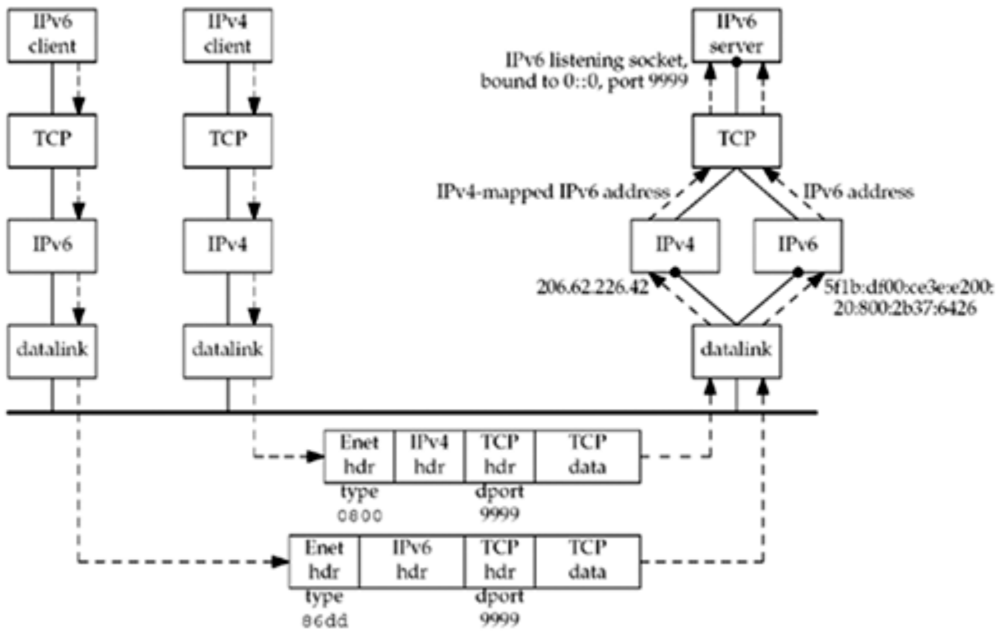
12.2 IPv4 Client, IPv6 Server

A general property of a dual-stack host is that IPv6 servers can handle both IPv4 and IPv6 clients. This is done using IPv4-mapped IPv6 addresses (Figure A.10). Figure 12.2 shows an example of this.

Figure 12.2. IPv6 server on dual-stack host serving IPv4 and IPv6 clients.



We have an IPv4 client and an IPv6 client on the left. The server on the right is written using IPv6 and it is running on a dual-stack host. The server has created an IPv6 listening TCP socket that is bound to the IPv6 wildcard address and TCP port 9999.

We assume the clients and server are on the same Ethernet. They could also be connected by routers, as long as all the routers support IPv4 and IPv6, but that adds nothing to this discussion. Section B.3 discusses a different case where IPv6 clients and servers are connected by IPv4-only routers.

We assume both clients send SYN segments to establish a connection with the server. The IPv4 client host will send the SYN in an IPv4 datagram and the IPv6 client host will send the SYN in an IPv6 datagram. The TCP segment from the IPv4 client appears on the wire as an Ethernet header followed by an IPv4 header, a TCP header, and the TCP data. The Ethernet header contains a type field of `0x0800`, which identifies the frame as an IPv4 frame. The TCP header contains the destination port of 9999. (Appendix A talks more about the formats and contents of these headers.) The destination IP address in the IPv4 header, which we do not show, would be 206.62.226.42.

The TCP segment from the IPv6 client appears on the wire as an Ethernet header followed by an IPv6 header, a TCP header, and the TCP data. The Ethernet header contains a type field of `0x86dd`, which identifies the frame as an IPv6 frame. The TCP header has the same format as the TCP header in the IPv4 packet and contains the destination port of 9999. The destination IP address in the IPv6 header, which we do not show, would be `5f1b:df00:ce3e:e200:20:800:2b37:6426`.

The receiving datalink looks at the Ethernet type field and passes each frame to the appropriate IP module. The IPv4 module, probably in conjunction with the TCP module, detects that the destination socket is an IPv6 socket, and the source IPv4 address in the IPv4 header is converted into the equivalent IPv4-mapped IPv6 address. That mapped address is returned to the IPv6 socket as the client's IPv6 address when `accept` returns to the server with the IPv4 client connection. All remaining datagrams for this connection are IPv4 datagrams.

When `accept` returns to the server with the IPv6 client connection, the client's IPv6 address does not change from whatever source address appears in the IPv6 header. All remaining datagrams for this connection are IPv6 datagrams.

We can summarize the steps that allow an IPv4 TCP client to communicate with an IPv6 server as follows:

1. The IPv6 server starts, creates an IPv6 listening socket, and we assume it `binds` the wildcard address to the socket.
2. The IPv4 client calls `gethostbyname` and finds an A record for the server. The server host will have both an A record and a AAAA record since it supports both protocols, but the IPv4 client asks for only an A record.
3. The client calls `connect` and the client's host sends an IPv4 SYN to the server.
4. The server host receives the IPv4 SYN directed to the IPv6 listening socket, sets a flag indicating that this connection is using IPv4-mapped IPv6 addresses, and responds with an IPv4 SYN/ACK. When the connection is established, the address returned to the server

by accept is the IPv4-mapped IPv6 address.

