

8 Program to simulate UDP Client Server.

What is UDP ?

UDP stands for User Datagram Protocol. It's one of the core protocols of the Internet Protocol (IP) suite, often used for sending short messages called datagrams between devices on a network.

Key Features of UDP:

- Connectionless: Unlike TCP, UDP doesn't establish a connection before sending data. It just sends it.
- Fast but unreliable: There's no guarantee of delivery, order, or error checking. If data gets lost or arrives out of order, UDP doesn't correct it.
- Low overhead: Because it skips checks and handshakes, it's faster and more lightweight than TCP.

Common Uses of UDP:

- Streaming (video/audio): Like Zoom, Skype, or YouTube live, where speed is more important than perfect delivery.
- Online gaming: Fast response is critical, and occasional data loss is acceptable.
- DNS (Domain Name System): When you type a website address, your computer sends a quick UDP query to get the IP.

```
#include <fstream>

#include "ns3/core-module.h"

#include "ns3/csma-module.h"

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

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

```
LogComponentEnable ("UdpClient",
LOG_LEVEL_INFO); LogComponentEnable
("UdpServer", LOG_LEVEL_INFO);
```

```
bool useV6 = false;
bool tracing = true;
Address
serverAddress;
```

```
CommandLine cmd (__FILE__);
cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
cmd.AddValue("tracing","Enable pcap tracing",
tracing);
```

```
cmd.Parse (argc, argv);
```

```
NS_LOG_INFO ("Create
nodes."); NodeContainer n;
n.Create (2);
```

```
InternetStackHelper internet;
internet.Install (n);
```

```
NS_LOG_INFO ("Create channels.");
```

```
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate
(5000000))); csma.SetChannelAttribute ("Delay", TimeValue (Milliseconds
(2))); csma.SetDeviceAttribute ("Mtu", UIntegerValue (1400));
NetDeviceContainer d = csma.Install (n);
```

```

NS_LOG_INFO ("Assign IP
Addresses."); if (useV6 == false)
{
    Ipv4AddressHelper ipv4;
    ipv4.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer i = ipv4.Assign (d);
    serverAddress = Address (i.GetAddress (1));
}
else
{
    Ipv6AddressHelper ipv6;
    ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix
(64)); Ipv6InterfaceContainer i6 = ipv6.Assign (d);
    serverAddress = Address(i6.GetAddress (1,1));
}

NS_LOG_INFO ("Create Applications.");

uint16_t port = 4000;
UdpServerHelper server (port);
ApplicationContainer apps = server.Install (n.Get
(1)); apps.Start (Seconds (1.0));
apps.Stop (Seconds (10.0));

uint32_t MaxPacketSize = 1024;
Time interPacketInterval = Seconds (0.05);
uint32_t maxPacketCount = 320;
UdpClientHelper client (serverAddress, port);
client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
client.SetAttribute ("Interval", TimeValue (interPacketInterval));

```

```

client.SetAttribute ("PacketSize", UIntegerValue
(MaxPacketSize)); apps = client.Install (n.Get (0));
apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));

```

```

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

```

Output:-

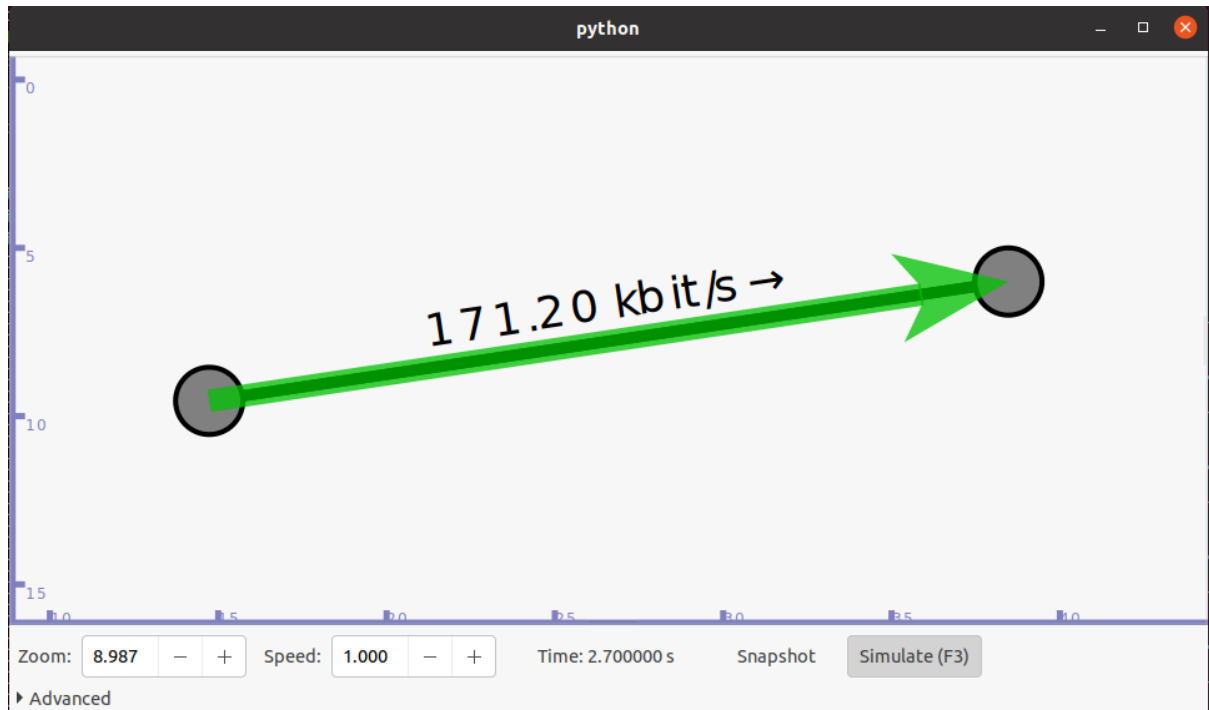
./waf --run scratch/UDF-Client-Server

```

bvimt@bvimt-VirtualBox: ~/workspace/ns-allinone-3.32/ns-3.32
bvimt@bvimt-VirtualBox:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/UDF-Client-Server
Waf: Entering directory '/home/bvimt/workspace/ns-allinone-3.32/ns-3.32/build'
[1987/2064] Compiling scratch/UDF-Client-Server.cc
[1988/2064] Compiling scratch/udp-client-server.cc
[1989/2064] Compiling scratch/subdir/scratch-simulator-subdir.cc
[1998/2064] Compiling scratch/third.cc
[2021/2064] Linking build/scratch/subdir/subdir
[2022/2064] Linking build/scratch/third
[2023/2064] Linking build/scratch/udp-client-server
[2024/2064] Linking build/scratch/UDF-Client-Server
Waf: Leaving directory '/home/bvimt/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.576s)
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 0 Time: +2s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 0 Uid: 0 TXtime: +2e+09ns RXtime: +2.01592e+09ns Delay: +1.59188e+07ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 11 Time: +2.05s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 1 Uid: 11 TXtime: +2.05e+09ns RXtime: +2.05371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 14 Time: +2.1s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 2 Uid: 14 TXtime: +2.1e+09ns RXtime: +2.10371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 17 Time: +2.15s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 3 Uid: 17 TXtime: +2.15e+09ns RXtime: +2.15371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 20 Time: +2.2s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 4 Uid: 20 TXtime: +2.2e+09ns RXtime: +2.20371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 23 Time: +2.25s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 5 Uid: 23 TXtime: +2.25e+09ns RXtime: +2.25371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 26 Time: +2.3s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 6 Uid: 26 TXtime: +2.3e+09ns RXtime: +2.30371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 29 Time: +2.35s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 7 Uid: 29 TXtime: +2.35e+09ns RXtime: +2.35371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 32 Time: +2.4s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 8 Uid: 32 TXtime: +2.4e+09ns RXtime: +2.40371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 35 Time: +2.45s

```

./waf --run scratch/UDF-Client-Server --vis



Command line input

