Assignment - Lab of Network Software

Yaman Parasher, PIXNET Batch 2019-21 04/06/2020

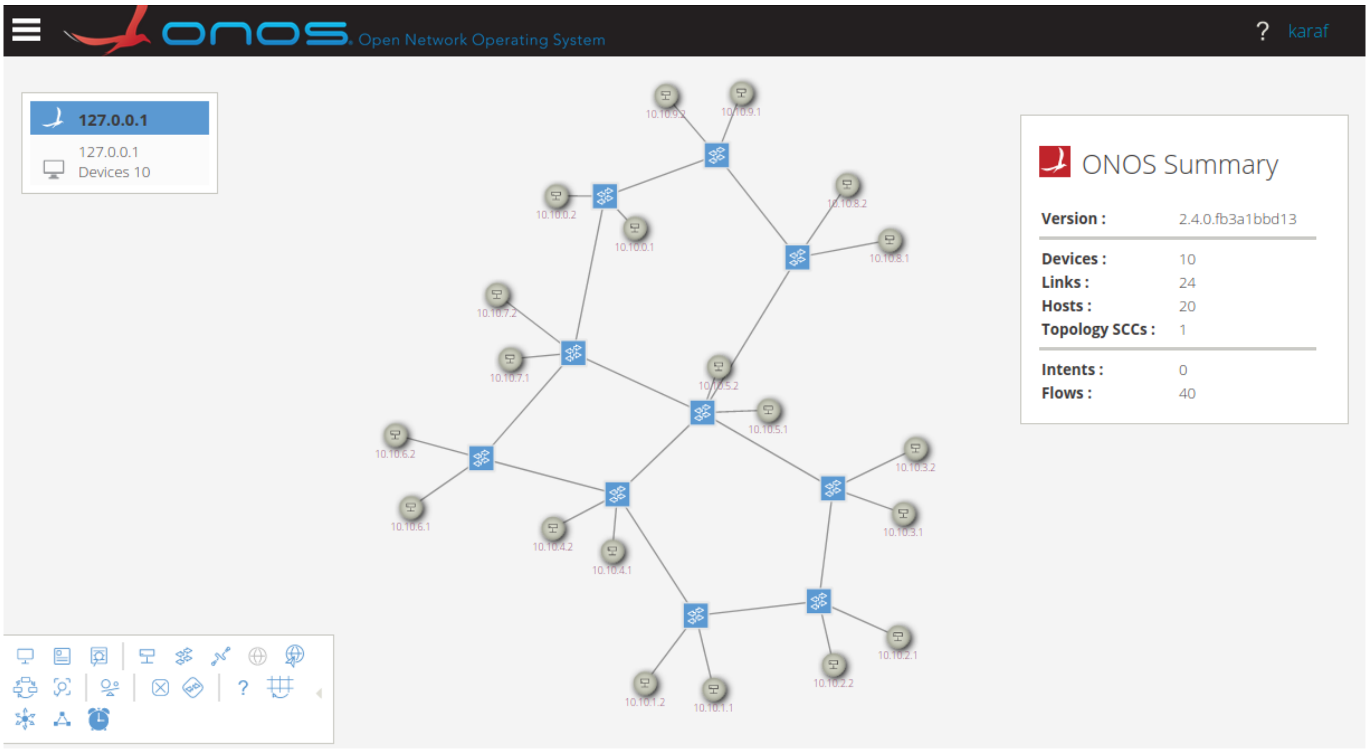
Introduction:

- TenPings app is a kind of hybrid bidriectional app that when receives an IP packet (i.e., a ping request) configures required flow rules for bidirectional communication between the source (src) and the destination (dst) hosts
- Moreover, TenPings app memorizes the history of received ping requests with time details in a separate datebase
- TenPings app also provides two CLI commands: show the history of performed pings using "tenping list" command & reset the history of performed pings using the “tenpings reset” command

Preliminary Steps:

1. Starting ONOS
2. Starting MIninet & displaying network topology on ONOS GUI interface
3. Starting “Reactive Forwarding” app in ONOS which will help in installation of flow rules one by one at each of the openflow switch
4. Running "pingall" command in Mininet
5. Showing List of disovered host on ONOS GUI interface by enabling the show host button located inside the bottom left bar of the interface.
6. Stopping the “Reactive Forwarding” app from the application section of ONOS GUI interface
7. Starting the “ARP proxy” app from the application section of ONOS GUI interface

According to step 4 of preliminary section, Figure 1 represents the network topology with OpenFlow switches and host connected together.

