Report

Comparing `learningSwitch` and `intentForward`

Differences:

	`learningSwitch`	`intentForward`
Learning Mechanism	Uses basic MAC address learning to build and maintain their MAC address tables. It learns the MAC addresses of devices connected to its ports by inspecting Ethernet frames and associating the source MAC addresses with the ingress port.	Goes beyond basic MAC address learning. It is designed to understand and act on higher-level network policies and user-defined intents. Instead of just learning MAC addresses, it understand the intent behind the network configuration and can enforce policies accordingly.
Automation and Policy Enforcement	Requires manual configuration and lack the ability to automatically enforce complex network policies. Network administrators need to configure VLANs, access control lists, and other policies individually.	Capable of automating network configuration and policy enforcement based on high-level user-defined intent. It interprets the administrator's intent and implement it across the network, simplifying complex network management tasks.
Network Visibility and Analytics	Provides limited visibility into network traffic and performance, making it challenging to troubleshoot issues or optimize network performance.	Offer advanced network visibility and analytics. It can provide detailed insights into network traffic, application performance, and security compliance, enabling better decision-making and troubleshooting.
Scalability	Typically less scalable and may require more manual configuration as the network grows	Designed to be more scalable and adaptable to network growth. Dynamically adjust to changes and updates in the network.
Ease of Management	Managing a network can be more time-consuming and may involve a significant amount of manual configuration and troubleshooting.	Designed to simplify network management by allowing administrators to express their intent in natural language, and the switch takes care of translating that intent into network configurations.
Dynamic Network Response	Responds to changes in the network, but often with manual intervention from administrators.	Dynamically adapt to changes in network conditions and automatically adjust configurations to meet the desired intent.

Behavior with varied topology:

	`learningSwitch`	`intentForward`

	`learningSwitch`	`intentForward`
Ring Network Topology	In a ring topology, it may not handle loops well. When a frame enters the network, it is flooded to all ports until the switch learns the MAC address/port associations. This can lead to broadcast storms if not properly managed and can degrade network performance in the presence of loops.	Designed to have more intelligent behavior. It can detect loops and automatically take actions to prevent broadcast storms or packet duplication.
Star Network Topology	In a star topology, it perform as expected. Learn MAC addresses and forward traffic based on MAC address table entries. Does not require complex loop detection and prevention mechanisms in star topologies.	Provides more advanced capabilities, such as the ability to enforce strict security policies and quality of service (QoS) settings in a star network. It can automatically apply the intended policies and quality of service settings to each port, ensuring network behavior aligns with the administrator's intent.

Difficulties faced and how I solved them

The first difficulty I faced doing this assignment was setting up the ONOS virtual machine on my laptop. Not only was the setup instructions in the tutorial incorrect with regards to the VM's network configuration, the VM image given also does not contain the necessary files for me to follow along. I solved this by asking my friend to help me with the setup steps, and he pointed out the correct network configuration for the VM. I also used the Piazza forum to find out about the correct VM images we were supposed to use.

Next, I had difficulty understanding the entire progamming assignment document. It is very unorganised and unclear about what I am expected to learn and do. For example, on one hand, I was told to implement `AddFirewallOnPortCommand` command line, however upon reading the source code I found out that it has already been implemented, and only later would I realise I have to implement a method in a completely different file. Exercise 1 did not mention anything about `star.py`, and I only figured out that it means I have to fill out the python source code once I finished reading Exercise 2. The folder structure shown is also completely incorrect. I had to figure all of this out by reading the Piazza forum, and asking my friend (again).

The programming assignment was difficult because information about usage of ONOS is very scattered on the internet. The ONOS Java API documentation did help with figuring out what methods are available and their return types, but ceases to be useful once a chained method call starts to grow out of control. The hint comments did the opposite of helping with the assignment - it added too much confusion as with what subroutine we are supposed to implement, and if such an operation is even possible. For example, "Insert the FlowRule to the designated output port" does not make sense. Again, I consulted the Piazza forum for answers, and basically copy-pasted TA Mao Yancan's replies.

In conclusion, learning and programming ONOS was EXTREMELY frustrating and difficult with the amount of indirection and mental overhead that this assignment has given. I do not even know whether I learnt anything valuable or relevant to SDN at all, because most of the time has been spent troubleshooting VM setup problems and looking for useful documentations for ONOS.