import random

class AccessPoint:

def \_\_init\_\_(self, name, mac, ip, x, y, ssid, password, range\_, channel):

self.name = name

self.mac = mac

self.ip = ip

self.coordinates = (x, y)

self.ssid = ssid

self.password = password

self.range = range\_

self.channel = channel

class Station:

def \_\_init\_\_(self, name, mac, ip, x, y):

self.name = name

self.mac = mac

self.ip = ip

self.coordinates = (x, y)

def check\_dead\_spot(x, y, dead\_spot):

return (dead\_spot[0][0] <= x <= dead\_spot[1][0]) and (dead\_spot[0][1] <= y <= dead\_spot[1][1])

# Define AP and STA information

ap\_data = [

("AP1", "00:00:00:00:00:01", "192.168.0.1", 10, 10, "studentID", "password1", 35, 1),

("AP2", "00:00:00:00:00:02", "192.168.0.2", 30, 10, "studentID", "password2", 35, 1),

("AP3", "00:00:00:00:00:03", "192.168.0.3", 50, 10, "studentID", "password3", 35, 6),

("AP4", "00:00:00:00:00:04", "192.168.0.4", 70, 10, "studentID", "password4", 50, 11),

("AP5", "00:00:00:00:00:05", "192.168.0.5", 90, 10, "studentID", "password5", 50, 11)

]

sta\_data = [

("STA1", "00:00:00:00:01:01", "192.168.0.10", 15, 15),

("STA2", "00:00:00:00:01:02", "192.168.0.11", 35, 15),

("STA3", "00:00:00:00:01:03", "192.168.0.12", 55, 15)

]

# WiFi dead spot coordinates

dead\_spot\_area = [(40, 0), (60, 20)] # Example dead spot area from (40,0) to (60,20)

# Instantiate APs and Stations

access\_points = [AccessPoint(\*data) for data in ap\_data]

stations = [Station(\*data) for data in sta\_data]

# Emulate environment

def emulate\_environment(aps, stas, dead\_spot):

print("Emulating environment with Access Points and Stations\n")

for ap in aps:

in\_dead\_spot = check\_dead\_spot(\*ap.coordinates, dead\_spot)

print(f"AP {ap.name} at {ap.coordinates} | Range: {ap.range}m | In Dead Spot: {in\_dead\_spot}")

for sta in stas:

closest\_ap = min(aps, key=lambda ap: ((sta.coordinates[0] - ap.coordinates[0]) \*\* 2 + (sta.coordinates[1] - ap.coordinates[1]) \*\* 2) \*\* 0.5)

print(f"{sta.name} at {sta.coordinates} connects to {closest\_ap.name} at {closest\_ap.coordinates}")

# Run the emulation

emulate\_environment(access\_points, stations, dead\_spot\_area)

A1ha59%%

from mininet.node import Controller, RemoteController, OVSKernelAP from mininet.link import adhoc from mininet.log import setLogLevel, info from mn\_wifi.net import Mininet\_wifi from mn\_wifi.node import Station def topology(): net = Mininet\_wifi(controller=Controller, accessPoint=OVSKernelAP) # Adding stations with different ad hoc protocols info("\*\*\* Creating stations\n") sta1 = net.addStation('sta1', ip6='2001:db8:0:1::1/64', position='30,30,0') sta2 = net.addStation('sta2', ip6='2001:db8:0:1::2/64', position='40,40,0') sta3 = net.addStation('sta3', ip6='2001:db8:0:1::3/64', position='50,50,0') # Ad hoc links with protocol specifications net.addLink(sta1, cls=adhoc, ssid='adhocUH', mode='g', channel=5) net.addLink(sta2, cls=adhoc, ssid='adhocUH', mode='g', channel=5) net.addLink(sta3, cls=adhoc, ssid='adhocUH', mode='g', channel=5) # Starting network info("\*\*\* Starting network\n") net.build() # Assign protocols to each station sta1.cmd('batmand -c -d 1 &') # Start batman sta2.cmd('modprobe batman-adv') # Start batman\_adv sta3.cmd('olsrd -d 2 &') # Start olsrd # Ping between nodes info("\*\*\* Testing connectivity\n") net.pingAll() # Capture ICMP traffic and initiate VoIP traffic info("\*\*\* Running VoIP simulation\n") sta1.cmd('ping6 -c 10 2001:db8:0:1::2 &') # ICMP test sta1.cmd('iperf -u -c 2001:db8:0:1::2 -t 120 &') # UDP VoIP test # Stop network net.stop() if \_\_name\_\_ == '\_\_main\_\_': setLogLevel('info') topology()

from mininet.node import Controller, RemoteController, OVSKernelAP

from mininet.link import adhoc

from mininet.log import setLogLevel, info

from mn\_wifi.net import Mininet\_wifi

from mn\_wifi.node import Station

def topology():

net = Mininet\_wifi(controller=Controller, accessPoint=OVSKernelAP)

# Adding stations with different ad hoc protocols

info("\*\*\* Creating stations\n")

sta1 = net.addStation('sta1', ip6='2001:db8:0:1::1/64', position='30,30,0')

sta2 = net.addStation('sta2', ip6='2001:db8:0:1::2/64', position='40,40,0')

sta3 = net.addStation('sta3', ip6='2001:db8:0:1::3/64', position='50,50,0')

# Ad hoc links with protocol specifications

net.addLink(sta1, cls=adhoc, ssid='adhocUH', mode='g', channel=5)

net.addLink(sta2, cls=adhoc, ssid='adhocUH', mode='g', channel=5)

net.addLink(sta3, cls=adhoc, ssid='adhocUH', mode='g', channel=5)

# Starting network

info("\*\*\* Starting network\n")

net.build()

# Assign protocols to each station

sta1.cmd('batmand -c -d 1 &') # Start batman

sta2.cmd('modprobe batman-adv') # Start batman\_adv

sta3.cmd('olsrd -d 2 &') # Start olsrd

# Ping between nodes

info("\*\*\* Testing connectivity\n")

net.pingAll()

# Capture ICMP traffic and initiate VoIP traffic

info("\*\*\* Running VoIP simulation\n")

sta1.cmd('ping6 -c 10 2001:db8:0:1::2 &') # ICMP test

sta1.cmd('iperf -u -c 2001:db8:0:1::2 -t 120 &') # UDP VoIP test

# Stop network

net.stop()

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

setLogLevel('info')

topology()