

Networking with Linux Lab

Module 2 & 3: Client Server Network topology using NS-3 and Animating the Network

Assignment 5: Write an NS3 program to simulate a star topology.

1. Execute the program with 10 nodes using command line arguments and display the results using NetAnim.

Aim: To execute a program with 10 nodes using command line arguments and visualize the results using NetAnim for network simulation and analysis

Theory: Star Topology

A star topology is a network architecture where each device (such as computers, printers, or servers) is connected directly to a central hub or switch. All data in a star network flows through this central point, which manages and controls the communication between devices.

This arrangement offers simplicity in setup and maintenance, easy troubleshooting, and efficient data transmission. However, if the central hub fails, the entire network may become inaccessible.

Command-Line Arguments

Command-line arguments are parameters supplied to a program when it is executed from the command line or terminal. They allow users to customize the behavior of a program without modifying its source code.

PyGraphviz is used to visualize network topologies. Network topology refers to the arrangement of nodes and connections in a network. Graph theory helps in modeling and understanding these topologies.

NetAnim is a network animation tool used with ns-3 for simulating and visualizing network behavior. It employs Discrete Event Simulation (DES) to model events like packet transmissions and node movements at discrete time points. The tool focuses on packet-level visualization, providing insights into data flow and network issues. Its real-time analysis feature aids in debugging and optimizing network protocols by offering a dynamic visual representation of the simulation.

Code:**> star.cc**

```

#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"

// 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[])
{
    //
    // 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.
//
pointToPoint.EnablePcapAll ("star");

// Set the bounding box for animation
star.BoundingBox (1, 1, 100, 100);

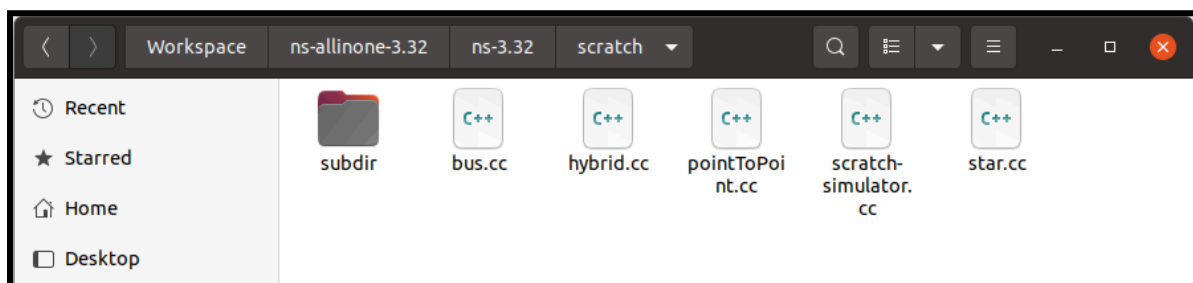
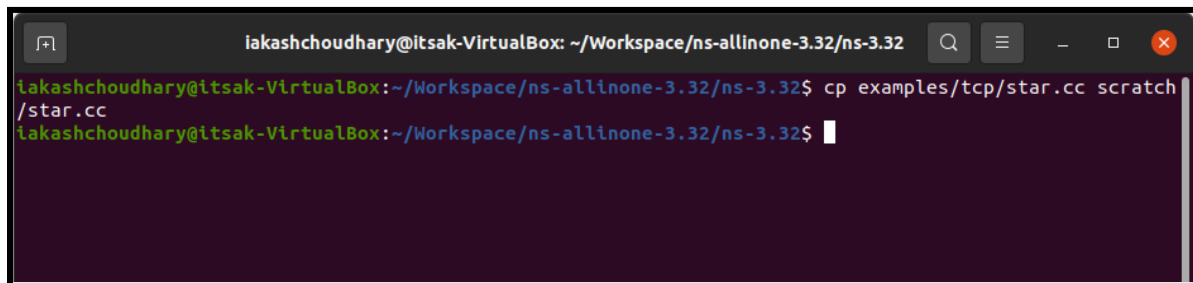
// Create the animation object and configure for specified xml output file
AnimationInterface anim ("star.xml");

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

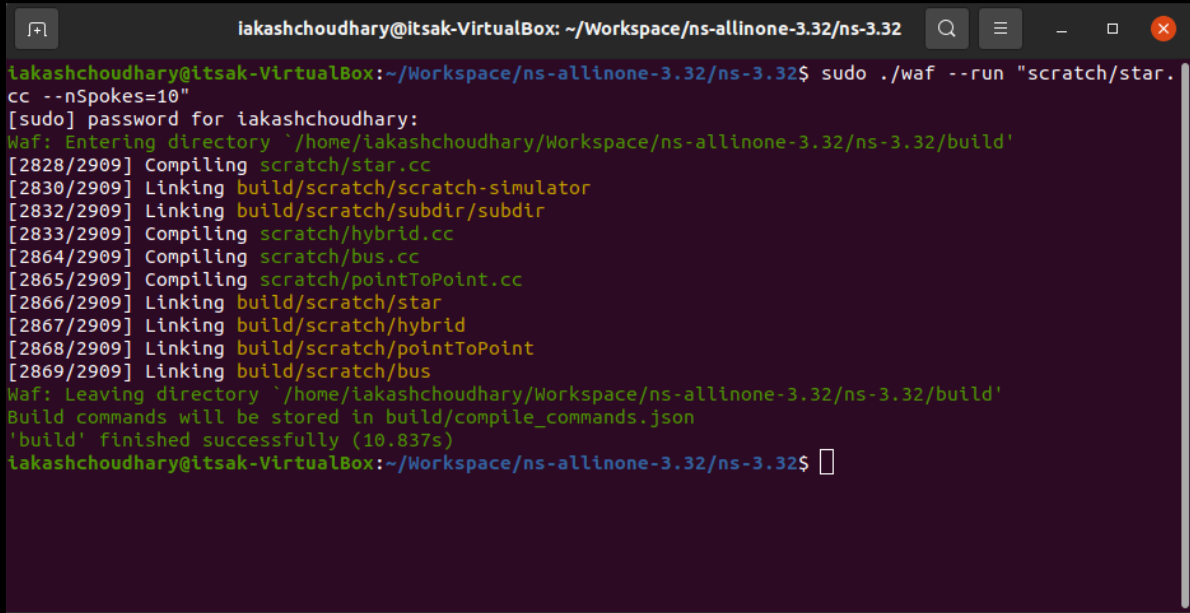
return 0;
}
```

