CptS/EE 455
Project #2

Instructor: Adam Hahn

Due: 10/14/2019 at 11:59 pm

Assignment: This project will implement a simple program that can send ARP requests and process ARP replys. The program should be run and validated within a Mininet environment. You should extend the C program you developed in Project 1, adding the required functionality.

Deliverable: Submit all code to blackboard by the due date. Ensure instructions to build and run the program are provided. You'll need to schedule a time with the TA to review both Project 1 and Project 2.

1 Overview

In this project, you must develop a program that can correctly send ARP requests and receive ARP responses. The program should take the `InterfaceName` (similar to Project 1), along with an IP address, `DestIP`, which will be the target of the ARP request.

Send ARP Request:

```
./455_proj2 <InterfaceName> <DestIP>
```

Examples:

```
./455_proj2 h1x1-eth0 10.0.0.2
```

2 Requirements

Develop a program that can successfully send ARP requests for an IP address and then return the resulting HW address in the ARP response. The program must perform the following tasks:

a) Correctly specify all fields in the ARP request payloads
b) During the creation of Ethernet segments, you'll need to specify the correct value for ARP `type`, which is ETH_P_ARP.
c) Correctly specific the destination and source Ethernet addresses on the ARP request.
d) Monitor for ARP responses associates with the request and return the correct HW address

3 Recommendations

The following list includes some recommended techniques that will likely be useful in the development of this router, but are not strictly required.

a) You'll also need to convert IP addresses from string to binary format. The `inet_aton()` function can perform this. Furthermore, the `struct in_addr` is useful for storing

addresses, which stores in an unsigned long `s_addr`.

```
struct in_addr addr
inet_aton("192.168.1.0", &addr);
```

b) You will also need to generated and parse ARP requests. To do this, create the following struct which can be used to easily manage these requests.

```
struct arp_hdr {
        uint16_t        ar_hrd;
        uint16_t        ar_pro;
        unsigned char   ar_hln;
        unsigned char   ar_pln;
        uint16_t        ar_op;
        unsigned char   ar_sha[6];
        unsigned char   ar_sip[4];
        unsigned char   ar_tha[6];
        unsigned char   ar_tip[4];
};
```

c) Use the following `ioctl` commands/options to get the IP address with an interface.

```
unsigned int get_ip_saddr(char *if_name, int sockfd){
   struct ifreq if_idx;
   memset(&if_idx, 0, sizeof(struct ifreq));
   strncpy(if_idx.ifr_name, if_name, IFNAMSIZ-1);
   if (ioctl(sockfd, SIOCGIFADDR, &if_idx) < 0)
       perror("SIOCGIFADDR");
    return ((struct sockaddr_in *)&if_idx.ifr_addr)->sin_addr.s_addr;
  }
```

4 Grading

Your project will be graded by the TA looking for correct functionality. We will test whether each host h1 can successfully send an ARP request and process the response from h2. Specifically, this will ensure Wireshark doesn't not raise any error on the ARP messages and that all correct ARP protocols values are included.