**#include "ns3/core-module.h"**

**#include "ns3/global-route-manager.h"**

**#include "ns3/wifi-module.h"**

**#include "ns3/mobility-module.h"**

**#include "ns3/bridge-module.h"**

**#include "ns3/point-to-point-module.h"**

**#include "ns3/network-module.h"**

**#include "ns3/applications-module.h"**

**#include "ns3/internet-module.h"**

**#include "ns3/netanim-module.h"**

using namespace ns3;

**NS\_LOG\_COMPONENT\_DEFINE ("WifiScriptExample");**

int main(int argc, char \*argv[])

{

CommandLine cmd;

cmd.Parse (argc, argv);

/\* Configuration. \*/

**LogComponentEnable ("UdpEchoClientApplication", LOG\_LEVEL\_INFO);**

**LogComponentEnable ("UdpEchoServerApplication", LOG\_LEVEL\_INFO);**

**std::string animFile="exp1.xml";**

/\* Build nodes. \*/

NodeContainer station\_0;

station\_0.Create (1);

NodeContainer station\_1;

station\_1.Create (1);

NodeContainer ap\_0;

ap\_0.Create (1);

NodeContainer term\_0;

term\_0.Create (1);

NodeContainer term\_1;

term\_1.Create (1);

/\* Build link. \*/

YansWifiPhyHelper wifiPhy\_ap\_0 = YansWifiPhyHelper::Default ();

YansWifiChannelHelper wifiChannel\_ap\_0 = YansWifiChannelHelper::Default ();

wifiPhy\_ap\_0.SetChannel (wifiChannel\_ap\_0.Create ());

PointToPointHelper p2p\_p2p\_0;

p2p\_p2p\_0.SetDeviceAttribute ("DataRate", DataRateValue (100000000));

p2p\_p2p\_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (**10**)));

PointToPointHelper p2p\_p2p\_1;

p2p\_p2p\_1.SetDeviceAttribute ("DataRate", DataRateValue (100000000));

p2p\_p2p\_1.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (**10**)));

/\* Build link net device container. \*/

NodeContainer all\_ap\_0;

NetDeviceContainer ndc\_ap\_0;

Ssid ssid\_ap\_0 = Ssid ("**ns-3-ssid**");

WifiHelper wifi\_ap\_0 = WifiHelper::Default ();

NqosWifiMacHelper wifiMac\_ap\_0 = NqosWifiMacHelper::Default ();

wifi\_ap\_0.SetRemoteStationManager ("ns3::**ArfWifiManager**");

/\*wifiMac\_ap\_0.SetType ("ns3::NqapWifiMac",

"Ssid", SsidValue (ssid\_ap\_0),

"BeaconGeneration", BooleanValue (true),

"BeaconInterval", TimeValue (Seconds (2.5)));\*/

**wifiMac\_ap\_0.SetType ("ns3::StaWifiMac","Ssid",SsidValue (ssid\_ap\_0),"ActiveProbing",BooleanValue (false));**

ndc\_ap\_0.Add (wifi\_ap\_0.Install (wifiPhy\_ap\_0, wifiMac\_ap\_0, **station\_0**));

**ndc\_ap\_0.Add (wifi\_ap\_0.Install (wifiPhy\_ap\_0, wifiMac\_ap\_0, station\_1));**

/\*wifiMac\_ap\_0.SetType ("ns3::NqstaWifiMac",

"Ssid", SsidValue (ssid\_ap\_0),

"ActiveProbing", BooleanValue (false));\*/

**wifiMac\_ap\_0.SetType ("ns3::ApWifiMac","Ssid",SsidValue (ssid\_ap\_0));**

ndc\_ap\_0.Add (wifi\_ap\_0.Install (wifiPhy\_ap\_0, wifiMac\_ap\_0, ap\_0 ));

MobilityHelper mobility\_ap\_0;

**mobility\_ap\_0.SetPositionAllocator ("ns3::GridPositionAllocator",**

**"MinX", DoubleValue (0.0),**

**"MinY", DoubleValue (0.0),**

**"DeltaX", DoubleValue (5.0),**

**"DeltaY", DoubleValue (10.0),**

**"GridWidth", UintegerValue (3),**

**"LayoutType", StringValue ("RowFirst"));**

**mobility\_ap\_0.SetMobilityModel ("ns3::RandomWalk2dMobilityModel","Bounds",**

**RectangleValue (Rectangle (-50, 50, -50, 50)));**

**mobility\_ap\_0.Install (station\_0);**

**mobility\_ap\_0.Install (station\_1);**

mobility\_ap\_0.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

mobility\_ap\_0.Install (ap\_0);

mobility\_ap\_0.Install(all\_ap\_0);

NodeContainer all\_p2p\_0;

all\_p2p\_0.Add (term\_0);

**all\_p2p\_0.Add (ap\_0);**

NetDeviceContainer ndc\_p2p\_0 = p2p\_p2p\_0.Install (all\_p2p\_0);

NodeContainer all\_p2p\_1;

all\_p2p\_1.Add (term\_0);

all\_p2p\_1.Add (term\_1);

NetDeviceContainer ndc\_p2p\_1 = p2p\_p2p\_1.Install (all\_p2p\_1);

/\* Install the IP stack. \*/

InternetStackHelper internetStackH;

internetStackH.Install (station\_0);

internetStackH.Install (station\_1);

internetStackH.Install (ap\_0);

internetStackH.Install (term\_0);

internetStackH.Install (term\_1);

/\* IP assign. \*/

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.0.0.0", "255.255.255.0");

Ipv4InterfaceContainer iface\_ndc\_ap\_0 = ipv4.Assign (ndc\_ap\_0);

ipv4.SetBase ("10.0.1.0", "255.255.255.0");

Ipv4InterfaceContainer iface\_ndc\_p2p\_0 = ipv4.Assign (ndc\_p2p\_0);

ipv4.SetBase ("10.0.2.0", "255.255.255.0");

Ipv4InterfaceContainer iface\_ndc\_p2p\_1 = ipv4.Assign (ndc\_p2p\_1);

/\* Generate Route. \*/

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

/\* Generate Application. \*/

uint16\_t port\_udpEcho\_0 = 123;

UdpEchoServerHelper server\_udpEcho\_0 (port\_udpEcho\_0);

ApplicationContainer apps\_udpEcho\_0 = server\_udpEcho\_0.Install (term\_1.Get(0));

apps\_udpEcho\_0.Start (Seconds (0.0));

apps\_udpEcho\_0.Stop (Seconds (5.0));

Time interPacketInterval\_udpEcho\_0 = Seconds (1.0);

UdpEchoClientHelper client\_udpEcho\_0 (iface\_ndc\_p2p\_1.GetAddress(1), 123);

client\_udpEcho\_0.SetAttribute ("MaxPackets", UintegerValue (1));

client\_udpEcho\_0.SetAttribute ("Interval", TimeValue (interPacketInterval\_udpEcho\_0));

client\_udpEcho\_0.SetAttribute ("PacketSize", UintegerValue (1024));

apps\_udpEcho\_0 = client\_udpEcho\_0.Install (station\_0.Get (0));

apps\_udpEcho\_0.Start (Seconds (0.1));

apps\_udpEcho\_0.Stop (Seconds (5.0));

/\* Simulation. \*/

**AnimationInterface anim(animFile);**

/\* Pcap output. \*/

/\* Stop the simulation after x seconds. \*/

uint32\_t stopTime = 6;

Simulator::Stop (Seconds (stopTime));

/\* Start and clean simulation. \*/

Simulator::Run ();

Simulator::Destroy ();

}