Communication Systems

CCE 3301

examples/wireless/wifi-simple-infra.cc

10.1.1.2

10.1.1.1

Using Pointers to Create the required Nodes

- Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue ("2200"));
- Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("2200"));
- Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue (phyMode));

For the class ns3::WifiRemoteStationManager set:
- the Fragmentation Threshold,
- the RTS/CTS threshold
- the PHY mode for non-unicast mode

For a list of parameters you can set check:
http://www.nsnam.org/doxygen-release/classns3_1_1_wifi_remote_station_manager.html
section GetTypeId

The PHY Mode

DsssRate1Mbps	OfdmRate6MbpsBW10MHz
DsssRate2Mbps	OfdmRate9MbpsBW10MHz
DsssRate5_5Mbps	OfdmRate12MbpsBW10MHz
DsssRate11Mbps	OfdmRate18MbpsBW10MHz
OfdmRate6Mbps	OfdmRate24MbpsBW10MHz
OfdmRate9Mbps	OfdmRate27MbpsBW10MHz
OfdmRate12Mbps	OfdmRate1_5MbpsBW5MHz
OfdmRate18Mbps	OfdmRate2_25MbpsBW5MHz
OfdmRate24Mbps	OfdmRate3MbpsBW5MHz
OfdmRate36Mbps	OfdmRate4_5MbpsBW5MHz
OfdmRate48Mbps	OfdmRate6MbpsBW5MHz
OfdmRate54Mbps	OfdmRate9MbpsBW5MHz
OfdmRate3MbpsBW10MHz	OfdmRate12MbpsBW5MHz
OfdmRate4_5MbpsBW10MHz	OfdmRate13_5MbpsBW5MHz

Choose one according to the IEEE 802.11 standard that you will be using

NodeContainer

- NodeContainer c;
- c.Create (2);

2 nodes are required, one for the STA and one for the AP

WifiHelper

Use a WifiHelper to install the PHY and MAC layers on a node:
- WifiHelper wifi;

Define the Standard:
- wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

All objects are configured for 802.11a unless ns3::WifiHelper::SetStandard is used

Possible Standards:
WIFI_PHY_STANDARD_80211a,
WIFI_PHY_STANDARD_80211b,
WIFI_PHY_STANDARD_80211_10Mhz,
WIFI_PHY_STANDARD_80211_5Mhz,
WIFI_PHY_STANDARD_holland,
WIFI_PHY_STANDARD_80211p_CCH,
WIFI_PHY_STANDARD_80211p_SCH,
WIFI_PHY_UNKNOWN

The PHY Layer

YANS stands for Yet Another Network Simulator

- YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

This helper will allow to set the parameters for the PHY layer

Set the Reception Gain

- wifiPhy.Set ("RxGain", DoubleValue (0));

In this example: we will define a Fixed RSS Loss Model, therefore RxGain must be 0

The PHY Layer

The following command will affect the level of information you will obtain in the PCAP traces

- wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

Possible Settings:

DLT_IEEE802_11: IEEE 802.11 Wireless LAN headers on packets
DLT_PRISM_HEADER: Include Prism monitor mode information
DLT_IEEE802_11_RADIO: Include Radiotap link layer information

The Channel

Set the properties for the Channel

- The Propagation Delay
- wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel")

Setting the speed to be equivalent to the speed of light
You may use a Random Propagation Delay Model – ns3::RandomPropagationDelayModel

- The Propagation Loss Model
- wifiChannel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue(rss));
- wifiPhy.SetChannel (wifiChannel.Create ());

Propagation Loss Models

- ns3::Cost231PropagationLossModel
- ns3::JakesPropagationLossModel,
- ns3::RandomPropagationLossModel
- ns3::FriisPropagationLossModel
- ns3::TwoRayGroundPropagationLossModel
- ns3::LogDistancePropagationLossModel
- ns3::ThreeLogDistancePropagationLossModel
- ns3::NakagamiPropagationLossModel
- ns3::FixedRssLossModel
- ns3::MatrixPropagationLossModel
- ns3::RangePropagationLossModel

The MAC Layer

Creating a Non QoS MAC Layer for a ns3::WifiNetDevice

- NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();

