Mininet-Wifi

Associates
Panagiotis Kontopoulos
pkontopoulous@di.uoa.gr
Alexandros Tsakrilis
alextsak@di.uoa.gr
Mininet-Wifi

- A fork of mininet
- Emulator for Software Defined Networking
- Simple to create topologies and run network scenarios
- [https://github.com/intrig-unicamp/mininet-wifi](https://github.com/intrig-unicamp/mininet-wifi)
- Virtualized Access Points
- Emulates mobility & supports various mobility models
- Remote controllers are also supported
Mininet-Wifi – Basic Classes

- **addHost()**: adds a host to a topology and returns the host name
- **addStation()**: adds a station to a topology and returns the station name
- **addAccessPoint()**: adds an access point to a topology and returns the access point name
- **addLink()**: adds a bidirectional link to a topology
- **addCar()**: adds a car to a topology and returns the car name
Some Background - SDN

APPLICATION LAYER

Business Applications

API

CONTROL LAYER

SDN Control Software

Network Services

Control Data Plane interface (e.g., OpenFlow)

INFRASTRUCTURE LAYER

Network Device

Network Device

Network Device

Network Device
Some Background - OVS

- Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache2.0 license.
- Network automation through programmatic extensions
OVS – Basic commands

- **ovs-vsctl**: Used for configuring the ovs-vswitchd configuration database (known as ovs-db)
- **ovs-ofctl**: A command line tool for monitoring and administering OpenFlow switches
- **ovs-dpctl**: Used to administer Open vSwitch datapaths
- **ovs-appctl**: Used for querying and controlling Open vSwitch daemons
OVS – Examples

- **ovs-ofctl dump-flows sw**: List the flows assigned to the bridge
- **ovs-ofctl show sw**: connects to the switch and prints out port state and OF capabilities
- **ovs-ofctl del-flows sw**: Deletes all flows from the switch
- **ovs-ofctl add-flow sw in_port=1,actions=output:2**: Adds a flow (rule) – port-based.
- **ovs-ofctl mod-flows switch in_port=1,actions=drop**: Modifies a flow entry
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

def topology():
    """Create a network."
    net = Mininet( controller=Controller, link=TCLink, switch=OVSKernelSwitch )

    print "*** Creating nodes"
    stal = net.addStation('stal')
    h1 = net.addHost( 'h1', ip="192.168.10.1/24" )
    ap1 = net.addAccessPoint( 'ap1', ssid="ssid_1", mode="g", channel="1" )
    c0 = net.addController('c0', controller=Controller, ip='127.0.0.1', port=6633 )

    print "*** Configuring wifi nodes"
    net.configureWifiNodes()

    print "*** Adding Link"
    # state the link bandwidth
    linkBW = {'bw':100}
    net.addLink(h1,ap1, cls=TCLink, **linkBW)
    net.addLink(stal, ap1)
Mininet-Wifi: Example - SinglePath

```python
print "*** Starting network"
net.build()
c0.start()
ap1.start( [c0] )

# Set the interface for the virtual device
sta1.cmd('ifconfig sta1-wlan0 192.168.10.10/24 up')
print "*** Running CLI"
CLI( net )
print "*** Stopping network"
net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
topology()
```
Mininet-Wifi: Example - SinglePath

user@mininetwifi:/mininet-wifi/netmanagement$ sudo python singlepath.py
 *** Creating nodes
 *** Configuring wifi nodes
 *** Adding Link
(100.00Mbit) (100.00Mbit) Associating sta1-wlan0 to ap1
 *** Starting network
 *** Configuring hosts
(100.00Mbit) *** Running CLI
 *** Starting CLI:
mininet-wifi>
Mininet-Wifi: Example

```
root@user-VirtualBox:~/mininet-wifi/netman# iperf -s -u -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)

[ 15] local 192.168.10.10 port 5001 connected with 192.168.10.1 port 56801
[ ID] Interval       Transfer  Bandwidth       Jitter    Lost/Total Datagrams
[ 15] 0.0- 1.0 sec  6.16 MBytes   51.7 Mbits/sec  0.453 ms    1810/ 6203 (29%)
[ 15] 1.0- 2.0 sec  6.10 MBytes   51.2 Mbits/sec  0.046 ms    3617/ 7969 (45%)
[ 15] 2.0- 3.0 sec  6.15 MBytes   51.6 Mbits/sec  0.178 ms    3658/ 8044 (45%)
[ 15] 3.0- 4.0 sec  6.16 MBytes   51.7 Mbits/sec  0.130 ms    3673/ 8066 (46%)
[ 15] 4.0- 5.0 sec  6.16 MBytes   51.7 Mbits/sec  0.051 ms    3663/ 8058 (45%)
[ 15] 0.0- 5.2 sec  32.2 MBytes   51.5 Mbits/sec  0.063 ms  17267/40206 (43%)
[ 15] 0.0- 5.2 sec  1 datagrams received out-of-order
```
Mininet-Wifi: Example

```
root@user-VirtualBox:~/mininet-wifi/netman# iperf -c 192.168.10.10 -u -b 100M -t 5

Client connecting to 192.168.10.10, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 160 KByte (default)

[15] local 192.168.10.1 port 56801 connected with 192.168.10.10 port 5001
[15] 0.0- 5.0 sec 56.4 MBytes 94.6 Mbits/sec
[15] Sent 40208 datagrams
[15] Server Report:
[15] 0.0- 5.2 sec 32.2 MBytes 51.5 Mbits/sec 0.063 ms 17257/40208 (43%)
[15] 0.0- 5.2 sec 1 datagrams received out-of-order
```

