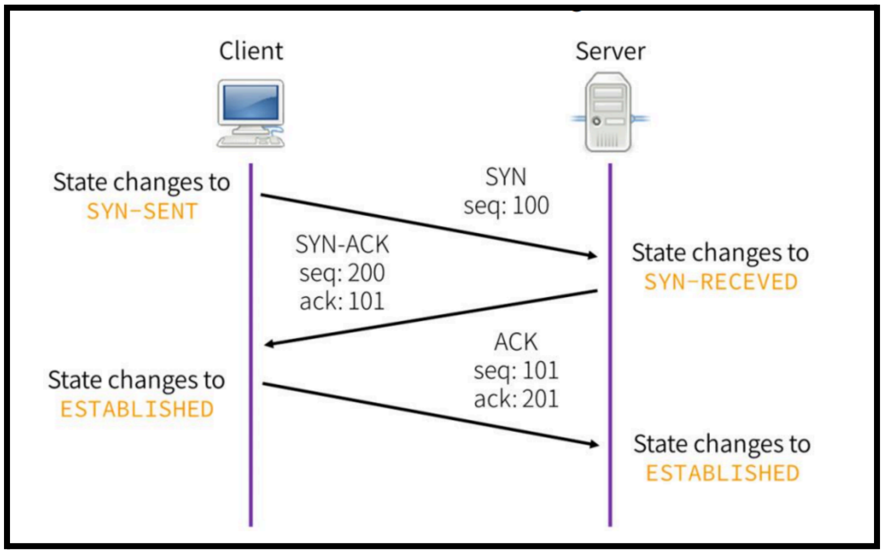
Networking with Linux Lab
Module 2 & 3: Client Server Network topology using NS-3 and Animating the Network

Assignment 10: Animate Three way handshake for TCP connection using NetAnim

Aim: To visually simulate and understand the three-way handshake process for establishing a TCP connection using NetAnim

Theory: Three-Way Handshake

TCP, or Transmission Control Protocol, ensures secure data transmission. Positive Acknowledgement with Retransmission (PAR) guarantees stable communication. In PAR, until receiving acknowledgment, a device keeps transmitting data units. The receiver discards damaged segments using checksum for error detection. If a segment is discarded, the sender resends it.


Client
Server
State changes to SYN-SENT
SYN seq: 100
SYN-ACK seq: 200 ack: 101
State changes to SYN-RECEVED
ACK seq: 101 ack: 201
State changes to ESTABLISHED
State changes to ESTABLISHED

The three-way handshake involves:

1. Client sends **SYN** to start communication and indicates the sequence number.
2. Server responds with **SYN-ACK**, acknowledging and indicating its sequence number.
3. Client confirms with **ACK**, establishing a secure connection for data transfer.

Code:

> tcp-star-server.cc

```
// Default Network topology, 9 nodes in a star
/*
        n2 n3 n4
        \ | /
        \|/
        n1---n0---n5
        /| \
        / | \
        n8 n7 n6
*/
// - CBR Traffic goes from the star "arms" to the "hub"
// - Tracing of queues and packet receptions to file
//   "tcp-star-server.tr"
// - pcap traces also generated in the following files
//   "tcp-star-server-$n-$i.pcap" where n and i represent node and interface
//   numbers respectively
// Usage examples for things you might want to tweak:
//       ./waf --run="tcp-star-server"
//       ./waf --run="tcp-star-server --nNodes=25"
//       ./waf --run="tcp-star-server --ns3::OnOffApplication::DataRate=10000"
//       ./waf --run="tcp-star-server --ns3::OnOffApplication::PacketSize=500"
// See the ns-3 tutorial for more info on the command line:
// http://www.nsnam.org/tutorials.html

#include <iostream>
#include <fstream>
#include <string>
#include <cassert>

#include "ns3/core-module.h"
#include "ns3/network-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"
#include "ns3/mobility-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpServer");

int
```

