Name: Meninato, Lorenzo lm4244 Date: 2021-12-22

Section: 001

Assignment 8: Final Project

Total in points (100 points total): ______________

Professor’s Comments:

Affirmation of my Independent Effort: *Lorenzo Meninato*

Introduction

In this project, I implemented the logic in an SDN controller for managing a layer-3 routing application and a distributed load balancer for redirecting new TCP connections to hosts in a round-robin order.

Related Work

There were several documents useful for completing this work:

http://mininet.org/
https://www.opennetworking.org/sdn-resources/onf-specifications/openflow
https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview
https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
http://homepages.inf.ed.ac.uk/mmarina/papers/sdn-chapter.pdf
http://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf

Algorithms

The main algorithm for computing the all pairs shortest paths was Dijkstra's algorithm Dijkstra's algorithm gives the shortest path between a given node and all other nodes in a graph. In order to compute the shortest paths between all nodes in the graph, I performed Dijkstra's algorithm on every node in the graph. For every switch, I use Djikstra's to calculate the distance to every other switch in the network. Further, when setting up the output packets on the correct port in order to reach the next switch in the path, for each switch I have to store the parent switch that was used to reach the shortest path to a given destination. Thus I use the data structure `HashMap<IOFSwitch, HashMap<IOFSwitch, IOFSwitch>>` so that given a switch and a destination, we can obtain the next switch in the path from the given switch with: `AllShortestsPaths.get(switch).get(destination)`. The remainder of the work for this path was just to set up the correct rules and install the correctly to every switch in the network - and rebuilding the flow tables accordingly as switches enter/leave/move throughout the network. I opted to simply recompute everything when changes occur to the system since it was simpler to implement, but this could be made more efficient in a future improvement.

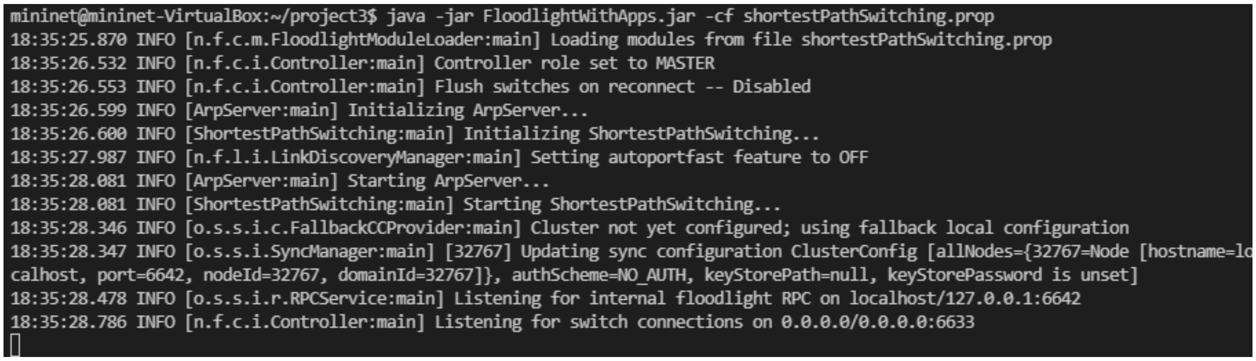
For Part 4, there were two main objectives. One was two install rules in every switch in the network to notify the controller whenever a client initiates a TCP connection with a virtual IP and for when the client issues an ARP request for the MAC address associated with a virtual IP. For this, I looped through every switch in the network by utilizing the `instances` class variable. And then I would install the corresponding IPv4 and ARP rules as well as all rules for all other packet types. Additionally, as each switched joined the network, I would install connection-specific rules for rewriting IP/MAC addresses of TCP packets sent between the client and the server.

Lastly, the connection-specific rules would match packets according to various types, like Ethernet type, source IP address, destination IP address, etc. To give more precedence to these rules, I ensured these rules were installed with a priority higher than the default. Further, there was an `IDLE_TIMEOUT` of 20 seconds to be given to these connection-specific rules, so that once TCP connections ended, by default the rules would be removed after 20 seconds.

