AIM: PROGRAM TO SIMULATE FTP USING TCP PROTOCOL

THEORY:

File Transfer Protocol (FTP):

File transfer protocol (FTP) is a set of rules that computers follow for the transferring of files from one system to another over the internet. It may be used by a business to transfer files from one computer system to another, or websites may use FTP to upload or download files from a website's server.

- File transfer protocol (FTP) is a way to download, upload, and transfer files from one location to another on the internet and between computer systems.
- File transfer protocol (FTP) enables computers on the internet to transfer files back and forth, and is an essential tool for those building and maintaining websites today.
- Many file transfer protocol (FTP) clients are available for free to download, although most websites (and web browsers) that offer downloads already have the FTP built-in, so downloading a separate piece of software isn't always required.

Understanding File Transfer Protocol (FTP):

File transfer protocol is one of many different protocols that dictate how computers behave on the internet. Other such protocols include the Hypertext Transfer Protocol (HTTP), the Internet Message Access Protocol (IMAP), and the Network Time Protocol (NTP). FTP enables computers on the internet to transfer files back and forth, and is an essential tool for those building and maintaining websites today.

In order to use FTP, a user must first download an FTP client (or access an FTP client through a web browser). A client is the software that will allow you to transfer files.

Most web browsers come with FTP clients—possibly via a downloadable extension—that enable users to transfer files from their computer to a server and vice versa. Some users may want to use a third-party FTP client because many of them offer extra features to improve your experience. Examples of FTP clients that are free to download include FileZilla Client, FTP Voyager, WinSCP, CoffeeCup Free FTP, and Core FTP.

Many people have used FTP before without even noticing it. If you have ever downloaded a file from a web page, chances are that you used FTP in the process. The first step for accessing an FTP server to download a file is to log in, which may occur automatically or by manually inputting a username and password. FTP will also require you to access an FTP server through a specific port number.

Once you have accessed the FTP server through your FTP client, you can now transfer files. Not all public FTP servers require you to sign in because some servers enable you to access them anonymously.

Depending on the FTP client you use, there will be different features available that allow you to modify the manner in which you upload and download files. For instance, if you use the free FTP client FileZilla, the program will enable you to set bandwidth limits for files, enabling you to control the speed at which you download or upload files. This can be helpful if you are managing multiple file transfers at once. Other features you may want to look for in an FTP client include public key authentication, the ability to set file compression levels, or tools that enable you to search a server using file masks.

TCP/IP Protocol:

TCP/IP stands for Transmission Control Protocol/Internet Protocol and is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP is also used as a communications protocol in a private computer network (an intranet or extranet). The entire IP suite -- a set of rules and procedures -- is commonly referred to as TCP/IP. TCP and IP are the two main protocols, though others are included in the suite. The TCP/IP protocol suite functions as an abstraction layer between internet applications and the routing and switching fabric.

TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination. TCP/IP requires little central management and is designed to make networks reliable with the ability to recover automatically from the failure of any device on the network.

The two main protocols in the IP suite serve specific functions. TCP defines how applications can create channels of communication across a network. It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.

IP defines how to address and route each packet to make sure it reaches the right destination. Each gateway computer on the network checks this IP address to determine where to forward the message.

A subnet mask tells a computer, or other network device, what portion of the IP address is used to represent the network and what part is used to represent hosts, or other computers, on the network.

Network address translation (NAT) is the virtualization of IP addresses. NAT helps improve security and decrease the number of IP addresses an organization needs.

How does TCP/IP work?

TCP/IP uses the client-server model of communication in which a user or machine (a client) is provided a service, like sending a webpage, by another computer (a server) in the network.

Collectively, the TCP/IP suite of protocols is classified as stateless, which means each client request is considered new because it is unrelated to previous requests. Being stateless frees up network paths so they can be used continuously.

The transport layer itself, however, is stateful. It transmits a single message, and its connection remains in place until all the packets in a message have been received and reassembled at the destination.

The TCP/IP model differs slightly from the seven-layer Open Systems Interconnection (OSI) networking model designed after it. The OSI reference model defines how applications can communicate over a network.

