# AIM: Program to simulate UDP server client.

**THEORY:**

**Understanding UDP server client:**

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of sender which the server uses to send data to the correct client.

In UDP, the client does not form a connection with the server like in TCP and instead, It just sends a datagram. Similarly, the server need not to accept a connection and just waits for datagrams to arrive. We can call a function called **connect()** in UDP but it does not result anything like it does in TCP. There is no 3-way handshake. It just checks for any immediate errors and store the peer's IP address and port number. connect() is storing peers address so no need to pass **server address** and **server address length** arguments in **send to()**. The entire process can be broken down into following steps **UDP Server:**

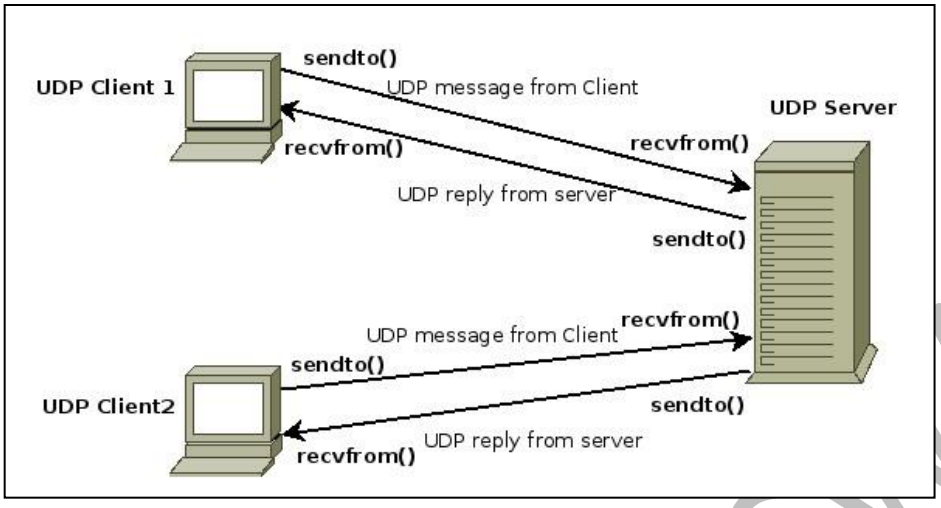
1. Create UDP socket.
2. Bind the socket to server address.
3. Wait until datagram packet arrives from client.
4. Process the datagram packet and send a reply to client.
5. Go back to Step 3.

**UDP Client:**

1. Create UDP socket.
2. Send message to server.
3. Wait until response from server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

**UDP Overview:**

UDP is the abbreviation of User Datagram Protocol. UDP makes use of Internet Protocol of the TCP/IP suit. In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.




UDP the InterSystems IRIS User Datagram Protocol (UDP) binding. Provides two-way message transfer between a server and a large number of clients. UDP is not connection-based; each data packet transmission is an independent event. Provides fast and lightweight data transmission for local packet broadcasts and remote multicasting. Inherently less reliable than TCP. Does not provide message acknowledgement.

**Properties of UDP:**

- The UDP does not provide guaranteed delivery of message packets. If for some issue in a network if a packet is lost it could be lost forever.

Since there is no guarantee of assured delivery of messages, UDP is considered an unreliable protocol.

- The underlying mechanisms that implement UDP involve no connection-based communication. There is no streaming of data between a UDP server or and an UDP Client.
- An UDP client can send "n" number of distinct packets to an UDP server and it could also receive "n" number of distinct packets as replies from the UDP server.
- Since UDP is connectionless protocol the overhead involved in UDP is less compared to a connection based protocol like TCP.

