l3routing.L3Routing not found
05:22:01.224 INFO [n.f.c.l.Controller:main] Controller role set to MASTER
05:22:01.245 INFO [n.f.c.l.Controller:main] Flush switches on reconnect -- Disabled
05:22:01.277 INFO [ArpServer:main] Initializing ArpServer...
05:22:01.277 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
05:22:02.412 INFO [n.f.l.LinkDiscoveryManager:main] Setting autopoportfast feature to OFF
05:22:02.488 INFO [ArpServer:main] Starting ArpServer...
05:22:02.490 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
05:22:02.749 INFO [o.s.s.l.c.FallbackController:main] Cluster not yet configured; using fallback local configuration
05:22:02.749 INFO [o.s.s.l.c.SynchManager:main] [32767] Updating sync configuration ClusterConfig [allNodes={32767=Node [hostname=localhost, port=6642, nodeId=32767, domainId=32767]}, authScheme=NO_AUTH, keys=torePath=null, keyStorePassword is unset]
05:22:03.063 INFO [o.s.s.l.c.RPCService:main] Listening for internal floodlight RPC on localhost/127.0.0.1:6642
05:24:01.523 INFO [n.f.c.l.OfChannelHandler:New I/O server worker #2-1] New switch connection from /127.0.0.1:58287
05:24:01.566 INFO [n.f.c.l.OfChannelHandler:New I/O server worker #2-1] Disconnected switch [/127.0.0.1:58287 DPID[?]]
05:24:02.546 INFO [n.f.c.l.OfChannelHandler:New I/O server worker #2-2] New switch connection from /127.0.0.1:58288
05:24:02.614 INFO [n.f.c.l.OfChannelHandler:New I/O server worker #2-2] Switch OFSwitchBase [/127.0.0.1:58288 DPID[00:00:00:00:00:00:01]] bound to class class net.floodlightcontroller.core.internal.OFSwitchImpl, writeThrottle=false, descriptionOFDescriptionStatistics [Vendor: Nictira, Inc., Model: Open vSwitch, Make: None, Version: 2.0.2, S/N: None]
05:24:02.627 WARN [n.f.c.l.c.s.notification:main] Switch 00:00:00:00:00:00:01 connected.
05:24:02.627 INFO [ShortestPathSwitching:main] Switch s1 added
05:24:02.651 INFO [ShortestPathSwitching:Topology Updates] Link s1:0 -> host updated
05:24:02.687 INFO [ShortestPathSwitching:Topology Updates] Link s1:1 -> host updated
05:24:02.687 INFO [ShortestPathSwitching:Topology Updates] Link s1:3 -> host updated
05:24:02.687 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
05:24:03.546 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h1 added
05:24:03.557 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h2 added
05:24:04.570 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h3 added
```



Debugging: Now for debugging we run commands (e.g., ping) in Mininet.