```
Activities Terminal
bvmitt@bvmitt-VirtualBox: ~/workspace/ns-allinone-3.32/ns-3.32
bvmitt@bvmitt-VirtualBox:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/star --vis
Waf: Entering directory '/home/bvmitt/workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory '/home/bvmitt/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.803s)
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.
bvmitt@bvmitt-VirtualBox:~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run "scratch/star --nSpokes=6" --vis
```

9 Program to simulate DHCP server and Clients.

What is DHCP?

DHCP stands for **Dynamic Host Configuration Protocol**. It's a network management protocol used to **automatically assign IP addresses** and other network configuration details to devices (clients) on a network.

DHCP Server

The **DHCP Server** is the system or device (like a router or dedicated server) that:

- Manages a pool of IP addresses.
- Assigns an IP address to each device that connects.
- Provides additional info like:
 - Subnet mask
 - Default gateway
 - DNS server

Think of it like a hotel front desk giving out room numbers and Wi-Fi details to guests when they check in.

☑ DHCP Clients

The **DHCP Clients** are devices like:

- Laptops
- Phones
- Printers
- Smart TVs
- Any network-enabled device

/*

* Network layout:

*

* R0 is a DHCP server. The DHCP server announced R1 as the default router.

* Nodes N1 will send UDP Echo packets to node A.

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

* Things to notice:

* 1) The routes in A are manually set to have R1 as the default router,

* just because using a dynamic outing in this example is an overkill.

- * 2) R1's address is set statically though the DHCP server helper interface.
- * This is useful to prevent address conflicts with the dynamic pool.
- * Not necessary if the DHCP pool is not conflicting with static addresses.
- * 3) N2 has a dynamically-assigned, static address (i.e., a fixed address assigned via DHCP).
- *
- */

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

```
#include "ns3/internet-apps-
```

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

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

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

```
point-module.h" #include
```

```
"ns3/applications-module.h"
```

```
using namespace ns3;
```

```
NS_LOG_COMPONENT_DEFINE ("DhcpExample");
```

```
int
```

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

```
{
```

```
CommandLine cmd (__FILE__);
```

```
bool verbose = true;
```

```
bool tracing = true;
```



```
cmd.AddValue ("verbose", "turn on the logs", verbose);

cmd.AddValue ("tracing", "turn on the tracing", tracing);


cmd.Parse (argc, argv);


// GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));


if (verbose)
{
    LogComponentEnable ("DhcpServer", LOG_LEVEL_ALL);

    LogComponentEnable ("DhcpClient", LOG_LEVEL_ALL);

    LogComponentEnable ("UdpEchoServerApplication",
        LOG_LEVEL_INFO); LogComponentEnable
        ("UdpEchoClientApplication", LOG_LEVEL_INFO);
}


Time stopTime = Seconds (20);


NS_LOG_INFO ("Create
nodes."); NodeContainer
nodes; NodeContainer router;

nodes.Create (3);

router.Create (2);


NodeContainer net (nodes, router);
```

```
NS_LOG_INFO ("Create  
channels."); CsmaHelper csma;  
  
csma.SetChannelAttribute ("DataRate", StringValue  
("5Mbps")); csma.SetChannelAttribute ("Delay", StringValue  
("2ms")); csma.SetDeviceAttribute ("Mtu", UIntegerValue  
(1500)); NetDeviceContainer devNet = csma.Install (net);  
  
NodeContainer p2pNodes;  
  
p2pNodes.Add (net.Get (4)); //R1 router. Adding R1 to point to point network  
  
p2pNodes.Create (1); // Creating node A  
  
PointToPointHelper pointToPoint;  
  
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));  
  
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));  
  
//netdevice for Point to point  
  
NetDeviceContainer p2pDevices;  
  
p2pDevices = pointToPoint.Install (p2pNodes);  
  
InternetStackHelper tcpip;  
  
tcpip.Install (nodes);  
  
tcpip.Install (router);  
  
tcpip.Install (p2pNodes.Get (1)); //A  
  
Ipv4AddressHelper address;
```

```
address.SetBase ("172.30.1.0", "255.255.255.0");
```

```
Ipv4InterfaceContainer p2pInterfaces;
```

```
p2pInterfaces = address.Assign
```

```
(p2pDevices);
```

```
// manually add a routing entry because we don't want to add a dynamic routing
```

```
Ipv4StaticRoutingHelper ipv4RoutingHelper;
```

```
Ptr<Ipv4> ipv4Ptr = p2pNodes.Get (1)->GetObject<Ipv4> (); //A
```

```
Ptr<Ipv4StaticRouting> staticRoutingA = ipv4RoutingHelper.GetStaticRouting
```

```
(ipv4Ptr); staticRoutingA->AddNetworkRouteTo (Ipv4Address ("172.30.0.0"),
```

```
Ipv4Mask ("/24"),
```

```
Ipv4Address ("172.30.1.1"), 1);
```

```
NS_LOG_INFO ("Setup the IP addresses and create DHCP
```

```
applications."); DhcpHelper dhcpHelper;
```

```
// The router must have a fixed IP.
```

```
/*InstallFixedAddress ( Ptr< NetDevice >
```

```
netDevice, Ipv4Address addr,
```

```
Ipv4Mask mask
```

```
) */
```

```
//Assign IP address of R1
```

```
Ipv4InterfaceContainer fixedNodes = dhcpHelper.InstallFixedAddress (devNet.Get (4), Ipv4Address  
("172.30.0.17"), Ipv4Mask ("/24"));
```

```
// Not really necessary, IP forwarding is enabled by default in IPv4.
```

```
fixedNodes.Get (0).first->SetAttribute ("IpForward", BooleanValue
```

```
(true));
```

```
// Configuring DHCP server R0
```

```
/*DhcpHelper::InstallDhcpServer ( Ptr< NetDevice >
```

```
netDevice, Ipv4Address
```

```
serverAddr, //IP address of R0
```

```
Ipv4Address
```

```
poolAddr
```

```
, Ipv4Mask
```

```
poolMas
```

```
k, Ipv4Address minAddr,
```

```
Ipv4Address maxAddr,
```

```
Ipv4Address gateway = Ipv4Address ()// R1
```

```
) */
```

```
ApplicationContainer dhcpServerApp = dhcpHelper.InstallDhcpServer (devNet.Get (3), Ipv4Address  
("172.30.0.12"),
```

