ECE 5600 Project (Phase 3)

OBJECTIVE

1. Be Familiar with the structure of IPv4 frame.
2. Understand and implement a send packet function via IP frame.
3. Know ICMP message types and implement ICMP echo.
4. Know the concepts of fragmentation and defragmentation.

BACKGROUND

Internet Protocol (IP) and Internet Message Control Protocol (ICMP) belong to the network layer. IP basically provides 2 services: send a packet and receive a packet. Unfortunately, it is difficult to test IP in isolation, so we will be implementing just enough of ICMP to send and respond to a “ping”. Fortunately, ping packets are (or can be) small enough to transmit on Ethernet without fragmentation, so you can defer the whole fragment/de-fragment problem for now.

PRE-LAB READING

Chapter 5. P431~455

PROJECT PROCEDURE

Step 1. Write code to implement IP (except for fragmentation – assume packets always fit in a single frame for now). Have your code loop dispatch frames to the IP code, which will check the header and dispatch the packet (with its length and source IP address) to a higher layer based on the protocol byte (in the IP header). For example, if the protocol byte is 1, the packet should be dispatched to ICMP.
Note: even though ICMP and IP are both network layer protocols, ICMP is layered upon IP.
(**Remember to discard packets with bad checksums**.)

Step 2. Implement a packet sending function.

You need to attach the IP header to an Ethernet frame as its data field. To compose a packet, make sure you add the correct IP header length (usually 20 bytes), calculate the checksum (the sample code contains the checksum calculation function), etc. and send the packet. For a specific given destination IP address, you also need to check if the address is within the lab or not. That can be implemented by providing an IP mask and gateway address so that if the address is not a local one, you can send it to the gateway instead. (The Network Lab gateway is 192.168.1.1)
The destination MAC address in the Ethernet header can be found by using codes from lab 2.

Step 3. Write code to implement the ICMP protocol for echo (ping) request. When ICMP gets an echo request (type = 8), it must respond by sending an echo reply (type = 0) to the originator’s IP address. The data in the packet should be echoed without alteration, but since the type field has changed from 8 to 0, you will need to re-compute the ICMP checksum.

Step 4. Test your code by pinging your IP address from a different computer. If you have implemented it properly, the ping will succeed. Copy the *first* four lines of the ping output (from the other computer) and paste them into a file.

Step 5. Test your code by pinging the gateway (192.168.1.1). Once that is working, ping an address outside your subnet (one of Google’s IP addresses is 74.125.127.103). Verify with Wireshark that your ICMP echo request went out and that the other computer responded. Copy & paste your echo request and the other computer’s response into the file (I’ll be checking that the packet data change with each ping, so record at least 2 request/reply pairs.)

REPORT REQUIREMENTS

1. Screenshot the Wireshark window when you successfully send a packet in Step 2. Highlight the packet you send and show the checksum value is correct.

2. In Step 3, you should provide the screenshot for echo request/reply pairs. Be careful that you should see two duplicate responses. Make sure the ICMP checksum is correct in your experiment.
3. In Step 4, paste the first four lines of ping output on your report.
4. In Step 5, copy & paste your echo request and the other computer's response into the file.
5. Online submissions only, for the screenshots, please make sure the pictures are large and clear enough in your lab report, otherwise, please archive all your files including the screenshots in .zip file.
6. **Project Due: Nov 13, 2017**