互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

第 1 章

iOS 网络功能介绍

本章内容：

- 理解 iOS 网络框架
- 面向开发者的关键网络 API
- 高效使用应用的运行循环

优秀的 iOS 应用需要简单、直观的用户界面。与之类似，与各种 Web Service 通信的优秀应用也需要一个良好架构的网络层。应用的架构必须具备适应需求变化的灵活性，以及能够优雅处理不断变化的网络情况的能力，同时又要保持着能够实现适当的可维护性与可伸缩性的核心设计原则。

在设计移动应用的架构时，必须精通一些核心概念，如运行循环、各种网络 API，以及如何将这些 API 与运行循环集成起来以创建响应式的网络应用框架。本章将会详细介绍运行循环以及如何在应用中高效使用它们。此外，本章还会概览关键 API 及其应用场景。

1.1 理解网络框架

在开始开发与网络交互的 iOS 应用前，需要理解 Objective-C 中网络层的组织形式，如图 1-1 所示。

每个 iOS 应用都位于某个网络框架栈之上，网络框架栈由 4 层构成。最上层是 Cocoa 层，包含了用于 URL 加载的 Objective-C API、Bonjour 与 Game Kit。Cocoa 层的下面是 Core Foundation 层，这是一套 C API，其中包含了 CFNetwork，这是大多数应用级别的网络代码的基础。CFNetwork 在 CFStream 与 CFSocket 之上提供了一个简单的网络接口。这两个类是针对 BSD socket 的轻量级封装，后者则形成了最下面的层，与无线硬件最为接近。BSD

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

socket 严格使用 C 来实现，向开发者提供了与远程设备或服务器进行通信的完全控制能力。

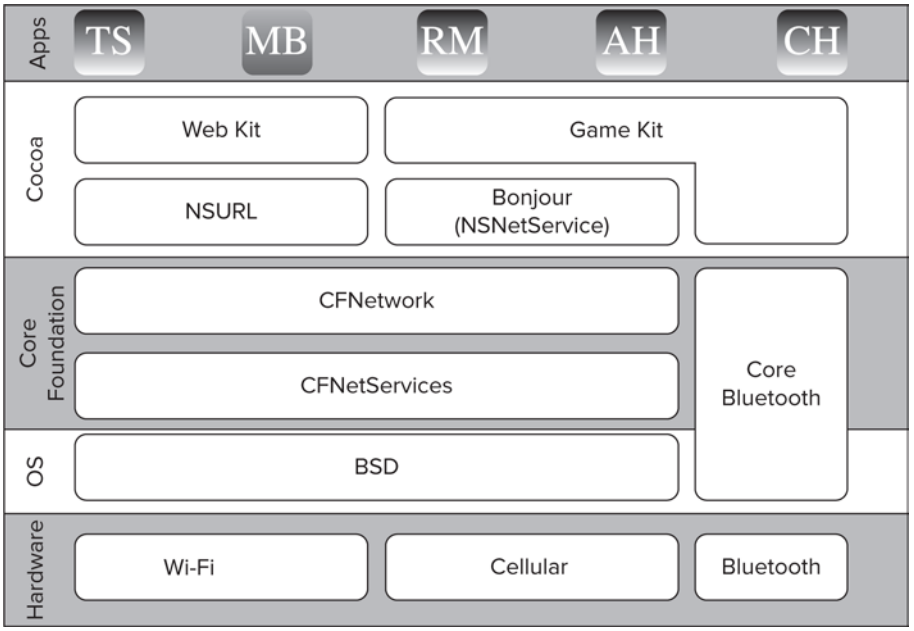

图 1-1

沿着框架栈向下移动每一层，你都会获得更为严密的控制，但却失去了上一层提供的易用性与抽象能力。虽然有时是可以的，不过 Apple 建议我们还是要位于 CFNetwork 层及其之上。BSD 层的原始 socket 无法访问系统范围的 VPN，也无法激活 Wi-Fi 和蜂窝无线电，而这些 CFNetwork 已经帮我们处理好了。