```
Ipv4Address ("172.30.0.0"), Ipv4Mask ("/24"),
```

```
Ipv4Address ("172.30.0.10"), Ipv4Address
```

```
("172.30.0.15"),
```

```
Ipv4Address ("172.30.0.17"));
```

```
// This is just to show how it can be done.
```

```
//DHCP static IP for N2
```

```
DynamicCast<DhcpServer> (dhcpServerApp.Get (0))->AddStaticDhcpEntry (devNet.Get (2)-  
>GetAddress (), Ipv4Address ("172.30.0.14"));
```

```
dhcpServerApp.Start (Seconds (0.0));
```

```
dhcpServerApp.Stop (stopTime);
```

```
// Configuring DHCP clients
```

```
// Netdevice for DHCP clients
```

```
NetDeviceContainer dhcpClientNetDevs;
```

```
dhcpClientNetDevs.Add (devNet.Get (0)); //
```

```
N0  dhcpClientNetDevs.Add  (devNet.Get
```

```
(1));  //  N2  dhcpClientNetDevs.Add
```

```
(devNet.Get (2)); //N3
```

```
// installing DHCP client Application on N0, N1 and N2
```

```
ApplicationContainer dhcpClients = dhcpHelper.InstallDhcpClient (dhcpClientNetDevs);
```

```
dhcpClients.Start (Seconds
```

```
(1.0)); dhcpClients.Stop
```

```
(stopTime);
```

```
UdpEchoServerHelper echoServer (9);
```

```
ApplicationContainer serverApps = echoServer.Install
```

```
(p2pNodes.Get (1)); serverApps.Start (Seconds (0.0));
```

```
serverApps.Stop (stopTime);
```

```
UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress
```

```
(1), 9); echoClient.SetAttribute ("MaxPackets", UIntegerValue
```

```
(100)); echoClient.SetAttribute ("Interval", TimeValue
```

```
(Seconds (1.0)); echoClient.SetAttribute ("PacketSize",
```

```
UIntegerValue (1024));
```

```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));
```

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

```
clientApps.Stop (stopTime);
```

```
Simulator::Stop (stopTime + Seconds (10.0));
```

```
if (tracing)
```

```
{
    csmc.EnablePcapAll ("dhcp-csmc");
    pointToPoint.EnablePcapAll ("dhcp-
    p2p");
}
```

```
NS_LOG_INFO ("Run
```

```
Simulation."); Simulator::Run
```

```
(); Simulator::Destroy ();
```

```
NS_LOG_INFO ("Done.");
```

```
}
```

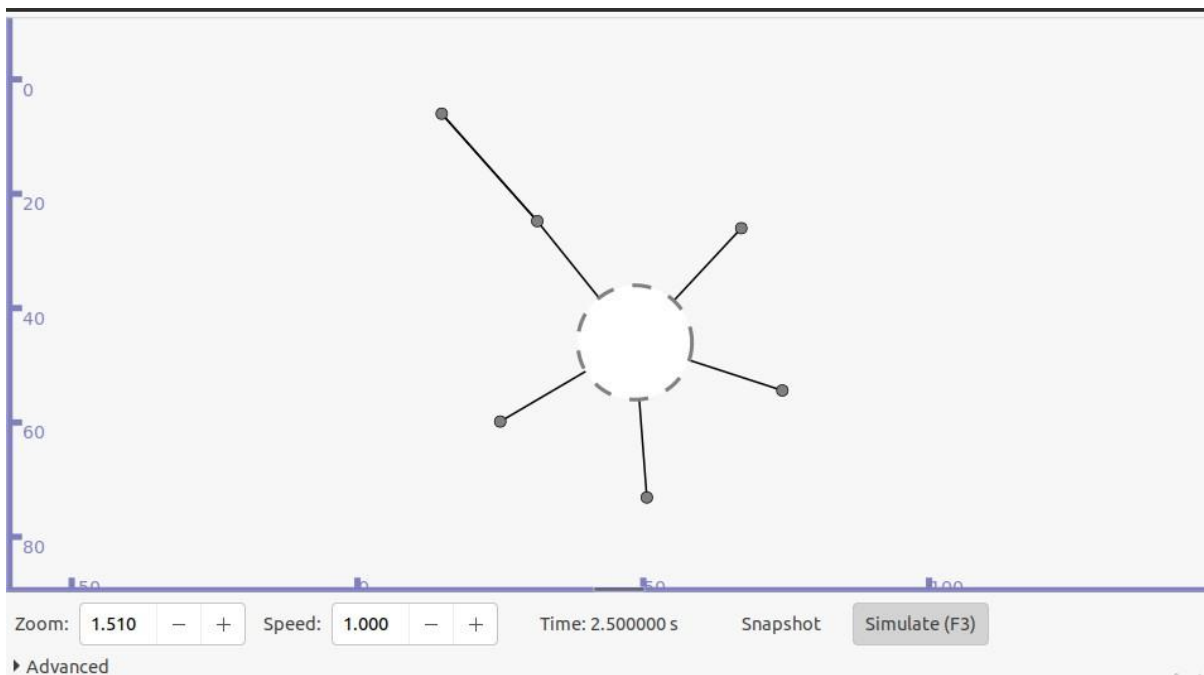
```
./waf --run sratch/dhcp.cc
```

```

bvimit@bvimit-VirtualBox: ~/workspace/ns-allinone-3.32/ns-3.32$ ./waf --run sratch/dhcp
Waf: Entering directory `/home/bvimit/workspace/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/bvimit/workspace/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.235s)
DhcpServer:DhcpServer(0x55c1c7a1bbe0)
DhcpServer:AddStaticDhcpEntry(0x55c1c7a1bbe0, 02-06-00:00:00:00:00:03, 172.30.0.14)
DhcpClient:DhcpClient(0x55c1c7a5a0e0)
DhcpClient:DhcpClient(0x55c1c7adf0a0)
DhcpClient:DhcpClient(0x55c1c7ae0520)
DhcpServer:StartApplication(0x55c1c7a1bbe0)
Adding 172.30.0.10 to the pool
Adding 172.30.0.11 to the pool
Adding 172.30.0.13 to the pool
Adding 172.30.0.14 to the pool
Adding 172.30.0.15 to the pool
DhcpClient:StartApplication(0x55c1c7a5a0e0)
My address is 02-06-00:00:00:00:00:01
My m_chaddr is 00-10-00:00:00:00:00:01:00:00:00:00:00:00:00:00:00:00:00:00:00
DhcpClient:Boot(0x55c1c7a5a0e0)
DHCP DISCOVER sent
DhcpClient:StartApplication(0x55c1c7adf0a0)
My address is 02-06-00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
My m_chaddr is 00-10-00:00:00:00:00:00:02:00:00:00:00:00:00:00:00:00:00:00:00

```

