NETWORK PROGRAMMING

ELEMENTARY UDP SOCKETS

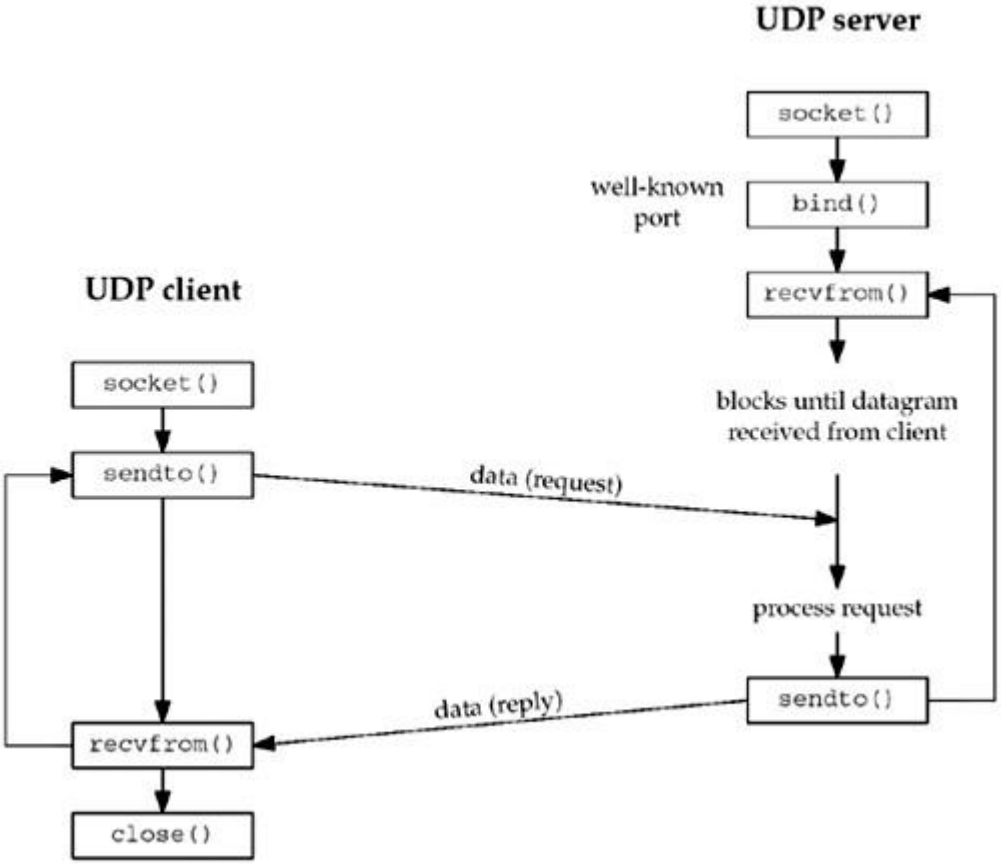
BY

MR.PRASAD SAWANT

ELEMENTARY UDP SOCKETS
CONNECTIONLESS, UNRELIABLE, DATAGRAM

- *recvfrom* and *sendto* functions
- UDP echo server
- UDP echo client
- Verify received responses
- *connect* function with UDP
- Rewrite *dg_cli* function with *connect*
- Lack of flow control with UDP
- Determine outgoing interface with UDP
- TCP and UDP echo server using *select*

SOCKET FUNCTIONS FOR UDP CLIENT/SERVER.

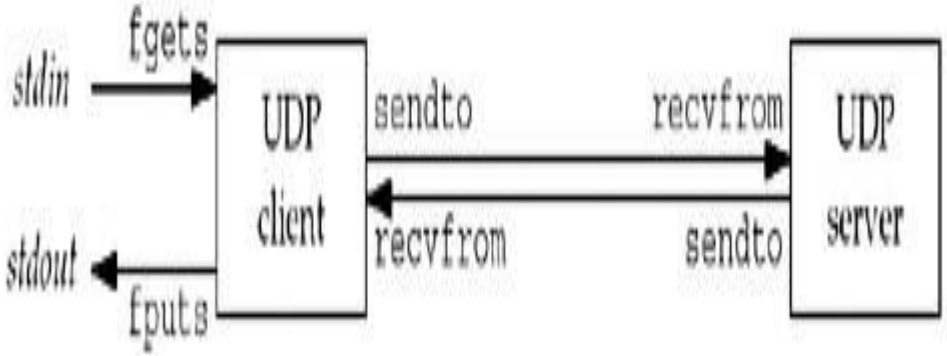


Recvfrom() and sendto()

```
#include <sys/socket.h>

ssize_t recvfrom(int sockfd, void *buff, size_t nbytes, int flags, struct
sockaddr *from, socklen_t *addrlen);

ssize_t sendto(int sockfd, const void *buff, size_t nbytes, int flags, const
struct sockaddr *to, socklen_t addrlen);
```

