Udp.c

```
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <stdlib.h>
#include <arpa/inet.h>

struct psd_udp {
      struct in_addr src;
      struct in_addr dst;
      unsigned char pad;
      unsigned char proto;
      unsigned short udp_len;
      struct udphdr udp;
};

unsigned short in_cksum(unsigned short *addr, int len)
{
      int nleft=len;

      int sum=0;
      unsigned short *w =addr;
      unsigned short answer=0;

      while(nleft>1)
      {
            sum+=*w++;
            nleft-=2;
      }

      if(nleft==1){
*(unsigned char *)&answer = *(unsigned char *)w;
            sum+=answer;

      }

      sum= (sum>>16) + (sum & 0xFFFFF);
      sum+= sum>>16;
      answer = ~sum;
      return answer;



}
```

```
unsigned short in_cksum_udp(int src, int dst, unsigned short *addr, int len)
{
        struct psd_udp buf;

        memset(&buf, 0, sizeof(buf));
        buf.src.s_addr = src;
        buf.dst.s_addr = dst;
        buf.pad = 0;
        buf.proto = IPPROTO_UDP;
        buf.udp_len = htons(len);
        memcpy(&(buf.udp), addr, len);
        return in_cksum((unsigned short *)&buf, 12 + len);
}



void *run(void *arg)
{
        struct ip ip;
        struct udphdr udp;
        int sd;
        const int on = 1;
        struct sockaddr_in sin;


        u_char *packet;
        packet = (u_char *)malloc(60);


        ip.ip_hl=0x5;
        ip.ip_v=0x4;
        ip.ip_tos=0x0;
        ip.ip_len=60;
        ip.ip_id=htons(12830);
        ip.ip_off=0x0;
        ip.ip_ttl=64;
        ip.ip_p=IPPROTO_UDP;
        ip.ip_sum=0x0;
        ip.ip_src.s_addr=inet_addr("127.0.0.1");
        ip.ip_dst.s_addr= inet_addr("127.0.0.1");

        ip.ip_sum=in_cksum((unsigned short *)&ip , sizeof(ip));
        memcpy(packet, &ip, sizeof(ip));

        udp.uh_sport = htons(45512);
        udp.uh_dport = htons(23);
```

```
       udp.uh_ulen = htons(8);
       udp.uh_sum = in_cksum_udp(ip.ip_src.s_addr, ip.ip_dst.s_addr, (unsigned
short*)&udp, sizeof(udp));
       memcpy(packet + 20, &udp, sizeof(udp));

       //fill IP and UDP here

       if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)) < 0) {
              perror("raw socket");
              exit(1);
       }

       if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
              perror("setsockopt");
              exit(1);
       }


       memset(&sin, 0, sizeof(sin));
       sin.sin_family = AF_INET;
       sin.sin_addr.s_addr = ip.ip_dst.s_addr;
       int c=0;
       while(1)
       {
       // c+=1;

       if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0)  {
              perror("sendto");
              exit(1);
       }
       }

       }


int main(int argc, char **argv)
{
       run(NULL);
       return 0;
}
```

Tcp.c

```
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

unsigned short in_cksum(unsigned short *addr, int len) {
    int nleft = len;
    int sum = 0;
    unsigned short *w = addr;
    unsigned short answer = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(unsigned char *)&answer = *(unsigned char *)w;
        sum += answer;
    }

    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += sum >> 16;
    answer = ~sum;
    return answer;
}

unsigned short in_cksum_tcp(int src, int dst, unsigned short *addr, int
len) {
    struct pseudo_header {
        struct in_addr src;
        struct in_addr dst;
        unsigned char pad;
        unsigned char proto;
        unsigned short tcp_len;
        struct tcphdr tcp;
    } psh;
    memset(&psh, 0, sizeof(psh));
    psh.src.s_addr = src;
    psh.dst.s_addr = dst;
    psh.pad = 0;
    psh.proto = IPPROTO_TCP;
    psh.tcp_len = htons(len);
