EXPERIMENT 5

BUILDING TOPOLOGIES IN NS-3 (STAR TOPOLOGY)

OBJECTIVE:

- Implement the star network topology

BACKGROUND:

You have implemented point to point, CSMA and Wi-Fi topologies. In today's lab we are going to strengthen those skills with implementation of the Star topology shown in figure 5.1:

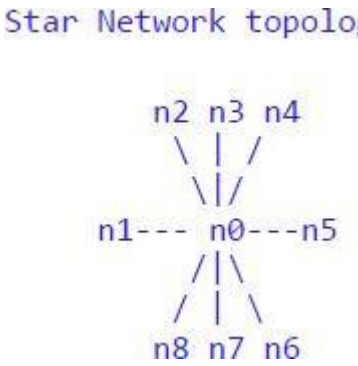


Figure 5.1: Star Network

Each link is a point to point link. The central node (n0) is the packet sink. The other nodes generate traffic, which is sent to the sink. The code for this topology is in the examples folder (examples/tcp/star.cc). In this particular example, pcap tracing has been enabled, while logging is not enabled. Enable logging by inserting the following lines of code in the beginning of main:

```
LogComponentEnable ("Star", LOG_LEVEL_INFO);
LogComponentEnable ("PacketSink", LOG_LEVEL_INFO);
LogComponentEnable ("OnOffApplication", LOG_LEVEL_INFO);
  NS_LOG_INFO ("Star Topology Simulation");
```

Now running the code will show transmission and reception of packets in the terminal window.

TASK:

Modify the above topology as shown below in figure 5.2:

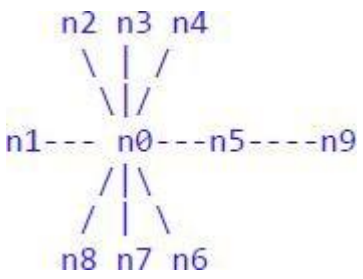


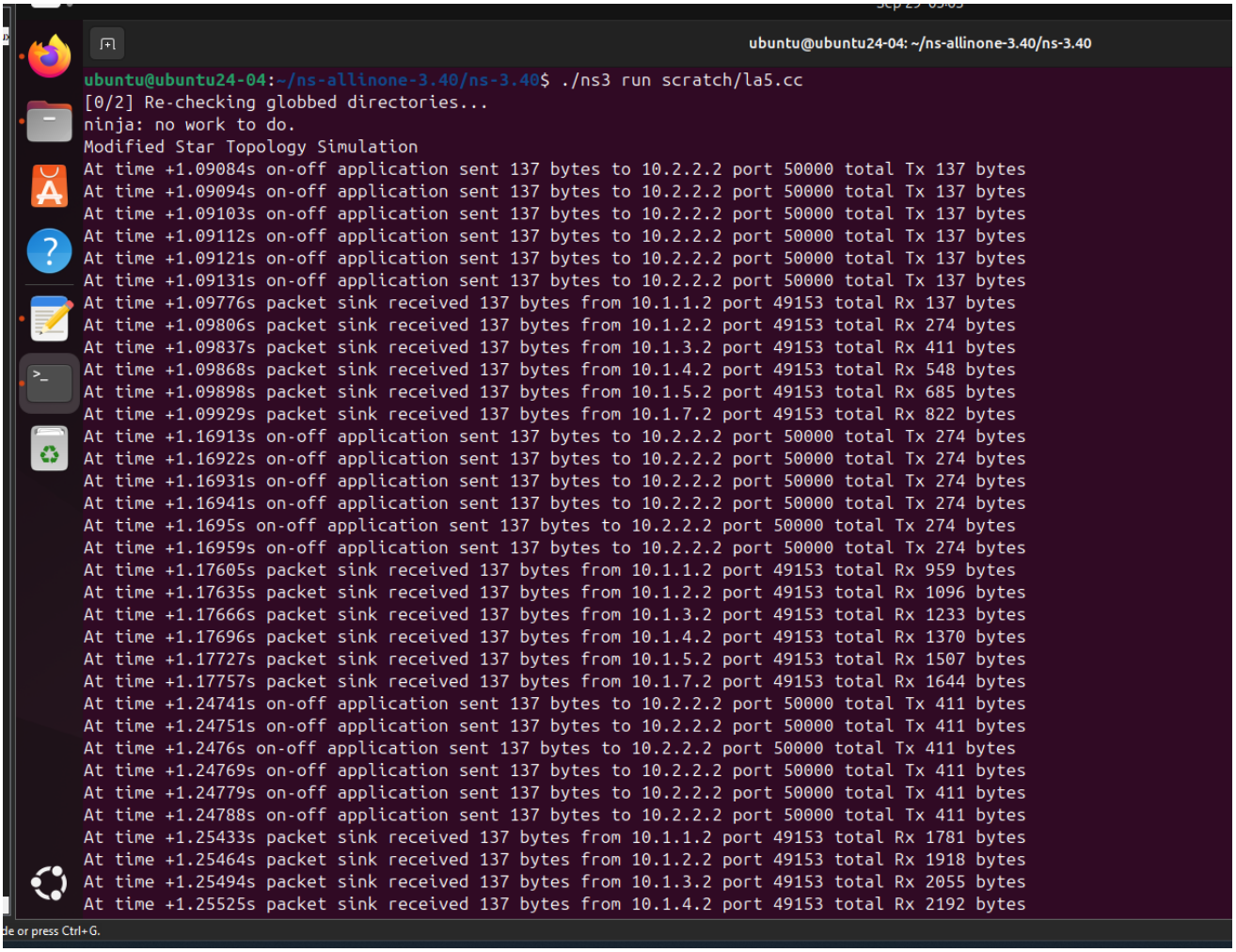
Figure 5.2: Modified Star Network

Do not install any applications on nodes 0 and 5. Make node 9 (n9) the packet sink and send all packets (from n1-n4 and n6-n8) to this sink. All links are point to point links.

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Star");

int main(int argc, char* argv[])
{
    LogComponentEnable("Star", LOG_LEVEL_INFO);
    LogComponentEnable("PacketSink", LOG_LEVEL_INFO);
    LogComponentEnable("OnOffApplication", LOG_LEVEL_INFO);
    NS_LOG_INFO("Modified Star Topology Simulation");

    Config::SetDefault("ns3::OnOffApplication::PacketSize", UintegerValue(137));
    Config::SetDefault("ns3::OnOffApplication::DataRate", StringValue("14kb/s"));


    uint32_t nSpokes = 7;

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

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    // Build star
    PointToPointStarHelper star(nSpokes, pointToPoint);

    // Create extra node n9 and connect to n5

    NodeContainer extra;
    extra.Create(2); // n5 and n9

    InternetStackHelper internet;
    star.InstallStack(internet);
    internet.Install(extra);

    NetDeviceContainer d0n5 = pointToPoint.Install(star.GetHub(), extra.Get(0));

    NetDeviceContainer d5n9 = pointToPoint.Install(extra.Get(0), extra.Get(1));

     // Assign addresses
    star.AssignIpv4Addresses(Ipv4AddressHelper("10.1.1.0", "255.255.255.0"));

    Ipv4AddressHelper address1,address2;
    address1.SetBase("10.2.1.0", "255.255.255.0");
    Ipv4InterfaceContainer if0n5 = address1.Assign(d0n5);

    address2.SetBase("10.2.2.0", "255.255.255.0");
    Ipv4InterfaceContainer if5n9 = address2.Assign(d5n9);
```

```
    // Install packet sink on n9
    uint16_t port = 50000;
    Address sinkLocalAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
    PacketSinkHelper sinkHelper("ns3::TcpSocketFactory", sinkLocalAddress);
    ApplicationContainer sinkApp = sinkHelper.Install(extra.Get(1));
    sinkApp.Start(Seconds(1.0));
    sinkApp.Stop(Seconds(10.0));


    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)
    {
        if (i == 5) continue; // skip n5 (forwarder only)

        AddressValue remoteAddress(InetSocketAddress(if5n9.GetAddress(1), port)); // sink at n9
        onOffHelper.SetAttribute("Remote", remoteAddress);
        spokeApps.Add(onOffHelper.Install(star.GetSpokeNode(i)));
    }
    spokeApps.Start(Seconds(1.0));
    spokeApps.Stop(Seconds(10.0));



    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    pointToPoint.EnablePcapAll("star_forwarding");

    Simulator::Run();
    Simulator::Destroy();
    return 0;

