Laboratório: QoS com Reserva de Recursos e Classificação de Tráfego em Redes SDN

Nilton J. Mocelin Jr.[1]

[1]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC)
Joinville, SC – Brasil

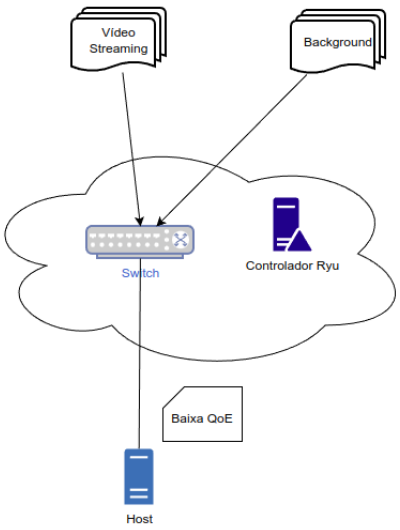
nilton.junior@edu.udesc.br

1. Objetivo

** Máquinas devem ser ligadas utilizando linux.*

Neste laboratório, os alunos irão criar uma topologia de rede que prioriza o tráfego de vídeo sobre tráfego UDP, utilizando os conceitos da arquitetura de redes SDN. Cada estudante terá uma rede com clientes que acessam o serviço de streaming de vídeo de uma rede que possui um servidor de vídeo e uma aplicação UDP gananciosa. Os estudantes devem implementar os procedimentos para descobrir qual fluxo é de vídeo e qual fluxo não é utilizando classificação de tráfego. Com isso, os alunos podem criar regras meter para reservar os recursos necessários para o streaming de vídeo e observar o impacto da disputa de recursos.

Figura 1. Rede ISP.



Simulador de ISPs

Os alunos irão acessar uma máquina virtual e desenvolver uma topologia de rede SDN simples, simulando um ISP e um cliente. O *professor* substituto também desenvolverá uma topologia de rede, que estará conectada a topologia dos estudantes. Na topologia do *professor*, existirá um servidor de vídeo e um *host* com uma aplicação UDP.

Todos os códigos utilizados podem ser encontrados no link: https://github.com/NiltonMocelin/laboratorioDMCA.

Figura 2. Disputa por Recursos de Rede.

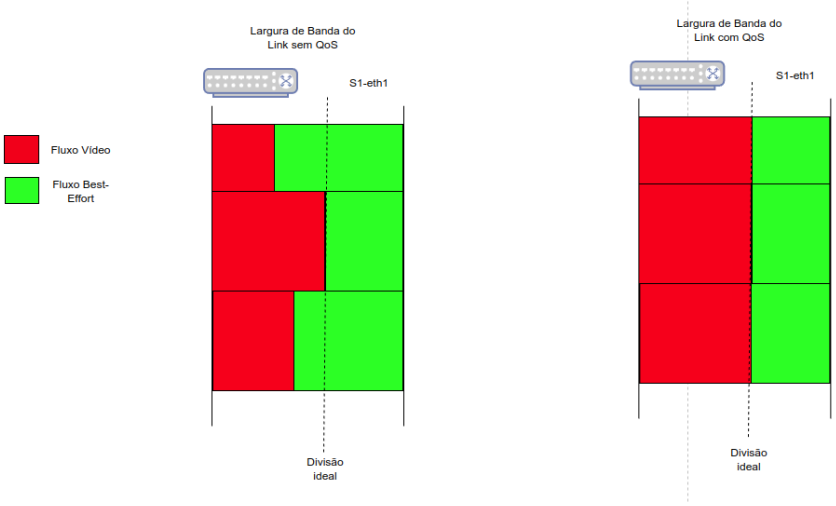
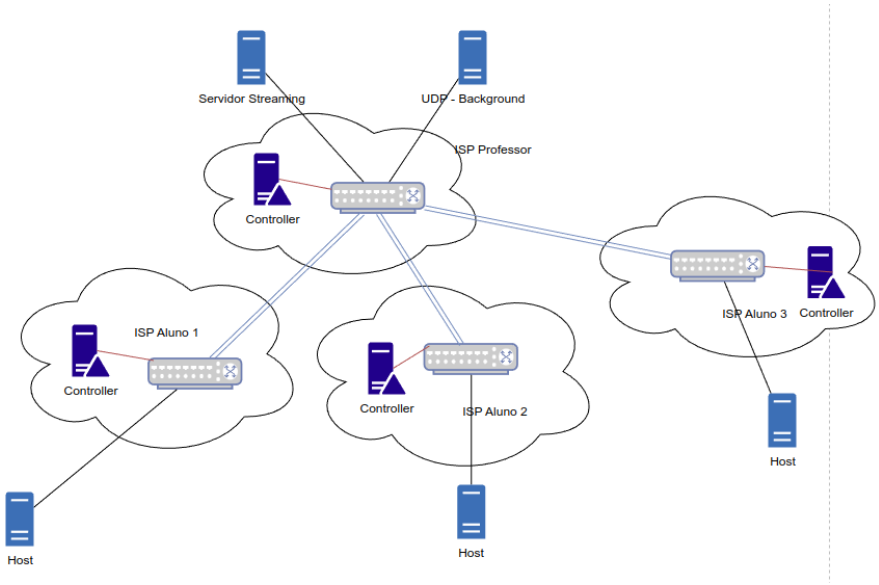