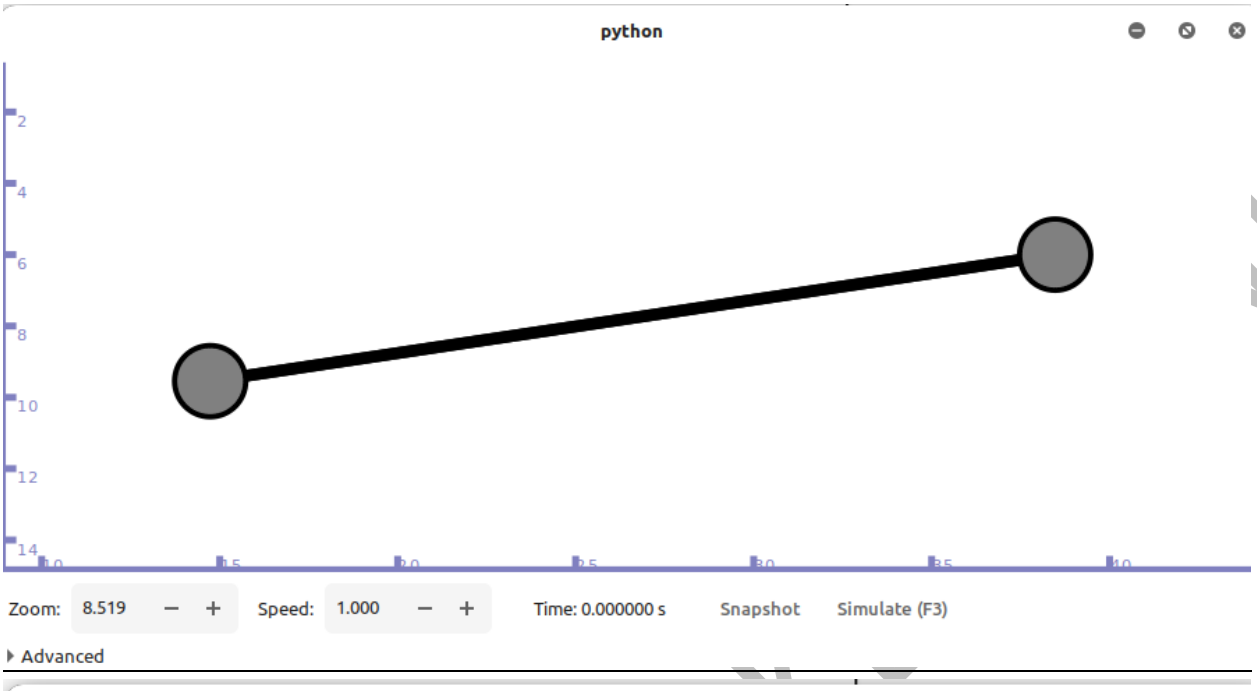
SOURCE CODE:

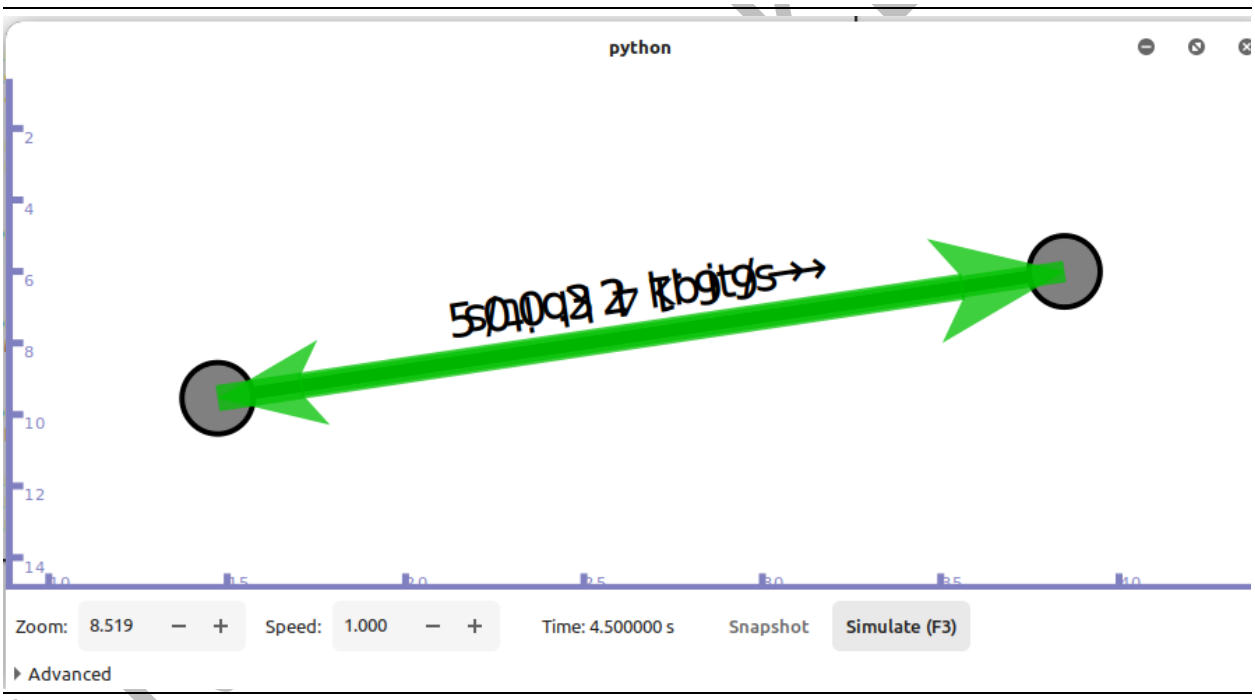
```
// Network topology
//
//       n0 ---------- n1
//            500 Kbps
//             5 ms
//
// - Flow from n0 to n1 using BulkSendApplication.
// - Tracing of queues and packet receptions to file "tcp-bulk-send.tr"
//   and pcap tracing available when tracing is turned on.

#include <string>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/packet-sink.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpBulkSendExample");

Int main (int argc, char *argv[])
{
  bool tracing = true;
  uint32_t maxBytes = 0;

// Allow the user to override any of the defaults at
// run-time, via command-line arguments

  CommandLine cmd (__FILE__);
  cmd.AddValue ("tracing", "Flag to enable/disable tracing", tracing);
  cmd.AddValue ("maxBytes",
                "Total number of bytes for application to send", maxBytes);
  cmd.Parse (argc, argv);

// Explicitly create the nodes required by the topology (shown above).

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

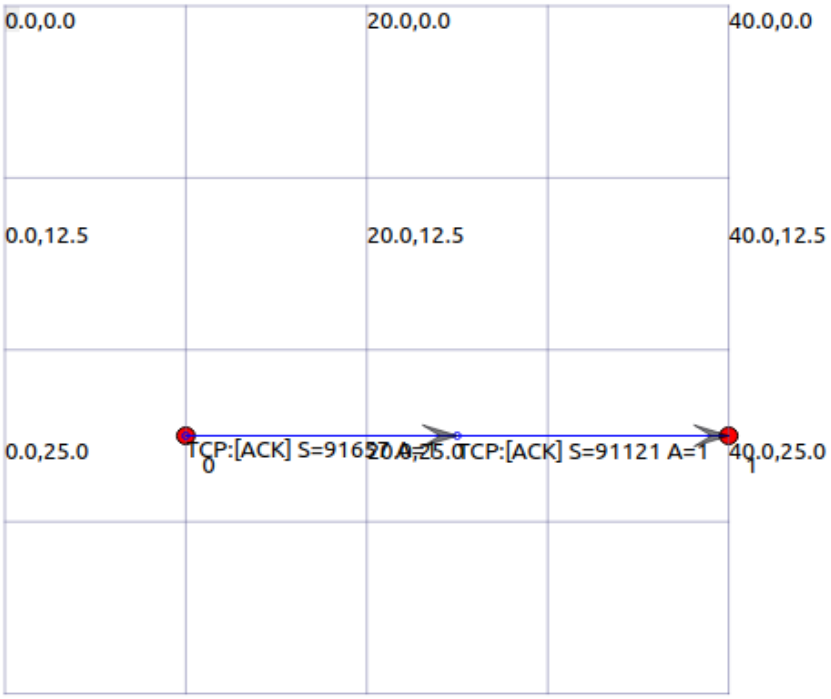
```
  NodeContainer nodes;
  nodes.Create (2);

