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

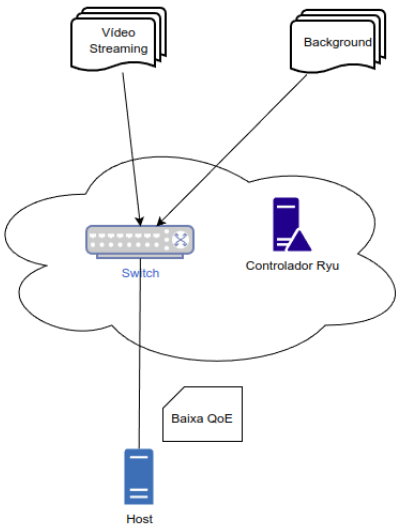
nilton.junior@edu.udesc.br

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.

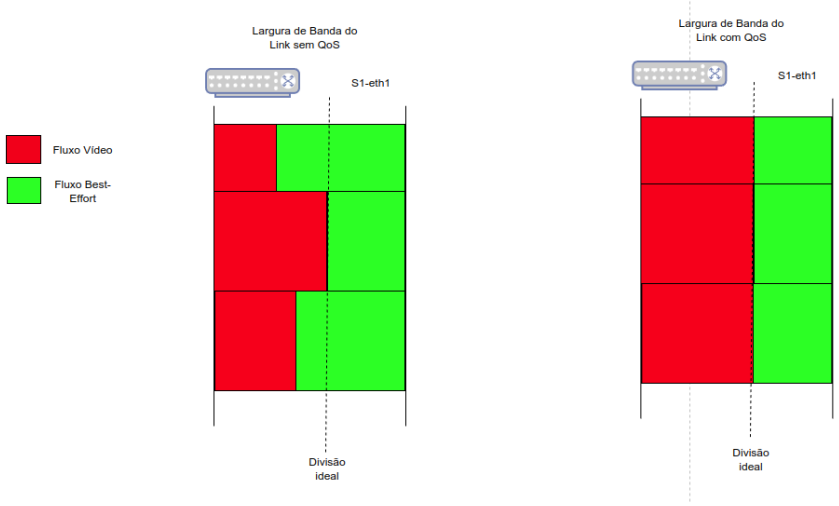
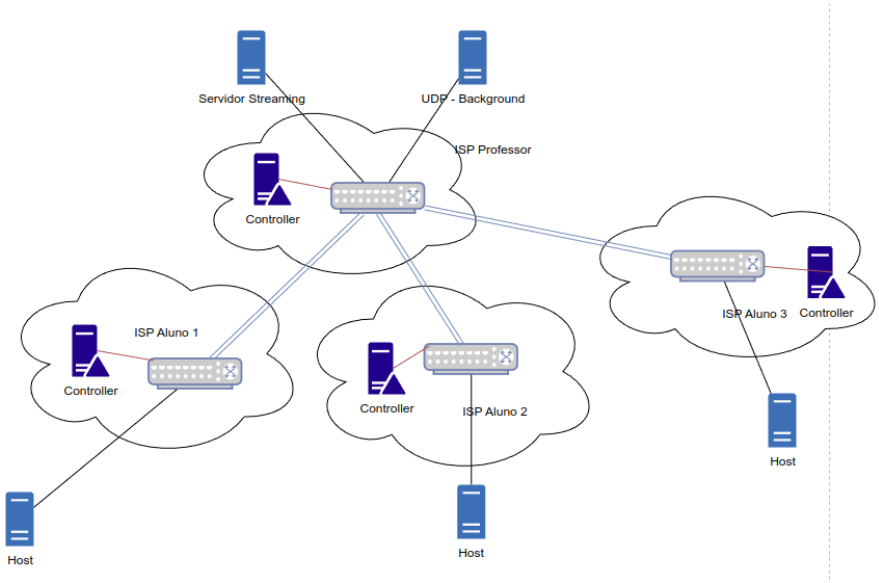


Figura 3. Topologia dos ISPs.



