Lab Exercise : TCP and UDP

AIM

To investigate the behaviour of TCP and UDP in greater detail.

EXPERIMENT 1: Understanding TCP Basics

Tools

For this experiment, we will use the Wireshark packet analyser that we used extensively in the previous lab. Before you begin go to the "Trace Files" link on the google class and download the trace for the TCP lab.

Exercise

Follow the steps described below. You will notice certain questions as you attempt the exercise. Write down the answers for your own reference.

IMPORTANT NOTE: When some of you conducted these experiments you must have noticed that the sequence numbers for the sender and receiver started from zero. The reason for this is that Ethereal by default scales down all real sequence numbers such that the first segment in the trace file always starts from 0. To turn off this feature, you have to click Edit->Preferences>Protocols->TCP and then disable the "Relative Sequence Numbers and Window Scaling" option. Note that the answers below reflect this change. If some of you conducted the experiment without this change, the sequence numbers that you observed will be different from the ones in the answers.

Step 1: Open an xterm and run Wireshark.
Step 2: Load the trace file tcp-ethereal-trace-1 by using the File pull down menu, choosing Open and selecting the appropriate trace file. This file captures the sequence of messages exchanged between a host and a remote server (gai.cs.umass.edu). The host transfers a 150 KB text file, which contains the text of Lewis Carrol's Alice's Adventure in Wonderland to the server. Note that the file is being transferred from the host to the server.

Step 3: Now filter out all non-TCP packets by typing "tcp" (without quotes) in the filter field towards the top of the Ethereal window. You should see a series of TCP and HTTP messages between the host in MIT and gaia.cs.umass.edu. The first three packets of the trace consist of the initial three-way handshake containing the SYN, SYN ACK and ACK messages. You should see an HTTP POST message and a series of "HTTP Continuation" messages being sent from the host in MIT to gaia.cs.umass.edu. Recall from the first lab exam that there is no such thing as an HTTP Continuation message this is Ethereal's way of indicating that there are multiple segments being used to carry a single HTTP message. You should also see TCP ACK segments being returned from gaia.cs.umass.edu to the host in MIT.

Question 1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? (Hint: To answer this question it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window")

Question 2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

Step 5: Since this lab is about TCP rather than HTTP, change Ethereal's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Ethereal do this, select Analyze->Enabled Protocols. Then uncheck the HTTP box and select OK.

Question 3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is in the segment that identifies the segment as a SYN segment?

Question 4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Question 5. What is the sequence number of the ACK segment sent by the client computer in response to the SYNACK? What is the value of the Acknowledgement filed in this ACK segment? Does this segment contain any data? What is it in the segment that identifies this segment as a ACK segment?

Question 6. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Ethereal window, looking for a segment with a "POST" within its DATA field.

Question 7. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST) sent from the client to the web server (Do not consider the ACKs received from the server as part of these six segments)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see page 237 in text, Section 3.5.3) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the measured RTT (SampleRTT) for the first segment, and then is computed using the EstimatedRTT equation on page 237 for all subsequent segments. Set α to 0.125.Note: Ethereal has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics->TCP Stream Graph>Round Trip Time Graph. However, do not use this graph to answer the above question.

Question 8. What is the length of each of the first six TCP segments?

Question 9. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

Question 10. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

Question 11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 245 in the text).

Question 12. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

EXPERIMENT 2: Understanding UDP Basics

Whenever possible, when answering a question below, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. To print a packet, use File->Print, choose Selected packet only, choose Packet summary line, and select the minimum amount of packet detail that you need to answer the question.

1. Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields.
2. By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.
3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)
5. What is the largest possible source port number? (Hint: see the hint in 4.)
6. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields).
7. Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets.

END OF LAB