

Apéndice C. Código fuente de las topologías Mininet

Este apéndice corresponde al código fuente en *Python* de cada una de las topologías implementadas en Mininet para la ejecución de experimentos de streaming de video codificado con HEVC/H265.

1. Código Fuente en lenguaje *Python* para la topología sencilla (H1S1H2.py)

```
# codigo fuente de topologia sencilla
# host transmisor h1
# conmutador s1
# host receptor h2
# controlador c0

# A continuacion se importan las librerias
# necesarias para la creacion e inicio de la red SDN

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import TCLink

# se crea la topologia

def topology():
    "Creando red"
    net = Mininet( controller=Controller, link=TCLink, switch=OVSKernelSwitch)

    print "**** Creando nodos ****"
    h1 = net.addHost( 'h1', ip="10.0.0.1/8", mac='00:00:00:00:00:01')
    h2 = net.addHost( 'h2', ip="10.0.0.2/8", mac='00:00:00:00:00:02')
    s1 = net.addSwitch( 's1', listenPort=6673, mac='00:00:00:00:00:03')
    c0 = net.addController('c0', controller=Controller, ip='127.0.0.1', port=6633 )

    print "**** Creando enlaces ****"
    linkopts0=dict(delay='10ms', loss=10)
    net.addLink(h1, s1, cls=TCLink, **linkopts0)
    net.addLink(s1, h2, cls=TCLink, **linkopts0)

    print "**** Iniciando la red ****"
    net.build()
    c0.start()
    s1.start( [c0] )

    print "**** Iniciando linea de comandos CLI ****"
    CLI ( net )

    print "**** Deteniendo emulacion de red ****"
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
```

```
topology()
```

2. Código Fuente en lenguaje *Python* para la topología split (divisor de tráfico)

(Split.py)

```
# codigo fuente de topologia split (divisor de trafico)
# host transmisor h1
# conmutadores s1, s2, s3, s4
# host receptor h3
# host adicional h2
# controlador c0

# A continuacion se importan las librerias
# necesarias para la creacion e inicio de la red SDN

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, OVSSwitch, UserSwitch
from mininet.link import Link, TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

# se crea la topología

def topology():

    "Creando red"

    net = Mininet( controller=Controller, link=TCLink, switch=OVSKernelSwitch )

    print "*** Creando nodos"

    h1 = net.addHost( 'h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8' )
    h2 = net.addHost( 'h2', mac='00:00:00:00:00:02', ip='10.0.0.2/8' )
    h3 = net.addHost( 'h3', mac='00:00:00:00:00:03', ip='10.0.0.3/8' )

    # dado que se trata de una topologia multicamino
    # se debe configurar los flujos o rutas en los conmutadores
    # a través de protocolo OpenFlow
    # estas rutas se configuran en script SplitFlows.sh

    s1 = net.addSwitch( 's1', protocols=["OpenFlow10,OpenFlow13"], listenPort=6674, mac='00:00:00:00:00:04' )
    s2 = net.addSwitch( 's2', protocols=["OpenFlow10,OpenFlow13"], listenPort=6675, mac='00:00:00:00:00:05' )
    s3 = net.addSwitch( 's3', protocols=["OpenFlow10,OpenFlow13"], listenPort=6676, mac='00:00:00:00:00:06' )
    s4 = net.addSwitch( 's4', protocols=["OpenFlow10,OpenFlow13"], listenPort=6679, mac='00:00:00:00:00:09' )
    c0 = net.addController( 'c0', controller=RemoteController, ip='127.0.0.1', port=6633 )

    print "*** Creando enlaces"

    # en la linea a continuacion se definen los valores de retardo y perdida de paquetes por enlace

    link1=dict(delay='10ms', loss=10)
```

```

net.addLink(h1, s1)
net.addLink(h2, s1)
net.addLink(s1, s2, cls=TCLink, **link1)
net.addLink(s1, s3)
net.addLink(s2, s4)
net.addLink(s3, s4)
net.addLink(s4, h3)

print "*** Iniciando red"

net.build()
c0.start()
s1.start( [c0] )
s2.start( [c0] )
s3.start( [c0] )
s4.start( [c0] )

print "*** Iniciando linea de comandos CLI"
CLI( net )
print "*** Finalizando red"
net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()

