Lab 3

Raw Socket

Create the socket

Cooked Sockets:

```
if ((s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP))==-1)
 perror("socket:");
/* ... */
if (sendto(s, buf, BUFLEN, 0, &si_other, slen)==-1)
 perror("sendto() : ");
```

Raw Sockets:

```
// Create a raw socket with UDP protocol
sd = socket(PF_INET, SOCK_RAW, IPPROTO_UDP);
if(sd < 0)
{
perror("socket() error");
// If something wrong just exit
exit(-1);
}
else
printf("socket() - Using SOCK_RAW socket and UDP protocol is OK.
```

Headers

```
//IP headers
struct ipheader {
unsigned char      iph_ihl:5, iph_ver:4;
unsigned char      iph_tos;
unsigned short int iph_len;
unsigned short int iph_ident;
unsigned char      iph_flag;
unsigned short int iph_offset;
unsigned char      iph_ttl;
unsigned char      iph_protocol;
unsigned short int iph_chksum;
unsigned int       iph_sourceip;
unsigned int       iph_destip;
};
```

```
// UDP header's structure
struct udpheader {


unsigned short int udph_srcport;
 unsigned short int udph_destport;
 unsigned short int udph_len;
 unsigned short int udph_chksum;


};
```

Fabricating the Headers

```
// Fabricate the IP header or we can use the
// standard header structures but assign our own values. ip->iph_ihl = 5;
ip->iph_ver = 4;
ip->iph_tos = 16; // Low delay
ip->iph_len = sizeof(struct ipheader) + sizeof(struct udpheader); ip->iph_ident = htons(54321);
ip->iph_ttl = 64; // hops
ip->iph_protocol = 17; // UDP
// Source IP address, can use spoofed address here!!! ip->iph_sourceip = inet_addr(argv[1]);
// The destination IP address
ip->iph_destip = inet_addr(argv[3]);

// Fabricate the UDP header
// Source port number, redundant
udp->udph_srcport = htons(atoi(argv[2]));
// Destination port number
udp->udph_destport = htons(atoi(argv[4]));
udp->udph_len = htons(sizeof(struct udpheader));
// Calculate the checksum for integrity
ip->iph_chksum = csum((unsigned short *)buffer, sizeof(struct ipheader) + sizeof(struct udpheader))
```

```
/ Inform the kernel do not fill up the packet structure
// we will build our own...
if(setsockopt(sd, IPPROTO_IP, IP_HDRINCL, val, sizeof(one)) < 0)
{
perror("setsockopt() error");
exit(-1);
}
else
printf("setsockopt() is OK.\n");
```

Sending Packet

```
int count;
for(count = 1; count <=20; count++)
{
if(sendto(sd, buffer, ip->iph_len, 0, (struct sockaddr *)&sin, sizeof(sin)) < 0) // Verify
{
perror("sendto() error");
exit(-1);
}
else
{
printf("Count #%u - sendto() is OK.\n", count);
sleep(2);
}
}
close(sd);
```