Both return: number of bytes read or written if OK, –1 on error

udp echo server: main function



udp echo server: main function

```
#include    "unp.h"

int
main(int argc, char **argv)
{
    int     sockfd;
    struct sockaddr_in servaddr, cliaddr;

    sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);

    Bind(sockfd, (SA *) &servaddr, sizeof(servaddr));

    dg_echo(sockfd, (SA *) &cliaddr, sizeof(cliaddr));
}
```

Create UDP Socket

Perform Server Processing

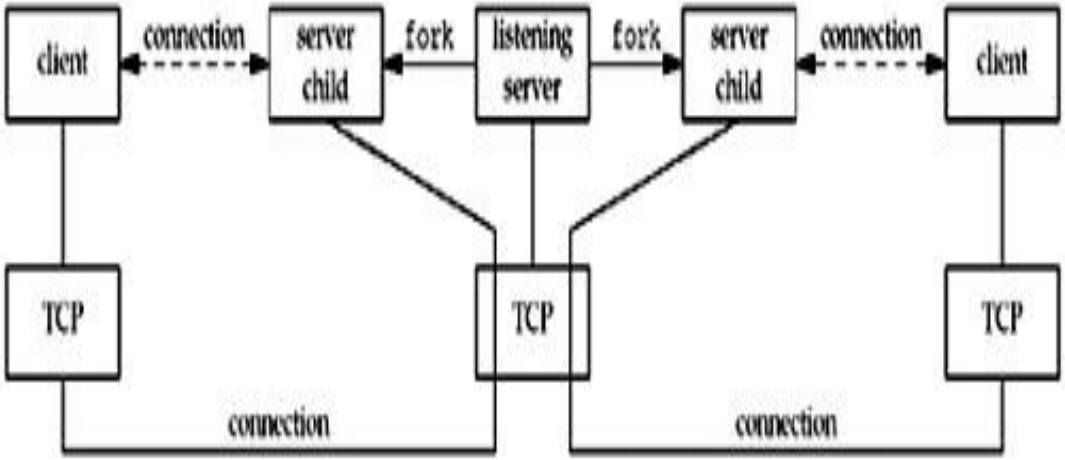
udp echo server: dg_echo function

lib/dg_echo.c

```
#include     "unp.h"

void
dg_echo(int sockfd, SA *pcliaddr, socklen_t clilen)
{
    int     n;
    socklen_t len;
    char    mesg[MAXLINE];

    for ( ; ; ) {
        len = clilen;
        n = Recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr, &len);

        Sendto(sockfd, mesg, n, 0, pcliaddr, len);
    }
}
```

reads the next datagram arriving at the server's port using recvfrom and sends it back using sendto

SUMMARY OF TCP CLIENT/SERVER WITH TWO CLIENTS.



UDP ECHO CLIENT: main FUNCTION

```
#include     "unp.h"

int
main(int argc, char **argv)
{
    int     sockfd;
    struct sockaddr_in servaddr;

    if(argc != 2)
       err_quit("usage: udpcli <IPaddress>");

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    Inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

    sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

    dg_cli(stdin, sockfd, (SA *) &servaddr, sizeof(servaddr));

    exit(0);
}
```

IPv4 socket address structure is filled in with the IP address and port number of the server

A UDP socket is created and the function dg_cli is called

UDP ECHO CLIENT: dg_cli FUNCTION

```
1 #include     "unp.h"

2 void
3 dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
4 {
5     int     n;
6     char    sendline[MAXLINE], recvline[MAXLINE + 1];

7     while (Fgets(sendline, MAXLINE, fp) != NULL) {

8         Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);

9         n = Recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);

10        recvline[n] = 0;        /* null terminate */
11        Fputs(recvline, stdout);
12    }
13 }
```

Line 7-12
read a line from standard input using fgets,
send the line to the server using sendto,
read back the server's echo using recvfrom,
and print the echoed line to standard output using fputs.