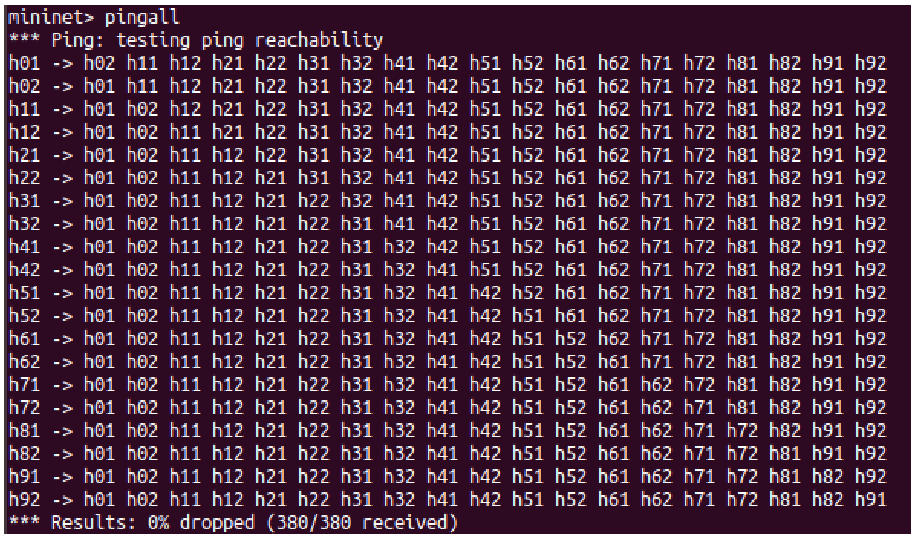


ONOS Summary

Version :	2.4.0.fb3a1bbd13
Devices :	10
Links :	24
Hosts :	20
Topology SCCs :	1
Intents :	0
Flows :	40

Figure 1: Schematic of Network Topology displaying Openflow Switches & Connected Hosts

The next Figure 2 represents the ping test performed for all the host connected in the topology. 0% dropped defines that all the host are able to communicate successfully with all other host present in the topology. Meaning they all are logially & physically connected & are reachable to each other.

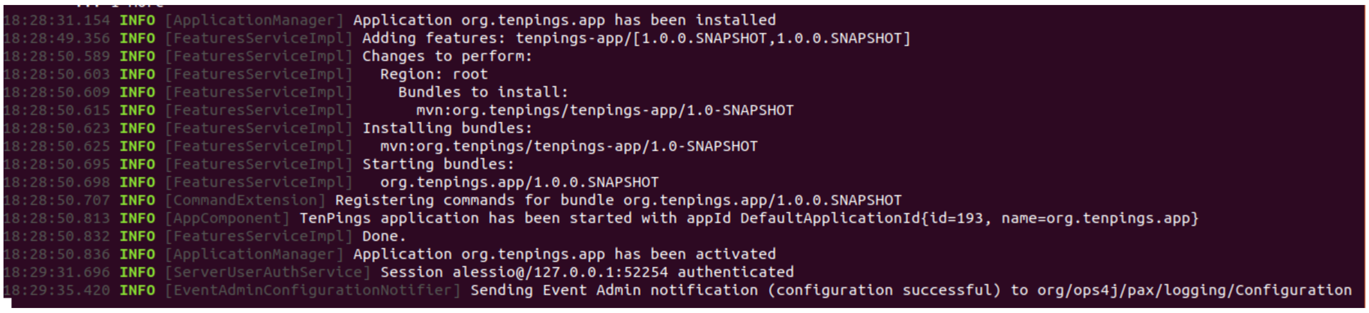


```
mininet> pingall
*** Ping: testing ping reachability
h01 -> h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h02 -> h01 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h11 -> h01 h02 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h12 -> h01 h02 h11 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h21 -> h01 h02 h11 h12 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h22 -> h01 h02 h11 h12 h21 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h31 -> h01 h02 h11 h12 h21 h22 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h32 -> h01 h02 h11 h12 h21 h22 h31 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h41 -> h01 h02 h11 h12 h21 h22 h31 h32 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h42 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h51 h52 h61 h62 h71 h72 h81 h82 h91 h92
h51 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h52 h61 h62 h71 h72 h81 h82 h91 h92
h52 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h61 h62 h71 h72 h81 h82 h91 h92
h61 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h62 h71 h72 h81 h82 h91 h92
h62 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h71 h72 h81 h82 h91 h92
h71 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h72 h81 h82 h91 h92
h72 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h81 h82 h91 h92
h81 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h82 h91 h92
h82 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h91 h92
h91 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h92
h92 -> h01 h02 h11 h12 h21 h22 h31 h32 h41 h42 h51 h52 h61 h62 h71 h72 h81 h82 h91
*** Results: 0% dropped (380/380 received)
```

Figure 2: Pingall Command of Mininet

TenPings app installation:

In this section, at the very first stage I build the downloaded tenpings app using "mvn clean install" command that triggers the jar packaging of the app. This includes compiling the sources, executing the tests and packaging the compiled files in a JAR file. Last step the install phase installs the resulting artifact into the local repository, so it can be used as dependencies by other Maven builds. After that I have also upload the app on the ONOS GUI interface to activate it directly.



```
18:28:31.154 INFO [ApplicationManager] Application org.tenpings.app has been installed
18:28:49.356 INFO [FeaturesServiceImpl] Adding features: tenpings-app/[1.0.0.SNAPSHOT,1.0.0.SNAPSHOT]
18:28:50.589 INFO [FeaturesServiceImpl] Changes to perform:
18:28:50.603 INFO [FeaturesServiceImpl]   Region: root
18:28:50.609 INFO [FeaturesServiceImpl]     Bundles to install:
18:28:50.615 INFO [FeaturesServiceImpl]       mvn:org.tenpings/tenpings-app/1.0-SNAPSHOT
18:28:50.623 INFO [FeaturesServiceImpl] Installing bundles:
18:28:50.625 INFO [FeaturesServiceImpl]   mvn:org.tenpings/tenpings-app/1.0-SNAPSHOT
18:28:50.695 INFO [FeaturesServiceImpl] Starting bundles:
18:28:50.698 INFO [FeaturesServiceImpl]   org.tenpings.app/1.0.0.SNAPSHOT
18:28:50.707 INFO [CommandExtension] Registering commands for bundle org.tenpings.app/1.0.0.SNAPSHOT
18:28:50.813 INFO [AppComponent] TenPings application has been started with appId DefaultApplicationId{id=193, name=org.tenpings.app}
18:28:50.832 INFO [FeaturesServiceImpl] Done.
18:28:50.836 INFO [ApplicationManager] Application org.tenpings.app has been activated
18:29:31.696 INFO [ServerUserAuthService] Session alessio@/127.0.0.1:52254 authenticated
18:29:35.420 INFO [EventAdminConfigurationNotifier] Sending Event Admin notification (configuration successful) to org/ops4j/pax/logging/Configuration
```

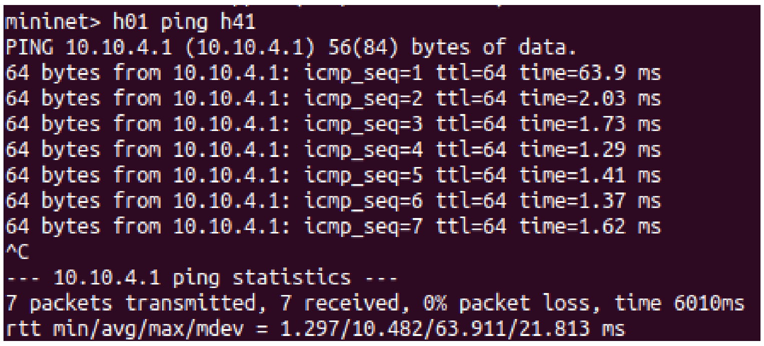
Figure 3: Log showing Installation of Tenpings App

Figure 3 represent the log generated after the installation & activation of the tenpings-app.

TenPings app execution:

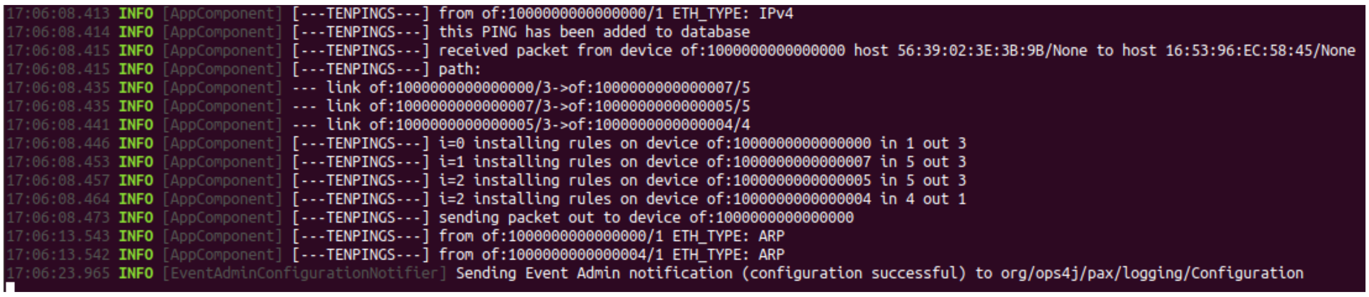
In this section, I performed a ping between two hosts. First between ***h01*** & ***h41*** and then between ***h21*** & ***h61***, till the transfer of 7 packets for each case. From Figure 4 & 5, it could be clearly seen that the first ping took a bit long time due because when the switch connected to source host (***h01***) sends the packet received from the source host to the controller, the controller took some time to discover destination of the packet and meanwhile do path computation so that it can install the flow reach on each device or switch traversed by the packet all the way to the destination, before sending out the packet out message to the first switch. The ARP packet here are managed by the ARP proxy application which simplies the environment by making the controller already know about the host that are connected.

