Department of Computer Science Vanderbilt University

CS-8395-50

Computer Networking



Network Traffic Control using SDN.

**Rupak Mohanty**

**Contents**

1. [Introduction 3](#_TOC_250011)
   1. [Scope 3](#_TOC_250010)
   2. Technology Stack .4
2. Background Information 4
   1. What is RYU? --------------------------------------------------------------------------------------------5
   2. How does SDN Switch Work --------------------------------------------------------------------------6
   3. What is a flow table? ------------------------------------------------------------------------------------7
3. [Design and Architecture 8](#_TOC_250004)
4. [Implementation Enviornment 11](#_TOC_250003)
5. [Analysis of Results 1](#_TOC_250002)3
6. [Conclusion 1](#_TOC_250001)4
7. [References 1](#_TOC_250000)5

# Introduction

Software Defined Networking (SDN) is an innovative network architecture, which can be considered as an effective approach to implement load balancing and traffic control as compared to traditional networking.

Software-defined networking (SDN) is an approach to networking that uses open protocols like OpenFlow to control software at the edge of the network. It is used to control access to switches and routers. [SDN allows users to virtualize their hardware and manage network services from a centralized location](https://www.bing.com/aclk?ld=e87Sck6RygX0IxUmPav8QBajVUCUzXB_yfvO9Po2rO7xP9ue6lksq2xzqbo8Pm7jr0YPcIgAsbZ-gCRCczFwbLUz_uYzuOE-0cuwQpXEKp0T1G6YSoY7qzNfDqAvFVbQOZFRgqPOd7BkMxMOjamB1E6EpckLcsi_pAzrNS8HDB4M-6-jlZ&u=&rlid=11914ee01ed01a4ea91f2b84a0fa592e).

Some of the advantages of SDN are:

* Cost reduction: SDN does not require expensive hardware or complex configurations. [It can use existing network devices or low-cost switches2](https://www.serverwatch.com/guides/software-defined-networking-advantages/).
* Overhead reduction: SDN simplifies network management and isolation by using logical rather than physical connections. [It eliminates the need for VLANs and other separation methods2](https://www.serverwatch.com/guides/software-defined-networking-advantages/).
* [Agility: SDN enables network administrators to dynamically adjust network policies and traffic flows according to the changing needs of applications and users3](https://fidelus.com/software-defined-networking-advantages/).
* [Automation: SDN supports network automation by allowing network devices to communicate with each other and respond to events without human intervention4](https://frontom.com/advantages-of-sdn/).
* [Innovation: SDN facilitates the development and deployment of new network applications and services by providing a programmable and flexible platform](https://www.g2.com/articles/software-defined-networking)

# Scope

# 

Different from traditional networking architecture, SDN separates control plane from data plane, providing flexible pro- grammability and enhancing the central control capability as well as adaptability to cloud computing. Besides, some urgent and complicated networking problems has been resolved based on SDN approach [1].

In this project, we are required to construct a simple SDN network topology using a virtual network testbed named Mininet. Then, program two controller applications based on Ryu framework to implement the function of network traffic forwarding and redirection, respectively. Under the circumstance of modern websites that process huge number of requests in an extremely short period of time, overburden may occur leading to the crash of single server. Load balance can efficiently resolve this practical problem by distributing networking traffics to a set of servers. The redirection application developed redistributes traffic to another server that is not originally appointed by the client, which can be considered as a simplified version to implement load balancing.

The contributions of our work are listed below:

* First, we constructed an SDN network topology with one client and two servers (server1 and server2) connected by SDN switch controlled by an SDN controller in Mininet. The reachability of every node in the topology is checked and ensured.
* Second, we programed two controller applications using Ryu framework to realize traffic forwarding and redirection.
* Third, Wireshark is used to test the respective performance of two applications developed and compare the testing results.