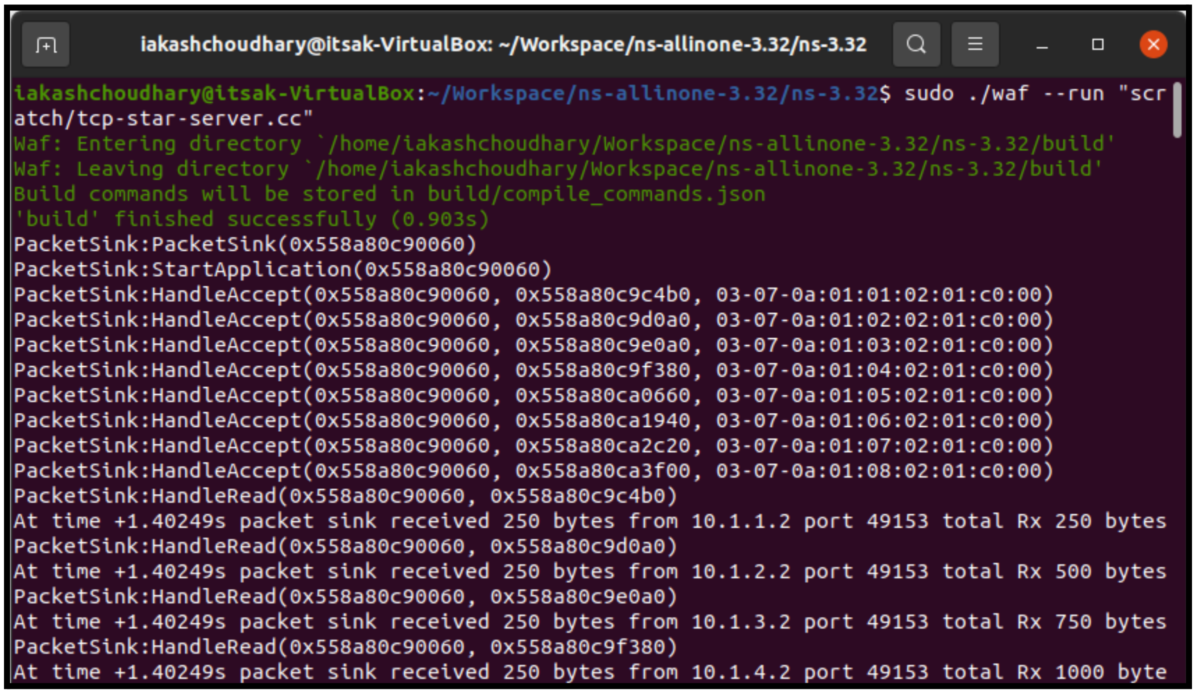
```
main (int argc, char *argv[])
{
  // Users may find it convenient to turn on explicit debugging
  // for selected modules; the below lines suggest how to do this

  //LogComponentEnable ("TcpServer", LOG_LEVEL_INFO);
  //LogComponentEnable ("TcpL4Protocol", LOG_LEVEL_ALL);
  //LogComponentEnable ("TcpSocketImpl", LOG_LEVEL_ALL);
  LogComponentEnable ("PacketSink", LOG_LEVEL_ALL);

  // Set up some default values for the simulation.
  Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (250));
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("5kb/s"));
  uint32_t N = 9; //number of nodes in the star

  // Allow the user to override any of the defaults and the above
  // Config::SetDefault()s at run-time, via command-line arguments
  CommandLine cmd (__FILE__);
  cmd.AddValue ("nNodes", "Number of nodes to place in the star", N);
  cmd.Parse (argc, argv);

  // Here, we will create N nodes in a star.
  NS_LOG_INFO ("Create nodes.");
  NodeContainer serverNode;
  NodeContainer clientNodes;
  serverNode.Create (1);
  clientNodes.Create (N-1);
  NodeContainer allNodes = NodeContainer (serverNode, clientNodes);

  // Install network stacks on the nodes
  InternetStackHelper internet;
  internet.Install (allNodes);

  //Collect an adjacency list of nodes for the p2p topology
  std::vector<NodeContainer> nodeAdjacencyList (N-1);
  for(uint32_t i=0; i<nodeAdjacencyList.size (); ++i)
        {
        nodeAdjacencyList[i] = NodeContainer (serverNode, clientNodes.Get (i));
        }

  // We create the channels first without any IP addressing information
  NS_LOG_INFO ("Create channels.");
  PointToPointHelper p2p;
  p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
  std::vector<NetDeviceContainer> deviceAdjacencyList (N-1);
```

```
for(uint32_t i=0; i<deviceAdjacencyList.size (); ++i)
        {
        deviceAdjacencyList[i] = p2p.Install (nodeAdjacencyList[i]);
        }

// Later, we add IP addresses.
NS_LOG_INFO ("Assign IP Addresses.");
Ipv4AddressHelper ipv4;
std::vector<Ipv4InterfaceContainer> interfaceAdjacencyList (N-1);
for(uint32_t i=0; i<interfaceAdjacencyList.size (); ++i)
        {
        std::ostringstream subnet;
        subnet<<"10.1."<<i+1<<".0";
        ipv4.SetBase (subnet.str ().c_str (), "255.255.255.0");
        interfaceAdjacencyList[i] = ipv4.Assign (deviceAdjacencyList[i]);
        }

//Turn on global static routing
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

// Create a packet sink on the star "hub" to receive these packets
uint16_t port = 50000;
Address sinkLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", sinkLocalAddress);
ApplicationContainer sinkApp = sinkHelper.Install (serverNode);
sinkApp.Start (Seconds (1.0));
sinkApp.Stop (Seconds (10.0));

// Create the OnOff applications to send TCP to the server
OnOffHelper clientHelper ("ns3::TcpSocketFactory", Address ());
                    clientHelper.SetAttribute          ("OnTime",          StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
                    clientHelper.SetAttribute          ("OffTime",          StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

//normally wouldn't need a loop here but the server IP address is different
//on each p2p subnet
ApplicationContainer clientApps;
for(uint32_t i=0; i<clientNodes.GetN (); ++i)
        {
        AddressValue remoteAddress
        (InetSocketAddress (interfaceAdjacencyList[i].GetAddress (0), port));
        clientHelper.SetAttribute ("Remote", remoteAddress);
        clientApps.Add (clientHelper.Install (clientNodes.Get (i)));
        }
clientApps.Start (Seconds (1.0));
```

```
  clientApps.Stop (Seconds (10.0));


  //configure tracing
  AsciiTraceHelper ascii;
  p2p.EnableAsciiAll (ascii.CreateFileStream ("tcp-star-server.tr"));
  p2p.EnablePcapAll ("tcp-star-server");

  // NetAnimation ---before simulator run
  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(serverNode);
  mobility.Install(clientNodes);

  AnimationInterface anim("tcp-star-server.xml");
  AnimationInterface::SetConstantPosition(serverNode.Get(0), 5, 5);
  AnimationInterface::SetConstantPosition(clientNodes.Get(0), 0, 5);
  AnimationInterface::SetConstantPosition(clientNodes.Get(1), 0, 0);
  AnimationInterface::SetConstantPosition(clientNodes.Get(2), 5, 0);
  AnimationInterface::SetConstantPosition(clientNodes.Get(3), 10, 0);
  AnimationInterface::SetConstantPosition(clientNodes.Get(4), 10, 5);
  AnimationInterface::SetConstantPosition(clientNodes.Get(5), 10, 10);
  AnimationInterface::SetConstantPosition(clientNodes.Get(6), 5, 10);
  AnimationInterface::SetConstantPosition(clientNodes.Get(7), 0, 10);
  anim.EnablePacketMetadata(true);

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

  return 0;
}
```

