Using Sockets and Socket Streams

This chapter describes ways to make socket connections that are completely under the control of your program. Most programs would be better served by higher-level APIs such as `NSURLConnection`, which was described in previous chapters. These APIs should be used only if you need to support some protocol other than the protocols supported by built-in Cocoa or Core Foundation functionality.

Choosing a Socket API

At almost every level of networking, software can be divided into two categories: clients (programs that connect to other apps) and services (programs that other apps connect to). At a high level, these lines are clear. Most programs written using high-level APIs are purely clients. At a lower level, however, the lines are often blurry.

Socket and stream programming generally falls into one of the following broad categories:

- Packet-based communication—Programs that operate on one packet at a time, listening for incoming packets, then sending packets in reply.

  With packet-based communication, the only differences between clients and servers are the contents of the packets that each program sends and receives, and (presumably) what each program does with the data. The networking code itself is identical.

- Stream-based clients—Programs that use TCP to send and receive data as two continuous streams of bytes, one in each direction.

  With stream-based communication, clients and servers are somewhat more distinct. The actual data handling part of clients and servers is similar, but the way that the program initially constructs the communication channel is very different.

The API you choose for socket-based connections depends on whether you are making a connection to another host or receiving a connection from another host. It also depends on whether you are using TCP or some other protocol. Here are a few factors to consider:

- In OS X, if you already have networking code that is shared with non-Apple platforms, you can use POSIX C networking APIs and continue to use your networking code as-is (on a separate thread). If your program is based on a Core Foundation or Cocoa (Foundation) run loop, you can also use the Core Foundation `CFStream` API to integrate the POSIX networking code into your overall architecture on the main thread. Alternatively, if you are using Grand Central Dispatch (GCD), you can add a socket as a dispatch source.

  In iOS, POSIX networking is discouraged because it does not activate the cellular radio or on-demand VPN. Thus, as a general rule, you should separate the networking code from any common data processing functionality and rewrite the networking code using higher-level APIs.

  > **Note:** If you use POSIX networking code, you should be aware that the POSIX networking API is not protocol-agnostic (you must handle some of the differences between IPv4 and IPv6 yourself). It is a connect-by-IP API rather than a connect-by-name API, which means that you must do a lot of extra work if you want to achieve the same initial-connection performance and robustness that higher-level APIs give you for free. Before you decide to reuse existing POSIX networking code, be sure to read Avoid Resolving DNS Names Before Connecting to a Host in Avoiding Common Networking Mistakes.

- For daemons and services that listen on a port, or for non-TCP connections, use POSIX or Core Foundation (`CFSocket`) C networking APIs.
- For client code in Objective-C, use Foundation Objective-C networking APIs. Foundation defines

high-level classes for managing URL connections, socket streams, network services, and other networking tasks. It is also the primary non-UI Objective-C framework in OS X and iOS, providing routines for run loops, string handling, collection objects, file access, and so on.

- For client code in C, use Core Foundation C networking APIs—part of the CFNetwork framework. The Core Foundation framework and the CFNetwork framework are two of the primary C-language frameworks in OS X and iOS. Together they define the functions and structures upon which the Foundation networking classes are built.

Note: In older versions of OS X, CFNetwork is a subframework of the Core Services framework.

To Learn More

To learn more about how to use sockets and socket streams, read Using Sockets and Socket Streams.