Flow Installation Mechanisms in SDN

Daniyal Ahmed¹(daniawan4@gmail.com),
Ateeqa Dawood²(ateeqadaud829@gmail.com),
Muhammad Rafey³(rafeykhan998@gmail.com),
Mudassar Hussain⁴(mudassar.hussain@uow.edu.pk)
Department of Computer Science
University of Wah
info@uow.edu.pk

ABSTRACT

Software Defined Networking (SDN) has developed tremendously in the recent years. This paper addresses different flow installation mechanisms. The idea of a programmable network which is the foundation of SDN ensures dynamic network management by separating the control plane from data and management planes. In SDN the control plane resides in SDN Controller, which is responsible for the installation of flows on the data plane. Whenever a flow enters into an SDN switch, flow tables are generated for those flows and installed at data plane from the control plane. These flows have their own properties because of their relevant protocols which are attached to them. A flow consists of a matching pattern, actions, and counter. Based on the respective flow, data communication takes place. This paper investigates three kinds of flow installation mechanisms, reactive, proactive and hybrid. Based on these three flow installation mechanisms, we have analyzed the performance of SDN by using different performance metrics. We have used the Mininet and POX controller to evaluate the performance analysis in Ubuntu and Windows-based environments. Finally, we have drawn conclusions.

Keywords: SDN, Flow Installation mechanisms, Mininet and POX Controller, Reactive, Proactive, OpenFlow.

1. INTRODUCTION

In this paper, we have discussed the flow installation mechanisms in Software Defined Networking (SDN). The main purpose of SDN is to separate the data plane from the control and management planes in middleboxes and forwarding devices (routers and switches) for improving network management and reliable communication. Whenever SDN is configured, the SDN Controller must be familiar with the flow entries. When a flow enters into an SDN Switch a separate flow table is generated for the flows. There are different kinds of data in these flows so, each flow should have to import its flow table in the controller database [1]. Otherwise, it may become arduous for a controller to manage them. When the flows are not active and there is no communication for specific hosts, the flows will get expired and removed from the switch [2]. The expiry time of flow differs and can be configured via the controller. There are two types of timeouts of flows, idle timeout and hard timeout. The idle timeout is the number of seconds after which a flow entry is removed from the flow table and the hardware provided because no packets match it. The hard timeout is the number of seconds after which the flow entry is removed from the flow table. If a flow entry has both an idle timeout and a hard timeout associated with it, the first timer to expire causes the flow entry to be removed. If the idle timer expires first, the flow entry is removed at that point only if there are no matching packets.
Otherwise, the flow entry is removed when the hard timeout expires. This work investigates three kinds of flow installation mechanisms, reactive, proactive and hybrid. In reactive flow installation whenever a new flow of packet comes into the switch, the Openflow switch looks for that flow into the flow table, if flow matches, then new flow is allowed to pass, but if the switch doesn’t recognize the flow, it creates a packet-in message and sends it towards the controller for further instructions. As it is a reactive approach, so a new flow is installed in a switch for that flow via the packet-out message from the controller. In Proactive approach, flows are populated in advance, i.e. before the first packet of a new flow arrives. The proactive approach minimizes the setup delay of the flows and reduces the overall number of signaling messages because flows are already installed for them. When flows and their actions are predefined in switches even before the packet arrives, as a result, all packets are sent or forwarded to nodes. Forwarding is done by just matching flows in the flow tables, as they are already there so the time is saved in a huge amount [3]. Hybrid flow method is the combination of both proactive and reactive approaches, a hybrid approach is considered to be a very flexible approach as compare to other approaches because it has the functionality of both approaches. When it is run it installs basic flows even before the commencement of communication like the proactive approach and when communication starts, it also installs flows for those hosts whose flows are not installed proactively. It provides great flexibility, robustness and reduces delays. Based on these three flow installation mechanisms, we have analyzed the performance of SDN by using different performance metrics. We used the POX controller and the Mininet Emulator to analyze the performance analysis of flows. In addition, the Windows operating system and Ubuntu operating system are also used for the implementation and analysis of our work.

