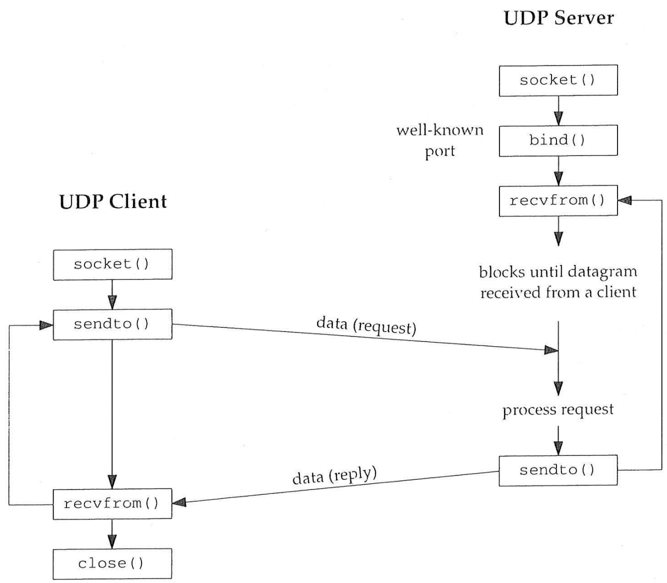

Figure 8.1 Socket functions for UDP client–server.

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.

TCP and UDP

Figures from:

UNIX NETWORK PROGRAMMING

W. R. Stevens

Vol. 1 sec. ed.

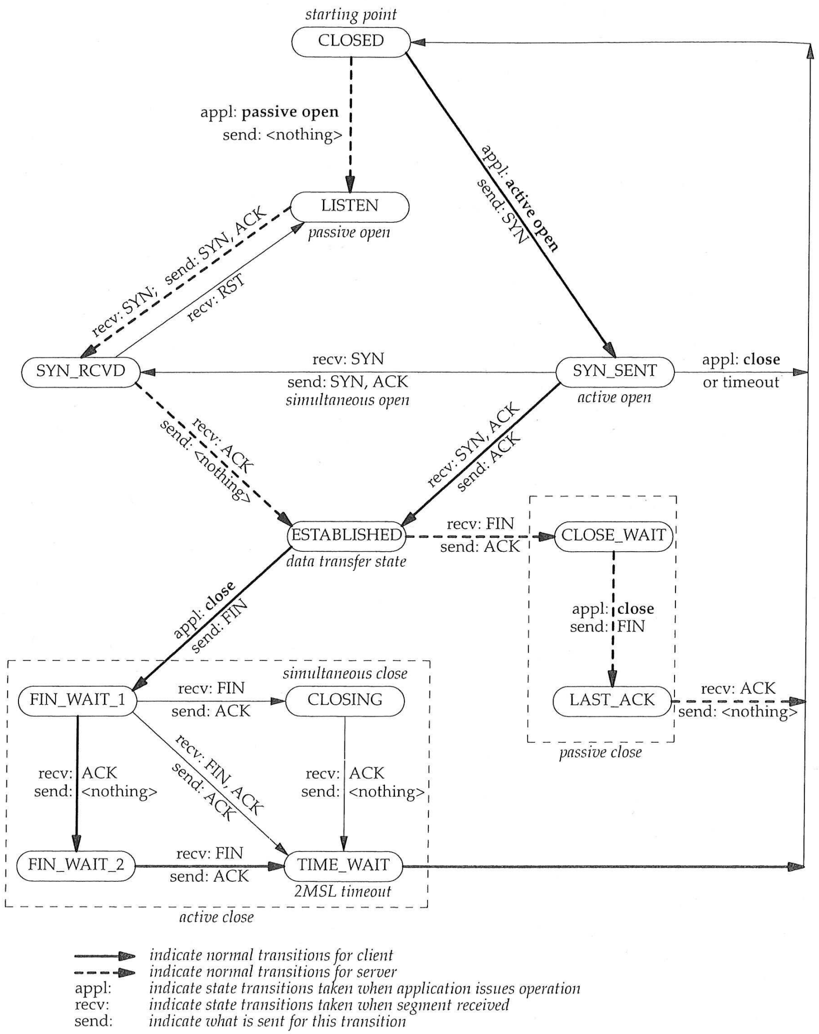

Figure 2.4 TCP state transition diagram.

Watching the Packets

Figure 2.5 shows the actual packet exchange that takes place for a complete TCP connection: the connection establishment, data transfer, and connection termination. We also show the TCP states through which each endpoint passes.

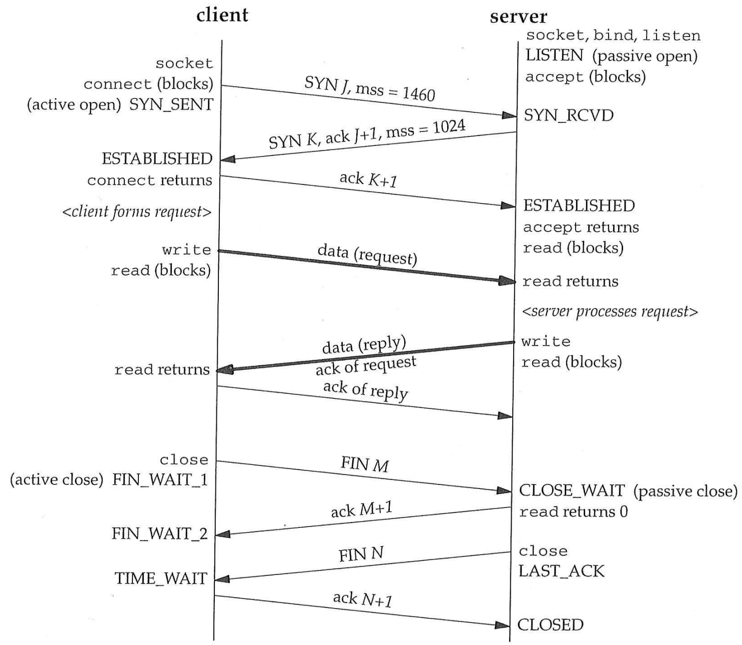

Figure 2.5 Packet exchange for TCP connection.

The client in this example announces an MSS of 1460 (typical for IPv4 on an Ethernet) and the server announces an MSS of 1024 (typical for older Berkeley-derived implementations on an Ethernet). It is OK for the MSS to be different in each direction. (See also Exercise 2.5.)

Once the connection is established, the client forms a request and sends it to the server. We assume this request fits into a single TCP segment (i.e., less than 1024 bytes given the server's announced MSS). The server processes the request and sends a reply, and we assume that the reply fits in a single segment (less than 1460 in this example).