Command & Screenshot:

> \$ cp examples/tcp/star.cc scratch/star.cc

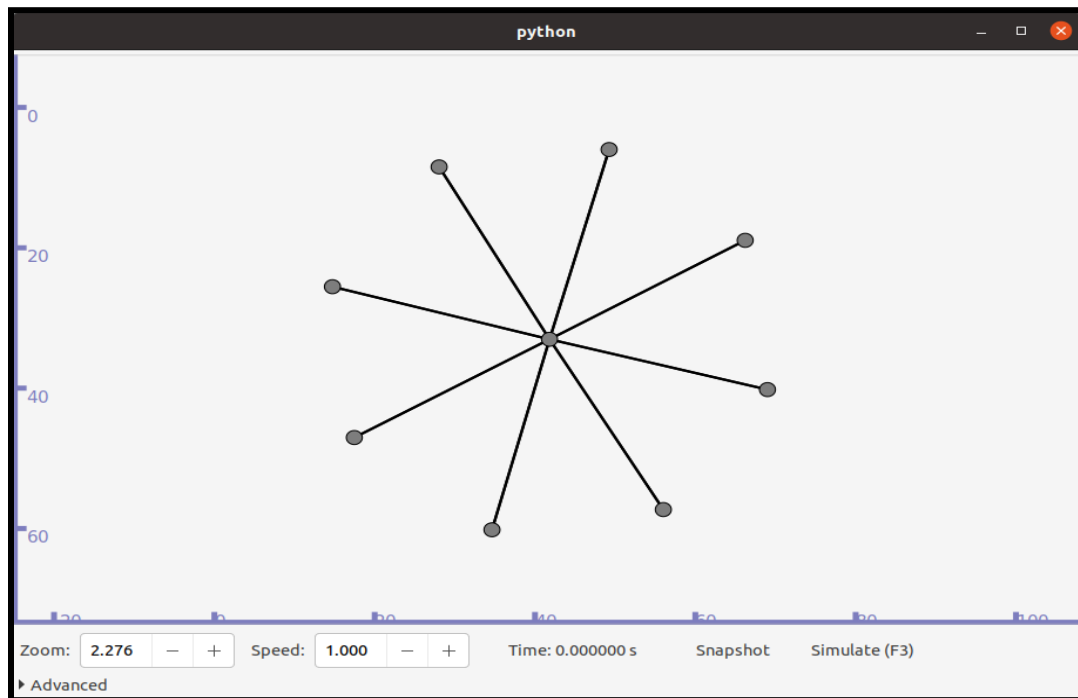


> \$ sudo ./waf --run "scratch/star.cc --nSpokes=10"