```
mininet@mininet-VirtualBox: ~/openflow
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** ARPing from host h1
*** Starting SimpleHTTPServer on host h1
*** ARPing from host h2
*** Starting SimpleHTTPServer on host h2
*** ARPing from host h3
*** Starting SimpleHTTPServer on host h3
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

```
mininet@mininet-VirtualBox: ~/openflow
mininet@mininet-VirtualBox:~/openflow$ java -jar FloodlightWithApps.jar -cf l3routing.prop
12:18:09.655 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from file l3routing.pr
12:18:10.271 ERROR [n.f.c.m.FloodlightModuleLoader:main] Could not find module: net.floodlight
controller.core.module.FloodlightModule: Provider edu.wisc.cs.sdn.apps.l3routing.L3Routing no
t found
12:18:10.312 INFO [n.f.c.l.Controller:main] Controller role set to MASTER
12:18:10.328 INFO [n.f.c.l.Controller:main] Flush switches on reconnect -- Disabled
12:18:10.353 INFO [ArpServer:main] Initializing ArpServer...
12:18:10.350 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
12:18:11.700 INFO [n.f.l.l.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
12:18:11.793 INFO [ArpServer:main] Starting ArpServer...
12:18:11.794 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
12:18:12.131 INFO [o.s.s.l.c.FallbackCPCProvider:main] Cluster not yet configured; using fallback
local configuration
12:18:12.131 INFO [o.s.s.l.c.SyncManager:main] [32767] Updating sync configuration ClusterConfig
[allNodes=[32767+Node [hostname=localhost, port=6642, nodeId=32767, domainId=32767]], authc
eme=NO_AUTH, keyStorePath=null, keyStorePassword is unset]
12:18:12.260 INFO [o.s.s.l.c.RPCService:main] Listening for internal floodlight RPC on localh
st/127.0.0.1:6642
12:18:12.544 INFO [n.f.c.l.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0
.0:6633
12:18:38.669 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] New switch connection from /127.0.0.1:41390
12:18:38.707 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] Disconnected switch [/127.0.0.1:41390 DPID[?]]
12:18:39.847 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-2] New switch connection from /127.0.0.1:41391
12:18:39.997 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-2] Switch OFSwitchBase [/127.0.0.1:41391 DPID[00:00:00:00:00:01]] bound to class class net.floodlightcontroller.core.internal.OFSw
itchImpl, writeThrottle=false, description OFDescriptionStatistics [vendor: NtcIra, Inc., Model: Open vSwitch, Make: None, Version: 2.0.2, S/N: None]
12:18:40.003 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-2] Clearing all flows on switch OFSwitchBase [/127.0.0.1:41391 DPID[00:00:00:00:00:01]]
12:18:40.011 WARN [n.f.c.l.Cs.notification:main] Switch 00:00:00:00:00:01 connected.
12:18:40.011 INFO [ShortestPathSwitching:main] Switch s1 added
12:18:40.047 INFO [ShortestPathSwitching:Topology Updates] Link s1:1 -> host updated
12:18:40.048 INFO [ShortestPathSwitching:Topology Updates] Link s1:3 -> host updated
12:18:40.048 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
12:18:40.050 INFO [ShortestPathSwitching:Topology Updates] Link s1:0 -> host updated
12:18:40.895 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h1 added
12:18:40.940 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h2 added
12:18:41.958 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h3 added
12:19:10.635 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.2 from 00:00:00:00:00:01
12:19:10.636 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.2->00:00:00:00:00:02
12:19:10.637 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1 from 00:00:00:00:00:02
12:19:10.638 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.1->00:00:00:00:00:01
12:19:10.651 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.3 from 00:00:00:00:00:01
12:19:10.652 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.3->00:00:00:00:00:03
12:19:10.656 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1 from 00:00:00:00:00:03
12:19:10.657 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.1->00:00:00:00:00:01
12:19:10.660 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.3 from 00:00:00:00:00:02
12:19:10.670 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.3->00:00:00:00:00:03
12:19:10.671 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.2 from 00:00:00:00:00:03
12:19:10.672 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.2->00:00:00:00:00:02
```

