ECE 5600 Project (Phase 2)

OBJECTIVE

1. Familiar with the structure of Ethernet frame.
2. Understand the ARP mechanism.
3. Know how to use Wireshark to analyze the captured frame, and locate the MAC addresses and IP addresses.
4. Construct the ARP cache.

BACKGROUND

Address resolution protocol (ARP) belongs to the MAC sublayer of the data-link layer. For the purposes of this project, it provides a single service to upper layers, namely to send a frame to a node on the local network whose protocol address (in our case, IP address) is known. Layers above ARP are blissfully unaware of what MAC addresses mean or that they even exist.

This is not as easy as it might seem. If ARP does not already know what MAC address goes with the specified IP address, it must broadcast an ARP request. Once the reply comes back, it must immediately send the frame. Care must be taken that in-coming packets are handled (at all protocol layers) while awaiting the ARP reply.

PRE-LAB READING

Chapter 5. P442~469

PROJECT PROCEDURE

1. Write a code loop whose job is to get packets from frameio and dispatch them to their

appropriate protocol stack. We will only be using two protocols in this project: 0x800 (IP) and 0x806 (ARP). You may dispatch frames in any way you please (function call, message queue, etc.), but this loop will need to run as its own thread (I recommend you use pthread, you will need to add –lpthread to your compile line) Hint: see example1.cpp for details. You can use example1.cpp as a template to start your project.

2. Write codes to respond to ARP requests. When your ARP implementation receives an ARP request and its destination IP address is your computer's, your machine must immediately send an ARP reply. The ARP reply must use the correct destination MAC address (i.e. not broadcast). In wireshark, you will see an ARP reply from the machine and a duplicate one generated from your code.
3. Test your code by pinging your IP address from a different computer (Use arping). Verify with wireshark that your protocol stack has responded. You should see the ARP request, your reply (duplicated, one is from your code, the other is from the machine itself), and ICMP frames on the wire. Copy & paste the ARP request, your ARP response to a file. Attention: It will broadcast ARP request and you will observe two ARP replies: one is from machine automatically and the other is from your code.
4. Implement a cache mechanism to facilitate Layer 2 communication. Specifically, upon receiving any ARP frame (even not for you) in your code, cache its source MAC and IP address.
5. Now, write codes to send an ARP frame with known IP address . If the IP address is already in the cache, compose the frame and send it immediately. Otherwise, send an ARP broadcast and get the ARP reply, cache the IP and MAC address pair. then send the frame now has known MAC addresses. (Recall that we need MAC address in layer 2 to communicate.) Again copy & paste your request and the other

computer’s response into the file.

Hints for this project can be found in canvas

REPORT REQUIREMENTS

1. Requirements for each procedure are listed in the hint file, please follow the instructions. Make sure your screenshots are clear in the reports otherwise please attach the file in your submissions.
2. Even if you work in groups, please submit your own lab report, the code can be same.
3. **Project due; October 16, 2017**

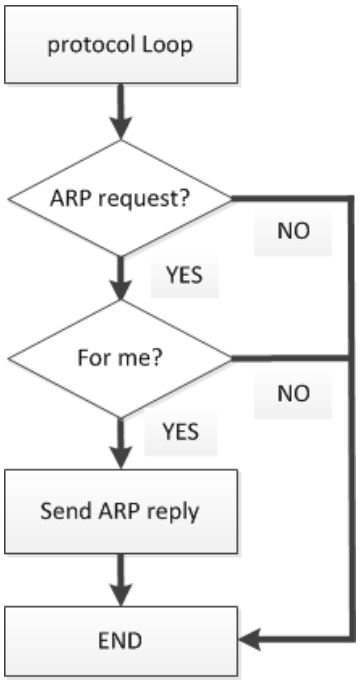
Hints for Project 2

PROCEDURE 1-3:

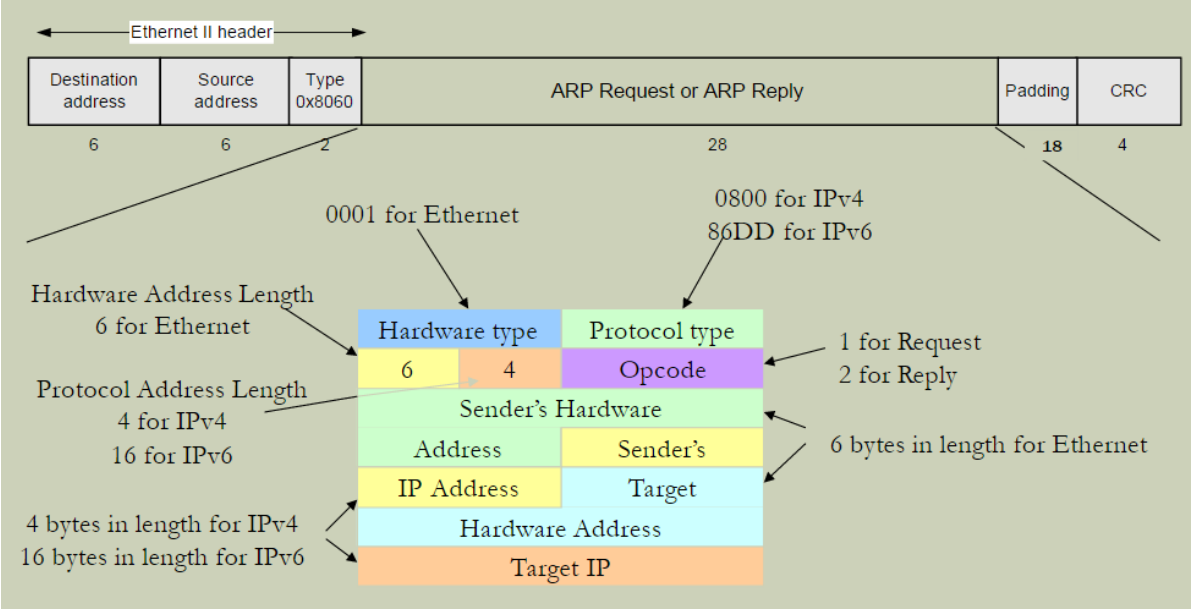
Procedure 1 is already done in the example code (protocol loop function) , it checks the buf.prot to see if it is an IP or ARP protocol. Then dispatches the packet either to ip_protocol_loop or arp_protocol_loop function.

What you need to do is checking the Opcode in the arp_protocol_loop function to see if the incoming frame is a request or reply. If it is a request, go on checking if its target IP address is the same as your machine's. If true, it means that the request is for your computer.