Results

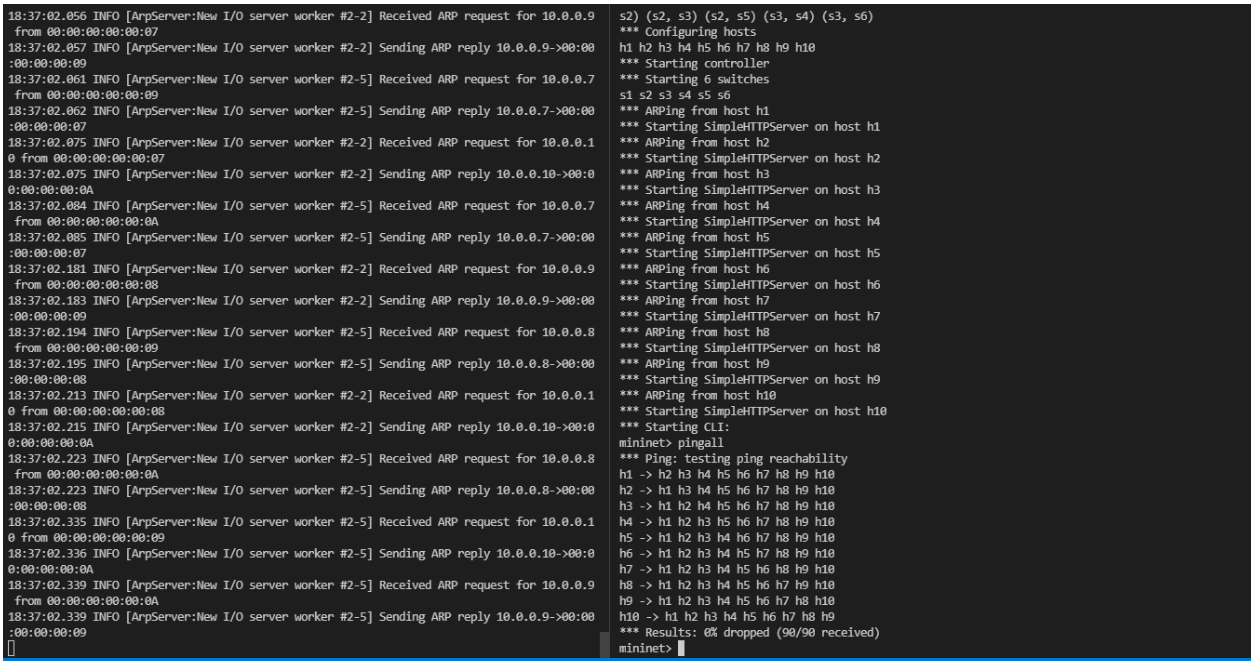
Below we can see some screenshots of both part 3 and part 4 applications working correctly:

Here for Part 3, we first set up the shortest path switching module:



```
mininet@mininet-VirtualBox:~/project3$ java -jar FloodlightWithApps.jar -cf shortestPathSwitching.prop
18:35:25.870 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from file shortestPathSwitching.prop
18:35:26.532 INFO [n.f.c.i.Controller:main] Controller role set to MASTER
18:35:26.553 INFO [n.f.c.i.Controller:main] Flush switches on reconnect -- Disabled
18:35:26.599 INFO [ArpServer:main] Initializing ArpServer...
18:35:26.600 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
18:35:27.987 INFO [n.f.l.i.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
18:35:28.081 INFO [ArpServer:main] Starting ArpServer...
18:35:28.081 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
18:35:28.346 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured; using fallback local configuration
18:35:28.347 INFO [o.s.s.i.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes={32767=Node [hostname=lo
calhost, port=6642, nodeId=32767, domainId=32767]}, authScheme=NO_AUTH, keyStorePath=null, keyStorePassword is unset]
18:35:28.478 INFO [o.s.s.i.r.RPCService:main] Listening for internal floodlight RPC on localhost/127.0.0.1:6642
18:35:28.786 INFO [n.f.c.i.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
▯
```

Then we can set up a second terminal and execute network operations via the mininet emulation layer:

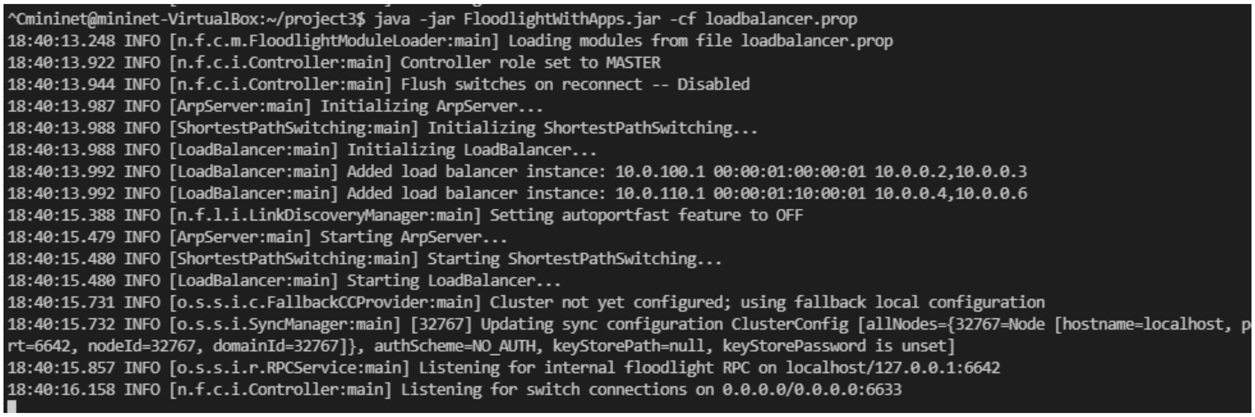


```
18:37:02.056 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.9
 from 00:00:00:00:00:07
18:37:02.057 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.9->00:00
:00:00:00:09
18:37:02.061 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.7
 from 00:00:00:00:00:09
18:37:02.062 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.7->00:00
:00:00:00:07
18:37:02.075 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1
0 from 00:00:00:00:00:07
18:37:02.075 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.10->00:0
0:00:00:00:0A
18:37:02.084 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.7
 from 00:00:00:00:00:0A
18:37:02.085 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.7->00:00
:00:00:00:07
18:37:02.181 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.9
 from 00:00:00:00:00:08
18:37:02.183 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.9->00:00
:00:00:00:09
18:37:02.194 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.8
 from 00:00:00:00:00:09
18:37:02.195 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.8->00:00
:00:00:00:08
18:37:02.213 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.1
0 from 00:00:00:00:00:08
18:37:02.215 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.10->00:0
0:00:00:00:0A
18:37:02.223 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.8
 from 00:00:00:00:00:0A
18:37:02.223 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.8->00:00
:00:00:00:08
18:37:02.335 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.1
0 from 00:00:00:00:00:09
18:37:02.336 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.10->00:0
0:00:00:00:0A
18:37:02.339 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.9
 from 00:00:00:00:00:0A
18:37:02.339 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.9->00:00
:00:00:00:09
▯
```

```
s2) (s2, s3) (s2, s5) (s3, s4) (s3, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
*** Starting 6 switches
s1 s2 s3 s4 s5 s6
*** 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
*** ARPing from host h4
*** Starting SimpleHTTPServer on host h4
*** ARPing from host h5
*** Starting SimpleHTTPServer on host h5
*** ARPing from host h6
*** Starting SimpleHTTPServer on host h6
*** ARPing from host h7
*** Starting SimpleHTTPServer on host h7
*** ARPing from host h8
*** Starting SimpleHTTPServer on host h8
*** ARPing from host h9
*** Starting SimpleHTTPServer on host h9
*** ARPing from host h10
*** Starting SimpleHTTPServer on host h10
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> █
```