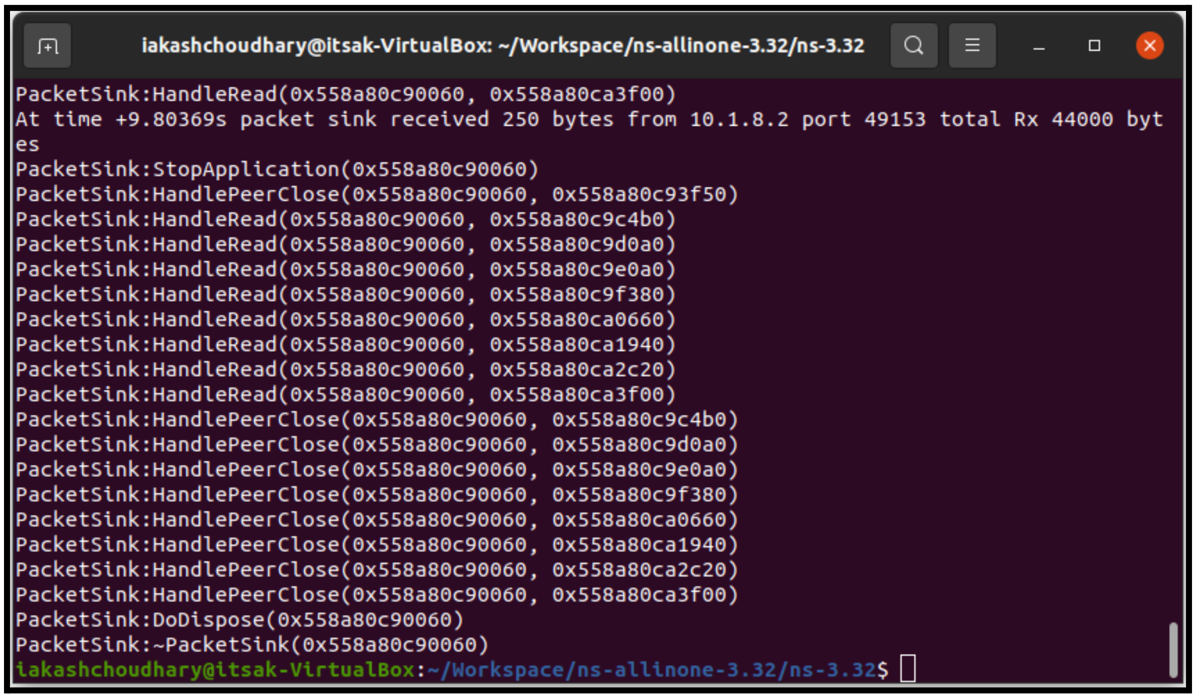
Command & Screenshot:

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



iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32

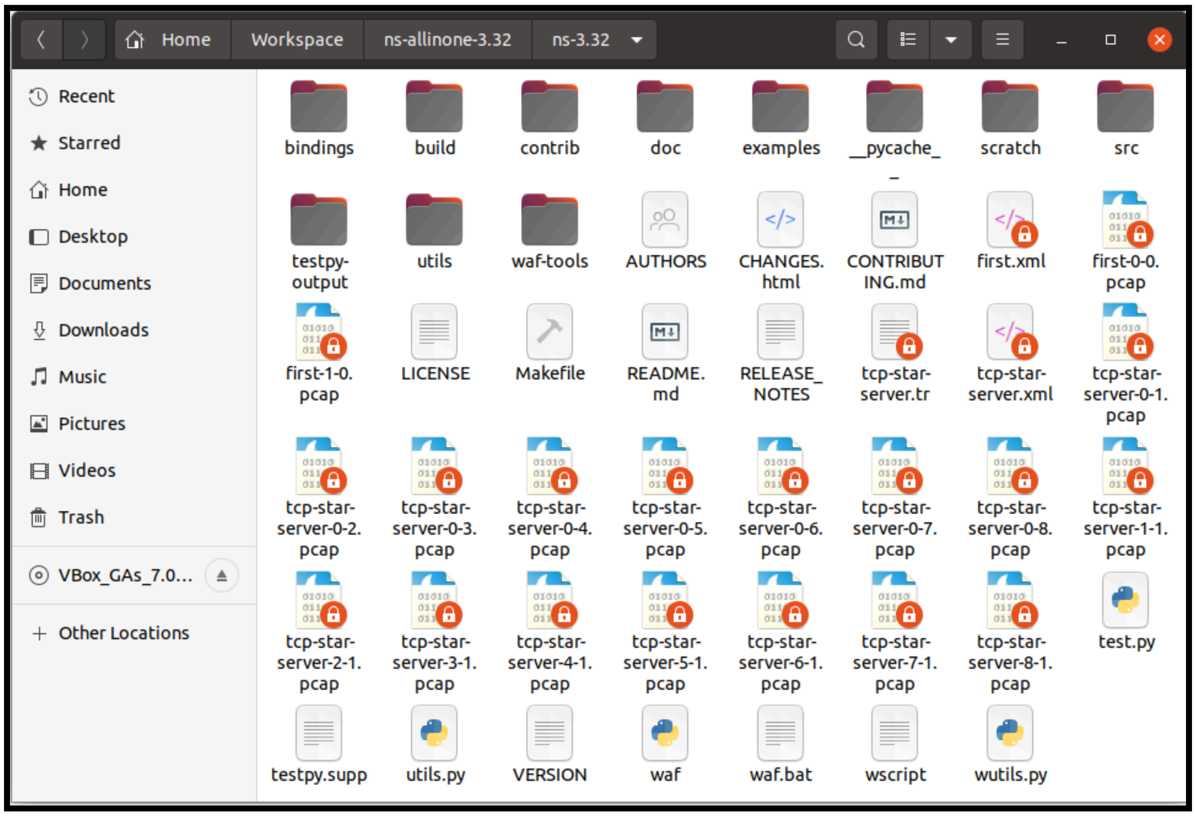
```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scr
atch/tcp-star-server.cc"
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.903s)
PacketSink:PacketSink(0x558a80c90060)
PacketSink:StartApplication(0x558a80c90060)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80c9c4b0, 03-07-0a:01:01:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80c9d0a0, 03-07-0a:01:02:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80c9e0a0, 03-07-0a:01:03:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80c9f380, 03-07-0a:01:04:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80ca0660, 03-07-0a:01:05:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80ca1940, 03-07-0a:01:06:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80ca2c20, 03-07-0a:01:07:02:01:c0:00)
PacketSink:HandleAccept(0x558a80c90060, 0x558a80ca3f00, 03-07-0a:01:08:02:01:c0:00)
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9c4b0)
At time +1.40249s packet sink received 250 bytes from 10.1.1.2 port 49153 total Rx 250 bytes
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9d0a0)
At time +1.40249s packet sink received 250 bytes from 10.1.2.2 port 49153 total Rx 500 bytes
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9e0a0)
At time +1.40249s packet sink received 250 bytes from 10.1.3.2 port 49153 total Rx 750 bytes
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9f380)
At time +1.40249s packet sink received 250 bytes from 10.1.4.2 port 49153 total Rx 1000 byte
```



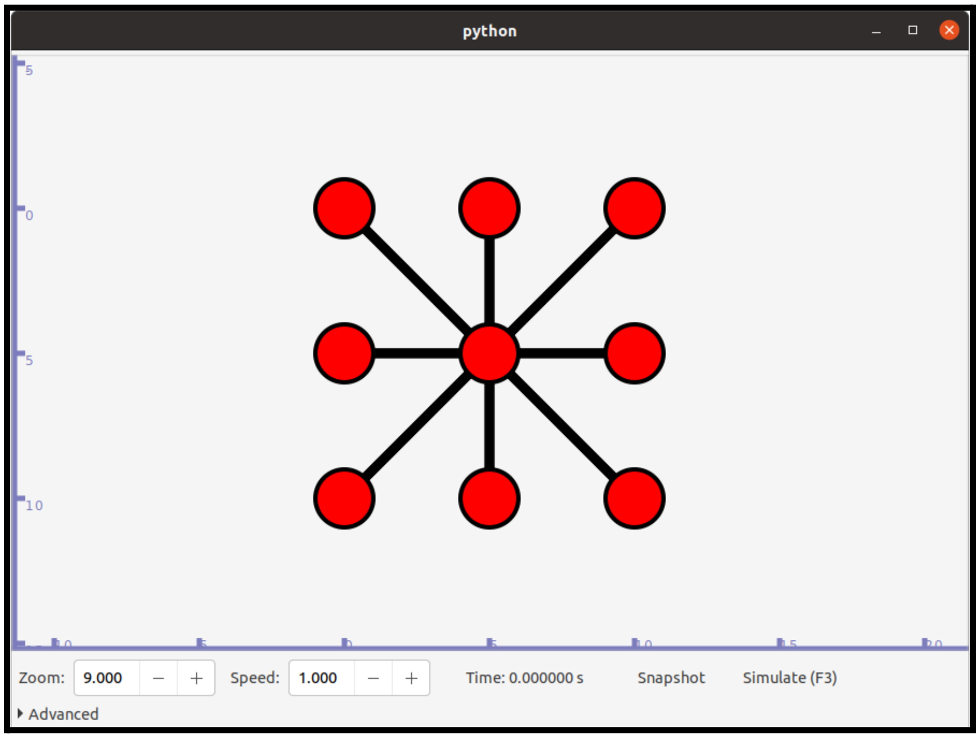
iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/ns-3.32