    memcpy(&(psh.tcp), addr, len);
    return in_cksum((unsigned short *)&psh, 12 + len);
}

void send_tcp_packet() {
    int sd;
    const int on = 1;
    struct sockaddr_in sin;
    u_char *packet;
    packet = (u_char *)malloc(60);

    struct ip ip;
    struct tcphdr tcp;

    ip.ip_hl = 0x5;
    ip.ip_v = 0x4;
    ip.ip_tos = 0x0;
    ip.ip_len = 60;
    ip.ip_id = htons(12830);
    ip.ip_off = 0x0;
    ip.ip_ttl = 64;
    ip.ip_p = IPPROTO_TCP;
    ip.ip_sum = 0x0;
    ip.ip_src.s_addr = inet_addr("127.0.0.1");
    ip.ip_dst.s_addr = inet_addr("127.0.0.1");

    ip.ip_sum = in_cksum((unsigned short *)&ip, sizeof(ip));
    memcpy(packet, &ip, sizeof(ip));

    tcp.th_sport = htons(45512);
    tcp.th_dport = htons(23);
    tcp.th_seq = htonl(12345);
    tcp.th_ack = htonl(0);
    tcp.th_off = 5;
    tcp.th_flags = TH_SYN;
    tcp.th_win = htons(5840);
    tcp.th_sum = 0;  // Set to 0 before calculating checksum
    tcp.th_sum = in_cksum_tcp(ip.ip_src.s_addr, ip.ip_dst.s_addr, (unsigned
short*)&tcp, sizeof(tcp));
    memcpy(packet + 20, &tcp, sizeof(tcp));

    if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
        perror("raw socket");
        exit(1);
    }

    if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
        perror("setsockopt");
        exit(1);
    }

    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = ip.ip_dst.s_addr;
    int c=0;
    while(c<10){
    c=c+1;
    if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0)  {
        perror("sendto");
        exit(1);
    }
    }
   // close(sd);
   // free(packet);
}

int main() {
    send_tcp_packet();
    return 0;
}
```

Tcp_mod.c

```
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

unsigned short in_cksum(unsigned short *addr, int len) {
    int nleft = len;
    int sum = 0;
    unsigned short *w = addr;
    unsigned short answer = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(unsigned char *)&answer = *(unsigned char *)w;
        sum += answer;
    }

    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += sum >> 16;
    answer = ~sum;
    return answer;
}

unsigned short in_cksum_tcp(int src, int dst, unsigned short *addr, int
len) {
    struct pseudo_header {
        struct in_addr src;
        struct in_addr dst;
        unsigned char pad;
        unsigned char proto;
        unsigned short tcp_len;
        struct tcphdr tcp;
```

```c
    } psh;

    memset(&psh, 0, sizeof(psh));
    psh.src.s_addr = src;
    psh.dst.s_addr = dst;
    psh.pad = 0;
    psh.proto = IPPROTO_TCP;
    psh.tcp_len = htons(len);
    memcpy(&(psh.tcp), addr, len);
    return in_cksum((unsigned short *)&psh, 12 + len);
}

void send_tcp_packet() {
    int sd;
    const int on = 1;
    struct sockaddr_in sin;
    u_char *packet;
    packet = (u_char *)malloc(60);

    struct ip ip;
    struct tcphdr tcp;

    ip.ip_hl = 0x5;
    ip.ip_v = 0x4;
    ip.ip_tos = 0x0;
    ip.ip_len = 60;
    ip.ip_id = htons(12830);
    ip.ip_off = 0x0;
    ip.ip_ttl = 64;
    ip.ip_p = IPPROTO_TCP;
    ip.ip_sum = 0x0;
    ip.ip_src.s_addr = inet_addr("127.0.0.1");
    ip.ip_dst.s_addr = inet_addr("127.0.0.1");

    ip.ip_sum = in_cksum((unsigned short *)&ip, sizeof(ip));
    memcpy(packet, &ip, sizeof(ip));

    tcp.th_sport = htons(45512);
    tcp.th_dport = htons(23);
    tcp.th_seq = htonl(12345);
    tcp.th_ack = htonl(0);
    tcp.th_off = 5;
    tcp.th_flags = TH_SYN;
    tcp.th_win = htons(5840);
    tcp.th_sum = 0;  // Set to 0 before calculating checksum
    tcp.th_sum = in_cksum_tcp(ip.ip_src.s_addr, ip.ip_dst.s_addr, (unsigned
short*)&tcp, sizeof(tcp));
    memcpy(packet + 20, &tcp, sizeof(tcp));