6. Run the command to execute the Layer 3 Routing Application.  
**java -jar FloodlightWithApps.jar -cf shortestPathSwitch.prop**

```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox:~$ cd openflow
mininet@mininet-VirtualBox:~/openflow$ java -jar FloodlightWithApps.jar -cf shortestPathSwitching.prop
12:27:10.102 INFO [n.f.c.n.FloodlightModuleLoader:main] Loading modules from file shortestPathSwitching.prop
12:27:10.515 ERROR [n.f.c.n.FloodlightModuleLoader:main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule: Provider edu.wisc.cs.sdn.apps.l3routing.L3Routing not found
12:27:10.535 INFO [n.f.c.l.Controller:main] Controller role set to MASTER
12:27:10.549 INFO [n.f.c.l.Controller:main] Flush switches on reconnect -- Disabled
12:27:10.569 INFO [ArpServer:main] Initializing ArpServer...
12:27:10.575 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
12:27:11.816 INFO [n.f.l.l.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
12:27:11.906 INFO [ArpServer:main] Starting ArpServer...
12:27:11.907 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
12:27:12.190 INFO [o.s.s.l.c.FallbackCProvider:main] Cluster not yet configured; using fallback local configuration
12:27:12.196 INFO [o.s.s.l.c.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes=(32767Node [hostname=localhost, port=6642, nodeId=32767, domainId=32767]), authScheme=NO_AUTH, keys
corePath=null, keyStorePassword is unset]
12:27:12.296 INFO [o.s.s.l.r.RPCService:main] Listening for Internal Floodlight RPC on localhost/127.0.0.1:6642
12:27:12.536 INFO [n.f.c.l.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
12:27:13.126 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] New switch connection from /127.0.0.1:41426
12:27:13.218 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] Switch OFSwitchBase [/127.0.0.1:41426 DPID[00:00:00:00:00:01]] bound to class class net.floodlightcontroller.core.internal.OFSw
itchImpl, writeTimeout=false, descriptionOrDescriptionStatistics [Vendor: Nicira, Inc., Model: Open vSwitch, Make: None, Version: 2.0.2, S/N: None]
12:27:13.215 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-1] Clearing all flows on switch OFSwitchBase [/127.0.0.1:41426 DPID[00:00:00:00:00:01]]
12:27:13.224 WARN [n.f.c.l.C.s.notification:main] Switch 00:00:00:00:00:00:01 connected.
12:27:13.224 INFO [ShortestPathSwitching:main] Switch s1 added
12:27:13.272 INFO [ShortestPathSwitching:Topology Updates] Link s1:0 -> host updated
12:27:13.402 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
12:27:13.402 INFO [ShortestPathSwitching:Topology Updates] Link s1:1 -> host updated
12:27:13.403 INFO [ShortestPathSwitching:Topology Updates] Link s1:3 -> host updated
12:27:13.403 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
```

We see the following output:

```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox:~$ cd openflow
mininet@mininet-VirtualBox:~/openflow$ java -jar FloodlightWithApps.jar -cf shortestPathSwitching.prop
12:34:00.314 INFO [n.f.c.n.FloodlightModuleLoader:main] Loading modules from file shortestPathSwitching.prop
12:34:00.822 ERROR [n.f.c.n.FloodlightModuleLoader:main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule: Provider edu.wisc.cs.sdn.apps.l3routing.L3Routing not found
12:34:00.868 INFO [n.f.c.l.Controller:main] Controller role set to MASTER
12:34:00.894 INFO [n.f.c.l.Controller:main] Flush switches on reconnect -- Disabled
12:34:00.935 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
12:34:02.296 INFO [n.f.l.l.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
12:34:02.385 INFO [ArpServer:main] Starting ArpServer...
12:34:02.386 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
12:34:02.684 INFO [o.s.s.l.c.FallbackCProvider:main] Cluster not yet configured; using fallback local configuration
12:34:02.684 INFO [o.s.s.l.c.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes=(32767Node [hostname=localhost, port=6642, nodeId=32767, domainId=32767]), authScheme=NO_AUTH, keys
corePath=null, keyStorePassword is unset]
12:34:02.761 INFO [o.s.s.l.r.RPCService:main] Listening for Internal Floodlight RPC on localhost/127.0.0.1:6642
12:34:02.966 INFO [n.f.c.l.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
12:34:19.966 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] New switch connection from /127.0.0.1:41427
12:34:20.979 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-1] Disconnected switch [/127.0.0.1:41427 DPID[?]]
12:34:21.117 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-2] New switch connection from /127.0.0.1:41428
12:34:21.230 INFO [n.f.c.l.OFChannelHandler:New I/O server worker #2-2] Switch OFSwitchBase [/127.0.0.1:41428 DPID[00:00:00:00:00:01]] bound to class class net.floodlightcontroller.core.internal.OFSw
itchImpl, writeTimeout=false, descriptionOrDescriptionStatistics [Vendor: Nicira, Inc., Model: Open vSwitch, Make: None, Version: 2.0.2, S/N: None]
12:34:21.238 INFO [n.f.c.OFSwitchBase:New I/O server worker #2-2] Clearing all flows on switch OFSwitchBase [/127.0.0.1:41428 DPID[00:00:00:00:00:01]]
12:34:21.249 WARN [n.f.c.l.C.s.notification:main] Switch 00:00:00:00:00:00:01 connected.
12:34:21.251 INFO [ShortestPathSwitching:main] Switch s1 added
12:34:21.283 INFO [ShortestPathSwitching:Topology Updates] Link s1:0 -> host updated
12:34:21.325 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
12:34:21.325 INFO [ShortestPathSwitching:Topology Updates] Link s1:1 -> host updated
12:34:21.325 INFO [ShortestPathSwitching:Topology Updates] Link s1:3 -> host updated
12:34:21.326 INFO [ShortestPathSwitching:Topology Updates] Link s1:2 -> host updated
12:34:22.167 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h1 added
12:34:22.191 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h2 added
12:34:23.208 INFO [ShortestPathSwitching:New I/O server worker #2-2] Host h3 added
12:34:30.733 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.2 from 00:00:00:00:00:01
12:34:30.734 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.2->00:00:00:00:00:02
12:34:30.735 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1 from 00:00:00:00:00:02
12:34:30.735 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.1->00:00:00:00:00:01
12:34:30.736 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.3 from 00:00:00:00:00:01
12:34:30.736 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.3->00:00:00:00:00:03
12:34:30.739 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1 from 00:00:00:00:00:03
12:34:30.739 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.1->00:00:00:00:00:01
12:34:30.750 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.3 from 00:00:00:00:00:02
12:34:30.761 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.3->00:00:00:00:00:03
12:34:30.767 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.2 from 00:00:00:00:00:03
12:34:30.771 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.2->00:00:00:00:00:02
```