A diagram of a network

Description automatically generated

# Technology Stack

# Linux environment (Ubuntu 20.04)

# Python2

# Python3

# Mininet (2.2.2)

# Ryu (4.34)

# Xterm (353)

# Wireshark

# Background Information’s

# A typical SDN architecture consists of three layers.

# Application layer: It contains the typical network applications like [intrusion detection](https://www.geeksforgeeks.org/intrusion-detection-system-ids/), [firewall](https://www.geeksforgeeks.org/introduction-to-firewall/), and [load balancing](https://www.geeksforgeeks.org/load-balancing-on-servers-random-algorithm/)

# Control layer: It consists of the SDN controller which acts as the brain of the network. It also allows hardware abstraction to the applications written on top of it.

# Infrastructure layer: This consists of physical switches which form the data plane and carries out the actual movement of data packets.

# A diagram of a software system Description automatically generated

In a traditional network, each [switch](https://www.geeksforgeeks.org/network-devices-hub-repeater-bridge-switch-router-gateways/) has its own data plane as well as the control plane. The control plane of various switches exchange [topology](https://www.geeksforgeeks.org/network-topologies-computer-networks/) information and hence construct a forwarding table that decides where an incoming data packet has to be forwarded via the data plane. Software-defined networking (SDN) is an approach via which we take the control plane away from the switch and assign it to a centralized unit called the SDN controller. Hence, a network administrator can shape traffic via a centralized console without having to touch the individual switches. The data plane still resides in the switch and when a packet enters a switch, its forwarding activity is decided based on the entries of flow tables, which are pre-assigned by the controller. A flow table consists of match fields (like input port number and packet header) and instructions. The packet is first matched against the match fields of the flow table entries. Then the instructions of the corresponding flow entry are executed. The instructions can be forwarding the packet via one or multiple ports, dropping the packet, or adding headers to the packet. If a packet doesn’t find a corresponding match in the flow table, the switch queries the controller which sends a new flow entry to the switch. The switch forwards or drops the packet based on this flow entry.

A diagram of a computer network

Description automatically generated A diagram of a distributed control plane

Description automatically generated

# 2.1 What is RYU?

Ryu is a component-based software defined networking framework. Ryu provides software components with well defined API that make it easy for developers to create new network management and control applications. Ryu supports various protocols for managing network devices, such as [OpenFlow](https://www.opennetworking.org/), Netconf, OF-config, etc

A diagram of a software application

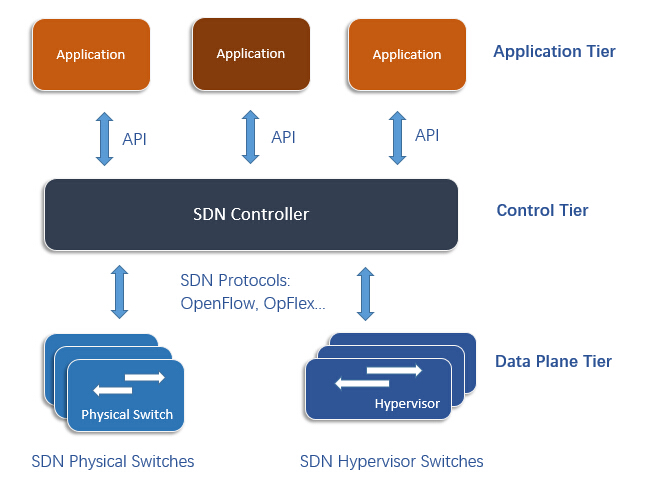
Description automatically generated

The network configuration function is divided into a data plane for handling data transfers and a control plane for handling control of the data plane. In the conventional approach, the two planes are tightly linked within individual network devices, so interworking between the control planes of the devices is necessary when constructing a single network with multiple network devices.

In the SDN approach, on the other hand, the data plane and control plane are separated, and their independence is maintained by specifying an application programming interface (API). Using the API to develop programs according to purpose and use enables higher independence of data plane control. A program that uses this kind of API is called an SDN application.