EXECUTION

```
root@localhost:~/Desktop/NP prasad/unpv13e/unpv13e/udpcliserv
File  Edit  View  Search  Terminal  Help
[root@localhost udpcliserv]# ./udpserv01 &
[3] 2628
[root@localhost udpcliserv]# ./udpcli01 127.0.0.1
hi
hi
prasad
prasad
pccc
pccc
```

PROBLEMS WITH UDP SOCKETS

Lost datagrams (client request or server reply):

- recvfrom blocks forever
- place a timeout, but don't know whether request or reply gets lost

Malicious datagrams inter-mixed with server replies:

- ignore any received datagrams not from that server
- allocate another socket address structure and compare returned address

PROBLEMS WITH UDP SOCKETS (CONT.)

For multi-homed server, verifying address may not work (server reply may go through another outgoing interface)

- solution 1: verify server domain name, instead
- solution 2: multi-homed UDP server creates one socket for every interface (IP addr), bind IP addresses to sockets, use *select* across all sockets

PROBLEMS WITH UDP SOCKETS (CONT.)

Server not running:

- ICMP port unreachable error (asynchronous error)
- asynchronous errors not returned for UDP sockets unless the socket has been connected (reason?: considering a client sending 3 datagrams to 3 servers, recvfrom has no way to know the destination of the datagram causing the error)
- recvfrom blocks forever
- solution: call *connect* on a UDP socket

Server Not Running

```
root@localhost:~/Desktop/NP prasad/unpv13e/unpv13e/udpcliserv
File  Edit  View  Search  Terminal  Help
[root@localhost udpcliserv]# ./udpcli02 127.0.0.1
hi
test
where is server ?
Oh.. Since server is not working thats why we are not able to see the echo
```

PROBLEMS WITH UDP SOCKETS (CONT.)

Lack of flow control:

- considering dg_cli in a client sendto 2000 1400-byte datagrams to the server
- client may overrun the server (e.g. 96% loss rate: mostly lost due to receive buffer overflow, some due to network congestion)
- use netstat -s to check the loss
- solution: use SO_RCVBUF option to enlarge buffer, use request-reply model instead of bulk transfer

Verifying Received Response

Recall Udp client main function we just replace the assignment

```
servaddr.sin_port = htons(SERV_PORT);
```

with

```
servaddr.sin_port = htons(7);
```

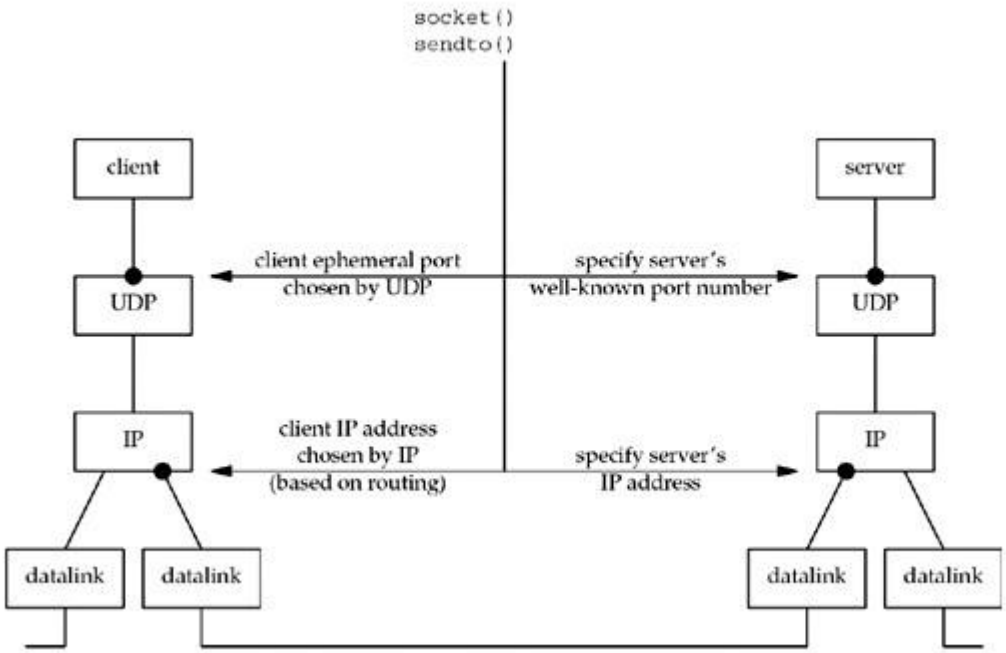
We do this so we can use any host running the standard echo server with our client.