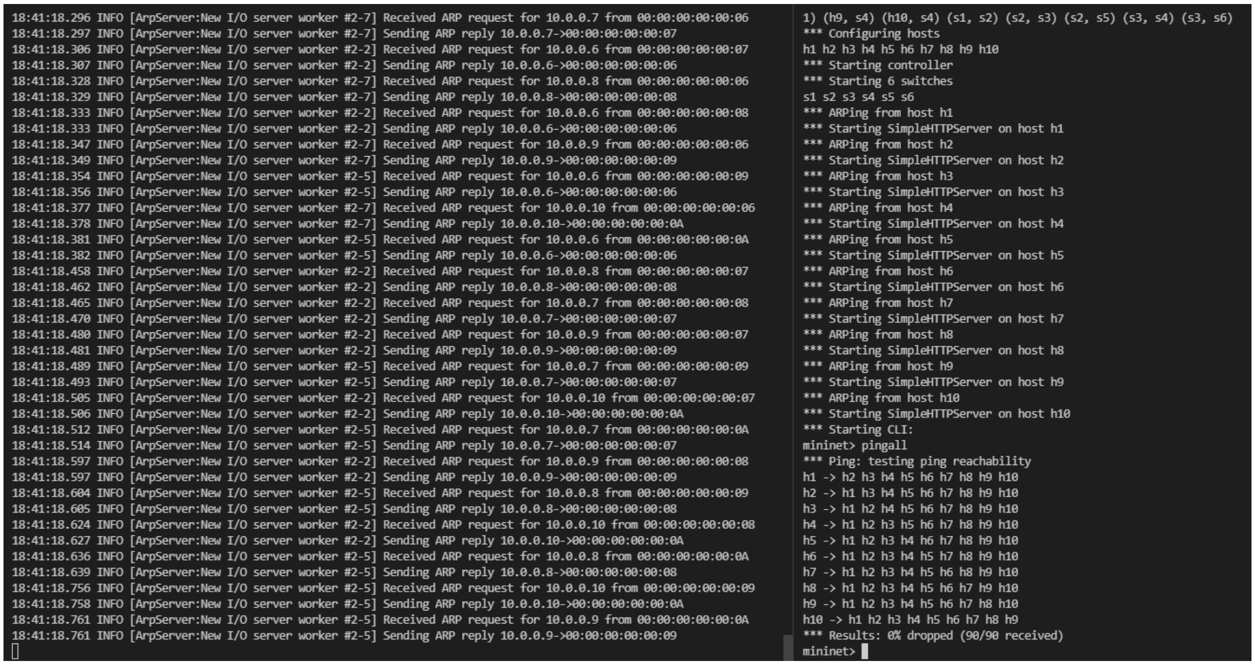
We can see that the ARP requests and replies are correctly handled by the level 3 routing application, and for various network topologies (I tested various), we have 0 packet loss and every host in the network can reach every other host!

Similarly, here is a screenshot for part 4 when the load balancing application starts up:



```
^Cmininet@mininet-VirtualBox:~/project3$ java -jar FloodlightWithApps.jar -cf loadbalancer.prop
18:40:13.248 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules from file loadbalancer.prop
18:40:13.922 INFO [n.f.c.i.Controller:main] Controller role set to MASTER
18:40:13.944 INFO [n.f.c.i.Controller:main] Flush switches on reconnect -- Disabled
18:40:13.987 INFO [ArpServer:main] Initializing ArpServer...
18:40:13.988 INFO [ShortestPathSwitching:main] Initializing ShortestPathSwitching...
18:40:13.988 INFO [LoadBalancer:main] Initializing LoadBalancer...
18:40:13.992 INFO [LoadBalancer:main] Added load balancer instance: 10.0.100.1 00:00:01:00:00:01 10.0.0.2,10.0.0.3
18:40:13.992 INFO [LoadBalancer:main] Added load balancer instance: 10.0.110.1 00:00:01:10:00:01 10.0.0.4,10.0.0.6
18:40:15.388 INFO [n.f.l.i.LinkDiscoveryManager:main] Setting autoportfast feature to OFF
18:40:15.479 INFO [ArpServer:main] Starting ArpServer...
18:40:15.480 INFO [ShortestPathSwitching:main] Starting ShortestPathSwitching...
18:40:15.480 INFO [LoadBalancer:main] Starting LoadBalancer...
18:40:15.731 INFO [o.s.s.i.c.FallbackCCProvider:main] Cluster not yet configured; using fallback local configuration
18:40:15.732 INFO [o.s.s.i.SyncManager:main] [32767] Updating sync configuration ClusterConfig [allNodes={32767=Node [hostname=localhost, p
rt=6642, nodeId=32767, domainId=32767]}, authScheme=NO_AUTH, keyStorePath=null, keyStorePassword is unset]
18:40:15.857 INFO [o.s.s.i.r.RPCService:main] Listening for internal floodlight RPC on localhost/127.0.0.1:6642
18:40:16.158 INFO [n.f.c.i.Controller:main] Listening for switch connections on 0.0.0.0/0.0.0.0:6633
```