# How does an SDN Switch Work?

* + - Whether it is a virtual (hypervisor-based) switch or a physical switch, an SDN switch only keeps the data plane (packet forwarding) in itself.
    - The control plane (high-level routing) is decoupled from the SDN switch hardware but implemented in the SDN controller (an application running on the server or somewhere), which lies between network devices and applications.
    - An SDN switch consists of ports and tables. Packets arrive and leave the switch through ports. Tables consist of rows containing a classifier and set of actions.
    - In a SDN network when a switch receives a packet for which it cannot find a match in its flow table, it will forward this packet to the controller. The controller will decide about this packet and will forward this packet to the sender switch for further actions.
    - The controller can download a flow to the switch, which includes the first classifier that best matches the packet and the actions. Actions administer the treatment of the packet, which can be forwarding it to the port(s), encapsulating and forwarding it to the controller, dropping the packet, or sending it to the normal processing pipeline.
    - Once the flow is downloaded to the switch table, it will switch similar packets at wire speed.



# What is a flow table?

* SDN flow tables are the data structures that store the rules for packet processing in SDN switches.
* Each flow table entry consists of a match field and an action field.
* The match field specifies the criteria for matching packets, such as source and destination addresses, port numbers, protocol types, etc.
* The action field specifies what to do with the matched packets, such as forward, drop, modify, or send to the controller.

A diagram of process processing

Description automatically generated

A screen shot of a computer

Description automatically generated

When a packet arrives at an SDN switch, it is compared with the flow table entries in order of priority. The first matching entry determines the action to be taken on the packet. If no entry matches, the switch can either drop the packet or send it to the controller for further instructions.

SDN flow tables enable flexible and dynamic control of network devices by allowing the controller to install, modify, or delete rules remotely.

# Design and Architecture

In this section, we design basic network architecture and algorithms to implement network traffic redirection.

# Classical network traffic redirection

Network traffic redirection is one of the ubiquitous networks forwarding requirements in cloud computation environment. From the perspective of virtual machine migration, several works have been proposed. For example, F. Travostino et al. [2] presented to utilize the traditional tunnel technol- ogy to realize network redirection after VM migration. K. Onoue et al. [3] also invented a kind of data transponder which ensures the original IP communication between virtual machines during redirection in a single network. B. Borisaniya et al. [4] further installed transponders in all networks to achieve cross-zone redirection. On the other hand, other works are more specific on the redirection of Content Delivery Networks. For instance, T. Boros al. [5] proposed using proxy servers to implement network redirection through rewriting the domain name of original servers according to Domain Name System (DNS) protocol. M. Ghaznavi et al. [6] also used TCP REPAIR flag to dump the state of a current TCP. session and loaded it on the desired host to implement network redirection.

Our work can be distinguished from the above-mentioned papers, since we mainly focused on handling TCP and ICMP packets, and SDN is applied to network redirection.

# SDN-based network traffic redirection

SDN provides a new architecture for the network which improves the programmability of the network communication model. Y.pu et al. [7] proposed a topology migration model based on Open switch which realized the network topology migration by using the Open switch control agent based on Linux. Their research provides a significant hint for solving the network redirection in the cloud environment. C. Liu et al [8] also mentioned a new real-time traffic redirection method for edge network based on load balancing which optimizes the network traffic based on the statistics collected by controller. The aforementioned related work provided the theoretical basis for the implementation of network redirection of this project and helped us to clarify the methodology of building SDN network.

# Design

In this section, we design basic network architecture and algorithms to implement network traffic redirection.

A diagram of a sdn switch

Description automatically generatedSDN architecture is a creative network architecture which has become a new approach of network virtualization impel- mentation [9]. The design concepts of this project is highly dependent on the working principle of SDN network architecture which is the central challenge of algorithm design

**Fig. 1. forwarding function of SDN network architecture**

A diagram of a sdn switch

Description automatically generated

According to Fig. 1. and Fig. 2., we can abstract the components of the SDN network into four classes: client, server, SDN switch and SDN controller. The two servers are connected with the client through a single switch and SDN switch can communicate with SDN controller through a specific data path. The controller could determine the actions of each packet going through SDN switch by flow entries installed in the switch which will control the whole network traffic. In addition, each host connects with switch through a specific interface and a port number is allocated to each interface.

The actions of SDN controller can be distilled into two functions: adding flow entry and handling packet in packet.

