**#include <iostream>**

**#include <fstream>**

**#include <string>**

**#include <cassert>**

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

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

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

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

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

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

**#include "ns3/ipv4-global-routing-helper.h"**

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

using namespace ns3;

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

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

{

CommandLine cmd;

cmd.Parse (argc, argv);

**Config::SetDefault ("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents", BooleanValue(true));**

/\* Configuration. \*/

/\* Build nodes. \*/

NodeContainer term\_0;

term\_0.Create (1);

NodeContainer term\_1;

term\_1.Create (1);

NodeContainer term\_2;

term\_2.Create (1);

NodeContainer term\_3;

term\_3.Create (1);

NodeContainer term\_4;

term\_4.Create (1);

NodeContainer term\_5;

term\_5.Create (1);

NodeContainer term\_6;

term\_6.Create (1);

/\* Build link. \*/

CsmaHelper csma\_hub\_0;

csma\_hub\_0.SetChannelAttribute ("DataRate", DataRateValue (**5000000**));

csma\_hub\_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (**2**)));

**PointToPointHelper p2p\_p2p\_0;**

**p2p\_p2p\_0.SetDeviceAttribute ("DataRate", DataRateValue (5000000));**

**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 (10000)));

PointToPointHelper p2p\_p2p\_2;

p2p\_p2p\_2.SetDeviceAttribute ("DataRate", DataRateValue (100000000));

p2p\_p2p\_2.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10000)));

PointToPointHelper p2p\_p2p\_3;

p2p\_p2p\_3.SetDeviceAttribute ("DataRate", DataRateValue (100000000));

p2p\_p2p\_3.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10000)));\*/

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

NodeContainer all\_hub\_0;

all\_hub\_0.Add (term\_2);

all\_hub\_0.Add (term\_3);

all\_hub\_0.Add (term\_4);

all\_hub\_0.Add (term\_5);

NetDeviceContainer ndc\_hub\_0 = csma\_hub\_0.Install (all\_hub\_0);

NodeContainer all\_p2p\_0;

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

all\_p2p\_0.Add (term\_2);

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

NodeContainer all\_p2p\_1;

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

all\_p2p\_1.Add (term\_2);

NetDeviceContainer ndc\_p2p\_1 = **p2p\_p2p\_0**.Install (all\_p2p\_1);

NodeContainer all\_p2p\_2;

all\_p2p\_2.Add (term\_1);

all\_p2p\_2.Add (term\_6);

NetDeviceContainer ndc\_p2p\_2 = **p2p\_p2p\_0**.Install (all\_p2p\_2);

NodeContainer all\_p2p\_3;

all\_p2p\_3.Add (term\_5);

all\_p2p\_3.Add (term\_6);

NetDeviceContainer ndc\_p2p\_3 = **p2p\_p2p\_0**.Install (all\_p2p\_3);

/\* Install the IP stack. \*/

InternetStackHelper internetStackH;

internetStackH.Install (term\_0);

internetStackH.Install (term\_1);

internetStackH.Install (term\_2);

internetStackH.Install (term\_3);

internetStackH.Install (term\_4);

internetStackH.Install (term\_5);

internetStackH.Install (term\_6);

/\* IP assign. \*/

Ipv4AddressHelper ipv4;

ipv4.SetBase (**"10.250.1.0", "255.255.255.0"**);

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

ipv4.SetBase **("10.0.3.0", "255.255.255.0");**

Ipv4InterfaceContainer iface\_ndc\_p2p\_2 = ipv4.Assign (ndc\_p2p\_2);

ipv4.SetBase **("172.16.1.0", "255.255.255.0");**

Ipv4InterfaceContainer iface\_ndc\_p2p\_3 = ipv4.Assign (ndc\_p2p\_3);

/\* Generate Route. \*/

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

/\* Generate Application. \*/

//NS\_LOG\_INFO ("Create Applications.");