## Debugging :

To debug, we view the contents of an SDN switches flow tables by running the following command in your mininet VM.

```
$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
```

```
Mininet_VM [Running] - Oracle VM VirtualBox
mininet@mininet-VirtualBox:~$ sudo ovs-ofctl -O OpenFlow13 dump-flows s1
[sudo] password for mininet:
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=566.463s, table=0, n_packets=4, n_bytes=392, priority=1,ip,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=565.445s, table=0, n_packets=4, n_bytes=392, priority=1,ip,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=566.476s, table=0, n_packets=4, n_bytes=392, priority=1,ip,nw_dst=10.0.0.1 actions=output:1
mininet@mininet-VirtualBox:~$
```

To debug further we trigger Event Handlers: We trigger the `linkDiscoveryUpdate(...)` event handler by running the following commands in Mininet

- `link s1 h1 down` — takes down the link between `s1` and `h1`; this will also result in a `deviceRemoved(...)` event and the `isAttachedToSwitch()` method for the Host object for `h1` will now return false
- `link s1 h1 up` — brings up the link between `s1` and `h1`; this will also result in a `deviceMoved(...)` event and the `isAttachedToSwitch()` method for the Host object for `h1` will now return true

```
mininet@mininet-VirtualBox: ~/openflow
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** ARPing from host h1
*** Starting SimpleHTTPServer on host h1
*** ARPing from host h2
*** Starting SimpleHTTPServer on host h2
*** ARPing from host h3
*** Starting SimpleHTTPServer on host h3
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> link s1 h1 down
mininet> link s1 h1 up
mininet>
```

