Five Reasons Why Synchronous Networking Is Bad

This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator.

128

Greetings All

This is a copy (with minor updates) of a post from the original DevForums (1). Someone asked me to resurrect it, so here it is.

Share and Enjoy
—
Quinn “The Eskimo!” @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@apple.com"`

(1) If you’re curious why this needs resurrection, see this post.

I wrote the following for an internal audience and various folks have suggested that I post it externally. In many ways this is a follow-up to my previous “Synchronous Networking” post, and if you haven't read that you should probably do so before continuing. (Update: That “Synchronous Networking” post eventually turned in to QA1693 “Synchronous Networking On The Main Thread”, which in turn got rolled in to Addressing Watchdog Terminations, so I haven’t bothered resurrecting it here.)

Five Reasons Why Synchronous Networking Is Bad

When you do networking, you have a choice of three different modes:

- Asynchronous — For example, you can schedule an `NSStream` on a run loop and have it call you when data arrives.
- Synchronous, polled — For example, you can periodically call `-[NSInputStream hasBytesAvailable]` to see if data is available.
- Synchronous, blocking — For example, you can call `-[NSInputStream read:maxLength:]` which will block if no data is available.

In virtually all cases I recommend that you use asynchronous networking. That’s because the other two options have fundamental pitfalls, as explained by the rest of this document.

To start, it’s clear that polled mode is bad: It forces you to choose between wasting CPU time or wasting network performance. If you poll frequently then, on a slow network, you will be wasting CPU time (and thus power, which is obviously an issue on mobile and notebook hardware, but has recently become an issue on desktop and server hardware as well). If you poll infrequently then, on a fast network, you will be squandering network performance.

It’s less obvious why synchronous blocking networking is bad. Each of the five sections below describes a serious problem with it.

1. Resource Usage

The number one reason synchronous blocking networking is bad is that it wastes resources. You can’t do synchronous blocking networking on the main thread, so you necessarily have to create a secondary thread to do the work. That thread is a waste of resources, most notably:

- Virtual address space (always consumed)
- A *wired* kernel stack (consumed while the thread is blocked)

This is especially bad when you’re handling lots of connections simultaneously, most of which are idle. You consume a bunch of wired kernel stacks to get exactly *no* work done.

2. Threads Are Evil

Because synchronous blocking networking requires a secondary thread, you have to deal with having multiple threads in your process. This raises a host of thread-related issues, most notably the problem of sharing data between threads. It’s best to avoid this if you can.

3. Cancellation

Synchronous blocking networking makes it very hard to support cancellation. You either have to jump through a bunch of hoops (see my SocketCancel sample code) or poll for cancellation (and did I mention that polling was bad?).

4. Timeouts

Implementing a timeout is tricky with synchronous blocking networking. Some synchronous blocking APIs implement some form of timeout support, but not all. And where they do, the support tends to be less robust than you’d like.

Again, you can work around this with various tricks but this serves to further undermines the illusion that synchronous blocking networking is easier.

5. Bidirectional

It’s not possible to support full bidirectional communications on a single synchronous blocking connection.

Network

Asked 5 months ago by eskimo

Reply to this question