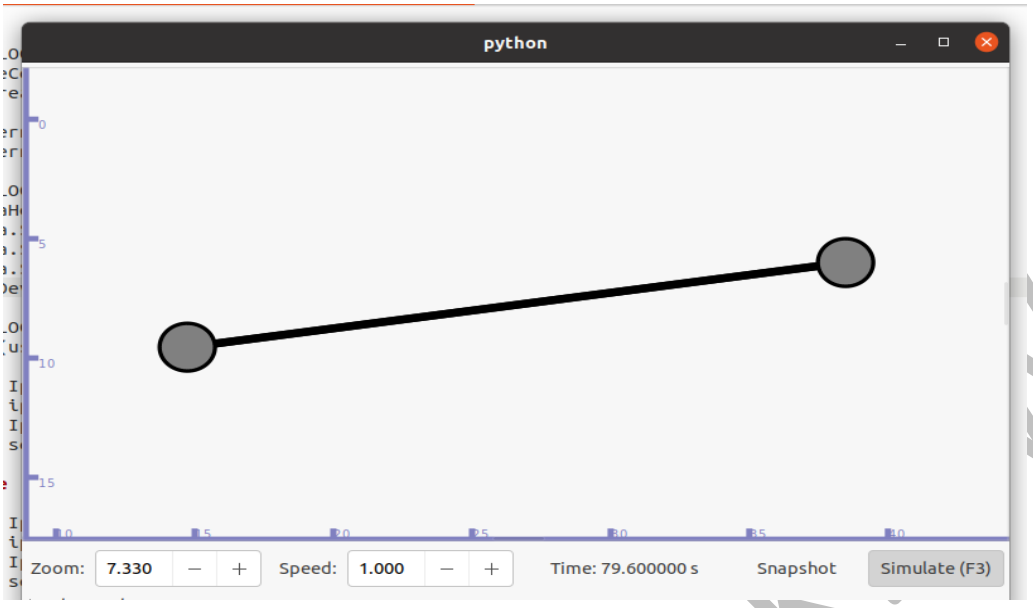
**SOURCE CODE:**

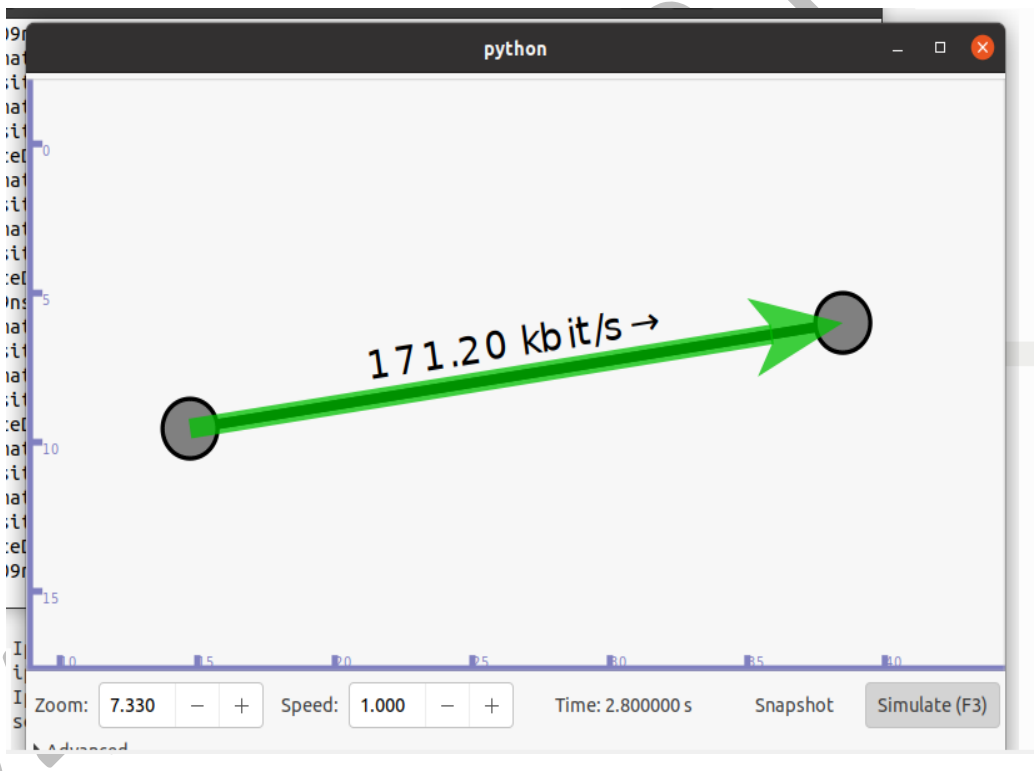
```
// Network topology
//
//       n0   n1
//       |    |
//       =======
//         LAN (CSMA)
//
// - UDP flow from n0 to n1 of 1024 byte packets at intervals of 50 ms
//   - maximum of 320 packets sent (or limited by simulation duration)
//   - option to use IPv4 or IPv6 addressing
```

```
//   - option to disable logging statements

#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
 #include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

int main (int argc, char *argv[])
{
 // Declare variables used in command-line arguments
 bool useV6 = false;
 bool logging = true;
 Address serverAddress;

 CommandLine cmd (__FILE__);
 cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
 cmd.AddValue ("logging", "Enable logging", logging);
 cmd.Parse (argc, argv);

 if (logging)
  {
   LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
   LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
  }

 NS_LOG_INFO ("Create nodes in above topology.");
 NodeContainer n;
 n.Create (2);

 InternetStackHelper internet;
 internet.Install (n);

 NS_LOG_INFO ("Create channel between the two nodes.");
 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)
  {
```