Once a packet is sent to a server, the switch will first match the existing flow entries in the flow table. If each field matches the information of the packet, the packet will forward or redirect according to the rules. Otherwise, packet in packet will be sent to the SDN controller. After the SDN controller receives the packet, it will create flow entry. Then, the packet out packet will be sent to SDN switch and forwarded to the right output port according to the network rules provided by the network administrator. Finally, the flow entry will be installed in switch and actions provided by the controller will be executed (forwarding or redirection).

# The workflow solution

Although the functionality mainly focuses on adding flow entry and handling packet in packet. However, since the packet in handling is similar to creating flow entry, we will just concentrate on creating flow entry and installing flow entry in this subsection. Firstly, we will define the specific implementation flow of each functionality and then abstract out the specific functions each functionality needed. In addition, the communication between client and server are based on TCP connection. Before the establishment of TCP connection between client and server, several initialization work should be completed in order to ensure the normal running of the SDN network. First, of all the controller should allocate a data path id through which controller can send message to switch. Secondly, a table-miss flow entry should be installed to the switch in order to match packets that should be sent to the controller. Thirdly, an empty mac to port table should be defined to record the mapping relationship between the port number of SDN switch and MAC address of each host

# 

# Creating the flow entry

In advance of TCP connection between client and server, ARP request and reply should be sent in order to generate the mapping relationship between the port number and MAC address in mac to port table. This information will be used in network redirection task later. While the first TCP packet sent from client to the switch, its original destination is server1. Since there are no flow entries matching the packet, it will be sent to the controller according to the table-miss flow entry.

Once the packet enters into the SDN controller, the information in the network protocol of this packet will be extracted including source ip, destination ip and protocols above the network layer. After the controller determines the packet as a TCP packet and the in port is the port connecting to client, it will define the port connecting to server2 as out port and set an action including changing packet fields of out port, source ip and source mac which helps instruct such kind of packet to redirect to the correct direction. The match field will also be set up to help switch recognize such kinds of packets. In addition, the priority should also be set since all the flow entry should have higher priority than table-miss flow entry. Finally, the controller will create a flow entry which includes match, action, priority, and data path.

The process of creating flow entry of TCP packet from server2 to client is quite like the previous one. However, the source ip and source mac should be server1 since TCP is connection-oriented protocol. Actually, the SDN switch deceives the client that it is connected with server1.

# Installing the flow entry

# After the creation of flow entry, the controller will install it to switch through data path which is a bridge between switch and controller. However, idle time should be added as a field of flow entry which helps remove useless flow entry. Once the flow entry has been installed to switch, packets with the same properties will be redirected directly without triggering packet in event.

# Algorithm Design

Here are the kernel pseudo code of networking traffic redirection function. Algorithm1 is the pseudo code for redirecting client to server2 and algorithm2 is for redirecting server2 back to client.

A screenshot of a computer program

Description automatically generated A screenshot of a computer code

Description automatically generated

# Implementation Environment

# The physical host running virtual machine (VM) is WIN-CSFKGOF6J4Q having CPU 8-core AMD Ryzen 5 45000U with Radeon Graphics with speed 2.3GHz and 16 GB main memory with speed 2667 MHz. The operating system installed is Windows 11 Home. VM is created by VirtualBox, where the version is VirtualBox 7.0. VM has the same memory speed and CPU power as the physical host. However, the main memory of VM is 4 GB. Ubuntu-22.04 64-bit is the operating system of VM and Python 3 is installed to develop the application. The development tool is Pycharm Python IDE. All actual coding work is extended by the framework of a SDN controller software Ryu that has been developed using Python, where the version installed is Ryu 4.34. The Python module used is only Ryu in redirection application.

# 4.1Actual Implementation

Main steps to implement redirection function is represented by a program flow chart in figure 2. Actions will be set to only forward incoming packet to the correct out port when packet in event occurs, after which determine whether the out port existed in the mac to port table. If exists, check ethernet type field in the frame. If it is IP, further check the upper layer protocol. If TCP is detected, identify which port the packet comes in. If the incoming port connects with client, configure rules of match and action. If the incoming port connects with server2, set distinct rules of match and action. OOP is the programming skill exploited to develop this application since the Ryu framework uses the idea of OOP.

