10.3 SCTP One-to-Many-Style Streaming Echo Client: `main` Function

Figure 10.3 shows our SCTP client main function.

Validate arguments and create a socket

9–15 The client validates the arguments passed to it. First, the client verifies that the caller provided a host to send messages to. It then checks if the "echo to all" option is being enabled (we will see this used in Section 10.5). Finally, the client creates an SCTP one-to-many-style socket.

Set up server address

16–20 The client translates the server address, passed on the command line, using the `inet_pton` function. It combines that with the server's well-known port number and uses the resulting address as the destination for the requests.

Set up for notifications of interest

21–23 The client explicitly sets the notification subscription provided by our one-to-many SCTP socket. Again, it wants no `MSG_NOTIFICATION` events. Therefore, the client turns these off (as was done in the server) and only enables the receipt of the `sctp_sndrcvinfo` structure.

Call echo processing function

24–28 If the `echo_to_all` flag is not set, the client calls the `sctpstr_cli` function, discussed in Section 10.4. If the `echo_to_all` flag is set, the client calls the `sctpstr_cli_echoall` function. We will discuss this function in Section 10.5 as we explore uses for SCTP streams.

Figure 10.3 SCTP streaming echo client main.

sctp/sctpclient01.c

```
 1 #include     "unp.h"

 2 int
 3 main(int argc, char **argv)
 4 {
 5     int     sock_fd;
 6     struct sockaddr_in servaddr;
 7     struct sctp_event_subscribe evnts;
 8     int     echo_to_all = 0;
 9     if (argc < 2)
10         err_quit("Missing host argument - use '%s host [echo] '\n", argv[0]) ;
11     if (argc > 2) {
12         printf("Echoing messages to all streams\n") ;
13         echo_to_all = 1;
14     }
15     sock_fd = Socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
16     bzero(&servaddr, sizeof (servaddr) ) ;
17     servaddr.sin_family = AF_INET;
18     servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
19     servaddr.sin_port = htons (SERV_PORT);
20     Inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

21     bzero(&evnts, sizeof (evnts)) ;
22     evnts.sctp_data_io_event = 1 ;
23     Setsockopt(sock_fd, IPPROTO_SCTP, SCTP_EVENTS, &evnts, sizeof (evnts)) ;
24     if (echo_to_all == 0)
25         sctpstr_cli (stdin, sock_fd, (SA *) &servaddr, sizeof (servaddr)) ;
26     else
27         sctpstr_cli_echoall(stdin, sock_fd, (SA *) &servaddr,
28                             sizeof (servaddr)) ;
29     Close (sock_fd) ;
30     return (0) ;
31 }
```

Finish up

29–31 On return from processing, the client closes the SCTP socket, which shuts down any SCTP associations using the socket. The client then returns from main with a return code of 0, indicating that the program ran successfully.