8.14 Determining Outgoing Interface with UDP

A connected UDP socket can also be used to determine the outgoing interface that will be used to a particular destination. This is because of a side effect of the `connect` function when applied to a UDP socket: The kernel chooses the local IP address (assuming the process has not already called `bind` to explicitly assign this). This local IP address is chosen by searching the routing table for the destination IP address, and then using the primary IP address for the resulting interface.

Figure 8.23 shows a simple UDP program that `connects` to a specified IP address and then calls `getsockname`, printing the local IP address and port.

Figure 8.23 UDP program that uses `connect` to determine outgoing interface.

udpcliserv/udpcli09.c

```
 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 }
```

If we run the program on the multihomed host `freebsd`, we have the following output:

```
freebsd % udpcli09 206.168.112.96
local address 12.106.32.254:52329

freebsd % udpcli09 192.168.42.2
local address 192.168.42.1:52330

freebsd % udpcli09 127.0.0.1
local address 127.0.0.1:52331
```

The first time we run the program, the command-line argument is an IP address that follows the default route. The kernel assigns the local IP address to the primary address of the interface to which the default route points. The second time, the argument is the IP address of a system connected to a second Ethernet interface, so the kernel assigns the local IP address to the primary address of this second interface. Calling `connect` on a UDP socket does not send anything to that host; it is entirely a local operation that saves the peer's IP address and port. We also see that calling `connect` on an unbound UDP socket also assigns an ephemeral port to the socket.

> Unfortunately, this technique does not work on all implementations, mostly SVR4-derived kernels. For example, this does not work on Solaris 2.5, but it works on AIX, HP-UX 11, MacOS X, FreeBSD, Linux, and Solaris 2.6 and later.