

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỆ THỐNG THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**Quản trị mạng và hệ thống**

**ĐỀ TÀI**

**Tìm hiểu và triển khai phần mềm Open vSwitch**

**Giảng viên hướng dẫn:** ThS. Đỗ Hoàng Hiển

Sinh viên thực hiện:

Đỗ Thành Tôn                       20520814

Nguyễn Minh Tấn               20521877

Trần Kiến Quốc                       20520293

Nguyễn Lê Thảo Ngọc           21521191

TP. Hồ Chí Minh, tháng 12 năm 2023

NHẬN XÉT CỦA GIẢNG VIÊN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………… *ngày……...tháng……năm 2023*

Người nhận xét

*(Ký tên và ghi rõ họ tên)*

MỤC LỤC

[**Chương 1: Tổng quan về Open vSwitch 3**](#_heading=h.30j0zll)

[1.1. Open vSwitch là gì? 3](#_heading=h.1hmsyys)

[1.2 Tại sao lại là Open vSwitch 3](#_heading=h.41mghml)

[1.3 Kiến trúc của Open vSwitch 4](#_heading=h.2grqrue)

[1.5 Xử lý gói tin của Open vSwitch 6](#_heading=h.vx1227)

[1.6 Kết luận chương 1 7](#_heading=h.3fwokq0)

[**Chương 2: Các kịch bản về Open vSwitch 8**](#_heading=h.1v1yuxt)

[Phần1: Triển khai Routing in Open vSwitch **8**](#_heading=h.4f1mdlm)

[1.1 Mininet 8](#_heading=h.2u6wntf)

[1.2 Định tuyến trên Open vSwitch 8](#_heading=h.19c6y18)

[2.1.Topology 9](#_heading=h.3tbugp1)

[2.2 Routing trên switch PBA\_S1 và PBB\_S1 11](#_heading=h.28h4qwu)

[Phần2: Triển khai VLAN Trunking trong OpenvSwitch **14**](#_heading=h.nmf14n)

[1.1. VLAN 14](#_heading=h.37m2jsg)

[1.2.Trunking 14](#_heading=h.1mrcu09)

[2.1. Thiết kế và tạo mô hình mạng 15](#_heading=h.46r0co2)

[2.2. Triển khai mô hình bằng mininet 16](#_heading=h.2lwamvv)

[2.3. Kiểm tra mô hình đã tạo trong mininet 16](#_heading=h.111kx3o)

[2.4. Xác thực lại cấu hình đã thiết lập 20](#_heading=h.3l18frh)

[Phần 3: Triển khai ACL trong OpenvSwitch **24**](#_heading=h.206ipza)

[1 Giới thiệu 24](#_heading=h.4k668n3)

[Phần 4: Triển khai QoS trong OpenvSwitch **29**](#_heading=h.2zbgiuw)

[1. Giới thiệu 29](#_heading=h.1egqt2p)

[1.1. VLAN 29](#_heading=h.3ygebqi)

[2.Xác minh tỉ lệ lưu lượng truy cập mặc định 30](#_heading=h.2dlolyb)

[3. Cấu hình chính sách QoS cho Switch S1 32](#_heading=h.sqyw64)

[4. Configuring Single-Rate Two-Colors Traffic Policing 34](#_heading=h.3cqmetx)

[5. Verifying metering configuration 35](#_heading=h.1rvwp1q)

[6. QoS Shaping 37](#_heading=h.4bvk7pj)

[6.1 QoS shaping là gì? 37](#_heading=h.2r0uhxc)

[6.2 Cấu hình QoS shaping 38](#_heading=h.1664s55)

[6.3 Xác minh cấu hình QoS shaping 39](#_heading=h.3q5sasy)

TÀI LIỆU THAM KHẢO

Open vSwitch 3.2.90 Documentation, Nhiều tác giả, 2023

[An OpenVSwitch Introduction From NSRC](https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0ahUKEwiy6sCB_pXRAhWKnpQKHblDC2wQFgg-MAc&url=https%3A%2F%2Fnsrc.org%2Fworkshops%2F2014%2Fnznog-sdn%2Fraw-attachment%2Fwiki%2FAgenda%2FOpenVSwitch.pdf&usg=AFQjCNFg9VULvEmHMXQAsuTOE6XLH6WbzQ&sig2=UlVrLltLct2F_xjgnqZiOA)

# Chương 1: Tổng quan về Open vSwitch

## 1.1. Open vSwitch là gì?

Open vSwitch (OVS) là là một multilayer switch dưới dạng phần mềm được cấp phép theo giấy phép mã nguồn mở Apache 2.0. Mục tiêu của OVS là xây dựng một nền tảng switch chất lượng cao trong công nghiệp hỗ trợ các giao diện quản lý tiêu chuẩn và các chức năng truyền tải có thể điều khiển bằng lập trình.

OVS được thiết kế phù hợp để hoạt động như một switch ảo trong môi trường Virtual Machine (VM). Ngoài việc trực quan hóa các giao diện quản lý và các lớp mạng ảo, OVS còn tương thích với nhiều phiên bản của máy chủ vật lý, công nghệ ảo hóa như Linux-based, Xen/XenServer, KVM và VirtualBox.

Phần lớn mã nguồn được viết bằng ngôn ngữ C - độc lập với nền tảng và dễ dàng chuyển sang các môi trường khác. Bản phát hành hiện tại của Open vSwitch hỗ trợ các tính năng sau:

- VLAN tagging và chuẩn 802.1q trunking và access ports

- NIC bonding với LACP trên upstream switch

- NetFlow, sFlow®

- Chính sách và cấu hình QoS (Qualiuty of Service)

- Geneve, GRE, VXLAN,  ERSPAN, GTP-U, SRv6, Bareudp, STT và LISP tunneling

- Kiểm soát lỗi kết nối 802.1ag

- OpenFlow 1.0 với nhiều tiện ích mở rộng

- Cấu hình liên kết giữa C và Python

- Truyền tải hiệu năng cao với module Linux Kernel

Open vSwitch cũng có thể hoạt động hoàn toàn trong userspace mà không cần sự hỗ trợ từ mô-đun kernel. Việc triển khai không gian người dùng này sẽ dễ chuyển đổi hơn so với chuyển đổi dựa trên kernel. OVS trong userspace có thể truy cập các thiết bị Linux hoặc DPDK. Lưu ý Open vSwitch với đường dẫn dữ liệu không gian người dùng và các thiết bị không phải DPDK được coi là thử nghiệm và đi kèm với chi phí về hiệu suất.

**1.2 Tại sao lại là Open vSwitch**

Hypervisor cần có khả năng kết nối lưu lượng giữa các máy ảo và với thế giới bên ngoài. Trên các hypervisor dựa trên Linux sử dụng switch layer 2 tích hợp sẵn (Linux bridge), nhanh và đáng tin cậy. Vậy tại sao Open vSwitch lại được sử dụng? Câu trả lời là Open vSwitch được nhắm mục tiêu vào việc triển khai ảo hóa đa máy chủ, một bối cảnh mà các ngăn xếp giao thức trước đó không phù hợp. Những môi trường này thường được đặc trưng bởi các điểm cuối năng động cao, các đối tượng trừu tượng logic và (đôi khi) tích hợp với hoặc giảm tải cho phần cứng switch mục đích đặc biệt. Các đặc điểm thiết kế sau đây giúp Open vSwitch đáp dứng được các yêu cầu đó.

Trạng thái linh hoạt:

Tất cả các trạng thái mạng mà được liên kết với một thực thể mạng (ví dụ như VM) phải dễ dàng được nhận dạng và có thể di chuyển giữa các máy chủ khác nhau. Điều này bao gồm các “trạng thái mềm” truyền thống, ví dụ như các entry vào bảng Layer 2, Layer 3, các trạng thái định tuyến chính sách, ACL, QoS, cấu hình giám sáng (Netflow, sFlow), v.v. OVS có hỗ trợ cho cả cấu hình và thay đổi trạng thái mạng. Ví dụ: nếu một máy ảo di chuyển giữa các máy chủ đầu cuối, thì các trạng thái cấu hình như (SPAN, ACL, QoS) hay bất kỳ trạng thái nào khác phức tạp hơn đều có thể được chuyển đổi theo, ngoài ra, các trạng thái mạng của OVS được phát triển theo hướng tự động hóa khả lập trình.

Phản hồi tốt với thay đổi của mạng

Môi trường ảo thường có tần suất thay đổi lớn. Máy ảo đến và đi, di chuyển qua lại theo thời gian, v.v. OVS hỗ trợ một số tính năng cho phép hệ thống điều khiển mạng phản hồi và thích ứng khi môi trường thay đổi. Diều này bao gồm hỗ trợ tính toán và giao diện hiển thị đơn giản như NetFlow, IPFX và sFLOW. Ngoài ra, OVS hỗ trợ cả OVSDB có thể tương tác từ xa, do đó, các phần mềm quản lý có thể theo dõi nhiều khía cạnh của mạng và phản hồi nếu có sự thay đổi. Tính năng này được sử dụng rộng rãi hiện nay. OVS đặc biệt hỗ trợ OpenFlow như một phương thức để truy cập vào khả năng điều khiển lưu lượng.

Duy trì các Tag logic

Các thiết bị chuyển mạch ảo phân tán (chẳng hạn như VMware vDS và Cisco’s Nexus 1000V) thường duy trì ngữ cảnh logic trong mạng thông qua việc gắn hoặc thao tác các thẻ trong packet mạng. Điều này có thể được sử dụng để xác định duy nhất một máy ảo (theo cách chống giả mạo phần cứng) hoặc để giữ một số ngữ cảnh khác liên quan trong miền logic. Phần lớn vấn đề của việc xây dựng một chuyển mạch ảo phân tán là quản lý hiệu quả và chính xác các thẻ này.