    if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
        perror("raw socket");
        exit(1);
    }

    if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
        perror("setsockopt");
        exit(1);
    }

    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = ip.ip_dst.s_addr;
    int c=0;
    while(c<10){
    c=c+1;
    if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0)  {
        perror("sendto");
        exit(1);
    }
    }
   // close(sd);
   // free(packet);
}

int main() {
    send_tcp_packet();
    return 0;
}
```

Tcp_data.c

```c
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

unsigned short in_cksum(unsigned short *addr, int len) {
    int nleft = len;
    int sum = 0;
    unsigned short *w = addr;
    unsigned short answer = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(unsigned char *)&answer = *(unsigned char *)w;
        sum += answer;
    }

    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += sum >> 16;
    answer = ~sum;
    return answer;
}

unsigned short in_cksum_tcp(int src, int dst, unsigned short *addr, int
len) {
    struct pseudo_header {
        struct in_addr src;
        struct in_addr dst;
        unsigned char pad;
        unsigned char proto;
        unsigned short tcp_len;
        struct tcphdr tcp;
    } psh;

    memset(&psh, 0, sizeof(psh));
    psh.src.s_addr = src;
    psh.dst.s_addr = dst;
    psh.pad = 0;
    psh.proto = IPPROTO_TCP;
    psh.tcp_len = htons(len);
    memcpy(&(psh.tcp), addr, len);
    return in_cksum((unsigned short *)&psh, 12 + len);
}

void send_tcp_packet() {
    int sd;
    const int on = 1;
    struct sockaddr_in sin;
    u_char *packet;
    // Increase the size of the packet to accommodate the "hello" message
    packet = (u_char *)malloc(60 + 5);  // 5 is the length of "hello"

    struct ip ip;
    struct tcphdr tcp;

    ip.ip_hl = 0x5;
    ip.ip_v = 0x4;
    ip.ip_tos = 0x0;
    // Update the IP length to include the TCP header and data
    ip.ip_len = 60 + 5;
    ip.ip_id = htons(12830);
    ip.ip_off = 0x0;
    ip.ip_ttl = 64;
    ip.ip_p = IPPROTO_TCP;
    ip.ip_sum = 0x0;
    ip.ip_src.s_addr = inet_addr("127.0.0.1");
    ip.ip_dst.s_addr = inet_addr("127.0.0.1");

    ip.ip_sum = in_cksum((unsigned short *)&ip, sizeof(ip));
    memcpy(packet, &ip, sizeof(ip));

    tcp.th_sport = htons(45512);
    tcp.th_dport = htons(23);
    tcp.th_seq = htonl(12345);
    tcp.th_ack = htonl(0);
    // Update the data offset to include the TCP header size
    tcp.th_off = 5 + 5;  // 5 is the length of "hello", each offset unit is
4 bytes
```

```c
    tcp.th_flags = TH_SYN;
    tcp.th_win = htons(5840);
    tcp.th_sum = 0;
    // Update the checksum calculation to include the data
    tcp.th_sum = in_cksum_tcp(ip.ip_src.s_addr, ip.ip_dst.s_addr, (unsigned
short*)&tcp, sizeof(tcp));
    memcpy(packet + 20, &tcp, sizeof(tcp));

    // Copy the "hello" message into the packet buffer
    memcpy(packet + 40, "hello", 5);

    if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
        perror("raw socket");
        exit(1);
    }

    if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
        perror("setsockopt");
        exit(1);
    }

    memset(&sin, 0, sizeof(sin));
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = ip.ip_dst.s_addr;

    int c = 0;
    while (c < 10) {
        c = c + 1;
        if (sendto(sd, packet, 60 + 5, 0, (struct sockaddr *)&sin,
sizeof(struct sockaddr)) < 0)  {
            perror("sendto");
            exit(1);
        }
    }

    close(sd);
    free(packet);
}

int main() {
    send_tcp_packet();
    return 0;
}
```

nmap help

```