Figura 3. Topologia dos ISPs.



Instalar uma Máquina Virtual

As máquinas da UDESC não possuem permissões o suficiente para que o usuário dos alunos possa instalar pacotes ou modificar o estado da máquina. Por isso, vamos instalar uma máquina virtual para a ferramenta VirtualBox.

- Primeiro vamos fazer *download* da máquina virtual do sistema operacional Ubuntu 20.04 server (disponibilizada pela Github do mininet). Acessem o link Donwload Mininet-VM.
- Inicializem o VirtualBox....
- Importar Mininet-VM: Arquivo $\rightarrow$ Importar Appliance $\rightarrow$ Selecionem o arquivo $.ovf$ $\rightarrow$ Importar.
- Mudar a configuração de rede da máquina virtual para modo bridge. Configurações $\rightarrow$ Rede $\rightarrow$ Adaptador1 $\rightarrow$ Conectado a: Placa em modo Bridge.

- Mudar a configuração gráfica: Configurações → Monitor → controladora gráfica: VMSVGA; Memória de Vídeo: 128MB; Habilitar Aceleração 3D.
- Aumentar a memória RAM: Configurações → Sistema → 2048MB.

2. Configurar VM

Após isso, inicializem a máquina virtual. Vamos configurar a VM para seguir com o laboratório.

Para fazer login na máquina:

Usuário: mininet Senha: mininet

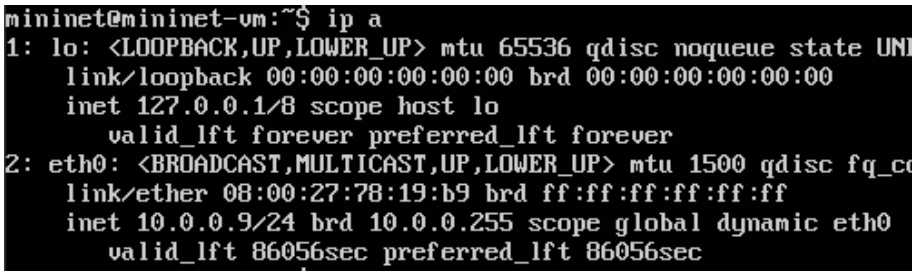
Criem o arquivo Xauthority, que permite encaminhar os dados do servidor de vídeo X11 da máquina virtual para a máquina física:

§ touch /.Xauthority

Identifiquem o endereço IP da máquina virtual:

§ ip a

Figura 4. Obter endereço IP da máquina virtual.



