CS 410 Networks and Networking
October 19th, 2012 **Name:** _____

Lab 5

Introduction

In this lab, you will explore the operation of the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), the two transport protocols of the Internet protocol architecture.

UDP is a simple protocol for exchanging messages from a sending application to a receiving application. UDP adds a small header to the message, and the resulting data unit is called a UDP datagram. When a UDP datagram is transmitted, the datagram is encapsulated in an IP header and delivered to its destination. There is one UDP datagram for each application message.

The operation of TCP is more complex. First, TCP is a connection-oriented protocol, in which a TCP client establishes a logical connection to a TCP server before data transmission can take place. Once a connection is established, data transfer can proceed in both directions. The data unit of TCP, called a TCP segment, consists of a TCP header and payload that contains application data. A sending application submits data to TCP as a single stream of bytes without indicating message boundaries in the byte stream. The TCP sender decides how many bytes are put into a segment.

TCP ensures reliable delivery of data, uses checksums, sequence numbers, acknowledgments, and timers to detect damaged or lost segments. The TCP receiver acknowledges the receipt of data by sending an acknowledgment segment (ACK). Multiple TCP segments can be acknowledged in a single ACK. When a TCP sender does not receive an ACK, the data is assumed lost and is retransmitted.

The lab covers the main features of UDP and TCP. Parts 1 and 2 compare the performance of data transmissions in TCP and UDP. Part 3 explores how TCP and UDP deal with IP fragmentation. Part 4 explores connection management. We won't be able to explore flow control, acknowledgments, retransmissions and congestion control because that requires expensive Cisco core routers.

Part 1: Learning how to use nTTCP

The nttcp command is a Linux tool used to generate synthetic UDP and TCP traffic loads. Together with ping and traceroute, nttcp is an essential utility program for debugging problems in IP networks. Running the nttcp tool consists of setting up a nttcp receiver on one host and then a nttcp sender on another host. Once the nttcp sender is started, it blasts the specified amount of data as fast as possible to the nttcp receiver.

A nttcp receiver process is started with the command
`nttcp -r [-lbuflen] [-nnumbufs] [-pport] [-u]`

A nttcp sender process is started with the command
`nttcp -t [-lblocksize] [-nnumblock] [-pport] [-u] [-D] IPaddress`

The options of the command are:

- `-t` Specifies the transmit mode (client/transmitter)
- `-i` Make nttcp wait for connections (server/receiver)
- `-u` Specifies to use UDP instead of TCP. By default, nttcp uses TCP to send data
- `-nnumblock` Number of data blocks to be transmitted (default value is 2048 blocks).
- `-lblocksize` Size of the data blocks that are passed to UDP or TCP in bytes (default is 4096 bytes). When UDP is used, this value is the number of data bytes in UDP datagram.
- `-T` print a title line at the output screen when finishing the transfer
- `-D` Disables buffering of data in TCP and forces immediate transmission of the data at the nttcp sender. Used only in the context of TCP.
- `-pport` Port number to send to or listen on. The port number must be identical at the sender and at the receiver. The default value is 5000.
- `IPaddress` IP address of the nttcp receiver to send data to.

By default, nttcp transmits data over a TCP connection. The nttcp sender opens a TCP connection to a nttcp receiver, transmits data, and then closes the connection. The nttcp receiver must be running when the nttcp sender is started. UDP data transfer is specified with the -u option. Since UDP is a connectionless protocol, the nttcp sender starts immediately sending UDP datagrams, regardless of whether a nttcp receiver is established.

Network Setup

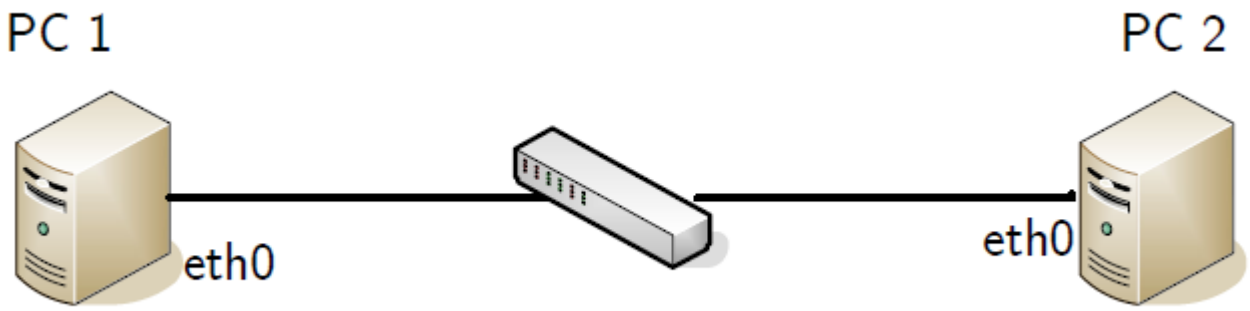

Figure 1: Figure 1

- Reboot the PCs and set up the network topology as show in Figure 1. Configure the IP addresses of the interfaces as given in Table 1.