  NS_LOG_INFO ("Create channels.");

// Explicitly create the point-to-point link required by the topology (shown above).

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

// Install the internet stack on the nodes

  InternetStackHelper internet;
  internet.Install (nodes);

// We've got the "hardware" in place.  Now we need to add IP addresses.

  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  NS_LOG_INFO ("Create Applications.");

// Create a BulkSendApplication and install it on node 0

  uint16_t port = 9;  // well-known echo port number

  BulkSendHelper source ("ns3::TcpSocketFactory",
                         InetSocketAddress (i.GetAddress (1), port));
  // Set the amount of data to send in bytes.  Zero is unlimited.
  source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
  ApplicationContainer sourceApps = source.Install (nodes.Get (0));
  sourceApps.Start (Seconds (0.0));
  sourceApps.Stop (Seconds (10.0));

// Create a PacketSinkApplication and install4 it on node 1
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
                         InetSocketAddress (Ipv4Address::GetAny (), port));
  ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
```

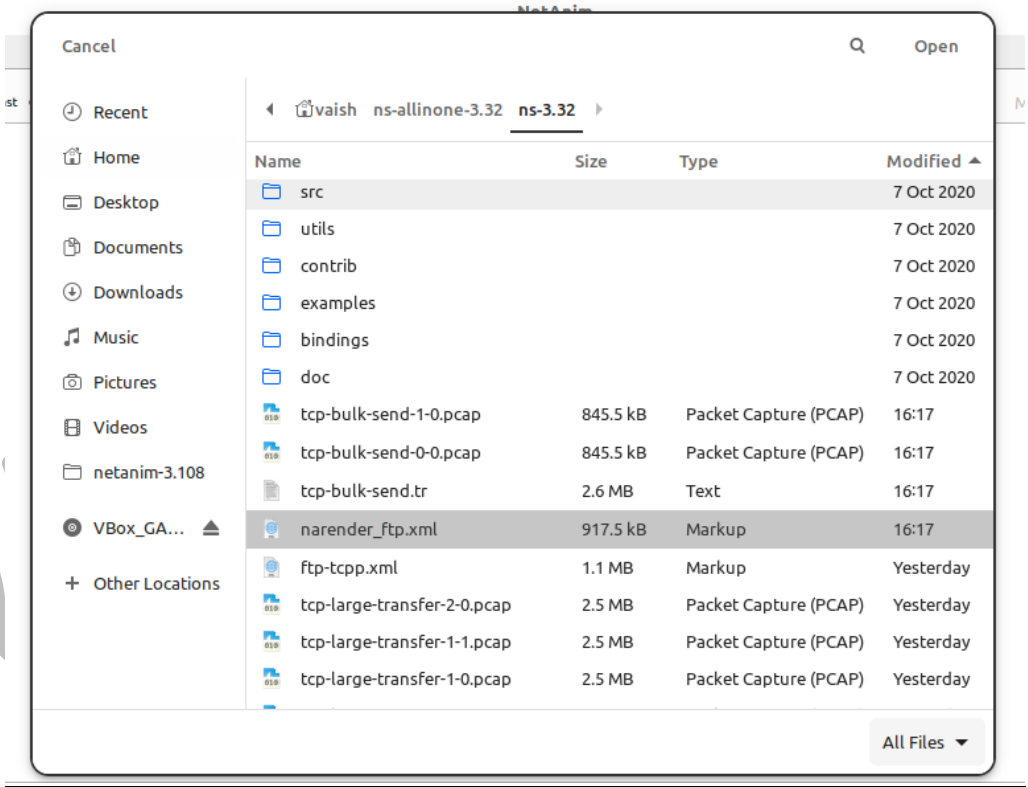
```
  sinkApps.Start (Seconds (0.0));
  sinkApps.Stop (Seconds (10.0));

