1 Задание

Построить иммитационую модель мобильной WIMAX-сети, с возможность передвижения со скоростью 100 км/ч без потери связи.

2 Описание используемого ПО

Для разработки исользовалось программое обеспечение NS-3(https://www.nsnam.org/).

3 Описание модели

Разработанная модель состоит из четырех узлов:

- Мобильная станция(NodeContainer ssNodes;).
- Базовая станция(NodeContainer bsNodes;) и узел ASN-щлюза (NodeContainer ASN_Node;), которые составляют Access Service Network(Сеть доступа).
- узла CSN(NodeContainer CSN_Node;).

4 Код разработанной модели

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wimax-module.h"
#include "ns3/csma-module.h"
#include <iostream>
#include "ns3/global-route-manager.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/vector.h"

NS_LOG_COMPONENT_DEFINE ("wimax");

using namespace ns3;

int main (int argc, char *argv[])
{
  int duration = 10;
  NodeContainer ssNodes;//узел мобильной станции
  Ptr<SubscriberStationNetDevice> ss;
  NetDeviceContainer ssDevs;
  Ipv4InterfaceContainer SSinterfaces;
  NodeContainer bsNodes;// узел базовой станции
  Ptr<BaseStationNetDevice> bs;
```

```
NetDeviceContainer bsDevs, bsDevsOne;
Ipv4InterfaceContainer BSinterfaces;
Ptr<SimpleOfdmWimaxChannel> channel;
NodeContainer CSN_Node;//узлы CSN
NodeContainer ASN_Node;//узлы asn
Ptr<ConstantPositionMobilityModel> BSPosition;
Ptr<RandomWaypointMobilityModel> SSPosition;
Ptr<RandomRectanglePositionAllocator> SSPosAllocator;

WimaxHelper::SchedulerType scheduler = WimaxHelper::SCHED_TYPE_SIMPLE;

LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

ssNodes.Create (1);
bsNodes.Create (1);

CSN_Node.Create (1);
ASN_Node.Create (1);

WimaxHelper wimax;

channel = CreateObject<SimpleOfdmWimaxChannel> ();
channel->SetPropagationModel (SimpleOfdmWimaxChannel::COST231_PROPAGATION);
ssDevs = wimax.Install (ssNodes,
                        WimaxHelper::DEVICE_TYPE_SUBSCRIBER_STATION,
                        WimaxHelper::SIMPLE_PHY_TYPE_OFDM,
                        channel,
                        scheduler);
Ptr<WimaxNetDevice> dev = wimax.Install (bsNodes.Get (0),
                                         WimaxHelper::DEVICE_TYPE_BASE_STATION,
                                         WimaxHelper::SIMPLE_PHY_TYPE_OFDM,
                                         channel,
                                         scheduler);
bsDevs.Add (dev);

SSPosition = CreateObject<RandomWaypointMobilityModel> ();
SSPosAllocator = CreateObject<RandomRectanglePositionAllocator> ();
Ptr<UniformRandomVariable> xVar = CreateObject<UniformRandomVariable> ();
xVar->SetAttribute ("Min", DoubleValue (0));
xVar->SetAttribute ("Max", DoubleValue (1000));
SSPosAllocator->SetX (xVar);
Ptr<UniformRandomVariable> yVar = CreateObject<UniformRandomVariable> ();
yVar->SetAttribute ("Min", DoubleValue (0));
yVar->SetAttribute ("Max", DoubleValue (0));
SSPosAllocator->SetY (yVar);
SSPosition->SetAttribute ("PositionAllocator", PointerValue (SSPosAllocator));
SSPosition->SetAttribute ("Speed", StringValue ("ns3::ConstantRandomVariable[Constant=30]"));
SSPosition->SetAttribute ("Pause", StringValue ("ns3::ConstantRandomVariable[Constant=0.01]"));

ss = ssDevs.Get (0)->GetObject<SubscriberStationNetDevice> ();
ss->SetModulationType (WimaxPhy::MODULATION_TYPE_QAM16_12);
ssNodes.Get (0)->AggregateObject (SSPosition);
bs = bsDevs.Get (0)->GetObject<BaseStationNetDevice> ();
CsmaHelper csmaASN_BS;
CsmaHelper csmaCSN_ASN;

NodeContainer LAN_ASN_BS;

LAN_ASN_BS.Add (bsNodes.Get (0));

LAN_ASN_BS.Add (ASN_Node.Get (0));

csmaASN_BS.SetChannelAttribute ("DataRate", DataRateValue (DataRate (10000000)));
csmaASN_BS.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

NetDeviceContainer LAN_ASN_BS_Devs = csmaASN_BS.Install (LAN_ASN_BS);

NetDeviceContainer BS_CSMADevs;

BS_CSMADevs.Add (LAN_ASN_BS_Devs.Get (0));

NetDeviceContainer ASN_Devs1;
ASN_Devs1.Add (LAN_ASN_BS_Devs.Get (1));

NodeContainer LAN_ASN_CSN;
LAN_ASN_CSN.Add (ASN_Node.Get (0));
LAN_ASN_CSN.Add (CSN_Node.Get (0));

csmaCSN_ASN.SetChannelAttribute ("DataRate", DataRateValue (DataRate (10000000)));
csmaCSN_ASN.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

NetDeviceContainer LAN_ASN_CSN_Devs = csmaCSN_ASN.Install (LAN_ASN_CSN);

NetDeviceContainer CSN_Devs;
NetDeviceContainer ASN_Devs2;
ASN_Devs2.Add (LAN_ASN_CSN_Devs.Get (0));
CSN_Devs.Add (LAN_ASN_CSN_Devs.Get (1));

