8.9 Server Not Running

The next scenario to examine is starting the client without starting the server. If we do so and type in a single line to the client, nothing happens. The client blocks forever in its call to `recvfrom`, waiting for a server reply that will never appear. But, this is an example where we need to understand more about the underlying protocols to understand what is happening to our networking application.

First we start `tcpdump` on the host `macosx`, and then we start the client on the same host, specifying the host `freebsd4` as the server host. We then type a single line, but the line is not echoed.

```
macosx % udpcli01 172.24.37.94
hello, world
```

we type this line but nothing is echoed back

Figure 8.10 shows the `tcpdump` output.

Figure 8.10 `tcpdump` output when server process not started on server host.

```
1 0.0                       arp who-has freebsd4 tell macosx
2 0.003576 ( 0.0036)        arp reply freebsd4 is-at 0:40:5:42:d6:de

3 0.003601 ( 0.0000)        macosx.51139 > freebsd4.9877: udp 13
4 0.009781 ( 0.0062)        freebsd4 > macosx: icmp: freebsd4 udp port 9877 unreachable
```

First we notice that an ARP request and reply are needed before the client host can send the UDP datagram to the server host. (We left this exchange in the output to reiterate the potential for an ARP request-reply before an IP datagram can be sent to another host or router on the local network.)

In line 3, we see the client datagram sent but the server host responds in line 4 with an ICMP "port unreachable." (The length of 13 accounts for the 12 characters and the newline.) This ICMP error, however, is not returned to the client process, for reasons that we will describe shortly. Instead, the client blocks forever in the call to `recvfrom` in Figure 8.8. We also note that ICMPv6 has a "port unreachable" error, similar to ICMPv4 (Figures A.15 and A.16), so the results described here are similar for IPv6.

We call this ICMP error an *asynchronous error*. The error was caused by `sendto`, but `sendto` returned successfully. Recall from Section 2.11 that a successful return from a UDP output operation only means there was room for the resulting IP datagram on the interface output queue. The ICMP error is not returned until later (4 ms later in Figure 8.10), which is why it is called asynchronous.

The basic rule is that an asynchronous error is not returned for a UDP socket unless the socket has been connected. We will describe how to call `connect` for a UDP socket in Section 8.11. Why this design decision was made when sockets were first implemented is rarely understood. (The implementation implications are discussed on pp. 748–749 of TCPv2.)

Consider a UDP client that sends three datagrams in a row to three different servers (i.e., three different IP addresses) on a single UDP socket. The client then enters a loop that calls `recvfrom` to read the replies. Two of the datagrams are correctly delivered (that is, the server was running on two of the three hosts) but the third host was not running the server. This third host responds with an ICMP port unreachable. This ICMP error message contains the IP header and UDP header of the datagram that caused the error. (ICMPv4 and ICMPv6 error messages always contain the IP header and all of the UDP header or part of the TCP header to allow the receiver of the ICMP error to determine which socket caused the error. We will show this in Figures 28.21 and 28.22.) The client that sent the three datagrams needs to know the destination of the datagram that caused the error to distinguish which of the three datagrams caused the error. But how can the kernel return this information to the process? The only piece of information that `recvfrom` can return is an `errno` value; `recvfrom` has no way of returning the destination IP address and destination UDP port number of the datagram in error. The decision was made, therefore, to return these asynchronous errors to the process only if the process connected the UDP socket to exactly one peer.

> Linux returns most ICMP "destination unreachable" errors even for unconnected sockets, as long as the `SO_BSDCOMPAT` socket option is not enabled. All the ICMP "destination unreachable" errors from Figure A.15 are returned, except codes 0, 1, 4, 5, 11, and 12.
>
> We return to this problem of asynchronous errors with UDP sockets in Section 28.7 and show an easy way to obtain these errors on unconnected sockets using a daemon of our own.