OVS bao gồm nhiều phương pháp để chỉ định và duy trì các quy tắc gắn thẻ, tất cả đều có thể truy cập được đối với một quy trình điều phối từ xa. Hơn nữa, trong nhiều trường hợp, các quy tắc gắn thẻ này được lưu trữ ở một biểu mẫu được tối ưu hóa để chúng không phải kết hợp với thiết bị mạng nặng. Ví dụ, điều này cho phép hàng nghìn quy tắc gắn thẻ hoặc ánh xạ lại địa chỉ được cấu hình, thay đổi và di chuyển. Tương tự, Open vSwitch hỗ trợ triển khai GRE có thể xử lý hàng nghìn đường hầm GRE đồng thời và hỗ trợ cấu hình từ xa để tạo, cấu hình và chia nhỏ đường hầm, tính năng này có thể được sử dụng để kết nối các mạng VM riêng trong các trung tâm dữ liệu khác nhau

Tích hợp phần cứng

Forwarding path của Open vSwitch (in-kernel datapath) được thiết kế để có thể "giảm tải" quá trình xử lý gói cho các chipset phần cứng, cho dù được đặt trong chuyển mạch vật lý cổ điển hay trong end-host NIC. Điều này cho phép đường dẫn điều khiển Open vSwitch có thể kiểm soát cả việc triển khai phần mềm thuần túy hoặc chuyển mạch vật lý.

Có nhiều nỗ lực đang diễn ra để chuyển Open vSwitch sang chipset phần cứng. Chúng bao gồm nhiều chipset silicon thương mại (Broadcom và Marvell), cũng như một số nền tảng dành riêng cho nhà cung cấp. Phần “Chuyển” trong tài liệu thảo luận về cách tạo một cổng như vậy.

Ưu điểm của việc tích hợp phần cứng không chỉ là hiệu năng trong môi trường ảo hóa. Nếu các bộ chuyển mạch vật lý cũng hiển thị các phần tóm tắt điều khiển Open vSwitch, thì cả môi trường lưu trữ ảo hóa và vật lý đều có thể được quản lý bằng cách sử dụng cùng một cơ chế để điều khiển mạng tự động.

**1.3 Kiến trúc của Open vSwitch**

A diagram of a computer server

Description automatically generated

Open vSwitch thường được sử dụng để kết nối các VMs/containers trong một host. Ví dụ như trên OpenStack compute node, nó được sử dụng với vai trò là integration bridge để kết nối các VMs chạy trên Compute node đó. Nó quản lý cả các port vật lý (eth0, eth1) và các port ảo (ví dụ như tap port của các VMs). Ba khối thành phần chính của Open vSwitch được mô tả như trên hình:

- vswitchd:

+ Là ovs daemon chạy trên user space

+ Công cụ tương tác: ovs-appctl

- ovsdb-server:

+ Là database server của Open vSwitch chạy trên user space

+ Công cụ tương tác: ovs-vsctl, ovsdb-client

- kernel module (datapath):

+ Là module thuộc kernel space, thực hiện công việc chuyển tiếp gói tin

+ Công cụ tương tác: ovs-dpctl

**1.4 Các thành phần của OVS**

Deamon OVS:

‘ovs-vswitchd’ là daemon của Open vSwitch chạy trên userspace. Nó đọc cấu hình của Open vSwitch từ ovsdb-server thông qua kênh IPC (Inter Process Communication) và đẩy cấu hình xuống ovs bridge (là các instance của thư viện ofproto). Nó cũng đẩy trạng thái và thông tin thống kê từ các ovs bridges vào trong database.

A diagram of a bridge

Description automatically generated

Hình: ‘vswitchd’: daemon chính của ovs

OVSDB:

Nếu như những cấu hình tạm thời ví dụ như flows được lưu trong datapath và ‘vswitchd’ thì các cấu hình bền vững sẽ được lưu trữ trong ovsdb và vẫn lưu giữ khi sau khi khởi động lại hệ thống. Các cấu hình này bao gồm cấu hình về bridge, port, interface, địa chỉ của OpenFlow controller ()nếu sử dụng),...

‘ovsdb-server’ cung cấp giao diện RPC(remote procedure call) tới ovsdb. Nó hỗ trợ trình khách JSON-RPC kết nối tới thông qua passive TCP/IP hoặc Unix domain sockets.

‘ovsdb-server’ chạy hoặc như một backup server hoặc như một active server. Tuy nhiên chỉ có active server mới xử lý giao dịch làm thay đổi ovsdb.

Datapath:

Module chính chịu trách nhiệm chuyển tiếp gói tin trong Open vsWitch, triển khai trong kernelspace nhằm mục đích đạt hiệu năng cao. Nó caches lại OpenFlow flows và thực thi các action trên các gói tin nhận được nếu các gói tin nó match với một flow đã tồn tại. Nếu gói tin không khớp với bất kì flow nào thì gói tin sẽ được chuyển lên ovs-vswitchd. Nếu flow matching tại ‘vswitchd’ thành công thì nó sẽ gửi gói tin lại cho kernel datapath kèm theo các action tương ứng để xử lý gói tin đồng thời thực hiện cache lại flow đó vào datapath để datapath xử lý những gói tin cùng loại đến tiếp sau. Hiệu năng cao đạt được ở đây là vì thực tế hầu hết các gói tin sẽ match flows thành công tại datapath và do đó sẽ được xử lý trực tiếp tại kernelspace.

Phân loại datapath mà Open vSwitch hỗ trợ:

- Linux upstream: là datapath triển khai bởi module của nhân đi cùng với bản phát hành Linux.

- Linux OVS tree: là datapath triển khai bởi module của nhân phát hành cùng với cây mã nguồn của Open vSwitch. Một số tính năng của module này có thể không hỗ trợ với các kernel phiên bản cũ, trong trường hợp này, Linux kernel version tối thiểu sẽ được đưa ra để tránh bị biên dịch lỗi

- Userpace datapath: datapath cho phép xử lý và chuyển tiếp gói tin ở userspace, điển hình là DPDK.

- Hyper-V: hay còn gọi là Windows datapath

**1.5 Xử lý gói tin của Open vSwitch**

A diagram of a computer process

Description automatically generated

Hình: OVS Packet Handling

Open vSwitch là một phần mềm switch hỗ trợ OpenFlow

OpenFlow controller chịu trách nhiệm hướng dẫn cho datapath biết làm sao xử lý các loại gói khác nhau, hay còn gọi là flows. Một flow mô tả hành động mà datapath thực hiện để xử lý các gói tin của cùng một loại như thế nào, hay còn được gọi là action. Các kiểu hành động bao gồm chuyển tiếp tới port khác, thay đổi vlan tag,... Quá trình tìm kiếm flow khớp với gói tin nhận được gọi là flow matching.

Nhằm mục đích đạt hiệu năng tốt, một phần của flows được cache trong datapath, và phần còn lại ở ‘vswitchd’.

Một gói tin đi vào Open vSwitch datapath sau khi nó được nhận trên một card mạng. Nếu gói tin khớp với flow nào đó trong datapath thì datapath sẽ thực thi các actions tương ứng mô tả trong flow entry. Nếu không (flow missing), datapath sẽ gửi gói tin lên ovs-vswitchd và tiến trình flow-matching khác được xử lý tại đây. Sau khi ovs-vswitchd xác định làm sao để xử lý gói tin, nó gửi trả gói tin lại cho datapath cùng với yêu cầu xử lý. Đồng thời, vswitchd cũng yêu cầu datapath cache lại flow để xử lý các gói tin tương tự sau đó.

**1.6 Kết luận chương 1**

Theo nhiều cách, Open vSwitch nhắm đến một điểm khác trong không gian thiết kế so với các ngăn xếp mạng hypervisor trước đó, tập trung vào nhu cầu kiểm soát mạng động và tự động trong môi trường ảo hóa dựa trên Linux quy mô lớn.

Mục tiêu của Open vSwitch là giữ cho mã trong kermal càng nhỏ càng tốt (tối ưu cho hiệu suất) và sử dụng lại các hệ thống con hiện có khi có thể (ví dụ: Open vSwitch sử dụng ngăn xếp QoS hiện có). Kể từ Linux 3.3, Open vSwitch được bao gồm như một phần của kernel và đóng gói cho các tiện ích userspace có sẵn trên hầu hết các bản phát hành phổ biến.

**Chương 2: Các kịch bản về Open vSwitch**

**Phần1: Triển khai Routing in Open vSwitch**

**1 Giới thiệu**

**1.1 Mininet**

Mininet là một công cụ giả lập mạng, bao gồm tập hợp các hosts đầu cuối, các switches, routers và các liên kết trên một Linux kernel.

Mininet sử dụng công nghệ ảo hóa (ở mức đơn giản) để tạo nên hệ thống mạng hoàn chỉnh, chạy chung trên cùng một kernel, hệ thống và user code.

Các host ảo, switch, liên kết và các controller trên mininet là các thực thể thực sự, được giả lập dưới dạng phần mềm thay vì phần cứng.

Một host mininet có thể thực hiện SSH(Secure Shell) vào đó, chạy bất kì phần mềm nào đã cài trên hệ thống linux (môi trường mà mininet đang chạy). Các phần mềm này có thể gửi gói tin thông các ethernet interface của mininet với tốc độ liên kết và trễ đặt trước.

Mininet cho phép tạo topo mạng nhanh chóng, tùy chỉnh được topo mạng, chạy được các phần mềm thực sự như web servers, TCP monitoring, Wireshark; tùy chỉnh được việc chuyển tiếp gói tin.

Mininet cũng dễ dàng sử dụng và không yêu cầu cấu hình đặc biệt gì về phần cứng để chạy: mininet có thể cài trên laptop, server, VM, cloud (linux)

**1.2 Định tuyến trên Open vSwitch**

Bộ định tuyến chịu trách nhiệm phân phối các gói tin trên các mạng khác nhau. Nó yêu cầu một giao thức định tuyến để trao đổi thông tin định tuyến với các bộ định tuyến khác. Thông tin này thay đổi theo thời gian dựa trên cấu hình mạng. Nếu bất kỳ liên kết nào bị lỗi, bộ định tuyến có thể chọn một tuyến đường mới tới một mạng cụ thể. Thiết bị đầu cuối không thể giao tiếp trực tiếp với các thiết bị bên ngoài mạng cục bộ. Khi một máy chủ gửi một gói đến một máy chủ khác trên một mạng khác, gói đó sẽ được chuyển tiếp đến cổng mặc định, định tuyến lưu lượng truy cập ra khỏi mạng cục bộ.