After this, I generate a log of the performed actions using log:tail command in the ONOS CLI which can be seen through Figure 5. Here it shows the installation of flow rules along the path computed when the controller discovered the packet from the switch which was directly connected to the source host ***h01***. Similary after running "tenpings list" command in the ONOS CLI, I got the details about list of ping performed. This could be seen in Figure 6. As instructed in the end, all the flow-rules installed by the app were removed using REST API interface to start from scratch.



```
mininet> h01 ping h41
PING 10.10.4.1 (10.10.4.1) 56(84) bytes of data.
64 bytes from 10.10.4.1: icmp_seq=1 ttl=64 time=63.9 ms
64 bytes from 10.10.4.1: icmp_seq=2 ttl=64 time=2.03 ms
64 bytes from 10.10.4.1: icmp_seq=3 ttl=64 time=1.73 ms
64 bytes from 10.10.4.1: icmp_seq=4 ttl=64 time=1.29 ms
64 bytes from 10.10.4.1: icmp_seq=5 ttl=64 time=1.41 ms
64 bytes from 10.10.4.1: icmp_seq=6 ttl=64 time=1.37 ms
64 bytes from 10.10.4.1: icmp_seq=7 ttl=64 time=1.62 ms
^C
--- 10.10.4.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 1.297/10.482/63.911/21.813 ms
```

Figure 4: Pinging request between ***h01*** & ***h41***



```
17:06:08.413 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
17:06:08.414 INFO [AppComponent] [---TENPINGS---] this PING has been added to database
17:06:08.415 INFO [AppComponent] [---TENPINGS---] received packet from device of:1000000000000000 host 56:39:02:3E:3B:9B/None to host 16:53:96:EC:58:45/None
17:06:08.415 INFO [AppComponent] [---TENPINGS---] path:
17:06:08.435 INFO [AppComponent] --- link of:1000000000000000/3->of:1000000000000007/5
17:06:08.435 INFO [AppComponent] --- link of:1000000000000007/3->of:1000000000000005/5
17:06:08.441 INFO [AppComponent] --- link of:1000000000000005/3->of:1000000000000004/4
17:06:08.446 INFO [AppComponent] [---TENPINGS---] i=0 installing rules on device of:1000000000000000 in 1 out 3
17:06:08.453 INFO [AppComponent] [---TENPINGS---] i=1 installing rules on device of:1000000000000007 in 5 out 3
17:06:08.457 INFO [AppComponent] [---TENPINGS---] i=2 installing rules on device of:1000000000000005 in 5 out 3
17:06:08.464 INFO [AppComponent] [---TENPINGS---] i=2 installing rules on device of:1000000000000004 in 4 out 1
17:06:08.473 INFO [AppComponent] [---TENPINGS---] sending packet out to device of:1000000000000000
17:06:13.543 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: ARP
17:06:13.542 INFO [AppComponent] [---TENPINGS---] from of:1000000000000004/1 ETH_TYPE: ARP
17:06:23.965 INFO [EventAdminConfigurationNotifier] Sending Event Admin notification (configuration successful) to org/ops4j/pax/logging/Configuration
```

Figure 5: Log generated by the First Ping between ***h01*** & ***h41***

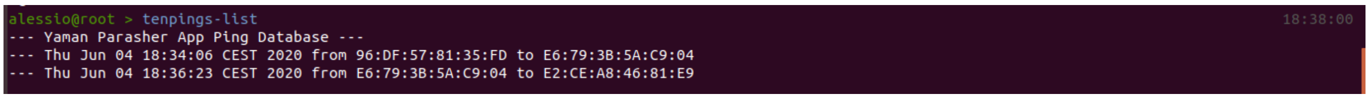
Todo 1

In this step, I was asked to change the string "Ping database" to some other string. After essential amendments in the code I see the reflected changes by re building & re installing the app again. I replace the given 'Ping database" string with " Yaman Parasher App Database" as shown in Figure 6

To do the same task, the changes that has been made in PingListCommand.java file, can be seen from the code below at line 12. The replacement of "Ping database" string with "Yaman Parasher App Ping Database".

Listing 1: Main code block for Todo 1

```
1  # PingListCommand.java
2  public class PingListCommand extends AbstractShellCommand
3  {
4  @Override
5  protected void doExecute()
6  {
7
8  # This line is required to use the functions and the variables that are defined in AppComponent
9  AppComponent appComponent = get(AppComponent.class);
10
11 # TODO-1 modify this string for STEP 4
print("--- Yaman Parasher App Ping Database ! ---");
print(appComponent.hostDatabaseToString());
}
}
```



```
alessio@root > tenpings-list                                                                 18:38:00
--- Yaman Parasher App Ping Database ---
--- Thu Jun 04 18:34:06 CEST 2020 from 96:DF:57:81:35:FD to E6:79:3B:5A:C9:04
--- Thu Jun 04 18:36:23 CEST 2020 from E6:79:3B:5A:C9:04 to E2:CE:A8:46:81:E9
```

Figure 6: Database Ping Entries

The Figure 6 above depicts the log generated by the tenping list command in the ONOS CLI.