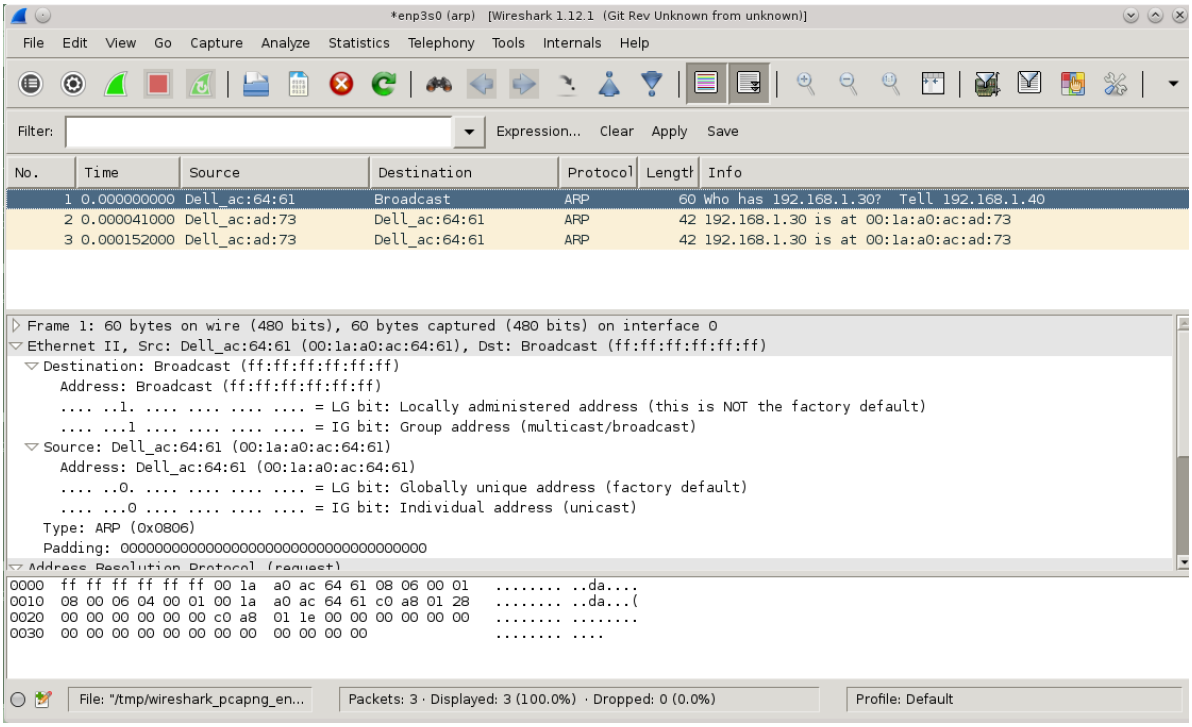
Then, you need to compose an ARP reply frame, you can do it by looking at a regular reply frame made by the computer from wireshark. (arping your IP address from other machine and see the reply) Make sure you know what each field means.
Write codes to respond to an ARP request.



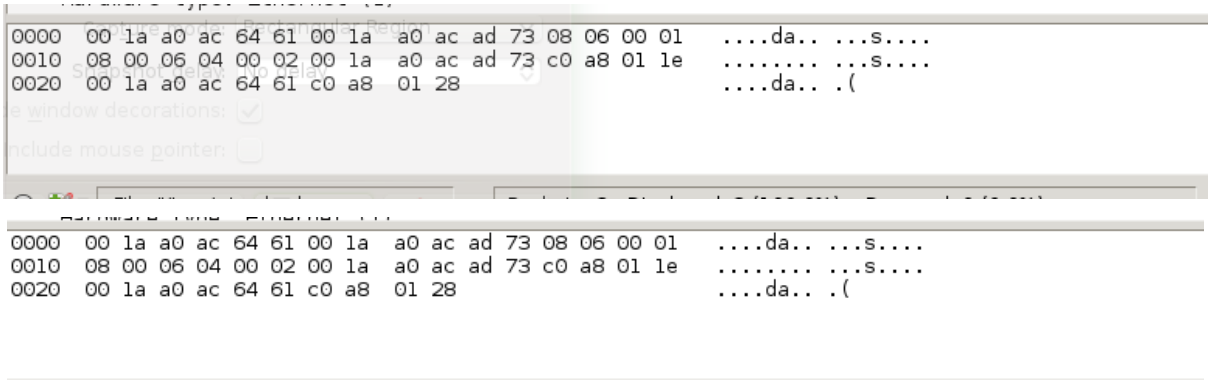
A brief flow chart.



Frame structure



Screenshots 1



Screenshots 3

Except for the screenshots, please also copy and paste the contents of your reply to the lab report.

Also, pay attention of the broadcast frame, it can be useful later.

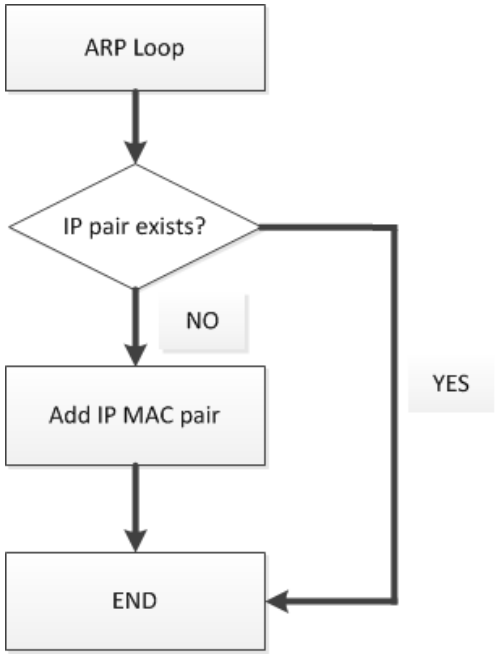
PROCEDURE 4-5:

First, set up a cache function to store the IP-MAC pair in the code.