Open vSwitch dựa vào giao thức OpenFlow để triển khai định tuyến. OpenFlow switches thực hiện chuyển tiếp gói bằng cách sử dụng chức năng packet-matching trong flow table.

Do đó, khi một gói đến switch, switch sẽ tra cứu trong flow table của nó và kiểm tra nếu có trung khớp. Sau cùng, switch sẽ quyết định hành động nào cần thực hiện dựa trên bảng flow table. Hành động đó có thể là:

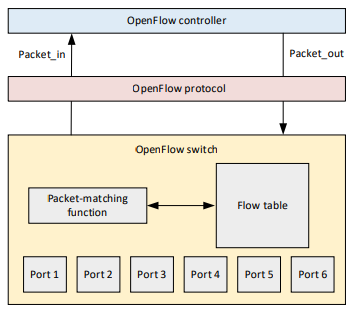
• Chuyển tiếp gói tin tới một cổng khác.

• Thả gói tin xuống.

• Chuyển gói tin tới bộ điều khiển.

Các luồng có thể được cài đặt thủ công với switch nếu không có bộ điều khiển nào được kết nối tới switch. Các luồng được cài đặt trong Open vSwitch (Open vSwitch vSwitchd) để điều khiển switch và triển khai giao thức OpenFlow. Câu lệnh ovs-ofctl là công cụ để giám sát và quản lý các switch được hỗ trợ bởi giao thức OpenFlow.

Mỗi flow table chứa một tập hợp các mục luồng bao gồm match fields, counters, và a set of instructions. Một

Open vSwitch có thể chứa nhiều flowtable. Switch bắt đầu khớp ở flow table đầu tiên và tiếp tục kiểm tra các flow table bổ sung để tìm cái hợp lệ. Theo mặc định, tất cả các flow table được lưu trữ trong bảng đầu tiên (table 0) nếu bảng số không được chỉ định cho một mục. Các gói tin khớp với các trường tiêu đề gói tin như switch input port, VLAN ID, Ethernet source/destination addresses, IP source/destination addresses, IP protocol, source/destination ports. Nếu một mục phù hợp được tìm thấy trong một bảng, các hướng dẫn liên quan đến mục nhập luồng cụ thể đó sẽ được thực thi.

Hình : Kiến trúc chuyển tiếp gói OpenFlow.

Hình cho thấy các chức năng cơ bản của một OpenFlow switch và mối quan hệ của nó với bộ điều khiển. Khi dữ liệu không khớp với gói tin đến, nó sẽ gửi packet\_in thông báo tới bộ điều khiển. Bộ điều khiển chạy các giao thức định tuyến và chuyển mạch và logic khác để xác định các bảng chuyển tiếp và logic trong dữ liệu. Do đó, khi bộ điều khiển có gói dữ liệu để chuyển tiếp qua bộ chuyển mạch, nó sẽ sử dụng OpenFlow packet\_out message. Flow entry sau đó được lưu trữ trong flow table nằm ở switch. Nếu không có bộ điều khiển nào được kết nối với switch, switch sẽ tra cứu trong flow table và thực hiện hành động dựa trên các flow entry được lưu trữ thủ công trong bộ switch. Nếu như không có sự trùng khớp nào trong flow table, switch sẽ loại bỏ gói tin.

**2.Triển khai**

**2.1.Topology**

A diagram of computer network

Description automatically generated

Setting

| Device | Interface | IP Address | Subnet | Default gateway |
| --- | --- | --- | --- | --- |
| KT\_PC\_A | KT\_PC\_A-eth1 | 192.168.1.10 | /24 | 192.168.1.1 |
| KD\_PC\_A | KD\_PC\_A-eth1 | 192.168.2.11 | /24 | 192.168.2.1 |

Đầu tiên, chúng ta dùng lệnh sudo ovs-vsctl show để hiển thị cấu hình Open vSwitch (OVS) trên hệ thống. Lệnh sẽ cung cấp cái nhìn toàn diện về cấu hình hiện tại của OVS, bao gồm thông tin về cầu nối, cổng và các cài đặt khác.

A screen shot of a computer

Description automatically generated

Tiếp đến dùng lệnh sudo ovs-ofctl dump-flows PBA\_S1  để in ra bảng điều khiển tất cả các luồng trong bảng của switch khớp với các luồng và flow table tại thời điểm này thì trống

A black and orange text

Description automatically generated

Kiểm tra kết nối giữa máy chủ KT\_PC\_A và máy chủ KD\_PC\_A bằng lệnh ping.

A screenshot of a computer program

Description automatically generated

Kết quả hiện thị không có kết nối giữa các máy chủ vì switch hiện tạikhông biết cách xử lý các luồng.

Địa chỉ IP là gateway(192.168.1.1) của máy chủ KT\_PC\_A và (192.168.2.1) là của máy chủ KD\_PC\_A



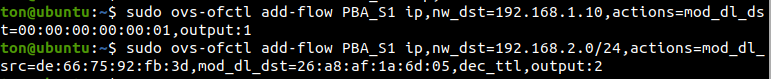
**2.2 Routing trên switch PBA\_S1 và PBB\_S1**

Trong phần này, ta sẽ cấu hình định tuyến trong switch PBA\_S1. Mục tiêu là sẽ cấu hình switch sao cho lưu lượng truy cập đến từ máy chủ KT\_PC\_A sẽ được gửi tới switch PBA\_S1. Ngoài ra, bạn sẽ thêm các luồng do đó switch PBA\_S1 sẽ chịu trách nhiệm chuyển bất kỳ dữ liệu nào nhận được từ switch PBB\_S1 tới switch đích yêu cầu (máy chủ KT\_PC\_A).

Thêm flow entry vô flow table của switch thông qua sudo ovs-ofctl add-flow



Lệnh thêm một luồng cái mà sẽ gửi yêu cầu ARP đến tất cả các cổng của switch.



Dòng 1: là lệnh thêm 1 flow entry cho switch PBA\_S1 sẽ cho lưu lượng truy cập đến máy chủ KT\_PC\_A(192.168.1.10).

Bất cứ khi nào switch PBA\_S1 nhận được bất kỳ lưu lượng truy cập nào đi đến máy chủ đích KT\_PC\_A(192.168.1.10). lưu lượng sẽ được chuyển tiếp đến cổng PBA\_S1 - eth1. “mod\_dl\_dst” có nhiệm vụ thay đổi địa chỉ MAC thành địa chỉ MAC chính xác của máy chủ PBA\_S1 (00:00:00:00:00:01)

Dòng 2: là lệnh thêm 1 flow entry cho switch PBA\_S1 cho phép lưu lượng truy cập đến mạng đích 192.168.2.0/24.

Bất cứ khi nào switch PBA\_S1 nhận được lưu lượng đi đến đích mạng 192.168.2.0/24, lưu lượng sẽ được chuyển tiếp đến cổng PBA\_S1 -eth2. Nguồn và MAC đích sẽ được thay đổi thành de:66:75:92:fb:3d (PBA\_S1 -eth2) và 26:a8:af:1a:6d:05 (PBB\_S1 -eth2) và giá trị TTL sẽ giảm đi một.

Và xác nhận lại các cấu hình bằng cách kiểm tra flowtable của switch bằng lệnh

sudo ovs-ofctl dump-flows PBA\_S1

A computer screen with white text

Description automatically generated

Theo như hình nhận thấy tất cả các quy trình thủ công đã được cài đặt trong flow table.

Làm tương tự cho PBB\_S1, ta được flow table như dưới

A screen shot of a computer code

Description automatically generated

Giờ đây có thể kiểm tra kết nối giữa 2 máy chủ KT\_PC\_A(192.168.1.10) và KD\_PC\_A(192.168.2.11)

A screenshot of a computer program

Description automatically generated

Có thể sử dụng một công cụ khác do Open vSwitch cung cấp để theo dõi gói tin như khi nó đi qua switch. ovs-appctl là công cụ theo dõi có thể được sử dụng để xác định những gì đang xảy ra với các gói khi chúng trải qua quá trình xử lý dữ liệu (ví dụ: đã sửa đổi trường, cổng đầu ra, v.v.).

sudo ovs-appctl ofproto/trace KT\_PC\_A in\_port=1,ip,nw\_src=192.168.1.10,nw\_dst=192.168.2.11,dl\_src=00:00:00:00:00:01,nw\_ttl=64

Input port: 1

Network layer protocol: IP

Source IP address: 192.168.1.10

Destination IP address: 192.168.2.11

Source MAC address: 00:00:00:00:00:01

Time-to-live (TTL): 64

A computer screen with white text

Description automatically generated

Hộp màu xám đầu tiên hiển thị các giá trị của các trường của flow trước khi flow được gửi đến switch. Lưu ý rằng nếu một trường không được chỉ định (ví dụ: dl\_src hoặc địa chỉ MAC nguồn), công cụ sẽ sử dụng giá trị 0 cho trường đó. Các giá trị ở đây khớp với những giá trị được người dùng chỉ định trong lệnh.

Hộp màu xám thứ hai hiển thị quy tắc phù hợp trong flowtable.

Hộp màu xám thứ ba hiển thị các giá trị của các trường của flow cuối cùng sau khi gói được xử lý trong switch.

**Phần 2: Triển khai VLAN Trunking trong OpenvSwitch**

**1. Giới thiệu**

**1.1. VLAN**

Ở cấp độ đơn giản nhất, VLAN (viết tắt của “mạng LAN ảo”) là một cách để phân vùng một switch thành nhiều switch.

Ví dụ: giả sử bạn có hai nhóm máy, nhóm A và nhóm B. Bạn muốn các máy trong nhóm A có thể nói chuyện với nhau và các máy trong nhóm B có thể nói chuyện với nhau. Tuy nhiên bạn không muốn các máy trong nhóm A có thể nói chuyện với các máy trong nhóm B. Bạn có thể thực hiện việc này bằng mô hình mạng có 2 switch, nhóm A sẽ kết nối với 1switch và nhóm B sẽ kết nối với switch còn lại.