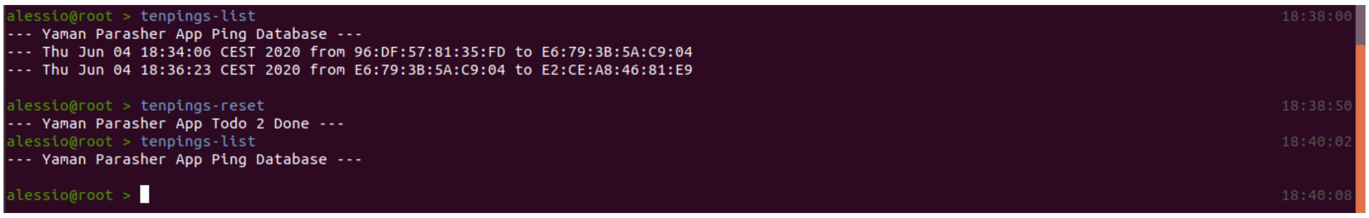
Todo 2

Here in this section, I was asked to clear the database of the performed pings by bringing changes in the file PingResetCommand.java so that when I entered the "tenpings reset" command in the ONOS CLI, I wont be able to discover any earlier ping entries.

To do the same task, the changes that has been made in PingResetCommand.java file, can be seen from the code below at line 12 & 13 mainly. The introduction of the concept of Java Map. Here pingInfoDatabase is a Java Map which use key value pair to perform various sort of actions. A Map is useful if you have to search, update or delete elements on the basis of a key.

Listing 2: Main code block for Todo 2

```
1  # PingResetCommand.java
2  @Service
3  @Command(scope = "onos", name = "tenpings-reset",
4  description = "Reset the database of executed pings")
5  public class PingResetCommand extends AbstractShellCommand
6  {
7  @Override
8  protected void doExecute()
9  {
10 # TODO-2: Here you have to clear the pingInfoDatabase that is
11 part of the AppComponent
12 AppComponent appComponent = get(AppComponent.class);
13 appComponent.pingInfoDatabase.clear();
14 }
15 }
```



```
alessio@root > tenpings-list                                                     18:38:00
--- Yaman Parasher App Ping Database ---
--- Thu Jun 04 18:34:06 CEST 2020 from 96:DF:57:81:35:FD to E6:79:3B:5A:C9:04
--- Thu Jun 04 18:36:23 CEST 2020 from E6:79:3B:5A:C9:04 to E2:CE:A8:46:81:E9

alessio@root > tenpings-reset                                                    18:38:50
--- Yaman Parasher App Todo 2 Done ---
alessio@root > tenpings-list                                                     18:40:02
--- Yaman Parasher App Ping Database ---

alessio@root >                                                                   18:40:08
```

Figure 7: Clearance of Ping Entries in Database

The Figure 7 represents the demonstration of clearance of ping database when "tenpings-reset" command was applied in the ONOS CLI. It clear the previous entries that were earlier discovered using the "tenpings-list" command earlier.

Todo 3

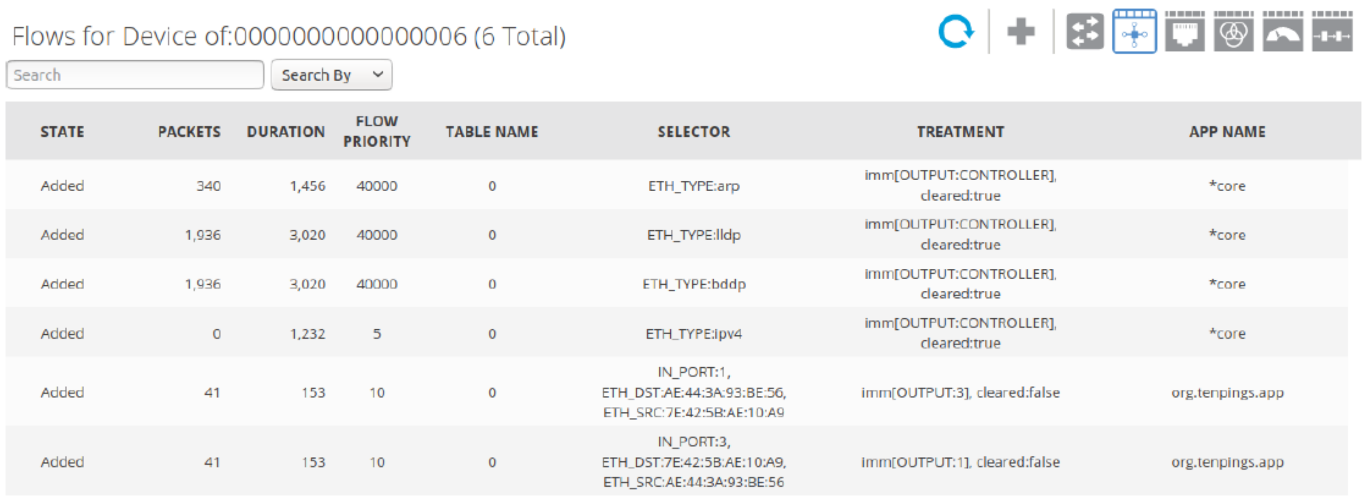
In this task, I was asked to modify the AppComponent.java file to automatically remove the installed flow rules 10 seconds after installation (e.g., using OpenFlow timers). For this, I first intiate a ping request in Mininet between two host for 30seconds.

