Star Topology

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"

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


// Network topology (default)
//
//        n2 n3 n4              .
//         \ | /                .
//          \|/                 .
//     n1--- n0---n5            .
//          /|\                 .
//         / | \                .
//        n8 n7 n6              .
//


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Star");

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

NodeContainer nodes;
```

```
nodes.Create(9);

  //
  // Set up some default values for the simulation.
  //
  Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (137));

  // ??? try and stick 15kb/s into the data rate
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("14kb/s"));

  //
  // Default number of nodes in the star.  Overridable by command line argument.
  //
  uint32_t nSpokes = 8;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("nSpokes", "Number of nodes to place in the star", nSpokes);
  cmd.Parse (argc, argv);

  NS_LOG_INFO ("Build star topology.");
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);

  NS_LOG_INFO ("Install internet stack on all nodes.");
  InternetStackHelper internet;
  star.InstallStack (internet);

  NS_LOG_INFO ("Assign IP Addresses.");
  star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));

  NS_LOG_INFO ("Create applications.");
  //
  // Create a packet sink on the star "hub" to receive packets.
  //
  uint16_t port = 50000;
  Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress);
  ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
  hubApp.Start (Seconds (1.0));
  hubApp.Stop (Seconds (10.0));

  //
  // Create OnOff applications to send TCP to the hub, one on each spoke node.
  //
  OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
  onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
```

```
  ApplicationContainer spokeApps;

  for (uint32_t i = 0; i < star.SpokeCount (); ++i)
    {
      AddressValue remoteAddress (InetSocketAddress (star.GetHubIpv4Address (i), port));
      onOffHelper.SetAttribute ("Remote", remoteAddress);
      spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
    }
  spokeApps.Start (Seconds (1.0));
  spokeApps.Stop (Seconds (10.0));

  NS_LOG_INFO ("Enable static global routing.");
  //
  // Turn on global static routing so we can actually be routed across the star.
  //
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  NS_LOG_INFO ("Enable pcap tracing.");
  //
  // Do pcap tracing on all point-to-point devices on all nodes.
  //

MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);
AnimationInterface anim("star.xml");
AnimationInterface::SetConstantPosition(nodes.Get(0),10,2);
AnimationInterface::SetConstantPosition(nodes.Get(1),11,5);
AnimationInterface::SetConstantPosition(nodes.Get(2),15,2);
AnimationInterface::SetConstantPosition(nodes.Get(3),19,7);
anim.EnablePacketMetadata(true);
pointToPoint.EnablePcapAll("star");

  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

  return 0;
}
```