2. SOFTWARE DEFINED NETWORKING (SDN)

The SDN is a modern network technology which is a new perspective in the world of computer networking that provides easy network administration and allows mannerly fast configuration of a network in order to enhance the network performance as well as network monitoring. It is signified to mark the reality that the unchanged traditional network architecture, which is localized and complicated while present-day networks need more flexibility and easy management. Moreover, it proposes a compact intelligent network in a single network element by decoupling the Data Plane (forwarding procedure of network packets) from Control plane (routing procedure). Control Plane made up of single or multiple controllers which are examined as of its brain where the entire intelligence is included. Although, the centralization of intelligence has its particular disadvantages when we talk about the security, flexibility, and scalability and that is the major problem of SDN on which various researches are being carried out to fix such kind of issues

2.1 SDN Applications

SDN Applications are the schemes that rightly and mannerly convey their network necessities and crave network conduct to the SDN Administrator through a Northbound Interface (NBI). Moreover, they could consume an abstracted sight of the network for its interior decision-making objective. An application contains single SDN application argumentation and multiple NBIs. These Applications may reveal themselves to other layers of abstracted network control, therefore presenting single or multiple effective NBIs via several NBI agents [4].

2.2 SDN Controller

The SDN Controller is a logically centralized establishment in charge of:
• Put the needs from the SDN application layer down to the SDN Datapath's and
• Deliver the SDN applications along with an abstract sight of the network.

An SDN Controller contains single or multiple NBI operator, the Control to Data-Plane Interface (CDPI) and the SDN Control Logic driver. Defined as a logically centralized entity, neither suggests nor prevents administration details. These administration details include a confederation of numerous controllers, the vertical association of controllers, transmission interfaces among controllers, and virtualization or slicing of network resources [4].

2.3 SDN Datapath

SDN Datapath is the logical network mechanism that reveals visibility and conclusive control over its proclaimed forwarding and data handling abilities. The logical presentation may enclose all or the part of physical surface resources. SDN Datapath contains a CDPI agent and a set of single or multiple traffic forwarding mechanisms and zero or multiple traffic handling tasks. These mechanisms and tasks may contain basic forwarding mechanisms among the Datapath’s outer interfaces or interior traffic handling or ending tasks. Single or multiple SDN Datapath’s may carry in one physical network segment—a homogeneous physical mixture of transmission resources, supervise as a unit. The SDN Datapath may also be interpreted over the multiple physical network segments [4].

2.4 SDN Control with Data-Plane Interface (CDPI)

The SDN CDPI is an interface between the SDN Administrator and SDN Datapath, that delivers at least:
• Mannerly control over all the forwarding actions,
• Potential notice,
• Statistics details, and
• Occurrence notification.

A single entity of SDN falls in a presumption that the CDPI has performed in an accessible, supplier impartial and practical method [4].

2.5 SDN North Bound Interfaces (NBI)

SDN NBIs are the interfaces among SDN applications and controller and normally deliver a precise network sight and allow the exact appearance of network behavior and needs as seen in figure 1. This could happen at every level of latitude (abstraction) and over the distinct groups of functionalities. A single entity of SDN falls in a presumption that interfaces are performed in an accessible, supplier impartial and practical method [4].

Figure 1: An effective SDN architectural overview [4]

3. FLOW INSTALLATION MECHANISMS IN SDN

The control plane resides in the SDN controller, which is responsible for the installation of flows on the data plane for the communication. There are three types of flow installation mechanisms in SDN, reactive, proactive and hybrid flow installation.

3.1 REACTIVE FLOW INSTALLATION

When the new flow of packet came into the switch, the OpenFlow switch looked
for that flow into the flow table, if flow matches, then the packet is allowed to pass. However, if the switch doesn’t recognize the flow, it creates a packet-in message and sends it towards the controller for further instructions. As it is a reactive approach, so a new flow is installed in the switch for that flow via the packet-out message from the controller [5].

3.2 PROACTIVE FLOW INSTALLATION

The proactive approach is a very good approach as it installs flows based on the current network environment before the packet reached a particular switch. In Proactive approach, flows are populated in advance, i.e., before the first packet of a new flow arrives. The proactive approach nullifies the setup delay of the flows and reduces the overall number of signaling messages because flows are already installed for them [5].

3.3 HYBRID FLOW INSTALLATION

Hybrid flow method is the combination of both proactive and reactive approaches. This approach is considered to be a very flexible approach as compared to other approaches, because, it has the functionality of both approaches. It installs basic flows even before communication is started like proactive approach and when communication starts it also installs flows for those hosts whose flows are not installed proactively, providing great flexibility, robustness and reduces delays [5].

4. IMPLEMENTATION

For the implementation of Flow Installation Mechanisms in SDN, we used Mininet Emulator and POX SDN controller platform for the implementation and evaluation of our work in Windows and Ubuntu-based operating systems. The Mininet network emulator is usually used for experimentation purposes as well as learning, testing, development, prototyping, and research. Different network topologies along with virtual controllers, switches, hosts, and links have been created by using Mininet. We have started by creating Mininet’s default topology. Moments default topology is known as minimal topology that includes two hosts connected to a single OpenFlow kernel Switch and a Controller using OpenFlow reference. After this, we created our own custom topology through a Python file containing a custom topology code written in python. The goal of the proposed work is to provide a custom topology framework which enables the creation of any custom network topology of user’s choice, including multiple IP networks.