```

3. Código fuente script SplitFlows.sh

Este script corresponde a lenguaje bash, en el cual se ejecuta una serie de comandos para configurar los flujos o rutas que seguirán los paquetes en la topología Split.py, esto a través de protocolo OpenFlow. En este script se configura el escenario en el cual el tráfico recibido en S1 desde h1, es dividido 50% hacia S2 y 50% hacia S3, generando un divisor de tráfico, y mediante modificaciones del enlace S1-S2, se generan retardos y desorden en la llegada de paquetes hacia h3.

script para configurar flujos o rutas en la topologia Split.py

```
ovs-ofctl -O OpenFlow13 add-group s1 group_id=0,type=select,bucket=weight:50,output=3,bucket=weight:50,output=4
```

```

ovs-ofctl -O OpenFlow13 add-flow s1 in_port=4,ip,nw_dst=10.0.0.1,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s1 in_port=4,ip,nw_dst=10.0.0.2,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s1 in_port=4,dl_type=0x0806,actions=flood
ovs-ofctl -O OpenFlow13 add-flow s1 in_port=1,actions=group=0
ovs-ofctl -O OpenFlow13 add-flow s1 in_port=2,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=1,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=2,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s3 in_port=1,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s3 in_port=2,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s4 in_port=1,actions=output=3

```

```
ovs-ofctl -O OpenFlow13 add-flow s4 in_port=2,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s4 in_port=3,actions=output=2
```

4. Código fuente topología NSFnet-14 (NSFnet14.py)

```
# codigo fuente de topologia NSFnet-14
# host transmisor h1
# 14 conmutadores
# host receptor h2
# controlador c0

# A continuacion se importan las librerias
# necesarias para la creacion e inicio de la red SDN

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, OVSSwitch, UserSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink

# se crea la topología

def topology():
    "Creando red"
    net = Mininet( controller=Controller, link=TCLink, switch=OVSKernelSwitch)

    print "**** Creando nodos ****"

    h1 = net.addHost('h1')
    h2 = net.addHost('h2')

    # dado que se trata de una topologia multicamino
    # se deben configurar los flujos o rutas en los conmutadores
    # a través de protocolo OpenFlow
    # estas rutas se configuran en script NSFnet14Flows.sh

    s1 = net.addSwitch('s1', protocols=["OpenFlow10,OpenFlow13"])
    s2 = net.addSwitch('s2', protocols=["OpenFlow10,OpenFlow13"])
    s3 = net.addSwitch('s3', protocols=["OpenFlow10,OpenFlow13"])
    s4 = net.addSwitch('s4', protocols=["OpenFlow10,OpenFlow13"])
    s5 = net.addSwitch('s5', protocols=["OpenFlow10,OpenFlow13"])
    s6 = net.addSwitch('s6', protocols=["OpenFlow10,OpenFlow13"])
    s7 = net.addSwitch('s7', protocols=["OpenFlow10,OpenFlow13"])
    s8 = net.addSwitch('s8', protocols=["OpenFlow10,OpenFlow13"])
    s9 = net.addSwitch('s9', protocols=["OpenFlow10,OpenFlow13"])
    s10 = net.addSwitch('s10', protocols=["OpenFlow10,OpenFlow13"])
    s11 = net.addSwitch('s11', protocols=["OpenFlow10,OpenFlow13"])
    s12 = net.addSwitch('s12', protocols=["OpenFlow10,OpenFlow13"])
    s13 = net.addSwitch('s13', protocols=["OpenFlow10,OpenFlow13"])
    s14 = net.addSwitch('s14', protocols=["OpenFlow10,OpenFlow13"])

    c0 = net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6633 )

    print "**** Creando enlaces ****"

    # en la linea a continuacion se definen los valores de retardo y perdida de paquetes por enlace
```

```

linkopts0=dict(delay='10ms', loss=10)

net.addLink(h1, s2, cls=TCLink, **linkopts0)
net.addLink(s13, h2, cls=TCLink, **linkopts0)
net.addLink(s1, s2, cls=TCLink, **linkopts0)
net.addLink(s1, s3, cls=TCLink, **linkopts0)
net.addLink(s1, s8, cls=TCLink, **linkopts0)
net.addLink(s2, s4, cls=TCLink, **linkopts0)
net.addLink(s2, s3, cls=TCLink, **linkopts0)
net.addLink(s3, s6, cls=TCLink, **linkopts0)
net.addLink(s4, s5, cls=TCLink, **linkopts0)
net.addLink(s4, s11, cls=TCLink, **linkopts0)
net.addLink(s5, s6, cls=TCLink, **linkopts0)
net.addLink(s5, s7, cls=TCLink, **linkopts0)
net.addLink(s6, s9, cls=TCLink, **linkopts0)
net.addLink(s6, s14, cls=TCLink, **linkopts0)
net.addLink(s7, s8, cls=TCLink, **linkopts0)
net.addLink(s8, s10, cls=TCLink, **linkopts0)
net.addLink(s9, s10, cls=TCLink, **linkopts0)
net.addLink(s10, s13, cls=TCLink, **linkopts0)
net.addLink(s11, s12, cls=TCLink, **linkopts0)
net.addLink(s11, s13, cls=TCLink, **linkopts0)
net.addLink(s12, s14, cls=TCLink, **linkopts0)

print "*** Iniciando red ***"
net.build()
c0.start()
s1.start([c0])
s2.start([c0])
s3.start([c0])
s4.start([c0])
s5.start([c0])
s6.start([c0])
s7.start([c0])
s8.start([c0])
s9.start([c0])
s10.start([c0])
s11.start([c0])
s12.start([c0])
s13.start([c0])
s14.start([c0])

print "*** Iniciando linea de comandos CLI ***"
CLI ( net )

print "*** Finalizando red ***"
net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    topology()

