

third.cc

Go to the documentation of this file.

```

1  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2  /*
3   * This program is free software; you can redistribute it and/or modify
4   * it under the terms of the GNU General Public License version 2 as
5   * published by the Free Software Foundation;
6   *
7   * This program is distributed in the hope that it will be useful,
8   * but WITHOUT ANY WARRANTY; without even the implied warranty of
9   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10  * GNU General Public License for more details.
11  *
12  * You should have received a copy of the GNU General Public License
13  * along with this program; if not, write to the Free Software
14  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
15  */
16
17 #include "ns3/core-module.h"
18 #include "ns3/point-to-point-module.h"
19 #include "ns3/network-module.h"
20 #include "ns3/applications-module.h"
21 #include "ns3/mobility-module.h"
22 #include "ns3/csma-module.h"
23 #include "ns3/internet-module.h"
24 #include "ns3/yans-wifi-helper.h"
25 #include "ns3/ssid.h"
26
27 // Default Network Topology
28 //
29 //   Wifi 10.1.3.0
30 //   *       *       *       *
31 //   |       |       |       |
32 //   |       |       |       | 10.1.1.0
33 // n5  n6  n7  n0 ----- n1  n2  n3  n4
34 //                point-to-point | | | |
35 //                                =====
36 //                                LAN 10.1.2.0
37
38 using namespace ns3;
39
40 NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
41
42 int
43 main (int argc, char *argv[])
44 {
45     bool verbose = true;
46     uint32_t nCsma = 3;
47     uint32_t nWifi = 3;
48     bool tracing = false;
49
50     CommandLine cmd (__FILE__);
51     cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
52     cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
53     cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
54     cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
55
56     cmd.Parse (argc,argv);
57
58     // The underlying restriction of 18 is due to the grid position
59     // allocator's configuration; the grid layout will exceed the
60     // bounding box if more than 18 nodes are provided.
61     if (nWifi > 18)
62     {
63         std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the bounding
64 box" << std::endl;
65         return 1;
66     }
67     if (verbose)

```

```

68 {
69     LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
70     LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
71 }
72
73 NodeContainer p2pNodes;
74 p2pNodes.Create (2);
75
76 PointToPointHelper pointToPoint;
77 pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
78 pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
79
80 NetDeviceContainer p2pDevices;
81 p2pDevices = pointToPoint.Install (p2pNodes);
82
83 NodeContainer csmaNodes;
84 csmaNodes.Add (p2pNodes.Get (1));
85 csmaNodes.Create (nCsma);
86
87 CsmaHelper csma;
88 csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
89 csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
90
91 NetDeviceContainer csmaDevices;
92 csmaDevices = csma.Install (csmaNodes);
93
94 NodeContainer wifiStaNodes;
95 wifiStaNodes.Create (nWifi);
96 NodeContainer wifiApNode = p2pNodes.Get (0);
97
98 YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
99 YansWifiPhyHelper phy;
100 phy.SetChannel (channel.Create ());
101
102 WifiHelper wifi;
103 wifi.SetRemoteStationManager ("ns3::AarfwifiManager");
104
105 WifiMacHelper mac;
106 Ssid ssid = Ssid ("ns-3-ssid");
107 mac.SetType ("ns3::StaWifiMac",
108             "Ssid", SsidValue (ssid),
109             "ActiveProbing", BooleanValue (false));
110
111 NetDeviceContainer staDevices;
112 staDevices = wifi.Install (phy, mac, wifiStaNodes);
113
114 mac.SetType ("ns3::ApWifiMac",
115             "Ssid", SsidValue (ssid));
116
117 NetDeviceContainer apDevices;
118 apDevices = wifi.Install (phy, mac, wifiApNode);
119
120 MobilityHelper mobility;
121
122 mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
123                               "MinX", DoubleValue (0.0),
124                               "MinY", DoubleValue (0.0),
125                               "DeltaX", DoubleValue (5.0),
126                               "DeltaY", DoubleValue (10.0),
127                               "GridWidth", UIntegerValue (3),
128                               "LayoutType", StringValue ("RowFirst"));
129
130 mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
131                             "Bounds", RectangleValue (Rectangle (-50, 50, -50, 50)));
132 mobility.Install (wifiStaNodes);
133
134 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
135 mobility.Install (wifiApNode);
136
137 InternetStackHelper stack;
138 stack.Install (csmaNodes);
139 stack.Install (wifiApNode);
140 stack.Install (wifiStaNodes);
141
142 Ipv4AddressHelper address;

```

```
143
144 address.SetBase ("10.1.1.0", "255.255.255.0");
145 Ipv4InterfaceContainer p2pInterfaces;
146 p2pInterfaces = address.Assign (p2pDevices);
147
148 address.SetBase ("10.1.2.0", "255.255.255.0");
149 Ipv4InterfaceContainer csmaInterfaces;
150 csmaInterfaces = address.Assign (csmaDevices);
151
152 address.SetBase ("10.1.3.0", "255.255.255.0");
153 address.Assign (staDevices);
154 address.Assign (apDevices);
155
156 UdpEchoServerHelper echoServer (9);
157
158 ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsmas));
159 serverApps.Start (Seconds (1.0));
160 serverApps.Stop (Seconds (10.0));
161
162 UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsmas), 9);
163 echoClient.SetAttribute ("MaxPackets", UIntegerValue (1));
164 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
165 echoClient.SetAttribute ("PacketSize", UIntegerValue (1024));
166
167 ApplicationContainer clientApps =
168     echoClient.Install (wifiStaNodes.Get (nWifi - 1));
169 clientApps.Start (Seconds (2.0));
170 clientApps.Stop (Seconds (10.0));
171
172 Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
173
174 Simulator::Stop (Seconds (10.0));
175
176 if (tracing)
177 {
178     phy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);
179     pointToPoint.EnablePcapAll ("third");
180     phy.EnablePcap ("third", apDevices.Get (0));
181     csma.EnablePcap ("third", csmaDevices.Get (0), true);
182 }
183
184 Simulator::Run ();
185 Simulator::Destroy ();
186 return 0;
187 }
```