version of dg_cli that verifies returned socket address

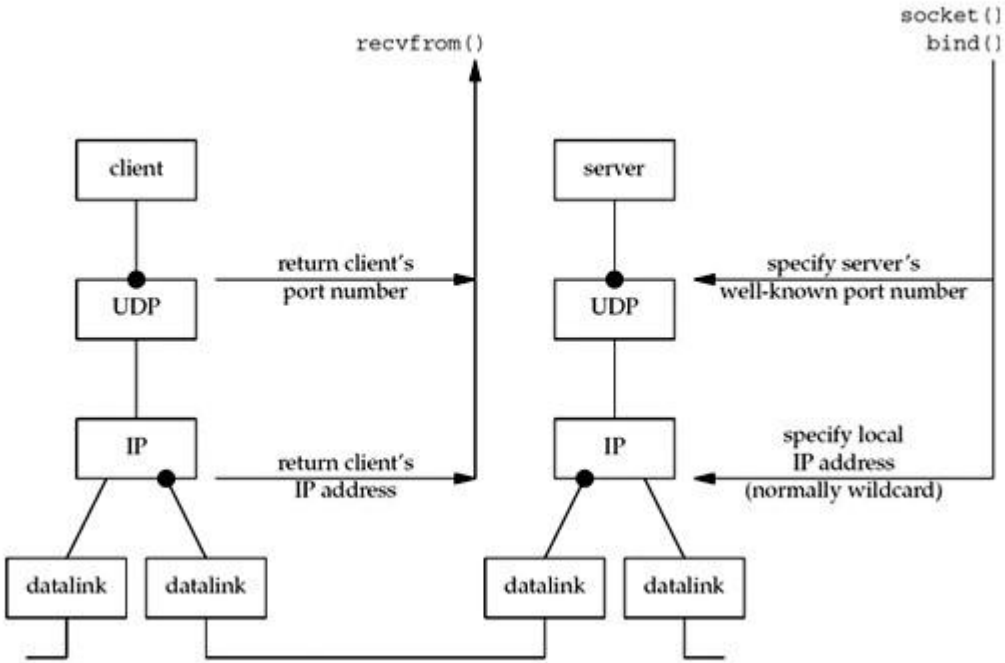
```
 1 #include     "unp.h"

 2 void
 3 dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
 4 {
 5     int     n;
 6     char    sendline[MAXLINE], recvline[MAXLINE + 1];
 7     socklen_t len;
 8     struct sockaddr *preply_addr;

 9     preply_addr = Malloc(servlen);

10     while (Fgets(sendline, MAXLINE, fp) != NULL) {

11         Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);

12         len = servlen;
13         n = Recvfrom(sockfd, recvline, MAXLINE, 0, preply_addr, &len);
14         if (len != servlen || memcmp(pservaddr, preply_addr, len) != 0) {
15             printf("reply from %s (ignored)\n", Sock_ntop(preply_addr, len));
16             continue;
17         }

18         recvline[n] = 0;      /* null terminate */
19         Fputs(recvline, stdout);
20     }
21 }
```

Line no 12- 18 value-result argument and then compare the socket address structures themselves using memcmp

SUMMARY OF UDP CLIENT/SERVER FROM CLIENT'S PERSPECTIVE



SUMMARY OF UDP CLIENT/SERVER FROM SERVER'S PERSPECTIVE.



Information available to server from arriving IP datagram.

From client's IP datagram	TCP server	UDP server
Source IP address	`accept`	`recvfrom`
Source port number	`accept`	`recvfrom`
Destination IP address	`getsockname`	`recvmsg`
Destination port number	`getsockname`	`getsockname`