We see the following output :

```
mininet>mininet-VirtualBox->cfd openflow
mininet>mininet-VirtualBox->-jopenflows java -jar Floodlightswitchapps.jar -cf shortestPathSwitching.prop
12:40:34.314 INFO [n.f.c.n.FloodlightModuleLoader.main] Loading modules from file shortestPathSwitching.prop
12:40:34.822 ERROR [n.f.c.n.FloodlightModuleLoader.main] Could not find module: net.floodlightcontroller...
12:40:34.868 INFO [n.f.c.i.ControllerMain] Controller role set to MASTER
12:40:34.900 INFO [n.f.c.i.ControllerMain] Load switches on reconnect -- Disabled
12:40:34.900 INFO [ArpserviceImpl] Initializing Arpservice...
12:40:34.930 INFO [ShortestPathSwitchingMain] Initializing ShortestPathSwitching...
12:40:34.936 INFO [ClusterManagerImpl] Setting autopilotFast feature to OFF
12:40:34.985 INFO [ArpserviceImpl] Starting Arpservice...
12:40:34.986 INFO [ShortestPathSwitchingMain] Starting ShortestPathSwitching...
12:40:34.988 INFO [n.f.c.s.fallbackconfigprovider.main] Cluster not yet configured; using fallback local configuration
12:40:34.988 INFO [n.f.c.s.synchronizermain] [32767] Updating sync configuration ClusterConfig [allNodes=[32767:mode {hostname=localhost, port=6642, nodeId=32767, domainId=32767}], authScheme=N/AUTH, keyStorePath=null, keyValuePairs={}]
12:40:34.992 INFO [n.f.c.i.fabricServiceMain] Listening for Internal Floodlight RPC on localhost/[277.0.0.1:6642]
12:40:34.996 INFO [n.f.c.i.controllermain] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
12:40:34.998 INFO [n.f.c.i.OfChannelHandlerNew I/O server worker #2-1] New switch connection from /127.0.0.1:61487
12:40:34.979 INFO [n.f.c.i.OfChannelHandlerNew I/O server worker #2-1] Disconnected switch /127.0.0.1:61487 DPID[]
12:40:34.211 INFO [n.f.c.i.OfChannelHandlerNew I/O server worker #2-2] New switch connection from /127.0.0.1:61428
12:40:34.230 INFO [n.f.c.i.OfChannelHandlerNew I/O server worker #2-2] Switch OFSwitchBase /127.0.0.1:61428 DPID[0x00:00:00:00:00:01], bound to class class net.floodlightcontroller.core.internal.OFSwitchBase
12:40:34.230 INFO [n.f.c.i.OfChannelHandlerNew I/O server worker #2-2] Vendor: Misticra, Inc., Model: Open Switch, Make: None, Version: 2.0.2, S/N: None]
12:40:34.238 INFO [n.f.c.i.OFSwitchBaseNew I/O server worker #2-2] Clearing all flows on switch OFSwitchBase /127.0.0.1:61428 DPID[0x00:00:00:00:00:01]
12:40:34.246 INFO [n.f.c.i.ControllerMain] Configuration successful
12:40:34.251 INFO [ShortestPathSwitchingMain] Switch s1 added
12:40:34.283 INFO [ShortestPathSwitchingTopology Updates] Link s1:0 -> host updated
12:40:34.321 INFO [ShortestPathSwitchingTopology Updates] Link s1:1 -> host updated
12:40:34.325 INFO [ShortestPathSwitchingTopology Updates] Link s1:1 -> host updated
12:40:34.325 INFO [ShortestPathSwitchingTopology Updates] Link s1:3 -> host updated
12:40:34.326 INFO [ShortestPathSwitchingTopology Updates] Link s1:2 -> host updated
12:40:34.222 INFO [ShortestPathSwitchingNew I/O server worker #2-2] Host h1 added
12:40:34.221 INFO [ShortestPathSwitchingNew I/O server worker #2-2] Host h2 added
12:40:34.228 INFO [ShortestPathSwitchingNew I/O server worker #2-2] Host h3 added
12:40:34.733 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.2 from 0x00:00:00:00:00:01
12:40:34.734 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.2-0x00:00:00:00:00:02
12:40:34.735 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.1 from 0x00:00:00:00:00:02
12:40:34.735 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.1-0x00:00:00:00:00:01
12:40:34.736 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.3 from 0x00:00:00:00:00:01
12:40:34.736 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.3-0x00:00:00:00:00:03
12:40:34.739 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.1 from 0x00:00:00:00:00:03
12:40:34.739 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.1-0x00:00:00:00:00:01
12:40:34.750 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.3 from 0x00:00:00:00:00:02
12:40:34.750 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.3-0x00:00:00:00:00:03
12:40:34.761 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.3-0x00:00:00:00:00:03
12:40:34.767 INFO [ArpserviceImpl] New I/O server worker #2-2 Received ARP request for 10.0.0.2 from 0x00:00:00:00:00:03
12:40:34.771 INFO [ArpserviceImpl] New I/O server worker #2-2 Sending ARP reply 10.0.0.2-0x00:00:00:00:00:02
12:40:34.819 INFO [n.f.c.i.c.notificationmain] Switch 0x00:00:00:00:00:01 port s1-eth1 changed: DOWN
12:40:34.819 INFO [ShortestPathSwitchingTopology Updates] Link s1:1 -> host down
12:40:34.824 INFO [n.f.d.i.DeviceScheduled-0] updateAttachmentPoint: att [AttachmentPoint {sws, ports, activeSens=mininet-n link s1 h1 up
12:40:34.825 INFO [ShortestPathSwitchingScheduled-0] Host h1 is no longer attached to a switch
12:40:34.830 WARN [n.f.c.i.c.notificationmain] Switch 0x00:00:00:00:00:01 port s1-eth1 changed: OTHER_UPDATE
12:40:34.854 INFO [ShortestPathSwitchingNew I/O server worker #2-2] Host h1 moved to s1:1
12:40:34.854 INFO [n.f.c.i.c.notificationmain] Switch 0x00:00:00:00:00:01 port s1-eth1 changed: UP
12:40:34.869 INFO [ShortestPathSwitchingTopology Updates] Link s1:1 -> host updated
```