```

5. Código fuente script NSFnet14Flows.sh

Teniendo en cuenta la topología mostrada en la Figura 1, y las rutas definidas para la transmisión del video desde H1 hacia H2, a continuación se presenta el script para la configuración de estos flujos a través de los nodos.

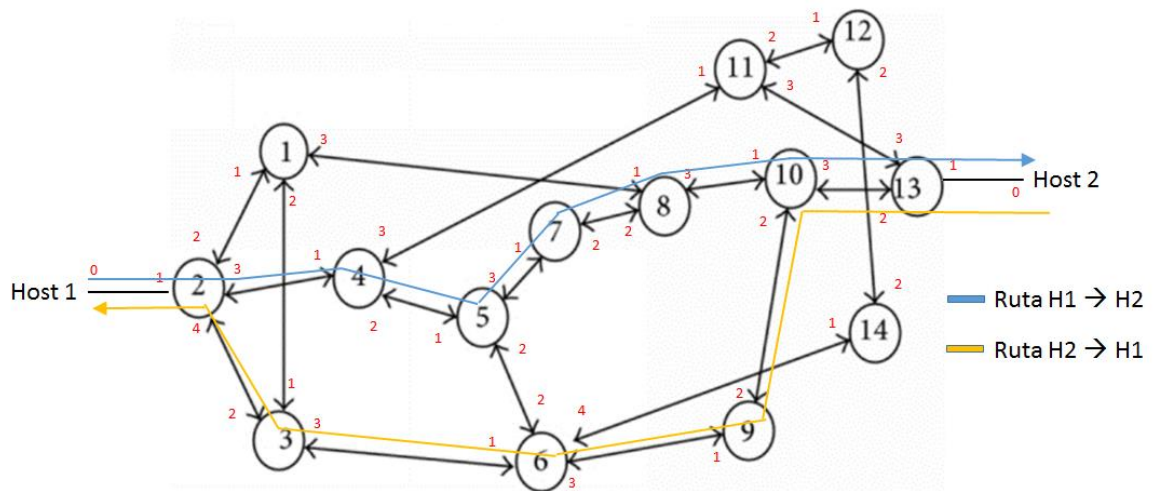


Figura 1. Topología SDN NSFnet14.py en Mininet.

script para configurar flujos o rutas en la topología NSFnet14.py

```
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=4,ip,nw_dst=10.0.0.1,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=4,dl_type=0x0806,actions=flood
```

A continuacion los flujos desde H1 hacia H2

```
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=1,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s4 in_port=1,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s5 in_port=1,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s7 in_port=1,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s8 in_port=2,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s10 in_port=1,actions=output=3
ovs-ofctl -O OpenFlow13 add-flow s13 in_port=2,actions=output=1
```

A continuacion los flujos desde H2 hacia H1

```
ovs-ofctl -O OpenFlow13 add-flow s13 in_port=1,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s10 in_port=3,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s9 in_port=2,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s6 in_port=3,actions=output=1
ovs-ofctl -O OpenFlow13 add-flow s3 in_port=3,actions=output=2
ovs-ofctl -O OpenFlow13 add-flow s2 in_port=4,actions=output=1
```