For the analysis of these three approaches, we used the topology which contains 1 controller, 3 switches, 4 hosts with 6 links. In addition, we used the POX SDN controller and OpenFlow v1.3 for the analysis of our research work.

4.1 REACTIVE FLOW INSTALLATION

For reactive approach before running the topology, we run the l2_learning module of pox controller, which is based upon the reactive flow installation. To run l2_learning, we changed directory to pox controller by command cd pox. After that following command was used to run l2_learning module:

```
sudo python pox.py forwarding.l2_learning
```

To run l2_learning module indirectly we used spanning tree that also uses l2_learning, following command is used to run it:

```
sudo python ./pox.py log.level -Debug samples.spanning_tree
```

When the topology is run the pox controller starts installing flows into the switches as shown in figure 2:
4.2 PROACTIVE FLOW INSTALLATION

For a proactive approach, two types of modules need to run simultaneously. First topo_proactive from forwarding and discovery from OpenFlow. discovery module and topo_proactive discovers the nodes in the network and installs flows proactively. When flows and their actions are predefined in switches even before the packet arrives, in result all packets are sent or forwarded to nodes only by just looking them in the flow tables. In this way, the time is saved in huge amount, which results in increased throughput. We used a python custom file “proactive.py” with network topology. This file installs flow proactively without controller running the modules in parallel. This is because custom file installs flows proactively based upon the modules discussed above. The running form of modules along with the installation of flows are as shown in figure 3.

To successfully run topology with the hybrid approach we used spanning_tree module within pox controller and OpenFlow. discovery installs flows at the beginning while l2_learning also establishes the connections in the topology even before communication occurs, its working is shown in figure 4. Command for Spanning_tree module:

```
sudo python ./pox.py log.level -Debug samples.spanning_tree
```

5. RESULTS AND EVALUATION

The amount of data traveled from one node to another in an instance of time is called throughput. The data can be traveled in megabits per second (Mbps) or gigabits per second (Gbps) in our simulation. To measure throughput, we used iperf3 tool, it can measure the maximum bandwidth that has been achieved on a network. Iperf3 works in command line interface (CLI) mode and works in a client-server environment. We checked the throughput of our simulation setup by using a transmission control protocol (TCP).

5.1 REACTIVE APPROACH RESULTS

In the reactive approach, it takes some amount of time to set-up flows and starts communication between 0 to 1 second. once communication started the throughput was calculated was 12.03 Gbps. The maximum throughput achieved at the 17th second which was 13.75 Gbps. The overall throughput graph for all instances of time is shown in Figure 5.
5.2 PROACTIVE APPROACH RESULTS

In the proactive approach, the throughput calculated in the second-time instance was 12.25 Gbps and at 7th-time instance it reached up to 13.95 Gbps. It can be noticed that in proactive approach throughput line after many ups and downs became stable because the flows were defined earlier, even before the communication was started so it didn’t take much time to set up, at some points like 10th and 15th second throughput drops to 11.5 and 11.2 Gbps which is the lowest in this graph. Overall the continuity and stability of data values make proactive approach better than the reactive approach as shown in Figure 6.

5.3 HYBRID APPROACH RESULTS

As Hybrid approach installs flows proactively and supports reactive flows, making it better than previous approaches, in graph 1st point starts from 12.1 Gbps and starts climbing up, stabilizing its throughput and reaches maximum of 14.6 Gbps in 20th second, which is the highest throughput of hybrid approach as compared to previous approaches, its lowest point is even above 12.0 Gbps, which makes it better than reactive and proactive approaches as shown in Figure 7. The hybrid approach is considered to be a very flexible approach as compare to other approaches because it has the functionality of both approaches.

6. CONCLUSION
In this paper, we have investigated reactive, proactive and hybrid flow installation mechanisms in SDN. In reactive flow installation approach, whenever a new flow of packet reaches the switch, the OpenFlow switch looks for that flow into the flow table, if flow matches, then the packet is allowed to pass. However, if the switch doesn’t recognize the flow then its switch creates a packet-in message and sends it towards the controller for further instructions. In Proactive flow installation approach, the flows are installed in the switch before any communication is started in the network. By comparing both reactive and proactive approaches, proactive is better because all necessary flows for the packets are installed in advance, hence, flow setup time and installation time is reduced. The third technique, which is the combination of the reactive and proactive approaches, we have a Hybrid approach. It uses best characteristics of both approaches for the installation of flows. The results suggest that the hybrid approach is the best of all three approaches and provides better throughput during data communication.