```
./waf --run sratch/dhcp.cc --vis
```



Wireshark

Wireshark - Apr 17 14:26 • dhcp-csma-0-0.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

Time	Source	Destination	No.	Protocol	Length	Info
0.000000	0.0.0.0	255.255.255.255	1	DHCP	290	DHCP Discover - Transaction ID 0xb8e60000
0.004986	172.30.0.12	255.255.255.255	2	DHCP	326	DHCP Offer - Transaction ID 0xb8e60000
0.007717	0.0.0.0	255.255.255.255	3	DHCP	290	DHCP Discover - Transaction ID 0xb8ceb000
0.010239	172.30.0.12	255.255.255.255	4	DHCP	326	DHCP Offer - Transaction ID 0xb8ceb000
0.013165	0.0.0.0	255.255.255.255	5	DHCP	290	DHCP Discover - Transaction ID 0xb8d85000
0.015687	172.30.0.12	255.255.255.255	6	DHCP	326	DHCP Offer - Transaction ID 0xb8d85000
5.004986	0.0.0.0	255.255.255.255	7	DHCP	296	DHCP Request - Transaction ID 0xb8e60000
5.009925	172.30.0.12	255.255.255.255	8	DHCP	290	DHCP ACK - Transaction ID 0xb8e60000
5.012713	0.0.0.0	255.255.255.255	9	DHCP	296	DHCP Request - Transaction ID 0xb8ceb000
5.015178	172.30.0.12	255.255.255.255	10	DHCP	290	DHCP ACK - Transaction ID 0xb8ceb000
5.018161	0.0.0.0	255.255.255.255	11	DHCP	296	DHCP Request - Transaction ID 0xb8d85000
5.020626	172.30.0.12	255.255.255.255	12	DHCP	290	DHCP ACK - Transaction ID 0xb8d85000

Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)

Ethernet II, Src: 00:00:00:00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255

User Datagram Protocol, Src Port: 68, Dst Port: 67

Dynamic Host Configuration Protocol (Discover)

0000 ff ff ff ff ff 00 00 00 00 01 08 00 45 00E..

0010 01 10 00 00 00 00 40 11 00 00 00 00 00 ff ff@.....

0020 ff ff 00 44 00 43 00 fc 00 00 01 01 06 00 b8 e6D.C.....

0030 00 00 00 01 00 00 00 00 00 00 00 00 00 00 0000000000

0040 00 00 00 00 00 00 00 00 00 00 01 00 00 00 0000000000

0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

00f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000000

dhcpc-csma-0-0.pcap Packets: 14 - Displayed: 14 (100.0%) Profile: Default

10 Program to simulate FTP using TCP.

What is FTP ?

FTP stands for **File Transfer Protocol**. It is a standard network protocol used to transfer files between a client and a server over a computer network, such as the internet or an intranet.

FTP works on a client-server model, where:

- **FTP Client** is a program or application used to connect to an FTP server.
- **FTP Server** is a machine that stores files and allows clients to upload, download, or manage those files.

Key features of FTP:

1. **File Transfers:** FTP allows users to upload and download files between their local machine and a remote server.
2. **File Management:** It provides capabilities for renaming, deleting, and managing files and directories on the server.
3. **Authentication:** FTP usually requires a username and password for access (though there are anonymous FTP servers that allow access without authentication).
4. **Modes:** FTP operates in two modes:
 - **Active Mode:** The client opens a port for data transfer, and the server connects to the client.
 - **Passive Mode:** The server opens a port for data transfer, and the client connects to it.

Common FTP Commands:

- `get`: Download a file from the server.
- `put`: Upload a file to the server.
- `ls`: List files in the current directory.
- `cd`: Change the directory on the server.


```
#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/csma-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FTP_Simulation");

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

    Time::SetResolution (Time::NS);
    LogComponentEnable ("OnOffApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("OnOffApplication", LOG_LEVEL_INFO);

    NodeContainer csmaNodes;

    csmaNodes.Create(2);

    // using csma helper to configure attributes
    // and creating connectivity between two nodes
```

```
CsmaHelper csma;

// set channel attributes for device
    csma.SetChannelAttribute ("DataRate",StringValue("1Mbps"));
    csma.SetChannelAttribute ("Delay",StringValue("20ms"));

// installing nodes to create, configure, install devices of required type
    NetDeviceContainer csmaDevices;

    csmaDevices = csma.Install(csmaNodes);

// installing protocol stack in nodes; once executed will install TCP,UDP, IP etc. on each node
of the nodeContainer

    InternetStackHelper stack;
    stack.Install(csmaNodes);

// assigning IP address to the devices, setting base address here
    Ipv4AddressHelper address;
    address.SetBase ("10.10.10.0","255.255.255.0");

    Ipv4InterfaceContainer csmaInterfaces =address.Assign(csmaDevices);

// assigning ip address to the devices, since devices contain nodes, while ip stack is created
within a node

//      uint32_t port_num=21; // port number for Rx
```

Address RxAddress(InetSocketAddress (csmalInterfaces.GetAddress(1), 21)); //Rx
obtaining a IPv4 address, and the port number to listen

PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", RxAddress); //
PacketSinkHelper is a class reference for creating socket references and passing address at
which Rx is instantiated

ApplicationContainer sinkApp = sinkHelper.Install (csmaNodes.Get(1)); //
ApplicationContainer, installs application on the node, in our case it is packet sink helper ;
node here is the server node

sinkApp.Start (Seconds (1.0));

sinkApp.Stop (Seconds (10.0));

//creating a TCP transmitter

// set up address and port number to send TCP traffic to the server

Address TxAddress(InetSocketAddress(csmalInterfaces.GetAddress(1),21));

OnOffHelper clientHelper ("ns3::TcpSocketFactory", TxAddress);

// setting up attributes for onoff application helper

clientHelper.SetAttribute("DataRate",StringValue("1Mbps"));

clientHelper.SetAttribute("PacketSize",UIntegerValue(1280));

ApplicationContainer Tx = clientHelper.Install (csmaNodes.Get
(0));

Tx.Start (Seconds (1.0));

Tx.Stop (Seconds (10.0));

```
csma.EnablePcap("ftpeg",csmaDevices.Get(1),true);
```

```
AsciiTraceHelper ascii;
```

```
csma.EnableAsciiAll(ascii.CreateFileStream  
("ftp_trace.tr"));
```

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

