Name:_________Dhanush Sureshbabu_________________
UCID:__________________________ds676___________________________________
Class Section: _______________004____

Wireshark Assignment 2: UDP Ping Client and Server

This wireshark assignment is to be used in conjunction with Programming Assignment 2. The wireshark trace is intended to demonstrate that your program is behaving as expected. **As such, please ensure that the program trace you provide shows at least 3 acknowledged ping requests and at least two "dropped" ping requests. (Since the server responds at random, this may require re-running your program to get a representative trace.)**

Test Environment:
Run the UDP Ping client and server on the same host[1]. Enable wireshark on the null/loopback interface and capture packets between client and server. Store the trace in a .pcap file. When analyzing the .pcap file, specify a filter such that only your application packets are shown in the packet summary window.

Note: When a question asks for "packet number", this refers to the "No." column in the summary window of the wireshark trace.

Using the wireshark trace and what you have learned so far, answer the following questions (your answers MUST be consistent with that shown in the wireshark trace you hand in):

1. What packet numbers correspond to ping requests?
   73,75,94,114,133,152,154,156,176,195

2. What packet numbers correspond to ping responses?
   74,113,153,155,175
3. How many ping requests were "lost" (no response from server)? 5

4. What is the loss rate of ping requests (%)? 50%

5. What is the IP address of the client? Which header includes this address in a packet? (2)

   127.0.0.1 The IP Header

6. What is the IP address of the server? Which header includes this address in a packet? (2)
   127.0.0.1 The IP Header

7. What port is the client listening on? Which header includes this port in a packet? (2)
   The client is listening on port 64248 and the UDP header includes this port in a packet
8. What port is the server listening on? Which header includes this port in a packet? (2)

[1] Of course, if you have access to 2 machines, and can run the client on one machine (on unencrypted link) and the server on the other, all the better.

The port is listening on port 17000 and the UDP header includes this port in a packet

9. In your program, where is the port number that the client listens on specified?

The port number is chosen by the UDP implementation software (Operating System) from a range of port numbers called Ephemeral Ports.

10. In your program, where is the port number that the server listens on specified?
    Port=argv[2]

11. What protocol in the protocol stack is responsible for forwarding a packet from a source host to a destination host?
    Network Layer->IP

12. What protocol in the protocol stack is responsible for multiplexing and de-multiplexing ping packets to and from the application processes?

    Transport Layer->UDP

13. What application data does the ping client send to the server and what is its length in bytes? (2)
    The ping client sends the message and sequence number to the server. This is 01 00 00 00 01 00 The total bytes are 8

14. What application data does the ping server send to the client and what is its length in bytes? (2)

    The ping server sends the message and sequence number to the server. This is 02 00 00 00 01 00 The total bytes is 8

15. How much overhead (in bytes) does the network stack (link layer, network layer and transport layer) in the operating system add to each ping packet? (3)
    NETWORK-20 bytes Transport-16 bytes =36 bytes

16. Network protocols format messages using network byte order. Explain network byte order and why is it used? (2)
    Network byte order is a standard way to represent multi-byte values so that communication can take place across a network without having to know the "endianness" of the two machines communicating.

    Total: 25

Submission Guidelines:
Please submit the following five types of individual files to Moodle by due date. **Please, <u>NO</u> zip files.**

- ✓ Submit the client and server source program files (please include name, UCID, section in comments at top of source files)
- ✓ Submit screenshots in .pdf format showing the trace output of the client and server and round-trip time results (be sure the .pdf is legible)
- ✓ Submit the README file (non-python programs only. see README submission format on Moodle)
- ✓ Submit the wireshark .pcap file captured while running the client and server programs, and the one used to answer questions in this document
- ✓ Submit this Word document with completed questions

Also, please hand in a **<u>paper copy</u>** of this Word document in the first class on or after due date. Alternatively, please bring to my office (GITC 4305). If I am not available, slip under my door. Do NOT send email.

Grading Rubric: Total of 50 points worth 5% of overall grade

- Questions (25)
- Program (25)
  - Ping request packet format as specified (4)
    - Message type = 1 (request)
    - Sequence number increments from 1
  - Ping response packet format as specified (4)
    - Message type = 2 (response)
    - Sequence number echoed back to client
  - Both messages in network byte order (2)
  - Client sends 10 ping requests, and server responds (randomly; at least 3 responses are demonstrated) (6)
  - Server randomly drops ping requests; at least 2 drops are demonstrated (4)
  - Client sends next ping message after time-out of 1 sec (1)
  - RTT calculated correctly for each acknowledged packet (1)
  - Min RTT tracked and printed (1)
  - Max RTT tracked and printed) (1)
  - Loss rate calculated and printed (1)

Academic Integrity
If academic integrity standards are not upheld, no credit is given. This includes copying of program or wireshark lab or .pcap file from any source, or hard-coding of results in your program.

Notes:

1. **Timestamp Resolution:**

When testing your program, especially when running client and server on the same host, you might find little, if any, difference in the RTT. The values will depend on the test environment, and timer resolution on your machine. If this is an issue, you should simulate longer delays, by adding in a random "wait" or "sleep" function on the server after receipt of a request, but before responding.

2. **Data structure Alignment and Padding**

Compilers on modern processors will typically try to align data structures to that optimal for the machine. For example, 4-byte values are stored at addresses divisible by 4, and 8-byte values are stored at addresses divisible by 8. For this reason, you may find that in data structures, such as “struct” in C/C++, the compiler adds padding in certain cases. For example, if using a C struct to store a 4-byte integer, followed by an 8-byte integer, 4-bytes of padding may be inserted between the 2 integers to ensure 64-bit alignment. Thus, a struct of 12 bytes may become a struct of 16 bytes with this padding. (You can determine the size of the structure and the relevant fields in the structure using the sizeof() operator, and also looking at the wireshark output). This padding is acceptable for this exercise if it occurs; just make sure to note this information when submitting your assignment.