UDP:

- User Datagram Protocol is described in RFC768.
- It implements a connectionless, unreliable datagram packet ser vice.
- Packets may be reordered or duplicated before they arrive.
- UDP generates and checks checksums to catch transmission errors.

When a UDP socket is created, its local and remote addresses are unspecified. Datagrams can be sent immediately using sendto() or sendmsg() with a valid destination address as an argument.

When connect is called on the socket the default destination address is set and datagrams can now be sent using send() or write() without specifying an destination address.

In order to receive packets the socket can be bound to an local address first by using bind. Otherwise the socket layer will automatically assign a free local port out of the range defined by net.ipv4.ip_local_port_range and bind the socket to INADDR_ANY.

All receive operations return only one packet. When the packet is smaller than the passed buffer only that much data is returned, when it is bigger the packet is truncated and the MSG_TRUNC flag is set. MSG_WAITALL is not supported.

When the MSG_DONTROUTE flag is set on sending, the destination address must refer to a local interface address and the packet is only sent to that interface.

UDP fragments a packet when its total length exceeds the interface MTU (Maximum Transmission Unit). A more network friendly alternative is to use path MTU discovery (refer to IP_MTU_DISCOVER section of ip(7)).

/* Sample UDP Server*/
```
#include <sys/socket.h>
#include <sys/types.h>
#include <resolv.h>
#include <unistd.h>
#include <stdio.h>

main() {
  int sock, length;
  struct sockaddr_in server;
  int msgsock;
  char buf[1024];
  socklen_t len_t;
  int rval;
  int i;

  sock = socket (AF_INET, SOCK_DGRAM, 0);
  if (sock < 0) {
    perror("opening stream socket");
  }
  server.sin_family = AF_INET;
  server.sin_addr.s_addr = INADDR_ANY;
  server.sin_port = 8889;

  if (bind (sock, (struct sockaddr *)&server, sizeof server) < 0) {
    perror ("binding stream socket");
  }
  len_t = sizeof (struct sockaddr);
  if ((rval = recvfrom(sock, buf, sizeof (buf), 0, (struct sockaddr
         *)&server, &len_t)) < 0){
    perror("reading socket");
  }else  {
    printf("%s\n",buf);
    printf("%d %d\n",server.sin_addr, server.sin_port);
  }
  close (sock);
}
```

```
/*Sample UDP Client*/
#include <sys/socket.h>
#include <sys/types.h>
#include <resolv.h>
#include <unistd.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

main() {
  int sock;
  struct sockaddr_in server;
  int msgsock;
  char buf[1024];
  struct hostent *hp;
  char *host = "127.0.0.1";
  int rval;

  sock = socket (AF_INET, SOCK_DGRAM, 0);
  if (sock < 0) {
    perror("opening stream socket");
  }

  bzero(&server, sizeof(server));
  hp = gethostbyname("localhost");
  bcopy((char*)hp->h_addr, (char*)&server.sin_addr, hp->h_length);
  server.sin_family = AF_INET;
  server.sin_port = 8889;
```

```
 strcpy(buf,"this is a test\n");
 if ((rval = sendto(sock, buf, sizeof (buf),0,(struct sockaddr *)&server, sizeof server)) <
 0){
   perror("writing socket");
 }
 close (sock);
}
```