// Set up tracing if enabled

MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);

AnimationInterface anim("narender_ftp.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition(nodes.Get(1),40,25);
  anim.EnablePacketMetadata(true);

  if (tracing)
    {
      AsciiTraceHelper ascii;
      pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("tcp-bulk-send.tr"));
      pointToPoint.EnablePcapAll ("tcp-bulk-send", false);
    }

// Now, do the actual simulation.

  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds (10.0));
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

  Ptr<PacketSink> sink1 = DynamicCast<PacketSink> (sinkApps.Get (0));
  std::cout << "Total Bytes Received: " << sink1->GetTotalRx () << std::endl;
}
```

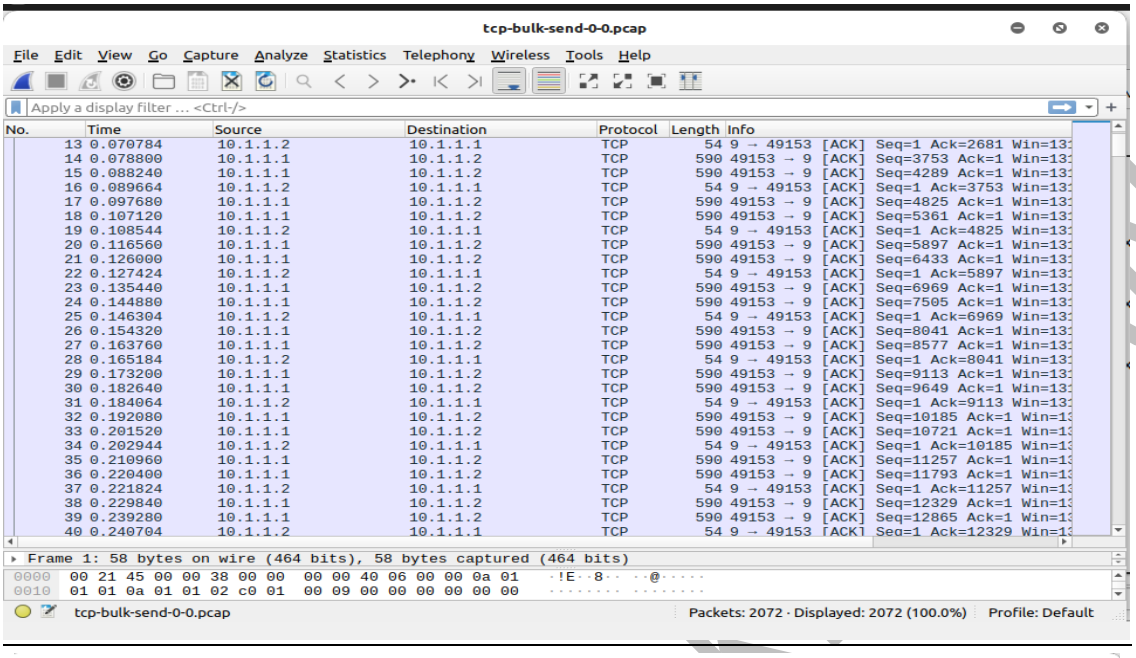
OUTPUT:

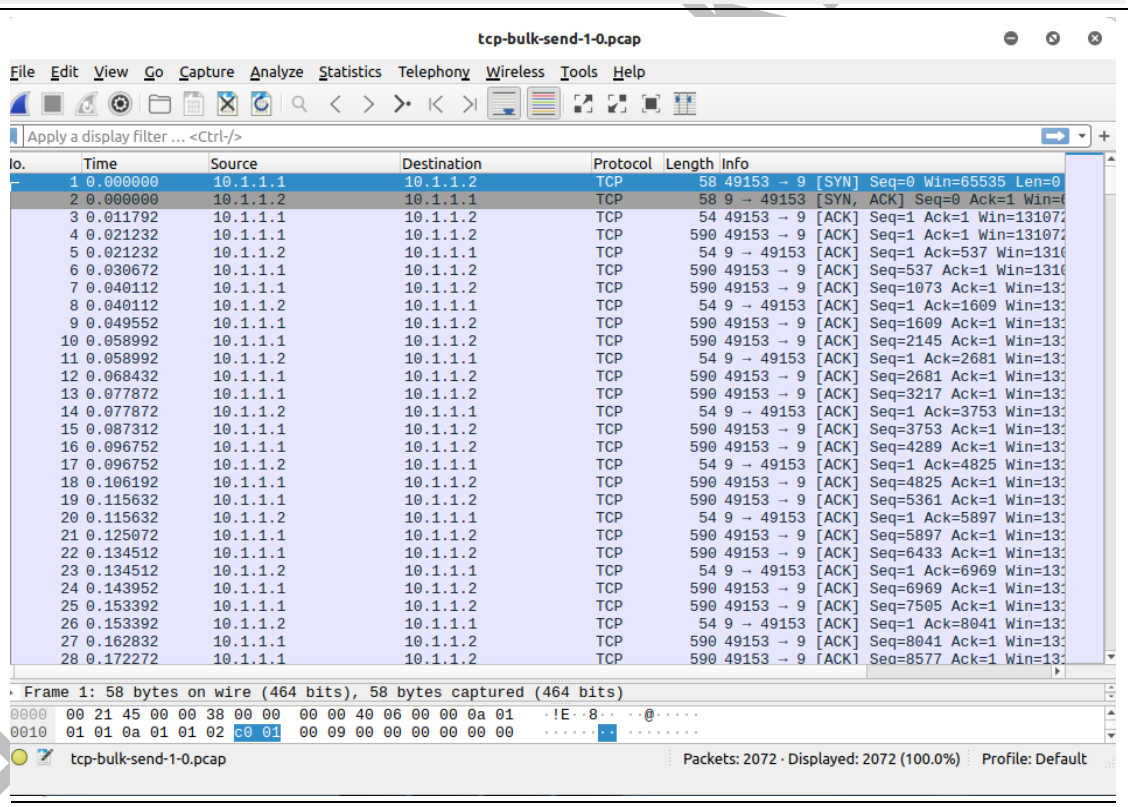

python
2
4
6
8
10
12
14
Zoom: 8.519
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced


python
2
4
6
8
10