Linux PC	Ethernet Interface eth0	Ethernet Interface eth1
PC1	10.0.1.11/24	10.0.5.11/24
PC2	10.0.2.22/24	10.0.5.22/24
PC3	10.0.1.33/24	10.0.2.33/24

- PC1 and PC2 are set up as hosts, and IP forwarding should be disabled. On PC1, this is done with the command

  `PC1% echo ''0'' >/proc/sys/net/ipv4/ip_forward`

- PC3 is set up as an IP router. Enable IP forwarding on PC3 with the command

  `PC3% echo ''1'' >/proc/sys/net/ipv4/ip_forward`

- Add default routes to the routing tables of PC1 and PC2, so that PC3 is the default gateway. For PC1 the command is as follows:

  `PC1% route add default gw 10.0.1.33`

- Verify that the setup is correct by issuing a ping command from PC1 and PC2 over both paths:

  `PC1% ping 10.0.2.22`

  `PC1% ping 10.0.5.22`

Part I: Transmitting Data with UDP

- On PC1, start wireshark to capture packets on interface eth0 between PC1 and PC2.
- On PC2, start a nttcp receiver that receives UDP traffic with the following command:

  `PC2% nttcp -i -l1024 -n10 -p4444 -u`
- On PC 1, start a nttcp sender that transmits UDP traffic by typing

  `PC1% nttcp t -T -l1024 -n10 -p4444 -u 10.0.2.22`
- Stop wireshark capture on PC1, and save the captured traffic to files.

Lab Report

Observe the captured traffic captured by Wireshark and answer the following questions. Include relevant parts of the Wireshark data to support each of your answers.

- How many packets are exchanged in the data transfer? How many packets are transmitted for each UDP datagram? What is the size of the UDP payload of these packets?
- Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and UDP headers, to the amount of application data transmitted.
- Inspect the fields in the UDP headers. Which fields in the headers do not change in different packets?
- Observe the port numbers in the UDP header. How did the nttcp sender select the source port number?

Part II: Transmitting Data with TCP

- On PC1, start wireshark to capture packets on interface eth0 between PC1 and PC2.
- On PC2, start a nttcp receiver that receives TCP traffic with the following command:
  `PC2% nttcp -i -l1024 -n10 -p4444`
- On PC 1, start a nttcp sender that transmits TCP traffic by typing
  `PC1% nttcp t -T -l1024 -n10 -p4444 -D 10.0.2.22`
- Stop wireshark capture on PC1, and save the captured traffic to files.

Lab Report

Observe the captured traffic captured by Wireshark and answer the following questions. Include relevant parts of the Wireshark data to support each of your answers.

- How many packets are exchanged in the data transfer? What are the sizes of the TCP segments?
- What is the range of the sequence numbers?
- How many packets are transmitted by PC1 and how many packets are transmitted by PC2?
- How many packets do not carry a payload, that is, how many packets are control packets?
- Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and UDP headers, to the amount of application data transmitted.
- Inspect the TCP headers. Which packets contain flags in the TCP header? Which types of flags do you observe?
- Compare the amount of data transmitted in the TCP and the UDP data transfers from this part and part I.
- Take the biggest UDP datagram and the biggest TCP segment that you observed, and compare the amount of application data that is transmitted in the UDP datagram and the TCP segment.

Part III: File Transfers Using TCP and UDP

The File Transfer Protocol (FTP) for copying files between hosts employs TCP as its transport protocol, thereby ensuring a reliable transfer of transmitted data. Two TCP connections are established for each FTP session: a control connection for exchanging commands and a data connection for the file transfer.

The Trivial File Transfer Protocol (TFTP) is a minimal protocol for transferring files without authentication. TFTP employs UDP for data transport. A TFTP session is initiated when a TFTP client sends a request to upload or download a file to UDP port 69 of a TFTP server. When the request is received, the TFTP server picks a free UDP port and uses this port to communicate with the TFTP client. Since UDP does not recover lost or corrupted data, TFTP is responsible for maintaining the integrity of the data exchange, TFTP transfers data in blocks of 512 bytes. A block must be acknowledged before the next block can be sent. When an acknowledgment is not received before a timer expires, the block is retransmitted.

- Create a large file in /tftpboot directory on PC1. This file should be at least 2 MB. Rename it to large.file. Make sure that the directory /tftpboot and everything inside are readable by all user groups by

  `chmod -R 777 /tftpboot`

- **FTP server setup:** Make sure that the following lines in `/etc/vsftpd.conf` are uncommented.

  `local_enable=YES`

  `write_enable=YES`

  `chroot_local_user=YES`

  Then, restart the vsftpd (FTP server daemon) by typing the command:

  `sudo /etc/init.d/vsftpd restart`

- Start wireshark on interface eth0 of PC1.

- Perform FTP file transfer: On PC2, perform the following steps:

  - Change the current directory to /labdata. If not exist, create it.
  - Invoke an FTP session to PC1 by typing
    `PC2% cd /labdata`
    `PC2% ftp 10.0.1.11`

- Log in as the root.

- Transfer file large.file from PC1 to directory /labdata on PC2 by typing

  `ftp>cd /tftpboot`

  `ftp>get large.file`

  `ftp>quit`

- Observe the output of the FTP session and save the output to a file.