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 hrefhttps://github.com/mininet/mininet/releases/download/2.3.0/mininet-2.3.0-210211-ubuntu-20.04.1-legacy-server-amd64-ovf.zipDonwload 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

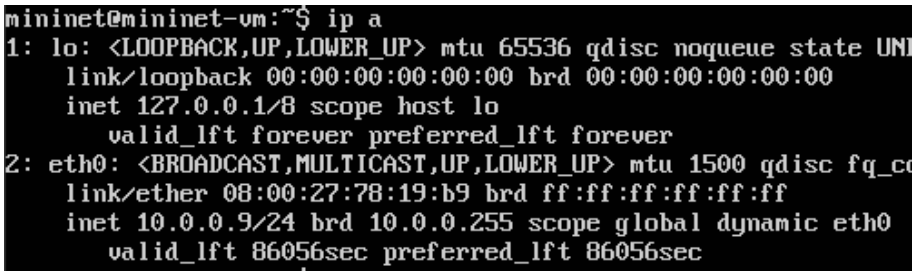
Criem o arquivo Xauthority, que permite encaminhar os dados do servidor de vídeo X11 da máquina virtual para a máquina física:

§ 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

O ISP possui um switch, um controlador e 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

O Host é a própria máquina virtual. Utilizando o terminal do VirtualBox (pois a conexão irá cair), vamos conectar a interface do eth0 ao switch:

§ ovs-vsctl add-port switch eth0

Para que o switch ovs possa gerenciar a interface eth0, é preciso associar o endereço IP de eth0 ao switch (porta do switch):

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

É preciso configurar o servidor DNS para a interface também:

§ sudo su
§ echo ”nameserver 8.8.8.8”> /etc/resolv.conf
§ exit

Após configurar o endereço IP ao switch, já é possível utilizar SSH novamente. Desta forma, a topologia configurada é a mesma mostrada na Figura **??**:

** imagem topologia **

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 e um cliente Iperf em outro:

Host 1:

§ iperf -S

Host 2:

§ iperf -c $ip - host1$

Essa velocidade de link não reflete um cenário real. Por isso, vamos configurar a velocidade do link entre o switch e o Host. No cenário proposto, o Host contratou 20Mbps de largura de banda. Vamos criar uma fila com esse limite, que é por onde o tráfego será tratado.

por isso vamos configurar o link utilizando a ferramenta linux traffic control (10Mbps += 1024kbps * 10):

Tc apenas configura o tráfego de saída. No entanto, o tráfego que chega no switch sai para eth0 e o tráfego gerado na máquina sai de eth0 para o switch. Desta forma, podemos limitar o link aplicando uma política para tráfego egresso em switch e eth0. (Outra forma seria utilizar uma fila limitada na porta do switch que conecta a interface eth0).

- Configurando o tráfego entre eth0 → switch: [§] sudo tc qdisc add dev eth0 root tbf rate 10240kbit latency 10ms burst 15400
- Configurando o tráfego entre switch → eth0: [§] sudo tc qdisc add dev switch root tbf rate 10240kbit latency 10ms burst 15400
- Caso seja necessário corrigir [§] sudo tc qdisc del dev $interface$ root.
- Definir a taxa de tráfego ingresso no switch: [§] ovs-vsctl set interface switch ingress_policing_rate=10000

  Utilizando iperf é possível verificar que a largura de banda do link foi configurada.

5. Testando QoS com congestionamento

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 $ip - servidor - iperf$

6. Configurando QoS

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

Vamos utilizar o controlador simpleswitch13modificado.py. As regras básicas já são implementadas por esse controlador. Sendo essas, a regra para que os pacotes que chegam da porta da máquina virtual sejam roteados para a porta local e os pacotes que chegam da porta local sejam roteados para a porta da máquina virtual.

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, o fluxo tcp não será afetado:

Figura 5. Modificação Simple Switch.

```
if pkt_ipv4:
    ip_src = pkt_ipv4.src
    ip_dst = pkt_ipv4.dst
    print('proto', pkt_ipv4.proto)

pkt_tcp = pkt.get_protocol(tcp.tcp)

pkt_udp = pkt.get_protocol(udp.udp)

out_port = 1
if in_port == 1:
    out_port == 4294967294

if pkt_udp and pkt_ipv4:

    addRegraM(datapath= datapath, meter_id = int(pkt_udp.src_port), banda = 1024) # 1Mbps
    #criar flow rule datapath, ip_src, ip_dst, out_port, src_port, dst_port, proto, meter_id
    addRegraF(datapath, ip_src, ip_dst, int(out_port), int(pkt_ipv4.proto), int(pkt_udp.src_port))
```

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

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