MobilityHelper mobility;
InternetStackHelper stack;
mobility.Install (bsNodes);
stack.Install (bsNodes);
mobility.Install (ssNodes);
stack.Install (ssNodes);
stack.Install (CSN_Node);
stack.Install (ASN_Node);

Ipv4AddressHelper address;

address.SetBase ("192.168.1.0", "255.255.255.0");
bsDevsOne.Add (bs);
BSinterfaces = address.Assign (bsDevsOne);
SSinterfaces = address.Assign (ssDevs);

address.SetBase ("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer BSCSMAInterfaces = address.Assign (BS_CSMADevs);
Ipv4InterfaceContainer ASNCSMAInterfaces1 = address.Assign (ASN_Devs1);

address.SetBase ("192.168.3.0", "255.255.255.0");
Ipv4InterfaceContainer ASNCSMAInterfaces2 = address.Assign (ASN_Devs2);
Ipv4InterfaceContainer CSNCSMAInterfaces = address.Assign (CSN_Devs);
```

```
Ipv4Address multicastGroup ("224.30.10.81");

Ipv4StaticRoutingHelper multicast;

Ptr<Node> multicastRouter = ASN_Node.Get (0);
Ptr<NetDevice> inputIf = ASN_Devs2.Get (0);

multicast.AddMulticastRoute (multicastRouter, CSNCSMAInterfaces.GetAddress(0), multicastGroup,
inputIf, ASN_Devs1);

Ptr<Node> sender = CSN_Node.Get (0);
Ptr<NetDevice> senderIf = CSN_Devs.Get (0);
multicast.SetDefaultMulticastRoute (sender, senderIf);

multicastRouter = bsNodes.Get (0);
inputIf = BS_CSMADevs.Get (0);

multicast.AddMulticastRoute (multicastRouter, SSinterfaces.GetAddress(0), multicastGroup,
inputIf, bsDevsOne);

Ipv4StaticRoutingHelper multicast1;

Ptr<Node> multicastRouter1 = bsNodes.Get (0);
Ptr<NetDevice> inputIf1 = bsDevsOne.Get (0);

multicast1.AddMulticastRoute (multicastRouter1, SSinterfaces.GetAddress(0), multicastGroup,
inputIf1, BS_CSMADevs);

Ptr<Node> sender1 = ssNodes.Get (0);
Ptr<NetDevice> senderIf1 = ssDevs.Get (0);
multicast1.SetDefaultMulticastRoute (sender1, senderIf1);

multicastRouter1 = ASN_Node.Get (0);
inputIf1 = ASN_Devs1.Get (0);

multicast1.AddMulticastRoute (multicastRouter1, CSNCSMAInterfaces.GetAddress(0), multicastGroup,
inputIf1, ASN_Devs2);

// настройка приложений на сервере
UdpServerHelper udpServer;
ApplicationContainer serverApps;
UdpClientHelper udpClient;
ApplicationContainer clientApps;

udpServer = UdpServerHelper (100);

serverApps = udpServer.Install (CSN_Node.Get (0));
serverApps.Start (Seconds (1));
serverApps.Stop (Seconds (duration));

udpClient = UdpClientHelper (multicastGroup, 100);
udpClient.SetAttribute ("MaxPackets", UintegerValue (3));
udpClient.SetAttribute ("Interval", TimeValue (Seconds (0.5)));
udpClient.SetAttribute ("PacketSize", UintegerValue (1024));

clientApps = udpClient.Install (ssNodes.Get (0));
```

```
  clientApps.Start (Seconds (3));
  clientApps.Stop (Seconds (duration));

  UdpServerHelper udpServer1;
  ApplicationContainer serverApps1;
  UdpClientHelper udpClient1;
  ApplicationContainer clientApps1;

  udpServer1 = UdpServerHelper (99);

  serverApps1 = udpServer1.Install (ssNodes.Get (0));
  serverApps1.Start (Seconds (1));
  serverApps1.Stop (Seconds (duration));

  udpClient1 = UdpClientHelper (multicastGroup, 99);
  udpClient1.SetAttribute ("MaxPackets", UintegerValue (3));
  udpClient1.SetAttribute ("Interval", TimeValue (Seconds (0.5)));
  udpClient1.SetAttribute ("PacketSize", UintegerValue (1024));

  clientApps1 = udpClient1.Install (CSN_Node.Get (0));
  clientApps1.Start (Seconds (5));
  clientApps1.Stop (Seconds (duration));

  IpcsClassifierRecord MulticastClassifier (BSinterfaces.GetAddress(0),
                                            Ipv4Mask ("255.255.255.255"),
                                            SSinterfaces.GetAddress(0),
                                            Ipv4Mask ("255.255.255.255"),
                                            101,
                                            65000,
                                            0,
                                            100,
                                            17,
                                            1);

  ServiceFlow MulticastServiceFlow = wimax.CreateServiceFlow (ServiceFlow::SF_DIRECTION_DOWN,
                                                              ServiceFlow::SF_TYPE_UGS,
                                                              MulticastClassifier);

  ss->AddServiceFlow (MulticastServiceFlow);

  bsNodes.Get(0)->GetObject<MobilityModel>()->SetPosition(Vector(0,0,0));
  ssNodes.Get(0)->GetObject<MobilityModel>()->SetPosition(Vector(100,0,0));

  Simulator::Stop (Seconds (duration));

  Simulator::Run ();

  Ptr<MobilityModel> mob = ssNodes.Get(0)->GetObject<MobilityModel>();
  Vector pos = mob->GetPosition ();
  std::cout << "POS: x=" << pos.x << ", y=" << pos.y << std::endl;

  Simulator::Destroy ();
  return 0;
}
```