- TFTP server setup: At PC1, make sure the following line appear in /etc/inetd.conf
  `tftp dgram udp wait nobody /usr/sbin/tcpd in.tftpd`
- Perform TFTP file transfer: On PC2, perform the following steps:
  - Invoke an TFTP session to PC1 by typing
    `PC2% cd /labdata`
    `PC2% tftp 10.0.1.11`
    `tftp>get large.d`
    `tftp>quit`
    By default, TFTP copies data from the `directory /tftpboot`. Therefore, we can transfer file by using command get large.file immediately.
  - Observe the output of the TFTP session and save the output to a file.
- On PC1, stop wireshark output.

Lab Report

Include relevant parts of the Wireshark data to support each of your answers.

- From the timestamps recorded by wireshark, obtain the times it took to transfer the file with FTP and with TFTP. Use your knowledge of FTP, TFTP, TCP, and UDP to explain the outcome.
- Identify the TCP connections that are created in the FTP session, and record the port numbers at the source and at the destination.

Part IV: TCP Connection Management

TCP is a connection-oriented protocol. The establishment of a TCP connection is initiated when a TCP client sends a request for a connection to a TCP server. The TCP server must be running when the connection request is issued.

TCP requires three packets to open a connection. This procedure is called a three-way handshake. During the handshake the TCP client and TCP server negotiate essential parameters of the TCP connection, including the initial sequence numbers, the maximum segment size, and the size of the windows for the sliding window flow control. TCP requires three or four packets to close a connection. Each end of the connection is closed separately, and each part of the closing is called a half-close.

TCP does not have separate control packets for opening and closing connections. Instead, TCP uses bit flags in the TCP header to indicate that a TCP header carries control information. The flags involved in the opening and the closing of a connection are SYN, ACK, and FIN.

Opening and closing a TCP connection

- Telnet server setup: At PC2, make sure that the following lines exist in `/etc/securetty`

  `pts/0`

  `pts/1`

  `pts/2`

  `pts/3`

  `pts/4`

  `pts/5`

  Then, restart the telnet server service by typing the command `/etc/init.d/openbsd-inetd restart`

- Start wireshark on the eth0 interface of PC 1 to capture traffic of the Telnet connection. Do not set any filters.

- Establish a Telnet session from PC1 to PC2 as follows:

  `PC1% telnet 10.0.2.22`

- On PC1, type Ctrl-] at the Telnet prompt and type quit, to terminate the connection by client.

- The closing of a connection can also be initiated by the server application. Reestablish a Telnet session on PC1 to PC2. Do not type anything. After a while, the connection will be closed by the TCP server and a message displayed at the Telnet client application.

- Save the wireshark output.

Lab Report

Use the saved Wireshark output to answer the following questions. Include relevant parts of the Wireshark data to support each of your answers.

- Identify the packets of the three-way handshake. Which flags are set in the TCP headers? Explain how these flags are interpreted by the receiving TCP server or TCP client.
- During the connection setup, the TCP client and TCP server tell each other the first sequence number they will use for data transmission. What are the initial sequence numbers of the TCP client and the TCP server?
- Identify the first packet that contains application data. What is the sequence number used in the first byte of application data sent from the TCP client to the TCP server?
- The TCP client and TCP server exchange window sizes to get the maximum amount of data that the other side can send at any time. Determine the values of the window sizes for the TCP client and the TCP server.
- What is the MSS value that is negotiated between the TCP client and the TCP server?
- How long does it take to open a TCP connection?
- In step 5, identify the packets that are involved in closing the TCP connection. Which flags are set in these packets? Explain how these flags are interpreted by the receiving TCP server or TCP client.
- Describe how the closing of the connection in Step 5 is different from Step 4. How long does the Telnet server wait until it closes the TCP connection?

Requesting a connection to a nonexisting host

- Start a new traffic capture with wireshark on interface eth0 of PC1.
- Set a static entry in the ARP table for the IP address 10.0.3.100. (Note that the IP address does not exist.)

  `PC1% arp -s 10.0.3.100 00:01:02:03:04:05`
- From PC1, establish a Telnet session to the nonexisting host:

  `PC1% telnet 10.0.3.100`
- Wait until the TCP client gives up trying to establish a connection. Save the wireshark output.

Lab Report

Include relevant parts of the Wireshark output to support each of your answers.

- How often does the TCP client try to establish a connection? How much time elapses between repeated attempts to open a connection?
- Does the TCP client terminate or reset the connection when it gives up trying to establish a connection?
- Why does this experiment require setting a static ARP table entry?

Requesting a connection to a nonexisting port

- Start a new traffic capture with wireshark on interface eth0 of PC1.
- Establish a TCP connection to port 80 of PC2:

  `PC1% telnet 10.0.2.22 80`

  There should not be a TCP server running on PC2 that is listening at this port number. Observe the TCP segments of the packets that are transmitted.
- Save the wireshark output.

Lab report

Include relevant parts of the Wireshark data to support your answers.

- How does TCP at PC2 close this connection?
- How long does the process of ending the connection take?