A diagram of a flowchart

Description automatically generated

# 

Only part of core Python code is shown below. The following code aims to implement match plus action when detecting TCP packet sent from client to server1. match = parser. OFPMatch() is used to set match rules in the flow entry. The most critical rules contain source and destination IP address. Header field of packet only contain IP and MAC address of server1 since client designates to send traffic to server1. Hence, parser. OFPActionSetField() is for modifying IP and MAC address in the packet header to the address of server2. Forwarding packet to the port that connects with server2 is implemented using parser.OFPActionOutput().

A screenshot of a computer program

Description automatically generated

The code below implements match plus action when detect- ing TCP packet replied by server2 to client. The flow entry set by match = parser.OFPMatch() is the same as the previous one. Client originally desires to send traffic to server1 and TCP is connection-oriented. Thus, parser.OFPActionSetField() is for replacing source IP and MAC address in the packet header with address of server1 so that client can receive the traffics. Then, parser.OFPActionOutput() messages switch which port to send packets out.

A screenshot of a computer program

Description automatically generated

# Analysis of Results

Tests are conducted in the same virtual machine (VM) and physical host where implementation is finished.

The whole test is carried out in Mininet of which the version is 2.3.0.dev6. The network packet analysis software used for testing performance is Wireshark 3.2.3.

# Functional Testing

A computer screen with white text

Description automatically generatedDuring the testing process, we simulated a simple SDN networking topology using Mininet. We first run networkTopo.py (the topology file) to initiate the virtual topology. Then run ryu forward.py (the controller applica- tions) on controller’s terminal to activate SDN network. To check whether each node in this topology connects with each other successfully, we use pingall to verify it. The snapshot is shown below which means The SDN topology we built can work as normal. Next, to check if the network can conduct TCP connection, do not need to close ryu-forward.py just

wait for 5 seconds. First run server.py on both server2’s and server1’s terminal and then run client.py on client’s terminal to check if all the packets are forwarded to Server1. The result shown below in the snapshot indicates that the TCP packets can be forwarded to server1 successfully.

A screenshot of a computer

Description automatically generated

# Performance Testing

Wireshark is used to capture network packets. Actually, we mainly focus on TCP segment. We run redirection and forwarding controller applications 10 times each and calculate the latency caused by TCP 3-way handshake (from the first SYN till the last ACK). Finally, we draw a curve diagram below to help us make a comparison. According to the diagram, except the sixth time, the time consumed of traffic redirection is larger than simple forward- ing. However, the only exception is caused by packet loss of SYN. Overall, the traffic network redirection will cause larger latency.

A graph with blue lines and orange dots

Description automatically generated

# Conclusion

# SDN provides flexibility and programmability to networking management, and it has gained more popularity in recent years. In this project, we initially analyzed the task requirement document. Then, we prototyped a SDN network topology and designed algorithms for completing forwarding and redirection functions. Furthermore, we programed two controller applications in the virtual machine because of algorithm devised previously to implement the functionality required. Ultimately, we tested the developed controller application. The results show that applications can work. The networking latency obtained by Wireshark provides the performance comparison between the two applications, which suggested the high efficiency of the controller programs.

# The redirection program developed can only distribute all traffics to another single server, which may result in failure of that server if excessive traffic occurs concurrently. Thus, future work can upgrade the functionality in redirection application, e.g., dynamically redistribute traffic to several servers.

# References

D. Tatang, F. Quinkert, J. Frank, C. Ro¨pke, and T. Holz, “Sdn-guard: Protecting sdn controllers against sdn rootkits,” in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 297–302.

F. Travostino, “Seamless live migration of virtual machines over the man/wan,” in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 290–es. [Online]. Available: <https://doi-org.ez.xjtlu.edu.cn/10.1145/1188455.1188758>

K. Onoue, Y. Oyama, and A. Yonezawa, “A virtual machine migration system based on a cpu emulator,” in *First International Workshop on Virtualization Technology in Distributed Computing (VTDC 2006)*, 2006, pp. 3–3.

B. Borisaniya and D. Patel, “Towards virtual machine introspection based security framework for cloud.” *Sa¯dhana¯: Published by the Indian Academy of Sciences*, vol. 44, no. 2, 2019.