Abram um terminal na máquina física e conectem a máquina virtual por meio de SSH (terminal permite copiar e colar):

§ ssh -Y mininet@$< ip - vm >$

3. Instalar controlador Ryu

Atualizar a base de dados do gerenciador de pacotes - para poder instalar pacotes na máquina:

§ sudo apt-get update

Instalar o controlador Ryu:

§ sudo pip3 install ryu

Atualizar o switch virtual:

§ sudo apt install openvswitch-switch

Vamos clonar o repositório onde estão os códigos para realizar o laboratório:

§ git clone https://github.com/NiltonMocelin/DocenciaMCA.git

Adicionando o display atual a lista de autorizados (repetir sempre que tiver erro ao abrir o GUI de uma aplicação):

§ sudo xauth add $(xauth -f /.Xauthority list | tail -1)

Instalar um editor leve, geany:

§ sudo apt-get install geany

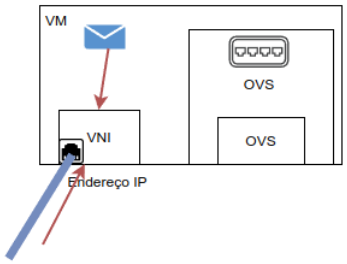
4. Configurando a Topologia

Vamos configurar uma topologia que simule o cenário de ISP proposto. O ISP possui um switch que se conecta a um controlador e a um Host. Para criar o *switch* OVS, vamos criar uma *bridge*:

§ sudo ovs-vsctl add-br switch
§ sudo ovs-vsctl set bridge switch protocols=OpenFlow10,OpenFlow11,OpenFlow12,OpenFlow13

A configuração da máquina virtual atualmente está mostra a Figura 5.

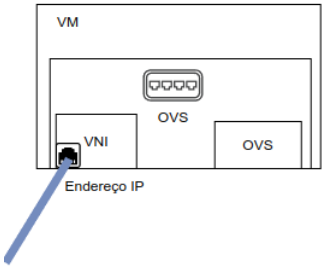
Figura 5. Configuração de rede da VM - Parte 1.



Vamos configurar a conexão entre o Host e o *switch*. Para isso, digitamos os seguintes comandos no terminal do VirtualBox (pois a conexão SSH irá cair). Conectando a interface do eth0 ao switch (Figura 6):

§ sudo ovs-vsctl add-port switch eth0

Figura 6. Configuração de rede da VM - Parte 2.



Só com isso o *switch* OVS não é capaz de configurar regras para a gerenciar a porta 'eth0', temos que fazer o tráfego de 'eth0' passar pelo *switch*. Para isso, é preciso associar o endereço IP de eth0 ao switch (porta do switch com mesmo nome). Primeiro vamos observar qual o endereço do *gateway* (... via 'endereço gateway' ...) e o endereço associado a interface eth0:

- Verificar *gateway*: [§] ip route.
- Verificar IP eth0: [§] ifconfig.

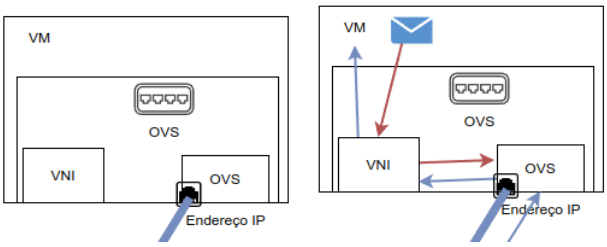
Então, passamos o endereço de eth0 para a porta switch:

§ sudo ifconfig eth0 0.0.0.0
§ sudo ifconfig switch $< antigo - ip - eth0 > /mascara$

- Configurar rota padrão com endereço do *gateway* anterior: [§] sudo ip route add default via 'endereço-gateway' dev switch.
- Verificar se o endereço IP e rota *default* foram configuradas.
- Caso tenha problemas, sempre se pode retornar ou deletar as configurações substituindo 'add' por 'del'.

Após configurar o endereço IP ao switch, já é possível utilizar SSH novamente e acessar a Internet. Desta forma, a topologia configurada é a mesma mostrada na Figura 7. A comunicação entre a interface 'eth0' e o *switch* acontece, pois automaticamente se cria uma regra de fluxo com a ação NORMAL. Essa ação faz com que os pacotes que chegam na interface do *switch* sejam analisados fora do *pipeline* do OVS, assim os pacotes trafegam normalmente como mostra a figura:

Figura 7. Configuração de rede da VM - Parte 3.



