A0201480W

CS4226 Project Report

Learning Switch vs Intent Forwarding

I feel that using learning switch and intent forwarding mainly differs in terms of programming experience, where a lot of things are much simplified and abstracted well when using intents, but programming manually without intents can give greater flexibility and customization. Both ways would deliver similar outcome and perhaps performance (assuming manual programming is done properly and expect similar results).

I have performed a test using star topology, where s5 is the middle switch connected to all other switches, but link s1 to s5 are down. You may refer to Figure 1 below for reference.

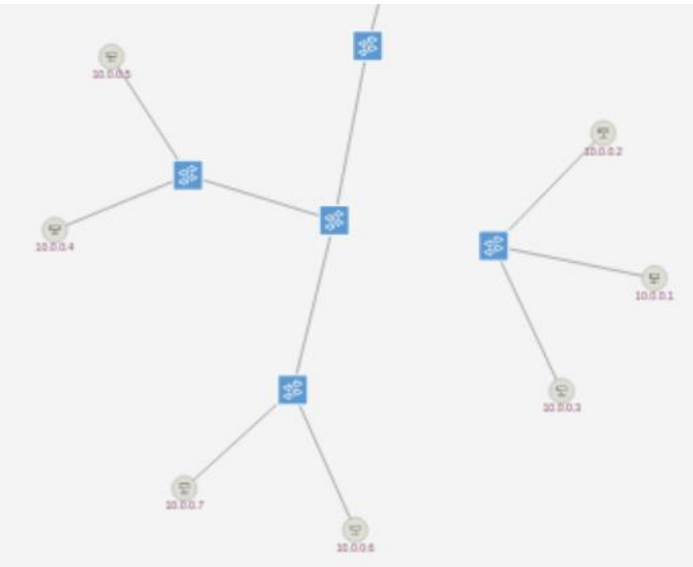

Figure 1: Star topology with link s1 s5 down

```
mininet> h3 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.241 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.036 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.040 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.037 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.031/0.077/0.241/0.082 ms
mininet> h6 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2000ms

mininet>
```

Figure 2a: Result using learning switch

```
mininet> h3 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.248 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.040 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.039 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.038/0.082/0.248/0.083 ms
mininet> h6 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
^C
--- 10.0.0.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3009ms

mininet>
```

Figure 2b: Result using intent forwarding

This test has a case where link s1 s5 are down. h3 will still be able to ping h1 (as they are still connected) but h6 will not be able to ping h1. Referring to Figure 2a and 2b, notice that there is no difference in terms of outcome ability to ping, nor major performance difference.

This abstracted APIs in intent forwarding is likely to be more robust as they have been developed longer and should there be new research or technology breakthrough, it is expected for the APIs to be updated ever faster especially since it is now open-source and more scalable than before.

In conclusion, if one would want to implement a standard usage which exists as an intent, intents should be used. Manual programming such as learning switch is good for learning purposes, or if we have special cases which behaves differently from standard usage, increasing its customizability/ flexibility.

Problems Encountered

Problem	Solution
Some files were corrupted which made the code unable to run as intended even when its supposed to work. Some bugs happened (still have not figured) which may be due to .pom or package clean related, but it costs me a lot of time to figure out.	As I realized that it may be due to file corruption, I had to redownload the original VM image and migrate my code over, which turns out to be working.
I tried to follow the guide in forum for integrating local VSCode and mininet VM. However I realized that I do not have ethernet, hence my mininet VM do not have internet access. As such, I was not able to integrate to my github repository or to check the ONOS GUI.	I had to resolve back to do my work inside the VM for internet access, so that I am able to check through ONOS GUI that my topology is set up properly. In addition to syncing to github, I have also learnt to clone checkpoints in VirtualBox as an additional tool for saving my work.
It is difficult to navigate through the documentation due to the lack of sequence and class diagrams, making it hard to see the bigger picture. Documentation is also dispersed in different places making things more disconnected, hence harder to grasp for new learners.	The walkthrough done by the TAs help a lot, and there is also some guidance in terms of documentation visualization and code. I had to spend some time learning from multiple sources to have a better grasp on how things work as a whole.
Debugging is not straight forward as we can't use the normal way of debugging in our IDE, as running the programme is not simply running the main file.	Tried to search online ways to debug onos / openflow debugging, but they turn out to be too complicated and I was still not sure how to incorporate into our programme to debug. Hence, resolved to logging info and had verbal discussion with friends, although still have not figured a good practice for debugging.