```
return(0);
```

```
}
```

```

bvinit@bvinit-VirtualBox:~/workspace/ns-allinnone-3.32/ns-3.32$ ./waf --run scratch/ftp
Waf: Entering directory '/home/bvinit/workspace/ns-allinnone-3.32/ns-3.32/build'
Waf: Leaving directory '/home/bvinit/workspace/ns-allinnone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.230s)
At time +2.01024s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 1280 bytes
At time +2.02048s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 2560 bytes
At time +2.03072s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 3840 bytes
At time +2.04096s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 5120 bytes
At time +2.0512s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 6400 bytes
At time +2.06144s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 7680 bytes
At time +2.07168s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 8960 bytes
At time +2.08192s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 10240 bytes
At time +2.09216s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 11520 bytes
At time +2.1024s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 12800 bytes
At time +2.11264s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 14080 bytes
At time +2.12288s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 15360 bytes
At time +2.13312s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 16640 bytes
At time +2.14336s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 17920 bytes
At time +2.1536s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 19200 bytes
At time +2.16384s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 20480 bytes
At time +2.17408s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 21760 bytes
At time +2.18432s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 23040 bytes
At time +2.19456s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 24320 bytes
At time +2.2048s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 25600 bytes
At time +2.21504s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 26880 bytes
At time +2.22528s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 28160 bytes
At time +2.23552s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 29440 bytes
At time +2.24576s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 30720 bytes
At time +2.256s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 32000 bytes
At time +2.26624s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 33280 bytes
At time +2.27648s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 34560 bytes
At time +2.28672s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 35840 bytes
At time +2.29696s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 37120 bytes
At time +2.3072s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 38400 bytes
At time +2.31744s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 39680 bytes
At time +2.32768s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 40960 bytes
At time +2.33792s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 42240 bytes
At time +2.34816s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 43520 bytes
At time +2.3584s on-off application sent 1280 bytes to 10.10.10.2 port 21 total Tx 44800 bytes

```

Wireshark Apr 12 16:35

ftpeg-1-0.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

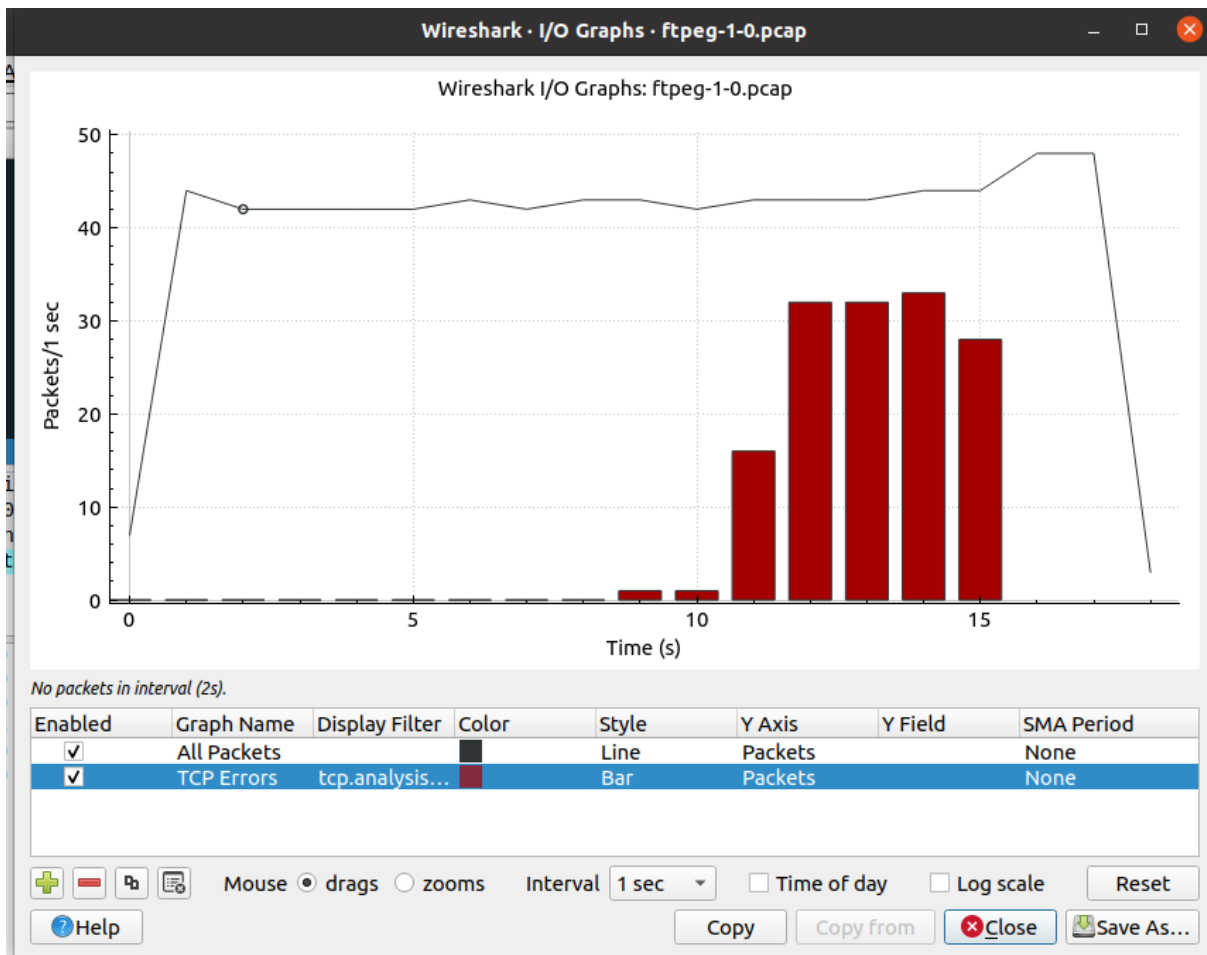
Time	Source	Destination	No.	Protocol	Length	Info
0.000000	00:00:00:00:00:01	Broadcast	1	ARP	64	Who has 10.10.10.2? Tell 10.10.10.1
0.000000	00:00:00:00:00:02	00:00:00:00:00:01	2	ARP	64	10.10.10.2 is at 00:00:00:00:00:02
0.041106	10.10.10.1	10.10.10.2	3	TCP	74	49153 → 21 [SYN] Seq=0 Win=65535 Len=0 TSval=1000 TSecr=0 WS=...