7. Run the command to execute the loadbalancer application.  
**java -jar FloodlightWithApps.jar -cf loadbalancer.prop**

Mininet [Running] - Oracle VM VirtualBox

mininet@mininet-VirtualBox: ~\$ openflow

```
mininet$ mininet-VirtualBox:~$ cd openflow
mininet$ mininet-VirtualBox:~$ java -jar FloodlightHttphs.jar -cf LoadBalancer.prop
66:08:05.712 INFO [n.f.c.n.FloodlightModuleLoader.main] Loading modules from file LoadBalancer.prop
66:08:10.077 ERROR [n.f.c.n.FloodlightModuleLoader.main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule; Provider edu.wisc.cs.sdn.apps.L3routing.L3routing not found
Exception in thread "main" net.floodlightcontroller.core.module.FloodlightModuleException: Module edu.wisc.cs.sdn.apps.LoadBalancer.LoadBalancer not found
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromList(FloodlightModuleLoader.java:178)
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromList(FloodlightModuleLoader.java:162)
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromConfig(FloodlightModuleLoader.java:208)
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.main(Main.java:55)
mininet$ mininet-VirtualBox:~$ openflow java -jar FloodlightHttphs.jar -cf LoadBalancer.prop
66:27:32.349 INFO [n.f.c.n.FloodlightModuleLoader.main] Loading default modules
66:27:32.725 INFO [n.f.c.n.FloodlightModuleLoader.main] Could not find module: net.floodlightcontroller.core.module.IFloodlightModule; Provider edu.wisc.cs.sdn.apps.L3routing.L3routing not found
66:27:32.752 INFO [n.f.c.i.c.Controller.main] Controller role set to MASTER
66:27:32.775 INFO [n.f.c.i.c.Controller.main] Flush switches on reconnect -- Disabled
66:27:32.807 ERROR [o.s.s.c.DelayingCProvider.main] Failed to initialize provider org.sdplatform.sync.internal.config.SyncStoreCProvider
org.sdplatform.sync.error.PersistException: Could not initialize persistent storage
    at org.sdplatform.sync.internal.store.Java8StorageEngine.<init>(Java8StorageEngine.java:106) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.StoreRegistry.register(StoreRegistry.java:119) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.SyncManager.registerPersistentStore(SyncManager.java:184) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.config.SyncStoreCProvider.init(SyncStoreCProvider.java:85) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.config.DelayingCProvider.init(DelayingCProvider.java:17) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.SyncManager.init(SyncManager.java:489) [FloodlightHttphs.jar]
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.loadModulesFromList(FloodlightModuleLoader.java:436) [FloodlightHttphs.jar]
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromList(FloodlightModuleLoader.java:333) [FloodlightHttphs.jar]
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromList(FloodlightModuleLoader.java:362) [FloodlightHttphs.jar]
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.LoadModulesFromConfig(FloodlightModuleLoader.java:209) [FloodlightHttphs.jar]
    at net.floodlightcontroller.core.module.FloodlightModuleLoader.main(Main.java:55) [FloodlightHttphs.jar]
Caused by: java.sql.SQLException: Failed to create database 'var/lib/floodlight/SyncDB', see the next exception for details.
    at org.apache.derby.impl.jdbc.EmbedConnection.getSQLException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedSQLException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection.createDatabase(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection3.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.Driver40.getNewEmbedConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.InternalDriver.connect(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.EmbeddedDataSource.getConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.EmbeddedConnection.openNewConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.EmbeddedConnection.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.Driver40.getNewPoolConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource.createPoolConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.getConnection(Java8StorageEngine.java:368) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.initTable(Java8StorageEngine.java:373) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.<init>(Java8StorageEngine.java:184) [FloodlightHttphs.jar]