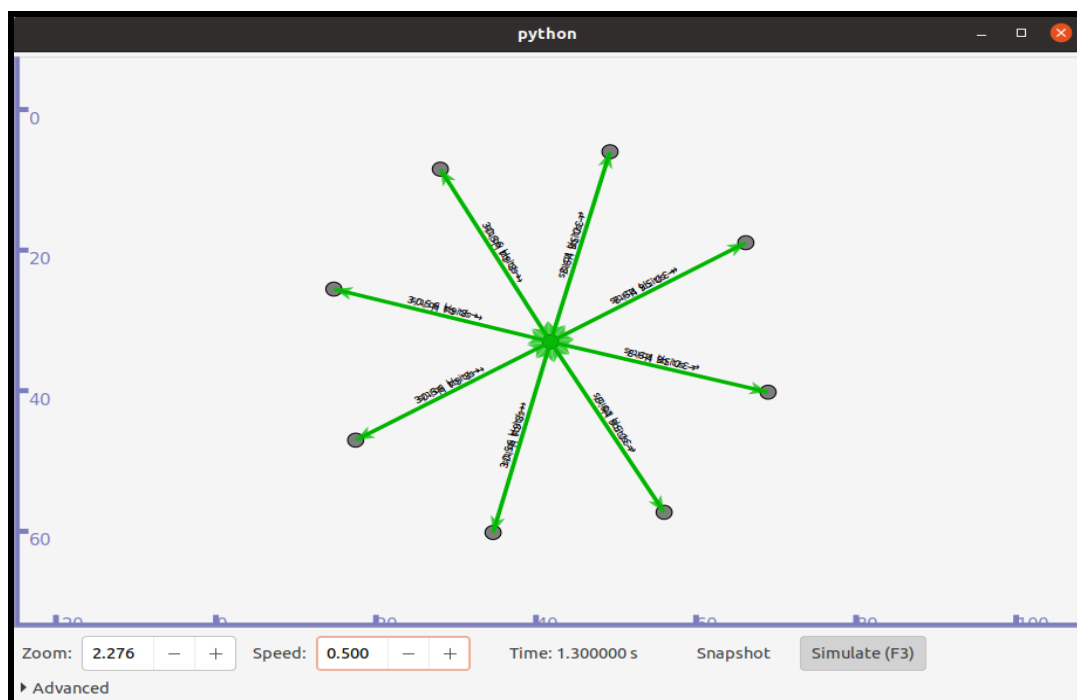
A terminal window titled 'iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32'. The user enters the command 'sudo ./waf --run "scratch/star.cc --nSpokes=10"'. The terminal shows the following output:

```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scratch/star.cc --nSpokes=10"
[sudo] password for iakashchoudhary:
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
[2828/2909] Compiling scratch/star.cc
[2830/2909] Linking build/scratch/scratch-simulator
[2832/2909] Linking build/scratch/subdir/subdir