0.044106	00:00:00:00:00:02	Broadcast	4	ARP	64	Who has 10.10.10.1? Tell 10.10.10.2
0.085131	00:00:00:00:00:01	00:00:00:00:00:02	5	ARP	64	10.10.10.1 is at 00:00:00:00:00:01
0.085131	10.10.10.2	10.10.10.1	6	TCP	74	21 → 49153 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 TSval=1070 ...
0.126285	10.10.10.1	10.10.10.2	7	TCP	70	49153 → 21 [ACK] Seq=1 Ack=1 Win=131872 Len=0 TSval=1135 TSecr=...
1.005576	10.10.10.1	10.10.10.2	8	FTP	606	Request: \000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000...
1.005576	10.10.10.2	10.10.10.1	9	TCP	70	21 → 49153 [ACK] Seq=1 Ack=537 Win=131872 Len=0 TSval=2035 TSecr=...
1.051360	10.10.10.1	10.10.10.2	10	FTP	606	Request: \000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000...
1.074400	10.10.10.1	10.10.10.2	11	FTP	278	Request: \000\000\000\000\000\000\000\000\000\000\000\000\000\000\000\000...
1.074400	10.10.10.2	10.10.10.1	12	TCP	70	21 → 49153 [ACK] Seq=1 Ack=1281 Win=131872 Len=0 TSval=2103 TSecr=...

Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0

Ethernet II, Src: 00:00:00:00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Address Resolution Protocol (request)

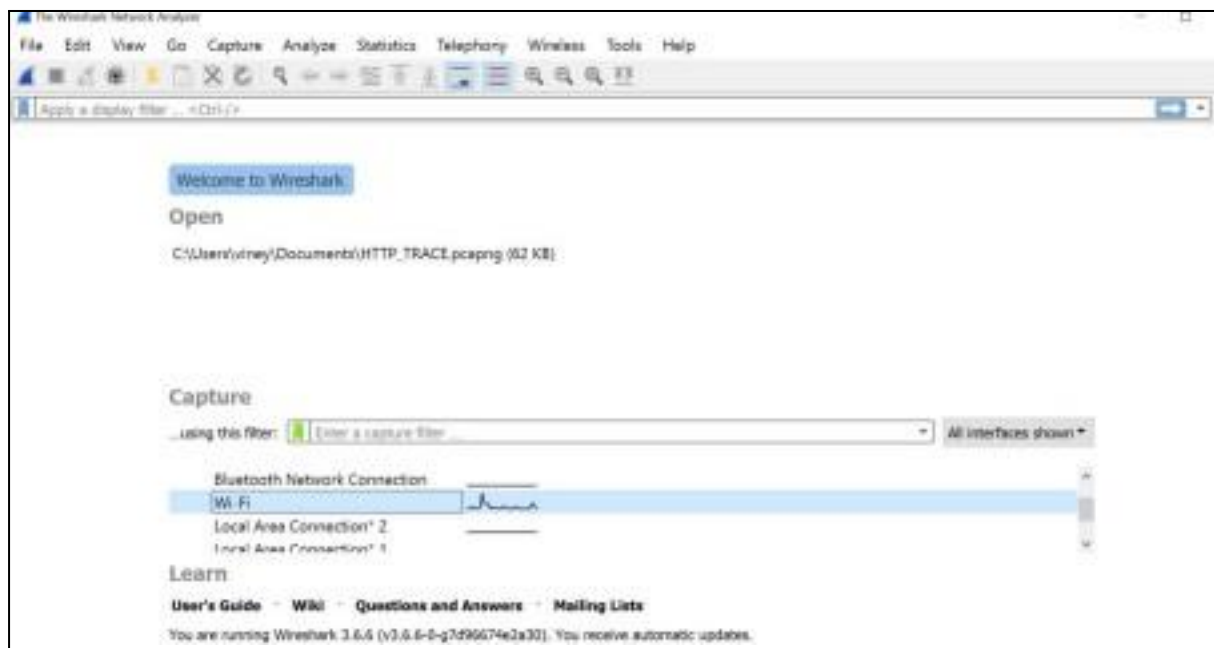
0000 ff ff ff ff ff ff 00 00 00 00 01 08 06 00 01
 0010 08 00 06 04 00 01 00 00 00 00 01 0a 0a 0a 01
 0020 ff ff ff ff ff ff 0a 0a 0a 02 00 00 00 00 00
 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00



11. Exercises for analyzing the network protocols using Wireshark · Capture the packets while browsing the any web site · Analyze the header fields of various protocols.

Analyse the network traffic using Wireshark

Step 1: Open Wireshark and select the network connection, which is Wifi.



Step 2: Start capturing packets, which will record the browsing history. And then stop capturing.

The screenshot shows the Wireshark interface with the 'Capture' menu highlighted. The packet list pane displays several DNS queries and TCP keep-alive messages. The packet details pane shows the structure of a DNS query (Standard query response) for the domain 'www.npr.org'. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
5936	108.776280893	8.8.8.8	10.0.2.15	DNS	216	Standard query response 0x98d3 Unknown (65) www.npr.org CNAME...
5937	108.776446590	10.0.2.15	8.8.8.8	DNS	98	Standard query 0xc8aa Unknown (65) e103193.dsca.akamaiedge.ne...
5938	108.780297566	8.8.8.8	10.0.2.15	DNS	162	Standard query response 0xc8aa Unknown (65) e103193.dsca.akam...
5939	108.780297578	8.8.8.8	10.0.2.15	DNS	336	Standard query response 0xfd94 Unknown (65) www.deccanherald...
5940	108.782220492	8.8.8.8	10.0.2.15	DNS	208	Standard query response 0x302c AAAA www.npr.org CNAME nprweb...
5941	109.076771491	10.0.2.15	34.107.221.82	TCP	54	[TCP Keep-Alive] 56590 → 80 [ACK] Seq=636 Ack=433 Win=64024 L...
5942	109.076811866	10.0.2.15	34.107.221.82	TCP	54	[TCP Keep-Alive] 48250 → 80 [ACK] Seq=301 Ack=299 Win=63942 L...
5943	109.077256994	34.107.221.82	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 56590 [ACK] Seq=433 Ack=637 Win=655...
5944	109.077257061	34.107.221.82	10.0.2.15	TCP	60	[TCP Keep-Alive ACK] 80 → 48250 [ACK] Seq=299 Ack=302 Win=655...

Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_98:81:d2 (08:00:27:98:81:d2), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8
 User Datagram Protocol, Src Port: 54381, Dst Port: 53
 Domain Name System (query)

0000 52 54 00 12 35 02 08 00 27 98 81 d2 08 00 45 00 RT: 5... '.....E:
 0010 00 47 45 2a 40 00 40 01 09 5d 0a 00 02 0f 08 08 -GE*0 @ [].....
 0020 08 08 d4 6d 00 35 00 33 1c 63 83 f9 01 00 00 01 ---m 5 3 -0.....
 0030 00 00 00 00 00 00 01 03 6e 74 70 06 75 62 75 6e 74n tp.ubunt
 0040 75 03 63 6f 6d 00 00 01 00 01 00 00 29 02 00 00 u:com... ..)