在开始设计应用的网络层之前，需要理解可用的各种 API 以及该如何使用它们。下一节将会介绍关键的 iOS 网络框架，并且简要介绍它们的使用方式。后面的章节将会对这里介绍的每一个 API 进行详细说明。

1.2　iOS 网络 API

框架栈的每一层都提供了一套关键 API，为开发者提供了各种功能与控制。每一层都比下一层提供了更高的抽象(参见图 1-1)。然而，这种抽象的代价就是丧失了某些控制。本节将会概览 iOS 中的关键 API 以及使用它们时的注意事项。

1.2.1　NSURLConnection

NSURLConnection 是一个 Cocoa 级别的 API，它提供了一个简单的方法来加载 URL 请求，可以与 Web Service 交互、获取图片或视频，或者只是简单地得到一个格式化的 HTML 文档。它构建在 NSStream 之上，并且在设计时针对如下 4 个常见的 URI 模式进行了优化支持：文件、HTTP、HTTPS 与 FTP。虽然 NSURLConnection 限制了你所能使用的协议，

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

但它对缓存读写的很多底层工作进行了抽象，包括对认证的内建支持，并且提供了一个健壮的缓存引擎。

NSURLConnection 接口中的内容并不多，它主要依赖于 NSURLConnectionDelegate 协议，应用可以凭借后者介入到连接生命周期的很多点上。在默认情况下，NSURLConnection 请求是异步的；不过有一个便捷方法可以发送同步请求。同步请求会阻塞调用线程，因此你也需要根据这一点来设计应用。第 3 章“构建请求”会详细介绍 NSURLConnection，还会介绍大量示例。

1.2.2　Game Kit

Game Kit 的核心功能在于为 iOS 应用提供了另一个点到点的网络选项。在传统的网络配置中，Game Kit 构建在 Bonjour 之上；然而，Game Kit 并不需要网络基础设施就能使用。它能创建自组(ad-hoc)的 Bluetooth Personal Area Networks(PAN)，这样在极少或是没有基础设施的地方，它就是非常棒的网络候选者了。

在创建网络时，Game Kit 只需要会话标识符、显示名与连接模式即可。不需要 socket 配置或是任何其他底层网络就可以实现连接点之间的通信。Game Kit 通过 GKSessionDelegate 协议进行通信。第 12 章“使用 Game Kit 实现设备间通信”会介绍如何将 Game Kit 集成到应用中。

1.2.3　Bonjour

Bonjour 是 Apple 对零配置(zeroconf)网络的实现。它提供了一种机制，可以检测并与网络中的设备或服务进行连接，同时无须了解设备的网络地址。Bonjour 通过名字、服务类型与域这个元组来引用服务。它对多播 DNS(mDNS)与基于 DNS 的服务探测(DNS-SD)所要求的底层网络进行了抽象。

在 Cocoa 层，NSNetService API 提供了一个接口，用于发布和解析 Bonjour 服务的地址信息。可以通过 NSNetServiceBrowser API 探测网络上可用的服务。发布 Bonjour 服务(甚至是使用 Cocoa 层的 API)需要理解 Core Foundation 才能配置好通信所需的 socket。第 13 章“使用 Bonjour 实现自组织网络”将会详细介绍零配置网络、Bonjour，此外还会通过示例介绍如何实现基于 Bonjour 的服务。

1.2.4　NSStream

NSStream 是一个 Cocoa 级别的 API，构建在 CFNetwork 之上，作为 NSURLConnection 的基础，旨在完成一些底层的网络任务。类似于 NSURLConnection，NSStream 提供了一种机制；用以与远程服务器或本地文件进行通信。不过，可以通过 NSStream 在诸如 telnet 或 SMTP 等 NSURLConnection 不支持的协议之上进行通信。