Default Setting: Adhoc Mode

- wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
  "DataMode",StringValue(phyMode),
  "ControlMode",StringValue(phyMode));

Here we are defining a Constant Rate MAC Layer – however there are other Rate Models that can be used

We are also setting the PHY Mode for the Data Packets and the Control Packets

The MAC Layer at the STA

Set the service Set ID i.e. the name which identifies a particular 802.11 wireless LAN

- Ssid ssid = Ssid ("wifi-default");

ns3::NqstaWifiMac - a non QOS STA state machine

STATION:

- wifiMac.SetType ("ns3::NqstaWifiMac",
  "Ssid", SsidValue (ssid),
  "ActiveProbing", BooleanValue (false));

A station sends a probe request frame when it needs to obtain information from another station

The first node is set as the STA

- NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, c.Get(0));

The MAC Layer at the AP

A non-QOS AP:

- wifiMac.SetType ("ns3::NqapWifiMac", "Ssid", SsidValue (ssid));

The last node will be the AP

- NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, c.Get(1));

The NetDeviceContainer devices has the two Net Devices

The Mobility Model

Set the Location of each node

- MobilityHelper mobility;
- Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();

creating a pointer of type ns3::ListPositionAllocator
ns3::ListPositionAllocator - Allocate positions from a deterministic list specified by the user

- positionAlloc->Add (Vector (0.0, 0.0, 0.0));
- positionAlloc->Add (Vector (5.0, 0.0, 0.0));

This is a 3D vector
Two nodes, therefore two vectors are required

- mobility.SetPositionAllocator (positionAlloc);

The devices will not roam around:

- mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
- mobility.Install (c);

InternetStackHelper and IP Addresses

aggregate IP/TCP/UDP functionality to all the nodes

- InternetStackHelper internet;
- internet.Install (c);

- Ipv4AddressHelper ipv4;
- Ipv4.SetBase ("10.1.1.0", "255.255.255.0");
- Ipv4InterfaceContainer i = ipv4.Assign (devices);

Sink – using Sockets

Get the TypeId for ns3::UdpSocketFactory

- TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");

The first node is the STA and it will act as a sink

On node 0 create an ns3::UdpSocketFactory

- Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);
- InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);

Ipv4Address::GetAny() – will get the address 0.0.0.0

- recvSink->Bind (local);
- recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Therefore, node 0 will receive packets from any IP Address
When it receives a packet, the function ReceivePacket will be called

ReceivePacket

- void ReceivePacket (Ptr<Socket> socket)
- {
- NS_LOG_UNCOND ("Received one packet!");
- }

Source – using Sockets

Node 1 will be a source of type ns3::UdpSocketFactory
This node will broadcast data on port 80

- Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);
- InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);
- source->SetAllowBroadcast (true);
- source->Connect (remote);

Using the ScheduleWithContext

- Simulator::ScheduleWithContext (source->GetNode ()->GetId (),

the unique id of the node the source socket is connected to

- Seconds (1.0),
- &GenerateTraffic,
- source,
- packetSize,
- numPackets,
- interPacketInterval);

Call the function GenerateTraffic, which requires four inputs:

source, packetSize, numPackets, interPacketInterval

GenerateTraffic

- static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize, uint32_t pktCount, Time pktInterval)
- {
- if (pktCount > 0)
- {

send a dummy packet of size pktSize

- socket->Send (Create<Packet> (pktSize));

calling itself with a decremented pktCount

- Simulator::Schedule (pktInterval, &GenerateTraffic, socket, pktSize,pktCount-1, pktInterval);
- }
- else
- {
- socket->Close ();
- }
- }

Assignment

Create a network consisting of 3 STAs and an AP

The STAs are located at:
(0.0, 0.0, 0.0),
(-10.0, 1.0, 0.0),
(-10.0, -1.0, 0.0)

The AP is located at:
(5.0, 0.0, 0.0)

Use:
- ns3::Cost231PropagationLossModel
- ns3::FriisPropagationLossModel

Find at what position node 3 must be in order not to receive the packet from the AP

Use:
- DsssRate1Mbps for Control Packets
- DsssRate11Mbps for Data Packets

Use only **PcapTracing**