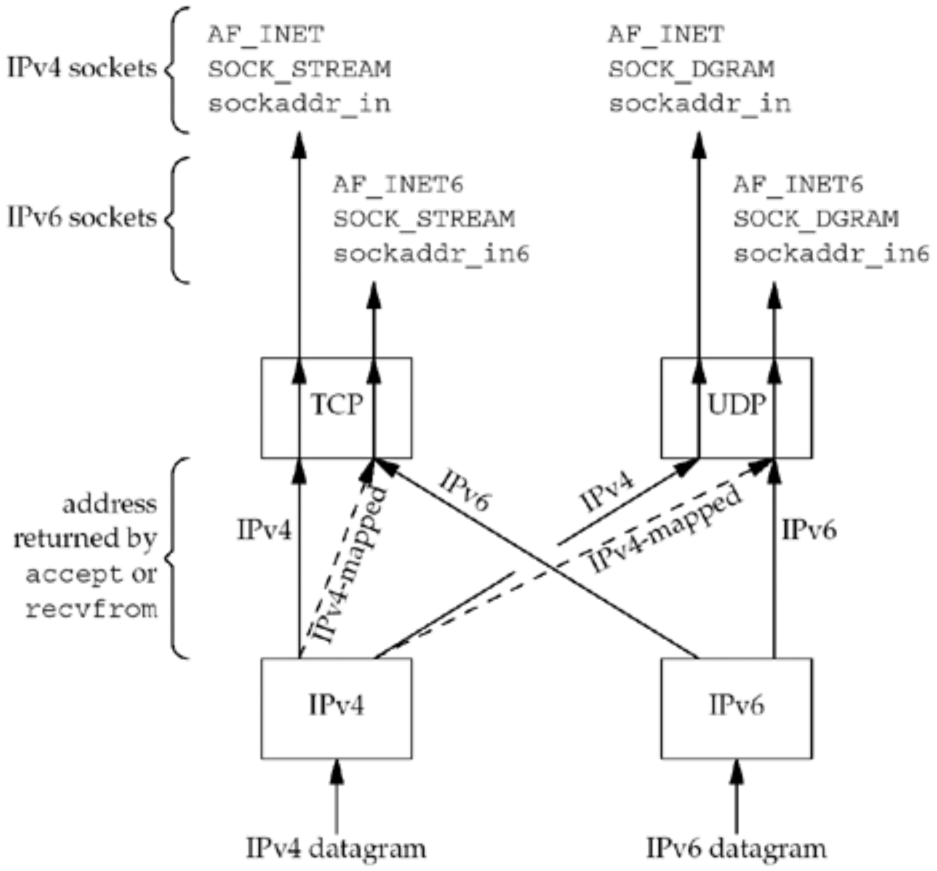
5. When the server host sends to the IPv4-mapped IPv6 address, its IP stack generates IPv4 datagrams to the IPv4 address. Therefore, all communication between this client and server takes place using IPv4 datagrams.
6. Unless the server explicitly checks whether this IPv6 address is an IPv4-mapped IPv6 address (using the IN6_IS_ADDR_V4MAPPED macro described in Section 12.4), the server never knows that it is communicating with an IPv4 client. The dual-protocol stack handles this detail. Similarly, the IPv4 client has no idea that it is communicating with an IPv6 server.

An underlying assumption in this scenario is that the dual-stack server host has both an IPv4 address and an IPv6 address. This will work until all the IPv4 addresses are taken.

The scenario is similar for an IPv6 UDP server, but the address format can change for each datagram. For example, if the IPv6 server receives a datagram from an IPv4 client, the address returned by recvfrom will be the client's IPv4-mapped IPv6 address. The server responds to this client's request by calling sendto with the IPv4-mapped IPv6 address as the destination. This address format tells the kernel to send an IPv4 datagram to the client. But the next datagram received for the server could be an IPv6 datagram, and recvfrom will return the IPv6 address. If the server responds, the kernel will generate an IPv6 datagram.

Figure 12.3 summarizes how a received IPv4 or IPv6 datagram is processed, depending on the type of the receiving socket, for TCP and UDP, assuming a dual-stack host.

Figure 12.3. Processing of received IPv4 or IPv6 datagrams, depending on type of receiving socket.



- If an IPv4 datagram is received for an IPv4 socket, nothing special is done. These are the two arrows labeled "IPv4" in the figure: one to TCP and one to UDP. IPv4 datagrams are exchanged between the client and server.
- If an IPv6 datagram is received for an IPv6 socket, nothing special is done. These are the two arrows labeled "IPv6" in the figure: one to TCP and one to UDP. IPv6 datagrams are exchanged between the client and server.
- When an IPv4 datagram is received for an IPv6 socket, the kernel returns the corresponding IPv4-mapped IPv6 address as the address returned by accept (TCP) or recvfrom (UDP). These are the two dashed arrows in the figure. This mapping is possible because an IPv4 address can always be represented as an IPv6 address. IPv4 datagrams are exchanged between the client and server.
- The converse of the previous bullet is false: In general, an IPv6 address cannot be represented as an IPv4 address; therefore, there are no arrows from the IPv6 protocol box to the two IPv4 sockets

Most dual-stack hosts should use the following rules in dealing with listening sockets:

1. A listening IPv4 socket can accept incoming connections from only IPv4 clients.
2. If a server has a listening IPv6 socket that has bound the wildcard address and the `IPV6_V6ONLY` socket option (Section 7.8) is not set, that socket can accept incoming connections from either IPv4 clients or IPv6 clients. For a connection from an IPv4 client, the server's local address for the connection will be the corresponding IPv4-mapped IPv6 address.
3. If a server has a listening IPv6 socket that has bound an IPv6 address other than an IPv4-mapped IPv6 address, or has bound the wildcard address but has set the `IPv6_V6ONLY` socket option (Section 7.8), that socket can accept incoming connections from IPv6 clients only.