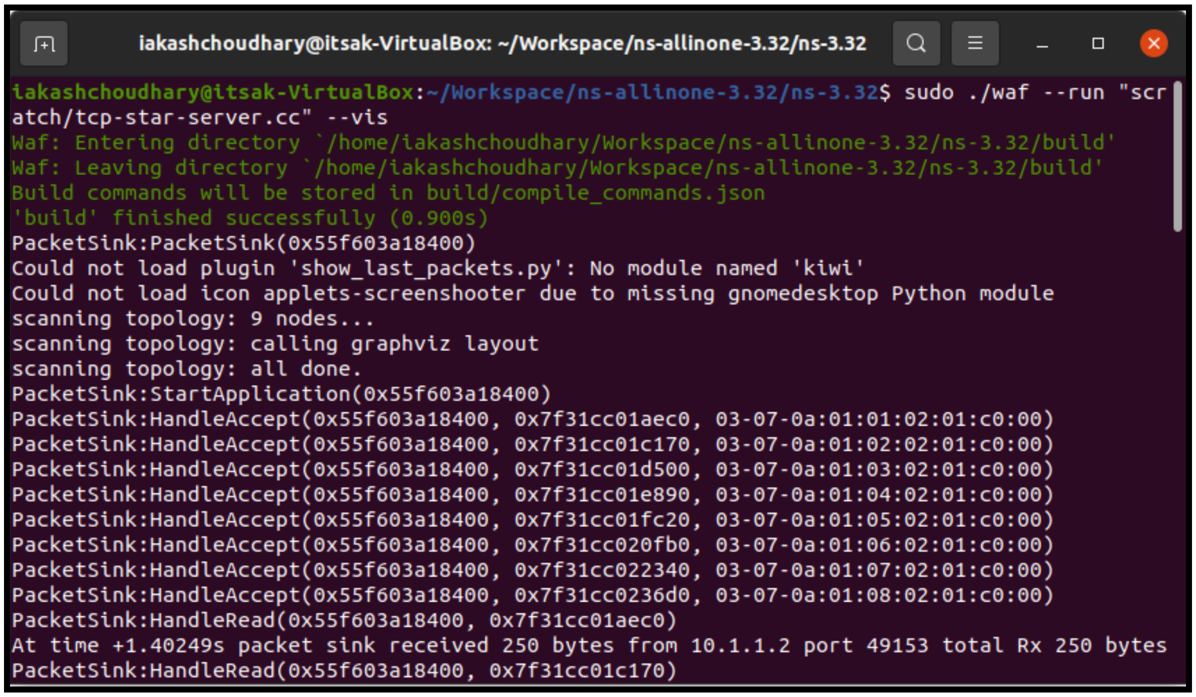
```
PacketSink:HandleRead(0x558a80c90060, 0x558a80ca3f00)
At time +9.80369s packet sink received 250 bytes from 10.1.8.2 port 49153 total Rx 44000 byt
es
PacketSink:StopApplication(0x558a80c90060)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80c93f50)
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9c4b0)
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9d0a0)
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9e0a0)
PacketSink:HandleRead(0x558a80c90060, 0x558a80c9f380)
PacketSink:HandleRead(0x558a80c90060, 0x558a80ca0660)
PacketSink:HandleRead(0x558a80c90060, 0x558a80ca1940)
PacketSink:HandleRead(0x558a80c90060, 0x558a80ca2c20)
PacketSink:HandleRead(0x558a80c90060, 0x558a80ca3f00)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80c9c4b0)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80c9d0a0)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80c9e0a0)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80c9f380)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80ca0660)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80ca1940)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80ca2c20)
PacketSink:HandlePeerClose(0x558a80c90060, 0x558a80ca3f00)
PacketSink:DoDispose(0x558a80c90060)
PacketSink:~PacketSink(0x558a80c90060)
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$
```

> Using the above command, run `$ **sudo ./waf --run "scratch/tcp-star-server.cc"**` to generate files such as **tcp-star-server.xml**, **.tr**, and other **.pcap** files.