[2833/2909] Compiling scratch/hybrid.cc
[2864/2909] Compiling scratch/bus.cc
[2865/2909] Compiling scratch/pointToPoint.cc
[2866/2909] Linking build/scratch/star
[2867/2909] Linking build/scratch/hybrid
[2868/2909] Linking build/scratch/pointToPoint
[2869/2909] Linking build/scratch/bus
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (10.837s)
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

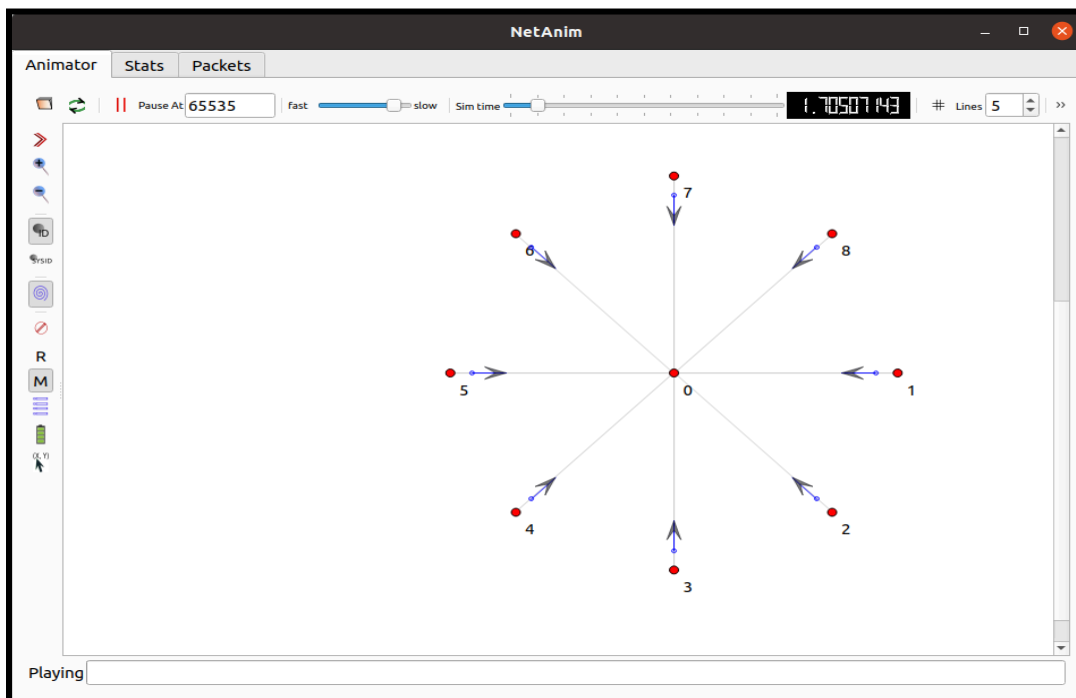
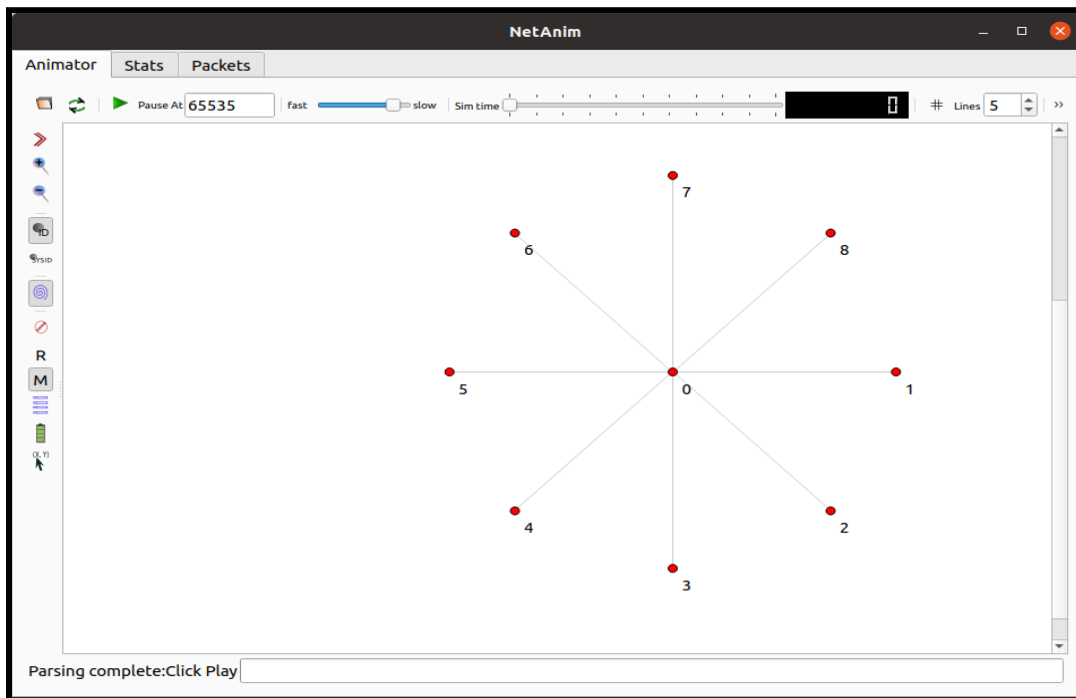
> Perform animation using pygraphviz: `$ sudo ./waf --run "scratch/star.cc" --vis`