5. Configuração de velocidade dos links

Podemos observar que a velocidade do link atual pode chegar até 5Gbps entre Host e máquina física ou 1Gbps (velocidade do cabo Ethernet Gigabyte) entre dois Hosts. Isso pode ser verificado iniciando um servidor Iperf em um Host (VM) e um cliente Iperf em outro (máquina física):

Host 1:

§ iperf -S

Host 2:

§ iperf -c $ip - host1$

Essa velocidade de link não reflete um cenário proposto. Por isso, vamos configurar a velocidade do link entre o switch e o Host. No cenário proposto, o Host contrata 20Mbps de largura de banda. Por isso, vamos configurar o link utilizando a ferramenta linux traffic control (20Mbps += 1024kbps * 20) para tratar o tráfego egresso e a ferramenta do OVS que limita o tráfego ingresso.

O tráfego que sai de 'eth0' é o tráfego originado na VM. Este tráfego deve ser encaminhado para a interface do *switch*, que deve ser enviado para fora da VM. Por outro lado, o tráfego que chega na interface do *switch* é externo e deve ser encaminhado para 'eth0'. Esse comportamento deve ser levado em consideração quando for criar as regras e configurações dos links.

- Configurando velocidade entre switch $\rightarrow$ eth0: [§] sudo tc qdisc add dev switch root tbf rate 20240kbit latency 10ms burst 15400

- Configurando velocidade entre 'eth0' $\rightarrow$ *switch*: [§] sudo ovs-vsctl set interface switch ingress_policing_rate=20000
- Caso seja necessário corrigir [§] sudo tc qdisc del dev $interface$ root.

A configuração dos links foi finalizada, no entanto, é como se o tráfego ingresso em 'eth0' entrasse por um link e saísse por outro, leve isso em consideração. Utilizando Iperf é possível verificar que a largura de banda do link foi configurada.

6. Testando QoS com congestionamento

Execute um dos servidores de vídeo de http_server em uma máquina. Acesse com o servidor com outra máquina. Verifique a qualidade de experiência obtida. Após isso, estresse o link com um fluxo UDP.

- Servidor 1: [§] python3 server.py
- Ou, servidor 2: [§] python3 server2.py
- Em outra máquina, acesse o link apresentado pelo servidor: mplayer, firefox, fplay...

Abra um servidor iperf em um novo terminal:

§ iperf -u -s

Enquanto acessa a aplicação de vídeo, outra máquina deve se conectar ao servidor iperf para verificar o efeito do congestionamento na aplicação:

§ iperf -u -b 150m -c $ip - servidor - iperf$

7. Configurando regras de QoS manualmente

Criar regras de fluxo na mão:

- Remover controlador anterior: [§]
- sudo ovs-vsctl del-controller switch
- Limpar regras antigas (vai cair SSH): [§] sudo ovs-ofctl del-flows switch
- 'eth0' $\rightarrow$ switch $\rightarrow$ Fora VM: [§] sudo ovs-ofctl add-flow switch in_port=1, actions=output:LOCAL
- Fora VM $\rightarrow$ switch $\rightarrow$ 'eth1' $\rightarrow$ VM: [§] sudo ovs-ofctl add-flow switch in_port=LOCAL, actions=output:1
- Adicionando regra *meter*: [§] sudo ovs-ofctl -O OpenFlow13 add-meter switch meter=1,kbps,band=type=drop,rate=2000
- Criando regra de fluxo limitada pela *meter*: [§] sudo ovs-ofctl -O OpenFlow13 add-flow switch udp,in_port=1,actions=meter=1,output:LOCAL