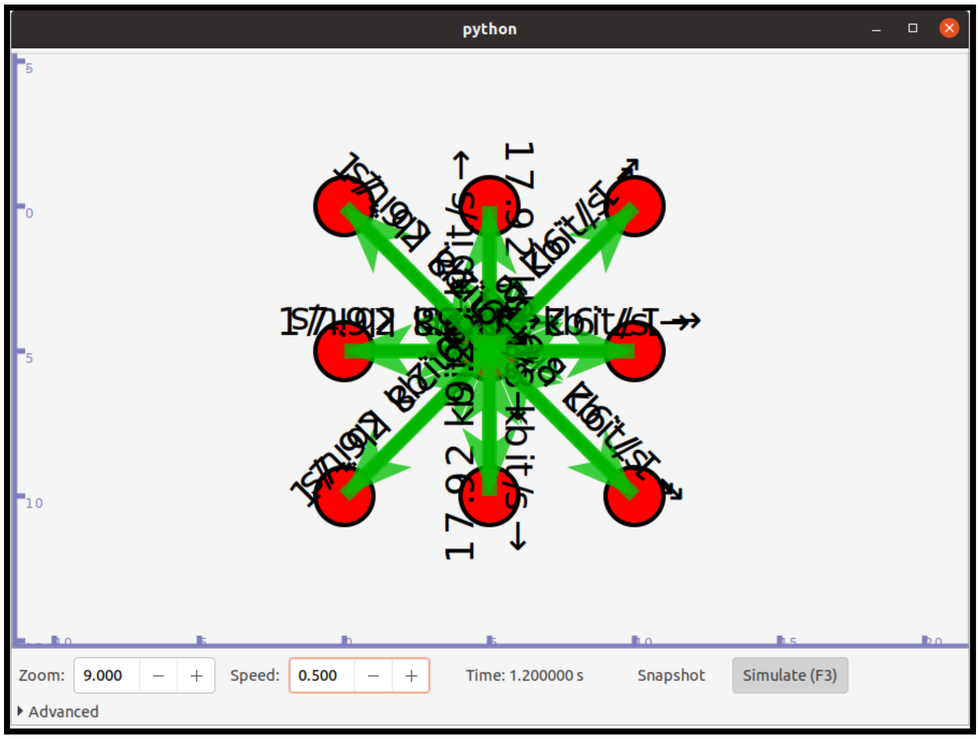


> Perform animation using pygraphviz: **$ sudo ./waf --run "scratch/tcp-star-server.cc" --vis**



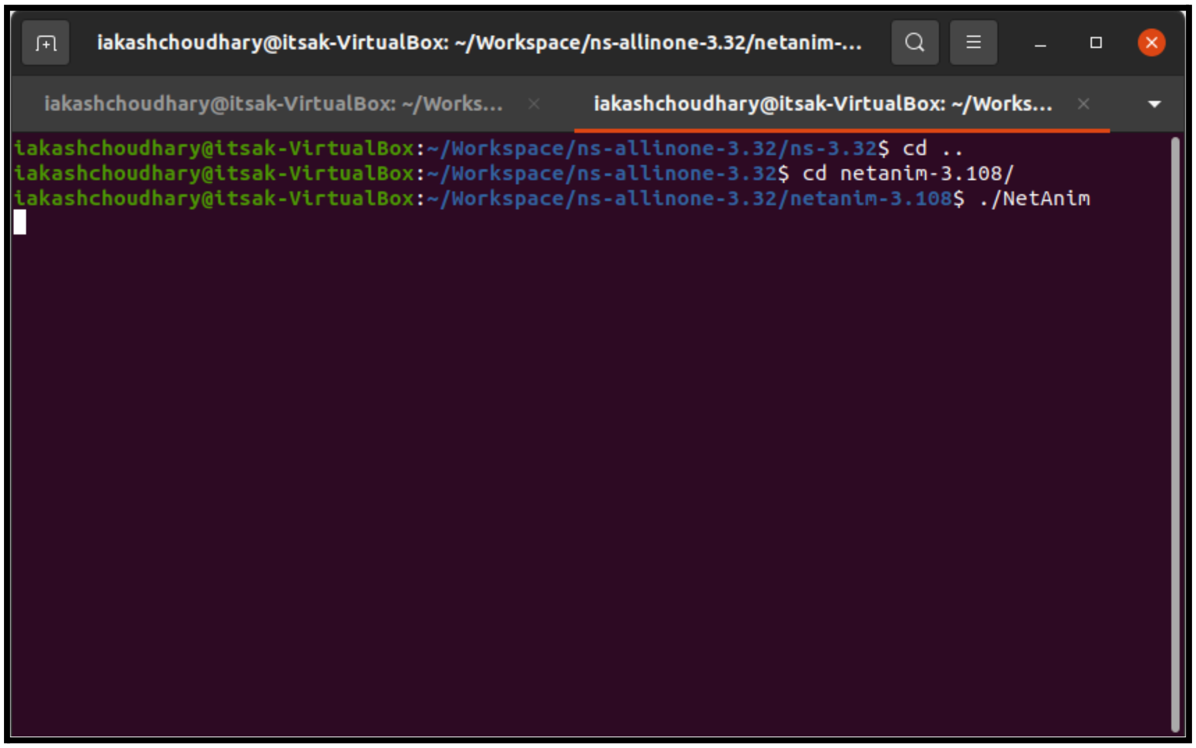


```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ sudo ./waf --run "scr
atch/tcp-star-server.cc" --vis
Waf: Entering directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/iakashchoudhary/Workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.900s)
PacketSink:PacketSink(0x55f603a18400)
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.
PacketSink:StartApplication(0x55f603a18400)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc01aec0, 03-07-0a:01:01:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc01c170, 03-07-0a:01:02:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc01d500, 03-07-0a:01:03:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc01e890, 03-07-0a:01:04:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc01fc20, 03-07-0a:01:05:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc020fb0, 03-07-0a:01:06:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc022340, 03-07-0a:01:07:02:01:c0:00)
PacketSink:HandleAccept(0x55f603a18400, 0x7f31cc0236d0, 03-07-0a:01:08:02:01:c0:00)
PacketSink:HandleRead(0x55f603a18400, 0x7f31cc01aec0)
At time +1.40249s packet sink received 250 bytes from 10.1.1.2 port 49153 total Rx 250 bytes
PacketSink:HandleRead(0x55f603a18400, 0x7f31cc01c170)
```