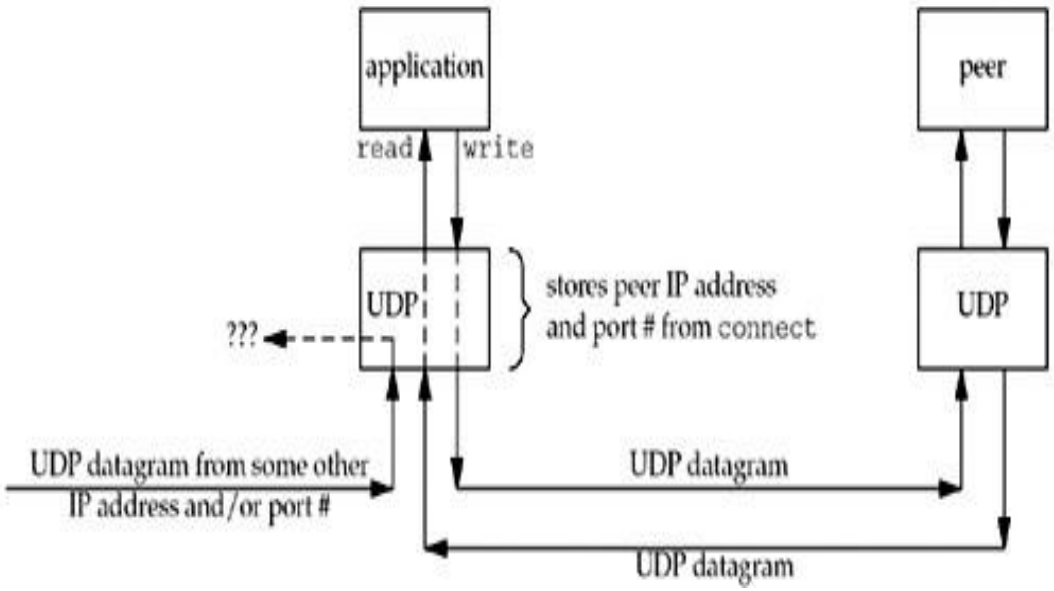
connect FUNCTION WITH UDP

1. An unconnected UDP socket, the default when we create a UDP socket

2. A connected UDP socket, the result of calling connect on a UDP socket

connect FUNCTION WITH UDP

Figure 8.15. Connected UDP socket.



dg_cli function (revisited)using connect

```
#include     "unp.h"

void
dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
    int     n;
    char    sendline[MAXLINE], recvline[MAXLINE + 1];

    Connect(sockfd, (SA *) pservaddr, servlen);

    while (Fgets(sendline, MAXLINE, fp) != NULL) {

        Write(sockfd, sendline, strlen(sendline));

        n = Read(sockfd, recvline, MAXLINE);

        recvline[n] = 0;        /* null terminate */
        Fputs(recvline, stdout);
    }
}
```

call to connect and replacing the calls to sendto and recvfrom with calls to write and read.

root@localhost:~/Desktop/NP prasad/unpv13e/unpv13e/udpcliserv

File Edit View Search Terminal Help

```
[root@localhost udpcliserv]# ./udpcli04 127.0.0.1
hi pcccs
read error: Connection refused
[root@localhost udpcliserv]#
```

LACK OF FLOW CONTROL WITH UDP
dg_cli FUNCTION THAT WRITES A FIXED NUMBER OF DATAGRAMS TO THE SERVER.

udpcliserv/dgcliloop1.c

```
#include     "unp.h"

#define NDG      2000          /*  datagrams to send */
#define DGLEN    1400          /*  length of each datagram */

void
dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
    int     i;
    char    sendline[DGLEN];

    for (i = 0; i < NDG; i++) {
        Sendto(sockfd, sendline, DGLEN, 0, pservaddr, servlen);
    }
}
```

dg_echo FUNCTION THAT COUNTS RECEIVED DATAGRAMS

```
#include     "unp.h"

static void recvfrom_int(int);
static int count;

void
dg_echo(int sockfd, SA *pcliaddr, socklen_t clilen)
{
    socklen_t len;
    char    mesg[MAXLINE];

    Signal(SIGINT, recvfrom_int);

    for ( ; ; ) {
        len = clilen;
        Recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr, &len);

        count++;
    }
}

static void
recvfrom_int(int signo)
{
    printf("\nreceived %d datagrams\n", count);
    exit(0);
}
```

DETERMINING OUTGOING INTERFACE WITH UDP

```
 1 #include     "unp.h"

 2 int
 3 main(int argc, char **argv)
 4 {
 5     int     sockfd;
 6     socklen_t len;
 7     struct sockaddr_in cliaddr, servaddr;

 8     if (argc != 2)
 9         err_quit("usage: udpcli <IPaddress>");

10     sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

11     bzero(&servaddr, sizeof(servaddr));
12     servaddr.sin_family = AF_INET;
13     servaddr.sin_port = htons(SERV_PORT);
14     Inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

15     Connect(sockfd, (SA *) &servaddr, sizeof(servaddr));

16     len = sizeof(cliaddr);
17     Getsockname(sockfd, (SA *) &cliaddr, &len);
18     printf("local address %s\n", Sock_ntop((SA *) &cliaddr, len));

19     exit(0);
20 }
```

Line 17
calls getsockname, printing the local IP address and port.

IF WE RUN THE PROGRAM ON THE MULTIHOMED HOST LINUX , WE HAVE THE FOLLOWING OUTPUT:

```
root@localhost:~/Desktop/NP prasad/unpv13e/unpv13e/udpcliserv
File  Edit  View  Search  Terminal  Help
[root@localhost udpcliserv]# ./udpcli09 127.0.0.1
local address 127.0.0.1:35741
[root@localhost udpcliserv]#
```