```
iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scratch/star.
cc" --vis
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.921s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 9 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```



> Perform animation using NetAnim: \$./NetAnim | Open star.xml



Conclusion: The program executed successfully with 10 nodes and the results were visualized using NetAnim.

2. Save the log information in a file (command: `./waf --run scratch/star.cc > log.out 2>&1`).

Aim: To run the "star.cc" script using the "./waf" command, redirecting both standard output and standard error streams to a file named "log.out" for logging purposes

Theory:

The command `./waf --run scratch/star.cc > log.out 2>&1` is used to execute a program named "waf" with the argument "--run scratch/star.cc". This command redirects both standard output (stdout) and standard error (stderr) to a file named "log.out".

In short, this command runs a program, captures any output or errors generated during its execution, and saves them in a file called "log.out" for later reference or analysis.

Code:

> **star.cc**

```
#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"
```

```
// 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[])
{
```

```
//
// 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.
//
pointToPoint.EnablePcapAll ("star");

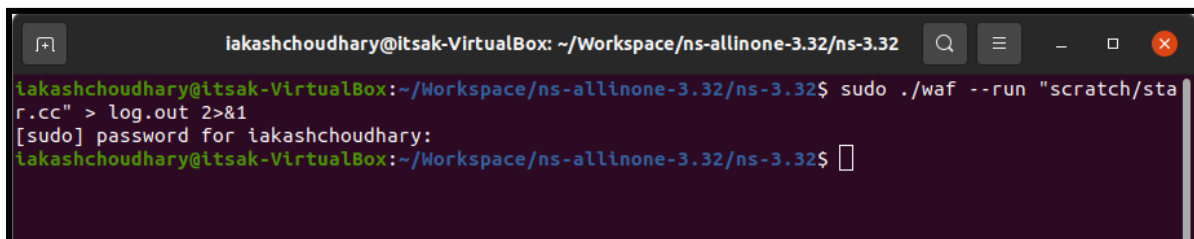
NS_LOG_INFO ("Run Simulation.");
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");

return 0;
}

```

Command & Screenshot:

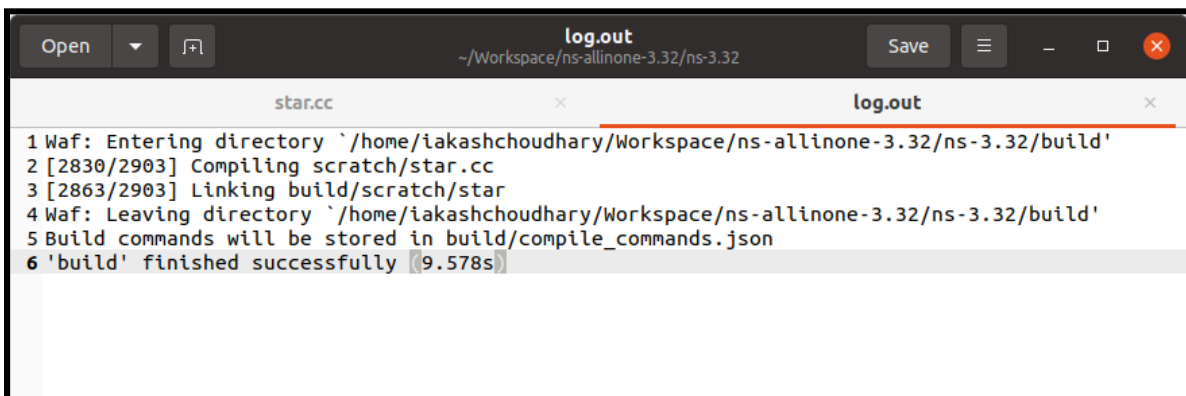
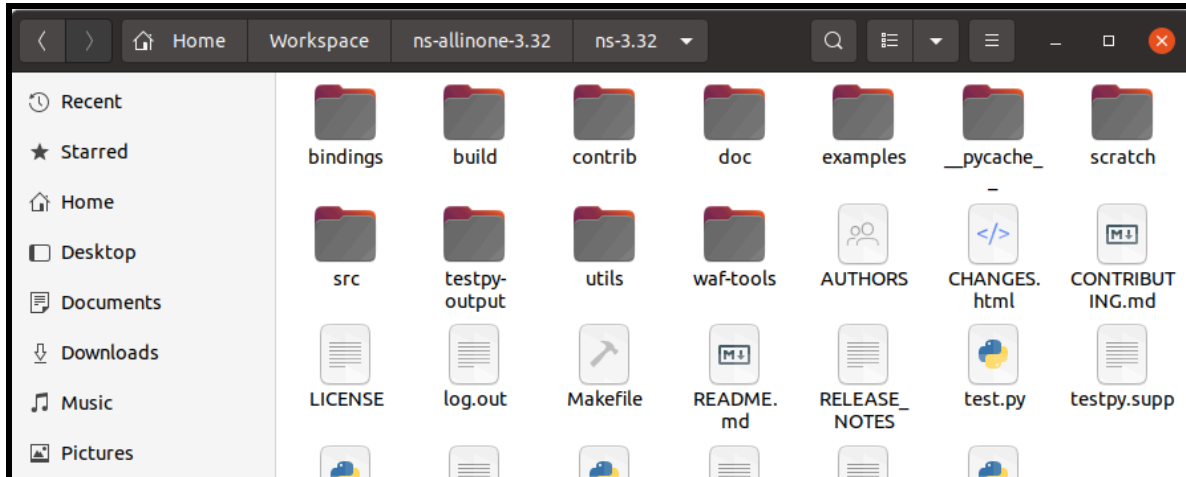
> \$ sudo ./waf --run "scratch/star.cc" > log.out 2>&1



```

lakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32
lakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scratch/sta
r.cc" > log.out 2>&1
[sudo] password for lakashchoudhary:
lakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$

```



Conclusion: The log information from running the `./waf` command with the `scratch/star.cc` script has been saved to the file `log.out`, indicating successful execution and capturing any errors or output messages.

3. Generate a .pcap file for analysis using Wireshark.

Aim: To analyze network traffic and investigate potential security threats or performance issues using Wireshark by examining the generated .pcap file

Theory: Wireshark

Wireshark is a popular open-source network protocol analyzer that allows users to capture and inspect data traveling back and forth on a network in real-time. It is commonly used for network troubleshooting, analysis, software and protocol development, and education. Wireshark supports a wide range of protocols and provides a comprehensive set of features for capturing, analyzing, and displaying network traffic.

Code:

> **star.cc**

```
#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"
```

```
// 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[])
{
    //
```

```

// 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.
//
pointToPoint.EnablePcapAll ("star");

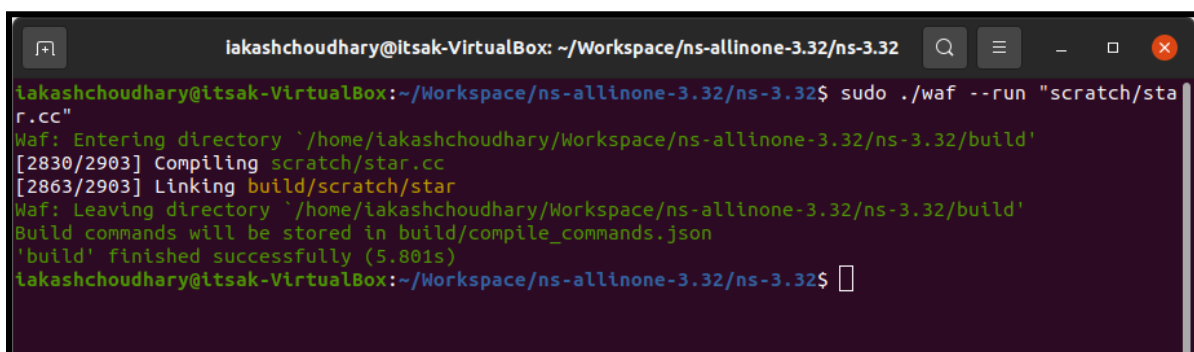
NS_LOG_INFO ("Run Simulation.");
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");

return 0;
}

