If you haven't used the Network Extension framework before, you might be surprised at how rich and powerful it is. There's probably no other framework on iOS (it's also available on macOS) that gives you such low-level access to the system.

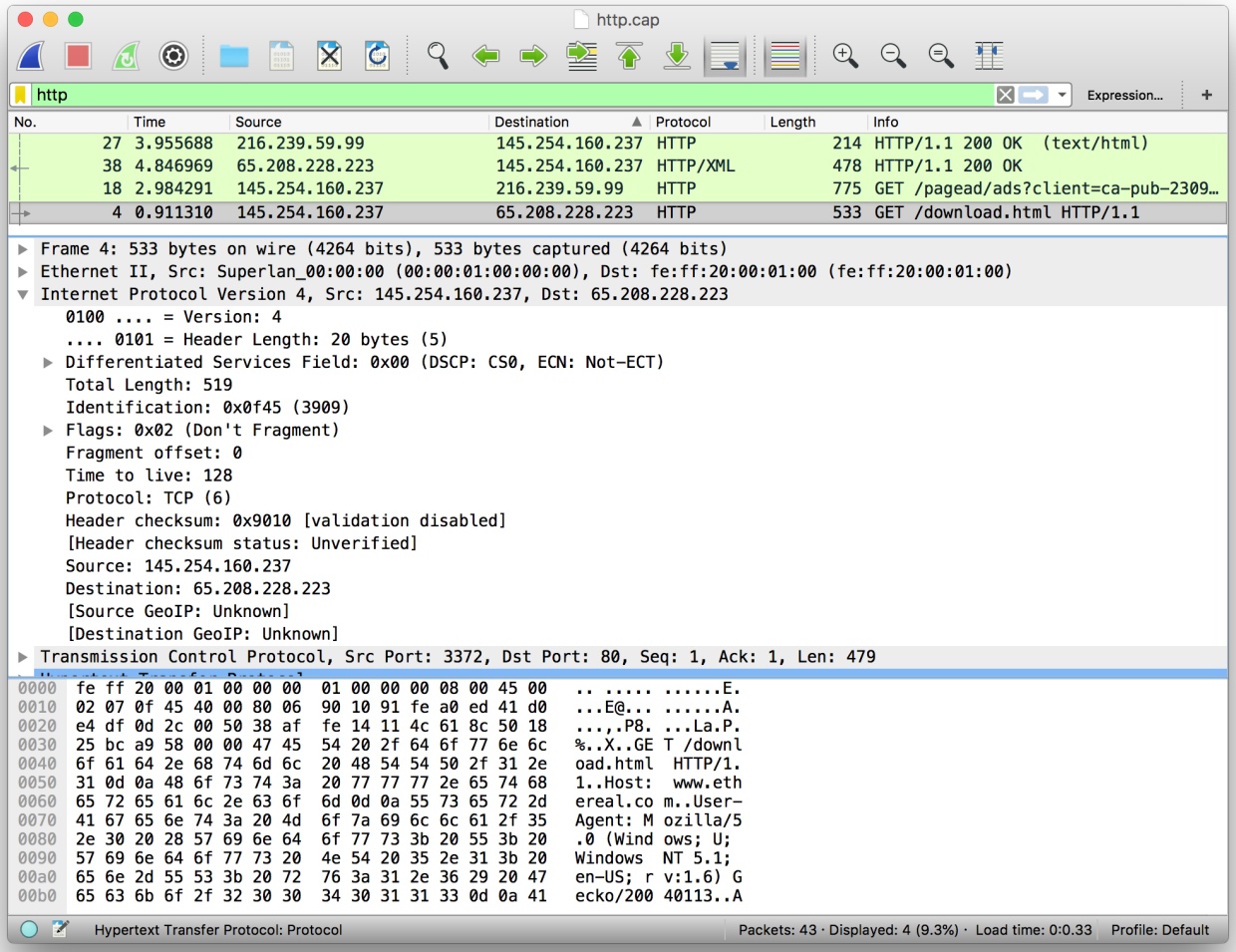
The Network Extension is an expansive framework. It provides a set of APIs that can be used to customize core networking features of the OS. What makes it especially exciting is that there are some surprising applications of some of the extensions which probably weren't envisioned by Apple (e.g. recently released Charles for iOS is built using network extensions).

This is **the first post in a series** about the Network Extension framework. I'm going to start with a high-level overview, dive deep into some of the most interesting extensions and explore a couple of apps that already take the powerful features provided by this fantastic framework.