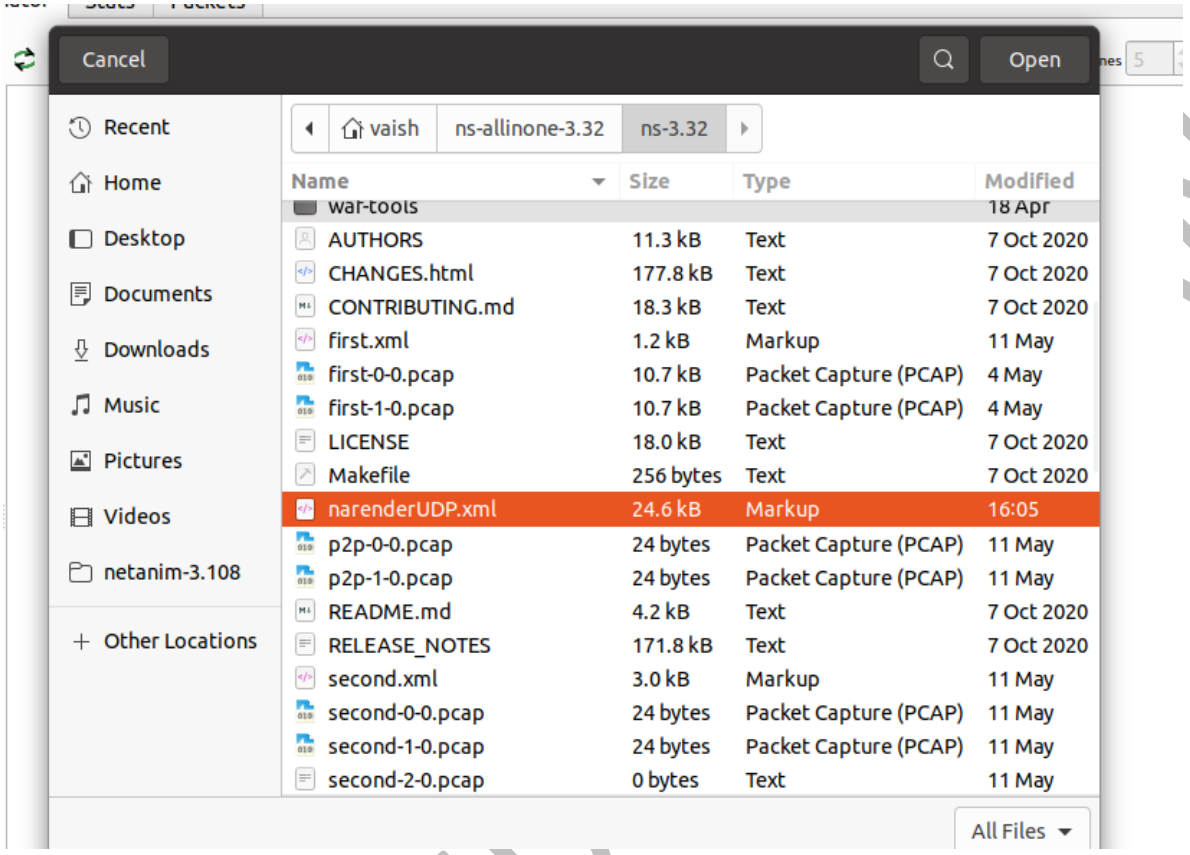
```
    Ipv4AddressHelper ipv4;
    ipv4.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer i = ipv4.Assign (d);
    serverAddress = Address (i.GetAddress (1));
   }
  else
   {
    Ipv6AddressHelper ipv6;
    ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
    Ipv6InterfaceContainer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

  NS_LOG_INFO ("Create UdpServer application on node 1.");
  uint16_t port = 4000;
  UdpServerHelper server (port);
  ApplicationContainer apps = server.Install (n.Get (1));
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (10.0));

  NS_LOG_INFO ("Create UdpClient application on node 0 to send to node 1.");