    ... 10 common frames omitted
Caused by: org.apache.derby.impl.jdbc.EmbedSQLException: Failed to create database 'var/lib/floodlight/SyncDB', see the next exception for details.
    at org.apache.derby.impl.jdbc.SQLExceptionFactory.getSQLException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.SQLExceptionFactory40.wrapArgsForTransportAcrossDRDA(Unknown Source) [FloodlightHttphs.jar]
    ... 26 common frames omitted
Caused by: org.apache.derby.impl.jdbc.EmbedSQLException: Directory [/var/lib/floodlight/SyncDB] cannot be created.
    at org.apache.derby.impl.jdbc.SQLExceptionFactory.getSQLException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.SQLExceptionFactory40.wrapArgsForTransportAcrossDRDA(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.SQLExceptionFactory40.getSQLException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.jdbc.EmbedConnection3.<init>(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.Driver40.getNewPoolConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource.createPoolConnection(Unknown Source) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.getConnection(Java8StorageEngine.java:368) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.initTable(Java8StorageEngine.java:373) [FloodlightHttphs.jar]
    at org.sdplatform.sync.internal.store.Java8StorageEngine.<init>(Java8StorageEngine.java:184) [FloodlightHttphs.jar]
    ... 10 common frames omitted
Caused by: java.io.IOException: StandardException: Directory [/var/lib/floodlight/SyncDB] cannot be created.
    at org.apache.derby.impl.error.StandardException.newException(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.services.monitor.StorageFactoryService.run(Unknown Source) [FloodlightHttphs.jar]
    at java.security.AccessController.doPrivileged(Native Method) [na:1.7.0_65]
    at org.apache.derby.impl.services.monitor.StorageFactoryService.createServiceRoot(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.services.monitor.BaseMonitor.createServiceRoot(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.services.monitor.BaseMonitor.createPersistentService(Unknown Source) [FloodlightHttphs.jar]
    at org.apache.derby.impl.services.monitor.Monitor.createPersistentService(Unknown Source) [FloodlightHttphs.jar]
    ... 26 common frames omitted
66:27:33.515 INFO [n.f.i.l.LinkDiscoveryManager.main] Setting autoportfast feature to OFF
66:27:33.618 INFO [o.s.s.c.FallbackCProvider.main] Cluster not yet configured, using fallback local configuration
66:27:33.621 INFO [o.s.s.c.SyncManager.main] [32767] Updating sync configuration ClusterConfig [allNodes=[32767,node] [hostname=localhost, port=6642, nodeId=32767, domainId=32767], authScheme=CHALLENGE, RES
PONSE, keyStorePath=/etc/floodlight/auth_credentials.jcks, keyStorePassword is unset]
66:27:33.749 INFO [o.s.s.c.RPCService.main] Listening for internal floodlight RPC on localhost/127.0.0.1:6642
66:27:34.099 INFO [n.f.c.i.c.Controller.main] Listening for switch connections on 0.0.0.0:0.0:6633
66:27:41.317 INFO [n.f.j.j2ydsorServerDebugger.main] Starting Debugger on :6655
66:27:41.321 INFO [n.f.c.i.c.ChannelHandler-New I/O server worker #2-1] New switch connection from /127.0.0.1:58318
66:31:34.250 INFO [n.f.c.i.c.ChannelHandler-New I/O server worker #2-1] Disconnected switch [/127.0.0.1:58318 DPID[?]]
66:31:34.261 INFO [n.f.c.i.c.ChannelHandler-New I/O server worker #2-2] New switch connection from /127.0.0.1:58318
66:31:34.250 INFO [n.f.c.i.c.ChannelHandler-New I/O server worker #2-1] Switch OFFSwitchBase [127.0.0.1:58318 DPID[00:00:00:00:00:00:00:00:00]] bound to class class net.floodlightcontroller.core.internal.OFSw
itchImpl, writeThrottle=false, description OFDescriptionStatistics [Vendor: Nicira, Inc., Model: Open vSwitch, Make: None, Version: 2.0.2, S/N: None]
66:31:35.274 INFO [n.f.c.i.c.OFSwitchBase-New I/O server worker #2-2] Clearing all flows on switch OFSwitchBase [127.0.0.1:58318 DPID[00:00:00:00:00:00:00:00:00]]
66:31:35.279 WARN [n.f.c.i.c.c.notification.main] Switch 00:00:00:00:00:00:00:00:00 connected.
```