 0050 00 00 00 00 00

Activate Windows
Go to Settings to activate Windows.

wireshark_enp0s3_20250313135425_85QQXa.pcapng Packets: 5944 · Displayed: 5944 (100.0%) Profile: Default

Step 3: Search for all TCP protocol packets.

The screenshot shows the Wireshark interface with the 'tcp' filter applied. The packet list pane displays several TCP packets, including a SYN packet. The packet details pane shows the structure of a TCP packet (SYN). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
13	-2.395215385	10.0.2.15	34.160.144.191	TCP	74	40286 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 ..
14	-2.390649568	34.160.144.191	10.0.2.15	TCP	60	443 → 40286 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
15	-2.390591005	10.0.2.15	34.160.144.191	TCP	54	40286 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
16	-2.388823820	10.0.2.15	34.160.144.191	TLsv1.2	274	Client Hello
17	-2.387525959	34.160.144.191	10.0.2.15	TCP	60	443 → 40286 [ACK] Seq=1 Ack=221 Win=65535 Len=0
18	-2.381469797	34.160.144.191	10.0.2.15	TLsv1.2	2974	Server Hello, Certificate
19	-2.381458234	10.0.2.15	34.160.144.191	TCP	54	40286 → 443 [ACK] Seq=221 Ack=2921 Win=62780 Len=0
20	-2.381166247	34.160.144.191	10.0.2.15	TLsv1.2	143	Server Key Exchange, Server Hello Done
21	-2.381161403	10.0.2.15	34.160.144.191	TCP	54	40286 → 443 [ACK] Seq=221 Ack=3010 Win=62780 Len=0
22	-2.374932656	10.0.2.15	34.160.144.191	TLsv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake ...

Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_98:81:d2 (08:00:27:98:81:d2), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.160.144.191
 Transmission Control Protocol, Src Port: 40286, Dst Port: 443, Seq: 0, Len: 0

0000 52 54 00 12 35 02 08 00 27 98 81 d2 08 00 45 00 RT: 5... '.....E:
 0010 00 3c 75 f4 40 00 40 06 05 5a 0a 00 02 0f 22 a0 <u @ @ -Z.....
 0020 90 bf 9d 5e 01 bb b9 44 7c 25 00 00 00 00 a0 02 ...A...D [%.....
 0030 fa f0 bf 9c 00 00 02 04 05 b4 04 02 08 0a 91 bd
 0040 8c 8b 00 00 00 00 01 03 08 07

Activate Windows
Go to Settings to activate Windows.

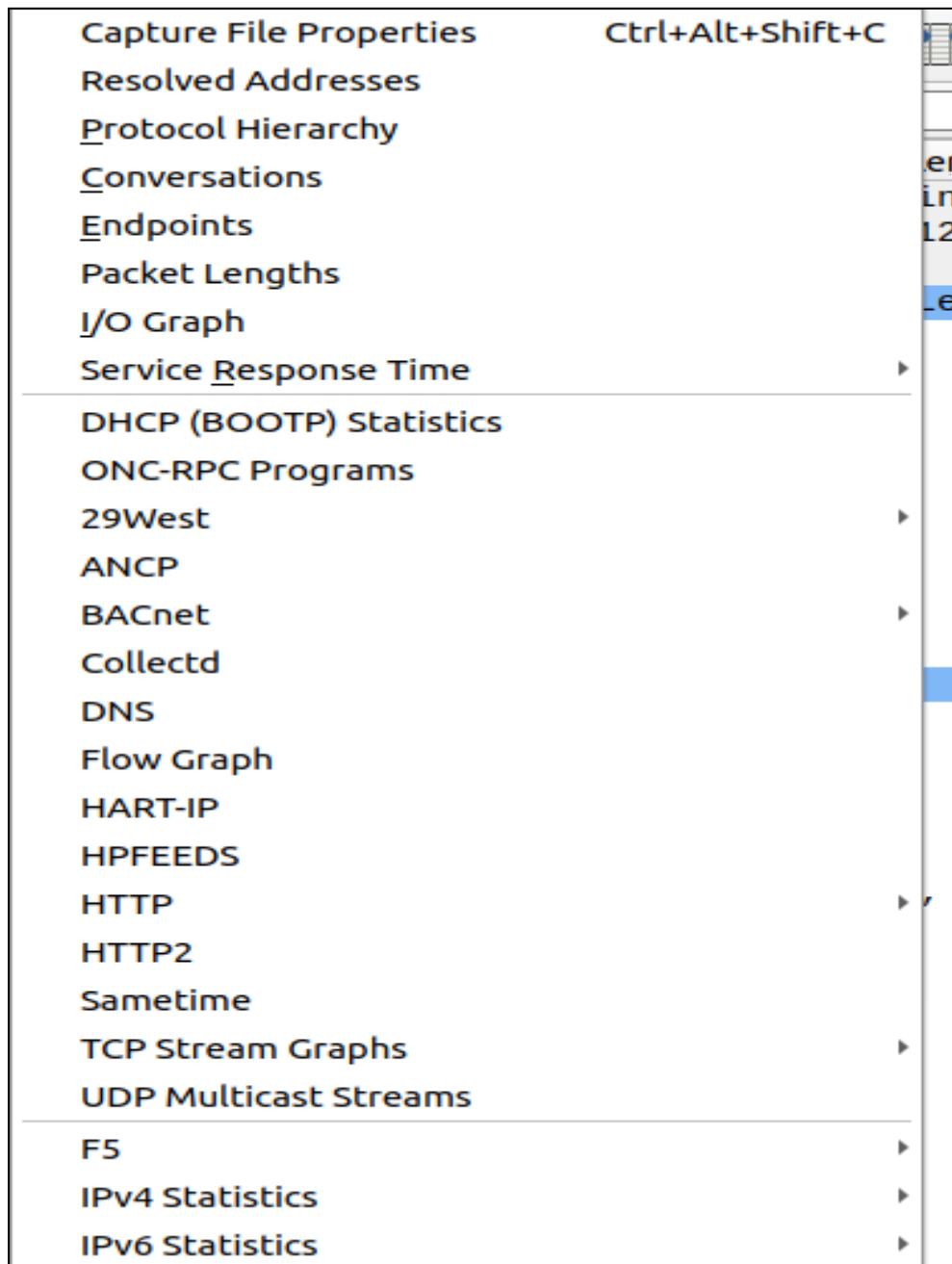
You can select any packets to get the information about that.

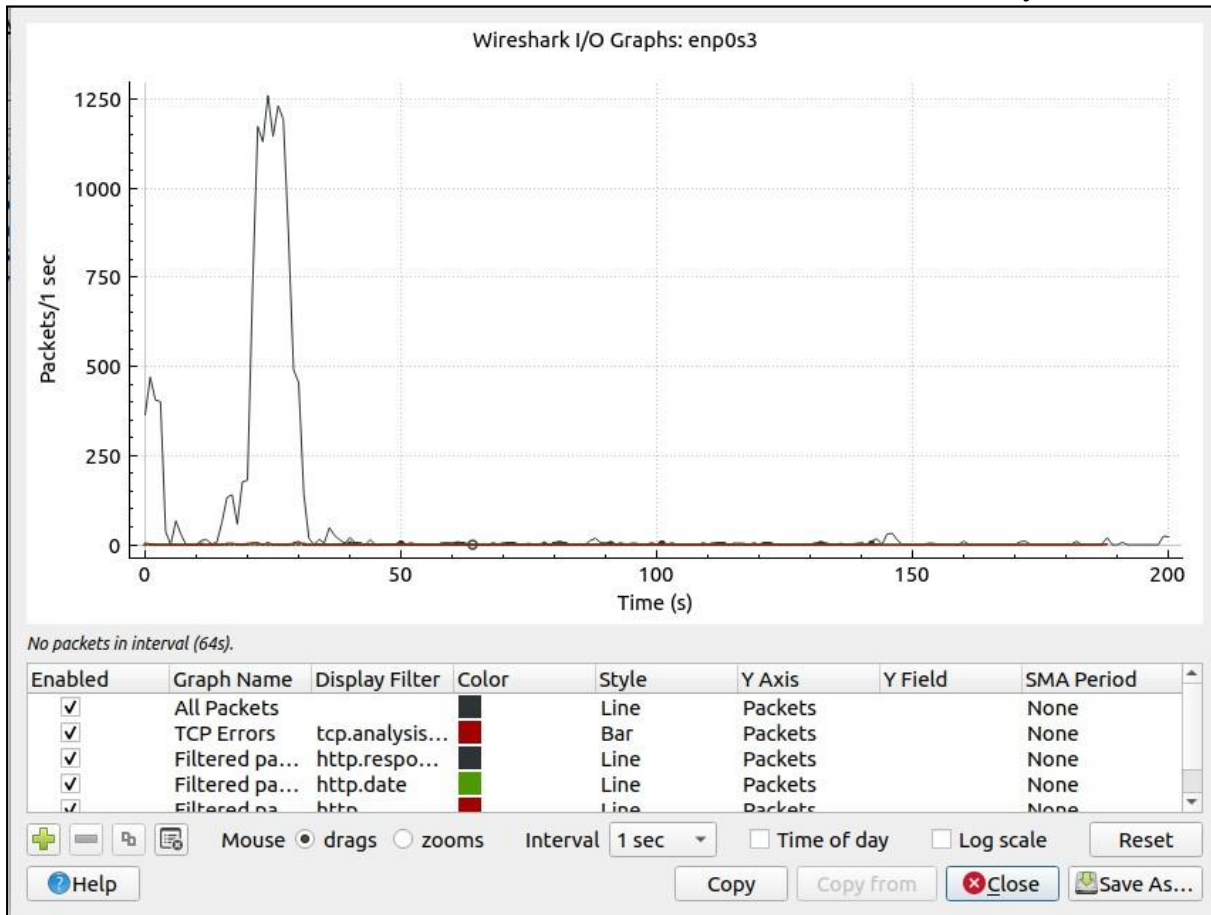
‣ Frame 13: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
‣ Ethernet II, Src: PcsCompu_98:81:d2 (08:00:27:98:81:d2), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
‣ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.160.144.191
‣ Transmission Control Protocol, Src Port: 40286, Dst Port: 443, Seq: 0, Len: 0

Then expand the Transmission Control Protocol column to get additional details.

‣ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.160.144.191
‣ Transmission Control Protocol, Src Port: 40286, Dst Port: 443, Seq: 0, Len: 0
Source Port: 40286
Destination Port: 443
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 3108273189
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 0
Acknowledgment number (raw): 0
1010 ... = Header Length: 40 bytes (10)
‣ Flags: 0x002 (SYN)
Window size value: 64240
[Calculated window size: 64240]
Checksum: 0xbf9c [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
‣ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
‣ [Timestamps]

You can analyze the statistics graphically. Click on the statistics option.





Step 4: Search for all HTTP protocol packets.

No.	Time	Source	Destination	Protocol	Length	Info
25	0.161335685	10.0.2.15	104.115.39.49	OCSP	493	Request
27	0.164853643	104.115.39.49	10.0.2.15	OCSP	943	Response
50	0.283661854	10.0.2.15	34.107.221.82	HTTP	355	GET /canonical.html HTTP/1.1
52	0.290749685	34.107.221.82	10.0.2.15	HTTP	352	HTTP/1.1 200 OK (text/html)
