**Trabalho final REC(Parte 1)**

**Guilherme M. Utiama1, Gustavo Daniel1**

1Universidade do Estado de Santa Catarina

**Aplicação de redes**

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.util import dumpNodeConnections

from mininet.log import setLogLevel

class SingleSwitchTopo(Topo):

def build(self, n=2):

switch = self.addSwitch(’s1’)

for h in range(n):

host = self.addHost(”h %s” % (h + 1))

self.addLink(host, switch)

def simpleTest():

”Create and test a simple network”

topo = SingleSwitchTopo(n=4)

net = Mininet(topo)

net.start()

print ”Dumping host connections”

dumpNodeConnections(net.hosts)

print ”Testing network connectivity”

net.pingAll()

net.stop()

if \_\_name\_\_== ’\_\_main\_\_’:

setLogLevel(’info’)

simpleTest()

Através desta aplicação, obtém-se o seguinte resultado:

**Aplicação para limitar o uso da cpu**

from mininet.topo import Topo

from mininet.net import Mininet

from mininet.node import CPULimitedHost

from mininet.link import TCLink

from mininet.util import dumpNodeConnections

from mininet.log import setLogLevel

class SingleSwitchTopo( Topo ):

"Single switch connected to n hosts."

def build( self, n=2 ):

switch = self.addSwitch( 's1' )

for h in range(n):

host = self.addHost( 'h%s' % (h + 1), cpu=.5/n )

self.addLink( host, switch, bw=10, delay='5ms', loss=2,

max\_queue\_size=1000, use\_htb=True )

def perfTest():

"Create network and run simple performance test"

topo = SingleSwitchTopo( n=4 )

net = Mininet( topo=topo, host=CPULimitedHost, link=TCLink )

net.start()

print "Dumping host connections"

dumpNodeConnections( net.hosts )

print "Testing network connectivity"

net.pingAll()

print "Testing bandwidth between h1 and h4"

h1, h4 = net.get( 'h1', 'h4' )

net.iperf( (h1, h4) )

net.stop()

if \_\_name\_\_ == '\_\_main\_\_':

setLogLevel( 'info' )

perfTest()

Através desta aplicação, obtém-se o seguinte resultado no terminal:

**Aplicação de monitoramento de arquivos**

def monitorTest( N=3, seconds=3 ):

"Run pings and monitor multiple hosts"

topo = SingleSwitchTopo( N )

net = Mininet( topo )

net.start()

hosts = net.hosts

print "Starting test..."

server = hosts[ 0 ]

outfiles, errfiles = {}, {}

for h in hosts:

outfiles[ h ] = '/tmp/%s.out' % h.name

errfiles[ h ] = '/tmp/%s.err' % h.name

h.cmd( 'echo >', outfiles[ h ] )

h.cmd( 'echo >', errfiles[ h ] )

h.cmdPrint('ping', server.IP(), '>', outfiles[ h ], '2>', errfiles[ h ], '&' )

print "Monitoring output for", seconds, "seconds"

for h, line in monitorFiles( outfiles, seconds, timeoutms=500 ):

if h:

print '%s: %s' % ( h.name, line )

for h in hosts:

h.cmd('kill %ping')

net.stop()

**Interface de linha de comando**

from mininet.topo import SingleSwitchTopo

from mininet.net import Mininet

from mininet.cli import CLI

net = Mininet(SingleSwitchTopo(2))

net.start()

CLI(net)

net.stop()

**Aplicação de um controlador próprio**

from mininet.net import Mininet

from mininet.node import Controller

from mininet.topo import SingleSwitchTopo

from mininet.log import setLogLevel

import os

class POXBridge( Controller ):

"Custom Controller class to invoke POX forwarding.l2\_learning"

def start( self ):

"Start POX learning switch"

self.pox = '%s/pox/pox.py' % os.environ[ 'HOME' ]

self.cmd( self.pox, 'forwarding.l2\_learning &' )

def stop( self ):

"Stop POX"

self.cmd( 'kill %' + self.pox )

controllers = { 'poxbridge': POXBridge }

if \_\_name\_\_ == '\_\_main\_\_':

setLogLevel( 'info' )

net = Mininet( topo=SingleSwitchTopo( 2 ), controller=POXBridge )

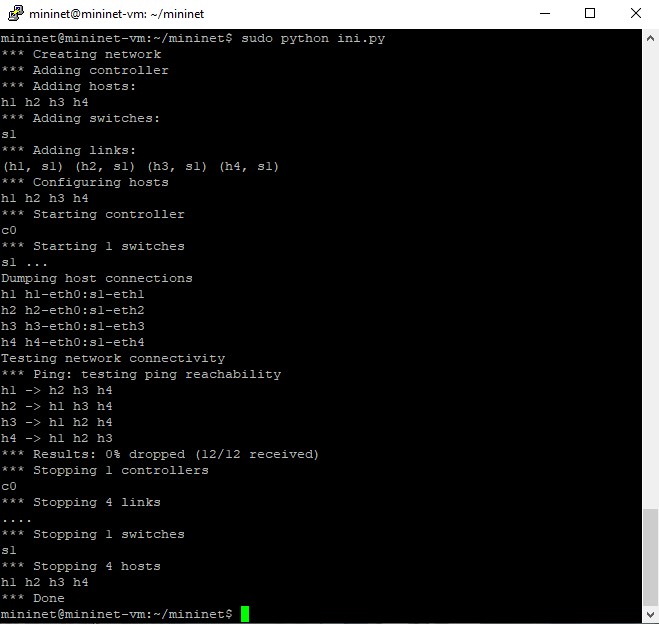
net.start()