## Results and Conclusion:

For this project, I developed a Software Defined Networking layer-3 routing application that constructs routing tables based on a global view of the network topology. The routing table is installed in each SDN switch by the application, and each SDN switch forwards packets according to the installed route table. The application installs routing table entries that match packets based on their destination IP address and execute an output action to send the packet out a specific port on the SDN switch. Traffic is forwarded to the intended host using the shortest path. We use the Bellman-Ford algorithm to compute the shortest paths to reach a host from every other host. We implement Bellman-Ford using a queue to define which edges to explore next

We also develop a second SDN application. We implement a load balancer. It is provided with a list of virtual IPs and a set of hosts among which connections to the virtual IPs should be load balanced. When the clients initiate TCP connections with a specific virtual IP, the SDN switches will send the TCP SYN packet to the SDN controller. Our SDN application selects a host from a pre-defined set and install rules in an SDN switch to rewrite the IP and MAC addresses of packets associated with the connection.

The assignment gave us a chance to implement a software defined network(SDN's). Some of the challenges that I faced during the implementation were in setting up the environment and refactoring the source code to resolve module related errors. I had to understand and gather some domain related knowledge(regarding Mininet, Openflow etc.) to work on the problem. After this assignment I feel more comfortable in working in virtualized environments(VM's). The key takeaway from the project is the advantages SDN's bring to the networking domain and the significant improvement they offer over traditional networking.

## References and Related works:

1. Mininet network emulator documentation (<http://mininet.org>)
2. Openflow documentation (<https://www.opennetworking.org/sdn-resources/onf-specifications/openflow>)
3. <https://www.opennetworking.org/sdn-resources/sdndefinition>
4. Floodlight Java-based SDN controller documentation (<https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview>)
5. <http://pages.cs.wisc.edu/~agember/cs640/s15/assign5/>
6. <https://www.ibm.com/services/network/sdn-versus-traditional-networking>