 113  nmap --help
 114  sudo nmap -sS 10.10.100.177
 115  sudo nmap  10.10.100.177
 116  ifconfig
 117  sudo apt install net-tools
 118  ifconfig
 119  sudo apt update
 120  sudo apt upgrade
 121  sudo apt install wireshark
 122  sudo wiresshark
 123  wireshark
 124  nslookup
 125  ifconfig
 126  nmap 10.10.54.23*
 127  nmap 10.10.54.*
 128  sudo nmap 10.10.54.*
 129  nmap 10.10.54.*
 130  nmap 10.10.54.1-25
 131  nmap 10.10.54.224 -p 0-100
 132  nmap 10.10.54.224 -pn 0-100
 133* nmap 10
 134  nmap -Pn 10.10.54.224 -p80-100
 135  netstat -tl
 136  netstat -tl -n
 137  nmap 10.10.54.224 -p631
 138  nmap -sS 10.10.54.224
 139  sudo nmap -sS 10.10.54.224
 140  sudo nmap -sS 10.10.54.23
 141  sudo snap connect nmap:network-control
 142  sudo nmap -sS 10.10.54.224
 143  sudo nmap -sS 10.10.54.224 -p631
 144  telnet 10.10.54.224:53
 145  netstat -tl -n
 146  nc -help
 147  nc -l -p 500
 148  nc -l -p 1500
 149  netstat -tl -n
 150  nc -l -p 1500
 151  netstat -tl -n
 152  nc -l -p 1500
 153  netstat -tl -n
```

```
154  sudo nmap -sT 10.10.100.185 -p1500
155  nc -l -p 1500
```

udp_data .c

```c
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>

struct psd_udp {
	struct in_addr src;
	struct in_addr dst;
	unsigned char pad;
	unsigned char proto;
	unsigned short udp_len;
	struct udphdr udp;
};

unsigned short in_cksum(unsigned short *addr, int len) {
	int nleft = len;
	int sum = 0;
	unsigned short *w = addr;
	unsigned short answer = 0;

	while (nleft > 1) {
	sum += *w++;
	nleft -= 2;
	}

	if (nleft == 1) {
	*(unsigned char *)&answer = *(unsigned char *)w;
	sum += answer;
	}

	sum = (sum >> 16) + (sum & 0xFFFF);
	sum += sum >> 16;
	answer = ~sum;
```

```c
      return answer;
}

unsigned short in_cksum_udp(int src, int dst, unsigned short *addr, int
len) {
      struct psd_udp buf;

      memset(&buf, 0, sizeof(buf));
      buf.src.s_addr = src;
      buf.dst.s_addr = dst;
      buf.pad = 0;
      buf.proto = IPPROTO_UDP;
      buf.udp_len = htons(len);
      memcpy(&(buf.udp), addr, len);
      return in_cksum((unsigned short *)&buf, 12 + len);
}

void *run(void *arg) {
      struct ip ip;
      struct udphdr udp;
      int sd;
      const int on = 1;
      struct sockaddr_in sin;

      u_char *packet;
      packet = (u_char *)malloc(60);

      ip.ip_hl = 0x5;
      ip.ip_v = 0x4;
      ip.ip_tos = 0x0;
      ip.ip_len = 60;
      ip.ip_id = htons(12830);
      ip.ip_off = 0x0;
      ip.ip_ttl = 64;
      ip.ip_p = IPPROTO_UDP;
      ip.ip_sum = 0x0;
      ip.ip_src.s_addr = inet_addr("127.0.0.1");
      ip.ip_dst.s_addr = inet_addr("127.0.0.1");

      ip.ip_sum = in_cksum((unsigned short *)&ip, sizeof(ip));
      memcpy(packet, &ip, sizeof(ip));

      udp.uh_sport = htons(45512);
```

```
	udp.uh_dport = htons(23);
	udp.uh_ulen = htons(14);  // Length of UDP header + message
	udp.uh_sum = in_cksum_udp(ip.ip_src.s_addr, ip.ip_dst.s_addr,
(unsigned short *)&udp, sizeof(udp));

	// Copy the UDP header
	memcpy(packet + 20, &udp, sizeof(udp));

	// Add the message "hello" after the UDP header
	const char *message = "hello";
	memcpy(packet + 28, message, strlen(message));

	// fill IP and UDP here

	if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)) < 0) {
	perror("raw socket");
	exit(1);
	}

	if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
	perror("setsockopt");
	exit(1);
	}

	memset(&sin, 0, sizeof(sin));
	sin.sin_family = AF_INET;
	sin.sin_addr.s_addr = ip.ip_dst.s_addr;
	int c = 0;
	while (1) {
	// c+=1;

	if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0) {
		perror("sendto");
		exit(1);
	}
	}
}

int main(int argc, char **argv) {
	run(NULL);
	return 0;
}
```