NSStream 提供的额外控制是有代价的。它并没有提供对处理 HTTP/S 响应状态码或认证的内建支持。它所发出与接收的数据都位于 C 缓冲中，Objective-C 开发者对此可能不太熟悉。它还无法管理多个外发请求，需要子类化才能添加这个特性。NSStream 是异步的，

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

通过 NSStreamDelegate 实现通信更新。第 8 章“底层网络”与第 13 章“使用 Bonjour 实现自组织网络”将会介绍 NSStream 的不同实现。

1.2.5　CFNetwork

CFNetwork API 位于基础的 BSD socket 之上，用在 NSStream、URL 加载系统、Bonjour 与 Game Kit API 的实现中。它为 HTTP 与 FTP 等高级协议提供了原生支持。CFNetwork 与 BSD socket 之间的主要差别在于运行循环集成。如果应用使用了 CFNetwork，那么输入与输出事件都会在线程的运行循环中进行调度。如果输入与输出事件发生在辅助线程中，就需要以恰当的模式开始运行循环。本章稍后的 1.3 节“运行循环”将会对此进行详细介绍。

CFNetwork 比 URL 加载系统提供了更多的配置选项，这个结果是喜忧参半的。在使用 CFNetwork 创建 HTTP 请求时可以使用这些配置选项。在创建请求时，需要手工将与请求一同发送的 HTTP 头和 Cookie 信息添加进去。但对于 NSURLConnection 来说，标准的头与 Cookie 信息会被自动添加进去。

CFNetwork 基础设施构建在 Core Foundation 层的 CFSocket 与 CFStream API 之上。CFNetwork 包含了针对特定协议的 API，比如用于与 FTP 服务器通信的 CFFTP、用于发送和接收 HTTP 消息的 CFHTTP、用于发布与浏览 Bonjour 服务的 CFNetServices 等。第 8 章将会详细介绍 CFNetwork，第 13 章则会概览 Bonjour。

1.2.6　BSD socket

BSD socket 构成了大多数 Internet 活动的基础，是网络框架层次体系中的最底层。BSD socket 使用 C 实现，但也可以用在 Objective-C 代码中。并不推荐使用 BSD socket API，因为它并没有在操作系统中插入任何钩子(hook)。比如，BSD socket 无法穿过系统范围的 VPN，如果 Wi-Fi 或是蜂窝无线电被关闭了，调用其 API 也无法自动激活。Apple 建议至少使用 CFNetwork 或是更高层的 API。第 8 章将会详细介绍 BSD socket 与 CFNetwork，并通过示例展示如何将其集成到应用中。

在实现各种网络 API 时，需要理解它们是如何与应用集成的。下一节将会介绍运行循环的概念，它会监控操作系统的网络事件并将这些事件转发给应用。

1.3　运行循环

运行循环是由类 NSRunLoop 表示的，有些线程可以让操作系统唤醒睡眠的线程以管理到来的事件，而运行循环则是这些线程的基础组件。运行循环是这样一种循环，可以在一个周期内调度任务并处理到来的事件。iOS 应用中的每个线程最多只有一个运行循环。对于主线程来说，运行循环会为你开始，在应用委托的 applicationDidFinishLaunchingWithOptions: 方法调用后就可以访问了。

不过，辅助线程必须显式运行自己的运行循环。在辅助线程中开始运行循环之前，你

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

至少要添加一个输入源或定时器；否则，运行循环就会立刻退出。运行循环向开发者提供了与线程交互的能力，不过有时并不是必需的。比如，没有任何其他交互而处理大数据集的线程可能就不会开始运行循环。然而，如果辅助线程与网络交互，就需要开启运行循环。

运行循环会从两类源中接收事件：输入源与定时器。输入源(通常是基于端口的或是自定义的)会异步向应用发送事件。这两类源的主要差别在于内核会自动发出基于端口的源信号，而自定义源就需要从不同的线程中手动发出。可以通过实现与 CFRunLoopSourceRef 相关的几个回调函数来创建自定义输入源。