This cache function will store any ip-mac pair even the frame is not for you.

One possible way is to use struct, like the ether_frame.

```
Struct ipmac
{
    octet ip[4];
    octet mac[6];
};
```

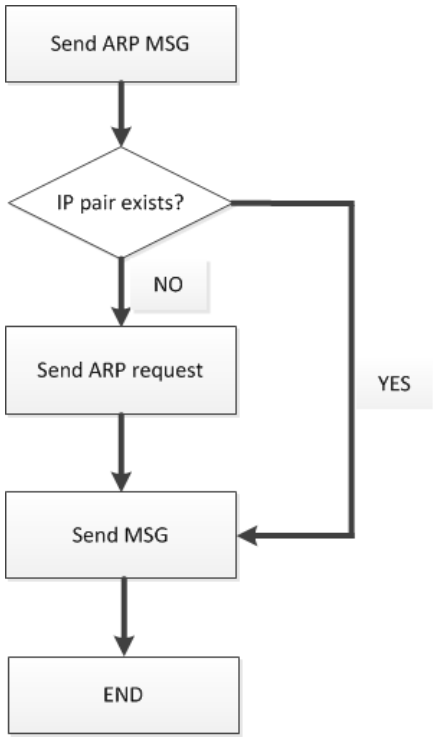


A brief flow chart.

Basically, what you need to do in procedure 4 is to send an ethernet frame. For consistency, you will send an ARP message to the destination with known IP address. (The IP address either comes from input from console or pre-defined in the code, depends on the implementation detail of your own codes).

With only known IP address, you can't send a frame in layer 2 protocol. Because you need to know the corresponding MAC address, since we have IP-MAC cache, first, you should look up that table. If the pair exists, compose an ARP frame and send directly, if not, then send ARP broadcast first(again, check wireshark to see the details of a broadcast frame), when the target machine replies, cache the IP-MAC pair, then send the frame(So, next time ,if you send to the same IP address, you just need to check ip-mac table).

For simplicity, the ARP msg can also be the ARP reply in procedure 3.



A brief flow chart.

In the report, show me two scenarios:

IP-MAC does not exist, your code will send an ARP broadcast, then the target machine replies, then your code add this pair, sends the ARP msg directly, the whole procedure should be highlighted in wireshark.

IP-MAC exists, ARP msg sends directly, also screenshots the wireshark.

EXTRA CREDIT PART

If you feel good about procedure 1-5, consider to add a timer about the ip-mac cache, which is also the mechanism in our computer. Specifically, when adding a new pair, set up a timer, when the time expires (say, 20 seconds), delete the pair.

In the following page, I list one picture which can explain the procedure 4-5, and also the extra credit part.

Always highlight the important results!

If you have further concerns, please send me the email.

AN EXAMPLE

When I run my code, it listens to every traffic and adds ip-mac pair if doesn't exist (if exists, it will output ip mac pair already exists).

My code always waits for my input from console, p 192 168 1 40 is the command to send ARP msg (Not IP msg in the picture) to the destination 192.168.1.40.

As you can see 192.168.1.40 is in cache, so it will compose an ARP frame with known mac address directly sent.

If you try to send to 192.168.1.50, it's not in cache, so send ARP broadcast first, waits for the reply, then adds to the cache, and send ARP msg.

This code has a timer for the cache, after certain time, 192.168.1.40 is deleted from cache, so if I still try to send msg to this address, it will send ARP broadcast first, waits for the reply, then adds to the cache again, and then send ARP msg.