```

Command & Screenshot:

> \$ sudo ./waf --run "scratch/star.cc"

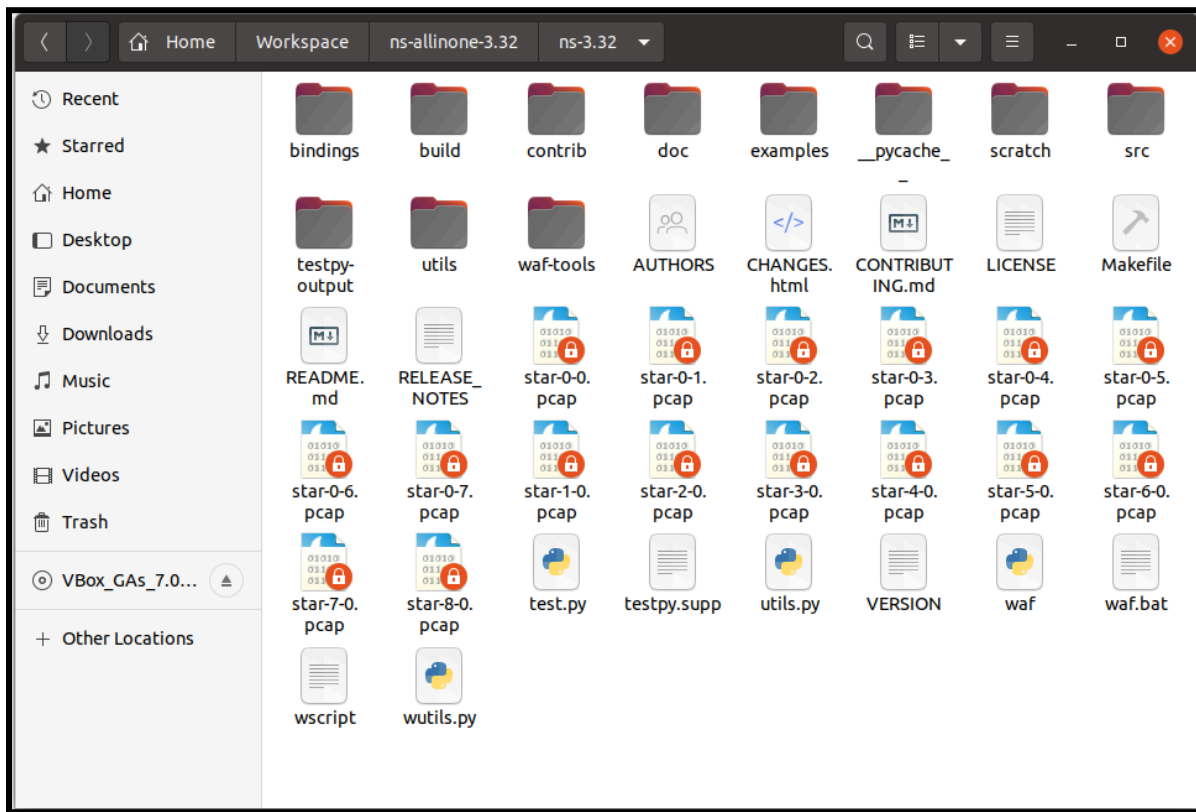


```

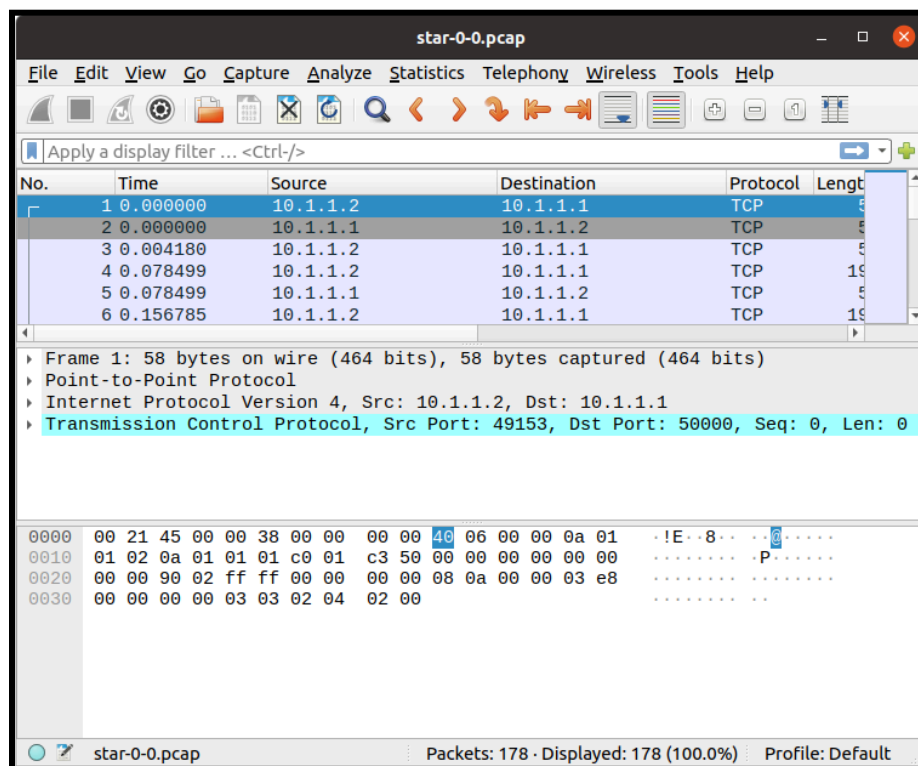
iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scratch/star.cc"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
[2830/2903] Compiling scratch/star.cc
[2863/2903] Linking build/scratch/star
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (5.801s)