定时器会生成基于时间的通知，它为应用(特别是线程)提供了一种机制以在未来的某个时间执行某个具体任务。定时器事件是同步发出的，并且与特定的模式有关，后面将会对此进行介绍。如果这个特定的模式当前并没有被监控，那么事件就会被忽略掉，线程也不会收到通知，直到运行循环“运行”在相应的模式下为止。

可以配置定时器以触发一次或是重复触发。重新调度是基于调度的触发时间而不是实际的触发时间。如果定时器触发，同时运行循环正在执行一个应用处理器方法，那么它会等待，直到下一次通过运行循环来调用定时器处理器为止，这一般是通过设定@selector()实现的。如果触发处理器被推迟到了下一次调用发生的那个点，那么定时器只会触发一个事件，之前延迟的事件则会被压制住。

运行循环也可以有观察者，它们不会被监控，这为对象提供了一种方式，使之可以在运行循环执行过程中的某个活动发生时收到回调。这些活动包括进入或退出运行循环、运行循环睡眠或唤醒、运行循环处理输入源或定时器之前等。CFRunLoopActivity 枚举的文档中对此有说明。观察者可以配置成触发一次，这样当触发后就会将其删除，也可以配置成重复的。要想添加运行循环观察者，请使用 Core Foundation 函数 CFRunLoopObserverRef()。

运行循环模式

每次通过运行循环都是在你所指定的模式下的一次运行。运行循环模式是由操作系统所用的一种约定，用于过滤监控的源并发布事件，比如调用委托方法等。模式包含了应该监控的输入源与定时器，以及当运行循环事件发生时应该通知的观察者。

在 iOS 中有两个预定义的运行循环模式。NSDefaultRunLoopMode(Core Foundation 中的 kCFRunLoopDefaultMode)是系统默认的，在开始运行循环及配置输入源时通常会使用它。NSRunLoopCommonModes(Core Foundation 中的 kCFRunLoopCommonModes)是个可配置的模式集合。通过在输入源实例上调用 scheduleInRunLoop:forMode:等方法，将 NSRunLoopCommonModes 赋给输入源会将其与当前组中的所有模式关联起来。

说明：

Mac OS X包含了 3 个额外的预定义运行循环模式，这可以在不同的文档中看到。

NSConnectionReplyMode、NSModalPanelRunLoopMode与NSEventTrackingRunLoopMode提

互动出版网 China-Pub.com 专业图书网上第一专营店 www.china-pub.com 独家提供样章

供了额外的过滤选项，不过iOS中并没有提供。

虽然NSRunLoopCommonModes是可配置的，但这是个底层过程，需要调用Core Foundation函数CFRunLoopAddCommonMode()。这会自动注册输入源、定时器与新模式的观察者，而不必手工将其添加到每个新模式中。可以通过指定自定义字符串(如@"CustomRunLoopMode")来定义自定义运行循环模式。要想提高自定义运行循环的效率，你至少需要添加一个输入源、定时器或是观察者。

虽然本节概览了运行循环，不过 Apple 提供了一些关于运行循环管理的颇有深度的资源，如果要开发高级的、基于网络的多线程应用，那么应该看看这些资源。开发者文档位于 https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Multithreading/RunLoopManagement/RunLoopManagement.html。第 8 章“底层网络”、第 13 章“使 用 Bonjour 实现自组织网络”将会介绍因运行循环集成而获益的各种网络技术。

1.4　小结

理解 iOS 网络栈以及应用如何与运行循环交互是 iOS 开发者需要掌握的重要概念。良好架构的网络层为应用提供了极大的灵活性。与之类似，设计低劣的网络层则会对成功与可伸缩性带来不利影响。

本章介绍的工具概览了各种网络 API 并对它们进行了比较。对于它们的使用方式，本章虽然做了简单介绍，不过后续章节将会更深入地进行阐述。