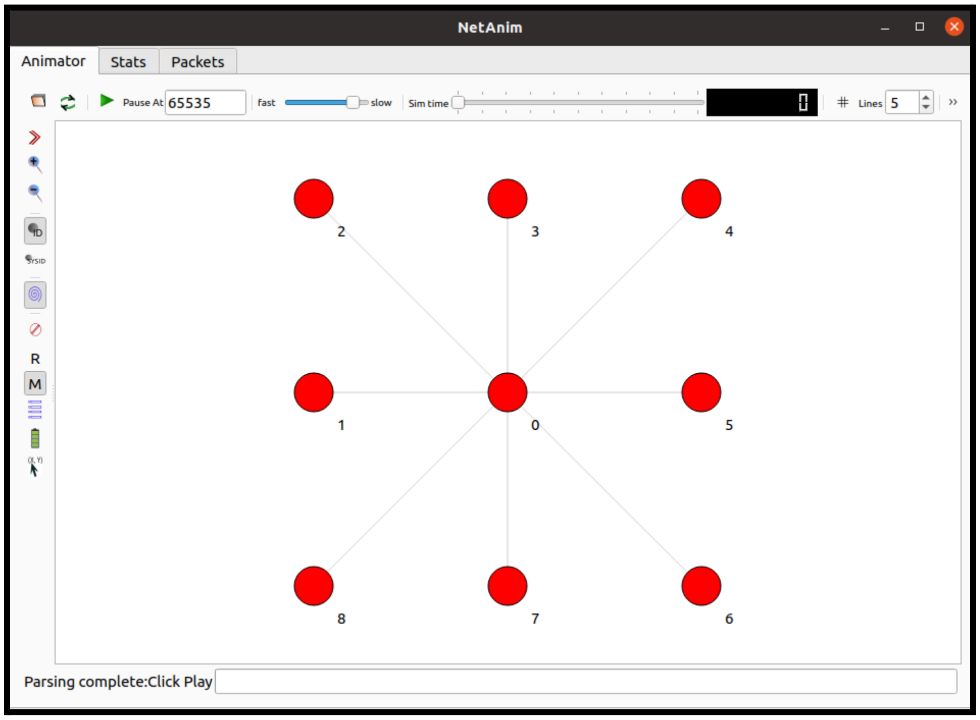

python
Zoom: 9.000
Speed: 0.500
Time: 1.200000 s
Snapshot
Simulate (F3)
Advanced

> Perform animation using NetAnim:

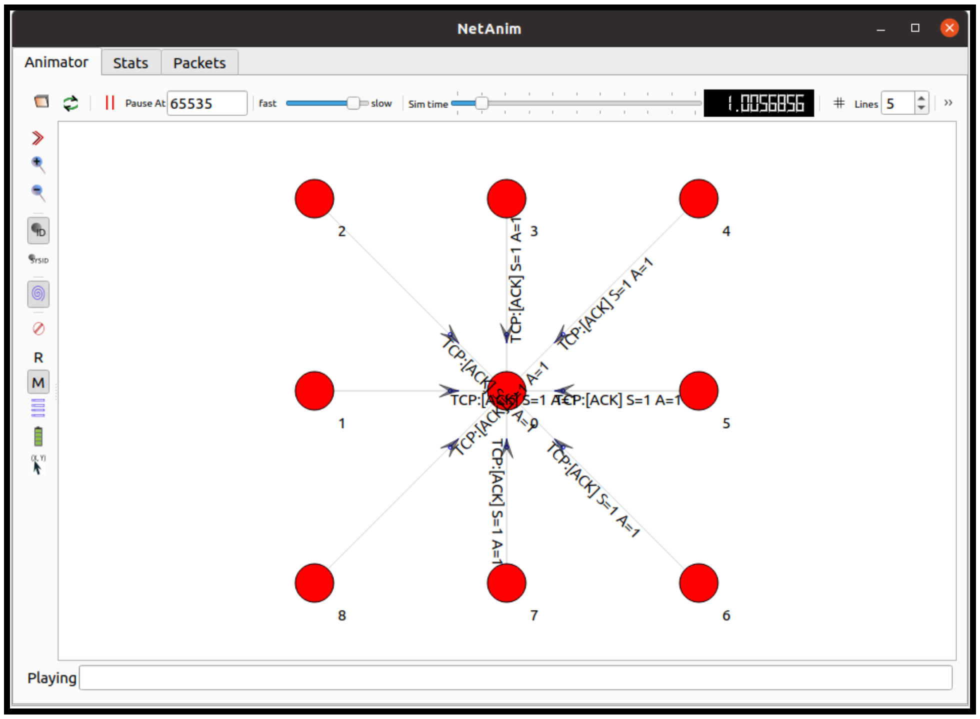
Open the Terminal from ns-3.32; go back to the previous directory by entering **$ cd ..**; then change the directory to netanim-3.108 by typing **$ cd netanim-3.108/**; lastly, open NetAnim by using the command **$ ./NetAnim** | Open **tcp-star-server.xml**.


iakashchoudhary@itsak-VirtualBox: ~/Workspace/ns-allinone-3.32/netanim-...
iakashchoudhary@itsak-VirtualBox: ~/Works...
iakashchoudhary@itsak-VirtualBox: ~/Works...