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$

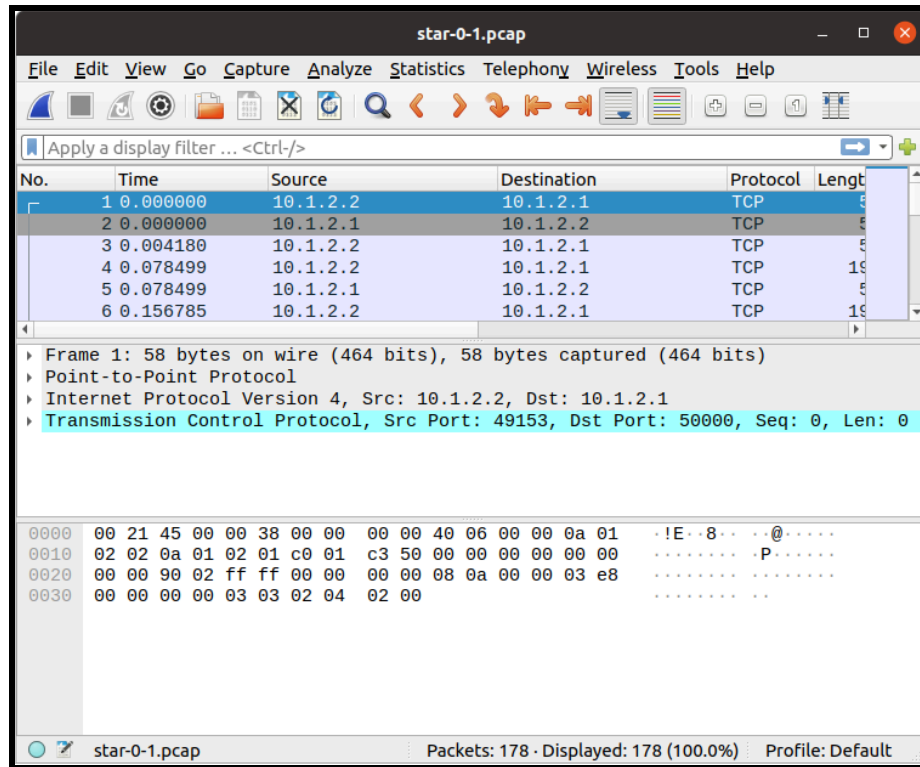
```

> After executing the above command, a **.pcap file is created** for Wireshark analysis.



> To open each .pcap file, simply **double-click** on it.





Conclusion: The .pcap file exhibits network traffic patterns and can be analyzed using Wireshark for further insights into network activity.