To do the same task, the changes that has been made in AppComponent.java file, can be seen from the code below at line 27 & 49 . The addition of .withHardTimeout(10) in the direct & reverse direction flow rules.

Listing 3: Main code block for Todo 3

```
1  # AppComponent.java
2  private void installRules(PacketContext context, DeviceId deviceId,
3  PortNumber inPort, PortNumber outPort)
4  {
5  # TODO-3 work inside this function
6  Ethernet inPkt = context.inPacket().parsed();
7
8  # Build the rule in the direct direction
9  TrafficSelector selector = DefaultTrafficSelector.builder()
10 .matchInPort(inPort)
11 .matchEthSrc(inPkt.getSourceMAC())
12 .matchEthDst(inPkt.getDestinationMAC())
13 .build();
14
15 TrafficTreatment treatment = DefaultTrafficTreatment.builder()
16 .setOutput(outPort)
17 .build();
18
19 FlowRule directRule = DefaultFlowRule.builder()
20 .forDevice(deviceId)
.forTable(0)
.withSelector(selector)
.withTreatment(treatment)
.withPriority(10)
.fromApp(appId)
.makePermanent()
.withHardTimeout(10)
.build();

# Build the rule in the reverse direction
TrafficSelector reverseSelector = DefaultTrafficSelector.builder()
.matchInPort(outPort)
.matchEthSrc(inPkt.getDestinationMAC())
.matchEthDst(inPkt.getSourceMAC())
.build();

TrafficTreatment reverseTreatment = DefaultTrafficTreatment.builder()
.setOutput(inPort)
.build();

FlowRule reverseRule = DefaultFlowRule.builder()
.forDevice(deviceId)
.forTable(0)
.withSelector(reverseSelector)
.withTreatment(reverseTreatment)
.withPriority(10)
.fromApp(appId)
.makePermanent()
.withHardTimeout(10)
.build();

# Install both flow rules in the data plane
flowRuleService.applyFlowRules(directRule, reverseRule);
}
```

As we can see in Figure 8 and 9, after more than 10 seconds, the flowrules that were initially installed by tenpings app were automatically removed, which proved that proposed solution introduced by the addition of .withHardTimeout(10) works completely fine.

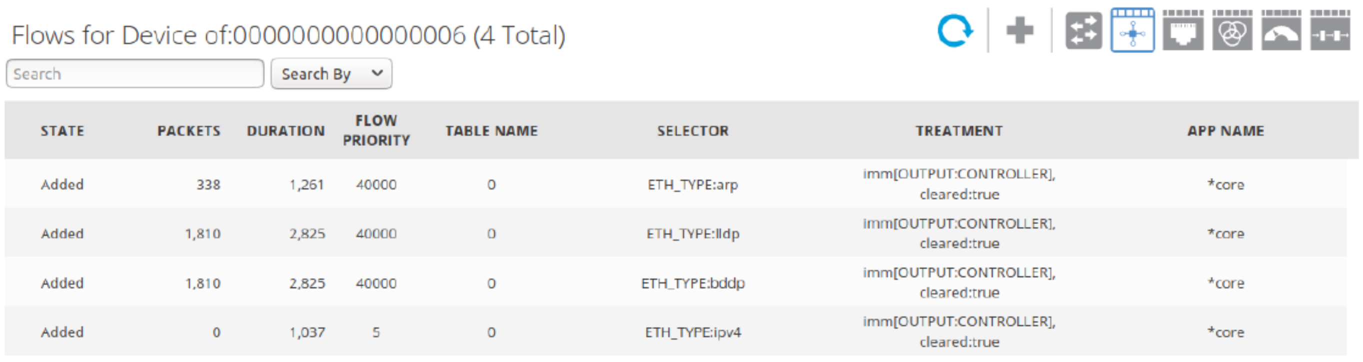


Flows for Device of:0000000000000006 (6 Total)

Search | Search By

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	340	1,456	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1,936	3,020	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1,936	3,020	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,232	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	41	153	10	0	IN_PORT:1, ETH_DST:AE:44:3A:93:BE:56, ETH_SRC:7E:42:5B:AE:10:A9	imm[OUTPUT:3], cleared:false	org.tenpings.app
Added	41	153	10	0	IN_PORT:3, ETH_DST:7E:42:5B:AE:10:A9, ETH_SRC:AE:44:3A:93:BE:56	imm[OUTPUT:1], cleared:false	org.tenpings.app

Figure 8: Flowrules Before



Flows for Device of:0000000000000006 (4 Total)

Search | Search By

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	338	1,261	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1,810	2,825	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1,810	2,825	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	0	1,037	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core