  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));

  csma.EnablePcapAll ("udp");

  AnimationInterface anim("narenderUDP.xml");
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

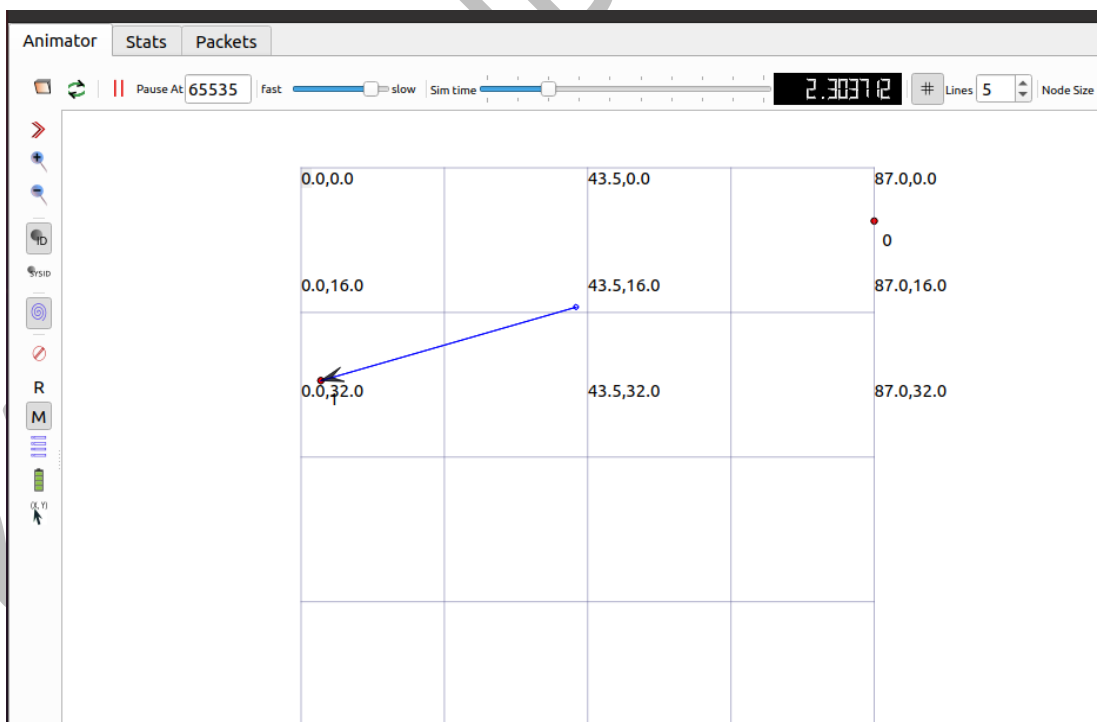
**OUTPUT:**


python
Zoom: 7.330
Speed: 1.000
Time: 79.600000 s
Snapshot
Simulate (F3)

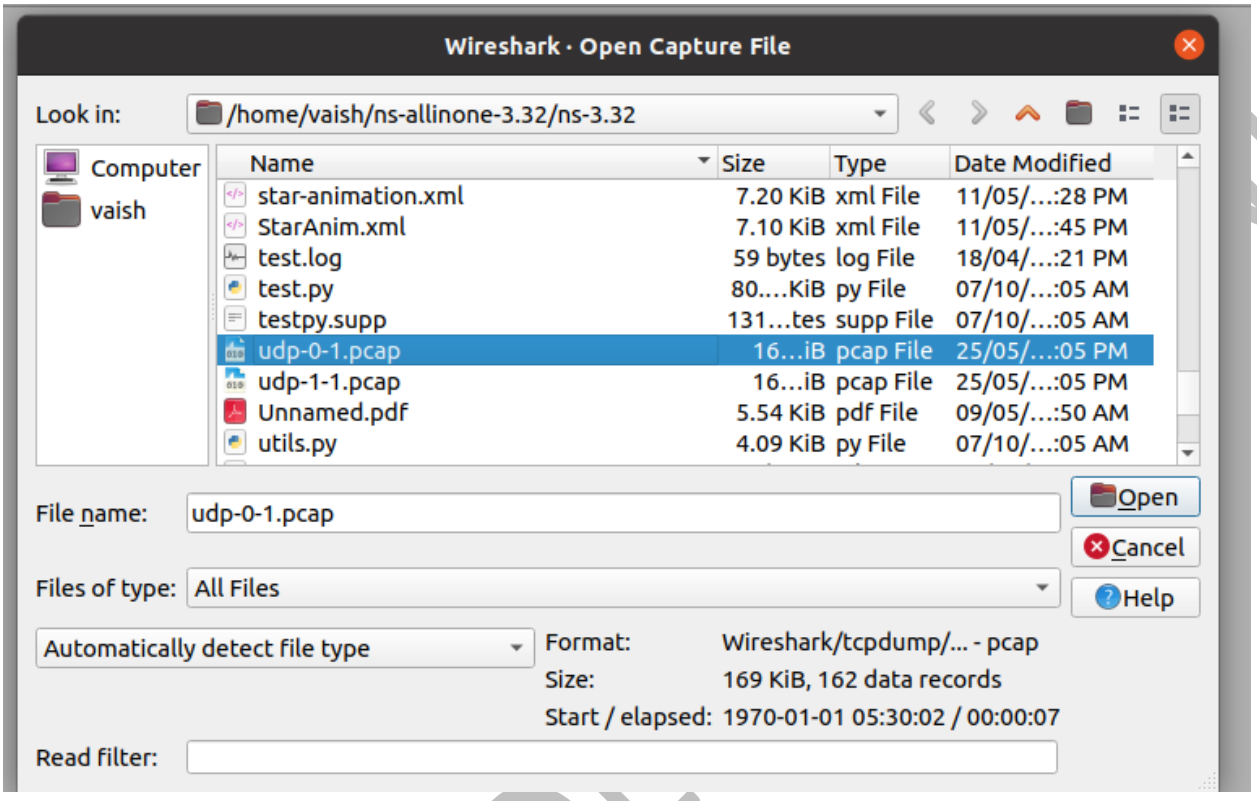

python
171.20 kbit/s →
Zoom: 7.330
Speed: 1.000
Time: 2.800000 s
Snapshot
Simulate (F3)

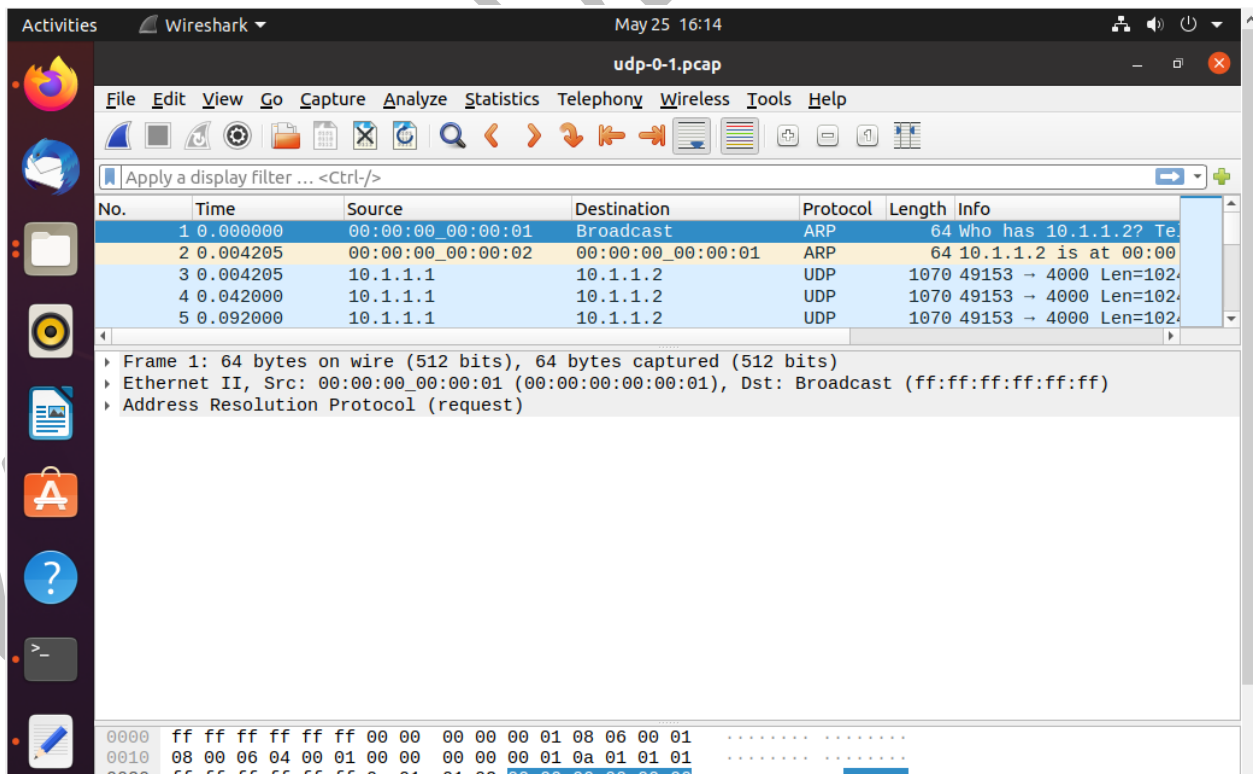
**NetAnim:**

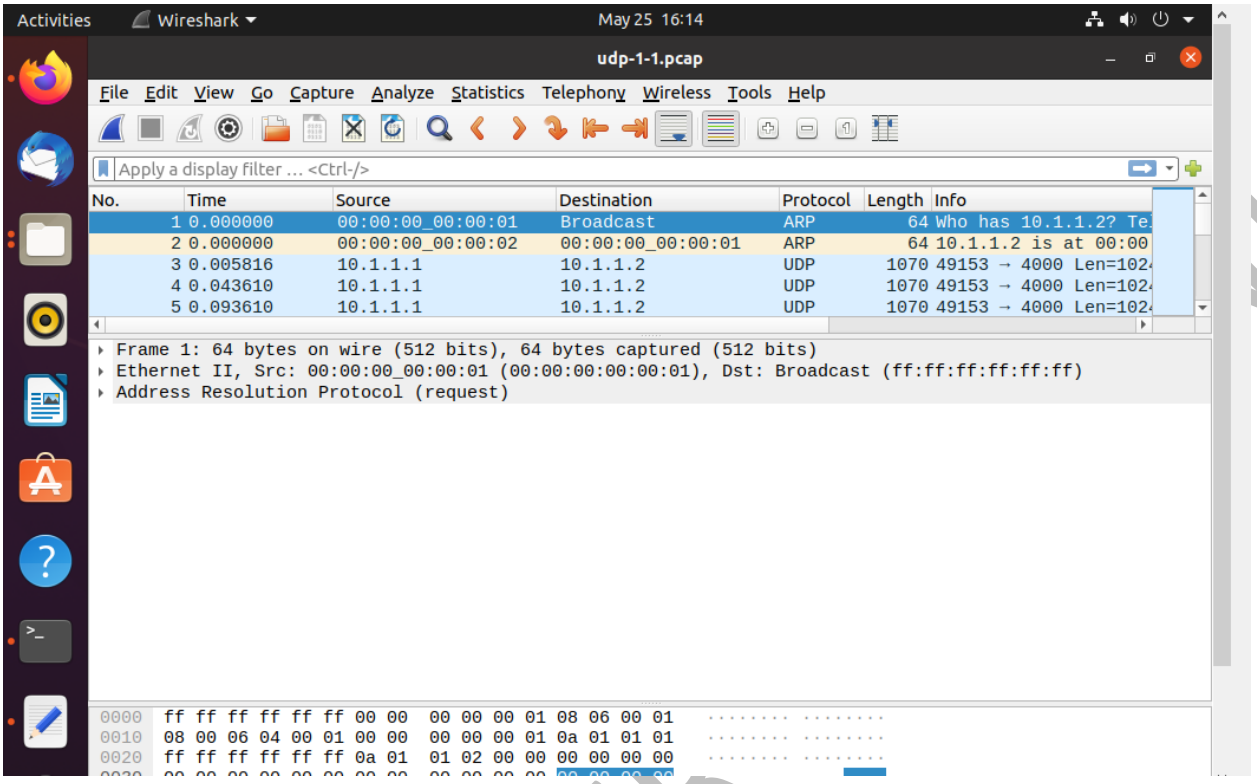

Cancel
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
vaish
ns-allinone-3.32
ns-3.32
Name
Size
Type
Modified
AUTHORS 11.3 kB Text 7 Oct 2020
CHANGES.html 177.8 kB Text 7 Oct 2020