**Mininet-Wifi: Example – Experiment Setup**

```python
def topology():
    "Create a network."
    net = Mininet(controller=Controller, link=TCLink, switch=OVSKernelSwitch, accessPoint=OVSKernelAP)

    print "*** Creating nodes"
    cars = []
    stas = []
    for x in range(0, 4):
        car.append(x)
        stas.append(x)
    for x in range(0, 4):
        car[x] = net.addCar('car%s' % (x), wans=2, ip='10.0.0.%s/8' % (x + 1), \
                           mac='00:00:00:00:00:00' % x, mode='g', position='%d, %d' % ((120 - (x * 20)), (100 - (x * 0))))

    rsul = net.addAccessPoint('rsul', ssid='rsul', dpid='3000000000000000', mode='g', channel='6', position='160,75,0')
    eNodeB1 = net.addAccessPoint('eNodeB1', ssid='eNodeB1', dpid='1800000000000000', mode='ac', channel='11', position='140,100,0', range=70)
    eNodeB2 = net.addAccessPoint('eNodeB2', ssid='eNodeB2', dpid='2000000000000000', mode='ac', channel='1', position='80,75,0')
    c1 = net.addController('c1', controller=Controller)
    client = net.addHost ('client')
    switch = net.addSwitch ('switch', dpid='4000000000000000')

    net.plotNode(client, position='125,230,0')
    net.plotNode(switch, position='125,200,0')

    print "*** Configuring wifi nodes"
    net.configureWifiNodes()
```
Mininet-Wifi: Example – Creating links and starting network

```
print "*** Creating links"
net.addLink(eNodeB1, switch)
net.addLink(eNodeB2, switch)
net.addLink(rsu1, switch)
net.addLink(switch, client)
net.addLink(rsu1, car[0])
net.addLink(eNodeB2, car[0])
net.addLink(eNodeB1, car[3])
```

```
print "*** Starting network"
net.build()
c1.start()
eNodeB1.start([c1])
eNodeB2.start([c1])
rsu1.start([c1])
switch.start([c1])
for sw in net.vehicles:
    sw.start([c1])
```
Channel bonding enables two or more network interfaces to act as one. It increases the bandwidth and provides redundancy.
Mininet-Wifi: Example - Experiment

```python
print "Moving nodes"
car[0].moveNodeTo('190,100,0')
car[1].moveNodeTo('150,100,0')
car[2].moveNodeTo('120,100,0')
car[3].moveNodeTo('90,100,0')
```

```python
print "applying first rule"
# Insert your code here
# Update the flow table on switch
# Example: os.system('ovs-ofctl mod-flows switch in_port=1,actions=output:3')
```
Project – Download & Installation

- Download the VM image (.ova file) from eclass
  - Download VirtualBox
    (https://www.virtualbox.org/wiki/Downloads)
  - Insert the ova file into VirtualBox
  - Username: mininet, Password: mininet
  - All the tools that you will need are pre-installed
Project - Walkthrough

- 4 cars, 2 eNodeBs, 1 RSU (Road Side Unit), ovs, controller, client (safety center)
- The experiment has 3 phases:
  - **Phase 1**: 3-hop V2V communication between the cars (in-band controlling) & V2I connectivity between car3 and eNodeB1
  - **Phase 2**: V2I communication between car0 and RSU, eNodeB2
  - **Phase 3**: V2I communication between car0 eNodeB2
Project - Tasks

• **Task 1:** Run the scenario described in the paper (*IEEE-Access* 2017 – *From theory to Experimental Evaluation: Resource Management in Software-Defined Vehicular Networks*)
  - **Calculate measurements:** Throughput, Packet loss, jitter, latency for each phase
  - Use bonding of interfaces in car0 if you want

• **Task 2:** Run in the same topology but with one car using bicasting
  - **Calculate measurements:** Throughput, Packet loss, jitter, latency
Project – Useful Links

- [http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt](http://openvswitch.org/support/dist-docs/ovs-ofctl.8.txt)
- [https://wiki.videolan.org/Documentation:Command_line/](https://wiki.videolan.org/Documentation:Command_line/)
- [https://www.es.net/assets/Uploads/201007-JTIPerf.pdf](https://www.es.net/assets/Uploads/201007-JTIPerf.pdf)
- [https://openmaniak.com/ipperf.php](https://openmaniak.com/ipperf.php)
- [https://www.youtube.com/user/ramonfontes](https://www.youtube.com/user/ramonfontes)
Questions ?