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");
  }
```

UDP:

- 用户数据报协议在RFC768中描述。
- 它实现了一种无连接的、不可靠的数据报分组服务。
- 数据包在到达时可能被重新排序或重复。
- UDP生成并校验校验和以捕获传输错误。

创建UDP套接字时，其本地和远程地址未指定。可以立即使用sendto()或sendmsg()并传入有效目标地址作为参数来发送数据报。

当在套接字上调用connect时，默认目标地址被设置，现在可以使用send()或write()发送数据报而无需指定目标地址。

为了接收数据包，套接字可以首先通过使用 bind 绑定到一个本地地址。否则，套接层将自动从 net.ipv4.ip_local_port_range 定义的范围内选择一个空闲的本地端口，并将套接字绑定到 INADDR_ANY。

所有接收操作仅返回一个数据包。当数据包小于传入的缓冲区时，只返回对应大小的数据；当数据包更大时，数据包会被截断，并设置 MSG_TRUNC 标志。不支持 MSG_WAITALL。

在发送时若设置了 MSG_DONTROUTE 标志，则目标地址必须指向一个本地接口地址，且该数据包只会被发送到该接口。

当数据包的总长度超过接口的 MTU（最大传输单元）时，UDP 会对其分片。一种对网络更友好的替代方法是使用路径 MTU 发现（参见 ip(7) 中的 IP_MTU_DISCOVER 部分）。

```
/* 示例 UDP 服务器 */
#include <sys/socket.h>
#include <sys/types.h>
#include <resolv.h>
#include <unistd.h>
#include <stdio.h>

main() {
  int 套接字, 长度;
  struct sockaddr_in server;
  int msgsock;
  字符 缓冲区[1024];
  socklen_t len_t;
  int rval;
  int i;

  套接字 = socket (AF_INET, SOCK_DGRAM, 0);
  如果 (袜子 < 0) {
    perror("打开流式套接字");
  }
  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;
```

```
  server.sin_family = AF_INET;
  server.sin_addr.s_addr = INADDR_ANY;
  server.sin_port = 8889;

  如果 (bind (sock, (struct sockaddr *)&server, sizeof server) < 0) {
    perror ("绑定流式套接字");
  }
  len_t = sizeof (struct sockaddr);
  如果 ((rval = recvfrom(sock, buf, sizeof (buf), 0, (struct sockaddr
         *)&server, &len_t)) < 0){
    perror("读取套接字");
  }否则  {
    printf("%s\n",buf);
    printf("%d %d\n",server.sin_addr, server.sin_port);
  }
  close (sock);
}
```

```
/*示例UDP客户端*/
#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("打开流式套接字");
 }

 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);
}
```

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); }