Nhưng nếu bạn chỉ có một switch thì bạn phải sử dụng Vlan để thực hiện điều tương tự, bằng cách định cấu hình các port cho các máy trong nhóm A làm “access port” cho cùng một Vlan và các port cho các máy của nhóm B làm “access port” cũng cho cùng một Vlan nhưng Vlan này phải nhất định khác với vlan của nhóm A . Switch sẽ chỉ chuyển tiếp các gói giữa các cổng được gán cho cùng một VLAN , vì thế dù chỉ có 1 switch nhưng nó sẽ hoạt động như 2 switch độc lập một cách hiệu quả, mỗi VLAN sẽ ứng với 1 switch độc lập này.

**1.2.Trunking**

Giả sử mỗi switch có 3 vlan như hình bên dưới:

A diagram of a computer network

Description automatically generated

Với các host thuộc cùng vlan nhưng lại kết nối với switch khác nhau thì để chúng có thể liên lạc được với nhau chúng ta phải tạo cho chúng một đường kết nối riêng. Thế nhưng nếu có quá nhiều vlan và nhiều switch trong mô hình thì cách làm này sẽ gây ra nhiều lãng phí và không hiệu quả.

Do đó để giải quyết vấn đề này chúng ta thường dùng trunking. Đây là 1 kỹ thuật cho phép dữ liệu của các VLAN (có thể khác nhau) cùng lưu thông qua một đường, đường này được gọi là đường trunk.

Vì kỹ thuật này cho phép dùng chung một kết nối vật lý cho dữ liệu của các VLAN đi qua nên dể phân biệt được chúng là dữ liệu của VLAN nào, người ta gắn vào các gói tin một dấu hiệu gọi là “tagging”. Hay nói cách khác là dùng một kiểu đóng gói riêng cho các gói tin di chuyển qua đường “trunk” này. Giao thức được sử dụng là 802.1Q (dot1 q).

Do đó khi dùng kỹ thuật này phải lưu ý số VLAN hoàn toàn tùy ý nhưng chúng ta phải đảm bảo rằng một Vlan nhất định phải được đánh tag nhất quán trên toàn mạng và các Vlan khác nhau phải được đánh tag khác nhau. (Điều đó nói lên rằng, Vlan 0 thường đồng nghĩa với gói không có tiêu đề Vlan và Vlan 4095 được bảo lưu).

**2. Thực nghiệm**

**2.1. Thiết kế và tạo mô hình mạng**

Mô hình gồm 2 switch và 4 host

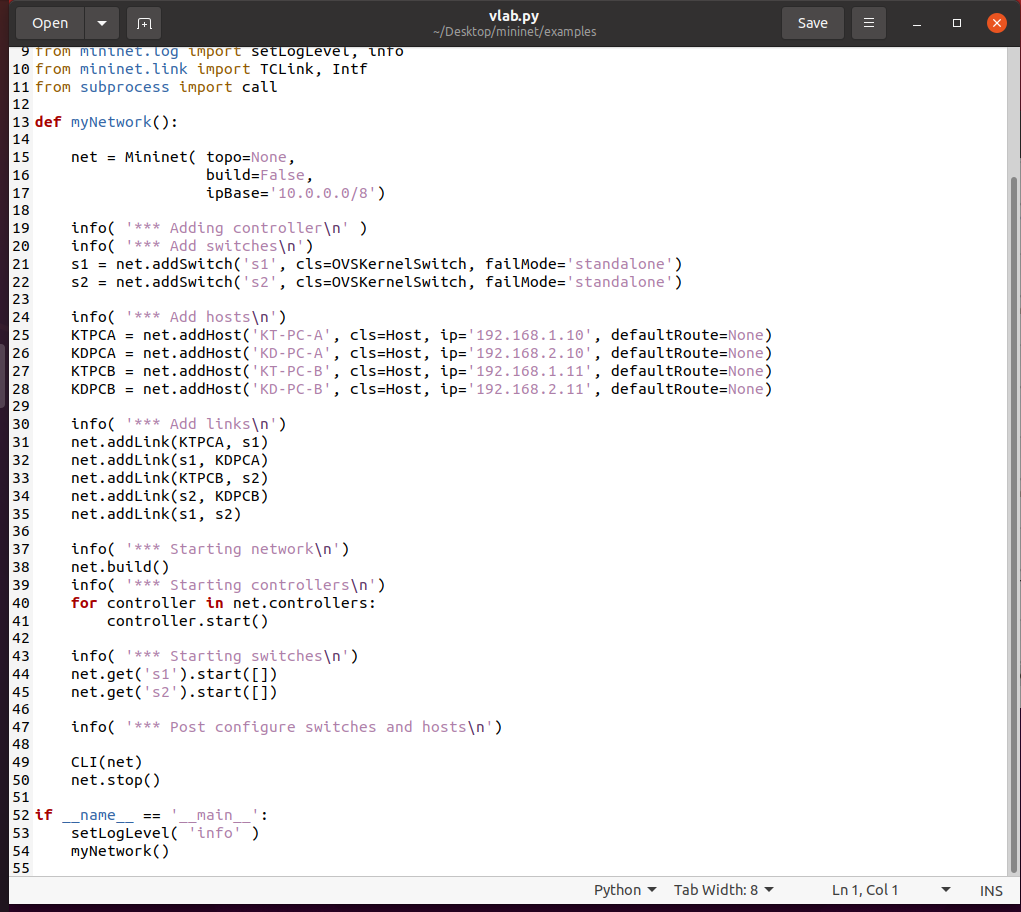
Ta sẽ cấu hình các thiết bị theo bảng bên dưới:

| Host | Interface | IP Address | Subnet | VLAN |
| --- | --- | --- | --- | --- |
| KT-PC-A | KT-PC-A -eth0 | 192.168.1.10 | /24 | 10 |
| KD-PC-A | KD-PC-A -eth0 | 192.168.2.10 | /24 | 20 |
| KT-PC-B | KT-PC-B -eth0 | 192.168.1.11 | /24 | 10 |
| KD-PC-B | KD-PC-B -eth0 | 192.168.2.11 | /24 | 20 |

A diagram of computer network

Description automatically generated

Chúng ta sẽ dùng ngôn ngữ python để viết chương trình tạo các host, switch và cấu hình thiết bị (IP, tên,…) tương ứng cho chúng



**2.2. Triển khai mô hình bằng mininet**

Thực hiện lệnh:

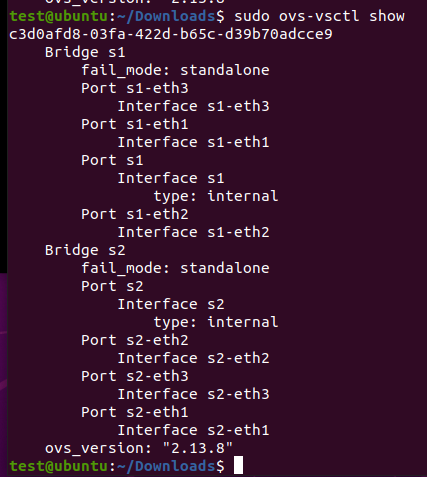
sudo python3{ten\_file\_thuc\_thi}

Để xây dựng mô hình mạng đã thiết kế trong mininet

A computer screen shot of a black background

Description automatically generated

Chúng ta dùng lệnh sudo ovs-vsctl show để hiển thị cấu hình Open vSwitch (OVS) trên này hệ thống. Lệnh này sẽ cung cấp cái nhìn toàn diện về cấu hình hiện tại của OVS, bao gồm thông tin về cầu nối, cổng cũng như các cài đặt khác (nếu có)



**2.3. Kiểm tra mô hình đã tạo trong mininet**

-Thực hiện lệnh để kiểm tra các kết nối và tên các interface nối với nhau

A screenshot of a computer code

Description automatically generated

*Giải thích kết quả hiện ra:*

 KT-PC-A-eth0<->s1-eth1 (OK OK)

*Ý nghĩa:*

 KT-PC-A(ứng với port eth0) kết nối thành công tới switch s1 tại port eth1 cuả switch.

Và ngược lại s1 (ứng với port eth1) kết nối thành công tới KT-PC-A tại port eth10 của host này.

-Thực hiện lệnh để kiểm tra IP của các host

Dùng lệnh xterm để mở 1 terminal của host. Sau đó dùng lệnh ifconfig để kiểm tra các thông tin địa chỉ của host này

+Host KT-PC-A

Mở terminal của KT-PC-A



Sau khi terminal hiện ra, nhập lệnh ifconfig để kiểm tra xem địa chỉ ip của KT-PC-A có đúng với thiết kế không.

A screenshot of a computer screen

Description automatically generated

+Thực hiện lại cho host KT-PC-B:

Mở terminal của KT-PC-B



Kiểm tra IP

A screenshot of a computer

Description automatically generated

+Tương tự cho KD-PC-A

Mở terminal của KD-PC-A



Thông tin IP của KD-PC-A

A screenshot of a computer screen

Description automatically generated

+Tương tự cho KD-PC-B

Mở terminal của KD-PC-B

A screenshot of a computer

Description automatically generated

Thông tin IP của KD-PC-B

A screenshot of a computer

Description automatically generated

**4 . Cấu hình VLANs và Trunking cho các thiết bị**

Sau khi kiểm tra mô hình bằng lệnh links, chúng ta biết được tên các port của mỗi thiết bị trong mô hình như sau:

A diagram of a computer network

Description automatically generated

Tiếp theo, chúng ta tiến hành dùng OpenvSwitch để cấu hình Vlan và tạo đường trunk cho mô hình mạng đã thiết kế.