Figure 9: Flowrules After

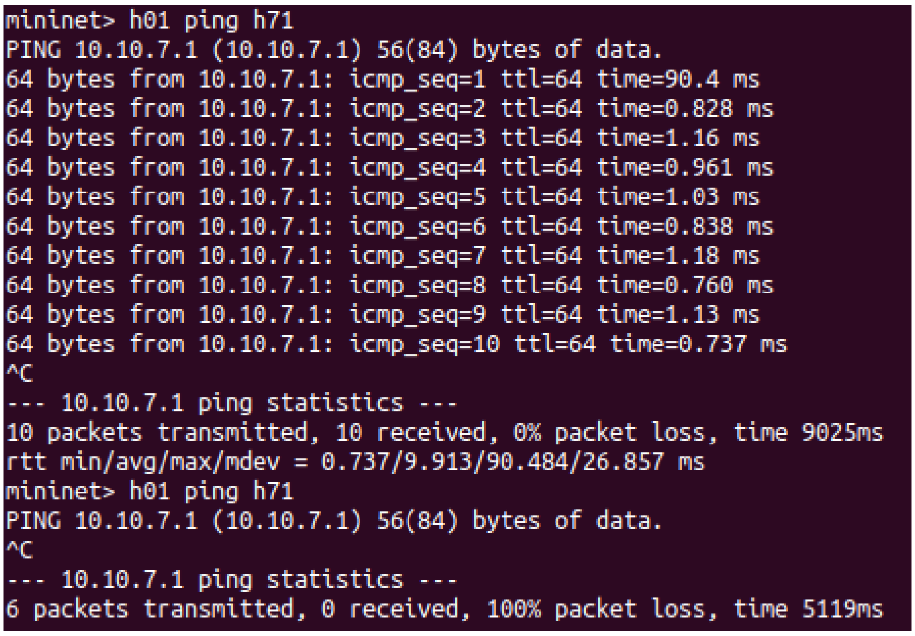
Todo 4

In this task, I was asked to modify the AppComponent.java file so that a ping is denied whenever the same ping (i.e., same src dst) is included in the local database. To do the same task, the changes that has been made in AppComponent.java file, can be seen from the code below with the introduction of if else block from line 3 until 11 . Here, its was checked from the database whether the ping entry between two host resides in the database or not. If it resides, I modify the code to send out a log error message saying that "Since, the ping has already been registered, the ping request is denied", othewise log message with the string "this PING has been added to database"

Listing 4: Main code block for Todo 4

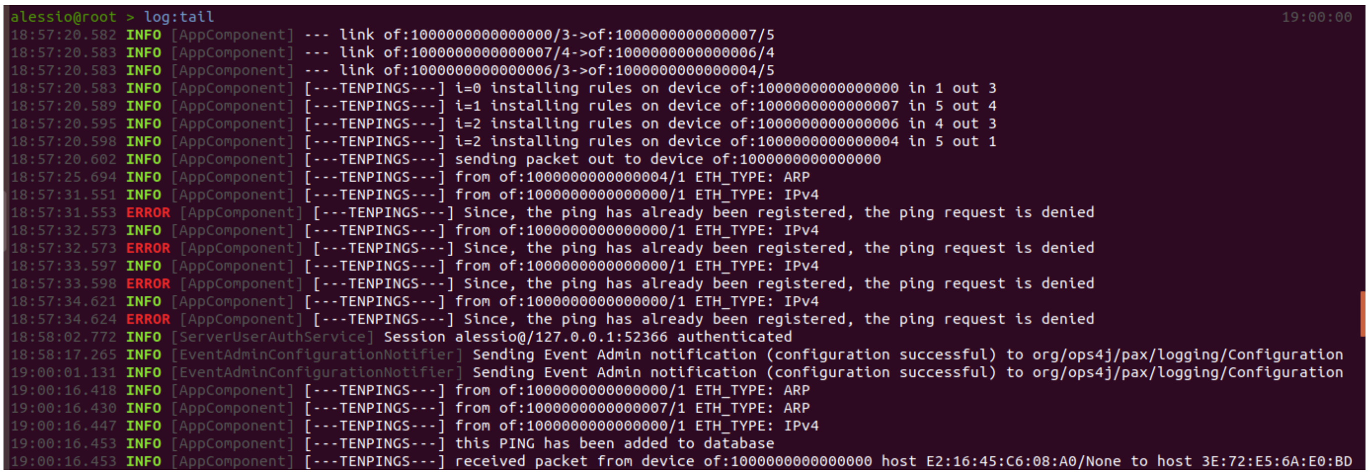
```
1  #Memorize that src has pinged dst
2  PingInfo pingInfo = new PingInfo(idSrc, idDst);
3  if(pingInfoDatabase.containsKey(idSrc.toString()+"-"+idDst.toString()))
```

```java
{
log.error("[---TENPINGS---] Since, the ping has already been registered, ↩
    the ping request is denied");
return;
}
else
{
pingInfoDatabase.put(idSrc.toString()+"-"+idDst.toString(), pingInfo);
log.info("[---TENPINGS---] this PING has been added to database");
```



```
mininet> h01 ping h71
PING 10.10.7.1 (10.10.7.1) 56(84) bytes of data.
64 bytes from 10.10.7.1: icmp_seq=1 ttl=64 time=90.4 ms
64 bytes from 10.10.7.1: icmp_seq=2 ttl=64 time=0.828 ms
64 bytes from 10.10.7.1: icmp_seq=3 ttl=64 time=1.16 ms
64 bytes from 10.10.7.1: icmp_seq=4 ttl=64 time=0.961 ms
64 bytes from 10.10.7.1: icmp_seq=5 ttl=64 time=1.03 ms
64 bytes from 10.10.7.1: icmp_seq=6 ttl=64 time=0.838 ms
64 bytes from 10.10.7.1: icmp_seq=7 ttl=64 time=1.18 ms
64 bytes from 10.10.7.1: icmp_seq=8 ttl=64 time=0.760 ms
64 bytes from 10.10.7.1: icmp_seq=9 ttl=64 time=1.13 ms
64 bytes from 10.10.7.1: icmp_seq=10 ttl=64 time=0.737 ms
^C
--- 10.10.7.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9025ms
rtt min/avg/max/mdev = 0.737/9.913/90.484/26.857 ms
mininet> h01 ping h71
PING 10.10.7.1 (10.10.7.1) 56(84) bytes of data.
^C
--- 10.10.7.1 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5119ms
```