Prerequisites

It's best to read this series when you're already familiar with the basics of network protocols. It should be enough to understand the basics of IP, TCP and UDP, and a bit about VPN. There is a lot of information available online. Regardless of whether you go through a tutorial or an RFC, I would recommend installing Wireshark and inspecting one or two sample captures.

> Wireshark is one of my favorite developer tools, I would definitely recommend trying it. It's a perfect instrument that would give you a good understanding of how internet protocols work together and what actually goes over the wire. And it's free!



Network Extensions Overview

Let's start with an overview of the available network extensions. There is already a good overview available in the Developer Library, but it does miss a few things.

I'm also going to highlight the requirements of each of the extensions. Some are only available on iOS 11, but that's easy to figure out. What's harder to notice is that some extensions only work on supervised devices, dramatically limiting where the apps that use them can be deployed. It's better to know about these limitations beforehand so that you wouldn't waste your time exploring an extension which you can't use.

> Unfortunately, the official documentation fails to highlight some of these limitations. For example, `NEDNSProxyProvider` is only available on supervised devices, but the Developer Library fails to mention that - I've opened a radar. Please keep in mind that sometimes these limitations get relaxed, so be sure to always check the official documentation. For example, there is a request by KrauseFx to make `NEFilterContentProvider` available on non-supervised devices.

There is no need to remember everything in the this section - it's just a high-level overview that provides a general idea of what the Network Extension framework is about.

Personal VPN

The **`NEVPNManager` (iOS 8+)** API gives apps the ability to create and manage a VPN configuration (one per app).

Network Tunneling Protocol Client

The `NETunnelProvider` APIs allow apps to implement the client side of a custom network tunneling protocol (e.g. VPN protocols).

- **`NEPacketTunnelProvider` (iOS 9+)** gives its subclasses access to reading and writing to a virtual network interface. In general, you would establish the connection with a VPN server, configure the tunnel, and start reading IP packets coming from the virtual network interface and sending them to the VPN server.
- **`NEAppProxyProvider` (iOS 9+, supervised devices, managed apps only)** is very similar to the `NEPacketTunnelProvider`, but operates on a TCP/UDP level instead of IP level.

The provider subclass should be added inside a special app extension target. The app extension will be automatically started by the system when necessary. In some cases, e.g. always-on VPN, the process running an extension will run indefinitely and will automatically be started even after the system restart.

> You won't be able to run any of these app extensions in a simulator, and to run in on a device you will need to create an entitlement.

On-Device Network Content Filter

The **`NEFilterProvider` (iOS 9+, supervised devices)** APIs give the ability to filter network traffic on iOS devices. This API was designed primarily for devices owned by schools. In general, most schools have some sort of content filtering enabled on their local WiFi networks, but that doesn't work with mobile networks. With `NEFilterProvider` the traffic filtering is going to work regardless of which network the device is connected to.

DNS Proxy

The **`NEDNSProxyProvider` (iOS 11+, supervised devices)** API allows apps to intercept all of the network traffic coming from the device. This allows apps to send DNS queries to preferred DNS servers, use custom DNS protocols (e.g. DNS over HTTPS which is supported by recently introduced Cloudflare DNS, by Google Public DNS and other DNS services), and more. One of the reasons to use custom (and encrypted) DNS protocols is to avoid DNS hijacking, which is a common practice among US mobile network operators.

Wi-Fi Hotspot Authentication and Configuration

The **`NEHotspotHelper` (iOS 9+)** API gives apps the ability to perform custom authentication for Wi-Fi Hotspots. It can also give users a way to seamlessly connect to a large network of Wi-Fi Hotspots. The **`NEHotspotConfiguration` (iOS 11+)** API lets apps configure these hotspots.

Upcoming Posts

The next post in the series is going to be a deep dive into `NEPacketTunnelProvider` (the most powerful and the most low-level extension) and into Apple's sample

project - SimpleTunnel. It's going to be a practical hands-on experience, please stay tuned.

> **Update:** The fact that this series never started turned into a bit of a running joke. I did end up writing three more posts on the Network Extension framework in 2020, starting with "How Does VPN Work?"

References

- Apple Developer Documentation: NetworkExtension Framework
- WWDC 2017. Session 707: Advances in Networking, Part 1
- WWDC 2017. Session 709: Advances in Networking, Part 2
- WWDC 2015. Session 717: What's New in NetworkExtension and VPN