-Thực hiện lệnh:

sudo ovs-vsctl set port <interface\_name> tag=<vlan-id>

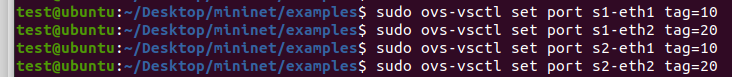
Để cấu hình port làm access port và gán port của switch vào VLAN tương ứng

-Thực hiện lệnh:

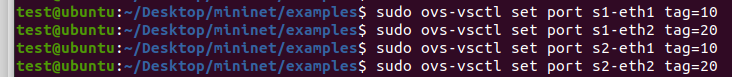
sudo ovs-vsctl set port {switchport\_name} trunk {VLAN1, VLAN2,….}

Để định cấu hình port làm trunk port. Khi các interface được đặt ở chế độ trunk, lưu lượng thuộc các Vlan  khác nhau vẫn có thể đi qua. Theo mặc định, OpenvSwitch chuyển tiếp tất cả lưu lượng VLAN. Dó đó nếu sử dụng lệnh trunk ở phía sau thì bạn sẽ hạn chế  swtich sao cho nó chỉ chuyển tiếp lưu lượng cho các Vlan đã chỉ định.

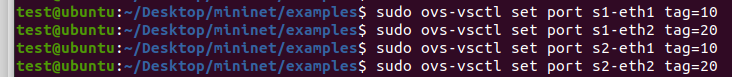
+Gắn port s1-eth1(port kết nối tới KT-PC-A) với VLAN10



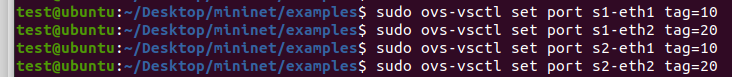
+Gắn port s1-eth2(port kết nối tới KD-PC-A) với VLAN20



+Gắn port s2-eth1(port kết nối tới KT-PC-B) với VLAN10



+Gắn port s2-eth2(port kết nối tới KD-PC-B) với VLAN20



+Cấu hình port eth3 của switch1 làm trunk port và chỉ định switch 1 chỉ cho phép lưu lượng thuộc VLAN 10 và VLAN 20 đi qua nó



+ Cấu hình port eth3 của switch2 làm trunk port và chỉ định switch 2 chỉ cho phép lưu lượng thuộc VLAN 10 và VLAN 20 đi qua nó



**2.4. Xác thực lại cấu hình đã thiết lập**

**a) Kiểm tra thông tin cấu hình bằng OpenvSwitch**

Thực hiện lại lệnh:  sudo ovs-vsctl show

Để xem cấu hình đã được thiết lập như mong muốn hay chưa.

A screenshot of a computer program

Description automatically generated

Chúng ta có thể thấy, port eth3 của switch s1 và switch s2 đã hoạt động như 1 trunk port. Các port còn lại thì đều được gán vào vlan tương ứng như mô hình đã thiết kế.

**b) Thực hiện ping giữa các host**

Trong mininet, chúng ta sẽ mở terminal của mỗi host bằng lệnh: xterm {ten\_host}

Để tiến hành ping từ host này tới các host khác trong mạng

+KT-PC-A ping tới KD-PC-A

=> Không thành công vì 2 host này khác vlan

A screenshot of a computer

Description automatically generated

+KT-PC-A ping tới KT-PC-B

=> Ping thành công vì 2 host này đều cùng thuộc vlan10

A screenshot of a computer

Description automatically generated

+KT-PC-A ping tới KD-PC-B

=> Không thành công vì 2 host này khác vlan

A screen shot of a computer screen

Description automatically generated

+KD-PC-A ping tới KT-PC-B

=> Không thành công vì 2 host này khác vlan

A screenshot of a computer program

Description automatically generated

+KD-PC-A ping tới KD-PC-B

=> Ping thành công vì 2 host này đều cùng thuộc vlan20

A screenshot of a computer program

Description automatically generated

+KT-PC-B ping tới KD-PC-B

=> Không thành công vì 2 host này khác vlan

A screenshot of a computer program

Description automatically generated

**Phần 3: Triển khai ACL trong OpenvSwitch**

**1 Giới thiệu**

**Tổng quan ACL**

ACL (Access Control List) được sử dụng như bộ lọc để kiểm soát lưu lượng mạng. Nó thực hiện lọc gói dựa trên cái chuỗi câu lệnh tuần tự ACE (Access Control Entry). ACL có tác dụng hạn chế truy cập và thiết bị của người dung vào mạng hoặc cho phép một máy chủ truy cập vào một phần mạng va ngăn chặn máy chủ khác truy cập vào vào cùng khu vực.

Các tiêu chí của ACL bao gồm: Ethernet source/destination addresses, IP source/destination addresses, IP protocol, source/destination ports.

Router được cấu hình ACL sẽ so sánh các header của gói tin từ ACE đầu tiên và tuần tự đến hết. Khi thực hiện khớp, bộ định tuyến sẽ thực hiện hướng dẫn, cho phép hoặc từ chối gói tin, sau khi match với ACE đầu tiên các ACE còn lại sẽ không được phân tích.

Nếu souce IP address không khớp với bất kỳ ACE nào, gói tin sẽ bị loại bỏ.

Topology

A computer network diagram with a computer and a router

Description automatically generated

Trong hình có 3 host được kết nối với router R1, ACL được cấu hình trong bộ định tuyến R1 để từ chối kết nối từ host PC-C đến host PC-A và cho phép mọi lưu lượng truy cập khác

Table 1: ACL table.

| Source | Destination | Protocol | Access |
| --- | --- | --- | --- |
| 192.168.7.30 | 192.168.7.10 | any | Deny |
| Any | Any | Any | Permit |

Xem xét bảng 1: Bảng này chứa các ACL đang chạy trong router R1, nếu PC-C (192.168.7.30) tạo lưu lượng truy cập cho máy chủ đích PC-A, router R1 sẽ phân tích bảng ACL và khớp với ACE đầu tiên, gói tin sẽ bị loại bỏ. Nếu máy chủ PC-B muốn giao tiếp với máy chủ PC-A, ACE thứ hai sẽ khớp

Thứ tự ACE trong ACL là cực kỳ quan trọng vì nó được so sánh từ đầu bảng tới cuối một cách tuần tự, thường các ACE từ chối hoặc chấp nhận các dịch vụ được xếp sau cùng. Trong ví dụ trên nếu đảo ngược thứ tự hai ACE với nhau thì PC-A và PC-C vẫn có thể liên lạc với nhau.

Triển khai ACL trong Open vSwitch

Trong một tường lửa dựa trên giao thức OpenFlow, các quy tắc được config sẵn trên Switch’s flow table. Mỗi gói tin được kiểm tra theo quy tắc của tường lửa từ mức ưu tiên cao nhất đến thấp nhất.

A diagram of a computer

Description automatically generated

Trong hình có 3 host được kết nối với Open vSwitch, s1. Switch s1 sẽ hoạt động như tưởng lửa để lọc gói tin. Nếu host PC-C tạo lưu lượng cho máy chủ đich PC-A, sw-s1 sẽ phân tích bảng ACL và ngăn chặn lưu lượng. Tất cả lưu lượng còn lại được cho phép

Table 2: Open vSwitch ACL table.

| Priority | Table | Source | Destination | Protocol | Action |
| --- | --- | --- | --- | --- | --- |
| 40000 | 0 | 192.168.7.30 | 192.168.7.10 | Any | Drop |
| 32768 | 0 | Any | Any | Any | Goto\_table=1 |

Table 3: Open vSwitch Forwarding table.

| Priority | Table | Source | Destination | Protocol | Action |
| --- | --- | --- | --- | --- | --- |
| 32768 | 1 | Any | Any | Any | Normal |

Xem xét bảng 3 và 4: Các bảng chứa quy tắc tưởng lửa trong sw-s1. Nếu host PC-C muốn giao tiếp với host PC-A, sw-s1 sẽ phân tích bảng dựa trên mức độ ưu tiên và lưu lượng truy cập sẽ bị loại bỏ. Đối với các lưu lượng khác, khớp với mục ưu tiên thứ 2 và switch sẽ kiểm tra bảng 1 để chuyển tiếp gói tin.

Các câu lệnh và mô hình triển khai trên hệ điều hành ubuntu

Topology dựa trên hình 2 và các bước khởi tạo sẽ nằm trong phần demo.

A diagram of a computer

Description automatically generated

File topo.py khởi tạo topology theo hình 2

A screen shot of a computer program

Description automatically generated

Đầu tiên, chúng ta dùng lệnh sudo ovs-vsctl show để hiển thị cấu hình Open vSwitch (OVS) trên hệ thống. Lệnh sẽ cung cấp cái nhìn toàn diện về cấu hình hiện tại của OVS, bao gồm thông tin về cầu nối, cổng và các cài đặt khác.

A computer screen shot of a program

Description automatically generated

Tiếp đến sử dụng lệnh để tạo 2 bảng ACL table 0 và table 1 chứa các ACE

table 0:

sudo ovs-ofctl add-flow s1 table=0.priority=40000,ip,nw\_src=192.168.7.30,nw\_dst=192.168.7.10,action=drop

sudo ovs-ofctl add-flow s1 table=0.priority=32768,action=goto\_table=1

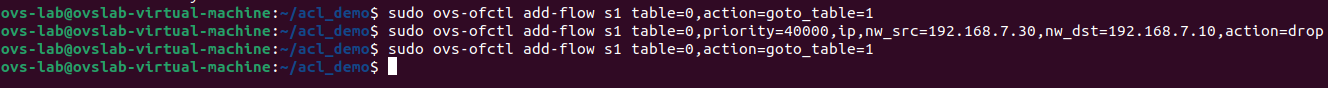


Table 1:

sudo ovs-ofctl add-flow s1 table=1,action=normal



Kiểm tra cấu hình bằng lệnh

sudo ovs-ofctl dump-flows s1

A computer screen shot of text

Description automatically generated

Kiểm tra kết nối giữa PC-A và PC-B sau khi cấu hình

A screenshot of a computer

Description automatically generated

PC-A và PC-B vẫn kết nối bình thường theo ACE thứ 2.

A screenshot of a computer

Description automatically generated

Kết nối giữa PC-A và PC-C bị chặn theo ACE thứ nhất.

A screenshot of a computer

Description automatically generated

Kết nối giữa PC-B và PC-C diễn ra bình thường theo ACE thứ 2.

Sử dụng công cụ ovs-appctl cung cấp bởi Open vSwitch để theo dõi gói tin khi nó đi qua switch.