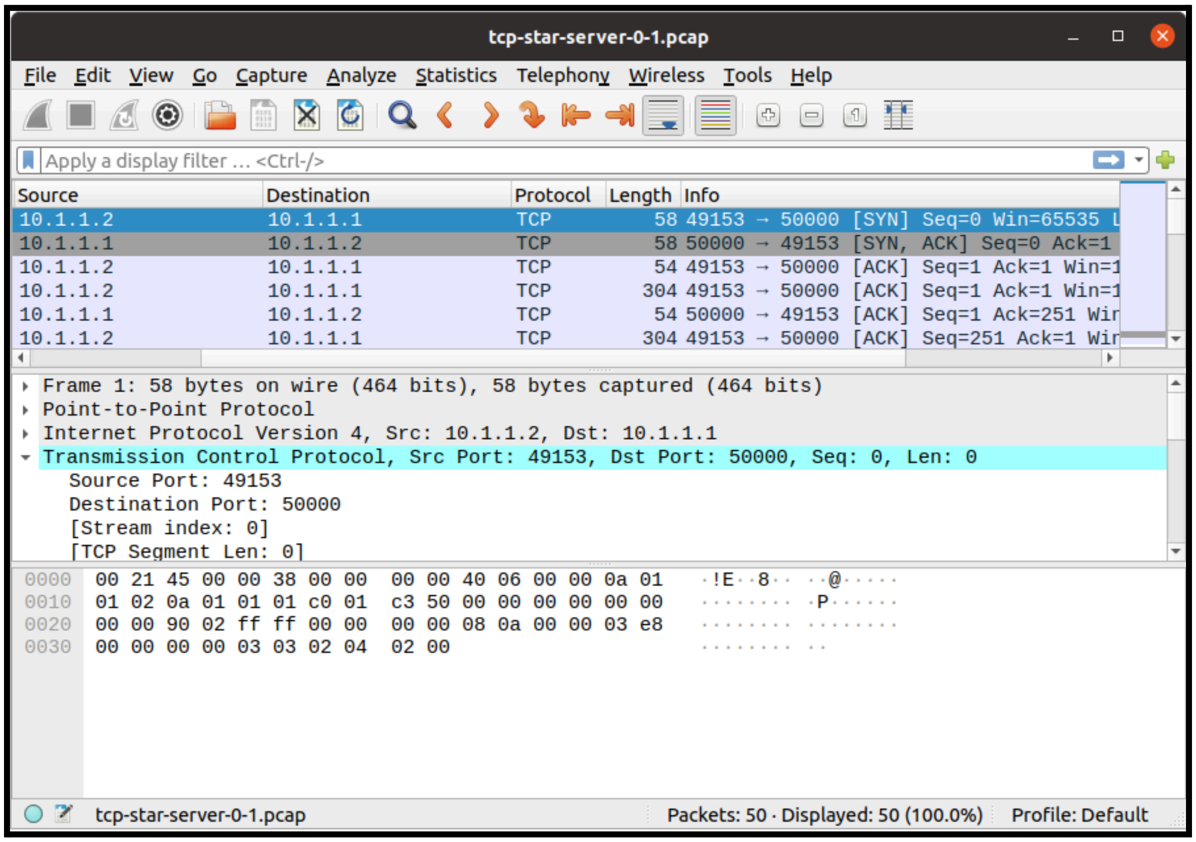
```
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/ns-3.32$ cd ..
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32$ cd netanim-3.108/
iakashchoudhary@itsak-VirtualBox:~/Workspace/ns-allinone-3.32/netanim-3.108$ ./NetAnim
```



Parsing complete:Click Play


NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
1.0056856
Lines 5
R
M
Playing
0
1
2
3
4
5
6
7
8
TCP:[ACK] S=1 A=1

> **Double click** on any of the **.pcap files** to **analyze** the network. Here, we are using the **tcp-star-server-0-1.pcap** file to **monitor the network**.


tcp-star-server-0-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
Source          Destination     Protocol  Length Info
10.1.1.2        10.1.1.1        TCP           58 49153 → 50000 [SYN] Seq=0 Win=65535 L
10.1.1.1        10.1.1.2        TCP           58 50000 → 49153 [SYN, ACK] Seq=0 Ack=1
10.1.1.2        10.1.1.1        TCP           54 49153 → 50000 [ACK] Seq=1 Ack=1 Win=1
10.1.1.2        10.1.1.1        TCP          304 49153 → 50000 [ACK] Seq=1 Ack=1 Win=1
10.1.1.1        10.1.1.2        TCP           54 50000 → 49153 [ACK] Seq=1 Ack=251 Wir
10.1.1.2        10.1.1.1        TCP          304 49153 → 50000 [ACK] Seq=251 Ack=1 Wir

▸ Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 10.1.1.2, Dst: 10.1.1.1
▾ Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
    Source Port: 49153
    Destination Port: 50000
    [Stream index: 0]
    [TCP Segment Len: 0]

0000  00 21 45 00 00 38 00 00  00 00 40 06 00 00 0a 01   ·!E··8·· ··@·····
0010  01 02 0a 01 01 01 c0 01  c3 50 00 00 00 00 00 00   ········ ·P······
0020  00 00 90 02 ff ff 00 00  00 00 08 0a 00 00 03 e8   ········ ········
0030  00 00 00 00 03 03 02 04  02 00                     ········ ··

tcp-star-server-0-1.pcap        Packets: 50 · Displayed: 50 (100.0%)    Profile: Default
```

Conclusion: NetAnim provides a GUI for simulating network operations. The TCP 3-way handshake is a protocol used to establish a connection between a client and server in a TCP/IP network. The client initiates the connection, the server responds with SYN-ACK, and the client acknowledges, establishing the connection.