8.2 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.

We will describe the *flags* argument in Chapter 14 when we discuss the recv, send, recvmsg, and sendmsg functions, since we do not need them with our simple UDP client/server example in this chapter. For now, we will always set the *flags* to 0.

The *to* argument for sendto is 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*. The 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*. Note that the final argument to sendto is an integer value, while the final argument to recvfrom is a pointer to an integer value (a value-result argument).

The final two arguments to recvfrom are similar to the final two arguments to accept: The contents of the socket address structure upon return tell us who sent the datagram (in the case of UDP) or who initiated the connection (in the case of TCP). The final two arguments to sendto are similar to the final two arguments to connect: We fill in the socket address structure with the protocol address of where to send the datagram (in the case of UDP) or with whom to establish a connection (in the case of TCP).

Both functions return the length of the data that was read or written as the value of the function. In the typical use of recvfrom, with a datagram protocol, the return value is the amount of user data in the datagram received.

Writing a datagram of length 0 is acceptable. In the case of UDP, this results in an IP datagram containing an IP header (normally 20 bytes for IPv4 and 40 bytes for IPv6), an 8-byte UDP header, and no data. This also means that a return value of 0 from recvfrom is acceptable for a datagram protocol: It does not mean that the peer has closed the connection, as does a return value of 0 from read on a TCP socket. Since UDP is connectionless, there is no such thing as closing a UDP connection.

If the *from* argument to recvfrom is a null pointer, then the corresponding length argument (*addrlen*) must also be a null pointer, and this indicates that we are not interested in knowing the protocol address of who sent us data.

Both recvfrom and sendto can be used with TCP, although there is normally no reason to do so.