CptS/EE 455
Project #1

Instructor: Adam Hahn

Due: 9/23/2019 at 11:59 pm

Assignment: This project will use basic C sockets to send raw Ethernet packets across the network. In this project, you will develop a program that can successfully send and receive Ethernet frames, including the correct methods for addresses and sockets.

You should use C to develop the program. If you are unfamiliar/uncomfortable with C, please discuss with the instructor and another language can be chosen. The following two files are necessary for this assignment

- `455_proj2.c` (*optional*) – This is the basic template of a C program that can be extended to send link-layer sockets frames.

Deliverable: Submit all code to blackboard by the due date. Ensure instructions to build and run the program are provided.

1) Requirements

Develop a program that can successfully send and receive Ethernet frames using raw sockets. You must perform the framing of the frames' destination address, source address, type and payload. The program must perform the following tasks:

a) Send a message (taken as a command line argument) to a specified Ethernet destination address. Here `Send` specifies the sending mode, `<InterfaceName>` represents the name of the interface to send on, `<DestHWAddr>` is the destination MAC address and <Message> is the value to send.

```
./455_proj2 Send <InterfaceName>  <DestHWAddr> <Message>
```

Example:

```
./455_proj2 Send h1–eth0 01:23:45:67:89:ab 'This is a test'
```

b) Read messages that are addressed to either the network interface's MAC address, or broadcast addresses. `<InterfaceName>` represents the name of the interface to receive on. Print the associated (i) source MAC addresses, (ii) Type, and (iii) payload data.

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

Example:

```
./455_proj2 Recv h2–eth0
```

c) Run the program in Mininet, using a basic 2 host topology and demonstrate host h1 sending and host h2 receiving

```
$ mn
```

d) For the Type, use the value `ETH_P_IP,` which is defined in `netinet/ether.h.` You'll need to perform the required host/network byte order conversions.

2 Recommendations

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

a) We need sockets that can send raw packets, so you can specify the appropriate headers, therefore, please use the `AF_PACKET, SOCK_RAW,` and `htons(ETH_P_ALL)` options as shown below:.

```
if ((sockfd = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) < 0){
    perror("socket() failed");
}
```

b) It may be useful to use the `netinet/ether.h` library, which defines the `struct ether_header`, (actually defined in `net/ethernet.h`) that specifies the Ethernet header format (destination and source MAC address, along with the Type).

c) You'll need to assign a socket to a specific interface, as specified by the `<InterfaceName>` argument. To do this you'll need to use a `struct ifreq`, which is passed to certain system calls `(man netdevice)`. By specifying the interface name, `strncpy(if_idx.ifr_name, if_name, IFNAMSIZ-1),` you can then assign the socket to the correct device, `ioctl(sockfd, SIOCGIFINDEX, &if_idx)`. The index is then stored in `if_idx.ifr_index.` The whole process is shown below.

```
struct ifreq if_idx;
memset(&if_idx, 0, sizeof(struct ifreq));
strncpy(if_idx.ifr_name, if_name, IFNAMSIZ-1);
 if (ioctl(sockfd, SIOCGIFINDEX, &if_idx) < 0)
        perror("SIOCGIFINDEX");
```

d) You'll also need to get the hardware MAC address of this interface, so that you can correctly specific source MAC address. To do this, we'll once again use the `struct ifreq` and `ioctl()` function. The address is stored in `if_idx.ifr_hwaddr.sa_data.` *Note: you need to use a different* `struct ifreq` *from Part C), as the* `ioctl()` *call will resent previous contents*. The following code should appropriately provide this.

```
if (ioctl(sockfd, SIOCGIFHWADDR, &if_idx) < 0)
```

```
        perror("SIOCGIFHWADDR");
```

e) When sending and receiving a raw packet, you'll need to provide a `sockaddr_ll,` which is specific type of `sockaddr` specific for link-layer addressing (as opposed to the network layer `sockaddr_in` discussed in class. For more information please refer to `man packet`. You'll need to at least complete the following fields:

Define:

```
 struct sockaddr_ll sk_addr;
 int sk_addr_size = sizeof(struct sockaddr_ll);
```

Receive:

```
 memset(&sk_addr, 0, sk_addr_size);
 recvLen = recvfrom(sockfd, buf, BUF_SIZ, 0, (struct
                    sockaddr*)&sk_addr, &sk_addr_size );
```

Send:

```
 memset(&sk_addr, 0, sk_addr_size);
 sk_addr.sll_ifindex = if_idx.ifr_ifindex;
 sk_addr.sll_halen = ETH_ALEN;
 byteSent = sendto(sockfd, buf, sendLen, 0,
                   (struct sockaddr*)&sk_addr,
                   sizeof(struct sockaddr_ll));
```

3 Grading

Your project will be graded by the TA looking for correct functionality. You must demonstrate the ability to run the two programs together, with one sending data and the other receiving it on two different hosts in Mininet. The program will be evaluated based on whether this communication is success, including whether the destination address, source address, type, and payload are all correctly implemented.