7. REFERENCES

[1] S. Singh, R. A. Khan, and A. Agrawal, “Flow Installation in Open Flow Based Software Defined Network ; A Security Perspective,” Int. J. Emerg. Trends Technol. Comput. Sci., vol. 4, no. 1, pp. 15-18, 2015.

[2] Juniper.net, “Understanding OpenFlow Flow Entry Timers on Devices Running Junos OS - Technical Documentation - Support - Juniper Networks,” 2018. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/concept/junos-sdn-openflow-flow-entry-timers-overview.html.

[3] Mininet Team, “Mininet overview.” [Online]. Available: http://mininet.org/overview/.

[4] Anon, “A,” 2018. [Online]. Available: https://en.wikipedia.org/wiki/software-defined_networking.

[5] NetworkStatic, “OpenFlow: Proactive vs Reactive,” Brent Salisbury’s Blog, 2018. [Online]. Available: http://networkstatic.net/openflow-proactive-vs-reactive-flows/. [Accessed: 04-Jun-2018].

[6] A. Greenberg et al., “A clean slate 4D approach to network control and management,” ACM SIGCOMM Comput. Commun. Rev., vol. 35, no. 3, p. 41, 2005.

[7] D. Maltz, G. Xie, and J. Zhan, Routing design in operational networks: A look from the inside. 2004.

[8] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: taking control of the enterprise,” Sigcomm ’07, 2007.

[9] N. McKeown et al., “OpenFlow: Enabling Innovation in Campus Networks,” ACM SIGCOMM Comput. Commun. Rev., 2008.

[10] S. Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, “NOX: Towards an Operating System for Networks,” ACM SIGCOMM Comput. Commun. Rev., pp. 105-110, 2008.

[11] A. Khurshid et al., “Veriflow: verifying network-wide invariants in real time,” in Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2013.

[12] P. Kazemian, G. Varghese, and N. McKeown, “Header space analysis: Static checking for networks,” in Presented as part of the 9th USENIX Symposium on Networked Systems Design and
Implementation (NSDI 12), 2012.

[13] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, “Flows Placement Problem in OpenFlow Networks: A Survey,” *IEEE Commun. Surv. Tutorials*, 2016.

[14] X. Jin et al., “Dynamic scheduling of network updates,” *ACM SIGCOMM Comput. Commun. Rev.*, 2014.

[15] J. Turner et al., “Supercharging PlanetLab - A high performance, multi-application, overlay network platform,” *Acm Sigcomm Comput. Commun. Rev.*, 2007.

[16] A. Voellmy, H. Kim, and N. Feamster, “Procera: a language for high-level reactive network control,” in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012.

[17] R. Moshref, M., Yu, M., Sharma, A. B., & Govindan, “vCRIB: Virtualized Flow Management in the Cloud,” in *HotCloud*, 2012.

[18] N. Kang, Z. Liu, J. Rexford, and D. Walker, “Optimizing the ‘one big switch’ abstraction in software-defined networks,” in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies - CoNEXT ’13*, 2013.

[19] A. A. R. Curtis, J. C. J. J. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: scaling flow management for high-performance networks,” *Proc. {ACM} {SIGCOMM} 2011 Conf. Appl. Technol. Archit. Protoc. Comput. Commun. Toronto, [ON], Canada, August 15-19, 2011*, 2011.

[20] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, “Infinite CacheFlow in software-defined networks,” in *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN ’14*, 2014.

[21] and N. R. S. A.S. Iyer, V. Mann, “SwitchReduce: Reducing switch state and controller involvement in OpenFlow networks,” *IFIP Netw. Conf.*, pp. 1-9, 2013.

[22] K. Agarwal, C. Dixon, E. Rozner, and J. Carter, “Shadow MACs: Scalable Label-switching for Commodity Ethernet,” *Proc. Third Work. Hot Top. Softw. Defin. Netw.*, 2014.

[23] E. Do Kim, S. I. Lee, Y. Choi, M. K. Shin, and H. J. Kim, “A flow entry management scheme for reducing controller overhead,” in *International Conference on Advanced Communication Technology, ICACT*, 2014.

[24] and D. L. Adam Zarek, Yashar Ganjali, “OpenFlow Timeouts Demystified,” *Dep. Comput. Sci. Univ. Toronto, Canada*, 2014.