Figure 10: Ping between ***h01*** & ***h71***



```
alessio@root > log:tail                                                                                                          19:00:00
18:57:20.582 INFO [AppComponent] --- link of:1000000000000000/3->of:1000000000000007/5
18:57:20.583 INFO [AppComponent] --- link of:1000000000000007/4->of:1000000000000006/4
18:57:20.583 INFO [AppComponent] --- link of:1000000000000006/3->of:1000000000000004/5
18:57:20.583 INFO [AppComponent] [---TENPINGS---] i=0 installing rules on device of:1000000000000000 in 1 out 3
18:57:20.589 INFO [AppComponent] [---TENPINGS---] i=1 installing rules on device of:1000000000000007 in 5 out 4
18:57:20.595 INFO [AppComponent] [---TENPINGS---] i=2 installing rules on device of:1000000000000006 in 4 out 3
18:57:20.598 INFO [AppComponent] [---TENPINGS---] i=2 installing rules on device of:1000000000000004 in 5 out 1
18:57:20.602 INFO [AppComponent] [---TENPINGS---] sending packet out to device of:1000000000000000
18:57:25.694 INFO [AppComponent] [---TENPINGS---] from of:1000000000000004/1 ETH_TYPE: ARP
18:57:31.551 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
18:57:31.553 ERROR [AppComponent] [---TENPINGS---] Since, the ping has already been registered, the ping request is denied
18:57:32.573 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
18:57:32.573 ERROR [AppComponent] [---TENPINGS---] Since, the ping has already been registered, the ping request is denied
18:57:33.597 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
18:57:33.598 ERROR [AppComponent] [---TENPINGS---] Since, the ping has already been registered, the ping request is denied
18:57:34.621 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
18:57:34.624 ERROR [AppComponent] [---TENPINGS---] Since, the ping has already been registered, the ping request is denied
18:58:02.772 INFO [ServerUserAuthService] Session alessio@/127.0.0.1:52366 authenticated
18:58:17.265 INFO [EventAdminConfigurationNotifier] Sending Event Admin notification (configuration successful) to org/ops4j/pax/logging/Configuration
19:00:01.131 INFO [EventAdminConfigurationNotifier] Sending Event Admin notification (configuration successful) to org/ops4j/pax/logging/Configuration
19:00:16.418 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: ARP
19:00:16.430 INFO [AppComponent] [---TENPINGS---] from of:1000000000000007/1 ETH_TYPE: ARP
19:00:16.447 INFO [AppComponent] [---TENPINGS---] from of:1000000000000000/1 ETH_TYPE: IPv4
19:00:16.453 INFO [AppComponent] [---TENPINGS---] this PING has been added to database
19:00:16.453 INFO [AppComponent] [---TENPINGS---] received packet from device of:1000000000000000 host E2:16:45:C6:08:A0/None to host 3E:72:E5:6A:E0:BD
```

Figure 11: Log generated for Second Ping Request

As you can see from the Figure 10 when , I ping host ***h01*** & ***h71*** for the first time it worked normally. However, next time when I try pinging both of them , it didn't worked out. It was only because of the changes that were made in the code (depicted above).The code blocked any extra ping request between the same pair of host after the first one. To give a wider picture, I took screenshot of the log generated in the ONOS CLI using log:tail command during the second ping request between the same pair of host. The figure represents the same string included in the code under the error warning informing that the ping entry between these two pair of host has been denied when tried the second time.