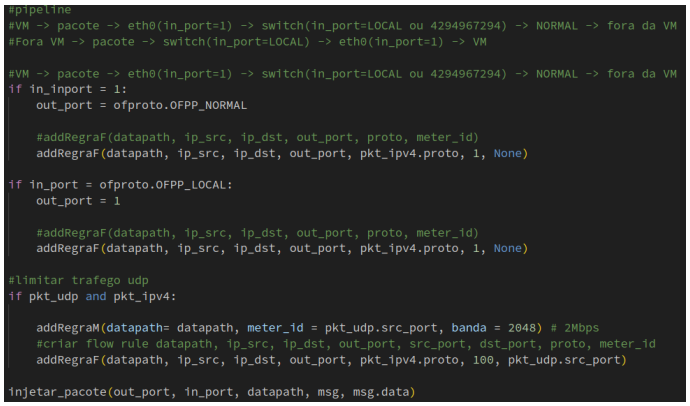
§ sudo ovs-ofctl -O OpenFlow13 add-flow switch udp,in_port=LOCAL,priority=10,actions=meter=1,output:1

8. Configurando regras de QoS com controlador

Vamos configura o switch para se conectar ao controlador remoto, que será configurado nos próximos passos:

§ sudo ovs-vsctl set-controller switch tcp:127.0.0.1:6653

Figura 8. Modificação Simple Switch.



```
#pipeline
#VM -> pacote -> eth0(in_port=1) -> switch(in_port=LOCAL ou 4294967294) -> NORMAL -> fora da VM
#Fora VM -> pacote -> switch(in_port=LOCAL) -> eth0(in_port=1) -> VM

#VM -> pacote -> eth0(in_port=1) -> switch(in_port=LOCAL ou 4294967294) -> NORMAL -> fora da VM
if in_inport = 1:
    out_port = ofproto.OFPP_NORMAL

    #addRegraF(datapath, ip_src, ip_dst, out_port, proto, meter_id)
    addRegraF(datapath, ip_src, ip_dst, out_port, pkt_ipv4.proto, 1, None)

if in_port = ofproto.OFPP_LOCAL:
    out_port = 1

    #addRegraF(datapath, ip_src, ip_dst, out_port, proto, meter_id)
    addRegraF(datapath, ip_src, ip_dst, out_port, pkt_ipv4.proto, 1, None)

#limitar trafego udp
if pkt_udp and pkt_ipv4:

    addRegraM(datapath= datapath, meter_id = pkt_udp.src_port, banda = 2048) # 2Mbps
    #criar flow rule datapath, ip_src, ip_dst, out_port, src_port, dst_port, proto, meter_id
    addRegraF(datapath, ip_src, ip_dst, out_port, pkt_ipv4.proto, 100, pkt_udp.src_port)

injetar_pacote(out_port, in_port, datapath, msg, msg.data)
```

Vamos utilizar o controlador simple_switch_13_modificado.py. A modificação realizada foi para criar regras que limitem fluxos UDP por uma meter de 1mb. Desta forma, mesmo que ocorra congestionamento por um fluxo UDP, um fluxo TCP não será afetado:

Para observar as regras de fluxo ativas, pode utilizar o comando:

§ sudo ovs-ofctl -O OpenFlow13 dump-flows switch

- Configurar o controlador remoto no *switch*: [§] sudo ovs-vsctl set-controller switch tcp:127.0.0.1:6653
- Para executar o controlador: [§] ryu-manager simple_switch_13_modigicado.py
- Realizar teste iperf UDP com e sem o controlador.
- Realizar teste streaming de vídeo com e sem iperf UDP.

9. Comando úteis

- iperf -s
- iperf -u -s
- iperf -c 'endereço-ip'
- iperf -u -b 15m -c 'endereço-ip' -t 1000
- sudo ovs-vsctl add-br switch
- sudo ovs-vsctl add-port switch eth0
- sudo ovs-vsctl set-controller switch tcp:127.0.0.1:6653
- sudo ovs-vsctl del-controller switch
- sudo ovs-vsctl set bridge switch protocols=OpenFlow10,OpenFlow11,OpenFlow12,OpenFlow13
- sudo ovs-ofctl -O OpenFlow13 dump-flows switch
- sudo ovs-ofctl -O OpenFlow13 dump-meters switch
- Link para criar regras na mão rapidamente: Comandos-link.
- sudo ovs-vsctl show