61	0.386101959	10.0.2.15	34.107.221.82	HTTP	372	GET /success.txt?ipv4 HTTP/1.1
67	0.397587645	34.107.221.82	10.0.2.15	HTTP	270	HTTP/1.1 200 OK (text/plain)

Step 5: Search for all HTTP request protocol packets with the GET method.

http.request.method==GET						
No.	Time	Source	Destination	Protocol	Length	Info
50	0.283661854	10.0.2.15	34.107.221.82	HTTP	355	GET /canonical.html HTTP/1.1
61	0.386101959	10.0.2.15	34.107.221.82	HTTP	372	GET /success.txt?ipv4 HTTP/1.1
1353	4.001235482	10.0.2.15	34.107.221.82	HTTP	372	GET /success.txt?ipv4 HTTP/1.1

Step 6: Search for all HTTP response protocol packets.

http.response						
No.	Time	Source	Destination	Protocol	Length	Info
27	0.164853643	104.115.39.49	10.0.2.15	OCSP	943	Response
52	0.290749685	34.107.221.82	10.0.2.15	HTTP	352	HTTP/1.1 200 OK (text/html)
67	0.397587645	34.107.221.82	10.0.2.15	HTTP	270	HTTP/1.1 200 OK (text/plain)
156	0.650575080	142.250.100.163	10.0.2.15	OCSP	1157	Response

12. Evaluate the network performance using metrics: throughput, delay, response time, packet loss, dropped packets etc. (All Topology).

Evaluating **network performance** involves analyzing several **key metrics** across all types of **topologies** (like star, bus, ring, mesh, etc.).

1. Throughput

Definition:

The amount of **data successfully transmitted** from source to destination in a given time, usually in **bps (bits per second)**.

High throughput = Good network performance.

How to measure:

- Use tools like iPerf, Wireshark, or real-time monitoring systems.
- Formula:
Throughput = (Total data received / Total time)

Factors affecting it:

- Bandwidth
- Network congestion
- Protocol overhead
- Packet loss

2. Delay (Latency)

Definition:

The time it takes for a packet to travel from source to destination.

Measured in: milliseconds (ms)

Types of delay:

- **Propagation delay:** Time to travel across the medium
- **Transmission delay:** Time to push bits onto the link
- **Processing delay:** Time to process the packet
- **Queuing delay:** Time waiting in queues

Lower delay = Better performance.

3. Response Time

Definition:

Time taken from the **user's request to the receipt of a response.**

Includes:

- Transmission time
- Server processing time
- Return time

Used for: Web services, applications, client-server models

Ideal: <100ms for web apps, <1s for interactive systems

4. Packet Loss

Definition:

The **percentage of packets that never reach** their destination.

Packet loss (%) = (Lost packets / Sent packets) × 100

Causes:

- Network congestion
- Faulty hardware
- Weak Wi-Fi signals

Low packet loss (ideally 0%) is desired.

5. Dropped Packets

Definition:

Packets that are intentionally **discarded** due to full buffers, congestion, or routing errors.

Difference from packet loss:

All dropped packets are lost, but not all lost packets are dropped intentionally.

Measured via:

- Router/switch logs
- Monitoring tools (e.g., SNMP, Wireshark)

Example Evaluation Table (All Topologies)

Metric	Star Topology	Mesh Topology	Ring Topology	Bus Topology
Throughput	High (centralized)	Very High (redundant links)	Moderate	Moderate (shared medium)
Delay	Low	Low (short paths)	Medium (one-way path)	High under load
Response Time	Fast	Fast	Slower with distance	Slower with collisions
Packet Loss	Low	Very Low (backup links)	Medium	High during collisions
Dropped Packets	Low	Very Low	Moderate	High under load

Tools You Can Use:

- **iPerf / iPerf3** (Throughput)
- **Ping** (Delay, packet loss)
- **Traceroute** (Delay across hops)
- **Wireshark** (Detailed analysis)
- **NetFlow / SNMP** (Enterprise monitoring)
- **Network Simulator (NS2/NS3, Cisco Packet Tracer)** for topology-based simulations