}
```

Output:



POST LAB QUESTION:

Modify the star topology as below in the figure 5.3 and paste your code in the space given below.

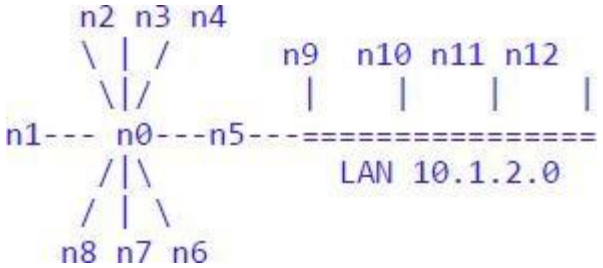


Figure 5.3

Do not install any application layer on n0 and n5. Make n12 packet sink and send all packets from n1n4 and n6-n11 to this sink.

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Star");

int main(int argc, char* argv[])
{
    LogComponentEnable("Star", LOG_LEVEL_INFO);
    LogComponentEnable("PacketSink", LOG_LEVEL_INFO);
    LogComponentEnable("OnOffApplication", LOG_LEVEL_INFO);
    NS_LOG_INFO("Star Topology Simulation");

    Config::SetDefault("ns3::OnOffApplication::PacketSize", UintegerValue(137));
    Config::SetDefault("ns3::OnOffApplication::DataRate", StringValue("14kb/s"));

    // Star has 8 spokes (n1–n8 around hub n0)

    uint32_t nSpokes = 8;

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


    // Point-to-Point helper

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    // Build star
    PointToPointStarHelper star(nSpokes, pointToPoint);

    // Create LAN nodes: n9–n12

    NodeContainer csmaNodes;
    csmaNodes.Create(4);


    // Connect n5 (spoke index 4) to LAN gateway (csmaNodes.Get(0)) via P2P
    NodeContainer p2pLan;

    p2pLan.Add(star.GetSpokeNode(4),csmaNodes.Get(0));   // n5 +  LAN gateway

    NetDeviceContainer p2pLanDevices = pointToPoint.Install(p2pLan);


    // CSMA LAN: n9–n12

    CsmaHelper csma;
    csma.SetChannelAttribute("DataRate", StringValue("5Mbps"));
    csma.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer csmaDevices = csma.Install(csmaNodes);

    // Install Internet stacks
    InternetStackHelper internet;
    star.InstallStack(internet);
    internet.Install(csmaNodes);
```

```
    // Assign IPs
    // ->Subnet 1: star (n0 + n1–n7)

    star.AssignIpv4Addresses(Ipv4AddressHelper("10.1.1.0", "255.255.255.0"));

    // ->Subnet 2: P2P link (n5 ↔ LAN gateway)

    Ipv4AddressHelper addressP2P;
    addressP2P.SetBase("10.2.1.0", "255.255.255.0");

    Ipv4InterfaceContainer ifP2P = addressP2P.Assign(p2pLanDevices);

    // ->Subnet 3: CSMA LAN (n9–n12)

    Ipv4AddressHelper addressLan;
    addressLan.SetBase("10.2.2.0", "255.255.255.0");

    Ipv4InterfaceContainer ifLan = addressLan.Assign(csmaDevices);



    uint16_t port = 50000;
    Address sinkLocalAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
    PacketSinkHelper sinkHelper("ns3::TcpSocketFactory", sinkLocalAddress);

    ApplicationContainer sinkApp = sinkHelper.Install(csmaNodes.Get(3)); // sink at n12

    sinkApp.Start(Seconds(1.0));
    sinkApp.Stop(Seconds(10.0));

    OnOffHelper onOffHelper("ns3::TcpSocketFactory", Address());
    onOffHelper.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));
    onOffHelper.SetAttribute("OffTime",    StringValue("ns3::ConstantRandomVariable[Constant=0]"));

    ApplicationContainer sourceApps;


     // Install on star spokes except hub (n0) and forwarder (n5)

for (uint32_t i = 0; i < star.SpokeCount(); ++i)
    {
        if (i == 4) continue; // skip n5

        AddressValue remoteAddress(InetSocketAddress(ifLan.GetAddress(3), port)); // sink at n12
        onOffHelper.SetAttribute("Remote", remoteAddress);
        sourceApps.Add(onOffHelper.Install(star.GetSpokeNode(i)));
    }


    // Install on LAN nodes except sink (n12)

for (uint32_t j = 0; j < 3; ++j) // n9–n11
    {
        AddressValue remoteAddress(InetSocketAddress(ifLan.GetAddress(3), port)); // sink at n12
        onOffHelper.SetAttribute("Remote", remoteAddress);
        sourceApps.Add(onOffHelper.Install(csmaNodes.Get(j)));
    }

    sourceApps.Start(Seconds(1.0));
    sourceApps.Stop(Seconds(10.0));



    Ipv4GlobalRoutingHelper::PopulateRoutingTables();

    // Tracing

    pointToPoint.EnablePcapAll("star_forwarding");
    csma.EnablePcap("lan", csmaDevices.Get(0), true);
```

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