Trường hợp 1:

* Network layter protocol: IP
* Source IP address: 192.168.7.10
* Destination Ip address 192.168.7.30
* Source MAC address: 00:00:00:00:00:01
* Destination MAC address: 00:00:00:00:00:03

A black rectangular frame with white dots

Description automatically generated with medium confidence

Xem xét hình : Ta thấy có sự trùng khớp với quy tắc thứ hai ở table 0 và switch được hướng dẫn để kiểm tra table 1. Từ table 1 ,switch chọn hoạt động bình thường và gói tin được chuyển tiếp, vậy cho thấy việc ping từ PC-A đến PC-C là có sự chuyển tiếp gói tin đến PC-C nhưng PC-A không nhận được phản hồi từ PC-C do ACE chặn giao tiếp từ nguồn PC-C đến PC-A, xem xét them trường hợp 2 để hiểu rõ.

Trường hợp 2:

* Network layter protocol: IP
* Source IP address: 192.168.7.30
* Destination Ip address 192.168.7.10
* Source MAC address: 00:00:00:00:00:03
* Destination MAC address: 00:00:00:00:00:01

A black background with white text

Description automatically generated with medium confidence

Xem xét hình : Ta thấy có sự trùng khớp với quy tắc đầu tiên của table 0 nên switch loại bỏ ACE còn lại, tiến hành chặn lưu lượng từ nguồn PC-C đến đích PC-A.

**Phần 4: Triển khai QoS trong OpenvSwitch**

**1. Giới thiệu**

QoS là từ viết tắt của Quality of Service, là một khái niệm chỉ ra tỷ lệ giữa dữ liệu mà một hệ thống mạng đã nhận và dữ liệu mà hệ thống đã gửi. Nó được sử dụng để đánh giá tính ổn định và hiệu quả của một hệ thống mạng hoặc dịch vụ mạng cụ thể.

QoS có thể được xem như một hệ thống để đảm bảo rằng các dịch vụ mạng được cung cấp một cách đồng bộ và hiệu quả, nhất là trong các môi trường mạng phức tạp với lưu lượng dữ liệu cao.

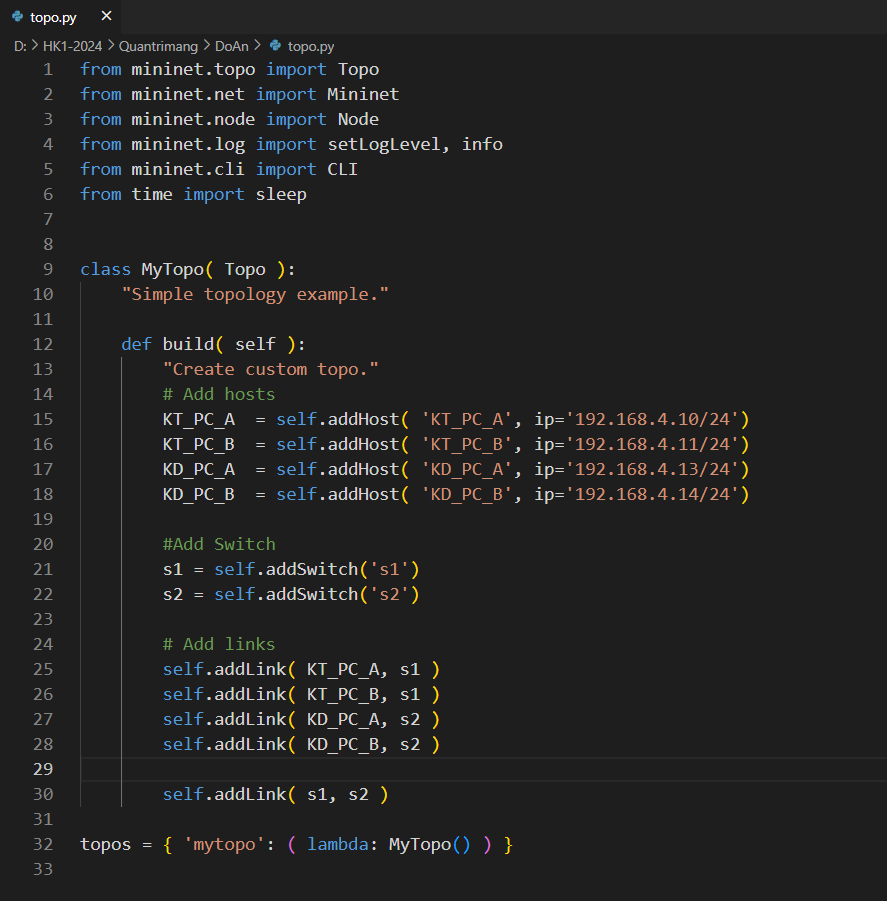
**1.1. VLAN**

Topology

A screenshot of a computer

Description automatically generated

Tạo các host và cấu hình địa chỉ IP



Ta nhập lệnh sau để chạy topology và dùng lệnh “links” kiểm tra kết nối:

Sudo mn  --custom ./topo.py --topo mytopo

A screenshot of a computer program

Description automatically generated

2.Xác minh tỉ lệ lưu lượng truy cập mặc định

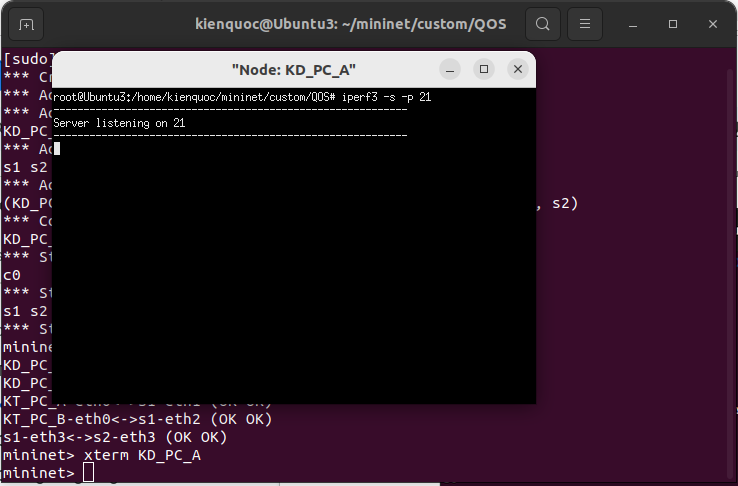
B1: Mở terminal KD\_PC\_A bằng cách nhập lệnh sau:

Xterm KD\_PC\_A

A screenshot of a computer

Description automatically generatedSau đó nhập lệnh sau để sử dụng cổng 21 dưới dạng máy chủ FTP:

Iperf3 -s -p 21



B2: Mở terminal của KT\_PC\_A bằng cách nhập lệnh:

 xterm KT\_PC\_A

A screenshot of a computer program

Description automatically generated

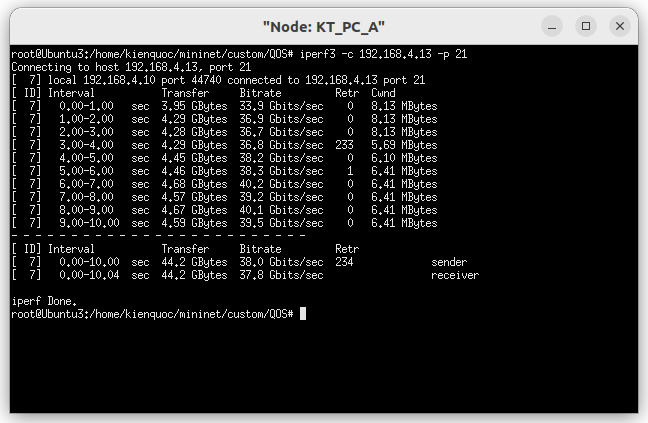
Sau đó nhập vào terminal lệnh sau để kiểm tra lưu lượng giữa KT\_PC\_A vs KD\_PC\_A:

Iperf3 -c 192.168.4.13 -p 21

A screenshot of a computer

Description automatically generated

B3: Xem kết quả



Tốc độ trung bình khoảng 37.8 Gbps

3. Cấu hình chính sách QoS cho Switch S1

B1: Tại terminal, ta nhập lệnh sau để đặt chính sách tỷ lệ xâm nhập cho interface s1-eth1 thành 10 megabit/s(Mbps). Nghĩa là tốc độ vào tối đa là 10Mbps

sudo ovs-vsctl set interface s1-eth1 ingress\_policing\_rate=10000

A screenshot of a computer

Description automatically generated

B2: Ta nhập lệnh sau để kiểm tra lưu lượng  
Iperf3 -c 192.168.4.13 -p 21

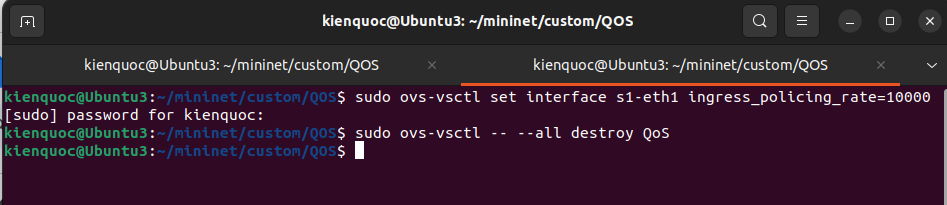
A screenshot of a computer

Description automatically generated

Lúc này tốc độ xâm nhập 10.7 Mbps

B3: Gõ lệnh sau để xóa tất cả chính sách QoS:

sudo ovs-vsctl -- --all destroy QoS



B4:  Gõ lệnh sau để đặt tốc độ tối đa thành 500Mbps cho interface s1-eth1:

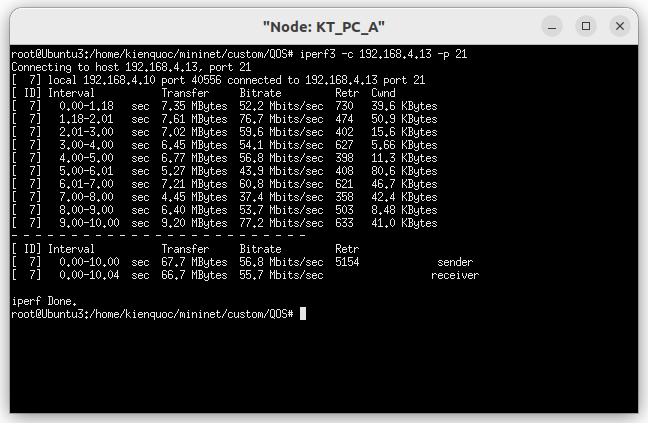
sudo ovs-vsctl set interface s1-eth1 ingress\_policing\_rate=500000

A screenshot of a computer program

Description automatically generated

B5: Gõ lệnh sau để kiểm tra thông lượng giữa h1 và h2:

Iperf3 -c 10.0.0.2 -p 21



Tốc độ lúc này khoảng 55Mbps mặc dù tốc độ tối đa là 500Mbps

B6:Xóa chính sách QoS bằng lệnh sau:

sudo ovs-vsctl -- --all destroy QoS

A screenshot of a computer program

Description automatically generated

**4. Configuring Single-Rate Two-Colors Traffic Policing**

**4.1 Single-rate two-color traffic policing là gì?**

Single-rate two-color traffic policing là một phương pháp kiểm soát lưu lượng trong mạng để đảm bảo rằng lưu lượng đi qua mạng không vượt quá một mức độ xác định. Phương pháp này thường được sử dụng trong các thiết bị mạng như bộ định tuyến hoặc bộ chuyển mạch để kiểm soát và quản lý lưu lượng đến và đi từ mạng.

Single-rate two-color traffic policing thực hiện kiểm tra lưu lượng dựa trên một tốc độ đơn và hai màu (two-color), thường là màu xanh (green) và màu đỏ (red). Nguyên tắc hoạt động cơ bản là như sau:

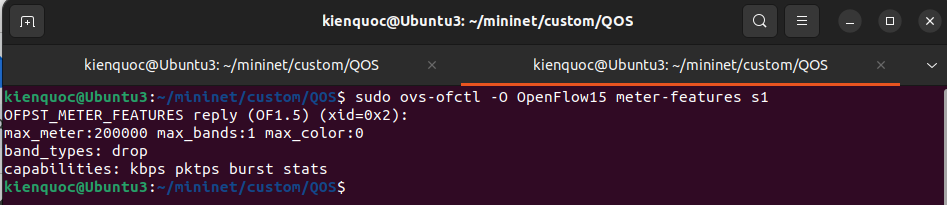
Lưu lượng xanh (green): Lưu lượng được coi là hợp lệ và được phép đi tiếp nếu nó không vượt quá tốc độ xác định (burst rate) trong một khoảng thời gian cụ thể. Lưu lượng xanh được gửi đi mà không bị giới hạn.

Lưu lượng đỏ (red): Lưu lượng vượt quá tốc độ xác định sẽ được coi là lưu lượng đỏ và có thể bị giới hạn hoặc loại bỏ.

**4.2 Cấu hình Single-Rate Two-colors traffic policing**

B1: Nhập lệnh sau để hiển thị các tính năng đo sáng được hỗ trợ bởi OVS.

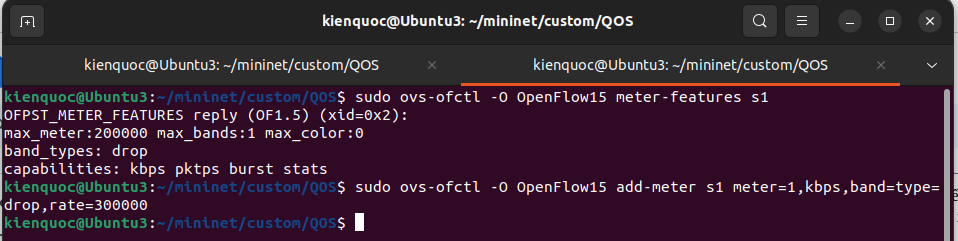
sudo ovs-ofctl -O OpenFlow15 meter-features s1



Band\_types: drop có nghĩa là OVS hỗ trợ 2 màu tốc độ đơn sáng

B2: Nhập lệnh sau để thêm đồng hồ đo vào switch S1 và giới hạn tốc độ ở mức 300Mbps

sudo ovs-ofctl -O OpenFlow15 add-meter s1 meter=1,kbps,band=type=drop,rate=300000



B3: Nhập lệnh sau để cài đặt quy trình thủ công trên switch S1 đối với port 21

sudo ovs-ofctl -O OpenFlow15 add-flow s1 tcp,tp\_dst=21,actions=meter:1,output:3

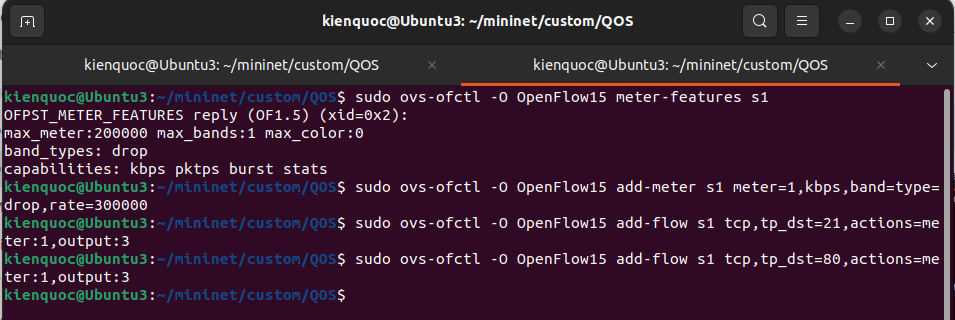
A screenshot of a computer program

Description automatically generated

Ta thấy trên port 21 giao thức TCP, đồng hồ được đặt thành 1 và interface ở đầu ra là s1-eth3. Khi nào port khớp trên TCP, switch S1 sẽ giới hạn tốc độ ở 300Mbps.

B4: Nhập lệnh sau để cài đặt cấu hình thủ công trên switch s1 (giao thức TCP, port 80)

sudo ovs-ofctl -O OpenFlow15 add-flow s1 tcp,tp\_dst=80,actions=meter:1,output:3



\* Một đồng hồ được áp dụng cho nhiều mức nhập luồng.

**5. Verifying metering configuration**

B1: Mở terminal KD\_PC\_B, nhập lệnh sau để chạy máy chủ HTTP sử dụng port 80

iperf3 -s -p 80

A screenshot of a computer

Description automatically generated

B2: Nhập lần lượt câu lệnh (1) trên terminal KT\_PC\_A và (2) trên terminal KT\_PC\_B (\*Không chạy chương trình)

iperf3 -c 192.168.4.13 -p 21 -b 300mb    (1)

iperf3 -c 192.168.4.14 -p 80 -b 300mb    (2)

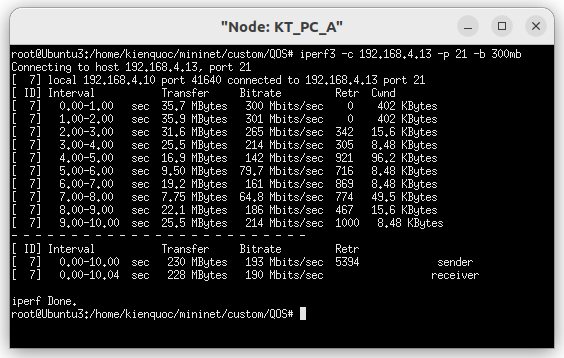
A screenshot of a computer

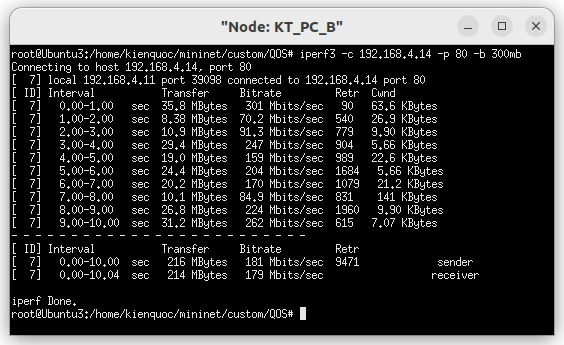
Description automatically generated

A screenshot of a computer

Description automatically generated

B3: Chạy lần lượt lệnh (1) và (2)





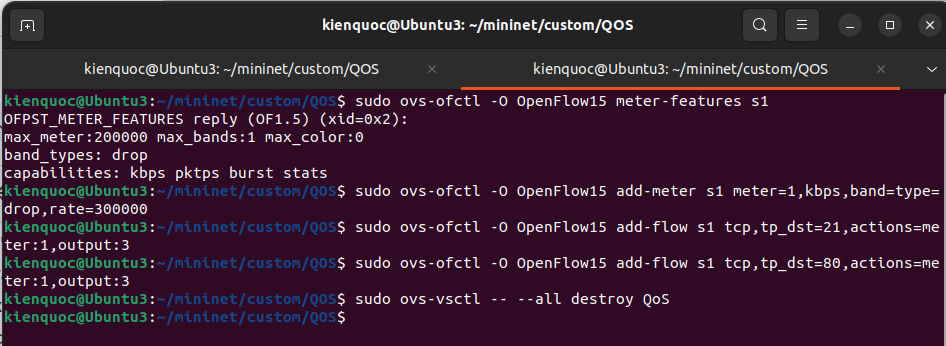
KT\_PC\_A: có tốc độ khoảng 190 Mbps

KT\_PC\_B: có tốc độ khoảng 180 Mbps

=> Mặc dù switch s1 nhận được khoảng 600Mbps từ hai máy chủ tuy nhiên chỉ có khoảng 400Mbps được phép và được đánh dấu là màu xanh lá cây, còn lại được đánh dấu màu đỏ.