12
14
Zoom: 8.519
Speed: 1.000
Time: 4.500000 s
Snapshot
Simulate (F3)
Advanced


0.0,0.0
20.0,0.0
40.0,0.0
0.0,12.5
20.0,12.5
40.0,12.5
0.0,25.0
TCP:[ACK] S=91121 A=1
40.0,25.0
0
1


Cancel
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
VBox_GA...
Other Locations
vaish
ns-allinone-3.32
ns-3.32
Name
Size
Type
Modified
src 7 Oct 2020
utils 7 Oct 2020
contrib 7 Oct 2020
examples 7 Oct 2020
bindings 7 Oct 2020
doc 7 Oct 2020
tcp-bulk-send-1-0.pcap 845.5 kB Packet Capture (PCAP) 16:17
tcp-bulk-send-0-0.pcap 845.5 kB Packet Capture (PCAP) 16:17
tcp-bulk-send.tr 2.6 MB Text 16:17
narender_ftp.xml 917.5 kB Markup 16:17
ftp-tcpp.xml 1.1 MB Markup Yesterday
tcp-large-transfer-2-0.pcap 2.5 MB Packet Capture (PCAP) Yesterday
tcp-large-transfer-1-1.pcap 2.5 MB Packet Capture (PCAP) Yesterday
tcp-large-transfer-1-0.pcap 2.5 MB Packet Capture (PCAP) Yesterday
All Files





CONCLUSION:

From this practical, I have learned about ftp bulk transfer in ns3.