And again we can see that for a sample network topology, the pingall command works with 0% packet loss:



```
18:41:18.296 INFO [ArpServer:New I/O server worker #2-7] Received ARP request for 10.0.0.7 from 00:00:00:00:00:06
18:41:18.297 INFO [ArpServer:New I/O server worker #2-7] Sending ARP reply 10.0.0.7->00:00:00:00:00:07
18:41:18.306 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.6 from 00:00:00:00:00:07
18:41:18.307 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.6->00:00:00:00:00:06
18:41:18.328 INFO [ArpServer:New I/O server worker #2-7] Received ARP request for 10.0.0.8 from 00:00:00:00:00:06
18:41:18.329 INFO [ArpServer:New I/O server worker #2-7] Sending ARP reply 10.0.0.8->00:00:00:00:00:08
18:41:18.333 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.6 from 00:00:00:00:00:08
18:41:18.333 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.6->00:00:00:00:00:06
18:41:18.347 INFO [ArpServer:New I/O server worker #2-7] Received ARP request for 10.0.0.9 from 00:00:00:00:00:06
18:41:18.349 INFO [ArpServer:New I/O server worker #2-7] Sending ARP reply 10.0.0.9->00:00:00:00:00:09
18:41:18.354 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.6 from 00:00:00:00:00:09
18:41:18.356 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.6->00:00:00:00:00:06
18:41:18.377 INFO [ArpServer:New I/O server worker #2-7] Received ARP request for 10.0.0.10 from 00:00:00:00:00:06
18:41:18.378 INFO [ArpServer:New I/O server worker #2-7] Sending ARP reply 10.0.0.10->00:00:00:00:00:0A
18:41:18.381 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.6 from 00:00:00:00:00:0A
18:41:18.382 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.6->00:00:00:00:00:06
18:41:18.458 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.8 from 00:00:00:00:00:07
18:41:18.462 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.8->00:00:00:00:00:08
18:41:18.465 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.7 from 00:00:00:00:00:08
18:41:18.470 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.7->00:00:00:00:00:07
18:41:18.480 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.9 from 00:00:00:00:00:07
18:41:18.481 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.9->00:00:00:00:00:09
18:41:18.489 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.7 from 00:00:00:00:00:09
18:41:18.493 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.7->00:00:00:00:00:07
18:41:18.505 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.10 from 00:00:00:00:00:07
18:41:18.506 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.10->00:00:00:00:00:0A
18:41:18.512 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.7 from 00:00:00:00:00:0A
18:41:18.514 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.7->00:00:00:00:00:07
18:41:18.597 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.9 from 00:00:00:00:00:08
18:41:18.597 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.9->00:00:00:00:00:09
18:41:18.604 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.8 from 00:00:00:00:00:09
18:41:18.605 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.8->00:00:00:00:00:08
18:41:18.624 INFO [ArpServer:New I/O server worker #2-2] Received ARP request for 10.0.0.10 from 00:00:00:00:00:08
18:41:18.627 INFO [ArpServer:New I/O server worker #2-2] Sending ARP reply 10.0.0.10->00:00:00:00:00:0A
18:41:18.636 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.8 from 00:00:00:00:00:0A
18:41:18.639 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.8->00:00:00:00:00:08
18:41:18.756 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.10 from 00:00:00:00:00:09
18:41:18.758 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.10->00:00:00:00:00:0A
18:41:18.761 INFO [ArpServer:New I/O server worker #2-5] Received ARP request for 10.0.0.9 from 00:00:00:00:00:0A
18:41:18.761 INFO [ArpServer:New I/O server worker #2-5] Sending ARP reply 10.0.0.9->00:00:00:00:00:09
```

```
1) (h9, s4) (h10, s4) (s1, s2) (s2, s3) (s2, s5) (s3, s4) (s3, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
*** Starting 6 switches
s1 s2 s3 s4 s5 s6
*** 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
*** ARPing from host h4
*** Starting SimpleHTTPServer on host h4
*** ARPing from host h5
*** Starting SimpleHTTPServer on host h5
*** ARPing from host h6
*** Starting SimpleHTTPServer on host h6
*** ARPing from host h7
*** Starting SimpleHTTPServer on host h7
*** ARPing from host h8
*** Starting SimpleHTTPServer on host h8
*** ARPing from host h9
*** Starting SimpleHTTPServer on host h9
*** ARPing from host h10
*** Starting SimpleHTTPServer on host h10
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet>
```

Conclusion

While more robust testing could be performed, this a suitable implementation of a layer-3 routing application and a distributed load balancer. Software-defined networking is clearly a powerful tool for manipulating computer networks programmatically. It is much more flexible, dynamic, and straightforward to implement as opposed to the equivalent hardware based solution.