B8: Gõ lệnh sau để xóa bỏ chính sách QoS

sudo ovs-vsctl -- --all destroy QoS



**6. QoS Shaping**

**6.1 QoS shaping là gì?**

QoS shaping, còn được gọi là hình dạng lưu lượng, là một công cụ QoS (Chất lượng dịch vụ) mà chúng ta có thể sử dụng để thực thi các tốc độ bit thấp hơn so với khả năng của giao diện vật lý. QoS shaping cho phép lưu lượng truy cập ưu tiên cao hơn luồng ở mức tối ưu ngay cả khi băng thông được sử dụng cao (ưu tiên lưu lượng truy cập). Nó giảm khả năng các gói quan trọng hơn bị trễ hoặc bị bỏ qua khi chúng rời giao diện bằng cách đặt giới hạn băng thông cho các gói ít quan trọng hơn.

QoS shaping cho phép bạn quản lý các dòng lưu lượng truy cập, rời khỏi một giao diện để phù hợp với tốc độ của giao diện đích và đảm bảo rằng nó tuân theo các chính sách đã được thỏa thuận. Các gói lưu lượng truy cập tuân theo một hồ sơ cụ thể có thể được điều chỉnh để đáp ứng nhu cầu phía dưới, loại bỏ tắc nghẽn trong các topologies với sự không khớp về tốc độ dữ liệu.

Bằng cách áp dụng một tốc độ lưu lượng truy cập tối đa cho hàng đợi egress của mỗi giao diện, các chính sách hình dạng lưu lượng truy cập điều chỉnh và làm mịn dòng gói. Các gói vượt quá ngưỡng sẽ được xếp hàng và sẽ được truyền sau. Quá trình này tương tự như cảnh sát lưu lượng truy cập; tuy nhiên, các gói không bị bỏ qua. Vì các gói được đệm, hình dạng lưu lượng truy cập giảm thiểu mất gói (dựa trên độ dài hàng đợi), dẫn đến hành vi lưu lượng truy cập TCP tốt hơn.

**6.2 Cấu hình QoS shaping**

Để định cấu hình QoS trong Open vSwitch, bạn sẽ chỉ định băng thông tối đa và tạo các hàng đợi với mức độ ưu tiên khác nhau. Bước tiếp theo là thêm các hàng đợi này vào các cổng trên switch s1 nơi bạn muốn triển khai QoS. Cuối cùng, bạn sẽ ánh xạ ID hàng đợi được yêu cầu với luồng.

B1: Thực thi tập lệnh cấu hình QoS

Đây là tập lệnh các lệnh được thực thi:

ovs-vsctl -- set port s1-eth3 \

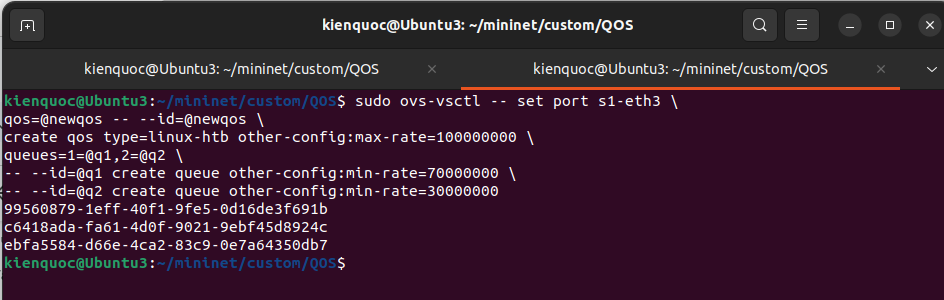
qos=@newqos -- --id=@newqos \

create qos type=linux-htb other-config:max-rate=100000000 \

queues=1=@q1,2=@q2 \

-- --id=@q1 create queue other-config:min-rate=70000000 \

-- --id=@q2 create queue other-config:min-rate=30000000



QoS được tạo với tốc độ truyền dữ liệu tối đa là 100 Mbps và được gắn vào cổng s1-eth3. Hai hàng đợi q1 với hàng đợi id = 1 và q2 với hàng đợi id = 2 đã được tạo. Hàng đợi 1 (q1) có tốc độ truyền tối thiểu là 70 Mbps. q2 có tốc độ truyền tối thiểu là 30 Mbps. Phương thức QoS chỉ đi ra, nghĩa là các tốc độ này sẽ được áp dụng khi các gói được chuyển tiếp ra khỏi cổng.

B2: Nhập lệnh sau để cài đặt luồng nhập thủ công cho q1 trên switch S1

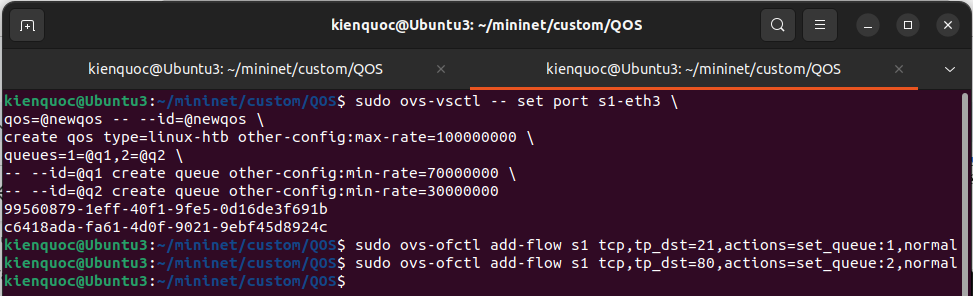
sudo ovs-ofctl add-flow s1 tcp,tp\_dst=21,actions=set\_queue:1,normal

A screenshot of a computer program

Description automatically generated

B3: Nhập lệnh sau để cài đặt luồng nhập thủ công cho q2 trên switch S1

sudo ovs-ofctl add-flow s1 tcp,tp\_dst=80,actions=set\_queue:2,normal



**6.3 Xác minh cấu hình QoS shaping**

Trong phần này, bạn sẽ xác minh cấu hình định hình QoS. Trong phần 1, bạn sẽ xác minh tốc độ lưu lượng truy cập cho từng máy chủ. Trong phần 2, bạn sẽ xác minh rằng máy chủ h1 sẽ có mức độ ưu tiên cao hơn khi cạnh tranh với máy chủ h3.

**6.3.1 Xác minh QoS và tốc độ lưu lượng cho từng máy chủ**

B1: Nhập lệnh sau để xác minh cấu hình QoS cho interface s1-eth3

sudo ovs-appctl -t ovs-vswitchd qos/show s1-eth3

A screenshot of a computer

Description automatically generated

Nó liệt kê tất cả các thông tin liên quan đến hàng đợi QoS. Bạn sẽ thấy hai hàng đợi q1 và q2 được thêm vào danh sách. Theo mặc định, Open vSwitch sử dụng hàng đợi mặc định trong đó min\_rate là 12.000 và max\_rate là 100.000.000.

B2: Nhập lệnh sau để kiểm tra thông lượng giữa 2 máy chủ KT\_PC\_A vs KD\_PC\_A. Tại thời điểm này tốc độ tối đa là 100Mbps.

iperf3 -c 192.168.4.13 -p 21

A screenshot of a computer

Description automatically generated

Hình trên cho thấy tốc độ khoảng 82 Mbps

B3: Nhập lệnh sau để chạy máy chủ dưới dạng HTTP, sử dụng cổng 80

iperf3 -s -p 80

A screenshot of a computer

Description automatically generated

B4:Nhập lệnh sau để kiểm tra thông lượng giữa 2 máy chủ KT\_PC\_B vs KD\_PC\_B. Tại thời điểm này tốc độ tối đa là 100Mbps.

iperf3 -c 192.168.4.14 -p 80

A screenshot of a computer

Description automatically generated

Hình trên cho thấy tốc độ khoảng 70 Mbps

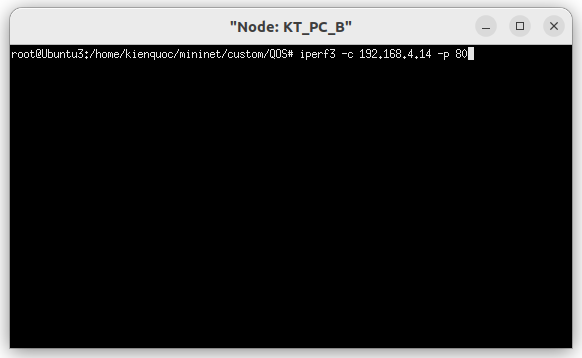
**6.3.2 Xác minh tỷ lệ lưu lượng truy cập cho các máy chủ cạnh tranh**

B1: Nhập lệnh (1) và (2) để kiểm tra thông lượng lần lượt giữa KT\_PC\_A vs KD\_PC\_A và KT\_PC\_B vs KD\_PC\_B trong 20s:

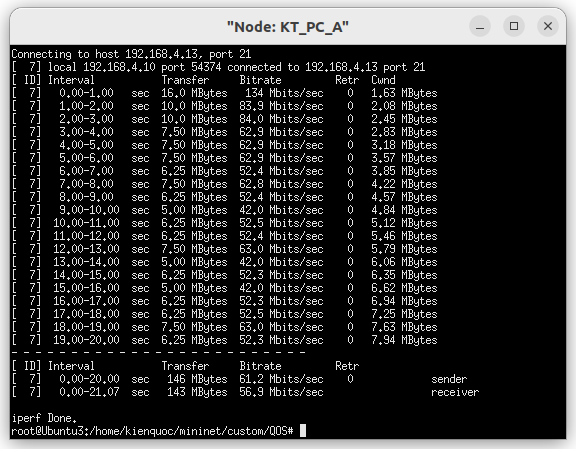
iperf3 -c 192.168.4.13 -t 20 -p 21    (1)

iperf3 -c 192.168.4.14 -t 20 -p 80    (2)

A screenshot of a computer

Description automatically generated

B2: Chạy lần lượt lệnh (1) và (2)



A screenshot of a computer

Description automatically generated

So sánh kết quả giữa 2 máy chủ ta rút ra được nhận xét:

-Máy chủ KT\_PC\_A có tốc độ bit đầu tiên khoảng 84Mbps sau đó giảm dần 63Mbps do có sự cạnh tranh từ KT\_PC\_B

-Máy chủ KT\_PC\_B có tốc độ trong khoảng 30Mbps do có độ ưu tiên thấp hơn máy chủ KT\_PC\_A