**uint16\_t port = 9; // Discard port (RFC 863)**

**OnOffHelper onoff ("ns3::UdpSocketFactory",**

**InetSocketAddress (iface\_ndc\_p2p\_3.GetAddress (1), port));**

**onoff.SetAttribute ("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));**

**onoff.SetAttribute ("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));**

**onoff.SetAttribute ("DataRate", StringValue ("2kbps"));**

**onoff.SetAttribute ("PacketSize", UintegerValue (50));**

**ApplicationContainer apps = onoff.Install (term\_1.Get(0));**

**apps.Start (Seconds (1.0));**

**apps.Stop (Seconds (10.0));**

**OnOffHelper onoff2 ("ns3::UdpSocketFactory",InetSocketAddress (iface\_ndc\_p2p\_2.GetAddress (1),port));**

**onoff2.SetAttribute ("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));**

**onoff2.SetAttribute ("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));**

**onoff2.SetAttribute ("DataRate", StringValue ("2kbps"));**

**onoff2.SetAttribute ("PacketSize", UintegerValue (50));**

**ApplicationContainer apps2 = onoff2.Install (term\_1.Get(0));**

**apps2.Start (Seconds (11.0));**

**apps2.Stop (Seconds (16.0));**

**PacketSinkHelper sink ("ns3::UdpSocketFactory",Address (InetSocketAddress (Ipv4Address::GetAny(), port)));**

**apps = sink.Install (term\_6.Get(0));**

**apps.Start (Seconds (1.0));**

**apps.Stop (Seconds (10.0));**

**PacketSinkHelper sink2 ("ns3::UdpSocketFactory",**

**Address (InetSocketAddress (Ipv4Address::GetAny (), port)));**

**apps2 = sink2.Install (term\_6.Get(0));**

**apps2.Start (Seconds (11.0));**

**apps2.Stop (Seconds (16.0));**

**AsciiTraceHelper ascii;**

**Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("dynamic-global routing.tr");**

**p2p\_p2p\_0.EnableAsciiAll (stream);**

**csma\_hub\_0.EnableAsciiAll (stream);**

**internetStackH.EnableAsciiIpv4All (stream);**

**p2p\_p2p\_0.EnablePcapAll ("dynamic-global-routing");**

**csma\_hub\_0.EnablePcapAll ("dynamic-global-routing", false);**

**Ptr<Node> n1 = term\_1.Get(0);**

**Ptr<Ipv4> ipv41 = n1->GetObject<Ipv4> ();**

**uint32\_t ipv4ifIndex1 = 2;**

**Simulator::Schedule (Seconds (2),&Ipv4::SetDown,ipv41, ipv4ifIndex1);**

**Simulator::Schedule (Seconds (4),&Ipv4::SetUp,ipv41, ipv4ifIndex1);**

**Ptr<Node> n6 = term\_6.Get(0);**

**Ptr<Ipv4> ipv46 = n6->GetObject<Ipv4> ();**

**uint32\_t ipv4ifIndex6 = 2;**

**Simulator::Schedule (Seconds (6),&Ipv4::SetDown,ipv46, ipv4ifIndex6);**

**Simulator::Schedule (Seconds (8),&Ipv4::SetUp,ipv46, ipv4ifIndex6);**

**Simulator::Schedule (Seconds (12),&Ipv4::SetDown,ipv41, ipv4ifIndex1);**

**Simulator::Schedule (Seconds (14),&Ipv4::SetUp,ipv41, ipv4ifIndex1);**

**Ipv4GlobalRoutingHelper g;**

**Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("dynamic-global-routing.routes", std::ios::out);**

**g.PrintRoutingTableAllAt (Seconds (12), routingStream);**

**AnimationInterface anim("exp1.xml");**

**Ptr<Node> n=term\_0.Get(0);**

**anim.SetConstantPosition(term\_0.Get(0),0.0,75.0);**

**anim.SetConstantPosition(term\_1.Get(0),25.0,50.0);**

**anim.SetConstantPosition(term\_2.Get(0),0.0,25.0);**

**anim.SetConstantPosition(term\_3.Get(0),80.0,50.0);**

**anim.SetConstantPosition(term\_4.Get(0),50.0,50.0);**

**anim.SetConstantPosition(term\_5.Get(0),75.5,25.0);**

**anim.SetConstantPosition(term\_6.Get(0),100.0,25.0);**

NS\_LOG\_INFO ("Run Simulation.");

Simulator::Run ();

Simulator::Destroy ();

NS\_LOG\_INFO ("Done.");

}