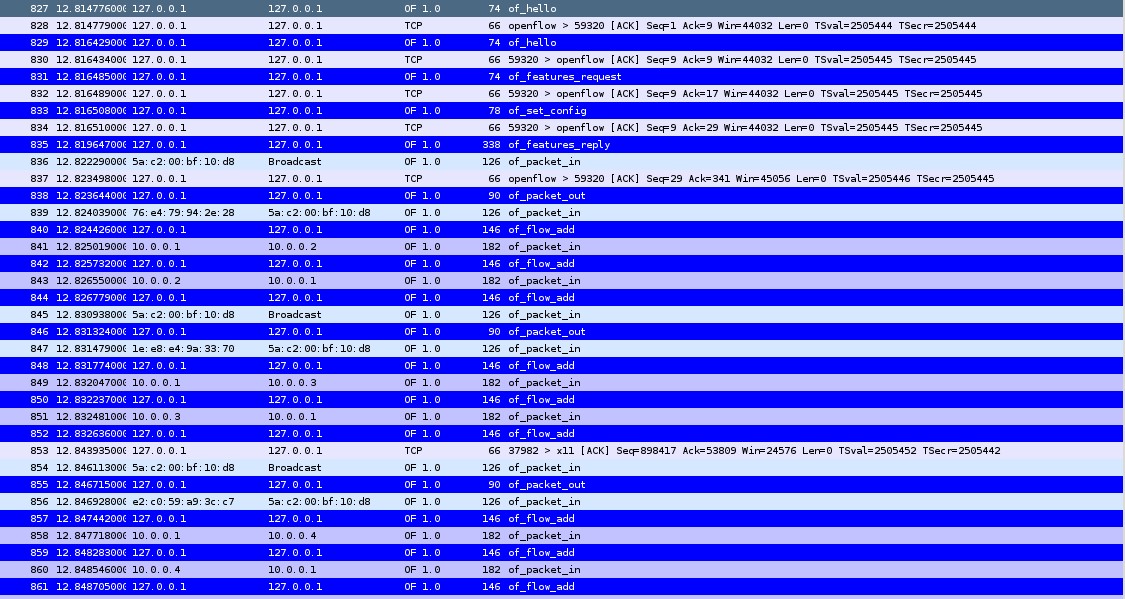
net.pingAll()

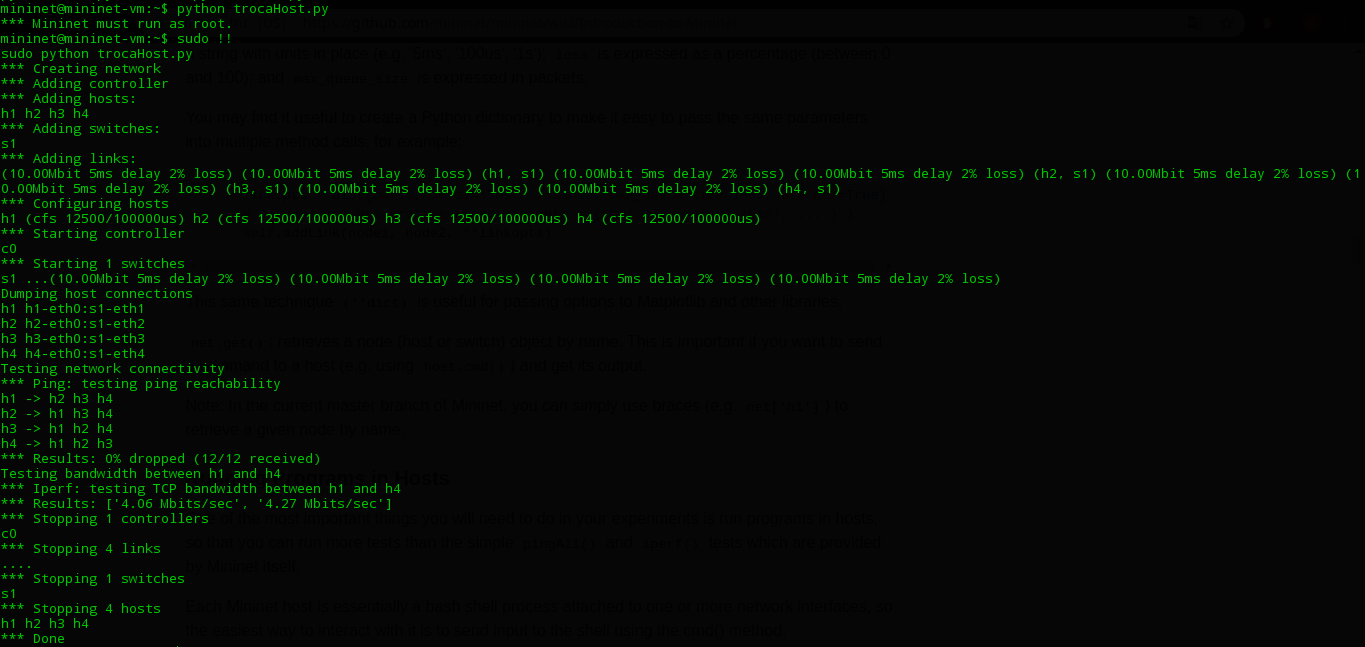
net.stop()