CONTRIBUTING.md 18.3 kB Text 7 Oct 2020
first.xml 1.2 kB Markup 11 May
first-0-0.pcap 10.7 kB Packet Capture (PCAP) 4 May
first-1-0.pcap 10.7 kB Packet Capture (PCAP) 4 May
LICENSE 18.0 kB Text 7 Oct 2020
Makefile 256 bytes Text 7 Oct 2020
narenderUDP.xml 24.6 kB Markup 16:05
p2p-0-0.pcap 24 bytes Packet Capture (PCAP) 11 May
p2p-1-0.pcap 24 bytes Packet Capture (PCAP) 11 May
README.md 4.2 kB Text 7 Oct 2020
RELEASE_NOTES 171.8 kB Text 7 Oct 2020
second.xml 3.0 kB Markup 11 May
second-0-0.pcap 24 bytes Packet Capture (PCAP) 11 May
second-1-0.pcap 24 bytes Packet Capture (PCAP) 11 May
second-2-0.pcap 0 bytes Text 11 May
All Files


Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.303712
Lines 5
Node Size
0.0,0.0
43.5,0.0
87.0,0.0
0
0.0,16.0
43.5,16.0
87.0,16.0
0.0,32.0
43.5,32.0
87.0,32.0

**WIRESHARK:**


Wireshark · Open Capture File
Look in: /home/vaish/ns-allinone-3.32/ns-3.32
Computer
vaish
Name Size Type Date Modified
star-animation.xml 7.20 KiB xml File 11/05/...:28 PM
StarAnim.xml 7.10 KiB xml File 11/05/...:45 PM
test.log 59 bytes log File 18/04/...:21 PM
test.py 80....KiB py File 07/10/...:05 AM
testpy.supp 131...tes supp File 07/10/...:05 AM
udp-0-1.pcap 16...iB pcap File 25/05/...:05 PM
udp-1-1.pcap 16...iB pcap File 25/05/...:05 PM
Unnamed.pdf 5.54 KiB pdf File 09/05/...:50 AM
utils.py 4.09 KiB py File 07/10/...:05 AM
File name: udp-0-1.pcap
Open
Cancel
Files of type: All Files
Help
Automatically detect file type
Format: Wireshark/tcpdump/... - pcap
Size: 169 KiB, 162 data records
Start / elapsed: 1970-01-01 05:30:02 / 00:00:07
Read filter:


Activities
Wireshark
May 25 16:14
udp-0-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.2? Te
2 0.004205 00:00:00_00:00:02 00:00:00_00:00:01 ARP 64 10.1.1.2 is at 00:00
3 0.004205 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
4 0.042000 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
5 0.092000 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
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 00 01 08 06 00 01
0010 08 00 06 04 00 01 00 00 00 00 00 01 0a 01 01 01




**CONCLUSION:**

From this practical, I have successfully implemented the simulation of UDP server client.