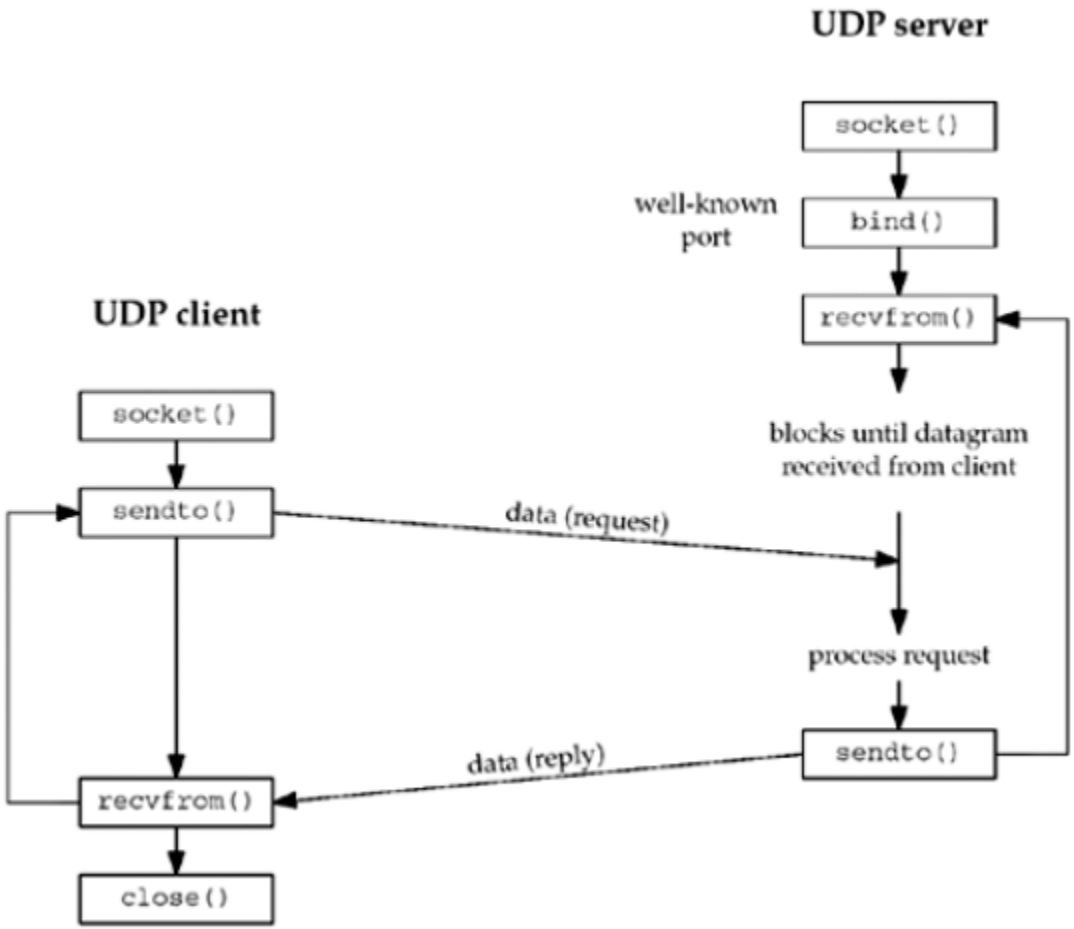
Elementary UDP Sockets

Socket functions for UDP client/server



recvfrom and *sendto* Functions

- These two functions are similar to the standard read and write functions, but three additional arguments are required

```
#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

- The first three arguments, *sockfd, buff*, and *nbytes*, are identical to the first three arguments for read and write: descriptor, pointer to buffer to read into or write from, and number of bytes to read or write

```
#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

- For sendto: a socket address structure containing the protocol address (e.g., IP address and port number) of where the data is to be sent
- The size of this socket address structure is specified by *addrlen*
- For recvfrom: function fills in the socket address structure pointed to by *from* with the protocol address of who sent the datagram
- The number of bytes stored in this socket address structure is also returned to the caller in the integer pointed to by *addrlen*
- For now, we will always set the *flags* to 0

Simple echo client/server using UDP

UDP echo server

```
#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));
}
```

dg_echo Function

```
#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);
    }
}
```

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);
}
```

UDP Echo Client: dg_cli Function

```
#include "unp.h"
void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
{
    int n;
    char sendline[MAXLINE], recvline[MAXLINE + 1];
    while (Fgets(sendline, MAXLINE, fp) != NULL) {
        Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);
        n = Recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
        recvline[n] = 0; /* null terminate */
        Fputs(recvline, stdout);
    }
}
```

Thank you