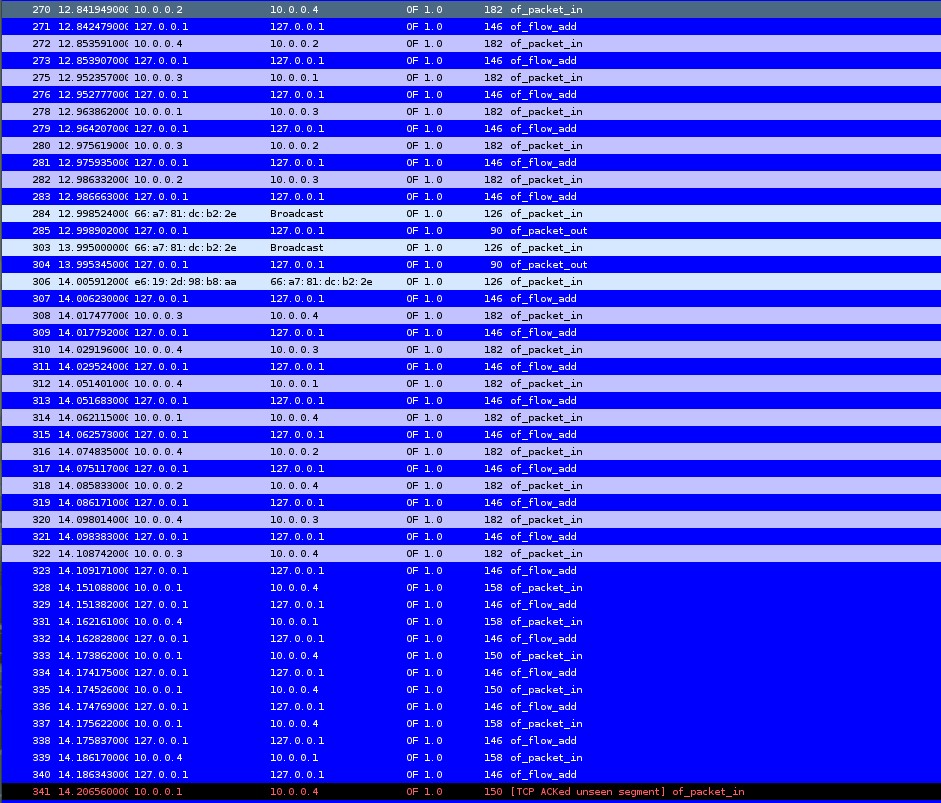
**Resultados obtidos:**

1. **Aplicação de redes**

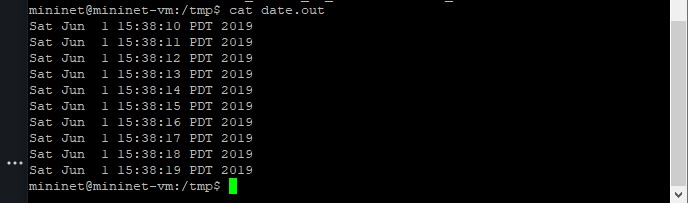
****

****

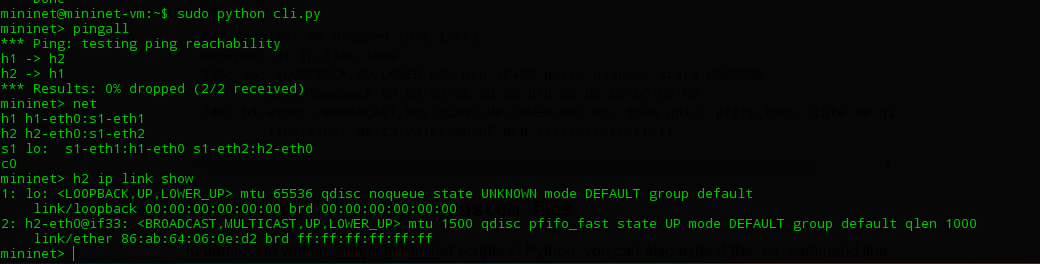
1. **Aplicação para limitar o uso da cpu**

****

1. **Aplicação de monitoramento de arquivos**

****

1. **Interface de linha de comando**

****

1. **Aplicação de um controlador próprio**

